VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV MANAGEMENTU

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF MANAGEMENT

VYUŽITÍ PROSTŘEDKŮ UMĚLÉ INTELIGENCE PRO PODPORU ROZHODOVÁNÍ V PODNIKU

THE USE OF MEANS OF ARTIFICIAL INTELLIGENCE FOR THE DECISION MAKING SUPPORT IN THE FIRM

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. PETR JÁGR
AUTHOR

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

BRNO 2012

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2011/2012
Ústav managementu

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jágr Petr, Bc.

Řízení a ekonomika podniku (6208T097)

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 prostředků umělé inteligence pro podporu rozhodování v podniku

v anglickém jazyce:

The Use of Means of Artificial Intelligence for the Decision Making Support in the Firm

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DOSTÁL, P. Pokročilé metody analýz a modelování v podnikatelství a veřejné správě. 1. vyd. Brno : CERM, 2008. 340 s. ISBN 978-80-7204-605-8.
DOSTÁL, P. Advanced Decision Making in Business and Public Services. Brno : CERM, 2011. 168 s., ISBN 978-80-7204-747-5.
SCHWARZ, J., SEKANINA, L. Aplikované evoluční algoritmy. Brno : VUT – FIT, 2006. Studijní opora, 101 s.
THE MATHWORKS. MATLAB – User's Guide. The MathWorks, Inc. 2010.
THE MATHWORKS. MATLAB – Global Optimization Toolbox - User's Guide. The MathWorks, Inc. 2010.
ZBOŘIL, F., ZBOŘIL, F., Jr. Základy umělé inteligence, Brno : VUT – FIT, 2006. Studijní opora, 142 s.

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

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2011/2012.

L.S.

PhDr. Martina Rašticová, Ph.D.	doc. RNDr. Anna Putnová, Ph.D., MBA
Ředitel ústavu	Děkan fakulty

V Brně, dne 19.01.2012

Abstrakt

Diplomová práce se zabývá využitím prostředků umělé inteligence jako podpora manažerského rozhodování v podniku. Součástí práce je aplikace využívající genetické a grafové algoritmy při optimalizaci umístění výrobních závodů a logistických skladů z hlediska nákladů na přepravu.

Abstract

The master's thesis deals with the use of artificial intelligence as support for managerial decision making in the company. This thesis contains the application which utilize genetic and graph algorithms to optimize the location of production facilities and logistic warehouses according to transport cost aspects.

Klíčová slova

Umělá inteligence, genetický algoritmus, Dijkstrův algoritmus, dopravní náklady, Matlab, Global optimization toolbox, grafické uživatelské rozhraní, SWOT analýza, Porterova analýza

Keywords

Artificial intelligence, genetic algorithm, Dijkstra algorithm, transportation costs, Matlab, Global optimization toolbox, graphic user interface, SWOT analysis, Porter analysis

Bibliografická citace

JÁGR, P. *Využití prostředků umělé inteligence pro podporu rozhodování v podniku.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2012. 127 s. Vedoucí diplomové práce prof. Ing. Petr Dostál, CSc.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu zákona č. 121/200 Sb. O právu autorském a o právech souvisejících s právem autorským).

datum podpis

Poděkování

Děkuji panu prof. Ing. Petru Dostálovi CSc. za odbornou pomoc a připomínky při řešení diplomové práce.

Obsah

1 Úvod

Jak sám název diplomové práce napovídá, v následujícím textu se budu zabývat problematikou umělé inteligence a jejího využití pro podporu důležitých rozhodnutích v podniku.

V dnešní technicky pokrokové době pronikají do rozhodovacích procesů moderní podpůrné simulační nástroje téměř na všech úrovních. V zájmu každého podniku je využít tyto příležitosti nové doby co nejlépe, protože získaná konkurenční výhoda může představovat zásadní faktor budoucího úspěchu.

Před několika desítkami let bylo obtížné si představit podnikové využití informačních technologií jinak něž např. při hromadném zpracovávání statistických dat nebo řešení jiných jednodušších úloh a problémů. V posledních letech, kdy dochází k ucelení teoretických poznatků včetně jejich potřebné formalizace na poli umělé inteligence, jsme svědky stále častějšího uplatňování informačních technologií i na experimentální úrovni.

Umělá inteligence je sám o sobě velmi široký, téměř až filozofický, pojem, který zahrnuje mnoho různorodých oblastí. V této práci se zaměřím hlavně na genetické algoritmy (GA), jež jsou podmnožinou evolučních algoritmů. GA představují uplatnění některých vybraných principů a pravidel, se kterými se běžně setkáváme v přírodě, v počítačovém programu. K tomuto účelu využiji podporu softwarového nástroje MATLAB a jeho součásti nazvané Global Optimization Toolbox, která obsahuje všechny potřebné funkce.

Jádro práce bude tvořit aplikace hledající optimální rozmístění výrobních závodů a logistických skladů koncernu Knorr-Bremse a to pouze z hlediska přepravních nákladů. Pokud bychom chtěli uvažovat i s dalšími podstatnými druhy nákladů, mohli bychom dospět k poměrně náročnému funkčnímu modelu a tím by se složitost aplikace a práce s ní zvýšila nad neúnosnou mez. Aplikace bude dále využitelná při porovnávání celkových dopravních nákladů s různými dodavateli, což může hrát významnou roli při volbě, se kterým se naváže obchodní spolupráce.

Závěry práce budou tvořit zvažovaný faktor při rozhodování o budoucím rozvoji

a způsobu expanze společnosti na další trhy. Samotná expanze je však z mnohem větší míry ovlivněna potenciálem trhu a dalšími podstatnějšími vlivy. Práce samotná je pro společnost pokus o nový přístup k problémům optimalizace dopravních nákladů, protože dosud se podobná rozhodnutí řeší hlavně na základě zkušeností a odhadů vedoucích pracovníků. Tento přístup se stává se zvětšujícím se množstvím výrobních závodů náročný. Aplikaci bude možno využít i na regionální úrovni při hledání např. optimálního umístění logistického skladu.

Součástí práce bude diskuze o nedostatcích a problémech spojených se zaváděním podobných experimentálních postupů do praxe a míra jejich relevance.

V závěru zvážíme budoucí možnosti rozvoje v této oblasti a návrhy na pokračování v realizované aplikaci.

2 Vymezení problému a cíle práce

Cílem práce je nalezení optimálního rozmístění výrobních závodů z hlediska logistických nákladů. Protože výrobní program společnosti zahrnuje převážně na přepravu náročné strojní součásti, představují logistické náklady v celkových nákladech významnou část. Přeprava mezi pobočkami je realizována nákladní kamionovou dopravou a do zámořských oblastí lodí. Ve výjimečných případech je použita doprava letecká nebo vlaková.

Kamionová doprava je realizována na základě uzavřených dlouhodobých vztahů se smluvním přepravce a na snížení jejího využívání se budu zaměřovat. Díky obecné povaze problému však bude možné navržené řešení jednoduše aplikovat i na lodní, leteckou nebo i kombinovanou dopravu.

Cílem práce bude navrhnutí aplikace věrně zachycující vztahy přepravních nákladů a rozmístění dílčích výrobních závodů, jejich dodavatelů a odběratelů. Do těchto nákladů je započítáváno několik méně podstatných specifických položek, jako jsou např. poplatky za užití dálniční sítě, poplatky spojené s provozem vozu na pozemních komunikacích apod., které díky zajišťování dopravy subdodavatelem a existencí smluvní ceny za kilometr zanedbám.

Na problém můžeme pohlížet z globálního hlediska, kdy sledujeme všechny přepravní náklady mezi všemi pobočkami nebo nás zajímají náklady na přepravu určité podmnožiny výrobků. Závěry u rozdílných množin výrobků se ze své podstaty mohou lišit, protože ve výpočtech jsou uvažováni jiní dodavatelé, jiní odběratelé, ale i jiné výrobní závody.

Vytvořená aplikace bude schopná pracovat s orientovaným grafem představujícím samotné přepravní náklady. Těmito grafy bude popsáno několik množin výrobků z aktuálního výrobního programu a budou navrhnuta možná opatření pro optimalizaci přepravních nákladů. Bude možné určit optimální rozložení výrobních operací v již existujících závodech nebo nalézt vhodného dodavatele.

Cílem práce není nalezení optimálního rozložení výrobních operací produktu mezi existujícími výrobními závody, i když to podstatně ovlivňuje celkové náklady na

dopravu. Úvaha o rozložení výrobních operací musí předcházet vytvoření orientovaného grafu, se kterým pracuje vyvinutá aplikace, a musí v ní být zahrnuta dlouhodobá strategie o pronikání na nové trhy s ohledem na předpokládané výrobní náklady jednotlivých regionů. Když bychom to řekli zjednodušeně, než začneme hledat optimální rozmístění výrobních závodů, musíme si určit, jak bude vypadat síť továren a jaké budou toky materiálu mezi nimi.

3 Teoretická východiska práce

Pro správné pochopení obsahu práce si musíme nejdříve definovat teoretické prerekvizity. Seznámíme se se základy algoritmizace, s pojmem *umělá inteligence*, se stěžejní látkou *genetických algoritmů* a popíšeme si *graf* tak, jak ho chápeme podle teorie grafů.

3.1 Umělá inteligence

Pojem inteligence můžeme chápat jako určitou složitou vlastnost entity, která jí umožňuje poznávat prostředí a využívat nabyté poznatky pro přizpůsobení se změnám. Tuto vlastnost bychom mohli nazvat jako kognitivní, protože inteligence zahrnuje chápavost, předvídavost, hodnocení, paměť a usuzování.(24)

Umělá inteligence je inteligence uměle vytvořeného systému. V našem případě budeme mluvit o umělé inteligenci simulované počítačovým programem. Program pak bude simulovat činnost inteligentní entity při řešení velmi složitých úloh, kde není možné použít numerické řešení. Jednání programu se bude pro okolí jevit jako inteligentní.

3.1.1 Turingův test

Abychom mohli o některém umělém systému říci, že je inteligentní, musí splnit předpoklady pro inteligentní chování. Tyto předpoklady můžeme testovat např. pomocí Turingova testu.

Tento test zformuloval Alan Turing v roce 1950 a poměřuje inteligenci umělé entity s inteligencí člověka. Test je od svého publikování podrobován neustálým kritikám, ale i přes své mnohé nedostatky se jedná o jeden z nejlepších dosud známých testů umělé inteligence.

Protože však pracuje na úrovni psaného textu, nebudeme ho využívat k testování inteligence nedostatečně komplexní aplikace vytvořené v rámci této práce.

3.2 Algoritmus

Algoritmem rozumíme formalizovaný postup dále nedělitelných kroků zpracovávajících vstupní informace za účelem dosáhnutí výstupních informací. Speciálním typem je algoritmus bez výstupních informací, který tak nepřináší žádný užitek v podobě výsledku. Takové typy algoritmů se používají výjimečně např. pro testovací účely. Druhým speciálním typem je algoritmus bez vstupních informací. Takové algoritmy se používají např. pro výpočet Ludolfova čísla π, Eulerova čísla e nebo prvočísel. Provedení jednoho kroku algoritmu nazýváme iterací. Algoritmy s nekonečným počtem iterací se nazývají nekonečné a patří mezi ně výše zmiňované výpočty iracionálních čísel a prvočísel. U provádění nekonečných algoritmů máme většinou další podmínky, při jejichž splnění provádění algoritmu ukončíme.

3.2.1 Determinismus

Další důležitou vlastností algoritmu je jeho předvídatelnost označovaná jako determinismus. Takový algoritmus má v každém kroku definovaný bezprostředně následující stav, ve kterém se bude nacházet. Při stejných vstupních informacích musí být výstupní informace rovněž stejné. Takové algoritmy nesmí využívat náhodná čísla a další nepředvídatelné proměnné. Pokud by tak činil, označujeme ho jako nedeterministický resp. stochastický. Příkladem deterministického algoritmu může být výpočet součinu dvou čísel. Za nedeterministický algoritmus můžeme považovat například genetický algoritmus, který při svém běhu používá náhodná čísla. Genetický algoritmus se kvůli tomu nikdy nechová stejně.

3.2.2 Rekurze

Rekurzivní algoritmus využívá v průběhu provádění opakované volání sebe sama s jinými vstupními parametry. Tento přístup v řadě případů vede k velmi elegantním řešením určitých problémů, ale při špatném použití může vést ke vzniku nekonečných algoritmů. Paměťová náročnost rekurzivních algoritmů bývá zpravidla vyšší než jejich ekvivalenty využívající iterační cykly, a proto by jejich použití měl předcházet detailní rozbor problému.

3.2.3 Časová a prostorová složitost

Mluvíme-li o algoritmech, musíme ještě zmínit pojem *časová složitost* a *prostorová složitost*. Tyto složitosti nám neříkají jak dlouho budeme algoritmus provádět nebo kolik nám obsadí místa v paměti, ale určuje, jak je doba provádění a obsazený paměťový prostor závislý na velikosti vstupních informací. Tyto závislosti zjišťujeme hlavně pro velká množství vstupních dat. Mějme hypotetický algoritmus sčítající všechny prvky pole. Tomuto algoritmu se prodlouží doba provádění o tolik, o kolik se prodlouží samotné pole. Tzn. je tu lineární závislost časové složitosti, kterou označujeme $O(N)$.

Prostorová složitost je v našem případě rovněž lineární, protože paměťové nároky se zvyšují úměrně s množstvím vstupních informací. Paměťové místo pro mezivýsledek zanedbáme, protože oproti velkým vstupním datům nehraje důležitou roli. Z časové a prostorové složitosti zřetelně vyplývá, že se vždy snažíme hledat a používat takové algoritmy, které mají funkci složitosti rostoucí co nejpomaleji. V tabulce 3.1 můžeme vidět několik příkladů složitostí pro různé množství vstupních informací.

	O(1)	O(N)	O(N log N)	O(N^2)	O(N^3)	O(2^N)
1	1	1	1	1	1	2
10	1	10	10	100	1000	1024
100	1	100	200	10000	1000000	10^30
1000	1	1000	3000	1000000	10^9	10^300
1000000	1	1000000	6000000	10^12	10^18	10^3000

Tabulka 3.1: Některé příklady složitostí

Zjednodušeně by se dal algoritmus připodobnit k receptu na vaření, kde pomocí přesného postupu dostaneme ze vstupních ingrediencí požadovaný výsledek.

3.3 Evoluční algoritmy

Evoluční algoritmy se snaží pomocí matematických postupů zachytit evoluční modely procesů odehrávajících se v přírodě.(17) Slouží k tomu obecné poznatky o fungování evoluce, jak je popsali přední světoví genetici a biologové Charles Darwin, Alfred Russel Wallace nebo Gregor Mendel.

První větší pokus o vysvětlení evoluce provedl francouzský přírodovědec Jean Baptiste Lamarck. Ten zastával názor, že vlastnosti, které jedinec získá v průběhu svého

života, jsou dědičné a přenášejí se na následující generaci.(23) Tento názor je v současnosti širokou vědeckou obcí odmítán.

Darwin vysvětloval evoluci na základě života komunity, která se potýká s omezenými zdroji a jedinci jsou pak nuceni spolu o tyto zdroje soupeřit. Pokud je jedinec neúspěšný při získávání zdrojů, zemře nebo je u něho snížena schopnost předat své genetické informace potomkům. Jedinci, kteří jsou v získávání zdrojů úspěšnější než ostatní, mají šanci na předání svých genetických informací potomkům vyšší. Potomci úspěšnějších jedinců začnou postupně vytlačovat potomky méně úspěšných až nakonec méně úspěšní vymizí. Toto vysvětlení evoluce si získalo pozitivnější přijetí, než které se dostalo Darwinovu předchůdci. Stále ale přetrvávalo několik nezodpovězených otázek a pochybností.

Společnými rysy Darwinovy teorie jsou:(17)

- Jedinci v populaci jsou geneticky variabilní a proměnlivost je vzhledem k prostředí náhodná.
- Množství jedinců populace není omezeno schopností růstu, ale omezenými dostupnými zdroji. Proto se reprodukčního věku dožívají jen nejschopnější jedinci – jedná se o „boj o přežití“.
- Četné potomstvo mají hlavně nejlépe vybavení jedinci, kteří tak přenášejí své genetické informace do následující generace ve zvýšené míře.
- Tímto mechanismem se druhy organismů přizpůsobují prostředí, které se navíc může postupně měnit.

Matematické evoluční modely jsou charakteristické několika společnými rysy. Pracují s množinou potenciálních řešení, ze které se vybírá podmnožina nejlépe vyhovujících řešení. Ta se v další iteraci vhodně doplní řešeními, které vzešly z operací mutace a křížení. Proces se opakuje podobně, jako se v přírodě střídají generace.

3.3.1 Genetické algoritmy

Genetické algoritmy (GA) jsou pravděpodobně nejrozšířenějším typem evolučních algoritmů. Mezi průkopníky a popularizátory GA patří John Holland, jež v roce 1975

uveřejnil práci Adaptation in Natural and Artificial Systems, které se dostalo poměrně velké pozornosti vědeckých kruhů. GA byly v této publikaci prezentovány jako účinný prohledávací mechanismus pro adaptivní systémy umělé inteligence.(17) V knize definoval jeden ze základních znaků GA, kterým je křížení a dále méně významná inverze.

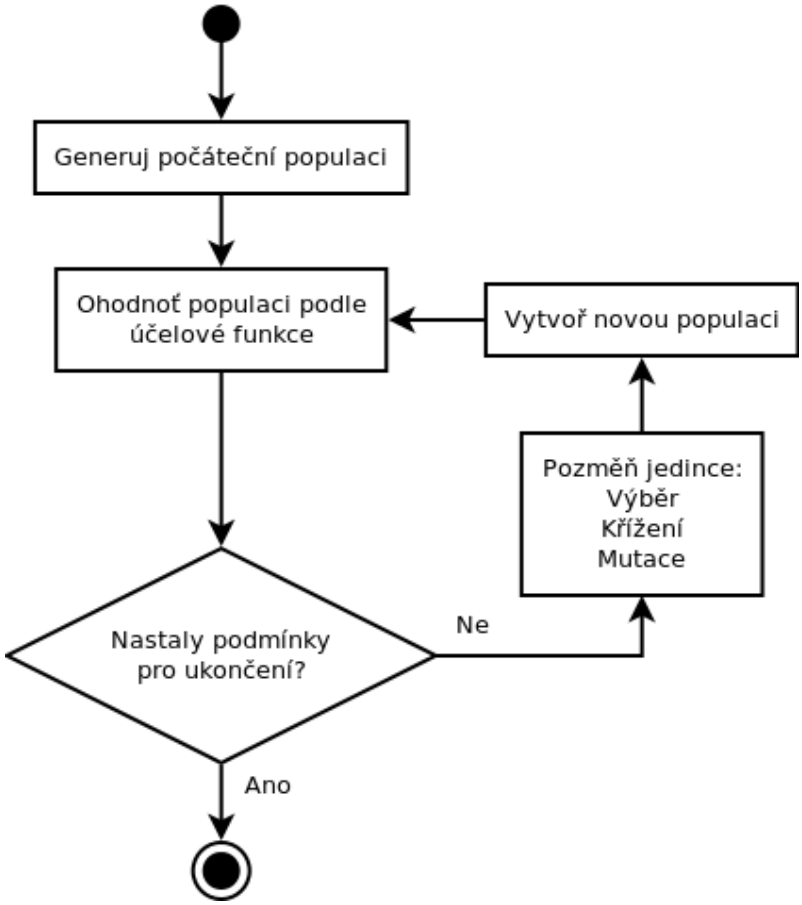
Již tak nezanedbatelný zájem o genetické algoritmy akceleroval student Johna Hollanda David Goldberg, který v roce 1989 vydal publikaci Genetic Algorithms in Search, Optimization, and Machine Learning. Čtenář zde nalezne detailně popsané GA včetně příkladů implementace. Od této doby také mluvíme o oblasti vědy zvané evolutionary computing, která je součástí soft computingu zahrnujícího také neuronové sítě a fuzzy logiku.

GA doznaly takto velkého rozšíření díky schopnosti řešit velmi složité úlohy, které by se pomocí jiných numerických metod řešily velmi obtížně nebo by je nešlo v rozumném časovém úseku vyřešit vůbec.

Pokud hovoříme o klasické jednoduché verzi genetického algoritmu, jsou za jeho prvky považovány:(17)

- Reprezentace jedince v populaci je binární.
- Ke křížení dochází pouze v jednom bodě.
- Mutace je prováděna pouze na jednom bitu.
- Proporcionální výběr rodičů podle hodnoty účelové funkce.
- Obnova populace proběhne zcela právě za dobu jedné generace.

Schéma průběhu genetického algoritmu můžeme vidět na obrázku 3.1.



Obrázek 3.1: Průběh GA

3.3.1.1 Definice pojmů

Jedince v populaci označujeme jako chromozóm nebo genom. Ten nese genetickou informaci v podobě binárního vektoru o konstantní délce *n*. Tyto prvky chromozómu nazýváme gen. Hodnota, kterou může nabývat gen je Alela.

$X=(X_0, X_1, \ldots, X_{n-1})$, kde X_i je i-tou proměnnou řetězce

$x=(x_0, x_1, \ldots, x_{n-1})$ je řetězec konkrétních instancí proměnných

$X_i=x_i, x_i \in \{0,1\}$

$D=(X^1, X^2, \ldots, X^N), X^j \in D$ je množina N řetězců, která specifikuje populaci D

Účelovou funkci definovanou nad množinou binárních vektorů o délce *n* označíme

$f:\{0,1\}^n \rightarrow \mathbb{R}$

Tato funkce ohodnotí každý chromozóm výsledkem v podobě reálného čísla.

Cílem je nalezení extrému funkce f . Podle charakteru účelové funkce volíme hledání minima nebo maxima.

Genetický algoritmus můžeme neformálně popsat jako posloupnost následujících kroků:

1. Nastav čítač generací na hodnotu $t=0$, generuj počáteční populaci $D(0)$ velikostí N .
2. Ohodnoť jedince populace $D(t)$ pomocí funkce $F(X)$.
3. Vygeneruj generaci potomků $U(t)$ s velikostí $M \leq N$ pomocí křížení a mutace.
4. Vytvoř novou populaci $D(t+1)$, která vznikne z $D(t)$, kde se nahradí odpovídající jedinci potomky $U(t)$.
5. Inkrementuj t .
6. Pokud není splněna podmínka pro ukončení (dosažen počet generací, nepřekročeno minimální mezigenerační zlepšení hodnotící funkce atd.), pokračuj bodem dva.

3.3.1.2 Vytvoření populace

Počáteční populace bývá generována náhodně, ale někdy je vytvořena rychlou heuristickou funkcí, která vytvoří jedince soustředěné blíže k hledanému řešení. Taková populace rychleji dospěje k nějakému řešení, ale riziko předčasné konvergence je vyšší. Množství jedinců v populaci je zásadní parametr genetického algoritmu a experimentální pokusy naznačují vhodnou velikost populace v intervalu $<n, 2n>$, kdy n představuje délku binární reprezentace chromozómu.

3.3.1.3 Obnova populace

Pro obnovu populace se používají dva základní způsoby:

1. Generativní GA s úplnou obnovou – rodiče vždy zcela vymřou a nedostanou se do následující generace.
2. GA s částečnou obnovou – nahrazen je pouze jeden nejslabší jedinec populace.

Často se však využívá jistá kombinace těchto metod, kdy může být nahrazováno jen několik málo desítek procent staré populace.

Při vytváření populace nebo její obnově bývá nutné brát v potaz omezující podmínky. Těmi mohou být např. zamezení vzniku duplicitních jedinců nebo neakceptování jedinců v určitých definovaných intervalech. Stejného cíle můžeme dosáhnout zakomponováním odpovídajícího postihu nevyhovujících jedinců do účelové funkce.

3.3.1.4 Výběr

Operátor výběru slouží k vytvoření nové populace $P(t+1)$. Výběr se provádí z předešlé populace $P(t)$ s možným opakováním. Nejčastějším způsobem výběru jedinců je podle výsledku jejich hodnotící funkce. Čím vyšší hodnota (v případě hledání maxima), tím větší pravděpodobnost výběru.

Při výběru jedinců budoucí populace existuje několik možných rizik, které by mohly zhatit správný vývoj populace a tím by GA konvergoval k nevyhovujícímu lokálnímu řešení nebo by konvergoval příliš pomalu. Proces výběru musí dostatečně upřednostňovat více vyhovující jedince, na druhé straně musí vybírat populaci pestrou.

Parametr vyjadřující toto chování se nazývá Selekční intenzita nebo také Selekční tlak a můžeme si ho vyjádřit jako podíl rozdílu průměrné hodnoty účelové funkce po výběru s průměrnou hodnotou účelové funkce před výběrem a rozptylem účelových hodnot před selekcí.(17)

$$I = \frac{\overline{M*} - \overline{M}}{\overline{\sigma}}$$

$\overline{M*}$ průměrná hodnota účelové funkce po výběru

$\overline{M}$ průměrná hodnota účelové funkce před výběrem

$\overline{\sigma}$ rozptyl hodnot účelové funkce před výběrem

Pro počet generací, za kterou je schopný genetický algoritmus změnit všechny chromozómy původní populace na novou složenou z lepších jedinců bez použití křížení a mutace, se používá označení *doba převládnutí*. Čím delší je, tím je menší riziko předčasné konvergence k lokálnímu optimu.

Při výběru nové populace je důležitým parametrem *ztráta různorodosti chromozómů*. K této ztrátě dochází při vylučování jedinců s nízkou hodnotou účelové funkce a opět se tím zvyšuje riziko předčasné konvergence k lokálnímu optimu. Ztrátu různorodosti vyjadřujeme poměrem nevybraných chromozómů ku celkovému počtu chromozómů. Při hodnotě blízké nule dochází k pomalé konvergenci a naopak při hodnotě blízké jedné dochází k předčasné konvergenci díky ztrátě různorodosti.

Proporcionální výběr – Jedná se o nejčastěji používaný způsob výběru nové generace podle hodnoty účelové funkce. Pravděpodobnost výběru je rovna poměru hodnoty účelové funkce ku součtu hodnot účelových funkcí všech chromozómů.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Existují modifikace tohoto způsobu výběru za účelem znevýhodněním jedince s příliš dobrým ohodnocením účelové funkce. Toho se může dosáhnout např. tzv. *komprimací účelové funkce*, kdy se ke každé její hodnotě přičte konstanta, která u podprůměrných jedinců zvýší podíl ohodnocení oproti celku a u nadprůměrných jedinců sníží.(17)

Výběr odseknutím – Tato metoda výběru je velmi jednoduchá a skládá se ze setřídění jedinců v populaci a následného výběru jen těch s nejlepším ohodnocením. Z pole jedinců jsou odseknuti všichni ti, kteří se nedostali do skupiny postupující do další generace. Poměrně velkou nevýhodou této metody výběru je velká ztráta genetického materiálů populace.(2)

Lineární uspořádání – Stejně jako v předchozí metodě vycházíme ze setříděné populace, kdy nejhoršího jedince umístíme na první pozici $i = 1$ a nejlepšího na poslední $i = N$. Vzorec pro výpočet pravděpodobnosti výběru jedince z pozice i je následující:(2)

$$p_i = \frac{1}{N} * \eta^- + (\eta^+ + \eta^-) * \frac{i-1}{N-1} \qquad i \in \{1, 2, \ldots, N\}$$

vztah $\frac{\eta^-}{N}$ vyjadřuje pravděpodobnost výběru nejhoršího jedince a $\frac{\eta^+}{N}$ udává

pravděpodobnost výběru nejlepšího jedince. Nevýhodou tohoto uspořádání je omezený selekční tlak.

Exponenciální uspořádání – Představuje vylepšení lineárního uspořádání v podobě rozložení pravděpodobnosti s exponenciální závislostí. Toto uspořádání může být aplikováno podle následujících vzorců:(2)(17)

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}} \quad , \quad i \in \{1,2,..,N\} \quad \text{, nebo}$$

$$p_{\exp-rank}(i) = \frac{1-e^{-i}}{c} \quad , \quad i \in \{1,2,..,N\}$$

Přičemž konstanta $c \in (0,1)$.

Výhodou exponenciálního uspořádání je možnost alokování dvou instancí nejlepšího jedince. Tento algoritmus dosahuje nejlepších výsledků a pomocí vhodného parametru c můžeme dosáhnout požadované selekční intenzity a neztratit přitom různorodost populace. Časová složitost algoritmu pro vysoké N pak odpovídá $O(N \ln N)$.

Turnajový výběr – Pro svoji jednoduchost a uspokojivé výsledky je další velmi často používanou metodou výběru, kdy z celé populace náhodně vybereme skupinu několika jedinců a z nich vybereme nejlepšího, který postupuje do následující generace. Toto opakujeme tolikrát, kolik má následující generace jedinců. Dojde tím k jejímu naplnění.

3.3.1.5 Křížení

Operátor křížení je základním mechanismem v evoluci a představuje proto důležitý způsob změny jedinců v genetických algoritmech. Operace křížení je podrobována diskuzím o užitečnosti, protože mění velké části chromozómů a tím rozbíjí ověřené části bitové posloupnosti. Pro křížení hovoří probíhající výměna informací mezi jedinci, která se v přírodě běžně děje. V některých případech je možné nahradit křížení operací mutace s velmi malou pravděpodobností.

Podle teorie stavebních bloků jsou GA schopny identifikovat dobré části

chromozomu a s těmi pak pracují jako s nedělitelnou jednotkou. Tyto nedělitelné bloky pak dále sestavují ve větší bloky a vytváří tím hledané řešení. Proces křížení nenastává vždy, ale jen s určitou pravděpodobností.

Výběr křížících se rodičů probíhá většinou zcela náhodně, ale existují i varianty, kdy jsou rodiče vybíráni podobnými způsoby jako jedinci, kteří přímo postupují do další generace.

Křížením N rodičů může vznikat M potomků. Nejčastěji používaným způsobem bývá jednobodové křížení dvou rodičů a vznik jednoho nebo dvou potomků. V případě dvou potomků tito nesou opačné části genetických informací rodičů. Při inspiraci přírodou by výsledek křížení byl v drtivé většině jeden potomek a s velmi malou pravděpodobností dva a více potomků. V takovém případě by většinu sourozenců tvořili rozdílní jedinci, s jinými body křížení.

Důsledkem křížení jedinců může být zlepšení hodnoty účelové funkce a tím se posunujeme k lepší skladbě populace. Volba druhu křížení je závislá na charakteru řešené úlohy a nelze obecně doporučit jeden nejlepší způsob.

Jednobodové a vícebodové křížení – Představuje nejjednodušší způsob křížení a vychází z procesu uskutečňovaného v přírodě.

Jednobodové křížení probíhá mezi dvěma rodiči, kde je náhodně zvolen bod křížení v rozmezí $0..n-2$. Potomek pak bude obsahovat proměnné prvního rodiče až do bodu křížení a zbytek proměnných převezme od druhého rodiče. V tabulce 3.2 můžeme vidět příklad křížení s bodem křížení 2. Při jednobodovém křížení může být výsledkem jeden nebo dva potomci.

Rodič 1	1	1	1	1	1	1	1	1
Rodič 2	0	0	0	0	0	0	0	0
Potomek	1	1	1	0	0	0	0	0

Tabulka 3.2: Jednobodové křížení

Vícebodové křížení je obdobný způsob jako jednobodové, pouze s rozdílem v množství bodů křížení. Ty jsou náhodně vybrány ze stejného intervalu $0..n-2$, přičemž v bodě křížení dochází ke změně rodiče, od kterého se získává genetická

informace pro potomka. V tabulce 3.3 můžeme vidět vícebodové křížení se zvolenými body křížení 0, 2 a 5. Tímto druhem křížení můžeme vytvořit jednoho až $2^{k+1}-2$ různých potomků, přičemž k vyjadřuje počet bodů křížení a potomkem nemůže být jedinec identický s rodičem.

Rodič 1	1	1	1	1	1	1	1	1
Rodič 2	0	0	0	0	0	0	0	0
Potomek	0	1	1	0	0	0	1	1

Tabulka 3.3: Vícebodové křížení

Používaných metod křížení bychom mohli nalézt ještě více. Zajímavým postupem je křížení používané při řešení problému obchodního cestujícího, kde v chromozomu vybereme náhodné pozice a genům mezi pozicemi změníme pořadí ze vzestupného na sestupné.

Jednotné křížení – Také nazývané jako uniformní je alternativní způsob provádění operace křížení. Algoritmus prochází jednotlivé binární prvky (geny) chromozómu a s pravděpodobností p_u provede jejich výměnu. Tento způsob křížení má řadu odpůrců, kteří poukazují na příliš velký zásah do struktury chromozómu, kde jsou s velkou pravděpodobností poškozeny stavební bloky. Jako obhajoba této metody křížení by mohlo sloužit vnášení poměrně velké různorodosti do populace. To předurčuje uniformní křížení pro použití na velmi složité funkce s mnoha lokálními extrémy. Zabrání se tím předčasné konvergenci. Příklad uniformního křížení můžeme vidět na následující tabulce 3.4, kde došlo ke změně genu z 50%.

Rodič 1	1	1	1	1	1	1	1	1
Rodič 2	0	0	0	0	0	0	0	0
Potomek	1	1	0	1	0	0	1	0

Tabulka 3.4: Uniformní křížení

3.3.1.6 Mutace

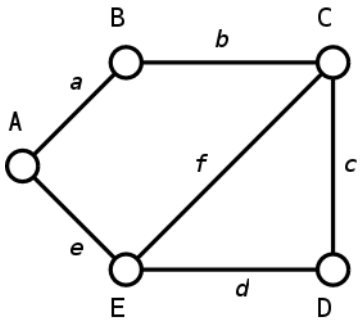
Tento reprodukční operátor zasahuje do genetické informace populace velmi málo, ale jeho význam je velký. Operace modifikuje binární prvky chromozómu s pravděpodobností p_m . Nejčastěji se můžeme setkat s jednoduchou bitovou negací, která mívá pravděpodobnost v rozmezí 0,0005 až 0,01.(17)

V přírodě má mutace na jedince zásadní vliv a stejně je tomu u genetických

algoritmů. Mutace přináší do populace potřebnou různorodost, na druhé straně má často na „mutanty“ fatální vliv v podobě výrazného zhoršení účelové funkce. Příliš vysoká pravděpodobnost mutace p_m má za následek nestabilitu vývoje a malá nepřináší do populace potřebnou různorodost.

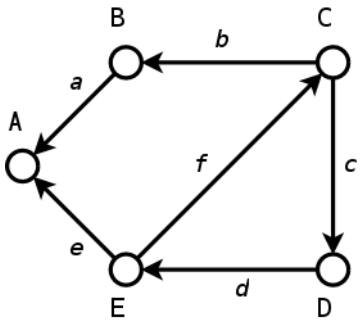
3.4 Graf podle teorie grafů

Neorientovaným grafem podle teorie grafů jak je uvedena v (13) je myšlena množina vrcholů V a množina hran H taková, že každá hrana $h \in H$ je přiřazena neuspořádané dvojici (tj. dvouprvkové množině) vrcholů $u, v \in V$. Pokud existuje hrana $h \in H$ přiřazená dvojici vrcholů $u, v \in V$, píšeme $h \equiv \{u, v\}$. Pokud je jedné dvojici přiřazeno více hran, nazýváme hrany jako vícenásobné. Neorientovaný graf můžeme vidět na obrázku 3.2.



