VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

VYUŽITÍ UMĚLÉ INTELIGENCE V PODNIKATELSTVÍ

THE USE OF ARTIFICIAL INTELLIGENCE IN BUSINESS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Ing. GABRIEL MATUS
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. PETR DOSTÁL, CSc.
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Matus Gabriel, Ing.

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Využití umělé inteligence v podnikatelství

v anglickém jazyce:

The Use of Artificial Intelligence in Business

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DOSTÁL, P. Pokročilé metody rozhodování v podnikatelství a veřejné správě. Brno:CERM, 2012. 718 s. ISBN 978-80-7204-798-7.
DOSTÁL, P. Advanced Decision Making in Business and Public Services. Brno:CERM, 2011. 168 s. ISBN 978-80-7204-747-5.
HANSELMAN, D. a B. LITTLEFIELD. Mastering MATLAB. Pearson Education International Ltd., 2012. 852 s. ISBN 978-0-13-185714-2.
MAŘÍK, V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence. Praha:ACADEMIA, 2013. 2473 s. 978-80-200-2276-9.

Vedoucí diplomové práce: prof. Ing. Petr Dostál, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc. Ředitel ústavu	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Děkan fakulty

V Brně, dne 30.11.2015

Abstrakt

Práca sa zaoberá s optimalizačným problémom cestujúceho obchodníka (TSP) a skúma jeho možnosti vo využití v podnikaní. Ide o optimalizačnú úlohu nákladov na cestovanie, usporenie času a zbytočne prejazdených kilometrov. K práce patrí jeden program s GUI písaný v programe MATLAB. Program pomocou neurónových sietí vypočíta najefektívnejšiu trasu medzi miestami, s ktorými sa obchodník musí stretnúť počas služobnej cesty. Algoritmus je možné využiť v rôznych prípadoch, kde sa zameriava na optimalizáciu trasy, napr. vývoz tovaru, správa skladu, plošné spoje či záchranárska služba. Program komunikuje so serverom Google Maps API, ktorý poskytuje aktuálne informácie o trase..

Kľúčové slová

Optimalizácia, TSP, problém obchodného cestujúceho, umelé neurónové siete, plánovanie trasy, MATLAB, logistika, Google Maps APIs

Abstract

This work deals with traveling salesman problem (TSP) and examines it's possibilities to use in business. It is about the optimization of the travel cost, saving time and unnecessary mileage. Part of the work is a program with a GUI written in program MATLAB. Program uses neural networks to calculate the most effective path between places, where the trader has to reach. It's possible to use the algorithm for many purposes, e.g. distribution of goods, store management, planning of PCBs or rescue services. Program communicates with the Google Maps API server, which provides the actual information of the path.

Keywords

Optimization, TSP, traveling salesman problem, artificial neuron network, planning of the path, MATLAB, logistics, Google Maps APIs

Bibliografická citace:

MATUS, G. *Využití umělé inteligence v podnikatelství*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 71 s. Vedoucí diplomové práce prof. Ing. Petr Dostál, CSc..

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Využití umělé inteligence v podnikatelství jsem vypracoval samostatně pod vedením vedoucího diplomového projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne: 21.1.2016 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce prof. Ing. Petru Dostálovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady za bezchybnou komunikaci při zpracování mé diplomové práce.

Děkuji i panu Lukáše Sáblíkovi za odbornou pomoc a připomínky z praxe při řešení diplomové práce.

Chtěl bych se poděkovat i mým kolegům Ing. Anikó Molnárovej a Ing. et Ing. Františkovi Hortaiovi za všestrannou pomoc a podporu během celého studia.

V Brně dne: 21.1.2016 .....

podpis autora

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Diplomová práca sa zameriava na využitie umelej inteligencie vo sfére podnikania. Umelú inteligenciu, ako nástroj na optimalizáciu jednotlivých procesov v podnikaní už použili a to viacerými spôsobmi. Optimalizovať sa dajú skoro všetky procesy, ktoré sme schopní charakterizovať a tieto charakteristiky aj číselne vyjadriť.

Optimalizovať znamená docieliť taký riadiaci proces, ktorý by podľa potreby maximalizoval či minimalizoval pri svojej práci výsledok daného procesu. Aby tento systém mohol čo najlepšie fungovať, potrebuje dokonale stanovené vstupné parametre a výstupné podmienky.

Umelá inteligencia, ako celok, je veľmi široký pojem technických prostriedkov, ktorými sa snažíme nahradiť ľudskú prácu pri vykonávaní systematických postupov. Táto práca sa zaoberá známou problematikou optimalizačného problému obchodného cestovateľa (TSP). Existujú rôzne systémy na vyriešenie tejto problematiky, rozdiely medzi nimi môžu byť presnosť, komplexnosť, výpočtová náročnosť ako aj použiteľnosť. Jedným najefektívnejším a najstabilnejším spôsobom sú genetické algoritmy. Pre porovnanie genetického algoritmu s iným algoritmom v práci sú použité aj umelé neurónové siete, ktoré sa dajú takisto použiť na optimalizáciu TSP.

1 CIELE PRÁCE, METÓDY A POSTUPY SPRACOVANIA

Práca sa zaoberá predovšetkým využitím umelej inteligencie v podnikateľskej sfére. Každý podnik sa snaží posilniť svoju pozíciu na trhu. K tomu používa rôznorodé metódy a nápady, ktorými môže konkurenciu potlačiť a prispieť k rastu svojho podniku.

V nasledujúcich kapitolách budú spomenuté základné teoretické východiská týkajúce sa optimalizácie. Pri optimalizácii sa snažíme vylepšiť už používané procesy, ktoré sú základnými podmienkami existencie podniku. Práca je zamieraná na optimalizáciu trasy pri logistických záležitostiach. Ide o tzv. problém obchodného cestujúceho (TSP). Je to známy problém, ktorý už mnohých inšpiroval. Základy tejto metódy pochádzajú z matematického pojmu teórie grafov.

Na aplikovanie metódy TSP vyvinuli viaceré postupy, ktoré môžu byť celkom odlišné a patria do množiny umelej inteligencie UI. Najznámejšie sú genetické evolučné algoritmy, ktoré sú používané v širokom spektre. Vďaka týmto schopnostiam je možné elegantne riešiť zložité problémy. Na druhej strane sú neurónové siete, ktoré napodobňujú funkcionalitu mozgu človeka. Skladajú sa z jednoduchých neurónov, ktoré spoločne tvoria rozsiahle siete a sú schopné riešiť problémy rôzneho druhu. Obidva prístupy sú zakomponované do práce a testované na danej problematike.

V sekcii analýza súčasného stavu je analyzované okolie podniku, v ktorom je práca implementovaná. Profil podniku je predaj potravín a ďalších produktov. Na českom trhu už spôsobí úspešne veľa rokov . Ako prvá zaviedla službu online obchodu, ktorá dostáva stále viac a viac pozornosti. Ide o novú službu a z tohto dôvodu je postihnutá začiatočnými ťažkosťami.

Vlastným návrhom riešenia je prostredie písané v jazyku MATLAB. Ide o grafický program, ktorý poskytuje riešenie na optimalizáciu trasy áut online obchodu. Je založený na podpore webových služieb od spoločnosti Google Inc. Služby sú poskytnuté pomocou webových serverov, ktoré komunikujú webovými adresami.

Postup pri spracovaní práce

Spracovanie bolo prevedené nasledujúcimi krokmi:

1. Získanie základných teoretických poznatkov o problematike TSP
2. Výber algoritmov riešiacich optimalizačné problémy
3. Charakterizácia obsahu práce
4. Návrh programového riešenia
5. Získanie spoluprácu z praxe
6. Získanie vstupných parametrov
7. Formulácia práce podľa potreby subjektu
8. Návrh finálnej verzie programu
9. Stanovenie podmienok využitia práce
10. Hodnotenie výsledkov práce

2 TEORETICKÉ VÝCHODISKÁ PRÁCE

2.1 Optimalizácia

Optimalizácia je matematická disciplína, v ktorom hľadáme minimum (maximum) danej funkcie $f(x)$ na danej množine M. Táto funkcia sa nazýva **účelová** či **cieľová**.

Optimalizovať sa dá čoskoro všetko vo svete, čo sme schopní pochopiť, spoznať, charakterizovať, číselne vyjadriť či vyhodnotiť. V každom prípade hľadáme taký **extrém** v priebehu jedného deja, ktorý je jedinečný v danom intervale. Z hľadiska intervalu pozorovania hovoríme o **lokálnom** a **globálom** extréme.

Cieľom optimalizácie z ekonomického hľadiska môže byť minimalizácia nákladov, maximalizácia výnosov v podnikaní. Aby sme k tomu docielili, musíme si zvoliť postup, presne definovanými krokmi, počas toho kontrolovať prejdenú cestu a neustále lepšie predvídať nasledujúce kroky. Je potrebné jasne definovať, pri akých podmienkach hľadáme optimum, lebo optimum môže byť ľahko subjektívny charakter.

2.2 Problém obchodného cestujúceho (TSP)

Problém obchodného cestujúceho je jeden z najznámejších optimalizačných problémov, ktorý má za ciel nájsť tú najkratšiu cestu, pri ktorej sa cestujúci dotkne každého mesta raz a iba raz a nakoniec prichádza naspäť do mesta, odkiaľ odštartoval.

Prvé spomienky týkajúce sa tejto tematiky pochádzajú sú ešte z 18. ho storočia. Najskôr od švajčiarskeho matematika a fyzika Leonharda Paula Eulera, ktorý si chcel posúvať rytiera na šachovnici tak, aby na každej pozícii stál iba raz.

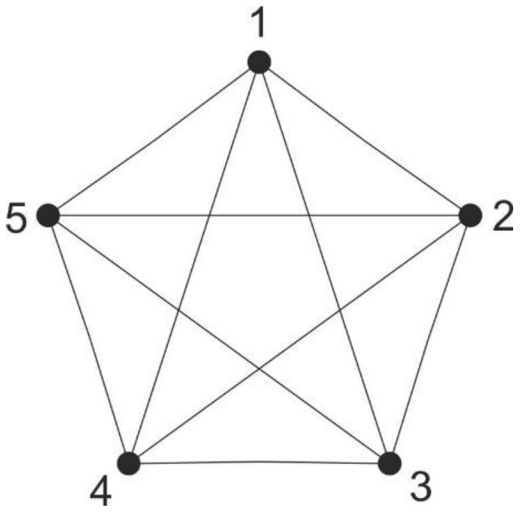
Z matematického hľadiska TSP sa dá opísať pomocou teórie grafu. Problematika sa dá predstaviť ako **hamiltonovskú kružnica**, z ktorej vyplývajú nasledujúce základné pojmy.

2.2.1 Teória grafov

Teória grafov je časť diskrétnej matematiky, ktorá skúma vlastnosti grafov. (KOPŘIVA, 2009)

2.2.1.1 Definícia grafu

Nech máme graf $G = (V, E)$, kde V je množina všetkých m miest, $V = \{v_1, \dots, v_m\}$ a E je množina dvojíc prvkov z V, $E = \{(r, s): r, s \in V, r \neq s\}$. Táto množina E je reprezentovaná vzdialenosťami medzi miestami (vrcholy grafu). Dá sa to definovať množinou D. V tomto prípade vzdialenosť medzi miestami r a s zapíšeme ako $D = (d_{r,s})$.



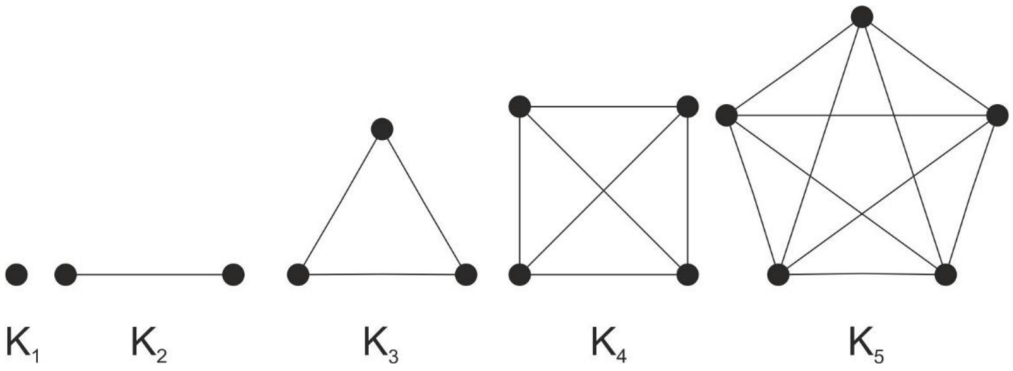
Obrázok 2.1: Teória grafu

2.2.1.2 Susedné vrcholy

Dva vrcholy r a s sú susedné v grafu G, keď $\{r, s\}$ je hrana grafu G. Keď hrana $e = \{r, s\}$, o hrane sa hovorí, že je **incidentná** vrcholmi r a s. Stupeň vrcholu je rovný počtu hrán s ním incidentných. Stupeň všetkých vrcholov grafu na (Obrázku 2.1) je $deg(1 \dots 5) = 4$.

2.2.1.3 Úplný graf

Úplný graf o n vrcholoch, označovaný ako K_n, je taký graf, ktorý obsahuje práve jednu hranu medzi každou dvojicou rôznych vrcholov. Obsahuje $\binom{n}{2}$ hrán. To platí pre grafy s $n \geq 1$ počtom hrán.



Obrázok 2.2: Úplné grafy pre 1 – 5 vrcholov

2.2.1.4 Podgraf

Podgraf je časť grafu, ktorá vznikne z pôvodného grafu vymazaním niektorých jeho vrcholov, všetkých hrán vedúcich do týchto vrcholov, poprípade vymazaním ďalších jeho hrán.

2.2.1.5 Hamiltonovská cesta a kružnica

Hamiltonovská kružnica je taký podgraf, ktorý má tvar kružnice a obsahuje všetky vrcholy pôvodného grafu. Sú to uzavrené hamiltonove cesty. Pri TSP sú podobné podmienky, ale okrem týchto si musíme zaistiť aj minimálnu dĺžku celej cesty. Kým z hamiltonovských ciest a kružníc môže existovať viac v jednom grafe, vyhovujúcou trasou je iba jedna z nich.

2.2.2 Základné typy TSP

Rozlišujeme tri základné typy TSP, ktoré sú:

- symetrický (sTSP)
- asymetrický (aTSP)
- problém viacerých obchodných cestujúcich (mTSP)

2.2.2.1 Symetrický problém obchodného cestujúceho (sTSP)

Nech máme graf $G = (V, E)$, kde V je množina všetkých m miest, $V = \{v_1, \dots, v_m\}$ a E je množina dvojíc prvkov z V, $E = \{(r, s): r, s \in V, r \neq s\}$ a

vzdialenosť medzi miestami r a s zapíšeme ako $D = (d_{r,s})$. V prípade keď medzi týmito dvomi miestami platí rovnosť $d_{r,s} = d_{s,r}$ pracujeme so symetrickou TSP. Výpočet takého TSP je mnoho jednoduchší, netreba uvažovať nad tým, ktorý smer spoju musíme použiť pre korektný výsledok.

2.2.2.2 Asymetrický problém obchodného cestujúceho (aTSP)

Nech máme graf $G = (V, E)$, kde V je množina všetkých m miest, $V = \{v_1, \dots, v_m\}$ a E je množina dvojíc prvkov z V, $E = \{(r, s): r, s \in V, r \neq s\}$a vzdialenosť medzi miestami r a s zapíšeme ako $D = (d_{r,s})$. V prípade keď medzi týmito dvoma miestami platí nerovnosť $d_{r,s} \neq d_{s,r}$ pracujeme s asymetrickou TSP. Výpočet sa nám komplikuje, musíme počítať napr. s jednosmernými cestami, zátarasami, uzávierkami ciest.

