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 OPTIMALIZÁCE SIETE MESTSKEJ HROMADNEJ DOPRAVY

SYSTEM FOR SUPPORTING THE OPTIMIZATION OF URBAN PUBLIC TRANSPORT NETWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ KATONA**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. FRANTIŠEK ZBOŘIL, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

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

Cieľom tejto práce je optimalizácia mestskej hromadnej dopravy, konkrétne časového rozpisu jednej linky. Zmyslom tohto textu je popis 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 široký rozsah vstupov, ktorými môže analytik obmedziť isté vlastnosti výsledného rozpisu. Vďaka tomuto nástroju by sa mala práca analytikov, ktorí majú nastarosť správu a vytváranie liniek, výrazne zjednodušiť a zvýšiť tak efektivitu meskej 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 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 inputs that an analyst can use to restrict certain properties of the resulting timetable. Thanks to this tool, the work of analysts responsible for managing and creating lines should be significantly simplified and thus increase the efficiency of urban public transport.

Kľúčové slová

optimalizácia, mestská hromadná doprava, algoritmy inšpirované prírodou, genetický algoritmus, NSGA-II, Pareto front, simulácia, DEVS

Keywords

optimization, urban public transport, nature-inspired algorithms, genetic algorithm, NSGA-II, Pareto front, simulation, DEVS

Citácia

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

Systém pre podporu optimalizáce 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 a študijnej opory predmetu IMS a pán Michael Kříž z Dopravného podniku mesta Brno. Uviedol som všetky literárne zdroje, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Lukáš Katona
7. mája 2025

Poďakovanie

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 používatela systému. Nakoniec by som chcel poďakovať môjmu priatelovi 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

Zoznam obrázkov

Kapitola 1

Úvod

Mestskú hromadnú opravu využíva takmer každý človek a preto sa oplatí pracovať na jej efektivite. Zmyslom tejto práce je nazrieť do problematiky vytvárania a správy jednotlivých liniek mestksej hromadnej dopravy, ich analýze a optimalizácie, vytvoriť nástroj, ktorý pomôže analytikom pri vytváraní nových liniek alebo optimalizácii existujúcich. Výsledkom celej práce je teoretické zefektívnenie mestskej hromadnej dopravy.

Mestská hromadná doprava je systém, ktorý je pre ľudí žijúcich v mestách a prímestských oblastiach existenčne dôležitý, preto sa oplatí investovať čas a námahu do jeho optimalizácie. Kedže je problematika mestksej hromadnej dopravy aj mne blízka, z pohľadu cestujúceho sa s ňou stretávam každý deň, rozhodol som sa zamerať na túto tému.

Ľudia od nepamäti nachádzjú inšpiráciu pre takmer všetky svoje vynálezy v prírode a tomuto trendu neunikla ani informatika, ktorá má od prírody zo všetkých vedných oborov asi najďalej. Algoritmy inšpirované prírodou prinášajú riešenia na problémy, ktoré by sme inými spôsobmi vedeli vyriešiť len za nereálne dlhý čas. Táto časť informaiky ma veľmi zaujala a preto som sa rozhodol nájsť ďalšie možné využitie pre jeden z týchto algoritmov, ktoré sa používajú pre komplexné problémy s veľkým množstvom navzájom sa ovplyvňujúcich premenných.

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. Nástroj obsahuje uživateľské rozhranie pre rýchlu manipuláciu so vstupnými dátami a prehľadné zobrazenie výsledkov. Použité technológie boli inšpírované prípadovou štúdiou [8], ktorá pomocou simulácie a evolučných algoritmov dokázala optimalizovať cesty autonómnych vozidiel.

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

Na zostrojenie simulačného modelu je potrebné najprv získať informácie, vlastnosti reálneho systému a vyhodnotiť, ktoré z nich sú pre nás relevantné. Kapitolo 2 popisuje konzultáciu s analytikom Dopravného podniku mesta Brno, ktorý popísal postupy a metódy, ktoré sa používajú pre optimalizáciu liniek v súčastnosti a vytýčil vlastnosti, na ktoré je dôležité sa zamerať pri simulácii a optimalizácii časového rozpisu linky.

Zo získaných informácií bol následne vytvorený simulačný model, ktorého popis je popísaný v kapitole 3. Výstupy simulačného modelu sú základom pre optimalizáciu časového rozpisu linky. Obsahujú sériu grafov, na ktorých používateľ vidí vyťaženosť linky v priebehu dňa a na jednotlivých zastávkach. Získané výsledky boli použité k validácii simulačného modelu a na ohodnotenie jednotlivých rozpisov.

Kapitola 4 popisuje proces optimalizácie časového rozpisu linky mestskej hromadnej dopravy. Jednotlivé rozpisy ohodnotené na základe štatistík získaných zo simulácie budú použité ako jedinci v genetickom algoritme. Genetický aloritmus následne tieto rozpisy kríži medzi sebou a mutuje, aby našiel čo optimálne riešenie.

Kapitola 5 popisuje experimenty optimalizácie. Aplikácia a optimalizačný algoritmus bol testovaný priebežne. Kapitola obsahuje konkrétne experimenty, ktoré sa realizovali po dokončení vývoja, skúmajúce správanie sa optimalizačného algoritmu vzhľadom na zmeny vstupov ako dĺžka trasy a kapacita vozidla.

Samotný algoritmus by nebol v praxi použiteľný bez používateľského rozhrania. Kapitola 6 popisuje návrh a implementácia uživateľského rozhrania analytického nástroja, umožňujúceho používateľovi jednoducho zadať vstupné dáta, spustiť analýzu a optimalizáciu časového rozpisu a vyhodnotiť výsledky. Jednou z najdôležitejších častí tohto rozhrania je zadávanie používateľských obmedzení, ktoré musí výsledný časový rozpis obsahovať.

Záver 7 obsahuje zhrnutie dosiahnutých výsledkov tejto bakalárskej práce a možné budúce rozšírenia aplikácie.

Kapitola 2

Relevantné vlastnosti mestskej hromadnej dopravy

Systém mestkej hromadnej dopravy je veľmi komplexný, obsahuje mnoho častí, na ktoré by bolo vhodné sa zamerať z pohľadu optimalizácie. Prvým rozhodnutím je výber častí mestskej hromadnej dopravy, ktorá je vhodná na optimalizáciu. Pre uskutočnenie správneho rozhodnutia sú potrebné informácie o reálnom systéme, doterajších postupoch, spôsoboch analýzy a optimalizácie.

2.1 Konzultácia s analytikom Dopravného podniku

Pre získanie relevantných informácií mi bolo doporučené obrátiť sa na pána Michaela Kříže, analytika Dopravného podniku mesta Brno (DPMB). Stretnutie s pánom Křížom prebehlo 7. 10. 2024 v priestoroch administratívy DPMB. Na základe získaných informácií som sa rozhodol zamerať na analýzu a optimalizáciu časového rozpisu jednej linky mestskej hromadnej dopravy. Optimalizácia viac ako jednej linky naraz by pravdepodobne prinieslo lepšie výsledky, ale len v čisto teoretickej rovine. V praxi je to z finančných a časových dôvodov neefektívne. Vzájomne sme sa zhodli na tom, že najlepšie bude optimalizovať jednu linku a to iba jedným smerom, nakoľko druhý smer linky sa dá považovať za samostatnú linku, minimálne v rámci nárokov cestujúcej verejnosti.

2.2 Získané relevantné vlastnosti

Vlastnosti, ktoré sú potrebné pre správnu analýzu a optimalizáciu časového rozpisu linky mestskej hromadnej dopravy sú:

- **Počet zastávok a informácie o nich** — potrebné pre zostavenie samotného rozpisu aj simulačného modelu, nie je dôležitý len počet, ale aj informácie o jednotlivých zastávkach.
- **Čas jazdy medzi jednotlivými zastávkami** — je to dôležitou 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 parame-

ter je dôležitý pre optimalizáciu, nakoľko pre dopravný podnik je výhodné mať čo najmenší počet vozidiel, ktoré vykonávajú obsluhu jednotlivých trás, šetrí to palivo, údržbu aj ľudské zdroje.

- **Počet odchodov vozidla z jeho prvej zastávky za hodinu** — myslí sa tým jeden riadok časového rozpisu, mal by byť priamo úmerný priemernému počtu cestujúcich čakajúcich na danú linku v danej hodine.
- **Počet cestujúcich čakajúcich na jednotlivých zastávkach** — časový rozpis by mal v čo najvyššej miere uspokojiť prepravné potreby väčšiny cestujúcich. Na prvotné testovacie účely boli použité hodnoty zodpovedajúce reálnemu počtu odchádzajúcich vozidiel za hodinu a ich kapacite dlhodobo existujúcej linky DPMB, ktorá je do veľej miery praxou optimalizovaná.
- **Doba čakania na zastávke** — užitočný parameter pre analytika, ktorý slúži na overenie kvality časového rozpisu, ak je doba čakania na zastávke dlhá, je to známka toho, že rozpis nie je optimálny a je potrebné ho upraviť.
- **Počet cestujúcich vo vozidie** — dôležitý parameter pre optimalizáciu. Vysoká preplnenosť vozidla má za dôsledok nekomfortné cestovanie, na druhej strane, príliš prázdne vozidlo je neefektívne pre dopravný podnik.
- **Kapacita vozidla** — každé vozidlo má presne definovanú maximálnu kapacitu, zároveň sa od naplnenosti vozidla odvíja komfort cestujúcich. Informácie o kapacite konkrétnych vozidiel sú získané z webovej stránky dopravného podniku [3].
- **Miest na sedenie** — to či cestujúci sedí alebo bude musieť počas jazdy stáť ovplyvňuje jeho spokojnosť, ktorá výrazne ovplivňuje ohodnotenie rozpisu z pohľadu optimalizácie.
- **Počet cestujúcich, ktorí nenásúpia kvôli preplnenosti vozidla** — v extrémnych prípadoch može nastať situácia, že bude vozidlo úplne plné a nebude možné aby do neho pristúpilo viac cestujúcich. Rozpisy, pri ktorých táto situácia vznikne budú veľmi negatívne ohodnocované z pohľadu optimalizácie aby sa táto nežiadúca situácia čo najviac eliminovala, viac v kapitole o optimalizácii 4.
- **Počet vystupujúcich cestujúcich na jednotlivých zastávkach** — tento parameter je potrebný pre simuláciu, keďže do vozidla s presne definovanou kapacitou cestujúci priebežne nastupujú aj vystupujú. Každá zastávka má pridelený koeficient dôležitosti, ktorý určuje aký podiel z prepravovaných cestujúcich vystúpi na danej zastávke. Prvá zastávka má tento koeficient 0 (cestujúci iba nastupujú), na konečnej zastávke je koeficient 1 (cestujúci iba vystupujú). Na ostatných zastávkach sa koeficient mení podľa toho, aký podiel cestujúcich na danej zastávke priemerne vystupuje pričom prestupné zastávky majú vyšší koeficient ako neprestupné.
- **Náklady na prevádzku** — náklady na prevádzku sú hlavným parametrom optimalizácie časových rozpisov jednotlivých liniek vyžadovaným DPMB. Náklady jednotlivých typov vozidiel sú súčasťou finančného plánu zahrnutého v zmluve [15].
- **Dĺžka trasy** — parameter potrebný na výpočet celkových nákladov na prevádzku linky. Dĺžky jednotlivých liniek sú obsahom zmluvy [15].

