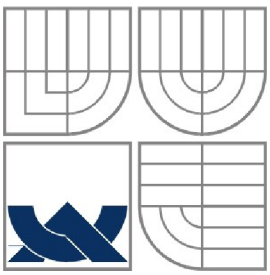
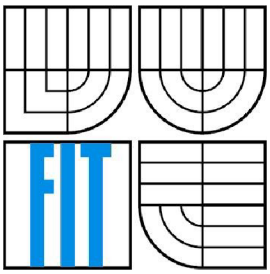


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EFEKTIVNÍ LOGISTIKA DOPRAVY VOZIDEL

EFECTIVE LOGISTIC OF TRANSPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB JANOVIČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN LUKÁŠ, Ph.D.

BRNO 2008

Zadanie:

1. Seznamte se s jazyky a prostředky pro tvorbu webových informačních systémů (XHTML, CSS, PHP, Javascript, MySQL).
2. Proveďte obecně analýzu požadavků pro informační dopravní logistiky vozidel.
3. Navrhněte vhodnou strukturu systému. Při analýze požadavků návrhu využijte vhodných technik.
4. Vyberte vhodnou metodu z umělé inteligence a pomocí této metody naimplementuje podle zadaných požadavků efektivní plánovač tras pro jednotlivá vozidla.
5. Daný systém implementujte tak, aby byl použitelný v praxi.
6. Zhodnoťte dosažené výsledky, porovnejte váš systém s existujícími systémy, navrhněte další možné rozšíření do budoucna.

Licenčná zmluva

Originál licenčnej zmluvy je uložený v archíve Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Táto práca sa zaoberá vyhľadávaním najkratšej cesty na základe vstupných podmienok ako sú počet vozidiel a vstupná mapa. Pojednáva o algoritmoch používaných na vyhľadávanie a ich konkrétne nasadenie v aplikácii, prípadne ich úprava pre danú problematiku. V práci je zaznamenaná celá štruktúra aplikácie ako spomínané vyhľadávanie, tak správa systému a správa objektov používaných v programe.

Kľúčové slová

logistika, umelá inteligencia, hľadanie trasy, doprava, webová aplikácia, cesta

Abstract

This thesis considers with searching the shortest route depends on input conditions as number of vehicles and input map. Presents algorithms used for searching, their use in applications and their adjustment for existing problems. In this thesis is included whole scructure of the application as mentioned searching, its administration and administration of objects used in program.

Keywords

logistic, artificial intelligence, route searching, transportation, web application, route

Citácia

Janovič Jakub: Efektívna logistika v doprave vozidiel. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Efektívna logistika dopravy vozidiel

Vyhlásenie

Vyhlasujem, že som túto bakalársku prácu vypracoval samostatne, pod vedením Ing. Romana Lukáša, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Jakub Janovič
12.5.2008

Poďakovanie

Ďakujem svojmu vedúcemu, Ing. Romanovi Lukášovi, Ph.D. za rady, ktoré mi poskytol a za čas, ktorý mi venoval. Taktiež ďakujem svojej rodine a priateľom, ktorí mi pomáhali na ceste až sem, k dokončeniu práce.

Obsah

Úvod

Už od príchodu prvých počítačov sa ľudia tieto stroje snažili stále zlepšovať. Prvé procesory zvládali prepočítavať len jednoduché úlohy, ale postupným vývojom sa nasadzovali na stále náročnejšie výpočty. Ľudia sa vždy snažia nejaký úkon zjednodušiť, automatizovať. Preto je tento vývoj nevyhnutný. Doba sa posúva stále dopredu, a tieto možnosti nie sú niekedy postačujúce, z toho dôvodu aj ľudia sa snažia výpočty urýchliť a to vynaliezaním stále rýchlejších algoritmov. Rozdelenie problémov na pod-problémy a ich jednotlivé riešenie a následné spájanie. Momentálne dokážeme riešiť rôzne špecifické úlohy ako modelovanie rozličných situácií, správa výrobných procesov, ale aj jednoduchšie aplikácie napríklad účtovníctvo, informačné systémy, atď.

Medzi najnáročnejšie problémy na výpočty sú aplikácie založené na modelovaní rôznych situácii pretože je nutné prechádzať veľké množstvo pozitívnych možností a výber najlepších z nich. Využívajú sa metódy z oblasti umelej inteligencie, simulácií. Jedným z týchto problémov je takzvaný problém obchodného cestujúceho. Jeho základom je prejsť všetky zadané mesta tou najkratšou cestou, pričom každé mesto navštívime len raz, a vrátime sa do pôvodného. Tento problém je jednoduchý pokiaľ sa jedná o nízky počet miest. Vtedy je výpočet pomerne krátky a jednoduchý. Ale komplikácie nastávajú pri väčších počtoch miest, pretože narastá počet rôznych ciest medzi nimi a výber najkratšej je náročný. Preto je potrebné nájsť algoritmus, ktorý aj pri zvýšenom počte miest nepotrebuje toľko nárokov, ako keby sme hľadanie realizovali hrubou silou.

V tejto práci bude problém obchodného cestujúceho nasadený ako jeden z viacerých problémov, pretože zadanie je odlišné. Bude treba vytvoriť si mapu miest a ciest medzi nimi a tieto sa budú následne vyberať do aplikácie a samotného výpočtu. Jednoducho povedané: máme mapu miest a potrebujeme obehnúť so svojim vozidlom len polovicu, nerovnomerne rozdelenú a mám k dispozícii *x* vozidiel. Postup pri hľadaní je taký, že celú hľadanú oblasť rozdelíme na zhluky miest, do ktorých chceme, aby sa vozidlá dostali. Každý jeden zhluk bude predstavovať jedno zo zadaných vozidiel. V tomto zhluku budeme prechádzať jednotlivé mestá algoritmom použitým v probléme obchodného cestujúceho. Toto nám vytvorí x zhlukov s x najkratších ciest. Tieto cesty sú ale vzdušné trasy, preto nastáva problém, že sa musíme držať na cestách, ktoré boli v mape vytvorené. Z tohto dôvodu bude potrebný aj algoritmus hľadania najkratšej možnej trasy po cestách medzi bodmi. Týmto sú vytvorené zhluky a cesta cez ne. Ďalej je potrebné zakomponovať východiskové mesto, odkiaľ budú vozidlá vychádzať. Takto sme teoreticky docielili celkový chod aplikácie a teoreticky našli požadovaný výsledok.

Ďalej bude nutné vytvoriť databázu vozidiel pre jednoduchšiu správu vozidiel. Tá bude súčasťou komplexnejšieho informačného systému, aby táto aplikácia mohla byť využívaná v reálnom nasadení. Celý problém je orientovaný na rýchly a spoľahlivý chod a čo najvšeobecnejšie pojatie.

Čiže chceme docieliť toho, aby bolo možné aplikáciu využívať v rôznych odvetviach., či už vo veľkých dopravných podnikoch alebo pre malú spoločnosť zaoberajúcu sa roznáškou novín.

Toto je len teoretický úvod, nasledujúce kapitoly sa budú venovať konkrétnym nasadeniam rozličných algoritmov použitých na docielenie tejto úlohy. Úvodná kapitola sa venuje všeobecným znalostiam algoritmov, ktoré budú následne použité. V 2. kapitole bude popísaný kompletný návrh celého systému s podmienkami úpravy algoritmov z 1. kapitoly. Nasleduje implementácia systému, prostriedky, ktoré boli využité a postupy, ktorými dosiahneme požadovaný cieľ. Predposledná kapitola zahŕňa výsledky, ktoré sme dosiahli, prípadné návrhy na zlepšenie. Záverom je zhodnotená celá práca.

1 Metódy a algoritmy

V tejto kapitole budú spomenuté jednotlivé metódy a algoritmy, ktoré sa využívajú na problémy ako sú problém obchodného cestujúceho, rozdelenie grafu do zhlukov, hľadanie najkratšej cesty medzi dvomi mestami.

Na riešenie týchto problémov boli dôkladne vyberané algoritmy, ktorých popis bol dostatočne popísaný v uvedených literatúrach. V následujúcich podkapitolách budú tieto algoritmy popísané a naznačený postup ich realizácie v oblasti informačných technológií.

1.1 K-means

Algoritmus k-means patrí do skupiny tzv. učenie bez učiteľa. Ide o hľadanie podobností medzi príkladmi trénovacej množiny a v klasifikácií príkladov s podobnými charakteristikami do skupín. Tieto skupiny nazývame zhluky. Umelý systém pritom nedostáva žiadnu informáciu o správnosti klasifikácie, preto ani o priebehu samotného učenia – jediná informácia, ktorú je možné zistiť je počet uzlov v zhluku.

Príklady trénovacej množiny sú prakticky vždy predstavované číselnými vektormi príznakov klasifikovaných dejov či objektov. Metódy určenia bez učiteľa sú následne založené na predpoklade, že tieto vektory, resp. ich koncové body tvoria v príslušnom n-rozmernom priestore zhluky.

Tieto zhluky môžu ale predstavovať rôzne tvary. Pri väčšom počte bodov ide o jednoznačné určenie a po znázornení na grafe sú vidieť zhluky jasne, ale pri menšom počte vektorov môže dôjsť k značnému rozptýleniu.

Ako najznámejšia a najefektívnejšia metóda na toto určenie sa používa algoritmus k-means clustering. Clustering je odvodené od slova cluster – zhluk, chumáč.

Algoritmus spočíva na tom, že do celej oblasti vektorov zasadí *k* vektorov a tie sa správajú ako centrum zhluku. Následne si všetky vektory z oblasti hľadajú najbližší stred zhluku, po nájdení sa k tomuto zhluku zaradí. Po priradení centier k všetkým vektorom sa centrum zhluku znovu prepočíta na základe vektorov, ktoré sú k nemu priradené. Tento postup sa opakuje dovtedy pokiaľ sa centrá zhlukov neposúvajú alebo posúvajú len po prípustnú hodnotu, prípadne vektory nemenia svoje priradenie v opakovaniach (viac viď. [1]).

