



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

FACULTY OF INFORMATION TECHNOLOGY

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**

DEPARTMENT OF INTELLIGENT SYSTEMS

**SYSTÉM PRE PODPORU OPTIMALIZÁCIE SIETE  
MESTSKEJ HROMADNEJ DOPRAVY**

SYSTEM FOR SUPPORTING THE OPTIMIZATION OF URBAN PUBLIC TRANSPORT NETWORK

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

**AUTOR PRÁCE**

AUTHOR

**LUKÁŠ KATONA**

**VEDOUCÍ PRÁCE**

SUPERVISOR

**doc. Ing. FRANTIŠEK ZBOŘIL, Ph.D.**

**BRNO 2025**

## Zadání bakalářské práce



163440

Ústav: Ústav inteligentních systémů (UITS)  
Student: **Katona Lukáš**  
Program: Informační technologie  
Název: **Systém pro podporu optimalizace sítě městské hromadné dopravy**  
Kategorie: Umělá inteligence  
Akademický rok: 2024/25

### Zadání:

1. Prostudujte problematiku vytváření sítě městské hromadné dopravy, její modelování a její optimalizace pro města do půl milionu obyvatel.
2. Pro město velikosti krajského města České republiky získejte nebo odhadněte data, ze kterých sestavte model zdejší městské hromadné dopravy.
3. Seznamte se s řešeními, které pro optimalizaci takové sítě používají metody umělé inteligence.
4. Vytvořte prostředí, které bude sloužit k modelování a optimalizaci takové sítě.
5. Na vhodně zvolených příkladech ověřte fungování vašeho systému a diskutujte dosažené výsledky.

### Literatura:

- Kate Han, Lee A. Christie, Alexandru-Ciprian Zăvoianu, and John McCall. 2021. Optimising the introduction of connected and autonomous vehicles in a public transport system using macro-level mobility simulations and evolutionary algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '21). Association for Computing Machinery, New York, NY, USA, 315–316.
- Yentl Van Tendeloo and Hans Vangheluwe. 2018. Discrete event system specification modeling and simulation. In Proceedings of the 2018 Winter Simulation Conference (WSC '18). IEEE Press, 162–176.
- 

Při obhajobě semestrální části projektu je požadováno:  
První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**  
Vedoucí ústavu: Kočí Radek, Ing., Ph.D.  
Datum zadání: 1.11.2024  
Termín pro odevzdání: 14.5.2025  
Datum schválení: 31.10.2024

## Abstrakt

Cielom tejto práce je optimalizácia mestskej hromadnej dopravy, konkrétne časového rozpisu jednej linky. Zmyslom tohto textu je popis celého riešenia daného problému, od analýzy až po výsledný systém. Informácie získané priamo z Dopravného podniku mesta Brno boli využité na zostrojenie simulačného modelu jednej linky mestskej hromadnej dopravy. Pomocou tohto modelu a genetického algoritmu sa systém pokúsi nájsť najoptimálnejší časový rozpis danej linky. Súčasťou práce je aj nástroj s používateľským rozhraním, ktorý je možné v praxi použiť na analýzu a optimalizáciu terajších časových rozpisov jednotlivých liniek. Nástroj poskytuje širokú škálu rôznych vstupov, ktorými môže analytik obmedziť isté vlastnosti výsledného rozpisu. Vďaka tomuto nástroju by sa mala práca analytikov, ktorí majú na starosť správu a vytváranie liniek, výrazne zjednodušiť a zvýšiť tak efektivitu mestskej hromadnej dopravy.

## Abstract

The aim of this work is the optimization of urban public transport, specifically the timetable of one line. The purpose of this text is to describe the entire solution of the given problem, from analysis to the resulting system. Information obtained directly from the Brno Public Transport Company was used to construct a simulation model of one urban public transport line. Using this model and a genetic algorithm, the system will attempt to find the most optimal timetable for the given line. Part of this work is a tool with a user interface that can be used in practice to analyze and optimize the current timetables of individual lines. The tool provides a wide range of different inputs that an analyst can use to restrict certain properties of the resulting timetable. Thanks to this tool, the work of analysts who are responsible for managing and creating lines should be significantly simplified and thus increase the efficiency of urban public transport.

## Klíčové slová

optimalizácia, mestská hromadná doprava, algoritmy inšpirované prírodou, genetický algoritmus, simulácia, DEVS

## Keywords

optimization, urban public transport, nature-inspired algorithms, genetic algorithm, simulation, DEVS

## Citácia

KATONA, Lukáš. *Systém pre podporu optimalizácie siete mestskej hromadnej dopravy*. Brno, 2025. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D.

# Systém pre podporu optimalizácie siete mestskej hromadnej dopravy

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostante pod vedením pána doc. Ing. Františka Zbořila Ph.D. Ďalšie informácie mi poskytol pán Dr. Ing. Petr Peringer z jeho prezentácií predmetov IMS a SNT a pán Michael Kříž z Dopravného podniku mesta Brno. Uviedol som všetky literárne zdroje, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....

Lukáš Katona

5. mája 2025

## Podakovanie

Rád by som poďakoval pánovi Zbořilovi za jeho cenné rady a konzultácie, hlavne zo strany optimalizácie a algoritmov z odvetvia umelej inteligencie. Pánovi Křížovi by som rád poďakoval za jeho čas a ochotu poskytnúť informácie o fungovaní MHD v Brne a zároveň za jeho pripomienky z pohľadu možného používateľa systému. Nakoniec by som chcel poďakovať môjmu priateľovi Danovi Valníčkovi a jeho starému otcovi za to, že mi vybavili kontakt na pána Kříže a tým umožnili získať potrebné informácie o riešenom probléme priamo z prvej ruky.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>3</b>
<b>2</b>	<b>Relevantné vlastnosti mestskej hromadnej dopravy</b>	<b>5</b>
2.1	Konzultácia s analytikom Dopravného podniku . . . . .	5
2.2	Získané informácie . . . . .	5
<b>3</b>	<b>Simulačný model linky mestskej hromadnej dopravy</b>	<b>7</b>
3.1	DEVS formalizmus . . . . .	7
3.2	Popis simulačného modelu . . . . .	10
3.3	Výstupy simulačného modelu . . . . .	17
<b>4</b>	<b>Optimalizácia časového rozpisu linky mestskej hromadnej dopravy</b>	<b>22</b>
4.1	Vstupy potrebné od používateľa . . . . .	22
4.2	Základná implementácia genetického algoritmu . . . . .	23
4.3	Multi-objektívny genetický algoritmus . . . . .	25
4.4	Výstupy optimalizácie . . . . .	32
<b>5</b>	<b>Experimenty</b>	<b>34</b>
5.1	Experiment s krátkou trasou a krátkym vozidlom . . . . .	35
5.2	Experiment s krátkou trasou a dlhým vozidlom . . . . .	36
5.3	Experiment s dlhou trasou a krátkym vozidlom . . . . .	37
5.4	Experiment s dlhou trasou a dlhým vozidlom . . . . .	38
<b>6</b>	<b>Nástroj na analýzu a optimalizáciu časových rozpisov</b>	<b>39</b>
6.1	Návrh užívateľského rozhrania . . . . .	39
6.2	Implementácia nástroja . . . . .	43
<b>7</b>	<b>Záver</b>	<b>46</b>
	<b>Literatúra</b>	<b>48</b>
<b>A</b>	<b>Experiment s krátkou trasou a krátkym vozidlom</b>	<b>50</b>
<b>B</b>	<b>Experiment s krátkou trasou a dlhým vozidlom</b>	<b>52</b>
<b>C</b>	<b>Experiment s dlhou trasou a krátkym vozidlom</b>	<b>54</b>
<b>D</b>	<b>Experiment s dlhou trasou a dlhým vozidlom</b>	<b>56</b>

# Zoznam obrázkov

3.1	Stavový automat modelu zastávky . . . . .	11
3.2	Stavový automat modelu vozidla . . . . .	13
3.3	Schéma zloženého simulačného modelu . . . . .	15
3.4	Počet cestujúcich prichádzajúcich na zastávku za hodinu . . . . .	19
3.5	Priemerný čas strávený čakaním za hodinu . . . . .	19
3.6	Priemerná naplnenosť vozidla na jednotlivých zastávkach . . . . .	21
4.1	Príklad Pareto frontu . . . . .	26
4.2	Populácia na začiatku a na konci optimalizácie . . . . .	33
6.1	Wireframe pre analýzu časového rozpisu . . . . .	40
6.2	Wireframe pre optimalizáciu časového rozpisu . . . . .	42
A.1	Počet cestujúcich prichádzajúcich na zastávku za hodinu . . . . .	50
A.2	Priemerný čas strávený čakaním za hodinu . . . . .	50
B.1	Počet cestujúcich prichádzajúcich na zastávku za hodinu . . . . .	52
B.2	Priemerný čas strávený čakaním za hodinu . . . . .	52
C.1	Počet cestujúcich prichádzajúcich na zastávku za hodinu . . . . .	54
C.2	Priemerný čas strávený čakaním za hodinu . . . . .	54
D.1	Počet cestujúcich prichádzajúcich na zastávku za hodinu . . . . .	56
D.2	Priemerný čas strávený čakaním za hodinu . . . . .	56

# Kapitola 1

## Úvod

Mestskú hromadnú opravu využíva takmer každý človek a preto sa oplatí pracovať na jej efektívite. Zmyslom tejto práce je nazrieť do problematiky vytvárania a správy jednotlivých liniek mestskej hromadnej dopravy, ich analýze a optimalizácie. Ďalej vytvoriť nástroj, ktorý pomôže analytikom pri vytváraní nových liniek alebo optimalizácii existujúcich. Tento nástroj popísať a otestovať na reálnych dátach. Výsledkom celej práce by teda malo byť teoretické zefektívnenie mestskej hromadnej dopravy.

Mestská hromadná doprava je systém, ktorý je pre mnoho ľudí veľmi dôležitý, preto sa oplatí investovať čas a námahu do jeho optimalizácie. Keďže aj mne je problém s dlhým čakaním na zastávke alebo preplneným autobusom blízky, rozhodol som sa venovať sa práve jemu.

Ľudia odnepamäti nachádzajú inspiráciu pre svoje vynálezy v prírode a tomuto trendu neunikla ani informatika, ktorá má od prírody zo všetkých oborov asi najďalej. Algoritmy inšpirované prírodou prinášajú riešenia na problémy, ktoré by sme inými spôsobmi vedeli vyriešiť len za nereálne dlhý čas. Táto časť informatiky ma veľmi zaujala a preto som sa rozhodol nájsť ďalšie využitie pre jeden z týchto algoritmov. Tieto algoritmy sa používajú pre komplexné problémy s veľkým množstvom navzájom sa ovplyvňujúcich premenných. Jedným z takýchto problémov môžu byť optimalizácie rôznych systémov.

Cieľom tejto práce je vytvoriť nástroj, ktorý za pomoci genetického algoritmu a používateľských obmedzení nájde optimálny časový rozpis jednej linky mestskej hromadnej dopravy. Zároveň sa tento nástroj bude dať použiť na analýzu už existujúcich rozpisov a ich porovnanie s novými riešeniami. Nástroj bude mať užívateľské rozhranie pre rýchlu manipuláciu so vstupnými dátami a prehľadné zobrazenie výsledkov.

Práca je rozdelená na 7 kapitol vrátane úvodu a záveru, z toho najdôležitejšie sú kapitoly 2 až 6.

Na zostrojenie simulačného modelu treba najprv vyhodnotiť, ktoré vlastnosti reálneho systému sú pre nás relevantné. V kapitole 2 je popísaná konzultácia s analytikom Dopravného podniku mesta Brno, ktorý poskytol informácie z reálneho života. Popísal postupy a metódy, ktoré sa používajú pre optimalizáciu liniek teraz a vytýčil vlastnosti na ktoré je potrebné sa zamerať.

Zo získaných informácií bol vytvorený simulačný model, ktorého popis, stručné zhrnutie implementácie a jednotlivé experimenty s ním sú popísané v kapitole 3. Výstupy simulačného modelu sú základom pre optimalizáciu časového rozpisu linky. Ide o sériu grafov, na ktorých môže používateľ vidieť vyťaženosť linky v priebehu dňa a naprieč zastávkami. Tieto informácie poslúžili k validácii simulačného modelu a na ohodnotenie jednotlivých rozpisov.

V kapitole 4 je popísaný proces optimalizácie časového rozpisu linky mestskej hromadnej dopravy. Jednotlivé rozpisy ohodnotené na základe informácií získaných zo simulácie sa použijú ako jedinci v genetickom algoritme. Genetický algoritmus bude tieto rozpisy krížiť medzi sebou a mutovať, aby našiel čo najlepšie riešenie. Výsledky optimalizácie budú porovnané s pôvodnými rozpismi a ohodnotené.

V kapitole 5, sú popísané experimenty optimalizácie. Aplikácia a optimalizačný algoritmus sa testoval priebežne počas vývoja. V tejto kapitole sú uvedené experimenty, ktoré sa realizovali až po dokončení vývoja. Konkrétne ide o experimenty, ktoré skúmajú správanie sa optimalizačného algoritmu vzhľadom na zmeny vstupov ako dĺžka trasy a kapacita vozidla.

Samotný program by v praxi nebol moc použiteľný bez používateľského rozhrania. V kapitole 6 je popísaný návrh a implementácia užívateľského rozhrania analytického nástroja. Tento nástroj umožňuje používateľovi jednoducho zadať vstupné dáta, spustiť analýzu a optimalizáciu časového rozpisu a zobrazíť výsledky. Jednou z najdôležitejších častí tohto rozhrania je zadávanie používateľských obmedzení, ktoré musí výsledný časový rozpis spĺňať.

V závere 7 je uvedené zhrnutie dosiahnutých výsledkov práce. Táto kapitola taktiež obsahuje možné budúce rozšírenia aplikácie.



## Kapitola 2

# Relevantné vlastnosti mestskej hromadnej dopravy

Systém mestskej hromadnej dopravy je veľmi komplexný, každá jeho časť stojí za zváženie možnej optimalizácie. Ako prvý krok je teda výber tej časti, ktorá sa bude optimalizovať. Pre toto rozhodnutie sú potrebné informácie o reálnom systéme, doterajších postupoch, spôsoboch analýzy a optimalizácie.

### 2.1 Konzultácia s analytikom Dopravného podniku

Pre získanie potrebných informácií som sa obrátil na pána Michaela Kříže, analytika Dopravného podniku mesta Brno. Stretnutie s pánom Křížom prebehlo 7. 10. 2024 v priestoroch dopravného podniku. Vďaka získaným informáciám som sa rozhodol zamerať na analýzu a optimalizáciu časového rozpisu jednej linky mestskej hromadnej dopravy. Optimalizovať viac ako jednu linku naraz by pravdepodobne prinieslo lepšie výsledky, ale to len teoreticky. V praxi by sa to z finančných a časových dôvodov neoplatilo. Zhodli sme sa na tom, že najlepšie bude optimalizovať jednu linku a to iba jedným smerom, pretože druhý smer linky sa dá považovať za samostatnú linku, minimálne v rámci nárokov cestujúcich.

### 2.2 Získané informácie

Informácie, ktoré sú dôležité pre analýzu a optimalizáciu časového rozpisu linky mestskej hromadnej dopravy sú:

- **Počet zastávok** — potrebné pre zostavenie samotného rozpisu aj simulačného modelu, nie je dôležitý len počet, ale aj informácie o jednotlivých zastávkach.
- **Čas potrebný na prejdienie od jednej zastávky k druhej** — tiež je to súčasťou rozpisu, zároveň sa touto informáciou riadi plánovanie udalostí v simulačnom modeli, viac v kapitole 3.
- **Počet odchodov vozidla z jeho prvej zastávky za celý deň** — celkový počet vozidiel, respektíve ciest ktoré vozidlá danej linky za deň absolvujú, tento parameter je dôležitý pre optimalizáciu, pre dopravný podnik je výhodné mať čo najmenší počet vozidiel, ktoré sú v prevádzke, šetrí to palivo, údržbu aj ľudské zdroje.