Obrázek 3.2:
Neorientovaný graf

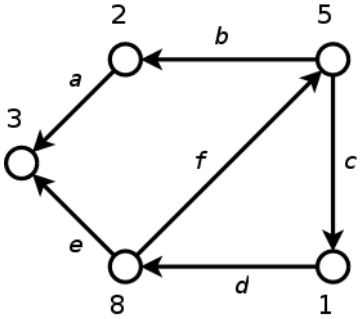
Podobně, orientovaným grafem je myšlena množina vrcholů V a množina hran H taková, že každá hrana $h \in H$ je přiřazena uspořádané dvojice $(u, v) \in V \times V$ vrcholů $u, v \in V$. Existuje-li jediná hrana $h \in H$ přiřazená dvojici (u, v) vrcholů, píšeme $h \equiv (u, v)$. Hrany, u níž jsou prvky uspořádané nebo neuspořádané dvojice totožné, označujeme smyčkou.(13) Orientovaný graf můžeme vidět na obrázku 3.3.



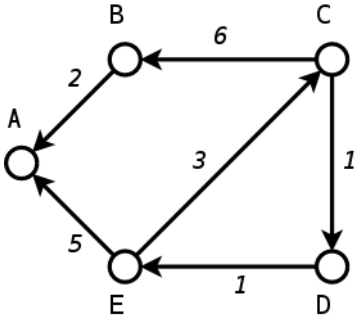
Obrázek 3.3:
Orientovaný graf

Oba grafy se skládají z vrcholů $V=\{A,B,C,D,E\}$ a množiny hran $H=\{a,b,c,d,e,f\}$. U neorientovaného grafu vidíme například hrany $a\equiv\{A,B\}$ nebo $b\equiv\{B,C\}$. U orientovaného existují např. hrany $a\equiv(B,A)$ nebo $b\equiv(C,B)$.

Máme-li graf s množinou vrcholů V a množinou hran H a dále $f:V\to\mathbb{R}$ a $g:H\to\mathbb{R}$ jsou zobrazení, pak f nazýváme vrcholovým ohodnocením a g hranovým ohodnocením grafu. Ukázky takových grafů můžeme vidět na obrázcích 3.4 a 3.5.



Obrázek 3.4: Vrcholově ohodnocený graf



Obrázek 3.5: Hranově ohodnocený graf

3.4.1 Dijkstrův algoritmus

Tento algoritmus je jedním z nejznámějších grafových algoritmů a slouží k nalezení nejkratší cesty v ohodnoceném grafu. Jeho autorem je nizozemský vědec, učitel a laureát Nobelovy ceny Edsger Wybe Dijkstra.

Jedná se o konečný algoritmus pracující nad hranově ohodnocenými grafy, pomocí

kterého jsme schopni nalézt délku nejkratší cesty z počátečního vrcholu do všech ostatních. Ohodnocení hran grafu musí být nezáporné. Široké uplatnění nachází při hledání optimálních dopravních tras, kdy disponujeme grafem (sítí) všech potenciálně využitelných ohodnocených cest. Jediný problém, který v takovém případě vyvstává, je přesnost grafu. Časová náročnost algoritmu je $O(|V|^2+|E|)$, přičemž V vyjadřuje počet vrcholů grafu a E značí počet hran grafu.(13) Vstupními informacemi algoritmu jsou orientovaný spojitý graf, počáteční bod grafu a koncový bod grafu. Pokud algoritmus nalezne cestu mezi těmito dvěma body, můžeme výpočet ukončit, protože ve zbylé části grafu již neexistuje kratší cesta než je právě nalezená.

Neformální popis algoritmu:

1. U počátečního bodu nastavíme vzdálenost k počátku na nulovou hodnotu. U ostatních bodů je hodnota nedefinovaná.
2. Nastavíme stav každého bodu jako dočasný.
3. Z dočasných bodů vybereme takový bod v , který má nejnižší definovanou hodnotu.
4. Prohlásíme bod v za trvalý a změníme mu stav z dočasný na trvalý.
5. Pokud mají sousední vrcholy ohodnocení vyšší než cesta přes vrchol v nebo mají ohodnocení nedefinované, aktualizujeme jejich ohodnocení na délku cesty přes vrchol v .
6. Pokud není koncový vrchol označen jako trvalý a všechny ostatní vrcholy ještě nejsou označeny jako trvalé, pokračujeme bodem 3.

3.5 MATLAB

Pro řešení úlohy mé diplomové práce využiji programové prostředí MATLAB, které poskytuje bohaté zázemí pro tvorbu aplikací řešících složité numerické a simulační úlohy. Budu používat funkce obsažené hlavně v balíku Global Optimization Toolbox, který je určen k řešení optimalizačních úloh.

3.5.1 Historie

Původní práce na předchůdci softwaru MATLAB začala již v sedmdesátých letech, kdy Cleve Moler, tehdy učitel na americké University of New Mexico, začal vytvářet programové vybavení pro své studenty. Tento software jim ulehčoval práci s knihovnami pro numerické výpočty, které byly vytvořeny v jazyce Fortran. Brzy se softwarový balík rozšířil i na další univerzity, a tak bylo jen otázkou času, kdy dojde ke komerčnímu využití produktu. To přišlo v roce 1984 v podobě založení společnosti MathWorks Inc. a komerčnímu nabízení balíku pod jménem MATLAB.(22)

Ve svých počátcích byl MATLAB velmi limitován hardwarovými prostředky tehdejší doby. Jistá limitace trvá až do dnešních dní, ale zdaleka už nesnižuje komfort práce tak, jako tomu bylo dříve.

3.5.2 Global Optimization Toolbox

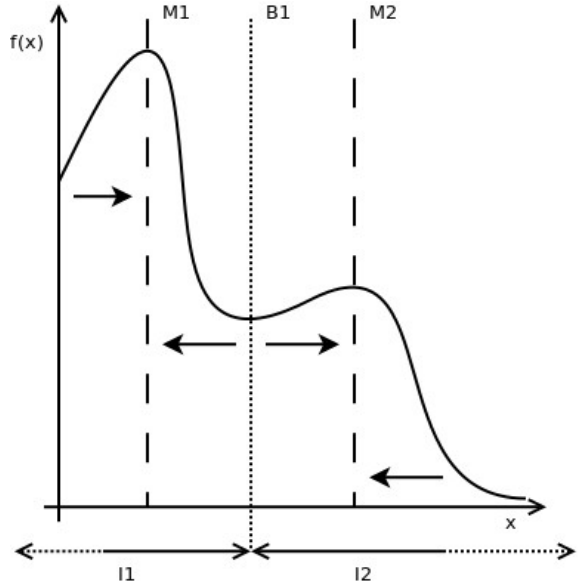
Tato knihovna funkcí programového prostředí MATLAB poskytuje prostředky, které hledají řešení problémů specifikovaných funkcemi s více extrémy. Tyto prostředky zahrnují skupiny Global search, MultiStart search, Pattern search, Simulated annealing search a Genetic algorithm search. Genetic algorithm search disponuje možností definování si vlastních funkcí pro vytvoření populace, hodnotící funkce a také funkce pro výběr rodičů, křížení a mutace. Těchto vlastností využijeme při návrhu algoritmu.

3.5.2.1 Globální optimalizace

Globální optimalizace má za cíl nalezení globálního optima. Toto optimum může být představováno minimem nebo maximem určité funkce. Při řešení takové úlohy narážíme na problém nalezení zdánlivého globálního optima, které je ve skutečnosti jen lokálním optimem. Pro hledání optim v intervalech, kde se nachází právě jedno, se v prostředí MATLAB využívá Optimization Toolbox .

Na obrázku 3.6 máme vyznačen rozdíl mezi globálním (M1) a lokálním (M2) maximem. Tyto maxima považujeme za globální a lokální optima, pokud hledáme nejvyšší ohodnocení funkce f(x). Optimalizační metody konvergují k optimům M1 a M2 právě tehdy, nachází-li se jejich výchozí body v intervalech I1 resp. I2. Směr konvergence je znázorněn plnými šipkami. Pokud neexistuje ještě další lokální

maximum funkce f(x), intervaly I1 a I2 nejsou omezené. Hranice mezi „spádovými oblastmi“ je označena jako B1. Při dvou rozměrech tvoří tyto oblasti plocha, při třech prostorový objekt atd.



Obrázek 3.6: Globální a lokální maximum

3.5.2.2 Charakteristiky optimalizačních metod

Počáteční body – Většina řešících algoritmů vyžaduje jako vstupní parametr počáteční body. Jedním z důvodů bývá zjištění rozměrů a dalších vlastností hledaného řešení. Genetické algoritmy sice tyto počáteční body řešení nevyžadují, ale vytvoří si je samy při generování počáteční populace. Pouze vyžadují zadání rozsahu řešení, aby vygenerovaná populace pokrývala všechny prohledávané oblasti rovnoměrně.

Iterace – Vyjadřuje jeden průchod optimalizačním algoritmem. Iterace některých optimalizačních metod využívajících náhodná čísla označujeme, stejně jako u kroku některých algoritmů, jako nedeterministické. Do těchto optimalizačních metod patří ty, které jsou založené např. na genetických algoritmech.

Gradient – Některé optimalizační metody používají svůj vlastní odhad nebo uživatelem poskytnuté informace pro výpočet hodnoty v další iteraci algoritmu. Tento odhad může být založen např. na vyšetření vlastností funkce v aktuálním nejlepším nalezeném řešení.

Konvergence – Řešící algoritmus může selhat, pokud mu je jako počáteční bod

zvolena pozice příliš daleko od optimálního řešení. Optimalizační metody založené na práci se sklonem vyšetřované funkce mohou při jejím vhodném tvaru a blízkému počátečnímu bodu konvergovat velmi rychle. Simulated annealing search a Genetic algorithm search nemusí pro některá zadání konvergovat dostatečně rychle.

Metody hledání globálních optim obsažené v Global Optimization Toolbox používají k řešení odlišné přístupy:

- Global search a MutiStart search na počátku vytvoří množství výchozích pozic, které pak pomocí lokálních vyhledávacích algoritmů zlepšují až naleznou lokální optima.
- Genetic algorithm search rovněž vytvoří množství výchozích pozic, které se v tomto případě nazývají populace, a ty postupně zlepšuje. Populace je sice generována náhodně, ale lze předpokládat, že pokrývá několik intervalů s lokálními optimy, a proto tento způsob nalézá globální řešení.
- Simulated annealing search využívá způsob náhodného hledání, kdy zvažované řešení porovnává s nejlepším dosud nalezeným. Pokud je zvažované lepší, přijme ho algoritmus jako nejlepší řešení. Aby se zabránilo uvíznutí v intervalu s jedním lokálním optimem, občas algoritmus přijme i horší řešení.
- Pattern search vždy, než přijme určité řešení, prozkoumá i jeho okolí. Pokud v okolí existuje neoptimální řešení, které konverguje k jinému optimu, prozkoumá i interval náležící k jinému intervalu.

3.5.2.3 Porovnání optimalizačních metod

Z porovnání, které je uvedeno v (19) vyplývají následující vlastnosti optimalizačních postupů.

Multistart – Rychle nalezne lokální řešení v očekávaném intervalu, ale nepodává dobré výsledky při hledání mimo něj. Má jednoduchý způsob použití.

Pattern search – Provede více vyhodnocení než Multistart a hledá v několika intervalech. Díky tomu nalézá lepší řešení než pomocí předchozí metody. Způsob

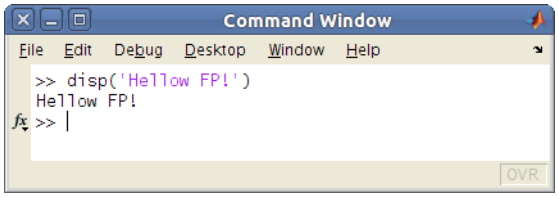
použití je obdobný.

Genetic algorithm search – Provede o mnoho více vyhodnocení než Pattern search. Šance dosáhnutí lepších výsledků než u předchozí metody je vyšší, ale kvůli nedeterministickému charakteru algoritmu není možné garantovat úroveň výsledku. Způsob použití není složitý a dává prostor pro zvýšení úspěšnosti hledání díky modifikacím funkcí zajišťujících operátory GA jako je např. vytváření populace.

Global search – Vyhodnotí mnohem více řešení než Pattern search. Prohledá mnoho intervalů a dosáhne proto také velmi dobrých výsledků. Nastavení algoritmu Global search je náročnější než u jiných řešících metod.

3.5.3 Tvorba aplikací v prostředí MATLAB

MATLAB je schopen provádět příkazy zadávané do speciálního řádku označovaného jako Command Window viz. Obrázek 3.7. Při jednom spuštění je možné zadat více středníkem zakončených příkazů současně. Tím se potlačí výpis návratové hodnoty příkazu.



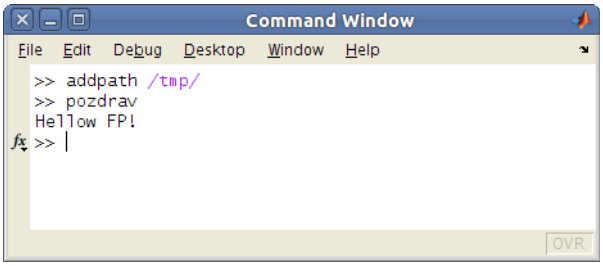
Obrázek 3.7: Command window

Při větším množství příkazů se tento způsob zadávání stává nepřehledným a proto existuje možnost zapisovat příkazy do zvláštních souborů, kterým se říká m-soubory. (20) M-soubor může obsahovat buď posloupnost příkazů, definici funkce nebo definici třídy. Pokud jméno první uložené funkce souhlasí se jménem souboru, lze tento soubor použít jako vstupní bod při spouštění složitějších programů. Jedná se tedy o obdobu funkce *main* používané v jiných jazycích. Pokud jméno funkce nesouhlasí se jménem souboru, MATLAB použije první nalezenou deklaraci funkce, přičemž ignoruje uvedený název a doplní místo něj jméno souboru bez přípony. Tento způsob spouštění sice funguje, ale rozhodně se nejedná o doporučeníhodný postup.

Hlavní výhodou m-souborů je archivace posloupnosti příkazů do trvalé paměti

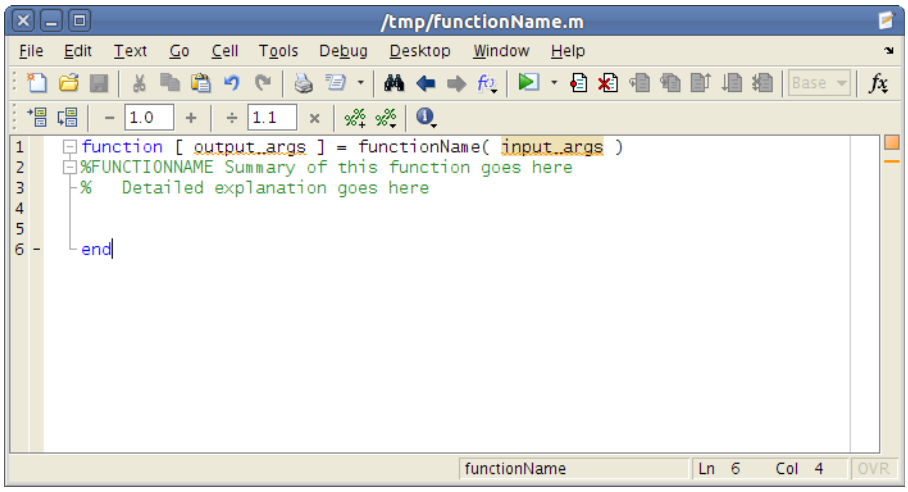
počítače a umístění těchto příkazů na uživatelem definovaná místa. Tím vzniká možnost provázání m-souborů za účelem vzniku složitějšího dobře strukturovaného programového celku.

Abychom mohli m-soubory spouštět pouhým napsáním jeho jména do příkazového okna, musí být umístěn v aktuálním adresáři nebo v některé z cest zadaných v proměnné *path*. Ukázku spuštění funkce můžeme vidět na obrázku 3.8, kde jsme přidali umístění souboru pozdrav.m do proměnné *path* a poté ho spustili bez zadávání absolutní cesty.



Obrázek 3.8: Spuštění funkce

M-soubory jsou ve skutečnosti obyčejné textové soubory a pro jejich tvorbu můžeme použít jakýkoliv textový editor. Doporučit lze jakýkoliv, který podporuje alespoň zvýrazňování syntaxe, ale nejlepší volbu nejspíš představuje integrovaný MATLAB editor (M-editor) zobrazený na obrázku 3.9. Kromě zvýrazňování syntaxe disponuje snad ještě užitečnější vlastností ladění kódu programu anglicky označované jako Debugger. Tato vlastnost se skládá z možnosti vkládat do kódu body pozastavení provádění algoritmu a ručního krokování programu. V okamžiku pozastavení má uživatel dostupné hodnoty všech aktuálně definovaných proměnných, díky čemuž může ověřit jejich správnost. Další vlastností, kterou ostatní textové editory nedisponují, je analýza zdrojového kódu, na jejíž základě editor nabízí změny v kódu jako je např. nepotřebné definování proměnné nebo neefektivní změna velikosti proměnné v každém prováděném cyklu.



Obrázek 3.9: MATLAB editor

3.5.3.1 Funkce

Jak jsme si naznačili výše, funkce slouží k rozdělení skriptu a jeho příkazů do přehlednějších bloků kódu. Funkcí se v jednom m-souboru může nacházet více, ale jen ta první, která by měla nést jméno stejné jako název souboru bez přípony, je použita pro počáteční bod programu a je přímo přístupná z ostatních souborů. Ostatní definované funkce přístupné nejsou a mohou být volány pouze v rámci svého souboru.

Každá funkce může přijímat vstupní parametry a zpět předávat výstupní. Funkce využívá v průběhu svého provádění vlastní prostor pro proměnné, do kterého nezasahují proměnné prostředí. Využívá tzv. lokální proměnné. Tím se liší od skriptu, který pracuje přímo s proměnnými prostředí (globální proměnné). V následujícím příkladu vidíme definici funkce *myfun* se vstupními parametry *in1* a *in2* a výstupními parametry *out1* a *out2*.(18)

```
function [out1, out2, ...] = myfun(in1, in2, ...)
```

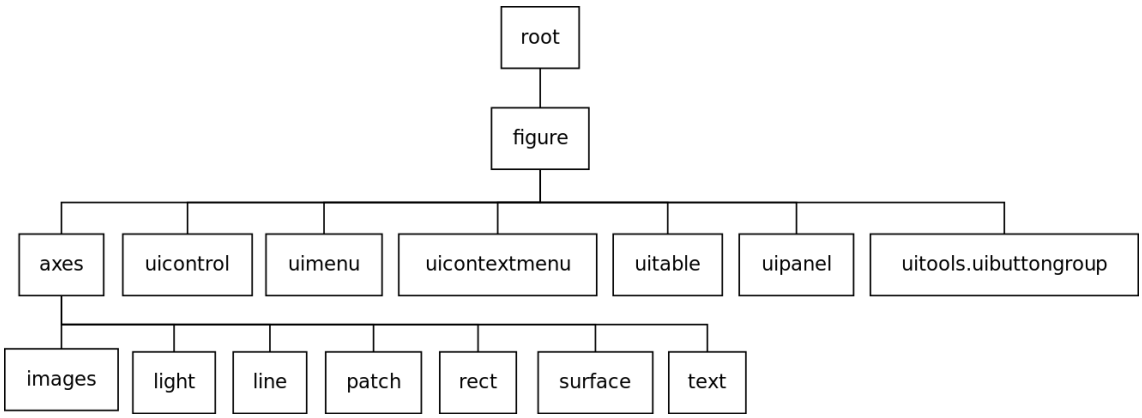
Takto definovaná funkce se sestává z veškerého kódu uloženého za deklarací až po znak konce souboru. Pokud bychom v souboru deklarovaly další funkci, předchozí tím ukončíme. Pro přehledné ukončování bloků kódu se doporučuje využívat rezervované slovo *end*. Při použití jakékoliv vnořené funkce, se použití *end* stává povinné pro všechny bloky existujících funkcí.

Blok příkazů funkce musí přiřadit všem výstupním parametrům hodnotu. Pokud

potřebujeme předávat nebo přijímat proměnný počet argumentů, využijeme k tomu klíčová slova *varargin* a *varargout*. Ty se zapisují jako poslední parametr ve výčtu argumentů a zastupují všechny volitelné argumenty.

3.5.3.2 Grafické objekty

Od svého vzniku se stal MATLAB velmi komplexním balíkem a uživatelům nabízí i možnost tvorby grafického uživatelského prostředí (GUI) pro zvýšení komfortu práce s vytvořenými aplikacemi. Toho využiji i v této práci a proto se seznámíme se základními způsoby zacházení se systémem grafických objektů Handle Graphics. Na obrázku 3.10 můžeme vidět nekompletní hierarchický strom grafických objektů. Hlavním objektem je *root*, který je jedinečný a může být rodičem dalších objektů. Každý existující grafický objekt může být jednoznačně identifikován pomocí ukazatele „handle“. Ukazatel je ve své podstatě celé číslo rozsahu Integer. Díky výsadnímu postavení objektu *root*, který představuje obrazovku samotného počítače, mu je přidělena hodnota ukazatele rovna nule. Tím je také vyřešen jednoduchý a neměnný způsob adresování pomocí absolutní hodnoty. Díky vlastnostem *Parent* a *Children*, kterými disponuje každý objekt, dochází k jejich provázanosti. *Parent* uchovává ukazatel na předchůdce a *Children* uchovává seznam ukazatelů na následníky. Objekt *root* z pochopitelných důvodů nedisponuje vlastností *Parent*. Vzniká tím stromová struktura podobná té na obrázku 3.10.



Obrázek 3.10: Schéma grafických objektů [8] + doplnění

Objekt *root* může mít potomky v podobě grafických oken *figure*. Do nichž jsou vykreslovány další uživatelem definované grafické prvky:(18)

- **Axes** – slouží ke grafickému znázornění hodnot a představuje jeden z nejdůležitějších grafických objektů. V rámci Axes můžeme používat jaderné grafické objekty tzv. základní kreslící primitivy.
- **Uicontrol** – sdružuje grafické ovládací prvky jako např. tlačítko, zaškrtávací políčko, přepínací tlačítko nebo posuvník.
- **Uimenu** – slouží pro vytváření položky v horním programovém menu. Objekt odvozený od *figure* vytvoří dolu vysouvací hlavní položku menu a objekt odvozený od *uimenu* vytvoří do boku vysouvací podmenu.
- **Uicontextmenu** – je určeno k vytváření kontextového menu grafických objektů.
- **Uitable** – komponenta tvoří základ pro tabulkové zobrazení a editování údajů podobně jako se editují a zobrazují údaje v tabulkových procesorech.
- **Uipanel** – vytváří ohraničenou oblast, do níž mohou být umisťovány další grafické prvky.
- **Uitools** – sada pomocných grafických nástrojů jako např. uiimport (průvodce importem dat) nebo uibuttongroup (zjednodušení práce se skupinou vzájemně se ovlivňujících tlačítek).
- **Uitoolbar** – vytvoří oblast, do které je možné umísťovat interaktivní snadno dostupné ikony.

3.5.3.3 Získání ukazatele na grafický objekt

Aby se práce s grafickými objekty co nejvíce zjednodušila, podporuje MATLAB několik příkazů pro získání jejich ukazatelů. Nejsme tedy odkázáni jenom na uchovávání si ukazatelů důležitých objektů v proměnných nebo dokonce na procházení a hledání objektu ve stromu objektů pomocí vlastností *Parent* a *Children*.

Získání ukazatele můžeme provést pomocí následujících příkazů:(18)

- **GCF** – vrací hodnotu ukazatele na aktuální okno *figure*. Pokud neexistuje, vytvoří jedno.

- **GCA** – vrací hodnotu ukazatele na aktuální plochu os *axes*. Pokud neexistuje, vytvoří jednu.
- **GCO** – vrací hodnotu ukazatele na naposledy myší označený objekt. Objekty *uimenu* se neuvažují a v okamžiku před prvním kliknutím na objekt se vrací ukazatel aktuálního okna *figure*.
- **GCBF** – vrací hodnotu ukazatele na okno *figure*, na němž je umístěn grafický objekt právě prováděného zpětného volání viz. kapitola 3.5.3.5 Zpětné volání. V případě, kdy se žádné zpětné volání neprovádí, vrací prázdnou matici.
- **GCBO** – vrací hodnotu ukazatele na objekt, kteréhož zpětné volání je právě prováděno. V případě, kdy se žádné zpětné volání neprovádí, vrací prázdnou matici.
- **FINDOBJ** – vrací seznam ukazatelů na objekty, které splňují definované vlastnosti. V případě absence parametrů s požadovanými vlastnostmi vrací seznam všech viditelných objektů.
- **FINDALL** – vrací seznam ukazatelů na všechny objekty, které jsou součástí hierarchie definovaných objektů a mají uvedené vlastnosti. Funkčností je příkaz obdobný předchozímu s rozdílem prohledávání i v neviditelných objektech.

3.5.3.4 Příkazy Get a Set

Jsou stěžejními příkazy pro manipulaci s vlastnostmi objektů po jejich vytvoření. Pracují s objekty na úrovni ukazatelů *handle* a čtou resp. nastavují vlastnosti pomocí klíče resp. dvojice argumentů klíč-hodnota.(18)

```
get(h,'PropertyName')
set(h,'PropertyName',PropertyValue)
```

V uvedeném příkladu pracujeme s ukazatelem *h*, kde pomocí *get* čteme hodnotu vlastnosti *PropertyName* a pomocí *set* nastavujeme dvojici *PropertyName-PropertyValue*.

3.5.3.5 Zpětné volání

Angl. callback představuje základní stavební kámen událostmi řízeného programování. V této kapitole se zaměřím na zpětné volání iniciované hlavně událostmi grafických objektů, protože tento druh programování je široce využíván i v jiných pro téma práce nesouvisejících oblastech, jako je simulace paralelních dynamických systémů pomocí balíku Simulink.

Každý grafický objekt disponuje minimálně dvěma událostmi, na které lze reagovat. Těmi jsou okamžik bezprostředně po vytvoření objektu a okamžik těsně před zrušením objektu. Objekt okna *figure* disponuje nejširší škálou událostí, kam patří stisknutí a uvolnění tlačítek myši, žádost o zavření okna, stisknutí a uvolnění klávesy, změna velikosti okna, otočení kolečka myši a pohyb kurzoru myši nad oknem. Ke všem těmto událostem objektu *figure* existuje možnost zpětného volání definované funkce.

Definice zpětně volané funkce může být následující:(18)

```
function myfile(obj, event, arg1, arg2)
```

Funkce nemusí předávat zpět žádné výstupní argumenty, protože se nikde dále nezpracovávají. Co ale musí obsahovat jsou první dva argumenty, ve kterých se předává ukazatel na objekt, kde nastala událost, a bližší informace spojené s událostí. Zbylé dva argumenty nejsou povinné, ale lze tak funkci předávat další proměnné dostupné v místě přiřazení zpětného volání k události.

Přiřazení zpětného volání k události může být následující:

```
set(h, 'StartFcn', {@myfile, 5, 6})
```

Argument *h* nám říká, jakému objektu přiřazujeme zpětné volání, a zbylá dvojice nastavuje reakci na událost *StartFcn* na hodnotu buňky, ve které je symbolický odkaz volané funkce *myfile* a za ním dva nepovinné argumenty. Definice reakce na události je dobré pro přehlednost nastavovat už v okamžiku vytváření objektu.

3.5.3.6 Pokročilé způsoby sdílení proměnných

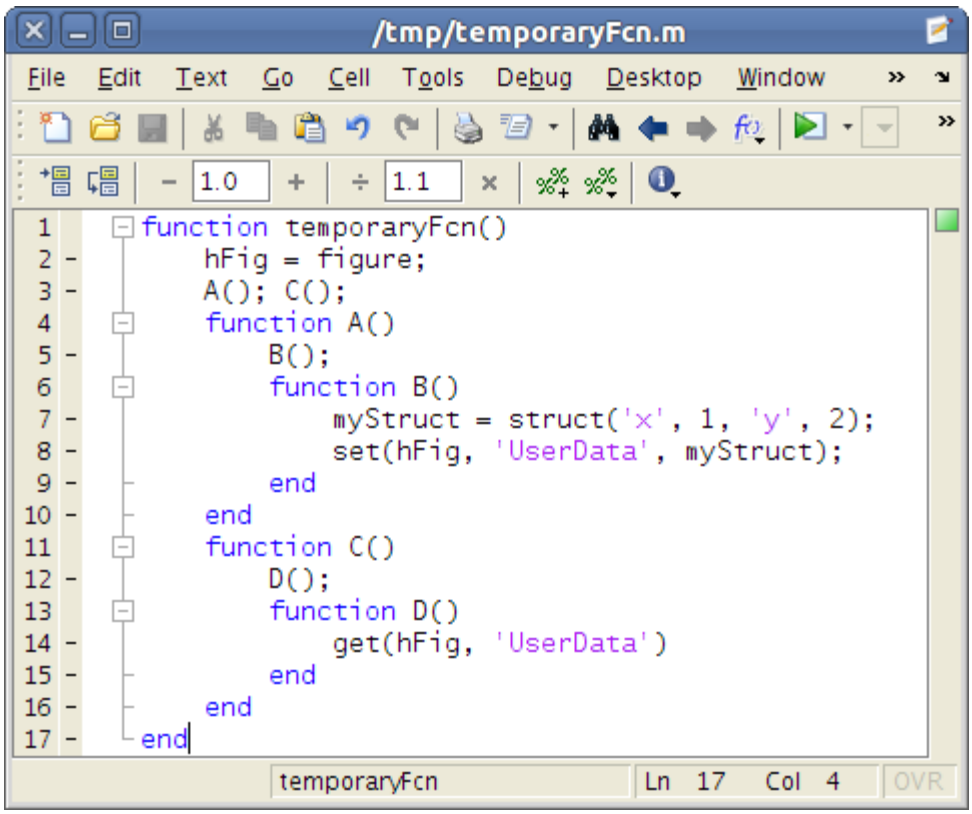
Skript uložený v m-souboru má k dispozici všechny definované proměnné prostředí. S použitím funkcí tato vlastnost spouštěného kódu zaniká a proměnné si do funkcí předáváme nejčastěji vstupními a výstupními parametry. Tento postup není ve všech

případech vhodný, a proto existuje ještě několik dalších mechanismů pro sdílení dat. Pomiňme nevhodné způsoby jako je např. umísťování proměnných do souboru.

Globální proměnné v jazyce MATLAB existují a definují se klíčovým slovem *global* před jménem proměnné. Toto klíčové slovo je nutné uvést před proměnnou ve všech z hlediska přístupnosti proměnných oddělených programových blocích. Způsob předávání a sdílení dat pomocí globálních proměnných je již řadu let považován za překonaný a není doporučeno ho ve vyšších programovacích jazycích používat. Mohou však existovat vzácné případy, kdy globální proměnné zvyšují šanci vzniku chyb v programu jen minimálně. Pak proti jejich použití nelze mnoho namítat.

Druhým způsobem sdílení dat představují vnořené funkce. Ty využíváme při členění větších funkcí do samostatných bloků. Vnořené funkce disponují vlastností jednosměrného sdílení prostoru proměnných s nadřazenými funkcemi, takže tím odpadá nutnost předávání proměnných z vyšších úrovní do nižších. Přístup z vyšší úrovně k proměnným definovaným v nižší úrovni není možný. V takovém případě využijeme např. návratové proměnné. Vnořené funkce na stejné úrovni nesdílejí prostory proměnných, ale mohou se navzájem volat.

Třetí způsob sdílení využívá vytvořené grafické objekty, do nichž jsou ukládány potřebné informace. Slouží k tomu vlastnost objektu zvaná *UserData*. která dokáže pojmout jakoukoliv platnou hodnotu proměnné prostředí MATLAB. Pro některá použití může představovat nevýhodu schopnost takto uložit pouze jednu proměnnou. Toto lze obejít uložením složitější struktury, což ale vytváří vyšší nároky při manipulaci s částmi uložené struktury. Ukázku využití vlastnosti *UserData* objektu *figure* vidíme na obrázku 3.11, kde strukturované pole vytvořené ve funkci *B()* je přístupné ve funkci *D()*. Rovněž vidíme jednosměrné sdílení prostoru proměnných, díky čemuž mohou funkce *B()* a *D()* přistupovat k ukazateli *hFig*.



Obrázek 3.11: Využití vlastnosti UserData u vnořených funkcí

Data aplikace nastavované a čtené pomocí *setappdata* a *getappdata* je další způsob předávání informací mezi prostory s oddělenými proměnnými. Ve své podstatě je to stejný způsob předávání jako předchozí *UserData*, ale s několika málo rozdíly. Hlavním je způsob přistupování na základě jména vlastnosti a ukazatele místo samotného ukazatele. Jako existuje vlastnost *'UserData'* nějakého objektu, tak mohou existovat další nově vytvořené vlastnosti *'DobraVlastnost'* a *'SpatnaVlastnost'*. Abychom k novým vlastnostem objektu mohli přistupovat pomocí *getappdata*, musíme disponovat jeho ukazatelem a jménem vlastnosti. Takto přístupná vlastnost je i *UserData*.