1.1.1 Algoritmus k-means clustering

1. Inicializuj *k* prototypov – náhodne vybraných v oblasti trénovacích vektorov.

$$\vec{w}_j = \vec{x}_p,\ j \in \langle 1, k \rangle,\ p \in \langle 1, P \rangle$$

2. Každý vektor $\vec{x}_p$ z trénovacej množiny priraď do zhluku *Cj*, *j*∈ ⟨1, *k*⟩, ktorého prototyp $\vec{w}_j$ má od vektoru $\vec{x}_p$ najmenšiu vzdialenosť:

$$|\vec{x}_p - \vec{w}_j| \leq |\vec{x}_p - \vec{w}_i| \qquad i, j \in \langle 1, k \rangle$$

3. Pre každý zhluk *Cj* , *j*∈ ⟨1, *k*⟩ prepočítaj prototyp tak, aby bol ťažiskom koncových bodov všetkých vektorov, ktoré sú k tomuto zhluku práve priradené (nech n_j je počet týchto vektorov):

$$\vec{w}_j = \frac{\sum_{\vec{x}_j \in C_j} \vec{x}_i}{n_j}$$

4. Vypočítaj „chybu“ aktuálneho stavu zhlukovania (súčet chýb všetkých zhlukov, ktoré sú dané súčtami štvorcov vzdialeností všetkých vektorov jednotlivých zhlukov od stredu týchto zhlukov):

$$\sum_{j=1}^{k} \sum_{\vec{x}_i \in C_j} |\vec{x}_i - \vec{w}_j|^2$$

5. Pokiaľ „chyba“ *E* klesla, alebo pokiaľ bol niektorý vektor priradený k inému zhluku, skoč na bod 2.

Obrázok 1.1 znázorňuje postup algoritmu k-means. Farebne sú rozdelené jednotlivé zhluky.

1.2 A*

Metóda A* (čítaj A s hviezdičkou, A star) je najznámejšou a najpoužívanejšou v oblasti riešenia úloh prehľadávaním stavového priestoru. Hľadá pomocou heureistiky a skúša vždy najslubnejší neprebádaný stav.

Pre každý navštívený vrchol *x* sa spočíta ohodnotenie najlepšej cesty *f(x)*, ktorá cez neho vedie podľa vzorca:

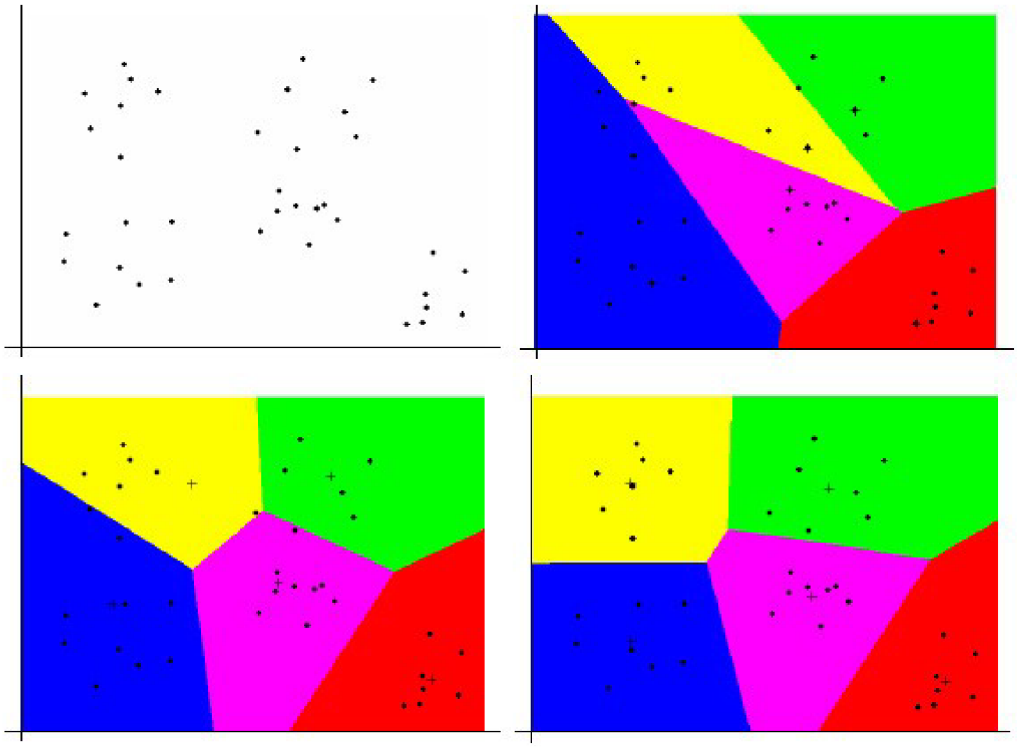
$$f(x) = g(x) + h(x)$$

Kde *g(x)* je súčet ohodnotenia najkratšej cesty zo štartu cez doposiaľ navštívené vrcholy do *x* a *h(x)* je odhad dĺžky cesty z *x* do cieľového stavu.

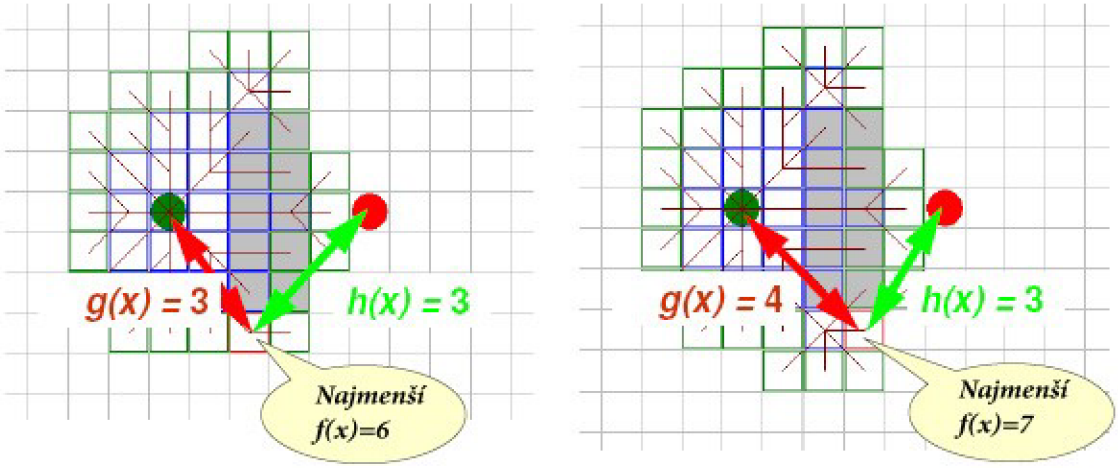
$$h(g) = 0 \quad \text{pre cieľový stav}$$

$$h(x) \geq 0 \quad \text{pre každý stav}$$

Na obrázku 1.2 je možné vidieť hodnoty *g(x)* a *h(x)* počas prehľadávania stavového priestoru. A zmena *f(x)* počas tohoto prehľadávania.



Obr. 1.1 Príklad postupu algoritmu k-means



Obr. 1.2 Príklad postupu algoritmu A*

V algoritme A* sa používajú dve dátové štruktúry a to open a closed. V štruktúre open (ďalej len OPEN) sa nachádza zoznam navštívených vrcholov, ktoré je potreba prebrať a štruktúra closed (ďalej len CLOSED) obsahuje už prejdené vrcholy. (Viac viď. [4])

1.2.1 Algoritmus A* teoreticky

1. Počiatočný bod vložíme do OPEN
2. Opakujeme následujúce:
   - Nájdeme bod s najmenším f(x) ohodnotením z OPEN, a poznačíme ho ako aktuálny bod
   - Vyberieme ho z OPEN a vložíme do CLOSED
   - Pre každé okolité body preveríme následujúce:
     - pokiaľ nie je priechodný alebo sa nachádza v CLOSED ignorujeme ho
     - Pokiaľ sa nenachádza v OPEN vložíme ho do OPEN a aktuálny bod zaznamenáme u vkladaného ako rodičovský a zaznamenáme hodnoty h(x) a f(x)
     - Pokiaľ sa už v OPEN nachádza, pozrieme či táto trasa k tomuto bodu je lepšia. Použijeme na zistenie hodnotu g(x) – bod s nižšou hodnotou g(x) znamená lepšiu cestu. Pokiaľ dôjdeme na lepšiu cestu rodiča tohoto bodu zmeníme na aktuálny a prepočítame hodnoty g(x)
   - zastavíme pokiaľ:
     - Vložíme cieľový bod do CLOSED, v tomto prípade bola trasa úspešne nájdená
     - Cesta sa nenájde, a OPEN je prázdny, v tomto prípade neexistuje cesta medzi danými dvomi bodmi
3. Zistíme si celú trasu a to tak, že prechádzame CLOSED od cieľového bodu a zisťujeme jeho rodiča. Týmto spôsobom sa dostaneme až na počiatočný bod. Otočením tohoto poľa získame hľadanú trasu. Podrobnejší popis na [5].

1.2.2 Algoritmus A* v pseudokóde

```
function A*(start,goal)
    var closed := the empty set
    var q := make_queue(path(start))
    while q is not empty
        var p := remove_first(q)
        var x := the last node of p
        if x in closed
            continue
        if x = goal
            return p
        add x to closed
        foreach y in successors(x)
            enqueue(q, p, y)
    return failure
```

1.3 Problém obchodného cestujúceho

Problém obchodného cestujúceho (Ďalej aj ako TSP – Travelling salesman problem) je obtiažny diskrétny optimalizačný problém, matematicky vyjadrujúci a zovšeobecňujúci úlohu nájdenia najkratšej možnej cesty prechádzajúce všetkými zadanými bodmi na mape.

Ide o takzvanú NP-úplnú úlohu. To znamená, že vo všeobecnom prípade nie je známe ani ako nájsť presné riešenie v rozumnom čase a dokonca ani či vôbec môže existovať algoritmus, ktorý takéto riešenie nájde v čase úmernom nejakej mocnine počtu uzlov.

Je možné nájsť množinu všetkých možných ciest a určiť tú najkratšiu, ale pri zvýšenom počte miest dochádza až k neriešiteľnosti príkladu, nakoľko počet kombinácií je natoľko vysoký, že čas, ktorý by sa venoval výpočtu je neefektívny a nie je možné túto metódu zasadiť do bežných aplikácií modelujúcich nachádzanie trás v reálnom živote.

Zlom v riešení problému obchodného cestujúceho prišiel v roku 1954 keď George Dantzig, Ray Fulkerson, and Selmer Johnson publikovali metódy na riešenie tohoto problému na mape s 49 mestami.

1.3.1 Rekordy v hľadaní najkratšej trasy

V tejto oblasti bolo vykonaných viacero jednotne zameraných hľadaní. Tieto výpočty trvali dlhší čas a boli uskutočnené na výskumné účely [6].