- **Počet odchodov vozidla z jeho prvej zastávky za hodinu** — myslí sa tým jeden riadok v časovom rozpise, mal by byť priamo úmerný priemernému počtu cestujúcich čakajúcich na danú linku v danej hodine.
- **Počet cestujúcich čakajúcich na jednotlivých zastávkach** — časový rozpis musí dostatočne uspokojiť potreby všetkých cestujúcich, zároveň je ťažké získať presné hodnoty. Pri reálnom použití by sa mala spraviť štúdia ktorá by tieto hodnoty získala. Na návrh pána Kříže som na testovacie účely použil hodnoty, ktoré sú úmerné počtu odchádzajúcich vozidiel v hodine a kapacite vozidiel, povedal, že linky v Brne sú už celkom dobre naplánované a preto by mali byť tieto hodnoty približne správne. Jedná sa o nemenný vstup optimalizácie, ktorému sa má časový rozpis prispôbiť, preto je teoreticky jedno aké hodnoty budú do systému zadane.
- **Doba čakania na zastávke** — užitočný parameter pre analytika, ktorý slúži na overenie toho, či je časový rozpis kvalitný.
- **Počet cestujúcich vo vozidle** — tretí dôležitý parameter pre optimalizáciu, vysoká preplnenosť vozidla má za dôsledok nekomfortné cestovanie, na druhej strane, príliš prázdne vozidlo je neefektívne využité zo strany dopravného podniku.
- **Kapacita vozidla** — každé vozidlo prenesie naraz len isté množstvo cestujúcich, zároveň sa od preplnenosti vozidla odvíja komfort cestujúcich. Informácie o kapacite konkrétnych vozidiel sú dostupné na webovej stránke dopravného podniku [3].
- **Miest na sedenie** — To či si cestujúci zadne alebo bude musieť počas jazdy stáť ovplyvňuje jeho spokojnosť a komfort. Čím viac ľudí bude musieť stáť v preplnených vozidlách, tým horšie bude rozpis ohodnotený.
- **Počet cestujúcich, ktorí nenásúpia kvôli preplnenosti vozidla** — v extrémnych prípadoch môže nastať, že bude vozidlo úplne plné a zvyšní cestujúci sa do neho už nezmestia, rozpisy pri ktorých sa toto bude diať budú veľmi negatívne ohodnocované aby sa táto nežiadúca situácia čo najviac eliminova, viac v kapitole o optimalizácii 4.
- **Počet vystupujúcich cestujúcich na jednotlivých zastávkach** — tento parameter je potrebný pre simuláciu, keďže do vozidla s istou kapacitou priebežne nastupuje veľa cestujúcich, musia z neho aj vystupovať, každá zastávka bude mať pridelený koeficient dôležitosti, ktorý bude určovať aký podiel cestujúcich vystúpi na danej zastávke. Na prvej zastávke je tento koeficient 0, keďže na nej cestujúci iba nastupujú, na poslednej zastávke je koeficient 1, keďže na nej cestujúci iba vystupujú a na zastávkach medzi nimi sa koeficient mení podľa toho aký podiel cestujúcich na danej zastávke priemerne vystupuje, prestupné zastávky budú mať vyšší koeficient ako tie, ktoré cestujúcich len zbierajú.
- **Náklady na prevádzku** — Náklady na prevádzku sú potrebné pre optimalizáciu, pretože zo strany dopravného podniku chceme, aby boli čo najnižšie. Náklady jednotlivých typov vozidiel sú súčasťou finančného plánu zahrnutého v zmluve [14].
- **Dĺžka trasy** — Tento parameter je potrebný na výpočet celkových nákladov na prevádzku linky. Dĺžky jednotlivých liniek sú spomenuté v zmluve [14].

## Kapitola 3

# Simulačný model linky mestskej hromadnej dopravy

Druhým hlavným krokom po získaní potrebných informácií je zostrojenie simulačného modelu. Problém optimalizácie časového rozpisu linky mestskej hromadnej dopravy je veľmi komplexný a vytvoriť program ktorý by ho riešil analyticky by bolo veľmi náročné. Preto sme sa rozhodli vytvoriť simulačný model, ktorý bude reprezentovať jednu linku MHD. Jeho súčasťou budú jednotlivé zastávky, vozidlá a cestujúci, ktorí budú vozidlá využívať na prepravu medzi zastávkami. Najdôležitejšou časťou bude časový rozpis, ktorý bude obsahovať odchody jednotlivých vozidiel. Na základe informácií získaných z priebehu simulácie sa následne tento rozpis ohodnotí a toto ohodnotenie sa využije v genetickom algoritme na optimalizáciu časového rozpisu. Viac informácií o optimalizácii a genetickom algoritme je v kapitole 4.

### 3.1 DEVS formalizmus

Na modelovanie bol použitý formalizmus DEVS (Discrete Event System Specification). Všetky informácie o tomto formalizme v tejto sekcii boli čerpané zo zborníka z konferencie WSC 2018 [16], kde je DEVS podrobne popísaný. Tento formalizmus je založený na diskrétnych udalostiach, pri ktorých sa celý modelový systém posúva skokovo v čase dopredu. Pri každej takejto udalosti sa mení vnútorný stav systému a vykonáva sa nejaká akcia. Veľkou výhodou tohto formalizmu je jeho modularita. Atomické modely sa dajú pomocou vstupných a výstupných signálov prepojiť do zloženého modelu.

#### Atomické modely

Atomický model je základným stavebným kameňom DEVS formalizmu. Tento model je definovaný ako štvorica  $\langle S, q_{init}, \delta_{int}, ta \rangle$ , kde:

- $S$  je množina stavov modelu
- $q_{init}$  je počiatočný stav modelu, ktorý je prvkom množiny  $S$ , v pôvodnej definícii DEVS tento prvok neexistoval, no podľa zdroja [16] je potrebný pre správne fungovanie modelu
- $\delta_{int}$  je funkcia, ktorá určuje prechody medzi stavmi modelu

- $ta$  je funkcia, ktorá určuje ako dlho model zotrúva v danom stave, musí byť definovaná pre každý stav modelu

Ako príklad môže poslúžiť jednoduchý model semaforu, ten sa za normálnej prevádzky nachádza v troch stavoch. Raz svieti na zeleno, inokedy na oranžovo alebo červeno. Každá farba zotrúva rozsvietená stanovený čas a tiež je dané, aká farba bude nasledovať. Formálny zápis by potom vyzeral takto:

$$\begin{aligned} & \langle S, q_{init}, \delta_{int}, ta \rangle \\ S &= \{\text{Zelená}, \text{Oranžová}, \text{Červená}\} \\ q_{init} &= \text{Zelená} \\ \delta_{int} &= \{\text{Zelená} \rightarrow \text{Oranžová}, \text{Oranžová} \rightarrow \text{Červená}, \text{Červená} \rightarrow \text{Zelená}\} \\ ta &= \{\text{Zelená} \rightarrow 57, \text{Oranžová} \rightarrow 3, \text{Červená} \rightarrow 60\} \end{aligned}$$

## Výstupné signály

Výstupné signály slúžia na to aby model mohol komunikovať so svojím okolím, konkrétne aby mohol informovať o zmenách svojeho vnútorného stavu. Každý model môže mať viacero výstupných signálov, ktoré sú definované ako množina  $Y$ . Okrem výstupných signálov, musia byť definované aj výstupné funkcie  $\lambda$ .

- $Y$  je množina výstupných signálov modelu, pomocou ktorých dáva model vedieť svojmu okoliu, že sa nastala nejaká udalosť, napríklad, že sa zmenil jeho stav
- $\lambda$  je funkcia, ktorá slúži na generovanie samotnej udalosti, táto funkcia je definovaná pre každý stav modelu a určuje aké výstupné signály sa budú odosielať do okolia modelu pri prechode do iného stavu

Pre príklad semaforu môžeme definovať výstupné signály, ktoré nám povedia, aká farba práve svieti na semafore. Pri definovaní týchto signálov a výstupnej funkcie si treba dať pozor na to, že tieto udalosti sú diskkrétne a keď chceme signalizovať napríklad to, že semafor svieti na červeno, musíme to urobiť ešte pred prechodom do stavu červenej farby a teda táto funkcia musí byť definovaná pre stav oranžovej farby.

$$\begin{aligned} Y &= \{\text{svieti\_zelená}, \text{svieti\_oranžová}, \text{svieti\_červená}\} \\ \lambda &= \{\text{Zelená} \rightarrow \text{svieti\_oranžová}, \text{Oranžová} \rightarrow \text{svieti\_červená}, \text{Červená} \rightarrow \text{svieti\_zelená}\} \end{aligned}$$

## Vstupné signály

Vstupné signály slúžia na ovládanie modelu z vonku. Napríklad spomínaný semafor doteraz pracoval len na základe svojich vnútorne definovaných stavov a prechodov medzi nimi. V prípade, že nastane nejaká externá udalosť, napríklad sa niekto rozhodne semafor vypnúť, je potrebné nejakým spôsobom informovať model o tejto udalosti. O to sa starajú vstupné signály, ktoré sú definované ako množina  $X$ . Taktiež je potrebné definovať aj funkciu externých prechodov  $\delta_{ext}$ .

- $X$  je množina vstupných signálov modelu, ide o externé udalosti, ktorých príchod model očakáva

- $\delta_{ext}$  je funkcia, ktorá určuje do akého stavu sa model dostane po príchode externého signálu, táto funkcia je definovaná pre každý stav modelu a pre každý vstupný signál

Príklad so semaforom rozšírime o nový stav, v ktorom je semafor vypnutý. Semafor v tomto stave zotrúva neobmedzene dlho, pokiaľ nepríde ďalší vstupný signál, ktorý ho zapne. Potom môžeme pridať vstupné signály a funkciu externých prechodov.

$$\begin{aligned}
S &= S \cup \{\text{Vypnutý}\} \\
ta &= ta \cup \{\text{Vypnutý} \rightarrow \infty\} \\
X &= \{\text{vypnúť}, \text{zapnúť}\} \\
\delta_{ext} &= \{(\text{Zelená}, \text{vypnúť}) \rightarrow \text{Vypnutý}, \\
&(\text{Oranžová}, \text{vypnúť}) \rightarrow \text{Vypnutý}, \\
&(\text{Červená}, \text{vypnúť}) \rightarrow \text{Vypnutý}, \\
&(\text{Vypnutý}, \text{zapnúť}) \rightarrow \text{Červená}\}
\end{aligned}$$

## Zložené modely

Jedinou úlohou zložených modelov je definícia vzťahov medzi jednotlivými atomickými modelmi. Zložený model nemá definované žiadne vlastné stavy ani prechody medzi nimi. To všetko je zabalené v atomických modeloch, z ktorých sa skladá. Zložený model je definovaný ako sedmica  $\langle D, \{M_i\}, \{I_i\}, X_{self}, Y_{self}, select, \{Z_{i,j}\} \rangle$ , kde:

- $D$  je množina všetkých atomických modelov, ktoré sú súčasťou zloženého modelu
- $\{M_i\} = \{\langle S_i, q_{init,i}, \delta_{int,i}, ta_i, Y_i, \lambda_i, X_i, \delta_{ext,i} \rangle \mid i \in D\}$  sú jednotlivé definície atomických modelov z množiny  $D$
- $\{I_i\}$  pre každý atomický model z  $D$  je definovaná množina modelov, na ktoré má vplyv, teda modely, ktoré sú prepojené s daným atomickým modelom v smere, že výstupný signál daného modelu je napojený na vstupný signál modelu, na ktorý má vplyv
- $X_{self}$  je množina vstupných signálov zloženého modelu, ktorými sa dá model ovládať z vonku
- $Y_{self}$  je množina výstupných signálov zloženého modelu, ktorými model komunikuje so svojím okolím
- $select$  je funkcia, ktorá určuje prioritu v skupine konfliktných modelov, prepojením atomických modelov sa môže stať, že v rovnakom momente bude chcieť viacero modelov poslať výstupný signál do jedného modelu, táto funkcia určuje, ktorý z nich bude mať prednosť
- $\{Z_{i,j}\}$  je množina prepojení medzi atomickými modelmi, slúži ako mapovacia funkcia medzi výstupným signálom jedného modelu a vstupným signálom druhého modelu, takto sa dajú prepojiť vstupné a výstupné signály nie len atomických modelov, ale aj samotného zloženého modelu

Príklad zloženého modelu bude ukázaný neskôr v nasledujúcej sekcii kde bude popísaný model použitý v simulácii problému, ktorý rieši táto práca. Jeden príklad, ktorý pri tejto práci veľmi pomohol pochopiť ako sa atomické modely skladajú do zložených je popísaný v článku [12].

## Zhrnutie

DEVS formalizmus umožňuje definíciu atomocických modelov, ktorým je možné priradiť stavy, prechody medzi nimi, vstupné a výstupné signály, a definovať ich správanie. Tieto atomické modely sa dajú spájať do zložených modelov definovaním väzieb medzi nimi. Všetky informácie o DEVS formalizme, definície jednotlivých prvkov a aj príklady sú čerpané zo zborníka z konferencie WSC 2018 [16]. Zároveň sú v tomto zborníku uvedené aj príklady kódu napísané v jazyku `Python`, s použitím knižnice `PythonPDEVS`. Trochu zložitejšie, no za to podrobnejšie a formálnejšie vysvetlenie DEVS formalizmu je popísané v originálnej špecifikácii v knihe [17].

## 3.2 Popis simulačného modelu

Na začiatok je potrebné definovať základné atomické prvky modelu (viď obrázok 3.3), ich stavy a prepojenie medzi nimi. Simulačný model je zložený z troch hlavných častí, tými sú:

- **Atomický model zastávky** — dôležitý pre príchod a odchod cestujúcich
- **Atomický model vozidla** — dôležitý pre prepravu cestujúcich medzi zastávkami

V simulácii sa budú nachádzať viaceré inštancie modelu zastávky, každá reprezentujúca jednu zastávku na linke. Vozidiel bude taktiež viacero, neberieme do úvahy fyzické vozidlá a fakt, že v reálnom živote sa tieto vozidlá na linke v priebehu dňa niekoľkokrát otočia. Dôležitá je trasa, ktorú prejde, preto jedna inštancia modelu vozidla reprezentuje jednu cestu, ktorú vozidlo prejde, inými slovami jeden spoj MHD. Cestujúcich v modeli nie je potrebné reprezentovať ako samostatný atomický model, sú reprezentovaní len jedným číslom, časom kedy prišli na zastávku. Sú súčasťou modelu zastávky, konkrétne tej na ktorej čakajú.

## Atomický model zastávky

Model zastávky slúži na reprezentáciu jednotlivých zastávok na linke. Každá reálna zastávka je v systéme ako samostatná inštancia modelu zastávky. Je pasívna, to znamená, že jej vnútorný stav sa plne odvíja od aktivácie jej vstupných signálov. Ako je vidieť vo formálnej definícii, tento model nemá definované žiadne interné prechody, ani výstupné signály.

$$\langle S_i, q_{init,i}, \delta_{int,i}, ta_i, Y_i, \lambda_i, X_i, \delta_{ext,i} \rangle$$

$$S = \{\text{Cestujúci čakajú}, \text{Vozidlo na zastávke}, \text{Cestujúci nastupujú}\}$$

$$q_{init} = \text{Cestujúci čakajú}$$

$$\delta_{int} = \emptyset$$

$$ta = \{\text{Cestujúci čakajú} \rightarrow \infty, \text{Vozidlo na zastávke} \rightarrow \infty, \text{Cestujúci nastupujú} \rightarrow \infty\}$$

$$Y = \emptyset$$

$$\lambda = \emptyset$$

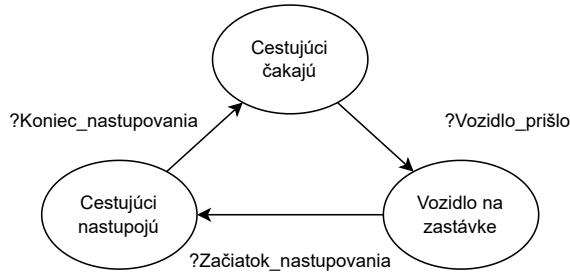
$$X = \{?Vozidlo\_prišlo, ?Začiatok\_nastupovania, ?Koniec\_nastupovania\}$$

$$\delta_{ext} = \{(\text{Cestujúci čakajú}, ?Vozidlo\_prišlo) \rightarrow \text{Vozidlo na zastávke},$$

$$(\text{Vozidlo na zastávke}, ?Začiatok\_nastupovania) \rightarrow \text{Cestujúci nastupujú},$$

$$(\text{Cestujúci nastupujú}, ?Koniec\_nastupovania) \rightarrow \text{Cestujúci čakajú}\}$$

Na obrázku 3.1 je graficky znázornený stavový automat. Je na ňom vidieť tri stavy modelu, a prechody medzi nimi, ktoré sú zapríčinené tromi vstupnými signálmi. Stavy modelu sú popísané v tabuľke 3.1 a akcie, ktoré sa vykonajú pri príchode vstupného signálu sú popísané v tabuľke 3.2. To kedy sa tieto vstupné signály aktivujú bude popísané neskôr v rámci definície zloženého modelu.



Obr. 3.1: Stavový automat modelu zastávky

Stav	Popis
Cestujúci čakajú	na zastávke nie je žiadne vozidlo
Vozidlo na zastávke	na zastávku práve prišlo vozidlo
Cestujúci nastupujú	cestujúci nastupujú do vozidla

Tabuľka 3.1: Stavy modelu zastávky

Vstupný signál	Akcia ktorú spúšťa
?Vozidlo_prišlo	výpočet časového intervalu medzi posledným príchodom vozidla a terajším príchodom
?Začiatok_nastupovania	generovanie príchodov cestujúcich
?Koniec_nastupovania	prepísanie času posledného príchodu vozidla

Tabuľka 3.2: Vstupné signály modelu zastávky

### Generovanie príchodov cestujúcich

Cestujúci sa generujú na základe exponenciálneho rozdelenia s parametrom  $\lambda$ . Tento parameter je v rámci jednej zastávky každú hodinu iný, pretože počet prichádzajúcich cestujúcich sa v priebehu dňa mení. V algoritme 1 je popísaný proces generovania cestujúcich na zastávke. Ak by stačil počet cestujúcich, ktorý za daný časový interval prišli, stačilo by nám Poissonovo rozdelenie. Avšak pre výpočet doby čakania na zastávke je potrebné vedieť aj čas príchodu jednotlivých cestujúcich. Preto je použité exponenciálne rozdelenie, ktoré generuje náhodný časový interval medzi príchodami jednotlivých cestujúcich. Vzťah medzi týmito dvoma rozdeleniami je a podrobnejší popis exponenciálneho rozdelenia je v článku [5]. Pripočítaním tohto intervalu k aktuálnemu modelovému času sa získa čas príchodu cestujúceho. Modelový čas sa o tento vygenerovaný časový údaj posunie. Tento proces sa opakuje až dokým modelový čas nedosiahne čas príchodu ďalšieho vozidla.