Posledním způsobem předávání proměnných je využívání funkce *guidata*. Tento způsob používá i nástroj pro tvorbu grafického rozhraní GUIDE. Od předchozích se liší jednodušší syntaxí v podobě pouze jedné funkce *guidata()*. Použití je následující:(18)

```
guidata(object_handle,data)
data = guidata(object_handle)
```

První řádek ukládá do grafického objektu specifikovaného ukazatelem informace

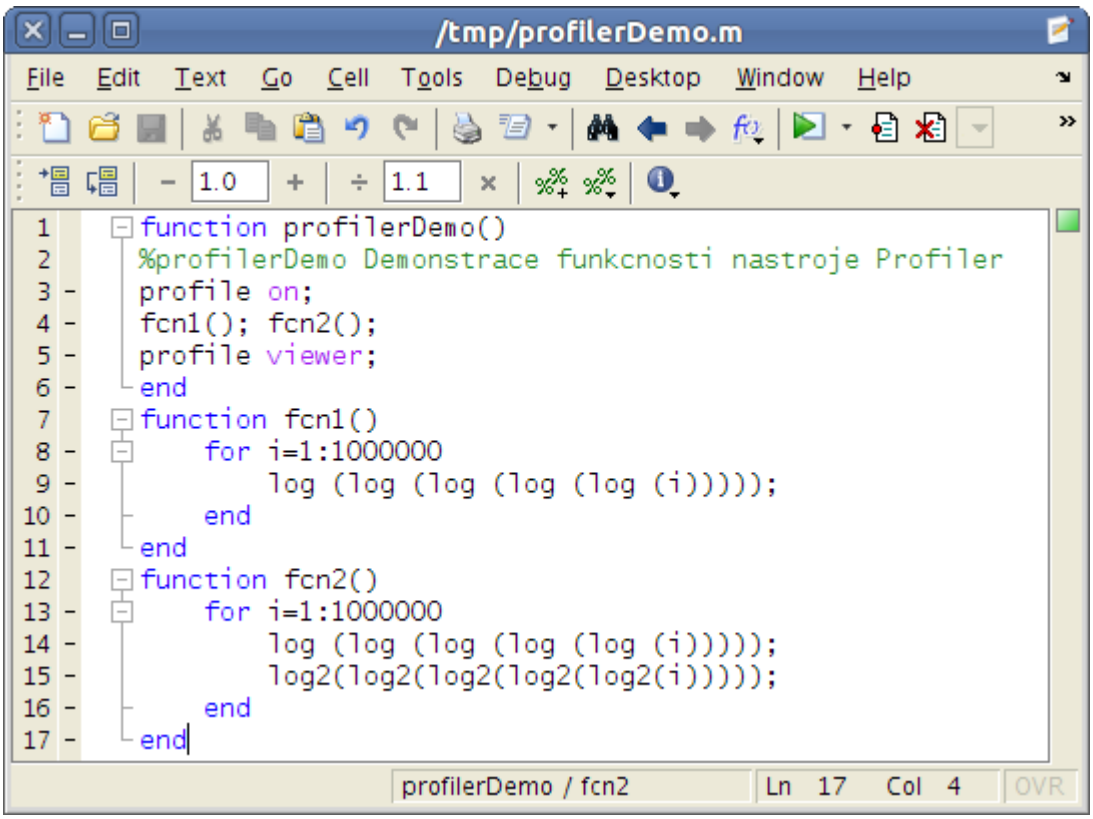
v podobě proměnné *data*. Druhý řádek načítá informace z objektu specifikovaného ukazatelem a předává je jako návratovou hodnotu do proměnné *data*. Tento způsob ukládání pomocí funkce *guidata()* je téměř identický, jako s využitím vlastnosti *UserData*. Trochu nelogicky působí nemožnost ukládat do jiných grafických objektů, než jsou okna *figure*. Pokud se použije ukazatel na nějaký jiný grafický objekt, data jsou uložena do nejbližšího rodičovského okna *figure*.

Poslední tři jmenované mechanismy ve výsledku ukládají a načítají vlastnosti objektu stejným způsobem. Existence tří takto podobných cest pro dosažení stejného výsledku může na leckoho působit zmatečným dojmem. Obzvlášť na nezkušeného uživatele.

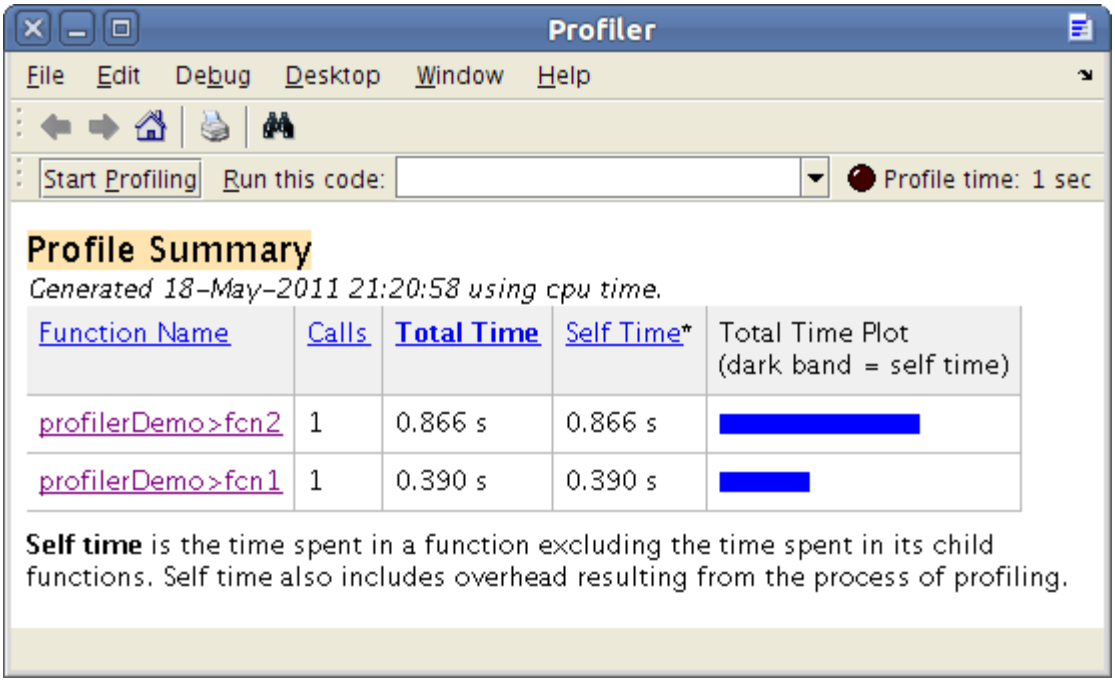
3.5.3.7 Analýza kódu pomocí nástroje Profiler

Nástroj Profiler je užitečným pomocníkem při vývoji složitějších a na výkon počítače náročnějších aplikací. Poskytuje programátorovi přehled o časové náročnosti jednotlivých funkcí a příkazů. Je žádoucí, aby měl tvůrce aplikace představu o náročnosti jednotlivých příkazů, ale při nevyhnutelném rozvětvení programu a využití několika cyklů jde identifikovat úzké hrdlo obtížně. A zde nastupuje nástroj Profiler, který změří, jak dlouho strávil procesor prováděním jednotlivých bloků kódu. Výsledky jsou přehledně prezentovány a dobře se z nich dají odhalit místa, kam by se měla soustředit pozornost při případné optimalizaci. Profiler bývá užitečný i pro ověření si správné provázanosti volaných funkcí, i když k tomuto účelu existuje specializovaný příkaz *depfun*.

Na obrázku 3.12 vidíme zdrojový kód demonstračního příkladu, na kterém si ukážeme analýzu pomocí nástroje Profiler. Příklad obsahuje spuštění dvou funkcí, které v cyklu provádí netriviální matematické operace. Funkce následují za příkazem spuštění zaznamenávání statistických informací a po jejich provedení se otevře grafické rozhraní nástroje Profiler zobrazené na obrázku 3.13



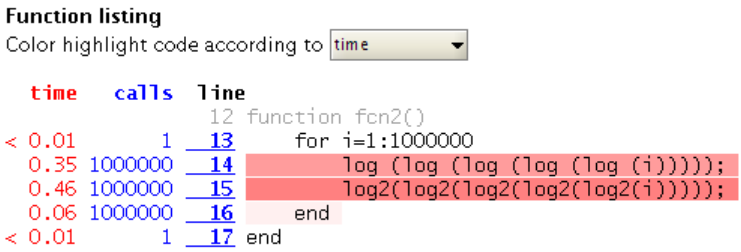
Obrázek 3.12: Zdrojový kód příkladu



Obrázek 3.13: Výsledky analýzy nástroje Profiler

Výsledek analýzy obsahuje seznam nejvíce vytěžujících funkcí. V našem případě je

funkce *fcn2* zpracována za zhruba dvojnásobně delší čas než funkce *fcn1*. Detail *fcn2* se nám zobrazí při kliknutí na odkaz funkce.



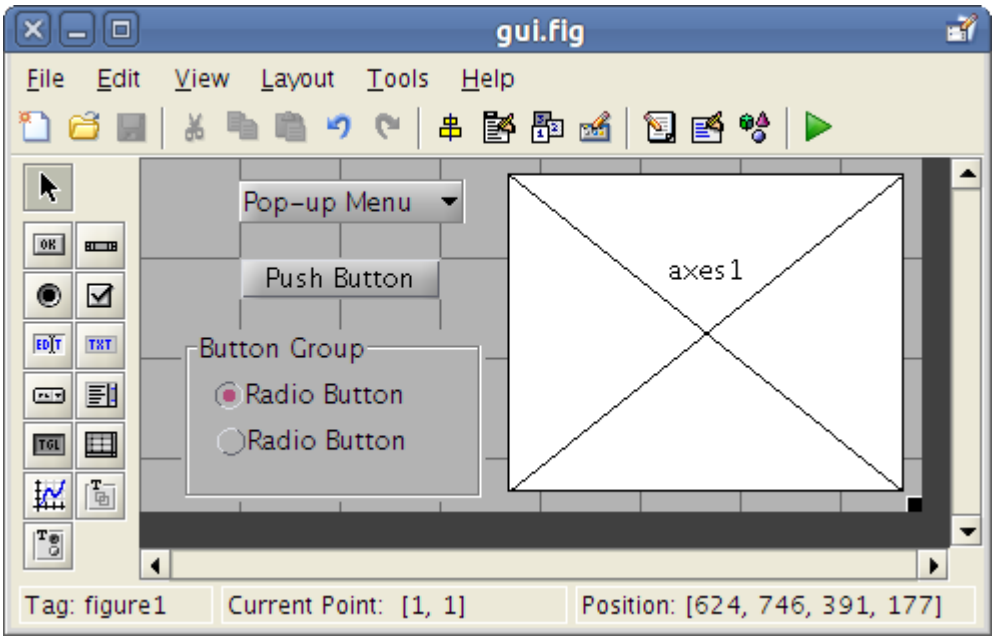
Obrázek 3.14: Doba procesoru strávená vykonáváním

Nejzajímavější informace nese spodní část výpisu, kde vidíme časovou náročnost jednotlivých řádků.

Tímto způsobem lze analyzovat i kód knihoven Matlabu, který nás může inspirovat v tvorbě vlastního méně výpočetně náročného řešení.

3.5.3.8 Tvorba grafického uživatelského rozhraní

V prostředí MATLAB lze vytvářet uživatelské prostředí dvěma způsoby. Prvním automatizovaným je využití integrovaného pomocného nástroje GUIDE na obrázku 3.15 a druhým je ruční zápis funkcí pro tvorbu grafických prvků. První způsob návrhu rozhraní je jednodušší a celkově rychlejší. Výsledkem návrhu jsou soubory s příponou *.fig* a *.m*. Pokud chceme mít nad vzhledem aplikace absolutní kontrolu a chceme porozumět detailům vytváření a práce s grafickými objekty, můžeme se přiklonit k druhé možnosti. Výsledkem pak bude pouze m-soubor, který současně s definicí grafického rozhraní může obsahovat i výkonný kód programu.



Obrázek 3.15: Nástroj GUIDE

V levé části okna vidíme ikony představující dostupné grafické objekty, které můžeme využít pro tvorbu uživatelského rozhraní. V horní liště stojí za zmínku Menu Editor pro tvorbu hlavní programové nabídky programu, Toolbar Editor pro úpravu trvale viditelné lišty ikon, M-file Editor pro editaci zdrojového souboru grafického rozhraní, Property Inspector pro zobrazení vlastností grafických objektů, Object Editor pro zobrazení hierarchie vztahů objektů a spuštění okna formuláře. Tyto nástroje jsou dostupné rovněž z hlavního programového menu.

4 Analýza současného stavu

Nyní si přiblížíme výchozí stav této diplomové práce a po představení si společnosti a jejích výrobků věnujeme pozornost oblasti logistiky a přepravních nákladů.

4.1 Skupina Knorr-Bremse

4.1.1 Historie

Původ koncernu sahá do roku 1905, kdy Ernst Theodor Georg Knorr (1859-1911) v Berlíně založil Knorr-Bremse GmbH a tím zužitkoval zkušenosti z dřívějšího působení ve společnosti Carpenter & Schulze. Georg Knorr je vynálezcem jednokomorové vzduchem poháněné brzdy pro vlakové soupravy a na výsledcích jeho práce vznikl poměrně rozšířený typ brzdy označovaný jako Kunze-Knorr.(1)

Obrázek 4.1: Georg Knorr

To byl počáteční vklad do začátků společnosti, která díky smlouvě s Pruskými státními drahami začala záhy expandovat.

V roce 1922 společnost vstoupila na vznikající trh se vzduchovými brzdovými systémy pro silniční užitková vozidla a jako první evropská společnost nabídla brzdový systém schopný současného brzdění všech čtyř kol nákladního automobilu zároveň s koly přívěsu. V průběhu druhé světové války továrna vyráběla i menší série kulometu MG35/36A pro potřeby německé armády. V nejisté poválečné době se poměrně zachovalý výrobní závod ocitl v sovětském sektoru vlivu, a proto začalo hledání výrobních prostor mimo tuto zónu. Mnichovská společnost Bayerische Flugzeugwerke AG vyrábějící letecké motory měla díky mírovým smlouvám a zákazu produkce leteckých motorů opačný problém. Téměř celý její výrobní program byl pozastaven, a tak hledala způsoby nového uplatnění. Došlo tedy k dohodě mezi oběma společnostmi a zbylé nezkonfiskované výrobní linky Knorr-Bremse GmbH se přesunuly do Mnichova, kde se nachází ústředí společnosti dodnes.

Se změnou hlavního závodu nastala i změna v zaměření na další oblasti a trhy.

V šedesátých letech byla vytvořena divize pneumatických systémů a v sedmdesátých letech vznikla divize dieselových motorů. Operace s brzdovými systémy pro užitková vozidla byly utlumovány. Změna strategie společnosti se ukázala jako špatný tah, protože nově vytvořené divize se dlouhodobě potácely na hranici ziskovosti nebo se propadaly do ztráty. Panovala velká roztříštěnost v cílech jednotlivých divizí, čímž se stala situace neudržitelnou.

Mezník ve vývoji představoval rok 1985, kdy se společnost celkově nacházela ve špatné finanční situaci a po konsolidaci rozhodovacích pravomocí byla očekávána zásadní reorganizace. V tento okamžik majoritní vlastník nečekaně oznámil záměr nabídnout svůj podíl k prodeji a výtěžek z prodeje věnovat náboženské charitě. To vyvolalo silné reakce u veřejnosti a zaměstnanců, kteří vyjadřovali obavy o budoucnost společnosti. I přes tyto nejisté vyhlídky se podařilo získat dlouholetému manažerovi společnosti Heinz Hermann Thielemu finanční podporu Deutsche Bank, aby mohl učinit nabídku na odkup podílu. Po provedení transakce okamžitě přistoupil k restrukturalizaci společnosti s cílem vytvoření globálního hráče na trhu brzdových systémů pro užitková vozidla a vlakové soupravy. Došlo ke spojení Knorr-Bremse GmbH s poválečným partnerem Süddeutsche Brake AG do společného podniku Knorr-Bremse AG, byly prodány divize dieselových motorů a divize zabývající se výrobou nářadí. V roce 1993 prodeje pokračovaly divizí na výrobu průmyslových pneumatických systémů a v r. 1997 divizí vyrábějící kovové odlitky.(1)

4.1.2 Akvizice

V průběhu novodobé historie došlo k několika důležitým akvizicím, jejichž cílem bylo rozšíření působnosti na další zahraniční trhy. Roku 1990 byla pohlcena část švýcarského Oerlikon Buehrle zabývajícího se brzdovými systémy traťových vozidel. Téhož roku došlo k převzetí původního závodu umístěného v Berlíně, kde stále působila společnost vzniklá na základech odsunuté poválečné výroby Knorr-Bremse. Důležitým průnikem na severoamerický trh brzdových systémů traťových vozidel bylo odkoupení části společnosti New York Air Brake, hlavního dodavatele brzdových systémů New Yorského metra. V roce 1999 byl s firmou Robert Bosch GmbH vytvořen společný podnik na výrobu elektronických brzdových jednotek, kde Knorr-Bremse AG vlastní

většinový podíl 60% a díky tomu disponuje právem samostatného rozhodování. Rok 2000 charakterizoval odkup významného britského výrobce brzd traťových vozidel a dlouholetého konkurenta Westinghouse Brakes Ltd.

V novém tisíciletí následovaly další neméně důležité akvizice. Posílení nastalo v oblasti vlakových pneumatických dveřních systémů nákupem 90% podílu v rakouské společnosti IFE. Pro firemní aktivity na trhu brzdových systémů pro užitková vozidla byla v roce 2002 pravděpodobně nejvýznamnější akvizice severoamerického výrobce Bendix s 2000 zaměstnanci, který byl do té doby součástí koncernu Honeywell International.

V roce 2008 byl otevřen společný podnik na výrobu torzních tlumičů pro největšího ruského výrobce nákladních automobilů Kamaz. Podíl je mezi společnostmi rozdělen rovnoměrně a vedením závodu je pověřena Knorr-Bremse AG.(4)

Obrázek 4.2: Heinz Hermann Thiele (druhý zprava) a Sergey Kogoghin (druhý zleva), ředitel JSC Kamaz, při otevírání nového závodu na dodávky torzních tlumičů pro vozy Kamaz.

V České republice se koncern rovněž angažoval. V dobách plánovaného hospodářství působil na českém trhu dlouhá léta výrobce brzdových systémů Autobrzdy Jablonec s.p.. Dodával brzdové systémy odběratelům Avia, Karosa, Liaz, Tatra, Škoda a dalším. Po transformaci vynucené změnou režimu byla vytvořena společnost ATESO a.s., která se záhy po svém vzniku začala potýkat s výrazným úbytkem zakázek. Kapacity závodu v Hejnicích v Libereckém kraji byly vytěžovány na 20% a hrozilo zastavení provozu. Rokem 1991 začíná spolupráce s Knorr-Bremse na součástkové výrobě. O dva roky později vzniká společný podnik Knorr-Autobrzdy Jablonec se sídlem v Hejnickém závodě s majoritním podílem 67% pro Knorr-Bremse AG. Roku 1998 se Knorr-Bremse stává 100% vlastníkem a dochází k rozšiřování výrobních kapacit a modernizování výroby.

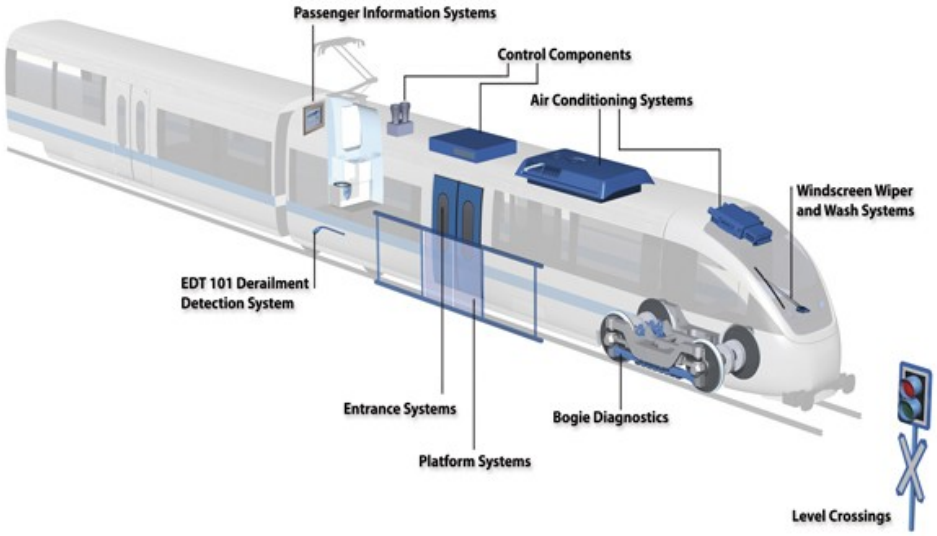
Za 15 let od změny vlastnické struktury v roce 1985 společnost Knorr-Bremse

provedla 28 akvizicí.(1)

4.1.3 Struktura společnosti

Společnost Knorr-Bremse AG je rozdělena na dvě obchodní divize, které si jsou rovnocenné a do jisté míry tvoří zdravou vnitropodnikovou rivalitu.

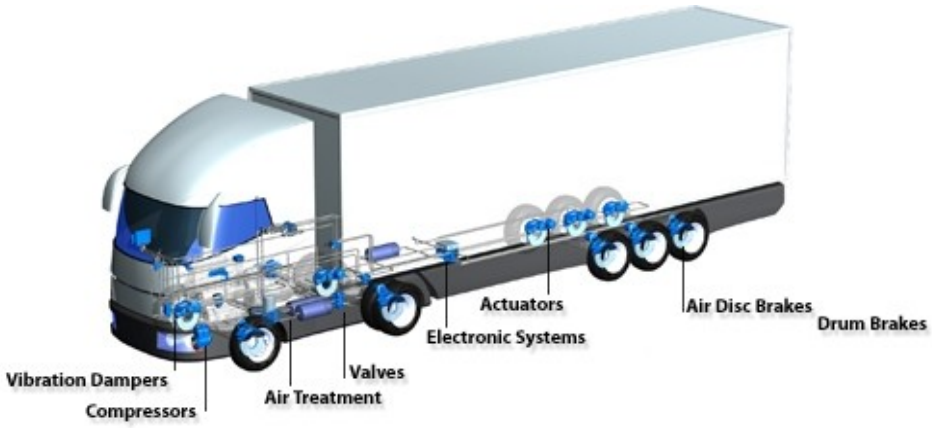
První divize se věnuje systémům pro celé spektrum kolejových vozidel, kam patří výroba brzdových systémů, částí řídících systémů, dveřních a nástupních systémů, sociálního zařízení, klimatizace, spojovacích spřáhel, informačních systémů pro cestující, systémů pro detekci rozpojení vozů vlakové soupravy, systémů pro čistění čelních skel a doplňkových elektronických systémů. Přehled některých systémů můžeme vidět na obrázku 4.3.(7)



Obrázek 4.3: Diagram komponent pro kolejová vozidla.

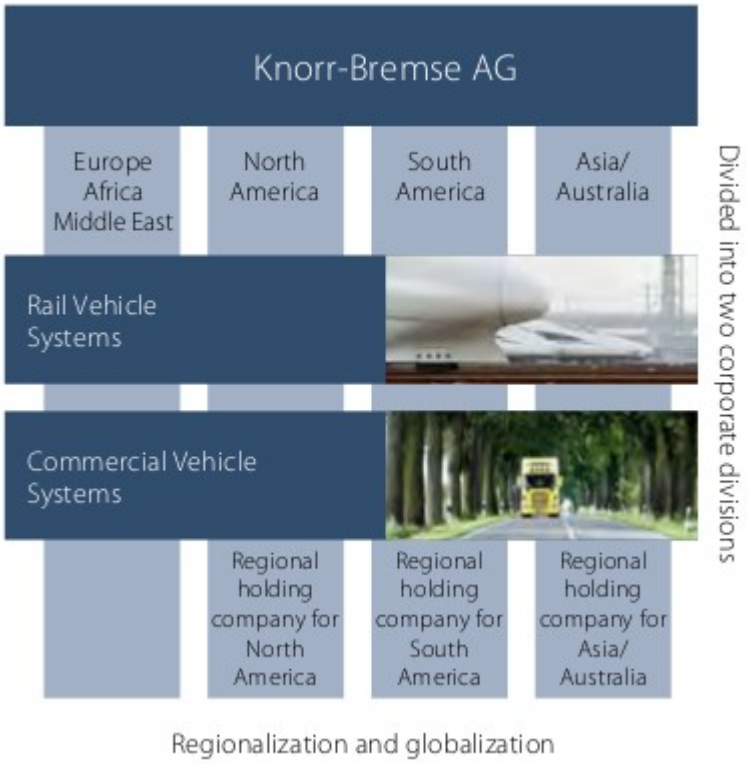
Druhá divize se soustředí na výrobu pro trh s užitkovými silničními vozidly, kde hlavní část produktového portfolia tvoří torzní tlumiče, vzduchové kompresory pro pneumatické systémy, filtrační jednotky pro pneumatické systémy, ventily, membránové brzdové válce, diskové a bubnové brzdy a elektronické obvody pro řízení protiblokovacích brzdových systémů, systémů regulace prokluzu kol, systémů pro regulaci zatížení brzd a rovnoměrnou distribuci brzdného účinku, stabilizačních systémů, systémů bočního naklápění vozidel a systémů pro udržování optimální vzdálenosti mezi vozidly. Některé vyráběné komponenty jsou znázorněny na obrázku

4.4.(7)



Obrázek 4.4: Diagram některých komponent pro silniční užitková vozidla.

Kvůli jednoduššímu a efektivnějšímu plnění specifických požadavků zákazníků jsou divize rozloženy do regionů. Vyšší nároky na zvládnutí spolupráce regionálních zastoupení jsou vyváženy spokojenějšími zákazníky, kteří následně využívají služeb koncernu Knorr-bremse opakovaně. Rozdělení je znázorněno na obrázku 4.5.(7)



Obrázek 4.5: Rozdělení obchodních divizí na regiony.

Další úroveň regionalizace je zakládání lokálních zastoupení a výrobních závodů. K co největšímu lokálnímu zastoupení také motivuje uplatnění se na tzv. Aftermarket trhu. Ten se skládá z více menších zákazníků, kteří díky nižší administrativní náročnosti a nižším přepravním nákladům upřednostňují lokální dodavatele náhradních dílů.

Koncern Knorr-Bremse AG takto pokrývá 85 regionů ve 29 zemích světa a proto je optimalizace dopravních nákladů jedna z priorit. Touto problematikou se také budu zabývat v této práci. Schématické územní zastoupení je zobrazeno na obrázku 4.6.

Obrázek 4.6: Rozložení zastoupení v jednotlivých zemích světa.

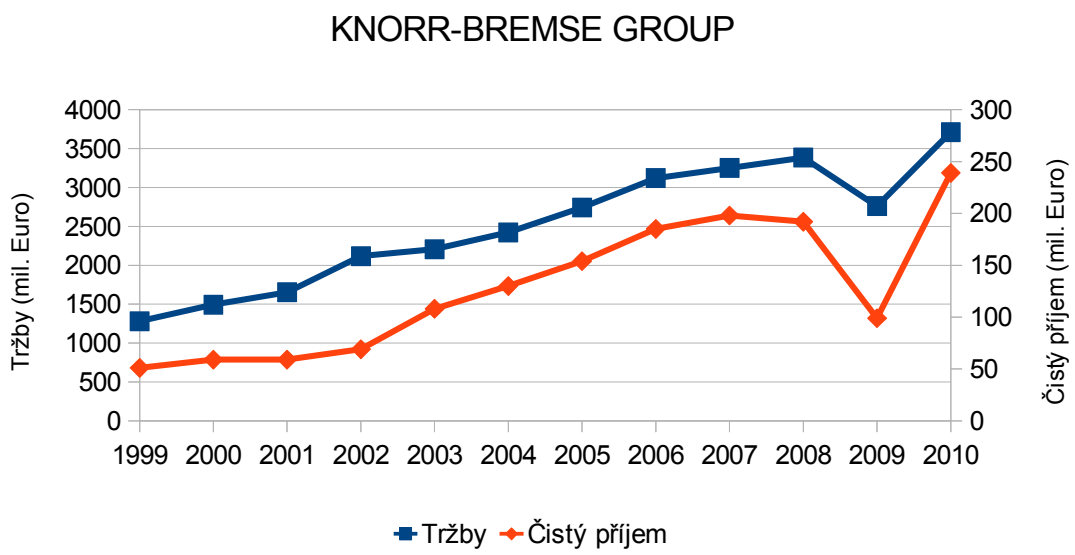
4.1.4 Klíčové hospodářské ukazatele

Pro hlubší představu o hospodaření koncernu si uvedeme několik základních ukazatelů. (5)(6)(7) Uvedené hodnoty jsou, až na počet zaměstnanců, v miliónech Euro.

Knorr-Bremse group	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Tržby	1281	1494	1653	2118	2206	2423	2743	3121	3251	3384	2761	3712
Čistý příjem	51	59	59	69	108	130	154	185	198	192	99	239
Zaměstnanci	-	-	9215	10959	10763	11143	12119	13035	13943	14999	14432	16277
Osobní náklady	-	-	413	496	488	508	538	592	622	686	641	721
Hodnota objednávek	-	-	1714	2377	2265	2447	2849	3541	3767	3209	3185	4040
Výzkum a vývoj	-	-	106	119	120	124	133	141	159	171	153	175

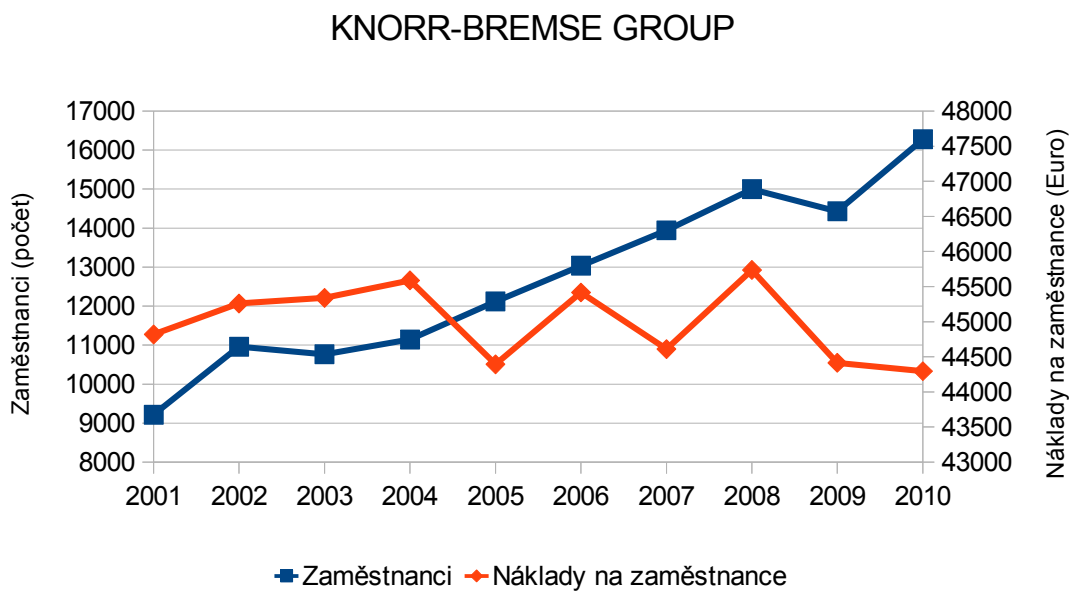
Tabulka 4.1: Hospodářské údaje koncernu Knorr-Bremse

Relevantní údaje o hospodaření nebudou pravděpodobně více než 10 let staré.



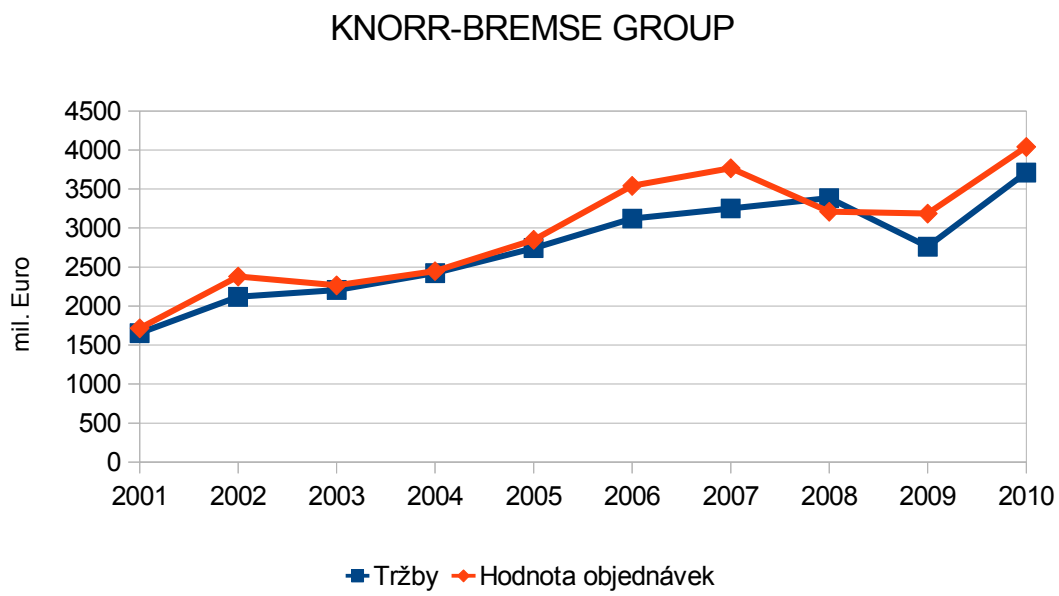
Jak můžeme pozorovat, rok 2009 byl pro společnost rokem krize. Zásadní úbytek objednávek se projevil již v druhé půlce roku 2008, což ovlivnilo jinak pozitivní vývoj v půlce první.(6) Poklesem byla zasažena hlavně divize systémů pro užitková vozidla, která si po roky 2006, 2007 a 2008 držela zhruba konstantní množství objednávek, ale v roce 2009 došlo k jejich snížení o více než 38%.(7) Divize kolejových vozidel byla díky úspěchu na Asijských a Amerických trzích krizí zasažena velmi mírně.(6)

Podle předběžných výsledků (11) dosáhla skupina za rok 2011 celkových tržeb 4,24 miliard Euro s rozložením 2,19 miliardy připadající na divizi kolejových vozidel (nárůst 8%) a 2.07 miliardy z divize užitkových vozidel (nárůst 22%). Divize kolejových vozidel zaznamenala nárůst tržeb hlavně v regionu severní Ameriky a Evropy. Divize užitkových vozidel pak měla největší zlepšení v Evropě a částečně v severní Americe. Nicméně poptávka trhu v Evropě stále nedosahuje předkrizové úrovně ze začátku roku 2008.(11)



Pozitivní trend je patrný z vývoje průměrných nákladů na jednoho zaměstnance, které jsou i přes pravidelnou valorizaci platů zhruba na stejné úrovni po celé sledované období. Příčinou je tvorba nových pracovních míst v regionech s nižší průměrnou mzdou. Rozložení pracovní síly je ke konci roku 2010 z 21,8% na americkém kontinentu, z 21,4% v regionu Asie/Austrálie a z 56,8% v Evropě.(7)

Posledním ze zajímavých ukazatelů, který si rozebereme, je hodnota uzavřených objednávek v relaci s tržbami. Pokud je jejich rozdíl příliš velký, může to naznačovat např. problémy při dostání nasmlouvaných závazků. To může být způsobeno neočekávaným příliš rychlým růstem poptávky a nedostatečnými výrobními kapacitami. Málo předvídatelný je také výpadek některého z dodavatelů např. kvůli přírodním katastrofám.



Uspokojování objednávek v letech 2006 a 2007 pravděpodobně nebylo ideální. Důvodem také mohou být finanční potíže některého z odběratelů. Rok 2008 charakterizuje dokončení objednávek z předešlého období.

Celkově můžeme hodnotit vývoj společnosti velice pozitivně. I přes obtížné období světové finanční a ekonomické krize se podařilo za poslední dekádu navýšit tržby zhruba na dvojnásobek. Velmi k tomu přispěla divize kolejových vozidel, která si připsala významnější růst. V roce 2003 tvořila divize kolejových vozidel 41% tržeb koncernu.(5). V roce 2010 to je již přes 54%.