2.2.2.3 Problém viacerých obchodných cestujúcich (mTSP)

Zmienený graf si musíme rozdeliť medzi viacerými cestujúcimi. Počet cestujúcich môže byť fixne daný, alebo môžeme s tým počítať ako premennou hodnotou. Počet začiatočných uzlov sa takisto môže zmeniť.

2.2.3 Časová náročnosť TSP

Táto problematika patrí medzi optimalizačné problémy, ktorú je ľahké definovať, ale riešenie vyžaduje zložitý výpočet. Ohľadom na zložitosť pre vysoký počet navštívených miest (počet vrcholov) náročnosť na výpočtový výkon, doba výpočtu môže tak narásť, že výpočet nebude možné zrealizovať. Podľa toho problém môžeme rozdeliť na **algoritmizovateľný** a **nealgoritmizovateľný**.

V prípade TSP ide o algoritmizovateľný proces, ktorý je aj časovo aj priestorovo náročný. Patrí do skupiny **NP-úplné** (nedeterministicky polynomiálny problém), ktoré sú tie najťažšie úlohy z nadmnožiny NP.

2.3 Umelá inteligencia

Inteligencia je taká vlastnosť živých organizmov, ktorá im pomôže, aby dostali v prírode mimoriadne postavenie. Je súbor rozumových schopností, schopnosť riešiť problémy v nových situáciách. So zvýšením výpočtovej rýchlosti sa snaha o napodobňovanie týchto schopností posilnila. Začali navrhovať a experimentálne overovať nové metódy, postupy a algoritmy s inteligentným chovaním. Takto sa vyvinula nová vedná disciplína – **umelá inteligencia (artificial intelligence)**.

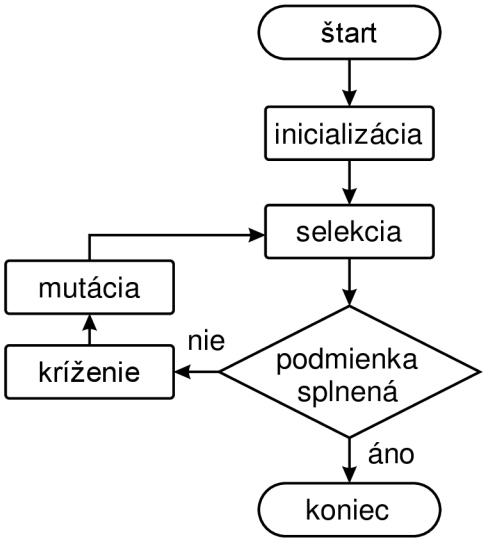
Dodnes nie je presne charakterizovaný výraz inteligencia a tak ani inteligencia umelá. Podľa **Turingovho testu** keď stroj bude schopný reagovať na podnety ľudského partnera takým spôsobom, že človek nebude schopný rozpoznať, či sa jedná o stroj, alebo o osobu prostredníctvom terminálu, potom sa dá stroj považovať za inteligentného. (KOPŘIVA, 2009)

2.3.1 Genetické algoritmy

Genetické algoritmy sú podmnožinou evolučných algoritmov. Sú to heuristické algoritmy a slúžia k riešeniu zložitých problémov, s ktorými si už exaktné algoritmy neporadia. Používajú techniky napodobňujúce evolučné procesy, ktoré poznáme z biológie a to sú:

- selekcia
- kríženie
- mutácia

Jedným najdôležitejším krokom je správne vytvorená začiatočná populácia. Ide o *n* počet jedincov a o správne zakódovanie informácie o riešenom problému. Najčastejšie prebieha inicializácia náhodne. Algoritmus je zobrazený pomocou vývojového diagramu:



Obrázok 2.3: Vývojový diagram reprodukcie

Reprodukcia je proces postupného vylepšovania populácie jedincov opakovanou aplikáciou genetických operátorov, ktorý vedie k evolúcii takých jedincov, ktorí lepšie vyhovujú stanoveným podmienkam než jedinci, z ktorých vznikli. Proces konverguje k situácii, keď je populácia tvorená len tými najlepšími jedincami. Hlavným princípom je kopírovanie a vymieňanie reťazcov - chromozómov. Chromozóm má pevnú dĺžku, jednotlivé pozície tvoria gény. Gény reprezentuje často binárna 0 alebo 1, ale všeobecne môžu mať ľubovoľnú hodnotu. Množinu obsahujúcu všetky možné chromozómy môžeme popísať rovnicou (DOSTÁL, 2008):

$$X = \{x_1, x_2, \dots, x_n\} \tag{2.1}$$

Množina chromozómov tvorí populáciu. Každý chromozóm v populácii má definovanú hodnotiacu funkciu, nazývanú *fitness funkcia*, ktorá charakterizuje vhodnosť chromozómu. Priraďuje každému $x \in X$ reálne číslo $f(x)$, tzv. $f : X \to R$. Optimalizačný problém má potom štruktúru:

$$x_{opt} = \arg\min_{x \in X} f(x) \tag{2.2}$$

Cieľom je dopracovať sa k takému chromozómu, ktorý disponuje hodnotou fitness funkcie $f(x)$ čo najbližšou k hodnote či rovnou $f(x_{opt})$. Najznámejšie genetické algoritmy, ktoré sú použiteľné na riešenie optimalizačného problému TSP sú:

- Neinformované metody
- Informované metody
- Exhausive search
- Backtracking
- Random search
- Greedy algorithm
- Hill climbing
- Simulated annealing
- Tabu search
- Ant colony
- Genetic search
- Participle swarms

Pri návrhu aplikácie bol použitý algoritmus ***simulované žíhanie*** (simulated annealing). V nasledujúcej podkapitole je opísané jeho fungovanie.

2.3.1.1 Simulované žíhanie (Simulated annealing)

Existuje mnoho optimalizačných algoritmov, ako už o tom bolo zmienené v predošlej kapitole. Táto metóda má prednosť v tom, že sa vyhýba pristihnutiu lokálnym maximám – riešenia, ktoré sú lepšie ako ostatné v okolí, ale nie sú tie najlepšie. Podľa experta algoritmov, *Stevena Skiena* – **„Riešenie pomocou simulovaného žíhania funguje nádherne“**. Problém obchodného cestujúceho je dobrým príkladom optimalizačného problému (KENDALL a kol.,1999; JURČÍK, 2014).

Je pravdepodobnostná (stochastická) optimalizačná metóda prehľadávania stavového priestoru a je založená na simulácii žíhania ocele. Základy má oproti ostatným podobným algoritmom z fyziky, namiesto biológie. Pri prehľadávaní stavového priestoru sa môže ľahko stať, že algoritmus uviazne v lokálnom minime. V metóde sa tomu snažíme zabrániť tým, že vykonávame aj zmeny k horšiemu, veľké hlavne spočiatku, a

vďaka tomu sa môžeme dostať z lokálneho minima. Veľkosť zmeny záleží na teplote. Čím väčšia je teplota, tým väčšie sú vykonané zmeny. Algoritmus pracuje iba s jedným kandidátnym riešením. Obyčajný gradientný algoritmus prijíma nové riešenie len ak je lepšie ako riešenie existujúce. Pri simulovanom žíhaní tak sú s určitou pravdepodobnosťou prijaté aj horšie riešenia. Pravdepodobnosť prijatia aj horšieho riešenia je priamo závislé na teplote a nepriamo na veľkosti zhoršenia. V priebehu výpočtu algoritmu je teplota postupne znižovaná na základe rýchlosti konvergencie. Ak algoritmus konverguje rýchlo (hodnotenie stavov rýchlo klesá), teplota sa znižuje tiež rýchlo a algoritmu je tak bránené pokračovať do horšieho hodnoteného stavu. Ak konverguje algoritmus pomaly (hodnotenie stavov moc neklesá), spomalí sa znižovanie teploty, a tak by sa prípadne podarilo vyslobodiť z lokálneho minima.

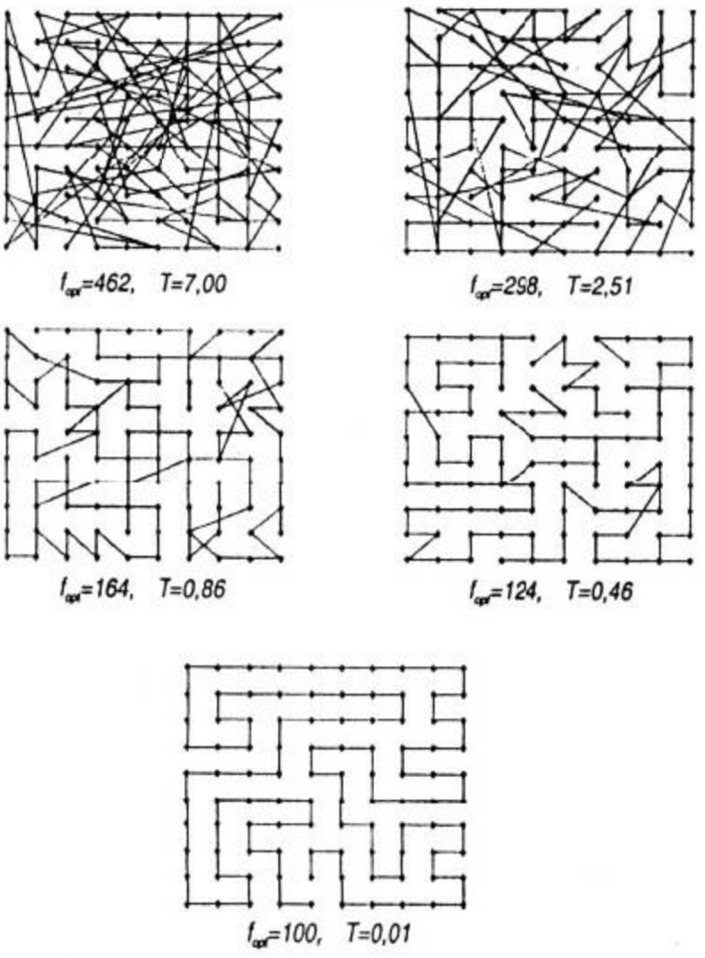
Ak nie sme spokojní s výsledkom, tak teplotu možno zvyšovať. Potom pravdepodobnosť, že bude algoritmus smerovať k horšiemu hodnotovému stavu tak stúpa, že sa algoritmu nepodarí (neustálym pokračovaním do čo najlepšie hodnotených stavov) približovať sa k cieľu.

Za podmienky, že proces ochladzovania je dostatočne pomalý, potom pre každú teplotu T je žíhané teleso v rovnovážnom stave. Tento stav je popísaný Boltzmannovým rozdelením pravdepodobnosti toho, že pri teplote T je teleso v stave i s energiou E_i (ELLISON GELTMAN, 2014)

$$W_T(E_i) = \frac{1}{Q(T)} e^{-\frac{E_i}{kT}} \tag{2.3}$$

Kde T je Boltzmannova konstanta a $Q(T)$ je partičná funkcia

$$Q(T) = \sum_i e^{-\frac{E_i}{kT}} \tag{2.4}$$



Obrázok 2.4: Simulované žíhanie – postupné znižovanie teploty (zdroj: KENDALL a kol, 1999)

2.3.2 Umelé neurónové siete

Umelé neurónové siete vychádzajú z biologických neurónových sietí mozgu človeka. Sieť sa skladá z mnoho samostatných neurónov ktoré sú husto prepojené pomocou synapsí. V mozgu z týchto synapsií a neurónov je obrovské množstvo, ktoréb by sa simulovať umelými sieťami nedalo. Umelé siete sú preto jednoduchším matematickým modelom biologických sietí. Základné termíny týchto sietí (JIRSÍK, 2012):

- **adaptácia** – schopnosť samooragnizácie siete, realizuje sa hlavne zmenami váh počas učenia
- **adaptivný systém** – systém so schopnosťou zlepšiť alebo upraviť svoju výkonnosť pri zvyšujúciom sa faktoru náročnosti, alebo vyhovieť neurčitostiam spôsobenými prostredím
- **architektúra** – štruktúra siete výkonných prvkov, ich vzájomné prepojenie
- **algoritmus** – metóda či procedúra pre získanie cieľa, alebo riešenie
- **excitace** – také pôsobenie neurónu, pri ktorom v pripojených neurónoch dochádza k rastu ich vnútorného potenciálu
- **energetická funkcia** – energia je mierou naučenosti, teda odchýlky medzi skutočnými a požadovanými hodnotami výstupu neurónovej siete pre danú trénovaciu množinu
- **klasifikácia** – schopnosť priradiť vstupný pozorovaný vzor alebo znak do určitej kategórie
- **znak alebo vlastnosť** – niečo čo charakterizuje vlastnosť nejakého objektu alebo situácie
- **heuristika** – metóda používaná podľa skúseností, pričom nemusí byť garantované riešenie problému
- **preučovanie** – učiaci sa proces, v ktorom sa maže istý počet váh. Oproti normálnemu učeniu, pri preučovaní siete určité množstvo informácií sa stratí a dostaneme sa k horšiemu výsledku
- **prevrátenie** – určenie vstupu z daného výstupu a systémového modelu
- **samoorganizácia** – schopnosť neurónovej siete učením prispôsobiť svoje chovanie k vyriešeniu daného problému

- **sieť** – spojenie vzájomne spojených entít
- **topológia** – popisuje druh a počet výkonných prvkov siete a štruktúru ich prepojenia
- **jednotka** – elementárny objekt umelej neurónovej siete

2.3.3 Model neurónu – perceptron

V umelej neurónovej sieti výkonným prvkom je sám neurón. Neurón sa dá matematicky formulovať, vstupné informácie spracováva a predá ďalej podľa vzťahu:

$$y = f\left(\sum_{i=1}^{N} w_i x_i + \Theta\right) \tag{2.5}$$

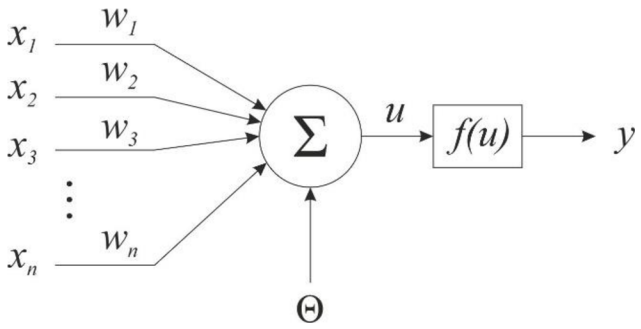
x_i – vstupy neurónu a počet týchto neurónov je N

w_i – synaptické váhy

F – prenosová funkcia dotyčného neurónu

Θ – prah daného neurónu

y – výstup neurónu



Obrázok 2.5: Model neurónu – perceptronu

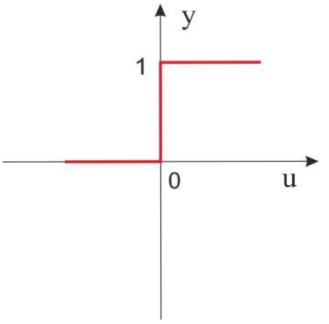
Perceptron pozostáva z jedného výkonného prvku modelovaného obvykle McCullochovým a Pittsovým modelom neurónu, ktorý má nastaviteľné váhové koeficienty a nastaviteľný prah. Algoritmus vhodný k nastaveniu parametrov perceptronu publikoval prvý F. Rosenblatt v roku 1958. Bol prvý funkčný perceptron, ktorý bol schopný riešiť iba lineárne separabilné úlohy.