---

#### Algoritmus 1: Generovanie cestujúcich

---

```

získanie  $\lambda$  pre danú zastáku a hodinu;
while modelový čas < čas príchodu vozidla do
    vygenerovanie času príchodu cestujúceho;
    pridanie cestujúceho do fronty;
    zmena modelového času na čas príchodu cestujúceho;

```

---

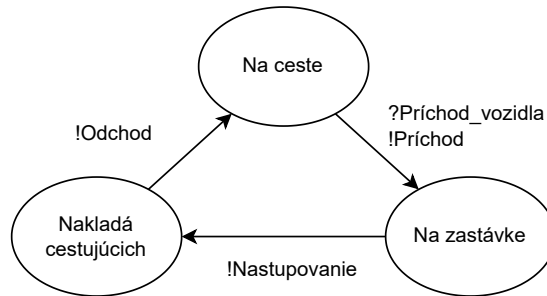


## Atomický model vozidla

Každé vozidlo, respektíve jedna trasa ktorú vozidlo podľa časového rozpisu prejde, je reprezentovaná jednou inštanciou modelu vozidla.

$$\begin{aligned} &\langle S_i, q_{init,i}, \delta_{int,i}, ta_i, Y_i, \lambda_i, X_i, \delta_{ext,i} \rangle \\ S &= \{\text{Na ceste}, \text{Na zastávke}, \text{Nakladá cestujúcich}\} \\ q_{init} &= \text{Na ceste} \\ \delta_{int} &= \{\text{Na ceste} \rightarrow \text{Na zastávke}, \text{Na zastávke} \rightarrow \text{Nakladá cestujúcich}, \\ &\quad \text{Nakladá cestujúcich} \rightarrow \text{Na ceste}\} \\ ta &= \{\text{Na ceste} \rightarrow \infty, \text{Na zastávke} \rightarrow 0, \text{Nakladá cestujúcich} \rightarrow 0\} \\ Y &= \{!Príchod, !Nastupovanie, !Odchod\} \\ \lambda &= \{\text{Na ceste} \rightarrow !Príchod, \text{Na zastávke} \rightarrow !Nastupovanie, \\ &\quad \text{Nakladá cestujúcich} \rightarrow !Odchod\} \\ X &= \{?Príchod\_vozidla\} \\ \delta_{ext} &= \{(\text{Na ceste}, ?Príchod\_vozidla) \rightarrow \text{Na zastávke}\} \end{aligned}$$

Na obrázku 3.2 je graficky znázornený stavový automat. Je na ňom vidieť tri stavy modelu, a prechody medzi nimi, ktoré zároveň generujú tri výstupné signály. Stavy modelu sú popísané v tabuľke 3.3 a akcie, ktoré sa vykonajú spolu s generovaním výstupných signálov sú popísané v tabuľke 3.4. Vo formálnej definícii vyššie je vidieť, že model vozidla zotráva v stave **Na ceste** neobmedzene dlho. Prechod z tohoto stavu je podmienený príchodom vstupného signálu *?Príchod\_vozidla*. Pôvod tohoto signálu bude popísaný neskôr v rámci definície zloženého modelu.



Obr. 3.2: Stavový automat modelu vozidla

Stav	Popis
Na ceste	vozidlo sa pohybuje medzi zastávkami
Na zastávke	vozidlo práve prišlo na zastávku
Nakladá cestujúcich	vozidlo nakladá cestujúcich

Tabuľka 3.3: Stavy modelu vozidla

Výstupný signál	Akcia ktorú spúšťa
!Príchod	výstup cestujúcich podľa koeficientu danej zastávky, ak je koeficient 0,7, vystúpi 70 percent cestujúcich
!Nastupovanie	nástup cestujúcich do vozidla, tí čo sa nezmestia odchádzajú nespokojní
!Odchod	okrem aktivácie prepojených vstupných signálov sa nič iné nevykonáva

Tabuľka 3.4: Výstupné signály modelu vozidla

### Obsluha zastávky

Každé vozidlo začína na prvej zastávke v rozpise, ale počas simulácie sa musí pohybovať medzi zastávkami. Proces ktorý vozidlo vykoná pri príchode na každú zastávku je popísaný v algoritme 2. Tento proces je vyvolaný príchodom vstupného signálu *?Príchod\_vozidla*.

---

#### Algoritmus 2: Obsluha zastávky

---

prepojenie výstupných signálov vozidla so vstupnými signálmi zastávky;  
aktivácia výstupného signálu **!Príchod**;  
aktivácia výstupného signálu **!Nastupovanie**;  
aktivácia výstupného signálu **!Odchod**;

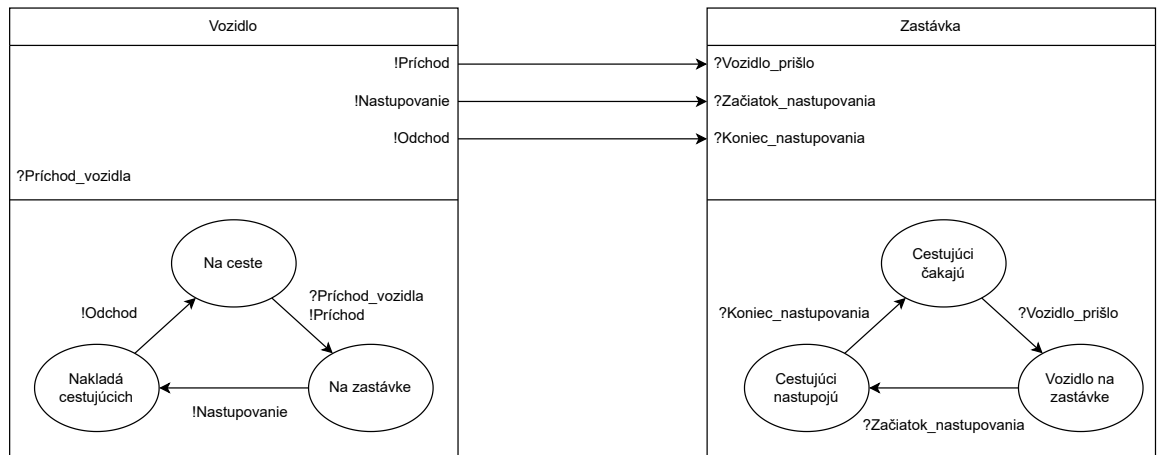
---

## Zložený model

Zložený simulačný model v tejto práci pozostáva z dvoch atomických modelov, modelu zastávky a vozidla. Prepojenie týchto atomických modelov je graficky znázornené na obrázku 3.3.

$$\begin{aligned}
 D &= \{Zastávka, Vozidlo\} \\
 M_{Zastvka} &= \langle \dots \rangle \\
 M_{Vozidlo} &= \langle \dots \rangle \\
 I_{Zastvka} &= \emptyset \\
 I_{Vozidlo} &= \{Zastávka\} \\
 X_{self} &= \emptyset \\
 Y_{self} &= \{!Príchod\_vozidla\} \\
 select &= \{\{Zastvka, Vozidlo\} \rightarrow Vozidlo, \{Zastvka\} \rightarrow Zastvka, \{Vozidlo\} \rightarrow Vozidlo\} \\
 Z_{i,j,self} &= \{!Príchod\_vozidla \rightarrow ?Príchod\_vozidla, \\
 &\quad !Príchod \rightarrow ?Vozidlo\_prišlo, \\
 &\quad !Nastupovanie \rightarrow ?Začiatok\_nastupovania, \\
 &\quad !Odchod \rightarrow ?Koniec\_nastupovania\}
 \end{aligned}$$

Samotný zložený model má navyše ešte jeden výstupný signál, ktorý je prepojený s modelom vozidla. Týmto signálom sa spúšťa rutina obsluhy zastávky v rámci vozidla. Vozidlo sa dostáva zo stavu **Na ceste** do stavu **Na zastávke**. Výstupný signál zloženého modelu *!Príchod\_vozidla* sa generuje pri spracovaní udalosti z kalenádara udalostí, oba tieto koncepty sú popísané neskôr.



Obr. 3.3: Schéma zloženého simulačného modelu

## Udalosť príchodu vozidla na zastávku

Udalosť príchodu vozidla na zastávku vyvolá spustenie rutiny obsluhy cestujúcich, ktorí čakajú na zastávke. Táto rutina je popísaná v algoritme 2. Parametre tejto udalosti sú:

- **Čas príchodu** — čas, kedy vozidlo prichádza na zastávku
- **Zastávka** — zastávka, na ktorej sa udalosť vykonáva
- **Vozidlo** — vozidlo, ktoré prichádza na zastávku

Vozidlo začína udalosť v stave **Na ceste**, príde na zastávku uvedenú v parametri udalosti, vykoná rutinu nástupu a výstupu cestujúcich a odíde na ďalšiu zastávku. Za jednu udalosť teda vozidlo prejde všetkými svojimi vnútornými stavmi, udalosť končí opäť v stave **Na ceste**. Zastávka ktorá je uvedená v parametri udalosti tiež prejde všetkými svojimi vnútornými stavmi vďaka prepojeniu výstupných signálov vozidla so vstupnými signálmi zastávky.

## Kalendár udalostí

Kalendár udalostí je základným prvkom modelu, ktorý riadi priebeh simulácie. V kalendári sú uložené všetky udalosti, ktoré sa majú v modeli stať. Tieto udalosti sú v prípade tohto modelu príchody vozidiel na zastávky. V rámci jednej udalosti sa vykoná rutina obsluhy cestujúcich čakajúcich na zastávke. Všetky udalosti sú naplánované na začiatku simulácie, pretože vďaka časovému rozpisu presne vieme všetky príchody vozidiel na jednotlivé zastávky. Príchody na prvú zastávku sú dané hlavným časovým rozpisom, príchody na ostatné zastávky sú odvodené od času potrebného na prejdienie z jednej zastávky na druhú. Pseudokód plánovania udalostí je uvedený v algoritme 3 a je prevzatý zo študijnej opory predmetu Modelovanie a simulácie [10], ktorý je vyučovaný na Fakulte informatiky Vysokého učení technického v Brne. V priebehu simulácie už netreba žiadne ďalšie udalosti plánovať.

---

### Algoritmus 3: Plánovanie udalostí

---

```
for čas v časovom rozpise počiatkovej zastávky do  
    Vytvorí sa vozidlo;  
    for zastávka na linke do  
        Pridá sa udalosť príchodu vozidla na danú zastávku;
```

---

Po naplánovaní všetkých udalostí sa začne simulácia. Postupne sa vykonávajú jednotlivé udalosti a simulácia sa diskretné posúva vpred, až kým nie je kalendár udalostí prázdny

alebo nie je dosiahnutý koncový čas simulácie. Pseudokód riadenia simulácie je popísaný v algoritme 4.

---

**Algoritmus 4:** Simulácia riadená udalosťami

---

```
while kalendár udalostí nie je prázdny do  
    Vyberie sa udalosť s najmenším časom v kalendári;  
    Udalosť sa odstráni z kalendára;  
    if čas udalosti > koncový čas simulácie then  
        | Simulácia končí;  
    Modelový čas sa nastaví na čas udalosti;  
    Udalosť sa vykoná (Vozidlo príde na zastávku a obsluži cestujúcich);
```

---

Pre potreby časového rozpisu stačí simulovať jeden deň, v prípade rôznych rozpisov v čase pracovných dní a v čase sviatkov a víkendov ide o kompletne iný rozpis, ktorý sa musí simulovať a optimalizovať samostatne. Všetky naplánované udalosti by mali byť v rámci jedného dňa, preto by simulácia mala končiť vždy vyprázdnením kalendára udalostí a nie prekročením koncového času.

### 3.3 Výstupy simulačného modelu

Hlavný zmysel simulácie je zber informácií, ktoré by sme inak museli získať z pozorovania reálnej prevádzky linky. Efektivitu každého jedného časového rozpisu, ktorý program vytvorí je potrebné otestovať a ohodnotiť aby sme našli ten najlepší z nich, skôr ako ho začneme používať v reálnej prevádzke. Na ohodnotenie jednotlivých rozpisov slúžia štatistiky získané počas simulácie. Štatistiky sa zbierajú pre každú zastávku 3.6. aj pre každé vozidlo 3.7. osobitne. Následne sa agregujú do jedného objektu **Statistics**, ktorého obsah je popísaný v tabuľke 3.5.

Parameter	Popis
<b>totalNumberOfBuses</b>	Počet spojov v priebehu dňa na linke
<b>busStopStatistics</b>	Agregované štatistiky zastávok
<b>busStatistics</b>	Agregované štatistiky vozidiel

Tabuľka 3.5: Trieda **Statistics**

#### Počet spojov v priebehu dňa na linke

Tento údaj slúži pre optimalizáciu, z pohľadu dopravného podniku chceme, aby bol počet spojov čo najmenší aby ušetrili na prevádzkových nákladoch. Z pohľadu cestujúcich chceme aby bol počet spojov čo najväčší, aby nemuseli čakať na spoj dlhšie ako je nutné. Je potrebné nájsť kompromis medzi týmito dvoma extrémami.

#### Koľko cestujúcich prišlo na zastávku za deň

Celkový počet cestujúcich za deň je dobrým indikátorom toho, ako sú namáhané jednotlivé časti linky. V prípadoch, kedy na zastávky v strede trasy prichádza oveľa viac cestujúcich ako na koncové zastávky, posilňuje dopravný podnik práve túto strednú časť ďalšími spojmi.

Parameter	Popis
totalPassengersArrived	Kolko cestujúcich prišlo na zastávku za deň
totalPassengersDeparted	Kolko cestujúcich vytúpilo na zastávke za deň
totalPassengersLeftUnboarded	Počet cestujúcich, ktorí sa nezmestili do vozidla
totalTimeSpentWaiting	Čas, ktorý cestujúci strávili čakaním na zastávke
passengersArrivedPerHour	Počet cestujúcich, ktorí prišli na zastávku v jednotlivých hodinách
passengersDepartedPerHour	Počet cestujúcich, ktorí vystúpili na zastávke v jednotlivých hodinách
passengersLeftUnboardedPerHour	Počet cestujúcich, ktorí sa nezmestili do vozidla v jednotlivých hodinách
timeSpentWaitingPerHour	Čas, ktorý cestujúci strávili čakaním na zastávke v jednotlivých hodinách

Tabuľka 3.6: Trieda `busStopStatistics`

Parameter	Popis
capacity	Kapacita vozidla
seats	Počet miest na sedenie
averageLoad	Priemerná naplnenosť vozidla
averageLoadInPercent	Priemerná naplnenosť vozidla v percentách
totalPassengersTransported	Celkový počet prevezených cestujúcich
loadPerBusStop	Naplnenosť vozidla na jednotlivých zastávkach
loadInPercentPerBusStop	Naplnenosť vozidla na jednotlivých zastávkach v percentách
passengerSatisfactions	Spokojnosť jednotlivých zákazníkov

Tabuľka 3.7: Trieda `busStatistics`

### Kolko cestujúcich vytúpilo na zastávke za deň

Je potrebné aby sme v rámci simulácie rátali aj s cestujúcimi, ktorí z vozidiel vystupujú, inak by sa kapacita vozidla hneď naplnila. Ide o dôležitú časť reálneho systému, ktorú nemôžeme zanedbať ani pri simulácii. Slúži na validáciu modelu.

### Počet cestujúcich, ktorí sa nezmestili do vozidla

Občas sa stane, že je vozidlo tak preplnené, že nie všetci cestujúci, ktorí čakali na zastávke do neho nastúpia. Rozpisy, pri ktorých sa tento jav vyskytne budú mať horšie hodnotenie.

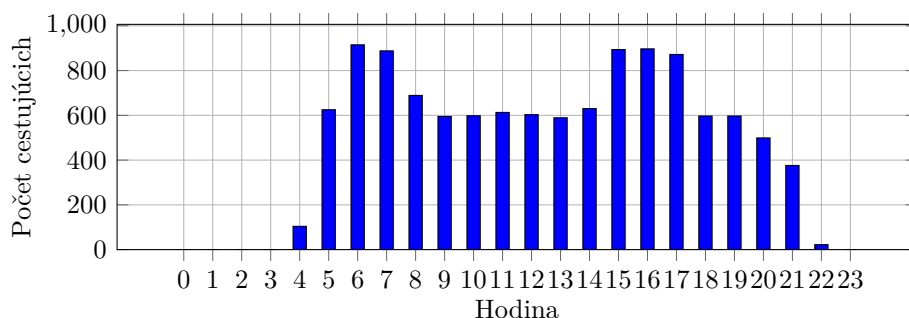
### Čas, ktorý cestujúci strávili čakaním na zastávke

Celkový čas, ktorý dokopy čakali cestujúci na zastávke. Možno atraktívnejšia informácia je priemerný čas, ktorý čakal jeden cestujúci. V oboch prípadoch chceme, aby bol tento čas čo najnižší.

### Počet cestujúcich, ktorí prišli na zastávku v jednotlivých hodinách

Tento údaj má skôr informatívny charakter, pretože ide o vstup programu. Algoritmus sa snaží vytvoriť rozpis, ktorý by bol schopný prepraviť všetkých cestujúcich. Na grafe 3.4 je

vidieť, že v priebehu dňa sa počet cestujúcich prichádzajúcich na zastávku mení. Obzvlášť v ranných a poobedných hodinách je tento počet výrazne vyšší ako počas dňa. V týchto časoch by malo chodiť viac spojov.



Obr. 3.4: Počet cestujúcich prichádzajúcich na zastávku za hodinu

### Počet cestujúcich, ktorí vystúpili na zastávke v jednotlivých hodinách

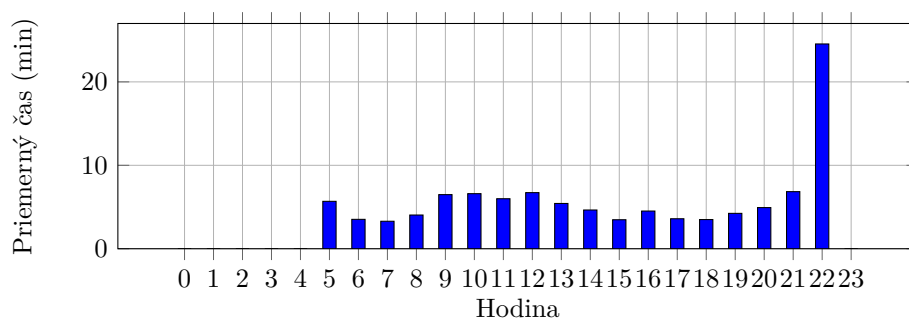
Ide o rovnaký údaj ako bol počet cestujúcich, ktorí vystúpili na zastávke za celý deň, ale rozdelený na jednotlivé hodiny dňa.

