



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ  
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER SYSTEMS

# VYUŽITÍ REGRESNÍCH METOD PRO PREDIKCI DOPRAVY

REGRESSION METHODS IN TRAFFIC PREDICTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. TOMÁŠ VAŇÁK

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. JIŘÍ PETRLÍK

BRNO 2013

## **Abstrakt**

Diplomová práce se zabývá možnostmi predikce dopravní situace na makroskopické úrovni s využitím údajů naměřených pomocí dopravních senzorů. Těmito senzory mohou být indukční smyčky, radarové detektory nebo kamery. Práce se zaměřuje na problematiku predikce dojezdových dob automobilů. V rámci diplomové práce byla navržena a implementována metoda dojezdových dob. Navržená metoda byla otestována pomocí dat z reálného provozu. Prvním cílem práce bude seznámení s metodami predikce, které budou využívány. Hlavním cílem práce je využít získaných znalostí k navržení a implementaci aplikace, která bude predikovat požadované dopravní veličiny.

## **Abstract**

Master thesis deals with possibilities of predicting traffic situation on the macroscopic level using data, that were recorded using traffic sensors. This sensors could be loop detectors, radar detectors or cameras. The main problem discussed in this thesis is the travel time of cars. A method for travel time prediction was designed and implemented as a part of this thesis. Data from real traffic were used to test the designed method. The first objective of this thesis is to become familiar with the prediction methods that will be used. The main objective is to use the acquired knowledge to design and to implement an application that will predict required traffic variables.

## **Klíčová slova**

Dojezdové doby, kartézské genetické programování, symbolická regrese, genetické programování

## **Keywords**

Travel times, cartesian genetic programming, symbolic regression, genetic programming

## **Citace**

Tomáš Vaňák: Využití regresních metod pro predikci dopravy, diplomová práce, Brno, FIT VUT v Brně, 2013

# Využití regresních metod pro predikci dopravy

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jiřího Petrlíka a že jsem uvedl všechny literární prameny a publikace, ze kterých jsem čerpal.

.....  
Tomáš Vaňák  
21. května 2013

## Poděkování

Děkuji tímto Ing. Jiřímu Petrlíkovi za jeho podporu a odbornou pomoc při tvorbě této práce.

© Tomáš Vaňák, 2013.

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

<b>1 Úvod</b>	<b>3</b>
<b>2 Genetické algoritmy a genetické programování</b>	<b>4</b>
2.1 Základní pojmy . . . . .	4
2.2 Genetický algoritmus . . . . .	4
2.3 Genetické programování . . . . .	8
<b>3 Kartézské genetické programování</b>	<b>11</b>
3.1 Reprezentace kandidátního řešení v CGP . . . . .	11
3.2 Vytvoření jedince z genotypu . . . . .	12
3.3 Genetické operátory . . . . .	13
3.4 Speciální varianty CGP . . . . .	14
3.4.1 Cyklické CGP . . . . .	14
3.4.2 Více-chromosomové CGP . . . . .	14
3.4.3 Sebe modifikující kartézské genetické programování . . . . .	15
<b>4 Symbolická regrese</b>	<b>16</b>
4.1 Příklad symbolické regrese . . . . .	16
<b>5 Predikce dopravy</b>	<b>19</b>
5.1 Stupně provozu . . . . .	19
5.2 Senzory . . . . .	19
5.3 Predikce dojezdových dob . . . . .	20
5.4 Predikce dojezdových dob s využitím soft-computingových metod . . . . .	22
5.4.1 Support vector regression . . . . .	22
5.4.2 Neuronové sítě . . . . .	23
<b>6 Metoda predikce</b>	<b>25</b>
<b>7 Implementace</b>	<b>27</b>
<b>8 Experimentální vyhodnocení</b>	<b>29</b>
8.1 Popis testovacích dat . . . . .	29
8.1.1 Údaje ze senzorů . . . . .	29
8.1.2 Dojezdové doby . . . . .	30
8.1.3 Úprava dat pro predikci . . . . .	31
8.2 Nastavení evoluce . . . . .	31
8.3 Mutace . . . . .	33
8.4 Velikost populace . . . . .	36

8.5	Rozměry CGP mřížky . . . . .	40
8.6	Použité funkce . . . . .	47
8.7	Závěrečné experimenty . . . . .	49
<b>9</b>	<b>Závěr</b>	<b>60</b>

# Kapitola 1

## Úvod

Efektivní řízení dopravy je problém, kterým se musí v dnešní době zabývat téměř všechna větší města. Pro úspěšné řízení dopravy je potřeba vhodným způsobem predikovat, jak se bude doprava vyvíjet.

Některá města byla založena už před stovkami i tisíci lety. V dřívějších dobách byla dopravní zátěž oproti dnešním dobám výrazně nižší. Města byla postavena podle tehdejších dopravních potřeb. Postupem času ale hustota dopravy narůstala a dopravní síť měst ji v dnešních dobách často nezvládá pojmout.

Jednou z možností jak řešit problémy s dopravou jsou stavební úpravy. Vybudováním nových silničních obchvatů kolem měst, mimoúrovňových křižovatek a podobných silničních prvků, můžeme výrazně pomoci odlehčit přetížené silnice ve městech. Tato řešení bývají poměrně účinná, bohužel ale i extrémně finančně náročná. Mnoho měst má navíc i velkou historickou hodnotu a jejich přestavba proto nepřichází v úvahu.

Další možností je vylepšit řízení dopravního provozu. Obvykle se to provádí pomocí světelných signálů a signalizačních zařízení na křižovatkách městských komunikací. Pro zvýšení efektivity tohoto způsobu řízení dopravy je potřeba spolehlivě zjišťovat aktuální dopravní situaci v důležitých místech dopravy. Je tedy například potřeba znát situaci na všech ramenech křižovatky, aby bylo minimalizováno zdržení řidiče při průjezdu křižovatkou. V ideálním případě by měl mít řidič neustále „zelenou“. Pomocí správné predikce dopravy je možné dopravu ve městě efektivně řídit.

Tato práce se zaměřuje na predikci dojezdových dob automobilů. V rámci diplomové práce byla navržena metoda pro predikci dojezdových dob založená na kartézském genetickém programování.

Text práce je rozdělen do deseti kapitol. Ve druhé kapitole jsou popsány základy genetických algoritmů a genetického programování. Třetí kapitola popisuje kartézské genetické programování a jeho varianty. Čtvrtá kapitola se zabývá symbolickou regresí a jejím řešením za pomoci kartézského genetického programování. Pátá kapitola popisuje problematiku predikce dopravy se zaměřením na dojezdové doby automobilů. V šesté kapitole je popsáno, jak byla použita dopravní data získávána v reálném provozu. Sedmá a osmá kapitola popisují, jakým způsobem byla implementována aplikace pro predikci dojezdových dob. V deváté kapitole jsou popsány provedené experimenty nad testovacími daty. Desátá kapitola obsahuje závěr práce.

## Kapitola 2

# Genetické algoritmy a genetické programování

Tato práce se zabývá predikcí dojezdových dob pomocí kartézského genetického programování, jež je speciálním případem genetického programování. Genetické programování patří do široké skupiny evolučních algoritmů. Tato kapitola obsahuje stručný popis základních principů evolučních algoritmů a pojmů, které se v této oblasti používají.

### 2.1 Základní pojmy

Evoluční algoritmy jsou metody náhodného heuristického prohledávání, inspirované principy přírodní evoluce, speciálně Darwinovou teorií přírodního výběru. Protože se jedná o inspiraci ve světě biologie, využívá se v evolučních algoritmech mnoho biologických metafor:

**Gen:** Je to základní stavební jednotka chromozomu. Nabývá různých hodnot předem definovaného typu (reálná čísla, celá čísla, ...), kterým se říká alely.

**Chromozom:** Jiným názvem taky jedinec (individuum). Reprezentuje konkrétní řešení (jedince). Obvykle je tvořen jako lineární pole genů.

**Genotyp:** Představuje kódovaný tvar řešení.

**Fenotyp:** Konkrétní řešení sestavené podle chromozomu (např. cesta v grafu).

**Populace:** Množina kandidátních řešení. Obsahuje konstantní počet jedinců (chromozomů).

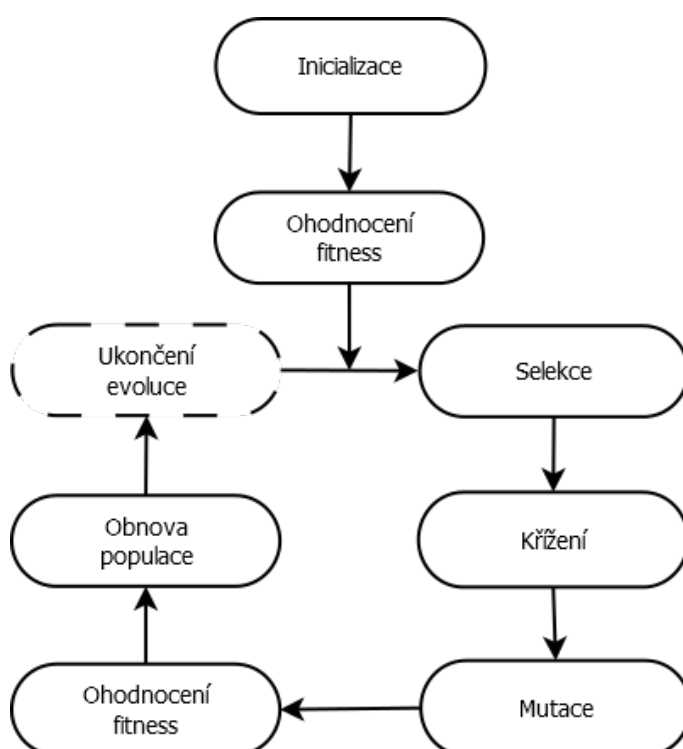
**Fitness funkce:** Vyjadřuje zdatnost (kvalitu řešení) jedince. Je zvykem ohodnocovat lepší jedince vyššími číselnými hodnotami. [23]

### 2.2 Genetický algoritmus

Průkopníkem v této oblasti byl John Holland, který studoval elementární procesy v populacích, jež jsou z hlediska evoluce nepostradatelné. Na základě těchto výzkumů navrhl genetický algoritmus jako abstrakci příslušných biologických procesů. Populace se postupně s každou generací vyvíjí. Jedinci v každé generaci vznikají z nejlepších jedinců předchozí

generace. Tento proces pokračuje, dokud ho nepřerušíme. Stejně jako v přírodě i při této evoluci může trvat mnoho generací, než je vytvořen dostatečně dobrý jedinec. V evolučních algoritmech je jedinec obvykle reprezentován vektorem diskrétních hodnot. Nový jedinec pak vzniká kombinací částí rodičovských vektorů.

Každý evoluční algoritmus začíná s množinou náhodných kandidátních řešení. Tuto množinu nazýváme počáteční populací. Jednotlivé iterace algoritmu se nazývají generace. V každé generaci jsou vytvořena nová řešení. Každé řešení je jeden jedinec populace. Jedinci jsou odvozeni ze současné populace pomocí genetických operátorů mutace a křížení. Následně se z této velké množiny rodičů a potomků vybere skupina jedinců, kteří se budou dále reprodukovat. Tento výběr se nazývá selekce. Neustálým vybíráním dobrých jedinců pro reprodukci a následně vytvoření nových řešení se postupně zvyšuje schopnost jedinců přizpůsobit se danému problému. Stejně jako v přírodě je umožněno účastnit se reprodukce jen silným jedincům. Průběh evolučního algoritmu je znázorněn na obrázku 2.1. [13]



Obrázek 2.1: Průběh genetických algoritmů

V prvním kroku genetického algoritmu jsou vytvořeni jedinci počáteční populace. Tvorba jedinců pro počáteční populaci může probíhat čistě náhodně anebo na základě našich znalostí řešeného problému, což může výrazně urychlit průběh evoluce. Hrozí ale, že vybereme jen určitou podmnožinu populace, zahrnující jen část našeho problému a díky tomu nalezneme evoluci pouze lokální maximum. Ohodnocení populace spočívá ve vypočtu hodnoty fitness funkce každého jedince. Dalším krokem je selekce pro následné křížení a mutaci. Výběr může probíhat mezi dvěma jedinci nebo mezi skupinami jedinců. Jsou upřednostňováni jedinci s vyšší fitness funkcí. Rozlišujeme mezi čtyřmi základními typy výběru:

Ruletový výběr: Fitness funkce jedinců se přepočítají tak, aby byly z intervalu (0, 1), jejich součet dal dohromady 1, ale poměr fitness jednotlivých jedinců zůstal stále stejný. Nová fitness určuje nyní velikost výseče na úsečce o velikosti 1 (pomyslné ruletě). Náhodně vygenerované číslo reprezentuje výběr jedince, na kterého ukázala ruleta. Čím má jedinec vyšší fitness, tím je pravděpodobnější, že bude vybrán. Jeden jedinec může být vybrán i vícekrát. Pokud je v populaci  $N > 0$  jedinců a předpokládáme fitness funkci každého jedince větší než 0, tedy  $f_i \geq 0$  (pro  $i = 1, \dots, N$ ), potom velikost výseče odpovídající  $i$ -tému jedinci je dána vztahem 2.1.

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j} \quad i \in 1, \dots, N \quad (2.1)$$

Turnajový výběr: Z populace je náhodně vybráno několik jedinců (obvykle 2). Je porovnána jejich fitness a do reprodukce postoupí ten lepší z nich.

Pravděpodobnostní výběr: Využívá kombinace předchozích dvou přístupů. Nejprve je ruletou vybráno několik jedinců (obvykle 2) a pak mezi nimi proběhne turnaj.

Uspořádaný výběr: Nejprve jsou jedinci v populaci uspořádáni podle hodnoty fitness funkce, selekce se pak provádí na základě pravděpodobnosti, která je úměrná pořadí jedince v tomto uspořádání. [23]

Na vybrané jedince jsou následně aplikovány genetické operátory křížení a mutace. Existuje několik základních typů křížení:

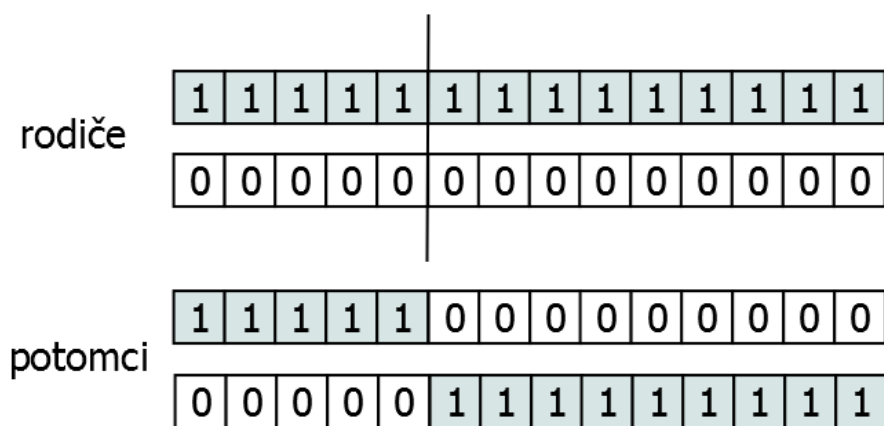
Jednobodové křížení: Vstupem jsou 2 jedinci z rodičovské populace a výstupem 2 jejich potomci. Nejprve je vygenerováno náhodné číslo, které udává bod křížení. U potomků se následně prohodí odpovídající si části chromozomů podle bodu křížení. Princip jednobodového křížení je znázorněn na obrázku 2.2.

Vícebodové křížení: Vstupem jsou 2 jedinci z rodičovské populace a výstupem 2 jejich potomci. Nejprve je vygenerováno  $k$  náhodných bodů křížení. U potomků se potom střídavě jedna část rodičovských chromozomů ponechá a druhá prohodí. Princip vícebodového křížení je znázorněn pomocí obrázku 2.3.

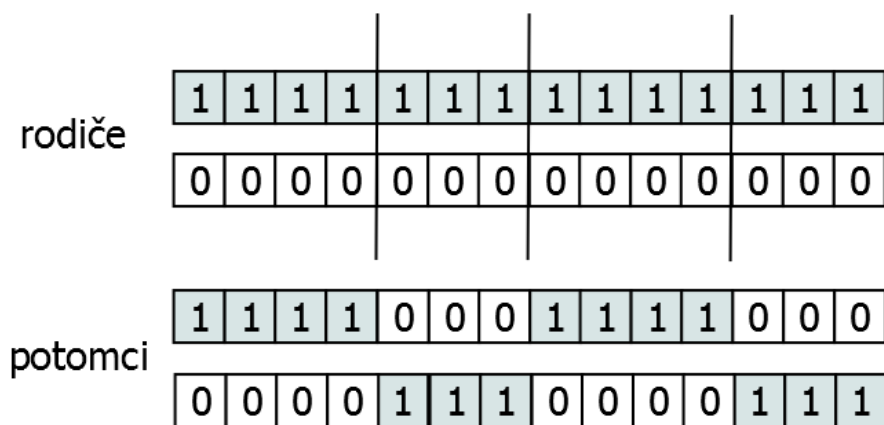
Uniformní křížení: Vstupem jsou 2 jedinci a výstupem 1 jejich potomek. Nejprve je náhodně vygenerována bitová maska stejné délky, jako jsou dlouhé genotypy rodičů. Jednotlivé hodnoty masky určují, ze kterého rodiče bude potomek obsahovat daný gen. Pokud je v masce jednička, obsahuje potomek na dané pozici gen z prvního rodiče, v opačném případě z druhého rodiče. Princip uniformního křížení je znázorněn na obrázku 2.4. [9] [21]

Při mutaci dojde u binárního zakódování ke změně hodnoty náhodně vybraného genu. Ten se náhodně nahradí za jiný (obrázek 2.5). Pro nové jedince musí být opět vypočítána hodnota jejich fitness.

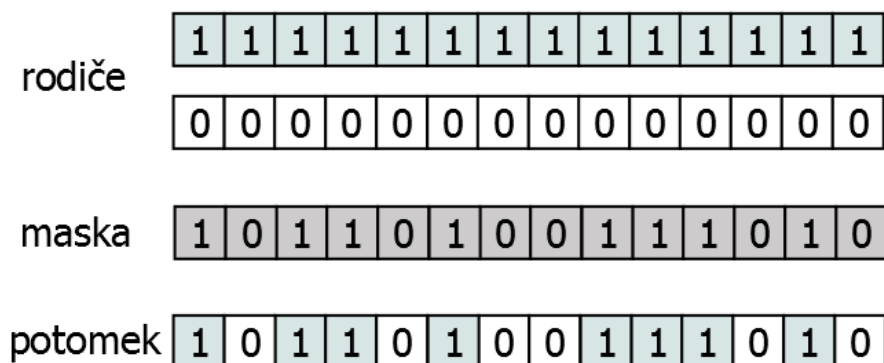
Ze staré populace rodičů a nové populace potomků je nutné vytvořit novou populaci. Pro výběr nové populace se využívá buď částečná nebo úplná obnova. Při úplné obnově populace rodičů vymírá a je kompletně nahrazena jejich potomky. Při částečné obnově část potomků



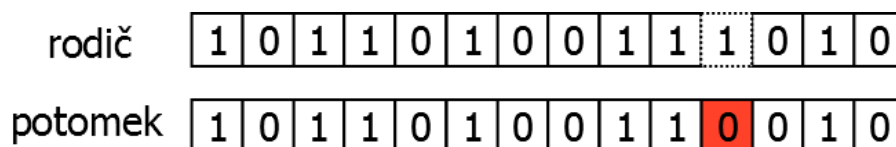
Obrázek 2.2: Jednobodové křížení



Obrázek 2.3: Vícebodové křížení



Obrázek 2.4: Uniformní křížení



Obrázek 2.5: Operátor mutace

nahradí část jedinců z populace rodičů. Existuje několik způsobů výběru jedinců. Jedním z nich je turnaj. Dalším způsobem může být nahrazení všech jedinců kromě několika málo nejsilnějších jedinců z původní populace. Tento způsob se nazývá elitismus. Posledním používaným způsobem je nahrazování jedinců s podobným genotypem. [23]

Po obnově populace dochází ke kontrole ukončujících podmínek evoluce. Pokud jsou podmínky splněny, je evoluce ukončena. Ukončovací podmínka může být definována jedním z těchto způsobů, případně jejich kombinací:

1. Byl překročen povolený počet generací.
2. Byl překročen časový limit pro evoluci.
3. Bylo nalezeno dostatečně kvalitní řešení.
4. Po provedení určitého počtu generací je fitness hodnota jedince neměnná.

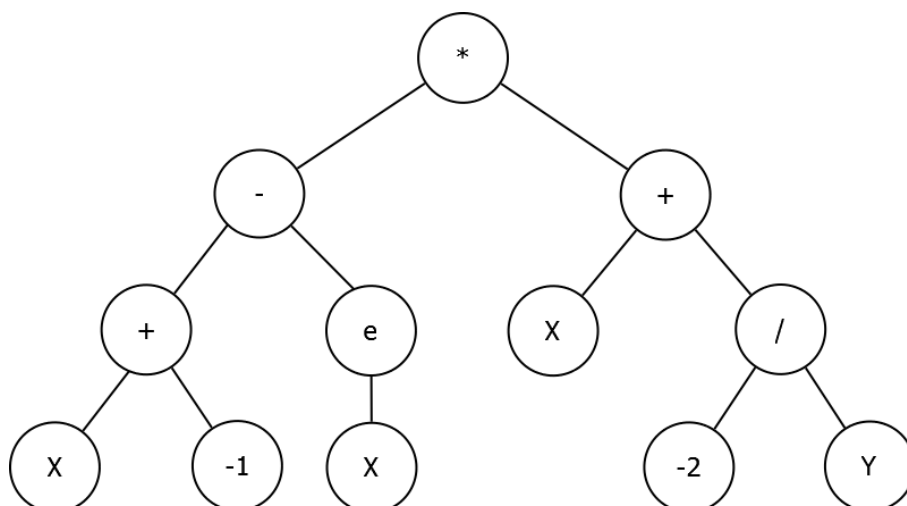
Pokud má být již evoluce ukončena, je jako výsledek evoluce vrácen nejlepší jedinec. Pokud nebyla žádná z podmínek ukončení evoluce dosažena, pokračuje se krokem selekce jedinců z populace.

## 2.3 Genetické programování

Genetické programování je evoluční výpočetní technika, která dovoluje počítačům řešit problémy, aniž by na ně byly explicitně naprogramovány. Genetické programování (GP) je rozšířením genetických algoritmů navržených Hollandem. Vlastní algoritmus je do značné míry podobný genetickému algoritmu. Jednou z odlišností je, že genetické programování často pracuje se spustitelnými strukturami. Často se jedná o programy. Na rozdíl od genetických algoritmů, kde jsou jedinci reprezentováni jako lineární pole bitů, jsou jedinci v genetickém programování reprezentováni ve formě syntaktických stromů. Výraz  $(x-1-e^x)*(x+\frac{-2}{y})$  je možné reprezentovat například ve formě stromu, který je znázorněn na obrázku 2.6. Uzly stromu reprezentují operandy. Listy obsahují proměnné a konstanty. V pokročilejších formách genetického programování mohou být programy složeny z více komponent (např. podprogramů). V takovém případě je GP reprezentováno množinou stromů, kde každá komponenta je reprezentována jedním stromem. Všechny tyto stromy jsou pak seskupeny pod speciálním kořenovým uzlem. [13] [29]

Při vytváření počáteční populace se může postupovat několika přístupy:

1. **Full** - v tomto případě je definována maximální možná výška stromu. Jakmile je dosaženo zadané výšky, je generování stromu ukončeno.
2. **Grow** - generují se stromy zcela náhodně. Když je vybrán ukončující terminál, je generování stromu ukončeno. V opačném případě generování stromu pokračuje.

