VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

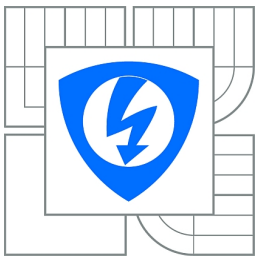
EVOLUCNÍ ALGORITMY
EVOLUTIONARY ALGORITHMS

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE
AUTHOR
BC. MARTIN BORTEL

VEDOUCÍ PRÁCE
SUPERVISOR
ING. PETRA LAMBERTOVÁ

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Bortel	***ID:***	83906
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Evoluční algoritmy

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce nastudujte a popište základní principy evolučních algoritmů. Zvýšenou pozornost věnujte hlavně genetickým algoritmům. V praktické části se zaměřte na využití evolučních algoritmů pro optimalizaci návrhu distribuční trasy.

DOPORUČENÁ LITERATURA:

[1] ZELINKA, Ivan, et al. Evoluční výpočetní techniky - principy a aplikace. [s.l.] : BEN, 2009. 536 s. ISBN 80-7300-218-3.
[2] HYNEK, Josef. Genetické algoritmy a genetické programování. [s.l.] : Grada, 2008. 200 s. ISBN 978-80-247-2695-3.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Petra Lambertová

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

Klíčová slova:

Evoluční algoritmus, Genetický algoritmus, Fitness, Gen, Chromozom

Key words:

Evolutionary algorithm, Genetic algorithm, Fitness, Gen, Chromosome

Abstrakt:

Práce se zabývá principy a základními vlastnostmi Evolučních a Genetických algoritmů. Jsou zde rozebrány operátory mutace, křížení a selekce a možnosti ukončení algoritmu. Uvedeny jsou příklady využití evolučních a genetických algoritmů v praxi. Využití technologií PHP&MySQL a Google Maps API k optimalizaci distribuční trasy, je důležitým bodem práce.

Abstract:

Thesis describes main attributes and principles of Evolutionary and Genetic algorithms. Crossover, mutation and selection are described as well as termination options. There are examples of practical use of evolutionary and genetic algorithms. Optimization of distribution routes using PHP&MySQL and Google Maps API technologies.

Citace:

BORTEL, M. Evoluční algoritmy. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 42 s. Vedoucí diplomové práce Ing. Petra Lambertová.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Evoluční algoritmy jsem vypracoval samostatně pod vedením vedoucí diplomové práce Ing. Petry Lambertové a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne 25.05.2011 .....

podpis autora

Poděkování

Děkuji vedoucí diplomové práce Ing. Petře Lambertové, za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 25.05.2011 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Evoluční algoritmus spadá do rodiny stochastických vyhledávacích technik, známých jako Evoluční Výpočetní Techniky (EVT). Mezi tyto techniky patří také Genetické algoritmy, Evoluční strategie nebo Evoluční programování. Evoluční výpočetní techniky popisují metody řešení problémů, pro které neexistuje proprietální funkce, či problémů, jejichž řešení konvenčními postupy má příliš vysokou časovou náročnost. Tyto techniky využívají pro řešení, nebo v tomto kontextu lépe řečeno pro šlechtění výsledku, zákonitosti evoluce, tak jak je popsal ve své knize „*O vzniku druhů přírodním výběrem, neboli uchováním prospěšných plemen v boji o život*“ Charles Darwin. Základní a možná nejdůležitější vlastností vyhledávacích algoritmů spadajících do rodiny evolučních výpočetních technik je skutečnost, že od samého začátku nepracují pouze s jedním řekněme parciálním řešením daného problému, ale hned s celou „populací“ takových řešení. To ve spojení s celkovou robustností a obecnou definicí těchto algoritmů umožňuje řešit takřka jakoukoliv úlohu.

Tento text se zabývá vlastnostmi části rodiny evolučních výpočetních technik a to evolučních algoritmů a zejména genetických algoritmů. Jsou zde shrnuty některé možné aplikace evolučních, potažmo genetických, algoritmů.

Část Optimalizace distribuční trasy se zabývá možností kombinace skriptovacího jazyka jako je PHP a MySQL úložiště, s využitím API rozhraní služby Google Maps, ve prospěch řešení optimalizačního úkolu.

1. Terminologie

V textu se budeme setkávat s některými pojmy známými z biologie, genetiky. Proto v této části shrňme ty nejdůležitější termíny, které se ve spojení s genetickými algoritmy, potažmo evolučními algoritmy, používají.

1.1. Gen

V souvislosti s evolučním algoritmem bude gen reprezentovat vlastnosti jedince. Na rozdíl od přírody je v tomto kontextu význam pojmu gen výrazně zjednodušen. Je reprezentován znakem, který popisuje vlastnost genu, např. genetický algoritmus nejčastěji používá pro reprezentaci genu binární hodnotu, tedy 0 nebo 1. Ale obecně může být pro zápis vlastnosti genu využit jiný zápis, např. dekadický.

Jako znázornění můžeme uvést problém obchodního cestujícího. Obchodní cestující má za úkol navštívit předem daný počet měst a trasa mezi nimi má být co možná nejkratší, nebo co možná nejrychlejší. Gen bude v tomto případě popisovat délku jednoho spojení mezi 2 městy. Pro tento problém bude zřejmě výhodnější využít dekadického zápisu genomu.

1.2. Chromozom

V přírodě je chromozom tvořen dvoušroubovicí kyseliny deoxyribonukleové (DNA). DNA je stočena a vytváří tím prostorový útvar známý jako chromozom.

Zde je chromozom souborem genů jedince. Je reprezentován numerickým nebo textovým řetězcem. Takový řetězec je složen z hodnot jednotlivých genů seřazených do posloupnosti. Chromozom nebo řetězec, který chromozom reprezentuje, může vypadat například takto: 1101011101.

Binární zápis je nejjednodušší variantou. Vlastnosti genů mohou být vyjádřeny i prostřednictvím jiných datových typů.

Datové typy, využívané pro deklaraci vlastností genů:

- **INTEGER**

  Dekadické celé číslo o velikosti 32bit (záleží na hardware).

  Rozsah: $< -2\ 147\ 483\ 648;\ \ 2\ 147\ 483\ 647 >$.

- **FLOAT**

  Dekadické reálné číslo s rozlišením až na 7 desetinných míst.

  Rozsah přibližně: $< \mathbf{1,5 \cdot 10^{-45}};\ \mathbf{3,4 \cdot 10^{38}} >$.

- **CHAR**

  Char je zkratkou pro character, což znamená znak.

  velikost jednoho znaku je 8bit, tedy 1Byte.

  pro win32, pokud není explicitně nastaveno jinak, je kódování typu CHAR unicode.

- **VARCHAR neboli STRING**

  Datový typ schopný uložit CHAR a FLOAT či INTEGER v jednom, všechny typy jsou však převedeny na typ CHAR a uloženy do paměti jako sekvence (pole) 8-bitových znaků CHAR.

V případě že bude použit dekadický zápis, může chromozom vypadat například takto: 96573946213.

Pro přehlednější vyjádření genetické informace lze využít jiného typu reprezentace. Geny, ze kterých je chromozom, složen mohou být zapsány ve dvojici s pořadím genu v chromozomu. Řetězec pak bude vypadat například takto:

- **TINYINT (BIN)**

  `{(0, 1)(1, 1)(2, 0)(3, 1)(4, 0)(5, 1)(6, 1)}`

- **INTEGER**

  `{(0; 9)(1; 6)(2; 5)(3; 7)(4; 3)(5; 9)(6; 4)}`

- **FLOAT**

  `{(0; 0,03)(1; 1,342)(2; 6,2)(3; 7,034)(4; -3,45)(5; 9,4837)(6; 4)}`

Zápis používá princip reprezentace genu dvojicí prvků. První prvek udává pozici v chromozomu a druhý vlastní hodnotu genu.

1.3. Genotyp

Genotyp je souhrn všech genů jedince. Představuje tedy celou jeho genetickou výbavu. Pro genetický algoritmus platí, že v chromozomu je obsažena tatáž informace jako v genotypu. V přírodě může genotyp obecně obsahovat větší množství informace, než která je obsažena v jednom chromozomu. Například u člověka je genotyp složen ze 46 chromozomů a každý z těchto chromozomů obsahuje množství genů reprezentovaných v DNA.

1.4. Fenotyp

Fenotyp je fyzickým projevem genotypu. Příkladem fenotypu může být grafické znázornění jedince z populace řešení „Problému obchodního cestujícího“.

1.5. Jedinec

Jedinec nebo individuum je základním článkem populace. Je definován genetickou informací, genotypem. Každý jedinec má unikátní genetickou výbavu. A v rámci GA je jedinec definován jediným chromozomem o pevné délce.

1.6. Populace

Populace je množinou všech jedinců, respektive partikulárních řešení zadaného problému.

2. Definice průběhu algoritmu

Následující schematický popis znázorňuje průběh genetického, potažmo evolučního algoritmu. Objevuje se zde pojem funkce „Fitness“, v literatuře můžeme najít také Hodnotící funkce, nebo pojem Úspěšnost jedince (fitness). Funkci Fitness a její výstup popisuji v odstavci nazvaném Funkce Fitness (viz 3).

I. Vynulování počítadla generací $t = 0$.

II. Náhodná generace první populace $P(0)$.

III. Vyhodnocení úspěšnosti všech jedinců v počáteční populaci (funkce Fitness).

IV. Selekce dvojic jedinců z populace $P(t)$, stanou se rodiči.

V. Vytvoření potomků $P'(t)$ populace $P(t)$.

VI. Ohodnocení individuí $P'(t)$ a vybrání nových dvojic – rodičů ${P_r}'(t)$

VII. Vytvoření nové populace $P(t+1)$ z populace $P(t)$, resp. ${P_r}'(t)$.

VIII. Inkrementace počítadla generací $t = t + 1$.

IX. Výpočet fitness individuí $P(t)$

X. Pokud t je rovno maximálnímu počtu generací, nebo je splněna jiná podmínka ukončení -> vrať jak výsledek populaci $P(t)$; Jinak pokračuj bodem 4.