### Počet cestujúcich, ktorí sa nezmestili do vozidla v jednotlivých hodinách

Tento údaj je vhodný na zistenie, ktorá časť dňa je poddimenzovaná a bolo by vhodné v tom čase posilniť linku o viacej spojov.

### Čas, ktorý cestujúci strávili čakaním na zastávke v jednotlivých hodinách

Čas strávený čakaním na zastávke by mal byť počas celého dňa približne rovnaký. Na grafe 3.5 je vidieť ako sa priemerný čas čakanie pohybuje približne 5 minút, ale o desiatej hodine večer tento čas náhle vzrastie. To znamená, že by v tejto hodine malo byť nasadených viac spojov.



Obr. 3.5: Priemerný čas strávený čakaním za hodinu

### Kapacita vozidla

Tento parameter je dôležitý, pretože na základe neho sa rozhoduje, koľko cestujúcich môže nastúpiť do vozidla. Zároveň slúži pre výpočet naplnenosti vozidla v percentách. Príliš malá

alebo príliš veľká naplnenosť vozidla môže byť nežiaduca. Na ohodnotenie toho, ako moc je vozidlo naplnené teda treba poznať aj jeho kapacitu.

### **Počet miest na sedenie**

Počet miest na sedenie hrá rolu pri výpočte spokojnosti nastupujúceho cestujúceho. Pokiaľ má možnosť si sadnúť jeho spokojnosť bude vyššia ako keď by mal stáť.

### **Priemerná naplnenosť vozidla**

Priemerná naplnenosť vozidla je dôležitý parameter, ktorý nám hovorí, ako moc sú vozidlá využívané. Z pohľadu cestujúcich je lepšie, ak je vozidlo prázdnejšie, ale z pohľadu dopravného podniku je to nevyužitý potenciál. Preto je potrebné nájsť kompromis medzi týmito dvoma požiadavkami. To aká naplnenosť je optimálna je už úlohou analytika, ktorý bude tento program využívať. Povedzme, že sa rozhodne pre naplnenosť 70%. Potom čím ďalej bude priemerná naplnenosť vozidiel od tejto hodnoty, tým horšie bude rozpis ohodnotený.

### **Celkový počet prevezených cestujúcich**

Celkový počet prevezených cestujúcich závisí od toho, koľko cestujúcich prišlo na jednotlivé zastávky a či sa im podarilo nastúpiť do vozidiel. V ideálnom prípade sa tento počet bude rovnať počtu cestujúcich, ktorí prišli na zastávky. V extrémnych prípadoch sa môže stať, že niektorí cestujúci zostanú na zastávkach a nezoberie ich žiadne vozidlo. V prípade štatistiky jednotlivých vozidiel, môže byť tento počet použitý na ich porovnanie. Počas dňa by mali všetky vozidlá prepraviť približne rovnaký počet cestujúcich. Ak tomu tak nie je a sú vyťažené nerovnomerne, rozpis by sa mal ohodnotiť negatívne.

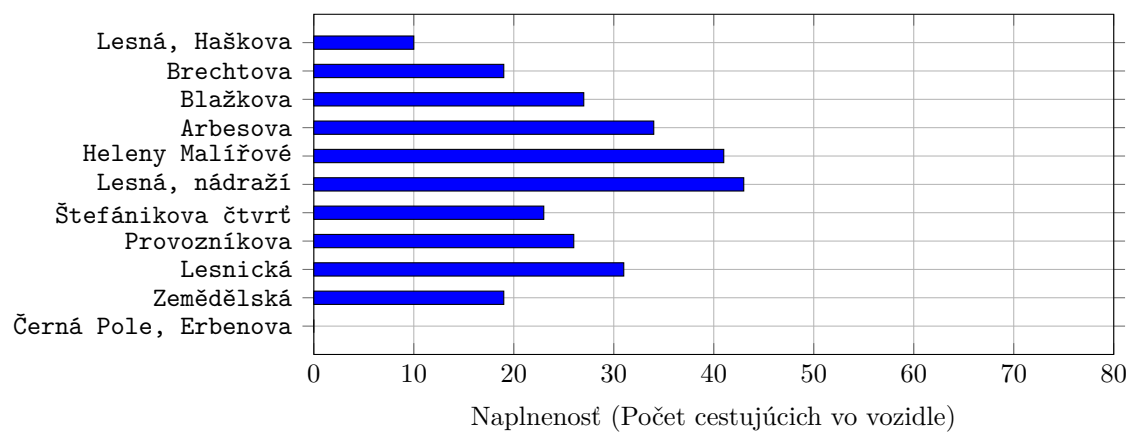
### **Naplnenosť vozidla na jednotlivých zastávkach**

Na grafe 3.6 je znázornené ako sa naplnenosť vozidiel mení počas ich jazdy. Konkrétne sa jedná o linku 46 v Brne. Naplnenosť sa postupne zvyšuje až kým nepríde na jednu z hlavných prestupných zastávok, Štefánikova čtvrt'. Tu vystúpi veľké množstvo cestujúcich a naplnenosť vozidla sa zníži. Pokiaľ má linka príliš veľké výkivy v naplnenosti vozidiel naprieč svojou trasou, môže to byť znakom toho, že je potrebné trasu na určitých úsekoch posilniť inou linkou.

### **Spokojnosť jednotlivých zákazníkov**

Spokojnosť zákazníkov je kľúčová pri optimalizácii časového rozpisu, na základe nej sa bude rozpis ohodnocovať. Spokojnosť sa počíta pre každého cestujúceho a nakonci sa vypočíta priemer za celý deň.





Obr. 3.6: Priemerná naplnenosť vozidla na jednotlivých zastávkach

## Kapitola 4

# Optimalizácia časového rozpisu linky mestskej hromadnej dopravy

Táto kapitola pojednáva o potrebných vstupoch, obmedzeniach, použitej optimalizačnej metóde a výstupoch optimalizácie.

Na optimalizáciu bol použitý genetický algoritmus. Genetický algoritmus je heuristická optimalizačná metóda, ktorá sa inšpiruje evolučnou biológiou. Je založená na princípe prírodného výberu, kde sa najlepšie jedince z populácie vyberajú na reprodukciu a vytvárajú novú generáciu. Vlastnosti jedincov by sa mali v priebehu generácií zlepšovať, až kým sa nenájde optimálne riešenie. Okrem nájdenia optimálneho riešenia, môže byť genetický algoritmus ukončený aj dosiahnutím maximálneho počtu generácií alebo stagnáciou kvality riešenia. Na ohodnotenie jedinca sa používa fitness funkcia, ktorá hodnotí kvalitu riešenia na základe zadaných kritérií. V prípade časových rozpisov MHD sa na výpočet fitness funkcie používajú štatistiky zo simulácie, ktoré sú popísané v kapitole 3.3.

### 4.1 Vstupy potrebné od používateľa

Na to aby mohol algoritmus fungovať, je potrebné zadať niekoľko vstupných parametrov. Najdôležitejším z nich sú informácie o zastávkach. O každej zastávke je potrebné zadať:

- **Názov zastávky** — názov zastávky, ktorý sa zobrazuje na výstupoch
- **Čas príchodu** — čas potrebný na prejsenie od prvej zastávky po túto zastávku
- **Štatistika o príchodoch cestujúcich** — priemerný počet prichádzajúcich cestujúcich v každej hodine dňa
- **Pravdepodobnosť výstupu cestujúceho** — údaj potrebný pre priebežné vyprázdenie vozidla

Tieto informácie sa zadávajú vo forme textového súboru, každá zastávka je na samostatnom riadoku a jednotlivé údaje sú oddelené dvojbodkou. Príklad toho ako by mali vyzerat záznamy o zastávkach:

```
Lesná, Haškova:0:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0  
Brechtova:1:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0.1  
Blažkova:2:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0.1
```

...

Ďalšie dôležité informácie o linke sú:

- **Kapacita vozidla** — maximálny počet cestujúcich, ktorý sa zmestí do vozidla
- **Miest na sedenie** — počet miest na sedenie v vozidle
- **Celkové náklady** — udávané v Kč/100 miesto-km (tabulková hodnota)
- **Dĺžka trasy** — dĺžka trasy v km

Kapacita vozidla a počet miest na sedenie sú potrebné pre výpočet naplnenosti vozidla, od ktorého sa potom odvíja spokojnosť cestujúcich. Celkové náklady na 100 miestokilometrov a dĺžka trasy sú potrebné pre výpočet nákladov na prevádzku linky. Podrobnejší popis výpočtu celkovej spokojnosti a nákladov na prevádzku je v kapitole 4.2.

Ďalšími dôležitými vstupnými parametrami sú parametre genetického algoritmu:

- **Veľkosť populácie** — počet jedincov v jednej generácii
- **Počet generácií** — počet generácií, ktoré sa majú vykonať
- **Pravdepodobnosť mutácie** — pravdepodobnosť, že sa v jedincovi vyskytne mutácia
- **Maximálny počet spojov za hodinu** — maximálny počet spojov, ktorý môže byť vygenerovaný za hodinu

Poslednou informáciou, ktorá je od používateľa potrebná sú obmedzenia na počet spojov za hodinu. Používateľ takto môže nastaviť fixný počet spojov, ktorý sa má vygenerovať za hodinu. Toto sa využíva hlavne pri plánovaní návaznosti liniek, kedy je potrebné aby sa niektoré spoje stretli na prestupnej zastávke. Obvykle sa to robí tak, že sa nastaví rovnaký počet spojov pre všetky prestupné linky, ktoré sa na tejto zastávke stretávajú. Toto obmedzenie využíva aj predbežná analýza, ktorá sa vykonáva pred optimalizáciou. V prípade ak pre niektorú hodinu neexistuje údaj o priemernom počte prichádzajúcich cestujúcich naprieč všetkými zastávkami, nastaví sa fixný počet spojov v danej hodine na 0. Tento krok výrazne prispieva k lepším výsledkom optimalizácie, pretože dopredu vieme, že v tejto hodine nebude žiaden spoj potrebný. Zároveň s nižším počtom spojov vo všetkých rozpisoch počas optimalizácie sa výpočet výrazne urýchli.

## 4.2 Základná implementácia genetického algoritmu

Na optimalizáciu časového rozpisu bol použitý genetický algoritmus. Genetický algoritmus je heuristická optimalizačná metóda, ktorá sa inšpiruje Darwinovou teóriou evolúcie. Podľa citátu od Charlesa Darwina, "V prírode je prežitie a úspech otázkou tých najschopnejších a najpríťažlivejších", aj genetický algoritmus sa snaží nájsť najlepšie riešenie problému pomocou evolučného procesu.

Používajú sa postupy ako je náhodná mutácia jedinca a rozne typy výberu a kríženia jedincov. Myšlienka je, že najlepší jedinci v generácii buď splňujú požiadavky alebo sú veľmi blízko k optimálnemu riešeniu. Predpokladá sa, že tieto jedinci majú najlepšie predispozície na to, aby ich potomkovia boli o krok bližšie k optimálnemu riešeniu. Využitie genetického algoritmu je efektívne v prípadoch, kedy by deterministické metódy trvali príliš dlho alebo

by vôbec neboli schopné nájsť optimálne riešenie. Genetický algoritmus bol prvýkrát predstavený okolo roku 1960 Johnom Hollandom na Univerzite v Michigane. Tieto informácie sú prevzaté zo zborníku z konferencie ICCES 2019 [6].

---

#### Algoritmus 5: Genetický algoritmus

---

```

Inicializácia prvej populácie;
Definovanie fitness funkcie;
Výpočet fitness funkcie pre každého jedinca v populácii;
while nie je nájdené optimálne riešenie alebo nebol dosiahnutý maximálny počet
generácií do
    Výber rodičov z populácie;
    Kríženie rodičov na vytvorenie potomkov;
    Mutácia potomkov;
    Výpočet fitness funkcie pre každého potomka;
Výber najlepšieho jedinca z populácie;

```

---

V prípade optimalizácie časového rozpisu MHD sa jedincami stávajú samotné časové rozpisy. Na základe simulácie jedného dňa na linke s použitým časovým rozpisom sa získajú štatistiky, ktoré sa následne použijú na ohodnotenie kvality jedinca.

### Reprezentácia jedinca

V genetickom algoritme je jedinec reprezentovaný takzvaným chromozómom. Chromozóm je reťazec génov, informácií o jedincovi. V prípade časového rozpisu MHD je chromozóm reprezentovaný ako pole obsahujúce 24 čísel (viď tabuľku 4.1), ktoré predstavujú počet odchodov vozidiel z prvej zastávky linky v jednotlivých hodinách dňa.

0	0	0	0	0	5	7	9	9	9	7	6	6	6	6	7	9	9	7	7	6	5	4	3
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Tabuľka 4.1: Príklad chromozómu

### Fitness funkcia — výpočet cieľov

Pre ohodnotenie kvality časového rozpisu sa zvolili dva hlavné ciele:

- **Spokojnosť cestujúcich** — priemerné percentuálne zhodnotenie spokojnosti všetkých cestujúcich, ktorí prišli za deň na zastávku
- **Náklady na prevádzku** — celkové náklady na prevádzku linky za deň, udávané v Kč/100 miesto-km

Výpočet spokojnosti cestujúceho sa vykonáva v čase nástupu do vozidla. Zhodnocuje sa aktuálna naplnenosť vozidla a na základe toho sa vypočíta percentuálne zhodnotenie spokojnosti. Potrebné parametre pre výpočet sú kapacita vozidla, počet miest na sedenie a aktuálna naplnenosť vozidla. Výpočet sa riadi nasledujúcimi troma podmienkami:

1. Ak sú vo vozidle voľné miesta na sedenie, cestujúci si sadne a jeho spokojnosť je 1
2. Ak je vozidlo úplne plné a cestujúci sa do neho nezmestí, jeho spokojnosť je 0

3. Ak je vo vozidle miesto, ale už len na státie, spokojnosť vozidla závisí od aktuálnej naplnenosti vozidla a je vypočítaná ako:

$$\text{spokojnosť} = 1 - \frac{\text{aktualna naplnenosť vozidla} - \text{počet miest na sedenie}}{\text{kapacita vozidla} - \text{počet miest na sedenie}} \quad (4.1)$$

Nakoniec sa vypočíta priemer spokojnosti všetkých cestujúcich. Hodnota sa pohybuje od 0 do 1 a genetický algoritmus uprednostňuje vyššie hodnoty, teda spokojnejších cestujúcich.

Celkové náklady na prevádzku linky sú vypočítané z počtu spojov v časovom rozpise, kapacity použitých vozidiel, dĺžky trasy a nákladov na jeden miestokilometer. Jeden miestokilometer je definovaný ako vzdialenosť jedného kilometra, ktorú prejde jedno miesto vo vozidle [7]. Potom vozidlo s kapacitou 80 miest, ktoré prejde 10 km, prejde 800 miestokilometrov. Miestokilometre sa môžu násobiť aj počtom vozidiel, potom dve takéto vozidlá by prešli 1600 miestokilometrov. Tento údaj je potrebný pretože náklady vo finančnom pláne Dopravného podniku mesta Brno sú udávané v českých korunách na 100 miestokilometrov. Tento finančný plán je súčasťou zmluvy medzi mestom Brno a Dopravným podnikom mesta Brna [14]. Celý výpočet nákladov na prevádzku linky teda závisí od štyroch premenných a je daný vzorcom:

$$\text{celkové náklady} = \frac{\text{dĺžka trasy} \cdot \text{počet spojov} \cdot \text{kapacita vozidla}}{100} \cdot \text{náklady} \quad (4.2)$$

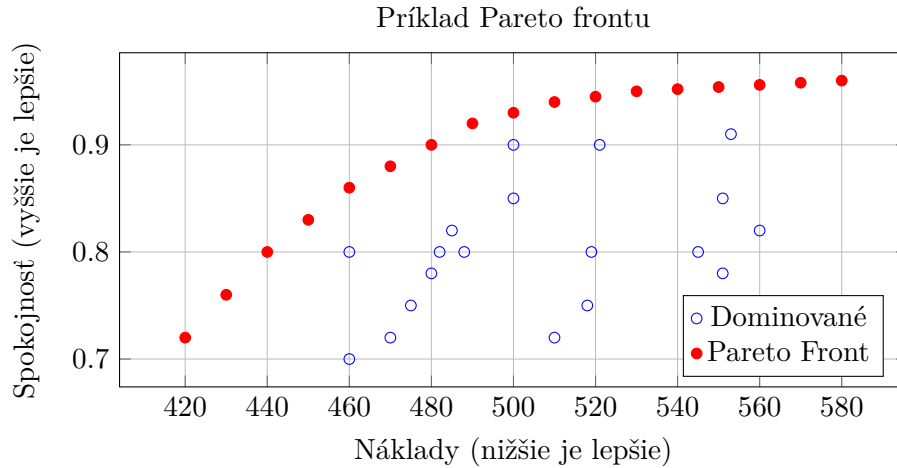
Tieto dva ciele sú v konflikte, pretože čím viac spojov je na linke, tým je spokojnosť cestujúcich vyššia, ale náklady na prevádzku sú tiež vyššie. Tento problém rieši multi-objektívna optimalizácia, ktorá sa snaží nájsť optimálne riešenie pre oba ciele naraz.

