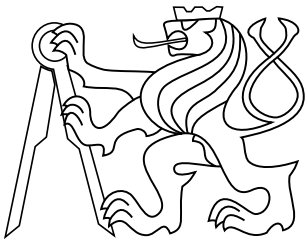
bachelor's thesis

Intelligent Algorithms for Petrol Station Inspections

Kateřina Jandová



2015

Štěpán Kopřiva, MSc.

Czech Technical University in Prague
Faculty of Electrical Engineering,

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Kateřina J a n d o v á

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Inteligentní algoritmy pro inspekci benzínových stanic

Pokyny pro vypracování:

1. Nastudujte problém inspekce benzínových stanic.
2. Nastudujte koncepty inspekčních her, bezpečnostních her a problém obchodního cestujícího.
3. Formulujte problém inspekce benzinových stanic za použití frameworku z oblasti teorie her.
4. Navrhněte vhodný algoritmus pro vyřešení výše uvedené hry.
5. Naimplementujte výše navržený algoritmus.
6. Otestujte algoritmus na otevřených datech poskytovaných Českou obchodní inspekcí.

Seznam odborné literatury:

[1] Stuart Russel, Peter Norvig – Artificial Intelligence: A modern approach, 2nd edition - 2003
[2] Rudolf Avenhaus – Applications of Inspection games – Mathematical Modelling and Analysis, 2004

Vedoucí bakalářské práce: MSc. Štěpán Kopřiva, MSc.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Kateřina J a n d o v á

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Inteligent Algorithms for Petrol Station Inspections

Guidelines:

1. Study the problem of petrol station inspections.
2. Study inspection games, security games and the traveling salesmen problem.
3. Formalize the problem of petrol station inspections as a game using suitable game theoretical framework.
4. Design an algorithm for solving the game defined above.
5. Implement the algorithm defined above.
6. Evaluate the algorithm on the subset of open data provided by Czech trade inspection.

Bibliography/Sources:

[1] Stuart Russel, Peter Norvig – Artificial Intelligence: A modern approach, 2nd edition - 2003
[2] Rudolf Avenhaus – Applications of Inspection games – Mathematical Modelling and Analysis, 2004

Bachelor Project Supervisor: MSc. Štěpán Kopřiva, MSc.

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2015

Acknowledgement

I wish to express my sincere thanks to my supervisor Štěpán Kopřiva, MSc., for providing all valuable advices and for his guidance. I would also like to thank my family for their support.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Podpis autora práce

Abstract

Efektivní kontroly pohonných hmot na benzinových stanicích jsou důležitým úkolem pro ochranu spotřebitelů. Kontroly benzinových stanic zajišťuje nejen stát, ale i soukromé firmy. V naší práci se zaměřujeme na vytvoření programu pro generování rozvrhu inspekcí benzinových stanic na základě herně-teoretického modelu stanic při uvažování rozpočtu na inspekce na daný rok. Pro vytvoření modelu jsme použili reálná data poskytovaná českou obchodní inspekcí. Pro spočtený rozvrh zároveň nalezneme optimalní trasu, po které inspektor projede zadané stanice. Výstupem programu je tedy rozvrh pro inspektora a přesná trasa, kterou má inspektor projet. Program musí vše spočítat tak, aby splnil rozpočet, který mu zadáme. Vzhledem ke složitosti úlohy program špatně škáluje a problém umí vyřešit pouze pro omezený počet vstupních benzinových stanic. Řešení programu trvá od desítek sekund až po jednotky minut v závislosti na počtu vstupních benzinových stanic.

Klíčová slova

Teorie her, Problém obchodního cestujícího, Optimalizace, Lineární programování, Pohonné hmoty, Benzinové stanice

Abstract

Effective inspection of fuel at petrol stations is important task to protect consumers. Inspections of petrol stations are ensured not only by the state but also by private companies. In our work we focus on developing a program for generating schedule of inspections. We create the program on basis of game theoretical model stations and we consider budget for inspections for one year. We use real data provided by the Czech Trade Inspection for creating the model. For the created schedule, we calculate the optimal route. The output of the program is schedule for the inspector and exact route which inspector has to ride out. We draw the route into the map. The program has to satisfy the budget which we define in Czech crowns. Due to the task's difficulty, the program can find solution only for limited amount of petrol stations and doesn't scale properly. Finding a solution can take as little as few seconds but with increasing amount of stations, this can take up to a few minutes.

Keywords

Game theory, Traveling salesman problem, Optimization, Linear programming, Fuels, Petrol station

Contents

1 Introduction

Fuel of poor quality is a problem, which touches many of us. Some petrol stations purposely add cheaper ingredients or foreign substances to the fuel in order to increase their profit per liter. Other petrol stations unintentionally change character of fuel by improper storing. If we refuel substandard fuel, engines of our vehicles can be damaged critically. This and losses on taxes are the reason why we have to inspect the petrol stations. The inspections can be performed by a public or private sector representative. Public sector agencies inspect the petrol stations on behalf of the government of state. The Czech Inspection Authority (CTIA) is state organization which makes the inspections in the Czech Republic.

The goal of this thesis is to create and implement algorithm which is able to solve the inspection problem of petrol stations and to produce a schedule which is hardly predictable for the petrol stations. We attempt to find not only one optimal route in a month for the given budget, but we look for game strategy for inspectors and then we find the optimal route for them.

In our work, we created two inspection models which model the problem of petrol station inspections. The second model was created according to real problem. After creating models, we create algorithm which solves the problem. We have evaluated the program on the real data from the CTIA. We have defined and performed a set of experiments on the real data.

1.1 Goals of the thesis

This thesis has the following goals:

- **Study the problem of petrol station inspections.** In the chapter 4 we study the problem of petrol station inspections. First, we present a basic overwiew of petrol stations inspection process. We explain the motivation to inspect petrol stations and the way petrol stations may possibly break the law. Then we briefly focus on technical standards for fuel, which are defined by law in the country. Finally we discuss who is the body running the inspection of petrol stations in the Czech Republic.
- **Study inspection games, security games and the traveling salesman problem.** We study inspection games, security games and the traveling salesman problem (TSP) in the chapter 2 and in the chapter 3. First, we describe inspection and security games in the chapter 2. Then we describe how to create a linear program for solving the game. In the chapter 2 we describe algorithms for solving TSP. We describe definition traveling salesman problem in the chapter 3. Then we explain the algorithm to solve the traveling salesman problem. We show and describe types of the TSP. We define TSP as linear program.
- **Formalize the problem of petrol station inspections as a game using suitable game theoretical model.** We formalize the problem of petrol station inspections as two-players game with calculating Nash equilibrium in the chapter 5. We show the problem like optimization problem with inspection optimizing.

We define benefits and costs of the cheating inspection stations. We describe all parameters for application and we define utilities function.

- **Design an algorithm for solving the game defined above.** We create algorithm which first solves the game theoretical model and then finds solution for TSP. We use the algorithm for computing the game theoretical part. We apply this algoritm twice. First utilization is for find potentially cheating petrol stations and calculating probabilities of cheating petrol station in the months is second application. We design algorithm for solving the traveling salesman problem. We describe it in chapter 6.
- **Implement the algorithm.** In the chapter 7 we describe implementation of the algorihtm, we describe programming language, libraries and databases, which we use in out application.
- **Evaluate the algorithm on the open data provided by Czech trade inspection.** We evaluate proposed algorithms on the real data. The results are provided in chapter 8.

1.2 Structure of the thesis

Chapter 2 describes published work related to our topic. The chapter provides an overwiev of game theoretical frameworks and TSP algorithms which are the two main sections of this chapter. In the first section, we show algorithms for solving the TSP problem. In the second section, we desribe inspection and security games.

Chapter 3 introduces frameworks and techniques we build our solution on. The chapter is divided into four sections. First section explains the linear programming framework, its complexity and its application for finding an optimal solution. Second part deals with traveling salesman problem. We define the problem and we show how to model TSP problem using the linear programming frameworks. In the third part we desribe the basics of the game theory framework. We write about games in extensive and normal forms, types of the game theory and the way game theoretical problems are solved using the linear programing framework. The last part explains what is IBM CPLEX.

Chapter 4 focuses on a basic overwiew of petrol stations inspection process. In this chapter we introduce overwiev of typical damage of vehicles and petrol station inspection specifics. We describe monitoring quality of fuels. We deal with companies performing the inspections. Finally, we summarize inspection process.

Chapter 5 focuses on the program formalization. We define physical inspection of potentially cheating petrol stations. We desribe the inspection policies of the inspector, benefits and costs of the cheating inspection stations. Then we formulate parametrs of petrol station. For the parametrs we create utilities functions and then we define the inspection schedule.

Chapter 6 introduces the algorithms we use to compute the solution of the problem. First we present the algorithm for finding potentially cheating petrol stations and for sampling of the probabilities. Then we use the same algorithm for search months in which the petrol station will cheat. We make algorithm for solving TSP and we create route for inspector.

Chapter 7 describes programming language, libraries, extensions and databases, which we use in this thesis. We provide the description of IBM CPLEX, GraphHopper, JavaFX, Leaflet and Postgis database.

Chapter 8 shows results of evaluation on real-world scenarios.

2 State of the art

2.1 Traveling salesman problem

We cover the following existing algorithms for solving TSP problem:

- Genetic algorithm
- Simulated annealing

In this chapter we describe state of the art algorithms, the remaining algorithms for solving TSP problem we describe in the section 3.2.

2.1.1 Genetic algorithm

Genetic algorithm is a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithm is inspired by Darwin's theory about evolution.

We create a random initial population. The initial population is random selection of the posibble solutions which are analogous chromosomes. Then we evaluate the fitness which is assigned to each solution (chromosome). Fintess depends on how close it actually is to solving the problem.

We choose choromosomes with a higher fitness value and they reproduce children which can mutate after reproduction (this process is known as "crossing over").

If we found generation which conatins solution in the required accuracy, the problem is solved. If not, then new generation will reproduce children in the same process. This will continue until a solution is found. [2]

We illustrate the genetic algorithm on the following graph which contain six nodes. Two edges between each pair of nodes denotes different distance. Figure 1 is only a sketch. It does not contain the real distances between nodes. [3]

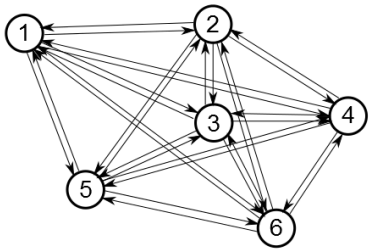


Figure 1 Directed weighted graph for TSP

Solution in TSP problem is usually represented by chromosome which has lenght as number of nodes in the problem. In our example we have lenght of chromosome five. Each node in the graph is one gene of the chromosome. No node can appear twice in the same chromosome. TSP has two representation methods: adjacency representation and path representation. We use path representation which we represent by lists the

gene (nodes). For example, we have a tour $\{1 \to 3 \to 2 \to 5 \to 4 \to 6\}$ and it may be represented as list $\{1, 3, 2, 5, 4\}$. We make the any sequence of the number of the nodes:

$$\begin{aligned} &\to\{1, 3, 2, 5, 4, 6\} \\ &\to\{6, 5, 4, 2, 3, 1\} \\ &\to\{3, 5, 1, 6, 2, 4\} \\ &\vdots \end{aligned} \tag{1}$$

We calculate fitness of the chromosomes by following process:

$$fitness = 1 \text{ to } 3 \text{ distance } +3 \text{ to } 2 \text{ distance } +2 \text{ to } 5 \text{ distance } +5 \text{ to } 4 \text{ distance } +4 \text{ to } 6 \text{ distance} \tag{2}$$

Fintess of the lists in the Example 1 is gradually 32, 58,41... We take chromosomes with the biggest fitness, it is chromosomes $\to \{6, 5, 4, 2, 3, 1\}$ and $\to \{3, 5, 1, 6, 2, 4\}$. This chromosomes are parents. On the parents we apply crossover operator. Part of the first parent is copied and the rest is taken in the same order as in the second parent.

Algorithm 1 Genetic algorithm

```
1: t = 0
2: initialize(P(t = 0))
3: evaluate(P(t = 0))
4: while isNotTerminated do
5:     P_p(t) = P(t).selectParents()
6:     P_e(t) = reproduction(P_p)
7:     mutate(P_e(t))
8:     evaluate(P_e(t))
9:     P(t + 1) = buildNextGenerationFrom(P_e(t), P(t))
10:    t = t + 1
11: end
```

Where t is the number of step of the algorithm. P is the population. P_p are the parents and P_e are the children of the population P.

2.1.2 Simulated annealing

Simulated annealing is a probabilistic method proposed in Kirkpatrick, Gelett and Vecchi (1983) and Cerny (1985) [4].

Algorithm 2 Simulated annealing

```
1: s = s_0
2: for k = 0 through k_max do
3:     T ← tepmerature(k/k_max)
4:     pick a random neighbour, s_new ← neighbour(s)
5:     if P(E(s), E(s_new), T) > random(0, 1), move to the new state then
6:         s ← s_new
7: return the final states
```

We define initial state. In the TSP each state is defined as permutation of the cities. The neighbours of a state are the set of permutations that are produced. At each step,

the simulated annealing heuristic considers some neighbouring state s_{new} of the current state s. It probabilistically decides between moving the system to the state s_{new} or staying in the state s. These probabilities P ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.

2.2 Game theory

Game theory is a branch of applied mathematics and economics that studies strategic situations where there are several stakeholders, each with different goals, whose actions can affect one another. In our work, we are going to model a domain which is usually modeled using the inspection games or security games.

2.2.1 Inspection games

Inspection games model is applied type of game theory. In inspection games the algorithm usually solves a game with two players or two coalition of players. One of the player (coalition) is called inspector and second player (coalition) is inspectee. Th inspection game may be defined by one inspector and a coalition of inspectee. The inspector controls the inspectee to check whether he complies the legal rules. The inspectee has tendency for violation these rules and the inspector's mission is to detect illegal behaviour of the inspectee. Generally, the inspector's resources are limited so that inspection can be only partial. We construct an optimal inspection scheme by a mathematical analysis, where penalty for the illegal behave is calculated strategically.

The concept of games is described in the work of Rudolf Avenhaus, Bernard von Stengel and Shmuel Zamir [6]. They explain application of games in practice, for example, in arms control and disarmament, accounting and auditing in economics and environmental control. Theory of inspection games is described in the article too.

We have to decide between two alternatives (H_0 or H_1) based on an observation of a random variable (if is it distribution of the random variable). We assume that the distribution of the random variable is strategically controlled by player called the inspectee. The inspectee can make two decision, either comply to the inspection rules or cheat. If he his behaviour complies to the rules, the distribution is according to the null hypothesis H_0. We denote variable ω called violation procedure and it marks illegaly behave of inspectee. Next random variable (which is also strategic variable of the inspectee) is Z, that means the distribution of the Z is under hypothesis H_1 and it depends on the violation procedure ω. The inspector has to decide between two actions: alarm (rejecting H_0) or no alarm (rejecting H_1). The decision is made according to the observation $z \in Z$. This model is shown in Figure 2.