3. Funkce Fitness

Česky bychom Fitness přeložili jako úspěšnost, vhodnost. A vyjadřuje míru schopnosti jedince se danému prostředí přizpůsobit, přežít a předat svou genetickou informaci dalším generacím.

Pro vytvoření nové populace z populace předchozí je nezbytné vydefinovat kritérium, podle kterého se budou vybírat úspěšní jedinci (rodiče). V přírodě jedinci mezi sebou soutěží v přirozeném prostředí, ti kteří se s životními nástrahami vypořádají nejlépe, mají obecně větší naději zplodit potomky, nebo mít potomků více než ti méně úspěšní. V prostředí umělého reprodukčního procesu je třeba podobnou soutěž „o přežití“ nějakým způsobem nasimulovat. K tomu slouží funkce Fitness.

Funkce fitness ve spojení s operátorem selekce (viz 4.3) nahrazují přirozený výběr známý z evolučního procesu v přírodě. Funkce fitness má za úkol zhodnotit jednotlivé jedince populace v závislosti na předpokládaném řešení. Individua, která se v tomto hodnocení nejvíce přiblíží k předpokládanému řešení, bývají ve fázi výběru, tzv. selekce, upřednostněni. Upřednostňování úspěšnějších jedinců však nemusí být nutně pravidlem.

Výstupem fitness funkce je míra úspěšnosti daného jedince, f-ce je volána nad každým jedincem populace bez výhrad.

4. Operátory

Operátory v oblasti genetických algoritmů představují sadu nástrojů, pomocí níž lze z rodičovské generace získat generaci další, potomky. Jak již bylo řečeno, genetický algoritmus implementuje základní principy reprodukce organizmů. Abychom získali novou generaci, musí proběhnout řada událostí. První z nich bude bezesporu výběr partnerů, kteří se spolu budou „pářit“. Následovat bude výměna genetické informace mezi partnery a její předání potomkům, toto lze provést několika způsoby a je zřejmé, že na výběru správné metody bude záviset konečný výsledek.

4.1. Operátor křížení

Křížení je jedním ze základních operátorů genetického algoritmu, který se dále dělí dle principu výměny genetické informace mezi rodičovskými chromozomy.

Je to procedura, v rámci které dochází k vzájemné výměně informace mezi dvěma rodičovskými chromozomy a k následnému vytvoření dvou potomků. Existuje více možností jak vybrat jednu či více částí rodičovského chromozomu. Výměna genetické informace s partnerským rodičovským chromozomem se může také různit. V následujících odstavcích uvádím ty nejdůležitější typů operátoru křížení.

4.1.1. Metoda bližšího souseda

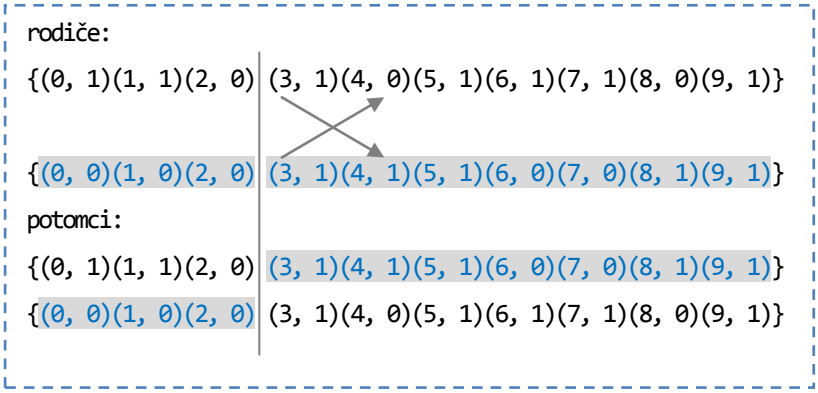
Pokud uvažujeme problém optimalizace distribuční trasy, vyplývá na povrch potřeba zachovat všechny geny z chromozomu pohromadě. Gen v tomto případě totiž definuje jeden bod na mapě (např. překladní místo). Chromozom definuje, jako posloupnost genů, jednu možnou trasu. Z tohoto důvodu nelze chromozomy křížit běžnými výměnami genetické informace. Můžeme pouze zaměňovat pořadí genů.

Principem metody je porovnání vzdálenosti mezi aktuálně zvoleným bodem (genem) a následujícím bodem v chromozomu obou rodičů. Ten následník, který je blíže, je vybrán.

4.1.2. 1-bodové křížení

První a zřejmě nejjednodušší typ křížení. V rodičovském chromozomu je určen jeden dělící gen, pro oba partnery stejný. Od tohoto genu dále je genetická informace mezi chromozomy zaměněna.

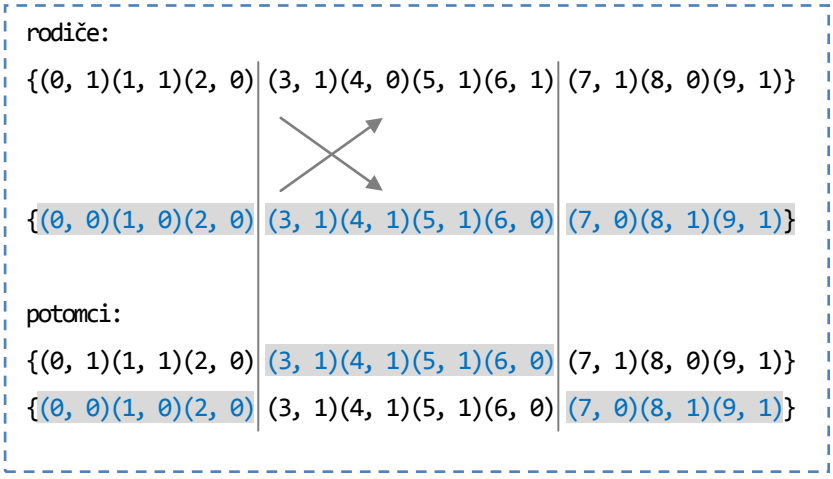
Příklad:



4.1.3. k-bodové křížení

Obecnější varianta předchozího typu křížení. V páru rodičovských chromozomů je určeno k dělících genů, pro oba partnery stejné.

Příklad: $k = 2$



4.1.4. Uniformní křížení

Třetí typ operátoru křížení je založen na výběru genů s pravděpodobností $P(0{,}5)$, které budou mezi rodiči vyměněny. Pravděpodobnostní hodnota se nazývá míra míšení (*mixing ratio*). Vybrán je stejný počet genů z obou rodičovských chromozomů a tyto geny jsou poté přiřazeny partnerskému rodičovskému chromozomu.

Příklad: $mixing\ ratio = 0{,}5$

```
rodiče:
1: { (0, 1)(1, 1)(2, 0)(3, 1)(4, 0)(5, 1)(6, 1)(7, 1)(8, 0)(9, 1) }
2: { (0, 0)(1, 0)(2, 0)(3, 1)(4, 1)(5, 1)(6, 0)(7, 0)(8, 1)(9, 1) }
potomci:
{(0, 1)1(1, 0)2(2, 0)2(3, 1)1(4, 1)2(5, 1)1(6, 1)1(7, 0)2(8, 0)1(9, 1)2}
{(0, 0)2(1, 1)1(2, 0)1(3, 1)2(4, 0)1(5, 1)2(6, 0)2(7, 1)1(8, 1)2(9, 1)1}
```

4.1.5. Aritmetické křížení

Operátor křížení, který doslova kombinuje dva chromozomy, jako by to byly vektory. Potomci vznikají dle vztahů popsaných rovnicemi(4.1) a (4.2)

$$4.1\ p_1 = \sum[a \cdot r_1 + (1 - a) \cdot r_2]$$

$$4.2\ p_2 = \sum[(1 - a) \cdot r_1 + a \cdot r_2]$$

$p_{1,2}$ … potomek 1,2

$r_{1,2}$ … rodič 1,2

a … náhodný váhový faktor, volený před každou operací křížení

Příklad:

```
Rodič 1: {(0; 0,4)(1; 0,6)(2; 0,1)(3; 1,4)}
Rodič 2: {(0; 0,5)(1; 2,1)(2; 4,5)(3; 5,6)}

Pokud a = 0.7, potomci budou vypadat následovně:
Potomek 1: {(0; 0,43)(1; 1,05)(2; 1,42)(3; 2,66)}
Potomek 2: {(0; 0,47)(1; 1,65)(2; 3,18)(3; 4,34)}
```

4.1.6. Heuristické křížení

Heuristický typ operátoru křížení určuje směr dalšího vývoje dle hodnoty fitness rodičů. Potomstvo je definováno rovnicemi (4.3) a (4.4). Může se stát, že $p1$ (první z potomků) nebude přípustný. Hodnota jednoho či více genů tohoto potomka se při špatně zvoleném n může dostat mimo povolený rozsah. Proto se do heuristického typu křížení zavádí uživatelský parametr k, který určuje počet pokusů o nalezení vhodného n, tak aby byl $p1$ přípustný. Pokud ani po k pokusů nebyl přijatelný chromozom ($p1$) nalezen, je přiřazen tomuto potomkovi rodičovský chromozom s nejnižší hodnotou fitness.

$$4.3\ p_1 = r_{best} + n \cdot [r_{best} - r_{worst}]$$

$$4.4\ p_2 = r_{best}$$

r_{best} … rodič s nejvyšší hodnotou fitness
r_{worst} …rodič s nejnižší hodnotou fitness
n … náhodné číslo z intervalu (0,1)
$p_{1,2}$ … potomci

4.2. Operátor mutace

Mutace je genetický operátor, který mění jeden nebo více hodnot genů v chromozomu jedince. Výsledkem mutace může zanesení zcela nové genetické informace do genomu jedince. Díky mutacím lze u genetického algoritmu zabránit uvíznutí v lokálním optimu.