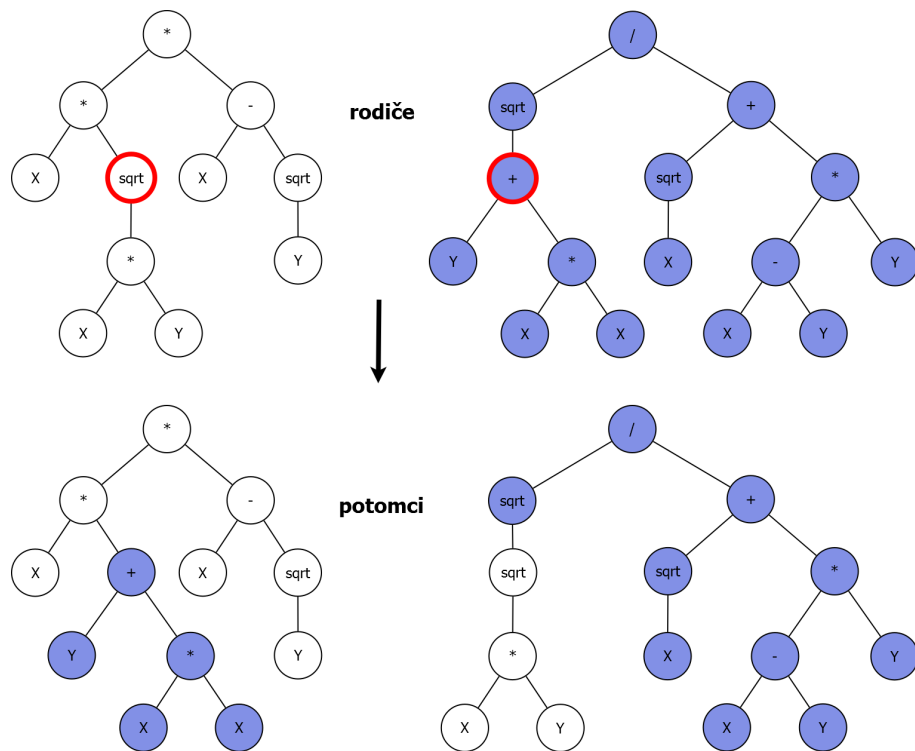


Obrázek 2.6: Strom reprezentující výraz  $(x - 1 - e^x) * (x + \frac{-2}{y})$

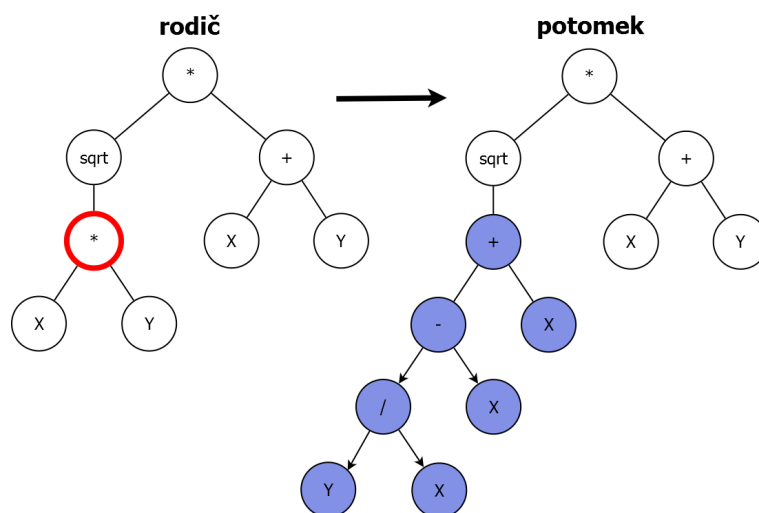
3. Ramped Half-and-half - je definována maximální výška stromu. Mohou být vygenerovány libovolné stromy menší nebo rovny maximální možné výšce. [8]

Další odlišností jsou genetické operátory pracující nad spustitelnými strukturami. Využívají se jednak tradiční operátory mutace a křížení, dále se však využívá pokročilých operátorů umožňujících například evolučně vytvářet podprogramy v rámci vyvíjených programů. Protože jsou jedinci v GP reprezentováni obvykle jako stromy, musí být patřičně upraveny i genetické operátory. Při křížení dvou jedinců se u každého jedince zvolí náhodně nelistový uzel. Ten bude představovat bod křížení. Podstromy vymezené tímto bodem křížení jsou následně u obou jedinců prohozeny. Princip činnosti operátoru křížení pro stromové struktury je znázorněn na obrázku 2.7. Genetický operátor mutace pracuje opět jen s jedním jedincem. Mutace nad stromem se provádí náhodným výběrem jednoho uzlu stromu. Tento uzel spolu s jeho podstromem je odstraněn a na jeho místo je vygenerován nový podstrom. Princip činnosti operátoru mutace pro stromové struktury je znázorněn na obrázku 2.8. Pro určení hodnoty fitness funkce je proveden kód kandidátního programu pro definovanou množinu vstupů.

Jedním z problémů genetického programování jsou introny. Jde o syntakticky správný kód, který však nemá žádný praktický význam. Může to být například operace typu  $a = a + 0$ . Při evoluci často dochází k narůstajícímu počtu intronů. V anglické terminologii je tento jev označován jako „bloat“. Obecně převládá mínění, že introny mají neblahý vliv na evoluci, protože jejich největší projev spočívá ve zpomalování vyhodnocování programů a tudíž k pomalejší evoluci. Existují však i názory, že introny chrání užitečný kód před destruktivními vlivy genetických operátorů. [25]



Obrázek 2.7: Princip činnosti operátoru křížení používaném v genetickém programování



Obrázek 2.8: Princip činnosti operátoru mutace používaném v genetickém programování

## Kapitola 3

# Kartézské genetické programování

Poprvé bylo kartézské genetické programování (CGP) představeno Julianem Millerem a Peterem Thomsonem v roce 1997 [17]. Na rozdíl od genetického programování je jedinec v CGP reprezentován jako acyklický orientovaný graf. Jednotlivé uzly jsou uspořádány ve 2D mřížce a představují jednoduché funkce. Vlastní průběh kartézského genetického programování je velmi podobný jednomu speciálnímu případu strategie  $(\mu + \lambda)$  a to strategii  $(1 + \lambda)$ . U strategie  $(\mu + \lambda)$  obsahuje populace  $\mu$  jedinců. Z těchto jedinců se pomocí genetických operátorů vygeneruje  $\lambda$  potomků. Nová populace tedy obsahuje  $\mu + \lambda$  jedinců. Z nich je ale selekcí vybráno do další populace jen  $\mu$  jedinců. [14] Průběh kartézského genetického programování je nastíněn pseudokódem 3.1.

```
1) nejlepsi = populace.vyberNejlepsihoJedince();
2) if (dosaeno konce evoluce) return nejlepsi;
3) populace.clear();
4) populace.insert(nejlepsi);
5) while (populace neni kompletne);
6) {
7)   populace.insert (zmutovany nejlepsi);
8) }
9) pokracuj bodem 1)
```

(3.1)

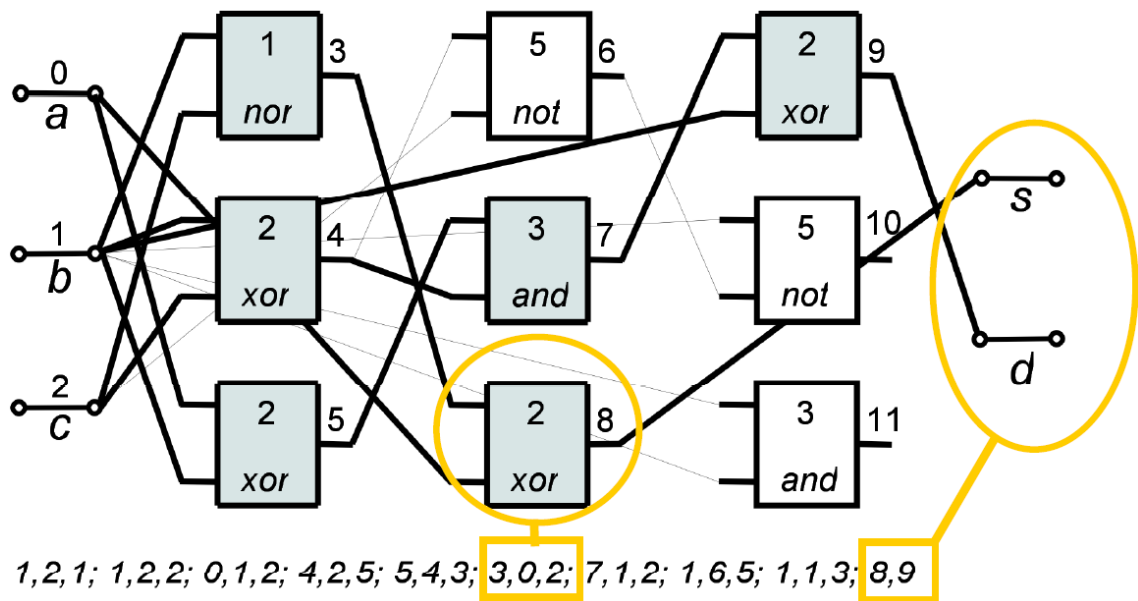
### 3.1 Reprezentace kandidátního řešení v CGP

Celý CGP graf je definován jako  $n$ -tice  $\{G, n_i, n_o, n_n, F, n_f, n_r, n_c, l\}$ , kde  $G$  reprezentuje genotyp a je sám množinou celočíselných hodnot reprezentující  $n_i$  vstupů grafu,  $n_n$  vstupů každého uzlu a  $n_o$  výstupů grafu. Množina  $F$  reprezentuje  $n_f$  funkcí uzlu. Hodnoty  $n_r$  a  $n_c$  udávají číslo uzlu v řádce a ve sloupci. Úroveň propojení v grafu definuje parametr  $l$ . O každém uzlu je potřeba uchovávat informaci, jakou funkci představuje a identifikátory vstupních uzlů. Každý jedinec je zakódován v poli integerů o pevné délce. Tato délka se dá vyjádřit vzorcem 3.2.

$$\|G\| = (n_r * n_c) * (n_n + 1) + n_o \quad (3.2)$$

Vstup libovolného uzlu může být připojen na libovolný primární vstup (na obrázku 3.1 označeny jako a, b, c). Výstupní uzly grafu mohou být připojeny na výstup kteréhokoliv uzlu. Vstupy vlastních uzlů grafu mohou být připojeny na výstup některého z uzlů, který

je v některém z předchozích sloupců. Je zakázáno propojovat uzly ve stejném sloupci. Tedy vstupy uzlu s indexem 11 na obrázku 3.1 mohou být připojeny jednak na primární vstupy grafu (indexovány 0 - 2) stejně jako na výstupy uzlů 3 - 8. Parametr  $l$  je takzvaný **L-back parametr**. Ten určuje, o kolik sloupců dopředu může být uzel připojen. Pokud je hodnota parametru L-back rovna počtu sloupců grafu, nejsme tímto parametrem nijak omezeni, ale při  $L = 1$  může být uzel 11 z obrázku 3.1 připojen pouze na uzly s indexem 6 - 8. Čím nižší je hodnota L-back parametru, tím menší jsou možnosti vytvoření různorodých grafů a je tedy daleko menší velikost stavového prostoru. [17] [25] [24]



Obrázek 3.1: Ukázka genotypu a fenotypu kartézského genetického programování [25]

### 3.2 Vytvoření jedince z genotypu

Na spodní části obrázku 3.1 je znázorněn příklad genotypu CGP a na horní části obrázku 3.1 je znázorněn příklad odpovídajícího fenotypu pro jednoduchou logickou funkci. Při dekódování grafu se postupuje od výstupních uzlů. Výstupní uzly jsou označeny jako **s** a **d**. Vstupem pro **s** je výstup uzlu 8. Jeho vstupem jsou uzly 3 a 4. Jejich vstupem jsou už primární vstupy grafu. Nejprve je tedy potřeba vyhodnotit uzel 3. Zjistí se, jakou funkci uzel představuje. V našem případě má funkce uzlu index 1 čemuž odpovídá logický člen NOR. Ten potřebuje 2 vstupní hodnoty. Vezmou se tedy vstupní hodnoty uzlu 3 (primární vstupy **b** a **c**), pošlou se na vstup funkce NOR a výsledek se pošle jako první vstup uzlu 8. Stejně se vyhodnotí i uzel 4, tím už má uzel 8 své dva potřebné vstupy, může být sám vyhodnocen a výsledek je poslán na výstup **s**. Tím je výstup **s** vyhodnocen. Obdobně se vyhodnotí i výstup **d**.

### 3.3 Genetické operátory

Některé uzly CGP mřížky nemusí být využity. To se může změnit při aplikování genetických operátorů. V kartézském genetickém programování se obvykle využívá výhradně operátor mutace. Křížení se většinou nepoužívá, protože dosud nebyl vymyšlen dostatečně vhodný operátor křížení a současné operátory křížení mívají velmi destruktivní důsledky a nevedou k dobrým výsledkům evoluce. [16] [25] Výjimkou bývá evoluční umění, kde může křížení v CGP vytvořit velmi zajímavé vzory. [5] Mutace bude ukázána na chromozomu z obrázku 3.1:

1, 2, 1<sup>3</sup> 1, 2, 2<sup>4</sup> 0, 1, 2<sup>5</sup> 4, 2, 5<sup>6</sup> 5, 4, 3<sup>7</sup> 3, 0, 2<sup>8</sup> 7, 1, 2<sup>9</sup> 1, 6, 5<sup>10</sup> 1, 1, 3<sup>11</sup> 8, 9

Červeně zbarvené jsou vstupy uzlu, černě funkce, kterou uzel představuje. Každá trojice čísel má index, který odpovídá indexu uzlu. Mutace probíhá takto:

- Náhodně je vybrána jedna hodnota chromozomu:

1, 2, 1<sup>3</sup> 1, 2, 2<sup>4</sup> 0, 1, 2<sup>5</sup> 4, 2, 5<sup>6</sup> 5, 4, 3<sup>7</sup> 3, 0, 2<sup>8</sup> 7, 1, 2<sup>9</sup> 1, 6, 5<sup>10</sup> 1, 1, 3<sup>11</sup> 8, 9

- Nyní je potřeba rozlišit jaká hodnota byla vybrána - funkce či vstup uzlu
- Byla vybrána funkce. Funkce může nabývat hodnot 0 - 5 (6 různých funkcí). Vygeneruje se tedy náhodná hodnota v intervalu 0 - 5 a tou se nahradí pětka v původním chromozomu. V případě vygenerování čísla 2 bude chromozom vypadat následovně:

1, 2, 1<sup>3</sup> 1, 2, 2<sup>4</sup> 0, 1, 2<sup>5</sup> 4, 2, 2<sup>6</sup> 5, 4, 3<sup>7</sup> 3, 0, 2<sup>8</sup> 7, 1, 2<sup>9</sup> 1, 6, 5<sup>10</sup> 1, 1, 3<sup>11</sup> 8, 9

- Při další mutaci může být vybráno jiné místo chromozomu:

1, 2, 1<sup>3</sup> 1, 2, 2<sup>4</sup> 0, 1, 2<sup>5</sup> 4, 2, 2<sup>6</sup> 5, 4, 3<sup>7</sup> 3, 0, 2<sup>8</sup> 7, 1, 2<sup>9</sup> 1, 6, 5<sup>10</sup> 1, 1, 3<sup>11</sup> 8, 9

Nyní byl vybrán vstup chromozomu. Zde je potřeba vzít v úvahu, které uzly je možné mít jako vstupní. Jde o uzel ze třetího sloupce, může tedy mít jako vstup:

- primární vstupy grafu
- výstupy uzlů z prvního a druhého sloupce

Dalším omezením je L-back parametr. Uvažujme stanovenou hodnotu L-back parametru L=1. Výstupy uzlů z prvního sloupce jsou tedy nedostupné. Jako povolené hodnoty na tom místě chromozomu nám zbyla množina {0, 1, 2, 6, 7, 8}. Z té náhodně jednu vybereme a dostaneme nový (zmutovaný) chromozom:

1, 2, 1<sup>3</sup> 1, 2, 2<sup>4</sup> 0, 1, 2<sup>5</sup> 4, 2, 5<sup>6</sup> 5, 4, 3<sup>7</sup> 3, 0, 2<sup>8</sup> 7, 1, 2<sup>9</sup> 1, 0, 5<sup>10</sup> 1, 1, 3<sup>11</sup> 8, 9

Jak bylo ukázáno, mutací se může změnit funkce uzlu či propojení uzlů. Z neaktivních uzlů se tak můžou stát aktivní, nebo naopak aktivní uzly se stanou neaktivní. Častým jevem jsou neutrální mutace. To je taková mutace, kdy má po zmutování jedinec pořád stejnou fitness. Může se to stát zmutováním neaktivního uzlu nebo je hodnota zmutována na stejnou hodnotu. Neutrální mutace se mohou výrazně projevit v pozdější fázi evoluce [24]. To může nastat, když je po mnoho generací mutována oblast GCP mřížky, která není používána. Evoluce může v takové neaktivní části vytvořit specifické propojení uzlů a jejich funkce. Toto propojení může být později dalšími mutacemi zahrnuto mezi aktivní uzly.

## 3.4 Speciální varianty CGP

Kartézské genetické programování je univerzální metoda pro řešení obecných úloh. Jsou ale případy, kdy CGP nedokáže nalézt výsledek nebo jeho nalezení trvá příliš velkou dobu. Aby se vylepšily vlastnosti CGP (nebo pro urychlení evoluce), upravují se vlastnosti CGP, čímž vznikají nové variaty kartézského genetického programování.

### 3.4.1 Cyklické CGP

V původní variantě CGP je řešení reprezentováno ve formě acyklického grafu. Graf tedy nemá žádnou zpětnou vazbu. Varianta CGP označovaná jako cyklické CGP umožňuje odstranit toto omezení. Reprezentace grafu používaného v CGP se jednoduše adaptuje na zakódování cyklického grafu. Je potřeba pouze odstranit omezení, že uzly grafu mohou jako vstupní hodnoty brát jen výstupy od uzlů z předchozích sloupců. Existuje ale pouze málo prací, které publikovali použití CGP bez tohoto omezení. Jednou z výjimek je práce od Khana, který zakódoval neuronovou síť v CGP[11]. Dovolil zpětnou vazbu a použil CGP k vytvoření rekurentní neuronové sítě. Další výjimkou je práce Walkera, který aplikoval CGP k evoluci strojového kódu sekvenčních a paralelních programů na procesoru MOVE[26]. Prohledávání cyklického grafu výrazně zesložití program (zpětná vazba způsobí další iterace). Dovolí to ale nalézt jak synchronní tak asynchronní obvody. [15]

### 3.4.2 Více-chromosomové CGP

Rozdíl mezi CGP a více-chromosomovým CGP je v tom, že genotyp více-chromosomového CGP (MC-CGP) je rozdělen do několika chromosomů stejné délky. Počet chromosomů do kterých je genotyp rozdělen je určen počtem výstupů, které daný problém vyžaduje. Každý chromosom je připojen na jeden konkrétní výstup programu. To dovoluje velkým problémům s násobnými výstupy (které jsou normálně zakódovány v jednom genotypu), aby byly rozděleny do několika menších problémů. Každý z těchto malých problémů je pak zakódován v jednom chromosomu. Každý chromosom má jeden výstup. Myšlenka je, že tento přístup by měl usnadnit řešení problému. Jednou možností by bylo provádět evoluci všech chromosomů postupně. Jinou možností je nechat vyvíjet všechny chromosomy zároveň. Tím, že je dovoleno každému podproblému, aby byl zakódován v chromosomu, je celý problém zakódován v jednom genotypu, ale propojení mezi podproblémy je odstraněno (mohlo by způsobit problémy v rozsahu fitness). Každý chromosom obsahuje shodný počet uzlů a je s ním zacházeno jako s individuálním jedincem s jedním programovým výstupem. Vstup každého uzlu, který je zakódován v chromosomu, může být připojen pouze na výstupy uzlů ve stejném chromosomu anebo na programové vstupy. Více-chromosomový přístup k CGP sdílí některé podobnosti s jinými GP technikami, jako je Parisian GP[3]. V této technice je řešení problému získáváno z jednotlivých podřešení. Nicméně v Parisian GP jedinec reprezentuje jenom část řešení a celé řešení sestává z množiny jedinců z celé populace. Toto odlišuje od multi-chromosomového přístupu k CGP, jehož každý chromosom kóduje řešení odlišného podproblému a řešení celého problému je obsaženo v jednom jedinci, který sestává z počtu chromosomů, každý z nich kóduje odlišný podproblém. Další odlišností je, že Parisian GP používá dvě odlišné fitness funkce. Lokální fitness funkci k přístupu k podílu každého jedince a globální fitness funkci k vyhodnocení, jak dobře jedinec řeší daný problém. Oproti tomu více-chromozomové CGP používá jednu fitness funkci k vyhodnocení, jak dobře řeší jedinec podproblém, který mu byl přidělen. Když jsou všechny

podproblémy vyřešeny, je vyřešen i celý problém. Všechny chromosomy pro jedince jsou obsaženy v jednom genotypu. To dovoluje sdílení a znovupoužití genetické informace spojené s vysokou fitness mezi chromosomy v genotypu jednoho jedince. Proto částečné řešení, které je přítomno ve všech chromosomech, je nutné zkonstruovat jenom v jednom chromosomu a je znovupoužito v jiném chromosomu. Teoreticky toto může znovu zavést konektivitu mezi chromosomy. Znovupoužití relevantní informace z jednoho chromosomu v jiném chromosomu je podobné propojení části genotypu týkající se jednoho podproblému s jinou částí genotypu týkající se jiného podproblému. [28] [27] [15]

### 3.4.3 Sebe modifikující kartézské genetické programování

V některých formách GP je drobná odlišnost mezi genotypem a fenotypem. Například ve stromově založeném GP je genotypem výraz v LISPu. Fenotypem je kompilovaný LISPový program. V biologii je genotypem genetická instrukce zakódovaná v každé buňce DNA. Za fenotyp je považována fyzická forma organismu. Toto by nemělo být zaměňováno s chováním organismu. V CGP je rozdíl mezi genotypem a fenotypem daleko více zřejmý. Genotyp je dekodován do kompaktnější formy, která je fenotypem. Při procesu dekodování mohou být genetické instrukce ignorovány, protože nejsou zahrnuty do mapování ze vstupu programu na výstup. Výše již bylo řečeno, že „zbytečné“ geny mohou být prospěšné v evolučním procesu. Při pohledu na biologii můžeme vidět, že v mnoha případech (jestli ne ve všech), je základní jednotkou při mapování z genotypu do fenotypu čas. Toto je možná nejjasnější, když uvážíme vývoj vícebuněčných organismů. V těchto systémech je rozdíl mezi genotypem a fenotypem enormní, protože ve fenotypu může být velké množství buněk. Tedy fenotyp je tvořen v průběhu času, v důsledku interakce organismu s prostředím. Zahrnutí pojmu čas v mapování z genotypu do fenotypu může být velmi prospěšné. Sebe-modifikující kartézské genetické programování (SMCGP) zahrnuje čas tak, že dovoluje novým druhům funkcí, aby vstoupily do genotypu. Jsou to sebe-modifikující funkce, tj. instrukce, které způsobí změnu samotného kódu. Když jsou tyto funkce dodržovány, genotyp se změní do nové podoby (tj. vyvine se v nový fenotyp). Nový fenotyp může také zahrnovat sebemodifikující funkce, takže když jsou tyto funkce aplikovány, může být vytvořen další fenotyp. Každá fáze provádění sebemodifikujících (SM) instrukcí se nazývá iterací. Jiné hledisko biologické buňky je, že je citlivá na libovolný počet vstupů. Tedy kromě času jsou zavedeny také CGP funkce, které si přidají (případně odeberou) programové vstupy a výstupy. Při použití ve shodě s SM funkcemi, toto dovoluje SMCGP programu získat více vstupů a dodat více výstupů. Tyto dvě nová vylepšení umožňují CGP, aby bylo schopné najít ještě lepší řešení problému. [7] [6]

## Kapitola 4

# Symbolická regrese

Symbolická regrese je metoda, která se snaží nalézt matematický výraz, který aproximuje zadanou funkci. Pro řešení úloh symbolické regrese je možné použít některé evoluční algoritmy, především pak genetické a kartézské genetické programování. Výsledkem má být předpis funkce, která co nejlépe popisuje referenční data, neboli aby pro zadané vstupy vracela požadované výstupy. Referenčními daty bývá množina bodů. [22] Tradiční lineární<sup>1</sup> a nelineární<sup>2</sup> regrese hledají přímku respektive křivku takovou, aby součet odchylek přímkou respektive křivkou od všech bodů byl nejmenší. Metody využívající k řešení problému symbolické regrese genetické algoritmy vytvoří na počátku náhodnou množinu matematických výrazů. Tyto výrazy mohou obsahovat základní matematické operace, jako je sčítání, odčítání, násobení a dělení, případně jiné složitější funkce, například goniometrické nebo logaritmické funkce. Genetický algoritmus následně využívá operátorů křížení a mutace k vytváření nových výrazů a upřednostňuje výrazy, které lépe vyhovují zadání úlohy. Jakmile je dosaženo dostatečně kvalitní řešení, je algoritmus ukončen. Symbolická regrese je často využívána k nalezení explicitních i diferenciálních rovnic. [4] [12]