2.2.2 Stackelberg game

In the following paragraph we describe Stackelberg game. The Stackelberg game has different two players: leader (moves first) and follower (moves second, after observes the leader's strategy). In the security game the leader is the defender and the follower is the attacker. "This models the capability of malicious attackers to employ surveillance in planning attacks. In this model, predictable defense strategies are vulnerable to exploitation by a determined adversary. Formally, the attacker's strategy in a Stackelberg

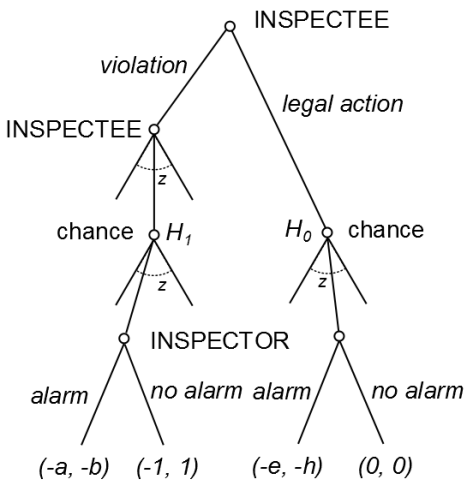


Figure 2 Inspection game

security game is a function that selects a strategy in response to each leader strategy: $F_A : \Delta_D \to \Delta_A$." [7]

Stackelberg equilibrium

In the Stackelberg equilibrium each player chooses a best response in each subgame of the original game. We have two types of Stackelberg equlibrium: strong (by Leitmann [9]) and weak (by Breton [10]). "In the weak Stackelberg equilibrium the follower choose the worst strategy for the leader. In the strong form the follower choose always optimal strategy for the leader. A strong Stackelberg equilibrium exists in all Stackelberg games, but a weak Stackelberg equilibrium may not." [11]

Definition according to the article [7]:
"A pair of strategies (δ_D, F_A) from a Strong Stackelberg Equilibrium if they satisfy the following:

1. The leader plays a best response:

$$P_D(\delta_D, F_A(\delta_D)) \leq P_D(\delta'_D, F_A(\delta'_D)) \forall \delta'_D \in \Delta_D.$$

2. The follower plays a best response:

$$P_A(\delta_D, F_A(\delta_D)) \leq P_A(\delta_D, \delta_A) \forall \delta_D \in \Delta_D, \delta_A \in \Delta_A.$$

3. The follower breaks ties optimally for the leader:

$$P_D(\delta_D, F_A(\delta_D)) \leq P_D(\delta_D, \delta_A) \forall \delta_D \in \Delta_D, \delta_A \in \Delta^*_A(\delta_A)."$$

2.2.3 Nash equilibrium

In the game theory is used a Nash equilibrium. It is a solution concept (strategy profile) of a non-cooperative game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy. [8]

2.2.4 Security games

Security games are models which are applied in protective environments. It efficiently solves many security problems. For example police patrols, bomb-sniffing dogs and security cameras. The security games are defined for example in the work of Christopher Kiekintveld, Manish Jain, Jason Tsai James Pita, Fernando Ordóñez, and Milind Tambe in their work [7].

We define security game as a normal form Stackelberg game. The security game is composed of two players, a defender D and an attacker A. The defender may not be one person, but could also make groups which cooperate to execute a joint strategy (for example police or army). The attacker can do the same (for example terrorist organization). The defender has a set of possible pure strategies. They are denoted $\sigma_D \in \Sigma_D$. The attacker has a set of possible strategies $\sigma_A \in \Sigma_A$. Players use mixed strategy. This means that the player may play with a probability distribution over pure strategies. It is defined as $\delta_D \in \Delta_D$ and $\delta_A \in \Delta_A$. We define payoffs for each player from defender by join all possible pure strategies outcomes: $P_D : \Sigma_A \times \Sigma_D \to \mathbb{R}$, similarly for the atacker.

3 Technical background

In this chapter we introduce frameworks and techniques we are building on. Firstly, we describe linear programming framework and its application for finding an optimal solution of the traveling salesman problem. CPLEX, solver of the linear programs is also introduced. We use it to solve our programs.

3.1 Linear Programing (LP)

Linear programing (LP) is a technique for optimization of linear objective function according to the set of linear inequalities (conditions). In LP the variables that are used in the optimization function refered to as the decision variables.

$$x_i$$

$$i = 1, 2, ..., n$$

To find the optimal solution solution we need to maximize or minimize linear function of the decision variables, which is called the objective function.

$$\xi = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

It follows that the set of feasible solutions of a linear programming is a convex polyhedron. The convex polyhedron is formed of the finite number half space, which are specified as linear conditions, then linear programming finds maximal or minimal point of polyhedron.

In linear programming are three possibilities: the task has at least one optimal solution, the task is inadmissible and the task is unlimited.

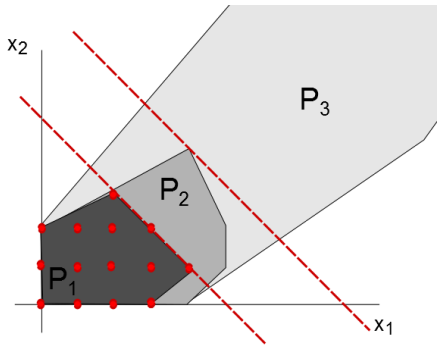


Figure 3 Convex polyhedron

At the Figure 3 we can see three convex polyhedrons in two-dimensional space. Red line corresponds to the objective function equal to zero. If task has optimal solution, then the solution may be either P_1 or P_2. In the case of P_1, the task has one optimal solution. The task has more optimal solutions in polyhedron P_2. The task is inadmissible if it has no solution. Therefore it is not at the figure. The task is unlimited, this means that task has infinitely many solutions (P_3).

Linear programming may be formalized in many ways. We can maximize or minimize objective function and conditions may contain both equalities and inequalities. There are two main possibilities how to represent linear programming problem. The first representation of linear programming is standard form (canonical form), example 3. Name can differ according to the source. In the normal form we maximize objective function and we use sign less than or equal to $\leq$ in conditions and every variables are greater than or equal to zero.

$$\begin{aligned} \max \mathbf{c^T x} \\ \mathbf{Ax} \leq \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3}$$

The second representation of linear programming is dictionary form. We maximize objective function, all conditions are equalities and all variables are positive or zero.

$$\begin{aligned} \max \mathbf{c^T x} \\ \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{4}$$

The standard form can be converted to the dictionary form and vice versa by some of the following modifications. We replace the equality $\mathbf{a_i^T x} = \mathbf{b_i}$ by two inequalities $\mathbf{a_i^T x} \geq \mathbf{b_i}$, $-\mathbf{a_i^T x} \geq -\mathbf{b_i}$. We convert the inequality $a_{i1}x_1 + ... + a_{in}x_n \leq b_i$ to equality by adding auxiliary slack variable $\lambda_i \geq 0$ as $a_{i1}x_1 + ... + a_{in}x_n + \lambda_i \leq b_i$. If the inequality is reversed $a_{i1}x_1 + ... + a_{in}x_n \geq b_i$, then we convert this inequality to equality by the same way - we add auxiliary slack variable $\lambda_i \geq 0$ but we subtract it $a_{i1}x_1 + ... + a_{in}x_n - \lambda_i \leq b_i$. If we have unbounded variable $x_i \in \mathbb{R}$, we divide the original variable to two non-negative variables $x_i^+ \geq 0$, $x_i^- \geq 0$ by adding condition $x_i = x_i^+ - x_i^-$.

Can one convert the non-linear function to the linear one? Yes, we can, if the function is piecewise affine.

$$\begin{aligned} f(\mathbf{x}) = \max_{k=1}^{K}(\mathbf{c_k^T x} + \mathbf{d_k}) \\ \mathbf{c_k} \in \mathbb{R} \\ d_k \in \mathbb{R} \end{aligned} \tag{5}$$

We have the task:

$$\begin{aligned} \min f(\mathbf{x}) \\ \mathbf{Ax} \geq \mathbf{b} \end{aligned} \tag{6}$$

The function f is piecewise affine. We can transform the task to the linear program by defining auxiliary variables z.

$$\begin{aligned} \min z \\ \mathbf{c_k^T x} + \mathbf{d_k} \leq z, \quad k = 1, ..., K \\ \mathbf{Ax} \geq \mathbf{b} \end{aligned} \tag{7}$$

We minimize over variable $(x_1, ..., x_n, z) \in \mathbb{R}^{n+1}$. We can use this transformation for function with absolute value too, because $|x| = \max\{-x, x\}$.

The transformation of piecewise affine function to the linear program is not possible if we maximize in the task.

3.1.1 Complexity of Linear Programming

The complexity of Linear Programming relies on the instance size and type of data. We have two standard complexity theory frameworks and hence measure the size. Standard complexity theory frameworks for analyzing LP solvers are often referred to as bit complexity and algebraic complexity.

Regarding the bit complexity: Data coefficients are assumed to be integers specified in binary form. The size of an instance is the total number of binary bits in the data for the instance. Bit complexity is very natural for combinatorial problems where each data coefficient is either 0 or 1. It means, that pure integer linear programming problems are NP-complete.

Regarding the algebraic complexity: data coefficients are assumed to be real numbers. The size of an instance is number of data coefficients for the instance. One considers $+, -, \div$ and inequality comparison are basic operations, the latter being used for branching. We can add two numbers is a single operation, this is contrast to bit complexity.

3.1.2 Algorithms

Integer linear programming is a hard problem for solving. Set of feasible solutions of a linear programming is a convex polyhedron. Optimal solution is a vertex of the convex polyhedron, which constist all feasible points. This points usually not integer. There are two cases of a linear program. First case is integer linear program:

$$\begin{aligned} \min -x - y & \\ x + 2y \leq & \quad 14 \\ 3x - y \geq & \quad 0 \\ x - y \leq & \quad 2 \end{aligned} \tag{8}$$

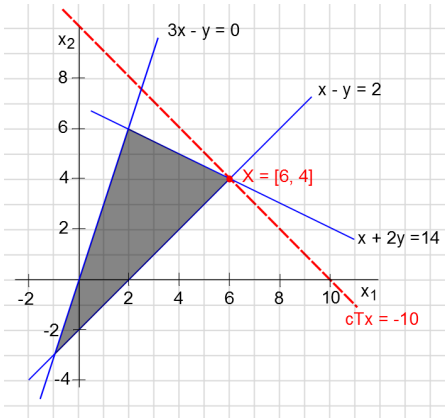


Figure 4 Convex polyhedron of feasible solutions in integer linear programming

The figure 4 shows integer linear program 8. All vertices of a convex polyhedron are integers, so this problem is called integer linear program (ILP). We can only move the objective function over integers in convex polyhedron and we get solutions. This way is not effective and we can not use it in general linear program. Integer linear program is easier to solve, but we meet with it in practice rarely. Linear program, where the

results are real numbers, is more frequent problem.

$$\begin{aligned} \min -x - y & \\ x + 2y \leq & \quad 15 \\ 3x - y \geq & \quad 0 \\ x - y \leq & \quad 2 \end{aligned} \tag{9}$$

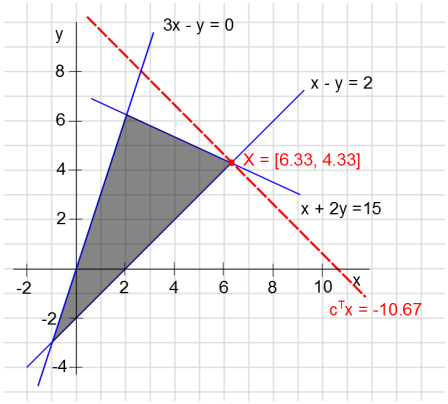


Figure 5 Convex polyhedron of feasible solutions in integer linear programming

Vertices at the figure 5 are real number. The task 9 is a hard problem. We must use some more sophisticated algorithm. We show here very important algorithms for solving LP the Branch and Cut algorithm and the Simplex method. The Branch and Cut algorithm is the part forms that it is composted: Branch-and-Bound algorithm and Cutting-plane method.

Branch and Bound algorithm

Branch and Bound algorithm is a general algorithm for solving large scale N P-hard combinatorial optimization problems. A Branch and Bound algorithm searches the complete space of solutions for a given problem for the best solution. The algorithm was proposed by Ailsa H Land and Alison G Doig [12]. Task, where the number of possible solutions is exponentially large. It is almost impossible to solve. We use of bounds for the function to be optimized combined with the value of the current best solution.

We represent algorithm Branch and Bound by search tree, which initially only contains the root. We decompose task into set of subproblems. Subproblems are represented as nodes of the tree (Figure 6). Each iteration of a Branch and Bound algorithm processes one such node. The iteration has three components (selection of the node to process, bound calculation, and branching).

The selection of the node to process depends on the chosen strategy.

For actual subproblem is calculated bound function and and compared to the current best solution. We discare the subproblem (and it's subproblems), if it can be established that the subspace cannot contain the optimal solution. Otherwise we save the subproblem.

It can be found more specify in [13].

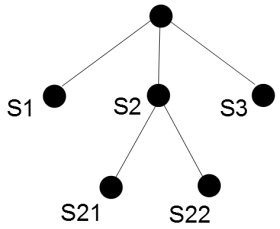


Figure 6 Search tree of Branch and Bound algorithm

Algorithm 3 Branch and Bound algorithm

```
1: Initialize:
2: x* ← null (it is current best solution)
3: LB* ← -∞ (it is lower bound)
4: L is a list of problems to solve
5: begin
6: while L is not empty do
7:     Select and remove problem from L
8:     Solve LP
9:     if unfeasible then
10:        contunue
11:    else
12:        LB ← bound
13:        x ← solution
14:    if LB ≤ LB* then
15:        continue
16:    if x is Integer then
17:        x* ← x
18:        LB* ← LB
19:        continue
20: Branch and add subproblems into L
```

Figure 7 Branch and Bound algorithm [14]

Simplex method

We will examine the set of feasible solutions in the LP form:

$$X = \{\mathbf{x} \in \mathbb{R} | \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\} \tag{10}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is wide ($m < n$) matrix with rank m (rows are linearly independent).

System $\mathbf{A}\mathbf{x} = \mathbf{b}$ have infinitely many solutions. If we put $n - m$ components of vector x is equal zero, the system has at most one solution. This leads to the followinf definitions:

- Set $J \subseteq \{1, 2, ..., n\}$ is base of task, if $|J| = m$ and columns (of matrix $\mathbf{A}$ with indices J) are linearly independent.
- Vector $\mathbf{x}$ is base solution for appropriate base J, if $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $x_j = 0$ for $j \notin J$.

- Base solution $\mathbf{x}$ is admissible, if $\mathbf{x} \geq \mathbf{0}$
- Base solution $\mathbf{x}$ is degenerate, if it has less than m non-zero components.
- Two bases are neighboring, if they have $m - 1$ common elements.

Martix $\mathbf{A}$ has rank m. This implies that exist at least one base. Every base appropriate just one base solution. One base solution may appropriates more than one base. This mean that the base solution is degenerate. [15]

3.2 Traveling Salesman problem

The TSP is probably the most widely studied combinatorial optimization problem, because it is a conceptually simple problem but hard to solve. It is an NP-complete problem. First, we describe TSP problem. Then we introduce types of the TSP and algorithms for solving the TSP problem.

Traveling Salesman Problem (TSP) is a NP-hard combinatorial optimization problem. The goal is to find a circle in the specified rated complete graph. The circle must pass through all vertices and cost of the sum of all edges is minimal. In other words, we find the shortest Hamiltonian circuit in weighted graph. Specifically, we use TSP to find tour between cities. We have list of cities and the distances between each pair of cities. We want to know shortest possible route given the circumstances that we visit each city exactly once and we return to the starting city.

TSP is usually defined by an undirected weighted graph (Figure 8). Cities are vertices in the graph, paths are edges in the graph and path's distance is edge's cost. We minimize the length of the path, which starts and ends at a specified vertex and which visits each other vertex exactly once. The model is usually complete graph (i.e. each pair of vertices is connected by an edge). If graph isn't complete (the path between all pairs of the cities doesn't exist), we can add an arbitrarily long edge without affecting optimal solution.

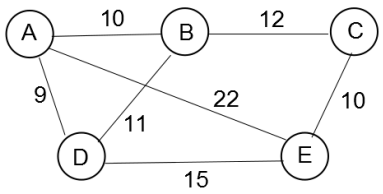


Figure 8 Undirected weighted graph

3.2.1 Symmetric and asymmetric TSP

Symmetric TSP: distance between two cities is the same in both directions (i.e. TSP is defined by undirected weighted graph, Figure 8). Symmetric TSP have less solution then asymmetric TSP. Symmetric TSP is simpler for finding an optimal solution.

Asymmetric TSP: distance between two cities can be different in each opposite directions and path may not exist in both directions (i.e. TSP is defined by directed weighted graph, Figure 9). Reason may be traffic collisions, one-way streets, and airfares for cities with different departure and arrival fees.

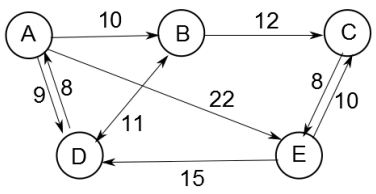


Figure 9 Directed weighted graph

3.2.2 Algorithms for solving TSP

Greedy algorithm

The greedy algorithm is very simple one. The algorithm chooses a node where it starts, let's mark it $node1$. The algorithm calculates all the distances from $node1$ to the remaining $n-1$ nodes. Then the algorithm chooses closest node and we go to it. Now the actual node is taken and we find next closest node from $node2$ to other $n-2$ nodes. The algorithm continues until all nodes are visited exactly once. Then the connection from the last node to the first one is added. The final sequence is returned as the best solution. The greedy algorithm is not computationally intensive, because the algorithm does not exchange any of nodes.