Tento operátor je použit na základě pravděpodobnostní hodnoty výskytu mutace, kterou nastaví uživatel. Pravděpodobnost výskytu by měla být nastavena s velkou opatrností, příliš velká hodnota bude mít za následek degradaci GA na náhodný vyhledávací algoritmus a neúměrné snížení efektivity algoritmu.

4.2.1. Replikační

Metoda, která vybere náhodně dvě pozice, resp. dva geny v chromozomu, a zamění jejich hodnoty mezi sebou.

Pokud budeme mít chromozom: `{(0, 1)(1, 0)(2, 1)(3, 0)}`, tučně označené geny jsou vybrány R_1 `[(1, 0)]` a R_2 `[(2, 1)]`. Výsledek potom bude: `{(0, 1)(1, 1)(2, 0)(3, 0)}`.

4.2.2. Přepnutí bitu

Jako druhý z možných typů operátoru mutace uvádím bitové přepnutí (Flip Bit). Tato metoda využívá jednoduchý princip: hodnota zvoleného genu chromozomu je zaměněna za opačnou. Pokud bude chromozom vypadat následovně: `{(0, 1)(1, 0)(2, 1)(3, 0)}` a gen, na kterém má být záměna bitu provedena, bude na pozici např. 2, po provedení mutace bude chromozom vypadat takto: `{(0, 1)(1, 0)(2, 0)(3, 0)}`.

Už název napovídá, že přepnutí bitu lze použít pouze pro geny, které jsou reprezentovány BIN zápisem, nebo datovým typem TINYINT (hodnoty 1 nebo 0).

4.2.3. Neuniformní

Tento typ operátoru mutace umožňuje postupné snižování pravděpodobnosti výskytu mutace v závislosti na zvyšování počtu generačních cyklů. Jinými slovy, tento typ mutace efektivně zabrání stagnaci populace v raném stádiu vývoje, ale umožňuje genetickému algoritmu jemné doladění řešení v pozdějších stádiích vývoje.

Neuniformní operátor mutace může být použit pouze pro geny, které jsou definovány jinými datovými typy než binárně (integer, float).

4.2.4. Uniformní

Operátor mutace uniformního typu zaměňuje hodnotu náhodně vybraných genů za hodnotu rovnoměrně náhodně vybranou z intervalu, který předem určí uživatel.

Uniformní operátor mutace může být použit pouze pro geny, které jsou definovány jinými datovými typy než binárně (integer, float).

4.3. Operátory selekce

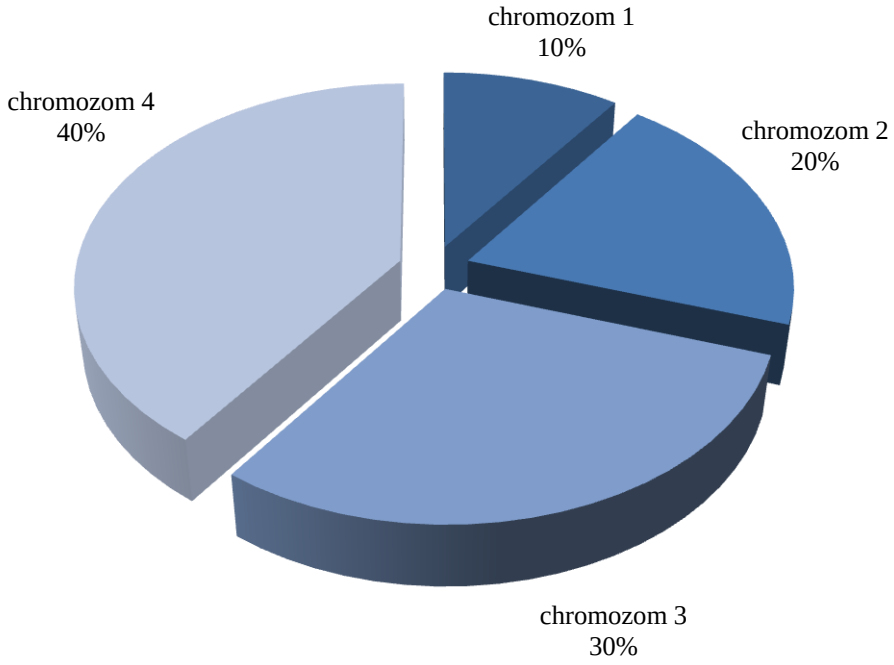
Po provedení zhodnocení jedinců v populaci následuje výběr partnerů, kteří „zplodí“ potomky. V přírodě by se tato fáze genetického algoritmu dala přirovnat k námluvám. Nejčastěji si namlouvá samec samici, snaží se ji zaujmout, upoutat na sebe pozornost, a přesvědčit ji o tom, že je ten nejlepší mezi svou konkurencí. V případě, že jedinci žijí ve stádě, má právo plodit potomky se samicemi ze stáda pouze nejsilnější samec. Pozice tzv. alfa samce musí být vybojována v souboji jeden na jednoho. Samec, který prohraje, je nejčastěji vykázán za hranice teritoria alfa samce, resp. stáda, ale existují výjimky z tohoto pravidla. Pokud předchozí zobecníme, vyplývá z toho, že evoluce zajistila poměrně rozmanitou množinu možností výběru nejzdatnějších z dané populace. Když máme principy evoluce simulovat, stojíme před úkolem jak najít k této množině alternativu, která umožní co nejlépe definovat, kteří jedinci určí správný směr evoluce.

Možností jak z rodičovské populace vybrat ty nejúspěšnější je mnoho, uvedeme si zde ty nejrozšířenější.

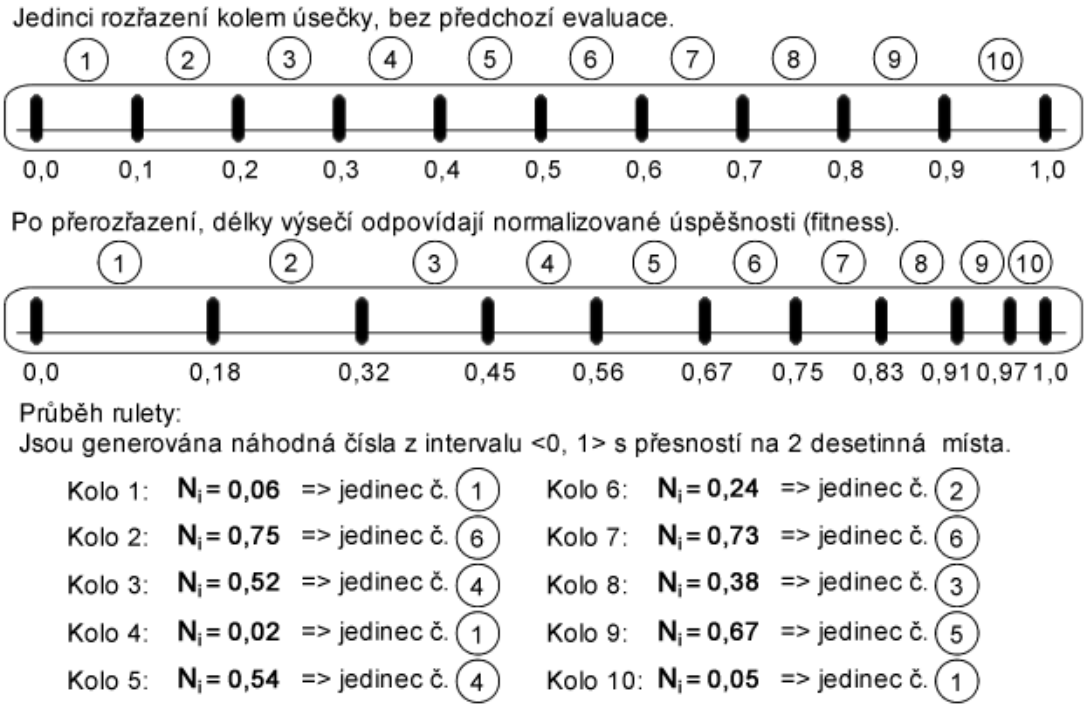
4.3.1. Ruleta

Nejjednodušší selekční schéma, můžeme o něm hovořit také jako o stochastickém vzorkování s náhradou [1]. Jedincům populace jsou přiděleny výseče kruhu, přičemž velikost přidělené výseče je přímo úměrné hodnotě fitness jedince. Následně je simulováno vložení kuličky do rulety, jejích běh a ustálení v jedné z výsečí, čímž je vybrán nový rodič. Kuliček může být ve stejném okamžiku vypuštěno více (maximálně však do počtu jedinců, ze kterých je vybíráno). V jednom běhu rulety tak můžou být vybrání všichni rodiče. Kruh a jeho výseče jsou zde použity pro snazší představu o principu. Pro efektivní využití tohoto typu selekce je použit interval od 0 do 1, ten je rozdělen na potřebný počet segmentů o délce úměrné hodnotě fitness jednotlivých jedinců. Simulace vložení kuličky do rulety je potom jednoduše provedeno vygenerováním náhodného čísla v intervalu <0, 1>. Jedinec, do jehož segmentu náhodně vygenerovaná hodnota spadá, je vybrán.