4.1.5 Hlavní odběratelé divize užitkových vozidel

Mezi hlavními odběrateli divize užitkových vozidel je několik největších světových výrobců nákladních automobilů. Silnou pozici má koncern zejména v Evropě, kde můžeme nalézt brzdové systémy Knorr-Bremse v automobilech značek Mercedes-Benz, Volvo, Renault, Iveco, DAF, Scania, Volkswagen, MAN a dalších. Z tuzemských odběratelů pro finální produkty lze jmenovat společnost TATRA Kopřivnice, Irisbus Iveco, SOR Libchavy, TEDOM nebo Zetor.

4.2 Divize systémů užitkových vozidel v ČR

V České republice působí koncern prostřednictvím tří společností, ve kterých vlastní

100% podíl.(14)(15)(16)

- IFE-CR a.s.
- IGE-CZ s.r.o.
- KNORR-BREMSE Systémy pro užitková vozidla ČR s.r.o.

První dvě patří do divize kolejových vozidel a třetí do divize silničních užitkových vozidel. V diplomové práci se budu věnovat problematice dopravních nákladů pouze u poslední jmenované.

4.2.1 Obchodní informace

Obchodní firma: KNORR-BREMSE Systémy pro užitková vozidla ČR, s.r.o.

Sídlo společnosti: Stráž nad Nisou, Svárovská 700, PSČ 463 03

Identifikační číslo: 473 11 096

Datum zápisu do OŘ: 1.ledna 1993

Jednatel: Matthias Sander

Základní kapitál: 138 188 000,- Kč

Předmět podnikání: Výroba a prodej brzdových systémů dopravních prostředků, jejich součástí a příslušenství nářadí a nástrojů, kovářství, zámečnictví, výroba nástrojů, broušení a leštění kovů kovoobrábění, výroba a opravy ostatních motorových dopravních prostředků, galvanizace kovů, koupě zboží za účelem jeho dalšího prodeje a prodej v rámci živností volných.(14)

Společnost je ze 100% vlastněna KNORR - BREMSE Systeme für Nutzfahrzeuge GmbH, D-8000 München 40, SRN, Moosacher Str. 80.(14)

Obrázek 4.7: Logo společnosti

4.2.2 Klíčové hospodářské ukazatele

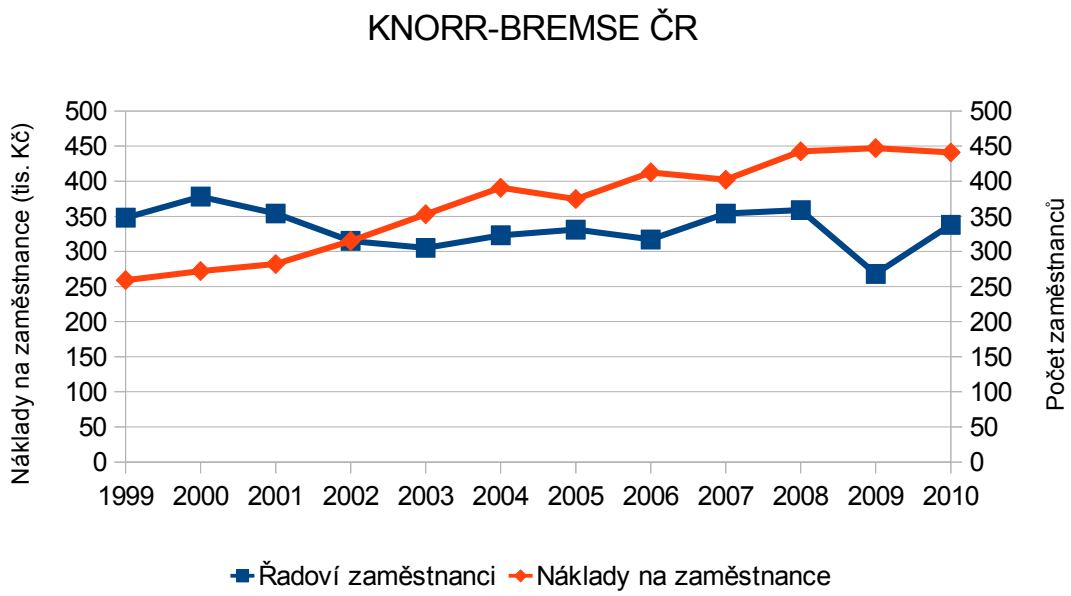
Uvedeme si některé zajímavé hospodářské ukazatele a postavíme je do relace s ukazateli celého koncernu.

Počet řadových zaměstnanců se v průběhu let drží zhruba na stejné úrovni a až na krizí postižený rok 2009 osciluje v rozmezí 305-378. Vedení společnosti se v roce 2010 stává z osmi pracovníků.(6)

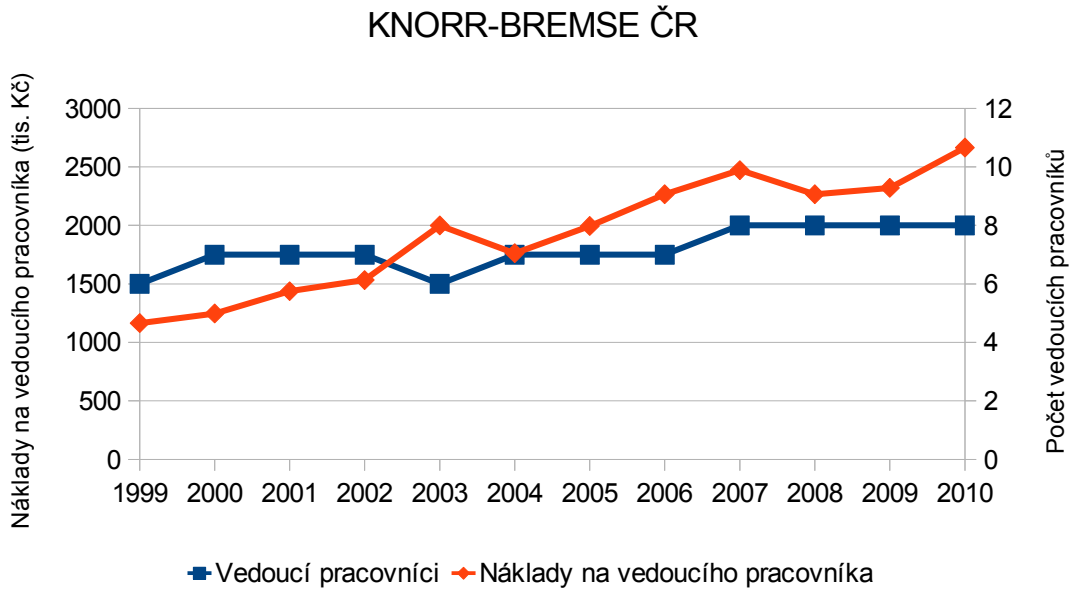
Knorr-Bremse ČR	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Řadoví zaměstnanci	348	378	354	315	305	323	331	317	354	359	268	338
Mzdové náklady	66652	76023	74180	73418	78755	91631	90116	95203	104018	116586	89023	109231
Soc. a zdr. pojištění	23077	26295	25304	25359	27478	32391	31371	33068	35770	39383	28758	37379
Sociální náklady	445	522	408	447	1441	2169	2507	2543	2583	2927	2091	2441
Celkem	90174	102840	99892	99224	107674	126191	123994	130814	142371	158896	119872	149051
Vedoucí pracovníci	6	7	7	7	6	7	7	7	8	8	8	8
Mzdové náklady	5170	6455	7446	7942	8868	9110	10251	11490	15042	15121	15514	17244
Soc. a zdr. pojištění	1810	2259	2606	2779	3104	3188	3675	4323	4672	2941	3036	3984
Sociální náklady	8	10	10	10	23	35	39	48	65	64	17	90
Celkem	6988	8724	10062	10731	11995	12333	13965	15861	19779	18126	18567	21318

Tabulka 4.2: Mzdové náklady Knorr-Bremse ČR(4)(5)(6)

Zajímavým ukazatelem jsou náklady na řadového zaměstnance, které se za deset let zvýšily o více než 62%.



Oproti tomu náklady na vedoucího pracovníka rostly mnohem markantněji a pomyslné nůžky se pravděpodobně budou rozevírat i nadále.



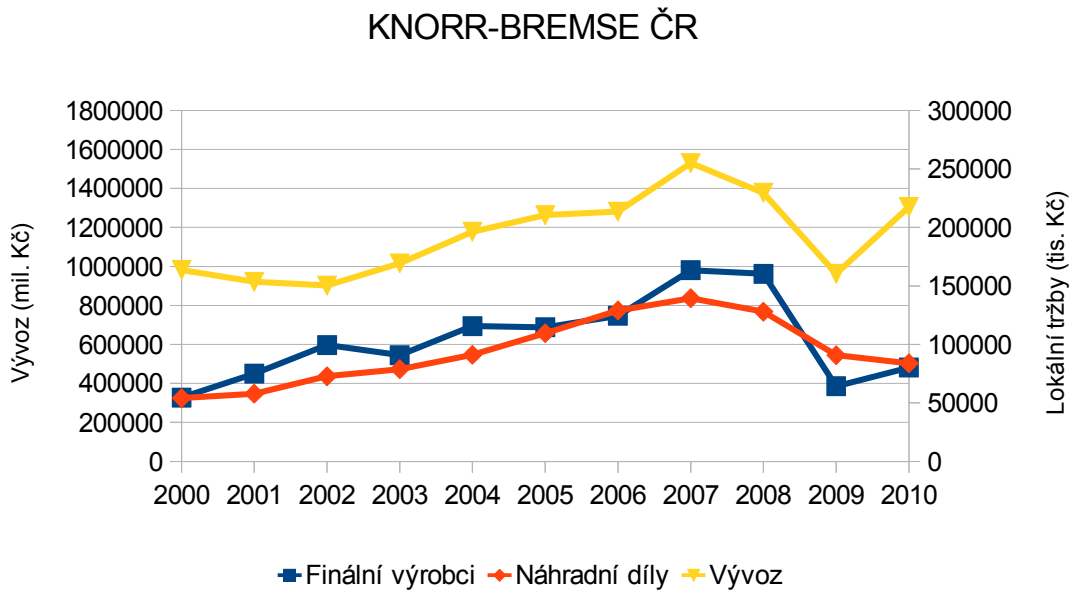
Náklady na vedoucí pracovníky se za posledních deset let zvýšily o 113%. V posledním sledovaném období, tj. roce 2010, dosahovaly hodnoty kolem 2,7 milionu Kč na osobu.

Velmi důležitým ukazatelem je velikost tržeb podle druhu odběratelů. Společnost rozlišuje odběratele na tuzemské finální výrobce, kteří používají výrobky pro začlenění

do větších celků. Typicky se jedná o výrobce nákladních automobilů nebo zemědělských strojů. Druhými neméně důležitými odběrateli jsou zákazníci na tzv. aftermarket trhu. Jedná se o různá lokální servisní střediska, která instalují zakoupené náhradní díly do existujících užitkových automobilů a zemědělských strojů. Zásadní objem tržeb však tvoří obchod se zahraničními společnostmi patřícími do skupiny Knorr-Bremse. S jinými zahraničními společnostmi nemá česká pobočka žádné obchodní vztahy.

Finální výrobci	54605	74942	99457	90837	115655	114677	124448	163506	160356	64263	80237
Náhradní díly	54104	57845	72804	78831	91077	109587	128941	139502	127988	90788	83823
Vývoz	982026	921185	901742	1015687	1177693	1263024	1280661	1529562	1377585	962084	1305373
Tržby celkem	1090735	1053972	1074003	1185355	1384425	1487288	1534050	1832570	1665929	1117135	1469433

Tabulka 4.3: Objem tržeb Knorr-Bremse ČR(4)(5)(6)



Z uvedeného grafu můžeme vyvodit několik zajímavých skutečností. Úbytek tržeb finálních výrobců v krizí postiženém roce 2009 byl na českém trhu razantnější než pokles tržeb ze zahraničního obchodu. To může naznačovat křehčí zázemí tuzemských výrobců, jejich málo diverzifikované produktové portfolio nebo zaměření se na menší množství trhů.

Naproti tomu trh s náhradními díly zaznamenal pokles poměrně pozvolný. To

plyne z jeho povahy, kdy si provozovatelé užitkových vozů nemohou dovolit vyměnit poškozený díl se zpožděním. V roce 2010 dále nedošlo k obratu trendu, jako tomu bylo u prodeje do zahraničí a finálním výrobcům a potvrzuje se tak vyšší setrvačnost aftermarket trhu.

	Tržby		Nákupy	
KNORR Německo	538659	47.37%	50215	46.19%
KNORR Maďarsko	290129	25.51%	18497	17.02%
KNORR Bendix (USA)	58329	5.13%	11845	10.90%
KNORR Francie	50719	4.46%	18300	16.83%
KNORR Itálie	50142	4.41%	7976	7.34%
KNORR Brazílie	34478	3.03%	0	0.00%
KNORR Velká Británie	25063	2.20%	510	0.47%
KNORR Čína	20445	1.80%	0	0.00%
KNORR Japonsko	20430	1.80%	0	0.00%
KNORR Španělsko	13267	1.17%	0	0.00%
KNORR Benelux	12889	1.13%	0	0.00%
KNORR Jižní Afrika	9171	0.81%	0	0.00%
KNORR Indie	5613	0.49%	1360	1.25%
KNORR Rakousko	5343	0.47%	0	0.00%
KNORR Švédsko	1189	0.10%	0	0.00%
KNORR Austrálie	931	0.08%	0	0.00%
KNORR Rusko	404	0.04%	0	0.00%
Celkem	1137201		108703	

Tabulka 4.4: Rozložení zahraničního obchodu za rok 2010.(6)

V tabulce jsou uvedeny všechny zahraniční společnosti koncernu, se kterými byl v roce 2010 uskutečněn nějaký obchod. Jedná se o hodnoty v milionech Kč. Klíčovou roli při nákupu a prodeji výrobků hraje několik málo poboček v čele s Německem, Maďarskem, USA, Francií a Itálií.

4.2.3 Výrobní závod Knorr-Bremse v ČR

Jak jsem se již zmínil v kapitole o akvizicích, první výrobní kapacity koncernu Knorr-Bremse v České republice tvořil bývalý závod Autobrzdy Jablonec s.p. V Hejnicích, který po privatizaci postupně přešel pod kontrolu české dceřinné společnosti. Otevřely se tím možnosti pro expanzi na nový trh a v případě přesunu výroby i pro snížení nákladů na pracovní sílu.

Závod se začal záhy rozšiřovat a modernizovat. Byly rozšířeny inženýrské sítě,

došlo k úpravám příjezdových cest a kvůli umístění v CHKO Jizerské hory byla zbudována čistička odpadních vod. Ve Frýdlantském výběžku, který trpí vysokou mírou nezaměstnanosti, byla existence spolehlivého zaměstnavatele s nadstandardními platovými podmínkami velmi kvitována.

Přes všechnu snahu byly dispoziční omezení stávajícího závodu dále obtížně řešitelné. Nevyhovující byla i existence rozšiřujících přidružených pracovišť v okolí a proto bylo rozhodnuto o přesunu do jiných výrobních prostor. Díky dosavadní úspěšné spolupráci v České republice a kvalifikované pracovní síle bylo zvoleno nové umístění v nedaleké průmyslové zóně Liberec sever. Přesun byl dokončen v roce 2010. Původní závod můžeme vidět na obrázku 4.8 a pro koncern Knorr-Bremse se v něm vyrábělo 19 let.(7)

Obrázek 4.8: Původní závod Knorr-Bremse ČR v Hejnicích

Nový závod v Liberci představuje významný posun díky využití bohatých zkušeností v koncernu Knorr-Bremse s tvorbou výrobních prostor. Byl zde využit nový přístup k výrobě a logistice podle celosvětově standardizovaného produkčního systému „Knorr-Bremse Production System“ (KPS) se zaměřením na optimální posloupnost a umístění operací přidávajících hodnotu výrobku „value-stream factory“. To vedlo ke zvýšení efektivnosti a přizpůsobitelnosti výrobního procesu.(7) Moderní modulární konstrukce stavby (obrázek 4.9) umožňuje v budoucnu jednoduché rozšíření výrobní plochy na přilehlých nezastavěných parcelách.

Obrázek 4.9: Nový závod Knorr-Bremse v Liberci dokončený v roce 2010. (12)

4.2.3.1 Výrobní program

Výrobní program se skládá z několika klíčových prvků brzdových systémů. Jmenovat můžeme následující. (12)

- Hlavní brzdiče řady MB (obrázek 4.10).
- Membránové válce řady BS (obrázek 4.11).
- Vysoušecí patrony řady FP (obrázek 4.12).
- Posilovače spojky řady VG (obrázek 4.13).


Obrázek 4.10: Hlavní brzdič MB


Obrázek 4.11: Membránový válec BS

Obrázek 4.12: Vysoušecí patrona FP

Obrázek 4.13: Posilovač spojky VG

Doplňujícím výrobním programem jsou přístroje ATESO, které setrvávají v nabídce od převzetí výrobního programu Autobrzdy Jablonec. Tyto přístroje již nepodléhají vývoji a do budoucna se počítá se zastavením jejich produkce.

4.2.3.2 Obchodní vztahy

Jak jsem již uvedl v kapitole 4.1.5 , lokálními odběrateli českého zastoupení jsou prakticky všichni tuzemští výrobci sériově vyráběných užitkových vozidel a odběratelé náhradních dílů. Třetí významnou skupinou jsou lokální dodavatelé materiálu, kam patří hlavně slévárny, šroubárny a výrobci umělohmotných výlisků nebo gumových těsnění. Pro chod výrobního závodu jsou také důležití dodavatelé mazacích směsí a emulzí nebo dodavatelé balícího materiálu.

Pro dopravu výrobků do a ze zahraničních poboček koncernu se využívá smluvní partner, který si účtuje cenu za přepravu na základě kilometrové tarifikace.

4.2.4 SWOT analýza

Pro přehled o současném stavu společnosti KNORR-BREMSE Systémy pro užitková vozidla ČR, s.r.o. si sestavíme SWOT analýzu, která nám dá konkrétní představu o vnitřním prostředí v podobě silných (Strengths) a slabých (Weaknesses) stránek a o vnějším prostředí pomocí příležitostí (Opportunities) a hrozeb (Threats).

4.2.4.1 Silné stránky

- Kvalifikovaná a dobře motivovaná pracovní síla tvořící kostru pracovního kolektivu, která má dlouhodobé zkušenosti s procesy uvnitř podniku,
- pravidelné zvyšování kvalifikace zaměstnanců na všech pozicích díky školením a stálé konfrontaci zkušeností se zahraničními týmy koncernu Knorr-Bremse,
- pověst kvalitního a spolehlivého dodavatele,
- přejímání nových technologií a výrobních postupů od koncernového vývojového pracoviště,
- mediální podpora koncernového oddělení vztahů s veřejností,
- právní podpora koncernového právního oddělení,
- certifikáty řízení kvality dle ISO 9001 a řízení ochrany životního prostředí dle ISO 14001,
- lehce modifikovatelné a rozšiřitelné výrobní prostory splňující všechny požadavky na moderní výrobní postupy,
- umístění v bezprostřední blízkosti rychlostní komunikace,
- široká základna potenciálních pracovníků díky umístění v okresním 100.000+ městě s bohatou tradicí výroby užitkových vozidel (LIAZ), ve kterém sídlí Technická univerzita Liberec,
- významné investice do výrobních linek a obráběcích CNC strojů jako součást strategie rozvoje,
- firemní kultura a důraz na utužování pracovního kolektivu na týmových sportovně-kulturních akcích.

4.2.4.2 Slabé stránky

- Některé nedořešené méně akutní změny v procesech po přesunu výroby do nových výrobních prostor,
- malá spolupráce s Technickou univerzitou v Liberci v oblasti vypisování

diplomových prací,

- díky provádění některých servisních úkonů smluvní zahraniční společností dochází k nadměrným prodlevám v zásazích.

4.2.4.3 Příležitosti

- Získání významnější pozice na tuzemském trhu s náhradními díly díky vyššímu důrazu na kompatibilitu s konkurenčními výrobky,
- rozšíření výrobního programu o nové produkty nebo ze zahraničí přesunuté výroby,
- přestěhování pracoviště galvanovny z již pronajatých prostor původního závodu v Hejnicích do nových prostor v Liberci,
- udržení nákladů na zaměstnance na dosavadní úrovni pomocí nárazového najímání dočasných pracovníků,
- získání exkluzivních dodavatelských vztahů s tuzemskými výrobci finálních produktů,
- získání dotací z Evropského sociálního fondu na podporu zaměstnanosti při obsazování nových pracovních pozic,
- rychlejší zavádění inovací z interního zaměstnaneckého programu „Dobrý nápad“.

4.2.4.4 Hrozby

- Finanční ztráty díky kolísavému kurzu České koruny a Eura,
- nejistota na evropském trhu kvůli pokračující neutěšené situaci v oblasti fiskální politiky některých států,
- přesunutí části výroby do úrovní rovnocenného závodu v maďarském Kecskemétu,
- utužení konkurence a ztráta některých odběratelů,
- platební neschopnost některých odběratelů.

4.2.5 Porterova analýza

V jistých ohledech se jedná o podobnou analýzu, jakou je SWOT, ale na rozdíl od ní přináší nové konkrétnější náhledy na problémy společnosti a míru konkurence v oblasti. Porterova analýza sleduje velikost pěti sil, které spolu interagují a ovlivňují schopnost společnosti dosahovat zisku. Těmito silami jsou:

1. Riziko vstupu nové konkurence.
2. Riziko vzniku substitutů.
3. Míra vlivu odběratelů při vyjednávání.
4. Míra vlivu dodavatelů při vyjednávání.
5. Míra stávající konkurence v odvětví.

Riziko vstupu nové konkurence je v oblasti brzdových systémů a jiných pneumatických systémů nízké z důvodu náročného vývoje a dlouhodobého testování produktu. Další z významných bariér pro vstup nových subjektů jsou celosvětově patentovaná technická řešení, jejichž nevyužití představuje významné zhoršení parametrů nově vyvíjeného produktu. Licencování takových technických řešení může představovat zásadní faktor, který může zhatit snahu o vstup na nový trh.

Riziko vzniku substitutů k existujícímu portfoliu společnosti je velmi nízké. U brzdových systémů probíhá průběžné zdokonalování již déle než jedno století a změna se nepředpokládá ani s příchodem elektrických a jiných alternativních pohonů. Může docházet pouze k dílčím změnám, ale většina pneumaticky nebo kapalinově ovládaných brzdových systémů zůstane z mechanického hlediska stejná.

Míra vlivu odběratelů při vyjednávání je silná. Světový trh s užitkovými vozidly pokračuje ve své globalizaci a očekává se spíše snižování počtu nezávislých výrobců. Na českém trhu se tak stalo s tradičním výrobcem užitkových vozidel do náročných terénů Tatra, která uzavřela dohodu s americkou skupinou Paccar Inc. (DAF) o dodávkách kabin a motorů. Paccar získal podíl ve společnosti Tatra 19%. Vyjednávací pozice Tatry se tím zlepšila, protože tu hrozí prohlubující se spolupráce s koncernem Paccar a využívání jeho dodavatelů. Naštěstí jsou brzdové systémy Knorr-Bremse dodávány i koncernu Paccar.

Míra vlivu dodavatelů při vyjednávání je střední až nízká. České pobočce jsou také dodávány díly ze zahraničních dceřinných společností, ale jejich ceny se řídí koncernovou politikou. Objemy obchodu s tuzemskými dodavateli jsou až na slévárnu Beneš a Lát, Zevetu bojkovice a Kamax poměrně nízké a objem dodávek jim nevytváří žádnou významnou vyjednávací pozici.

Míra stávající konkurence v odvětví je vysoká. Na celosvětovém trhu působí hned několik významných konkurentů jako např. WABCO Vehicle Control Systems/MeritorWabco, Benteler International, TRW Automotive nebo Haldex Commercial Vehicle Systems, kteří mají dlouhou zkušenost s výrobou brzdových systémů a mají dostatečně velké technologické a finanční zázemí. Menším hráčům v této oblasti pomáhá snaha výrobců užitkových vozidel rozdělit poptávku mezi několik subjektů z důvodu případného dokrytí výpadku jednoho dodavatele od ostatních. Pokud budeme uvažovat míru konkurence pouze u české pobočky, tak dospějeme k podobnému závěru. V blízkém okolí se nachází výrobní závod společnosti Benteler a ve zhruba 20Km vzdáleném Frýdlantě v Čechách je výrobní komplex společnosti TRW. Lokální odběratelé si tedy mohou vybrat produkty od tří velkých světových výrobců brzdových systémů prakticky se stejnými náklady na dopravu.

5 Vlastní řešení

V této kapitole se budu věnovat vlastnímu řešení problému hledání optimálního rozmístění skladovacích a výrobních prostor z pohledu dopravních nákladů.

Jak jsem uvedl v úvodu, k řešení problému jsem vytvořil aplikaci v prostředí MATLAB, která využívá rozšíření Global optimization Toolbox. Poskytované knihovny pro numerické výpočty dosahují velmi vysoké kvality a pokrývají široké spektrum úloh, nicméně jsem se při vývoji aplikace potýkal s některými nedostatky a nedokonalostmi. Tyto nejspíše vycházejí z historických souvislostí z počátků vývoje v 80. letech a pravděpodobně konzervativního přístupu společnosti Mathworks. Konkrétně jsem narazil na velice nedostatečné možnosti průvodce návrhu grafického rozhraní GUIDE a některé další méně zřejmé skutečnosti při vývoji grafického uživatelského prostředí. Rozhodl jsem se proto vytvořit grafické rozhraní aplikace bez použití pomocných nástrojů. Další výtka by mohla směřovat na velkou funkční redundanci v některých oblastech, jako je např. načítání a ukládání textových souborů, vykreslování obrázků nebo ukládání proměnných do grafických objektů. Je to pravděpodobně důsledek současného udržování několika vysoko a nízko úrovňových funkcí pro dosažení podobného cíle.

Pro některé pokročilé operace čtení vstupních souborů nebo práci s XML dokumenty není v prostředí MATLAB mnoho knihovních funkcí a proto jsem se rozhodl získat chybějící funkčnost využitím některých metod z platformy JAVA. Ve společnosti Mathworks si jsou nejspíš vědomi některých nedostatečně rozvinutých oblastí prostředí MATLAB a proto umožnili používat metody Javy přímo ve zdrojových souborech implementací syntaxe jazyka Java. Kvůli některým kolizím s původní syntaxí jazyka MATLAB však bylo nutné provést menší úpravy a proto nelze používat existující kód v jazyku Java bez úprav.

5.1 Databáze světových měst společnosti MaxMind Inc.

Při hledání optimálního umístění výrobních závodů a logistických skladů jsem se potýkal s problémem jak určit vhodné umístění z pohledu dostupné pracovní síly.

Výsledkem optimalizace pouze na základě geografické vzdálenosti může být umístění závodu nebo skladu v nedostupném terénu nebo neobydlené oblasti. Toto jsem vyřešil využitím veřejně přístupné offline databáze společnosti MaxMind Inc., která obsahuje více něž 3 miliony světových měst se souřadnicemi a u zhruba 48 tisíc měst obsahuje navíc i velikost populace, která začíná už na několika stovkách obyvatel. Databázi lze získat v textové formě v soubor formátu CSV, který zabírá zhruba 127MB, a je ji možné používat pro jakékoliv účely. Podmínky užití nařizují uvedení formulace "This product includes data created by MaxMind, available from http://www.maxmind.com/" v dokumentaci, čímž je tato podmínka pro mnou vyvinutou aplikaci splněna.

Obrázek 5.1: Logo společnosti MaxMind

Je velice obtížné si představit důvody, které by vedly k umístění výrobního nebo přepravního uzlu jinde, než ve městech uložených v databázi. Proto jsou při výpočtu pomocí geografické dopravní sítě souřadnice měst uvažovány jako potenciální body optimálního řešení.

Databáze má následující tvar:

```
Country,City,AccentCity,Region,Population,Latitude,Longitude
cz,brno,Brno,78,377407,49.2,16.6333333
cz,broumov,Broumov,70,8325,50.5833333,16.3333333
cz,brtnice,Brtnice,80,3641,49.3166667,15.6833333
cz,bruntal,Bruntál,85,17610,49.9833333,17.4666667
cz,brusperk,Brušperk,85,3573,49.7,18.2166667
cz,buchlovice,Buchlovice,90,2437,49.0833333,17.3333333
```

První sloupec obsahuje dvoupísmenný kód země podle normy ISO 3166, ve které se město nachází. Druhý sloupec tvoří jméno města ve znakové sadě ASCII tzn. bez diakritiky a dále bez velkých písmen. Pro oficiální lokální název je vyčleněn třetí sloupec v kódování ISO-8859-1. Čtvrtý sloupec nese označení státu nebo regionu města podle normy ISO-3166-2 pro USA a podle FIPS 10-4 pro ostatní části světa. Další sloupec, který může být prázdný, nám sděluje počet obyvatel města. Poslední dva sloupce jsou hodnoty ve stupních a představují zeměpisnou šířku a délku. Protože autorem databáze je americká společnost, je místo desetinné čárky používána desetinná tečka.

5.2 Výpočet vzdáleností mezi městy z databáze

Prvním krokem k určení vhodného umístění pro výrobní závod nebo logistický sklad je databáze měst. Samotná databáze by nebyla k ničemu, pokud bychom neznali dopravní vzdálenosti mezi městy (obecně dopravními uzly).

Nejjednodušší způsob určování vzdálenosti mezi uzly je využití obyčejné Pythagorovy věty. Zjistíme rozdíly světových souřadnic počátečního a cílového města, které dosadíme do věty a výsledek vynásobíme typickou ujetou vzdáleností na stupeň obvyklou v uvažovaných zeměpisných šířkách. Toto není příliš přesné určení dopravní vzdálenosti mezi městy, ale pokud bychom počítaly v oblasti s hustou silniční zástavbou, mohli bychom se dobrat použitelných výsledků. Nevýhoda spočívá v potřebě vzorových koeficientů pro převod vypočítané hodnoty na předpokládanou ujetou vzdálenost.

Z hlediska výpočtu složitější, ale rovněž jednoduchou a lehce přesnější metodou pro určení vzdálenosti je využití tzv. Haversine metody, která počítá vzdálenost na ideální kouli.(21) Výpočet můžeme rozdělit do dvou kroků:

1. $$var = \sin^2\left(\frac{lat2 - lat1}{2}\right) + \cos(lat1) * \cos(lat2) * \sin^2\left(\frac{lon2 - lon1}{2}\right)$$

2. $$distance = r * 2 * \arctan\left(\frac{\sqrt{(var)}}{\sqrt{(1 - var)}}\right)$$

lat1 – zeměpisná šířka prvního bodu

lon1 – zeměpisná délka prvního bodu

lat2 – zeměpisná šířka druhého bodu

lon2 – zeměpisná délka druhého bodu

r – poloměr uvažované ideální koule (v případě země 6371Km)

Země sice zcela neodpovídá ideální kouli, ale vzhledem k míře přesnosti, s jakou tu pracuji, není toto zásadní. U vypočítaných vzdáleností můžeme říci, že reálná dopravní vzdálenost nebude kratší. Hodnotu získanou pomocí metody Haversine bychom mohli přirovnat k vzdálenosti vzdušnou čarou. Při husté nebo dobře navržené

dopravní síti můžeme počítat s vyšší efektivností přepravy než v případě řídké sítě, která musí navíc překonávat terénní překážky. Vypočítanou vzdálenost vzdušnou čarou tedy násobíme koeficientem vetším než 1. Pokud budeme uvažovat vzdálenost mezi Brnem a Prahou, pomocí Haversina metody vypočítáme vzdálenost 184,3Km, přičemž vzdálenost po silniční dopravní síti, kterou tvoří dálnice D1, je 208Km. To odpovídá koeficientu přibližně 1,13. Při výpočtu vzdálenosti na vzorovém příkladu dosáhla vyvinutá aplikace pomocí Haversina metody délku nejkratšího dopravního spojení na dopravní síti mezi městy v ČR s počtem obyvatel vetším než 10000 vzdálenost 540.65Km. Ten samý příklad vypočtený pomocí Dijkstrova algoritmu a reálných vzdáleností dopravních cest mezi městy byl optimalizován až na hodnotu 579.46Km. To odpovídá koeficientu zhruba 1.07.

Výpočet dopravní vzdálenosti pouze ze zeměpisných souřadnic je pro silniční dopravu poměrně nepřesný. Tento způsob určování vzdálenosti se hodí hlavně při letecké dopravě mezi městy s vetším počtem obyvatel, která mívají nedaleko umístěné letiště. Pro silniční dopravu je vhodnější způsob uvedený v následující kapitole.

5.3 The Google Distance Matrix API

Nejpřesnějším způsobem určování silničních dopravních vzdáleností je výpočet na základě detailních informací o umístění dopravní trasy nebo fyzickým změřením pomocí projetí trasy vozidlem. Protože oba způsoby jsou pro potřeby aplikace vyvinuté v prostředí MATLAB nevyhovující, využiji webovou službu společnosti Google nazvanou Distance Matrix API, která v reálném čase odpovídá na dotazy o vzdálenosti mezi body. Tyto body mohou být specifikovány jak názvem, tak, a to je pro nás podstatnější, zeměpisnými souřadnicemi. Souřadnice budou čerpány z výše zmíněné databáze světových měst.

5.3.1 Formát dotazu

Dotaz webové službě se podává prostřednictvím parametrů uvedených ve webové adrese za znakem „?“. V případě využití více parametrů se oddělují znakem „&“. Obecný formát má následující tvar: (3)

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

Červeně označený řetězec specifikuje formát odpovědi webové služby. Ta může být ve formátu JSON nebo XML. Modře je označená část s parametry. Ty mohou být:

- *origins* (povinný) – jedna nebo více adres v podobě názvu místa nebo zeměpisné souřadnice.
- *destinations* (povinný) – jedna nebo více adres v podobě názvu místa nebo zeměpisné souřadnice.
- *sensor* (povinný) – nabývá hodnoty *true* nebo *false* a informuje webovou službu o využívání odpovědi v navigačním zařízení.

Další parametry služby si již uvádět nebudeme, protože jsou nepovinné a pro potřeby aplikace jsou vhodné jejich výchozí hodnoty.

Konkrétní dotaz na webovou službu Distance Matrix API může mít následující podobu:

```
http://maps.googleapis.com/maps/api/distancematrix/xml?
origins=Praha|Pardubice&destinations=Brno|Liberec&sensor=false
```

Odpovědí na tento dotaz bude XML dokument, který bude obsahovat údaje o vzdálenosti ze všech počátečních měst do všech cílových. Údaje mohou být zaneseny do tzv. matice sousednosti, která bude mít následující tvar:

	Brno	Liberec
Praha	209 km	109 km
Pardubice	146 km	152 km

5.3.2 Omezení služby

Poskytování služby je rozděleno na dvě kategorie. První je volné užití, kde jsou kladeny tyto omezující podmínky:(3)

- Počet cest mezi počátečními a koncovými pozicemi nesmí být v jednom dotazu vyšší než 100.
- Počet cest mezi počátečními a koncovými pozicemi nesmí přesáhnout 100 v průběhu posledních 10 sekund (dotazů může být učiněno několik).
- V průběhu posledních 24 hodin mohou být uskutečněny dotazy na 2500 cest

mezi počátečními a koncovými pozicemi.