Nemecko

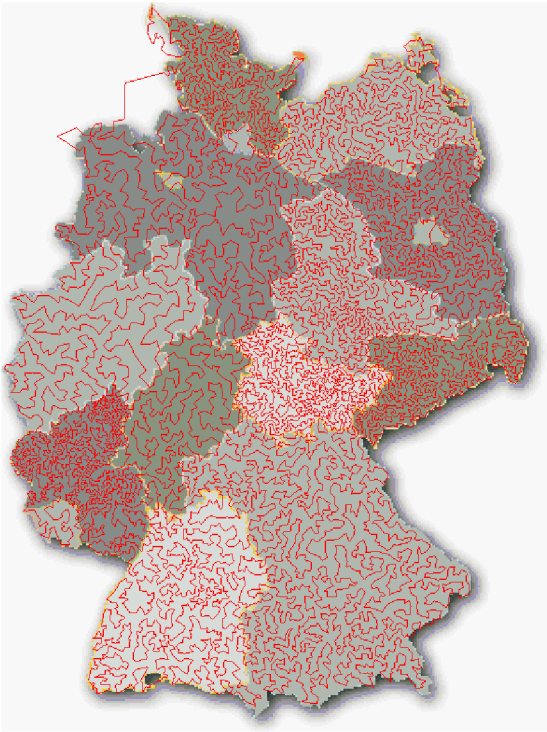
V apríli 2001 David Applegate, Robert Bixby, Vašek Chvátal a William Cook zrealizovali vyhľadávanie najkratšej cesty po 15 112 mestách v Nemecku. Optimálna cesta mala dĺžku asi 66 000 kilometrov.

Výpočet bol uskutočnený na sieti 110 procesorov umiestnených na Rice University a Princeton University. Doba výpočtu bola 22,6 CPU rokov realizovaná na Compaq EV6 Alpha procesoroch taktovaných na 500 MHz

Švédsko

V máji 2004 bol vyriešený problém obchodného cestujúceho s navštívením 24 978 miest. Cesta merala 855 597 TSPLIB [7] jednotiek (asi 72 500 km) a bolo dokázané, že kratšia neexistuje. Toto je momentálne najrozsiahlejšie zisťovanie po hľadaní trasy v Nemecku.

Výskum bol realizovaný na 96 dual core procesoroch Intel Xeon 2.8GHz v Georgia Tech School of Industrial and systém Enginneering, na pozadí zatiaľ, čo boli servery v normálnej činnosti.



Obr. 1.3 Obrázok zobrazujúci výsledok hľadania trasy v Nemecku

Finálny CPU čas bol 222 796 246 CPU sekúnd, čo je asi 7,1 CPU rokov po prevode. Tento výzkum bol zrealizovaný tímom pracovníkov ako z komerčnej tak univerzitnej oblasti.

- David Applegate, AT&T Labs
- Robert Bixby, ILOG and Rice University
- Vašek Chvátal, Rutgers University
- William Cook, Georgia tech
- Keld Helsgaun, Roskilde University

Výskum bol podporovaný a realizovaný na základe Office of Naval Research grantu a National Science Foundation Grant.

Optimálna cesta s 85 900 mestami

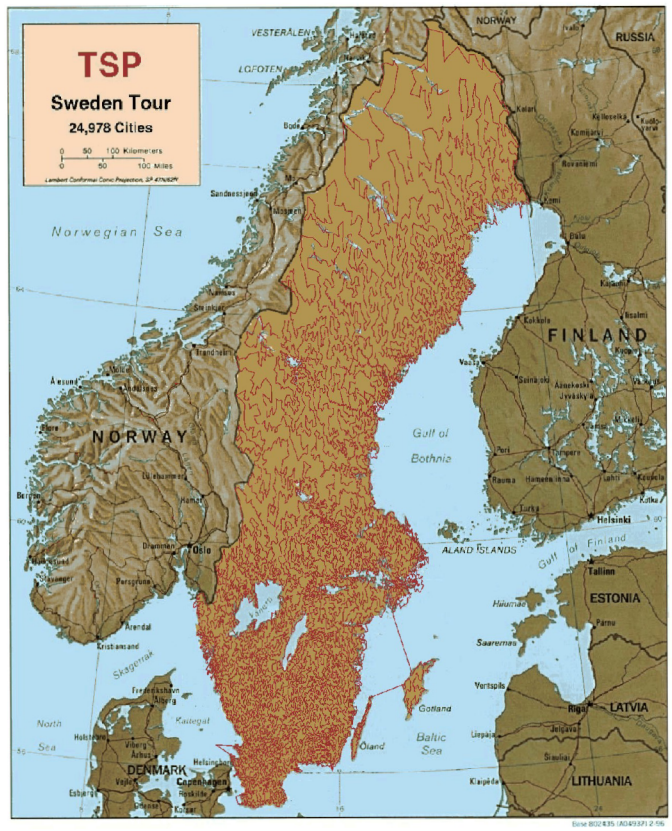
Najväčším doposiaľ realizovaným výpočtom bolo hľadanie optimálnej cesty cez 85 900 miest. Tentokrát ale nebol použitý reálny model, ale model z VLSI aplikácie vyvinutej v Bell Laboratories v 80-tých rokoch 20. storočia.

Výpočet našiel najoptimálnejšiu trasu, ktorá merala 142 382 641 TSPLIB jednotiek pomocou takzvaného Conocorde code [8].

Výsledok bol získaný v rokoch 2005, 2006 týmito pracovníkmi:

- David Applegate, AT&T Labs
- Robert Bixby, ILOG and Rice University

- Vašek Chvátal, Rutgers University
- William Cook, Georgia tech
- Daniel Espinoza, University of Chile
- Marcos Goycoolea, Universidad Adolfo Ibanez
- Keld Helsgaun, Roskilde University



Obr. 1.4 Obrázok zobrazujúci výsledok hľadania trasy vo Švédsku

2 Návrh systému

Bude nutné navrhnúť aplikáciu tak, aby splňovala viacero podmienok. Medzi hlavné určite patrí použiteľnosť. Aplikácia je použiteľná na viacero odvetví, preto sa nezameriava len na jednu konkrétnu problematiku. Ďalším prvkom je to, že aplikáciu bude možné jednoducho konkretizovať na samotné odvetvie tým, že sa prípadne pridajú nejaké moduly, ktoré by boli neskôr doimplementované. Nechceme vytvárať aplikácie zamerané len jedným smerom, čiže bude zameraná na jadro, čím je vyhľadávanie najkratšej trasy pre viacero vozidiel, ale budú tu obsiahnuté aj iné moduly, ako je správa vozidiel, správa zamestnancov. Ich práva na prístup do systému.

Ďalšou podmienkou je jednoduchosť. Systém je realizovaný tak, aby ho bol schopný používať ako skúsený odborník tak aj laik, preto je aplikácia zameraná najmä na jednoduchosť užívateľského rozhrania s tým, že aj neznalá osoba intuitívne dôjde k spôsobu jej ovládania. Dôležité je graficky reprezentovať vstupné a výstupné dáta. Bolo by možné túto aplikáciu spracovať aj v textovom režime, ale tým by táto podmienka nebola splnená. Preto budú použité rôzne grafické prvky ako je tvorba mapy, správa mapy, atď. Tá bude realizovaná „klikateľným“ prostredím na obrazovke a „vyskakovacími“ tabuľkami na správu aplikácie.

Inou nevyhnutnou podmienkou je rýchlosť aplikácie. V predchádzajúcej kapitole boli zmieňované algoritmy, ktorými sa budú trasy vyhľadávať. Tieto algoritmy sú pripravené na náročné výpočty, ale čím náročnejší výpočet je realizovaný, tým sa zvyšuje aj doba výpočtu, preto nebude systém až tak zameraný na presnosť, aj keď to je hlavný cieľ, ale aj na rýchlosť. Bude hľadaná cesta najrozumnejším smerom, a keď algoritmus zistí, že je dostatočne presný, výpočet sa zastaví. Výsledkom bude aktuálny stav výpočtu. Keby sme chceli dosiahnuť sto percentnú presnosť, mohlo by dôjsť k výpočtom trvajúcim minúty, hodiny až dni.

Spôsob na zrýchlenie výpočtu je použiť výkonnejší hardware, ale to tiež nie je cieľom. Aplikácia sa musí držať použiteľnosti v reálnom nasadení, preto nie je možné, aby bola schopná počítať len na tých najvýkonnejších serveroch.

Užívateľskou pomôckou je aj možnosť skrytia hlavného banneru stránky. Tým sa zakryje pri práci s aplikáciou nepodstatná časť rozhrania. Táto časť je ale nezbytná kvôli celkovému grafickému dojmu. Z hľadiska užívateľského rozhrania je dôležité, že celá aplikácia sa nemôže meniť pri zobrazovaní na rozličných platformách. Dôraz je kladený na jednotnosť prostredia za akýchkoľvek prostrediach. Nevyhnutnou je ale minimálna šírka obrazovky, ktorá musí korespondovať so šírkou mapy.

2.1 Správa mapy

Nezbytnou súčasťou je správa mapy, ktorá nám v jednoduchom grafickom prostredí umožní spravovať mestá a trasy – či už vytvárať mazať alebo editovať. Dôraz je kladený na jednoduchosť a intuitívnosť. Sú dané presné súradnice o veľkosti 850 x 448 a tu je možné mapu spravovať. Ako pomôcku je možné vložiť si obrázok na pozadie mapy, aby sa dala reprezentovať nejaká reálna oblasť a užívateľ nemusí vytvárať mapu „od oka".

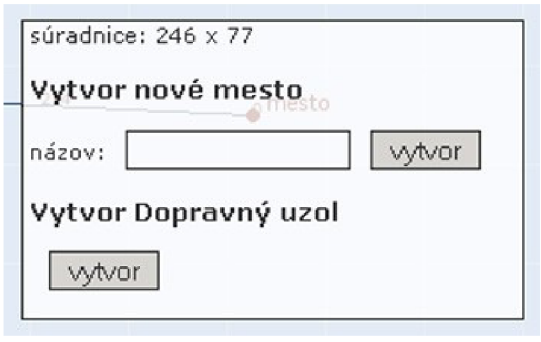
2.1.1 Tvorba mesta

Nové mesto sa na mape vytvára jednoduchým kliknutím na plochu mapy. Následne sa zjaví menu s kliknutými súradnicami s dotazom čo má užívateľ záujem vytvoriť ako je viditeľné na obrázku 2.1. Vpíše názov mesta a po potvrdení sa mu mesto vytvorí. Zmena sa prejaví okamžite, a vytvorené nové mesto vidí na mape.