Přidělení výsečí jedincům může vypadat následovně.



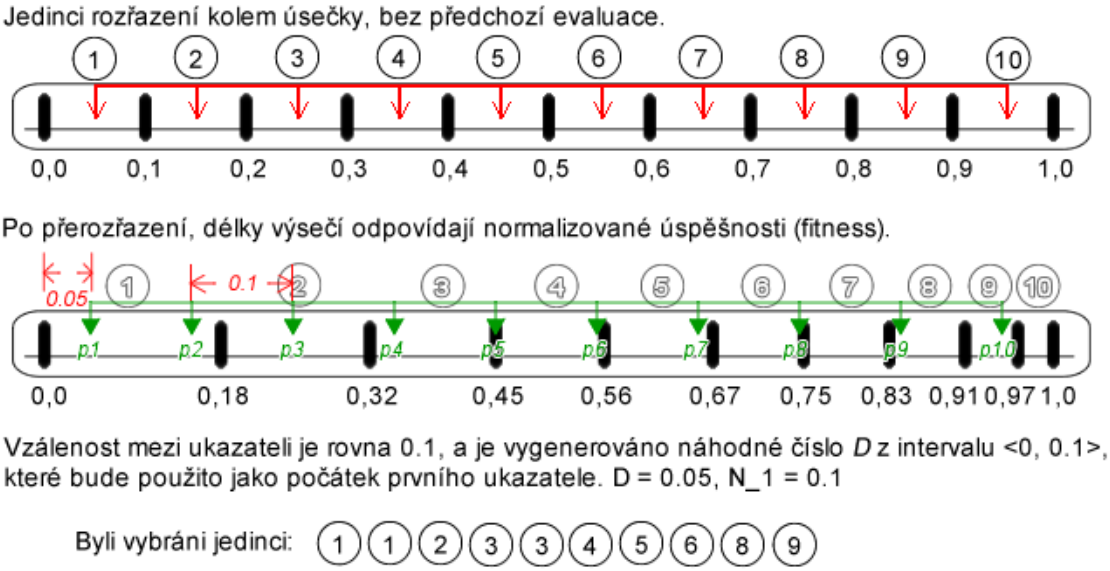
Obrázek 4.1 Znázornění jak může vypadat přidělení výsečí rulety jedincům.



Obrázek 4.2 Zobrazení lineárního rozdělení pro ruletový výběr.

4.3.2. Stochastické univerzální vzorkování

Podobně jako u ruletového selekčního mechanizmu, i u této metody jsou jedincům přiděleny části intervalu s ohledem na jejich hodnotu fitness. Poté jsou přes tento interval rozloženy ukazatele, počet ukazatelů je stejný jako počet jedinců kteří mají být vybráni (N). Vzdálenost mezi jednotlivými ukazateli je rovna $d = \frac{1}{N}$. Pozice prvního ukazatele je určena náhodně vygenerovaným číslem z intervalu $< 0, d >$.



Obrázek 4.3 Příklad stochastického univerzálního vzorkování nad 10 jedinci.

4.3.3. Místní selekce

Jedinci jsou rozděleni do oddělených prostředí nazývaných sousedství. Jedinci si mohou vyměňovat genetickou informaci pouze s jinými jedinci ze svého sousedství.

4.3.4. Turnaj

Pokud zvolíme tento typ selekce, populace bude rozdělena na několik skupin, přičemž počet skupin záleží na velikosti turnaje a celkovém počtu jedinců v populaci. Jedinci jsou do jednotlivých turnajových skupin vybírání zcela náhodně. Velikost turnaje je vstupní parametr této selekční metody a udává počet jedinců vybraných do jedné skupiny. Po rozdělení do skupin je z každé z nich vybrán ten jedinec, jehož úspěšnost (fitness) je nejvyšší. [4]

Velikost turnaje T může nabývat hodnot z intervalu od 2 po N, kde N je celkový počet jedinců v populaci.

Tabulka 4.1 Tabulka 4.1 Vztah mezi velikostí turnaje a intenzitou výběru.ukazuje vztah mezi velikostí turnajové skupiny a intenzitou selekce [4].

Velikost skupiny	1	2	3	5	10	30
Intenzita selekce	0	0,56	0,85	1,15	1,53	2,04

Tabulka 4.1 Vztah mezi velikostí turnaje a intenzitou výběru.

V publikaci [4] můžeme nalézt dopodrobna rozebranou analýzu výběru turnajem, pro účely tohoto textu jsem vybral pouze vztah popisující vazbu mezi velikostí turnaje (skupiny) a intenzitou výběru. Velikost skupiny (turnaje) je T a intenzita selekce pro T ($Int_{sel}(T)$) je dána vztahem.

$$Int_{sel}(T) \approx \sqrt{2\left(\ln(T) - \ln\left(\sqrt{4{,}14 \cdot \ln(T)}\right)\right)}$$

Rovnice 4.5

4.3.5. Elitářství

Elitářství není selekčním mechanizmem. Uvádím jej, protože je důležitým doplňkem výběru jedinců. Tento princip může v některých případech přispět k podstatnému zefektivnění průběhu algoritmu.

Podstatou elitářství je, že před aplikací některého z operátorů selekce je vybrán elitní jedinec, který má nejvyšší hodnotu fitness. Tento jedinec je přímo převeden do množiny rodičů, nepodléhá soutěži.

5. Metody ukončení algoritmu

Z podstaty algoritmů patřících do rodiny evolučních výpočetních technik vyplývá, že nemůžeme dopředu tušit, jak by měl vypadat žádaný výsledek. Proto je potřeba už na začátku vydefinovat kritéria, na jejichž základě bude běh algoritmu zastaven.

5.1. Počet generací

Pokud nemáme k dispozici žádné informace o tom, jak má řešení po skončení běhu algoritmu vypadat, je vhodné zvolit jako ukončující kritérium pevně stanovený počet generací. Při každém vytvoření nové generace populace je inkrementováno počítadlo, pokud dosáhne hodnoty maximálního počtu generací, algoritmus je ukončen. Jako řešení je vybrán nejlepší jedinec aktuální generace. Řešení může být ale i více, pokud je to požadováno. Potom jsou vybráni jako výsledek ti jedinci, kteří splňují minimální hodnotu fitness.

5.2. Evoluční čas

Před zahájením hledání řešení je zvolena maximální doba běhu algoritmu. Podobně jako u počtu generací je tato metoda ukončení vhodná v případě, že o požadovaném řešení je dopředu známo jen velmi málo informací. Nebo pokud je předpoklad, že k dosažení ideálního řešení bude zapotřebí příliš mnoho času.

5.3. Konvergence fitness

Tato metoda je vhodná pro případ, kdy máme dopředu dostupný odhad řešení. Potom můžeme jako ukončovací kritérium nastavit hodnotu funkce fitness. Jakmile se v populaci objeví jedinec, jehož ohodnocení se dostatečně přiblíží hodnotě fitness ukončovacího kritéria, běh algoritmu je zastaven a daný jedinec je vybrán jako řešení.

6. Praktické využití EA

První zmínky o výpočetní technice, využívající principů známých z biologie, nalézáme na konci 50. let 20. století, viz práce pánů Bremermanna, Friedberga, Boxe a dalších. Avšak převážně díky nedostatečné výpočetní kapacitě tehdejší počítačové techniky se tato oblast vědy nedostává do povědomí širší veřejnosti po tři nadcházející desetiletí. Až ke konci sedmdesátých let se zájem o tuto oblast obnovuje a jsou publikovány další práce. Asi nejznámější je práce *Adaptation in Natural and Artificial Systems (1975)*, Johna Hollanda.

Od sedmdesátých let minulého století však evoluční algoritmy a genetické algoritmy možná ještě více nabývají na důležitosti, jejich jedinečné vlastnosti se začínají uplatňovat ve stále větším počtu oborů.

Dnes můžeme nalézt využití evolučních algoritmů v mnoha oborech. Jako příklady uveďme optimalizační úlohy – rozmístění komunikací, telekomunikačních spojů, nakládání kontejnerů, učení chování robotů, struktury molekul, návrhy řešení problémů, které jsou obtížné na interpretaci – návrh uspořádání výrobních hal. EA jsou účinné při řešení různé plánovacích problémů, hlavně pokud jsou jednotlivé úlohy na sobě navzájem závislé – tvorba časových rozvrhů, predikce akciových trhů, apod. Evoluční algoritmy za posledních několik desítek let nalezly uplatnění napříč spektrem oborů.

Tento typ algoritmů velmi dobře funguje při řešení složitých problémů typu NP-úplné problémy, tzn. při řešení problémů, u kterých je výpočetní čas exponenciálně nebo faktoriálně závislý na počtu proměnných. Nemá však smysl používat je pro řešení relativně jednoduchých problémů nebo problémů, pro jejichž řešení existuje specializovaný algoritmus.

6.1. Využití genetického algoritmu při zpracování zvuku

6.1.1. Návrh oznamovacích melodií s využitím IGA

Skupina vědců z univerzity v Doshishe okolo Mitsunori Mikiho přišla s nápadem využít genetický algoritmus při návrhu oznamovacích melodií. Můžeme je definovat jako krátké a jednoduché melodie, které byly výrobcem záměrně implementovány do zařízení nebo prostředí, jako například zvuky varování u spotřebičů nebo vyzváněcí tón mobilního telefonu.