## Genetický algoritmus s viacero cieľmi

Genetický algoritmus, v ktorom sa snažíme dosiahnuť optimálneho kompromisu medzi viacero cieľmi sa nazýva multi-objektívny genetický algoritmus. Jeho myšlienka spočíva v nájdení Pareto frontu, čo je množina všetkých možných riešení, ktoré sú optimálne z pohľadu všetkých cieľov. To znamená, že neexistuje žiaden iný jedinec, ktorý by bol lepší v oboch cieľoch naraz. Ak je jeden jedinec lepší v oboch cieľoch, hovoríme, že dominuje iného jedinca. V prípade, že dominuje iba v jednom, ale v druhom nie, patria do rovnakej skupiny, sú si rovnocenní. Ak je jedinec horší v oboch cieľoch, hovoríme, že je dominovaný. Každá generácia sa rozdelí do rovnocenných skupín, pri výbere rodičov sa uprednostňujú jedinci z lepšej skupiny. Algoritmus sa snaží nájsť Pareto front, množinu tých najlepších možných jedincov. Postupnou evolúciou jedinci získavajú lepšie vlastnosti a čím novšia je generácia, tým viac sa približuje k hľadanému Pareto frontu. Na obrázku 4.1 je znázornený Pareto front pre dva ciele, spokojnosť a náklady. Multi-objektívne genetické algoritmy dokážu pracovať aj s viac ako dvoma cieľmi. Pre optimalizáciu časového rozpisu však stačia dva, jeden zo strany zákazníka (spokojnosť) a druhý zo strany dopravného podniku (náklady). Tieto základné informácie o fungovaní multi-objektívnych genetických algoritmov s použitím Pareto frontu sú prevzaté z článku [9].

## 4.3 Multi-objektívny genetický algoritmus

Pre účely multi-objektívnej optimalizácie bol použitý algoritmus NSGA-II (Non-dominated Sorting Genetic Algorithm II). V článku [2] je porovnanie tohto algoritmu s inými multi-objektívnymi genetickými algoritmi. NSGA-II vo viacerých testoch preukázal, že má



Obr. 4.1: Príklad Pareto frontu

lepšie rozloženie a diverzitu jedincov v Pareto fronte ako iné algoritmy. Práve pre túto vlastnosť bol zvolený aj pre túto prácu. Okrem testov je v článku [2] aj podrobný popis fungovania algoritmu. Algoritmy 10, 6, 7 a 8 boli z tohto článku prevzaté a mierne upravené pre účely tejto práce.

NSGA-II je založený na triedení jedincov do frontov na základe ich dominancie nad inými jedincami. Pri výbere jedincov pri krížení sa uprednostňujú jedinci z lepších frontov. V prípade, že treba porovnať dvoch jedincov z rovnakého frontu, použije sa zhuková vzdialenosť.

Zhluková vzdialenosť zaisťuje vysokú diverzitu jedincov, pretože sa uprednostňujú jedinci, ktorí sú od seba najďalej, teda majú veľké rozdiely v hodnotách cieľov. Pre jej výpočet sa najprv jedinci frontu zoradia podľa hodnoty jedného z cieľov. Zhuková vzdialenosť daného jedinca pre daný cieľ je pomer medzi vzdialenosťami jedinca od jeho susedov vo fronte a vzdialenosťou medzi prvým a posledným jedincom vo fronte. Zhukové vzdialenosti pre jednotlivé ciele sa následne sčítajú a tým sa získa celková zhuková vzdialenosť jedinca. Celý postup pre priradenie zhukovej vzdialenosti je popísaný v algoritme 7.

Oproti klasickému genetickému algoritmu NSGA-II spracováva elitizmus inak. V NSGA-II neexistuje konštanta, ktorá by udávala počet elitných jedincov, ktorí sa prenesú do ďalšej generácie. Pri vytváraní novej generácie sa najprv vytvorí nová populácia z rodičov a ich potomkov. Táto, dvojnásobne veľká populácia sa následne rozdelí do frontov a pre každého jedinca sa vypočíta zhuková vzdialenosť na základe jeho vzdialenosti od svojich susedov vo fronte. Potom sa najlepšie fronty prenesú do novej generácie. Posledný prenášaný front sa pravdepodobne nezmestí celý, vtedy sa vyberie toľko jedincov z daného frontu aby bola populácia rovnako veľká naprieč všetkými generáciami. Jedinci z posledného frontu sa vyberajú na základe zhukovej vzdialenosti, aby sa zachovala vyššia diverzita.

## Rýchle nedominantné triedenie

Rýchle nedominantné triedenie je základom NSGA-II. Triedi jedincov do frontov na základe ich dominancie nad inými jedincami. Jedince, ktoré sa navzájom nedominujú, patria do rovnakej skupiny, frontu, preto sa toto triedenie nazýva nedominantné. Triedenie sa vykonáva v dvoch krokoch. V prvom kroku sa musia porovnať všetci jedinci navzájom. Pre každého jedinca  $p$  v prvom cykle algoritmu 6 sa vypočíta množina  $S_p$ , ktorá obsahuje všet-

kých jedincov, ktorých dominuje  $p$  a počet  $n_p$ , ktorý udáva počet jedincov, ktorí dominujú  $p$ . Pokiaľ je  $n_p = 0$ , jedinec  $p$  nie je nikým dominovaný a pripojí sa do prvého frontu. Taktiež sa mu priradí hodnota  $p_{rank} = 1$ . Rank je hodnota, ktorá udáva v akom fronte sa jedinec nachádza, čím nižšia hodnota, tým je jedinec lepší.

Následuje druhý hlavný cyklus algoritmu 6, ktorý vytvorí zvyšné fronty. Princíp je taký, že pre každého jedinca  $p$  v posledne vytvorenom fronte sa upraví hodnoty  $n_q$  všetkých jedincov  $q$  z množiny  $S_p$ . To v podstate znamená odstránenie jedinca  $p$  z riešeného problému. Na konci tohto cyklu by sa mal z problému odstrániť celý front  $F_i$ . To znamená, že niektorí jedinci by sa mali stať tými najlepšími v tomto obmedzenom probléme. Práve títo jedinci sa pridávajú do ďalšieho frontu  $F_{i+1}$ . Tento cyklus sa opakuje až pokiaľ sa do najnovšieho frontu nemá kto pridať.

Toto triedenie teda iteratívne nachádza najlepší front a odstraňuje ho z problému až pokiaľ sa nenájdu všetky fronty.

---

**Algoritmus 6:** Rýchle nedominantné triedenie

---

**Input:** Populácia  $P$

**Output:** Fronty nedominovaných jedincov  $F$

$F \leftarrow \emptyset$  (množina frontov);

**for** každý jedinec  $p \in P$  **do**

$S_p \leftarrow \emptyset$  (množina jedincov, ktorých dominuje  $p$ );

$n_p \leftarrow 0$  (počet jedincov, ktorí dominujú  $p$ );

**for** každý jedinec  $q \in P : q \neq p$  **do**

**if**  $p$  dominuje  $q$  **then**

$S_p \leftarrow S_p \cup \{q\}$ ;

**else if**  $q$  dominuje  $p$  **then**

$n_p \leftarrow n_p + 1$ ;

**if**  $n_p = 0$  **then**

$p_{rank} \leftarrow 1$ ;

$F_1 \leftarrow F_1 \cup \{p\}$  (pridanie jedinca  $p$  do prvého frontu);

$i \leftarrow 1$ ;

**while**  $F_i \neq \emptyset$  **do**

$Q \leftarrow \emptyset$  (množina jedincov ďalšieho frontu);

**for** každý jedinec  $p \in F_i$  **do**

**for** každý jedinec  $q \in S_p$  **do**

$n_q \leftarrow n_q - 1$ ;

**if**  $n_q = 0$  **then**

$q_{rank} \leftarrow i + 1$ ;

$Q \leftarrow Q \cup \{q\}$ ;

$i \leftarrow i + 1$ ;

$F_i \leftarrow Q$ ;

**return**  $F$ ;

---

## Priradenie zhlukovej vzdialenosti

V rámci jedného frontu sa jedinci porovnávajú na základe zhlukovej vzdialenosti. Zhluková vzdialenosť je definovaná ako vzdialenosť jedinca od jeho susedov vo fronte. Vypočítava sa pre každý cieľ zvlášť a následne sa spočíta. Najdôležitejšou časťou algoritmu 7 je cyklus, ktorý sa vykonáva pre každý cieľ  $c$ , teda v prípade tejto práce pre spokojnosť zákazníkov a celkové náklady. Výpočet hodnôt oboch cieľov je popísaný v sekcii 4.2. Najprv sa jedinci zoradia podľa dosiahnutých hodnôt cieľa  $c$ . Následne sa vytýčia krajné body, jedincom s najmenšou a najväčšou hodnotou cieľa  $c$  sa priradí nekonečná zhluková vzdialenosť, pretože sú najviac diverzní. Pre všetkých ostatných jedincov vo fronte a pre cieľ  $c$  sa vypočíta zhluková vzdialenosť ako podiel rozdielu hodnôt susedov a rozdielu krajných jedincov. Tieto dielčie vzdialenosti sa postupne sčítavajú do celkovej zhlukovej vzdialenosti jedinca.

---

### Algoritmus 7: Priradenie zhlukovej vzdialenosti

---

**Input:** Front  $L$   
 $l \leftarrow |L|$  (počet jedincov frontu);  
**for** každý jedinec  $p \in L$  **do**  
     $p_{vzdialenosť} \leftarrow 0$  (inicializácia zhlukovej vzdialenosti);  
**for** každý cieľ  $c$  **do**  
     $L \leftarrow \text{sort}(L, c)$  (zoradenie jedincov podľa hodnôt cieľa  $c$ );  
     $L[0] \leftarrow \infty$  (priradenie nekonečnej vzdialenosti najmenšiemu jedincovi);  
     $L[l-1] \leftarrow \infty$  (priradenie nekonečnej vzdialenosti najväčšiemu jedincovi);  
    **for**  $i = 1$  **do**  $l-2$  **do**  
         $L[i]_{vzdialenosť} \leftarrow L[i]_{vzdialenosť} + \frac{L[i+1].c - L[i-1].c}{L[l-1].c - L[0].c};$

---

## Porovnávací operátor $\prec_n$

Informácie získané z rýchleho nedominantného triedenia 6 a priradenia zhlukovej vzdialenosti 7 sa použijú na porovnanie dvoch jedincov. Pokiaľ je jedinec  $p$  z lepšieho frontu ako jedinec  $q$ , teda má lepší, nižší rank,  $p$  dominuje  $q$ . Ak sú jedinci z rovnakého frontu, porovnávajú sa na základe zhlukovej vzdialenosti. Dominuje ten, ktorý má väčšiu zhlukovú vzdialenosť, to znamená, že sa výraznejšie odlišuje od svojich susedov vo fronte. Tento porovnávací operátor je znázornený v algoritme 8.

---

### Algoritmus 8: Porovnávací operátor $\prec_n$

---

**Input:** Jedinci  $p$  a  $q$   
**Output:** True ak  $p$  dominuje  $q$ , inak False  
**return**  $p_{rank} < q_{rank} \vee (p_{rank} = q_{rank} \wedge p_{vzdialenosť} > q_{vzdialenosť})$ ;  
(dá sa rozšíriť aj o obmedzenia);

---

Toto porovnávanie sa dá rozšíriť aj o rôzne obmedzenia. Napríklad, v rámci optimalizácie časového rozpisu máme dva ciele, spokojnosť zákazníkov a celkové náklady, ale porovnávanie sme rozšírili o to, v ktorom rozpise dôjde menej často k úplnému preplneniu vozidla. Sledujeme počet prípadov, kedy sa nejaký cestujúci nezmestil do vozidla. V prípade, že aspoň jeden z porovnávaných jedincov má tento počet väčší ako 0, vyhráva ten, ktorý má



tento počet menší. Všeobecne môže byť obmedzení aj viac, v takom prípade sa porovnáva počet porušenia daných obmedzení. Napríklad ak by sme mali tri rôzne obmedzenia, časový rozpis môže porušovať 0, 1, 2 alebo 3 z nich. Vyhráva ten rozpis ktorý ich porušuje menej. V prípade ak oba rozpisy porušujú rovnaký počet obmedzení, porovnáme ich podľa operátora  $\prec_n$ . Pre potreby tejto práce máme len jedno spomínané obmedzenie, bližšie informácie k implementácii viacerých obmedzení sú v článku [2].

## Inicializácia prvej generácie

Populácia prvej generácie sa musí spracovať trochu inak ako ostatné generácie. Hlavný cyklus algoritmu NSGA-II sa začína spojením rodičov a potomkov do jedného celku. Na začiatku však žiadnych potomkov nemáme. Algoritmus 9 popisuje ako sa vytvorí populácia prvej generácie a jej potomkovia. Najprv sa vytvorí prvá generácia jedincov, tým sa náhodne vygenerujú chromozómy. Následne sa tieto jedinci roztriedia na fronty pomocou rýchleho nedominantného triedenia. V rámci každého frontu sa vypočíta zhluková vzdialenosť každého jedinca. Na základe získaných ohodnotení sa vyberú tí lepší rodičia a za pomoci kríženia a mutácie sa vytvorí prvá generácia potomkov.

---

### Algoritmus 9: Inicializácia prvej generácie

---

**Input:** Veľkosť populácie  $N$   
 $t \leftarrow 0$  (počiatočná generácia);  
 $P_t \leftarrow \emptyset$  (inicializácia populácie);  
**for**  $i = 0$  **do**  $N - 1$  **do**  
     $p_i \leftarrow \text{vytvorenie\_jedinca}()$ ;  
     $P_t \leftarrow P_t \cup \{p_i\}$ ;  
 $F \leftarrow \text{rýchle\_nedominantné\_triedenie}(P_t)$ ;  
**for** *každý front*  $f \in F$  **do**  
     $\text{priradenie\_zhlukovej\_vzdialenosti}(f)$ ;  
 $Q_t \leftarrow \text{vytvorenie\_nových\_potomkov}(P_t)$  (kríženie a mutácia);

---

## Hlavná slučka algoritmu NSGA-II

Po inicializácii prvej generácie sa ostatné môžu spracovávať rovnako. Ako je vidieť v algoritme 10, hlavná slučka začína spojením rodičov a potomkov do jednej populácie. Novovzniknutá populácia sa následne rozdelí do frontov na základe ich dominancie. Následuje kombinovaný cyklus na spracovanie jednotlivých frontov, ktorý ma dve úlohy:

1. Priradenie zhlukovej vzdialenosti jedincov v aktuálnom fronte
2. Priradenie jedincov aktuálneho frontu do novej generácie

Tento cyklus sa vykonáva pokým sa celý nasledujúci front zmestí do novej generácie. Posledný front, ten ktorý sa už nezместí celý, sa musí spracovať osobitne. Prvky z tohto frontu sa usporiadajú na základe zhlukovej vzdialenosti a do novej generácie sa prenesú len tie jedince, ktoré sa tam zmestia. Z novovzniknutej populácie sa následne vytvorí nová generácia potomkov. Vo všeobecnosti sa táto slučka vykonáva pokiaľ nie je dosiahnutý maximálny

počet generácií alebo pokiaľ nebol nájdený optimálny jedinec. V prípade tejto práce sa však bere do úvahy iba maximálny počet generácií.

---

**Algoritmus 10:** Hlavná slučka algoritmu NSGA-II

---

```

 $R_t \leftarrow P_t \cup Q_t$  (zlučovanie rodičov a potomkov);
 $F \leftarrow \text{rýchle\_nedominantné\_triedenie}(R_t)$ ;
 $P_{t+1} \leftarrow \emptyset$  (inicializácia novej populácie);
 $i \leftarrow 0$ ;
while  $|P_{t+1}| + |F_i| \leq N$  do
    |  $\text{priradenie\_zhlukovej\_vzdialenosti}(F_i)$ ;
    |  $P_{t+1} \leftarrow P_{t+1} \cup F_i$ ;
    |  $i \leftarrow i + 1$ ;
 $\text{sort}(F_i, \prec_n)$ ;
 $P_{t+1} \leftarrow P_{t+1} \cup F_i[0 : N - |P_{t+1}|]$ ;
 $Q_{t+1} \leftarrow \text{vytvorenie\_nových\_potomkov}(P_{t+1})$  (kríženie a mutácia);
 $t \leftarrow t + 1$ ;

```

---

### Vytvorenie nových potomkov

Genetický algoritmus by sa nezaobišiel bez vytvárania nových potomkov. Tento proces sa dá rozdeliť na tri hlavné časti:

1. Výber rodičov
2. Kríženie rodičov
3. Mutácia potomkov

Z každého kríženia vzniknú dvaja noví potomkovia, to znamená, že tento proces asa musí vykonať  $N/2$  krát, pokiaľ je veľkosť populácie  $N$ . Druhá možnosť ako ukončiť tento proces je znázornená v algoritme 11, kde sa noví potomkovia pridávajú do množiny  $Q$  pokiaľ je ich počet menší ako  $N$ .

---

**Algoritmus 11:** Vytvorenie nových potomkov

---

```

Input: Populácia  $P$ 
Output: Noví potomkovia  $Q$ 
 $Q \leftarrow \emptyset$  (inicializácia nových potomkov);
while  $|Q| < N$  do
    |  $p_1 \leftarrow \text{výber\_rodiča}(P)$ ;
    |  $p_2 \leftarrow \text{výber\_rodiča}(P)$ ;
    |  $q_1, q_2 \leftarrow \text{kríženie\_rodičov}(p_1, p_2)$ ;
    |  $\text{mutácia\_jedinca}(q_1, )$ ;
    |  $\text{mutácia\_jedinca}(q_2)$ ;
    |  $Q \leftarrow Q \cup \{q_1, q_2\}$ ;
return  $Q$ ;

```

---

## Výber rodiča

Pred tým ako sa budú rodičia medzi sebou krížiť, je potrebné ich vybrať. Pre túto prácu bol zvolený turnajový výber rodiča. Turnajový výber funguje tak, že sa náhodne vyberie  $k$  jedincov z populácie, z ktorých sa turnajovým spôsobom vyberie ten najlepší. V algoritme 12 je zvolený turnajový výber s  $k = 2$ , *binary tournament*, teda sa vyberú dvaja náhodní jedinci a porovnajú sa pomocou porovnávacieho operátora  $\prec_n$ .