### 4.1 Příklad symbolické regrese

Jednoduchým příkladem na symbolickou regresi může být nalezení výrazu, který nejlépe aproximuje data uvedená v tabulce 4.1. Kromě trénovacích dat je potřeba určit množinu funkcí, ze kterých bude výsledný výraz sestavován. Mohou to být veškeré aritmetické operátory, analytické funkce apod. Musí být ale zajištěno, aby to všechno byly takzvané chráněné operace. To znamená, že jejich hodnota je vždy definována. Například dělení nulou není standardně definováno. Při symbolické regresi ale může být tento problém vyřešen například tím, že pokud bude výraz dělen nulou, výsledkem toho dělení bude nějaká konstanta (typicky 0 nebo 1). Pokud tedy bude děleno například  $8/0$ , neskončí toto dělení chybou, ale výsledkem bude nula. Obdobně výpočet logaritmu může být ošetřen pro nekladná čísla například způsobem, který je popsán pomocí algoritmu 4.1.

```
if  $x == 0$ 
    return 0
else
    return  $\ln||x||$ 
endif
```

(4.1)

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Linear\\_regression](https://en.wikipedia.org/wiki/Linear_regression)

<sup>2</sup>[https://en.wikipedia.org/wiki/Nonlinear\\_regression](https://en.wikipedia.org/wiki/Nonlinear_regression)

Obdobně další funkce, které nejsou pro některé hodnoty definovány. Fitness funkci je možné definovat několika způsoby. Jedním z nich je suma druhých mocnin odchylky nalezené funkce od referenčních bodů. Čím menší výsledná fitness bude, tím je řešení lepší. Výpočet této fitness funkce se dá vyjádřit vzorcem 4.2.

$$fitness(s) = \sum_{i=1}^m (s(x_i) - y_i)^2 \quad (4.2)$$

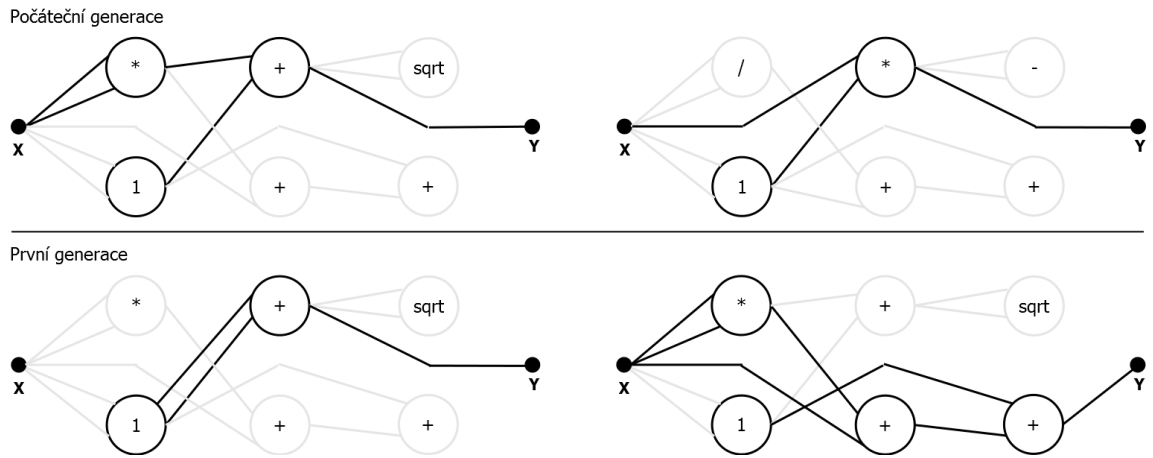
Další možností výpočtu fitness funkce je stanovení skóre řešení. Je definována mezní odchylka  $\varphi$ . Absolutní hodnota rozdílu nalezené funkce od referenčních dat je pak porovnávána s  $\varphi$  a pokud je větší než  $\varphi$ , je k fitness funkci přičtena jednička, v opačném případě se k fitness funkci nic nepřičte. Výpočet této fitness funkce se dá vyjádřit vzorcem 4.3.

$$fitness(s) = \sum_{i=1}^m \begin{cases} 1 & \text{pro } |s(x_i) - y_i| \geq \varphi \\ 0 & \text{pro } |s(x_i) - y_i| < \varphi \end{cases} \quad (4.3)$$

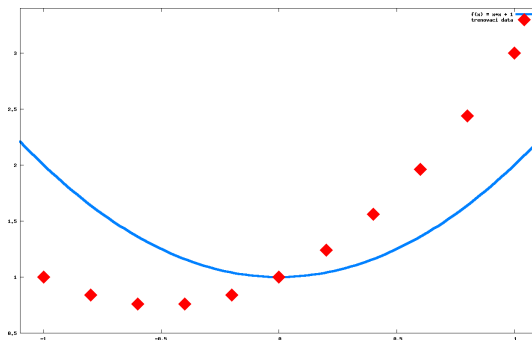
Při použití kartézského genetického programování pro symbolickou regresi musí být zvolena velikost populace a velikost mřížky CGP. V tomto příkladu bude použita pro jednoduchost velikost mřížky 3x2 a populace o velikosti 2. Ukončující podmínkou evoluce bude dosažení hodnoty fitness funkce nejlepšího jedince pod hodnotu 0,1 při použití fitness funkce dané vztahem 4.2. Pro počáteční populaci byli náhodně vygenerováni 2 jedinci. Jak je vidět na obrázku 4.1, první jedinec reprezentuje výraz  $(x * x) + 1$  (obrázek 4.2) a druhý jedinec reprezentuje výraz  $x * 1$  (obrázek 4.3). Při použití trénovacích dat z tabulky 4.1 a fitness funkce 4.2 je fitness jedince  $x^2 + 1$  rovna hodnotě 1,00. Fitness jedince  $x$  je rovna hodnotě 2,67. Fitness nejlepšího jedince je větší než povolují ukončující podmínky, a proto se bude pokračovat v evoluci. Selekcí je vybrán jedinec s nejvyšší fitness, což je jedinec reprezentující výraz  $x^2 + 1$ . Mutace proběhne záměnou dvou náhodně vybraných genů za nově vygenerované náhodné hodnoty, tak vzniknou dva noví jedinci (obrázek 4.1). Při první mutaci bylo změněno připojení jednoho ze vstupů 3. uzlu (počítáno po sloupcích). Výsledný jedinec tak reprezentuje výraz  $1 + 1$  (obrázek 4.4) a jeho fitness je rovna hodnotě 1,70 (hodnota fitness funkce potomka je v tomto případě horší, než hodnota fitness funkce předka). Při druhé mutaci bylo zmutováno připojení výstupního uzlu, čímž vznikl jedinec  $(x * x) + x + 1$  (obrázek 4.5) s fitness funkcí 0,00. Do první generace postoupí jeden jedinec z počáteční generace ( $x^2 + 1$ ) a jeden nový jedinec  $x^2 + x + 1$ . Jedinec s fitness hodnotou 1,70 nebyl dostatečně dobrý, a byl proto vyřazen. Následně dojde ke kontrole, zda nebyla splněna ukončující podmínka. Bylo nalezeno řešení s fitness funkcí lepší, než hodnota 0,1 (ukončující podmínka evoluce) a evoluce může být ukončena. Výsledkem symbolické regrese je tedy jedinec  $x^2 + x + 1$ .

X	-1,0	-0,8	-0,6	-0,4	-0,2	0,0	0,2	0,4	0,6	0,8	1,0
Y	1,0	0,84	0,76	0,76	0,84	1,0	1,24	1,56	1,96	2,44	3,0

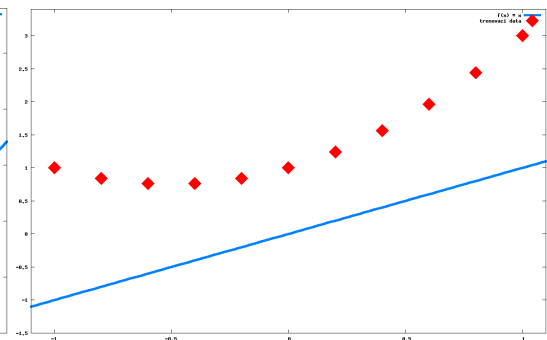
Tabulka 4.1: Příklad trénovacích dat pro úlohu symbolické regrese



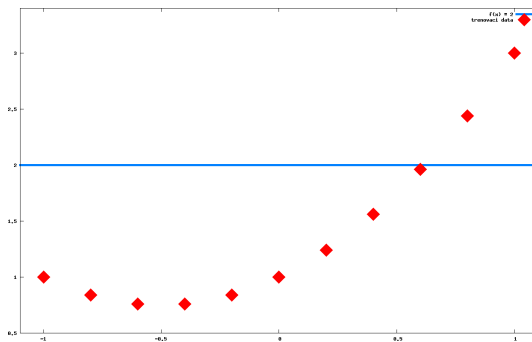
Obrázek 4.1: Symbolická regrese pomocí CGP



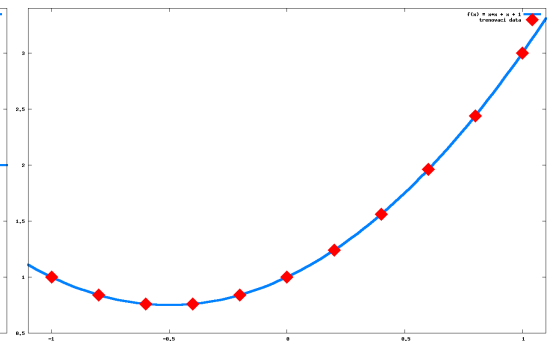
Obrázek 4.2: Vstupní data + fce  $x^2 + 1$



Obrázek 4.3: Vstupní data + fce  $x$



Obrázek 4.4: Vstupní data + fce  $2$



Obrázek 4.5: Vstupní data + fce  $x^2 + x + 1$

## Kapitola 5

# Predikce dopravy

Většina dopravních veličin, jako např. intenzita a obsazenost dopravního proudu, má typický průběh. Ve velké míře mohou být pozorovatelné denní, víkendové a sezónní vzory dopravní situace. Například denní vzory zahrnují ranní nástup dopravní zátěže, odpolední dopravní špičku a minimální provoz pozdě v noci. Týdenní vzory rozlišují dopravu ve všední dny a o víkendech. Sezónní vzory rozlišují zimní a letní dopravu. Tyto charakteristiky vedou na speciální způsob jejich předpovědi.

### 5.1 Stupně provozu

V České republice se pro vyjádření hustoty dopravy používá pětibodová stupnice, která je považována za standardní. První stupeň provozu znamená, že doprava je plynulá, na komunikaci se vyskytují pouze jednotlivá vozidla. Druhý stupeň říká, že na komunikacích se vyskytují malé skupinky vozidel, ale jízda je pořád plynulá, odbavování na křižovatkách je bez větších problémů. Při 3. stupni provozu se tvoří proudy vozidel a jízda již není při maximální povolené rychlosti plynulá. Čtvrtý dopravní stupeň udává nízkou plynulost provozu. Na silnicích se tvoří kolony, průjezd křižovatkami je pomalý, vozidla jsou nucena pohybovat se jen velmi pomalu. Pátý dopravní stupeň značí dopravní kolaps. Na silnicích jsou velké kolony a vozidla v nich stojí nebo popojíždí jen minimální rychlostí. [18] Ředitelství silnic a dálnic provozuje webový portál <http://www.dopravniinfo.cz/>, díky němuž je možné sledovat míru dopravy a další dopravní informace online. Bohužel prozatím je možné sledovat hustotu provozu jen na největších silničních komunikacích.

### 5.2 Senzory

Dopravní situace je detekována pomocí senzorů. Různé dopravní veličiny jsou detekovány různými typy senzorů.

#### **Senzory ve vozovce**

Mezi nejpoužívanější detektory patří indukční smyčky. Jedná se o kovovou smyčku zabudovanou pod povrchem vozovky. Ta kolem sebe vytváří magnetické pole, které je narušeno vždy, když přes smyčku přejede vozidlo. Tato změna je detekována a slouží ke zjišťování přítomnosti vozidel či vzdáleností mezi vozidly. Pomocí této techniky je možné identifikovat kolony. Při použití více smyček položených za sebou se dá měřit rychlost vozidel.

#### **Mikrovlnné detektory**

Instalují se bez zásahu do vozovky. Neničí vozovku a tím ani nesnižují její životnost. Dají

se snadno sejmout a přemístit. Pomocí nich se dá spočítat počet vozidel, jejich rychlost a dokonce i délka jednotlivých vozidel. Mikrovlnné detektory se dále dělí na Dopplerův radar a mikrovlnný radar s frekvenčně modulovanými spojitými vlnami. Dopplerův radar vysílá signál konstantní frekvence, a proto nemůže detekovat stojící vozidla.

#### Infračervené detektory

Jsou citlivé na světlo. Dělí se na aktivní a pasivní detektory. Aktivní detektory vysílají záření blízké infračervené oblasti a přijímají odražené světlo. Měření může být ovlivněno hustým deštěm a sněžením. Všechny objekty, které nemají teplotu rovnu absolutní nule<sup>1</sup>, vyzařují energii. Pasivní infračervené detektory tuto energii zachytávají. Díky tomu mohou detekovat intenzitu dopravy, počítat vozidla, klasifikovat vozidla, detekovat chodce na přechodech. Aktivní detektory mohou kromě těchto údajů navíc zaznamenávat rychlosti vozidel a obsazenost komunikací.

#### Ultrazvukové detektory

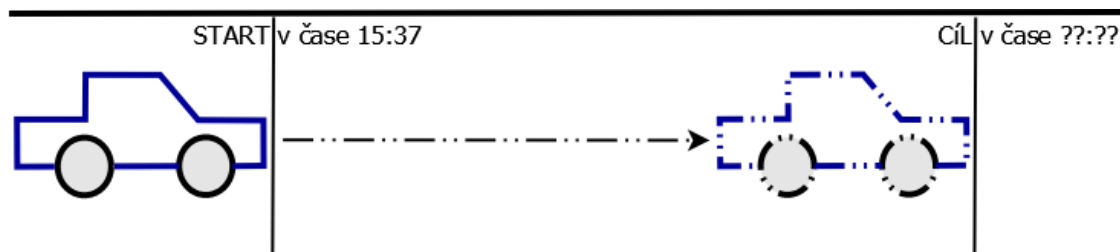
Vysílají ultrazvukové signály a odražené je přijímají zpět. Slouží ke zjišťování přítomnosti vozidel a k měření jejich vzdálenosti.

#### Videodetekce

Umožňuje sledovat více parametrů najednou. Oproti jiným detektorům umožňuje zaznamenat havárii vozidel. Nevýhodou je závislost na počasí a nutnost zpracování velkého množství dat. [20]

### 5.3 Predikce dojezdových dob

Jak je znázorněno na obrázku 5.1, u dojezdové doby se zjišťuje čas, za který automobil dorazí z aktuální pozice do nějaké cílové pozice, která nás zajímá. Například může být užitečná informace, za jak dlouho se automobil dostane od jedné světelné křižovatky ke druhé.



Obrázek 5.1: Dojezdová doba

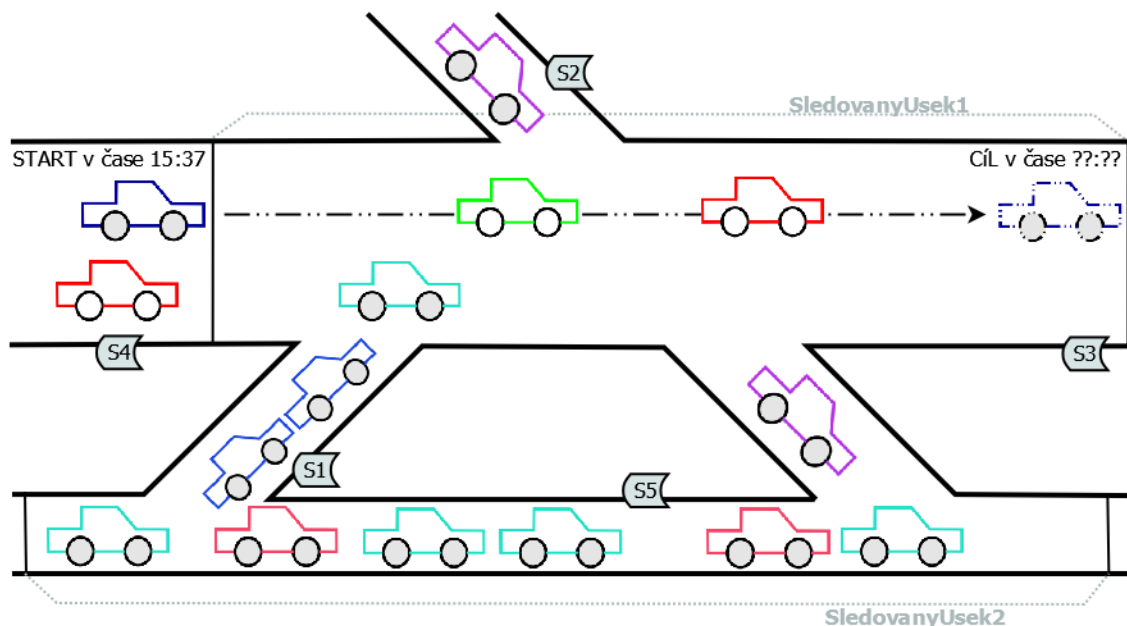
Tyto vlastnosti dělají predikci dojezdové doby velmi komplexní a je obtížné dosáhnout velké přesnosti. Nicméně ve velké míře mohou být pozorovatelné denní, víkendové a sezónní vzory dopravní situace. Například denní vzory zahrnují dopravní špičku a minimální provoz pozdě v noci. Týdenní vzory rozlišují dopravu ve všední dny a o víkendech. Sezónní vzory rozlišují zimní a letní dopravu. Tyto časově variabilní vlastnosti jsou podstatné pro chování dopravy a měly by být klíčem pro modelování dojezdových dob

Dojezdové doby jsou jedny z nejdůležitějších dat pro velké množství dopravních analýz. Mohou být použity při plánování dopravy, návrhu, obsluze a vyhodnocování. Dojezdové doby jsou zejména významné pro dopravní informace před cestou a během ní v pokročilých

<sup>1</sup>[https://cs.wikipedia.org/wiki/Absolutn%C3%AD\\_nula](https://cs.wikipedia.org/wiki/Absolutn%C3%AD_nula)

dopravních informačních systémech. Ty jsou velmi užitečné pro řidiče při plánování cesty. S přesnou predikcí dojezdové doby může navigační systém navrhnout optimální alternativní cestu anebo varovat řidiče před potencionálními dopravními zácpami. Řidič pak může lépe rozhodnout o optimálním času odjezdu anebo odhadnout jeho čas příjezdu na základě predikovaných dojezdových dob. Dojezdovou dobu (stejně jako většinu dopravních veličin) ovlivňuje spousta faktorů, jako například rychlost vozidla, aktuální dopravní zátěž, nepříznivé počasí (husté sněžení), uzavření jedné z okolních ulic z důvodu plánované opravy komunikace nebo naopak náhlá uzavírka z důvodu dopravní nehody. Když je zvýšený stupeň dopravní zátěže na jedné dopravní komunikaci, je velmi pravděpodobné, že na komunikaci s ní sousedící bude také vysoký stupeň dopravní zátěže. Jak je znázorněno na obrázku 5.2, doprava na dolní dopravní komunikaci pojmenované „SledovanýÚsek2“ dosahuje vyššího stupně, je proto velmi pravděpodobné, že se zvýší stupeň dopravy na silnici na ni přímo napojené a následně na silnici na které sledujeme dojezdovou dobu (pojmenované „SledovanýÚsek1“). Dojezdová doba se proto v důsledku zvýšení stupně dopravy na komunikaci „SledovanýÚsek1“ prodlouží. Nejjednodušším způsobem zjišťování dojezdové doby je zaznamenat, jak dlouho trvalo jednomu vozidlu projet sledovaným úsekem. Tím je získána dojezdová doba jednoho vozidla, například 1 minuta a 15 sekund. Nyní můžeme dalšímu vozidlu, které právě dorazilo na začátek sledovaného úseku předpovědět, že mu bude trvat 1 minutu a 15 sekund než dorazí na konec sledovaného úseku. Tento způsob předpovědi dojezdové doby je velmi jednoduchý a levný. Stačí k němu pouze 2 senzory. Jeden senzor na počátku sledovaného úseku, který zaznamená potřebné údaje k identifikaci vozidla. Takovým údajem může být například státní poznávací značka, která je na každém vozidle unikátní. K dané SPZ následně stačí zaznamenat pouze aktuální čas měření. Druhý senzor umístěný na konci úseku opět detekuje státní poznávací značky vozidel. Detekovanou značku přiřadí ke značce zaznamenané prvním senzorem a odečte od aktuálního času čas zaznamenaný prvním senzorem. Pro tento způsob predikce jsou tedy dostatečné 2 kamery a software pro rozpoznávání SPZ ve videu. Jeden z takových automatických systémů detekce a rozpoznávání poznávacích značek automobilů byl vyvinut v Japonské Osace [10]. Tento způsob predikce dojezdových dob se ale hodí pouze na místa, kde se dopravní provoz příliš nemění nebo se mění jen velmi pozvolna. Pokud je proměnlivost provozu příliš vysoká, přináší tento způsob předpovědi chybná data. Chyba je tím vyšší, čím je sledovaný úsek delší a čím více vlivů na něj může působit. Mezi tyto vlivy může patřit počet křižovatek a odboček na sledovaném úseku. Taková situace může být ukázána na obrázku 5.2. Byla změřena doba, jak dlouho trvalo nějakému vozidlu projet úsekem „SledovanýÚsek1“, na kterém byl momentálně provoz stupně 1. Tato doba byla například 2 minuty. Nyní, jak je ukázáno na obrázku 5.2, přijelo na začátek úseku další vozidlo (na obrázku znázorněno modrou barvou). Aktuální čas je změřen senzorem S4 jako 15:37. Podle použitého jednoduchého předpovědního modelu by mělo modré vozidlo dorazit na konec úseku za 2 minuty, tedy v čase 15:39. Na jiném silničním úseku, který je na obrázku 5.2 označen jako „SledovanýÚsek2“, se však postupně zvyšuje hustota provozu. Provoz tam již dosáhl 4. stupně a řidiči proto začali přejíždět silnicí z komunikace pojmenované „SledovanýÚsek2“ do komunikace „SledovanýÚsek1“. Díky tomu se zvýšil stupeň provozu na úseku „SledovanýÚsek1“. Modré vozidlo již nemá tak volnou cestu jako mělo první vozidlo a dorazí na konec úseku za 3 minuty. Predikce tedy byla díky neaktuálním údajům chybná. Lepším způsobem predikce by bylo využít i data o dopravě v okolních ulicích. V takovém případě by se daly lépe sledovat i náhlé změny v dopravě způsobené například dopravní nehodou. Při detekci zvýšení provozu na jedné silniční komunikaci a s tím souvisejícímu prodloužení dojezdové doby na dané komunikaci by se okamžitě adekvátně upravila dojezdová doba na okolních úsecích.

Cílem této práce je nalezení vztahů mezi dopravními veličinami, na základě kterých by bylo možné predikovat dojezdovou dobu v závislosti na měnícím se provozu v okolních ulicích.