Účelem oznamovací melodie je sdělit uživateli specifickou informaci. Například že operace, kterou inicioval, proběhla úspěšně, nebo že obsluha na vlakovém nádraží bude v nejbližším okamžiku hlásit informace o přijíždějícím/odjíždějícím vlaku. Tyto melodie bývají nejčastěji spojovány s rozvojem osobních počítačů – např. varovné hlášky operačních systémů či uživatelských programů (potvrzení vysypání koše, dokončení kopírování souboru, upozornění na nadcházející schůzku, apod.). Avšak s rozvojem spotřební elektroniky se význam krátkých informativních zvuků značně rozšířil, mezi tyto melodie můžeme zařadit také vyzváněcí tóny mobilních telefonů ale i upozornění některých typů dětských elektronických chůviček. Neopomeňme zdůraznit i jiné neméně rozšířené oznamovací melodie, např. jak už byly zmíněny informativní znělky na vlakových nádražích, letištích nebo zvuková indikace na přechodu pro chodce, známé je určitě také pípání elektronického teploměru signalizující dokončení měření. Podobných zvuků je v dnešní době kolem nás celá řada. Většina výrobců dopodrobna zkoumala univerzálnost a efektivitu předání informace danou melodií, ale jen málokterý z nich se zabýval otázkou, zda je přívětivá pro ucho posluchače.

V dnešní době si uživatel může na internetu vybrat z nepřeberného množství různých melodií, hlášek, znělek, ale priority a záliby lidí se různí v závislosti na jejich vnímavosti, citlivosti. Jako doplněk by proto bylo vhodné, pokud by uživatel mohl volně a snadno vytvořit oznamovací zvuk dle vlastní chuti a priorit.

Většina lidí nedokáže složit melodii, protože nemají potřebné hudební vzdělání. Neznají zápis not, pomlk, natož předznamenání, stupnic, zákonů harmonie, atp. Vědcům z univerzity v Doshishe se podařilo dosáhnout řešení, které za pomocí interaktivního genetického algoritmu umožní i laikovi složit krátkou melodii vyjadřující potřebnou informaci přesně dle jeho vkusu.

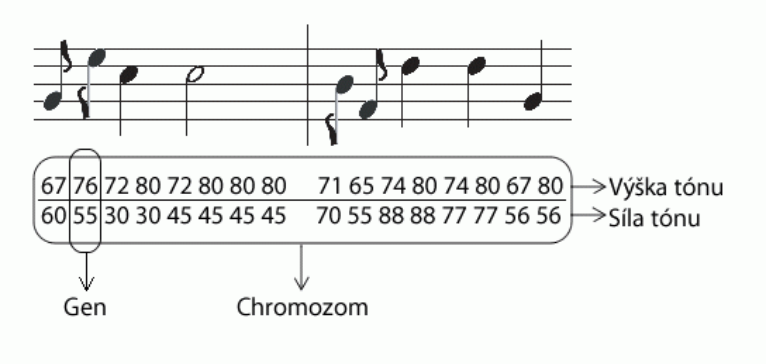
Interaktivní Genetický Algoritmus se oproti zde popisovanému genetickému algoritmu liší ve fázi ohodnocování. Nepoužívá předem definovanou fitness funkci, ale nabízí uživateli k výběru jedince a ten potom sám vybírá ty, kteří nejlépe odpovídají jeho vkusu.

Skupina došla po provedení několika zkoušek k závěru, že pro úspěšné použití této metody je zapotřebí zajistit stabilitu krátkých frázi složených z několika málo not, ne kratších než jeden takt. Jinými slovy, při křížení nesmí

dojít k roztržení fráze, proto nejkratší částí chromozomu (melodie) vstupující do procesu křížení je 1 takt. Ten je pak po dobu křížení dále nedělitelný.

Celková délka oznamovací melodie byla stanovena na 2 takty.

Pro reprezentaci tónů byly zvoleny hodnoty standardu SMF (Standard MIDI File) – výška tónů v rozmezí od 0 do 127, přičemž 60 je „velké C“, neboli C1. Každá změna o 1 znamená změnu noty o půltón. Pro generování oznamovacích melodií byla zvolena stupnice C Dur, tedy hodnoty tónů jsou omezeny na interval mezi 60 a 79. Hodnota 0, která nebyla použita pro notu, má zde přiřazen význam pauzy. Délka tónu v číselném zápise je vždy 1/8, proto je použita hodnota 80 jako prodloužení, např. sekvence 60, 80, 80, 80 bude znamenat půlovou notu C1. Sekvence je samozřejmě neúplná, ke každé hodnotě výšky tónu patří intenzita, síla jakou má být daná nota přehrána (viz Obrázek 6.1). Hodnoty 0 a 80 podléhají selekci stejně jako hodnoty reprezentující výšky tónů (60 – 79).



Obrázek 6.1 Reprezentace notového zápisu pro genetický algoritmus

Mitsunori Mikimu a spol. se podařilo vytvořit systém, který za pomoci interaktivního genetického algoritmu je schopen zkomponovat krátkou efektní melodii na základě interakce s uživatelem.

6.1.2. Přístup k audio steganografii na základě genetického algoritmu

Podobně jako všechny techniky skrývání informace v multimediálních datech musí audio steganografie splnit tři základní požadavky. Transparentnost při přehrání/zobrazení (v případě ideální transparentnosti posluchač nerozpozná rozdíl mezi původním audiem a audiem ukrývajícím zprávu), kapacita pro skrývání informace a robustnost (odolnost skrývané informace).

Odolnost skrývané informace rozdělujeme na dva typy, podle typu útoků, kterým musí čelit. Při prvním typu útoku se útočník snaží o odhalení skryté zprávy, při druhém je cílem útočníka zprávu zničit.

Aby byl útočník schopen skrytou informaci (zprávu) odhalit, či zničit, nejprve musí zjistit, kde se tato zpráva v audio souboru nachází. Jinými slovy musí odhalit, které bity původního souboru byly změněny. Vzhledem k tomu, že většina používaných technik steganografie používá pro uložení tajné informace nejméně důležité bity vzorků (LSB bity), je pro útočníka zjištění, kde se zpráva v audio souboru nachází, poměrně jednoduchou záležitostí. Při použití LSB bitů k uložení skryté informaci tuto informaci chrání pouze fakt, že potencionální útočník netuší, že se v zaslaném audio souboru vůbec nějaká skrytá informace nachází. V případě, že útočník získá podezření na existenci takové informace, zpráva je de facto odhalena. Použití genetického algoritmu má za cíl znesnadnit odhalení zprávy. Oproti používané technice nevyužívá všech LSB, anebo ukládá tajnou informaci i do jiných bitů vzorku (bitů z vyšších vrstev).

Vybírání vrstev a bitů pro uložení skrývané informace je kritické, náhodná selekce vzorků použitých pro vložení zavádí do výsledného audia nízko-úrovňový přídavný bílý gaussův šum (AWGN – Additive White Gaussian Noise). Z psycho akustiky je známo, že lidský sluchový aparát je vysoce citlivý na přítomnost AWGN. Tento fakt limituje počet bitů, které můžeme v původním audio souboru změnit, aniž by došlo k poslouchatelné změně. [8]

Vkládání nové informace do hlubších vrstev způsobuje větší chybu a snižuje tím kvalitu transparentnosti. Proto algoritmus, který vkládá data do hlubších vrstev by se měl zároveň pokoušet změnit další bity za účelem snížení chyby jím způsobené.

Genetický algoritmus je tomto případě vhodnou volbou pro redukci vznikající chyby. Jako chromozom bude vystupovat 1 vzorek (první rodiče budou vzorek původního audia a vzorek se změněným bitem pro vkládanou informaci), gen bude 1 bit vzorku a fitness funkce bude hodnotit výslednou chybu nad vzorkem. Při křížení a mutaci musí být dbáno na to, aby změněný bit nebyl ovlivněn (z operací křižení a mutace je vyjmuta vrstva, která nese skrývanou informaci). Jakmile se součet bitů vzorku (potomku) dostatečně přiblíží k součtu původního vzorku, potomek je vybrán jako řešení.

Příklad využití [8]:

- Bity vzorku: 00101111 = 47
- Cílová vrstva je 5, a bit zprávy má hodnotu 1
- Bez úpravy dalších bitů: 00111111 = 63 (chyba je 16)
- Po úpravě: 00110000 = 48 (chyba bude 1 pro vkládání 1 bitu)

- Bity vzorku: 00100111 = 39
- Cílové vrstvy jsou 4 a 5, bity zprávy mají hodnoty 1 a 1
- Bez úprav: 00111111 = 63 (chyba je 24)
- Po úpravě: 00011111 = 31 (chyba bude 8 pro vkládání 2 bitů)

Práce skupiny okolo PhD. M. Zamaniho [8] ukazuje velmi efektivní využití genetického algoritmu pro optimalizaci Steganografie nad audio daty, tedy skrývání tajné informace v audio datech.

7. Optimalizace distribuční trasy

Pro realizaci ukázkové aplikace, která má za úkol řešit problém optimalizace trasy, jsem zvolil prostředí programovacího jazyka PHP a nástrojů volně dostupné API od společnosti Google Inc ve spolupráci s databázovým úložištěm MySQL. Jedním z důvodů proč jsem zvolil právě toto prostředí, je velmi dobrá dostupnost pro uživatele a dostačující výpočetní kapacita.

7.1. Definice a rozebrání problému

Distribuční trasou považujeme jakoukoli trasu spojující tři a více geografických lokací.

Oproti známému optimalizačnímu problému Obchodního Cestujícího se může mírně lišit, zejména v tom že není podmínkou začínat a končit na tomtéž místě. Aplikace má za cíl najít nejoptimálnější trasu spojující všechny uživatelem zadané geografické body, bez ohledu na směr jízdy a pořadí.