Turnajový výber bol pre túto prácu zvolený, pretože je jednoduchý na implementáciu a má dobré výsledky v porovnaní s inými typmi výberu rodičov. V zborníku z konferencie CEC [4] je porovnanie roznych typov výberu rodičov. Turnajový výber má veľmi podobné výsledky ako náhodný výber rodiča, pravdepodobne z dôvodu malého počtu jedincov v turnaji. Vyšší počet by však mohol celú optimalizáciu spomaliť. Pokusmi sme odhadli počet jedincov v turnaji na 2.

---

**Algoritmus 12:** Výber rodiča

---

**Input:** Populácia  $P$

**Output:** Rodič  $p$

$p_1 \leftarrow$  náhodný jedinec z  $P$ ;

$p_2 \leftarrow$  náhodný jedinec z  $P$ ;

**if**  $p_1 \prec_n p_2$  **then**

**return**  $p_1$ ;

**else**

**return**  $p_2$ ;

---

## Kríženie rodičov

Kríženie znamená výmenu génov v chromozómoch rodičov. Tým vznikajú noví jedinci, potomkovia. V algoritme 13 je znázornené kríženie dvoch rodičov. Pre každý gén v chromozómoch rodičov sa náhodne vyberie jeden z rodičov a jeho gén sa priradí prvému potomkovi. Gén z druhého rodiča sa priradí druhému potomkovi. Tento proces sa opakuje pre všetky gény v chromozómoch rodičov. Je vidieť, že takýmto krížením vzniknú dvaja noví jedinci, preto stačí toto kríženie vykonať len  $N/2$  krát, pokiaľ je veľkosť populácie  $N$ .

Tento typ kríženia je známy ako *uniform crossover* a je veľmi populárny v genetických algoritmoch. Porovnanie rôznych typov kríženia je v zborníku z konferencie ICIIP [1]. Na základe výsledkov v tomto článku bol zvolený typ kríženia s najväčšou diverzitou výsledkov aby sa zamedzilo predčasnej konvergencii, teda aby sa algoritmus nezastavil na lokálnom minime.

## Mutácia jedinca

Posledným krokom v procese vytvorenia nových potomkov je mutácia. Mutácia zvyšuje diverzitu jedincov a zabráňuje predčasnému uviaznutiu v lokálnych extrémoch. V algoritme 14 je znázornená mutácia jedinca. Pre každý gén v chromozóme sa náhodne vyberie číslo z intervalu  $[0, 1]$ . Ak je toto číslo menšie ako pravdepodobnosť mutácie, gén sa nahradí náhodným génom.

---

**Algoritmus 13:** Kríženie rodičov

---

**Input:** Rodičia  $p_1$  a  $p_2$   
**Output:** Potomkovia  $q_1$  a  $q_2$   
**for** každý gén  $g$  v  $p_1$  **do**  
    **if**  $\text{random}(0, 1) < 0,5$  **then**  
         $q_1[g] \leftarrow p_1[g];$   
         $q_2[g] \leftarrow p_2[g];$   
    **else**  
         $q_1[g] \leftarrow p_2[g];$   
         $q_2[g] \leftarrow p_1[g];$   
**return**  $q_1, q_2;$

---

---

**Algoritmus 14:** Mutácia jedinca

---

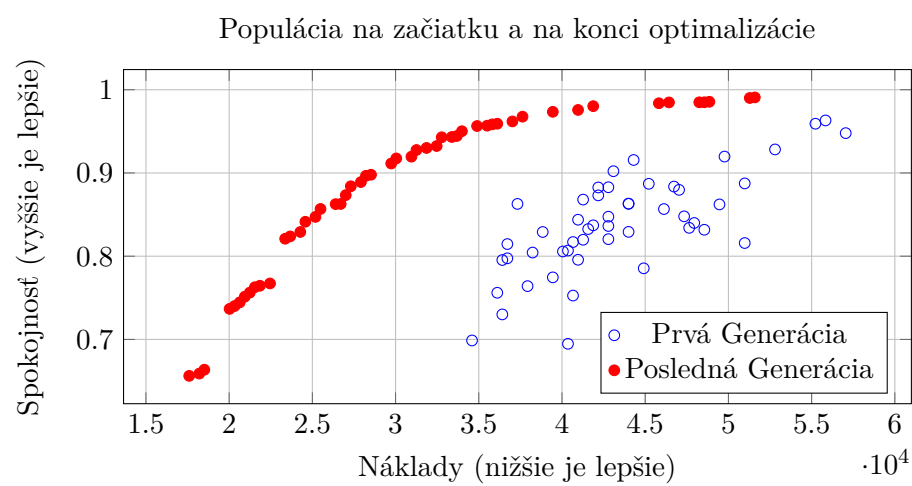
**Input:** Jedinca  $p$   
**for** každý gén  $g$  v  $p$  **do**  
    **if**  $\text{random}(0, 1) < \text{pravdepodobnosť mutácie}$  **then**  
         $p[g] \leftarrow \text{náhodný gén};$

---

## 4.4 Výstupy optimalizácie

Výstupom klasického genetického algoritmu by mal byť jedinec s najlepším ohodnotením. V prípade NSGA-II je to front s najlepšimi jedincami. Keďže všetky jedince v najlepšom fronte sú rovnako dobré, nedá sa jednoznačne určiť, ktorý z nich je najlepší. Ich ohodnotenie je závislé od viacerých cieľov, v prípade tejto práce dvoch. Po skončení optimalizácie sa celá posledná generácia približuje optimálnemu frontu riešení. To znamená, že všetky časové rozpisy v poslednej generácii sú rovnako dobrým výsledkom. Pre výber toho najlepšieho jedinca je potrebné rozhodnúť sa, ktorý z cieľov je dôležitejší, prípade ich váhy. Táto práca sa už musí nechať na používateľa programu, ktorý sa sám rozhodne, ktorý z rozpisov je pre neho najlepší. Z výsledkov priebežných experimentov počas vývoja programu sa ukázalo, že rozpisy s vyššou spokojnosťou cestujúcich majú z pravidla aj vyššie celkové náklady. Ako postupne klesajú náklady, klesá aj spokojnosť cestujúcich, pretože sa znižuje počet spojov, to má za následok preplnenejšie vozidlá a dlhšie čakacie doby na zastávkach. Tieto experimenty sú bližšie popísané v kapitole 5.

Na obrázku 4.2 je vidieť priaznivé výsledky optimalizácie. Algoritmus NSGA-II sa po 100 generáciach priblížil k optimálnemu frontu riešení. Oproti prvej generácii sa značne znížili náklady a zvýšila spokojnosť cestujúcich. Tento výsledok ukazuje, že genetický algoritmus dokáže efektívne nájsť kompromis medzi protichodnými cieľmi, ako sú náklady a spokojnosť. Použitie NSGA-II umožnilo zachovať diverzitu riešení, čo poskytuje používateľovi možnosť výberu najvhodnejšieho časového rozpisu podľa jeho preferencií.



Obr. 4.2: Populácia na začiatku a na konci optimalizácie

## Kapitola 5

# Experimenty

V tejto kapitole sú podrobne popísané experimentálne scenáre, ktoré boli navrhnuté na posúdenie výkonnosti optimalizačného algoritmu aplikovaného na plánovanie liniek MHD. Cieľom bolo preskúmať, ako algoritmus reaguje na zmeny v parametroch ako sú dĺžka trasy, dĺžka vozidla a počet zastávok.

Každý experiment bol vykonaný s rovnakými parametrami pre genetický algoritmus, aby sa zabezpečila konzistentnosť výsledkov:

- **Počet jedincov v populácii** — 50 pre krátku trasu, 10 pre dlhú trasu <sup>1</sup>
- **Počet generácií** — 100
- **Pravdepodobnosť mutácie** — 5%
- **Maximálny počet spojov za hodinu** — 15

Ostatné parametre sú špecifické pre jednotlivé experimenty a sú uvedené v tabuľkách v príslušných sekciách.

Parametre zariadenia na ktorom boli experimenty vykonané sú:

- **Operačný systém** — Windows 11 Pro 64-bit
- **Procesor** — Intel Core i7-9750H CPU @ 2.60GHz
- **RAM** — 16 GB DDR4

Všetky experimenty boli vykonané na rovnakom zariadení, aby sa zabezpečila konzistentnosť výsledkov, a v rovnakom čase, aby sa minimalizoval vplyv vonkajších faktorov. Každý experiment trval približne od 45 minút do 1.5 hodiny v závislosti od dĺžky trasy a počtu zastávok.

---

<sup>1</sup>Tento počet bol znížený z dôvodu dlhšej doby výpočtu.

## 5.1 Experiment s krátkou trasou a krátkým vozidlem

Cílem tohoto experimentu je zjistit, ako sa algoritmus vyrovnáva s krátkou trasou a krátkým vozidlom. Ako referencia bola zvolená linka 46 v Brne, ktorej trasa pozostáva z 11 zastávok. Parametry experimentu sú uvedené v tabuľke 5.1.

Vstup	Hodnota
Kapacita vozidla	80
Miest na sedenie	30
Celkové náklady	92,82 Kč/100 miesto-km
Dĺžka trasy	3,8 km

Tabuľka 5.1: Parametre vozidla a trasy

Ako je vidieť v tabuľke 5.2, optimalizácia prebehla úspešne. Algoritmus bol schopný vygenerovať rozpis, pri ktorom je spokojnosť cestujúcich 94% a počet cestujúcich. Vozidlá sú v priemere zaplnené do jednej tretiny, pretože ich je až 115, to je o 3 spoje menej ako je aktuálny rozpis linky 46.

Výstup	Hodnota
Počet príchodov cestujúcich	11603
Počet prevezených cestujúcich	11603
Počet neobslužených cestujúcich	0 (0,0 %)
Celkový čas strávený čakaním	54625 min
Priemerný čas strávený čakaním	5 min
Celkové náklady	32450 Kč
Priemerná spokojnosť cestujúcich	94,0 %
Celkový počet vozidiel	115
Priemerná naplnenosť vozidiel	25 (31,02 %)

Tabuľka 5.2: Výstupy simulácie

Celý rozpis zastávok, časový rozpis a grafy podrobnejšie popisujúce tento experiment sú uvedené v prílohe A.

## 5.2 Experiment s krátkou trasou a dlhým vozidlom

Cieľom tohto experimentu je zistiť, ako sa algoritmus vyrovnáva s krátkou trasou a dlhým vozidlom. Ako referencia bola zvolená linka 46 v Brne, ktorej trasa pozostáva z 11 zastávok. Parametry experimentu sú uvedené v tabuľke 5.3.

Vstup	Hodnota
Kapacita vozidla	160
Počet miest na sedenie	60
Celkové náklady	92,82 Kč/100 miesto-km
Dĺžka trasy	3,8 km

Tabuľka 5.3: Parametre vozidla a trasy

V tabuľke 5.4 sú vidieť zmeny oproti predošlému experimentu. Navýšenie kapacity jedného vozidla má vplyv na:

- Priemerná čakacia doba sa zvýšila o 2 minúty
- Spokojnosť cestujúcich sa zvýšila o 4,54%
- Počet vozidiel sa znížil o 43 kusov
- Priemerná naplnenosť vozidiel sa znížila o 6,42%

Výstup	Hodnota
Počet príchodov cestujúcich	11528
Počet prevezených cestujúcich	11528
Počet neobslužovaných cestujúcich	0 (0,0 %)
Celkový čas strávený čakaním	86013 min
Priemerný čas strávený čakaním	7 min
Celkové náklady	40633 Kč
Priemerná spokojnosť cestujúcich	98,54 %
Celkový počet vozidiel	72
Priemerná naplnenosť vozidiel	39 (24,6 %)

Tabuľka 5.4: Výstupy simulácie

Celý rozpis zastávok, časový rozpis a grafy podrobnejšie popisujúce tento experiment sú uvedené v prílohe B.



### 5.3 Experiment s dlhou trasou a krátkym vozidlom

Cieľom tohto experimentu je zistiť, ako sa algoritmus vyrovnáva s dlhou trasou a krátkym vozidlom. Ako referencia bola zvolená okružná linka 84 v Brne, ktorej trasa pozostáva zo 44 zastávok. Parametry experimentu sú uvedené v tabuľke 5.5.

Vstup	Hodnota
Kapacita vozidla	80
Počet miest na sedenie	30
Celkové náklady	92,82 Kč/100 miesto-km
Dĺžka trasy	23,0 km

Tabuľka 5.5: Parametre vozidla a trasy

Z tabuľky 5.6 je vidieť, že algoritmus bol schopný vygenerovať rozpis, pri ktorom je spokojnosť cestujúcich viac ako 98%. Priemerná čakacia doba, počet vozidiel a priemerná naplnenosť vozidiel sú veľmi podobné ako v prípade krátkej trasy. Oproti nej však vzrástol počet cestujúcich a teda aj celkový čas strávený čakaním. Taktiež sa zvýšili celkové náklady na prevádzku.

Výstup	Hodnota
Počet príchodov cestujúcich	49041
Počet prevezených cestujúcich	49041
Počet neobslužených cestujúcich	0 (0,0 %)
Celkový čas strávený čakaním	253052 min
Priemerný čas strávený čakaním	5 min
Celkové náklady	189576 Kč
Priemerná spokojnosť cestujúcich	98,1 %
Celkový počet vozidiel	111
Priemerná naplnenosť vozidiel	22 (27,67 %)

Tabuľka 5.6: Výstupy simulácie

Celý rozpis zastávok, časový rozpis a grafy podrobnejšie popisujúce tento experiment sú uvedené v prílohe C.

## 5.4 Experiment s dlhou trasou a dlhým vozidlom

Cieľom tohto experimentu je zistiť, ako sa algoritmus vyrovnáva s dlhou trasou a dlhým vozidlom. Ako referencia bola zvolená okružná linka 84 v Brne, ktorej trasa pozostáva zo 44 zastávok. Parametry experimentu sú uvedené v tabuľke 5.7.

Vstup	Hodnota
Kapacita vozidla	160
Počet miest na sedenie	60
Celkové náklady	92,82 Kč/100 miesto-km
Dĺžka trasy	23,0 km

Tabuľka 5.7: Parametre vozidla a trasy

Výstupy experimentu je možné vidieť v tabuľke 5.8. Oproti experimentu s dlhou cestou ale krátkym vozidlom sa zmenili nasledovné parametre:

- Priemerná čakacia doba sa zvýšila o 2 minúty
- Spokojnosť cestujúcich sa zvýšila o 1,88%
- Počet vozidiel sa znížil o 29 kusov
- Priemerná naplnenosť vozidiel sa znížila o 9,04%

Výstup	Hodnota
Počet príchodov cestujúcich	48358
Počet prevezených cestujúcich	48358
Počet neobslužených cestujúcich	0 (0,0 %)
Celkový čas strávený čakaním	314379 min
Priemerný čas strávený čakaním	7 min
Celkové náklady	280094 Kč
Priemerná spokojnosť cestujúcich	99,98 %
Celkový počet vozidiel	82
Priemerná naplnenosť vozidiel	30 (18,63 %)

Tabuľka 5.8: Výstupy simulácie

Celý rozpis zastávok, časový rozpis a grafy podrobnejšie popisujúce tento experiment sú uvedené v prílohe D.

## Kapitola 6

# Nástroj na analýzu a optimalizáciu časových rozpisov

Poslednou časťou tejto práce je implementácia nástroja na analýzu a optimalizáciu časových rozpisov. Tento nástroj bude slúžiť ako rozhranie medzi používateľom a optimalizačným algoritmom. Nástroj bude mať dve hlavné funkcie:

- **Analýza časového rozpisu** — nástroj prevezme existujúci časový rozpis a vykoná analýzu jeho kvality. Na to poslúži spustenie simulácie, zber štatistík a ich vizualizácia.
- **Optimalizácia časového rozpisu** — nástroj vygeneruje nové časové rozpisy na základe vstupných parametrov a požiadaviek používateľa. Na to poslúži optimalizačný algoritmus NSGA-II.

Užívateľské rozhranie by malo byť intuitívne a jednoduché. Hlavným cieľom tejto práce je optimalizácia, preto je kladený väčší dôraz na ňu ako na samotné užívateľské rozhranie. Toto rozhranie, bez ohľadu na to ako moc jednoduché bude, je dôležité, preto tento krok nesmie byť zanedbaný.

### 6.1 Návrh užívateľského rozhrania

Prvým krokom pri tvorbe užívateľských rozhraní je návrh. Najprv si treba premyslieť, čo všetko bude nástroj robiť a aké funkcie bude mať. Tento nástroj by mal vedieť analyzovať nahrané časové rozpisy, vizualizovať zistenia zo simulácie, prezentovať ich formou grafov a štatistík. Ďalej by mal vedieť optimalizovať časové rozpisy. Výpočet optimalizácie pomocou genetického algoritmu je časovo náročný úkon a preto by bolo vhodné používateľa informovať o priebehu optimalizácie. Po dokončení optimalizácie by mal mať používateľ možnosť vybrať si rozpis, ktorý sa mu páči najviac a uložiť si ho. Malo by byť možné analyzovať aj uložené rozpisy. Prípadne by nástroj mohol ponúkať exportovanie analýzy rozpisu.

Všetky tieto funkcie sa dajú rozdeliť medzi dve stránky, stránku pre analýzu a stránku pre optimalizáciu. Na prepínanie medzi stránkami poslúži jednoduché menu v hornej časti okna. Na ich návrh boli použité wireframy.