Obrázek 5.2: Dojezdová doba v závislosti na okolní dopravě

## 5.4 Predikce dojezdových dob s využitím soft-computingových metod

### 5.4.1 Support vector regression

Chun-Hsin Wu ve své práci Travel-Time Prediction With Support Vector Regression využil pro predikci dojezdových dob support vector machine (SVM<sup>2</sup>). Support vector machine byl navržen původně ruským výzkumníkem Vapnikem z AT&T Bellových laboratoří. Bylo intenzivně studováno jeho využití u klasifikace a regrese. SVM má velký potenciál a vysoký výkon v praktických aplikacích. To je do velké míry způsobeno principem minimalizace strukturálních rizik<sup>3</sup> u SVM, což má větší schopnost generalizace a je lepší než empirická minimalizace rizik (ERM<sup>4</sup>), která je využívána v neuronových sítích. U SVM výsledek garantuje globální minimum, zatímco ERM může nalézt jenom lokální minimum. Navíc SVM je adaptabilní ke komplexním systémům a robustní při práci s poškozenými daty. Tyto vylepšení nabízí SVM větší schopnost generalizace, což je slabé místo u přístupu neuronových sítí. Tradičně se mnoho studií zaměřuje na aplikaci SVM v klasifikaci dokumentů ke hledání vzorů. Pro inteligentní transportní systémy je také mnoho prací aplikujících SVM na dopravu, jako je detekce vozidel či rozpoznávání dopravních vzorů. Tyto výsledky jsou důkazem použitelnosti SVM v dopravě. Další aplikace SVM na předpovídání časových řad se nazývá support vector regression (SVR) [2]. Tato metoda má také mnoho objevů jako

<sup>2</sup>[http://en.wikipedia.org/wiki/Support\\_vector\\_machine](http://en.wikipedia.org/wiki/Support_vector_machine)

<sup>3</sup>[http://en.wikipedia.org/wiki/Structural\\_risk\\_minimization](http://en.wikipedia.org/wiki/Structural_risk_minimization)

<sup>4</sup>[http://en.wikipedia.org/wiki/Empirical\\_risk\\_minimization](http://en.wikipedia.org/wiki/Empirical_risk_minimization)

předpovídání finančních trhů, předpověď ceny elektřiny, spotřeby energií a rekonstrukce chaotických systémů. Výjimkou pro predikci dopravy je několik výsledků SVR analýzy časových řad pro inteligentní transportní systémy (ITS). Wu se pokusil pomocí SVR predikovat dojezdové doby pro řidiče na dálnici. Predikce pomocí support vector regression byla porovnáována se dvěma dalšími predikcemi. Jednou z nich byla metoda predikce aktuálních dojezdových dob a druhou byla metoda predikce pomocí historického průměru. Metoda predikce aktuální dojezdové doby počítala dojezdovou dobu v okamžiku provádění predikce. Dojezdová doba je pak definována vzorcem 5.1, kde  $\Delta$  je zpoždění dat,  $L$  je počet sekcí,  $x_{i+1} - x_i$  označuje vzdálenost sekcí na dálnici a  $v(x_i, t - \Delta)$  je počáteční rychlost na počátku sledovaného dálničního úseku.

$$T(t, \Delta) = \sum_{i=0}^{L-1} \frac{x_{i+1} - x_i}{v(x_i, t - \Delta)} \quad (5.1)$$

Metoda predikce za pomoci historického průměru získává dojezdové doby z průměrných dojezdových dob historických dat ve stejný čas dne a z odpovídajícího dne v týdnu. Dojezdová doba je pak vypočítána pomocí vzorce 5.2, kde  $w$  je počet víkendů pro které je model trénován a  $T(i, t)$  je minulá dojezdová doba v čase  $t$  historický víkend  $i$ .

$$\bar{T}(t) = \frac{1}{w} * \sum_{i=1}^w T(i, t) \quad (5.2)$$

Podle očekávání, prediktor založený na historickém průměru nemůže zachytit dopravní vzory, které jsou významně odlišné od posledního průměru a aktuální prediktor je obvykle pomalý k tomu, aby zachytil změny v dopravních vzorech. Nicméně SVR může konvergovat rychle a vyhnout se lokálnímu minimu. SVR predictor funguje v těchto experimentech velmi dobře. Výsledky ukazovaly, že SVR redukuje odchylku jak u střední průměrné odchylky, tak u střední kvadratické odchylky na méně než polovinu toho co dosáhly prediktory aktuálního času a historického času pro všechny odlišné vzdálenosti. Ve Wuových experimentech se zvětšující dojezdovou vzdáleností počet volných sekcí narůstal více než počet obsazených sekcí, takže dojezdové doby dlouhých vzdáleností jsou dominovány časy na volných sekcích. Proto není překvapující, že všechny tři prediktory predikují dobře pro dlouhé vzdálenosti (350km), toto ale komplikuje porovnání výkonnosti těchto tří prediktorů. Z toho důvodu byly prozkoumány speciálně ty testovací body dat, na kterých je chyba kteréhokoliv prediktoru větší než 5%. V tomto případě prediktor SVR nejenže vylepšuje celkový výsledek, ale také významně redukuje predikované chyby v případech, kde jsou predikovány velké chyby v kterémkoliv prediktoru. Support vector machine a SVR demonstrovali jejich úspěšnost v časově závislé analýze a statistickém učení. Výsledek ukazuje, že SVR prediktor významně překonává jiné základní prediktory. Toto dokazuje aplikovatelnost SVR na analýzu dojezdových dob. [30]

#### 5.4.2 Neuronové sítě

Park a Rilett se ve svém článku „Forecasting Freeway Link Travel Times with a Multi-layer Feedforward Neural Network“ zaměřili na využití neuronové sítě pro předpovídání dojezdových dob na více časových úsecích do budoucna. Model neuronové sítě byl vybrán, protože tento model má schopnost vzít v úvahu informace o prostorových a časových informacích o dojezdových dobách současně. Například může být očekáváno, že informace o

dojezdové době v souvislosti s předchozím časovým intervalem může být použita pro predikci, co se stane v blízké budoucnosti. Užitečné může být i spojení dojezdové doby z úseků navazujících na cílový úsek. To znamená, že narůstající zahlcení na předcházejícím úseku může být užitečný indikátor prodlužování dojezdové doby na cílovém úseku. Data použitá jako vstupní byly získány z Houstonu a Texasu. Pro predikci dojezdových dob byla zvolena plně propojená vícevrstvá dopředná neuronová síť kombinovaná s algoritmem backpropagation. Backpropagation byl vyvinut pro trénování vícevrstevných neuronových sítí s cílem minimalizovat chyby mezi aktuálním a požadovaným výstupem. Trénování neuronových sítí probíhalo několik desítek dnů a bylo provedeno 80 opakování každého experimentu. Ke změření výkonnosti modelu byla použita střední absolutní odchylka. Při predikci dojezdových dob v blízké době dávalo lepší výsledky, když byla použita informace pouze o nedávné dojezdové době na daném úseku, než použití doplňujících informací o dojezdových dobách z přilehlých úseků. Ale při predikci dojezdových dob více do budoucnosti tomu bylo přesně naopak. Výsledky predikce pomocí neuronové sítě byly porovnány s modelem daným Kalmanovým filtrem, navigační metodou ALI\_SCOUT<sup>5</sup>, historickým profilem dojezdových dob, profilem dojezdových dob měřených v reálném čase a modelem exponenciálního vyhlazování. Historický profil predikuje dojezdové doby na základě dlouhodobého průměru. Naopak real-time profil předpokládá, že aktuální dojezdová doba bude pokračovat i v budoucnu. Použití real-time predikce nikdy nepřineslo nejlepší výsledek pro jakoukoliv situaci. Neuronová síť dávala nejlepší výsledky při predikci 3 - 5 časových úseků do budoucnosti. Kalmanův filtr dával nejlepší výsledky při predikci 1 - 2 časové úseky do budoucna. Real-time profil, Kalmanův filtr, ALI\_SCOUT model a exponenciální vyhlazování dávaly podobné výsledky při predikci 3 - 5 časových úseků do budoucna. Neuronová síť dosahovala nejlepších výsledků, když využívala real-time dojezdové doby a historické dojezdové doby současně tak, že rozpoznávala vzory v historických datech. Rozpoznávání vzorů v historických dojezdových dobách umožňovalo neuronové síti predikovat dojezdové doby na základě aktuálních a nedávných podmínek. [19]

---

<sup>5</sup><http://www.umich.edu/~driving/publications/UMTRI-96-30.pdf>

## Kapitola 6

# Metoda predikce

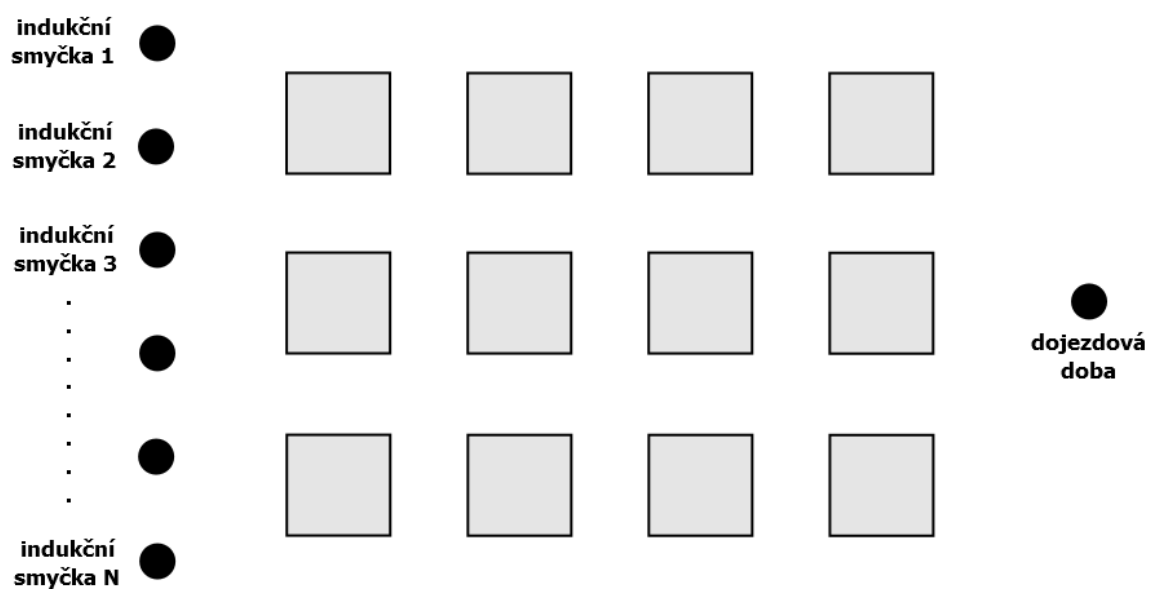
Vstupními daty kartézského genetického programování jsou jednotlivé hodnoty ze senzorů v konkrétním čase na dané sekci. Například když dopravu na sekci "XYZ" zaznamenává 20 senzorů, pak má kartézské genetické programování 20 vstupů. Mezi těmito vstupními daty může být obsazenost silniční komunikace, průměrná rychlost vozidel, atd. Nad vstupními daty je proveden výpočet, který evoluce vytvoří pomocí CGP mřížky. Výsledek tohoto výpočtu je přiveden na jediný výstup CGP mřížky. Tento výstup reprezentuje dojezdovou dobu.

Fitness funkce je vypočítána podle vzorce 4.2, kde  $x_i$  je hodnota vypočítaná matematickým výrazem vytvořeným CGP a  $y_i$  je skutečná naměřená dojezdová doba v daném čase. Výsledek vzorce 4.2 je navíc nakonec podělen celkovým počtem hodnot, ze kterých je dojezdová doba na sekci predikována a výsledek dělení je odmocněn. Tím je získána průměrná odchylka každé predikované hodnoty.

Pro predikci dojezdových dob byla použita základní varianta kartézského genetického programování popsaná v kapitole 3. Jedinou úpravou oproti popsané základní variantě je přidání jedné hodnoty do genotypu. V základní variantě je každý uzel CGP mřížky reprezentován třemi celočíselnými hodnotami. První dvě jsou vstupy uzlu. Poslední hodnotou je funkce uzlu. V CGP implementovaném v rámci této diplomové práce byla do genotypu přidána ještě jedna celočíselná hodnota. Ta představuje konstantu, která může být použita při výpočtu. Tato hodnota se použije, pokud je jako funkce uzlu zvolena právě konstanta. Jeden uzel CGP mřížky je pak znázorněn na obrázku 6.2. Celá CGP mřížka je pak znázorněna na obrázku 6.2. Na každý ze vstupů CGP mřížky je přivedena jedna hodnota z dopravního detektoru. Data použitá v této práci byla naměřena pomocí indučních smyček. Na každý vstup bude proto přivedena jedna hodnota z jednotlivých indučních smyček, které měřili dopravní data na dané silniční sekci. Tyto vstupní uzly CGP mřížky budou využity jako vstupy hlavních uzlů CGP a bude z nich vypočítána dojezdová doba. Ta bude jediným výstupem CGP mřížky.

1. VSTUP	2. VSTUP	KONSTANTA	FUNKCE
----------	----------	-----------	--------

Obrázek 6.1: Uzel implementované CGP mřížky



Obrázek 6.2: CGP mřížka

# Kapitola 7

## Implementace

Aplikace pro predikci dojezdových dob k této práci je napsána v programovacím jazyce C++. Vstupem programu jsou CSV soubory s naměřenými dopravními daty popsány v kapitole 8.1. Evoluce se pokusí z dat, která jsou v nich uložena, najít matematické závislosti v datech z jednotlivých senzorů tak, aby ve výsledku dávaly požadovou dojezdovou dobu. Program přijímá na vstupu několik parametrů:

- h Vypíše nápovědu
- g **generation** Položka generation udává maximální počet generací evoluce
- p **population** Velikost populace
- s **size** Rozměry grafu ve tvaru [vyska]x[sirka]
- l **lback** Parametr L-Back
- m **mut** Počet mutací, které se provedou na jedinci v jedné generaci
- d **dist** Maximalni odchylka fitness funkce nejlepšího jedince
- o **out** Nazev výstupního souboru
- sc **sigma** Parametr sigma pro fitness funkci „skóre řešení“
- se **section** Název sekce, pro kterou se bude predikovat

Příklad spuštění programu může být například s těmito parametry:

```
./cgpRegression -g 50000 -p 5 -s 3x4 -d 0.001 -m 10 -o cgOutput -se tripId-1
```

Prvním krokem je zpracování parametrů příkazového řádku. Pokud jsou parametry v pořádku, jsou načtena data ze souboru „trainSet.txt“, který je uložen v adresáři pojmenovaném stejně jako daná dopravní sekce, pro kterou mají být predikovány dojezdové doby. Data z „trainSet.txt“ slouží pro natrénování CGP. Následně se spustí evoluce, která se pokusí nalézt co nejlepší matematický vztah mezi hodnotami z různých senzorů. Po ukončení evoluce je načten soubor „testSet.txt“, na jehož datech se provede otestování predikce dojezdových dob.

Program je implementován ve 13 souborech:

cmdParam.[cpp—hpp] - třída pro zpracování parametrů příkazového řádku. Konstruktor třídy CMDparam přijímá jako parametry argumenty, se kterými byla aplikace spuštěna. Argumenty jsou rozparsovány a zkontrolována jejich správnost. Jednotlivé parametry je poté možné získat příslušnými přístupovými metodami.

dataImportExport.[cpp—hpp] - funkce pro načítání vstupních dat a export výsledků.

dateAndTime.[cpp—hpp] - třída pro práci s datem a časem.

evolution[cpp—hpp] - funkce provádějící běh evoluce. Nejdůležitější funkcí je „evolution()“. Ta implementuje samotný algoritmus kartézského genetického programování, který byl popsán v kapitole 3. Na počátku je opakovaným voláním funkce „createRandomChromosome()“ vytvořena náhodná populace. Jednotliví jedinci populace jsou ohodnocováni pomocí funkce „fitnessFunction()“. V průběhu evoluce jsou v pravidelných intervalech vypisovány průběžné výsledky evoluce. Tyto průběžné výsledky zahrnují fitness nejlepšího jedince a matematický předpis, který daný jedinec představuje. Velikost těchto intervalů je dána konstantou „TEMPOUTNUMBER“. Evoluce probíhá tak dlouho, dokud opakovaně volaná funkce „isEndOfEvolution()“ nezjistí, že bylo dosaženo některé z ukončujících podmínek evoluce.

chromosome[.cpp—hpp] - třída zapouzdřující práci s genotypem a CGP mřížkou. Po vytvoření nové instance třídy Chromosome a nastavení parametrů CGP (rozměry mřížky, počet mutací, L-back parametr) je potřeba zavolat metodu „init()“. Tato metoda zajistí vytvoření náhodné mřížky jedince. Zavoláním metody „mutate()“ dojde k náhodné mutaci jedince. Pro výpočet dojezdové doby slouží metoda „calculate()“. Ta přijímá pole s hodnotami ze senzorů a vrací dojezdovou dobu, která byla vypočítána pomocí rekurzivního průchodu CGP grafem, jak bylo ukázáno v kapitole 3.2.

kg.[cpp—hpp] - generátor pseudonáhodných čísel.

main.cpp - samotná aplikace, jenž volá příslušné metody a funkce.

## Kapitola 8

# Experimentální vyhodnocení

### 8.1 Popis testovacích dat

Za testovací data byla zvolena databáze Research Data Exchange<sup>1</sup>. Je to databáze dopravních dat volně dostupná pro široké spektrum využití. Sběr dopravních dat byl zajištěn organizací WSDOT<sup>2</sup>.

#### 8.1.1 Údaje ze senzorů

Jako primární množina dat jsou využívány údaje z indukčních smyček. Data jsou ukládána v 5-ti minutových intervalech. Tato úroveň agregace poskytuje dostatečně detailní pohled k identifikování nastávajícího zvyšování úrovně provozu. WSDOT agreguje 5-ti minutová data ve svém řídicím centru a zaznamenává je z dat získávaných z dopravních detektorů v 20-ti sekundových intervalech.

#### Popis dat

5-ti minutová data jsou organizována ve 3 tabulkách - Cabinets, Loops a LoopData. Každá z těchto tabulek je poskytována v oddělených CSV souborech. Tabulka Cabinet popisuje geografickou polohu měřících zařízení. Tabulka Loop popisuje specifické typy dat, která jsou shromažďována danou smyčkou a údaje o typu vozovky, jejím směru (na sever, jih, východ či západ) a typu smyčkového detektoru. Tabulka LoopData obsahuje aktuální data o obsazenosti a objemu dopravy, která jsou ukládána z každé smyčky v 5-ti minutových intervalech. Vzhledem k množství dat jsou aktuální data ze smyček separována v CSV souborech po 6 měsících. Tyto data nejsou nijak speciálně seřazena. Dokonce jsou ještě dělena do souborů po jednotlivých měsících, protože každý CSV soubor je příliš velký, aby byl načten exceleem. Tedy data pro srpen jsou uložena v souboru LoopData\_Aug.csv.

#### Testování kvality

Proces ověřování kvality 5-ti minutových intervalů začíná s umístěním jednoduché chybové značky dobré/špatné/podezřelé/chybějící pro každou hodnotu ze smyčky jako výsledek agregace 20-ti sekundové informace o objemu a obsazenosti dopravy na dopravní komunikaci, které zjišťují kontrolori v oblasti. Technik pak reviduje 5-ti minutová data s

---

<sup>1</sup><https://www.its-rde.net/>

<sup>2</sup><http://www.wsdot.com>

pomocí řady automatizovaných kontrol kvality dat. Výsledek lidského přezkoumání je schopen přepsat původní značku kvality novou značkou kvality indikující, že data pro smyčku jsou nevhodná pro použití ve většině analýzách. Tato ruční produkce značek kvality je spojena s každou smyčkou na celý měsíc (to znamená, že manuální produkce značek přepíše všechny původní značky pro celý měsíc). Nová chybová značka indikuje, zda by měla být data použita v automatických analýzách. Ne všechna data v daném měsíci, pro místo označené jako „nevhodné“, musí být ve skutečnosti „nevhodné“. Podobně ne všechna data v měsíci označená jako „dobrá“ jsou vždy správná. Například pro měsíc květen je 275940 5-ti minutových opakování s obsazeností komunikace větší než 100% a mají chybnou značku. Zbývajících 8951 dat (3% výskytů) nemají chybovou značku. Některá z nich jsou validní data. K jinému případu dochází na vyvýšených úsecích vozovky, kde nebyla manuální kontrola kvality aplikována a kde automatický proces neidentifikoval chybná data. Tato data by měla být použita s opatrností. Skutečnost, že originální značka nebyla přepsána, indikuje, že agregace dat na místě jsou považovány za validní pro použití k analýzám. Chyby v datech ze senzorů shromážděné a přenesené do hlavního dopravního centra mohou být způsobeny různými hardwarovými chybami. Tyto chyby mohou být klasifikovány jako permanentní anebo dočasné. Permanentní chyby vznikají, když zařízení přestane fungovat a nedodává validní data dokud není opraveno nebo vyměněno.

### 8.1.2 Dojezdové doby

Dojezdové doby jsou vypočítány pro 5-ti minutová data, která odpovídají testům kvality popsaným výše. Dojezdové doby jsou k dispozici pro 8 odlišných úseků. Nejdůležitějším předpokladem pro výpočet co nejpřesnějších dojezdových dob je mít co nejkvalitnější data. Protože dojezdové doby jsou počítány algoritmem trajektorie vozidel, proces identifikace a odpovědnosti za nevalidní data shromážděná podél koridoru má významný dopad na vypočítané dojezdové doby. Protože sensory WSDOT jsou od sebe vzdáleny asi 800m, předpokládá se, že měření rychlosti v místě detektoru jsou validní pro vzdálenost na půl cesty před detektorem a půl cesty k dalšímu detektoru a představuje jenom mírnou chybu ve výpočtu dojezdových dob. Nicméně přítomnost nevalidních dat - například ztráta všech dat ze smyčky, protože senzor se rozbije - může vytvořit případy, ve kterých jsou vzdálenosti dost velké a odhad rychlosti interpolací předchozích a následujících rychlostí může způsobit velkou chybu. Pokud se poškodily jenom jedna nebo dvě smyčky, předpokládá se pro výpočet dojezdových dob, že rychlosti jsou rovny průměrné rychlosti zbývajících smyček na dané sekci. Když musí být z výpočtu dojezdových dob odstraněny všechny smyčky na dané sekci, protože jejich data jsou chybná, je rychlost interpolována z předcházejících a následujících lokací, aby byla pokryta chybějící dálniční sekce.

### Popis dojezdových dob

5-ti minutové dojezdové doby jsou uloženy ve 3 oddělených tabulkách. Každá tabulka je poskytována v separátním CSV souboru. Spolu s dojezdovými dobami jsou v tabulkách uloženy i polohy, kde byly naměřeny, v jakou dobu, počátek a konec měřeného úseku atd.

### Testování kvality dat

Primární proces testování kvality dat sestává s obdobných procedur jako pro data z jednotlivých smyček. Pro dojezdové doby se ale provádí i několik dalších kontrol. Tyto kontroly zahrnují vykreslování průměru 80 a 95 percentilů dojezdových dob pro úsek podle času dne,

aby se mohly najít dojezdové doby, které nejsou reálné (například dojezdové doby nikdy nedosáhnou stavu úplně volného úseku, dokonce ani pozdě v noci). Tyto grafy jsou porovnávány s grafy vytvořenými předchozí rok. Když tyto dva grafy tvoří výrazně odlišné vzory, jsou provedeny další analýzy, aby bylo zjištěno, zda jsou tyto změny v dojezdových dobách způsobeny aktuálními změnami ve stavu dopravy anebo je to výsledek chybných dat. Tyto dodatečné kontroly vyžadují vytvoření časově závislého diagramu, který ilustruje odhadnutou rychlost v každé lokaci po celém úseku během dne. Přezkoumání těchto diagramů dovoluje analytikům zjistit, zda změny v dopravě ve specifické lokaci indikují chybu dat anebo na ně má vliv nějaká specifická událost. Například extrémní počasí může způsobit velmi extrémní změny v dojezdových dobách v jediném dni. I jeden jediný velmi pomalý den může znamenat změnu v průměrných dojezdových dobách. [1]

### 8.1.3 Úprava dat pro predikci

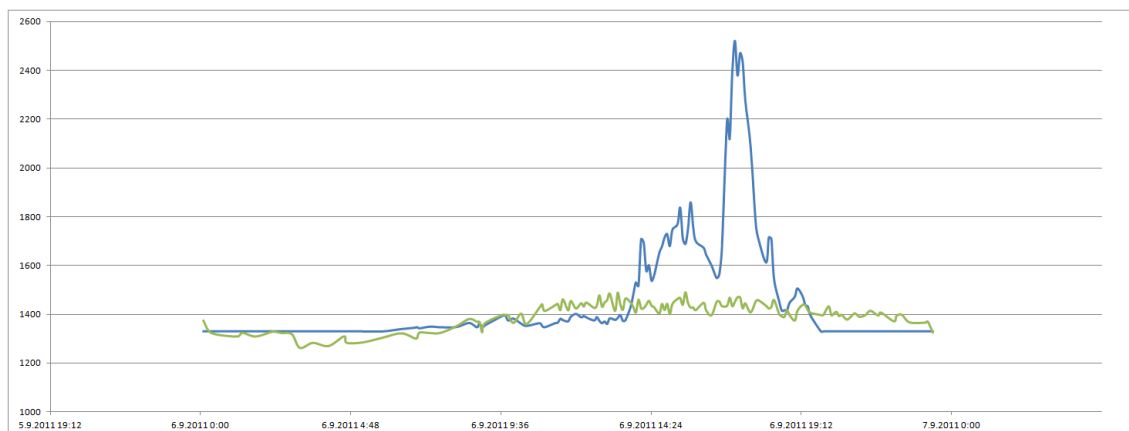
Pro tuto práci byla vybrána data naměřená na několika dopravních úsecích v Seattlu v září roku 2011. Data z každé dopravní sekce byla upravena pro snazší manipulaci. Byla použita jenom data, která byla označena jako „správná“. Pro každou sekci byl vytvořen CSV soubor, kde první sloupec obsahuje datum a čas naměřeného údaje. Ve druhém sloupci je dojezdová doba automobilů v daném čase. Další sloupce obsahují hodnoty ze senzorů na dané sekci. Z této databáze dat ze senzorů a k nim odpovídajícím dojezdovým dobám byly nakonec vytvořeny 2 soubory. V souboru „trainSet.txt“ jsou data z první poloviny měsíce a slouží pro natrénování CGP. V souboru „testSet.txt“ jsou data z druhé poloviny měsíce a slouží k otestování kvality predikce.