Kapitola 3

Simulačný model linky mestskej hromadnej dopravy

Krokom následujúcim po získaní potrebných informácií je zostrojenie simulačného modelu. Problém optimalizácie časového rozpisu linky mestskej hromadnej dopravy je veľmi komplexný. Program, ktorý by tento problém riešil analytickým spôsobom by bol veľmi neefektívny a mal by priveľkú časovú zložitosť výpočtu. Preto sme sa rozhodli využiť stochastický spôsob, to jest vytvorenie simulačného modelu, ktorého výstupy budú následne využité pri optimalizácii genetickým algoritmom. Simulačný model bude reprezentovať jednu linku MHD, idúcu jedným smerom. Časti modelu sú zastávky, vozidlá a cestujúci, pričom najdôležitejšou časťou je časový rozpis, ktorý obsahuje odchody jednotlivých vozidiel. Na základe informácii získaných z priebehu simulácie sa rozpis ohodnotí a toto ohodnotenie sa využije v genetickom algoritme na jeho optimalizáciu. Viac informácií o optimalizácii a genetickom algoritme je v kapitole 4.

3.1 DEVS — Discrete Event System Specification

Na modelovanie bol použitý formalizmus DEVS (Discrete Event System Specification). Všetky informácie o tomto formalizme v tejto sekcii sú čerpané zo zborníka z konferencie WSC 2018 [17], 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 [17] je potrebný pre správne fungovanie modelu
- δ_{int} je funkcia, ktorá určuje prechody medzi stavmi modelu

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

Vhodným príkladom je jednoduchý model semafóru, ten sa za normálnej prevádzky nachádza v troch stavoch. Raz svieti na zeleno, inokedy na oranžovo alebo červeno. Každá farba zotrvá rozsvietená stanovený čas a tiež je dané, aká farba bude následovať. Formálny zápis by potom vyzeral takto:

$$\langle S, q_{init}, \delta_{int}, ta \rangle$$

$$S = \{\text{Zelená}, \text{Oranžová}, \text{Červená}\}$$

$$q_{init} = \text{Zelená}$$

$$\delta_{int} = \{\text{Zelená} \rightarrow \text{Oranžová}, \text{Oranžová} \rightarrow \text{Červená}, \text{Červená} \rightarrow \text{Zelená}\}$$

$$ta = \{\text{Zelená} \rightarrow 57, \text{Oranžová} \rightarrow 3, \text{Červená} \rightarrow 60\}$$

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 λ.

- Y je množina výstupných signálov modelu, pomocou ktorých dáva model vedieť svojmu okoliu, že sa nastala nejaká udalosť, napríklad, že sa zmenil jeho stav
- λ 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

V náväznosti na príklad semafóru môžeme definovať výstupné signály, koré nám povedia, aká farba práve svieti na semafóre. Pri definovaní týchto signálov a výstupnej funkcie treba dbať na to, že tieto udalosti sú diskrétne a keď chceme signalizovať napríklad to, že semafór svieti na červeno, musíme to urobiť ešte pred prechodom do stavu červenej farby a teda táto funkcia musí byť definovaná pre stav oranžovej farby.

$$Y = \{\textit{svieti\_zelená}, \textit{svieti\_oranžová}, \textit{svieti\_červená}\}$$

$$\lambda = \{\text{Zelená} \rightarrow \textit{svieti\_oranžová}, \text{Oranžová} \rightarrow \textit{svieti\_červená}, \text{Červená} \rightarrow \textit{svieti\_zelená}\}$$

Vstupné signály

Vstupné signály slúžia na ovládanie modelu z vonku. Napríklad spomínaný semafór doteraz pracoval len na základe svojich vnútorne definovaných stavov a prechodov medzi nimi. V prípade, že nastane nejaká externá udalosť, napríklad sa niekto rozhodne semafór 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 δ_{ext}.

- X je množina vstupných signálov modelu, externých udalostí, ktorých príchod model očakáva

- δ_{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 semafórom rozšírime o nový stav, v ktorom je semafór vypnutý. Semafór v tomto stave zotrvá neobmedzene dlho, pokým nepríde ďalší vstupný signál, ktorý ho zapne. Potom môžeme pridať vstupné signály a funkciu externých prechodov.

$$S = S \cup \{\text{Vypnutý}\}$$

$$ta = ta \cup \{\text{Vypnutý} \rightarrow \infty\}$$

$$X = \{\textit{vypnúť}, \textit{zapnúť}\}$$

$$\delta_{ext} = \{(\text{Zelená}, \textit{vypnúť}) \rightarrow \text{Vypnutý},$$
$$(\text{Oranžová}, \textit{vypnúť}) \rightarrow \text{Vypnutý},$$
$$(\text{Červená}, \textit{vypnúť}) \rightarrow \text{Vypnutý},$$
$$(\text{Vypnutý}, \textit{zapnúť}) \rightarrow \text{Červená}\}$$

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 súčasťou 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
- $\{M_i\} = \{\langle S_i, q_{init,i}, \delta_{int,i}, ta_i, Y_i, \lambda_i, X_i, \delta_{ext,i}\rangle | i \in D\}$ 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 funckcia, ktorá určuje prioritu v skupine konfliktných modelov, prepojovaním 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 popísaný v následujúcej sekcii, kde bude použitý na modelovanie linky mestskej hromadnej dopravy. Skladanie atomických modelov do žložených mi pomohol lepše pochopiť zrozumitelne popísaný príklad v článku [13].

Zhrnutie

DEVS formalizmus umožňuje definíciu atomických modelov, ktorým je možné priradiť stavy, prechody medzi nimi, vstupné a výstupné signály, ako aj definovať ich správanie. Tieto atomické modely sa dajú spájať do zložených modelov definovaním väzieb medzi nimi navzájom. Všetky informácie o DEVS formalizme, definície jednotlivých prvkov a aj príklady sú čerpané zo zborníka z konferencie WSC 2018 [17]. Tento zborník obsahuje aj príklady kódu napísané v jazyku `Python`, s použitím knižnice `PythonPDEVS`. Trochu zložitejším, no o to podrobnejším a formálnejším spôsobom je popísané vysvetlenie DEVS formalizmu je popísané v originálnej špecifikácii v knihe [18].

3.2 Popis simulačného modelu

Ako prvé je potrebné definovať základné atomické prvky modelu, ich stavy a prepojenie medzi nimi. Simulačný model je zložený z dvoch hlavných častí, tými sú:

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

Simulácia obsahuje viaceré inštancie modelu zastávky, každá reprezentujúca jednu konkrétnu zastávku na linke, viaceré inštancie modelu vozidla, ktoré reprezentujú jednu cestu od nástupnej na konečnú zastávku (nejedná sa o fyzické vozidlá). Atomický model cestujúcich nie je potrebný, sú súčasťou modelu zastávky, na ktorej čakajú. Sú reprezentovaní jedným číslom, časom kedy prišli na zastávku.

Atomický model zastávky

Model zastávky predstavuje jednotlivé zastávky na linke, každá reálna zastávka je v systéme ako samostatná inštancia modelu zastávky. Je pasívný, to znamená, že jeho vnútorný stav sa plne odvíja od aktivácie vstupných signálov. Ako je vidieť vo formálnej definícii, tento model nemá definované žiadne interné prechody, ani výstupné signály.

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

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

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

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

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

$$Y = \emptyset$$

$$\lambda = \emptyset$$

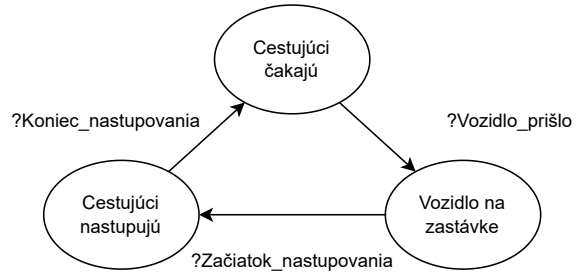
$$X = \{\textit{?Vozidlo\_prišlo}, \textit{?Začiatok\_nastupovania}, \textit{?Koniec\_nastupovania}\}$$

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

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

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

Na obrázku 3.1 je graficky znázornený stavový automat modelu zastávky. Zobrazuje tri stavy modelu a prechody medzi nimi, ktoré sú vyvolané 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 v rámci definície zloženého modelu neskôr.