3.2.3 2-Opt algorithm

The idea of the 2-Opt algorithm is to exchange two edges and see if the cost improves. The 2-Opt algorithm consist from three steps:

- We set initial solution and objective function value. We have the solution from user or from greedy algorithm.
- We find two edges and their endpoints.
- We swap the endpoints.

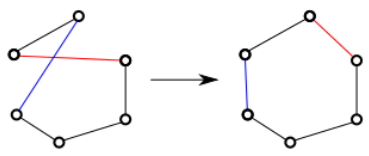


Figure 10 A single 2-Opt algorithm's move

Algorithm 4 2-Opt algorithm

```
1: S = some initial solution
2: z = objective function value of the initial solution S
3: noChange = true
4: while noChange do
5:     S*= S
6:     z*= z
7:     for all possible edge-pairs in S do
8:         S' = tour by swapping end points in edge-pair
9:         z' = objective function value of S'
10:        if z' < z* then
11:            S* = S'
12:            z* = z'
13:            noChange = false
14:    S = S*
15:    z = z*
16: return S
```

We can see that 2-Opt algorithm considers only two edge exchange. We have initiall soulution S defined by user or greedy algorithm and we compute objective function value z. We set initial solution S as the best solution S^* and we set z as z^* in the same manner. If the objective function value z' (objective function value computed from solution with replaced edges) is smaller than the objective function value $z*$, it is stored as candidate for future decision (we set solution S' as the best solution S^* and z' as z^*). If not, solution S' and its objective function value z' are discared.

Greedy 2-Opt algorithm

"Like the 2-Opt algorithm, greedy 2-opt algorithm also considers pairwise exchanges. We define $edge1, edge2, .., edgeN$ for each edge in the graph and N means number of the edges in the graph. Initially, it considers transposing $edge1$ and $edge2$. If the result's objective function value is less than the previous one, two edges are immediately transposed. If not, the algorithm will go on to $edge3$ and evaluate the exchange, and so on until find the improvement. If $edge1$ and $edge2$ are transposed, the algorithm will take it as an initial solution and repeat the algorithm until it is impossible to improve the solution any further. Greedy 2-opt algorithm makes the exchange permanent whenever an improvement is found and thus consumes less computational time than 2-Opt algorithm. On the other hand, greedy 2-opt produces slightly worse solutions than 2-Opt algorithm." [1]

3-Opt algorithm and greedy 3-Opt algorithm

The 3-Opt algorithm is similar to the 2-Opt algorithm, but the 3-Opt algorithm changes three edges. The same difference is between Greedy 2-Opt algorithm and greedy 3-Opt algorithm.

Neural network

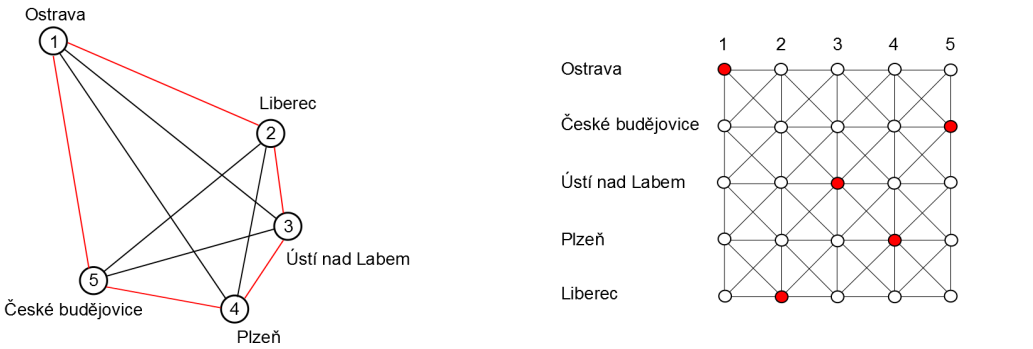
Neural networks are used for a lot of mathematical problems. It is not suprising that it is applied to the TSP. First attempts apply neural network were in 1985, but the

technology is quite young and it is still to examine. Neural networks are powerful parallel devices. They are made up of a large number of simple elements that can process their inputs in parallel.

The most frequent types of neural networks fot the solving TSP problem are the Hopfield-Tank network, the elastic net and the self-organizing map. According to the work of Jean-Yves Potvin [5]: the elastic net and the self-organizing map appear to be the best approaches for solving the TSP. The Hopfield-Tank network has been the dominant neural method for solving combinatioral optimiyation problems over the last decade. It was the first to be applied to the TSP and still many researchers are working on that model, trying to explain its failures and successes.

The Hopfield-Tank model

The network or graph defining the TSP is very different from the neural network. As a consequence, the TSP must be mapped onto the neural network structure.



a) Graph of TSP

b) Interpretaton of the Hopfield-Tank model

Figure 11 Different between ordinally TSP graph and interpretaton of the Hopfield-Tank model

In the Figure 11a is TSP defined over a transportation network. A feasible solution is the tour Ostrava → Liberec → Ústí nad Labem → Plzeň → České Budějovice, as shown by red line. In the Figure 11b is the Hopfield network. It is defined as a 5×5 matrix of nodes that are indicate cities in a time interval. Each row corresponds to a particular city and each column to a particular position in the tour. The nodes are used to encode solutions to the TSP. The red nodes are the activated units that encode the current solution. In the Figure 11b is not all connection. In fact, there is a connection between each pair of nodes.

We start from some arbitrarily chosen initial configuration. It can be either feasible or non-feasible. The Hopfield network evolves by updating the activation of each node. The update rule of any given node involves the activation of the nodes which are connected to as well as the weights on the connections. We update until the network settles into a stable configuration.

We summarize the algorithm (according to Jean-Yves Potvin [5]) to a few points:

- Choose a scheme which allows the activation levels of the nodes.
- Design an energy function whose minimum corresponds to the best solution of the problem.
- Derive the connectivity of the network from the energy function.
- Set up the initial activation levels of the units.

Integer linear programming formulation

Asymetric and symmetric TSP can be formulated as integer linear problem:

$$\begin{aligned}
&\min \sum_{i=0}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} \\
&\sum_{i=0, i \neq j}^{n} x_{ij} = 1 && j = 0, ..., n \\
&\sum_{j=0, j \neq i}^{n} x_{ij} = 1 && j = 0, ..., n \\
&0 \leq x_{ij} \leq 1 && i, j = 0, ..., n \\
&u_1 = 1 \\
&2 \leq u_i \leq n && \forall i \neq 1 \\
&u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) && \forall i \neq 1, \forall j \neq 1
\end{aligned} \tag{11}$$

In formulation 11: x_{ij} is a boolean indication whether from city i to city j leading a path (Formulation 12). We have 1, ..., n cities.

$$x_{ij} = \begin{cases} 1, \text{ the path goes from city } i \text{ to city } j \\ 0, \text{ otherwise} \end{cases} \tag{12}$$

u_i is an artificial variable for $i = 0, ..., n$. c_{ij} is the distance from city i to city j.

The first equality (in formulation 11) implies that each city be arrived at from exactly one other city. The second equality implies that from each city is a path to exactly one other city. The third constraint implies that exists only one circuit, which contain all cities. The solution doesn't contain two or more disjointed circuits.

3.2.4 Computing and solution

To compute the NP hard problems, we can also use one of the following methods:

- Optimal algorithms which compute the optimal solution (they will work reasonably fast only for small problem sizes).
- Heuristic algorithms (algorithms, which have either seemingly or probably good solutions, but solutions may not be optimal).
- We can find "subproblems" for which are possible better or exact heuristics.

We show a few examples of these algoritms.

Example of exact algorithms is branch and bound algorithm, which works well for 40-60 cities.

Heuristic and approximation algorithm is nearest neighbor algorithm or christofides algorithm. The nearest neighbor algorithm choose the nearest unvisited city in every move. This algorithm is quick and it find short path, but the algorithm doesm't find the shortest path. The path can be 25% longer that the shortest possible path.

The christofides algorithm solves the problem of the metric traveling salesman problem (only metric). In the worst case path is lenght 3/2 of the optimal solution. The disadvantage of this algorithm is difficult to implement it. Other algorithms for solving TSP [16] and [17].

3.3 Game theory

Game theory concerns with the behaviour of decision makers whose decisions affect each other. Game theory studies strategic decision-making. Specifically, it is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" [18]. It analyses the behavior of the agents is from a rational rather than a psychological or sociological aspect. Its methods can be applied in principle to all interactive situations, especially in economics, political science, evolutionary biology, and computer science. Game theory applies to a wide range of behavioral relations, it includes human as well as non-human players (computers, animals, plants).

Game theory began with the existence of mixed-strategy equilibria in two-person zero-sum games, introduced by John von Neumann [19]. His paper was followed by book Theory of Games and Economic Behavior, which studied cooperative games of several players.

The game theory has precisely defined mathematical objects:

a game - every confrontation of players
player - a person, who by their behavior can affect game's output
(player is rational, if he wants to achieve an optimal outcome of the game)
strategy - assignment of one behavior for the game
utility (payoff) - player's loss (gain) at the end a game round or game
utility matrix - assignment utility for the selected strategy

Fully defined game must specify the players of the game, actions available to each player at each decision point, and the utilities for each outcome.

Both Cooperative and non-cooperative games can be described using normal form and extensive form.

3.3.1 Normal form

The normal form of the game is usually represented by a matrix. The matrix describes players and their possible stategies and possible utilities.

	Player II chooses M	Player II chooses N
Player I chooses K	6, 5	1, 1
Player I chooses L	-1, -1	5, 6

Table 1 Game Theory - normal form

In the example in the Table 1 describes a two-player game. First player's task is to select his strategy - a row in the matrix. Second player's task is to select his strategy - a column in the matrix. Each player has two options for action. Utilities are written inside the matrix (table). In each cell the first number is utility forthe player I (it is marked in red). Second number in the cell is utility for the player II (it is marked in blue). This means, that if the player I chooses K and the player II chooses M, the first player acquires utility 6 and the second player acquires utility 5.

In the case of the normal form game, we expect that players select actions concurrently or that players do not know, what action chooses their adversary.

3.3.2 Extensive form

The extensive form of the game is usually represented by a tree (Figure 12).

Every extensive form game has an equivalent normal form game. But if we transformation the extensive form to the normal form, it can be exponentially larger in the size of the representation. It is not computationally practical [20].

Number 1 in Figure 12 represents the first player and number 2 represents the second player. Each node is a decision point, where the player chooses his action. Player who chooses the action is determined by number, which is written above the node. Edges represent possible actions of the players. The utility is specified in the leaves of tree. 'The extensive form can be viewed as a multi-player generalization of a decision tree' [21].

We have two players at the Figure 12. First player is red and second player is blue. Player 1 plays first and chooses between K and L. Player 2 plays second. He may know Player 1's move. He chooses between M and N. If the first player chooses L and the second player chooses N, the first player gains utility 5 and the second player gains utility 6.

The extensive form is usually used to formalize games with a time sequencing of moves. The extensive form can appear even game when players choose moves at the same time and also games with incomplete information.

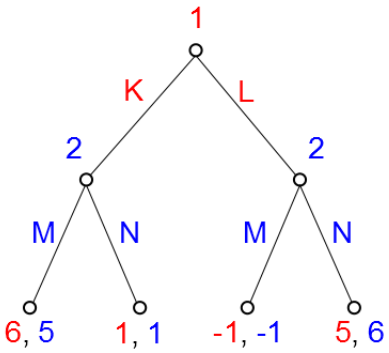


Figure 12 Extensive form represented by a tree

3.3.3 Types of games

There are a lot of game types in the game Theory. We will describe the basic game types.

Zero-sum / Non-zero-sum games

The total benefit in zero-sum games is always zero (for all players in the game, for every combination of strategies). This means, that a player's benefits are only equal at expense of others. For example, go or poker may be modelled as zero-sum games.

In the real world, many situations will be modelled using non-zero sum game concept because some results have total benefit more or less than zero. Profit of one player might not necessarily mean the lost for another player.

	Player II chooses M	Player II chooses N
Player I chooses K	6, -6	0, 0
Player I chooses L	-1, -1	-5, 5

Table 2 Example of a two-player zero sum game

Perfect information and imperfect information

A game with perfect information is a game where all players know the moves previously made by all other players. Most games are imperfect-information games. For example game of perfect information is chess and games of imperfect information are many card games, such as poker.

Perfect information is often confused with complete information, which is a similar concept. Game of complete information is the game, where every player know the strategies and payoffs of the other players but not necessarily the actions taken. Games of imperfect information contain "moves by nature" [19].

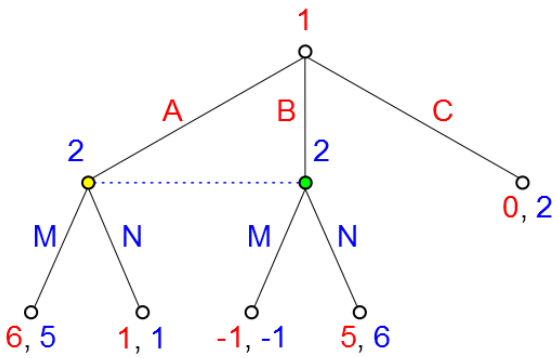


Figure 13 The game of imperfect information

In the Figure 13 is game for two players. The dotted line represents ignorance on the part of player 2. The second player do not know, if he is in the yellow node or in the green node.

Cooperative / Non-cooperative games

A game is cooperative if the players are able to form commitments. For example the legal system requires them to adhere to their commitments. This is not possible in non-cooperative games. Hybrid games contain cooperative and non-cooperative elements.

Formally "Cooperative game theory focuses on how much players can appropriate given the value each coalition of player can create, while non-cooperative game theory focuses on which moves players should rationally make." [22].

Agents compete and cooperate in cooperate model of game theory. They create and capture value in unstructed interactions. Cooperative game theory analyses situations where agents can cooperate to create value by joining coalitions. Agents also compete to capture value. Cooperative game theory is descibed by a set of agents and a function that returns the value each coalition. In non-cooperative game theory agents maximize their utilitz in a defined procedure. They rely ona detailed description of the moves.

Symmetric / Asymmetric games

A symmetric game is a game in which the results of the strategy depends on the strategies of other players. It does not depend in who is playing. The game is symmetric if the players can be replaced without change the payoff. Mostly it was study games 2x2 and they were symmetrical. Example of symmetric game is the Prisoner's Dilemma.

	Player II chooses M	Player II chooses N
Player I chooses K	1, 1	2, 3
Player I chooses L	3, 2	5, 5

Table 3 The symmetric game

Simultaneous / Sequential games

Simultaneous game is the game, where both players move concurrently or the player who moves later he has not information on the previous move opponent. Sequential games are games in which players know something about the opponent's previous move. Players may not have perfect knowledge.

3.3.4 Game theory as a linear program

We have matrix game with random strategy. Matrix game is defined by payoff (utility) matrix. Game with random strategy is the game, where player make him choices at random according to some fixed probability distribution. We formalize action of players, y_i label the probabilty that the row player selects action i. The vector y is called a stochastic vector. Vector is a stochastic vector if it satisfies the conditions in the Figure 13. [23]

$$\begin{aligned} y &\leq 0 \\ e^T y &= 1 \end{aligned} \tag{13}$$

In the Figure 15 e is the unit vector. We define variable for the column player in the same way, vector x is composed of probabilities x_i. Vector x is called a stochastic vector. Let x_i is the probability that the column player select action i.

The column player has the following expected payoff (in the Figure 14).

$$\sum_{i,j} y_i a_{ij} x_i = y^T A x \tag{14}$$

The column player has strategy x, because he decides to play according to stochastic vector x. The row player have to count up him best defense. It is the strategy y^*. The optimal strategy y^* the following minimum (in the Figure 15).

$$\begin{aligned} &\min y^T A x \\ &s.t. e^T y \quad = 1 \\ &y \quad \leq 0 \end{aligned} \tag{15}$$

"From the fundamental theorem of linear programming, we know that this problem has a basic optimal solution. For this problem, the basic solutions are simply y vectors that are zero in every component except for one, which is one. That is, the basic optimal solutions correspond to deterministic strategies."[24]

3.4 IBM ILOG CPLEX

IBM ILOG CPLEX is software package for matematical operations. It is often called olny CPLEX. The CPLEX is named for the simplex method as implemented in the C programming language. Now it has also support in other mathematical problems and it offers interface in java too. The CPLEX is solver ILP and LP problem from IBM and it representation commercial solver.