## 8.2 Nastavení evoluce

Po prvotním spuštění predikce dojezdových dob nebyly výsledky nijak zvlášť dobré, jak je vidět na obrázku 8.1. CGP se příliš nedařilo nalézt vhodný předpis, který by odpovídal skutečné dojezdové době. Při prvním spuštění byly použity následující parametry evoluce:

- počet jedinců populace - 5
- velikost CGP mřížky - šířka 8, výška 7
- počet mutací jedince v jedné generaci - 10
- ukončující podmínky evoluce:
  - nalezení ideálního (přesného) řešení
  - nejlepší řešení v aktuální generaci má fitness menší než 0,001
  - dosažení maximálního počtu generací 50000
- funkce CGP mřížky jsou uvedeny v tabulce 8.1
- jako fitness funkce byl použit vzorec 4.2 s tím rozdílem, že výsledek tohoto vzorce byl podělen počtem testovacích bodů, čímž se získala průměrná odchylka jednoho bodu

Na obrázku 8.1 je na x-ové ose čas v jednom dni, na y-ové ose dojezdová doba. Modrou barvou je skutečně naměřená dojezdová doba a zeleně je dojezdová doba predikovaná pomocí CGP. Pro ostatní sekce a dny byly výsledky podobně nekvalitní. Dalším cílem proto bylo zjistit, jaké parametry evoluce jsou pro predikci dojezdových dob nejlepší.



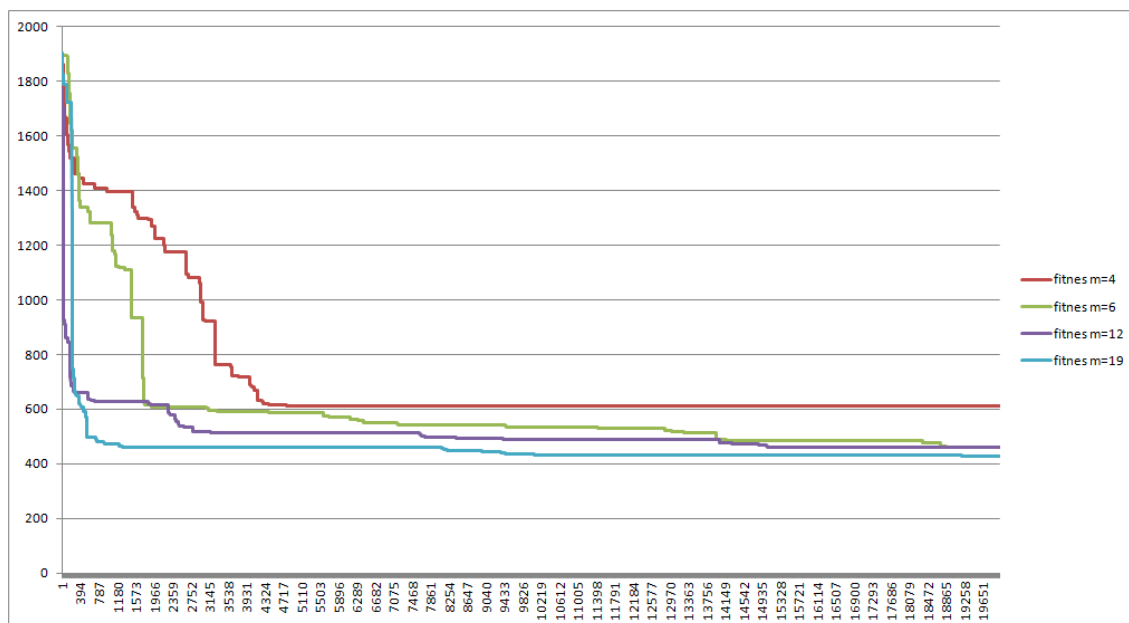
Obrázek 8.1: První pokus predikce dojezdových dob. Na x-ové ose se nachází čas během dne, na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečná dojezdová doba, zelenou barvou predikovaná dojezdová doba.

Funkce	Definice funkce
F1	$IN1 + IN2$
F2	$IN1 - IN2$
F3	$IN * IN2$
F4	$\text{if}(IN2 == 0, 0) 0, 0 \text{ else } IN1/IN2$
F5	KONSTANTA
F6	$\sin(IN1)$
F7	$\cos(IN1)$
F8	$\tan(IN1)$
F9	$\sqrt{ IN1 }$
F10	$ IN1 $
F11	$\text{if}(IN1 == 0, 0) 0, 0 \text{ else } \ln IN1$
F12	$e^{IN1}$

Tabulka 8.1: Funkce použité v CGP mřížce

### 8.3 Mutace

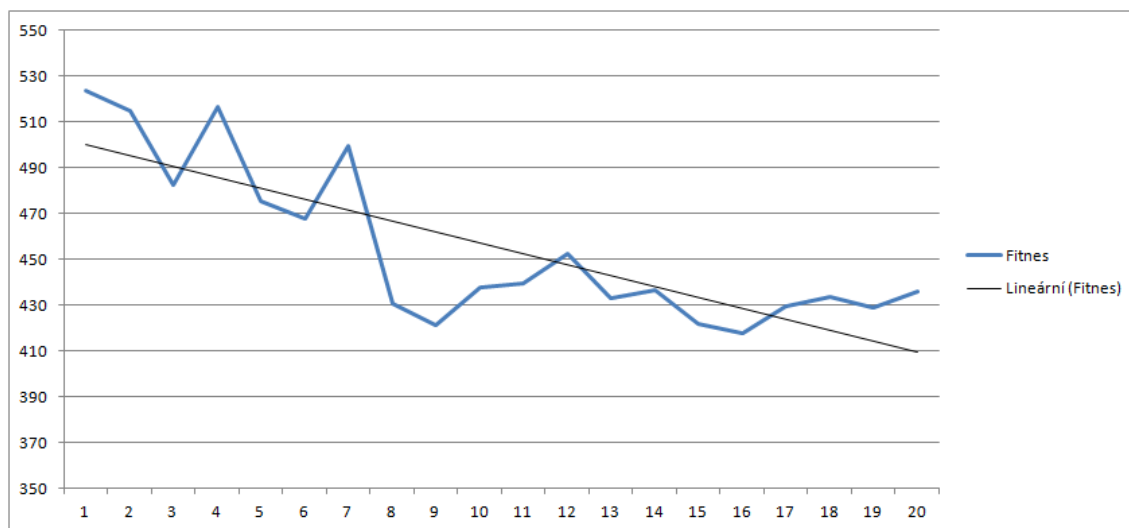
První pokusy byly zaměřeny na zjištění, jaký vliv má na evoluci počet mutací. Byly provedeny 4 opakování evoluce s různým nastavením počtu mutací a pro každé nastavení vybrán nejlepší výsledek. Na obrázku 8.2 jsou vidět výsledky konvergence fitness funkce pro evoluce, které měly nastaven počet mutací 4, 6, 12 a 19. Na x-ové ose jsou generace, ve kterých probíhala evoluce a na y-ové ose je hodnota fitness funkce v každé generaci. Z grafu je patrné, že čím vyšší je počet mutací, tím rychleji fitness konverguje. Zároveň je vidět, že čím větší je počet mutací, tím lepší výsledek je evoluce schopna nalézt. V tabulce 8.2 jsou vidět jednotlivé hodnoty fitness pro grafy z obrázku 8.2. Obdobnou informaci je možné vyčíst z obrázku 8.3, na kterém jsou graficky zobrazeny hodnoty z tabulky 8.3. Obrázek 8.3 znázorňuje závislost počtu mutací na velikosti fitness. Na x-ové ose je počet mutací a na y-ové ose je zprůměrovaná fitness z několika opakovaných běhů evoluce pro daný počet mutací. Na samotné křivce to není až tak patrné. Když se ale grafem proloží přímka, je na ní vidět, že má sestupný charakter, což naznačuje závislost mezi počtem mutací a kvalitou nalezeného řešení. Samotná křivka nemá příliš sestupný charakter, protože z důvodů obrovské časové náročnosti nebylo možné provést dostatečně mnoho opakování evoluce. Při dalších opakování evoluce by se křivka začala stále více podobat proložené přímce. Kvalita výsledků se ale zvyšuje se vzrůstajícím počtem mutací jenom do určité hranice. Na obrázku 8.4 je znázorněna fitness v závislosti na počtu mutací, ale oproti grafu na obrázku 8.3 jde o dlouhodobější průběh. Graf ukazuje, že se vzrůstajícím počtem mutací se fitness zlepšuje, ale od určitého počtu mutací se zlepšující tentence zastavuje a s přibývajícím počtem mutací evoluce nachází dokonce horší výsledky. Mezním počtem mutací je v tomto konkrétním případě 16 mutací. To odpovídá zmutování 14% z celého chromozomu.



Obrázek 8.2: Srovnání různých nastavení mutací. Na x-ové ose jsou generace, ve kterých probíhala (prvních 20000 generací). Na y-ové ose je hodnota fitness.

Generace	4 mutace	6 mutací	12 mutací	19 mutací
0	1864	1902	1773	1904
250	1520	1554	663	744
500	1426	1339	659	593
750	1410	1282	628	479
1000	1397	1282	628	474
1250	1397	1117	628	466
1500	1397	932	628	462
1750	1298	625	628	459
2000	1223	609	616	459
2250	1176	609	616	459
2500	1174	609	539	459
2750	1081	609	536	459
3000	990	609	516	459
3250	922	593	515	459
3500	763	592	515	459
3750	720	592	515	459
4000	718	591	515	459
4250	632	591	515	459
4500	617	586	515	459
4750	617	586	515	459
5000	611	586	515	459

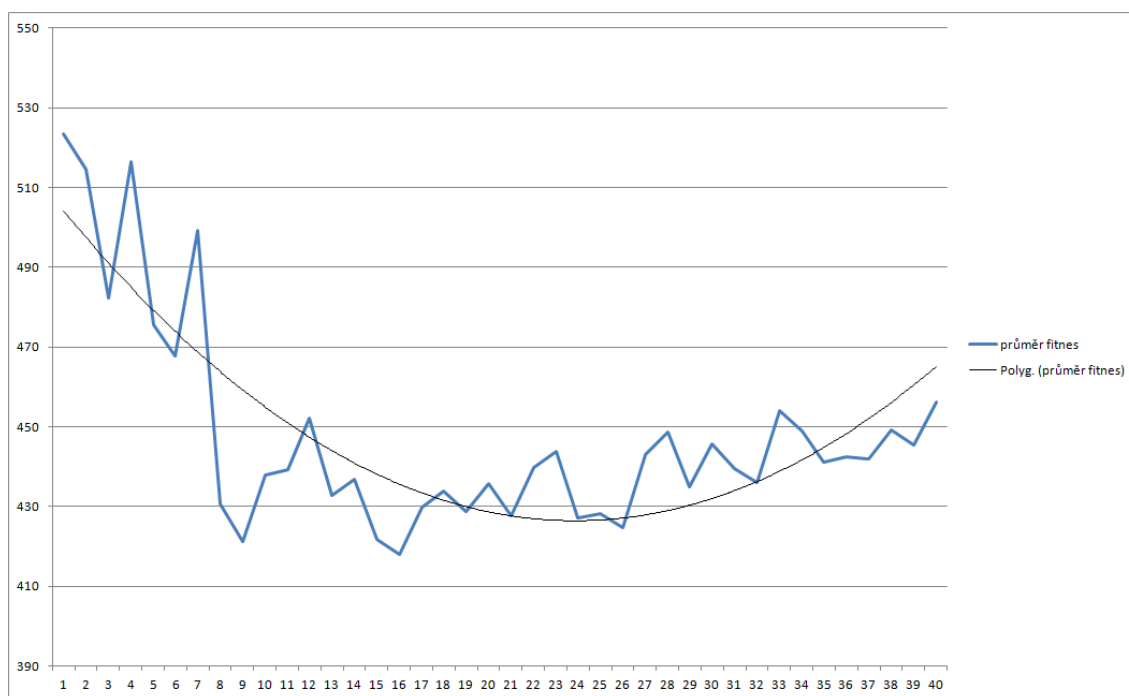
Tabulka 8.2: Srovnání konvergence fitness pro různá nastavení mutací během prvních 5000 generací evoluce.



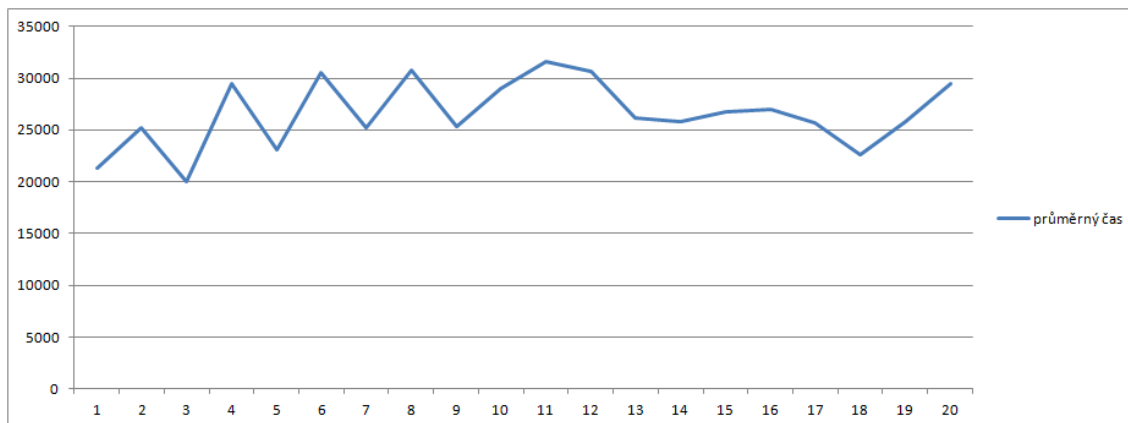
Obrázek 8.3: Závislost fitness na počtu mutací. Na x-ové ose je počet mutací, který byl aplikován na jedince během jedné generace. Na y-ové ose je průměrná fitness výsledného řešení.

Počet mutací					Průměrná fitness
1	621,024	466,549	509,072	497,386	523,5
2	508,384	435,864	466,329	648,05	514,7
3	468,203	543,657	420,654	496,379	482,2
4	522,533	571,248	538,273	433,664	516,4
5	448,718	444,75	554,345	454,528	475,6
6	507,5	426,909	476,229	460,633	467,8
7	427,883	488,461	519,795	561,063	499,3
8	422,222	418,965	437,881	443,853	430,7
9	426,524	429,364	404,914	423,623	421,1
10	413,36	469,396	412,382	456,635	438,0
11	423,305	502,138	415,758	415,781	439,2
12	433,698	467,525	448,812	458,778	452,2
13	443,413	441,804	429,692	416,48	432,8
14	407,905	433,192	457,731	448,142	436,7
15	456,951	405,621	410,447	414,524	421,9
16	423,95	409,668	435,809	402,634	418,0
17	416,358	423,11	427,381	452,116	429,7
18	419,911	445,911	435,68		433,8
19	430,287	430,02	427,101	427,923	428,8
20	446,07	416,074	437,826	443,309	435,8

Tabulka 8.3: Závislost fitness na počtu mutací



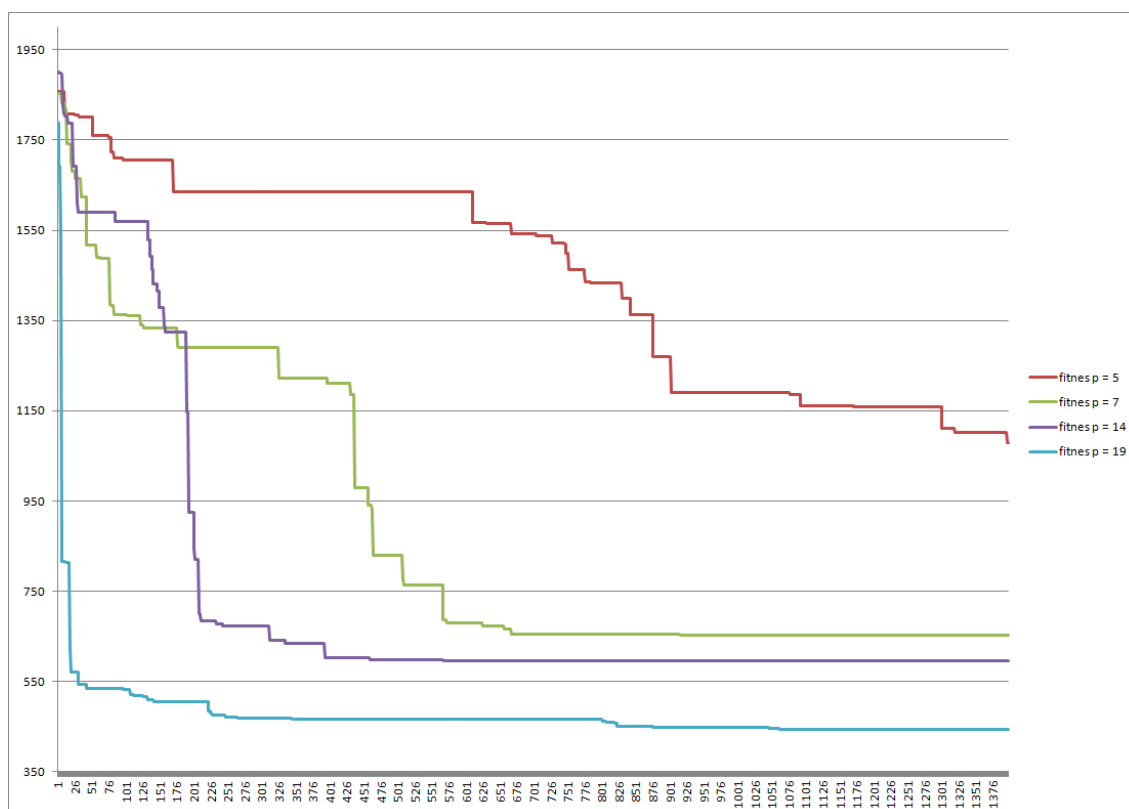
Obrázek 8.4: Závislost fitness na počtu mutací. Na x-ové ose je počet mutací, který byl aplikován na jedince během jedné generace. Na y-ové ose je průměrná fitness výsledného řešení.



Obrázek 8.5: Doba evoluce pro různý počet mutací. Na x-ové ose je počet mutací, který byl aplikován na jedince během jedné generace. Na y-ové ose je čas v sekundách potřebný pro běh evoluce.

## 8.4 Velikost populace

Po mutacích byly provedeny experimenty s velikostí populace. V tabulce 8.4 jsou hodnoty konvergence fitness ze začátku evoluce pro nejlepší výsledky ze čtyř opakovaných spuštění evoluce pro různé velikosti populací, konkrétně pro populace o velikostech 5, 7, 14 a 19 jedinců. Hodnoty z tabulky 8.4 jsou znázorněny na obrázku 8.6. Na x-ové ose jsou generace, ve kterých probíhala evoluce a na y-ové ose je hodnota fitness v každé generaci pro jednotlivé evoluce s různými velikostmi populací. Z grafu je patrné, že čím větší je velikost populace, tím rychleji fitness konverguje. Je to způsobeno tím, že čím větší populace je, tím větší stavový prostor je prohledáván a je tak mnohem větší šance na nalezení lepšího řešení. Tato výhoda má ale jednu velkou nevýhodu. Při zvětšování velikosti populace výrazně narůstá čas potřebný pro běh evoluce, což je vidět na grafu 8.8. Na x-ové ose jsou různě velké populace a na y-ové ose je čas potřebný k vykonání evoluce. Graf znázorňuje, že čím větší populace je, tím větší jsou časové nároky na dobu běhu evoluce. To je výrazná nevýhoda oproti mutacím. Vyšší počet mutací měl, stejně jako u velikosti populace, pozitivní vliv na běh evoluce, ale bez negativního vlivu na dobu běhu evoluce, jak je znázorněno na obrázku 8.5. V tabulce 8.5 jsou fitness jednotlivých běhů evoluce a na obrázku 8.7 je graf srovnání průměrných hodnot fitness pro jednotlivé velikosti populací. Na x-ové ose je počet populací, a na y-ové ose je průměrná velikost fitness. Křivka má sestupný charakter, což ukazuje, že čím větší populace je k dispozici, tím lepšího výsledku evoluce dosahuje. Sestupná tendence je opět vidět lépe při proložení přímkou. To, že větší počet jedinců populace má za následek lepší výsledek evoluce je dáno možností prohledávat větší stavový prostor a je tak větší šance pro nalezení lepšího řešení problému.



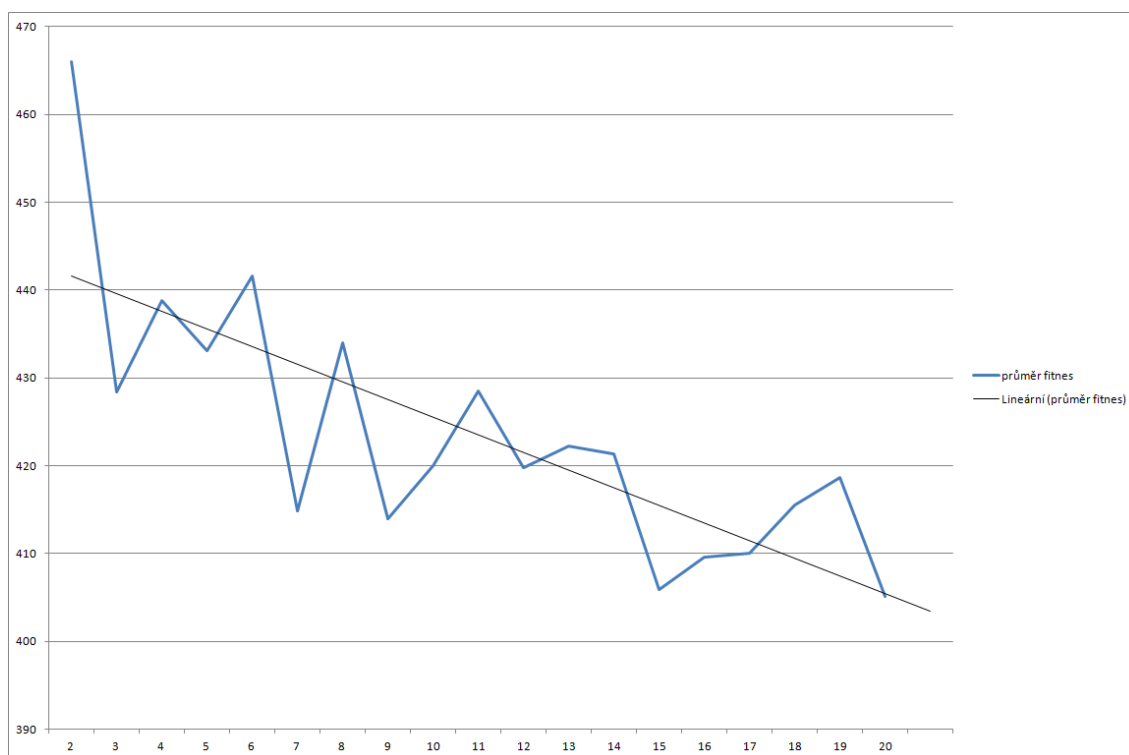
Obrázek 8.6: Srovnání evoluce pro různé velikosti populace. Na x-ové ose jsou generace, ve kterých evoluce probíhala (prvních 1500 generací). Na y-ové ose je hodnota fitness.