2.3.4 Prenosová funkcia

Podľa súčtu váhových vstupov a prahu transformuje výstup z neurónu. Najznámejšie typy prenosových funkcií:

- **skoková prenosová funkcia**

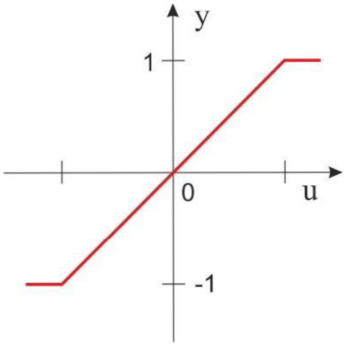
$$y = \begin{Bmatrix} 0\ pre\ u < 0 \\ 1\ pre\ u\ \geq 0 \end{Bmatrix} \tag{2.6}$$



Obrázok 2.6: Skoková prenosová funkcia

- **čiastočne lineárna prenosová funkcia**

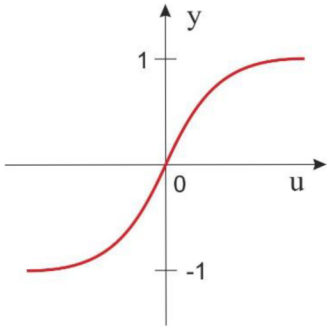
$$y = \begin{Bmatrix} 1\ pre\ u \geq \frac{1}{2} \\ u\ pre -\frac{1}{2} < u < \frac{1}{2} \\ 0\ pre\ u \leq -\frac{1}{2} \end{Bmatrix} \tag{2.7}$$



Obrázok 2.7: Čiastočne lineárna prenosová funkcia

- **prenosová funkcia hyperbolického tangenta**

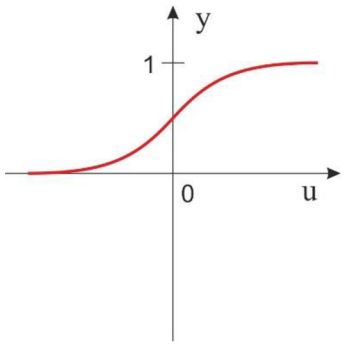
$$y = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \tag{2.8}$$



Obrázok 2.8: Prenosová funkcia hyperbolického tangenta

- **sigmoidná prenosová funkcia**

$$y = \frac{1}{1 + e^{-u}} \tag{2.9}$$



Obrázok 2.9: Sigmoidná prenosová funkcia

2.3.5 Topológie a druhy neurónových sietí

Neurónové siete si môžeme rozdeliť podľa viacerých kritérií:

Podľa počtu vrstiev:

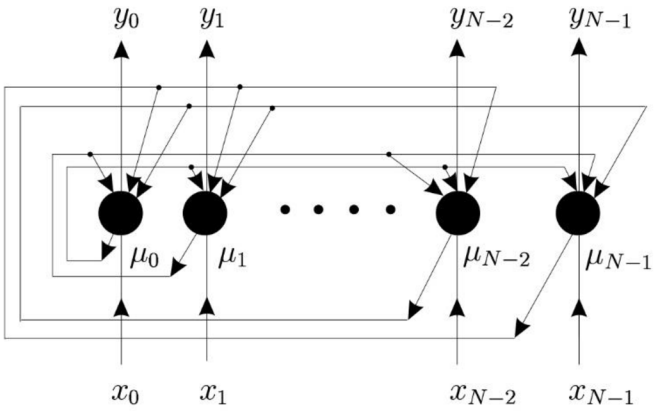
- jednovrstvová (Hopfieldová sieť, Kohonenová sieť, ...)
- viacvrstvová (ART sieť, Perceptron, klasická viacvrstvová sieť s algoritmom Back-propagation)

Podľa algoritmu učenia:

- s učiteľom (sieť s algoritmom Back-propagation, ...)
  - deterministicky (algoritmus Back-propagation)
  - stochasticky (náhodné nastavovanie váh)
- bez učiteľa (Hopfieldová sieť, Kohonenová sieť,...)

2.3.6 Hopfieldova sieťová štruktúra

Hopfieldov model je krokom smerom od biologickej realite. Používa symetrické spojenie medzi neurónmi, ktoré v prírode neexistujú. Patrí skôr do teórie lineárnych dynamických systémov.



Obrázok 2.10: Hopfieldova sieťová štruktúra (zdroj: JIRSÍK, 2012)

Každý neurón v sieti je vstupným aj výstupným neurónom (jedna vrstva) – spätnoväzbová (rekurentná) sieť. Výstup z neurónu je vstupom ostatných neurónov.

2.3.6.1 Učenie v Hopfieldovej sieti

Proces učenia je jednorazový, čiže prejde sieť množinu všetkých vstupných vzorov a je naučená.

Krok 1. Učenie

- Nastavenie váh podľa vstupných vzorov:

$$w_{ij} = \begin{cases} \sum_{S=0}^{M-1} x_i^s x_j^s, & pro\ i \neq j \\ 0, & pro\ i = j, 0 \leq i, j \leq N-1 \end{cases} \tag{2.10}$$

- V tejto rovnici w_{ij} je váha medzi *i*-tým a *j*-tým neurónom (ide o štvorcovú maticu). Výrazy x_i a x_j v rovnici nadobúdajú hodnoty ± 1.

Krok 2. Opakovanie alebo koniec učenia

- V prípade, že sme nevyčerpali všetky učície vzory, vrátime sa späť ku **Kroku 1**. Keď sme už použili všetky vzory ku testovaniu, ukončíme proces.

2.3.6.2 Vybavovanie v Hopfieldovej sieti

Krok 1. Inicializácia neurónovej siete

- Nastavenie začiatočných stavov podľa predložených vzorov:

$$\mu_i(0) = x_i, 0 \leq i \leq N-1 \tag{2.11}$$

- V tejto rovnici $\mu_i(t)$ je výstup z i-teho neurónu v čase t a x_i je element obrazca, ktorý môže nadobúdať hodnoty len **0** alebo **1** (unipolárne), poprípade **+1** alebo **- 1** (bipolárne).

Krok 2. Iterácia až do nájdenia odpovede

$$\mu_i(t+1) = f_h\left[\sum_{i=0}^{N-1} t_{ij}\mu_j(t)\right], 0 \leq j \leq N-1 \tag{2.12}$$

- Funkcia f_h je nelinearita bez posunutia. **Krok 2** prevedieme toľko krát, kým sa váhy prestanú meniť – rozdiel medzi týmto a predošlým krokom je nulový. Výstupy neurónov $y_i = \mu_i\left(t_{posledný}\right)$ sú priamo jednotlivé body obrazca.

Krok 3. Opakovanie procesu vybavovania

- Po ukončení iterácie môžeme zadať nový vzor a prejsť ku **Kroku 1**. V opačnom prípade ukončíme vybavovanie.

2.3.6.3 Optimalizácia pomocou siete Hopfieldova typu

Hopfieldová sieť je dynamickým systémom, pri ktorom sa dá merať jej energia. Má charakter postupného zníženia, ktorý sa dá využiť na optimalizačné problémy, kde hľadáme minimum (resp. maximum) určitej matematickej funkcie. Jednou z nich je práve problém obchodného cestujúceho.

Pri aplikovaní Hopfieldovej siete používa sa štvorcová matica neurónov a ich počet je n^2, kde n je počet navštívených miest. Podľa topológie siete používa $n^2 \cdot n^2$počet váh, ktorými sú spojené všetky neuróny s ostatnými neurónmi.

Výstupnou maticou je matica reprezentujúca výstupy z neurónov. Z toho vyplýva, že jej dimenzia bude tá istá, ako dimenzia neurónov vo vrstve.

Platná cesta – tento výraz pochádza ešte z teórie grafu. Cesta je platná, keď dodrží všetky podmienky Hamiltonovskej kružnici. V tomto prípade platnú cestu si môžeme definovať s tým, že po výpočte optimálnej trasy výstupná matica musí mať jeden a iba jeden nenulový neurón v každom riadku, jeden a iba jeden nenulový neurón v každom stĺpci napr.:

V =

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

Tabuľka 1: Platná cesta reprezentujúca výstupnú maticu Hopfieldovej siete (vlastný zdroj)

Úlohou siete je minimalizovať energetickú funkciu *E*, ktorá je navrhnutá tak, že minimálnu energiu získa iba v tom prípade, ak sú dodržané požiadavky na výstupnú maticu.

$$E = \underbrace{A \sum_{r} \sum_{ij} x_{ri} . x_{rj}}_{1} + \underbrace{B \sum_{r} \sum_{ij} x_{ir} . x_{jr}}_{2} + \underbrace{C((\sum_{ij} x_{ij}) - m)^2}_{3} + \underbrace{D \sum_{rs} (\sum_{i} d(r,s) x_{ri} (x_{s,i+1} + x_{s,i-1}))}_{4} \tag{2.13}$$

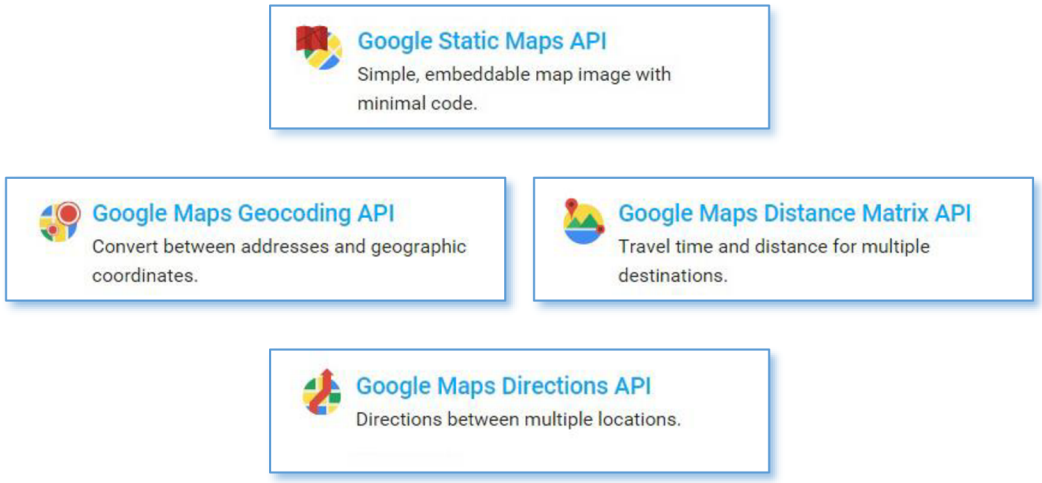
1. Prvý člen rastie s počtom nenulových neurónov v jenom riadku
2. Druhý člen rastie s počtom nenulových neurónov v jenom stĺpci,
3. Tretí člen je najmenší pre práve *m* spoje medzi miestami
4. Štvrtý člen počíta dĺžku cesty

Konštanty **A, B, C** a **D** určujú dôraz pre jednotlivé členy vo vzorci. Na voľbe ich hodnôt veľmi záleží. Doporučené sú: A = B = D = 500, C = 200.

2.4 Google Maps APIs

Google Inc. je jednou z najväčších korporácií, ktorá dodáva webové služby a produkty všeobecne prístupné a užitočné. Ich cieľom je roztriediť všetky informácie na svete. Služba **API** (*Application Programming Interface*) je zbierka funkcií a tried, ktoré určujú akým spôsobom sa majú funkcie knižníc volať zo zdrojového kódu programu. API funkcie sú programové celky, ktoré programátor volá namiesto vlastného naprogramovania.

Google ponúka Maps API na rôznych platformách ako sú **Android, iOS, Web APIs** a **Web Service APIs.** V práci sú použité rozhranie webové služby Google Maps:



Na použitie služby musíme zostaviť otázku na server pomocou webovej adresy:

```
https://maps.googleapis.com/maps/api/staticmap?parameters
```

Pri požiadavke sú stanovené podmienky:

- dĺžka webovej adresy nemôže byť dlhšia, než 2048 charakterov.
- Počet trasových bodov môže byť max. 8.
- Počet ciest medzi počiatočnými a koncovými body nesmie byť v jednej otázke viac ako 100.
- Za posledných 24 hodín sa dá požiadať o max. 2500 otázok a to zvlášť pri všetkých typov služieb

Odpoveď na našu otázku môže mať dve kódovanie:

- *json* – (odporúčané) indikuje výstup ako JavaScript Object Notation (JSON)
- *xml* – indikuje výstup formou XML

2.4.1 Google Static Maps API

Pomocou tejto služby si môžeme stiahnuť vybranú časť mapy z Google Mapsu pomocou URL dotazu. Služba potrebuje od nás nasledujúce parametre:

Alokačné parametre

- *center* – definuje stred mapy, pri dotaze je použitý pár parametrov {*latitude,longitude*} (napr. „49.217479, 16.606918“) alebo reťazec adresy (napr. NC Královo pole, Cimburková 4., Brno)
- *zoom* – definuje zväčšenie mapy, zadáva sa ako celé číslo z intervalu (0;20). **0** znamená najvzdialenejší pohľad a **20** najbližší.

Parametre mapy

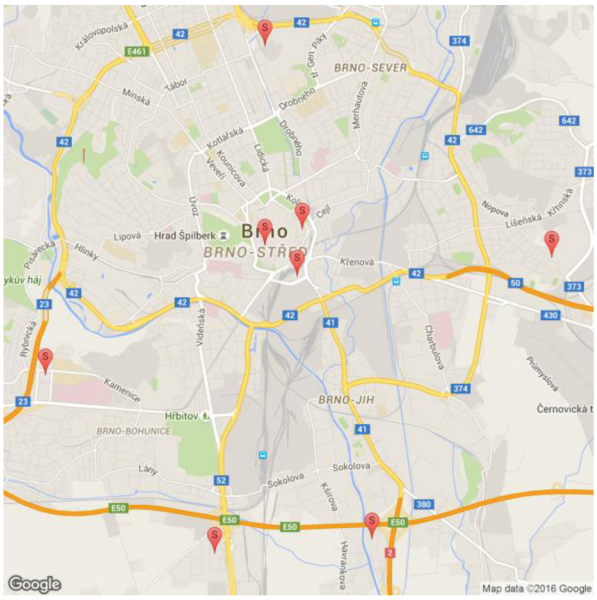
- *size* – definuje obdĺžnikové rozmery mapy spôsobom {*horizontal_value*}x{*vertical_value*}. Na tento parameter ešte spôsobí ďalší parameter *scale*.
- *scale* – definuje počet pixelov získaného obrazu. Môže nadobudnúť hodnoty (**1,2** a **4** pre prémiové zákazníci). *scale=2* znamená dvojnásobný počet pixelov než pri *scale=1*. Je to užitočné, keď pracujeme s veľkými plochami, alebo ho potrebujeme vytlačiť.
- *maptype* – definuje typ mapy, ktorá môže byť *roadmap*, *satellite*, *hybrid* a *terrain*.

Parametre funkcií

- *markers* – umožňuje vyznačiť body na mape. Pri použití tohto parametru nemusíme si definovať parametre *center* a *zoom*. Služba vypočíta spoločný stred mapy a optimálne zväčšenie.

Pri konštrukcii URL otázky musíme dodržať preddefinovanú formu:

```
https://maps.googleapis.com/maps/api/staticmap?parameters
```



Obrázok 2.11: Príklad Google Static Maps API

(zdroj: https://developers.google.com/maps/documentation/static-maps/intro)

2.4.2 Google Maps Geocoding API

Táto služba nám pomôže pomocou URL dotazu získať príslušné geografické koordináty (*Latitude/Longitude*) miesta, keď poznáme jeho presnú adresu.

Potrebný parameter

- *address* – presná adresa bytu podľa národnej poštovej služby. Následné upresňujúce adresy môžu výsledok kladne ovplyvniť.

```
https://maps.googleapis.com/maps/api/geocode/output?parameters
```

Príklad:

https://maps.googleapis.com/maps/api/geocode/json?address=NC+Kralovo+pole,+Cimburkova+4.,+Brno

```
lat =
      49.2175

lon =
      16.6069
```

(zdroj: https://developers.google.com/maps/documentation/geocoding/intro)

2.4.3 Google Maps Distance Matrix API

Služba prevedie vzdialenosti a časy medzi pôvodnou destináciou a ostatnými destináciami. Získané hodnoty sú vypočítané podľa odporúčanej cesty API. Informácie sú uložené v príslušných radoch, ktoré obsahujú kľúčové slová *distance* a *duration* pre všetky dvojice.