2.1.2 Tvorba dopravného uzlu

Dopravný uzol slúži na prirodzenejšiu tvorbu celej mapy a ciest medzi nimi. Ide o miesto kde nie je zreteľná cesta priamo medzi dvomi mestami, ale je treba spraviť akúsi križovatku, alebo niekde v mieste cesty sa nachádza odbočka k inému mestu alebo dopravnému uzlu.

Po kliknutí na mapu, kde sa nenachádza žiadne mesto, alebo uzol sa zobrazí užívateľovi menu s dotazom čo chce vytvoriť obrázok 2.1. Je tam spomínaná položka pre tvorbu mesta, alebo tlačítko pre tvorbu dopravného uzlu. Po potvrdení sa mapa znovu nahrá aj so zobrazeným novo-vytvoreným dopravným uzlom.



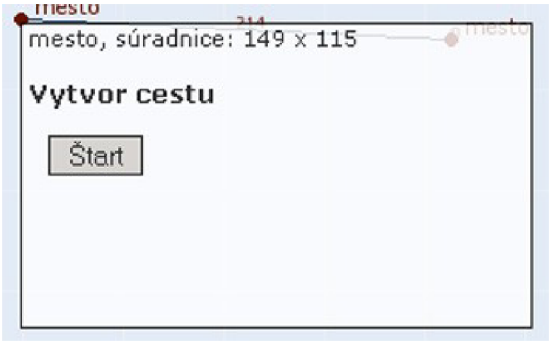
Obr. 2.1 Vytváranie miest a dopravných uzlov

2.1.3 Tvorba ciest

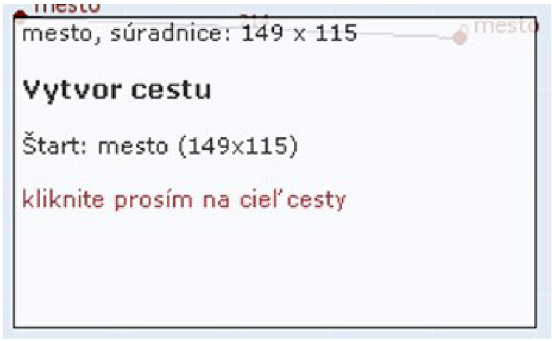
Cesty spájajú jednotlivé mestá alebo dopravné uzly a tým vytvárajú dopravnú sieť na mape. V aplikácii je možné hodnotu trasy ponechať na samotnom výpočte vzdialenosti miest a uzlov, alebo manuálne zadať. Toto je z dôvodu, že nie všetky cesty v reálnom prostredí sú jednotné. Niektoré cesty sú napr. veľmi kľukaté, keďže je nutné prekonať nerovnosti prostredia. Túto hodnotu užívateľ

môže využiť aj ak chce si cesty ohodnotiť podľa toho aká je na reálnej ceste povolená rýchlosť, prípadne ako rýchlo sa môže na tejto ceste pohybovať vozidlo. Pre rýchlejšie komunikácie využije nižšie ohodnotenie cesty, pretože v reálnom prostredí sa môže na nej pohybovať rýchlejšie. Naopak pre komunikácie, ktoré sú náročnejšie ako napríklad poľné cesty sa využije vyššie ohodnotenie cesty.

Cesta sa vytvára kliknutím na mesto alebo dopravný uzol. Následne je užívateľovi ponúknutá možnosť vytvoriť komunikáciu, ktorá prechádza týmto mestom. Kliknutím na iné mesto, alebo dopravný uzol sa ukáže ponuka na spojenie týchto bodov aj so spomínaným určením hodnoty trasy. Postup tvorby cesty je vidno na obrázkoch 2.2.1, 2.2.2 a 2.2.3. Na obrázkoch si možno takisto všimnúť, že okienko ukazuje rohom na kliknuté mesto.



Obr. 2.2.1 Tvorba cesty: 1. krok



Obr. 2.2.2 Tvorba cesty: 2. krok

Obr. 2.2.3 Tvorba cesty: 3. krok

2.1.4 Vloženie vlastnej mapy

V ponuke sa nachádza aj možnosť vloženia si vlastnej priehľadnej mapy, ktorá slúži na jednoduchšiu správu mapy. Vložením obrázka sa vytvorí priehľadná vrstva prekrývajúca originálnu mapu. Ďalej celá správa mapy funguje ako bolo popísané vyššie.

2.2 Návrh trasy

Táto časť sa venuje jadru celej aplikácie a to je vyhľadávanie najkratších ciest. Je tu možné vybrať konkrétne mestá, ktorými je treba prejsť. Ďalšou vstupnou podmienkou je počet vozidiel, ktoré do tejto oblasti vyjedú.

Vstupné údaje sa odošlú a sú generované najoptimálnejšie trasy. Počet trás je identický s počtom vozidiel. Do oblasti, do ktorej majú vozidlá ísť je generované pomocou algoritmu k-means. V týchto zhlukoch sa vytvárajú samostatné cesty. Pre každý zhluk sa hľadá práve jedna cesta pomocou algoritmu založeného na probléme obchodného cestujúceho.

Týmto sa ale vytvoria len vzdušné najoptimálnejšie trasy, preto je nevyhnutné ich konvertovať na trasy po existujúcich cestných komunikáciach. To je realizované na základe algoritmu A*.

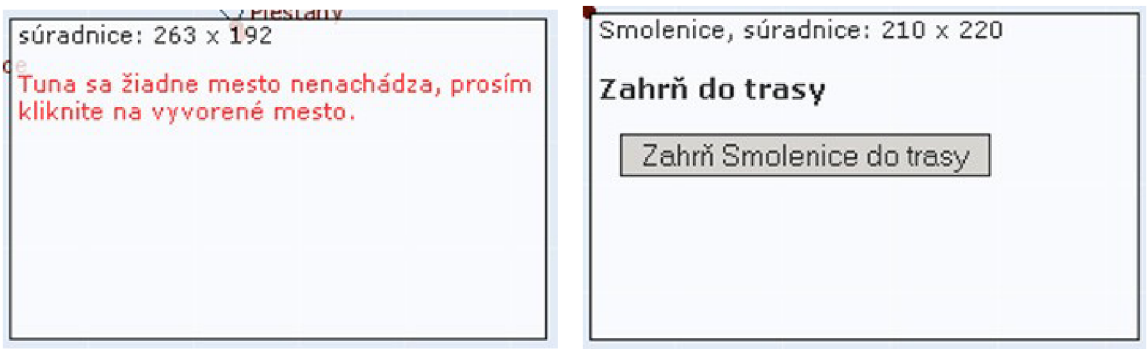
Postup:

- užívateľ zadá mestá, ktoré žiada, aby boli prejdené
- tieto mestá sa rozdelia do zhlukov
- v rámci každého zhluku sa nájde najkratšia vzdušná cesta
- prechádzaním miest vzdušnej cesty sa prechádzajú aj samotné mestá po reálnych komunikáciach

2.2.1 Výber miest

Výber miest prebieha podobne ako pri správe mapy. Po kliknutí na mapu sa zobrazí okno s ponukou. Pokiaľ bolo kliknuté inde ako na mesto, aplikácia vyzve užívateľa, aby klikol na miesto kde leží mesto (obrázok 2.3.1). Naopak pokiaľ sa klikne na mesto, v okienku sa objaví možnosť pridať toto mesto do návrhu trasy (obrázok 2.3.2)..

Po pridaní je zoznam pridaných miest vidieť na spodnej časti stránky. Spolu s ním sa tu nachádza aj zoznam dostupných vozidiel. Tie si užívateľ vyberie a po navrhnutej trase odošle požiadavok tlačítkom *Pokračuj* (obrázok 2.4).



Obr. 2.3.1 a 2.3.2 Menu po kliknutí na mapu, kliknutie na miesto kde mesto nie je a kde je

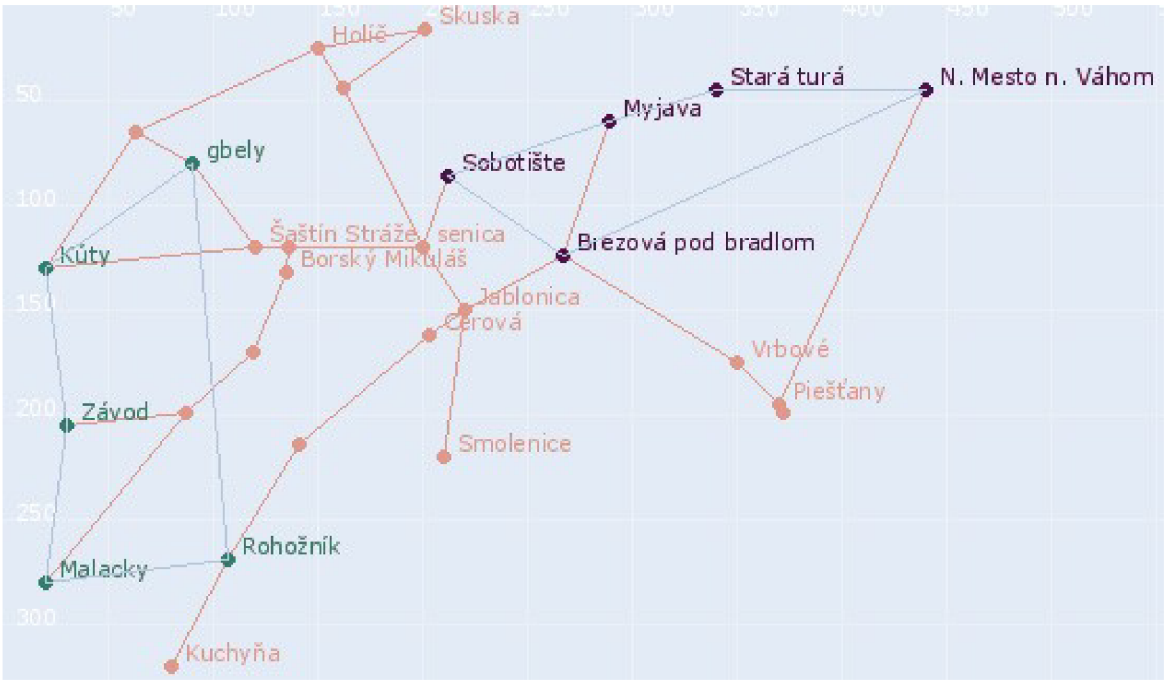
Obr. 2.4 Výber vozidiel pre trasu.

2.2.2 Rozdelenie miest do zhlukov a TSP

Tieto 2 postupy sa udejú v jednom kroku. To je z toho dôvodu, aby užívateľ nemusel prechádzať väčším množstvom krokov k dosiahnutiu cieľa. Najprv sa vygenerujú zhluky metódou k-means a potom postupne pre každý zhluk sa vytvára vlastná vzdušná najoptimálnejšia cesta pomocou metód TSP.