Originally developer of IBM ILOG CPLEX is Robert E. Bixby and it was sold through CPLEX Optimization Inc., which was take over by company ILOG in 1997. ILOG was purchased by company IBM in january 2009.

Formally decsription from IBM: "IBM ILOG CPLEX Optimization Studio is an analytical decision support toolkit for rapid development and deployment of optimization models using mathematical and constraint programming. It combines an integrated development environment (IDE) with the powerful Optimization Programming Language (OPL) and high-performance ILOG CPLEX optimizer solvers." [25]

4 Petrol Station Inspection Problem

In this chapter, we present a basic overwiew of petrol stations inspection process. At first, we explain the motivation to inspect petrol stations and the way petrol stations may possibly break the law. Then we focus on a technical standards for fuel, which is defined by law in the country. In the next part of this chapter, we discuss who is the body running the inspection of petrol stations in the Czech Republic. We divide inspection authorities to the two parts. Frist part is the public sector and second part is the private sector. Inspection entities of the public sector control random all petrol stations. Inspection entities of the private sector make inspections for legal entities. We also describe number of inspections and their success.

What does one need to inspect the petrol stations? Some petrol stations purposely add cheaper ingredients or foreign substances to the fuel, in order to increase their profit per liter. Other petrol stations unintentionally change character of fuel by improper storage. Fuel of poor quality can cause large damages to a car or motorbike.

There are three main conditions on fuel which are often not met:

- Fuel does not fulfill the requirement of the relevant standards or the requirement of a manufacturer vehicles.
- Fuel properties are significantly impaired by improper mixtures of alternative fuels, post-production manipulation, improper storage, purposely addition of cheaper ingredients or foreign substances.
- Poor fuel clogs the fuel system, changes the process of combustion and affects the properties of engine oil by soot, which arise by intersection unburned fuel or its components.

Overview of typical damage of vehicles

The most of typical damage is loss of power or irregular engine running or obvious defect of fuel system.

The next damage cathegory is glitch with permanent restrictions function of engine. This damage can be caused by many thinks like previously damage. To locate the problem, the fuel needs to be subscribed from the vehicle and analysed.

The worst damage is crush of engine. According to the range damage, inspector makes the sampling and advanced analysis of fuel and lubricants with the evaluation.

4.1 Petrol station inspection specifics

By inspection in context of this work we mean control performed by the inspector in person at the petrol station. The inspector needs to take a fuel sample. Sampling is performed in the proper way into clean and marked test-tube, which is immediately unmistakably scaled, with the standard CTIA (The Czech Trade Inspection Authority) EN ISO 3170. [26] The sampling protocol always documents conditions and place of collection.

Properties of fuels are evaluated by a set of test on the modern equipment in a laboratory accredited according to CTIA EN ISO/IEC 17025. Clear and strict rules are sure to objectivity.

The results of the test are evaluated with the offsetting uncertainties determination and in relation to relevant quality standards or to relevant requirements of manufacturers vehicles. For judicial resolution of the dispute review is processed by renowned forensic expert.

4.2 Monitoring quality of fuels

The precise composition of fuel is described in the act No. 133/2010 Coll. The act describes requirements for fuels, methods of monitoring composition and quality of fuels and their evidence. Now we will focus on quality of fuels. We show needed quality of the most frequently fuels. They are motor gasoline CTIA EN 228 and motor oil fuel CTIA EN 590. Quality indicators of motor gasoline are in the Table 4 and of motor oil fuel in the Table 5.

Indicator of quality	**Unit**	**Minimum**	**Maximum**
1. Octane number by research method	-	95	-
2. Octane number by engine method	-	85	-
3. Density at $15°C$	kg/m^3	720.0	775.0
4. Vapor pressure, DVPE method - summer	kPa	-	60.0
5. During the distillation:			
5.1 the evaporated amount at $100°C$	$\%(V/V)$	46.0	-
5.2 the evaporated amount at $150°C$	$\%(V/V)$	75.0	-
5.3 finish temperature distillation	$°C$	-	210
6. Hydrocarbon composition:			
6.1 olefins	$\%(V/V)$	-	18.0
6.2 aromatic hydrocarbons	$\%(V/V)$	-	35.0
6.3 benzen	$\%(V/V)$	-	1.0
7. The content of oxygen	$\%(m/m)$	-	3.7
8. The content of oxygenates:			
8.1 methanol	$\%(V/V)$	-	3.0
8.2 ethanol	$\%(V/V)$	-	10.0
8.3 isopropanol	$\%(V/V)$	-	12.0
8.4 tercbutanol	$\%(V/V)$	-	15.0
8.5 isobutanol	$\%(V/V)$	-	15.0
8.6 ethers (containing 5 or more atoms of carbon in molecule)	$\%(V/V)$	-	22.0
8.7 other oxygenates	$\%(V/V)$	-	15.0
9. The content of sulfur	mg/kg	-	10.0
10. The content of lead	mg/l	-	5.0
11. The content of manganese	mg/l	-	6.0
12. Oxidation stability	min	360	-

Table 4 Quality indicators of motor gasoline [27]

Indicator of quality	Unit	Minimum	Maximum
1. Cetane number	-	51.0	-
2. Cetane index	-	46.0	-
3. Density at $15°C$	kg/m^3	-	845.0
4. During the distillation:			
4.1 the evaporated amount at $250°C$	$\%(V/V)$	-	65.0
4.2 the evaporated amount at $350°C$	$\%(V/V)$	85.0	-
4.3 temperature at which distils 95% (V/V)	$°C$	-	360
5. The content of poly cyclic aromatic hydrocarbons	$\%(mim)$	-	8.0
6. The content of FAME	$\%(V/V)$	-	7.0
7. The content of sulfur	mg/kg	-	10.0
8. The content of water	mg/kg	-	200.0
9. Filterability:			
9.1 winter	$°C$	-	-20
9.2 transitional period	$°C$	-	-10
10. Flash-point	$°C$	more than 55	-
11. Oxidation stability	g/m^3	-	25.0

Table 5 Quality indicators of motor oil fuel [27]

4.3 Who performs the inspections

The inspections can be performed by a public or private sector representative. Public sektor control petrol stations for state. In the Czech Republic, The Czech Inspection Authority (CTIA) is state organization, which makes the inspections. CTIA inspects all products in Czech republic except food and tobacco products. An example of the inspection body from private sector is the SGS company, which is the largest worldwide inspection company. It controls petrol stations for costumer or legal entity.

4.3.1 The Czech Inspection Authority (CTIA)

The Czech Trade Inspection Authority (CTIA) is an administrative government institution falling under the jurisdiction of the Ministry and Trade of the Czech Republic. The minister of industry and trade appoints the director general of the CTIA. The CTIA was established under Act No. 64/1986 Coll. It consists of the Central Inspectorate and Regional Branch Inspectorates. The Regional Branch Inspectorate has offices in major regional cities.

The Czech Trade Inspection Authority controls and monitors individuals and businesses, which are selling or buying products in the Czech republic, are providing services or similar activities on the domestic market, are providing consumer credit, and are operating marketplaces.

Penalities

"The Czech Trade Inspection Authority may, in some cases, impose fines of up to 50 million CZK for violations of laws committed by the audited subject. For minor violations, CTIA inspectors may impose immediate fines of up to 5,000 CZK. This also applies to private individuals selling either produce from small farms or forest-harvested

crops. In addition to fines, the CTIA also imposes bans on the sale of products, or their introduction onto the Czech market, if these products do not comply with Czech regulations." [28]

4.3.2 Private sector notified bodies

SGS is the world's leading inspection, verification, testing and certification company. They operate a network of more than 1,650 offices and laboratories around the world. Their service comprise inspection, testing, certification and verification.

SGS deals with petrol station inspections in the Czech Republic. They control quality of fuel for private sector. SGS performs the fuel inspections based on contracts with large fuel petrol station networks.

SGS make independent inspections and they give so-called Quality seal to the petrol station, which are satisfing periodically the inspections. Reason for creation quality seal is described in their web page [29]. They write than independent inspections show significant differences in fuel quality, but this inspections does not facilitate orientation in station market for the normal customer.

Program Quality seal with the trademark is used to guide the motorists in search of quality petrol station. In the petrol stations systematic care about the quality of fuel and these fuels signify European standards in the long term.

4.4 Inspection process data

We have described informaiton about petrol station inspections. Now we discus application of inspection. CTIA is a member of the open-data initiative, therefore we were able to perform an analysis based on the real data. The Czech Trade Inspection Authority publishes data about it's inspections on web page [28]. The petrol stations with license need to fulfill requirements by a legal regular and CTIA may do periodicaly inspection.

We present process of inspections and data from CTIA. In the Table 6 we show the data. In the CTIA web page are approximately 3 800 publicly accessible petrol stations. CTIA control approximately 3 800 petrol stations every year. It means that each petrol staton is controled once per year in average.

125 inspections are detected as inconvenient (from 3 800 inspection), which is 3.3 % of all inspections. The averange fine for unsatisfactory inspection is 210 000 CZK and total fines are 26.25 million CZK.

Number of control per year	3 800
Number of petrol stations	3 800
Detected controls as inconvenient (per year)	125
Detected controls as inconvenient (per year) [%]	3.3
Averange penalty [CZK million]	0.21
Total penalties [CZK million]	26.25

Table 6 Petrol station inspection data provided by CTIA

Unsatisfactory stations not detected (per year)	121
Unsatisfactory stations not detected (per year) [%]	3.2
Total number of violations	246
Average harm [million CZK]	0.42
Total harm [million CZK]	103.3

Table 7 Estimated petrol station inspection data

In the Table 7, we estimate that there are exists another 3.2 % unsatisfactory petrol stations, because the total number of petrol stations who contravene some regulation is not known. Consequently all together 6.5 % of all stations contravene at least one regulation. We estimate the averange and total harm too. Averange harm will be twice the averange cost, which is 420 000 CZK. We have summary 6.5 % cheating petrol stations, then the total harm is 103.3 million CZK.

We want to improve inspection and reduce cheating of petrol stations. We create the following prediction. We expect decrease cheating of petrol stations. In the time when we start new control system the caught violations of petrol stations very fast increases, because we have better control system and we catch more cheating petrol stations. At this moment another petrol stations register change in the inspections and some of them finish with cheating. Consequently, the total violations will be decreases. After the transition caught violations and total violations settle down.

We focus on the three important thinks about inspections:

- Unpredictability of inspections
- Save costs
- Reduce bribing inspectors

We ensure unpredictability of inspections by sampling from different inspection strategies. We initialize game theory and we solve it by LP. We save costs by solving path of inspectors as Travel Inspection Problem (TSP). Inspectiors have tendency for taking bribes; therefore, we want reduce it. We use our program for it. The program generates results every month in the morning before the inspection day. Then inspectors do not know what petrol stations they will control. We have one more measure against bribing. We choose every time another inspector for the petrol station.

5 Problem formalization

Formally, the inspection problem of petrol stations can be seen as an optimization problem with inspection optimizing a joint criterion function for the inspector. In the following sections, we define the inspection policies of the inspector, benefits and costs of the cheating inspection stations. We formulate three utilities function. First utility function is overall cheating level of the petrol stations. Second utility function defines cost of potentially cheating the petrol station in individual months and third utility function represents cost of route for the inspector. Then we use mathematical programming that allows us to capture constraints posed on the our problem easily.

5.1 Game theoretical model

We create two-person game between inspector and petrol stations. For simplicity, we define inspector as first player and N petrol stations as second player, which decides where he will cheat (in which petrol stations) and when he will cheat. For example, second player chooses the five petrol stations from $N = 10$ petrol stations where he cheats. And then he chooses January, May, June, October and December when he cheats in the five choosen petrol stations. The task of the inspector (first player) is calculating where and when the second player cheats. This means that he calculates in which petrol stations the second player cheats and in which months the second player cheats.

We use asymmetric zero-sum game, the inspector's utils are equal to petrol station's utils. Our game is imperfect information, because the inspector does not know petrol station's decision. The game is non-cooperative and we calculate Nash equilibrium. This means that each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy. All this types of game are described in Chapter 3.

5.2 Physical inspection of potentially cheating petrol stations

We define the problem where one inspector must visit the given petrol stations using the least costs. More precisely we plan route into several petrol stations for the one inspector. We define time period such as one month for the route (or for the one schedule for inspector).

When we get properties about the petrol stations. We get the budget for the inspections. We find potentially cheating petrol stations, which we going to inspect. Then we search months for the choosen petrol stations. We select the months according to the probability of cheating petrol stations in the each month. We calculate costs of routes and we check it with budget. The final cost of the year inspections can not be bigger than budget. User give us month and we construct route and schedule for the inspector in the month.

5.3 Petrol station attributes

Real petrol station has a lot of properties, but we can not comprise all of the properties, we will use just a subset of the properties. In this section, we describe properties, which we consider. We define the following parametrs for each petrol stations.

1. The population of the town (or the population of the nearest town), where the petrol station is located.
2. Size of the road, where the petrol station is located.
3. The fine, which the petrol station get, when the petrol station cheats.
4. The profit of the cheating petrol station.
5. The fuel consumption chart according to motnhs. The fuel consumption chart has the same values for the all petrol station. This parametr is for calculating the probability of cheating petrol station in the each month.

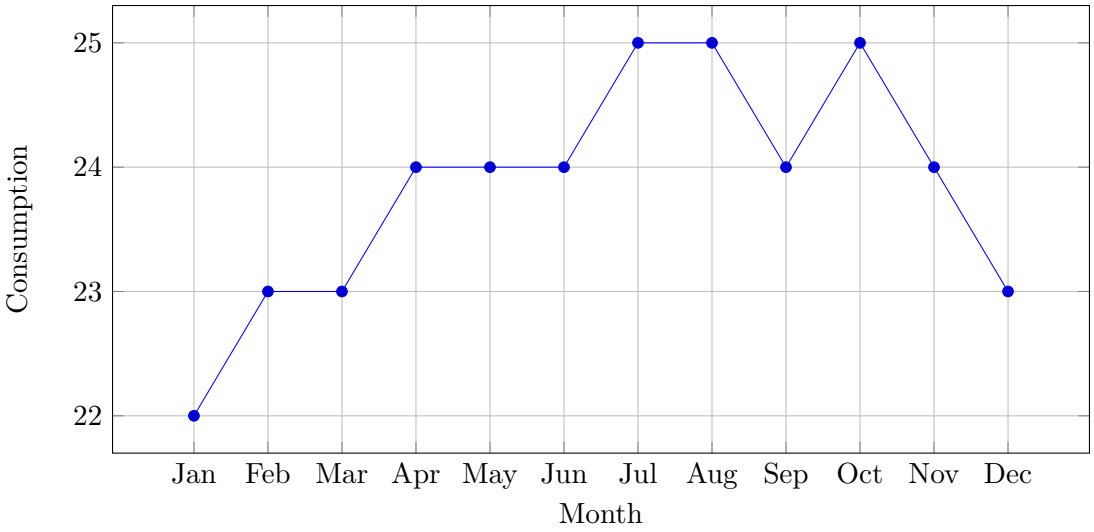


Figure 14 Fuel consumption chart (fuel consumption is in thousand barrels per day) [30]

The population of the town is readed from the table created from the cities and their population. The petrol stations has adress and we find the population in the table according to the adress. We find the type of road in the OSM. We derive size of road from the type of road, for example the biggest road is 'motorway' and smaller road is 'secondary_link'.

The fine and the profit of petrol station, we deduce from attendance of petrol stations. We ascertained attendance of petrol station for one day. We have number of costumers from the several petrol stations from each type of route. We decide types of routes to the groups and we calculate average number of costumers c for the each group. We have the following types of route (from the largest to the smallest): motorway, motorway_link, trunk, trunk_link, primary, primary_link, secondary, secondary_link, tertiary, tertiary_link, road, residential, living_street, unclassified. We found average fine from the CTIA, it is 360 000 CZK. Average fine for one costumer is 997 CZK. Then we calculate fines for the each group according to the following figure:

$$\text{fine}_i = c_i \cdot 997$$

Where c_i is number of costumers in the i-th group. The Table 8 describes fines in each group.