Potrebné parametre

- *origins* – jedna alebo viac adries a/alebo textové zemepisné súradnice, oddelené s charakterom **{ | }** , z ktorého sa vypočíta vzdialenosť a čas. Ak sa predá adresa ako reťazec, bude prevedená na zemepisné súradnice a následne budú vypočítané vzdialenosti.
- *destinations – jedna alebo viac adries a/alebo textové zemepisné súradnice,* oddelené s charakterom **|** , do ktorých sa počíta vzdialenosť a čas. Ak sa predá adresa ako reťazec, bude prevedená na zemepisné súradnice a následne budú vypočítané vzdialenosti.

Režimy cesty

- *driving (default)* – označuje výpočet vzdialeností s použitím cestnej siete.
- *walking* – výpočet vzdialeností pre peších pešie zóny a cez chodníky (ak sú k dispozícii).

- *bycicling* – výpočet vzdialeností pre jazdu na bicykli cez cyklotrasy a preferovaných ulíc (ak sú k dispozícii).
- *transit* – výpočet vzdialeností prostredníctvom verejných tranzitných trás (ak sú k dispozícii).

```
https://maps.googleapis.com/maps/api/distancematrix/output?parameters
```

Príklad:

https://maps.googleapis.com/maps/api/distancematrix/json?origins=49.177520,16.566122&destinations=49.189515,16.612934|49.191795,16.660315|49.193276,16.606979|49.195234,16.613926|49.157532,16.626835|49.217479,16.606918|49.155801,16.597555&mode=driving&sensor=false&language=en-EN&units=metric

```
matrix =

          []    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}
    {1x2 cell}    [       0]    [    5951]    [   10152]    [    6222]    [    6492]    [   11371]    [    7895]    [    5052]
    {1x2 cell}    [    5951]    [       0]    [    5974]    [    1655]    [    1375]    [    6352]    [    4548]    [    5124]
    {1x2 cell}    [   10152]    [    5974]    [       0]    [    6772]    [    5429]    [   10499]    [    6657]    [   11947]
    {1x2 cell}    [    6222]    [    1655]    [    6772]    [       0]    [    1546]    [    7023]    [    3626]    [    5434]
    {1x2 cell}    [    6492]    [    1375]    [    5429]    [    1546]    [       0]    [    6612]    [    3242]    [    5938]
    {1x2 cell}    [   11371]    [    6352]    [   10499]    [    7023]    [    6612]    [       0]    [   17794]    [    6338]
    {1x2 cell}    [    7895]    [    4548]    [    6657]    [    3626]    [    3242]    [   17794]    [       0]    [   14950]
    {1x2 cell}    [    5052]    [    5124]    [   11947]    [    5434]    [    5938]    [    6338]    [   14950]    [       0]
```

(zdroj: https://developers.google.com/maps/documentation/distance-matrix/intro)

2.4.4 Google Maps Directions API

Služba vypočíta smery medzi jednotlivými miestami. Smery sú špecifikované pôvodnou destináciou, konečnou destináciou a zastávkami a to pomocou reťazca charakterov alebo zemepisnými súradnicami.

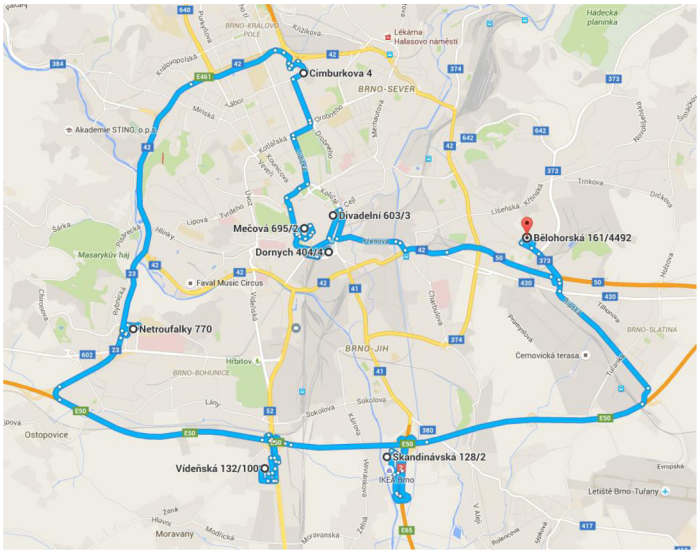
Potrebné parametre

- *origin* – adresa alebo zemepisné súradnice, z ktorého sa počítajú smery. Ak je zadaná adresa ako reťazec, bude prevedená na zemepisné súradnice a následne budú vypočítané smery.
- *destination* – adresa alebo zemepisné súradnice, do ktorého sa počítajú smery. Ak je zadaná adresa ako reťazec, bude prevedená na zemepisné súradnice a následne budú vypočítané smery. Parametre sú podobné ako u *origin*.

Voliteľné parametre

- *waypoints* – určuje trasové body, cez ktoré cesta vedie. Sú to adresy alebo zemepisné súradnice.

```
https://maps.googleapis.com/maps/api/directions/output?parameters
```



Obrázok 2.12: Google Maps Direction výsledok pre predajne Tesca v Brne (zdroj: https://developers.google.com/maps/documentation/directions/intro)

3 ANALÝZA SÚČASNÉHO STAVU

Tesco má v Česku viac ako **200** obchodov, ktoré zahŕňajú hypermarkety aj menšie lokálne formáty predajní. Spoločnosť dlhodobo zamestnáva takmer **14 000** ľudí a patrí tak medzi najvýznamnejších privátnych zamestnávateľov.

Tesco Stores ČR a.s v Českej republike prevádzkuje tiež **17** čerpacích staníc a **7** obchodných centier. Tesco prevádzkuje tiež franšízovú sieť **Žabka**, ktorá má viac ako **100** obchodov.

Ako prví umožnili českým zákazníkom nákupy potravín **on-line**. Služba v súčasnosti funguje celkom v siedmich krajoch ČR a je dostupná takmer štyrom miliónom Čechov.

3.1 Obecné údaje o spoločnosti

Pri voľbe podnikového subjektu som sa snažil naviazať spojenie s firmou, ktorá je na trhu dlhšiu dobu a vykazuje stabilné výsledky podnikania. Dôležitá bola aj aktuálnosť a rastúci záujem o ich služby.

Spoločnosť Tesco je jednou z najväčších maloobchodníkov a českým zákazníkom poskytuje širokú škálu tovaru a služieb prostredníctvom rozsiahlej siete obchodov rozličných formátov a prvá online obchod s potravinami a ďalším tovarom. (*zdroj: Veřejný rejstřík a Sbírka listin - Ministerstvo spravedlnosti České republiky*)

- **Dátum zápisu:** 23. marca 1992
- **Spisová značka:** B 1377 vedená na Mestskom súde v Prahe
- **Obchodná firma:** Tesco Stores ČR a.s.
- **Sídlo:** Praha 10, Vršovická 1527 / 68b, PSČ 10000
- Identifikačné číslo: 453 08 314
- **Právna forma:** Akciová spoločnosť
- **Základný kapitál:** 13 263 310 000,- Kč

3.1.1 Histórie firmy

Jack Cohen, vyslúžilý vojak britského kráľovského letectva, začal v roku 1919 predávať v stánku potraviny z nadbytočných vojnových zásob. O päť rokov neskôr uviedol na trh prvý výrobok vlastnej značky - čaj Tesco, ktorého názov vznikol z iniciálok mien spoločníkov TE Stockwell a Jack Cohen.

- V roku **1929** Jack Cohen otvára svoj prvý kamenný obchod v severnom Londýne
- **1932** - Tesco pozostáva komanditnou spoločnosťou
- **1934** - Jack Cohen kupuje v severnom Londýne pozemok, na ktorom stavia potravinový sklad. Zakladá nový systém centrálnej kontroly zásob a zabezpečuje prevádzku päťdesiatich obchodných jednotiek
- **1947** - Tesco vstupuje na burzu
- **1956** - Tesco otvára prvý samoobslužný obchod
- **1961** - Supermarket Tesco v Leicesteri je zapísaný do Guinessovej knihy rekordov ako najväčší obchod v Európe.
- Na český trh vstupuje spoločnosť Tesco v roku **1996**, kedy tiež rozširuje svoje pôsobenie na Slovensko a Maďarsko. Už za dva roky otvára v **Prahe na Zličíne** prvý hypermarket a **len o rok neskôr** víta prvých návštevníkov v Obchodnom centre Letňany, dnes jedného z najväčších v Českej republike.
- Od roku **2002** sa v predajných pultoch objavujú **prvé výrobky pod vlastnou značkou Tesco**.
- v roku **2005** je otvorená prvá **čerpacia stanica** v Karlových Varoch.
- Svoju sieť predajní Tesco významne rozširuje v roku **2006**, keď preberá **27** obchodov od **Edeka** a **11** obchodov od **Carrefouru**.
- Od roku **2007** je predstavený nový formát predajne **Tesco Expres**.
- Od septembra roku **2010** môžu zákazníci využívať ojedinelý vernostný program **Tesco Clubcard**.
- Od septembra **2010** ponúka Tesco v spolupráci so spoločnosťou Home Credit **Tesco Finančné služby**.
- Ďalšie rozširovanie predajní Tesco v Českej republike prebieha v roku **2011**, kedy od investičnej spoločnosti Penta spoločnosť kupuje sieť **129** obchodov **Žabka** a **47** predajní **Koruna**.

- Pod novou značkou **Tesco City** bol v auguste roku **2011** po rekonštrukcii otvorený obchodný dom Tesco v Pardubiciach.
- Ako prvý v Českej republike spúšťa spoločnosť Tesco v roku **2012** jedinečnú službu **Potraviny on-line**, ktorá umožňuje nákupy potravín cez internet.
- Od júna **2012** môžu zákazníci využívať v predajniach Žabka možnosť **bezkontaktné platby**.
- Nový spôsob nakupovania **"Klikni & vyzdvihni"** uviedlo Tesco v roku **2012**. Zákazníci si svoj nákup vyberú cez internet a sami ho vyzdvihnú v čase, ktorý im vyhovuje.
- V máji **2013** spustilo Tesco nového mobilného operátora **Tesco Mobile**. Partnerom v ČR je spoločnosť Telefónica.
- V ČR prebehla v novembri **2013** prvá **Národná potravinová zbierka** na pomoc ľuďom v núdzi, do ktorej sa zapojilo 35 obchodov Tesco. Spoločnosť Tesco bola iniciátorom tohto projektu v ČR.
- Od roku **2013** majú zákazníci Tesco možnosť využiť službu **Garancia kvality**, ktorá prináša možnosť vrátenia peňazí za tovar, s ktorým nie sú spokojní a to bez udania dôvodu.

3.1.2 Predmet podnikania

Predmetom podnikania spoločnosti sú:

- Montáž, opravy, revízie a skúšky plynových zariadení a plnenie nádob plynmi
- Činnosť účtovných poradcov, vedenie účtovníctva, vedenie daňovej evidencie
- Pekárstvo, cukrárstvo
- Výroba nebezpečných chemických látok a nebezpečných chemických prípravkov a predaj chemických látok a chemických prípravkov klasifikovaných ako veľmi toxické a toxické
- Poskytovanie služieb v oblasti bezpečnosti a ochrany zdravia pri práci
- Výroba, obchod a služby neuvedené v prílohách 1 až 3 živnostenského zákona
- Stráženie majetku a osôb
- Mäsiarstvo a údenárstvo
- Hostinská činnosť

- Očná optika
- Prevádzkovanie neštátneho zdravotníckeho zariadenia – optometria – ambulantnej starostlivosti
- Predaj kvasného liehu, konzumného liehu a liehovín

3.1.3 Vízia a stratégia firmy

V každom podnikaní je dôležité mať jasný cieľ. Vízia nám pomáha nastaviť smer, ktorým sa chceme uberať. Tesco je spoločnosť, ktorá stojí na svojich zákazníkoch a zamestnancoch, ktorí podporujú rast podnikania po celom svete a pomáhajú vytvárať ďalšie príležitosti pre rast.

Záleží im na tom, aby spoločnosť Tesco bola oceňovaná zákazníkmi v krajinách, kde pôsobia, miestnymi komunitami, zamestnanci a spolupracovníkmi a samozrejme tiež ich akcionármi. Preto stanovili nasledujúce vízie skladajúce sa z piatich elementov. Každý z nich popisuje, akou spoločnosťou chce Tesco byť.

1 - Chcenie a potreba po celom svete

Zásadné pre nich je, aby ich obchody boli nielen voľbou pre zákazníkov, ale aj miestom, kde ľudia chcú pracovať, ktoré je dobré pre ich podnikanie, ktoré oceňujú lokálne komunity a kde chce každý akcionár investovať.

2 - Spoločnosť príležitostí a rastu

Či už ide o jedlo alebo nepotravinársky tovar, knihy alebo digitálnu zábavu, bankovníctvo alebo stravovanie mimo domova, ich podnikanie je plné príležitostí ako pre zákazníkov, tak pre ich kolegov. Chcú, aby ich spoločnosť zakaždým ponúkala niečo nové.

3 - Moderné, inovatívne, plné príležitosťami

Úspech spoločnosti Tesco bol vždy založený na snahe lepšie než ktokoľvek iný porozumieť potrebám zákazníkov - a potom inovovať, aby sa ich život stal trochu

jednoduchším. Tento postoj, ktorý priniesol on-line nakupovanie, rozšírenú predajnú dobu, množstvo vlastných značiek, rozsah formátov od Express po Extra a mnohé ďalšie veci, ktoré ich robia tým, kto sú.

4 - Lokálny víťaz s globálnym zázemím

Maloobchod je vždy lokálny, pretože kultúry, chute, klíma, pravidlá sú odlišné. Ale základné zručnosti, ktoré sa naučili na jednom mieste môžu byť použité aj inde. Napríklad rozšírenie služby Potraviny online do ôsmich medzinárodných trhov v celej skupine by nebolo možné bez toho, čo sa naučili vo Veľkej Británii.

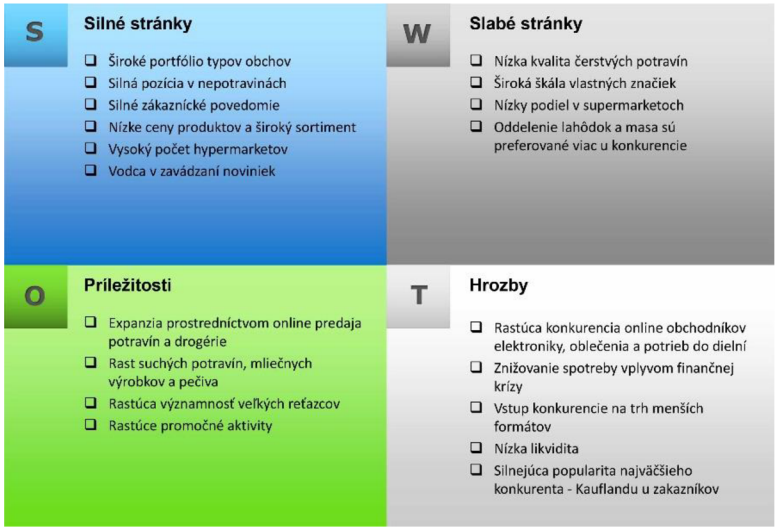
5 - Inšpiratívna spoločnosť a dôveryhodný partner

Chcú, aby Tesco bolo spoločnosťou, ktorá získava dôveru, nielen rešpekt z toho, čo robia - či už ide o príjemný nákup v našich obchodoch, náš sľub dobré ceny alebo naše odhodlanie zabezpečiť zákazníkom vysokú kvalitu potravín. Chcú byť obchod, ktorému zákazníci, kolegovia a komunity dôverujú a sú mu lojálni.