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 a aktuálnym príchodom vozidla
?Začiatok_nastupovania	generovanie príchodov cestujúcich
?Koniec_nastupovania	prepísanie času posledného príchodu vozidla

Tabuľka 3.2: Vstupné signály modelu zastávky

Generovanie príchodov cestujúcich

Generovanie príchodov cestujúcich prebieha na základe exponenciálneho rozdelenia s parametrom λ v časovom intervale medzi príchodom posledného a aktuálneho vozidla. 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 príchodu cestujúcich na zastávku. Ak by nám na správne fungovanie modelu stačil iba počet cestujúcich, na jeho výpočet by sme použili Poissonovo rozdelenie. Pre výpočet doby čakania na zastávke je potrebné vedieť aj čas príchodu jednotlivých cestujúcich. Preto je v simulačnom modeli použité exponenciálne rozdelenie, ktoré generuje náhodný časový interval medzi príchodami jednotlivých cestujúcich. Vzťah medzi týmito dvoma rozdeleniami a podrobnejší popis exponenciálneho rozdelenia je v článku [5]. Pripočítaním náhodného časového intervalu k aktuálnemu modelovému času sa vygeneruje čas príchodu cestujúceho, ktorý sa uloží do fronty predstavujúcej všetkých cestujúcich čakajúcich na zastávke. Modelový čas sa zmení na čas príchodu cestujúceho. Tento proces sa opakuje až kým modelový čas nedosiahne čas príchodu aktuálneho vozidla.

Algoritmus 1: Generovanie príchodov cestujúcich

```
získanie λ pre danú zastávku 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.

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

$$S = \{\text{Na ceste}, \text{Na zastávke}, \text{Nakladá cestujúcich}\}$$

$$q_{init} = \text{Na ceste}$$

$$\delta_{int} = \{\text{Na ceste} \rightarrow \text{Na zastávke}, \text{Na zastávke} \rightarrow \text{Nakladá cestujúcich},$$

$$\text{Nakladá cestujúcich} \rightarrow \text{Na ceste}\}$$

$$ta = \{\text{Na ceste} \rightarrow \infty, \text{Na zastávke} \rightarrow 0, \text{Nakladá cestujúcich} \rightarrow 0\}$$

$$Y = \{\textit{!Príchod}, \textit{!Nastupovanie}, \textit{!Odchod}\}$$

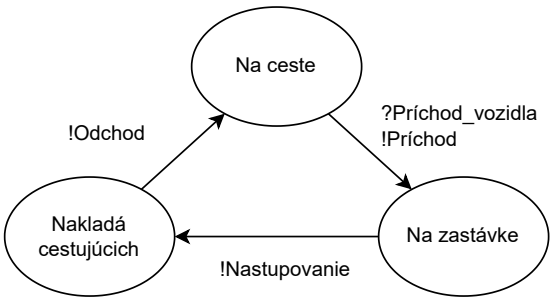
$$\lambda = \{\text{Na ceste} \rightarrow \textit{!Príchod}, \text{Na zastávke} \rightarrow \textit{!Nastupovanie},$$

$$\text{Nakladá cestujúcich} \rightarrow \textit{!Odchod}\}$$

$$X = \{\textit{?Príchod\_vozidla}\}$$

$$\delta_{ext} = \{(\text{Na ceste}, \textit{?Príchod\_vozidla}) \rightarrow \text{Na zastávke}\}$$

Na obrázku 3.2 je graficky znázornený stavový automat modelu vozidla. Zobrazuje 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 uvedené, že model vozidla zotrváva v stave **Na ceste** neobmedzene dlho. Prechod z tohoto stavu je podmienený príchodom vstupného signálu *?Príchod_vozidla*. Pôvod tohoto signálu bude popísaný v rámci definície zloženého modelu neskôr.



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, napríklad ak je koeficient 0,7, vystúpi 70 percent cestujúcich
!Nastupovanie	nástup cestujúcich do vozidla, tí ktorí sa nezmestia do vozidla sú nespokojní a následne opúšťajú systém
!Odchod	okrem aktivácie prepojených vstupných signálov sa nič iné nevykonáva

Tabuľka 3.4: Výstupné signály modelu vozidla

Obsluha zastávky

Každé vozidlo začina na prvej zastávke v rozpise a 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.

$$D = \{\text{Zastávka}, \text{Vozidlo}\}$$

$$M_{Zastvka} = \langle \ldots \rangle$$

$$M_{Vozidlo} = \langle \ldots \rangle$$

$$I_{Zastvka} = \emptyset$$

$$I_{Vozidlo} = \{\text{Zastávka}\}$$

$$X_{self} = \emptyset$$

$$Y_{self} = \{!Príchod\_vozidla\}$$

$$select = \{\{Zastvka, Vozidlo\} \rightarrow Vozidlo, \{Zastvka\} \rightarrow Zastvka, \{Vozidlo\} \rightarrow Vozidlo\}$$

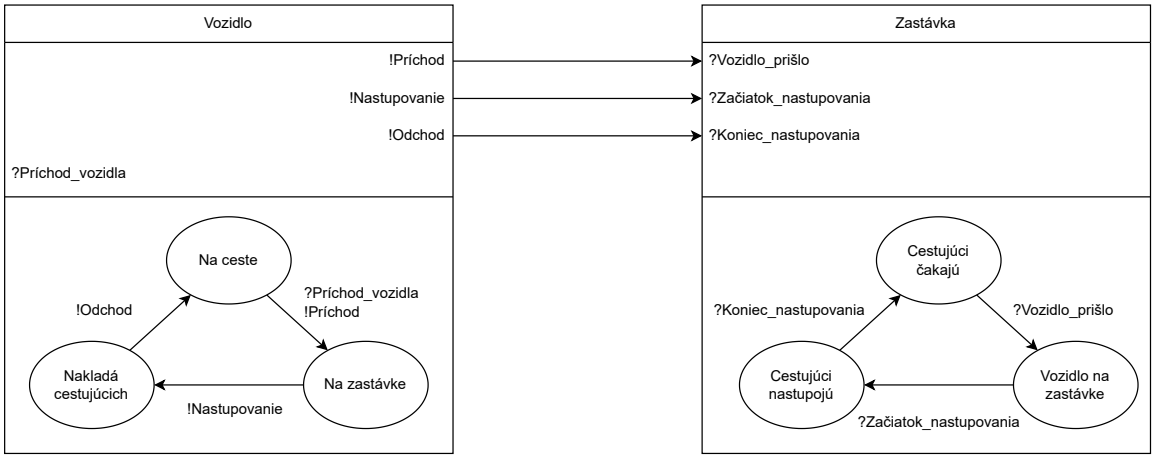
$$Z_{i,j,self} = \{!Príchod\_vozidla \rightarrow ?Príchod\_vozidla,$$

$$!Príchod \rightarrow ?Vozidlo\_prišlo,$$

$$!Nastupovanie \rightarrow ?Začiatok\_nastupovania,$$

$$!Odchod \rightarrow ?Koniec\_nastupovania\}$$

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 obslúženia 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 odchádza na ďalšiu zastávku. Za jednu udalosť teda vozidlo prechádza všetkými svojimi vnútorným stavmi, udalosť končí opäť v stave **Na ceste**. Zastávka, ktorá je uvedená v parametri udalosti, rovnako prechádza 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 vykonať. Vykonávané udalosti sú v prípade tohto modelu príchody vozidiel na zastávky. V rámci jednej udalosti sa vykoná rutina obslúženia cestujúcich čakajúcich na zastávke. Všetky udalosti sú naplánované na začiatku simulácie, pretože na základe časovému rozpisu sú presne definované všetky príchody vozidiel na jednotlivé zastávky. Príchody na počiatočnú zastávku sú dané hlavným časovým rozpisom, príchody na ostatné zastávky sú odvodené od času potrebného na presun z jednej zastávky na následujúcu. Pseudokód plánovania udalostí je uvedený v algoritme 3 a spolu s kódom v implementácii je prevzatý zo študijnej opory predmetu Modelovanie a simulácie [11], ktorý je vyučovaný na Fakulte informatiky Vysokého učení technického v Brne. V priebehu simulácie už nie je potrebné plánovať žiadne ďalšie udalosti.

Algoritmus 3: Plánovanie udalostí

```
for čas v časovom rozpise počiatočnej zastávky do
    Vytvorí sa vozidlo (inicializuje sa objekt);
    for zastávka na linke do
        Pridá sa udalosť príchodu vozidla na danú zastávku;
```

Po naplánovaní všetkých udalostí sa spustí simulácia. Postupne sa vykonávajú jednotlivé udalosti a simulácia sa diskrétne posúva vpred dovtedy, až kým nie je kalendár udalostí

prázdny alebo nie je dosiahnutý koncový čas simulácie. Pseudokód riadenia simulácie je popísaný v algoritme 4, teno kód je tiež prevzatý zo študijnej opory [11].

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 obslúži cestujúcich);
```

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

3.3 Výstupy simulačného modelu

Hlavný zmysel simulácie je postupné generovanie informácií, ktoré by sme inak museli získať z pozorovania reélnej prevádzky linky. Efektivitu každého jedného časového rozpisu, ktorý program vytvorí, je potrebné otestovať a ohodnotiť, pre výber najlepšieho 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 3.7. aj pre každé vozidlo 3.6 osobitne. Následne sa agregujú do jednoho objektu `Statistics`, ktorého obsah je popísaný v tabuľke 3.5.

Parameter	**Popis**
`totalNumberOfBuses`	Počet spojov v priebehu dňa na linke
`busStopStatistics`	Agregované štatistiky zastávok
`busStatistics`	Agregované štatistiky vozidiel

Tabuľka 3.5: Trieda `Statistics`

Parameter	**Popis**
`capacity`	Celková 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 3.6: Trieda `busStatistics`

Parameter	Popis
`totalPassengersArrived`	Koľko cestujúcich prišlo na zastávku za deň
`totalPassengersDeparted`	Koľko 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 za hodinu
`passengersDepartedPerHour`	Počet cestujúcich, ktorí vystúpili na zastávke za hodinu
`passengersLeftUnboardedPerHour`	Počet cestujúcich, ktorí sa nezmestili do vozidla za hodinu
`timeSpentWaitingPerHour`	Čas, ktorý cestujúci strávili čakaním na zastávke za hodinu

Tabuľka 3.7: Trieda `busStopStatistics`

Počet spojov v priebehu dňa na linke

Tento údaj slúži pre optimalizáciu prevádzkových nákladov. Z pohľadu dopravného podniku je potrebné dosiahnuť, aby bol počet spojov čo najnižší. Z pohľadu cestujúcich je potrebné aby bol počet spojov čo najväčší, aby nemuseli čakať dlho na spoj a estovali komfortne. Preto je dôležité nájsť správny kompromis medzi týmito dvoma požiadavkami.