<b>Generace</b>	<b>5 jedinců</b>	<b>7 jedinců</b>	<b>14 jedinců</b>	<b>19 jedinců</b>
0	1857,75	1854,22	1900,12	1789,7
60	1760,4	1490,8	1590,76	535,301
120	1706,43	1359,58	1569,68	519,004
180	1634,51	1290,42	1323,68	505,524
240	1634,51	1290,42	677,292	475,479
300	1634,51	1289,94	674,156	469,335
360	1634,51	1221,22	635,349	466,027
420	1634,51	1211,63	603,092	466,027
480	1634,51	830,213	597,537	466,027
540	1634,51	764,081	597,537	466,027
600	1634,51	679,57	595,177	466,027
660	1563,91	667,083	595,177	466,027
720	1538,18	654,902	595,177	466,027
780	1435,86	653,789	595,177	466,027
840	1400,08	653,789	595,177	449,635
900	1270,58	653,789	595,177	447,26
960	1191,31	653,115	595,177	447,26
1020	1191,31	653,115	595,177	447,26
1080	1186,81	653,115	595,177	444,197
1140	1161,93	653,115	595,177	444,197
1200	1159,75	653,115	595,177	444,197

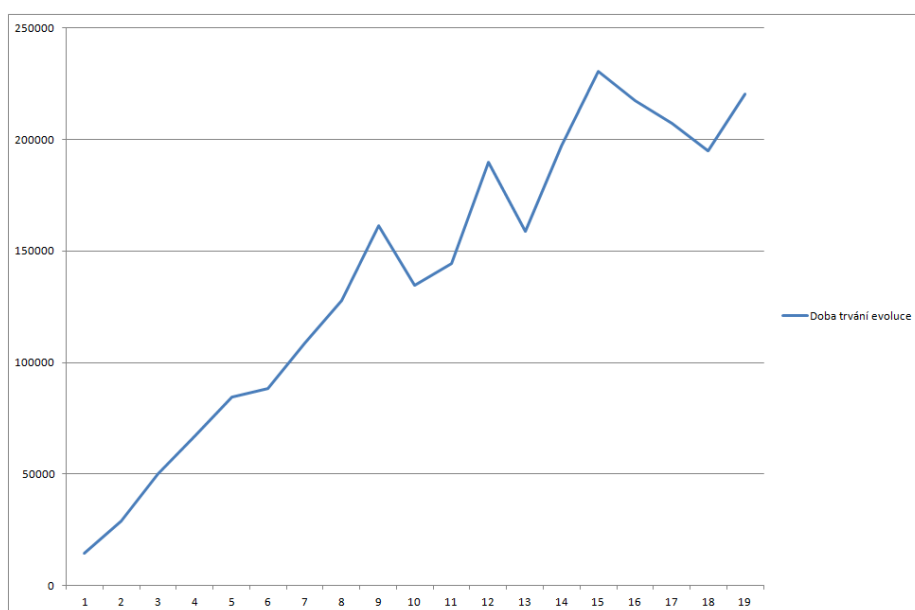
Tabulka 8.4: Srovnání konvergence fitness pro různé velikosti populace

<b>Počet jedinců</b>					<b>Průměrná fitness</b>
2	447,836	465,222	425,412	525,73	466,05
3	412,107	430,618	456,351	414,497	428,39
4	436,256	441,971	445,967	430,977	438,79
5	447,277	450,86	434,892	399,578	433,15
6	421,147	426,311	432,382	486,699	441,63
7	419,137	412,847	419,43	408,068	414,87
8	439,463	402,843	415,418	478,125	433,96
9	425,732	399,113	411,086	419,847	413,94
10	418,155	430,422	407,274	424,198	420,01
11	400,588	419,549	461,365	432,528	428,51
12	426,477	458,726	418,356	375,351	419,73
13	409,991	437,69	415,395	425,928	422,25
14	420,126	427,911	421,275	416,097	421,35
15	392,727	413,211	409,577	407,9	405,85
16	408,489	407,861	395,26	426,664	409,57
17	416,64	396,83	413,805	412,715	410,00
18	428,775	407,202	404,272	421,869	415,53
19	409,348	419,862	429,518	416,09	418,70
20	395,624	401,291	413,199	410,191	405,08

Tabulka 8.5: Hodnoty fitness pro jednotlivé velikosti populací



Obrázek 8.7: Závislost fitness na velikosti populací. Na x-ové ose je počet jedinců populace. Na y-ové ose je hodnota fitness.



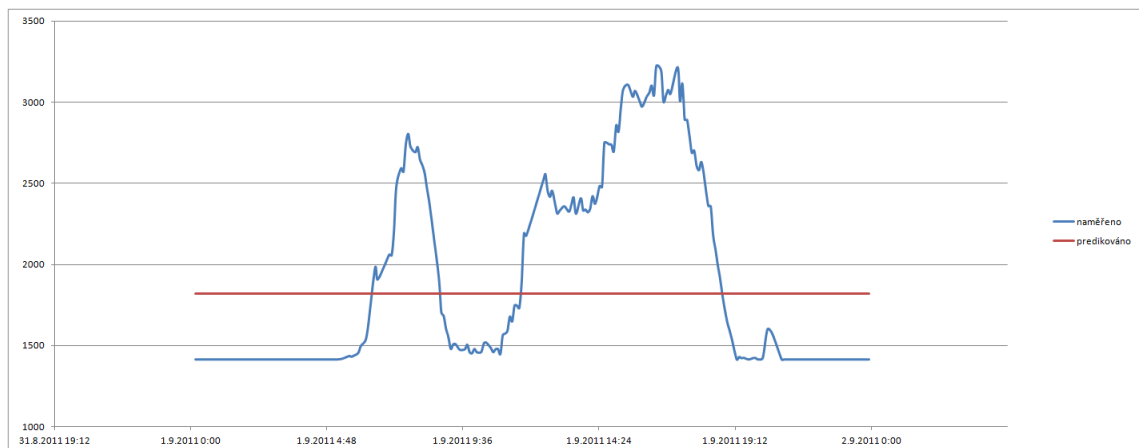
Obrázek 8.8: Doba evoluce různě velikých populací. Na x-ové ose je počet jedinců populace. Na y-ové ose je čas v sekundách potřebný pro běh evoluce.

## 8.5 Rozměry CGP mřížky

Byly provedeny experimenty s různými nastaveními rozměrů CGP mřížky. Jedním z výsledků bylo, že nemá smysl používat mřížku o šířce 2. Pro takto malou mřížku byla výsledkem vždy jen konstantní funkce. Vůbec nebyly brány v úvahu hodnoty z jednotlivých senzorů. V tabulce 8.6 jsou vidět nalezené fitness pro mřížky s šířkou 2 a různými výškami. Nejlepší nalezené řešení představovalo výraz  $(-79 * -23)$ . To je konstantní funkce o hodnotě 1817. Graf této funkce je vidět na obrázku 8.9 spolu se skutečně naměřenou dojezdovou dobou. Pro mřížky o šířce 3 a 4 (neboli se třemi či čtyřmi sloupci) platí podobný závěr. Opět byly často nacházeny konstantní funkce, ale v některých případech už byly pro výpočet dojezdové doby brány za vstupní hodnoty i data ze senzorů.

výška CGP mřížky	hodnoty fitness			
2	570,962	571,792	570,962	571,792
6	570,961	571,013	570,989	570,961
10	571,033	571,028	570,962	570,962
15	570,989	570,989	570,962	713,414
20	570,962	571,013	570,961	570,962

Tabulka 8.6: Hodnoty fitness pro CGP mřížku šířky 2 a různé výšky



Obrázek 8.9: Nejlepší řešení pro mřížku šířky 2. Na x-ové ose je čas během dne. Na y-ové ose je dojezdová doba. Modrou barvou je skutečně naměřená dojezdová doba, hnědou barvou je predikovaná dojezdová doba pro CGP mřížku se dvěma sloupci.

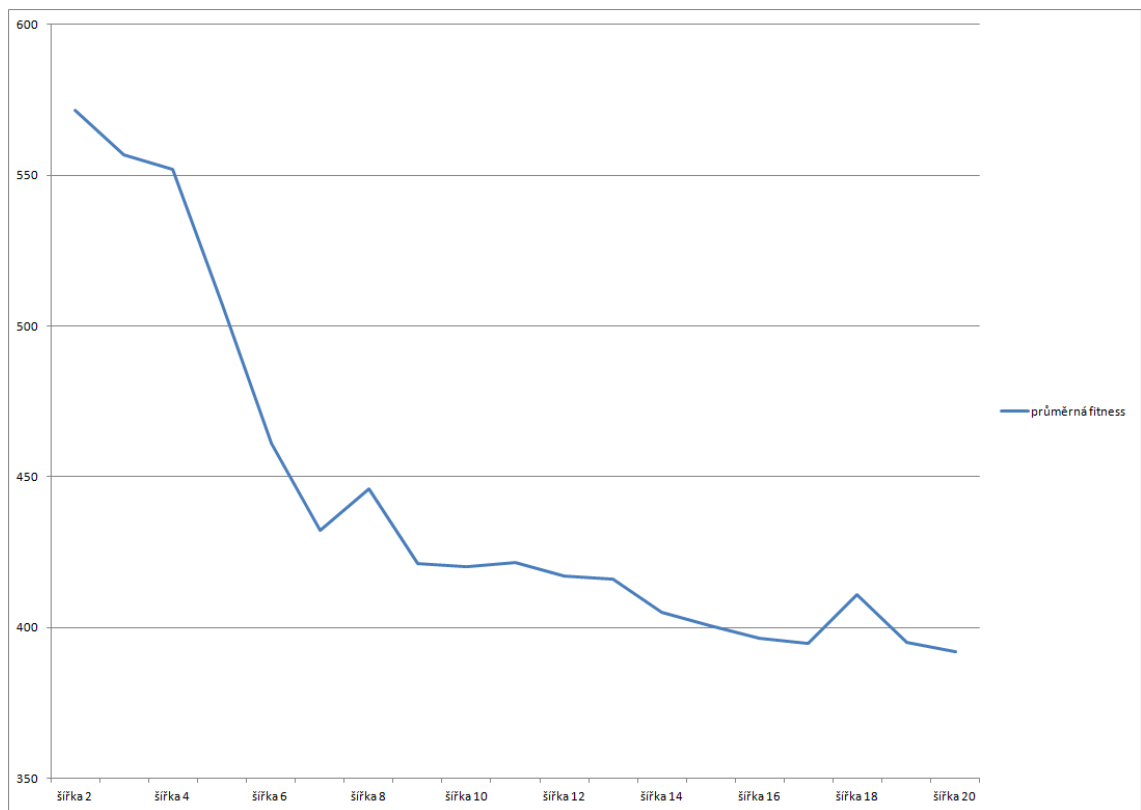
Při vzrůstajícím počtu sloupců CGP mřížky se ale výsledky začaly vylepšovat. V tabulce 8.7 jsou hodnoty fitness pro CGP mřížku se 4 řádky a různým počtem sloupců. V prvním řádku tabulky je počet sloupců CGP mřížky. V dalších sloupcích jsou fitness získané z jednotlivých běhů evoluce a v posledním sloupci je průměr těchto fitness. Na obrázku 8.7 jsou pak průměrné hodnoty fitness vyneseny do grafu. Na grafu je vidět sestupná tendence. Čím vyšší je počet sloupců CGP mřížky, tím lepší řešení je evoluce schopna nalézt. Po mřížku o šířce 7 se fitness vylepšuje poměrně rychlým tempem. S dalším narůstáním počtu sloupců ale pokles fitness (a tedy nalezení lepšího jedince) pomalu ustává. Jak bylo řečeno v kapitole 8.3, nejlepších výsledků dosahuje evoluce, pokud je mutováno 14% chromosomu. Této hodnoty je dosaženo právě u mřížky o rozměrech 4 řádky a 7 sloupců. To je dohromady 28 uzlů CGP mřížky. Každý uzel je zakódován čtyřmi celočíselnými hodnotami. Chromosom má tedy délku  $28 * 4 = 112$  genů. U testování velikosti CGP mřížky byl každý jedinec mutován vždy desetkrát, což je u mřížky o rozměrech 4 řádky a 7 sloupců přes 11% chromosomu, tedy téměř ideální hodnota. Aby se i u větších mřížek fitness vylepšovala rychleji, bylo by třeba spolu s narůstající velikostí mřížky zvyšovat i počet mutací, díky čemuž by se zlepšily vlastnosti evoluce. Stejně jako na výslednou fitness, má šířka CGP mřížky pozitivní vliv i na rychlost konvergence fitness. Na obrázku 8.11 je vidět konvergence fitness během prvních generací evoluce. Konkrétně jde o evoluce, které mají CGP mřížku o výšce 4 a šířkách 5, 7, 16 a 19. Jak graf ukazuje, čím má mřížka více sloupců, tím rychleji fitness konverguje. Bohužel tyto výhody vzrůstajícího počtu sloupců mají velkou nevýhodu v době provádění evoluce. Obrázek 8.12 ukazuje, jak se vzrůstajícím počtem sloupců CGP mřížky stoupá časová náročnost evoluce. Na x-ové ose se nachází počet sloupců CGP mřížky. Na y-ové ose je čas v sekundách potřebný pro běh evoluce. Nárůst času potřebného pro evoluci sice není kvadratický, ale je horší než lineární. Na obrázcích 8.13 a 8.14 je vidět závislost fitness na počtu řádků CGP mřížky. Na x-ové ose je počet řádků CGP mřížky a na y-ové ose je hodnota fitness. Při proložení přímkou je vidět, že s narůstajícím počtem řádků CGP mřížky hodnota fitness mírně klesá. Narozdíl od narůstajícího počtu sloupců nemá počet řádků vliv na časovou náročnost evoluce. To ukazuje obrázek 8.15. Na x-ové ose jsou počty řádků CGP mřížky a na y-ové ose je čas v sekundách potřebný pro běh evoluce.

Počet sloupců	Výsledné hodnoty fitness při jednotlivých testech				Průměrná fitness
2	571,792	570,962	571,792	571,792	571,58
3	570,989	570,962	570,962	513,613	556,63
4	494,75	570,989	570,989	570,962	551,92
5	571	494,338	425,933	536,78	507,01
6	434,695	447,686	474,724	486,978	461,02
7	430,641	454,623	412,101	432,115	432,37
8	419,786	459,723	486,697	418,157	446,09
9	434,477	412,818	410,603	426,52	421,10
10	413,455	420,272	408,988	438,47	420,30
11	425,739	418,297	404,705	437,514	421,56
12	407,905	412,502	419,059	429,457	417,23
13	415,781	408,966	425,146	414,478	416,09
14	405,213	399,785	405,147	409,743	404,97
15	404,438	398,685	408,128	390,578	400,46
16	384,072	396,842	395,202	409,385	396,38
17	381,454	408,358	395,419	394,064	394,82
18	406,963	432,03	416,961	387,883	410,96
19	383,406	400,207	401,579		395,06
20	393,898	387,034	407,838	378,653	391,86

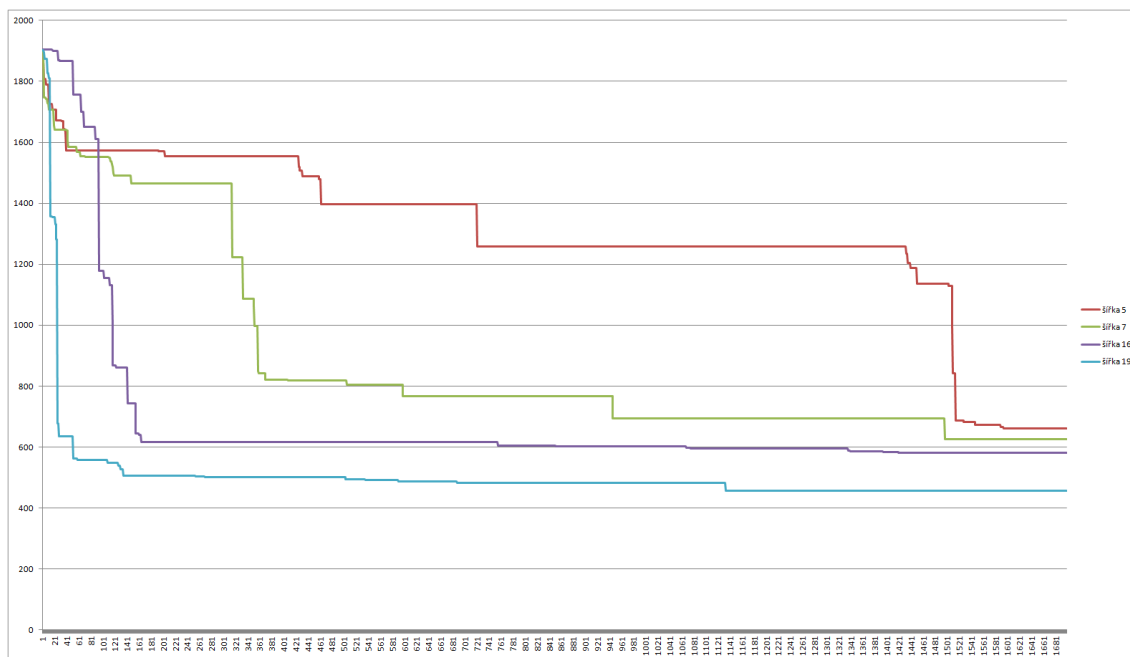
Tabulka 8.7: Fitness pro CGP mřížku o výšce 4 a různou šířku

počet řádků					průměrná fitness
2	448,979	451,603	426,851	432,764	440,05
3	422,242	454,86	450,336	393,413	430,21
4	413,455	420,272	408,988	438,47	420,30
5	431,333	416,454	415,611	457,997	430,35
6	419,56	408,308	418,278	435,931	420,52
7	412,69	411,035	436,404	434,896	423,76
8	414,966	457,26	401,758	426,984	425,24
9	417,136	434,856	429,929	390,444	418,09
10	430,49	405,327	415,422	410,272	415,38

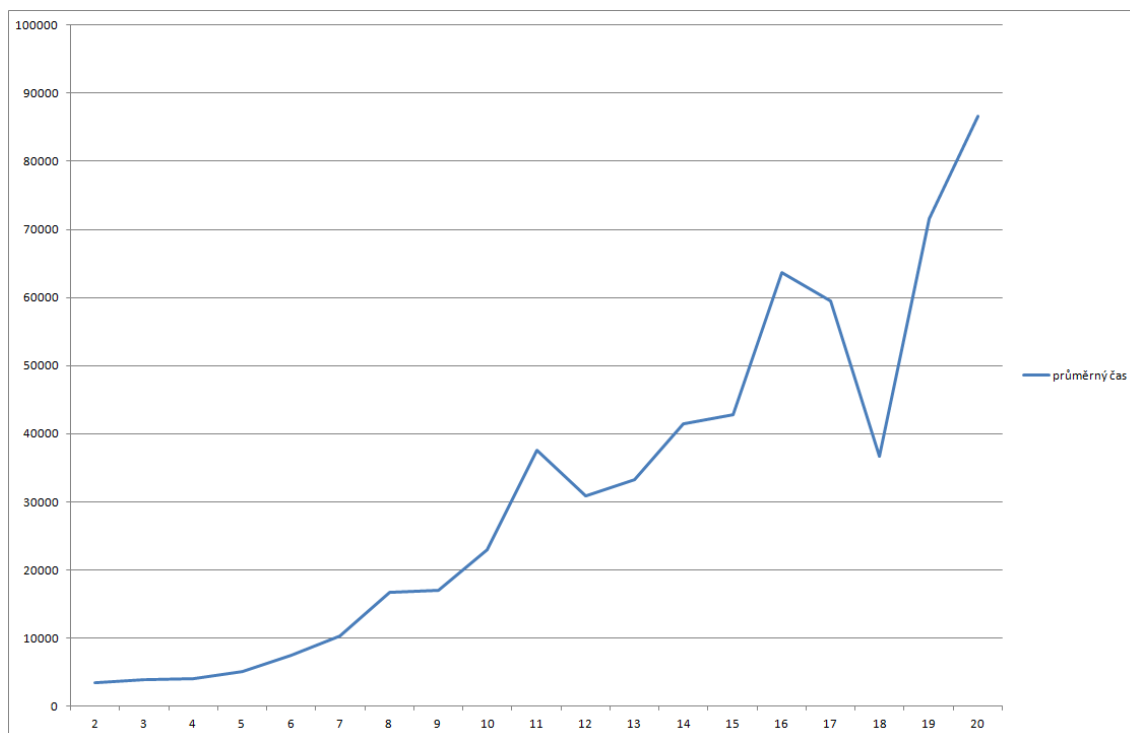
Tabulka 8.8: Fitness pro CGP mřížku o šířce 10 a různé výšky



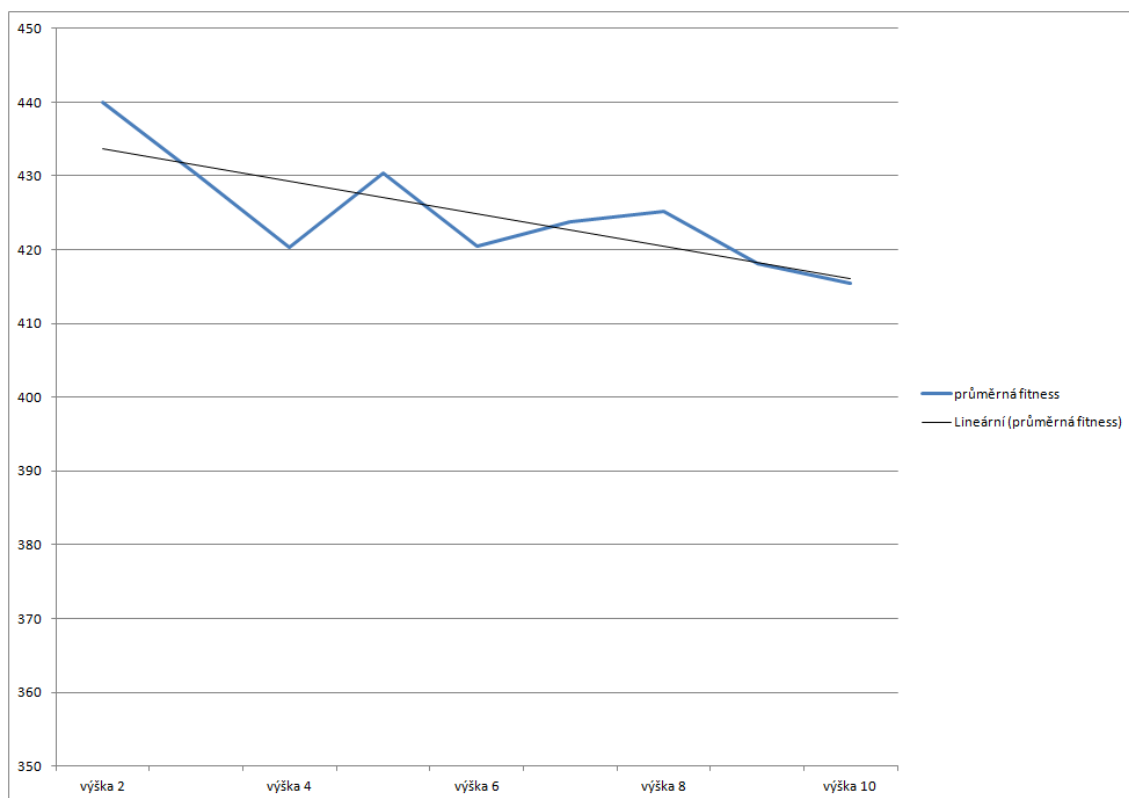
Obrázek 8.10: Fitness pro CGP mřížku o počtu řádků 4 a různém počtu sloupců. Na x-ové ose je šířka CGP mřížky (počet sloupců). Na y-ové ose je průměrná hodnota fitness.