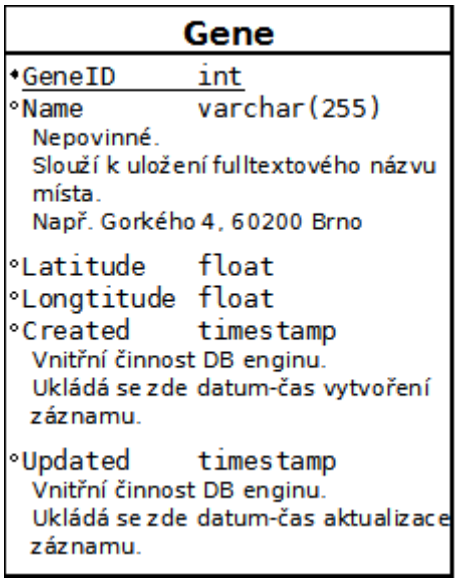
Většina optimalizačních řešení pro trasy volí jako hodnotící kritérium přímou vzdálenost mezi souřadnicemi. V reálném prostředí nelze cestovat přímočaře. Proto jsem zvolil Google Maps API, které umožňuje spočítat lomenou trasu. S velmi slibnou přesností tak kopíruje komunikace. Díky svému webovému rozhraní se může změnit ve vhodnou navigaci a zdroj dat pro optimalizační nástroj.

7.2. Vhodné rozvržení evolučního algoritmu

Evoluční algoritmus má cyklický průběh. Po vytvoření prvotní generace se dostáváme do cyklického opakování tří základních úkonů. Ohodnocení jedinců populace (generace). Selekce vhodných kandidátů pro předání informace další generaci, rodičů. A křížení, vznik nových jedinců.

7.2.1. Gen

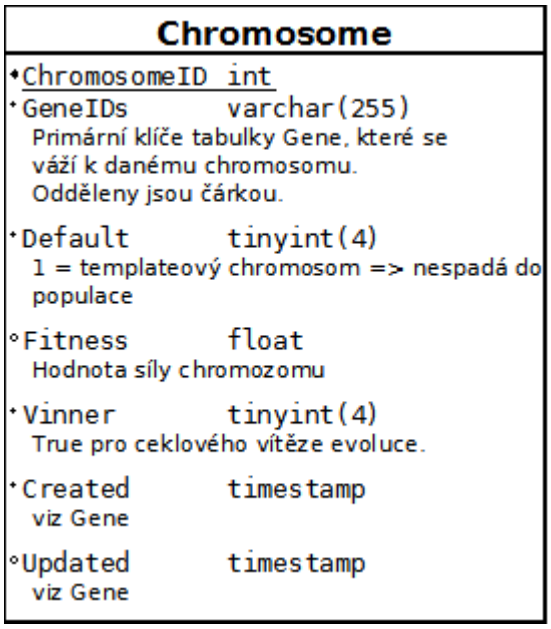
Základní stavební jednotka jedince. V tomto případě je tvořena informací o geografické poloze jednoho bodu na trase. Datová reprezentace je tvořena záznamem v databázové tabulce `gene`. Tabulka obsahuje sloupec primárního klíče `GeneID`, fulltextový název místa (adresa), pokud byla zadána `Name` a nejdůležitější dvojicí sloupců jsou `Longtitude` a `Latitude`, které nesou GPS otisk adresy bodu na trase.



Obrázek 7.1 Databázová tabulka gene

7.2.2. Chromozom

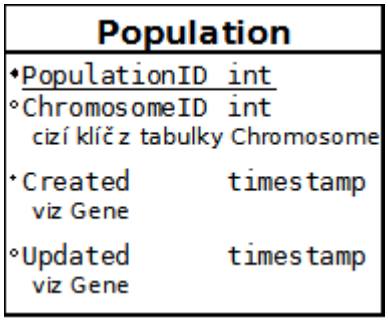
Nebo také Jedinec, představuje jednu „túru“ [10]. Jedná se o posloupnost genů. Datové úložiště pro jednotlivce je tabulka `chromosome`. Stejně jako gen obsahuje primární klíč `ChromosomeID`, druhý sloupec `GeneIDs` ukládá informaci o posloupnosti genů, která chromozom definuje. `GeneIDs` jsou čárkami odděleny identifikátory záznamů v tabulce `gene`. Do tabulky chromosome je uložen i template chromozomu – první vytvořený jedinec, ten se ovšem soutěže neúčastní. Jedinci, kteří se stanou součástí první generace, jsou vygenerování náhodným přeuspořádáním pozic `GeneIDs`. Template chromozomu má příznak `Default` nastaven na true. Sloupec Fitness je připraven pro uložení informace o síle daného jedince. `Vinner` je podobně jako `Default` příznak true / false, označuje jedince, který je vybrán při ukončení chodu algoritmu jako nejúspěšnější řešení.



Obrázek 7.2 Databázová tabulka chromosome

7.2.3. Populace

Populace je kolekce chromozomů reprezentující jednu generaci. Databázová tabulka `populace` je relací svázána s `chromosome`.



Obrázek 7.3 Databázová tabulka population

7.2.4. Rodiče

Po provedení selekce získáme z populace páry chromozomů, které slouží jako předloha pro operaci křížení, resp. mutace. Pro uchování těchto jedinců jsem zvolil využití ještě jedné db tabulky s názvem `parent`, obsahuje sloupce: primární klíč: `ParentID` [PK] a dva cizí klíče: `ChromosomeID` [FK], `PopulationID` [FK].

Nad rodiči je provedeno křížení a nově vzniklí jedinci jsou uloženi do tabulky `chromosome` spolu s relační vazbou v tabulce `population`, `PopulationID` je inkrementováno o 1.

7.3. Maps Google API – directions

Rozhraní Maps Google API umožňuje datovou komunikaci mezi službou google maps a vlastní aplikací. Data jsou transportována ve dvou možných podobách, XML a JSON. Veškerá komunikace probíhá prostřednictvím http dotazu a odpovědi.

Služba directions je zaměřena na vyhledání trasy z výchozího místa, do cílového místa, případně přes další body mezi těmito dvěmi. Pro účely optimalizace ji však můžeme použít, pokud získaná data správně použijeme.

7.3.1. Google API – Dotaz

Http dotaz služby Maps Google API sestává z pevné adresy a několika parametrů, které rozhodují o tom, jaká data můžeme v odpovědi čekat.

Pevná (neměnná) část dotazu:

- http://maps.googleapis.com/maps/api/directions/{encoding}?

Parametry:

- encoding: JSON / XML
- origin: výchozí místo, může být adresa nebo zeměpisná šířka / délka odděleny čárkou.
- Destination: cílové místo, podobně jako výchozí místo.
- waypoints: další body na trase, tento parametr je o něco složitější. Sestává z více adres, GPS souřadnic oddělených znakem „roura“ => | . Před vlastními adresami se nachází ještě jeden „podparametr“:
    - optimize:“true“ nebo “false“ – tento příznak dává službě vědět, zda si přejeme aby námi vložené „waypoints“ přerovnala tak aby výsledná trasa byla výhodnější. Pro účely této aplikace je trvale nastavena na false.
- sensor:“true“ nebo “false“, tento parametr říká službě, zda máme vlastní GPS zařízení se kterým by případně mohla komunikovat. Pro účely této aplikace zůstává nastavena na false.

  Hodnoty jsou parametrům přiřazovány následovně parametr=hodnota. Jednotlivé parametry jsou od sebe odděleny ampersandem „&“.

  Protože dotaz se v ničem neliší od jiné webové adresy, veškeré speciální znaky musí být nahrazeny bezpečnými zástupnými hodnotami. V PHP pro tento účel nalezneme funkci urlencode().

7.3.2. Google API – Odpověď

Pokud je dotaz sestaven správně, odpovědí bude soubor ve formátu JSON (.json) nebo XML (.xml). Zvolil jsem formát JSON, s pomocí funkcí `json_decode()` a `json_encode()` je práce s tímto formátem pro účely této aplikace efektivnější. PHP obsahuje funkci pro získání obsahu ze vzdáleného souboru: `file_get_content()`. Tato funkce vrátí obsah souboru jako string. Poté pomocí funkce `json_decode()` můžeme převést data z formátu JSON do nativního PHP formátu. Při použití parametru `$assoc = true`, je json převeden na asociativní pole.

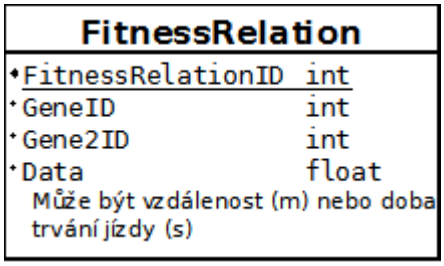
Odpověď obsahuje GPS souřadnice všech zadaných „zastávek“, vzdálenosti mezi jednotlivými body, a protože je to odpověď na vyhledání trasy obsahuje i spoustu pro nás doplňkových dat ohledně instrukcí podél cesty.

7.4. Fitness

Fitness neboli síla jedince, také schopnost prosadit se v boji o přežití mezi ostatními jedinci. Tato aplikace hodnotí sílu jedince nepřímo úměrně k celkové vzdálenosti mezi jednotlivými geny. Tedy Fitness je rovna součtu vzdáleností mezi geny.

7.4.1. Fitnessrelation

Při zpracování příchozích dat ze služby google maps přichází na řadu další databázová tabulka. `fitnessrelation` uchovává vztahy mezi jednotlivými geny (vzdálenost, případně čas, atd.). Abychom mohli hodnotit fitness jedince, musíme nejprve stanovit vzdálenosti mezi jednotlivými body trasy (geny). Tabulka `fitnessrelation` tyto informace ukládá do struktury: `FitnessRelationID` [PK], `GeneID` [FK], `Gene2ID` [FK], `Data`. Kombinace `GeneID`, `Gene2ID` a `Data` tak obsahuje potřebnou informaci.