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.

Koľko cestujúcich vystúpilo 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

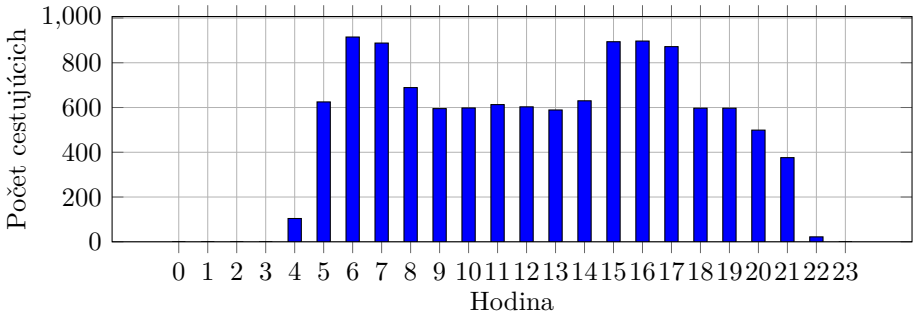
Môže nastať situácia, že je vozidlo tak preplnené, že nie je možné aby všetci cestujúci čakajúci na zastávke do neho nastúpili. Rozpisy, pri ktorých sa tento jav vyskytne, budú mať horšie hodnotenie.

Čas, ktorý cestujúci strávili čakaním na zastávke

Čas, ktorý cestujúci strávili čakaním na zastávke sledujeme v dvoch rôznych údajoch, raz ako celkový čas čakania a raz ako priemerný čas čakania jedného cestujúceho. V oboch prípadoch chceme, aby bol tento čas čo najnižší.

Počet cestujúcich, ktorí prišli na zastávku za hodinu

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 za hodinu

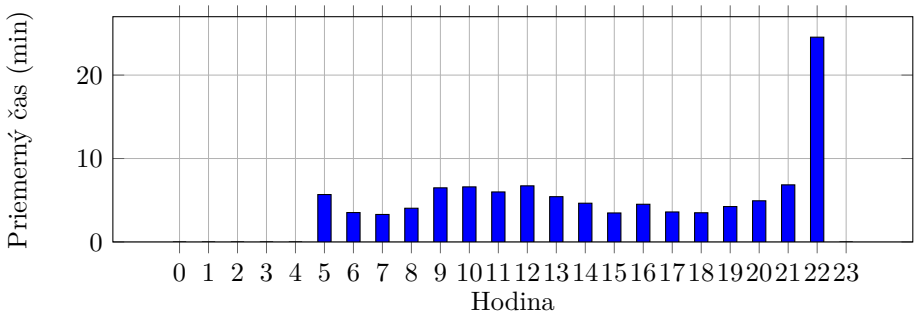
Jedná sa o rovnaký údaj ako je počet cestujúcich, ktorí vystúpili na zastávke za celý deň, ale rozdelený na jednotlivé hodiny dňa.

Počet cestujúcich, ktorí sa nezmestili do vozidla za hodinu

Tento údaj informuje, že linka je v určitej časti dňa poddimenzovaná a bolo by vhodné v tomto čase linku posilniť.

Čas, ktorý cestujúci strávili čakaním na zastávke za hodinu

Čas strávený čakaním na zastávke by mal byť počas celého dňa približne rovnaký. Na grafe 3.5 je vidieť ako sa priemerný čas čakania pohybuje v rozsahu približne 5 minút, ale o desiatej hodine večer tento čas náhle narastá. Z uvedeného vyplíva, že by v tejto hodine malo byť nasadených viac vozidiel.



Obr. 3.5: Priemerný čas strávený čakaním za hodinu

Celková 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 nežiadúca.

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 keby mal stáť.

Priemerná naplnenosť vozidla

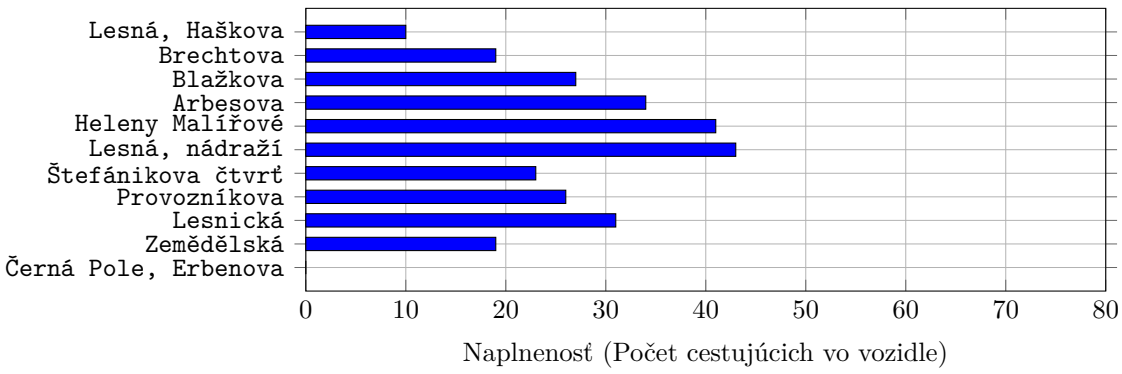
Priemerná naplnenosť vozidla je dôležitý parameter, ktorý nás informuje, ako veľmi sú vozidlá využívané. Z pohľadu cestujúcich je lepšie, ak je vozidlo prázdnejšie, ale z pohľadu efektivity dopravného podniku je prázdnejšie vozidlo nevyužitý ekonomický potenciál. Preto je potrebné nájsť kompromis medzi týmito dvoma požiadavkami.

Celkový počet prevezených cestujúcich

Celkový počet prevezených cestujúcich závisí od toho, koľko cestujúcich prišlo na jednotlivé zastávky a či sa im podarilo nastúpiť do jednotlivých vozidiel. V ideálnom prípade sa tento počet bude rovnať celkovému počtu cestujúcich, ktorí prišli na zastávky. V extrémnych prípadoch sa môže stať, že niektorí cestujúci nenastúpia do preplnených vozidiel.

Naplnenosť vozidla na jednotlivých zastávkach

Graf 3.6 znázorňuje ako sa naplnenosť vozidiel mení počas ich jazdy na linke 46 v Brne. Naplnenosť sa postupne zvyšuje až kým nepríde na jednu z hlavných prestupných zastávok, Štefánikova čtvrť, kde vystúpi veľké množstvo cestujúcich. Pokiaľ má linka veľké výkyvy v naplnenosti vozidiel, je potrebné trasu na určitých úsekoch posilniť inou linkou.



Obr. 3.6: Priemerná naplnenosť vozidla na jednotlivých zastávkach linky 46

Spokojnosť jednotlivých zákazníkov

Spokojnosť zákazníkov je kľúčová pri optimalizácii, na základe nej sa rozpis ohodnocuje. Spokojnosť program počíta za každého cestujúceho a nakoniec sa výsledok zpriemeruje.

Kapitola 4

Optimalizácia časového rozpisu linky mestskej hormadnej dopravy

Táto kapitola pojednáva o potrebných vstupoch, obmedzeniach, použitej optimalizačnej metóde a výstupoch optimalizácie. Na optimalizáciu časového rozpisu 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 takzvaná 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 Potrebné používateľské vstupy

Pre správne fungovanie algoritmu, je potrebné zadať niekoľko povinných vstupných parametrov týkajúcich sa zastávok, linky a samotného genetického algoritmu.

Vstupné parametre zastávky

- **Názov zastávky** — názov zastávky, zobrazovaný na výstupoch
- **Čas príchodu** — predpokladaný čas potrebný na prejazd od počiatočnej zastávky
- **Štatistika o príchodoch cestujúcich** — priemerný počet prichádzajúcich cestujúcich v každej hodine dňa
- **Pravdepodobnosť výstupu cestujúceho** — koeficient potrebný pre priebežné vyprázdnenie vozidla

Tieto informácie sa importujú formou textového súboru, každá zastávka je na samostatnom riadku a jednotlivé údaje sú oddelené dvojbodkou:

```
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
...
```

Vstupné parametre linky

- **Kapacita vozidla** — maximálny možný počet cestujúcich vo vozidle
- **Miesta na sedenie** — počet miest na sedenie vo vozidle
- **Celkové náklady** — udávané v korunách na 100 miestokilometrov (vysvetlné v sekcii 4.2)
- **Dĺžka trasy** — dĺžka trasy v kilometroch

Kapacita vozidla a počet miest na sedenie sú potrebné pre výpočet naplnenosti vozidla, od ktorej 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.

Vstupné parametre genetického algoritmu

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

Vstupné obmedzenia časového rozpisu

Poslednou informáciou, ktorá je od používateľa potrebná je obmedzenie počtu spojov za hodinu. Používateľ takto môže nastavuje fixný počet spojov, ktorý sa musí vygenerovať za hodinu. Toto obmedzenie sa využíva hľavne pri plánovaní náväznosti 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 je využívané aj v predbežnej analýze, ktorá sa vykonáva pred optimalizáciou. V prípade že pre niektorú hodinu neexistuje údaj o priemernom počte prichádzajúcich cestujúcich na všetky zastávky linky, automaticky sa nastaví fixný počet spojov v danej hodine na 0 (napríklad nočná linka nepremáva cez deň). Zadaním vstupných ombedzení časového rozpisu sa znižuje náročnosť výpočtu a zvyšuje kvalita nájdeného riešenia.