Group of route types	Calculation	Result [CZK]
motorway, motorway_link	$700 \cdot 997$	$= 697900$
trunk, trunk_link	$429 \cdot 997$	$= 427713$
primary, primary_link	$319 \cdot 997$	$= 318043$
secondary, secondary_link	$201 \cdot 997$	$= 200307$
tertiary, tertiary_link	$160 \cdot 997$	$= 159520$
road, residential, ...	$130 \cdot 997$	$= 129610$

Table 8 Fines for petrol stations

The profit petrol station from the cheating, we deduce from the previous numbers of costumers. We assume that each costumer buys approximately 40 l of fuel and the petrol station has 3 CZK profit from 1 l if the petrol station cheats. In the Table 9 are the results of the calculating profit.

Group of route types	Calculation	Result [CZK]
motorway, motorway_link	$700 \cdot 40 \cdot 3$	$= 84000$
trunk, trunk_link	$429 \cdot 40 \cdot 3$	$= 51480$
primary, primary_link	$319 \cdot 40 \cdot 3$	$= 38280$
secondary, secondary_link	$201 \cdot 40 \cdot 3$	$= 24120$
tertiary, tertiary_link	$160 \cdot 40 \cdot 3$	$= 19200$
road, residential, ...	$130 \cdot 40 \cdot 3$	$= 15600$

Table 9 Petrol stations profits from the cheating

We assume the graph of consumption from the information in the report Monthly Energy Review January 2015 from U.S. [30].

5.4 Utility functions

As was mentioned above, we use three utility functions:

1. Utility function f_1 for calculate cost of cheating petrol stations
2. Ulitity function f_2 for calculate cost of potentially cheating the petrol station in each month.
3. Utility function f_3 for calculate cost of route (distances)

We describe each utility function and their composition. We use parameters defined above. Each parametr in utility function is multiplied by constant.

First utility function f_1 is consisted from the population of the town and from size of the road, where the petrol station a is located.

$$f_1(a) = \alpha \cdot \text{population}_a + \beta \cdot \text{road}_a,$$

$$\alpha = 0.5$$

$$\beta = 0.5$$

We define number 0 as the biggest road and the higher number means smaller type of road.

We define two options of the utility function f_2. We construct two model of our program and for each we define different utility function f_2. We describe the models in the Chapter 6.

The fine, the profit of the cheating petrol station a and the fuel consumption in the month m form second utility function f_2 for our first model of application.

$$f_2(a, m) = -\alpha \cdot \text{fine}_a - \beta \cdot \text{profit}_a - \gamma \cdot \text{fuel\_consumption}_m$$
$$\alpha = 0.4$$
$$\beta = 0.3$$
$$\gamma = 0.3$$

The fine, the profit of the list cheating petrol stations (petrol stations $1, 2, ..., x$) and the fuel consumption in the month m form second utility function f_2 for our second model of application. For the final fine, we sum fines of the each petrol stations. We sum profits of the each petrol stations by the same way.

$$f_2(m) = \alpha \cdot \text{fine} + \beta \cdot \text{profit} + \gamma \cdot \text{fuel\_consumption}_m$$
$$\alpha = 0.4$$
$$\beta = 0.3$$
$$\gamma = 0.3$$
$$\text{fine} = \sum_{i=1}^{x} \text{fine}_i$$
$$\text{profit} = \sum_{i=1}^{x} \text{profit}_i$$

Third utility function f_3 is very simple. It is equal distance between petrol stations a and b. It does not contain constant.

$$f_3(a, b) = \text{distance}_{ab}$$

5.5 Inspector schedule

The program creates schedule for the inspector. Schedule is list of the places, which inspector going to control in the given month. Inspector always starts and ends his route in the Prague office. Program draws the route in the open maps called GraphHopper [31].

6 Algorithms

In this chapter, we introduce the algorithms which we use to compute the solution of the problem. First, we use algorithm for computing the game theoretical part. We apply this algoritm twice. First utilization is for find potentially cheating petrol stations and calculating probabilities of cheating petrol station in the months is second application. We design Travel Salesman problem for the finding route.

We construct two models. First model is simplier. The second model continue from the first model and the second model improving the first model. First model is with fixed number of petrol stations and months. The second model calculate this numbers and the second model need to get budget, which we can spend for the inspections.

6.1 Model with given number of petrol stations and months

6.1.1 Finding potentially cheating petrol stations

First, we calculate probability of cheating for each petrol station independent of time. We have N petrol stations. We get budget B_S for visiting petrol stations. We construct utility function $f_1(a)$, which is defined above in section 5.4.

We create utility matrix M_1. Columns in matrix are petrol stations, which can potentially cheat. Number of cheating petrol stations can be from 0 to N. We construct permutation from zeros and ones. Permutation's size is N. Number 1 means that petrol station cheat, number 0 means that the petrol station is not cheating. Inspections are in the rows in matrix. We mark them by similar way like petrol stations in columns. Number 1, if inspector realizes the inspection and number 0 if he does not realize inspection.

For example: matrix M_1 for $N = 3$ and $B_S = 1$:

			0	1	0	0	1	0	1	1
			0	0	1	0	1	1	0	1
			0	0	0	1	0	1	1	1
1	0	0								
0	1	0								
0	0	1								

Where N is the number of all petrol stations (from which we choose). The permutations in the rows allow to choose only one petrol station. If we want to choose two petrol stations, the matrix will look like following matrix:

			0	1	0	0	1	0	1	1
			0	0	1	0	1	1	0	1
			0	0	0	1	0	1	1	1
1	1	0								
0	1	1								
1	0	1								

According to utilitity function $f_1(a)$ we calculate utility values $u_1, u_2, ..., u_N$ for each petrol station. Then we calculate values in the utility matrix M_1. We will find the minimum value, it means that smallest number in matrix is the best. We fill the matrix pursuant to this rules:

Inspection	Station			
0	0	$\rightarrow$		0
0	1	$\rightarrow$	$+$	$u_i, 1 \leq i \leq N$
1	0	$\rightarrow$	$+$	$\frac{u_i}{10}, 1 \leq i \leq N$
1	1	$\rightarrow$	$-$	$u_i, 1 \leq i \leq N$

Final matrix will be:

	0 0 0	1 0 0	0 1 0	0 0 1	1 1 0	0 1 1	1 0 1	1 1 1
1 0 0	$\frac{u_1}{10}$	$-u_1$	$\frac{u_1}{10} + u_2$	$\frac{u_1}{10} + u_3$	0	$\frac{u_1}{10} +$ $u_2 + u_3$	0	$-u_1 +$ $u_2 + u_3$
0 1 0	$\frac{u_2}{10}$	$\frac{u_2}{10} + u_1$	$-u_2$	$\frac{u_2}{10} + u_3$	0	0	$\frac{u_2}{10} +$ $u_1 + u_3$	$u_1 -$ $u_2 + u_3$
0 0 1	$\frac{u_3}{10}$	$\frac{u_3}{10} + u_1$	$\frac{u_3}{10} + u_2$	$-u_3$	$\frac{u_3}{10} +$ $u_1 + u_2$	0	0	$u_1 +$ $u_2 - u_3$

Now we solve following LP:

$$\begin{gathered} \min \ z \\ \sum_{i=1}^{R} p_i \cdot u_{ij} \leq z \quad j \in 1, 2, ..., C \\ \sum_{i=1}^{R} p_i = 1 \end{gathered} \tag{16}$$

Where R is number of rows in the matrix M_1 and C is number of columns. p_i is probability of potentially cheating petrol station. u_{ij} is utility value in the matrix M_1 and z is slack variable.

6.2 Sampling

We calculated probabilites of potentially cheating petrol station $p_1, ..., p_R$. Now we have to choose B_S petrol stations. Number of petrol stations B_S is provided by calculation in LP (problem desribed above). We use random generator for production real number r, which satisfies the condition $0 \leq r \leq 1$. We choose one probability from the result of LP (probabilities of potentially cheating petrol stations) according to real number r. We introduce exapmle of sampling for three petrol station:

We define $N = 3$, $B_S = 1$ We have result of LP: $\{p_1, p_2, p_3\}$. Then we generate real number r.

If $0 \leq r \leq p_1$ then we choose first result of LP (first petrol station).
If $p_1 < r \leq p_2$ then we choose second result of LP (second petrol station).
If $p_2 < r \leq p_3$ then we choose third result of LP (third petrol station).

If we define budget for inspection petrol stations $B_S > 1$, then LP reurn list of probabilities by the same size as for $B_S = 1$ (if N is the equally). The difference is in the permutation of the inspections in matrix M_1.

For example: $N = 3, B_S = 2$, we define matrix M_1:

			0	1	0	0	1	0	1	
			0	0	1	0	1	1	0	results
			0	0	0	1	0	1	1	
1	1	0								p_1
0	1	1								p_2
1	0	1								p_3

Then we calculate utility values in matrix M_1 and calculate LP. Result of LP is list of probabilities $\{p_1, p_2, p_3\}$. In sampling we choose for example probability p_1. It means, that we choosen first and second petrol stations for inspection.

6.3 Search months, in which the petrol station will cheat

We have choosen petrol stations for the inspection. Now we make following calculation for each choosen petrol station. We calculate probability of the petrol station for each month. We have 12 months. We get budget B_M for months, in which we inspect the petrol station. We construct function $f_2(a, m)$ for petrol station a. The function f_2 is defined in section 5.4.

We create utility matrix M_2 by the same way as utility matrix M_1. But columns in matrix are months, in which the petrol station can potentially cheat. Months, when we inspect the petrol station, is in rows. We make permutations, calculate values in matrix M_2 and calculate the same LP as in the subsection 6.1.1, but for months.

When we have the probabilities of months, we sample on the probabilities as in the subsection 6.2. Then we save choosen months and we repeat all of this subsection for the next choosen petrol station.

6.4 Calculating route

We have list of the petrol stations to inspection and months when we inspect each petrol station. Now we construct route for the inspector. The begining of route is in the Prague office. Then the inspector continues to the choosen petrol stations office. The end of the route is in the Prague office as well. We choose month μ when we want to calculate route.

We have N_μ choosen petrol stations $\{s_1, s_2, ..., s_{N_\mu}\}$ in month μ. First we calculate distances between each two petrol stations $\{s_1, s_2, ..., s_{N_\mu}\}$. We construct utility matrix M_3 by using utility function $f_3(a, b)$ (is defined in section 5.4). $f_3(a, b)$ means distance between petrol stations a and b. We create the following utility matrix M_3:

	s_1	s_2	$\cdots$	s_{N_μ}
s_1	max_value	$f_3(s_1, s_2)$	$\cdots$	$f_3(s_1, s_{N_\mu})$
s_2	$f_3(s_2, s_1)$	max_value	$\cdots$	$f_3(s_2, s_{N_\mu})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_{N_μ}	$f_3(s_{N_\mu}, s_1)$	$f_3(s_{N_\mu}, s_2)$	$\cdots$	max_value

Where *max_value* means the maximum number of integer. We used it, because we will find minimum value in the LP and we can not allow route from the petrol station to the same petrol station.

Then we calculate the LP problem:

$$\begin{aligned}
&\min \sum_{i=0}^{N_\mu} \sum_{j\neq i, j=1}^{N_\mu} c_{ij} x_{ij} \\
&\sum_{i=0, i\neq j}^{N_\mu} x_{ij} = 1 && j = 0, ..., N_\mu \\
&\sum_{j=0, j\neq i}^{N_\mu} x_{ij} = 1 && j = 0, ..., N_\mu \\
&0 \leq x_{ij} \leq 1 && i, j = 0, ..., N_\mu \\
&u_1 = 1 \\
&2 \leq u_i \leq N_\mu && \forall i \neq 1 \\
&u_i - u_j + 1 \leq (N_\mu - 1)(1 - x_{ij}) && \forall i \neq 1, \forall j \neq 1
\end{aligned} \tag{17}$$

Where c_{ij} is value in the matrix M_3 (it is distance between petron stations). $u_1, \cdots, u_{N_\mu}$ are variables which guarantees that we make one continuous route (not two or more separated routes). $u_i = t$ if the petrol station i is visited in step t.

Result of the LP is matrix X containing zeros and ones. Matrix X can look like:

	s_1	s_2	$\cdots$	s_{N_μ}
s_1	0	1	$\cdots$	0
s_2	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_{N_μ}	1	0	$\cdots$	0

It means that the route starts in the petrol station (realistically in the Prague office) s_1, then the inspector continue to the petrol station s_{N_μ} and he ends in the petrol station (realistically in the Prague office) s_2.

$$s_1 \rightarrow s_{N_\mu} \rightarrow s_2 \rightarrow s_1$$

We obtain list of petrol stations to inspecting in the month μ.

6.5 Model with budget

This model is similar like previous model. We introduce differences from the model described above.

We do not know how many we inspect petrol stations. We calculate it by LP. Therefore we construct different matrix for finding potentially cheating petrol stations. Number of cheating petrol stations can be 0 to N.

For example: matrix M_1 for $N = 3$:

			0 0 0	1 0 0	0 1 0	0 0 1	1 1 0	0 1 1	1 0 1	1 1 1
0	0	0								
1	0	0								
0	1	0								
0	0	1								
1	1	0								
0	1	1								
1	0	1								
1	1	1								

In the matrix M_1 we construct permutations for every number of petrols stations. It is different from previous model where we create permutation with the fixed number of ones (number of petrol stations). The rules for filling the matrix are the same. Therefore, the filled matrix M_1 look like following matrix.

Final matrix will be:

	0 0 0	1 0 0	0 1 0	0 0 1	1 1 0	0 1 1	1 0 1	1 1 1
0 0 0	0	u_1	u_2	u_3	u_1+u_2	u_2+u_3	u_1+u_3	u_1+ u_2+u_3
1 0 0	$\frac{u_1}{10}$	$-u_1$	$\frac{u_1}{10}+u_2$	$\frac{u_1}{10}+u_3$	0	$\frac{u_1}{10}+$ u_2+u_3	0	$-u_1+$ u_2+u_3
0 1 0	$\frac{u_2}{10}$	$\frac{u_2}{10}+u_1$	$-u_2$	$\frac{u_2}{10}+u_3$	0	0	$\frac{u_2}{10}+$ u_1+u_3	u_1- u_2+u_3
0 0 1	$\frac{u_3}{10}$	$\frac{u_3}{10}+u_1$	$\frac{u_3}{10}+u_2$	$-u_3$	$\frac{u_3}{10}+$ u_1+u_2	0	0	u_1+ u_2-u_3
1 1 0	$\frac{u_1}{10}+\frac{u_2}{10}$	$-u_1+$ $\frac{u_2}{10}$	$\frac{u_1}{10}-u_2$	$\frac{u_1}{10}+$ $\frac{u_2}{10}+u_3$	$-u_1-$ u_2	$\frac{u_1}{10}-$ u_2+u_3	$-u_1+$ $\frac{u_2}{10}+u_3$	$-u_1-$ u_2+u_3
0 1 1	$\frac{u_2}{10}+\frac{u_3}{10}$	u_1+ $\frac{u}{10}+\frac{u_3}{10}$	$-u_2+$ $\frac{u_3}{10}$	$\frac{u_2}{10}-u_3$	u_1- $u_2+\frac{u_3}{10}$	$-u_2-$ u_3	u_1+ $\frac{u_2}{10}-u_3$	u_1- u_2-u_3
1 0 1	$\frac{u_1}{10}+\frac{u_3}{10}$	$-u_1+$ $\frac{u_3}{10}$	$\frac{u_1}{10}+$ $u_2+\frac{u_3}{10}$	$\frac{u_1}{10}-u_3$	$-u_1+$ $u_2+\frac{u_3}{10}$	$\frac{u_1}{10}+$ u_2-u_3	$-u_1-$ u_3	$-u_1+$ u_2-u_3
1 1 1	$\frac{u_1}{10}+$ $\frac{u_2}{10}+\frac{u_3}{10}$	$-u_1+$ $\frac{u_2}{10}+\frac{u_3}{10}$	$\frac{u_1}{10}-$ $u_2+\frac{u_3}{10}$	$\frac{u_1}{10}+$ $\frac{u_2}{10}-u_3$	$-u_1-$ $u_2+\frac{u_3}{10}$	$\frac{u_1}{10}-$ u_2-u_3	$-u_1+$ $\frac{u_2}{10}-u_3$	$-u_1-$ u_2-u_3

We calculate choosen petrol stations by calculating probabilities and sampling like in the previous model, but in this model we also calculate number of choosen petrol stations.