Toto sa prejaví graficky na mape (obrázok 2.5), a je možné si už v tomto kroku vytvoriť približnú predstavu výsledku. Tento krok je aj najnáročnejší z pohľadu výpočtu, najmä preto, že sa vykonávajú výpočty spojené s problémom obchodného cestujúceho.

Ružovou farbou sú na obrázku znázornené všetky mestá a spoje medzi nimi existujúce. Dvomi tmavšími farbami je vidieť rozdelenie požadovaných miest do zhlukov. Bledo-modrou sú znázornené vzdušné trasy medzi nimi generované na základe TSP.



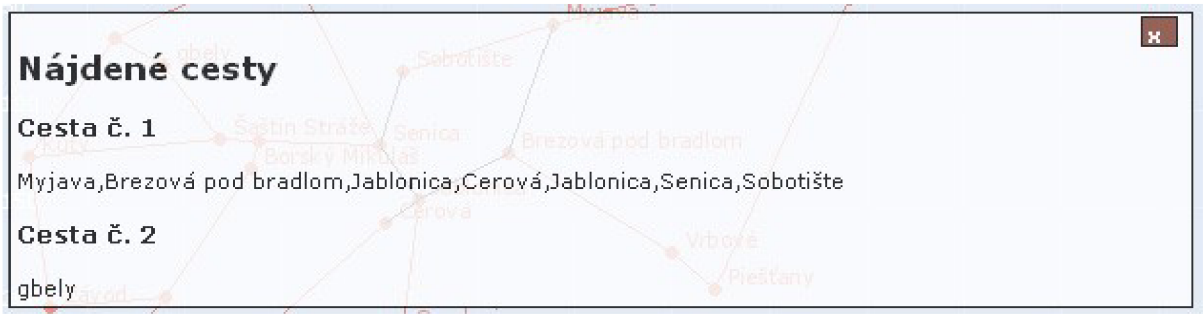
Obr. 2.5 Zobrazenie mapy po druhom kroku generovania najoptimálnejších trás

2.2.3 Vyhľadanie najkratšej trasy

Toto je posledný krok pri vyhľadávaní najoptimálnejších (najkratších) trás. V tomto kroku je použitý algoritmus A*. V každom zhluku je po trase vytvorenej TSP tento algoritmus aplikovaný, a generujú sa tak výsledné cesty.

Tieto sú opäť znázornené graficky (obrázok 2.6), ale je možnosť si pozrieť výsledok aj v textovej podobe (obrázok 2.7).

Obr. 2.6 Výsledok znázornený na mape



Obr. 2.7 Výsledok zobrazený v textovej podobe

2.3 Správa vozidiel

Užívateľ má možnosť jednoduchým a intuitívnym spôsobom spravovať vozidlá v aplikacii. Môže vozidlá pridávať, editovať mazať čo sú štandardné úkony v informačných systémoch. Ďalej si prezerá štatistiky používania vozidiel, na akých trasách boli použité

2.4 Správa užívateľov

Každý užívateľ sa dostane do systému len po korektnom prihlásení. Nie je možné manipulovať s aplikáciu neregistrovaným užívateľom.

2.4.1 Práva

Práva užívateľov nie sú pevne zadefinované, ale je ich možno voliť na každú položku zvlášť. To znamená, že ten kto má práva na správu užívateľov môže im udeľovať aj práva na iné položky ako sú návrh trasy, správa mapy, správa vozidiel. Užívateľovi, ktorý nebude mať dostatočné práva na nejakú časť aplikácie, bude táto skutočnosť oznámená.

3 Implementácia

Táto kapitola sa sústreďuje na štruktúru programu a približuje prepojenie celého výsledného systému. Ide o webovú aplikáciu preto je nutné použiť programovacích postupov využívaných pri tvorbe webu. Systém je implementovaný za použití technológií PHP, MySQL, HTML, CSS, JavaScript. V následujúcich riadkoch sa dočítate o dôvodoch prečo sa technológií použili a v akom rozmedzí, ako boli implementované algoritmy spomínané v kapitole 1 a ako bol tvorený interface medzi užívateľom a systémom spomínaný v kapitole 2.

Celý systém pracuje s kódovaním UTF-8, s ktorým nedochádza k problémom na rôznych platformách.

3.1 Technológie

V tejto časti sa dozviete, prečo boli použité konkrétne programovacie jazyky a technológie, a akým spôsobom sú v aplikácii využité.

3.1.1 PHP

Hypertextový preprocesor (PHP) je skriptovací programovací jazyk určený predovšetkým pre programovanie dynamických internetových aplikácií. Najčastejšie sa začleňuje do štruktúry HTML, XHTML, čo je veľmi výhodné pre tvorbu webových aplikácií. PHP ale je možné využiť aj pri tvorbe konzolových a desktopových aplikácií [2] a [9].

Skripty sú vykonávané na strane serveru, a k užívateľovi sú prenášané až výsledné dáta. Syntax je kombináciou viacerých programovacích jazykov ako sú Perl, C, Pascal, Java. Je nezávislý na platforme a veľmi dobre sa pomocou neho pristupuje k databázam.

V systéme sa používa PHP5. Ten bol nasadený kvôli možnosti objektového programovania, keďže v predchádzajúcich verziách ešte nebolo možné objekty použiť. Prechodom z PHP4 na PHP5 vznikli mnohé nové funkcie, ktoré uľahčujú tvorbu programu a aj to bol dôvod prečo použiť PHP5.

PHP je v aplikacii využité z najväčšej časti. Sú v ňom implementované všetky vyhľadávacie algoritmy, samotné vykresľovanie mapy, prihlasovanie užívateľov, správa prvkov aplikácie.

3.1.2 MySQL

Ide o databázový systém, vytvorený švédskou firmou MySQL AB. Je považovaný za úspešného priekopníka v dvojnom licencovaní – je k dispozícií pod bezplatnou licenciou GPL ale aj pod platenou licenciou.

MySQL je multiplatformná databáza. Komunikácia s ňou prebieha pomocou jazyka SQL. Je použiteľná multiplatformne, prečo sa stala veľmi obľúbenou. Jej hlavnou vlastnosťou je rýchlosť a to aj za cenu niektorých zjednodušení: má jednoduché spôsoby zálohovania a donedávna ani neboli podporované pohľady, triggery a úložné procedúry [12].

V aplikacii je použitá na uchovávanie dát o mape ako sú mestá, dopravné uzly, cesty, užívatelia, vozidlá a práca so systémom. Použitá je verzia MySQL 5.

3.1.3 HTML

Hypertext markup language (HTML) je značkovací jazyk pre hypertext. Je jedným z jazykov pre vytváranie stránok WWW, ktorý umožňuje publikáciu dokumentov na internete. Jazyk je aplikáciu skôr vyvinutého rozsiahleho univerzálneho značkovacieho jazyka SGML. Vývoj HTML bol ovplyvnený vývojom webových prehliadačov, ktoré spätne ovplyvňovali definíciu jazyka [10].

V aplikacii je z dôvodu jednoduchého značkovania dát získaných z výpočtov a dôležitý je aj pri samotnom layoute stránky.

3.1.4 CSS

Cascading Style Sheets (CSS) je jazyk pre popis spôsobou zobrazenia stránok písaných v HTML, XHTML, alebo XML. Bol navrhnutý štandadizačnou organizáciou W3C. Hlavým zmyslom je oddeliť obsahovú formu, ktorá príde užívateľovi vo forme (X)HTML, XML, a vzhľad [11].

Aplikácia využíva CSS len na vzhľad. Sú nimi definované všetky grafické prvky.

3.1.5 JavaScript

Ide o multiplatformný objektovo orientovaný skriptovací jazyk. Z pravidla sa používa ako interpretovaný programovací jazyk pre WWW stránky, často vkladaný priamo do HTML kódu stránky. Sú ním najčastejšie ovládané rôzne interaktívne prvky GUI alebo animácie a efekty obrázkov. Viac [3].

Syntax patrí do rodiny jazykov C/C++/Java. S programovacím jazykom Java – ako vyplíva z názvu – ale nemá nič spoločné. Program v JavaScripte sa spúšťa zväčša až po nahraní ku klientovi. Nemôže pracovať so súbormi – tým je zabezpečená bezpečnosť užívateľa.

V systéme sa s ním stretneme takmer všade. Jeho najväčším uplatnením je při správe mapy a tvorby trasy. Keďže ide o mapu, sú použité klikateľné miesta, vyskakovacie okná, … A tie je najefektívnejšie implementovať pomocu JavaSriptu.

3.2 GUI (Grafické uživatelské rozhranie)

Stránka je ladená v jednoduchom štýle. Navrchu sa nachádza baner s motívom cesty a po bokoch fakultné a univerzitné logo. Väčšinu obrazovky zaberá telo stránky, na ktorom sa vyskytuje buď mapa alebo tabuľky pre správu vozidiel a užívateľov. V patičke stránky sú uvedené kontaktné údaje.

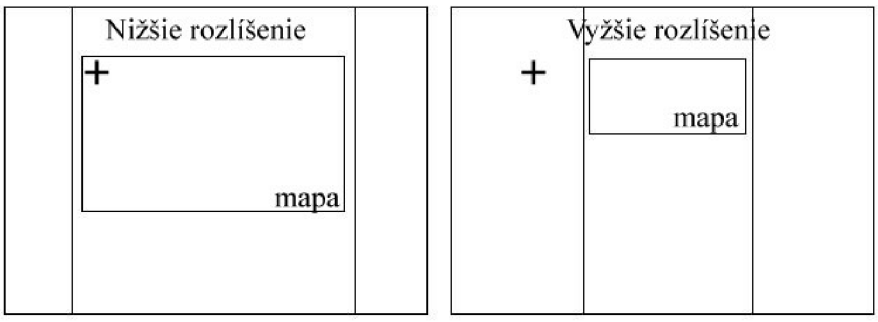
Celá práca v aplikácii je orientovaná najmä na prácu s myšou, čiže sa tu vyskytuje množstvo klikateľných miest, napríklad tlačítko šipky pre skrytie baneru, na efektívnejšiu prácu s mapou.

Obrázky boli vytvárané tak, aby nenarušovali celkový dojem stránky ladenej do bledomodrej farby s prvkami vínovo červenej a čiernej. Používané obrázky sú vo formáte gif a png. Tieto formáty sa používajú často na webových aplikáciach, pretože pri nich klient so serverom komunikuje rýchlejšie.