4.2 Základný genetický algoritmus

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 pomocou evolučného procesu.

Používajú sa postupy ako náhodná mutácia jedinca, rôzne typy výberu a kríženia jedincov. Základnou premysou je, že najlepší jedinci v generácii buď splňujú zadané požiadavky alebo sú veľmi blízko k ich splneniu. Predpokladá sa, že títo jedinci majú najvhodnejšie predispozície na to, aby ich potomkovia lepšie spĺňali zadané podmienky. 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. Uvedené 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 chromozómom pole obsahujúce 24 čísel (viď tabuľku 4.1), ktoré predstavujú počet odchodov vozidiel z počiatočnej 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 boli definované dva hlavné ciele:

- **Spokojnosť cestujúcich** — priemerné percentuálne hodnotenie spokojnosti všetkých cestujúcich, ktorí prišli za daný deň na zastávku
- **Náklady na prevádzku** — celkové náklady na prevádzku linky za deň, udávané v korunách na 100 miestokilometrov

Výpočet spokojnosti cestujúceho sa vykonáva v čase nástupu do vozidla. Hodnotí sa aktuálna naplnenosť vozidla, na základe ktorej sa vypočíta percentuálne hodnotenie spokojnosti. Potrebné parametre pre výpočet sú kapacita vozidla, počet miest na sedenie a aktuálna naplnenosť vozidla.

Výpočet sa riadi následujúcimi 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 cestujúceho a je vypočítaná podľa rovnice 4.1 následovne:

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

Spokojnosť všetkých cestujúcich sa vypočíta ako priemer spokojnosti jednotlivých cestujúcich. Hodnota spokojnosti sa pohybuje od 0 do 1 a genetický algoritmus automaticky uprednostňuje vyššie hodnoty, teda dosiahnutie vyššej spokojnosti u cestujúcich.

Celkové náklady na prevádzku linky sú vypočítané z počtu a kapacity použitých vozidiel v časovom rozpise, dĺžky trasy a nákladov na jeden miestokilometer. Jeden miestokilometer je definovaný ako vzdialenosť jedného kilometra, ktorú prejde jedno miesto vo vozidle [7]. Napríklad vozidlo s kapacitu 80 miest, ktoré prejde 10 kilometrov, prejde 800 miestokilometrov. Miestokilometre sa môžu násobiť aj počtom vozidiel, potom dve takéto vozidlá by prešli 1600 miestokilometrov. Miestokilometer je potrebný pre výpočet efektivity linky, 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 podníkom mesta Brna [15]. Celý výpočet nákladov na prevádzku linky teda závisí od štyroch premenných a je daný rovnicou 4.2:

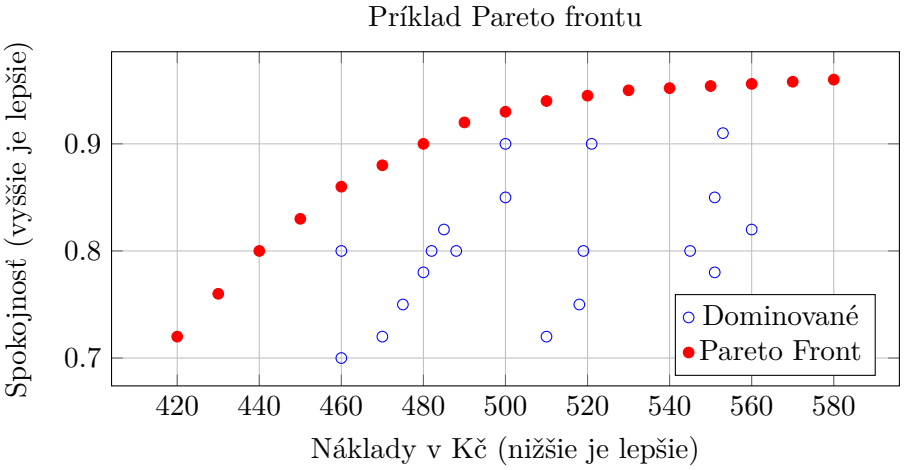
$$\text{celkové náklady} = \frac{\text{dĺžka trasy} \cdot \text{počet spojov} \cdot \text{kapacita vozidla}}{100} \cdot \text{náklady} \tag{4.2}$$

Tieto dva ciele sú v neustálom konflikte — viac spojov na linke a vyššia spokojnosť cestujúcich voči rastúcim nákladom na prevádzku DPMB. Túto situáciu rieši multi-objektívna optimalizácia, ktorá sa snaží nájsť optimálne riešenie pre oba ciele naraz.

Genetický algoritmus s viacerými cieľmi

Genetický algoritmus, ktorým sa snažíme dosiahnuť optimálneho kompromisu medzi viacerými cieľmi sa nazýva multi-objektívny genetický algoritmus. Jeho podstata 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 existuje jeden jedinec, ktorý je lepší v oboch cieľoch, hovoríme, že dominuje iného jedinca. V prípade, že dominuje iba v jednom cieli, 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 boli prevzaté z článku [10].



Obr. 4.1: Príklad Pareto frontu

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 algoritmami. NSGA-II vo viacerých testoch preukázal, že má lepšie rozloženie a diverzitu jedincov v Pareto fronte ako iné algoritmy. Práve pre túto vlastnosť bol zvolený aj pre optimalizáciu časového rozpisu. 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 naše účely.

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 zhluková 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. Zhluková 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. Zhlukové vzdialenosti pre jednotlivé ciele sa následne sčítajú a tým sa získa celková zhluková vzdialenosť jedinca. Celý postup pre priradenie zhlukovej 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 zhluková vzdialenosť na základe jeho vzdialenosti od svojich susedov vo fronte. Potom sa najepšie fronty prenesú do novej generácie. Posledný prenášaný front sa pravdepodobne do novej generácie 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 zhlukovej 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 všetkých jedincov, ktorých dominuje p a počet n_p, ktorý udáva počet jedincov, ktorí dominujú p. Pokiaľ je $n_p = 0$, jedinec p je nie je nikým dominovaný a pripojí sa do prvého frontu. Taktiež sa mu priradí hodnota $p_{rank} = 1$. Rank udáva kvalitu jedinca.

Následuje druhý hlavný cyklus algoritmu 6, ktorý vytvorí zvyšné fronty. Princíp je taký, že pre každého jedinca p v posledne vytvorenom fronte sa upravia hodnoty n_q všetkých jedincov q z množiny S_p. To vpodstate 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 pridajú do ďalšieho frontu F_{i+1}. Tento cyklus sa opakuje až pokým sa do najnovšieho frontu nemá kto pridať.

Algoritmus 6: Rýchle nedominantné triedenie

```
Input: Populácia P
Output: Fronty nedominovaných jedincov F
F ← ∅ (množina frontov);
for každý jedinec p ∈ P do
    S_p ← ∅ (množina jedincov, ktorých dominuje p);
    n_p ← 0 (počet jedincov, ktorí dominujú p);
    for každý jedinec q ∈ P : q ≠ p do
        if p dominuje q then
            S_p ← S_p ∪ {q};
        else if q dominuje p then
            n_p ← n_p + 1;
    if n_p = 0 then
        p_rank ← 1;
        F_1 ← F_1 ∪ {p} (pridanie jedinca p do prvého frontu);
i ← 1;
while F_i ≠ ∅ do
    Q ← ∅ (množina jedincov ďalšieho frontu);
    for každý jedinec p ∈ F_i do
        for každý jedinec q ∈ S_p do
            n_q ← n_q − 1;
            if n_q = 0 then
                q_rank ← i + 1;
                Q ← Q ∪ {q};
    i ← i + 1;
    F_i ← Q;
return F;
```

Priradenie zhlukovej vzdialenosti

V rámci jednoho 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 optimalizácie časového rozpisu 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 čiastočné 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**
  $p_{vzdialenost} \leftarrow 0$ (inicializácia zhlukovej vzdialenosti);
for *každý cieľ* c **do**
  $L \leftarrow sort(L, c)$ (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**
    $L[i]_{vzdialenost} \leftarrow L[i]_{vzdialenost} + \frac{L[i+1].c - L[i-1].c}{L[l-1].c - L[0].c}$;

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 $p_{rank} < q_{rank} \vee (p_{rank} = q_{rank} \wedge p_{vzdialenost} > q_{vzdialenost})$;
(dá sa rozšíriť aj o obmedzenia);

Toto porovnávanie sa dá rozšíriť aj o rôzne obmedzenia. Napríklad, v rámci optimalizácie časového rozpisu máme dva cieľe, 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 nižší. 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 $\prec_n$. V optimalizácii časového ropisu máme len jedno obmedzenie, počet cestujúcich, ktorí sa nezmestili do vozidla, bližšie informácie k implementácii viacerých obmedzení sú v článku [2].

Inicializácia prvej generácie

Populácia prvej genrácie je potrebné spracovať iným spôsobom ako ostatné generácie. Hlavný cyklus algoritmu NSGA-II sa začína spojením rodičov a potomkov do jedného celku. Algoritmus 9 popisuje ako sa vytvorí populácia prvej generácie a jej potomkovia. Najprv sa vytvorí prvá generácia jedincov (rodičov), tým sa náhodne vygenerujú chromozómy. Následne sa týto jedinci roztriedia na fronty pomocou rýchleho nedominantného triedenia. V rámci každého frontu sa vypočíta zhluková vzdialenosť každého jedinca. Na základe získaných ohodnotení sa vyberú najvhodnejší rodičia a za pomoci kríženia a mutácie sa vytvorí prvá generácia potomkov.