- Maximální délka dotazu ve formě webové adresy může mít 2048 znaků.
- Aplikace přistupující k webové službě Distance Matrix API může zobrazovat získaná data pouze na Google mapě.

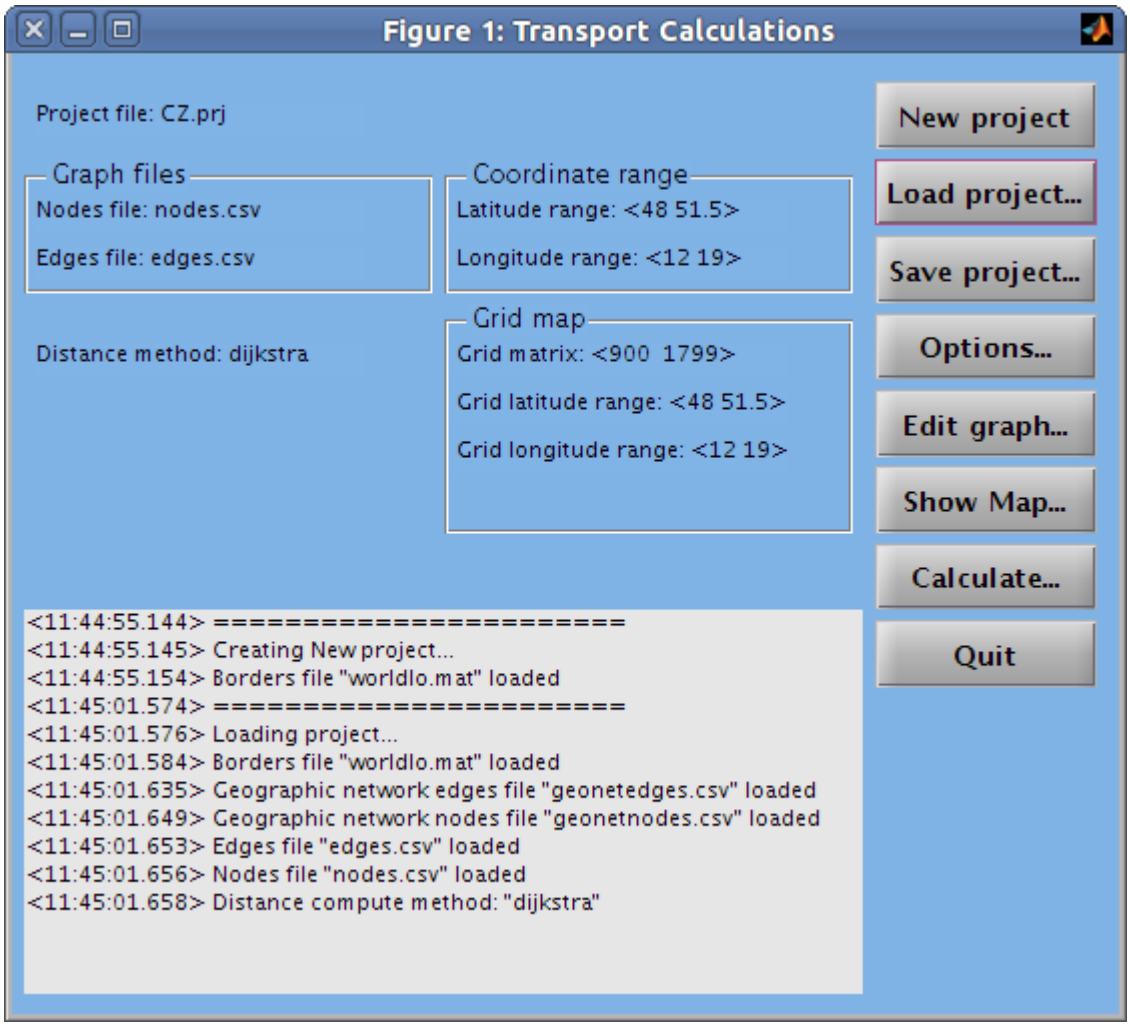
Podmínku danou posledním omezením jsem v aplikaci splnil zobrazením Google mapy, která se identifikuje logem „Google maps“.

5.4 Grafické prostředí aplikace

Grafické prostředí aplikace jsem vytvořil pomocí grafických prvků znázorněných na obrázku 3.10, kterým jsem přiřazoval potřebné vlastnosti až v rámci zdrojového kódu. Jedná se tedy o „ruční“ tvorbu grafického prostředí, která je časově náročnější, ale na druhou stranu přináší větší kontrolu nad obsahem aplikace a umožňuje někdy nastavovat parametry, které nejsou součástí nástroje GUIDE. Někdy jsou tak dostupná užitečná rozšíření, která však jsou nedokumentovaná a není zajištěna stejná funkčnost i v následující verzi prostředí MATLAB. Grafické prvky jsou v relativním vztahu s velikostí okna, které je rovněž v relativním vztahu s velikostí obrazovky. Plocha okna a grafických prvků bude tedy vždy zabírat stejnou plochu na každém zobrazovacím zařízení.

5.4.1 Hlavní okno

Hlavní okno aplikace nese důležité údaje řešené úlohy a základní ovládací prvky. Ve spodní části je informativní pole, kde se zobrazují všechny důležité události s časem, kdy nastaly. Panel „Graph files“ obsahuje informace o načtených souborech grafu toků. Panel „Coordinate range“ obsahuje údaje o rozsahu, nad kterým probíhá optimalizace. Hledané optimální řešení se uvažuje pouze tomto rozsahu. Panel „Grid map“ nese údaje o dvoustavové mapě, která slouží k určení, zda-li se uvažovaná pozice nachází na souši. Dále je v hlavním okně ještě uvedena informace o jméně hlavního souboru úlohy a metodě pro výpočet vzdálenosti bodů.

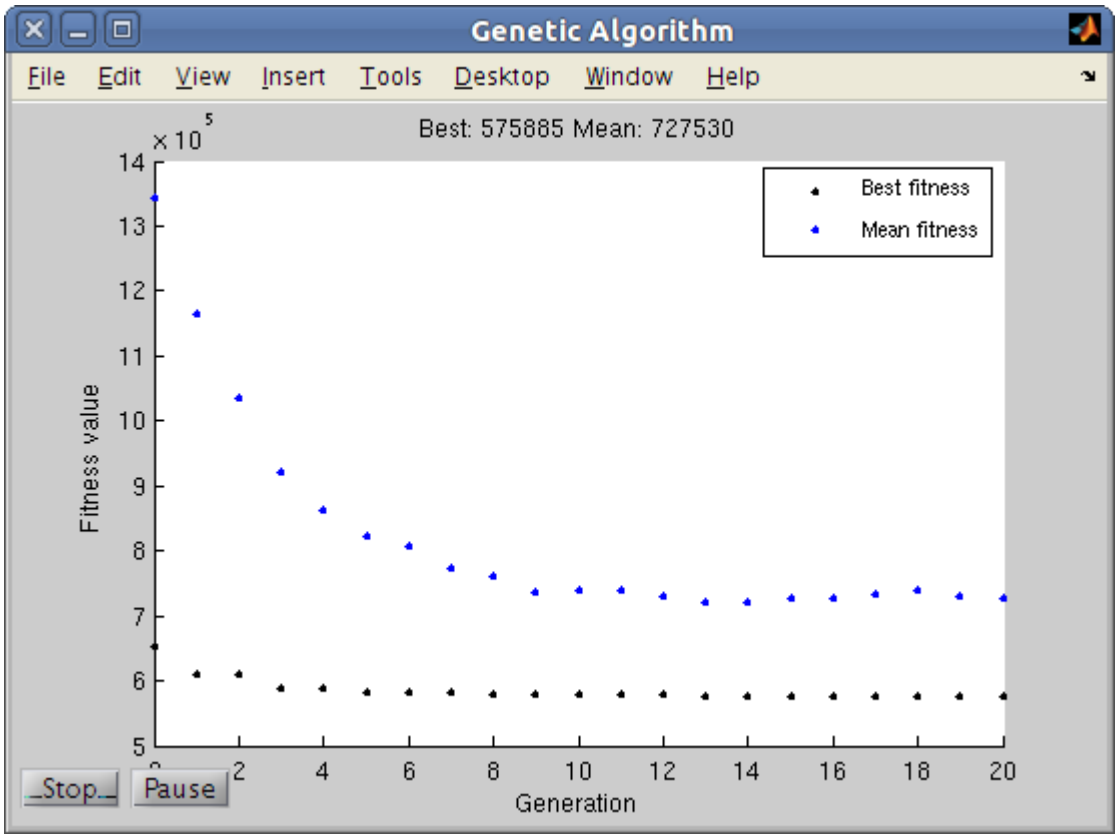


Obrázek 5.2: Hlavní okno aplikace

Z důvodu potenciálního využití aplikace uživateli, kteří pochází ze zahraničí, je rozhraní vytvořeno v Anglickém jazyce. Funkce ovládacích prvků je následující:

- *New project* – Vytvoří novou úlohu se základními parametry. Toto se děje i při prvotním spuštění aplikace.
- *Load project...* – Tímto prvkem otevřeme dialog pro načtení hlavního souboru úlohy.
- *Save project...* – Otevře dialog pro výběr hlavního souboru úlohy, do kterého je zapsána konfigurace a následně zapsány další datové soubory.
- *Options...* – Otevře okno programu s nastavením aktuální úlohy.

- *Edit graph...* – Otevře okno programu, ve kterém je možné editovat graf toků materiálu, polotovarů a finálních produktů.
- *Show Map...* – Otevře okno programu se znázorněním všech definovaných uzlů grafu toků a v případě existence dopravní sítě i ji.
- *Calculate...* – Spustí proces optimalizace nad definovaným grafem toků a v grafu s hodnotami nejlepšího jedince v generaci a průměru celé generace zobrazí průběžné výsledky. Po dokončení optimalizace je v hlavním okně v informačním poli zobrazen výsledek nejlepšího nalezeného řešení.



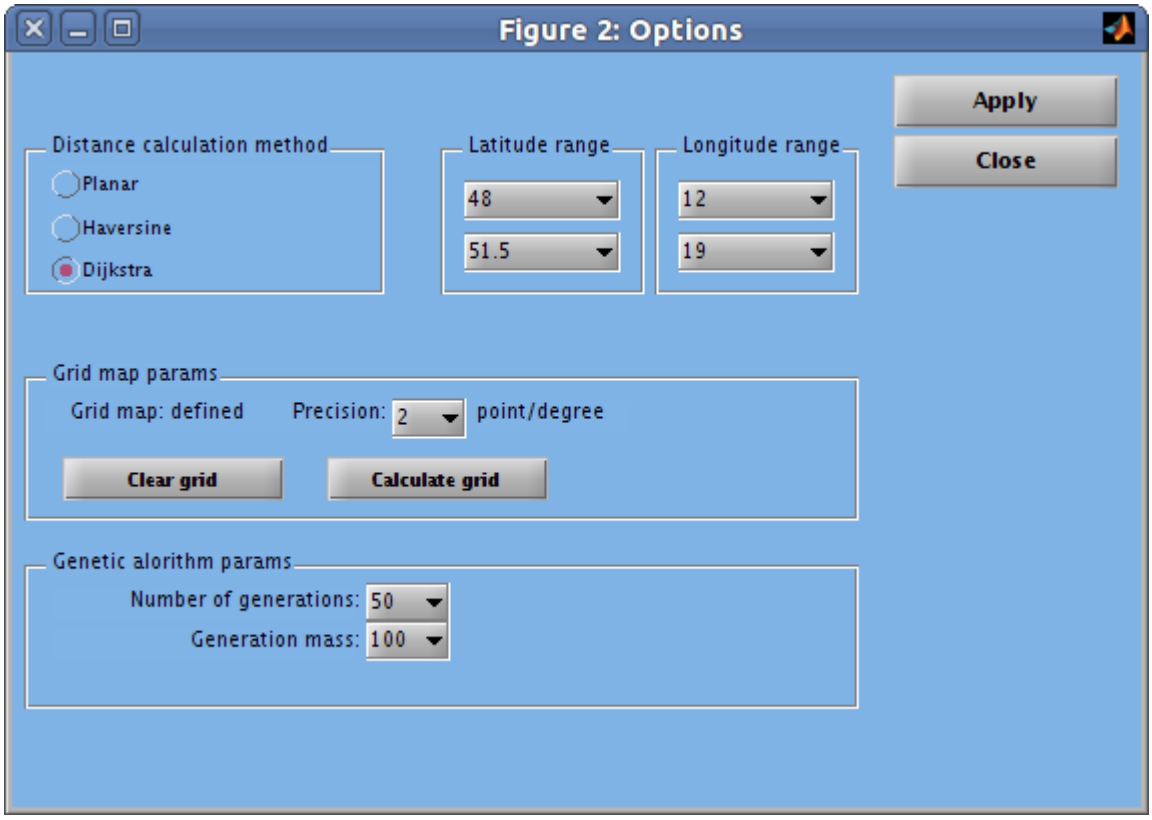
Obrázek 5.3: Průběh optimalizace úlohy

- *Quit* – Ukončí aplikaci a veškerá neuložená data budou ztracena.

5.4.2 Okno nastavení

Okno nastavení je přístupné z hlavního okna pomocí tlačítka *Options...* a umožňuje měnit zásadní parametry řešené úlohy. Tyto parametry jsou téměř všechny ukládány do hlavního souboru uložené úlohy, který má příponu *.prj*.

Panel „Distance calculation method“ slouží ke zvolení metody pro výpočet vzdálenosti mezi uzly uvažovaného řešení. Pokud není dostupná geografická sít definující dopravní cesty a jejich vzdálenosti, není možné zvolit metodu s využitím Dijkstrova algoritmu. Panely „Latitude range“ a „Longitude range“ definují oblast, nad kterou bude genetický algoritmus hledat optimální řešení. Panel „Grid map params“ informuje o existenci dvoustavové mapy, která definuje pevninu a vodní plochu. Tuto mapu můžeme vymazat pomocí tlačítka „Clear grid“. Dále se v panelu nachází menu pro zvolení počtu bodů nově generované mapy a poslední prvek v podobě tlačítka „Calculate grid“ slouží k vygenerování nové mapy. Poslední panel s názvem „Genetic algorithm params“ obsahuje dvě menu pro zvolení zásadních parametrů výpočtu pomocí genetického algoritmu a těmi jsou počet generací výpočtu a množství jedinců v jedné generaci.

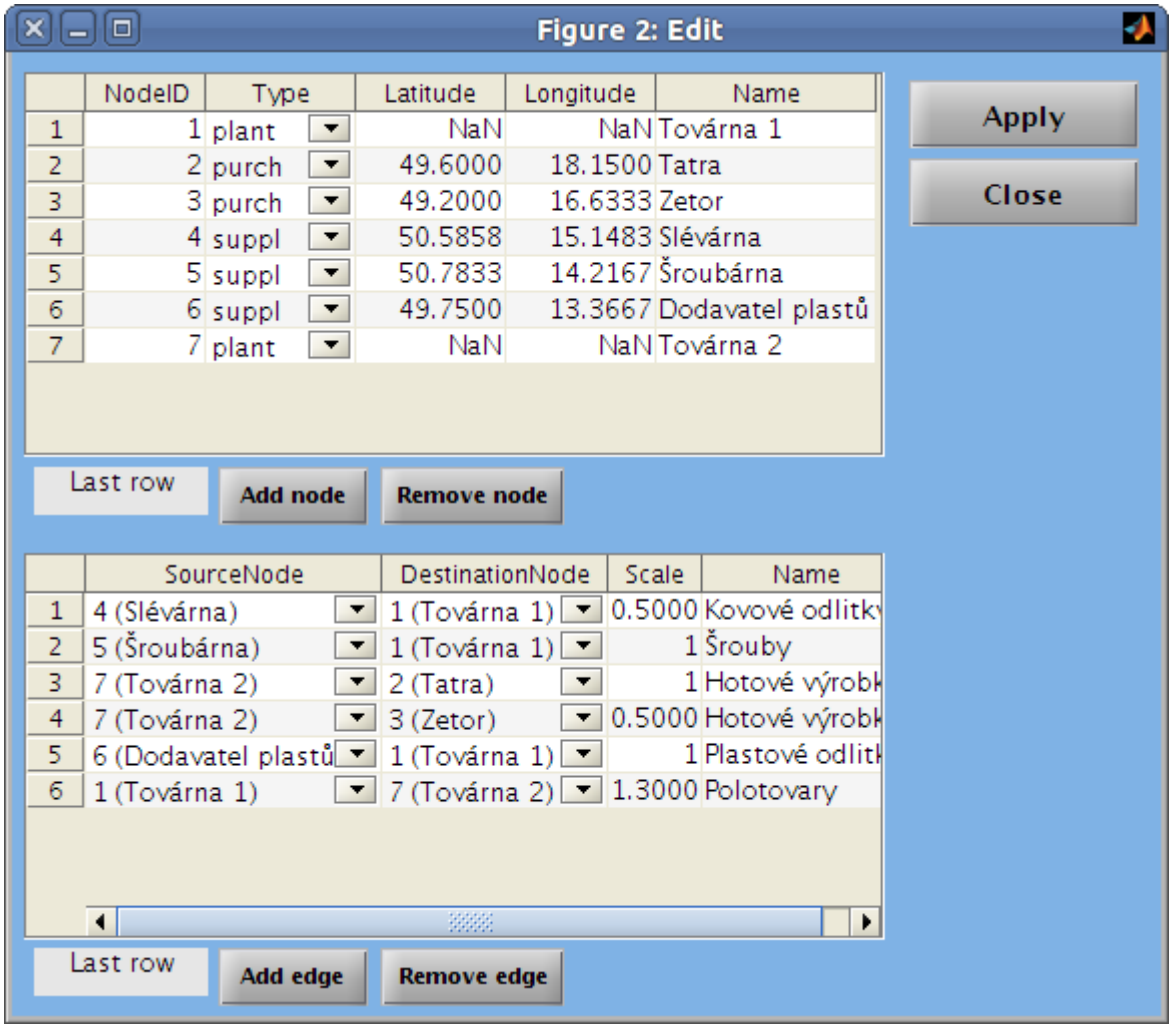


Obrázek 5.4: Okno nastavení.

Dále se v okně nachází tlačítko „Apply“ pro zavření okna a zapsání provedených změn a tlačítko „Close“ pro zavření bez uložení.

5.4.3 Okno editace grafu toků

Graf toků tvoří zásadní část, která popisuje vytížení dopravních cest a tím jim přiřazuje určitou důležitost. Samotné vytížení může být zaměněno např. za výši obratu nebo zisku generovaného z uvedené dopravní cesty. Na obrázku 5.5 je zobrazena editace na fiktivním grafu toků.



Obrázek 5.5: Editace parametrů grafu toků.

Okno obsahuje dvě tabulky, ve kterých je uložen seznam uzlů grafu a cest grafu. Tyto tabulky jsou editovatelné a mohou jim být přidávány nebo odebírány řádky prostřednictvím přidružených tlačítek. Informační pole vedle tlačítek zobrazuje

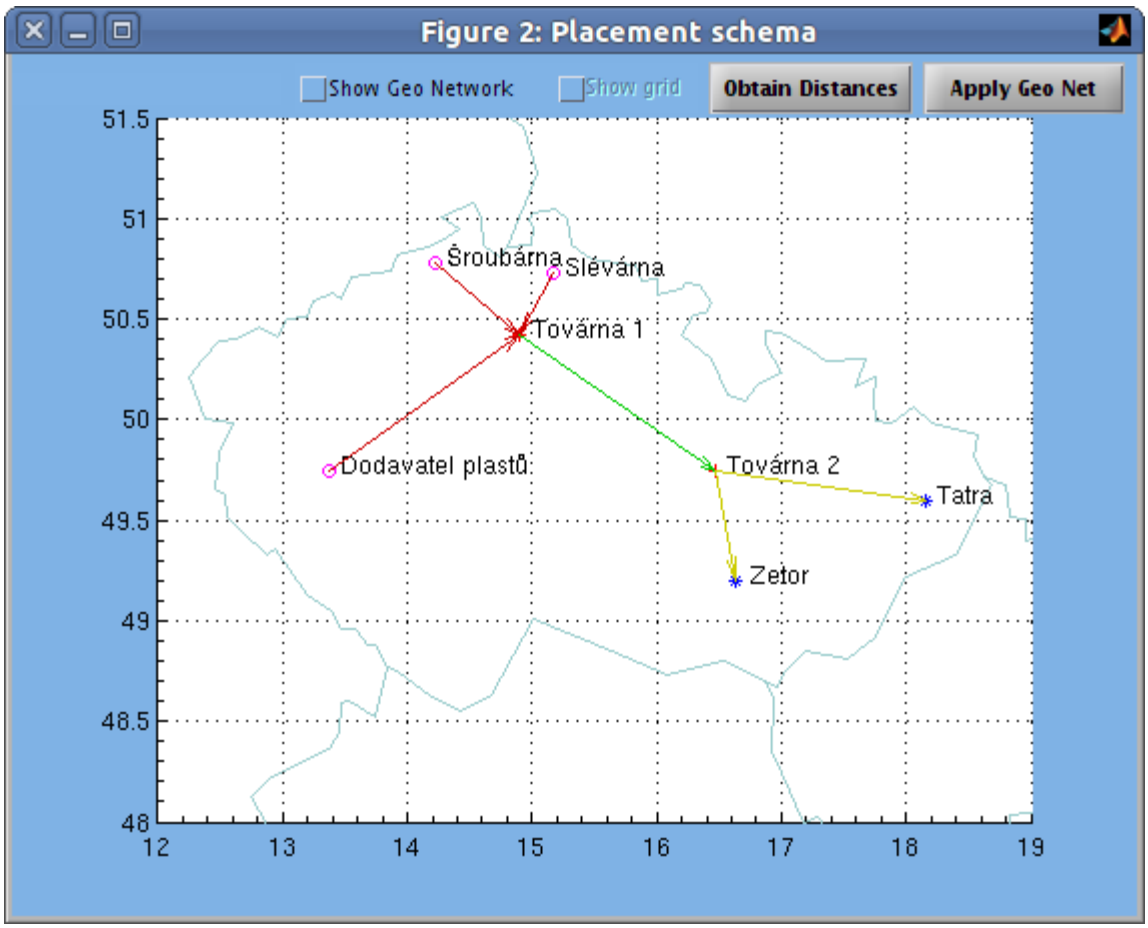
označené řádky. V případě žádného označeného se uvažuje poslední řádek tabulky. Tlačítko „Remove node“ a „Remove edge“ odebere označené řádky z tabulky s uzly resp. hranami a tlačítko „Add node“ a „Add edge“ označené řádky přidá na konec tabulky s uzly resp. hranami.

V tabulce se seznamem uzlů grafu je první sloupec tvořen unikátním identifikačním klíčem. Tento není možné editovat a aplikace se stará o dodržení podmínky jedinečnosti sama. Tabulka se seznamem cest pracuje s tímto jednoznačným identifikátorem a musí se vždy nacházet v konzistentním stavu tzn. tabulka s cestami nemůže obsahovat cestu od nebo k neexistujícímu uzlu.

Provedené změny v grafu toků zapíšeme pomocí tlačítka „Apply“ a tím okno zavřeme. Tlačítko „Close“ okno zavře bez uložení změn.

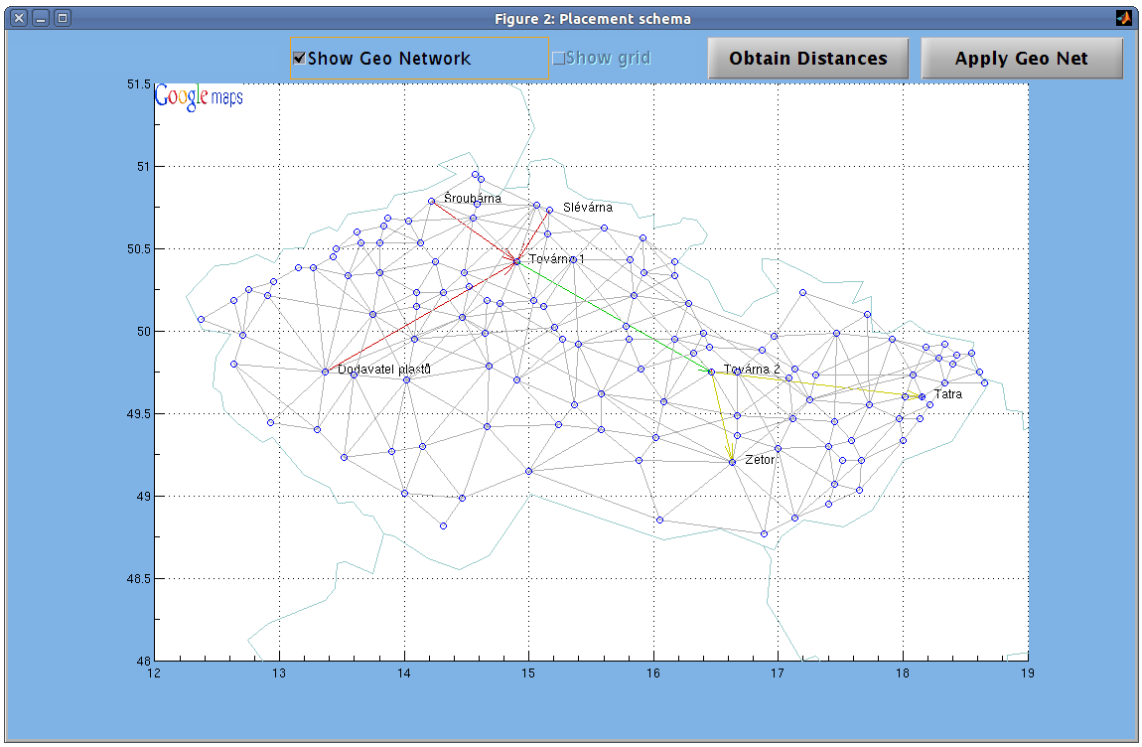
5.4.4 Okno zobrazení mapy

Hlavní část okna tvoří vykreslení oblasti specifikované v nastavení v panelech „Latitude range“ a „Longitude range“. Každý bod grafu toků s udanou polohou je zobrazen a každá cesta mezi uzly s polohou je zobrazena. Pro lepší orientaci se zobrazí i státní hranice a pobřeží kontinentů. V případě obrázku 5.6 není pobřeží vykresleno.



Obrázek 5.6: Výsledek optimalizace zobrazen na mapě.

Na mapě vidíme tok materiálu naznačený šipkami. Od dodavatelů, které tvoří Šroubárna, Slévárna a Dodavatel plastů, je materiál dovážen do výrobního závodu Továrna 1. Ten materiál zpracuje a polotovar putuje do Továrny 2, která může výrobek dokončit. Továrna 2 může také představoval logistický sklad, kde jsou dodávky z první továrny rozdělovány. Hotové výrobky pak putují k odběratelům Zetor a Tatra. V praxi by bylo udržování dvou továren pravděpodobně neoptimální, ale musíme mít na paměti, že uvedená aplikace se zabývá pouze dopravními náklady. Ty jsou v uvedeném případě nejmenší. Při zobrazení dopravní sítě zjistíme, že Továrna 1 se nachází v Mladé Boleslavi a Továrna 2 ve Svitavách.



Obrázek 5.7: Zobrazení dopravní sítě.

Na obrázku 5.7 vidíme dopravní síť, ve které figurují všechny města v ČR s počtem obyvatel větším než 10000 a šedivě označené dopravní spojení. Cesty jsou vytvořeny uživatelem a jejich vzdálenosti, pokud již nejsou známy, se zjistí stisknutím tlačítka „Obtain Distances“. Aplikace se začne dotazovat webové služby Google Distance Matrix API a označí cesty, ke kterým byly získány nové vzdálenosti. Pokud se u cesty nepodaří zjistit vzdálenost, je červeně označena a neuvažuje se při optimalizaci dopravních nákladů. Pro uložení změn v dopravní síti slouží tlačítko „Apply Geo Net“. V případě existence moří a oceánů v zobrazeném rozsahu je aktivní zaškrtávací políčko „Show grid“, které zobrazí plochy nevyhovující k umístění uzlu grafu toků.

5.5 Architektura aplikace

Aplikace se skládá z několika souborů:

5.5.1 compute.m

Soubor obsahuje rutinní operace pro přípravu dat uložených v hlavní datové struktuře project. Vytváří nastavení potřebná pro spuštění genetického algoritmu. Součástí souboru jsou funkce pro výpočet všech nejkratších cest mezi všemi vrcholy. Stěžejní

částí jsou pro potřeby algoritmu upravené funkce pro vytvoření populace a pro mutaci populace. Zvláštností funkce mutace je zužování rozsahu u mutovaných genů. Čím pozdější generace, tím užší rozsah a větší šance nalezení lepšího řešení v blízkosti původně mutovaného.

5.5.2 dijkstraAlg.m

V souboru je zapsán Dijkstrův algoritmus, který je používán pro hledání nejkratších cest v grafu, který představuje dopravní síť. Jedná se o implementaci standardního algoritmu s rozšířením o ukládání předchůdců v nejkratší nalezené cestě. To je užitečné pro pozdější rekonstrukci cesty.

5.5.3 findNearestNeighbor.m

Zde je uložena jednoduchá, ale často používaná metoda pro nalezení nejbližšího uzlu dopravní sítě. Málokteré uvažované řešení se nachází přesně v uzlu dopravní sítě a proto se vzdálenost od tohoto uzlu započítává k cestě po dopravní síti. Pomocí operací genetického algoritmu dochází k tíhnutí nejlepších řešení k dopravnímu uzlu.

5.5.4 fitnessDijkstra.m

Provádí převod z genetickým algoritmem používaných diskrétních hodnot na radiány a vypočítává délku cesty pro všechny souřadnice uvažovaného řešení.

5.5.5 fitnessHaversine.m

Platí prakticky to samé, jako v předchozím případě, pouze s rozdílem určování délky ne na základě dopravní sítě, ale povrchové vzdálenosti geometrického útvaru koule s poloměrem 6371Km.

5.5.6 fitnessPlanar.m

Rovněž platí to, co v předchozí dvou případech. Výpočet vzdálenosti se provádí na základě Pythagorovi věty. Tato metoda je nejméně přesná a kvůli absenci převodní tabulky ze stupňů na vzdálenost je použitelná pouze pro odhad pravděpodobného umístění výrobních závodů na malé rozloze.

5.5.7 gui.m

Jedná se o hlavní soubor aplikace. Funkce *gui*, kterou obsahuje je vstupní bod programu a jejím spuštěním se zobrazí grafické rozhraní.

Součástí je i několik funkcí pro vytvoření grafického rozhraní, pro vytvoření nové optimalizační úlohy s výchozími hodnotami, pro aktualizaci grafických informačních prvků, pro výpis informačních a chybových řetězců a funkce pro reakci na události hlavního okna zvané „Callback“.

5.5.8 worldlo.mat

Soubor obsahuje řadu světových geografických informací a pochází z dřívějších vydání prostředí MATLAB. Společnost Mathworks se rozhodla některé informace v souboru obsažené kvůli obtížnému aktualizování a někdy nejasnému požadovanému stavu nedistribuovat. To se týká hlavně státních hranic, které jsou často nejasné a vedou se o ně spory. Proto jsou v aplikaci zobrazené státní hranice spíše informativního charakteru.

5.6 Používané datové soubory

Aplikace používá pro načítání a ukládání řadu souborů. Jejich počet a formát vychází z vnitřního uspořádání datových struktur.

Hlavní soubor úlohy se skládá z volitelného jména a výchozí přípony *prj*. Všechny ostatní datové soubory jsou ukládány do adresáře, který se skládá ze jména hlavního souboru projektu (bez přípony) a přidaného řetězce „_files“. Zde můžeme nalézt následující soubory:

5.6.1 edges.csv

Obsahuje seznam cest mezi vrcholy v grafu toků a má následující tvar:

```
SourceNode;DestinationNode;Scale;Name;
4;1;0.5;Kovové odlitky;
5;1;1;Šrouby;
7;2;1;Hotové výrobky;
```

Sloupce určují popořadě počáteční uzel cesty, koncový uzel cesty, váha resp. důležitost cesty a jméno cesty.

5.6.2 nodes.csv

Obsahuje seznam vrcholů grafu toků a má následující tvar:

```
NodeID;Type;Latitude;Longitude;Name;
1;plant;NaN;NaN;Továrna 1;
2;purch;49.6;18.15;Tatra;
3;purch;49.2;16.6333;Zetor;
```

Sloupce určují popořadě jednoznačné identifikační číslo uzlu, typ uzlu, zeměpisná šířka, zeměpisná délka a jméno uzlu. Nedefinovaná zeměpisná šířka a délka určuje souřadnice, které se budou hledat v rámci optimalizace pomocí genetických algoritmů.

5.6.3 geonetedges.csv

Obsahuje seznam cest mezi vrcholy v grafu dopravní sítě a má následující tvar:

```
SourceNode;DestinationNode;Distance;
86;46;27177.7;
86;2;31146.8;
2;92;42210.6;
92;83;16867;
```

Sloupce určují popořadě zdrojový uzel, cílový uzel a vzdálenost mezi uzly v Km.

5.6.4 geonetnodes.csv

Obsahuje seznam vrcholů v grafu dopravní sítě a má následující tvar:

```
NodeID;Latitude;Longtitude;Name;
1;49.7833;14.6833;Benešov;
2;49.95;14.0833;Beroun;
3;50.5333;13.8;Bílina;
4;49.3667;16.6667;Blansko;
```

Sloupce určují popořadě jednoznačné identifikační číslo uzlu, zeměpisná šířka a délka a poslední název.

5.6.5 gridMap.mat

Soubor v binárním formátu .mat, který obsahuje proměnnou *grid*, *gridLatitude* a *gridLongitude*. První proměnná nese dvoustavové logické hodnoty a představuje mapu o rozložení pevniny v rozsahu uvedených v dalších dvou proměnných. V případě potřeby lze data uchovaná v tomto souboru poměrně lehce vygenerovat v okně nastavení a při uložení úlohy na disk se vytvoří i tento soubor.

6 Přínosy vlastního řešení

Jedním z hlavních požadavků bylo vytvořit univerzální nástroj pro hledání optimálního umístění výrobních závodů a logistických skladů. To se podařilo, ale pro hmatatelný přínos je nutné tento nástroj využít v reálné situaci. Proto aplikaci využiji na několika případech vycházejících z aktuální situ ace závodu v ČR a koncernu v Evropě.

6.1 Optimální umístění existujícího výrobního závodu v rámci ČR

Na optimální polohu výrobního závodu můžeme pohlížet několika způsoby. Předně bych chtěl zdůraznit, že se jedná o optimalizaci pouze dopravních nákladů. Teoreticky by nás měla zajímat délka dopravních cest všech uskutečněných přeprav materiálu a výrobků z a do závodu. To ovšem narušuje jednotná cena za přepravu v rámci ČR u některých menších zásilek. Ty však díky malé důležitosti můžeme zanedbat.