Obrázek 7.4 Databázová tabulka fitnessrelation

Vztah mezi počtem genů a počtem jednotlivých párů je dán vztahem:

$$\mathrm{K(k,n)} = \binom{n}{k} = \frac{n!}{k!\,(n-k)!};\ \mathrm{k} = 2, \mathrm{n} = \text{počet } gen\text{ů } v\ chromozomu$$

Rovnice 7.1 Variace k-tic z n počtu prvků

$$K(2,10) = \frac{10!}{2!\,(10-2)!} = \frac{10!}{2 \cdot 8!} = 45$$

Rovnice 7.2 Příklad kominací 2 z 10

$$K(2,10) = \frac{20!}{2!\,(20-2)!} = \frac{20!}{2 \cdot 18!} = 190$$

Rovnice 7.3 Příklad kombinací 2 z 20

Tato operace je pravděpodobně nejnáročnější na čas, s rostoucím počtem genů v chromozomu (tedy zastávek na cestě) citelně roste počet výsledných kombinací. Pro každou kombinaci je potřeba sestavit nový dotaz pro Maps Google API a zpracovat odpověď. Zdržení je však pouze na začátku běhu, jakmile jsou kombinace uloženy v databázi, přístup k nim se značně urychlí.

7.5. Křížení

V sekci Operátor křížení jsem zmínil metodu křížení, při níž se vlastní genetická informace nevyměňuje mezi rodiči. Pouze se mění pořadí jednotlivých genů mezi sebou, na základě vzdálenosti k následníkovi.

7.5.1. Nevhodné metody

Metody křížení typu 1-bodové, k-bodové, uniformní, apod. by zavinily ztrátu citlivé informace. Pokud bychom rozdělili chromozomy popisující tury na dvě části a jednu zaměnili s protějším chromozomem, s největší pravděpodobností by nastala situace, že jeden chromozom by obsahoval dvakrát jednu část túry, a druhý tu opačnou taktéž dvakrát.

Pro příklad zvolíme jasnou situaci: chromozom1 = (1;2;3;4;5;6), chromozom2 = (6;5;4;3;2;1). Řekněme, že rozdělovat budeme v půli: chromozom1‘ = (1;2;3;3;2;1) a chromozom2‘ = (6;5;4;4;5;6).

7.5.2. Vhodné metody

První vhodnou je již zmíněná metoda bližšího souseda. Další vhodná metoda se zakládá na náhodném výběru úseku genu a záměně těch genu, které jsou v obou rodičích na stejných pozicích.

```
chromozom1 = (2,5,3,4,7,1,6);
chromozom2 = (5,2,3,4,1,6,7);
 potomek1 = (2,5,4,3,7,1,6);
 potomek2 = (5,2,4,3,1,6,7);
```

7.6. Ukončovací podmínka

Jako ukončovací podmínku jsem zvolil počet generací. Ale jako kontrolní hodnotu jsem přidal počítadlo stagnace vývoje hodnoty fitness. V případě, že jedna z podmínek je splněna (maximální počet generací, nebo přílišná stagnace) cyklus je ukončen a jako nejlepší jedinec je vybrán ten, který první dosáhl nejnižší hodnoty Fitness.

7.7. Shrnutí aplikace

Aplikace se skládá ze dvou základních vrstev. První se zabývá komunikací s databází, základními třídami, které zde hrají roli jsou DBObject a DBCollection. Propojení s funkcemi přímo komunikujícími s databází slouží třída Db, kterou odvodil od známého pluginu AdoDb wue a byl tak laskav, že ji zpřístupnil k volnému využití.

Druhá vrstva je zodpovědná za běh evolučního algoritmu. Obsahuje třídy obstarávající Geny, Chromozomy a Populace. Také třída FitnessRelation, která má na starost získávání a ukládání informací o vztazích mezi geny.

Řídící skripty jsou uloženy ve složce /scripts/. Hlavní spouštěcí skript je obsažen v souboru /scripts/run.php, odtud je řízena logika.

7.7.1. Průběh aplikace

Výpis první populace, kolekce chromozomů. Ve stavovém řádku je vypsána aktuální nejlepší Fitness.

První populace

ChromosomeID	GeneIDs	Fitness
11	3,4,1,2	175471
2	1,2,4,3	190479
6	2,1,3,4	202751
9	2,1,3,4	202751
5	2,3,4,1	350146
7	1,3,4,2	350148
8	2,4,3,1	350148
10	3,2,4,1	482127
3	3,2,4,1	482127
4	2,3,1,4	494399

Nejlepší Fitness chromozomu: **175471**
Nejlepší Fitness populace: **3280547**

Obrázek 7.5První populace

Následující screen zobrazuje 29. generaci. Ve stavovém řádku je vidět aktuální nejlepší Fitness chromozomu (175471) a stagnace Fitness Populace (110).

Populace #29

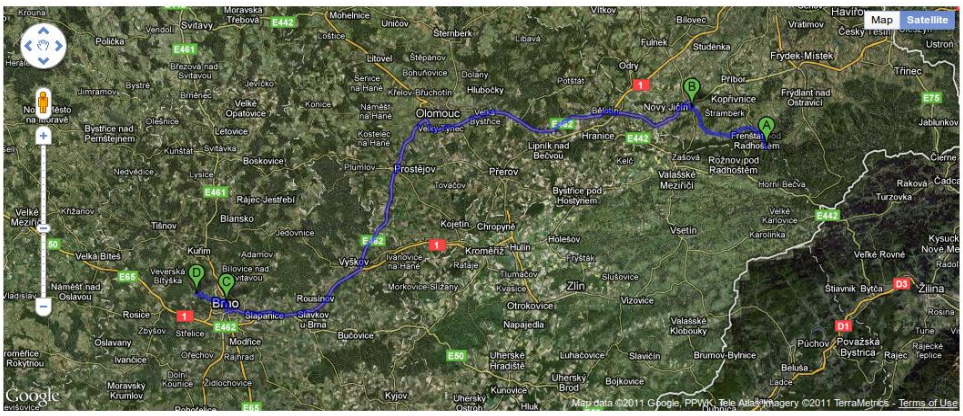
ChromosomeID	GeneIDs	Fitness
330	3,4,1,2	175471
332	3,4,1,2	175471
335	3,4,2,1	190479
338	2,1,3,4	202751
339	2,1,3,4	202751
333	2,1,3,4	202751
331	1,2,3,4	217757
336	3,1,2,4	334732
337	3,1,2,4	334732
334	2,3,4,1	350146

Nejlepší Fitness chromozomu: **175471**
Ukazatel stagnace: 110

Obrázek 7.6 Populace č.29

Závěrečné zobrazení nejlepšího chromozomu. Převedeno do rozhraní Google Maps JS, lze z mapou pracovat, případně i přetáhnutím doupravit trasu.

Google JS MAP



Obrázek 7.7 Výsledná mapa optimalizované trasy

Závěr

Tato práce se zabývá podstatou části rodiny výpočetních technik, které s úspěchem staví na principech známých z přírody a evoluce. V posledních čtyřech dekádách postupně získávaly tyto metody řešení hlavně optimalizačních problémů na váze. S přibývající výpočetní silou hardwaru, ať už specializovaného nebo určeného do osobních počítačů, vzrůstá obliba a počet aplikací pro tento typ algoritmu.

Práce rozebírá základní vlastnosti a principy Evolučních Algoritmů (EA). Jsou zde uvedeny definice stěžejních termínů z genetiky. Je nastíněn průběh a fungování evolučního optimalizačního algoritmu. Jsou objasněny možnosti operátorů křížení, mutace a výběru. V kapitole Praktické využití EA jsou nastíněny některé z mnoha možností aplikace EA v praxi spolu s uvedením konkrétních příkladů.

Praktická část práce se zaměřuje na možnost využití skriptovacího jazyku PHP, databázového úložiště MySQL spolu s rozhraním Maps Google API, pří řešení optimalizační úlohy. Zadání úlohy zní, optimalizace distribuční trasy. Věřím, že zvolená kombinace použitých technologií má perspektivu tuto úlohu s úspěchem řešit. Přináší zejména rozšíření možnosti kalkulovat s reálnou vzdáleností mezi body trasy, tak jak se po ní obchodník skutečně vydá. Oproti jiným řešením, počítajícím s přímočarým pohybem mezi body trasy.

V rámci navrhování a realizace aplikace jsem odkryl mnoho problémů, některé se mi podařilo vyřešit. Avšak ještě by bylo třeba více času, aby tato aplikace byla připravena k většímu využití. Při snaze o ucelený běh aplikace přes vícero generací jsem narazil na překážku synchronního běhu skriptu. Pokud běží skript v celku bez přerušování, skončí chybou indikující překročení maximální doby běhu skriptu. Při nastavení na 60s se nezdá být rozumné tuto dobu prodlužovat. Uživatel určitě nebude potěšen několika minutovým pohledem na předchozí webovou stránku nebo na obrázek vybízející k čekání.

Jako řešení se zdá vhodná asynchronní metoda spouštění skriptu, pomocí technologie AJAX. Tato technologie, propojující vlastnosti a funkce skriptovacího jazyku ze strany serveru i klienta, umožňuje spouštět jednotlivé úseky kódu nezávisle na stavu vykonání předchozího kódu. Tato vlastnost by byla zejména vhodná pro umožnění lepší interakce uživatele s možnostmi evolučního algoritmu. Zejména možnost změny parametrů (křížení, mutace, selekce) v průběhu chodu algoritmu.

Seznam použité literatury

[1] ZELINKA, Ivan, et al. Evoluční výpočetní techniky - principy a aplikace. [s.l.] : BEN, 2009. 536 s. ISBN 80-7300-218-3

[2] HYNEK, Josef. *Genetické algoritmy a genetické programování.*. [s.l.] : Grada, 2008. 200 s. ISBN 978-80-247-2695-3.