Algoritmus 9: Inicializácia prvej generácie

```
Input: Veľkosť populácie N
t ← 0 (počiatočná generácia);
P_t ← ∅ (inicializácia populácie);
for i = 0 do N − 1 do
    p_i ← vytvorenie_jedinca();
    P_t ← P_t ∪ {p_i};
F ← rýchle_nedominantné_triedenie(P_t);
for každý front f ∈ F do
    priradenie_zhlukovej_vzdialenosti(f);
Q_t ← vytvorenie_nových_potomkov(P_t) (kríženie a mutácia);
```

Hlavná slučka algoritmu NSGA-II

Po inicializácii prvej generácie sa následné generácie spracovávajú rovnako. Ako je vidieť v algoritme 10, hlavná slučka začína spojením rodičov a potomkov do jednej populácie. Novo vzniknutá 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 dovtedy, kým sa celý následujúci front zmestí do novo vytváranej generácie. Posledný front, ten ktorý sa už nezmestí celý, sa spracováva osobitne. Prvky z tohto frontu sa usporiadajú na základe zhlukovej vzdialenosti a do novej generácie sa prenesú len najlepšie 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 berie 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);
$F \leftarrow$ 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**
    priradenie_zhlukovej_vzdialenosti(F_i);
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$;
    $i \leftarrow i + 1$;
$sort(F_i, \prec_n)$;
$P_{t+1} \leftarrow P_{t+1} \cup F_i[0 : N - |P_{t+1}|]$;
$Q_{t+1} \leftarrow$ 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 skladá z výberu rodičov, kríženia rodičov a mutácie nových 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**
    $p_1, p_2 \leftarrow$ výber_rodiča(P);
    $q_1, q_2 \leftarrow$ kríženie_rodičov(p_1, p_2);
    mutácia_jedinca(q_1, 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 výber vhodných rodičov na kríženie 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 zvolený pre jednoduchosť na implementáciu a jeho dobré výsledky v porovnaní s inými typmi výberu rodičov, ktoré je uvedené v zborníku z konferencie CEC [4].

Algoritmus 12: Výber rodiča

Input: Populácia P
Output: Rodič p
$p_1, p_2 \leftarrow$ náhodný jedinec z P;
if $p_1 \prec_n p_2$ **then**
    return p_1;
else
    return p_2;

Kríženie rodičov

Výmenou génov v chromozómoch rodičov vznikajú dvaja noví jedinci, potomkovia. 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. Na základe porovnania rôznych typov kríženia zo zborníku z konferencie ICIIP [1] bol zvolený typ kríženia s najväčšou diverzitou výsledkov, aby sa zamedzilo predčasnej konvergencii. Ide o uniformné kríženie 13.

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 $random(0, 1) < 0,5$ **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;

Mutácia jedinca

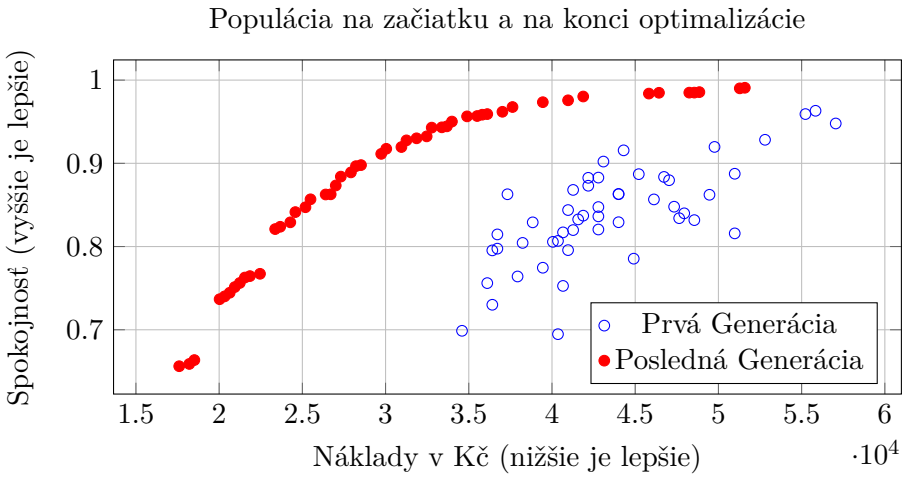
Posledným krokom v procese vytvorenia nových potomkov je mutácia. Mutácia zvyšuje diverzitu jedincov a zabraňuje predčasnému uviaznutiu v lokálnych extrémoch. V algoritme 14 je znázornená mutácia jedinca. Pre každý gén v chromozóme sa náhodne vyberie číslo z intervalu $[0, 1]$. Ak je toto číslo menšie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

Algoritmus 14: Mutácia jedinca

Input: Jedinec p
for *každý gén* g *v* p **do**
    if $random(0, 1) <$ *pravdepodobnosť mutácie* **then**
        $p[g] \leftarrow$ 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 najlepšími 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 optimalizácie časového rozpisu 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. Toto rozhodnutie sa musí nechať na používateľa programu, 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.



Obr. 4.2: Populácia na začiatku a na konci optimalizácie

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.

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 reakcie algoritmu 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 [1]
- **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.

[1] Tento počet bol znížený z dôvodu dlhej doby výpočtu.

5.1 Experiment s krátkou trasou a krátkym vozidlom

Cieľom tohto experimentu je zistiť, ako sa algoritmus vyrovnáva s krátkou trasou a krátkym 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 neobslúžených cestujúcich je 0. Vozidlá sú v priemere zaplnené do jednej tretiny, pretože ich je až 115, to je o 3 vozidlá menej ako je aktuálny rozpis linky 46. Z toho vyplíva, že musia byť menšie aj celkové prevádzkové náklady linky 46.

Výstup	**Hodnota**
Počet príchodov cestujúcich	11603
Počet prevezených cestujúcich	11603
Počet neobslúžených 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 výsledky optimalizácie rovnakej linky ako v predošlom experimente, avšak s vozidlom s vyššou kapacitou. Navýšenie kapacity 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%
- Celkové náklady sa zvýšili o 8183 Kč

Výstup	**Hodnota**
Počet príchodov cestujúcich	11528
Počet prevezených cestujúcich	11528
Počet neobslúžených 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 neobslúžených 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 následovné parametre:

- Priemerná čakacia doba sa zvýšila o 2 minúty
- Spokojnosť cestiujúcich sa zvýšila o 1,88%
- Počet vozidiel sa znížil o 29 kusov
- Priemerná naplnenosť vozidiel sa znížila o 9,04%
- Celkové náklady sa zvýšili o 90518 Kč

Výstup	**Hodnota**
Počet príchodov cestujúcich	48358
Počet prevezených cestujúcich	48358
Počet neobslúžených 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 slúži ako rozhranie medzi požívateľom 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.

Používateľské rozhranie by malo byť intuitívne a jednoduché. Hlavným cieľom tejto práce je optimalizácia, preto je kladený väčší dôraz na ňu ako na samotné používateľské rozhranie. Toto rozhranie je pre používanie optimalizačného algoritmu nevyhnutné, bez ohľadu na jeho jednoduchosť, nesmie byť jeho vývoj zanedbaný.

6.1 Návrh používateľského rozhrania

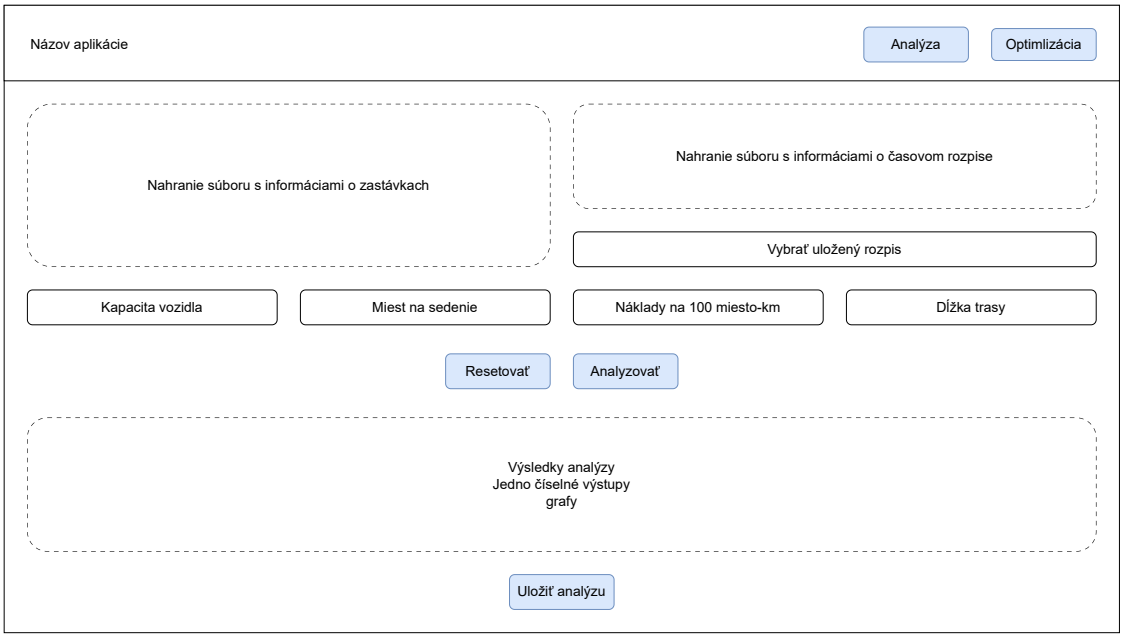
Prvý krok pri tvorbe používateľských rozhraní je jeho návrh. Je dôležité si premyslieť funkcionalitu akú bude rozhranie poskytovať. Tento nástroj by mal vedieť analyzovať nainportované časové rozpisy, vizualizovať výsledky simulácie, prezentovať ich formou grafov a štatistík a optimalizovať časové rozpisy. Výpočet optimalizácie pomocou genetického algoritmu je časovo náročný úkon a preto je nutné používateľa informovať o jeho priebehu formou zobrazenia aktuálnej generácie. Po dokončení optimalizácie bude mať používateľ možnosť vybrať si rozpis, ktorý najviac zodpovedá jeho potrebám a uložiť si ho. Používateľské rozhranie by malo umožniť aj analýzu už uložených rozpisov, prípadne export jednotlivých analýz do napríklad textového súboru alebo PDF.