3.1.4 SWOT analýza

SWOT analýza odhaľuje silné a slabé stránky spoločnosti vo zrovnaní s príležitosťami a hrozbami trhu. Analýza ďalej sa sústredí aj na riešenie a príležitosti kde a ako ich môže využiť vo svoj prospech a naopak eliminovať hrozby, ktoré majú negatívny vplyv na pozíciu Tesca v budúcnosti. Skratka SWOT je odvozená zo slov:

- **Strengths** – silné stránky
- **Weaknesses** – slabé stránky
- **Opportunities** – príležitosti
- **Threats** – hrozby



Obrázok 3.1: SWOT analýza spoločnosti Tesco Stores a.s. (zdroj: KRIŽAN, 2011)

3.2 Online potraviny

Tesco bolo prvým maloobchodným reťazcom, ktorý v Českej republike otvoril svoju online predajňu. Táto aktivita bola založená predovšetkým na bohatých skúsenostiach z online predaja, ktorými Tesco disponuje vo svojej britskej centrále.

Obľuba internetových predajov u českých spotrebiteľov v posledných rokoch rýchlo rastie, útraty v tuzemských internetových obchodoch podľa odhadu Asociácie pre elektronickú komerciu v roku 2011 vzrástli na 37 mld. Kč, čo bolo o 4 mld. viac než v roku 2010. Hlavným dôvodom využitia internetových obchodov je nielen zvyčajne nižšia cena, ale i pohodlie a úspora času. Internetový predaj potravín je veľmi obľúbený napríklad vo Veľkej Británii, kde sa touto cestou predajú tri percentá všetkých potravín. Po online predaji tovaru každodennej spotreby s dodaním priamo domov je v Českej republike evidentný záujem – už v prvých dňom po spustení internetového obchodu Tesco sa doň zaregistrovalo niekoľko tisíc zákazníkov. Najväčší záujem je o čerstvé a trvanlivé potraviny, hlavne o základné položky, akými sú nápoje, ovocie, zelenina alebo pečivo. V budúcnosti plánuje spustenie internetového obchodu v Česku napríklad obchodný reťazec Ahold, ktorý prevádzkuje supermarkety a hypermarkety Albert, či obchodný reťazec Billa.

Stránky online obchodu: http://jaknakupovat.itesco.cz a http://nakup.itesco.cz

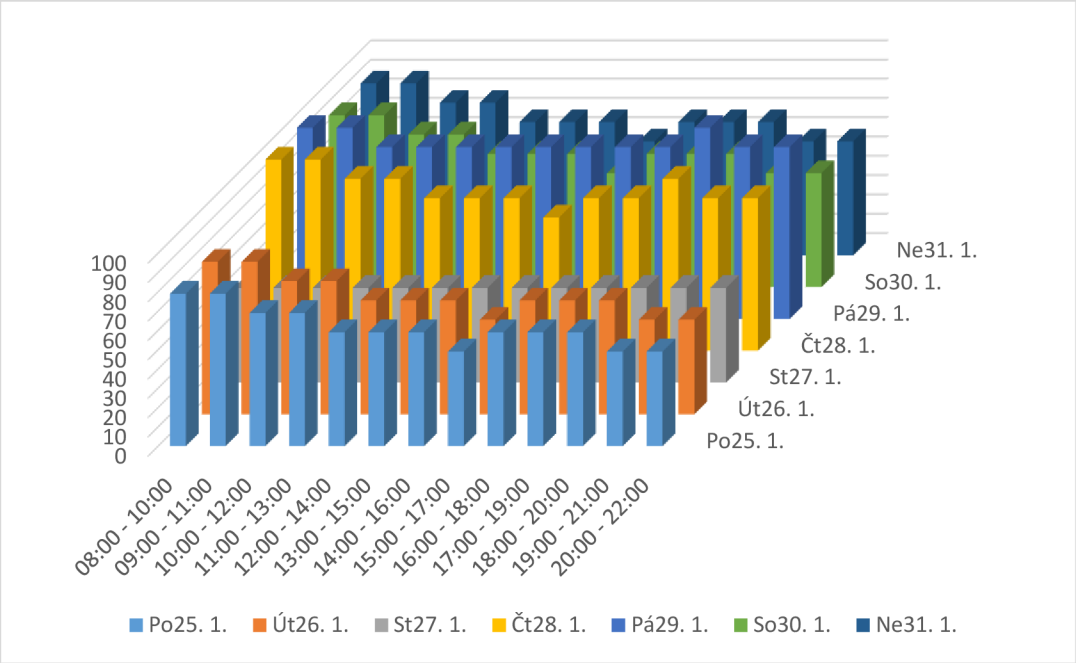
Pri nákupu cez online si môžeme zvoliť z dvoch spôsobov, ako sa k nákupu dostaneme:

- Doručenie ku dverám
- Klikni a vyzdvihni

Obrázok 3.2: Spôsoby doručenia tovaru (zdroj: http://nakup.itesco.cz)

Pri nákupu sa postupuje nasledovne:

- **Online účet** – keď sme stálym zákazníkom tejto služby je vhodný si nechať vytvoriť vlastný účet na stránkach, kde si pred vytvorením môžeme kontrolovať, či bývame v rozvozovej oblasti a to pomocou zadaní PSČ.
- **Výber spôsobu doručenia** – na *obrázku 3.2* je vidno, že môžeme si zvoliť buď osobný odber buď vývoz tovaru ku dverám.
- **Výber času doručenia** – v prípade, že sme zvolili vývoz tovaru, musíme si vopred vybrať čas jeho doručenia. Služba je dostupná denne od **8:00 – 22:00**. Cena dovozu záleží aj na vybranom časovom úseku. Na nasledujúcom *obrázku 3.3* si môžeme všimnúť, aké sú tie cenové rozdiely výrazné. Aktuálne sú ceny medzi **49 Kč** až **99 Kč**. Cenu ovplyvňujú hlavne:
    - **Štatistika** – sú dni, ktoré sú menej frekventované, ako napr. utorok a streda.
    - **Dopravné špičky** – ranné a večerné špičky negatívne ovplyvňujú doručenie tovaru v čas
    - **Extrémy** – večerné hodiny od **19:00 – 22:00** sú menej atraktívne a tak aj lacnejšie



Obrázok 3.3: Rozdelenie ceny dovozu podľa dňa a času (vlastný zdroj)

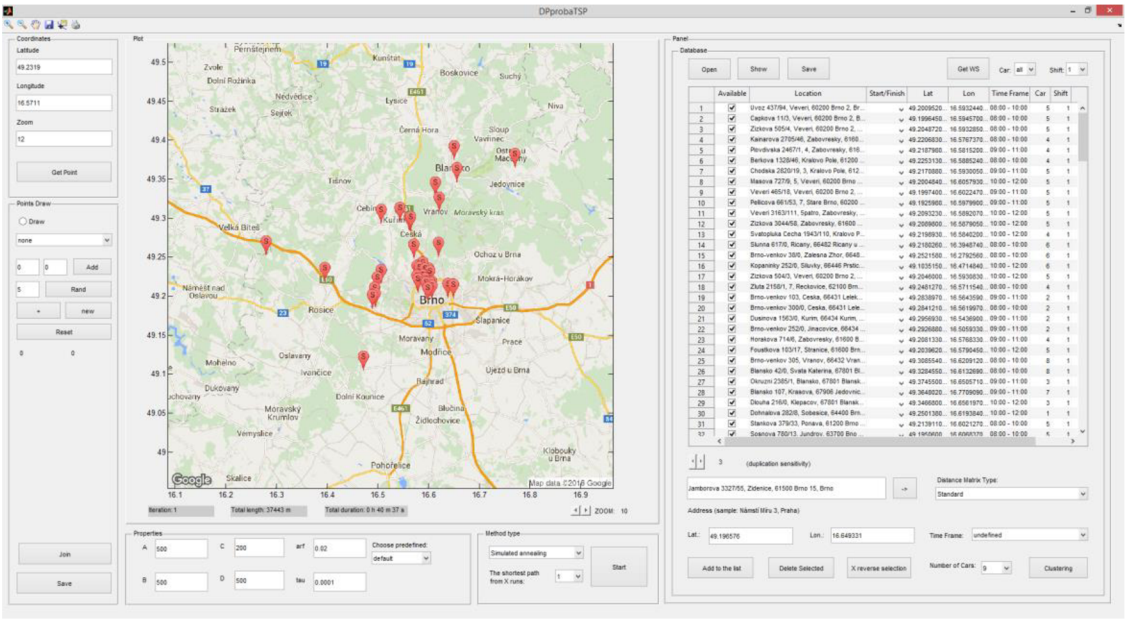
3.2.1 Produkty

Na webovej stránke spoločnosti si môžeme vyhľadať zoznam minulých nákupov. Tie produkty, ktoré sú často v košíku sú zapamätávané ako *obľúbené*. Môžeme si vytvárať *nákupné zoznamy*, ktoré nám uľahčia práci pri zostavení objednávky a urýchlia celý proces. Vyberať si môžeme z ôsmich kategórií potravín a ostatných tovarov:

- **Čerstvé potraviny** (pečivo, ovocie, zelenina, mlieko, syry, údeniny, mäso...)
- **Trvanlivé potraviny**
- **Mrazené potraviny**
- **Nápoje** (čaj, káva, rozpustné nápoje, džúsy, minerálne vody, pivo, víno, liehoviny...)
- **Drogérie** (pracie prostriedky, čistiace prostriedky, starostlivosť o kože, vlasy...)
- **Dieťa** (dojčenská výživa, kozmetika, príslušenstvo...)
- **Domáci maznáčik** (výrobky pre mačky, psy a ostatné zvieratá)
- **Domov a zábava** (knihy, domov, zábava, hračky, športové potreby, auto...)

4 VLASTNÉ NÁVRHY RIEŠENIA

4.1 Jednotlivé logické celky programu



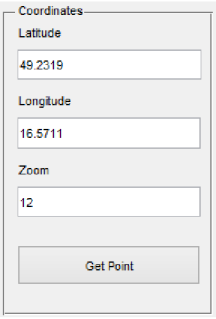
Obrázok 4.1: Hlavné okno programu (vlastný zdroj)

Program obsahuje 5 základných panelov:

- Panel na zadávanie koordinátou v geografickom súradnicovom systéme
- Panel na zadávenie koordinátou v Karteziánskej sústave súradníc
- Panel pre výber a nastavenie algoritmu
- Panel pre správu databázy
- Panel na zobrazenie výsledkov

4.1.1 Panel na zadávanie koordinátou v geografickom súradnicovom systéme

Užívateľ môže pridať miesto do záznamu pomocou tohto panelu. Po zadaní koordinátou program stiahne z Google serveru mapu so zväčšením definovaným užívateľom.



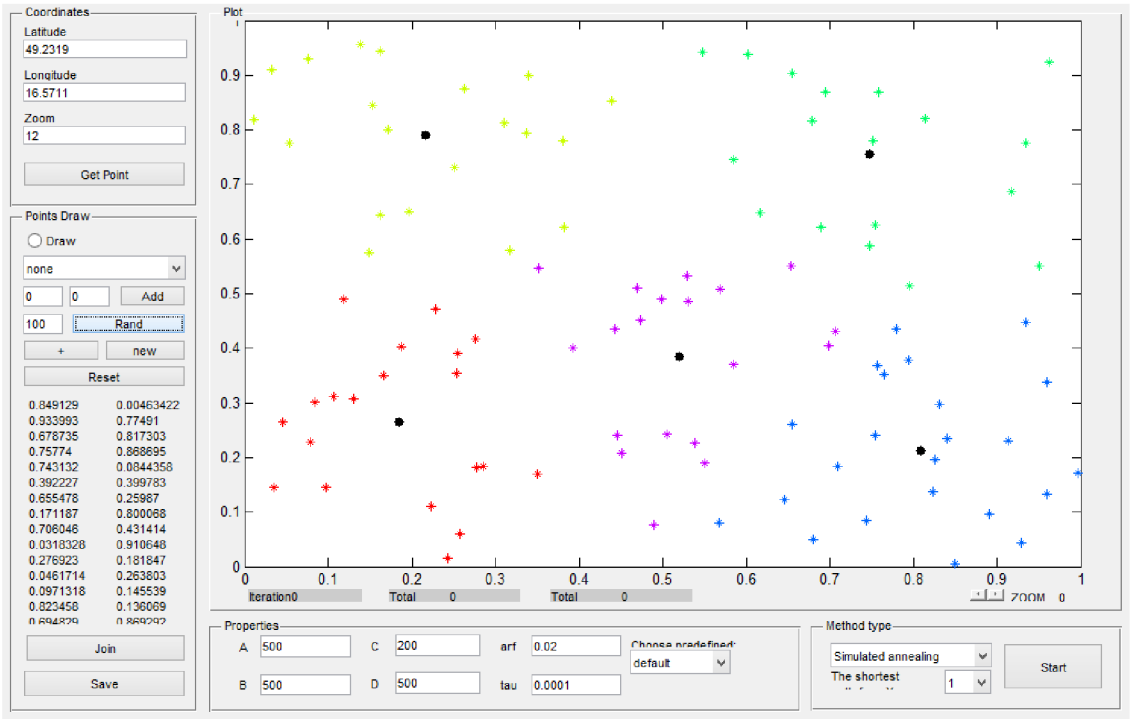
- **Latitude** – zemepisná šírka
- **Longitude** – zemepisná dĺžka
- **Zoom** – nastavenie zväčšenia v rozmeroch (*0 – 20*)

Obrázok 4.2: Panel na zadávanie koordinátou v geografickom súradnicovom systéme (vlastný zdroj)

4.1.2 Panel na zadávenie a kreslenie bodov

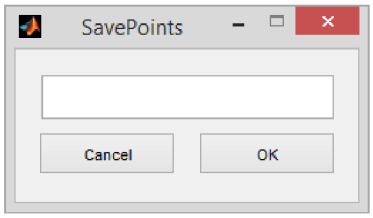
Po aktivácii opcie **Draw** sa plocha na kreslenie prepne na čistú, šedú podobu, kde si užívateľ môže definovať body, ktoré bude chcieť prepojiť. Na tento účel máme na výber štyri možnosti:

1. Otvoriť uložené
2. Zadať pomocou *XY* koordinátami
3. Generovať náhodne
4. Pridať po jednom pomocou kurzoru / vymeniť posledný zadaný bod na nový



Obrázok 4.3: Vykreslené a zhlukované body (vlastný zdroj)

Stlačením **Join** sa tieto body prepoja, zelené spojenie je medzi prvým a posledným bodom. Táto funkcia má hlavne teoretický význam. Pomocou nej sa dá simulovať, či je algoritmus funkčný. Sú to tzv. trénovacie body. Obľúbenú trasu si môžeme aj uložiť.



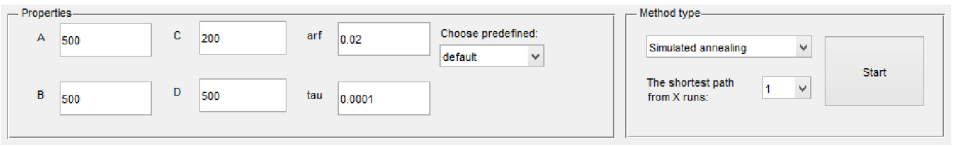
Obrázok 4.4: Uloženie zadávaných bodov (vlastný zdroj)

4.1.3 Panel pre výber a nastavenie algoritmu

Do práce sú implementované dva algoritmy:

- Simulované žíhanie
- Neurónové siete

Predvolený je genetický algoritmus – simulované žíhanie. U neurónových sietí si môžeme definovať konštanty, ktorými nastavujeme dôraz na jednotlivé častí vo vzorci. Najpoužívanejšie hodnoty sú uložené a program poskytuje výber.

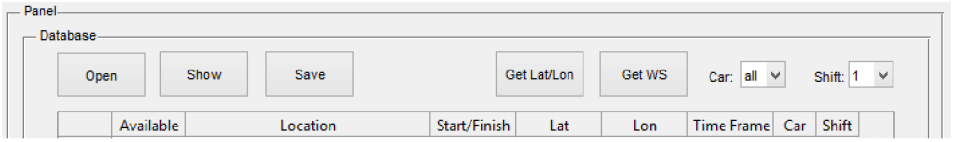


Obrázok 4.5: Výber a nastavenie algoritmov (vlastný zdroj)

Je tam ešte položka, ktorá slúži na definovanie počtu opakovaného výpočtu optimálnej trasy.