3.2.1 Mapa

V tvorbe webových stránok sa v minulosti na vnorovanie stránok používal tag *iframe,* ktorý vložil do dokumentu ďalšiu stránku. Tento spôsob sa pomaly vytratil a na vnorovanie stránok sa momentálne využívajú iné technológie ako HTML. Veľmi obľúbenou je použitie PHP. Napriek tomu ale je v aplikácii *iframe* použitý.

Je to z dôvodu „klikateľnosti“ mapy. Pokiaľ by bola mapa vložená bez *iframe*-u nebolo by možné JavaScriptom zistiť miesto kliknutia, pretože pri rozličných prehliadačoch je stránka na iných miestach a teda nie je zaručené správnosť detekovania kliknutia na ploche stránky (obrázok 3.1). krížik na obrázku znázorňuje rovnaké súradnice kliknutia, ale iné miesta vzhľadom na stránku.



Obr. 3.1 Obrázok demonštrujúci rozdelenie miesta kliknutia na rozličných rozlíšeniach

Tým, že je vložená ďalšia stránka pomocou iframe-u docielime toho, že na okraji nového *frame*-u sa súradnice počítajú opäť od (0,0).

Mapa je tvorená obrázkom, ktorý je uložený na serveri. Pri každom generovaní mapy sa tento obrázok prepisuje a je možné sa k nemu dostať aj priamou cestou. Toto slúži na prezeranie obrázku mapy inak ako aplikáciou. Obrázok je vo formáte png.

Generovanie mapy je zabezpečené funkciami na tvorbu obrázkov v PHP.

3.3 Implementácia vyhľadávacích algoritmov

Všetky vyhľadávacie algoritmy sú implementované v zvlášť php súboroch na kvôli jednoduchšej úprave. Súbor pritom pracuje ako knižnica prikladaná k iným skriptom. Vyhľadávacie algoritmy sú v nich implementované na objektovo orientovanej úrovni.

3.3.1 K-means

Implementácia je realizovaná v súbore *k-means.php*. Obsahuje 2 dátové štruktúry a 1 funkciu. Konkrétne to je dátová štruktúra *cluster()*, ktorá je abstraktom zhluku. Každý takýto zhluk má atribút nesúci informáciu o bodoch, ktoré zhluk obsahuje, bod, ktorý je stredom zhluku a metódu na prepočítanie tohoto stredu.

Ďalšou štruktúrou je samotný bod *point()*. Obsahuje atribúty súradníc a názvu bodu, a metódu na získanie vzdialenosti tohoto bodu od iného.

Poslednou položkou je funkcia *distributeOverClusters()*. Funkcia má na vstupe počet clusterov(zhlukov), na základe ktorých sa mapa delí a pole bodov, ktoré majú byť analyzované. Výsledkom je pole zhlukov vyhodnotených práve touto funkciou.

3.3.2 A*

Vyhľadávací algoritmus A* je zložitejší, a preto si vyžadoval viacero prvkov a ich atribútov. Pri vyhľadávaní sa využívajú zásobníky, body, previazania medzi nimi, atď. Obsah je umiestnený v súbore *aStar.php*.

Elementárnou dátovou štruktúrou je bod zo zásobníku. Ten nesie informáciu o tom aký to je bod, zo štruktúry *point()* použitej v k-means. Nasleduje atribút otcovského bodu, hodnota H a G využívaná pri výpočtoch A* a metóda na výpočet hodnoty F.

Štruktúry pre zásobníky Open a Closed majú rovnaké jadro: body obsiahnuté v zásobníku, vkladanie bodov do zásobníkov a detekcia či sa bod v zásobníku nachádza. Open zásobník navyše obsahuje metódu na vyňatie bodu zo zásobníku a vyhľadanie najlepšieho bodu zo zásobníku. Najlepší je v tomto prípade myslený s najmenšou hodnotou F.

Jedinou funkciou tohoto súboru je *aStar()*, ktorá má za úlohu nájdenie najkratšej cety medzi dvomi bodmi. Na vstupe je cieľový a počiatočný bod cesty. Na základe pravidiel spomínaných v kapitole 1 je vypočítaná najkratšia cesta a vrátená funkciou vo forme poľa bodov štruktúry.

3.3.3 Problém obchodného cestujúceho

Objekty v tomto algoritme sú obsiahnuté v súbore tsp.php (traveling salesman problem). Obsahuje položky spojené s mestami, trasami a celou mapou.

Základným prvkom je objekt mesto. Nesie identifikačné číslo mesta, jeho názov a súradnice na mape. Jedinou metódou je získanie priamej vzdialenosti medzi mestami.

Nad ňou je objekt *routeCity()*. Ten obsahuje 3 atribúty: spojenie 3 miest. Konkrétne to sú aktuálne mesto jeho jeden a druhý sused. Objekt slúži na zisťovanie susediacich miest.

Ďalšou položkou je objekt cesty, ktorá obsahuje mestá v celej okružnej trase. A metódu na prepočítanie dĺžky trasy. Tá je prepočítavaná na základe susedov z prechádzajúceho objektu.

Po vygenerovaní náhodných trás na počiatku algoritmu sú všetky zapísané v štruktúre ciest, ktorá navyše obsahuje metódu na rekalkuláciu najkratšej z nich a inú na získanie konkrétnej cesty. Tá sa získava volaním metódy kde vstupom je objekt cesta výstupom je pole miest, ktoré sú v ceste obsiahnuté zoradené podľa toho akú následnosť cesta nesie.

3.4 Databáza

Databáza bola vytváraná v programe phpMyAdmin. Ide o aplikáciu slúžiacu na správu a tvorbu databáz MySQL. Verzia phpMyAdminu je 2.9.1.1., ktorá postačovala na všetky potreby aplikácie.

Databáza nie je nejako náročná, ide len o uschovávanie dát a rýchlu prácu s nimi, MySQL bola volená z dôvodu jej rozšírenia a silnej komunity používateľov. Databáza obsahuje 7 tabuliek.

- Mesta: tu je obsiahnutá x-ová a y-ová súradnica výskytu na mape jeho názov a primárny kľúč id.
- Cesty: slúžia na spojenie dvoch miest alebo dopravných uzlov. Identifikovaná je primárnym kľúčom id, obsahuje položky štart, ciel a hodnotu trasy.
- Uzol: je identický s mestom ale nenesie v sebe žiaden názov.
- Uzivatel: ide o tabuľku charakterizujúca jednotlivých užívateľov. Primárny kľúč je id a ďalšími položkami, ktoré netreba popisovať sú: login, heslo, meno a práva.
- Vozidlo: opäť id ako primárny kľúč, názov a popis
- Trasy: Posledná tabuľka nesie záznamy o realizovaných trasách. Jej stĺpce sú: id, id_vozidla, popis

Všetky stĺpce sú najmä dvoch typov: *integer* pre primárne kľúče, a previazania medzi nimi. Druhým typom je *varchar*. Oba typy sú alokované tak, aby spĺňali požadované vlastnosti tj. pre popisy istých prvkov je volený väčší rozsah a pre iné naopak menší

Pripojenie k databáze MySQL je zabezpečené v súbore *mysql.php* kde sú dva príkazy, ktorými dôjde k pripojeniu na MySQL server a výberu danej databázy.

Celá aplikácia pracuje s 1 databázou a tabuľkami v nej.

3.5 Ostatné

V táto časť práce sa venuje ostatným prvkom implementácie, ktoré neboli doposiaľ spomenuté, ale sú tiež dôležité.

3.5.1 Bezpečnosť

Do aplikácie je možné sa dostať len s platným menom a heslom. Celý systém je orientovaný na jediný súbor *index.php* v tomto je modul prihlasovania implementovaný a na základe adresy sa vkladajú do neho ostatné stránky. K ostatným stránkam sa užívateľ môže dostať, ale nie je možné získať nejaké konkrétne dáta bez prístupu na *index.php*.

Práva súborov: všetky súbory majú nastavené práva na 644 teda čítanie pre všetkých a zápis pre vlastníka. Výnimkou sú obrázky reprezentujúce mapy, tie majú práva na zápis pre všetkých, aby mohlo dôjsť k prepisovaniu obrázka.

Komunikácia zo serverom prebieha po HTTP (Hypertext transfer protokol). Heslá uložené v databázy sú kódované šifrovacím algoritmom MD5.

3.5.2 Sessions

Sessions riešia problém bezstavovosti protokolu http, udržíme pomocou nich informácie o stave aplikácie a prácu s ňou. Mechanizmus sessions od prvých prídavných doplnkov pre PHP3 vykonal veľa zmien a od PHP 4.1 je práca so sessions štandardne dobre funkčná a použiteľná.

V systéme sú využívané na detekciu prihlásenia užívateľa. Keď sa prihlási do systému vygeneruje sa hodnota *$_SESSION['login']* a v nej informácia o prihlásení. Po odhlásení užívateľa sa táto hodnota vyprázdni a aplikácia užívateľa berie ako neprihláseného.

Ďalším spôsobom ako sú sessions využité je prenášanie dát v krokoch pri generovaní trás. Generujú sa rôzne dátové štruktúry, ktoré je najlepšie prenášať práve pomocou sessions.

3.5.3 Formuláre

Formuláre sú odosielané metódou post. To je z dôvodu krajšej URL, pretože pri použití metódy get sa hodnoty formulára prenášajú práve v tejto adrese.

Formuláre sú využívané najmä na správu užívateľov a dopravných prostriedkov, ale jeden je aj generovaný v správe mapy / návrhu trasy. Generovaný je na základe na súradnice, na ktorú bolo na mape kliknuté. Ak užívateľ klikne v správe mapy na prázdnu plochu ukáže sa formulár pre tvorbu mesta alebo dopravného uzlu, naopak ak klikne na mesto alebo dopravný uzol ponúkne sa mu možnosť vytvárať cestu. V návrhu trasy formulár vedie k správnemu vytvoreniu trasy.

4 Testy a výsledky

Táto kapitola poukazuje na výsledky aplikácie. Sú tu zhrnuté výpočty či už s pozitívnym nájdením, alebo negatívnym. Sumarizuje úspešnosť v hľadaní. Výsledky sú ukázané na základe testov rozdelených do 3 kategórií podľa náročnosti.

Testy sú realizované na mapke s 22 mestami a 5 dopravnými uzlami. Ide o model časti Slovenska Záhorie a jeho okolie.

Testy prebiehali na serveri s procesorom Intel Dual Core taktovaný na 2.4GHz, s 3GB RAM.