Všetky uvedené funkcie budú rozdelené medzi dve stránky, stránku pre analýzu a stránku pre optimalizáciu. Na prepínanie medzi stránkami bude slúžiť jednoduché menu v hornej časti okna. Na ich návrh boli použité wireframy.

Stránka pre analýzu časového rozpisu

Na stránke analýzy časového rozpisu bude možné:

- naimportovať informácie o zastávkach
- naimportovať informácie o časovom rozpise
- vybrať uložený časový rozpis
- zadať parametre pre simuláciu
- resetovať stránku (vyprázdniť vstupy)
- spustiť simuláciu
- zobraziť výsledky analýzy, vrátane grafov a štatistík
- uložiť výsledky analýzy



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. Na import informácií o zastávke a nového časového ropisu bude slúžiť vyberač súborov. Importný súbor bude vo formáte TXT. Uživateľské rozhranie by malo obsahovať nápovedu, ktorá používateľovi poskytne vzor formátu vsupných súborov.
Formát potrebný na import 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 import 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 číslenými 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 neobslúžených 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 neobslúžených cestujúcich za hodinu

Posledným grafom bude:

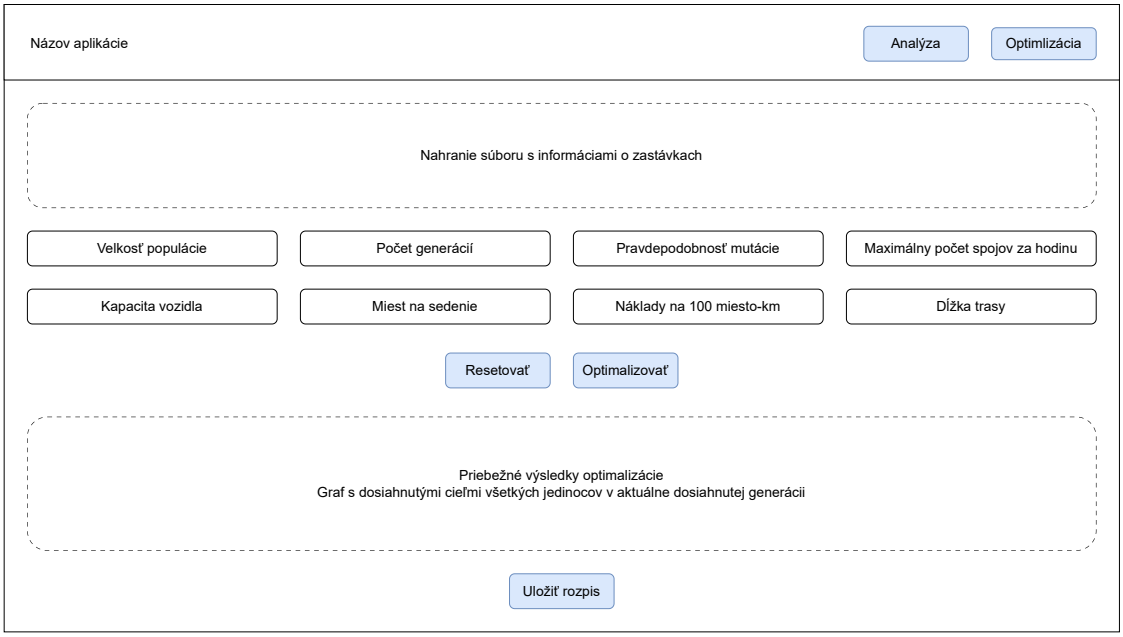
- Priemerná naplnenosť vozidiel na linke

Po stlačení tlačítka `Uložiť analýzu` sa otvorí vyberač súborov, aby si používateľ mohol z adresárovej štruktúry vybrať kam sa vygenerovaného analýza uloží. Analýza sa bude ukladať vo formáte TXT.

Stránka pre optimalizáciu časového rozpisu

Na stránke optimalizácie časového rozpisu bude možné:

- naimportovať informácie o zastávkach
- zadať parametre pre optimalizáciu
- Resetovať stránku
- spustiť optimalizáciu
- zobraziť priebežne dosiahnuté cieľe aktuálne spracovanej generácie
- vybrať rozpis z generácie
- uložiť vybraný 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 v tomto prípade nie je potrebný, pretože z pohľadu optimalizácie je časový rozpis výstupom a nie vstupom. Číselné vstupné parametre pre simuláciu sú rovnaké ako na stránke analýzy, ale pre genetický algoritmus je navyše potrebné zadať štyri nové a to `Velkosť 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:

- Ktorú generáciu algoritmus aktuálne spracováva
- 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 pri uložení automaticky pomenuje časom a dátumom jeho vytvorenia

6.2 Implementácia nástroja

Po návrhu nasleduje implementácia. Prvým krokom implementácie je výber vhodných technológií a druhým je návrh architektúry aplikácie.

Použité technológie

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

`Python` bol však vybraný vďaka jednoduchému použitiu a možnostiam naprogramovania ako backendu, tak aj frontendu (na programovanie frontendu bol využitý framework `Reflex` [12]).

Vďaka tomuto 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 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 [12].

Takáto verzatilita jazyka a frameworku bola pri programovaní tohto nástroja veľmi vítaná, treba však dávať pozor na prehladnosť a previazanosť kódu aby sa v budúcnosti dal ľahko rozširovať. Je potrebné zvoliť si dobrú architektúru, rozčlenenie aplikácie na moduly a tie na jednotlivé webové komponenty alebo backendové skripty, pretože tieto technológie umožňujú veľkú variabilitu písania zdrojového kódu aplikácie.

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. Adresárová štruktúra sa nachádza v prílohe E.

Súbor	**Obsah**
Backendové skripty	
EventCalendar.py	trieda kalendára udalostí a udalosti
Genetics.py	trieda genetického algoritmu a jedného jedinca
InputParser.py	trieda s metódami na spracovanie vstupu
models.py	model zastávky, vozidla a časového rozpisu
RandomNumberGenerator.py	trieda generátora náhodných čísel
Simulation.py	trieda na riadenie simulácie
Statistics.py	trieda na zber a spracovanie štatistík
Webové komponenty	
analyzeLine.py	analýza časového rozpisu
busStopChart.py	zobrazenie grafu so zastávkami na osi x
busStopTable.py	zobrazenie tabuľky so zastávkami
constraintInput.py	zadanie obmedzení počtu spojov za hodinu
footer.py	pätička webovej aplikácie
hourChart.py	zobrazenie grafu s hodinami na osi x
infoCard.py	zobrazenie jednočíselných štatistík
infoUpload.py	import súboru
layout.py	základné rozloženie stránky
navbar.py	vrchné navigačné menu
numberInput.py	zadanie jednočíselných vstupov
optimizeLine.py	optimalizácia časového rozpisu
rozpis.txt	vzorový príklad informácií o časovom rozpise
timeTable.py	zobrazenie časového rozpisu
zastavky.txt	vzorový príklad informácií o zastávkach
Stránky aplikácie	
analyzePage.py	stránka pre anaýlzu časového rozpisu
homePage.py	úvodná stránka
optimizePage.py	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 [9], ktorá bola vygenerovaná pomocou nástroja `Sphinx` [14].

Inštalácia a spustenie aplikácie

Postup inštalácie a spustenia aplikácie na lokálnom počítači je nasledovný:

1. Nainštalujte si `Python` vo verzii 3.10.
2. Nainštalujte si `pip` vo verzii 24.3.
3. (Volitelné) v koreňovom adresári projektu vytvorte virtuálne prostredie príkazom `python -m venv venv`.
4. Aktivujte virtuálne prostredie príkazom `venv\Scripts\activate` na Windowse alebo `source venv/bin/activate` na Linuxe.
5. Nainštalujte potrebné knižnice príkazom `pip install -r requirements.txt`.
6. Pomocou príkazu `cd src/sprout` sa presuňte do adresára s aplikáciou.
7. Spustite aplikáciu príkazom `reflex run`.

Kapitola 7

Záver

Cieľom tejto práce bolo vyvinúť nástroj, ktorý by umožnil zjednodušene optimalizovať mestskú hromadnú dopravu. Ako cieľ optimalizácie bol vybraný časový rozpis linky MHD. Optimalizácia bola realizovaná pomocou multi-objektívneho genetického algoritmu NSGA-II.

Táto práca mala splniť následujúce body:

1. Preskúmajte problematiku vytvárania siete mestskej 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 [17] a príklad, ktorý pomohol k lepšiemu porozumeniu tejto témy je čerpaný z článku [13].

2. Pre mesto veľkosti krajského mesta Českej republiky získajte alebo odhadnite dáta, z ktorých zostavíte model miestnej mestskej hromadnej dopravy. Informácie boli získané priamo z Dopravného podniku mesta Brno, podrobnejší popis 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 vedeckého článku [16], 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 prispôsobiť rôznym situáciam.

Aplikáciu by bolo možné rozšíriť o porovnanie dvoch rozpisov, alebo exportovanie analýzy vo formáte PDF, rozšírenie nástroja o verzatilnejšie zadávanie vstupov nakoľko momentálny import údajov o zastávkach a časovom rozpise je možný len vo formáte TXT. Pri analýze a optimalizácii sa využíva simulačný model, ktorý je možné spraviť podrobnejší, aby sa viac približoval realite a dodal tak ešte kvalitnejšie výsledky. V rámci optimalizácie by sa mohli niektoré výpočty sparalelizovať, aby sa skrátila doba čakania na výsledok. Bolo by možné rozšíriť optimalizáciu o inteligentné ohodnocovanie rozpisov v poslednej generácii, aby sa eliminoval ručný výber najvhodnejšieho rozpisu používateľom.

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. 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] Kate, H.; Christie, L.; Zavoianu, C. a Mccall, J. Optimising the introduction of connected and autonomous vehicles in a public transport system using macro-level mobility simulations and evolutionary algorithms. In:. S.l.: [b.n.], 2021, s. 315–316.