4.1.4 Panel pre správu databáze

Po štarte programu sa zobrazí kombinácia **Panel** – **Database**. Slúži na správu databázy, ktorá obsahuje informácie o miestach.

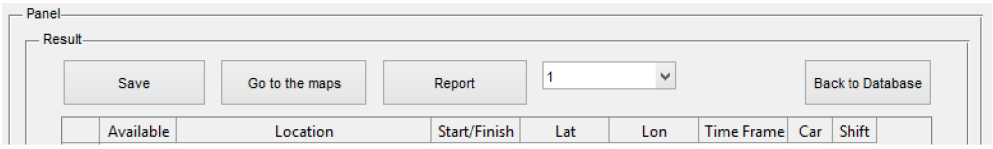


Obrázok 4.6: Správa databáze (vlastný zdroj)

- **Open** – načítanie databázy z Excel súboru
- **Show** – vykreslenie všetky miesta, ktoré majú položku **Available – True**
- **Save** – uloženie všetkých miest, ktoré majú položku **Available – True**
- **Get Lat/Lon –** automaticky doplní zemepisné súradnice k adresám
- **Get WS** – automatické priradenie jednotlivých miest do vhodnej smeny
- **Shift** – výber smeny
- **Car** – výber auta

Po odštartovaní výpočtu sa panel prepne do kombinácie **Panel – Result**. Panel obsahuje miesta vo vypočítanom poradí. Po výpočte máme viac možností:

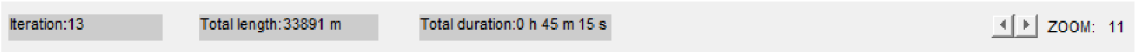
- **Save** – uloženie trasy do Excelovského súboru
- **Go to the maps** – po výbere zo zoznamu si môžeme vykresliť trasu v Google Mapsu v tom poradí, ako je to optimálne, najkratšie.
- **Report** – generovanie reportu, ktorý obsahuje výsledky všetkých trias. Ide o Excelovskú tabuľku, ktorá má počet zošitov ako počet nastavených smien.
- **Back to Database** – v prípade, že je potrebné nastaviť vstupy inak, sa môžeme vrátiť do panela *Database* a vykonať zmeny.



Obrázok 4.7: Panel výsledkov (vlastný zdroj)

4.1.5 Panel na zobrazenie výsledkov

Zobrazovací panel slúži na vykreslenie mapy získanej z Google serveru. Okrem vyznačení miest obsahuje aj vypočítané trasy po skončení algoritmu. Poskytuje nám aj informácie o počte potrebných **iterácií** pri optimalizácii, **celkovej dĺžky** trasy v metroch a **celkové trvanie cesty** vo formáte hh:mm:ss.



Obrázok 4.8: Panel na zobrazenie vypočítaných trias (vlastný zdroj)

Pri prosbe od serveru je potrebné nastaviť aj optimálne zväčšenie mapy tak, aby všetky miesta, ktoré potrebujeme mať v jednej mape boli viditeľné. Program automaticky počíta

najoptimálnejšie zväčšenie. Stáva sa, že niektoré miesta sú relatívne ďaleko od ostatných a nútia ku zníženiu zväčšenia. Doladenie hodnoty môžeme korigovať manuálne.

4.2 Popis postupu

Vstupné informácie

V práci som si stanovil nasledujúce vstupné podmienky:

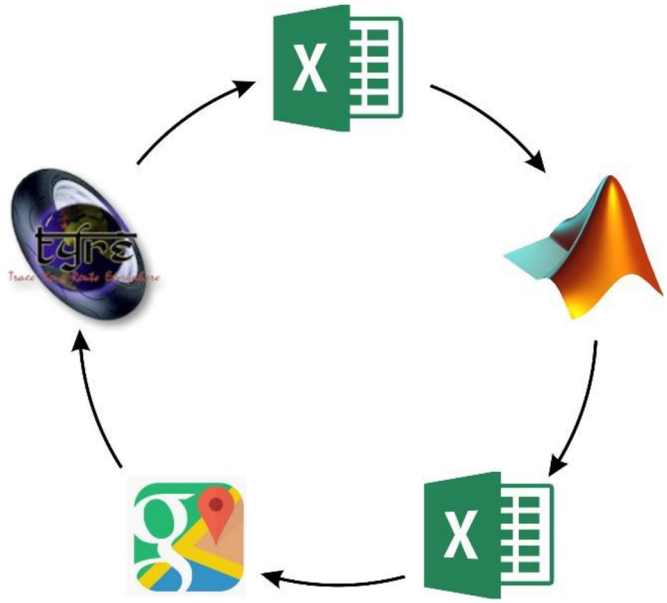
1. Zoznam miest
2. Tovar sa objednáva v daných dvojhodinových časový úsekoch
3. Na vývoz je k dispozícii max. 12 áut
4. Vývoz je rozdelený do smien
5. Pri vývoze sa počíta aj s extra časmi, ktoré sú tvorené z položiek:
    a. Trvanie cesty z jedného miesta do druhého
    b. Lokalita, kde sa miesto nachádza
    c. Forma platby – karta/online/hotovosť
    d. Štatistika – získané informácie od šoférov
    e. Priepustnosť premávky – pracovné extrémy
6. Existuje hranica vzdialenosti vývozu (oblasť, v ktorej je služba dostupná)

Výstupné údaje

Excelovská tabuľka, ktorá obsahuje report výpočtu:

1. Rozdelenie miest do smien
2. V každej smene je rozdelenie miest podľa predošlého zhlukovania na jednotlivé autá
3. Zoradenie miest podľa najkratšej cesty pre všetky autá
4. Stanovené časy príjazdu pri jednotlivých miest
5. Kontrola možnosti dodržania plánovanej trasy
6. Vykreslenie trasy pomocou Google Mapsu
7. Generovanie .gpx súboru pre nahranie trasy do GPS pomocou programu Tyre

Pri riešení problematiky som bral do úvahy vyššie zmienené vstupné podmienky. Celý cyklus pri používaní aplikácie TSP je nasledovný:



Obrázok 4.9: Cyklus pri používaní aplikácie TSP (vlastný zdroj)

1. Dátový vstup – Excel tabuľka obsahujúca miest
2. Spracovanie údajov v programe písanom v MATLABu
3. Report – uložený v Excel tabuľke
4. Zobrazenie vypočítaných trias na Google Mapsu
5. Vytvorenie .gpx súboru pre GPS

4.2.1 Databáza

Prvým krokom je zostavenie databázy. Môžeme si otvoriť uloženú v Microsoft Excelu alebo zadať po jednom pomocou grafického rozhrania. Databáza musí mať nasledujúci preddefinovaný tvar:

	A	B	C	D	E	F
1	TRUE	Uvoz 437/94, Veveri, 60200 Brno 2, Brno	Start&Finish	49.200952	16.593244	08:00 - 10:00
2	FALSE	Capkova 11/3, Veveri, 60200 Brno 2, Brno		49.199645	16.59457	08:00 - 10:00
3	TRUE	Zizkova 505/4, Veveri, 60200 Brno 2, Brno		49.204872	16.593285	08:00 - 10:00

Obrázok 4.10: Vzorová databáza (vlastný zdroj)

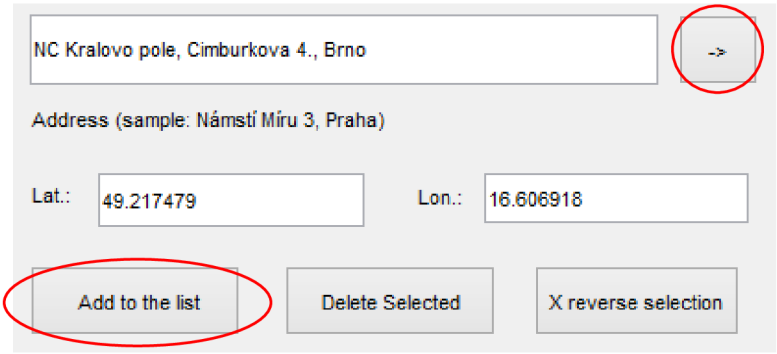
Jednotlivé stĺpce znamenajú:

- **A** – môže byť TRUE/FALSE, po otvorení databázy miesto označené TRUE bude zahrnuté do výpočtu optimálnej trasy.
- **B** – prvoradá informácia o navštívenom mieste, zákazník pri objednaní tovaru musí zadať presnú adresu, kam tovar treba doviesť.
- **C** – pri dodaní tovaru musí auto vychádzať z jedného miesta (najčastejšie obchod so skladom) a tam sa aj vrátiť. Iba jedno miesto môže byť nastavené ako Start&Finish.
- **D a E** – druhoradá informácia o navštívenom mieste, sú to geografické hodnoty miesta. Najčastejšie túto informáciu nepoznáme, ale je nevyhnutná k výpočtu. Po otvorení databázy v programe môžeme tieto informácie získať od Google Maps Geocoding API jedným kliknutím 

. Hodnoty majú 6 desatinnú presnosť.
- **F** – obsahuje časový slot, kedy musí byť tovar u zákazníka. V tomto prípade ide o dvojhodinové úseky (od 8:00 do 22:00).

Pri zadávaní miest pomocou grafického rozhrania je potrebné vedieť aspoň jednu informáciu o mieste. Môže to byť **adresa** alebo geografické súradnice v poradí **1. zemepisná dĺžka** a **2. zemepisná šírka**. Po zadaní adresy stlačíme 

. Automaticky sa vyplnia položky o zemepisných súradniciach. Stlačením **Add to the lsit** môžeme miesto pridať do databázy.



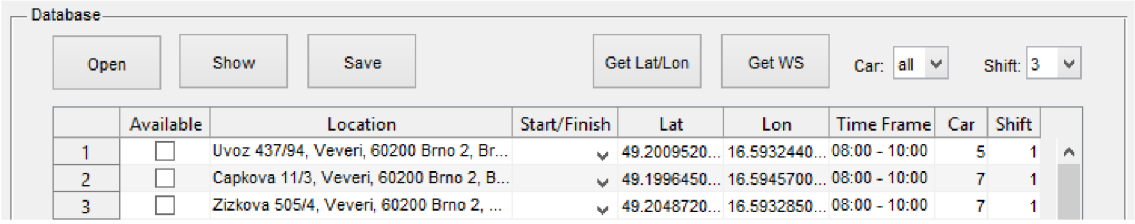
Obrázok 4.11: Zadávanie vstupov pomocou grafického rozhrania (vlastný zdroj)

Pozn.: pri zadávaní adresy musíme dávať pozor, aby sme nepoužívali diakritiku.

Pri zadávaní manuálnym spôsobom prebieha kontrola duplicity miest. Algoritmus pozoruje duplicity podľa zemepisných súradníc a to s presnosťou podľa voľby od 1 – 5. Tieto hodnoty znamenajú citlivosť kontroly, ktorá je založená na desatinných súradníc. 1 znamená najväčšiu citlivosť, 5 najmenšiu.

4.2.2 Predspracovanie dát

Po načítaní miest do okna **Database** si musíme najprv rozdeliť navštívené miesta do jednotlivých smien. Pomocou **Get WS** rozdelenie prebehne automaticky podľa fixne danej podmienky. Po výbere smeny pomocou **Shift** sa vykreslia miesta, ktoré patria do zvolenej smeny. Stlačením **Show** si môžeme vykresliť práve označené položky (stĺpec **Available**) do zobrazovacieho panelu.



Obrázok 4.12: Predspracovanie dát (vlastný zdroj)

V práci som definoval 3 smeny po časových úsekoch 8:00 – 12:00, 12:00 – 18:00 a 18:00 – 22:00.

Pozn.: adresy na *obrázku 4.2* sú iba náhodne generované pre testovanie funkčnosti aplikácie.

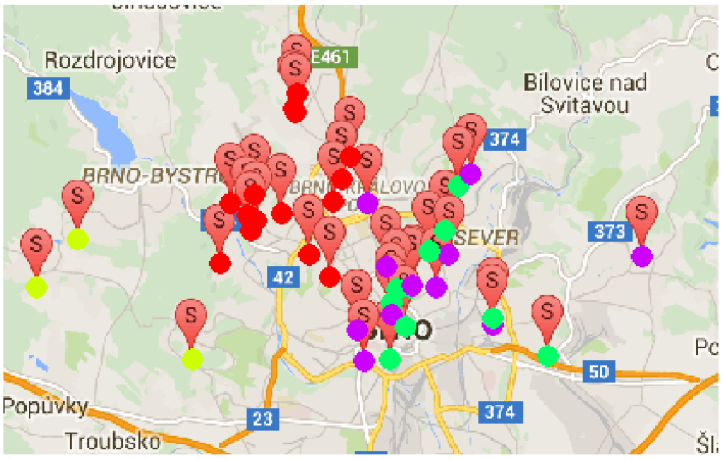
Po druhé prebehne zhlukovanie miest v danej smene. Je to automatický proces, ktorý používa metódu **K-means**. Po vybraní smeny zo zoznamu počet miest vo smene sa priemeruje a nastaví sa počet áut. Je to začiatočná podmienka stavená na minimalizovanie potrebného počtu áut. Po stlačení **Clustering** algoritmus zhlukuje miesta a vykreslí ich do zobrazovacieho panelu farebne odlíšene.

Aby sa nestalo, že jedno auto má za prácu navštíviť 20 miest a druhé iba 2, algoritmus spočíta 50 možných variant zhlukovania a vyberie to najvhodnejšie podľa vzorcu:

$$max(i_{11} \cdot i_{21} \cdot \ldots \cdot i_{n1}) \tag{4.1}$$

kde i_{11} – je počet pridelených miest pre prvé auto v prvom cykle. Výsledok bude mať najväčšiu hodnotu v prípade, keď každé auto bude mať ten istý počet pridelených miest. Pravdepodobnosť rovnakého počtu je veľmi malá, hlavne keď pracujeme s reálnymi dátami.

Po získaní najlepšej varianty ešte vždy sa môže stať, že niektoré auto dostane príliš veľký počet miest. Algoritmus počíta aj s touto situáciou. Rieši to zvýšením počtu áut o jedno a náhodným pridelením polovice miest k pridanému autu.



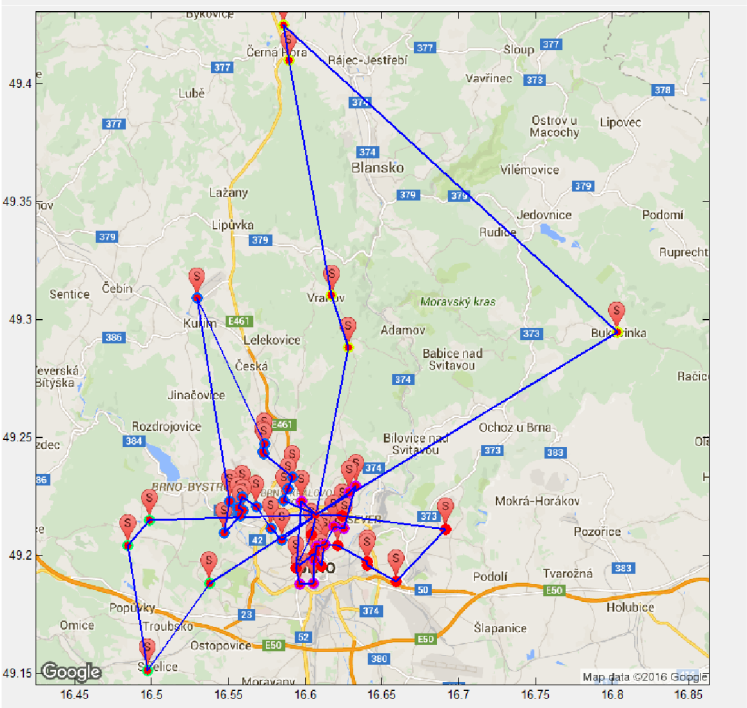
Obrázok 4.13: Zhlukovanie miest pre jednotlivé autá (vlastný zdroj)

4.2.3 Výpočet optimálnej trasy

Po implementovaní rozraďovania na smeny a zhlukovania pre jednotlivé autá sa môže začať výpočet optimálnej trasy. Problém viacerých obchodných cestujúcich je rozdelený pomocou zhlukovania na menšie TSP. Teraz sa už výpočet môže použiť pre všetky autá zvlášť po jednom.