6.2 Vnitrostátní doprava

Pro výpočet použiji předpokládané náklady na přepravu odvíjející se od výše objemu obchodu s jednotlivými dodavateli a odběrateli v ČR. Tržby za dodané produkty v roce 2010 dosahovaly celkem 164,06 mil. Kč.

Hlavními českými dodavateli materiálu za rok 2010 jsou:

- Odlitky ze společnosti BENEŠ a LÁT a.s. se závody v Poříčanech, Slaná-Sutice a Průhonicích
- Lisované díly ze společnosti ZEVETA Bojkovice a.s.
- Šrouby ze společnosti KAMAX s.r.o. se závodem v Turnově

Hlavními českými finálními odběrateli produktů za rok 2010 jsou:

- TATRA a.s. s výrobním závodem v Kopřivnici
- Zetor a.s. s výrobním závodem v Brně
- SOR Libchavy spol. s r.o.

- TEDOM a.s. s výrobním závodem v Jablonci nad Nisou
- Iveco Czech Republic a. s. (Irisbus) s výrobním závodem ve Vysokém Mýtě

Hlavními českými odběrateli produktů za rok 2010 určených pro druhotný trh jsou:

- DORBAS s.r.o. se sídlem ve Fryštáku
- L.A.F. a.s. se sídlem v Liberci
- BRZDY BAUMRUKR s.r.o. se sídlem v Praze
- Karex a.s. se sídlem v Cerekvicích nad Loučnou
- MAKRO - ND spol. s r. o. se sídlem v Moravanech
- TURBOSOL SERVIS spol. s r.o. se sídlem v Brně
- Penax - Petr Sýba s.r.o. se sídlem v Bílině
- AUTOCARRO PARTS s.r.o. se sídlem v Opavě
- TANAP, spol. s r.o. se sídlem v Chotěboři
- KAMAS TRUCK s.r.o. se sídlem ve Velkém meziříčí
- AP servis Poděbrady, s.r.o.
- KDH AUTO MORAVA s.r.o. se sídlem v Přerově a Ostravě

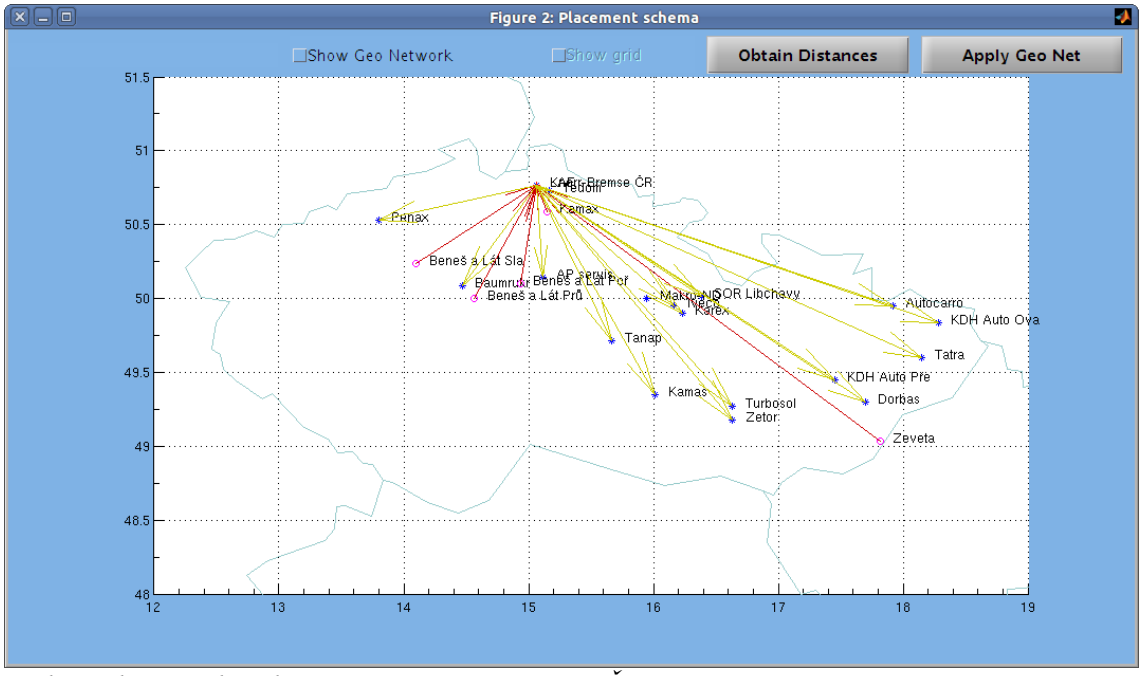
Předpokládané vyjádření množství nákladu dopravovaného od dodavatelů a k odběratelům je zachyceno v následující tabulce:

Odběratelé	Množství	Odběratelé	Množství	Dodavatelé	Množství
Tatra	7	Baumrukr	1.8	Beneš a Lát Poř	5
Zetor	3	Turbosol	1.3	Beneš a Lát Sla	1
SOR Libchavy	4	Penax	1.1	Beneš a Lát Prů	2
Tedom	1.5	Autocarro	1.2	Zeveta	7
Iveco	5	Tanap	0.9	Kamax	3
Dorbas	3	Kamas	0.7		
Karex	2.3	AP servis	0.4		
Makro-ND	2.1	KDH Auto Pře	0.2		
LAF	2	KDH Auto Ova	0.3		

Tabulka 6.1: Množství přepraveného materiálu

Přesné množství dodaných dílů je předmětem obchodního tajemství společnosti Knorr-Bremse ČR a proto jsou uvedené hodnoty přibližné a jsou vyjádřeny pouze relativně. I přes to je možné navrhnout opatření pro optimalizaci dopravních nákladů.

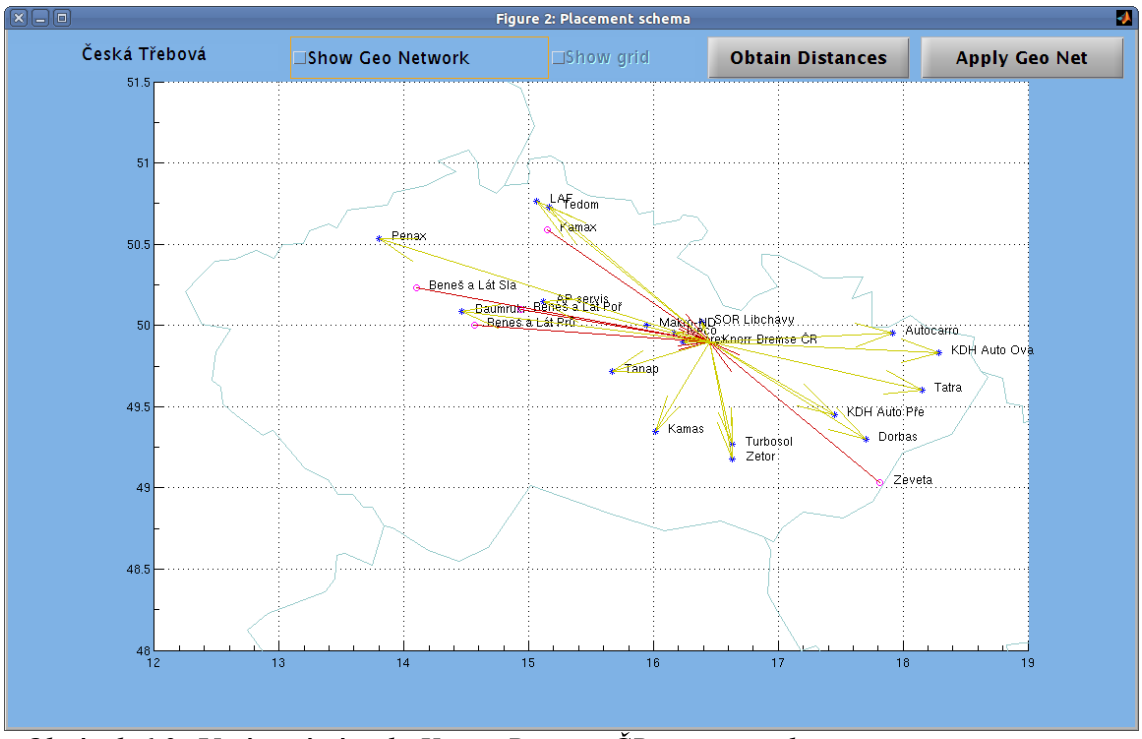
Rozmístění odběratelů, dodavatelů a dopravních cest je znázorněno na obrázku 6.1:



Obrázek 6.1: Aktuální situace Knorr-Bremse ČR

Červeně se zobrazují cesty od dodavatelů a žlutě cesty k odběratelům. Tato situace není optimální. Součet délky tras vynásobený příslušným koeficientem podle množství přepravovaného materiálu dosahuje hodnoty zhruba 9358. Toto číslo vystihuje aktuální situaci. Pokusíme se o nalezení lepšího umístění závodu pomocí optimalizace za přispění genetického algoritmu.

Po optimalizaci na nejkratší vzdálenosti k českým dodavatelům a odběratelům je výsledné řešení uvedeno na následujícím obrázku 6.2:



Obrázek 6.2: Umístění závodu Knorr-Bremse ČR po optimalizaci

Můžeme si povšimnout umístění závodu zhruba ve středu mezi dodavateli a odběrateli v uzlu dopravní sítě, který představuje město Česká Třebová. Toto je optimální umístění s hodnotou náročnosti dopravy 5907. V porovnání s aktuální lokací se v optimálním umístění zkrátí délka vnitrostátních dopravních cest na zhruba 63%.

Prostor pro potenciální snížení nákladů bezprostředně spojených s množstvím ujeté vzdálenosti představuje až 37%.

6.3 Mezinárodní doprava

Protože zásadní část tržeb dceřinné společnosti Knorr-Bremse v ČR představuje obchod se zahraničními společnostmi koncernu, provedeme si optimalizaci umístění závodu v situaci, kdy je drtivá většina výrobků exportována do zahraničí. Pro určení množství produktů dopravených do zahraničí velmi dobře poslouží obrat uvedený v tabulce 4.3. Pro rozdělení exportu na zahraniční společnosti použijeme udaje v tabulce 4.4. Pro zjednodušení úlohy zanedbáme nevýznamné partnery a budeme uvažovat pouze prvních 5 společností, které dohromady tvoří téměř 87% exportu a více než 98% importu.

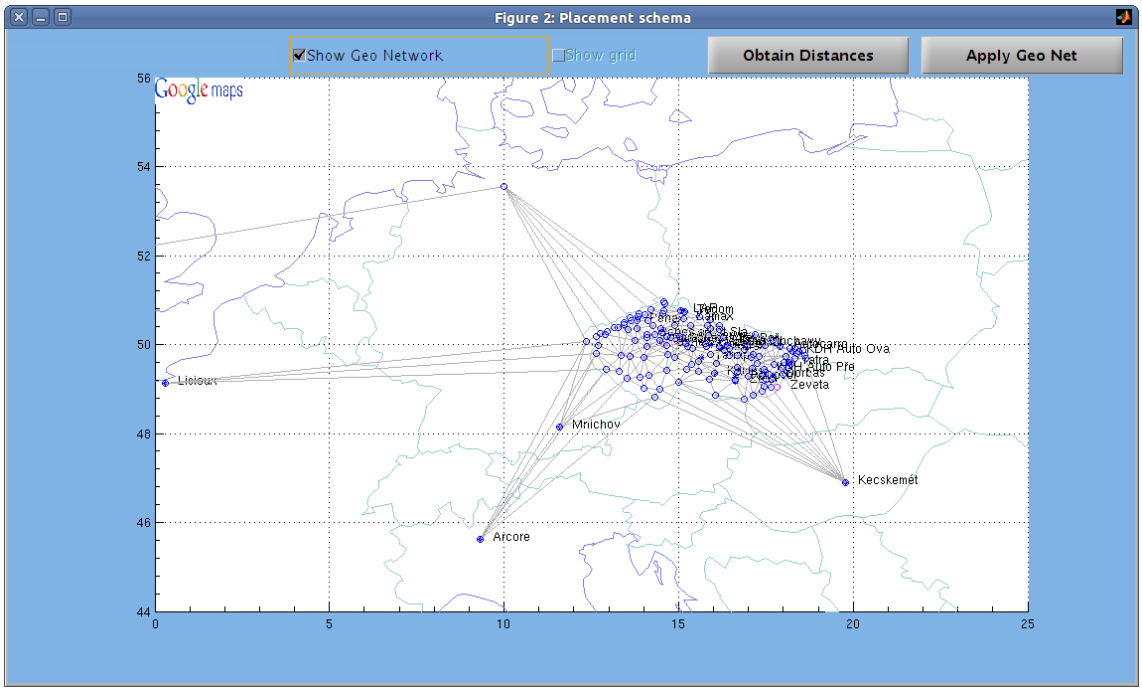
K obchodním partnerům uvedeným v předchozí kapitole přidáme ještě následující zahraničních cílová místa:

Odběratelé	Množství
KNORR Německo	164
KNORR Maďarsko	88
KNORR Bendix (USA)	18
KNORR Francie	15
KNORR Itálie	15

Tabulka 6.3: Množství odebraných výrobků

Množství odebraných výrobků je v relaci s množstvím tuzemských odběratelů. Pro současnou pozici výrobního závodu v Liberci je vypočítaná hodnota náročnosti dopravy 301510. Toto číslo velmi ovlivnily objemné dodávky do zahraničí.

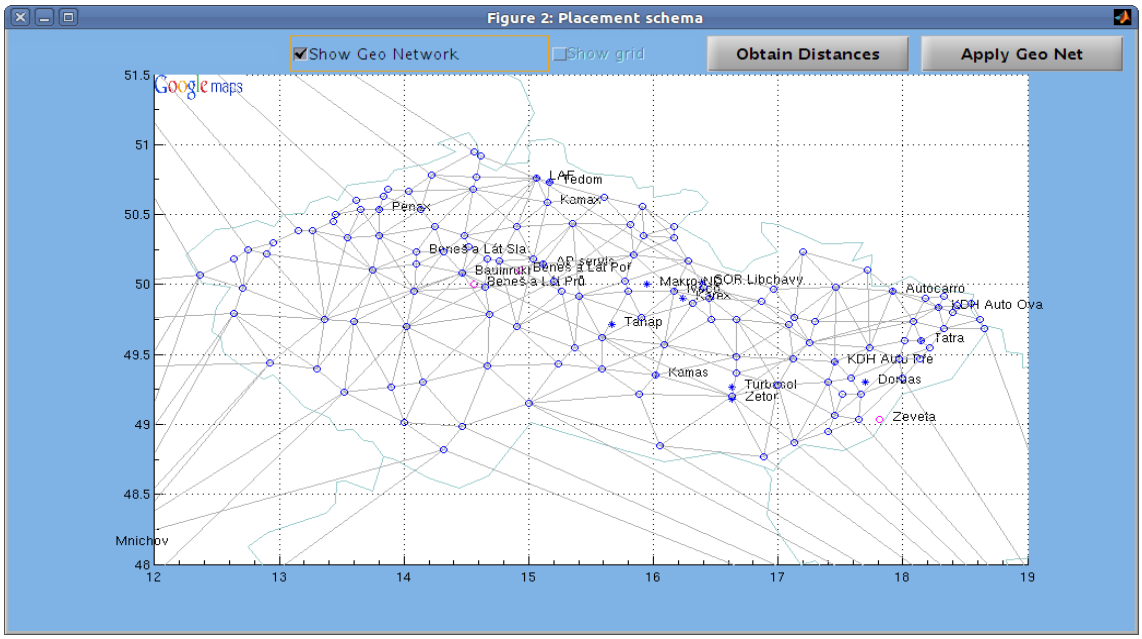
Dopravní síť, nad kterou bude probíhat optimalizace je následující:



Obrázek 6.3: Rozšířená dopravní síť

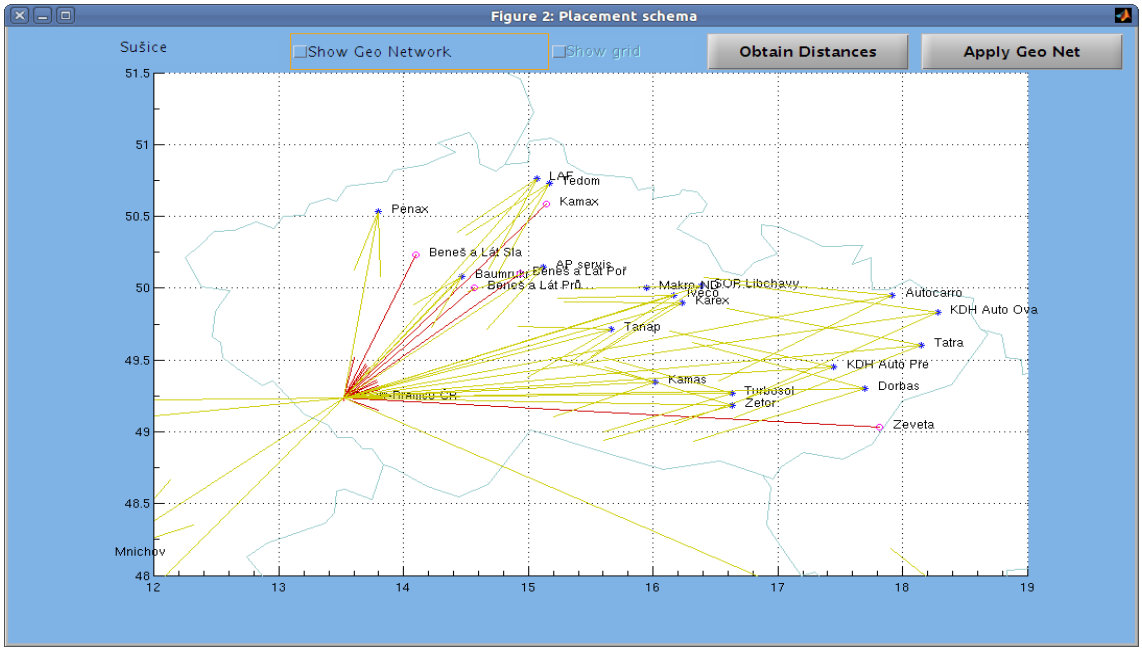
Přeprava do amerického závodu je realizována prostřednictvím přístavu v Hamburku. Ostatní závody v Mnichově, Lisieux, Arcore a Kecskemétu mají přímé spojení prostřednictvím silniční dopravní sítě.

Pro hledání řešení ve správném rozsahu musíme změnit parametry zeměpisné šířky a délky, jako to je na následujícím obrázku:



Obrázek 6.4: Užší rozsah přípustných řešení

Výsledkem optimalizace je hodnota 267618, která představuje snížení dopravní vzdálenosti na 88.76%. To již nepředstavuje tak vysoké procentuální zlepšení jako v případě optimalizace v rámci tuzemských dodavatelů, ale s přihlédnutím k faktu, že se ze závodu vypravuje do zahraničí zhruba jeden až dva plně naložené kamióny s přívěsem denně, může hrát toto snížení vzdálenosti podstatnou roli. Graficky znázorněný výsledek optimalizace se zahraničními cestami má následující podobu:



Obrázek 6.5: Optimální uzel v podobě města Sušice

7 Závěr

V diplomové práci jsem si dal nelehký úkol využít genetické algoritmy pro podporu manažerského rozhodování v podniku a to prostřednictvím optimalizace umístění výrobního závodu společnosti KNORR-BREMSE Systémy pro užitková vozidla ČR, s.r.o.. Toto zadání se ukázalo býti natolik obecné a náročné, až muselo dojít k jeho zúžení na konkrétnější problém, kterým se stala optimalizace dopravních vzdáleností materiálu a produktů společnosti.

Úkolem řešení bylo vyvinutí grafické aplikace v prostředí MATLAB, která by optimalizaci vykonávala. To se podařilo a obecnější pojetí aplikace dává možnost jejího použití k efektivnímu řešení dopravních problémů podobného typu. Nabízí se také možnost pokračování ve vývoji a zvyšování funkcionality např. o řešení obecného dopravního problému, který spočívá v optimálním výběru dodavatelů s omezenou kapacitou pro zásobování několika odběratelských míst. V průběhu vývoje jsem se dále seznámil se zajímavou webovou službou společnosti Google, která může lehce posloužit k řešení dopravních problémů. Zajímavým tématem do budoucna bude v této oblasti identifikování aktuálně přetížených dopravních komunikací a uzpůsobování cesty v reálném čase. To by mohlo přinést snížení dopravních nákladů mnoha společnostem.

Závěry práce jsou pro společnost Knorr-Bremse minimálně k zamyšlení. V rámci tuzemských odběratelů se podařilo optimálním umístěním výrobního závodu v České Třebové snížit vzdálenosti mezi dodavateli a odběrateli o velmi solidních 37%. Solidní výsledek optimalizace byl pozorovatelný i při zaměření se na celkovou, tedy hlavně přeshraniční, dopravu, kdy přesunutím výrobního závodu pouze v rámci ČR do města Sušice lze dosáhnout snížení dopravní vzdálenosti o více než 11%.

Problematika optimalizace rozmístění výrobních kapacit nebude pravděpodobně asi nikdy zcela uzavřena a bude se tedy jednat o trvalou snahu hledání lepších řešení.

8 Literatura

(1) BRENNAN, G.. *Knorr-Bremse AG*. International Directory of Company Histories. 2007 [cit. 2011-5-22]. Dostupné z <http://www.encyclopedia.com/doc/1G2-3480000060.html>

(2) DOSTÁL, P. *Pokročilé metody analýz a modelování v podnikatelství a veřejné správě*. 1. Vyd. Brno: CERM, 2008. 340 s. ISBN 978-80-7204-605-8.

(3) GOOGLE INC. *The Google Distance Matrix API Documentation.* [online]. 2012 [cit. 2012-1-15]. Dostupné z <http://code.google.com/apis/maps/documentation/distancematrix/>

(4) KNORR-BREMSE ČR. *Výroční zpráva 2006.* [online]. 2012 [cit. 2012-1-6]. Dostupné z <http://www.justice.cz/xqw/xervlet/insl/getFile?listina.@slCis=500119840&listina.@rozliseni=pdf&listina.@klic=352e679ebfc7658511ea63c8976493e2>

(5) *KNORR-BREMSE ČR. Výroční zpráva 2008.* [online]. 2012 [cit. 2012-1-6]. Dostupné z <http://www.justice.cz/xqw/xervlet/insl/getFile?listina.@slCis=500203921&listina.@rozliseni=pdf&listina.@klic=5c1cc18a2fa8a33c7716a973d2d29f56>

(6) KNORR-BREMSE ČR. *Výroční zpráva 2010.* [online]. 2012 [cit. 2012-1-6]. Dostupné z <http://www.justice.cz/xqw/xervlet/insl/getFile?listina.@slCis=500280237&listina.@rozliseni=pdf&listina.@klic=ee0995e455da8f220cb4ba554a366670>

(7) KNORR-BREMSE GROUP. *Knorr-Bremse and KAMAZ open joint production plant in Russia – press release*. [online]. 2008-10-28 [cit. 2011-5-22]. Dostupné z <http://www.knorr-bremse.cz/cz/press/pressreleases/press_detail_1344.jsp>

(8) KNORR-BREMSE GROUP. *Knorr-Bremse Annual Report 2005*. [online]. 2012 [cit. 2012-1-3]. Dostupné z <http://www.knorr-bremse.de/media/documents/group/ff_annual_reports/gb05_en.pdf>

(9) KNORR-BREMSE GROUP. *Knorr-Bremse Annual Report 2008*. [online].

2012 [cit. 2012-1-3]. Dostupné z <http://www.knorr-bremse.de/media/documents/group/ff_annual_reports/GB_2008_EN_small-safe.pdf>

(10) KNORR-BREMSE GROUP. *Knorr-Bremse Annual Report 2010*. [online]. 2012 [cit. 2012-1-3]. Dostupné z <http://www.knorr-bremse.de/media/documents/group/ff_annual_reports/anualreport2010/Anual_Report_2010_Knorr-Bremse_AG.pdf>

(11) KNORR-BREMSE GROUP. *Knorr-Bremse tops EUR 4 billion in sales (Press release)*. [online]. 2012 [cit. 2012-1-15]. Dostupné z <http://www.knorr-bremse.de/en/press/pressreleases/press_detail_17728.jsp>

(12) KNORR-BREMSE GROUP. *Knorr-Bremse v České republice*. [online]. 2011 [cit. 2011-5-22]. Dostupné z <http://www.knorr-bremse.cz/cz/group/kbinczechrepublic/knorrbremse_cz.jsp>

(13) KOVÁR, M. *Diskrétní matematika*. Brno. 2002-2003. S. 87-113.

(14) MSP ČR. *Úplný výpis z rejstříku (KNORR-BREMSE Systémy pro užitková vozidla ČR s.r.o.)*. [online]. 2011 [cit. 14. 12. 2011]. Dostupné z: <http://www.justice.cz/xqw/xervlet/insl/report?sysinf.vypis.CEK=106520&sysinf.vypis.rozsah=uplny&sysinf.@typ=transformace&sysinf.@strana=report&sysinf.vypis.typ=XHTML&sysinf.vypis.klic=6d350e097309c54e33ffbb28b7c2c1f8&sysinf.spis.@oddil=C&sysinf.spis.@vloz ka=3720&sysinf.spis.@soud=Krajsk%FDm%20soudem%20v%20%DAst%ED%20nad%20Labem&sysinf.platnost=14.12.2011>

(15) MSP ČR. *Úplný výpis z rejstříku (IFE-CR a.s)*. [online]. 2011 [cit. 14. 12. 2011]. Dostupné z: <http://www.justice.cz/xqw/xervlet/insl/report?sysinf.vypis.CEK=177028&sysinf.vypis.rozsah=aktualni&sysinf.@typ=transformace&sysinf.@strana=report&sysinf.vypis.typ=XHTML&sysinf.vypis.klic=28dc75c3a32ed8569ad6b2b5ae7b09f0&sysinf.spis.@oddil=B&sysinf.spis.@vlo zka=414&sysinf.spis.@soud=Krajsk%FDm%20soudem%20v%20Brn%EC&sysinf.platnost=14.12.2011>

(16) MSP ČR. *Úplný výpis z rejstříku (IFE-CZ s.r.o)*. [online]. 2011 [cit. 14. 12. 2011]. Dostupné z: <http://www.justice.cz/xqw/xervlet/insl/report?sysinf.vypis.CEK=293226&sysinf.vypis.rozsah=aktualni&sysinf.@typ=transformace&sysinf.@strana=report&sysinf.vypis.typ=XHTML&sysinf.vypis.klic=f9c594f33bfbc0c395763706f988c78c&sysinf.spis.@oddil=C&sysinf.spis.@vlozka=28862&sysinf.spis.@soud=Krajsk%FDm%20soudem%20v%20Brn%EC&sysinf.platnost=14.12.2011>

(17) SCHWARZ, J., SEKANINA, L. *Aplikované evoluční algoritmy*. Brno: VUT – FIT, 2006. Studijní opora, 101 s.

(18) THE MATHWORKS. *MATLAB – Documentation*. Dostupné z <http://www.mathworks.com/help/techdoc/>

(19) THE MATHWORKS. *MATLAB – Global Optimization Toolbox – User's Guide*. The MathWorks Inc.. 2010.

(20) THE MATHWORKS. *MATLAB – User's Guide*. The MathWorks Inc.. 2010.

(21) VENESS, C. *Movable Type Scripts*. [online]. 2012 [cit. 14.2.2011]. Dostupné z: <http://www.movable-type.co.uk/scripts/latlong.html>

(22) *Wikipedia: MATLAB*. [online]. 2011 [cit. 2011-5-22]. Dostupné z <http://en.wikipedia.org/wiki/MATLAB>. Poslední aktualizace 2011-5-11.

(23) *Wikipedie: Lamarckismus*. [online]. [cit. 2011-5-22]. Dostupné z <http://cs.wikipedia.org/wiki/Lamarckismus>. Poslední aktualizace 2011-3-26.

(24) ZBOŘIL, F., ZBOŘIL, F., Jr. *Základy umělé inteligence*. Brno: VUT – FIT, 2006. Studijní opora. 142 s.

9 Seznam obrázků

10 Seznam tabulek

11 Přílohy

compute.m

```
function [calculatedNodes, fval, pop] = compute(project)

% assigning variables due to performance
nodes = project.nodes.data;
edges = project.edges.data;
geonodes = project.geonodes.data;
geoedges = project.geoedges.data;
grid = project.gridMap;
gridRes = size(project.gridMap);
computeMethod = project.properties.get('distComputeMethod');
numOfGen = str2double(project.properties.getProperty('GANumOfGen'));
genMass = str2double(project.properties.getProperty('GAGenMass'));

if isempty(grid)
    msgbox('gridMap can not be empty.', 'modal');
    fval = [];
    pop = [];
    return;
end

% convert SourceNode and DestinationNode to row index in node array
% to speed up fitness function
for i = 1:size(edges,1)
    edges{i,1} = find(edges{i,1} == [nodes{:,1}]);
    edges{i,2} = find(edges{i,2} == [nodes{:,1}]);
end

% convert coordinates to radians
for i=1:size(nodes,1)
    nodes{i,3} = toRadians('degrees', nodes{i,3});
    nodes{i,4} = toRadians('degrees', nodes{i,4});
end

for i=1:size(geonodes,1)
    geonodes{i,2} = toRadians('degrees', geonodes{i,2});
    geonodes{i,3} = toRadians('degrees', geonodes{i,3});
end


% determine indexes of unallocated nodes
unalNodesIdx = find(isnan([nodes{:,3}]) == 1);

if isempty(unalNodesIdx)
    pop = [];
    switch computeMethod
        case 'planar'
            fval = fitnessPlanar([],nodes,edges,[],[],[]);
        case 'haversine'
            fval = fitnessHaversine([],nodes,edges,[],[],[]);
        case 'dijkstra'
            fval = fitnessDijkstra([], getShortestPaths(), geonodes, nodes,...
                edges, [], [], []);
    end
else
    % convert range to radians
    lat = str2num(project.properties.getProperty('latitudeRange'));
    lon = str2num(project.properties.getProperty('longitudeRange'));
    latR=toRadians('degrees',lat);
    lonR=toRadians('degrees',lon);

    % create array of values for faster integer(index) to angle conversion
```

```
    rangeArrR.lat = linspace(latR(1),latR(2),gridRes(1));
    rangeArrR.long = linspace(lonR(1),lonR(2),gridRes(2));

    % create population range
    popRange = repmat([1 1;gridRes],1,length(unalNodesIdx));

    options = gaoptimset('PopInitRange', popRange,...
                  'PlotFcns',@gaplotbestf,...
                  'Generations', numOfGen,...
                  'EliteCount', 20,...
                  'PopulationSize', genMass,...
                  ...%'TolFun', 1e-9,...
                  'StallGenLimit', 1000,...
                  'CreationFcn',@createPopulationI,...
                  'CrossoverFcn',@crossoversinglepoint,...
                  'MutationFcn',@mutatePopulationIDecRange);

    switch computeMethod
        case 'planar'
            fitnessFcn = @(coords)fitnessPlanar(coords, nodes, edges,...
                unalNodesIdx, rangeArrR, grid);
        case 'haversine'
            fitnessFcn = @(coords)fitnessHaversine(coords, nodes, edges,...
                unalNodesIdx, rangeArrR, grid);
        case 'dijkstra'
            sPaths = getShortestPaths();
            fitnessFcn = @(coords)fitnessDijkstra(coords, sPaths, geonodes,...
                nodes, edges, unalNodesIdx, rangeArrR, grid);
    end

    [coords, fval, ~, ~, pop, ~]=ga(fitnessFcn,length(unalNodesIdx)*2,options);

    for i=1:size(pop,1)
        pop(i,1:2:end) = toDegrees('radians',rangeArrR.lat(pop(i,1:2:end)));
        pop(i,2:2:end) = toDegrees('radians',rangeArrR.long(pop(i,2:2:end)));
    end

    % fill nodes with calculated coordinates
    for i=1:length(unalNodesIdx)
        project.nodes.data{unalNodesIdx(i),3} = toDegrees('radians',...
            rangeArrR.lat(coords(1)));
        project.nodes.data{unalNodesIdx(i),4} = toDegrees('radians',...
            rangeArrR.long(coords(2)));
        coords = coords(1,3:end); % remove used coordinates
    end
end

calculatedNodes = project.nodes.data;


    function sPaths = getShortestPaths()
        nodeIds = [geonodes{:,1}]';
        adjMat = Inf(length(nodeIds));
        adjMat(1:length(nodeIds)+1:length(nodeIds)*length(nodeIds)) = 0;

        % build adjacency matrix
        for ii=1:(size(geoedges,1))
            startIdx = geoedges{ii,1} == nodeIds(:);
            endIdx = geoedges{ii,2} == nodeIds(:);
            adjMat(startIdx, endIdx) = geoedges{ii,3};
            adjMat(endIdx, startIdx) = geoedges{ii,3};

        end

        sPaths = struct('distances',{}, 'predecessors', {});
        %compute shortest path for every node in geo network (faster fitnessFcn)
        for ii=1:(size(geonodes,1))
            sPaths(ii) = dijkstraAlg(adjMat, ii);
```

```
        end
    end

end

function population = createPopulationI(GenomeLen,~,opt)
    popLat = randi(opt.PopInitRange(2,1), opt.PopulationSize, GenomeLen/2);
    popLon = randi(opt.PopInitRange(2,2), opt.PopulationSize, GenomeLen/2);
    population = zeros(opt.PopulationSize,GenomeLen);
    population(:,1:2:end) = popLat;
    population(:,2:2:end) = popLon;
end

% mutate parents to integer children
function mChildren = mutatePopulationIDecRange(parents,opt,~, ...
    ~,state,~,thisPopulation)

    mutationRate = 0.2; % gene mutation probability

    mChildren = thisPopulation(parents(:),:);
    scale = 1-state.Generation/opt.Generations;
    upperLat = opt.PopInitRange(2,1);
    upperLon = opt.PopInitRange(2,2);

    for i=1:size(mChildren,1)
        if rand < mutationRate
            for j=1:size(mChildren,2)
                if (mod(j,2) == 0)
                    upperDiff = ceil((upperLon - mChildren(i,j))*scale);
                else
                    upperDiff = ceil((upperLat - mChildren(i,j))*scale);
                end

                lowerDiff = ceil((mChildren(i,j)-1)*scale);
                r = [mChildren(i,j)-lowerDiff,...
                    mChildren(i,j)+upperDiff];
                mChildren(i,j) = randi(r);
            end
        end
    end
end
```