[9] Katona, L. *Sprout Documentation*. 2025. Dostupné z: https://sprout.readthedocs.io/en/latest/. Citované dňa 4. 5. 2025.

[10] 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.

[11] Peringer, P. *Modelování a simulace IMS: Studijní opora.* Brno: Fakulta informačních technologií, Vysoké učení technické v Brně, 2022. Vydáno dne 2. února 2022.

[12] Pynecone, Inc.. *Reflex Documentation.* 2025. Dostupné z: `https://reflex.dev/docs/`. Citované dňa 27. 4. 2025.

[13] 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.

[14] Sphinx Developers. *Sphinx Documentation.* 2025. Dostupné z: `https://www.sphinx-doc.org/`. Citované dňa 4. 5. 2025.

[15] 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`. Zverejnené dňa 26. 11. 2024, citované dňa 27. 4. 2025.

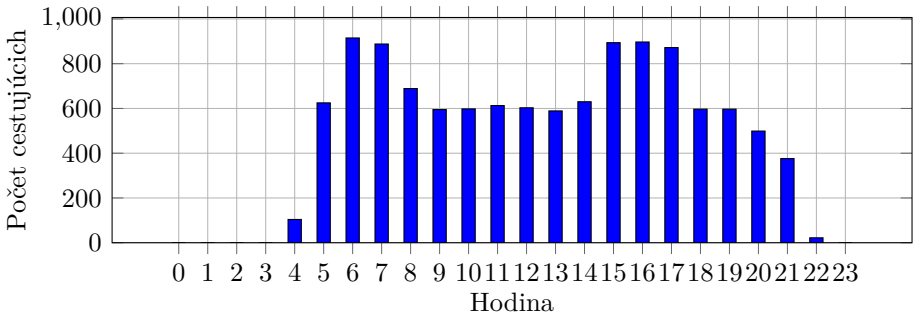
[16] Tang, J.; Yang, Y.; Hao, W.; Liu, F. a Wang, Y. A Data-Driven Timetable Optimization of Urban Bus Line Based on Multi-Objective Genetic Algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 2021, zv. 22, č. 4, s. 2417–2429.

[17] 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.

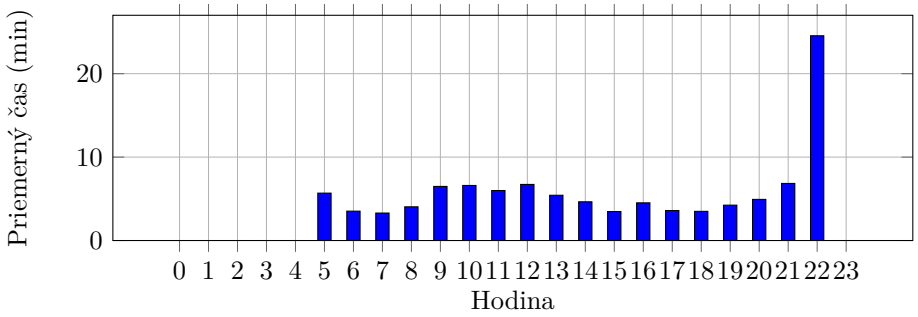
[18] Zeigler, B. P.; Praehofer, H. a Kim, T. G. *Theory of Modeling and Simulation.* San Diego: Elsevier Science, 2000. ISBN 9780127784557.

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

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

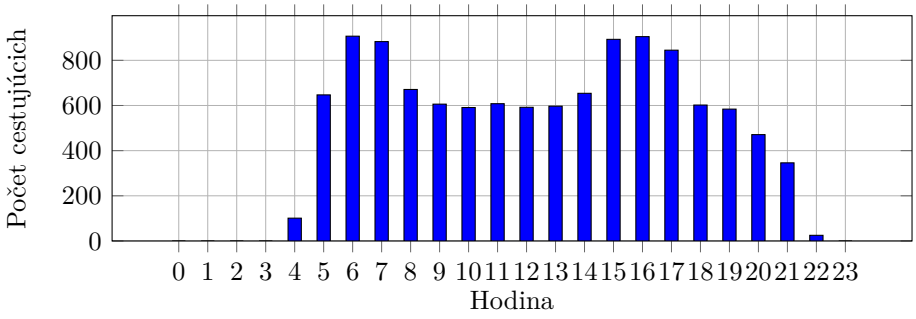
Tabuľka A.1: Rozpis zastávok

h	**Odchody**
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

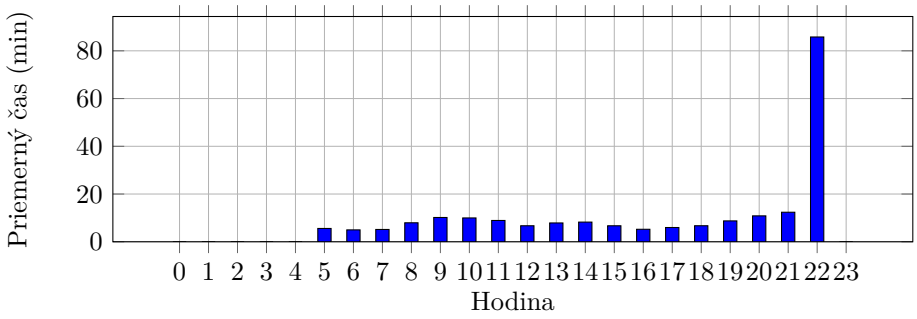
Tabuľka A.2: Časový rozpis

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

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

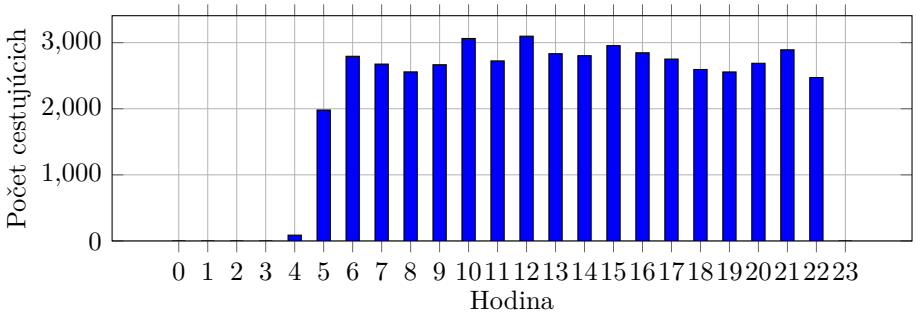
Tabuľka B.1: Rozpis zastávok

h	**Odchody**
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

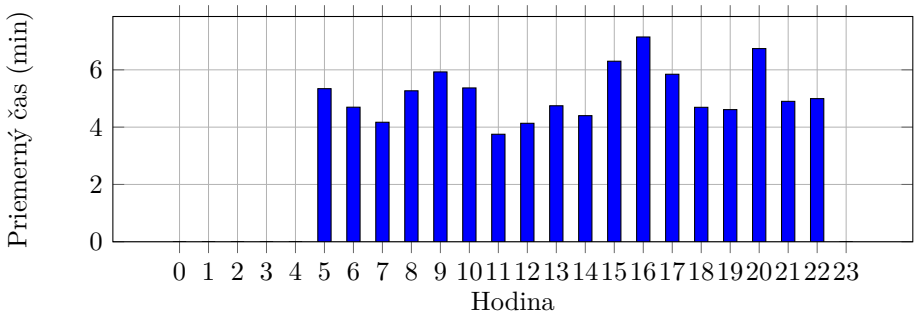
Tabuľka B.2: Časový rozpis

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

Zastávka	#
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 strojírna	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

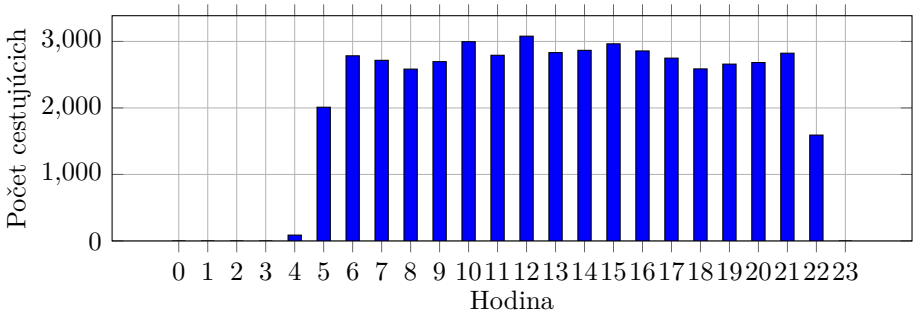
Tabuľka C.1: Rozpis zastávok

h	**Odchody**
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

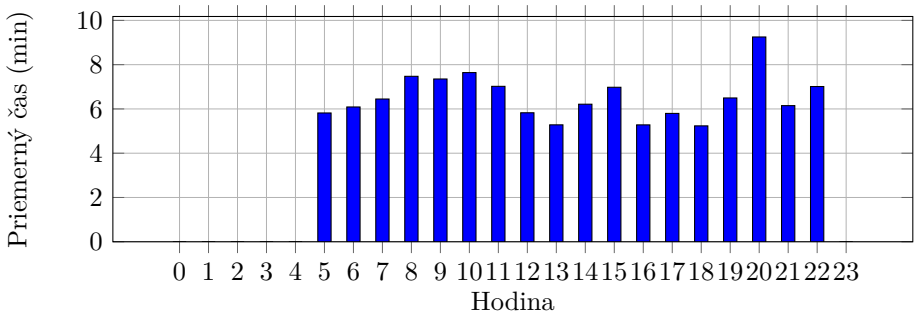
Tabuľka C.2: Časový rozpis

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

Zastávka	#
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 strojírna	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

Tabuľka D.1: Rozpis zastávok

h	**Odchody**
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

Tabuľka D.2: Časový rozpis

Príloha E

Adresárová štruktúra nástroja

```
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
```