We calculate months for all choosen petrol stations together. We do not know how many months we could choose. We construct a litle different utility function, because we calculate it for all petrol stations (not for each petrol station). The utility function is defined in the section 5.4. Then we calculate choosen month by the same way like we choose the petrol stations.

We continue and we calculate route for the each month where we inspect a petrol station. We sum the distances of the routes from each month and we check it with the budget. Budget is in the Czech crowns. Therefore, we convert distance (in km) d to the cost of the distance d by following way:

$$B_e = d \cdot c$$

Where B_e is exhausted budget for the year and c is cost for one kilometer.

Now we can compare exhausted budget B_e with the final budget B. We repeat all the calculates (from choosing the petrol stations) until the exhausted budget B_e will be smaller than final budget B. If the exhausted budget B_e will be bigger than final budget B, we do not include the last result.

User chooses one month for drawing of the route to the map.

7 Implementation

In this chapter we describe entire program. More precisely, we describe programming language, libraries and database, which we use in this thesis. We create java application using the following extension:

- CPLEX
- GraphHopper
- JavaFX
- Leaflet
- Database

7.1 Program flow architecture

7.1.1 Model with given number of petrol stations and months

The following scheme shows process of the entire first model:

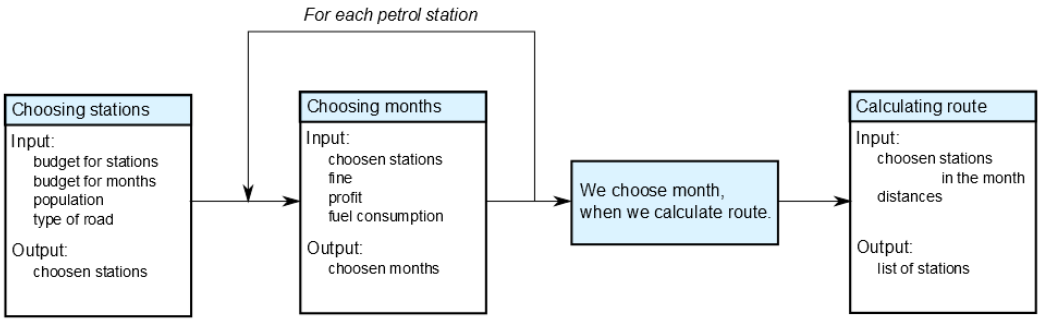


Figure 15 Scheme of model

First, we want to find potentially cheating petrol stations and we find them by IBM CPLEX. We need to define budget for station (how many petrol station we inspect) and budget for months (how many months in a year we inspect one petrol station). Then for each petrol station we find months when we inspect the petrol station. User choose month when we calculate inspector schedule. Then we find the route and we calculate distance.

7.1.2 Model with budget

The following scheme shows process of the entire second model:

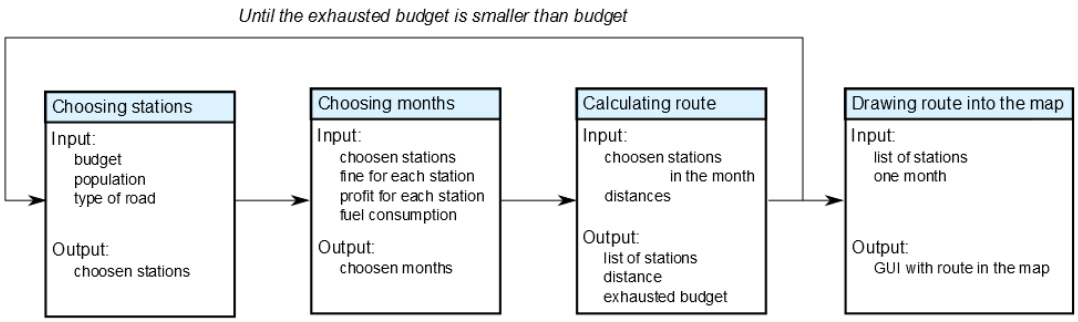


Figure 16 Scheme of program

We want to find potentially cheating petrol stations and we find them by IBM CPLEX. We need to define budget how many money we can spend for the year. We also need to know population of the city where the petrol station is located and type of the road beside the petrol station. The second last parametrs we use for creating utility function. Outputs from this part are the choosen stations.

Then for all petrol stations we find months when we inspect the petrol stations. We know choosen stations from previous step and we need to know fuel consumption, fine and profit of the petrol station. The choosen moths for each petrol station are output.

Then we create the route for inspections in the each month. We have choosen stations in the month and we calculate distances between each two choosen petrol stations. We find the routes by IBM CPLEX. We calculate distance of the all routes (from the each month) and from distance we calculate exhausted budget for the year.

We repeat the calculations until we exhaust the budget. Then the user chooses one month and we create route in GUI by calculating points in GraphHopper.

7.2 IBM CPLEX

IBM CPLEX is solver to solve the linear program instances. In our solution, we formulate three linear integer programs in total and all of them are solved using CPLEX. Twice in the game theoretical part and once in the part of traveling salesman problem. In the game theoretical part, we apply CPLEX for calculate probabilities of potentially cheating petrol stations and probabilities of months when the petrol station cheat.

7.3 GraphHopper

GraphHopper is a fast, flexible and memory efficient Java program providing the route directions service. It works with latest worldwide OpenStreetMap data. The GraphHopper is released under Apache License. [31]

The GraphHopper is applied in the part of traveling salesman problem. We use it for computing distances of any two places in Czech republic represented by their addresses. The distances we set to the CPLEX. But we apply it also for creating route with all

waypoints and instructions for navigation. The waypoints are used for representing route in the map (in application's GUI). Graphopper excepts the web service provides also an API functionality. In our program, we use the API to get the distances.

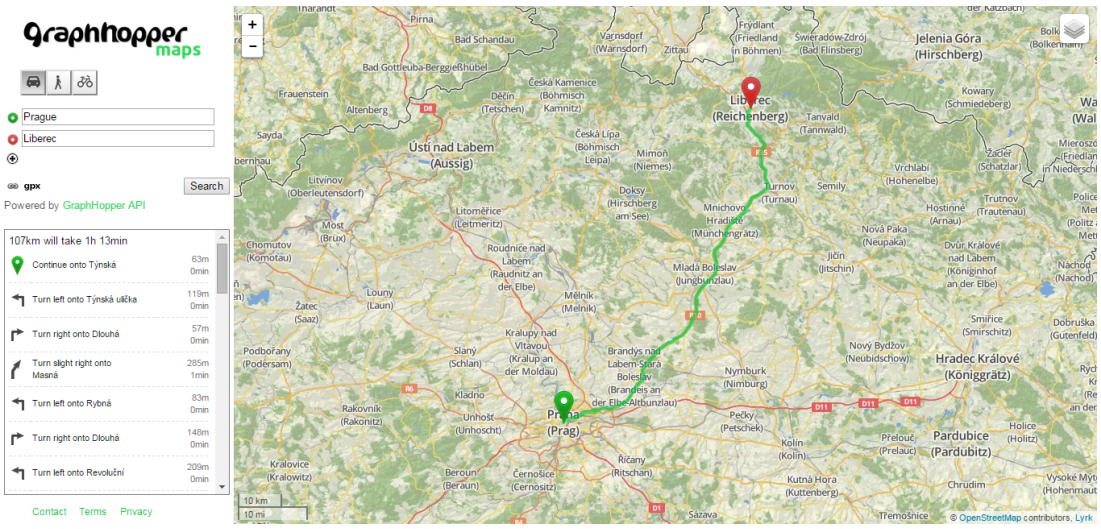


Figure 17 GraphHopper

7.4 JavaFX

JavaFX is built on a software platform based on Java Platform. JavaFX is created by company Oracle. It is used for the development of applications called RIA (Rich Internet Applications). It was established in response to the massive expansion of platforms such as Adobe Flash and Microsoft Silverlight especially. JavaFX completely replaced outdated Swing in March 2014, as a tool for creating GUI in Java. [32]

We use JavaFX for create graphical user interface for our java application. It is look like:

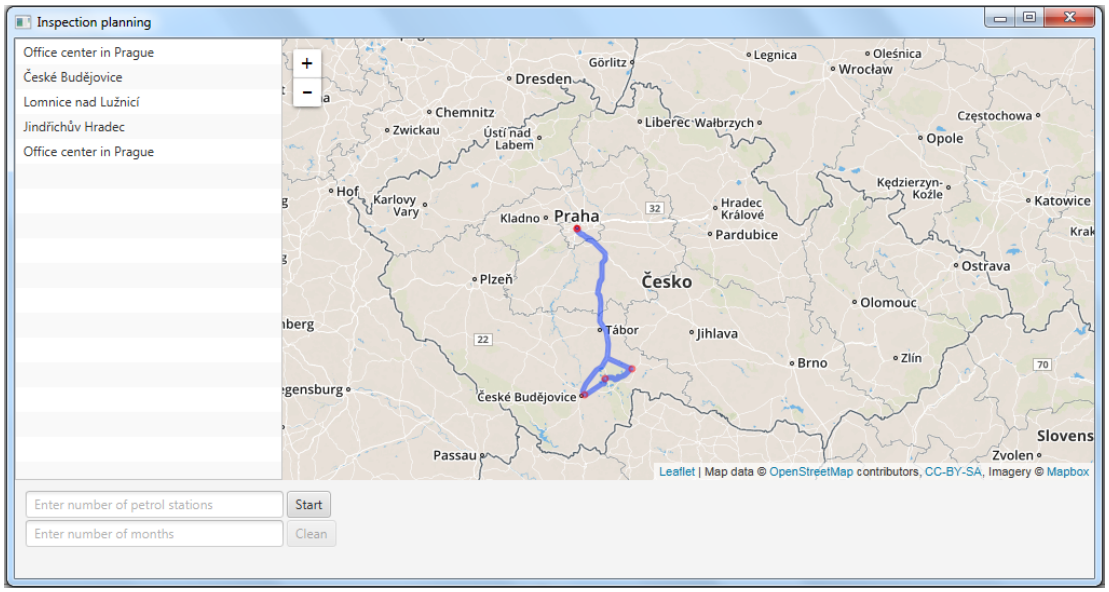


Figure 18 Graphical user interface

In the left column in the Figure 18 is list of the petrol stations, which the inspector have to control. The list starts and ends in the Prague office.

In the JavaFX we apply Leaflet described in the following section.

7.5 Leaflet

Leaflet is a contemporary open-source JavaScript library for interactive maps. It is developed by Vladimir Agafonkin and by a team of specialized contributors.

'Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It can be extended with a huge amount of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.' [33]

We use leaflet in our GUI constructed by JavaFX. In the GUI is box with using leaflet for creating route in the map.

7.6 PostGIS database

We use PostGIS database of the open street maps for the computing and drawing the route. The database is created by Department of Computer Science CTU.

7.7 Deployment scheme

We show implementation in the following scheme:

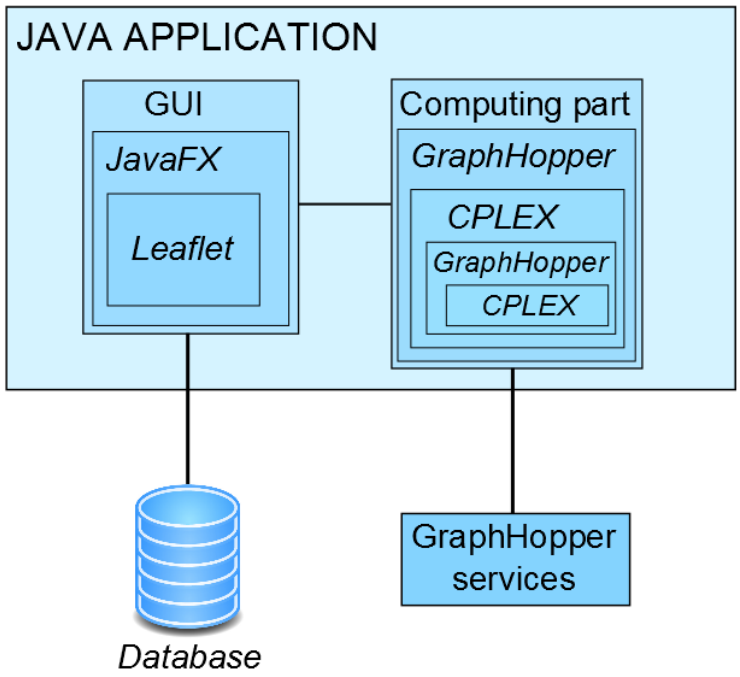


Figure 19 Application Architecture diagram

We compute choosen petrol stations and choosen months in the IBM CPlEX. Then we get the petrol stations and the months to the GraphHopper, where we calculate distances between each two petrol stations in the choosen month by user. We create distance matrix and we put it to the IBM CPLEX for solving the TSP. Now we have list of petrol stations for the inspection. We get the list of petrol stations (more precisely list of cities, where we inspect the petrol stations) to the GraphHopper for creating exactly route with all waypoints. The GraphHopper uses its own services on the server.

Then we create GUI by JavaFX. We draw the exactly way in the Leaflet. Leaflet uses the PostGIS database.

8 Evaluation

In this chapter, we evaluate algorithms described above. We deal with the influence of parameters on algorithm runtime and quality of the solution. Tests in this chapter were performed on real data from the Czech Trade Inspection Authority. We divide this chapter into two sections according to the two models. First model has less variables than the second model. The first model considers the number of petrol stations and number of months when the inspector can inspect the stations. The second model is extended by a budget parameter. We calculate optimal number of petrol stations and number of months for inspection in the second model.

8.1 Model with fixed petrol stations and fixed months

8.1.1 Choose parameters for testing

The model considers the following three parameters:

- budget for the petrol stations
- budget for month
- month when we create route

First, we define budget for the petrol stations $B_S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. We determine constant budget for month $B_M = 7$. Month m when we create route is choosen according to number of petrol station in each month. We choose the month where is the biggest number of petrol stations.

Why we define $B_M = 7$? Because the biggest number of petrol stations in our choosen month will be equal to budget for the petrol staions B_S. Consequently, we calculate TSP with corresponding number of petrol stations. If we choose $B_M = 3$, the biggest number of petrol stations in our choosen month can be diffrent. For example: for $B_S = 4$ we would calculate TSP with four petrol station (if each petrol station has dangerous the same month) and for $B_S = 5$ we would calculate TSP with three petrol stations. That is reason why we define bufget for month equal seven.

We calculate two main tests and we create two graphs for the test.

1. Algorithm performance measurement
2. Final route length measurement

We show first test by graph dependent of the speed program on the number of petrol stations ($B_S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$). Second graph shows dependency of the calculated final distance on the number of petrol stations.

8.1.2 Algorithm performance

We test our program on the notebook with procesor Intel Core i3-2350, 2.30GHz and RAM 4GB. We run the program three times for different values of the petrol stations budget $B_S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. We calculate the same test three times for another group of the petrol stations. We test program on the petrol stations in central bohemian region including Prague. We divide this petrol stations to the three groups

and we test program on each group of the petrol stations. It means that we have three measurements. We perform each measurement for different values of the petrol stations budget $B_S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. We test algorithm performance in each measurement for each petrol stations budget.