## Stránka pre analýzu časového rozpisu

Na stránke analýzy časového rozpisu bude možné:

- nahrať informácie o zastávkach
- nahrať informácie o časovom rozpise
- nahrať uložený časový rozpis
- zadať parametre pre simuláciu
- Resetovať stránku
- spustiť simuláciu
- zobrazíť výsledky simulácie, vrátane grafov a štatistík
- uložiť výsledky analýzy

The wireframe shows a web interface for simulation analysis. At the top, there is a header with the text 'Názov aplikácie' on the left and two buttons, 'Analýza' and 'Optimizácia', on the right. Below the header, the main content area is divided into several sections. On the left, there is a large dashed box labeled 'Nahratie súboru s informáciami o zastávkach'. To its right, there is another dashed box labeled 'Nahratie súboru s informáciami o časovom rozpise'. Below these, there is a button labeled 'Vybrať uložený rozpis'. Further down, there are four input fields: 'Kapacita vozidla', 'Miest na sedenie', 'Náklady na 100 miesto-km', and 'Dĺžka trasy'. Below these fields are two buttons: 'Resetovať' and 'Analyzovať'. At the bottom, there is a large dashed box labeled 'Výsledky analýzy' with the subtitle 'Jedno číselné výstupy' and 'grafy'. Below this box is a button labeled 'Uložiť analýzu'.

Obr. 6.1: Wireframe pre analýzu časového rozpisu

Na obrázku 6.1 je znázornený wireframe pre túto stránku. Funkcie spomenuté vyššie sú usporiadané rovnako ako jednotlivé elementy stránky. Pre nahranie informácií o zastávke a novom časovom rozpise bude slúžiť vyberač súborov. Informácie by mali byť uložené v textovom súbore. Užívateľské rozhranie by malo obsahovať nápovedu, ktorá používateľovi ukáže aký formát majú mať vstupné súbory.

Formát potrebný na spracovanie informácií o zastávkach:

Lesná, Haškova:0:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0

Brechtova:1:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0.1

Blažkova:2:[(5, 60), (6, 90), ..., (21, 40), (22, 30)]:0.1

...

Formát potrebný na spracovanie informácií o časovom rozpise:

04:

05:01,11,21,31,41,51

06:01,08,14,21,28,34,41,48,54

07:01,08,14,21,28,34,41,48,54

08:01,08,18,28,38,48,58

09:08,18,28,38,48,58

...

Výber uloženého rozpisu môže byť realizovaný formou rozbaľovacej ponuky. Rozpisy v tejto ponuke by mali byť pomenované, aby bol výber prehľadný. Vstupné parametre **Kapacita vozidla**, **Miest na sedenie**, **Náklady na 100 miesto-km** a **Dĺžka trasy** budú realizované jednoduchými textovými vstupmi, s obmedzením len na numerické znaky, prípadne číselnými obmedzeniami potrebnými pre dané parametre. Tlačítko **Resetovať** vymaže všetky vstupy aj výstupy analýzy. Tlačítko **Analyzovať** spustí simuláciu časového rozpisu a zobrazí výsledky analýzy. Medzi tieto výsledky budú patriť agregované hodnoty:

- Počet príchodov cestujúcich
- Počet prevezených cestujúcich
- Počet neobslužených cestujúcich
- Celkový čas strávený čakaním
- Priemerný čas strávený čakaním
- Celkové náklady
- Priemerná spokojnosť cestujúcich
- Celkový počet vozidiel
- Priemerná naplnenosť vozidiel

Taktiež budú na stránke zobrazené grafy s hodnotami rôznych štatistík za hodinu aby boli vidieť zmeny v priebehu dňa, konkrétne:

- Cestujúci prichádzajúci za hodinu
- Priemerný čas strávený čakaním za hodinu
- Počet prípadov kedy sa cestujúci nezmestili do vozidla za hodinu

Posledným grafom bude:

- Priemerná naplnenosť vozidiel naprieč zastávkami

Po stlačení tlačítka **Uložiť analýzu** sa otvorí vyberač súborov, aby si používateľ mohol vybrať kam sa analýza uloží. Analýza sa bude ukladať vo forme textového súboru.

## Stránka pre optimalizáciu časového rozpisu

Na stránke optimalizácie časového rozpisu bude možné:

- nahrať informácie o zastávkach
- zadať parametre pre optimalizáciu
- Resetovať stránku
- spustiť optimalizáciu
- zobrazíť priebežne dosiahnuté ciele aktuálne spracovanej generácie
- vybrať rozpis z generácie
- uložiť vybraný rozpis

The wireframe shows a web interface for scheduling optimization. At the top, there's a header with 'Názov aplikácie' on the left and two buttons, 'Analýza' and 'Optimalizácia', on the right. Below the header is a large dashed box containing the text 'Nahratie súboru s informáciami o zastávkach'. Underneath this box are eight input fields arranged in two rows of four: 'Veľkosť populácie', 'Počet generácií', 'Pravdepodobnosť mutácie', 'Maximálny počet spojov za hodinu', 'Kapacita vozidla', 'Miest na sedenie', 'Náklady na 100 miesto-km', and 'Dĺžka trasy'. Below these fields are two buttons: 'Resetovať' and 'Optimalizovať'. At the bottom of the main content area is another large dashed box containing the text 'Priebežné výsledky optimalizácie' and 'Graf s dosiahnutými cieľmi všetkých jednotníc v aktuálne dosiahnutej generácii'. Below this box is a button labeled 'Uložiť rozpis'.

Obr. 6.2: Wireframe pre optimalizáciu časového rozpisu

Na obrázku 6.2 je znázornený wireframe pre túto stránku. Pre nahranie informácií o zastávke bude slúžiť rovnaký vyberač súborov ako na stránke analýzy. Vyberač pre časový rozpis tu nie je potrebný, pretože z hľadiska optimalizácie je časový rozpis výstupom a nie vstupom. Číselné vstupné parametre pre simuláciu sú rovnaké ako na stránke analýzy, ale pre genetický algoritmus tu sú štyri nové a to **Veľkosť populácie**, **Počet generácií**, **Pravdepodobnosť mutácie** a **Maximálny počet spojov za hodinu**. Tlačítko **Resetovať** vymaže všetky vstupy aj výstupy optimalizácie. Tlačítko **Optimalizovať** spustí optimalizáciu časového rozpisu. Používateľ by mal počas optimalizácie byť schopný vidieť:

- V koľkej generácii sa algoritmus nachádza
- Koľko generácií bude algoritmus spracovávať

- Kedy optimalizácia začala
- Kedy optimalizácia skončila (ak už skončila)
- Ako dlho optimalizácia trvala (ak už skončila)
- Graf s priebežnými dosiahnutými cieľmi jednotlivých jedincov v aktuálne dosiahnutej generácii

Po dokončení optimalizácie by mal byť používateľ schopný prezrieť si všetky rozpisy v poslednej generácii. To bude zabezpečené posuvným tiahlom, a zobrazením aktuálne vybraného rozpisu. Tiahlo bude mať toľko krokov, koľko je rozpisov v generácii. Vyplnením textového vstupu pre názov rozpisu a stlačením tlačítka **Uložiť rozpis** sa rozpis uloží. Nepomenovaný rozpis sa označí časom a dátumom vytvorenia.

## 6.2 Implementácia nástroja

Po návrhu následuje implementácia. Predtým však treba vybrať technológie, ktoré sú na to vhodné a navrhnuť architektúru aplikácie.

### Použité technológie

Ako programovací jazyk bol zvolený **Python** a to preto, lebo je veľmi verzatilný. Výhodnejšie by možno bolo naprogramovať optimalizačný algoritmus v jazyku **C** alebo **C++**, ktoré sú rýchlejšie.

**Python** bol však vybraný vďaka jednoduchému použitiu a tomu, že sa v ňom dá pohodlne napísať ako backend, tak aj frontend. Na to aby sa mohol v tomto jazyku písať aj frontend bol využitý framework **Reflex**.

Vďaka tomtu frameworku je možné písať webové komponenty a celé stránky priamo v jazyku **Python**. Každý komponent je písaný formou funkcie, ktorá vracia jeho obsah. Tým sa dajú komponenty zhlukovať do väčších komponentov alebo stránok. Vstupy komponentov sa dajú predávať ako parametre jeho funkcie, alebo môže komponent pracovať s premennými, ktoré sú súčasťou triedy **State**. **State** je trieda, ktorá počas behu aplikácie uchováva jej stav. K premenným tohto stavu sa dá jednoducho pristupovať, čítať ich hodnoty aj prepisovať ich. Takýchto tried môže byť v aplikácii viacej a jeden komponent môže pracovať s premennými aj viacerých stavov. Stav okrem premenných obsahuje aj metódy, ktoré sa volajú pri danej udalosti. Medzi udalosti patrí napríklad stlačenie konkrétneho tlačítka, alebo zmena hodnoty v jednom z textových vstupov. V týchto metódach sa môže pristupovať aj k premenným iného stavu.

Informácie o použití tohoto frameworku boli čerpané z jeho oficiálnej dokumentácie [11].

Takáto verzatilita jazyka a frameworku bola pri tejto práci veľmi výtaná. Treba si však dávať pozor, pretože priveľká sloboda v písaní kódu môže mať za následok neprehľadný, príliš previazaný kód, ktorý sa bude v budúcnosti ťažko rozširovať. Pretože sa s týmito technológiami dá aplikácia napísať naozaj veľmi veľa spôsobmi, je potrebné zvoliť si dobrú architektúru, rozčlenenie aplikácie na moduly, a tie na jednotlivé webové komponenty alebo backendové skripty.

## Architektúra aplikácie

Aplikácia sa volá SPROUT, čo znamená *Smart Performance and Resource Optimization for Urban Transport*, po slovensky inteligentná optimalizácia výkonu a zdrojov verejnej dopravy. Touto skratkou je označený aj hlavný adresár, ktorý obsahuje všetky zdrojové kódy aplikácie. V ňom sa nachádzajú adresáre **backend**, v ktorom sú skripty obsahujúce logiku, simulačný model, kalendár udalostí, genetický algoritmus a podobné, **components**, ktorý obsahuje webové komponenty a **pages**, kde sú uložené zdrojové kódy jednotlivých stránok aplikácie. Ďalej sa tu nachádza **sprout.py**, súbor, ktorý obsahuje koreňový element aplikácie a smerovanie medzi jednotlivými stránkami. Posledným súborom v tomto adresári je **\_\_init\_\_.py**, ktorý slúži na inicializáciu adresára ako **Python** modulu. Tento súbor sa nachádza aj v ostatných adresároch.

```
sprout
├── backend
│   ├── __init__.py
│   ├── EventCalendar.py
│   ├── Genetics.py
│   ├── InputParser.py
│   ├── models.py
│   ├── RandomNumberGenerator.py
│   ├── Simulation.py
│   └── Statistics.py
├── components
│   ├── __init__.py
│   ├── analyzeLine.py
│   ├── busStopChart.py
│   ├── busStopTable.py
│   ├── constraintInput.py
│   ├── footer.py
│   ├── hourChart.py
│   ├── infoCard.py
│   ├── infoUpload.py
│   ├── layout.py
│   ├── navbar.py
│   ├── numberInput.py
│   ├── optimizeLine.py
│   ├── rozpis.txt
│   ├── timeTable.py
│   └── zastavky.txt
├── pages
│   ├── __init__.py
│   ├── analyzePage.py
│   ├── homePage.py
│   └── optimizePage.py
├── __init__.py
└── sprout.py
```



Súbor	Obsah
Backendové skripty EventCalendar.py Genetics.py InputParser.py models.py RandomNumberGenerator.py Simulation.py Statistics.py	trieda kalendára udalostí a udalsti trida genetického algoritmu a jedného jedinca trieda s metódami na spracovanie vstupu model zastávky, vozidla a časového rozpisu trieda generátora náhodných čísel trieda na riadenie simulácie trieda na zber a spracovanie štatistík
Webové komponenty analyzeLine.py busStopChart.py busStopTable.py constraintInput.py footer.py hourChart.py infoCard.py infoUpload.py layout.py navbar.py numberInput.py optimizeLine.py rozpis.txt timeTable.py zastavky.txt	analýza časového rozpisu zobrazenie grafu so zastávkami na jednej z osí zobrazenie tabuľky so zastávkami zadanie obmedzení počtu spojov za hodinu pätička webovej aplikácie zobrazenie grafu s hodinami na jednej s osí zobrazenie jednočíselných štatistík nahranie súboru základné rozloženie stránky horné navigačné menu zadanie jednočíselných vstupov optimalizácia časového rozpisu vzorový príklad informácií o časovom rozpise zobrazenie časového rozpisu vzorový príklad informácií o zastávkach
Stránky aplikácie analyzePage.py homePage.py optimizePage.py	stránka pre analýzu časového rozpisu úvodná stránka stránka pre optimalizáciu časového rozpisu
sprout.py	koreňový komponent a smerovanie

Tabuľka 6.1: Zdrojové kódy

V tabuľke 6.1 je stručne popísaný obsah jednotlivých súborov zdrojových kódov a ich rozdelenie medzi jednotlivé moduly. Podrobnejší popis celého zdrojového kódu je dostupný v programovej dokumentácii [8], ktorá bola vygenerovaná pomocou nástroja **Sphinx** [13].

## Kapitola 7

# Záver

Cieľom tejto práce bolo vyvynúť nástroj, ktorý by v určitom ohlade optimalizoval mestskú hromadnú dopravu. Ako cieľ optimalizácie bol vybraný časový rozpis. Optimalizácia bola realizovaná pomocou multi-objektívneho genetického algoritmu NSGA-II.

Táto práca mala splniť nasledujúce body:

1. Preskúmajte problematiku vytvárania sieteestskej hromadnej dopravy, jej modelovania a optimalizácie pre mestá do pol milióna obyvateľov. Na návrh môjho garanta bol simulačný model navrhnutý pomocou DEVS, poznatky o tomto formalizme boli čerpané zo zborníku z konferencie WSC [16] a príklad, ktorý pomohol k porozumeniu je z článku [12].
2. Pre mesto veľkosti krajského mesta Českej republiky získajte alebo odhadnite dáta, z ktorých zostavíte model miestnejestskej hromadnej dopravy. Informácie boli získané priamo z Dopravného podniku mesta Brno, viac informácií je v kapitole 2.
3. Oboznámte sa s riešeniami, ktoré na optimalizáciu takejto siete používajú metódy umelej inteligencie. Inšpirácia na použitie genetického algoritmu vychádza z článku [15], konkrétne ide o multi-objektívny NSGA-II.
4. Vytvorte prostredie, ktoré bude slúžiť na modelovanie a optimalizáciu takejto siete. Bol vytvorený nástroj na analýzu a optimalizáciu časových rozpisov, ktorý je popísaný v kapitole 6, na zostrojenie modelu slúžia používateľské vstupy, ktoré nástroj spracuje a sám zostrojí model.
5. Na vhodne zvolených príkladoch overte fungovanie vášho systému a diskutujte dosiahnuté výsledky. Fungovanie systému bolo otestované v rámci štyroch experimentov a priebežného testovania počas vývoja. Experimenty sú v kapitole 5. Výsledky preukázali, že algoritmus je schopný sa prispôbiť rôznym situáciám.

Aplikáciu by bolo možné rozšíriť napríklad o porovnanie dvoch rozpisov, alebo exportovanie analýzy vo formáte PDF, alebo export optimalizácie. Taktiež by bolo dobré zadať verzatilnejšie zadávanie vstupov, pretože momentálne sa informácie o zastávkach dajú nahrať len ako textový súbor. Pri analýze a optimalizácii sa využíva simulačný model, ktorý by sa mohol spraviť odrobnejší, aby sa viac približoval realite a dodal tak kvalitnejšie výsledky. V rámci optimalizácie by sa mohli niektoré výpočty sparalelizovať aby sa tak skrátila doba čakania na výsledok. Dalo by sa naimplementovať inteligentné ohodnocovanie rozpisov vo finálnej generácii na základe nových používateľských vstupov, aby používateľ nemusel tento krok robiť ručne.



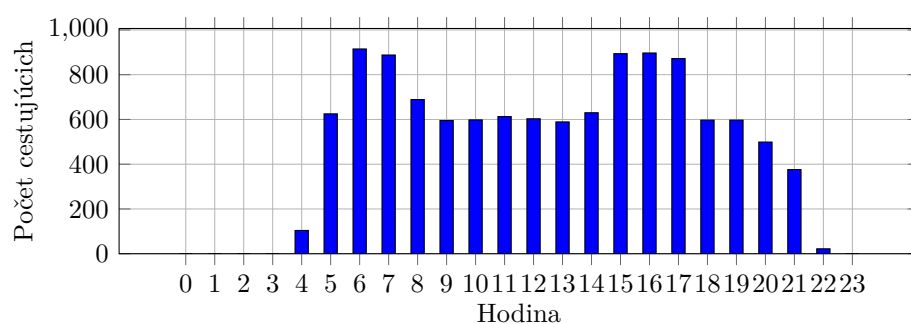
# Literatúra

- [1] BALA, A. a SHARMA, A. K. A comparative study of modified crossover operators. In: *2015 Third International Conference on Image Information Processing (ICIIP)*. S.l.: IEEE, 2015, s. 281–284.
- [2] DEB, K.; PRATAP, A.; AGARWAL, S. a MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 2002, zv. 6, č. 2, s. 182–197.
- [3] DOPRAVNÍ PODNIK MĚSTA BRNA. *Náš vozový park – Autobusy*. 2025. Dostupné z: <https://www.dpmb.cz/nas-vozovy-park#autobusy>. Citované dňa 27. 4. 2025.
- [4] FARIAS, R. G. G. a DE MAGALHÃES, C. S. Parent Selection Strategies in Niching Genetic Algorithms. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. S.l.: IEEE, 2018, s. 1–8.
- [5] GERAGHTY, M. A. *7.2: Exponential Distribution*. LibreTexts Statistics, 2022. Dostupné z: [https://stats.libretexts.org/Bookshelves/Introductory\\_Statistics/Inferential\\_Statistics\\_and\\_Probability\\_-\\_A\\_Holistic\\_Approach\\_\(Geraghty\)/07:\\_Continuous\\_Random\\_Variables/7.02:\\_Exponential\\_Distribution](https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Inferential_Statistics_and_Probability_-_A_Holistic_Approach_(Geraghty)/07:_Continuous_Random_Variables/7.02:_Exponential_Distribution). Citované dňa 4. 5. 2025.
- [6] IMMANUEL, S. D. a CHAKRABORTY, U. K. Genetic Algorithm: An Approach on Optimization. In: *2019 International Conference on Communication and Electronics Systems (ICCES)*. S.l.: IEEE, 2019, s. 701–708.
- [7] INTERNATIONAL AIR TRANSPORT ASSOCIATION (IATA). *Demystifying Key Air Traffic Metrics: Understanding RPKs and ASKs*. 2024. Dostupné z: <https://www.iata.org/en/publications/newsletters/iata-knowledge-hub/demystifying-key-air-traffic-metrics-understanding-rpks-and-asks/>. Citované dňa 4. 5. 2025.
- [8] KATONA, L. *Sprout Documentation*. 2025. Dostupné z: <https://sprout.readthedocs.io/en/latest/>. Citované dňa 4. 5. 2025.
- [9] NGATCHOU, P.; ZAREI, A. a EL SHARKAWI, A. Pareto Multi Objective Optimization. In: *Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems*. S.l.: IEEE, 2005, s. 84–91.
- [10] PERINGER, P. *Modelování a simulace IMS: Studijní opora*. Brno: Fakulta informačních technologií, Vysoké učení technické v Brně, 2022. Vydáno dne 2. února 2022.