4.1 Jednoduché testy

Jednoduché testy boli vykonávané na základe jednoduchých vstupných podmienok. Vstupom je buď malý počet miest, alebo vozidiel, a ich kombinácia.

4.1.1 Mestá blízko seba s jedným vozidlom (test č.1)

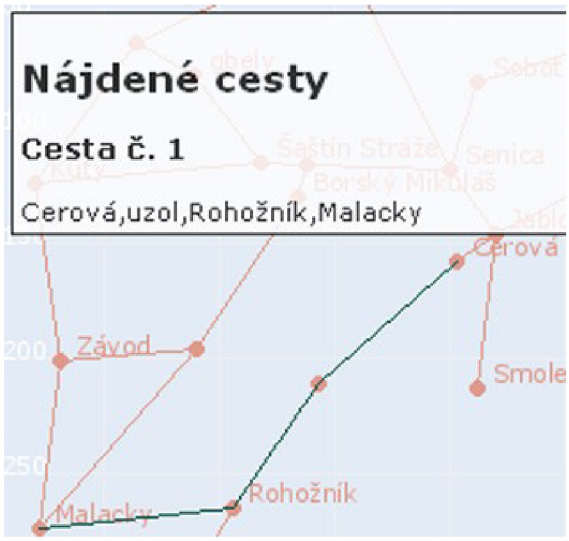
Tento test je jedným zo základných, ide o nájdenie optimálnej trasy medzi 2 mestami, nepriamo spojenými, cesta medzi nimi je jednoduchá – prechod medzi 4 mestami.

Výsledok je výborný, podľa obrázku 4.1 je evidentné, že ide o najkratšiu trasu.

4.1.2 Test s dvomi vozidlami (test č.2)

Ide o testovanie viacerých vozidiel na nájdenie 2 trás medzi 4 mestami. Predpokladáme, že mestá sa rozdelia do 2 skupín a v každej skupine bude disponovať jedno vozidlo.

Výsledok spĺňa očakávania. Mapa bola rozdelená na 2 časti a v každú časť prechádza jedno vozidlo. Konkrétne prejde každé vozidlo 2 mestá.



Obr. 4.1 Výsledok 1. testu

Obr. 4.2 Výsledok 2. testu

4.2 Stredne náročné testy

Tieto testy sú orientovné na strednú náročnosť výpočtov. Očakávajú sa kladné výsledky s prípadnými menšími odlišnsťami od reálnej najoptimálnejšej trasy.

4.2.1 Viacej miest pre jedno vozidlo (test č.3)

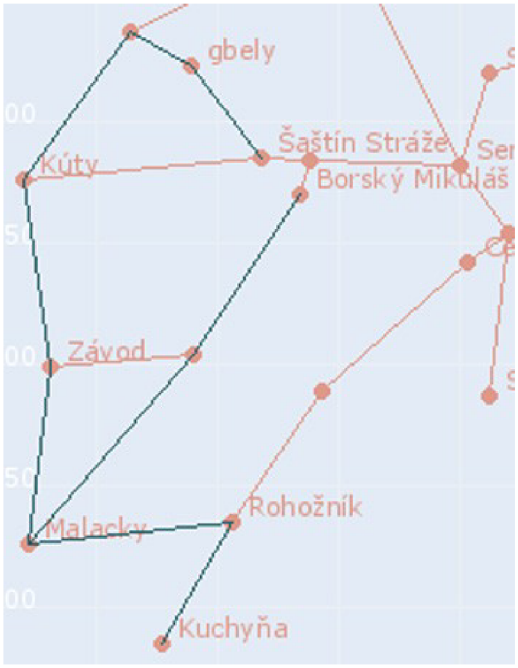
Vybraných bolo 8 miest, rozložených v blískosti seba. Tieto mestá má obísť jedno vozidlo.

Výsledkom je trasa, ktorá nie je priama - cesta sa vetví, čo hodnotím ako plus, nakoľko nie je cieľom obísť mesta okružnou trasou, ale najrýchlejšou. K tomuto výsledku prikladám mimo grafického výstupu aj textový, aby bolo možné vidieť spôsob vetvenia trasy.

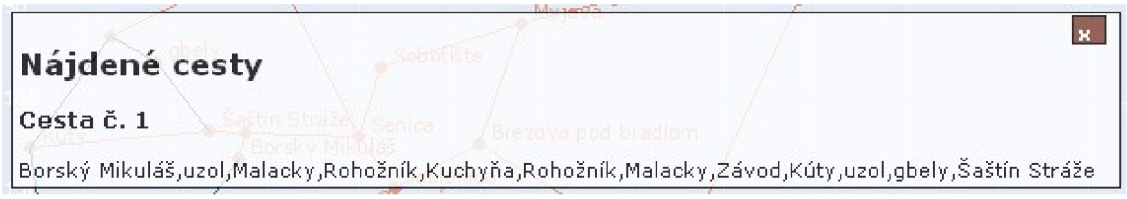
4.2.2 Viacej miest pre 2 vozidlá (test č. 4)

V tomto teste je skúmaný postup algoritmov pre výpočet 2 trás pre 2 vozidlá na prejazd 4 mestami rôzne rozloženými po mape. Očakávaný je kladný výsledok s prípadnou nepresnosťou rozdelenia miest do zhlukov.

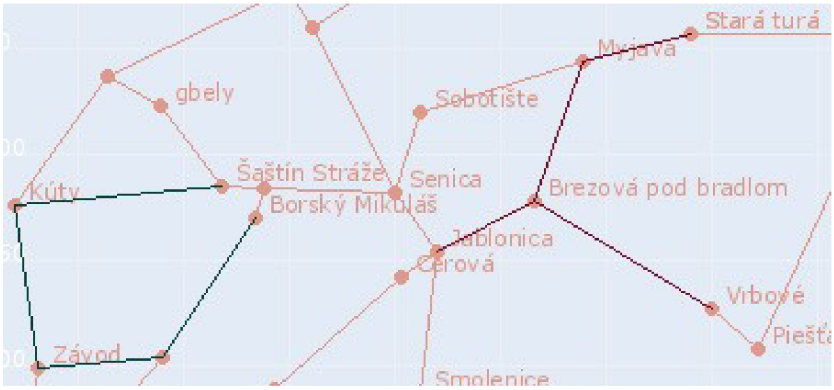
Výsledkom sú 2 trasy pričom jedna je priama a druhá je vetvená.U prvej trasy je možné ale zbadať neefektívne vypočítaný začiatok/koniec trasy. Táto trasa by mohla byť optimálnejšia, ak by štartovné mesto a koncové mesto boli vzdialené viac od seba.



Obr. 4.3 Výsledok testu č.3 v grafickej podobe



Obr. 4.4 Výsledok testu č. 3 v textovej podobe.



Obr. 4.5 Výsledok testu č. 4

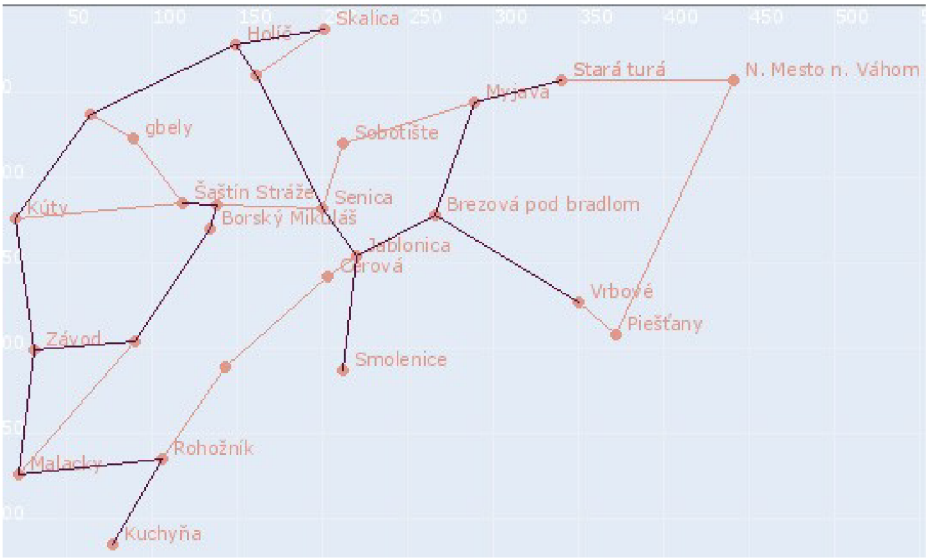
4.3 Náročné testy

Náročnejšie testy sú zamerané na náročné výpočty spôsobené veľkým počtom vstupných údajov a ich rozmanitosi.

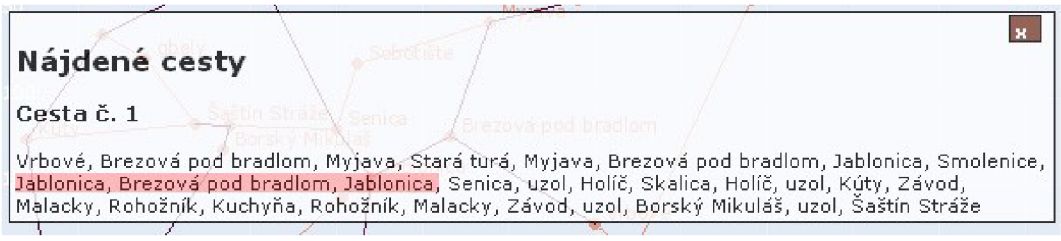
4.3.1 Viacero miest v jednej trase (test č. 5)

Test je zameraný na veľký počet miest (15) prechádaných 1 vozidlom. Hlavnou časťou výpočtu sú postupy spojené s TSP. Preto odhadujeme väčšiu časovú náročnosť, a aj prípadné nepresnosti.

Výsledokom je pomerne optimálna trasa. Jediným výkyvom je skutočnosť, že v rámci trasy je jedna dvojica miest prejdených raz navyše (vyznačené na obrázku 4.7). To môže byť spôsobené nesprávnym rozložením cesty po výpočtoch TSP. Táto nepresnosť je zanedbateľná tým, že si ju uživateľ jednoucho všimne a v reálnom nasadení túto skutočnosť bude ignorovať a nadbytočnú trasu nezahrnie do reálnej. Pozitívom je ale rýchlosť. Trasa bola vygenerovaná do 0,3 sekundy.



Obr. 4.6 Výsledok testu č. 5 – grafická podoba



Obr. 4.7 Výsledok testu č. 5 – textová podoba

4.3.2 Viacero miest, viacero vozidiel (test č.6)

Posledný test je zameraný na prechod mapou cez viacero miest (všetkých 22) pomocou 5 vozidiel.Na základe predchádzajúcich testov sú očakávané pomerne dobré výsledky. V tomto teste je ukázané aj rozdelenie zhlukov (2. krok výpočtu).