dijkstraAlg.m

```
function [solvedGraph] = dijkstraAlg(adjMat, startNode)
    solvedGraph = [];

    % each node has one predecessor (needed for path reconstruction)
    graph.predecessors = nan(size(adjMat(:,1)));
    % each node has infinite distance from starting node
    graph.distances = Inf(size(adjMat(:,1)));
    % each node is marked as temporary
    graph.temporary = true(size(adjMat(:,1)));
    % starting point has distance set as zero
    graph.distances(startNode) = 0;


    auxDistances = graph.distances;
    while any(graph.temporary)
        %search for temporary node with smallest distance from starting node
        [~, node] = min(auxDistances);

        % disqualify from smallest distance node search
        auxDistances(node) = NaN;

        % mark node as permanent
        graph.temporary(node) = false;
```

```
        % search for distances to node neighbors
        distToNeighbors = adjMat(node, :);
        for i=1:length(distToNeighbors)
            % we are interested only about temporary neighbors
            if graph.temporary(i)
                % eventually actualize neighbor value and change
                % pointer to predecessor
                if (graph.distances(i) > graph.distances(node) +...
                        distToNeighbors(i))

                    [graph.distances(i), auxDistances(i)] =...
                        deal(graph.distances(node) + distToNeighbors(i));
                    graph.predecessors(i) = node;
                end
            end
        end
    end
    solvedGraph.distances = graph.distances;
    solvedGraph.predecessors = graph.predecessors;
end
```

findNearestNeighbor.m

```
function [nodeIndex, distance] = findNearestNeighbor(geoC, lat1, lon1, degRad)

    if strcmp(degRad, 'degrees')
        lat1 = toRadians('degrees', lat1);
        lon1 = toRadians('degrees', lon1);
        geoC = [[geoC{:, 2}] ; [geoC{:, 3}]]';
        geoC(:) = toRadians('degrees', geoC(:));
    end

    a=sin((geoC(:,1)-lat1)/2).^2 + cos(lat1) .* cos(geoC(:,1)) .* sin((geoC(:,2)-
lon1)/2).^2;
    distances = 6371000 * 2 * atan2(sqrt(a(:)),sqrt(1 - a(:)));
    [distance, nodeIndex] = min(distances);
end
```

fitnessDijkstra.m

```
function [distance] = fitnessDijkstra(newCoords, shortestPaths, geonodes, nodes,...
    edges, unalNodesIdx, rangeArrR, datagrid)

    if ~isempty(newCoords)
        xv = newCoords(1:2:length(newCoords));
        yv = newCoords(2:2:length(newCoords));

        xvLatR = rangeArrR.lat(xv);
        yvLonR = rangeArrR.long(yv);


        % fill nodes with generated coordinates and
        % check water/land position by mapgrid
        for i=1:length(unalNodesIdx)
            if ~datagrid(xv(i),yv(i))
                distance = inf;
                return;
            end
            node = unalNodesIdx(i);
            nodes{node,3} = xvLatR(i);
            nodes{node,4} = yvLonR(i);
        end
    end
    % calculate network distance
    geoCoords = horzcat([geonodes{1:end,2}]', [geonodes{1:end,3}]');
    calcDist = arrayfun(@dijkstraDistance,...
                         [nodes{[edges{:,1}]',3}]',...
```

```
                        [nodes{[edges{:,1}]',4}]',...
                        [nodes{[edges{:,2}]',3}]',...
                        [nodes{[edges{:,2}]',4}]',...
                        [edges{:,3}]');

    distance = sum(calcDist);

    function [dist] = dijkstraDistance(lat1,lon1,lat2,lon2, k)
        [neighbor1, distance1] = findNearestNeighbor(geoCoords, lat1, lon1, []);
        [neighbor2, distance2] = findNearestNeighbor(geoCoords, lat2, lon2, []);
        dd = shortestPaths(neighbor2).distances(neighbor1)*k;
        dist = dd + distance1 + distance2;
    end
end
```

fitnessHaversine.m

```
function [distance] = fitnessHaversine(newCoords, nodes, edges, unalNodesIdx,...
    rangeArrR, datagrid)

    if ~isempty(newCoords)
        xv = newCoords(1:2:length(newCoords));
        yv = newCoords(2:2:length(newCoords));

        xvLatR = rangeArrR.lat(xv);
        yvLonR = rangeArrR.long(yv);

        % fill nodes with generated coordinates and
        % check water/land position by mapgrid
        for i=1:length(unalNodesIdx)
            if ~datagrid(xv(i),yv(i))
                distance = inf;
                return;
            end
            node = unalNodesIdx(i);
            nodes{node,3} = xvLatR(i);
            nodes{node,4} = yvLonR(i);
        end
    end

    % calculate network distance
    calcDist = 0;
    for i=1:size(edges,1)
        calcDist = calcDist + haversineDistance(...
                            nodes{edges{i,1},3},...
                            nodes{edges{i,1},4},...
                            nodes{edges{i,2},3},...
                            nodes{edges{i,2},4})...
                            * edges{i,3};
    end
    distance = calcDist;
end

function [dist] = haversineDistance(lat1,lon1,lat2,lon2)
    % earth radius is 6371km
    a = sin((lat2-lat1)/2).^2 + cos(lat1) .* cos(lat2) .* sin((lon2-lon1)/2).^2;
    dist = 6371000 * 2 * atan2(sqrt(a),sqrt(1 - a));
end
```

fitnessPlanar.m

```
function [distance] = fitnessPlanar(newCoords, nodes, edges, unalNodesIdx,...
    rangeArrR, datagrid)

    if ~isempty(newCoords)
        xv = newCoords(1:2:length(newCoords));
        yv = newCoords(2:2:length(newCoords));
```

```
        xvLatR = rangeArrR.lat(xv);
        yvLonR = rangeArrR.long(yv);

        % fill nodes with generated coordinates and
        % check water/land position by mapgrid
        for i=1:length(unalNodesIdx)
            if ~datagrid(xv(i),yv(i))
                distance = inf;
                return;
            end
            node = unalNodesIdx(i);
            nodes{node,3} = xvLatR(i);
            nodes{node,4} = yvLonR(i);
        end
    end

    % calculate network distance
    calcDist = 0;
    for i=1:size(edges,1)
        calcDist = calcDist + planarDistance(...
                              nodes{edges{i,1},3},...
                              nodes{edges{i,1},4},...
                              nodes{edges{i,2},3},...
                              nodes{edges{i,2},4})...
                              * edges{i,3};
    end
    distance = calcDist;
end

function [dist] = planarDistance(lat1, lon1, lat2, lon2)
    dist = sqrt((lat1 - lat2)^2 + (lon1 - lon2)^2);
end
```