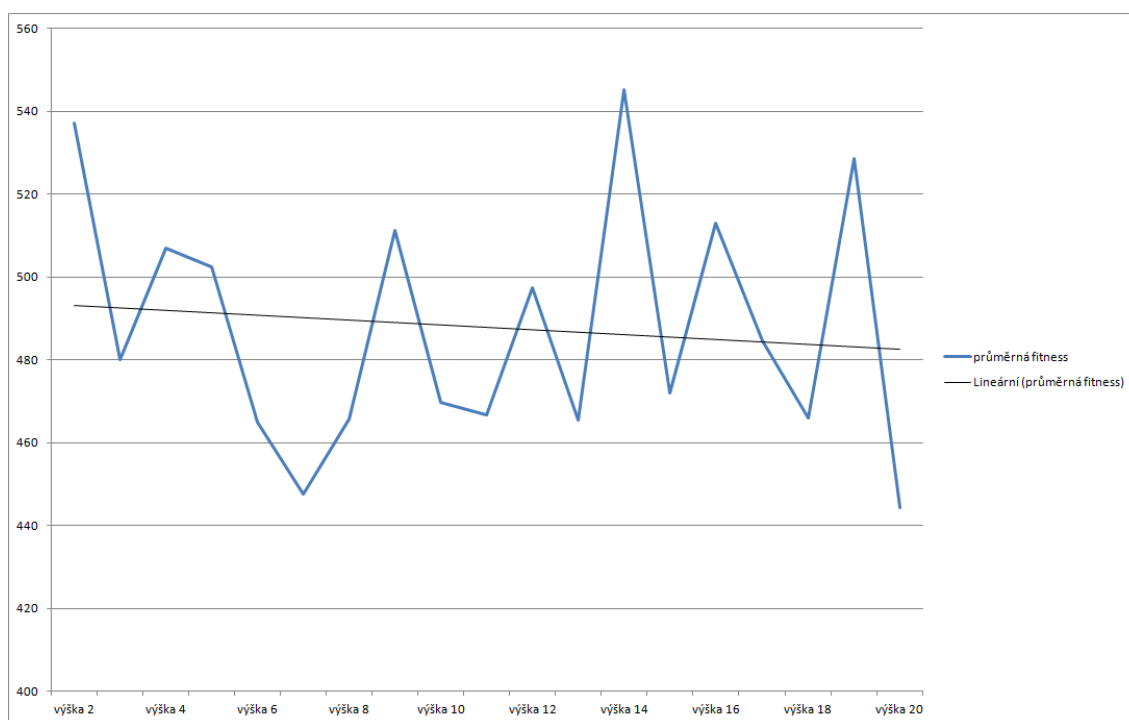
Obrázek 8.11: Konvergence fitness pro různý počet sloupců CGP mřížky. Na x-ové ose jsou generace, ve kterých evoluce probíhala. Na y-ové ose je hodnota fitness.



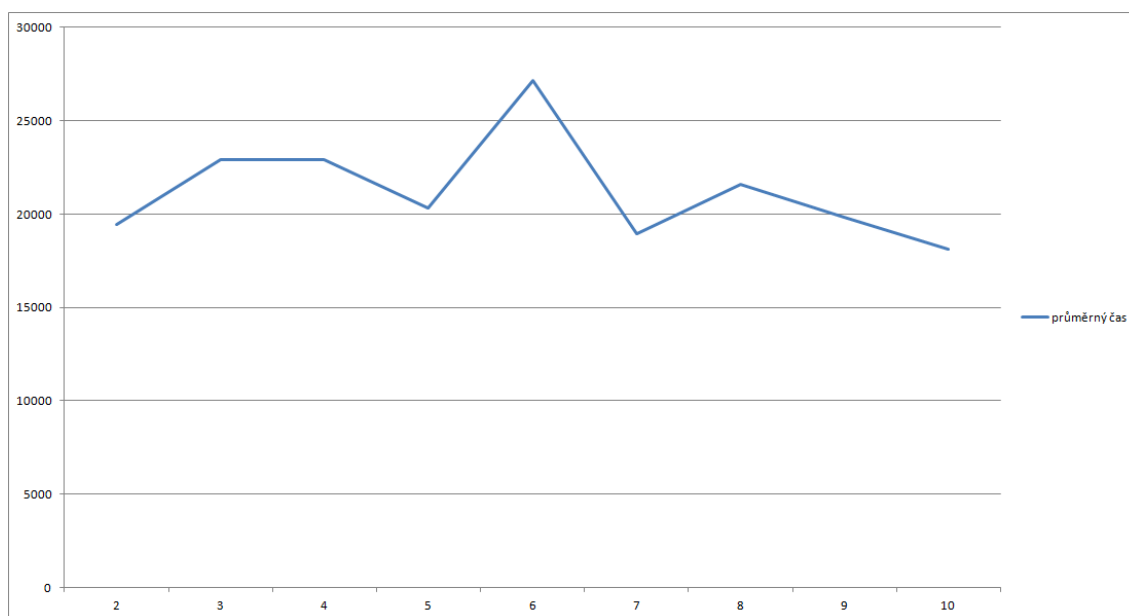
Obrázek 8.12: Vzdávající časová závislost na počtu sloupců CGP mřížky. Na x-ové ose je šířka CGP mřížky (počet sloupců). Na y-ové ose je čas potřebný pro běh evoluce.



Obrázek 8.13: Závislost fitness na výšce CGP mřížky pro mřížku o šířce 10. Na x-ové ose je počet řádků CGP mřížky (výška mřížky). Na y-ové ose je hodnota fitness.



Obrázek 8.14: Závislost fitness na výšce CGP mřížky pro mřížku o šířce 5. Na x-ové ose je počet řádků CGP mřížky (výška mřížky). Na y-ové ose je hodnota fitness.



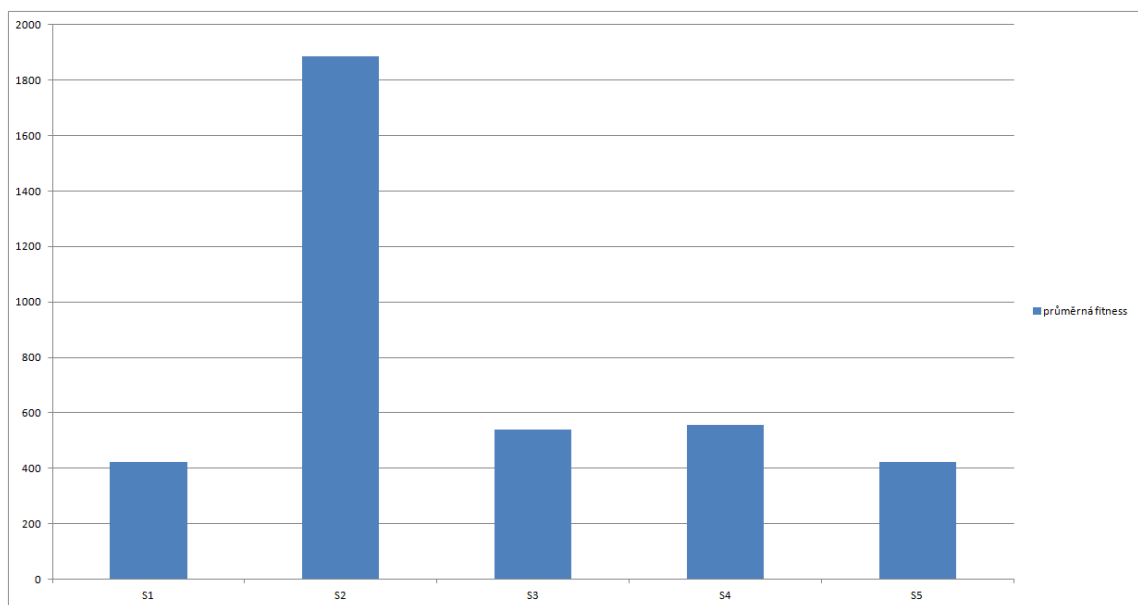
Obrázek 8.15: Časová závislost na vzrůstajícím počtu řádků CGP mřížky. Na x-ové ose je výška CGP mřížky (počet řádků). Na y-ové ose je čas potřebný pro běh evoluce.

## 8.6 Použité funkce

Pro zjištění, které z používaných funkcí (uvedených v tabulce 8.1) jsou vhodné a které jsou nevhodné pro predikci dojezdových dob, bylo vytvořeno 5 skupin funkcí:

- S1 - všechny funkce z tabulky 8.1
- S2 - goniometrické funkce, v tabulce 8.1 označeny jako F6, F7 a F8
- S3 - základní aritmetické funkce, v tabulce označeny jako F1, F2, F3 a F4
- S4 - základní aritmetické funkce + goniometrické funkce, neboli S2 + S3 dohromady
- S5 - všechny funkce z tabulky 8.1 kromě goniometrických funkcí

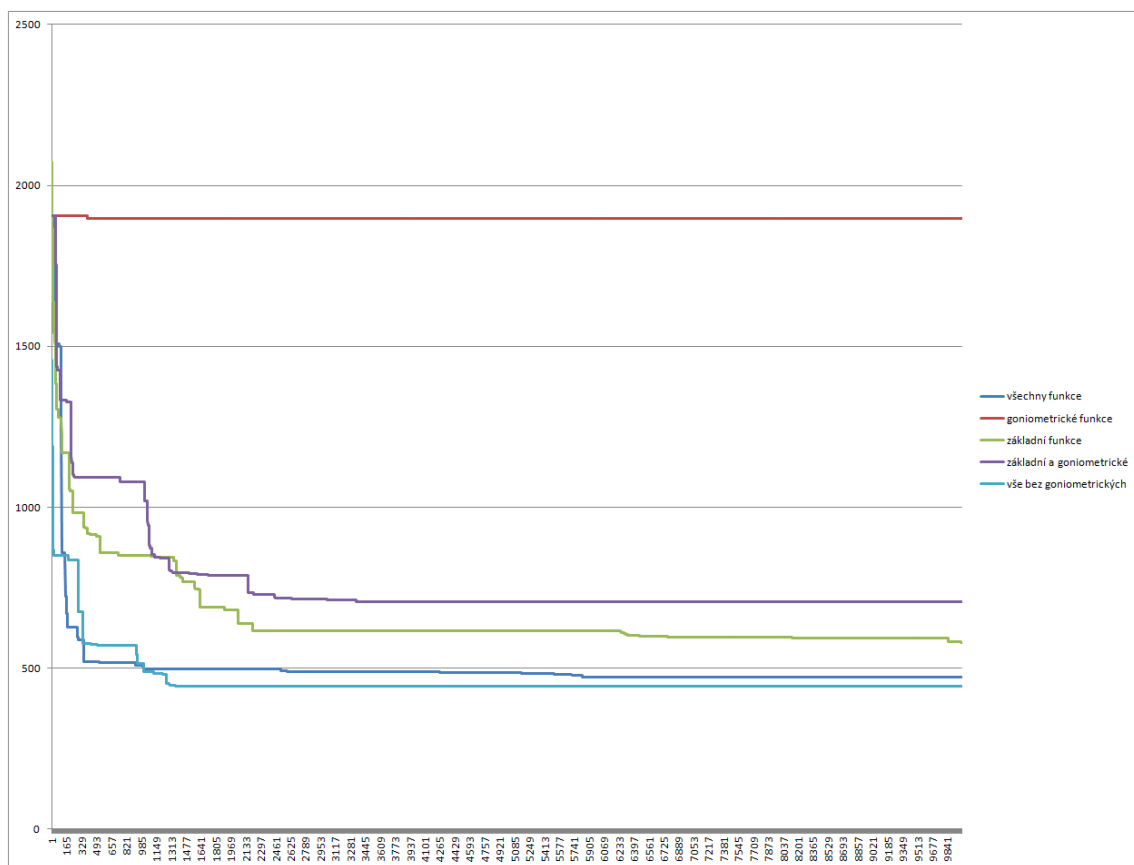
Pro každou tuto skupinu funkcí bylo vyzkoušeno, jakých výsledků je s nimi evoluce schopna dosáhnout. V tabulce 8.9 jsou výsledné hodnoty fitness pro evoluce s různě použitými funkcemi. Na obrázku 8.16 jsou pak znázorněny průměrné fitness u těchto skupin funkcí. Je vidět, že samotné goniometrické funkce jsou naprosto nevhodné pro predikci dojezdových dob. Samotné aritmetické funkce jsou výrazně lepší, ale i ony samy nestačí k nalezení matematického výrazu popisujícího dojezdové doby. Při kombinaci aritmetických a goniometrických funkcí se pak projeví neefektivita goniometrických funkcí, které zpomalují nalezení nejlepšího možného výrazu. To je vidět ještě lépe na obrázku 8.17. Zde je vidět konvergence fitness pro různé skupiny funkcí. Nejlepších výsledků dosahuje evoluce při použití všech funkcí CGP mřížky. Při použití množiny funkcí S1 a S5 je dosahováno prakticky stejných výsledků. To je dáno buď tím, že použití goniometrických funkcí je vhodné pouze ve spojení s dalšími funkcemi anebo se při použití velké množiny funkcí negativní efekt goniometrických funkcí příliš neprojeví. Toto by mohlo být obsahem dalších experimentů.



Obrázek 8.16: Průměrná fitness s různými použitými funkcemi. Na x-ové ose jsou různé skupiny funkcí. Na y-ové ose je hodnota fitness.

Skupina funkcí					Průměrná fitness
S1	422,625	448,388	417,875	401,599	422,62175
S2	1887,61	1883,61	1887,61	1879,04	1884,4675
S3	536,613	520,474	542,282	564,033	540,8505
S4	554,262	544,932	567,569	554,141	555,226
S5	432,233	419,512	426,342	413,762	422,96225

Tabulka 8.9: Fitness pro různé použité funkce



Obrázek 8.17: Konvergence fitness pro různé použité funkce CGP mřížky. Na x-ové ose jsou generace ve kterých probíhala evoluce. Na y-ové ose je hodnota fitness.

## 8.7 Závěrečné experimenty

Z dosavadních experimentů vyplynuly vhodné parametry pro evoluci. Predikce dojezdových dob dosahuje lepších výsledků v případě, že populace obsahuje velké množství jedinců. Stejně tak čím má CGP mřížka větší počet sloupců, tím je predikce přesnější. Bohužel tyto dva parametry nemohou být nastaveny příliš vysoké, protože mají výrazný vliv na dobu trvání evoluce. Větší počet řádků CGP mřížky má na lepší výsledek evoluce jen malý vliv. S velikostí CGP mřížky souvisí i počet mutací. Nejlepších výsledků dosahuje evoluce, pokud počet mutací odpovídá 14% délky chromozomu. Lepších výsledků predikce dosahuje i tehdy, když má k dispozici rozsáhlejší množinu funkcí, ale se sporným přínosem goniometrických funkcí. Z těchto výsledků byly odvozeny „lepší“ parametry pro predikci dojezdových dob. Pro predikci dojezdových dob byla nakonec navržena populace o osmi jedincích. Velikost CGP mřížky byla stanovena na 15 sloupců a 15 řádků. Z toho vyplývá počet mutací jedince v jedné generaci na  $\frac{15 \cdot 15 \cdot 4}{100} \cdot 14 = 126$ . Jako množina funkcí byly zvoleny všechny funkce z tabulky 8.1 kromě goniometrických funkcí. Bylo by výhodnější zvolit větší velikost populace i větší šířku CGP mřížky, ale z časových důvodů byly zvoleny uvedené parametry. Evoluce byla čtyřikrát spuštěna nad daty ze všech 8 dopravních sekcí. Z výsledných predikovaných dojezdových dob byly vybrány nejlepší výsledky. V tabulce 8.10 je ukázka výsledků predikce dojezdových dob pro sekci „tripId-3“ z každé hodiny v den 12. září 2011. Na obrázku 8.18 jsou uvedená data vynesena do grafu. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba. 12. září patří do první poloviny měsíce, a proto se jedná o trénovací data, na kterých se evoluce snažila „naučit“ predikovat dojezdové doby. V tabulce 8.11 je opět ukázka predikce dat ze sekce „tripId-3“, tentokrát ale pro den 19. září 2011. Tento den patří do druhé poloviny měsíce, proto se jedná o predikci nad testovacími daty. Na obrázku 8.19 jsou data z tabulky 8.11 vynesena do grafu. U obou grafů je vidět, že se evoluce snaží vystihnout různé změny v dopravní situaci, ale nedaří se jí zachytit přesný okamžik nástupu této změny ani její intenzitu. Výraz 8.1 je výsledný matematický výraz predikující dojezdovou dobu pro sekci tripId-3.

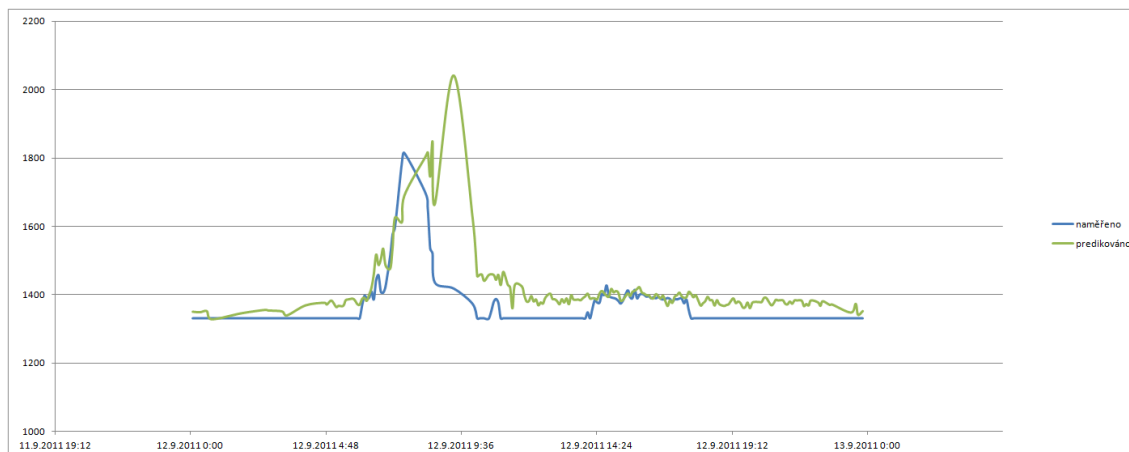
$$\begin{aligned}
 & ((\log(\text{abs}((\text{abs}("X1233.occupancy") * \exp("X906.occupancy")))) - \\
 & \quad ((\text{abs}("X704.occupancy" + "X475.occupancy")) / \\
 & \quad \quad \log(\text{abs}("X861.occupancy")))) + \\
 & \quad (((("X887.occupancy" * "X1101.intensity") / \\
 & \quad ("X1101.occupancy" * "X1115.occupancy")) * \\
 & \text{abs}(\log(\text{abs}(\text{sqrt}(\text{abs}("X1080.occupancy")) * "X1080.intensity"))))) + \\
 & \quad ((\text{abs}(\log(\text{abs}("X861.occupancy")) / \\
 & \quad "X1101.intensity") * "X475.occupancy")) * \\
 & \text{abs}(\text{sqrt}(\text{abs}("X1040.occupancy" - "X887.intensity")))) + (24 * 57))
 \end{aligned} \tag{8.1}$$

Hodnoty „X1233“ a podobné jsou identifikátory různých senzorů. Na výsledném matematickém předpisu je vidět, že evoluce využívá rozmanité množství funkcí. Výraz obsahuje i bloat. Například  $\text{abs}(\log(\text{abs}(\dots)))$  či  $\text{abs}(\text{sqrt}(\text{abs}(\dots)))$ . Problém nepřesné predikce může být způsoben využitím jen malé množiny senzorů. Na sekci „tripId-3“ se nachází 20 různých senzorů a každý z nich měří 2 různé veličiny - intenzitu dopravy a obsazenost silniční komunikace. To je dohromady 40 různých hodnot popisujících dopravu na sekci „tripId-3“.

Evoluce ale pro predikci využila jenom 13 různých hodnot z 10 různých senzorů. Využití většího množství senzorů by nejspíš umožnilo použití větší CGP mřížky.

Čas	Skutečná dojezdová doba	Predikovaná dojezdová doba
0:45	1331,4	1328,4
1:50	1331,4	1346,38
2:40	1331,4	1356,03
3:25	1331,4	1339,54
4:45	1331,4	1376,71
5:30	1331,4	1383,54
6:40	1457,7	1487,19
7:35	1815,2	1689,86
8:20	1698	1803,9
9:20	1418,7	2041,4
10:20	1331,4	1458,77
11:30	1331,4	1428,97
12:35	1331,4	1391,58
13:20	1331,4	1389,49
14:20	1384,3	1389,55
15:20	1383,3	1384,95
16:20	1390	1390,49
17:30	1375,5	1391,14
18:30	1331,4	1383,82
19:30	1331,4	1375,59
20:35	1331,4	1369,84
21:30	1331,4	1383,49
22:25	1331,4	1381,1
23:35	1331,4	1373,64

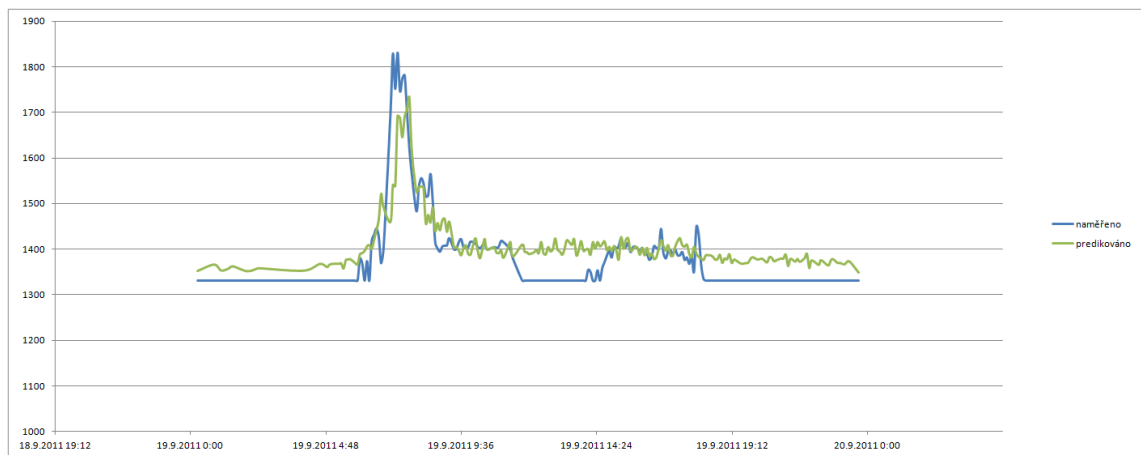
Tabulka 8.10: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-3 ze dne 12. září 2011



Obrázek 8.18: Výsledek predikce dojezdových dob pro den 12. září na sekci tripId-3. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba, zelenou barvou predikovaná dojezdová doba

Čas	Skutečná dojezdová doba	Predikovaná dojezdová doba
0:15	1331,4	1352,28
1:30	1331,4	1362,17
2:25	1331,4	1358,18
4:00	1331,4	1352,68
4:35	1331,4	1367,61
5:30	1331,4	1375,93
6:30	1431,7	1420,33
7:30	1774,1	1645,9
8:30	1565	1458,51
9:30	1415,4	1397,22
10:30	1400	1399,38
11:25	1379,9	1384,17
12:30	1331,4	1391,3
13:30	1331,4	1409,53
14:30	1331,4	1406,04
15:30	1414,2	1423,94
16:30	1403	1382,2
17:30	1376,6	1405,44
18:30	1331,4	1383,46
19:30	1331,4	1368,3
20:30	1331,4	1382,23
21:30	1331,4	1378,66
22:35	1331,4	1364,42
23:20	1331,4	1373,24

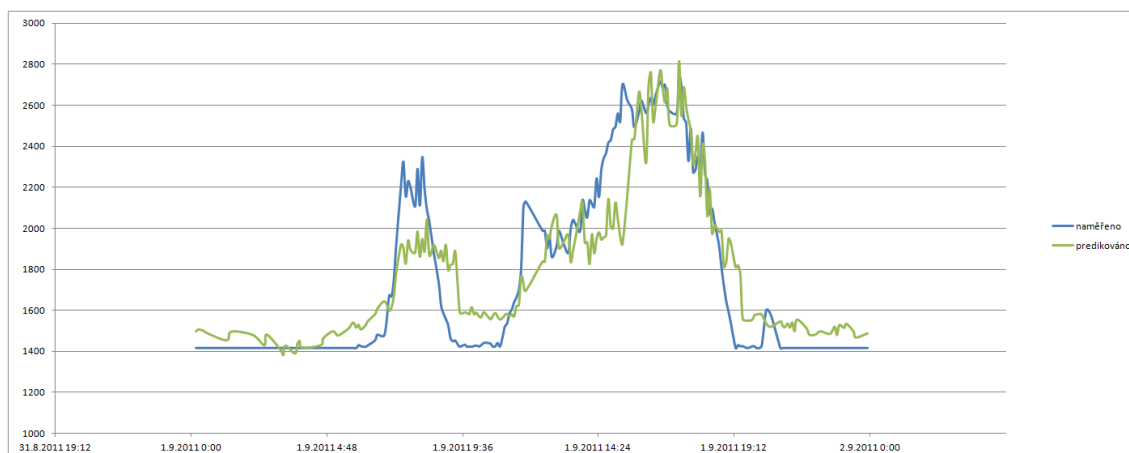
Tabulka 8.11: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-3 ze dne 19. září 2011