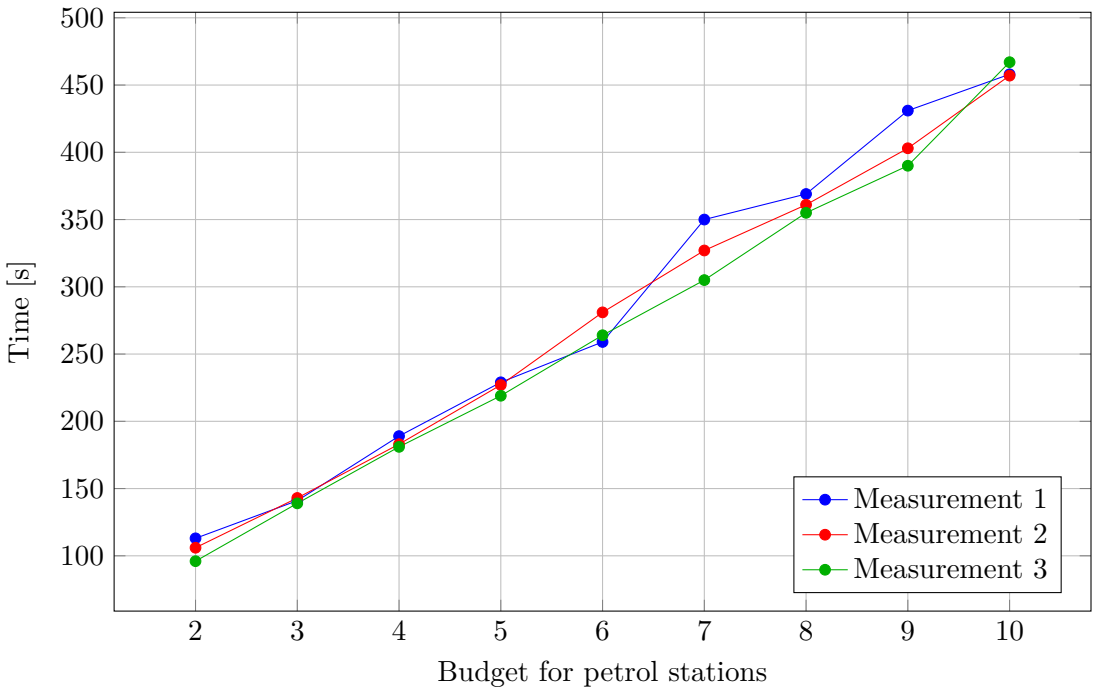


Figure 20 Algorithm performance for various number of petrol stations

Budget for petrol stations	Measurement 1 [s]	Measurement 2 [s]	Measurement 3 [s]	Average [s]
2	113	106	96	105
3	141	143	139	141
4	189	183	181	184
5	229	227	219	225
6	259	281	264	268
7	350	327	305	327
8	369	361	355	362
9	431	403	390	408
10	458	457	467	461

Table 10 Algorithm performance for various number of petrol stations

In the Figure 20 we registered three tests. We get the expected results. With the increasing number of stations is also growing the time required to calculate task. Between three measurements are only very small differents. In the Figure 21 we show average from testing algorithm performance.

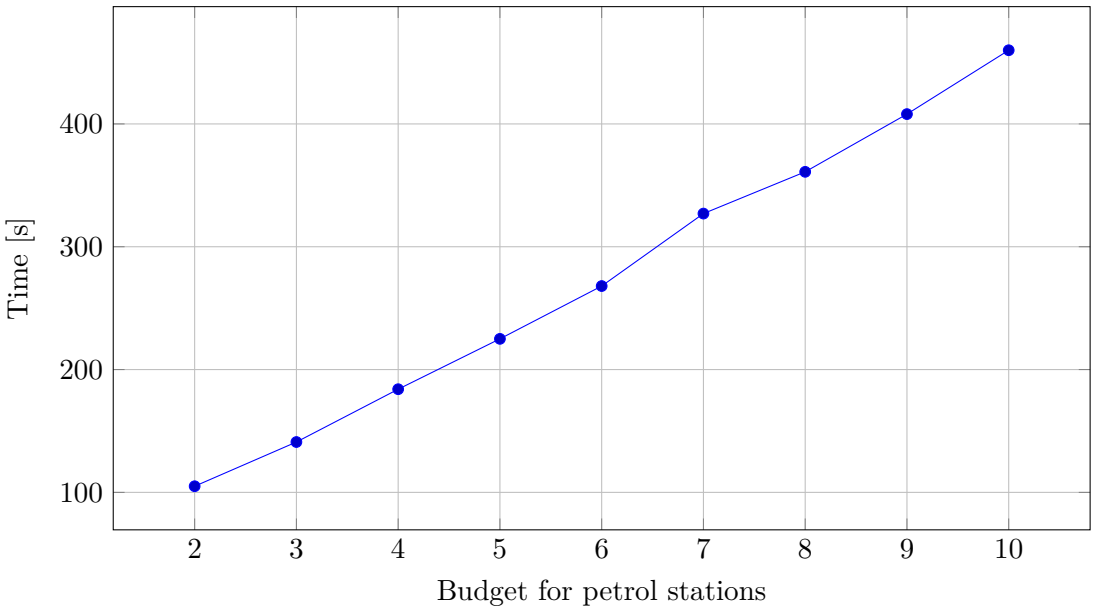


Figure 21 Average of testing algorithm performance for various number of petrol stations

8.1.3 Final route length measurement

In the first test, we test not only algorithm performance but also final route length. Therefore, we measure two variables in the first test. We examine dependency of distance on the number of stations. The results are in the Figure 22. We provide three tests on the three groups of the petrol stations from central bohemian region like in previous section 8.1.2.

Results of three measurements are very different, because we get very unlike routes. We explain the reason for $B_S = 2$, where the different is the biggest. In the first measurement we get the following route:

Prague office → Okřesaneč → Kamýk nad Vltavou → Prague office

The first route's distance is 289 km.

Route of distance measurement 2:

Prague office → Tuchlovice → Petrovice → Prague office

The distance of the second route is 221 km.

Route of distance measurement 3:

Prague office → Prague 4 → Prague 3 → Prague office

In the third measurement, length of route is the smallest, because we inspect petrol stations only in the Prague. That is reason why in the graph are very noticeable differences between each measurenent.

We can see that with increasing number of petrol stations differences between distances of three measurement are smaller. If we construct route with more petrol stations, differences are not as big as for two petrol stations, where difference is big. We show examples.

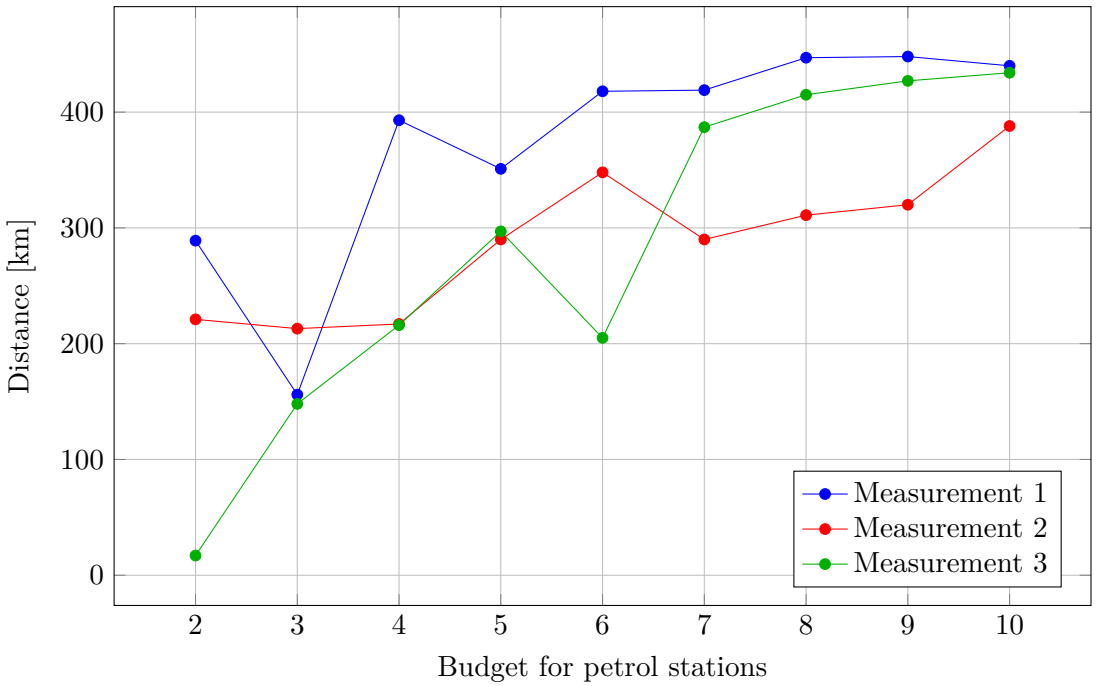


Figure 22 Dependency of the final distance route on the number of petrol stations

Budget for petrol stations	Measurement 1 [km]	Measurement 2 [km]	Measurement 3 [km]	Average [km]
2	289	221	17	176
3	156	213	148	172
4	393	217	216	275
5	351	290	297	312
6	418	348	205	323
7	419	290	387	365
8	447	311	415	391
9	448	320	427	398
10	440	388	434	420

Table 11 Dependency of the final distance route on the number of petrol stations

Now we can compare Figure 24 and Figure 25.

We have number of stations $B_S = 2$. We get very long route, when one station of the choosen petrol stations is long way from Prague office (in the Figure 24a). Unlike in the Figure 24b, we can get all the choosen petrol stations in the Prague and the route will be very short. This is why the differences are so big.

If we have number of stations $B_S = 10$, the one remote petrol station does not cause very big different in final distance. Probability that we get all choosen petrol stations in the Prague is very small. This is why the differences are smaller than in the previous example with $B_S = 2$.

In the average with the increasing number of stations is also growing the distances of the final route.

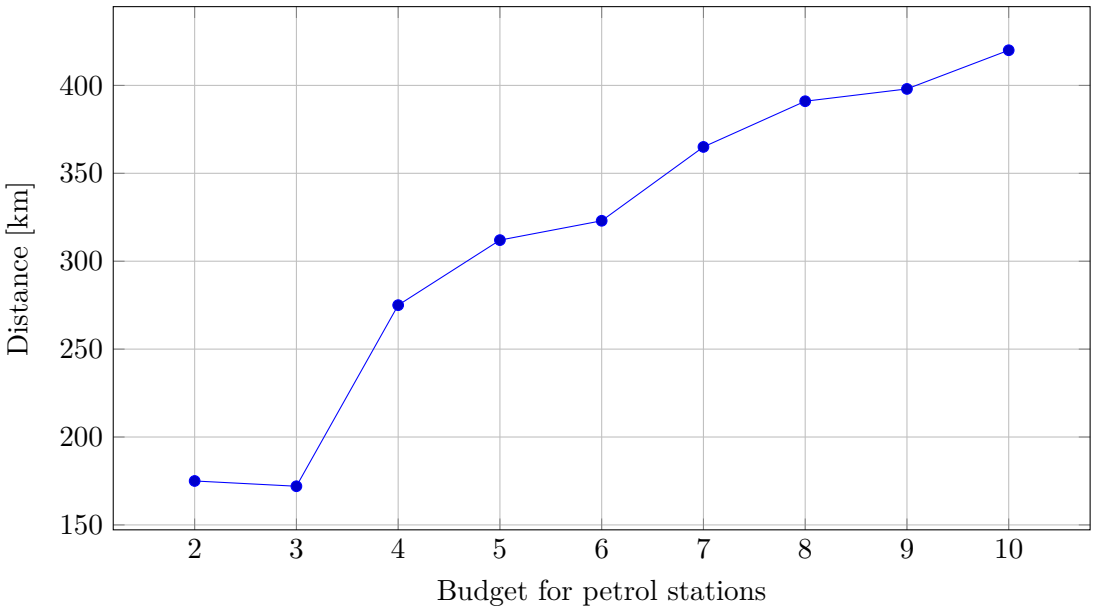
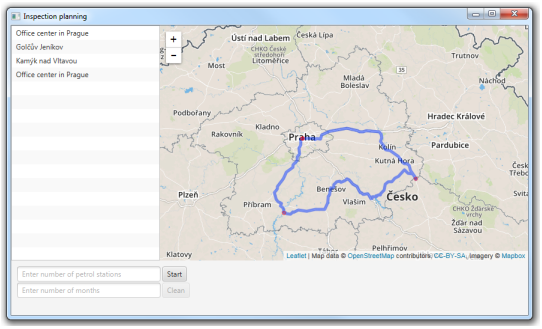
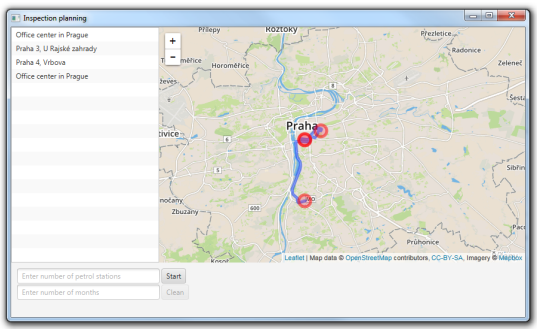


Figure 23 Average of testing dependency of the final distance route on the number of petrol stations

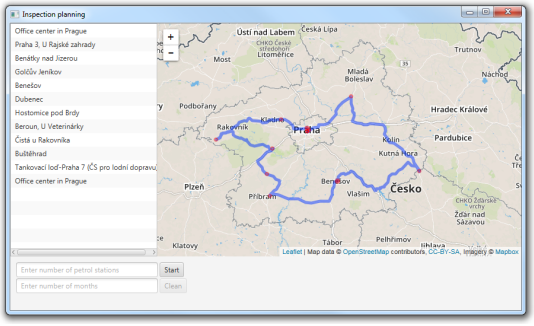


a) Route from measurement 1

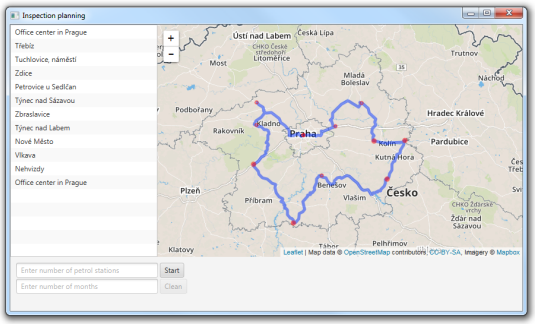


b) Route from measurement 3

Figure 24 Example for two petrol stations $B_S = 2$



a) Route from measurement 1



b) Route from measurement 3

Figure 25 Example for ten petrol stations $B_S = 10$

8.2 Model with budget

8.2.1 Choose parameters for testing

The model considers the following parameters:

- budget
- price for one kilometer

We define the budget for spending money

$$B = \{50000, 70000, 100000, 150000, 200000, 250000, 300000\}$$

and we determine constant price for one kilometer $p = 7$. Both parameters are in CZK.

We test two properties of the model (the same properties like in the previous tests in Section 8.1):

1. Algorithm performance measurement
2. Final route length measurement

8.2.2 Algorithm performance

We measure speed of the second model for different budget values. We provide the same test three times for another group petrol stations like in previous tests. All of the tested petrol stations are from central bohemian region including Prague.