gui.m

```
function gui(varargin)
    clc;
    if isempty(varargin)
        initialize();
    else

    end
end

% initialize on start
function initialize()
    createMainFigure();
    %create default project
    newProject_Callback();
end

%%%%%%%%%%%%%
% Callbacks %
%%%%%%%%%%%%%

% show actual state of task on map
function showMap_Callback(~, ~, ~)

project = guidata(findobj('Tag', 'MainFigure'));

try
    nodes = project.nodes.data;
    edges = project.edges.data;
catch ex
    nodes = [];
    edges = [];
end
```

```
try
    geonodes = project.geonodes.data;
    geoedges = project.geoedges.data;
    selectedNodeHandle = [];
catch ex
    geonodes = [];
    geoedges = [];
end

% Create new figure with axes
hAxesFigure = findobj('Tag', 'AxesFigure');
hAxesObject = findobj('Tag', 'AxesObject');
if isempty(hAxesFigure)
    figure(getpref('DefaultFigure'),...
        'Name', 'Placement schema',...
        'Tag', 'AxesFigure',...
        'Menubar', 'none',...
        'KeyPressFcn', @closeFigure_Callback);
    hAxesObject = axes('Tag', 'AxesObject',...
        'XLim',str2num(project.properties.getProperty('longitudeRange')),...
        'YLim',str2num(project.properties.getProperty('latitudeRange')),...
        'GridLineStyle',':',...
        'XGrid','on',...
        'YGrid','on',...
        'XColor', 'k',...
        'YColor', 'k',...
        'XMinorTick', 'on',...
        'YMinorTick', 'on',...
        'Layer', 'bottom',...
        'Color', 'w');

    hNodeInfo = uicontrol(getpref('DefaultText'),...
        'HorizontalAlignment', 'center',...
        'Tag', 'SelectedNodeInfo',...
        'UserData', 'Nodes file: ',...
        'Position', [0 0.94 0.24 0.04]);

    hGridCheckbox = uicontrol(getpref('DefaultCheckbox'),...
        'Tag', 'ShowGridCheckbox',...
        'Position', [0.48 0.93 0.13 0.06],...
        'String', 'Show grid',...
        'FontSize', 0.4,...
        'Callback', @showGrid_Callback);

    uicontrol(getpref('DefaultCheckbox'),...
        'Tag', 'ShowPopCheckbox',...
        'Position', [0.91 0.86 0.09 0.06],...
        'String', 'Show Pop',...
        'FontSize', 0.4,...
        'Visible','off',...
        'Callback', @showPop_Callback);


    uicontrol(getpref('DefaultCheckbox'),...
        'Tag', 'ShowGeoNetCheckbox',...
        'Position', [0.25 0.93 0.23 0.06],...
        'String', 'Show Geo Network',...
        'FontSize', 0.4,...
        'Callback', @showGeoNet_Callback);

    uicontrol(getpref('DefaultButton'),...
        'Position',[0.81 0.93 0.18 0.06],...
        'String','Apply Geo Net',...
        'Callback',@applyGeoMap_Callback,...
        'Tag','ApplyGeoMapBtn',...
        'TooltipString', 'Apply values');
```

```
    uicontrol(getpref('DefaultButton'),...
        'Position',[0.62 0.93 0.18 0.06],...
        'String','Obtain Distances',...
        'Callback',@obtainDistances_Callback,...
        'Tag','ApplyGeoMapBtn',...
        'TooltipString', 'Obtain distances from Google Distance Matrix API');
end
axes(hAxesObject);
cla;
xlim = str2num(project.properties.getProperty('longitudeRange'));
ylim = str2num(project.properties.getProperty('latitudeRange'));
set(gca, 'XLim',xlim,...
         'YLim',ylim);
hold on;
drawGrid();
drawGoogleLogo();
drawCoastBorders();
drawNodes();
drawEdges();
drawFinalPopulation();
hold off;

    function drawCoastBorders()
        geoshow(project.worldBorders.POline(1), 'Color', [0.6 0.8 0.8]);
        geoshow(project.worldBorders.POline(2).lat,...
                project.worldBorders.POline(2).long,...
                'Color', [0.5 0.5 0.9]);
    end

    function drawGoogleLogo()
        logo = imread('Google_maps_logo.png');

        hGMapsLogo = imagesc([xlim(1) (xlim(1)+(diff(xlim)*0.1))],...
                     [(ylim(2)-(diff(ylim)*0.05)) ylim(2)], flipdim(logo ,1));
        set(hGMapsLogo, 'Tag', 'GMapsLogo',...
                    'Visible', 'off');
    end

    function drawGrid()
        if isempty(project.gridMap)
           return;
        end

        if ~diff(project.gridMap)
            set(hGridCheckbox, 'Enable', 'off');
            return;
        end

        x = str2num(project.properties.getProperty('gridLatitude'));
        y = str2num(project.properties.getProperty('gridLongitude'));
        hGrid = imagesc(y, x, project.gridMap);
        colormap(gray);
        set(hGrid, 'Tag', 'GridImage',...
            'Visible', 'off');
    end

    function drawNodes()
        for i=1:size(nodes,1)
            pt_Y = nodes{i,3};
            pt_X = nodes{i,4};
            if (~isnan(pt_X) && ~isnan(pt_Y))
                switch nodes{i,2}
                    case {'suppl'}
                        scatter(pt_X, pt_Y, 'mo');
                    case {'plant'}
                        scatter(pt_X, pt_Y, 'r+');
                    case {'purch'}
                        scatter(pt_X, pt_Y, 'b*');
```

```
                end
                text(pt_X+(0.015*diff(str2num(project.properties.getProperty(...
                    'longitudeRange')))), pt_Y, nodes{i,5},...
                     'Color', 'k');
            end
        end
    end

    function drawEdges()
        resourceSX = [];
        resourceSY = [];
        resourceDX = [];
        resourceDY = [];
        semiproductSX = [];
        semiproductSY = [];
        semiproductDX = [];
        semiproductDY = [];
        productSX = [];
        productSY = [];
        productDX = [];
        productDY = [];

        for i=1:size(edges,1)
            sNInd = find([nodes{:,1}] == edges{i,1});
            eNInd = find([nodes{:,1}] == edges{i,2});

            if isempty(sNInd) || isempty(eNInd)
                continue;
            end
            SY = nodes{sNInd,3};
            SX = nodes{sNInd,4};
            EY = nodes{eNInd,3};
            EX = nodes{eNInd,4};
            if ~isnan([SX SY EX EY])
                if strcmp(char(nodes{sNInd,2}), 'suppl')
                    resourceSX = [resourceSX SX];
                    resourceSY = [resourceSY SY];

                    resourceDX = [resourceDX EX-SX];
                    resourceDY = [resourceDY EY-SY];
                elseif strcmp(char(nodes{eNInd,2}), 'purch')
                    productSX = [productSX SX];
                    productSY = [productSY SY];

                    productDX = [productDX EX-SX];
                    productDY = [productDY EY-SY];
                else

                    semiproductSX = [semiproductSX SX];
                    semiproductSY = [semiproductSY SY];

                    semiproductDX = [semiproductDX EX-SX];
                    semiproductDY = [semiproductDY EY-SY];
                end
            end
        end
        quiver(resourceSX, resourceSY, resourceDX, resourceDY, 0,...
            'Color', [0.8 0 0]);
        quiver(semiproductSX, semiproductSY, semiproductDX, semiproductDY, 0,...
            'Color', [0 0.8 0]);
        quiver(productSX, productSY, productDX, productDY, 0,...
            'Color', [0.8 0.8 0]);
    end

    function drawGeoNetwork()
        if isempty(geoedges)
           return;
        end
```

```
        lineX = [];
        lineY = [];
        for i=1:size(geoedges,1)
            try
                sNInd = find([geonodes{:,1}] == geoedges{i,1});
                eNInd = find([geonodes{:,1}] == geoedges{i,2});
            catch ex
                writeError(['Unknown node in geo edges on row: ' num2str(i)]);
            end
            if ~isempty(sNInd) && ~isempty(eNInd)
                lineY = [lineY [geonodes{sNInd,2}; geonodes{eNInd,2}]];
                lineX = [lineX [geonodes{sNInd,3}; geonodes{eNInd,3}]];
            end
        end

        nLines = plot(lineX, lineY,...
            'Color', [0.7 0.7 0.7],...
            'LineStyle', '-',...
            'Tag', 'GeoEdges',...
            'ButtonDownFcn', @selectLine_Callback);

        for i=1:size(geoedges,1)
            a = geoedges(i,1:2);
            set(nLines(i), 'UserData', a);
        end

        if isempty(geonodes)
           return;
        end

        for i=1:size(geonodes,1);
            scatter(geonodes{i,3},geonodes{i,2}, 'bo',...
                'Tag', 'GeoNodes',...
                'UserData', geonodes(i,:),...
                'ButtonDownFcn', @selectPoint_Callback);
        end
    end

    function drawFinalPopulation()
        try
            popX = project.finalpopulation(:,1:2:end);
            popY = project.finalpopulation(:,2:2:end);
        catch e
            return;
        end
        for i=1:size(popX,2)
            scatter(popY(:,i), popX(:,i), 'Tag', 'FinalPop', 'Visible', 'off');
            drawnow;
        end


    end

    function showGrid_Callback(hObject, ~, ~)
        hGridImage = findobj('Tag', 'GridImage');
        if (get(hObject,'Value') == get(hObject,'Max'))
            set(hGridImage, 'Visible', 'on');
        else
            set(hGridImage, 'Visible', 'off');
        end
    end

    function showPop_Callback(hObject, ~, ~)
        hPop = findobj('Tag', 'FinalPop');
        if (get(hObject,'Value') == get(hObject,'Max'))
            set(hPop, 'Visible', 'on');
        else
            set(hPop, 'Visible', 'off');
```

```
        end
    end

    function showGeoNet_Callback(hObject, ~, ~)
        hGeoNetwork = findobj('Tag', 'GeoNodes', '-or', 'Tag', 'GeoEdges');
        if (get(hObject,'Value') == get(hObject,'Max'))
            if isempty(hGeoNetwork)
                hold on;
                drawGeoNetwork();
                hold off;
            else
                set(hGeoNetwork, 'Visible', 'on');
            end
            set(findobj('Tag', 'GMapsLogo'), 'Visible', 'on');
        else
            set(hGeoNetwork, 'Visible', 'off');
            set(findobj('Tag', 'GMapsLogo'), 'Visible', 'off');
        end
    end

    function selectPoint_Callback(hObject, ~, ~)
        if strcmp(get(hObject, 'Selected'), 'on')
            set(hObject, 'Selected', 'off');
            selectedNodeHandle = [];
            set(hNodeInfo, 'String', '');
        elseif isempty(selectedNodeHandle)
            set(hObject, 'Selected', 'on');
            selectedNodeHandle = hObject;
            actualNode = get(hObject, 'UserData');
            set(hNodeInfo, 'String', actualNode{1,4});
        else
            set(hNodeInfo, 'String', '');
            set(selectedNodeHandle, 'Selected', 'off');
            oldNode = get(selectedNodeHandle, 'UserData');
            selectedNodeHandle = [];
            actualNode = get(hObject, 'UserData');

            if (find([geoedges{:,1}] == actualNode{1} &...
                     [geoedges{:,2}] == oldNode{1}))
                return;
            end

            if (find([geoedges{:,2}] == actualNode{1} &...
                     [geoedges{:,1}] == oldNode{1}))
                return;
            end

            lat = [oldNode{2} actualNode{2}];
            long = [oldNode{3} actualNode{3}];

            hold on;
            hLine = plot(long,...
                lat,...
                'Color', [0.7 0.7 0.7],...
                'LineStyle', '-',...
                'Tag', 'GeoEdges',...
                'ButtonDownFcn', @selectLine_Callback);
            uistack(hLine, 'bottom');
            hold off;
            edgeElement = {oldNode{1} actualNode{1} NaN};
            set(hLine, 'UserData', {oldNode{1} actualNode{1}});
            geoedges = vertcat(geoedges, edgeElement);

        end
    end

    function selectLine_Callback(hObject, ~, ~)
        line = get(hObject, 'UserData');
```

```
        delete(hObject);
        for i=1:size(geoedges,1)
            a = [geoedges{i,1:2}];
            b = [line{:}];
            if isequal(a,b)
                geoedges(i,:) = [];
                return;
            end
        end
    end

    function applyGeoMap_Callback(~, ~)
        project.geonodes.data = geonodes;
        project.geoedges.data = geoedges;
        guidata(findobj('Tag','MainFigure'), project);
        close(gcbf);
    end

    function obtainDistances_Callback(~, ~)
        startCoords = cell(size(geoedges,1),1);
        endCoords = cell(size(geoedges,1),2);
        requests = createRequests();
        tim = timer('Tag', 'DistanceTimer',...
            'TimerFcn', @timer_Callback,...
            'ExecutionMode', 'fixedSpacing',...
            'Period', 1.5);
        start(tim);

        function timer_Callback(~, ~, ~)
            disp([num2str(size(requests,1)) ' requests left']);
            if ~isempty(requests)
                req = requests{1};
                requests(1) = [];

                answer = urlread(req);
                distanceArray = parseAnswer(answer);

                endIndexes = endCoords{1,2};
                endCoords(1,:) = [];

                for i=1:size(endIndexes,2)
                    geoedges{endIndexes(i),3} = distanceArray(i);
                    edge = geoedges(endIndexes(i),1:2);
                    hLine = findobj('UserData', edge);
                    if length(hLine)>1
                        disp(['duplicate edge: ' num2str([edge{:}])]);
                    else
                        if isnan(distanceArray(i))
                            set(hLine, 'Color', 'r');
                        else
                            set(hLine, 'Color', 'g');
                        end
                    end
                end
            else
                stop(tim);
                delete(timerfind);
            end

            function [distArr] = parseAnswer(answer)
                distArr = [];

                import org.xml.sax.InputSource
                import javax.xml.parsers.*
                import java.io.*
                iS = InputSource();
                iS.setCharacterStream(StringReader(answer));
                doc = xmlread(iS);
```

```
            distanceMatrixNode = doc.getDocumentElement;
            entries = distanceMatrixNode.getChildNodes;
            for ii=0:entries.getLength - 1
            if strcmpi(entries.item(ii).getNodeName, 'row')
                rowNode = entries.item(ii).getChildNodes;
                for j=0:rowNode.getLength - 1
                    elementNode = rowNode.item(j).getChildNodes;
                    for k=0:elementNode.getLength - 1
                    if strcmpi(elementNode.item(k).getNodeName, 'status')
                        statusNode = elementNode.item(k).getChildNodes;
                        if ~strcmpi(statusNode.getTextContent, 'OK')
                            distArr = [distArr NaN];
                        end
                    end
                    if strcmpi(elementNode.item(k).getNodeName, 'distance')
                    distanceNode = elementNode.item(k).getChildNodes;
                    for l=0:distanceNode.getLength - 1
                    if strcmpi(distanceNode.item(l).getNodeName, 'value')
                    distArr = [distArr ...
                        str2double(distanceNode.item(l).getTextContent)];
                    end
                    end
                    end
                    end
                end
            end
            end
        end
    end

    function [httpRequests] = createRequests()

        for i=1:size(geoedges,1)
            if ~isnan(geoedges{i,3})
                continue;
            end
            if isempty(startCoords{geoedges{i,1}})
                startCoords{geoedges{i,1}} =...
                    [num2str(geonodes{geoedges{i,1},2})...
                    ','...
                    num2str(geonodes{geoedges{i,1},3})];
                %startCoords{geoedges{ii,1},2} = i;
            end
            if isempty(endCoords{geoedges{i,1}})
                endCoords{geoedges{i,1},1} =...
                    [num2str(geonodes{geoedges{i,2},2})...
                    ','...
                    num2str(geonodes{geoedges{i,2},3})];
                endCoords{geoedges{i,1},2} = i;
            else
                endCoords{geoedges{i,1},1} =...
                    [endCoords{geoedges{i,1},1}...
                    '|'...
                    num2str(geonodes{geoedges{i,2},2})...
                    ','...
                    num2str(geonodes{geoedges{i,2},3})];
                endCoords{geoedges{i,1},2} =...
                    [endCoords{geoedges{i,1},2} i];
            end

        end
        startCoords(cellfun(@isempty,startCoords)) = [];
        endCoords(cellfun(@isempty,endCoords(:,1)),:) = [];

        httpRequests = cell(size(startCoords,1),1);
        for iii=1:size(startCoords,1)
            httpRequests{iii} =...
```

```
          ['http://maps.googleapis.com/maps/api/distancematrix/xml?origins=',...
                   startCoords{iii},...
                   '&destinations=',...
                   endCoords{iii,1},...
                   '&sensor=false'];
            end
        end
    end
end

% quit application
function quit_Callback(~, ~, ~)
close all;
end

% load project as guidata
function load_Callback(~, ~, ~)
    mFilePath = fileparts(mfilename('fullpath'));
    [projFileName, projPath] = uigetfile(strcat(mFilePath, filesep, '*.prj'),...
            'Select Project File to Open (.prj file)');
    if (projFileName == 0)
        return;
    end

    config = java.util.Properties;
    try
        file = strcat(projPath, projFileName);
        config.load(java.io.FileInputStream(file));
    catch ex

    end
    writeInfo('======================');
    writeInfo('Loading project...');
    [~, projName, ~] = fileparts(file);
    projFiles = [projName, '_files'];
    project = createProject();
    project.properties.setProperty('projFile', projFileName);
    en = config.keys();
    while (en.hasMoreElements())
        key = char(en.nextElement());
        if isempty(config.get(key)), continue, end
        switch key
            case 'nodesFile'
                hFile = fopen(fullfile(projPath, projFiles, config.get(key)));
                nHeaders = textscan(hFile,'%s %s %s %s %s',1,'Delimiter',';',...
                    'CollectOutput', true);
                nData = textscan(hFile, '%s %s %s %s %s', 'Delimiter', ';',...
                    'HeaderLines', 1, 'CollectOutput', true);
                fclose(hFile);
                project.nodes.headers = nHeaders{:};
                project.nodes.data = str2DoubleInCell(nData{:},...
                    [1, 3, 4]);
                project.properties.setProperty(key,config.get(key));
                writeInfo(strcat('Nodes file "', config.get(key) ,'" loaded'));
            case 'edgesFile'
                hFile = fopen(fullfile(projPath, projFiles, config.get(key)));
                eHeaders = textscan(hFile, '%s %s %s %s', 1, 'Delimiter',';',...
                    'CollectOutput', true);
                eData = textscan(hFile, '%s %s %s %s', 'Delimiter', ';',...
                    'HeaderLines', 1, 'CollectOutput', true);
                fclose(hFile);
                project.edges.headers = eHeaders{:};
                project.edges.data = str2DoubleInCell(eData{:},...
                    [1, 2, 3]);
                project.properties.setProperty(key,config.get(key));
                writeInfo(strcat('Edges file "', config.get(key) ,'" loaded'));
            case 'geoNetworkNodesFile'
                hFile = fopen(fullfile(projPath, projFiles, config.get(key)));
```

```
                nHeaders = textscan(hFile, '%s %s %s %s', 1,'Delimiter',';',...
                    'CollectOutput', true);
                nData = textscan(hFile, '%s %s %s %s', 'Delimiter', ';',...
                    'HeaderLines', 1, 'CollectOutput', true);
                fclose(hFile);
                project.geonodes.headers = nHeaders{:};
                project.geonodes.data =str2DoubleInCell(nData{:},...
                    [1, 2, 3]);
                project.properties.setProperty(key,config.get(key));
                writeInfo(strcat('Geographic network nodes file "',...
                    config.get(key) ,'" loaded'));
            case 'geoNetworkEdgesFile'
                hFile = fopen(fullfile(projPath, projFiles, config.get(key)));
                eHeaders = textscan(hFile, '%s %s %s', 1, 'Delimiter', ';',...
                    'CollectOutput', true);
                eData = textscan(hFile, '%s %s %s', 'Delimiter', ';',...
                    'HeaderLines', 1, 'CollectOutput', true);
                fclose(hFile);
                project.geoedges.headers = eHeaders{:};
                project.geoedges.data =str2DoubleInCell(eData{:},...
                    [1, 2, 3]);
                project.properties.setProperty(key,config.get(key));
                writeInfo(strcat('Geographic network edges file "',...
                    config.get(key) ,'" loaded'));
            case 'latitudeRange'
                project.properties.setProperty(key, config.getProperty(key));
            case 'longitudeRange'
                project.properties.setProperty(key, config.getProperty(key));
            case 'GANumOfGen'
                project.properties.setProperty(key, config.getProperty(key));
            case 'GAGenMass'
                project.properties.setProperty(key, config.getProperty(key));
            case 'gridMapFile'
                project.properties.setProperty(key, config.get(key));
                data = load(fullfile(projPath, projFiles,...
                    config.get(key)), '-mat');
                project.gridMap = data.grid;
                project.properties.setProperty('gridLatitude',...
                    data.latitudeRange);
                project.properties.setProperty('gridLongitude',...
                    data.longitudeRange);
            case 'distComputeMethod'
                project.properties.setProperty(key, config.get(key));
                writeInfo(strcat('Distance compute method: "',...
                    config.get(key),'"'));
            otherwise
                writeError(strcat('Unknown config element: "',key,'->',...
                    config.get(key),'"'));

        end
    end
    guidata(findobj('Tag','MainFigure'), project);
    actualizeInfoFields();
    % enable buttons
    set(findobj('Tag','SaveBtn'),'Enable', 'on');
    set(findobj('Tag','EditBtn'),'Enable', 'on');
    set(findobj('Tag','ShowBtn'),'Enable', 'on');
    set(findobj('Tag','CalcBtn'),'Enable', 'on');
end

% save Node and Edge tabs
function save_Callback(~, ~, ~)
    project = guidata(findobj('Tag','MainFigure'));
    if isempty(project)
        writeError('Project is empty.');
        return;
    end
```

```
    [fileName, pathName] = uiputfile(strcat(fileparts(mfilename('fullpath')),...
        filesep, '*.prj'), 'Enter Name of Project File');

    if ~fileName
        return;
    end

    saveProjFile(project.properties, strcat(pathName, fileName));
    project.properties.setProperty('projFile', fileName);

    % save other files in subdirectory "_files"
    [~, projName, ~] = fileparts(fileName);
    filesDir = [pathName, projName, '_files'];
    if ~isdir(filesDir)
        mkdir(filesDir);
    end

    saveGraph();
    saveGeoNetwork();
    saveGridMap();
    actualizeInfoFields();

    function saveProjFile(projProperties, projFileName)
        propToSave = projProperties.clone();

        propToRemove = {'projFile',...
                        'gridLatitude',...
                        'gridLongitude'};
        for i=1:length(propToRemove)
            try
                propToSave.remove(propToRemove{i});
            catch ex
            end
        end
        propToSave.store(java.io.FileOutputStream(projFileName),...
                'Project configuration');
    end

    function saveGridMap()
        if isempty(project.gridMap), return, end;
        grid = project.gridMap;
        latitudeRange = project.properties.getProperty('gridLatitude');
        longitudeRange = project.properties.getProperty('gridLongitude');
        fileStr = strcat(filesDir, filesep, 'gridMap.mat');
        save(fileStr, 'grid', 'latitudeRange', 'longitudeRange');
        project.properties.setProperty('gridMapFile','gridMap.mat');
    end

    function saveGraph()
        nodesFileStr = strcat(filesDir, filesep, 'nodes.csv');
        edgesFileStr = strcat(filesDir, filesep, 'edges.csv');

        nHeader = project.nodes.headers;
        nData = project.nodes.data;
        eHeader = project.edges.headers;
        eData = project.edges.data;

        hNodesFile = fopen(nodesFileStr, 'w');
        hEdgesFile = fopen(edgesFileStr, 'w');

        fprintf(hNodesFile, '%s;%s;%s;%s;%s;\r\n', nHeader{:});
        fprintf(hEdgesFile, '%s;%s;%s;%s;\r\n', eHeader{1,:});

        for i = 1:size(nData,1)
            fprintf(hNodesFile, '%g;%s;%g;%g;%s;\r\n', nData{i,:});
        end
        for i = 1:size(eData,1)
            fprintf(hEdgesFile, '%g;%g;%g;%s;\r\n', eData{i,:});
```

```
        end

        fclose(hNodesFile);
        fclose(hEdgesFile);

        project.properties.setProperty('nodesFile','nodes.csv');
        project.properties.setProperty('edgesFile','edges.csv');
    end

    function saveGeoNetwork()
        nodesFileStr = strcat(filesDir, filesep, 'geonetnodes.csv');
        edgesFileStr = strcat(filesDir, filesep, 'geonetedges.csv');


        nHeader = project.geonodes.headers;
        nData = project.geonodes.data;
        eHeader = project.geoedges.headers;
        eData = project.geoedges.data;

        hNodesFile = fopen(nodesFileStr, 'w');
        hEdgesFile = fopen(edgesFileStr, 'w');

        fprintf(hNodesFile, '%s;%s;%s;%s;\r\n', nHeader{:});
        fprintf(hEdgesFile, '%s;%s;%s;\r\n', eHeader{1,:});

        for i = 1:size(nData,1)
            fprintf(hNodesFile, '%g;%g;%g;%s;\r\n', nData{i,:});
        end
        for i = 1:size(eData,1)
            fprintf(hEdgesFile, '%g;%g;%g;\r\n', eData{i,:});
        end

        fclose(hNodesFile);
        fclose(hEdgesFile);

        project.properties.setProperty('geoNetworkNodesFile','geonetnodes.csv');
        project.properties.setProperty('geoNetworkEdgesFile','geonetedges.csv');
    end
end

% open edit window
function editGraph_Callback(~, ~, ~)

% Spreadsheet areas
nodeTablePosNorm = [0.01 0.58 0.75 0.4];
edgeTablePosNorm = [0.01 0.08 0.75 0.4];
% Tab control buttons
nodeRowTextField = [0.02 0.52 0.15 0.05];
addNodeBtnNorm = [0.18 0.51 0.13 0.06];
remNodeBtnNorm = [0.32 0.51 0.16 0.06];

edgeRowTextField = [0.02 0.02 0.15 0.05];
addEdgeBtnNorm = [0.18 0.01 0.13 0.06];
remEdgeBtnNorm = [0.32 0.01 0.16 0.06];

% Buttons
applyBtnNorm = [0.78 0.9 0.2 0.07];
closeBtnNorm = [0.78 0.82 0.2 0.07];

project = guidata(findobj('Tag','MainFigure'));

hEditWindow = figure(getpref('DefaultFigure'),...
    'Name','Edit',...
    'Tag','EditWindow',...
    'Menubar','none');


createGUIObjects();
```

```
% create GUI elements
function createGUIObjects()
    uicontrol(getpref('DefaultButton'),...
        'Position',applyBtnNorm,...
        'String','Apply',...
        'Callback',@apply_Callback,...
        'Tag','ApplyGraphBtn',...
        'TooltipString', 'Apply values');

    uicontrol(getpref('DefaultButton'),...
        'Position',closeBtnNorm,...
        'String','Close',...
        'Callback',@closeFigure_Callback,...
        'Tag','CloseBtn',...
        'TooltipString', 'Close Figure');

    uicontrol('Parent', hEditWindow,...
        'Style','Text',...
        'Tag', 'nodeRowTextField',...
        'Units', 'normalized',...
        'Position', nodeRowTextField,...
        'BackgroundColor',[0.9 0.9 0.9],...
        'FontUnits', 'normalized',...
        'FontSize', 0.5,...
        'String', 'Last row');
    uicontrol('Parent', hEditWindow,...
        getpref('DefaultButton'),...
        'Position', addNodeBtnNorm,...
        'String','Add node',...
        'Callback',@addNode_Callback);
    uicontrol('Parent', hEditWindow,...
        getpref('DefaultButton'),...
        'Position', remNodeBtnNorm,...
        'String','Remove node',...
        'Callback',@removeNode_Callback);


    uicontrol('Parent', hEditWindow,...
        'Style','Text',...
        'Tag', 'edgeRowTextField',...
        'Units', 'normalized',...
        'Position', edgeRowTextField,...
        'BackgroundColor',[0.9 0.9 0.9],...
        'FontUnits', 'normalized',...
        'FontSize', 0.5,...
        'String', 'Last row');
    uicontrol('Parent', hEditWindow,...
        getpref('DefaultButton'),...
        'Position', addEdgeBtnNorm,...
        'String','Add edge',...
        'Callback',@addEdge_Callback);
    uicontrol('Parent', hEditWindow,...
        getpref('DefaultButton'),...
        'Position', remEdgeBtnNorm,...
        'String','Remove edge',...
        'Callback',@removeEdge_Callback);

    uitable('Parent', hEditWindow,...
        'Tag','NodeTab',...
        'Units', 'normalized',...
        'Position', nodeTablePosNorm,...
        'ColumnEditable', [false true true true true],...
        'CellSelectionCallback', @tabSelection_Callback,...
        'CellEditCallback', @tabEdit_Callback,...
        'ColumnName', project.nodes.headers,...
        'ColumnWidth',{60 'auto' 'auto' 'auto' 140},...
        'Data', project.nodes.data,...
```

```
            'ColumnFormat', getNodeTabColumnFormat());
        uitable('Parent', hEditWindow,...
            'Tag','EdgeTab',...
            'Units', 'normalized',...
            'Position', edgeTablePosNorm,...
            'ColumnEditable', [true true true true],...
            'CellSelectionCallback', @tabSelection_Callback,...
            'ColumnName', project.edges.headers,...
            'ColumnWidth',{160 160 'auto' 140},...
            'Data', getLabelledEdges(),...
            'ColumnFormat', getEdgeTabColumnFormat(project.nodes.data));
    end

    % Cell selection change in Node table
    function tabSelection_Callback(hObject, eventdata, ~)
        set(hObject, 'UserData', eventdata.Indices);
        if isempty(eventdata.Indices)
            str = 'Last row';
        else
            str = ['Row:' sprintf('%i,', unique(eventdata.Indices(:,1)))];
        end

        switch get(hObject, 'Tag')
            case 'NodeTab'
                set(findobj('Tag','nodeRowTextField'),'String', str);
            case 'EdgeTab'
                set(findobj('Tag','edgeRowTextField'),'String', str);
            otherwise
        end
    end

    function tabEdit_Callback(hObject, ~, ~)
        set(findobj('Tag', 'EdgeTab'), 'ColumnFormat',...
            getEdgeTabColumnFormat(get(hObject, 'Data')));
    end

    % get Node table column format
    function colFormat = getNodeTabColumnFormat()
        cfNodeType = {'suppl' 'plant' 'purch'};
        colFormat = {'numeric',cfNodeType,'numeric','numeric','char'};
    end

    % get Edge table column format
    function colFormat = getEdgeTabColumnFormat(nodeData)

        labelledIds = getLabelledIds();
        nodeList = sort(labelledIds)';
        colFormat =  {nodeList, nodeList, 'numeric', 'char'};

        function labelledIds = getLabelledIds()
            %nodeData = project.nodes.data;
            nodeIds = cellfun(@num2str,nodeData(:,1), 'UniformOutput', false);
            nodeNames = nodeData(:,5);
            labelledIds = strcat(nodeIds, {' ('}, nodeNames, {')'});
        end
    end

    % get edge table with node names in source and destination column
    function labelledEdges = getLabelledEdges()
        nodeData = project.nodes.data;
        edgeData = project.edges.data;
        sLabNodes = {};
        eLabNodes = {};
        for i=1:size(edgeData,1)
            sNInd = [nodeData{:,1}] == edgeData{i,1};
            eNInd = [nodeData{:,1}] == edgeData{i,2};

            sNodeId = edgeData(i,1);
```

```
            sNodeName = nodeData(sNInd,5);
            eNodeId = edgeData(i,2);
            eNodeName = nodeData(eNInd,5);

            sLabelledNode=strcat({num2str(sNodeId{:})},{' ('},sNodeName,{')'});
            eLabelledNode=strcat({num2str(eNodeId{:})},{' ('},eNodeName,{')'});
            sLabNodes = [sLabNodes; sLabelledNode];
            eLabNodes = [eLabNodes; eLabelledNode];
        end

        labelledEdges = [sLabNodes, eLabNodes, edgeData(:,3:4)];
    end

    % Add node to Node table
    function addNode_Callback(~, ~, ~)
        nodes = get(findobj('Tag','NodeTab'),'Data');
        rows = get(findobj('Tag','NodeTab'), 'UserData');
        if isempty(rows)
            cell = (copyRows(nodes, size(nodes, 1), true));
        else
            cell = (copyRows(nodes, unique(rows(:,1)), true));
        end
        set(findobj('Tag','EdgeTab'),...
                    'ColumnFormat', getEdgeTabColumnFormat(cell));
        set(findobj('Tag','NodeTab'),'Data', cell(:,:));
    end

    % Remove node from Node table
    function removeNode_Callback(~, ~, ~)
        nodes = get(findobj('Tag','NodeTab'),'Data');
        rows = get(findobj('Tag','NodeTab'), 'UserData');
        if isempty(rows)
            cell = removeRows(nodes, size(nodes, 1));
        else
            cell = removeRows(nodes, unique(rows(:,1)));
        end
        set(findobj('Tag','EdgeTab'),...
                    'ColumnFormat', getEdgeTabColumnFormat(cell));
        set(findobj('Tag','NodeTab'),'Data', cell(:,:));
    end

    % Add edge to Edge table
    function addEdge_Callback(~, ~, ~)
        edges = get(findobj('Tag','EdgeTab'),'Data');
        rows = get(findobj('Tag','EdgeTab'), 'UserData');
        if isempty(rows)
            cell = (copyRows(edges, size(edges, 1), false));
        else
            cell = (copyRows(edges, unique(rows(:,1)), false));
        end
        set(findobj('Tag','EdgeTab'),'Data', cell(:,:));
    end

    % Remove edge from Edge table
    function removeEdge_Callback(~, ~, ~)
    edges = get(findobj('Tag','EdgeTab'),'Data');
    rows = get(findobj('Tag','EdgeTab'), 'UserData');
    if isempty(rows)
        cell = removeRows(edges, size(edges, 1));
    else
        cell = removeRows(edges, unique(rows(:,1)));
    end
    set(findobj('Tag','EdgeTab'),'Data', cell(:,:));
    end

    % copy specified rows at the end
    function cell = copyRows(cell, rows, primaryKey)
        if isempty(cell)
```

```
            return;
        end

        if isempty(rows)
            return;
        end

        newRows = cell(rows,:);

        if primaryKey
            freeKeys = setdiff(1:size(cell,1)+length(rows), [cell{:,1}]);
            freeKeys = freeKeys(1:length(rows));
            newRows(:,1) = num2cell(freeKeys);
        end
        cell = [cell;newRows];
    end

    % remove specified rows from cell
    function cell = removeRows(cell, rows)
        if isempty(cell) || isempty(rows)
            return;
        end

        if isempty(setdiff(1:size(cell,1),rows))
            errordlg('You can not remove all rows.','Removing error','modal');
            return;
        end

        cell(rows,:) = [];
    end

    % save graph
    function apply_Callback(~, ~, ~)
        if ~graphConsistent()
            return;
        end

        edges = get(findobj('Tag','EdgeTab'), 'Data');
        edges(:,1) = strtok(edges(:,1));
        edges(:,2) = strtok(edges(:,2));
        project.edges.data = str2DoubleInCell(edges, [1, 2]);
        project.nodes.data = get(findobj('Tag','NodeTab'), 'Data');;
        guidata(findobj('Tag','MainFigure'), project);
        close(gcbf);

        % check graph consistency.
        function result = graphConsistent()
            nodeData = get(findobj('Tag','NodeTab'),'Data');
            edgeData = get(findobj('Tag','EdgeTab'),'Data');
            if isempty(nodeData)
                errordlg('Node table should not be empty.', 'modal');
                result = false;
                return;
            end
            if isempty(edgeData)
                errordlg('Edge table should not be empty.', 'modal');
                result = false;
                return;
            end
            sNodeIds = strtok(edgeData(:,1));
            eNodeIds = strtok(edgeData(:,2));
            nodeIdsList = cellfun(@num2str,nodeData(:,1),'UniformOutput',false);

            sourceIdx = ismember(sNodeIds, nodeIdsList);
            destinationIdx = ismember(eNodeIds, nodeIdsList);

            sourceInc = find(~sourceIdx);
            destInc = find(~destinationIdx);
```

```
            messageStr = '';
            if ~isempty(sourceInc)
                messageStr = ['Incosistency in Edge table. '...
                    'Column: SourceNode Row: ' sprintf('%i,', sourceInc) '.'];
            end

            if ~isempty(destInc)
                if isempty(messageStr)
                    messageStr = [messageStr 'Incosistency in Edge table.'];
                end
                messageStr = [messageStr ' Column: DestinationNode Row: '...
                    sprintf('%i,', destInc) '.'];
            end

            if ~isempty(messageStr)
                errordlg(messageStr, 'modal');
                result = false;
            else
                result = true;
            end
        end
    end
end

% open option window
function options_Callback(~, ~, ~)


% Buttons
applyBtnNorm = [0.78 0.9 0.2 0.07];
closeBtnNorm = [0.78 0.82 0.2 0.07];

project = guidata(findobj('Tag','MainFigure'));
newProjProp = project.properties.clone();
newGrid = project.gridMap;
figure(getpref('DefaultFigure'),...
    'Name','Options',...
    'Tag','OptionsWindow',...
    'Menubar','none');

% GUI elements
createButtons();
createDistanceMethodObjects();
createCoordRangeObjects();
createGridObjects();
createGAObjects();

adjustPopups();

    function createButtons()
        uicontrol(getpref('DefaultButton'),...
            'Position',applyBtnNorm,...
            'String','Apply',...
            'Callback',@applyOptions_Callback,...
            'Tag','ApplyOptionsBtn',...
            'TooltipString', 'Apply values');

        uicontrol(getpref('DefaultButton'),...
            'Position',closeBtnNorm,...
            'String','Close',...
            'Callback',@closeFigure_Callback,...
            'Tag','CloseBtn',...
            'TooltipString', 'Close Figure');
    end

    function createDistanceMethodObjects()
        hDistanceMethod = uibuttongroup('visible','on',...
```

```
          'Units', 'normalized',...
          'FontUnits', 'normalized',...
          'FontSize', 0.12,...
          'Position',[0.01 0.68 0.32 0.21],...
          'Title', 'Distance calculation method',...
          'SelectionChangeFcn', @distanceMethod_Callback,...
          'BackgroundColor', getpref('DefaultRadiobutton', 'BackgroundColor'));

        uicontrol(getpref('DefaultRadiobutton'),...
                  'Parent', hDistanceMethod,...
                  'String','Planar',...
                  'UserData', 'planar',...
                  'Tag', 'RadiobuttonPlanar',...
                  'Position',[0.05 0.66 0.95 0.33],...
                  'FontSize', 0.4);

        uicontrol(getpref('DefaultRadiobutton'),...
                  'Parent', hDistanceMethod,...
                  'String','Haversine',...
                  'UserData', 'haversine',...
                  'Tag', 'RadiobuttonHaversine',...
                  'Position',[0.05 0.33 0.95 0.33],...
                  'FontSize', 0.4);

        hDijkstraRadio = uicontrol(getpref('DefaultRadiobutton'),...
                  'Parent', hDistanceMethod,...
                  'String','Dijkstra',...
                  'UserData', 'dijkstra',...
                  'Tag', 'RadiobuttonDijkstra',...
                  'Position',[0.05 0 0.95 0.33],...
                  'FontSize', 0.4);

        if isempty(project.geonodes.data)
            set(hDijkstraRadio, 'Enable', 'off');
        end

        obj = findobj('UserData',...
            char(project.properties.getProperty('distComputeMethod')));
        set(hDistanceMethod, 'SelectedObject', obj);

        function distanceMethod_Callback(~, eventdata)
            switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
                case {'RadiobuttonHaversine', 'RadiobuttonPlanar'}

                    newProjProp.setProperty('distComputeMethod',...
                        get(eventdata.NewValue,'UserData'));
                case 'RadiobuttonDijkstra'
                    newProjProp.setProperty('distComputeMethod',...
                        get(eventdata.NewValue,'UserData'));
                otherwise
            end
        end
    end

    function createCoordRangeObjects()
        hLatPanel = uipanel(getpref('DefaultPanel'),...
            'FontSize', 0.12,...
            'Tag', 'LatPanel',...
            'Title','Latitude range',...
            'Position',[0.38 0.68 0.18 0.21]);

        uicontrol(getpref('DefaultPopupmenu'),...
                  'Parent', hLatPanel,...
                  'Tag', 'LatPopupFrom',...
                  'Position',[0.1 0.7 0.8 0.15],...
                  'Callback', @changePopup_Callback);

        uicontrol(getpref('DefaultPopupmenu'),...
```

```
                'Parent', hLatPanel,...
                'Tag', 'LatPopupTo',...
                'Position',[0.1 0.3 0.8 0.15],...
                'Callback', @changePopup_Callback);


    hLongPanel = uipanel(getpref('DefaultPanel'),...
        'FontSize', 0.12,...
        'Tag', 'LongPanel',...
        'Title','Longitude range',...
        'Position',[0.57 0.68 0.18 0.21]);

    uicontrol(getpref('DefaultPopupmenu'),...
                'Parent', hLongPanel,...
                'Tag', 'LongPopupFrom',...
                'Position',[0.1 0.7 0.8 0.15],...
                'Callback', @changePopup_Callback);

    uicontrol(getpref('DefaultPopupmenu'),...
                'Parent', hLongPanel,...
                'Tag', 'LongPopupTo',...
                'Position',[0.1 0.3 0.8 0.15],...
                'Callback', @changePopup_Callback);
end

function createGridObjects()
    hGridPanel = uipanel(getpref('DefaultPanel'),...
        'FontSize', 0.12,...
        'Tag', 'GridPanel',...
        'Title','Grid map params',...
        'Position',[0.01 0.38 0.74 0.21]);

    uicontrol(getpref('DefaultPopupmenu'),...
                'Parent', hGridPanel,...
                'Tag', 'GridPrecisPopup',...
                'Position',[0.44 0.77 0.09 0.15],...
                'Value', 2);

    uicontrol(getpref('DefaultButton'),...
        'Parent', hGridPanel,...
        'Position',[0.04 0.12 0.27 0.35],...
        'String','Clear grid',...
        'Callback',@clearGrid_Callback,...
        'Tag','ClearGridBtn',...
        'TooltipString', 'Clear grid');
    uicontrol(getpref('DefaultButton'),...
        'Parent', hGridPanel,...
        'Position',[0.36 0.12 0.27 0.35],...
        'String','Calculate grid',...
        'Callback',@calculateGrid_Callback,...
        'Tag','CalculateGridBtn',...
        'TooltipString', 'Calculate grid');

    uicontrol(getpref('DefaultText'),...
              'Tag', 'GridMapField',...
              'Parent', hGridPanel,...
              'String', 'Grid map: ',...
              'HorizontalAlignment', 'right',...
              'Position', [0.02 0.65 0.15 0.25]);

    t1 = uicontrol(getpref('DefaultText'),...
              'Tag', 'GridInfoField',...
              'Parent', hGridPanel,...
              'UserData', {'empty', 'defined'},...
              'Position', [0.17 0.65 0.13 0.25]);

    uicontrol(getpref('DefaultText'),...
              'Tag', 'GridPixDeg',...
```

```
                'Parent', hGridPanel,...
                'String', 'point/degree',...
                'Position', [0.54 0.65 0.19 0.25]);

        uicontrol(getpref('DefaultText'),...
                'Tag', 'GridPrecField',...
                'Parent', hGridPanel,...
                'String', 'Precision:',...
                'HorizontalAlignment', 'right',...
                'Position', [0.3 0.65 0.13 0.25]);

        cell = get(t1, 'UserData');
        if isempty(project.gridMap)
            set(t1, 'String', cell{1});
        else
            set(t1, 'String', cell{2});
        end
    end

    function createGAObjects()
        hGAPanel = uipanel(getpref('DefaultPanel'),...
            'FontSize', 0.12,...
            'Tag', 'GAPanel',...
            'Title','Genetic alorithm params',...
            'Position',[0.01 0.13 0.74 0.21]);

        uicontrol(getpref('DefaultText'),...
                'Tag', 'GANumOfGenField',...
                'Parent', hGAPanel,...
                'String', 'Number of generations:',...
                'HorizontalAlignment', 'right',...
                'Position', [0.03 0.65 0.37 0.25]);

        uicontrol(getpref('DefaultPopupmenu'),...
                 'Parent', hGAPanel,...
                 'Tag', 'GANumOfGenPopup',...
                 'Position',[0.41 0.8 0.1 0.15],...
                 'String', {1:200},...
                 'Value', 20,...
                 'Callback', @changePopup_Callback);

        uicontrol(getpref('DefaultText'),...
                'Tag', 'GAGenMassField',...
                'Parent', hGAPanel,...
                'String', 'Generation mass:',...
                'HorizontalAlignment', 'right',...
                'Position', [0.03 0.35 0.37 0.25]);

        uicontrol(getpref('DefaultPopupmenu'),...
                 'Parent', hGAPanel,...
                 'Tag', 'GAGenMassPopup',...
                 'Position',[0.41 0.5 0.1 0.15],...
                 'String', {50:50:5000},...
                 'Value', 20,...
                 'Callback', @changePopup_Callback);
    end

    function calculateGrid_Callback(hObject, ~)

        btnLabel = get(hObject, 'String');
        set(hObject, 'String', 'Working...');
        drawnow;
        str = get(findobj('Tag','GridPrecisPopup'), 'String');
        val = get(findobj('Tag','GridPrecisPopup'), 'Value');
        precision = str2double(str{val});

        str = get(findobj('Tag', 'LatPopupFrom'), 'String');
        val = get(findobj('Tag', 'LatPopupFrom'), 'Value');
```

```
        latFrom = str{val};

        str = get(findobj('Tag', 'LatPopupTo'), 'String');
        val = get(findobj('Tag', 'LatPopupTo'), 'Value');
        latTo = str{val};

        str = get(findobj('Tag', 'LongPopupFrom'), 'String');
        val = get(findobj('Tag', 'LongPopupFrom'), 'Value');
        longFrom = str{val};

        str = get(findobj('Tag', 'LongPopupTo'), 'String');
        val = get(findobj('Tag', 'LongPopupTo'), 'Value');
        longTo = str{val};

        % join coast coordinates
        landAreas = shaperead('landareas.shp');
        x = [];
        y = [];
        for i=1:length(landAreas)
            x = [x landAreas(i).X];
            y = [y landAreas(i).Y];
        end

        latRange = [str2double(latFrom) str2double(latTo)];
        longRange = [str2double(longFrom) str2double(longTo)];
        [Z, ~] = vec2mtx(y , x, precision, latRange, longRange, 'filled');
        Z(Z == 0)=1;
        Z(Z == 2)=0;
        newGrid = Z;
        newProjProp.setProperty('gridLatitude', [latFrom ' ' latTo]);
        newProjProp.setProperty('gridLongitude', [longFrom ' ' longTo]);

        hGridInfo = findobj('Tag', 'GridInfoField');
        cell = get(hGridInfo, 'UserData');
        set(hGridInfo, 'String', cell{2});
        set(hObject, 'String', btnLabel);
    end

    function clearGrid_Callback(~, ~, ~)
        newGrid = [];
        newProjProp.setProperty('gridLatitude', '');
        newProjProp.setProperty('gridLongitude', '');
        hGridInfo = findobj('Tag', 'GridInfoField');
        cell = get(hGridInfo, 'UserData');
        set(hGridInfo, 'String', cell{1});
    end

    function changePopup_Callback(hObject, ~, ~)

            switch get(hObject, 'Tag')
                case {'LatPopupFrom', 'LatPopupTo'}
                    p1 = findobj('Tag', 'LatPopupFrom');
                    p2 = findobj('Tag', 'LatPopupTo');
                    str1 = get(p1, 'String');
                    str2 = get(p2, 'String');
                    strRange=[str1{get(p1,'Value')} ' ' str2{get(p2,'Value')}];
                    newProjProp.setProperty('latitudeRange', strRange);
                case {'LongPopupFrom', 'LongPopupTo'}
                    p1 = findobj('Tag', 'LongPopupFrom');
                    p2 = findobj('Tag', 'LongPopupTo');
                    str1 = get(p1, 'String');
                    str2 = get(p2, 'String');
                    strRange=[str1{get(p1,'Value')} ' ' str2{get(p2,'Value')}];
                    newProjProp.setProperty('longitudeRange', strRange);
                case 'GANumOfGenPopup'
                    str = get(hObject, 'String');
                    val = get(hObject, 'Value');
                    numOfGen = str(val);
```

```
                    newProjProp.setProperty('GANumOfGen', numOfGen);
                case 'GAGenMassPopup'
                    str = get(hObject, 'String');
                    val = get(hObject, 'Value');
                    genMass = str(val);
                    newProjProp.setProperty('GAGenMass', genMass);
            end
            adjustPopups();
    end

    function adjustPopups()
            p1 = findobj('Tag', 'LatPopupFrom');
            p2 = findobj('Tag', 'LatPopupTo');
            p3 = findobj('Tag', 'LongPopupFrom');
            p4 = findobj('Tag', 'LongPopupTo');
            latRange = str2num(newProjProp.getProperty('latitudeRange'));
            longRan = str2num(newProjProp.getProperty('longitudeRange'));
            p1Str = {(-90:0.5:latRange(2)-1)'};
            p2Str = {(latRange(1)+1:0.5:90)'};
            p3Str = {(-180:0.5:longRan(2)-1)'};
            p4Str = {(longRan(1)+1:0.5:180)'};
            set(p1, 'String', p1Str, 'Value', find(p1Str{:}==latRange(1)));
            set(p2, 'String', p2Str, 'Value', find(p2Str{:}==latRange(2)));
            set(p3, 'String', p3Str, 'Value', find(p3Str{:}==longRan(1)));
            set(p4, 'String', p4Str, 'Value', find(p4Str{:}==longRan(2)));

% limit precision to 1620000 pixels (5 points per degree for max range)
            maxPixPerDeg=floor(sqrt(1620000/(diff(latRange)*diff(longRan))));
            hGridPrecisPopup = findobj('Tag', 'GridPrecisPopup');
            actualValue = get(hGridPrecisPopup, 'Value');
            set(hGridPrecisPopup, 'String', {1:maxPixPerDeg},...
                'Value', min([maxPixPerDeg, actualValue]));

            p5 = findobj('Tag', 'GANumOfGenPopup');
            p5str = get(p5, 'String');
            numOfGen = str2double(newProjProp.getProperty('GANumOfGen'));
            set(p5, 'Value', find(numOfGen == cellfun(@str2double, p5str)));
            p6 = findobj('Tag', 'GAGenMassPopup');
            p6str = get(p6, 'String');
            genMass = str2double(newProjProp.getProperty('GAGenMass'));
            set(p6, 'Value', find(genMass == cellfun(@str2double, p6str)));

        end

    function applyOptions_Callback(~, ~)
        project.properties = newProjProp;
        project.gridMap = newGrid;
        guidata(findobj('Tag','MainFigure'), project);
        actualizeInfoFields();
        close(gcbf);
    end

end

% realize calculation with loaded values
function calculate_Callback(~, ~, ~)

project = guidata(findobj('Tag', 'MainFigure'));

[project.nodes.data, fval, pop] = compute(project);
if ~isempty(pop)
    project.finalpopulation = pop;
end
guidata(findobj('Tag', 'MainFigure'), project);
writeInfo(['Optimization result: ' num2str(fval/1000)]);

end
```

```
% close figure
function closeFigure_Callback(~, ~, ~)
    close(gcbf);
end

% create new project with default values
function newProject_Callback(~, ~, ~)
    p = guidata(findobj('Tag','MainFigure'));
    if ~isempty(p)
        choice = questdlg('Overwrite actual project?', ...
        'Warning', 'Yes', 'No', 'No');
        if ~strcmp(choice, 'Yes')
            return;
        end
    end

    writeInfo('======================');
    writeInfo('Creating New project...');

    guidata(findobj('Tag','MainFigure'), createProject());
    actualizeInfoFields();
end

%%%%%%%%%%%%%%%%%%%%
% Common functions %
%%%%%%%%%%%%%%%%%%%%

% create main window
function hMainFigure = createMainFigure()


% Main figure info area
messageFieldPosNorm = [0.01 0.02 0.758 0.4];
infoLeftPosNorm =[repmat(0.02 ,9,1) (0.91:-0.05:0.51)' repmat([0.3 0.04],9,1)];
infoRightPosNorm =[repmat(0.4 ,9,1) (0.91:-0.05:0.51)' repmat([0.3 0.04],9,1)];

% Main figure buttons
btnPosNorm = [repmat(0.78 ,8,1) (0.9:-0.08:0.34)' repmat([0.2 0.07],8,1)];

formColor = [0.5 0.7 0.9];

createGlobalPreferences();

% Main figure
hMainFigure = figure(getpref('DefaultFigure'),...
    'Name','Transport Calculations',...
    'Tag','MainFigure',...
    'Menubar','none');

createButtons();
createTextFields();

    function createGlobalPreferences()
        %try to remove old preferences
        prefCell = {'DefaultFigure',...
            'DefaultButton',...
            'DefaultText',...
            'DefaultPanel',...
            'DefaultRadiobutton',...
            'DefaultPopupmenu',...
            'DefaultCheckbox'};

        for i=1:length(prefCell)
            try
                rmpref(prefCell{i});
            catch ex
                writeInfo(['Missing preferences for ''' prefCell{i} ''': '...
                    ex.message]);
```

```
    end
end


% Default figure preferences
setpref('DefaultFigure',{'Units',...
                         'Position',...
                         'Color'}...
                        ,{'normalized',...
                          [0.1 0.3 0.5 0.5],...
                          formColor});

% Default button preferences
setpref('DefaultButton',{'Units',...
                         'Style',...
                         'FontUnits',...
                         'FontSize',...
                         'FontWeight'}...
                        ,{'normalized',...
                          'Push',...
                          'normalized',...
                          0.4,...
                          'bold'});

% Default text field preferences
setpref('DefaultText',{'Units',...
                       'Style',...
                       'FontUnits',...
                       'FontSize',...
                       'HorizontalAlignment',...
                       'BackgroundColor'}...
                      ,{'normalized',...
                        'Text',...
                        'normalized',...
                        0.6,...
                        'left',...
                        formColor});

% Default panel preferences
setpref('DefaultPanel',{'Units',...
                        'FontUnits',...
                        'FontSize',...
                        'BackgroundColor'}...
                       ,{'normalized',...
                         'normalized',...
                         0.18,...
                         formColor});

setpref('DefaultRadiobutton',{'Style',...
                              'Units',...
                              'FontUnits',...
                              'FontSize',...
                              'BackgroundColor'}...
                             ,{'Radio',...
                               'normalized',...
                               'normalized',...
                               0.18,...
                               formColor});

setpref('DefaultPopupmenu',{'Style',...
                            'Units',...
                            'FontUnits',...
                            'FontSize'}...
                           ,{'popupmenu',...
                             'normalized',...
                             'normalized',...
                             1});
```

```
        setpref('DefaultCheckbox',{'Style',...
                                   'Units',...
                                   'FontUnits',...
                                   'FontSize',...
                                   'BackgroundColor'
                                    }...
                                  ,{'checkbox',...
                                   'normalized',...
                                   'normalized',...
                                    1,...
                                    formColor});
    end

    function createButtons()
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(1,:),...
            'String','New project',...
            'Callback',@newProject_Callback,...
            'Tag','NewBtn',...
            'TooltipString', 'New project');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(2,:),...
            'String','Load project...',...
            'Callback',@load_Callback,...
            'Tag','LoadBtn',...
            'TooltipString', 'Load project');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(3,:),...
            'String','Save project...',...
            'Callback',@save_Callback,...
            'Tag','SaveBtn');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(4,:),...
            'String','Options...',...
            'Callback',@options_Callback,...
            'Tag','OptionBtn');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(5,:),...
            'String','Edit graph...',...
            'Callback',@editGraph_Callback,...
            'Tag','EditBtn');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(6,:),...
            'String','Show Map...',...
            'Callback',@showMap_Callback,...
            'Tag','ShowBtn');
        uicontrol(getpref('DefaultButton'),...
            'Position',btnPosNorm(7,:),...
            'String','Calculate...',...
            'Callback',@calculate_Callback,...
            'Tag','CalcBtn');
        uicontrol(getpref('DefaultButton'),...
            'Position', btnPosNorm(8,:),...
            'String','Quit',...
            'Callback',@quit_Callback,...
            'Tag','QuitBtn');
    end

    function createTextFields()
        % project file
        uicontrol(getpref('DefaultText'),...
             'Tag', 'ProjectFileField',...
             'UserData', 'Project file: ',...
             'Position', infoLeftPosNorm(1,:));


        % graph files
        uipanel(getpref('DefaultPanel'),...
```

```
            'Title','Graph files',...
            'Position',[0.01 0.75 0.37 0.14]);
        uicontrol(getpref('DefaultText'),...
            'Tag', 'NodesFileField',...
            'UserData', 'Nodes file: ',...
            'Position', infoLeftPosNorm(3,:));
        uicontrol(getpref('DefaultText'),...
            'Tag', 'EdgesFileField',...
            'UserData', 'Edges file: ',...
            'Position', infoLeftPosNorm(4,:));


        % coordinates
        uipanel(getpref('DefaultPanel'),...
            'Title','Coordinate range',...
            'Position',[0.39 0.75 0.37 0.14]);
        uicontrol(getpref('DefaultText'),...
            'Tag', 'LatitudeRange',...
            'UserData', 'Latitude range: ',...
            'Position', infoRightPosNorm(3,:));
        uicontrol(getpref('DefaultText'),...
            'Tag', 'LongitudeRange',...
            'UserData', 'Longitude range: ',...
            'Position', infoRightPosNorm(4,:));

        % Distance method
        uicontrol(getpref('DefaultText'),...
            'Tag', 'ComputeDistMethod',...
            'UserData', 'Distance method: ',...
            'Position', infoLeftPosNorm(6,:));

        % grid
        uipanel(getpref('DefaultPanel'),...
            'Title','Grid map',...
            'Position',[0.39 0.5 0.37 0.24],...
            'FontSize', 0.105);
        uicontrol(getpref('DefaultText'),...
            'Tag', 'GridMatrix',...
            'UserData', 'Grid size: ',...
            'Position', infoRightPosNorm(6,:));
        uicontrol(getpref('DefaultText'),...
            'Tag', 'GridLatitude',...
            'UserData', 'Grid latitude range: ',...
            'Position', infoRightPosNorm(7,:));
        uicontrol(getpref('DefaultText'),...
            'Tag', 'GridLongitude',...
            'UserData', 'Grid longitude range: ',...
            'Position', infoRightPosNorm(8,:));


        uicontrol('Tag', 'MessageField',...
            'Style', 'Text',...
            'Units', 'normalized',...
            'FontUnits', 'normalized',...
            'FontSize', 0.06,...
            'HorizontalAlignment', 'left',...
            'BackgroundColor', [0.9 0.9 0.9],...
            'Position', messageFieldPosNorm);
    end
end

% create new project with default values
function project = createProject()
    mFilePath = fileparts(mfilename('fullpath'));

    project = struct;
    % default properties
    project.properties = java.util.Properties;
```

```
    project.properties.setProperty('projFile', '');
    project.properties.setProperty('nodesFile', '');
    project.properties.setProperty('edgesFile', '');
    project.properties.setProperty('latitudeRange', '-90 90');
    project.properties.setProperty('longitudeRange', '-180 180');
    project.properties.setProperty('gridMapFile', '');
    project.properties.setProperty('gridLatitude', '');
    project.properties.setProperty('gridLongitude', '');
    project.properties.setProperty('distComputeMethod', 'haversine');
    project.properties.setProperty('geoNetworkNodesFile', '');
    project.properties.setProperty('geoNetworkEdgesFile', '');
    project.properties.setProperty('GANumOfGen', '30');
    project.properties.setProperty('GAGenMass', '500');

    % default graph
    project.nodes.headers = {'NodeID', 'Type', 'Latitude','Longtitude', 'Name'};
    project.nodes.data = {1,'plant',48.133333,11.566667,'Munich';...
                          2,'purch',51.507222,-0.1275,'London'};
    project.edges.headers = {'SourceNode', 'DestinationNode', 'Cost', 'Name'};
    project.edges.data = {1,2,1,'Path 1'};

    % default geographic network
    project.geonodes.data = [];
    project.geonodes.headers = [];
    project.geoedges.data = [];
    project.geoedges.headers = [];

    project.gridMap = [];

    % load borders
    worldFile = fullfile(mFilePath, filesep, 'worldlo.mat');
    project.worldBorders = load(worldFile, 'POline');
    writeInfo(strcat('Borders file "worldlo.mat" loaded'));
end

% actualize information text fields on main figure
function actualizeInfoFields()
    project = guidata(findobj('Tag','MainFigure'));

    setInfo('ProjectFileField', project.properties.get('projFile'));
    setInfo('NodesFileField', project.properties.get('nodesFile'));
    setInfo('EdgesFileField', project.properties.get('edgesFile'));
    setInfo('LatitudeRange', ['<' project.properties.get('latitudeRange') '>']);
   setInfo('LongitudeRange',['<' project.properties.get('longitudeRange') '>']);
    setInfo('ComputeDistMethod', project.properties.get('distComputeMethod'));

    setInfo('GridLatitude', ['<' project.properties.get('gridLatitude') '>']);
    setInfo('GridLongitude', ['<' project.properties.get('gridLongitude') '>']);
    setInfo('GridMatrix', ['<' num2str(size(project.gridMap)) '>']);

    function strOut = setInfo(objTag, str)
        hObject = findobj('Tag',objTag);
        if isempty(str)
            strOut = [get(hObject, 'UserData') 'NA'];
        else
            strOut = [get(hObject, 'UserData') str];
        end
        set(findobj('Tag', objTag), 'String', strOut);
    end
end

% write info
function writeInfo(str)
    hField = findobj('Tag', 'MessageField');
    str = [datestr(now, '<HH:MM:SS.FFF> '), str];
    if isempty(hField)
        disp(str);
        return;
```

```
    end

    oldContent = get(hField,'String');
    if isempty(oldContent)
        set(hField,'String', str);
    elseif iscell(oldContent)
        ccc = {oldContent{end-min(10,end)+1:end} str};
        set(hField,'String', ccc);
    else
        ccc = {oldContent; str};
        set(hField,'String', ccc);
    end
end

% write error
function writeError(str)
    hField = findobj('Tag', 'MessageField');
    str = [datestr(now, 'ERROR: <HH:MM:SS.FFF> '), str];
    oldContent = get(hField,'String');
    if isempty(hField)
        disp(str);
        return;
    end

    if isempty(oldContent)
        set(hField,'String', str);
    elseif iscell(oldContent)
        ccc = {oldContent{end-min(10,end)+1:end} str};
        set(hField,'String', ccc);
    else
        ccc = {oldContent; str};
        set(hField,'String', ccc);
    end
end

% convert string to double in specified cell columns
function cell = str2DoubleInCell(cell, columns)
    for i=1:size(cell,1)
        for j=columns
            cell{i,j} = str2double(strrep(cell{i,j}, ' ', ''));
        end
    end
end
```

Licenční smlouva společnosti MaxMind Inc.

```
OPEN DATA LICENSE for MaxMind WorldCities and Postal Code Databases

Copyright (c) 2008 MaxMind Inc.  All Rights Reserved.

All advertising materials and documentation mentioning features or use of
this database must display the following acknowledgment:
"This product includes data created by MaxMind, available from
http://www.maxmind.com/"

Redistribution and use with or without modification, are permitted provided
that the following conditions are met:
1. Redistributions must retain the above copyright notice, this list of
conditions and the following disclaimer in the documentation and/or other
materials provided with the distribution.
2. All advertising materials and documentation mentioning features or use of
this database must display the following acknowledgement:
"This product includes data created by MaxMind, available from
http://www.maxmind.com/"
3. "MaxMind" may not be used to endorse or promote products derived from this
database without specific prior written permission.

THIS DATABASE IS PROVIDED BY MAXMIND.COM ``AS IS'' AND ANY
```

```
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR a PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL MAXMIND.COM BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
DATABASE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```