Obrázek 8.19: Výsledek predikce dojezdových dob pro den 19. září na sekci tripId-3. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba, zelenou barvou predikovaná dojezdová doba

O sekci „tripId-5“ platí obdobné výsledky. V tabulce 8.12 je ukázka predikovaných dat pro sekci „tripId-5“ ze dne 1. září 2011. Hodnoty jsou pak graficky znázorněny na obrázku 8.20. Tyto dojezdové doby jsou predikovány pro testovací data. Výsledky pro trénovací data (pro 19. září 2011) jsou uvedeny v tabulce 8.13 a graficky znázorněny na obrázku 8.13. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba. Na grafu je vidět že ani u dopravní sekce „tripId-5“ evoluce nezachytí přesný okamžik změny intenzity dopravy. Stejně jako na sekci „tripId-3“ i na sekci „tripId-5“ získává data o dopravě 20 senzorů a každý z nich měří intenzitu a obsazenost. Matematický výraz 8.2, který evoluce našla jako nejlepší pro predikci dojezdových dob, využívá ale jenom 10 unikátních hodnot ze 40 možných. Oproti výsledku ze sekce „tripId-3“ tento výraz neobsahuje bloat.

$$\begin{aligned}
 & ((((((X2445.intensity - X1607.intensity) + X1601.intensity) + \\
 & (X1572.occupancy / -35) * (X1701.occupancy / \\
 & \quad abs(X1802.intensity)))) + \\
 & ((X1431.occupancy + X1370.occupancy) + \\
 & (X2454.intensity - X1810.intensity))) - \\
 & (30 * (-49 - \log(abs(X1701.occupancy / abs(X1802.intensity))))))
 \end{aligned} \tag{8.2}$$



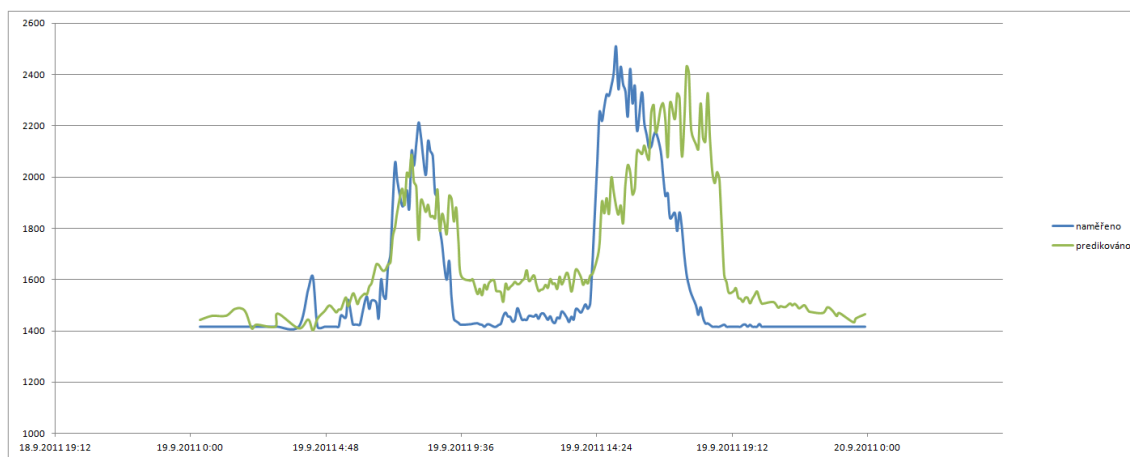
Obrázek 8.20: Výsledek predikce dojezdových dob pro den 1. září na sekci tripId-5. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba, zelenou barvou predikovaná dojezdová doba

Čas	Skutečná dojezdová doba	Predikovaná dojezdová doba
0:35	1416,6	1486
1:25	1416,6	1495
2:35	1416,6	1428,58
3:20	1416,6	1426,16
4:35	1416,6	1428,53
5:35	1416,6	1513,49
6:30	1454,6	1580,17
7:30	2324,4	1905,56
8:35	1872,2	1917,08
9:30	1422,9	1587,17
10:35	1437,9	1555,18
11:30	1664,2	1619,69
12:30	1988,4	1836,09
13:30	2041	1891,3
14:30	2283,7	1944,82
15:35	2578,4	2431,34
16:35	2715,3	2770,08
17:30	2511,3	2593,28
18:30	2032,4	2013,3
19:30	1426,2	1557,91
20:30	1576,5	1518,24
21:25	1416,6	1553,24
22:35	1416,6	1483,86
23:30	1416,6	1465,79

Tabulka 8.12: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-5 ze dne 1. září 2011

<b>Čas</b>	<b>Skutečná dojezdová doba</b>	<b>Predikovaná dojezdová doba</b>
0:45	1416,6	1457,44
1:35	1416,6	1485,35
2:20	1416,6	1423,22
3:05	1416,6	1466,53
4:30	1416,6	1446,73
5:30	1452,5	1529,5
6:35	1511,3	1659,07
7:30	1888,8	1954,22
8:30	2101,8	1845,58
9:30	1431,4	1739,04
10:30	1424,9	1560,71
11:30	1444,4	1590,46
12:30	1469,1	1562,67
13:30	1455,9	1552,31
14:30	2255,1	1741,41
15:30	2237,9	2046,28
16:30	2175,2	2173,28
17:30	1700,1	2207,13
18:30	1416,6	2017,19
19:30	1416,6	1524,52
20:20	1416,6	1506,05
21:30	1416,6	1496,54
22:35	1416,6	1491,18
23:35	1416,6	1447,29

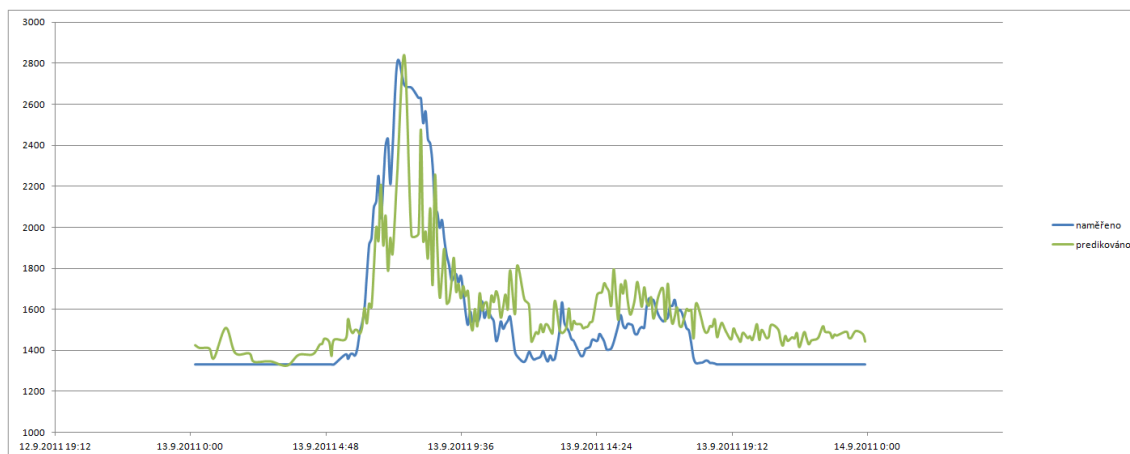
Tabulka 8.13: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-5 ze dne 19. září 2011



Obrázek 8.21: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-5 ze dne 19. září 2011. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba.

Výsledky pro trénovací data ze dne 13. září 2011 sekce „tripId-7“ jsou uvedeny v tabulce 8.14 a graficky znázorněny na obrázku 8.22. Pro testovací data ze dne 21. září 2011 jsou obdobné výsledky v tabulce 8.15 a graficky znázorněny na obrázku 8.23. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba. Na grafech je vidět, že pro sekci „tripId-7“ se evoluci dařilo o něco více zachytit výkyvy v dopravě než na sekcích „tripId-3“ a „tripId-5“, ale opět nejsou výsledky příliš kvalitní. Stejně jako na předchozích zmiňovaných sekcích i na sekci „tripId-7“ zaznamenává dopravu 20 senzorů a každý z nich zaznamenává 2 dopravní veličiny. Evoluce pro výpočet dojezdových dob na sekci „tripId-7“ našla matematický výraz 8.3. Tento výraz využívá jen 12 různých hodnot z 11 různých senzorů. Stejně jako u sekce „tripId-5“, ani tento výraz neobsahuje bloat.

$$\begin{aligned}
 & ((-17 * -83) + (((X906.intensity + \\
 & (X912.occupancy - X777.intensity)) * \\
 & \text{sqrt}(\text{abs}((X912.occupancy - X1115.occupancy) / \\
 & ((X1101.intensity / X475.intensity) * \text{sqrt}(\text{abs}(X1080.intensity)))) + \\
 & (X906.occupancy / X1080.intensity)))))) - \\
 & (\log(\text{abs}(\text{sqrt}(\text{abs}(X620.intensity)) - X704.occupancy) / \\
 & X1040.intensity))) * \\
 & (\log(\text{abs}(X887.occupancy * X906.occupancy))) + 73))
 \end{aligned} \tag{8.3}$$



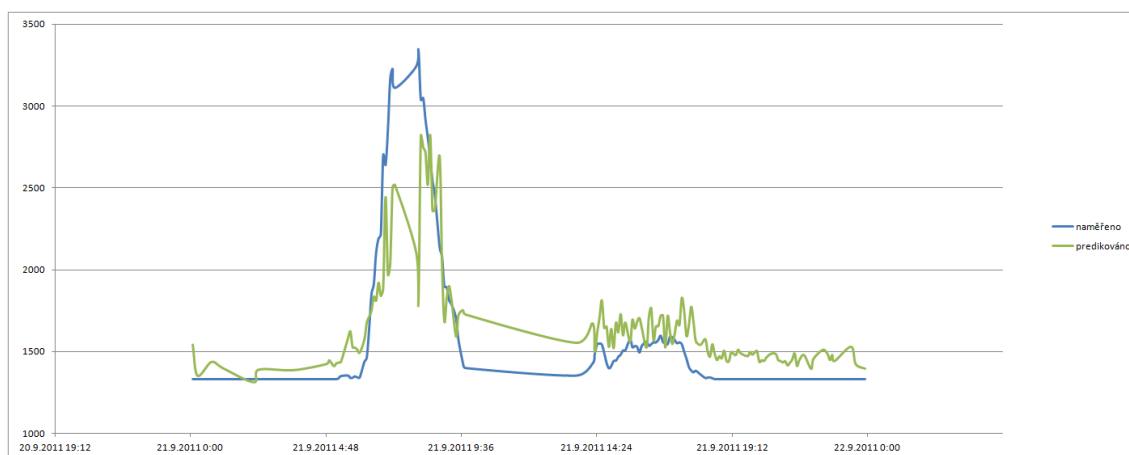
Obrázek 8.22: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-7 ze dne 13. září 2011. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba.

Čas	Skutečná dojezdová doba	Predikovaná dojezdová doba
0:20	1331,4	1411,42
1:35	1331,4	1387,32
2:50	1331,4	1346,69
3:50	1331,4	1377,98
4:35	1331,4	1428,33
5:30	1380,6	1455,18
6:30	2096,5	1799,03
7:35	2692,9	2836,14
8:30	2407,3	2091,62
9:30	1730,5	1724,16
10:30	1591,7	1633,29
11:30	1396,2	1576,36
12:30	1397	1488,86
13:30	1456,6	1500,83
14:30	1480,3	1680,8
15:30	1529,1	1641,42
16:35	1574,6	1662,05
17:35	1508,6	1600,51
18:30	1338,5	1515,89
19:30	1331,4	1442,22
20:30	1331,4	1466,67
21:30	1331,4	1484,46
22:30	1331,4	1490,66
23:35	1331,4	1494,44

Tabulka 8.14: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-7 ze dne 13. září 2011

Čas	Skutečná dojezdová doba	Predikovaná dojezdová doba
0:15	1331,4	1351,37
1:10	1331,4	1395,68
2:25	1331,4	1387,62
3:40	1331,4	1385,19
4:50	1331,4	1423,39
5:35	1353,1	1577,99
6:30	1912,1	1835,21
7:15	3111,8	2519,37
8:30	2677	2825,97
9:30	1571,8	1722,21
13:40	1353	1552,88
14:30	1549,1	1707,04
15:35	1572,6	1559,11
16:30	1555,8	1652,92
17:30	1499,2	1748,85
18:30	1336,8	1544,14
19:30	1331,4	1489,11
20:35	1331,4	1485,82
21:30	1331,4	1410,87
22:35	1331,4	1483,31
23:35	1331,4	1421,49

Tabulka 8.15: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-7 ze dne 21. září 2011



Obrázek 8.23: Srovnání skutečné a predikované dojezdové doby pro sekci tripId-7 ze dne 21. září 2011. Na x-ové ose je čas během dne a na y-ové ose je dojezdová doba. Modrou barvou je znázorněna skutečně naměřená dojezdová doba a zelenou barvou predikovaná dojezdová doba.

## Kapitola 9

# Závěr

V úvodní kapitole této práce byly popsány principy genetických algoritmů. Na názorných ukázkách bylo předvedeno, jak se provádí křížení a mutace a jak se tyto genetické operátory liší u různých typů genetických algoritmů. V další kapitole bylo podrobněji popsáno kartézské genetické programování jakožto metoda umělé inteligence, pomocí které byla predikována dojezdová doba. V další části byla definována symbolická regrese a je popsán její průběh s využitím kartézského genetického programování. V kapitole 5 byly stručně popsány přístroje pro získávání informací v dopravě, byla vysvětlena problematika dojezdových dob a popsány výsledky predikce dojezdových dob pomocí soft-computingových metod support vector regression a neuronových sítí. V dalších kapitole byla popsána metoda použitá v této práci pro predikci dojezdových dob. Kapitola 7 popisuje aplikaci navrženou v rámci této práce, která se snaží pomocí kartézského genetického programování predikovat dojezdovou dobu automobilů. Předposlední kapitola popisuje experimenty prováděné nad daty z reálného provozu.

Tato práce měla za cíl prozkoumat možnosti využití symbolické regrese a genetických algoritmů pro predikování dopravy. Byla navržena metoda predikce dojezdových dob využívající kartézské genetické programování. Metoda byla otestována nad dopravními daty získanými z reálného provozu. Ze získaných výsledků vyplývá, že kartézské genetické programování nepatří mezi nejlepší metody predikce dojezdových dob. Tato metoda dává přibližné, nikoliv však přesné výsledky. Bohužel však pro velmi vysokou časovou náročnost evoluce nebylo možné důsledně prozkoumat všechny možné parametry, které mohou průběh evoluce ovlivňovat. Mezi další výzkum by mohlo patřit podrobnější prozkoumání vhodnosti použití funkcí, které reprezentují jednotlivé uzly CGP mřížky. V této práci byly vybrány jen některé základní množiny funkcí. Existuje však ještě mnoho variant, které vyzkoušeny nebyly. Další možností by bylo přidat do množiny funkcí i některé nové funkce. Další experimenty by se mohly podrobněji zabývat nejvhodnějším poměrem mezi počtem sloupců CGP mřížky a počtem řádků CGP mřížky. Výsledky této práce mohou sloužit jako základ při dalším výzkumu využití symbolické regrese a genetických algoritmů pro predikci dopravních jevů.

# Literatura

- [1] Research Data Exchange. ©2013.  
URL <https://www.its-rde.net/>
- [2] Basak, D.; Pal, S.; Patranabis, D. C.: Support vector regression. *Neural Information Processing-Letters and Reviews*, ročník 11, č. 10, 2007: s. 203–224.
- [3] Collet, P.; Lutton, E.; Raynal, F.; aj.: Polar IFS+Parisian Genetic Programming=Efficient IFS Inverse Problem Solving. *Genetic Programming and Evolvable Machines*, ročník 1, č. 4, 2000: s. 339–361, ISSN 1389-2576, doi:10.1023/A:1010065123132.  
URL <http://dx.doi.org/10.1023/A%3A1010065123132>
- [4] Davidson, J.; Savic, D.; Walters, G.: Symbolic and numerical regression: experiments and applications. *Information Sciences*, ročník 150, č. 1-2, 2003: s. 95 – 117, ISSN 0020-0255, doi:10.1016/S0020-0255(02)00371-7.  
URL <http://www.sciencedirect.com/science/article/pii/S0020025502003717>
- [5] DiPaola, S.; Sorenson, N.: CGP, Creativity and Art. In *Cartesian Genetic Programming*, editace J. F. Miller, Natural Computing Series, Springer Berlin Heidelberg, 2011, ISBN 978-3-642-17309-7, s. 293–307, doi:10.1007/978-3-642-17310-3\_10.  
URL [http://dx.doi.org/10.1007/978-3-642-17310-3\\_10](http://dx.doi.org/10.1007/978-3-642-17310-3_10)
- [6] Harding, S.; Miller, J.; Banzhaf, W.: Self modifying Cartesian Genetic Programming: Parity. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, 2009, s. 285–292, doi:10.1109/CEC.2009.4982960.  
URL <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4982960>
- [7] Harding, S.; Miller, J.; Banzhaf, W.: Self-Modifying Cartesian Genetic Programming. In *Cartesian Genetic Programming*, editace J. F. Miller, Natural Computing Series, Springer Berlin Heidelberg, 2011, ISBN 978-3-642-17309-7, s. 101–124, doi:10.1007/978-3-642-17310-3\_4.  
URL [http://dx.doi.org/10.1007/978-3-642-17310-3\\_4](http://dx.doi.org/10.1007/978-3-642-17310-3_4)
- [8] Hynek, J.: *Genetické algoritmy a genetické programování*. Grada, první vydání, 2008, ISBN 97-880-247-2695-3.
- [9] Jurgen, B.: *Evolutionary optimization in dynamic environments*. Kluwer Academic, vyd. 1 vydání, 2002, ISBN 07-923-7631-5.
- [10] Kanayama, K.; Fujikawa, Y.; Fujimoto, K.; aj.: Development of vehicle-license number recognition system using real-time image processing and its application to

- travel-time measurement. In *Vehicular Technology Conference, 1991. Gateway to the Future Technology in Motion., 41st IEEE*, 1991, ISSN 1090-3038, s. 798–804, doi:10.1109/VETEC.1991.140605.
- [11] Khan, M.; Khan, G.; Miller, J.: Efficient representation of Recurrent Neural Networks for markovian/non-markovian non-linear control problems. In *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, 2010, s. 615–620, doi:10.1109/ISDA.2010.5687197.  
URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=5687197](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5687197)
- [12] Koza, J. R.: Genetically breeding populations of computer programs to solve problems in artificial intelligence. In *Tools for Artificial Intelligence, 1990., Proceedings of the 2nd International IEEE Conference on*, 1990, s. 819–827, doi:10.1109/TAI.1990.130444.  
URL <ftp://reports.stanford.edu/pub/ctr/reports/cs/tr/90/1314/CS-TR-90-1314.pdf>
- [13] Langdon, W. B.: *Genetic programming and data structures*. Kluwer Academic Publishers, 1998, ISBN 07-923-8135-1.
- [14] Mařík, V.; Štěpánková, O.; Lažanský, J.: *Umělá inteligence (3)*. Academia, vyd. 1. vydání, 2001, ISBN 80-200-0472-6.
- [15] Miller, J.: *Cartesian genetic programming*. Springer Berlin Heidelberg, 2011, ISBN 97-836-421-7309-7.
- [16] Miller, J.; Smith, S.: Redundancy and computational efficiency in Cartesian genetic programming. *Evolutionary Computation, IEEE Transactions on*, ročník 10, č. 2, 2006: s. 167–174, ISSN 1089-778X, doi:10.1109/TEVC.2006.871253.
- [17] Miller, J.; Thomson, P.: Cartesian Genetic Programming. In *Genetic Programming, Lecture Notes in Computer Science*, ročník 1802, editace R. Poli; W. Banzhaf; W. Langdon; J. Miller; P. Nordin; T. Fogarty, Springer Berlin Heidelberg, 2000, ISBN 978-3-540-67339-2, s. 121–132, doi:10.1007/978-3-540-46239-2\_9  
URL [http://dx.doi.org/10.1007/978-3-540-46239-2\\_9](http://dx.doi.org/10.1007/978-3-540-46239-2_9)
- [18] Paličková, L.; Rosenauer, J.: Jak by měla vypadat hlášení pro Zelenou vlnu? [online]. 2005-04-08.  
URL [http://www.rozhlas.cz/zelenavlna/skoladz/\\_zprava/183270](http://www.rozhlas.cz/zelenavlna/skoladz/_zprava/183270)
- [19] Park, D.; Rilett, L. R.: Forecasting freeway link travel times with a multilayer feedforward neural network. *Computer-Aided Civil and Infrastructure Engineering*, ročník 14, č. 5, 1999: s. 357–367.
- [20] Příbil, P.: Teorie provozu na pozemních komunikacích - TEPR, přednášky k předmětu. přednášky k předmětu, 2012.  
URL [https://www.fd.cvut.cz/personal/burkejos/TEPR/Prednasky/Kap.%20%20Detektory/Dopravní%20detektory\\_short%20%5Bre%5B017e%5Dim%20kompatibility%5D.pdf](https://www.fd.cvut.cz/personal/burkejos/TEPR/Prednasky/Kap.%20%20Detektory/Dopravní%20detektory_short%20%5Bre%5B017e%5Dim%20kompatibility%5D.pdf)
- [21] Pospíchal, J.; Kvasnička, V.; Tiňo, P.: *Evolučné algoritmy*. Slovenská technická univerzita, prvé vydání, 2000, ISBN 80-227-1377-5.

- [22] Schmidt, M.; Lipson, H.: Distilling Free-Form Natural Laws from Experimental Data. *Science*, ročník 324, č. 5923, 2009: s. 81–85, doi:10.1126/science.1165893, <http://www.sciencemag.org/content/324/5923/81.full.pdf>.  
URL <http://www.sciencemag.org/content/324/5923/81.abstract>
- [23] Schwarzl, J.: Aplikované evoluční algoritmy - EVO. přednášky k předmětu, 2008.
- [24] Sekanina, L.: *Evoluční hardware*. Academia, vyd. 1. vydání, 2009, ISBN 97-880-200-1729-1.
- [25] Sekanina, L.: Biologií inspirované počítače - BIN. přednášky k předmětu, 2011.
- [26] Walker, J.; Liu, Y.; Tempesti, G.; aj.: Automatic Code Generation on a MOVE Processor Using Cartesian Genetic Programming. In *Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science*, ročník 6274, editace G. Tempesti; A. Tyrrell; J. Miller, Springer Berlin Heidelberg, 2010, ISBN 978-3-642-15322-8, s. 238–249, doi:10.1007/978-3-642-15323-5\_21.  
URL [http://dx.doi.org/10.1007/978-3-642-15323-5\\_21](http://dx.doi.org/10.1007/978-3-642-15323-5_21)
- [27] Walker, J.; Miller, J.: The Automatic Acquisition, Evolution and Reuse of Modules in Cartesian Genetic Programming. *Evolutionary Computation, IEEE Transactions on*, ročník 12, č. 4, 2008: s. 397–417, ISSN 1089-778X, doi:10.1109/TEVC.2007.903549.
- [28] Walker, J.; Volk, K.; Smith, S.; aj.: Parallel evolution using multi-chromosome cartesian genetic programming. *Genetic Programming and Evolvable Machines*, ročník 10, č. 4, 2009: s. 417–445, ISSN 1389-2576, doi:10.1007/s10710-009-9093-2.  
URL <http://dx.doi.org/10.1007/s10710-009-9093-2>
- [29] Weise, T.: *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published): Germany, 2009.  
URL <http://www.it-weise.de/projects/book.pdf>
- [30] Wu, C.-H.; Ho, J.-M.; Lee, D.: Travel-time prediction with support vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, ročník 5, č. 4, 2004: s. 276–281, ISSN 1524-9050, doi:10.1109/TITS.2004.837813.