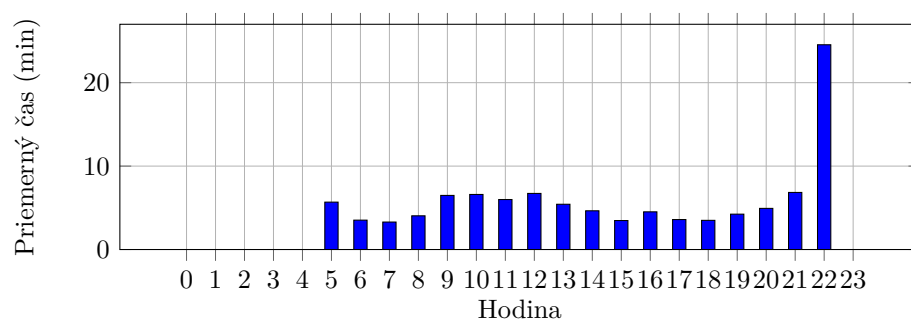
- [11] PYNECONE, INC.. *Reflex Documentation*. 2025. Dostupné z: <https://reflex.dev/docs/>. Citované dňa 27. 4. 2025.
- [12] SEO, K.-M.; CHOI, C.; KIM, T. a KIM, J.-H. DEVS-based combat modeling for engagement-level simulation. *Simulation: Transactions of The Society for Modeling and Simulation International*, 2014, zv. 90, č. 7, s. 759–781.
- [13] SPHINX DEVELOPERS. *Sphinx Documentation*. 2025. Dostupné z: <https://www.sphinx-doc.org/>. Citované dňa 4. 5. 2025.
- [14] STATUTÁRNÍ MĚSTO BRNO a DOPRAVNÍ PODNIK MĚSTA BRNA, A.S.. *Smlouva o veřejných službách v přepravě cestujících — zajištění dopravní obslužnosti a poskytování veřejných služeb v přepravě cestujících ve vymezeném území*. 2024. Dostupné z: <https://smlouvy.gov.cz/smlouva/31154992>. Zveřejnené dňa 26. 11. 2024, citované dňa 27. 4. 2025.
- [15] TANG, J.; YANG, Y.; HAO, W.; LIU, F. a WANG, Y. A Data-Driven Timetable Optimization of Urban Bus Line Based on Multi-Objective Genetic Algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 2021, zv. 22, č. 4, s. 2417–2429.
- [16] TENDELOO, Y. V. a VANGHELUWE, H. Discrete Event System Specification Modeling and Simulation. In: *2018 Winter Simulation Conference (WSC)*. S.l.: IEEE, 2018, s. 162–176.
- [17] ZEIGLER, B. P.; PRAEHOFFER, H. a KIM, T. G. *Theory of Modeling and Simulation*. San Diego: Elsevier Science, 2000. ISBN 9780127784557.

## Príloha A

# Experiment s krátkou trasou a krátkym vozidlom



Obr. A.1: Počet cestujúcich prichádzajúcich na zastávku za hodinu



Obr. A.2: Priemerný čas strávený čakaním za hodinu

<b>Zastávka</b>	<b>#</b>
Lesná, Haškova	0
Brechtova	1
Blažkova	2
Arbesova	3
Heleny Malířové	4
Lesná, nádraží	5
Štefánikova čtvrť	6
Provozníková	7
Lesnická	9
Zemědělská	10
Černá Pole, Erbenova	11

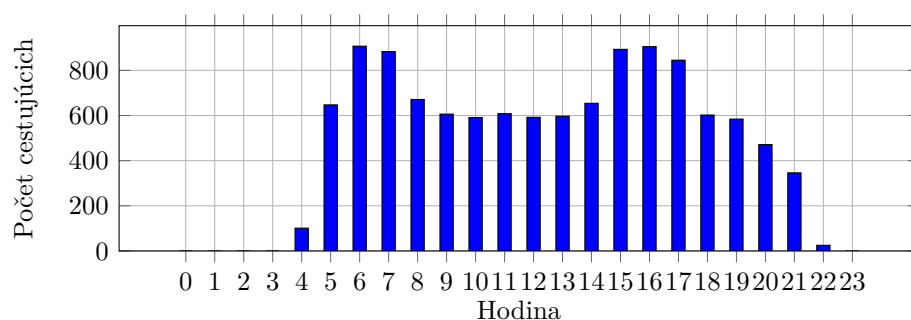
Tabuľka A.1: Rozpis zastávok

<b>h</b>	<b>Odchody</b>
05	00, 10, 20, 30, 40, 50
06	00, 06, 12, 18, 24, 30, 36, 42, 48, 54
07	00, 06, 13, 20, 26, 33, 40, 46, 53
08	00, 08, 17, 25, 34, 42, 51
09	00, 15, 30, 45
10	00, 12, 24, 36, 48
11	00, 12, 24, 36, 48
12	00, 15, 30, 45
13	00, 08, 17, 25, 34, 42, 51
14	00, 10, 20, 30, 40, 50
15	00, 06, 12, 18, 24, 30, 36, 42, 48, 54
16	00, 10, 20, 30, 40, 50
17	00, 06, 12, 18, 24, 30, 36, 42, 48, 54
18	00, 07, 15, 22, 30, 37, 45, 52
19	00, 08, 17, 25, 34, 42, 51
20	00, 10, 20, 30, 40, 50
21	00, 15, 30, 45
22	00

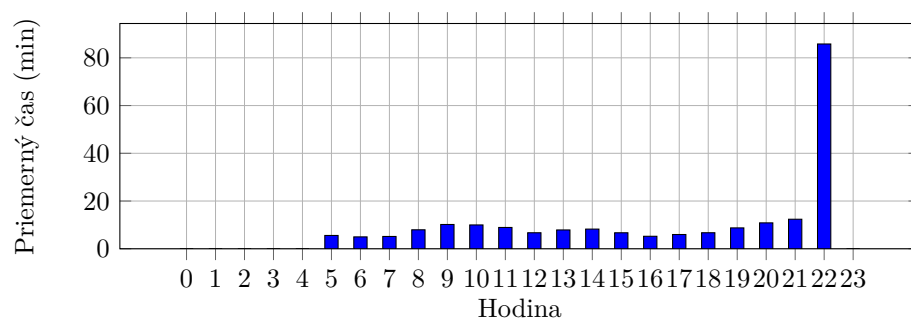
Tabuľka A.2: Časový rozpis

## Príloha B

# Experiment s krátkou trasou a dlhým vozidlom



Obr. B.1: Počet cestujúcich prichádzajúcich na zastávku za hodinu



Obr. B.2: Priemerný čas strávený čakaním za hodinu



<b>Zastávka</b>	<b>#</b>
Lesná, Haškova	0
Brechtova	1
Blažkova	2
Arbesova	3
Heleny Malířové	4
Lesná, nádraží	5
Štefánikova čtvrť	6
Provozníková	7
Lesnická	9
Zemědělská	10
Černá Pole, Erbenova	11

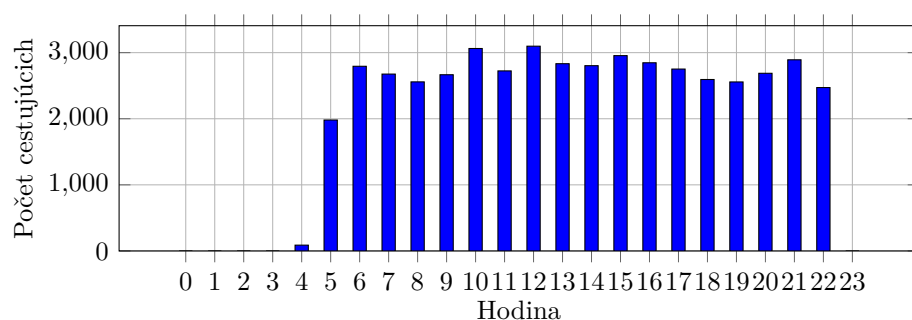
Tabuľka B.1: Rozpis zastávok

<b>h</b>	<b>Odchody</b>
05	00, 10, 20, 30, 40, 50
06	00, 10, 20, 30, 40, 50
07	00, 10, 20, 30, 40, 50
08	00, 20, 40
09	00, 20, 40
10	00, 20, 40
11	00, 15, 30, 45
12	00, 12, 24, 36, 48
13	00, 20, 40
14	00, 15, 30, 45
15	00, 12, 24, 36, 48
16	00, 10, 20, 30, 40, 50
17	00, 12, 24, 36, 48
18	00, 15, 30, 45
19	00, 20, 40
20	00, 20, 40
21	00, 30
22	00

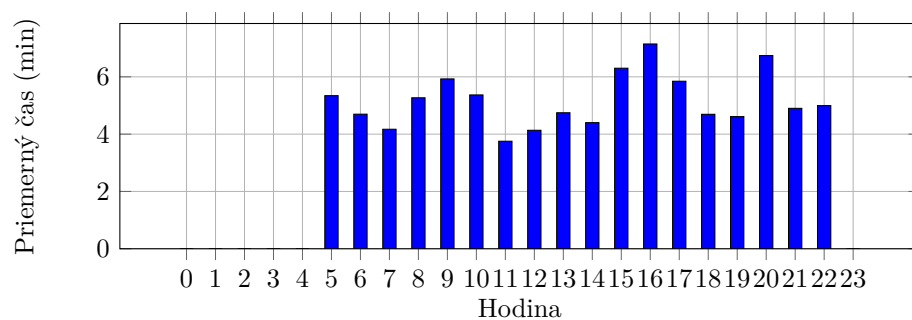
Tabuľka B.2: Časový rozpis

## Príloha C

# Experiment s dlhou trasou a krátkym vozidlom



Obr. C.1: Počet cestujúcich prichádzajúcich na zastávku za hodinu



Obr. C.2: Priemerný čas strávený čakaním za hodinu

<b>Zastávka</b>	<b>#</b>
Židenice, Stará osada	0
Gajdošova	2
Otakara Ševčíka	3
Škroupova	4
Tržní	6
Hladíkova	8
Autobusové nádraží	10
Opuštěná	12
Křídlovická	15
Poříčí	16
Mendlovo náměstí	20
Křížkovského	21
Výstaviště	22
Riviéra	24
Pisárky	26
Anthropos	28
Pod Jurankou	29
Veslařská	30
Jundrov, hřiště	31
Jundrovský most	32
Vozovna Komín	34
Hlavní	35
Štursova	36
Rosického náměstí	37
Přívrat	39
Záhřebská	40
Skácelova	42
Slovanské náměstí	43
Husitská	44
Semilasso	46
Královo Pole, nádraží	47
Mojmírovo náměstí	48
Kociánka	49
Královopolská strojárna	50
Divišova čtvrť	51
U Tunýlku	52
Halasovo náměstí	54
Poliklinika Lesná	55
Lesná, nádraží	56
Štefánikova čtvrť	57
Merhautova	59
Tomkovo náměstí	61
Židenice, kasárna	63
Židenice, Stará osada	65

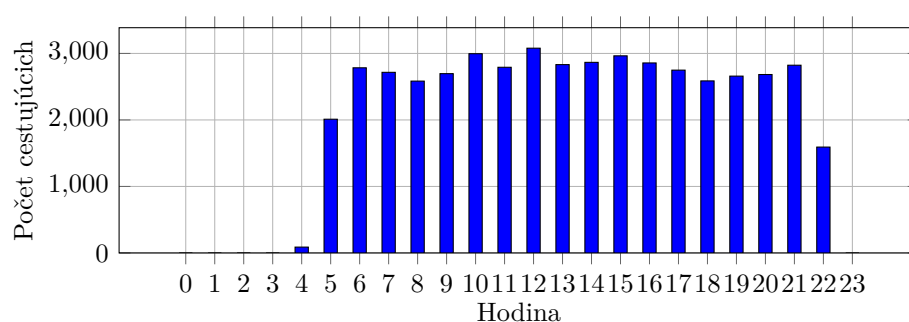
Tabulka C.1: Rozpis zastávek

<b>h</b>	<b>Odchody</b>
05	00, 10, 20, 30, 40, 50
06	00, 07, 15, 22, 30, 37, 45, 52
07	00, 10, 20, 30, 40, 50
08	00, 12, 24, 36, 48
09	00, 12, 24, 36, 48
10	00, 07, 15, 22, 30, 37, 45, 52
11	00, 07, 15, 22, 30, 37, 45, 52
12	00, 10, 20, 30, 40, 50
13	00, 07, 15, 22, 30, 37, 45, 52
14	00, 12, 24, 36, 48
15	00, 15, 30, 45
16	00, 12, 24, 36, 48
17	00, 10, 20, 30, 40, 50
18	00, 07, 15, 22, 30, 37, 45, 52
19	00, 15, 30, 45
20	00, 10, 20, 30, 40, 50
21	00, 08, 17, 25, 34, 42, 51
22	00, 10, 20, 30, 40, 50

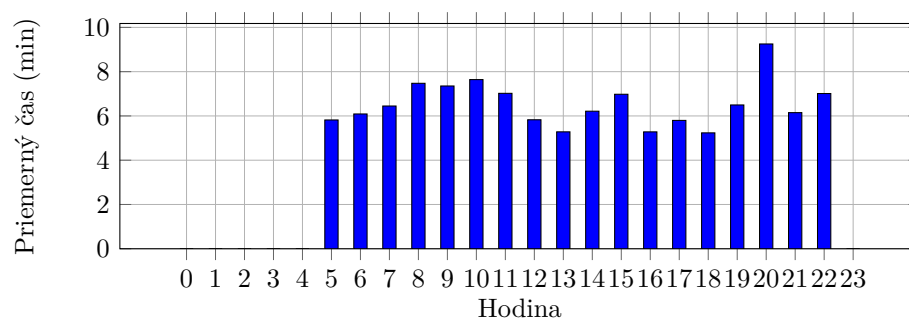
Tabulka C.2: Časový rozpis

## Príloha D

# Experiment s dlhou trasou a dlhým vozidlom



Obr. D.1: Počet cestujúcich prichádzajúcich na zastávku za hodinu



Obr. D.2: Priemerný čas strávený čakaním za hodinu

<b>Zastávka</b>	<b>#</b>
Židenice, Stará osada	0
Gajdošova	2
Otakara Ševčíka	3
Škroupova	4
Tržní	6
Hladíkova	8
Autobusové nádraží	10
Opuštěná	12
Křídlovická	15
Poříčí	16
Mendlovo náměstí	20
Křížkovského	21
Výstaviště	22
Riviéra	24
Pisárky	26
Anthropos	28
Pod Jurankou	29
Veslařská	30
Jundrov, hřiště	31
Jundrovský most	32
Vozovna Komín	34
Hlavní	35
Štursova	36
Rosického náměstí	37
Přívrat	39
Záhřebská	40
Skácelova	42
Slovanské náměstí	43
Husitská	44
Semilasso	46
Královo Pole, nádraží	47
Mojmírovo náměstí	48
Kociánka	49
Královopolská strojárna	50
Divišova čtvrť	51
U Tunýlku	52
Halasovo náměstí	54
Poliklinika Lesná	55
Lesná, nádraží	56
Štefánikova čtvrť	57
Merhautova	59
Tomkovo náměstí	61
Židenice, kasárna	63
Židenice, Stará osada	65

Tabulka D.1: Rozpis zastávek

<b>h</b>	<b>Odchody</b>
05	00, 12, 24, 36, 48
06	00, 12, 24, 36, 48
07	00, 15, 30, 45
08	00, 15, 30, 45
09	00, 15, 30, 45
10	00, 15, 30, 45
11	00, 12, 24, 36, 48
12	00, 10, 20, 30, 40, 50
13	00, 12, 24, 36, 48
14	00, 15, 30, 45
15	00, 10, 20, 30, 40, 50
16	00, 12, 24, 36, 48
17	00, 10, 20, 30, 40, 50
18	00, 12, 24, 36, 48
19	00, 20, 40
20	00, 12, 24, 36, 48
21	00, 12, 24, 36, 48
22	00

Tabulka D.2: Časový rozpis