Algoritmus automaticky prevedie výpočet od prvého auta až do posledného. Proces optimalizácie je krok za krom viditeľný a trasy zostanú vykreslené po jeho skončení. Informácie o výsledkoch sú uložené do bufferu a sú k dispozícii, pokiaľ sa proces znova nespustí. V tomto kroku máme dve možnosti:

1. Nechať si vykresliť trasy na **Google Mapsu**, aby sme si mohli vytvoriť **.gpx** súbory – prípad, keď trasy potrebujeme priamo nahrať do navigačného systému (digitálna reprezentácia výsledkov)
2. Vygenerovať si report – všetky informácie o trasách sú uložené v Excelovskej tabuľke, ktorá je automaticky pomenovaná podľa aktuálneho dátumu a času (tlačová reprezentácia výsledkov)



Obrázok 4.14: Vypočítané trasy pre jednotlivé autá (vlastný zdroj)

Aplikácia funguje nasledujúcim spôsobom:

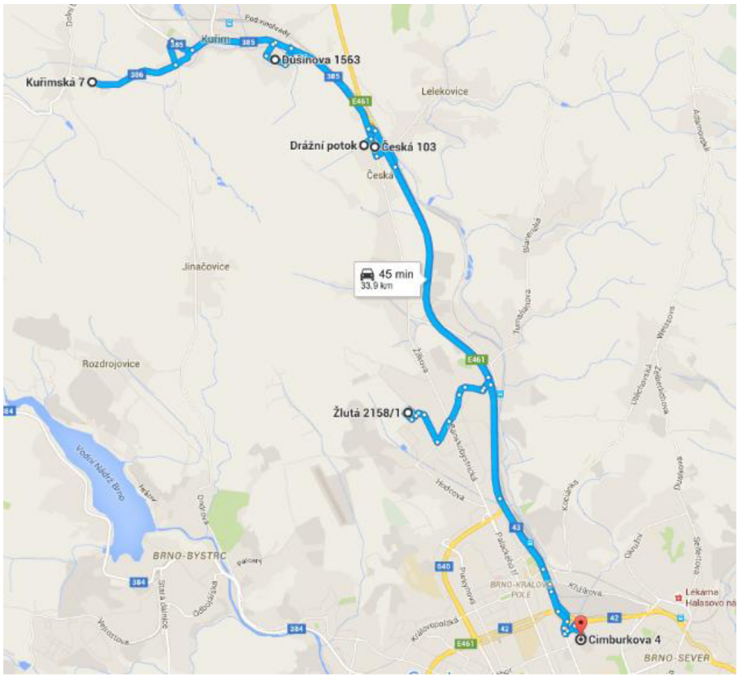
1. Výber vstupu podľa toho, či pracujeme s databázou miest alebo s generovanými dátami v Karteziánskom súradnicovom systéme
2. Načítanie vstupov:
    a. Databáza miest
    b. Počet áut
    c. Číslo smeny
    d. Adresa začiatočná/konečná
3. Separovanie miesta do menších štruktúr podľa áut
4. Nastavenie počtu hlavného cyklu podľa počtu áut
5. Získanie matice vzdialeností medzi miestami podľa:
    a. Priamej vzdialenosti
    b. Presných údajov od Google Distance Matrix API v metroch
    c. Presných údajov od Google Distance Matrix API v sekundách
6. Podľa voľby algoritmu
    a. Simulované žíhanie
    b. Neurónové siete

7. Načítanie parametrov zadaných užívateľom
8. Samotný beh algoritmu
9. Výstupom je vektor indexov ukazujúci na jednotlivé miesta v zozname
10. Nastavenie poradia miest, aby začiatočná adresa bola aj prvá aj posledná v zozname
11. Získanie presných informácií o vytvorenej trase z Google Maps Directions API
12. Výpočet celkovej dĺžky trasy a potreného času na jeho zvládnutie zatiaľ bez zohľadnenia na extra časy
13. Uloženie všetkých informácií o trase do bufferu
14. Opakovanie predošlých bodov od 5 až do 13 podľa počtu áut vo smene
15. Uloženie výsledkov do premennej štruktúry

V tomto kroku máme optimalizované trasy iba pre jednu smenu. Aby sme nestratili výsledky stlačením **Report** musíme si údaje uložiť do tabuľky a potom môžeme výpočet aplikovať podobne aj pre ostatné smeny.

4.2.4 Zobrazenie trasy v Google Mapsu

Pre bližšie informácie o danej trase nám ponúka webová služba Google Maps. Jedným kliknutím na **Go to the maps** sa nám otvorí webový prehliadač (ktorý je nastavený ako predvolený). Od tohto kroku je užívateľ schopný použiť každú výhodu, ktorú nám táto služba ponúka. Zmeniť poradie miest, zvoliť si inú variantu trasy, získať podrobné informácie o trase.



Obrázok 4.15: Zobrazenie trasy v Google Mapsu

4.2.5 Výstup z programu

Pre zobrazenie výsledkov program generuje report v podobe Excel tabuľky. Jednotlivé smeny sú naplánované po jednom. Z tohto dôvodu ich môžeme reportovať v jednej tabuľke, kde program sleduje, ktorá smena je práve aktívna a uloží výsledky podľa čísla smeny do zošitu s tým číslom. Druhá možnosť je uložiť ich zvlášť do novej tabuľky.

Report

Směna 1 (8:00-12:00)

auto 1	Časový úsek	Adresa	Vzdialenost	Trvání	Extra čas [sec]	Plánovaný čas příjezdu	Status
1	08:00 - 10:00	NC Kralovo pole, Cimburkova 4., Brno	0	0	0	8:00:00	OK
2	09:00 - 11:00	Brno-venkov 103, Ceska, 66431 Lelekovice, Ceska	8.9 km	8 mins	840	8:22:01	OK
3	08:00 - 10:00	Brno-venkov 300/0, Ceska, 66431 Lelekovice, Ceska	0.3 km	1 min	840	8:36:50	OK
4	09:00 - 11:00	Dusinova 1563/0, Kurim, 66434 Kurim, Kurim	2.6 km	4 mins	840	8:55:01	OK
5	09:00 - 11:00	Brno-venkov 252/0, Jinacovice, 66434 Kurim, Jinacovice	3.8 km	7 mins	840	9:16:02	OK
6	08:00 - 10:00	Zluta 2158/1, 7, Reckovice, 62100 Brno 21, Brno	11.7 km	16 mins	840	9:45:42	OK
7	08:00 - 10:00	NC Kralovo pole, Cimburkova 4., Brno	6.7 km	10 mins	840	10:09:15	OK

Tabuľka 2: Report výsledkov pre jedno auto v prvej smene (vlastný zdroj)

Po generovaní výsledku hodnoty sú ešte zmeniteľné a dajú sa nastaviť extra časy zvlášť pre každé miesto. Stĺpec **Plánovaný čas príjazdu** obsahuje vzorec na kumulovanie všetkých extra časov a zohľadňuje odjazd z predošlého miesta.

4.3 Nahranie trasy do GPS

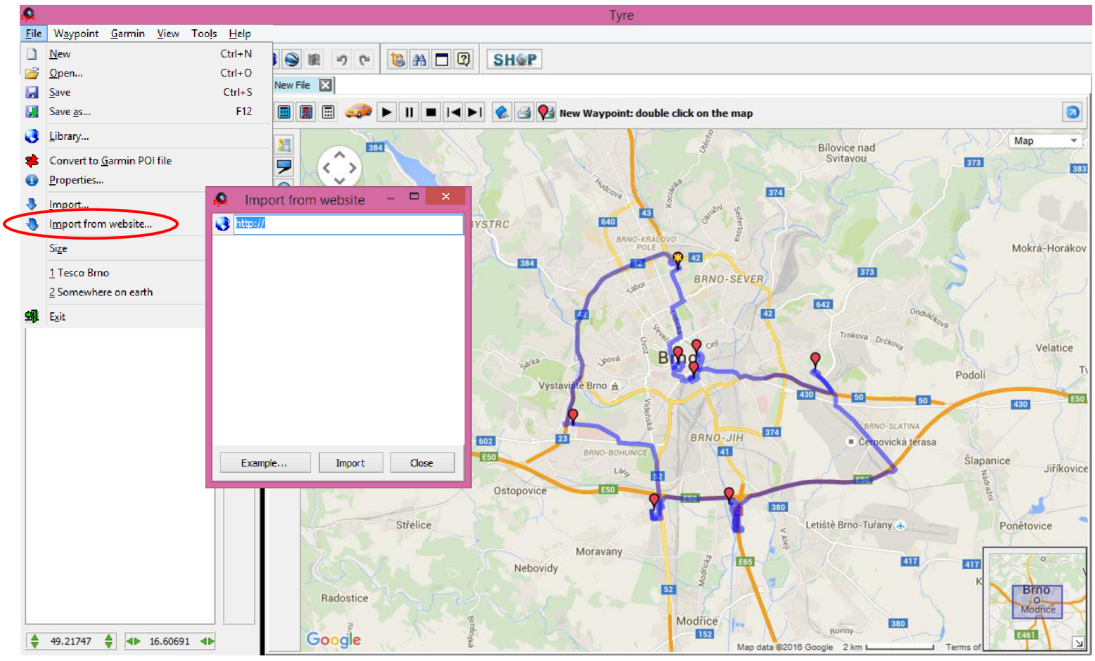
Posledným krokom celého procesu je nahrať optimalizovanú trasu do GPS. Aj tento krok by mohol zobrať zbytočne veľa času, keby sme museli zadávať jednotlivé destinácie do navigačného systému ručne. Okrem časového hľadiska si musíme brať do úvahy aj možnosť ľudskej chyby.

Firma používa navigačné zariadenie typu Garmin nüvi 1350, Garmin nüvi 52LM, Be On The Road (System Windows Embedded CE). Tieto zariadenie si môžeme spojiť s počítačom a tvária sa ako úložisko vo Windows Exploreru. Po otvorení tohto úložiska nájdeme zložku **...\Garmin\GPX**, kam si môžeme nahrať **.gpx** súbory, ktoré obsahujú uložené trasy z predošlých výjazdov.

4.3.1 Prepojenie Google Maps s GPS

V práci som používal webové rozhranie od spoločnosti Google. Po výpočte optimálnej trasy s jedným klikom si môžeme náš výsledok zobraziť v Google Mapsu.

Pomocou freeware programu **Tyre** si môžeme trasu importovať a následne uložiť v kompatibilnom formáte **.gpx**. (*http://www.tyretotravel.com/download-tyre*).



Obrázok 4.16: Nahranie URL do programu Tyre (vlastný zdroj)

V tomto kroku si môžeme ešte trasu naladiť podľa potreby. Uložiť trasu si môžeme nasledovne: ***File -> Save As -> XY.gpx***

4.3.2 Odhalené problémy pri vývoji softvéru

Pri programovaní tejto aplikácie som sa stretol niekoľkými zásadnými problémami, agresívnymi podmienkami, matematickými prekážkami a problémami so spravovaním databázy. Toto prispeli ku komplexnosti programu. Boli to:

1. Definovanie formy vstupu, aby obsahoval všetky nevyhnutné informácie

2. Prekonať prekážky, ktoré boli stanovené zo strany Google Maps API, kde maximálny počet zastávok medzi začiatočným a konečným bodom je **8**. Pre jedno auto je ale stanovený maximálny počet **13** (bez začiatočného a koncového bodu). To znamená, že pri získaní údajov zo serveru som si musel otázku rozdeliť a následne korektne spojiť údaje. Pri rozdeľovaní miest do menších skupín nastáva problém, keď v poslednej skupine zostáva iba jedno miesto.
3. Nastavenie osí po zobrazení mapy v zobrazovacom paneli, aby kreslenie na mapu bolo presné. Bolo treba vypočítať relatívne rozdelenie zemepisných súradníc na Karteziánske súradnice. Hranice osí sú vypočítané podľa **rozlíšenia** mapy, veľkosti **zväčšenia** a **pomeru** podľa týchto vzorcov:

```
x1 = newCenterLon - 1280*0.002739726/8/(2^(zoom-10));
x2 = newCenterLon + 1280*0.002739726/8/(2^(zoom-10));
y1 = newCenterLat - 1280*0.001782531/8/(2^(zoom-10));
y2 = newCenterLat + 1280*0.001782531/8/(2^(zoom-10));
```

5 ZÁVER

Cieľom práce bolo optimalizovať náklady pri podnikaní v zvolenom objekte. Pre tento účel boli použité metódy z oblasti umelej inteligencie. Pre užívateľské možnosti bola vytvorená aplikácia s grafickým rozhraním písaná v skriptovacom jazyku MATLAB. Z optimalizačných problémov týkajúci sa profilu spoločnosti Tesco Stores ČR a.s bol vybraný známy nedeterministicky – polynomiálny problém obchodného cestujúceho. Tento problém sa týka online obchodu spoločnosti.

Pri voľbe podnikového subjektu som sa snažil nadviazať spojenie s takou firmou, ktorá je na trhu dlhšiu dobu a vykazuje stabilné výsledky podnikania. Dôležitá bola aj aktuálnosť a rastúci záujem o ich služby. Tesco Stores ČR a.s je jedným z najväčších obchodných celkov v Českej Republike, kde pôsobí už viac ako 20 rokov. Na Morave stanovil základné podmienky a vybudoval novú službu – **online obchod** v roku 2012.

Na optimalizáciu trias áut, čo znamená nižšie náklady na palivo a údržbu, boli použité dva celkom rozdielne algoritmy. Simulované žíhaní patrí do kategórie evolučných genetických algoritmov. Je jedným najefektívnejším optimalizačným algoritmom vo svojej kategórii. Podľa experta algoritmov, *Stevena Skiena* – **„Riešenie pomocou simulovaného žíhania funguje nádherne“**.

Druhým nástrojom boli neurónové siete usporiadané podľa Hopfieldovej sieťovej štruktúry. Tvoria dynamický systém pri ktorého sa dá merať jeho energia. Používa jednu vrstvu neurónov a jeden cyklus učenia. Dajú sa efektívne využiť na predstavenú problematiku. Pri vhodnom nastavení môžu byť výsledky oveľa optimálnejšie, než u genetických algoritmoch.

Treba dodať, že v práci je používaný výraz „optimálne“ a nie „najoptimálnejšie“. Je to z dôvodu, ži pri optimalizácie môžeme ukotviť pri lokálnom extréme namiesto globálneho. Napriek tomu sa dajú považovať výsledky za použiteľné.

Problém obchodného cestujúceho je známou problematikou a existuje k jej riešeniu rada metód. Nároky a podmienky podnikového subjektu viedli k náročnému vývoju softvéru. Hlavným cieľom podniku je, aby dostali program, ktorý kontroluje výsledky aktuálneho plánovača a prípade ponúkne lepšie riešenie. Program je schopný

kontrolu previesť a navyše zhlukovať, optimalizovať a reportovať trasy pre autá až od čistej databázy miest, ktoré majú byť navštívené počas jedného dňa.

V programe je použité veľké množstvo programátorských a matematických riešení. Napriek tomu neobsahuje všetky potrebné funkcie, aby mohol pracovať samé a nahradiť súčasný plánovač. Takéto je zohľadnenie časových úsekov pri vypočítaní optimálnej cesty. Zohľadnenie časových úsekov ale prináša so sebou skutočnosť, že cesta možno nebude najkratšia, maximálne najkratšia možná.