Výsledok sa dá považovať za najoptimálnejší.

Obr. 4.8 Test č.6 rozdelenie miest do zhlukov a výpočet TSP

Obr. 4.9 Výsledok testu č. 6 – grafická podoba

Obr. 4.10 Výsledok testu č. 6 – textová podoba

4.4 Zhrnutie

Výstup aplikácie je zväčša najoptimálnejšia trasa. Výkyvy nástávajú pri výpočte trasy pre jedno vozidlo, ak je prechádzaných viacero miest. Keďže je aplikácia tvorená viacmenej so základom pomôcť pri logistike, nie je to chyba až tak závažná. Výtup je len vodítko, pre užívateľa ako navrhnúť reálnu trasu dopravy.

Problémy nastávajú najmä pre spôsob vyhľadávania algoritmu pre problém obchodného cestujúceho. Výsledky ale niesú nepoužiteľné. Výstup prichádza v čase pre takýto typ aplikácie vyžadovaný. Nebolo cieľom ponúknuť čo najefektívnejšiu cestu za cenu času generovania, ale dostatočne efektívnu vzhľadom na dobu výpočtu. Doba výpočtu bola závislá od náročnosti tesu, no nepresahovala dobu 0,5 sekundy.

5 Záver

Výsledkom práce je webová aplikácia zahrňujúca viacero postupov umelej inteligencie. Ide o komplexný systém na vytváranie ciest na základe vstupnej mapy a požiadavkov užívateľa. Aplikácia generuje najefektívnejšie trasy, ktoré užívateľovi pomôžu pri reálnej tvorbe logistiky.

Bolo riešených viacero problémov z oblasti umelej inteligencie, ktoré boli aplikované do užívateľsky prívetivého rozhrania zakomponovaného do informačného systému. Boli navrhnuté koncepcie na jednoduchú správu systému a tie následne implementované pomocou programovacích jazykov a metód spojenými s tvorbou webových aplikácií. Záverom tieto postupy boli dôkladne otestované, výsledky zhodnotené.

Za vlastný prínos považujem najmä samotné vytvorenie aplikácie. So systémom podobným tomuto sa málokto už stretol. V dnešnej dobe je internet zaplnený redakčnými systémami a rôznymi aplikáciami, ktoré „len“ komunikujú s databázou. Tieto systémy sú stále viac automatizované a na to, aby sa prinieslo niečo nové, je treba hľadať nové možnosti. Web je v súčasnosti veľmi rozšírený, ale chýbajú aplikácie s konkrétnym zameraním. Tieto ľudia vyhľadávajú, preto aj samotná aplikácia je zameraná práve na to, o čom má byť a to je logistika. To, že je to informačný systém som sa snažil zatieniť i keď nie odstrániť.

Ide o systém, ktorý spája z môjho pohľadu 3 obsiahle celky:

- umelá inteligencia – vyhľadávanie trás
- práca s mapami – generovanie máp
- informačný systém – správa databázy, správa prvkov aplikácie

Dosiahnuté výsledky sú z môjho pohľadu výborné, čo okrem iného dokazuje kapitola zaoberajúca sa testovaním. Keďže ide o systém využívajúci tak náročné riešenia problémov ako je najmä problém obchodného cestujúceho, nie je možné dotiahnuť projekt k dokonalosti za tak krátky čas. Z tohoto dôvodu bude vývoj aplikácie pokračovať. Budúci vývoj sa zameria hlavne na nasledujúce body:

- **vyhľadávacie algoritmy:** postupným vylepšovaním algoritmov umelej inteligencie bude dosiahnutých presnejších a rýchlejších výpočtov.
- **mapy:** ďalším cieľom je zkvalitniť možnosti práce s mapou, konkrétne možnosť pohybovania mapou, jej približovanie/odďalovanie, vytvorenie tzv. nekonečnej mapy, ktorá nebude obmedzená na pevné rozmery.
- **moduly:** vytvorenie možnosti jednoduchého vkladania modulov, ktoré si uživateľ môže sám dopĺňať. Príkladom môže byť správa účtovníctva, správa personálu, modul skladu, atď.

Z uvedeného je pochopiteľné, že práca na aplikácii bude pokračovať a mohla by sa stať námetom na ďalšie podobné systémy, ktoré by boli viac špecifické.

Postupy pri tvorbe časti informačného systému boli zvolené na základe skúseností nadobudnutých z doterajšej praxe s realizovaním projektov založených na báze informačných systémov. Ostatné prvky aplikácie boli naštudované a aplikované do systému.

Literatúra

[1] Zbořil, F., Zbořil, F.: *Základy umělé inteligence: Studijní opora* FIT VUT v Brně, 2006, 147 s.

[2] Kosek, J.: PHP - Tvorba interaktivních internetových aplikací, Grada Publishing, 1999.

[3] PÍSEK, Slavoj. *JavaScript : efektní nástroj oživení WWW stránek.* 1. vyd. Praha : Grada Publishing, 2001. 232 s. ISBN 80-247-0014-X

[4] Bajer, L.: *Algoritmy pro pathfinding*, MFF UK, 2006, [cit 2008-5-8] Dostupný z WWW: <http://ksvi.mff.cuni.cz/~brom/hagents/H-likeAgents4_Bajer060410.pdf>

[5] Lester, P.: *A* Pathfinding for Beginners* [online]. 2005, [cit 2008-5-8] Dostupný z WWW: <http://www.policyalmanac.org/games/aStarTutorial.htm>

[6] *Optimal TSP Tours* [online]. 2003 [cit. 2008-5-8]. Dostupný z WWW: <http://www.tsp.gatech.edu/optimal/index.html>.

[7] *TSPLIB* [online]. 2007 [cit. 2008-05-07]. Dostupný z WWW: <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.

[8] *Concorde TSP Solver* [online]. 2005 [cit. 2008-05-07]. Dostupný z WWW: <http://www.tsp.gatech.edu/concorde.html>.

[9] *PHP - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-05-02]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[10] *HyperText Markup Language - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/HTML>.

[11] *Cascading Style Sheets - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-05-02]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>.

[12] *MySQL - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-05-11]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

Zoznam príloh

Príloha č. 1 – Ovládanie

Obr. 1: hlavný banner s menu

Návrh trasy – tu užívateľ môže navrhnúť trasu. Podmienkami bude vybrať mestá, ktoré chce obísť a vozidlá, ktoré má záujem použiť.

Správa mapy – vytváranie máp, možnosť prekryť plochu obrázkom reálnej mapy na zjednodušenie samotnej tvorby. Kliknutím užívateľ vytvára nové mestá, dopravné uzly a cesty.

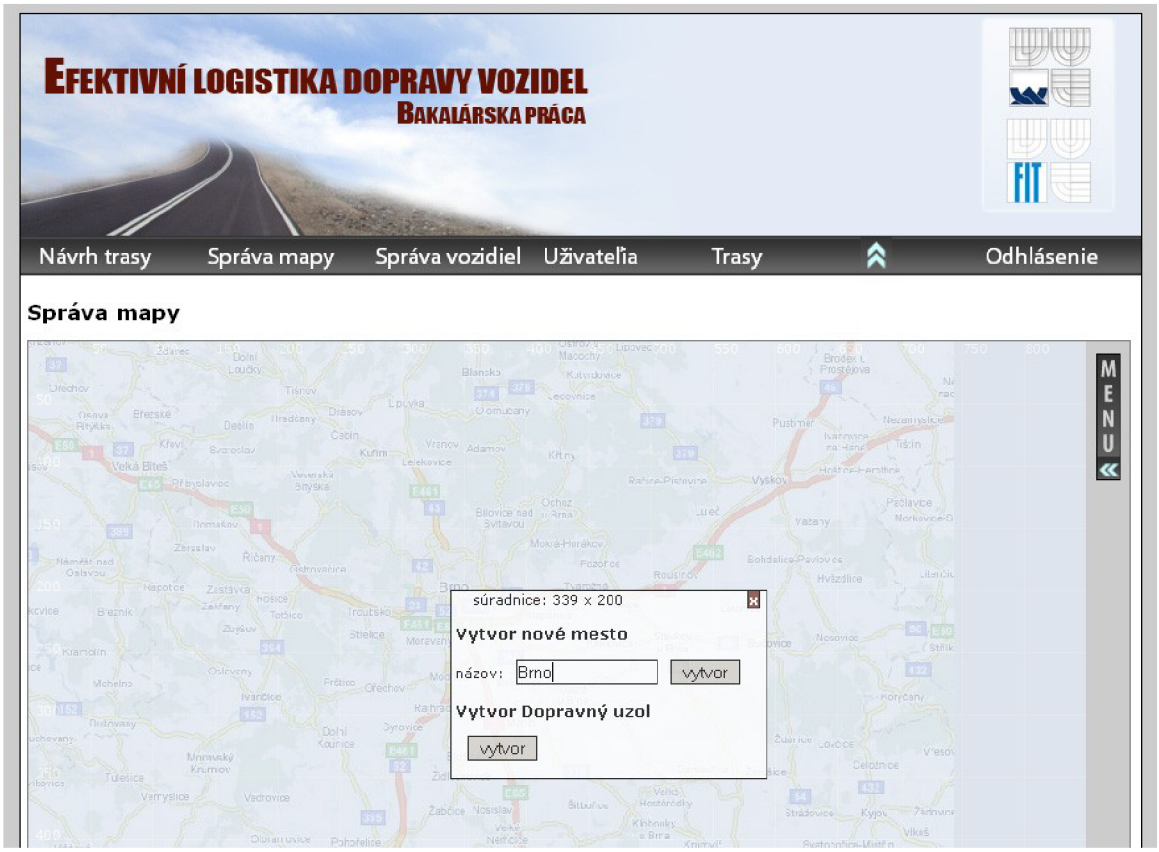
Správa vozidiel – vytváranie, editácia a mazanie vozidiel, ktoré ďalej využíva pri tvorbe trás.

Užívatelia – správa užívateľov (vytváranie, editácia, mazanie), možnosť prideľovať práva na vstup do sekcií.

Trasy – zoznam realizovaných trás .

Odhlásenie – odhlásenie užívateľa zo systému.

Príloha č. 2 – Obrázky aplikácie



Obr. 1. Tvorba mesta, so zapnutou mapou na pozadí

Obr. 2. Vytváranie uživateľa

Príloha č. 3 - Zoznam použitých obrázkov