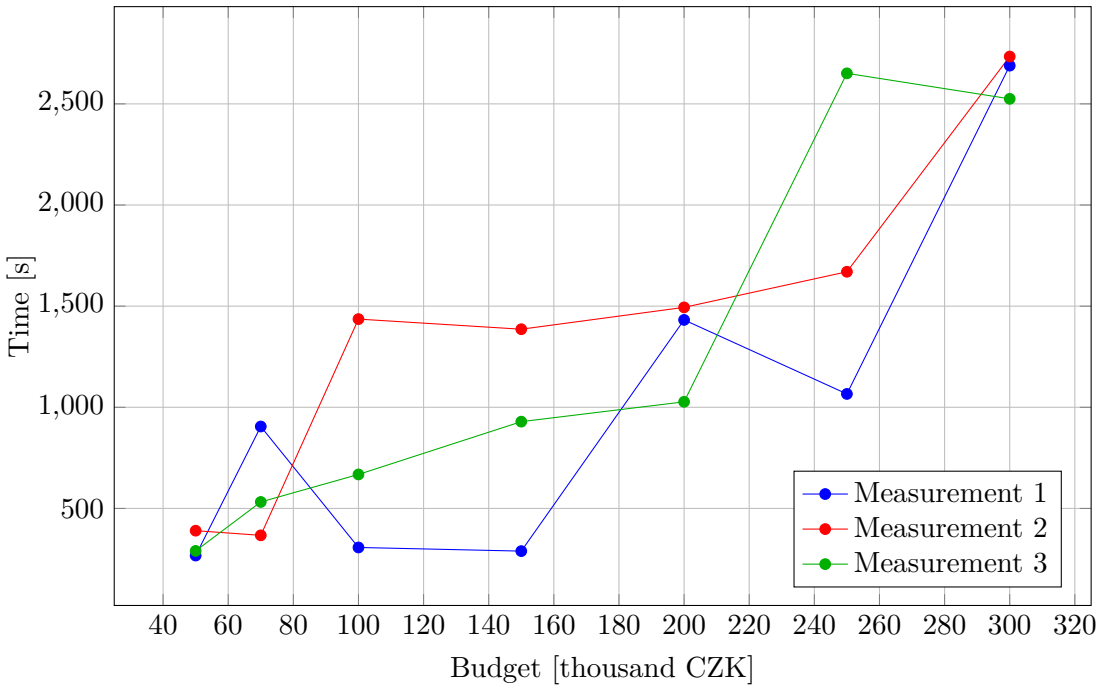


Figure 26 Algorithm performance for various budget

Budget [CZK]	Measurement 1 [s]	Measurement 2 [s]	Measurement 3 [s]	Average [s]
50 000	267	390	290	316
70 000	905	367	532	601
100 000	307	1 436	668	804
150 000	289	1 386	929	868
200 000	1 432	1 494	1 027	1 318
250 000	1 066	1 670	2 651	1 796
300 000	2 689	2 734	2 525	2 649

Table 12 Algorithm performance for various budget

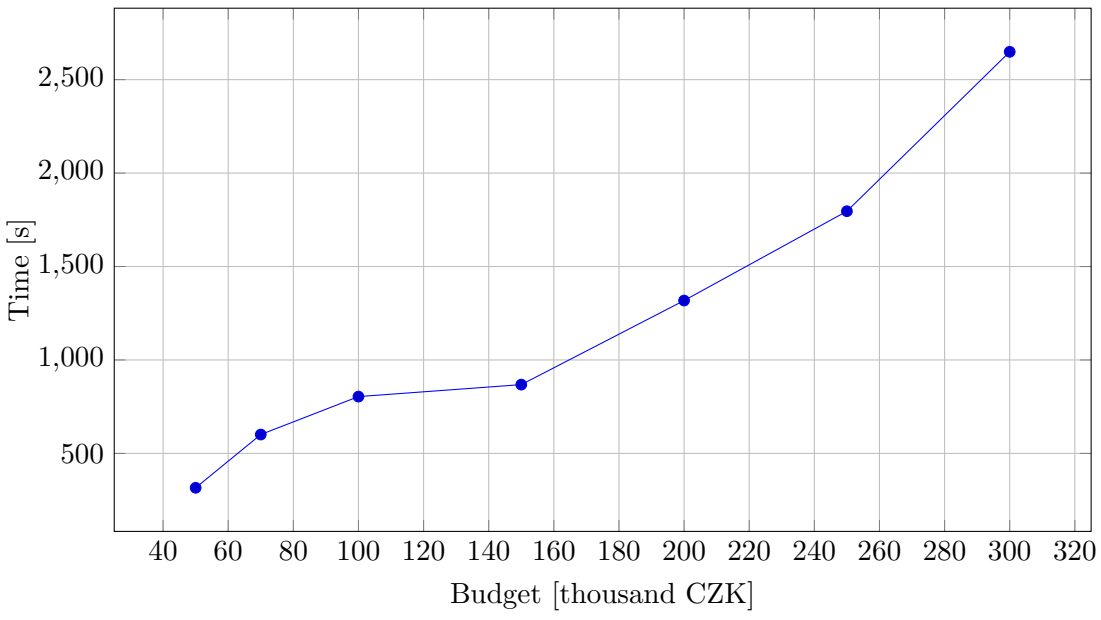


Figure 27 Average of testing algorithm performance for various budget

In the Figure 26, we registred three measurements and we get the unexpected results. We would expect that with increasing budget will be also growing the time required to find solution. The time varies solution to solution in the one measurement but also the time varies solution to solution between each measurements.

This differences are caused by different number of stations in calculation and different number of months for inspection. The algorithm repeats the calculation until it exhausts the budget. If the algorithm chooses ten petrol station in first round, the algorithm can exhaust the budget in one round and time for finding solution will be very short. If the algorithm chooses two petrol stations in first round, then it also chooses two petrol stations and so on. Time for finding solution will be very long (for the same budget like previous example with ten choosen petrol stations).

8.2.3 Final route length measurement

We calulate final length of route for the year. In the previous test in Section 8.1.3 we calculated final lenght for the choosen month. We study dependency of length of final route on the budget. We provide the same test three times like in prevous tests. Each test is on the other group of the petrol stations from central bohemian region.

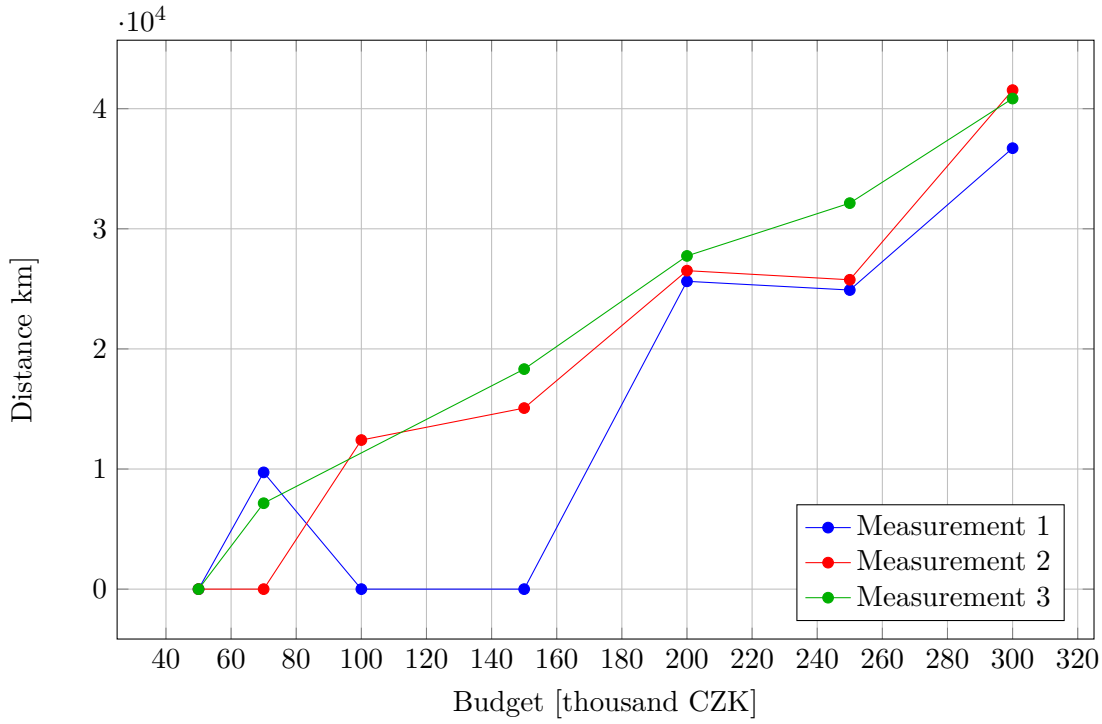


Figure 28 Dependency of the final distance route on the budget

The results are in the Figure 28. We get almost the expected results. With the increasing budget is also growing the length of final route. But in all measurements for budget $B = 70000$ result is null. It will happend if the algorithm chooses more petrol stations or more months and the price of the length of route exceeds the budget B. Concurrently the time for finding correct solution will be very small as we can see in the Figure 26.

Budget [CZK]	Measurement 1 [km]	Measurement 2 [km]	Measurement 3 [km]	Average [km]
50 000	0	0	0	0
70 000	9 721	0	7 135	5 625
100 000	0	12 412	10 729	7 714
150 000	0	15 074	18 319	11 131
200 000	25 629	26 516	27 742	26 629
250 000	24 903	25 756	32 139	27 599
300 000	36 714	41 548	40 845	39 702

Table 13 Dependency of the final distance route on the budget

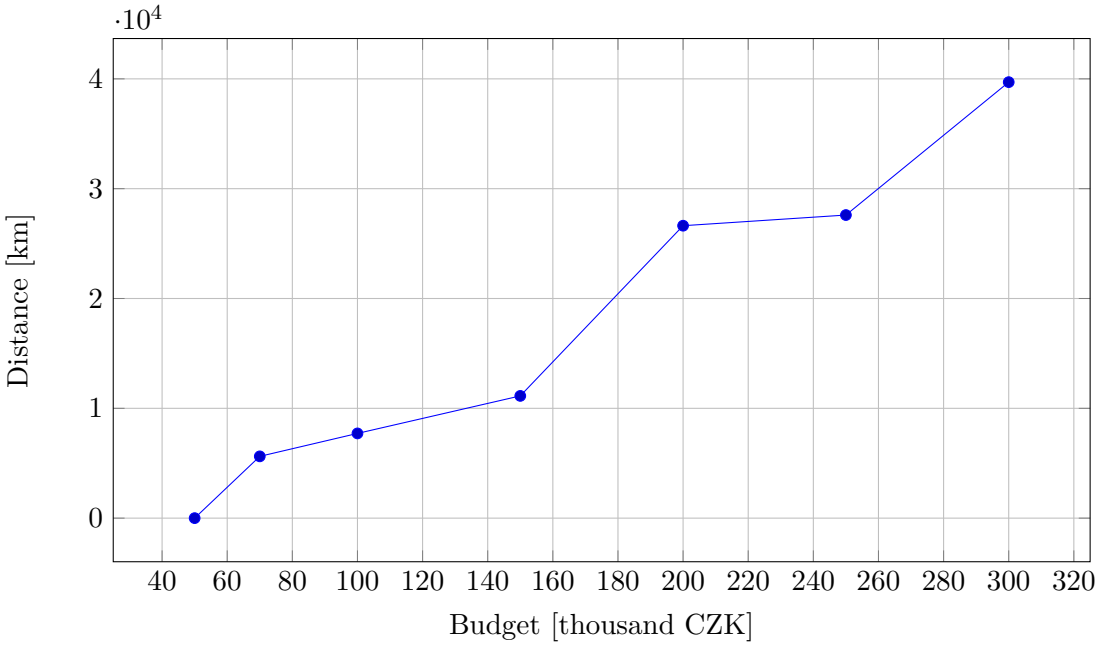


Figure 29 Average of testing dependency of the final distance route on the budget

9 Conclusion

We have addressed the inspection scheduling problem for one inspector. We have addressed all points of the assignment. We studied the problem of petrol station inspections and we also studied inspection games, security games and traveling salesman problem. This problems and algorithms are described in the Chapters 2, 3 and 4. We have formalized the problem of petrol station inspections as a game using suitable game theoretical approach, which we model as a two-player game, where one player is the inspecting entity and the other is the petrol stations.

We have constructed two models of the inspection problem. We have designed and implemented algorithm for computing the solution for both models. Then we evaluated the algorithms on the open data provided by Czech trade inspection.

The first model is with fixed petrol stations and number of months. The first model is simplier than the second model. The first model considers fixed input parameters fixed petrol stations and number of months. We can define number of petrol station and we can study dependency of the algorithm performance on the number of petrol stations and dependency of the distance route on the number of petrol stations. In the experiments, the performance of the algorithm depended on the number of petrol stations appeared as we expected. With the increasing number of stations is also growing the time required to calculate task. For example result of algorithm performance for two petrol stations $B_S = 2$ is 105 s (in average) and result of algorithm performance for ten petrol stations $B_S = 10$ is 461 s (in average).

The second model extends the first model. User specifies to the program budget which we can spend for the inspections. The algorithm from the second model calculates number of petrol stations and number of months automatically. This model is more realistic than the first model, because in real life the inspection authority defines the budget for spending money. The computation time for the algorithm from the second model is longer than for the algorithm from the first model, because the second program calculates the TSP for each month. The first program calculates the TSP only for the month choosen by user. Moreover the program which uses the second program repeats all calculations until it exhausts the budget.

We have addressed inspector problem for one inspector by game theoretical model and by traveling salesman problem. We solved the problems by linear program. This solution is very slow. For example program finds correct solution in 390 s. This time is for the first program with fixed petrol stations $B_S = 9$ and number of months $B_M = 7$. Time of the finding correct solution in the second program is even far longer. The second program finds solution for the budget $B = 200000$ CZK for 1432 s.

Bibliography

[1] Min Zhang Byung-In Kim Jae-Ik Shim. "Comparison of TSP Algorithms". In: *Comparison of TSP Algorithms* (1998).

[2] Otunbanowo Kehinde Adewole Philip Akinwale Adio Taofiki. "A Genetic Algorithm for Solving, Travelling Salesman Problem". In: *A Genetic Algorithm for Solving, Travelling Salesman Problem* (2011).

[3] Nikhil Kumar Sardar Prof. Sharadindu Roy Uttam Kumar Panja. "Efficient technique to solve travelling salesman problem using genetic algorithm". In: *Efficient technique to solve travelling salesman problem using genetic algorithm* (2014).

[4] John Tsitsiklis Dimitris Bertsimas. "Simulated annealing". In: *Simulated annealing* (1998).

[5] Jean-Yves Potvin. "The Traveling Salesman Problem: A Neural Network Perspective". In: *The Traveling Salesman Problem: A Neural Network Perspective* ().

[6] Bernard von Stengel Rudolf Avenhaus and Shmuel Zamir. "Inspection games". In: *Inspection games* (2001).

[7] Jason Tsai James Pita Christopher Kiekintveld Manish Jain, Fernando Ordóñez, and Milind Tambe. "Computing Optimal Randomized Resource Allocations for Massive Security Games". In: *Computing Optimal Randomized Resource Allocations for Massive Security Games* (2009).

[8] Martin J. Osborne. *A Course in Game Theory*. 1994.

[9] G. Leitmann. *Optimization Theory and Applications*. 1978.

[10] M. Breton. *Optimization Theory and Applications*. 1988.

[11] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. 1995.

[12] Ailsa H Land and Alison G Doig. *An automatic method of solving discrete programming problems*. 1960.

[13] Jens Clausen. "Branch and Bound Algorithms - Principles and Examples." In: *Branch and Bound Algorithms - Principles and Examples.* (1999).

[14] Jakub Ondráček. "Intelligent Algorithms for Monitoring of the Environment Around Oil Pipe Systems Using Unmanned Aerial Systems". Bachelor's thesis. Czech Technical University in Prague, 2014.

[15] Tomáš Werner. *Optimalizace*. 2014.

[16] *Travelling salesman problem*. URL: `http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Travelling_salesman_problem.html` (visited on 12/18/2014).

[17] Pavel Mička. *Problém obchodního cestujícího*. URL: `http://www.algoritmy.net/article/5407/Obchodni-cestujici`.

[18] Roger B. Myerson. *Game Theory: Analysis of Conflict*. 1991.

[19] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. 1944.

[20] Yoav Shoham Kevin Leyton-Brown. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. 2008.

[21] Jean Tirole Drew Fudenberg. *Game theory*. 1991.

[22] Jens Clausen. "Cooperative and Non-Cooperative Game Theory". In: *Cooperative and Non-Cooperative Game Theory* (2014).

[23] Robert J. Vanderbei. *Linear Programming: Chapter 11, Game Theory*. URL: `http://www.princeton.edu/~rvdb/542/lectures/lec8.pdfi` (visited on 10/17/2007).

[24] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. 2001.

[25] IBM. *IBM ILOG CPLEX Optimization Studio*. URL: `http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/`.

[26] *Analýza paliv a mziv pro reklamace posouzení vlivu na závadu motoru*. URL: `http://www.dobrapumpa.cz/files/dobrapumpa/certifikaty/SGS_letA4_reklamace_analyza-paliv-a-maziv.pdf`.

[27] Ministry of Industry and Trade. *Předpis č. 133/2010 Sb.* URL: `http://www.zakonyprolidi.cz/cs/2010-133#p3`.

[28] The Czech Trade Inspection Authority. *Působnost úřadu*. URL: `http://www.coi.cz/`.

[29] SGS. *Působnost úřadu*. URL: `http://www.dobrapumpa.cz/`.

[30] U.S. Energy Information Administration. "Monthly Energy Review January 2015". In: *Monthly Energy Review January 2015* (2015).

[31] *GraphHopper*. URL: `http://graphhopper.com/`.

[32] *JavaFX*. URL: `http://docs.oracle.com/javase/8/javase-clienttechnologies.htm`.

[33] *Leaflet*. URL: `http://leafletjs.com/`.