[3] BAKER, James. E. Reducing bias and inefficiency in the selection algorithm. *Proceeding : Proceedings of the Second International Conference on Genetic algorithms and their application*. 1987, s. 14 - 21. ISSN 0-8058-0158-8.

[4] POHLHEIM, Hartmut. *GEATbx : Algorithms* [online]. 3.80. Prosinec 2006 [cit. 2010-11-20]. Evolutionary Algorithms. Dostupné z WWW: <http://www.geatbx.com>.

[5] GOLDBERG, David E.; DEB, Kalyanmoy. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms*. 1991, 91-53076, s. 27. Dostupný také z WWW: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.9494&rep=rep1&type=pdf>. ISSN 1-55860-170-8

[6] BLICKLE, Thobias; THIELE, Lothar. *A Comparison of Selection Schemes used in Genetic Algorithms*. 2. Edition. Zurich : TIK, ETH, 1996. 77 s. Dostupné z WWW: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.9494&rep=rep1&type=pdf>.

[7] MIKI, Mitsunori, et al. Design of Sign Sounds using an Interactive Genetic Algorithm. *Systems, Man and Cybernetics, 2006. SMC* . 8.-11. Říjen 2006, 9772573, s. 4. Dostupný také z WWW: <http://www-im.dwc.doshisha.ac.jp/~wake-lab/member/swake/pdf/SS-MelodyGA-SMC06.pdf>. ISSN 1-4244-0099-6

[8] ZAMANI, Mazdak, et al. A Genetic-Algorithm-Based Approach for Audio Steganography. *World Academy of Science : Engineering and Technology*. 2009, 54, s. 4. Dostupný také z WWW: <http://www.waset.org/journals/waset/v54/v54-63.pdf>.

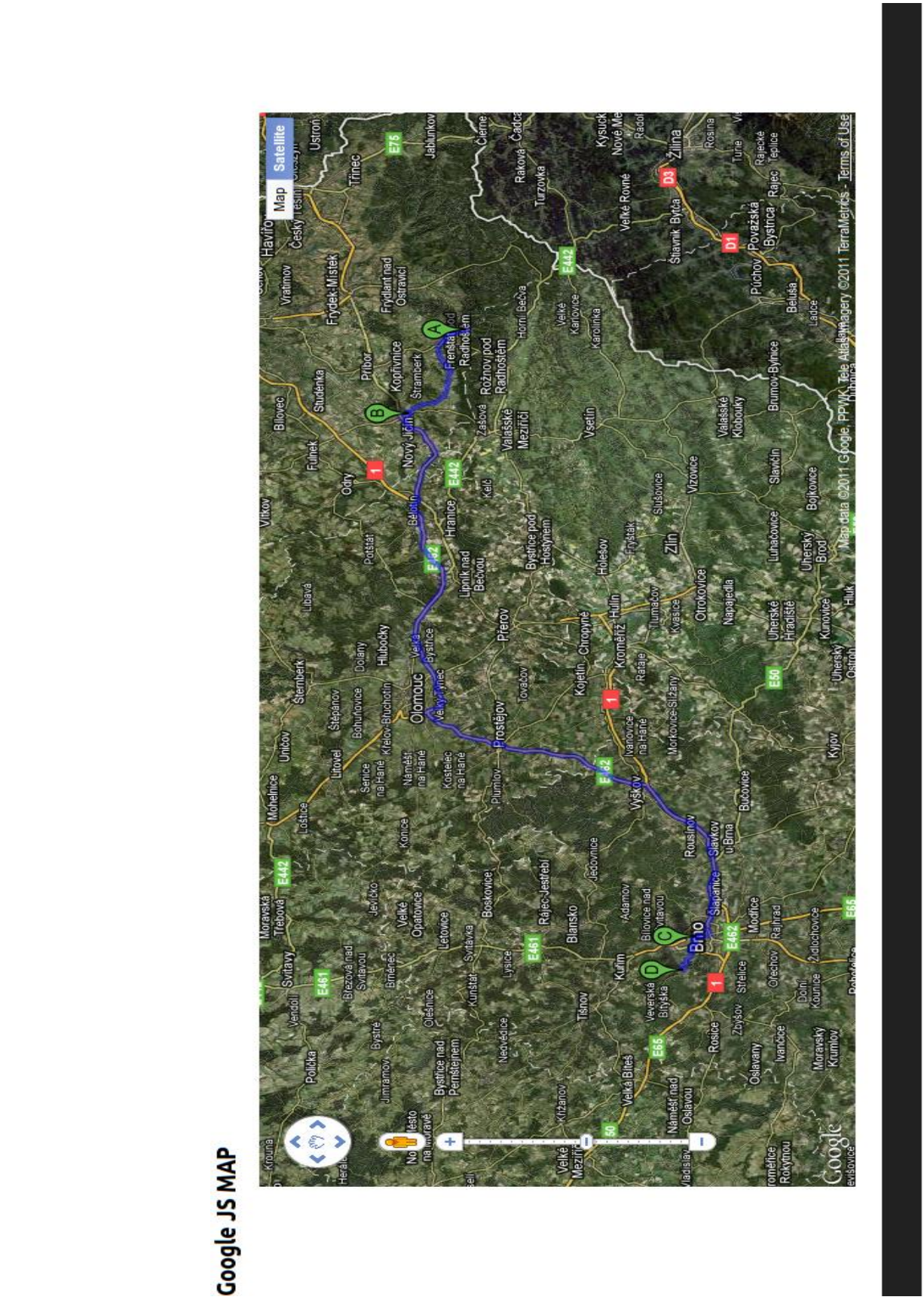
[9] DASGUPTA, D.; MICHALEWITZ, Z. . *Evolutionary algorithms in engineering applications*. Berlin Heidelberg : Springer-Verlag, 1997. 547 s. ISBN 3-540-62021-4.

[10] STÁREK, Onřej. *TSP : Programování* [online]. 2005 [cit. 2011-05-25]. Řešení problému obchodního cestujícího. Dostupné z WWW: <http://stareko.webzdarma.cz/skola/tsp/zprava.php>.

Seznam zkratek, veličin a symbolů

EA	–	Evoluční Algoritmus
GA	–	Genetický Algoritmus
API	–	Application Program Interface (programové rozhraní k aplikaci)
db	–	databáze, databázový
PK	–	Primární Klíč (Primary Key)
FK	–	Cizí Klíč (Foreign Key)
XML	–	Extensible Markup Language
JSON	–	JavaScript Object Notation
http	–	HyperText Transfer Protocol

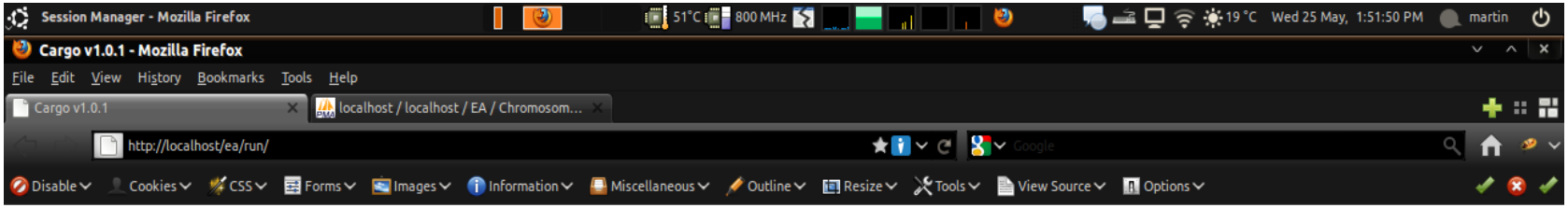
Přílohy


Google JS MAP
Map
Satellite
Brno
Olomouc
Zlín
Žilina
Map data ©2011 Google, PPWK, Tele Atlas
Imagery ©2011 TerraMetrics - Terms of Use
Google

Populace #29

ChromosomeID	GeneIDs	Fitness
330	3,4,1,2	175471
332	3,4,1,2	175471
335	3,4,2,1	190479
338	2,1,3,4	202751
339	2,1,3,4	202751
333	2,1,3,4	202751
331	1,2,3,4	217757
336	3,1,2,4	334732
337	3,1,2,4	334732
334	2,3,4,1	350146

Nejlepší Fitness chromozomu: **175471**
Ukazatel stagnace: 110



První populace

ChromosomeID	GeneIDs	Fitness
11	3,4,1,2	175471
2	1,2,4,3	190479
6	2,1,3,4	202751
9	2,1,3,4	202751
5	2,3,4,1	350146
7	1,3,4,2	350148
8	2,4,3,1	350148
10	3,2,4,1	482127
3	3,2,4,1	482127
4	2,3,1,4	494399

Nejlepší Fitness chromozomu: **175471**
Nejlepší Fitness populace: **3280547**

Spuštění kódu:

1. Pro spuštění kódu je třeba mít instalován PHP engin (instalovat lze např. v rámci balíku XAMPP - http://www.apachefriends.org/en/xampp.html).
2. Uložte složku se zdrojovými kódy na místo, do kterého máte právo zapisovat (alespoň složka /design/smarty/ musí mít nastavena práva pro zápis).
3. Otevřete soubor /include/baseconfig.inc.php a upravte položky $config[‘baseurl’] a $config[‘basepath’] dle nastavení vašeho serveru a umístení zdrojových kódů.
4. Naimportujte databázi do vašeho mysql rozhraní. Záloha struktury db je umístěna ve složce /_local/_sql/. Pokud jste si db pojmenovali jinak než EA, tak změňte příslušnou hodnotu v baseconfig.inc.php.