Pri písaní práce som sa dozvedel veľa nových informácií o umelej inteligencii, programovaní a možnostiach riešenia problémov. Komunikácia s podnikovým subjektom a stanovenie hraníc obsahu práce boli rovnako užitočnými skúsenosťami zo života.

Použitá literatúra

- DOSTÁL, Petr. *Pokročilé metody analýz a modelování v podnikatelství a veřejné správě*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2008, 340 s. ISBN 978-80-7204-605-8.
- DOSTÁL, Petr. *Vybrané metody rozhodování v podnikové sféře: Chosen methods of decision-making in business sphere : zkrácená verze habilitační práce*. Brno: VUTIUM, 2005, 22 s. ISBN 80-214-3083-4.
- ELLISON GELTMAN, Katrina. *The Simulated Annealing Algorithm. Katrina Ellison Geltman* [online]. 2014-2-20 [cit. 2016-01-21]. Dostupné z: http://katrinaeg.com/simulated-annealing.html
- HANSELMAN, D. a B. LITTLEFIELD. Mastering MATLAB. Pearson Education International Ltd., 2012. 852 s. ISBN 978-0-13-185714-2.
- JÁGR, P. *Využiti prostředků uměle inteligence pro podporu rozhodovani v podniku*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2012. 127 s. Vedoucí diplomové práce prof. Ing. Petr Dostál, CSc.
- JIRSÍK, Václav a Petr HRÁČEK. *Neuronově sítě, expertní systémy a rozpoznávání řeči*. Brno, 2012. Skriptum. VUT v Brně. Vedoucí práce Ing. Václav Jirsík, CSc.
- JURČÍK, L. *Evoluční algoritmy při řešení problému obchodního cestujícího*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 79 s. Vedoucí diplomové práce prof. Ing. Petr Dostál, CSc.
- KENDALL G. and BURKE E.K. *Evaluation of Two Dimensional Bin Packing Problem using the No Fit Polygon. Proceedings of the 26th International Conference on Computers and Industrial Engineering*. Melbourne, Australia, 1999, pp 286-291
- KOPŘIVA, J.: *Srovnání algoritmů při řešení problému obchodního cestujícího*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2009, 77 s. Vedoucí diplomové práce doc. Ing. Petr Dostál, CSc.

- KRIŽAN, D.: *Strategická analýza spoločnosti Tesco stores ČR a.s.* Praha: Vysoká škola ekonomická v Praze, Fakulta podnikohospodářská, 2011, 94 s. Vedoucí diplomové práce doc. Ing. Helena Sedláčková CSc.
- MAŘÍK, V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence. Praha:ACADEMIA, 2013. 2473 s. 978-80-200-2276-9.
- ROSA, Š. Využití prostředků umělé inteligence pro podporu rozhodování v podniku. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2012. 88 s. Vedoucí diplomové práce prof. Ing. Petr Dostál, CSc.
- *Veřejný rejstřík a Sbírka listin - Ministerstvo spravedlnosti České republiky* [online]. 2015 [cit. 2016-01-21]. Dostupné z: https://or.justice.cz/ias/ui/rejstrik

Použité skratky

- **API** – Application Programming Interface
- **TSP** – Traveling Salesman problem
- **sTSP** – symetric TSP
- **aTSP** – asymetric TSP
- **mTSP** – multiple TSP
- **NP-complet** – Nondeterministic Polynomial & Nondeterministic Polynomial Hard
- **JSON** – JavaScript Object Notation
- **XML** – Extensible Markup Language
- **SWOT** – **S**trengths, **W**eaknesses, **O**pportunities and **T**hreats
- **GUI** – Graphical User Interface
- **GPS** – Global Positioning System

Zoznam príloh

- Google Static Maps API
- Google Maps Geocoding API
- Google Maps Distance Matrix API
- Google Maps Directions API
- Algoritmus na výpočet TSP pomocou Hopfieldovej neurónovej sieti
- Algoritmus na výpočet TSP pomocou simulovaného žíhaní

Prílohy

Google Static Maps API

```
% QUERY URL CONSTRUCTION EXAMPLE - BRNO STRED

preamble =
'https://maps.googleapis.com/maps/api/staticmap
?center=';
lat = '49.192465';
lon = '16.605168';
zoom = '&zoom=13';
size = '&size=800x800';
type = '&maptype=roadmap';
scale = '&scale=1'
markers = '&markers=size:mid%7Ccolor:red%7C';

url = [preamble lat ',' lon zoom size type
scale markers lat ',' lon]
waitfor(urlwrite(url,'map.jpg'));
imshow('map.jpg');
```

URL RESULT

Google Maps Geocoding API

```
text = 'brno, stred';
text = strrep(text,' ','+');
preamble =
'https://maps.googleapis.com/maps/api/geocode/json?address=';
url = [preamble text]

urlwrite(url,'geocode.txt');
a = textread('geocode.txt','%s','whitespace','\n');
k = strfind(a, 'location');
index = find(not(cellfun('isempty', k)));

lat = a{index(1)+1};
lat = sscanf(lat,'"lat" : %f',[1,inf])';
lon = a{index(1)+2};
lon = sscanf(lon,'"lng" : %f',[1,inf])';
```

RESULT:

```
>> lat

lat =

   49.1925

>> lon

lon =

   16.6052
```

Google Maps Distance Matrix API

```
preamble = 'https://maps.googleapis.com/maps/api/distancematrix/json?origins=';
destination = '&destinations=';
opt = '&mode=driving&sensor=false&language=en-EN&units=metric';

waypoint =  [49.177520,16.566122;49.155801,16.597555;49.157532,16.626835;
             49.191795,16.660315;49.195234,16.613926;49.189515,16.612934;
             49.193276,16.606979;49.217479,16.606918];

distance = [];
dis_text = [];
duration = [];
dur_text = [];
result = zeros(size(waypoint,1));
buffer = zeros(size(waypoint,1),2);
matrix = cell(size(waypoint,1)+1);

for i = 2:size(waypoint,1)+1
   matrix{1,i} = [waypoint(i-1,1),waypoint(i-1,2)];
   buffer(i-1,:) = [waypoint(i-1,1),waypoint(i-1,2)];
end
matrix(:,1) = matrix(1,:);
waypoint = buffer;

for i = 1:size(waypoint,1)-1
    from = [num2str(waypoint(i,1),'%f') ',' num2str(waypoint(i,2),'%f')];

    b = [];
    for k = (i+1):size(waypoint,1)
        b = [b num2str(waypoint(k,1),'%f') ',' num2str(waypoint(k,2),'%f') '|'];
    end
    b = b(1:end-1);

    url = [preamble from destination b opt]
    urlwrite(url,'matrix.txt');
    a = textread('matrix.txt','%s', 'whitespace', '\n');

    if logical(sum(not(cellfun('isempty', strfind(a, 'error_message')))))
        errordlg('Too many waypoints in the request. The maximum allowed
waypoints for this request is 8, plus the origin, and destination. "routes" :
[], "status" : "MAX_WAYPOINTS_EXCEEDED".','Error_message!');
        distance = 0;
        duration = 0;
        dis_text = 0;
        dur_text = 0;
        return;
    end

    l = strfind(a, 'value');
    t = strfind(a, 'text');
    index_k = find(not(cellfun('isempty', l)));
    index_t = find(not(cellfun('isempty', t)));

    for m = 1:2:size(index_k,1)-1
        a{index_k(m)}(1:10)=[];
        a{index_t(m)}(1:9)=[];
        a{index_k(m+1)}(1:10)=[];
        a{index_t(m+1)}(1:9)=[];

        distance = [distance str2double(a{index_k(m)})];
        duration = [duration str2double(a{index_k(m+1)})];
        dis_text = [dis_text regexp(a{index_t(m)}, '(?<=")[^"]+(?=")',
'match')];
        dur_text = [dur_text regexp(a{index_t(m+1)}, '(?<=")[^"]+(?=")',
'match')];
    end
    result(i,i+1:end) = distance;
    distance = [];
    dis_text = [];
    duration = [];
```

```
    dur_text = [];
end

result = result + result';
for i = 2:size(waypoint,1)+1
    for j = 2:size(waypoint,1)+1
        matrix{i,j} = result(i-1,j-1);
    end
end

matrix
```

```
matrix =

  Columns 1 through 8

             [] [1x2 double] [1x2 double] [1x2 double] [1x2 double] [1x2 double] [1x2 double] [1x2 double]
   [1x2 double] [         0] [      5052] [     11371] [     10152] [      6492] [      5951] [      6222]
   [1x2 double] [      5052] [         0] [      5352] [     11855] [      6111] [      5318] [      6207]
   [1x2 double] [     11371] [      5352] [         0] [     10261] [      6554] [      6221] [      9525]
   [1x2 double] [     10152] [     11855] [     10261] [         0] [      5429] [      5460] [      6772]
   [1x2 double] [      6492] [      6111] [      6554] [      5429] [         0] [      1286] [      2405]
   [1x2 double] [      5951] [      5318] [      6221] [      5460] [      1286] [         0] [      1655]
   [1x2 double] [      6222] [      6207] [      9525] [      6772] [      2405] [      1655] [         0]
   [1x2 double] [      7895] [     14678] [     17794] [      6657] [      3242] [      4548] [      3626]

  Column 9

   [1x2 double]
   [      7895]
   [     14678]
   [     17794]
   [      6657]
   [      3242]
   [      4548]
   [      3626]
   [         0]
```

Google Maps Directions API

```
preamble = 'https://maps.googleapis.com/maps/api/directions/json?origin=';
fromto = '49.177520,16.566122';
destination = '&destination=';
waypoints = '&waypoints=';
opt = '&mode=driving&sensor=false&language=en-EN&units=metric';

waypoint = [49.177520,16.566122;49.155801,16.597555;49.157532,16.626835;
            49.191795,16.660315;49.195234,16.613926;49.189515,16.612934;
            49.193276,16.606979;49.217479,16.606918;49.177520,16.566122];

b = [];
for i = 2:size(waypoint,1)-1
    b = [b num2str(waypoint(i,1),'%f') ',' num2str(waypoint(i,2),'%f') '|'];
end
b = b(1:end-1);

url = [preamble fromto destination fromto waypoints b opt]
urlwrite(url,'distance.txt');
a = textread('distance.txt','%s', 'whitespace', '\n');

k = strfind(a, 'value');
t = strfind(a, 'text');
w = strfind(a, 'via_waypoint');
index_k = find(not(cellfun('isempty', k)));
index_t = find(not(cellfun('isempty', t)));
index_w = find(not(cellfun('isempty', w)));

a{index_k(1)}(1:10)=[];
a{index_t(1)}(1:9)=[];
a{index_k(2)}(1:10)=[];

a{index_t(2)}(1:9)=[];
distance = str2double(a{index_k(1)});
duration = str2double(a{index_k(2)});
dis_text = regexp(a{index_t(1)}, '(?<=")[^"]+(?=")', 'match');
dur_text = regexp(a{index_t(2)}, '(?<=")[^"]+(?=")', 'match');

for k = 1:size(index_w,1)-1
    for i = 1:size(a,1)
        if index_w(k)>index_k(i)

        else
           row_k1 = index_k(i);
           row_k2 = index_k(i+1);
           row_t1 = index_t(i);
           row_t2 = index_t(i+1);

           a{row_k1}(1:10)=[];
           a{row_t1}(1:9)=[];
           a{row_k2}(1:10)=[];
           a{row_t2}(1:9)=[];
           distance(end+1) = str2double(a{row_k1});
           duration(end+1) = str2double(a{row_k2});
           dis_text(end+1) = regexp(a{row_t1}, '(?<=")[^"]+(?=")', 'match');
           dur_text(end+1) = regexp(a{row_t2}, '(?<=")[^"]+(?=")', 'match');
           break;
        end
    end
end

distance
dis_text
duration
dur_text
```

```
distance =

5051        5382       10261        5428        1286        1655       3627
7610

dis_text =

'5.1 km'    '5.4 km'    '10.3 km'    '5.4 km'    '1.3 km'    '1.7 km'    '3.6 km'    '7.6 km'

duration =

556   562   839   597   235   520   599   553

dur_text =

'9 mins'     '9 mins'     '14 mins'     '10 mins'     '4 mins'     '9 mins'     '10 mins'
'9 mins'
```

Algoritmus na výpočet TSP pomocou Hopfieldovej neurónovej sieti

```
bandiag = zeros(N);
for i=1:N-1;
    bandiag(i,i+1)=1;
    bandiag(i+1,i)=1;
end
    bandiag(1,N) = 1;
    bandiag(N,1) = 1;

noI = (ones(N) - eye(N));
matA = kron(noI,eye(N));
matB = kron(eye(N), noI);
matC = ones(N^2);
matD = kron(bandiag,matrix);

w = -A*matA-B*matB-C*matC-D*matD;
konec = false;
itera = 0;
struktura = {};

for m = 1:val10
hold off
po = 0
while ~konec
    V = rand(N);
    U = atanh(2*V-1)*u0;
    V = V(:);
    U = U(:);
    U_old = 100*U;
    t = 0;
    po = po + 1;
    while sum((U_old-U).^2)>1e-1&&t<2000
        itera = itera + 1;
        U_old = U;
        U=U+lamda*(w*V-U+C*N);
        V=(1+tanh(U/u0))/2;
        V(V<0.3) = 0;
        V(V>0.7) = 1;
        t = t+1;
    end
    V = reshape(V,[N N]);
    konec = all(sum(V,1)==1)&&all(sum(V,2))&&sum(V(:))==N;

    if po > 2000
        break;
    end
end

konec = 0;

k = zeros(1,N+1);
for i = 1:N
   k(i) = find(V(:,i));
end
k(end) = k(1);
out = k;

way = loc(out(:),:);
total = 0;

for i = 1:size(loc,1)-1
    total = total + sqrt((way(i,1)-way(i+1,1))^2+(way(i,2)-way(i+1,2))^2);
end
```

Algoritmus na výpočet TSP pomocou simulovaného žíhaní

```
NumCity = N;
distance = matrix;

count = 20;
all_dE = zeros(count, 1);
for i = 1:count
    path = randperm(NumCity);
    energy = sum(distance((path-1)*NumCity + [path(2:NumCity)
path(1)]));
    new_path = path;
    index = round(rand(2,1)*NumCity+.5);
    inversion_index = (min(index):max(index));
    new_path(inversion_index) = fliplr(path(inversion_index));
    all_dE(i) = abs(energy - ...
        sum(sum(diff(loc([new_path new_path(1)],:))'.^2)));
end
dE = max(all_dE);

temp = 10*dE;
fprintf('Initial energy = %f\n\n',energy);

out = [path path(1)];
hold on
plot(loc(out(:), 1), loc(out(:), 2),'r.', 'Markersize', 20);
h = plot(loc(out(:), 1), loc(out(:), 2),'LineWidth',2); hold off

MaxTrialN = NumCity*100;
MaxAcceptN = NumCity*10;
StopTolerance = 0.005;
TempRatio = 0.5;
minE = inf;
maxE = -1;

itera = 0;
while (maxE - minE)/maxE > StopTolerance,
    minE = inf;
    maxE = 0;
    TrialN = 0;
    AcceptN = 0;
    while TrialN < MaxTrialN & AcceptN < MaxAcceptN,
        new_path = path;
        index = round(rand(2,1)*NumCity+.5);
        inversion_index = (min(index):max(index));
        new_path(inversion_index) = fliplr(path(inversion_index));
        new_energy = sum(distance( ...
            (new_path-1)*NumCity+[new_path(2:NumCity)new_path(1)]));
        if rand < exp((energy - new_energy)/temp)
            energy = new_energy;
            path = new_path;
            minE = min(minE, energy);
            maxE = max(maxE, energy);
            AcceptN = AcceptN + 1;
        end
        TrialN = TrialN + 1;
    end
```

```
    out = [path path(1)];
    set(h, 'xdata', loc(out(:), 1), 'ydata',
            loc(out(:),2),'LineWidth',2);
    drawnow;
end
```