

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Fakulta riadenia a informatiky

Memetický algoritmus na riešenie úlohy rozmiestňovania nabíjacích
staníc pre elektrické autobusy

Bakalárska práca

JURAJ DOSTÁL

Študijný program: Informatika a riadenie

Študijný odbor: Informatika

Školiace pracovisko: Žilinská univerzita v Žiline,

Vedúci bakalárskej práce: Ing. Maroš Janovec, PhD.

Žilina 2025

ZADANIE TĚMY BAKALÁRSKEJ PRÁCE.

Študijný program: Informatika a riadenie

Meno a priezvisko

Juraj Dostál

Osobné číslo

561253

Názov práce v slovenskom aj anglickom jazyku

Memetický algoritmus na riešenie úlohy rozmiestňovania nabíjajúcich staníc pre elektrické autobusy

Memetic algorithm for charging infrastructure location problem for electric buses

Zadanie úlohy, ciele, pokyny pre vypracovanie

(Ak je málo miesta, použite opačnú stranu)

Cieľ bakalárskej práce:

Cieľom práce je vytvorenie aplikácie, v ktorej bude implementovaný Memetický algoritmus pre riešenie úlohy rozmiestňovania nabíjajúcich staníc pre elektrické autobusy. Aplikácia má slúžiť na riešenie zadanej úlohy na základe vstupných dát k úlohe zadaných používateľom. Pre otestovanie metódy sa vykoná séria experimentov na niekoľkých úlohách (testovacia sada úloh bude vybraná po dohode s vedúcim práce) a porovná sa efektívnosť Memetického algoritmu s inými spôsobmi riešenia úlohy.

Obsah:

Postup práce bude nasledovný:

- vytvorenie prehľadu o Memetickom algoritme a jeho existujúcich implementáciách,
- analýza úlohy rozmiestňovania nabíjacej infraštruktúry,
- návrh Memetického algoritmu na riešenie úlohy rozmiestňovania nabíjacej infraštruktúry,
- implementácia navrhnutej metódy v ľubovoľnom programovacom jazyku,
- otestovanie algoritmu na testovacej sade úloh a porovnanie dosiahnutých riešení s ďalšími spôsobmi riešenia (heuristické/exaktné).

Meno a pracovisko vedúceho BP: Ing. Maroš Janovec, PhD., KMMOA, ŽU

Meno a pracovisko tutora BP:

garant štud. programu
(dátum a podpis)

Čestné vyhlásenie

Vyhlasujem, že som zadanú bakalársku prácu vypracoval samostatne, pod odborným vedením vedúceho práce/školiťa a používal som len literatúru uvedenú v práci.

Žilina 2. mája 2025

podpis

Pod'akovanie

Chcel by som sa srdečne poďakovať vedúcemu práce Ing. Marošovi Janovcovi, PhD. za neoceniteľné rady, odbornú pomoc a vedenie pri vypracovávaní bakalárskej práce.

Abstrakt

DOSTÁL, Juraj: Memetický algoritmus na riešenie úlohy rozmiestňovania nabíjacích staníc pre elektrické autobusy. [Bakalárska práca] – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra matematických metód a operačnej analýzy. – Školiteľ: Ing. Maroš Janovec, PhD. – Žilina: FRI UNIZA, 2025. Počet strán: 46

Táto bakalárska práca sa zaoberá návrhom a implementáciou aplikácie pre riešenie úlohy rozmiestnenia statických nabíjacích staníc a bodov pre elektrické autobusy. V tejto práci sa implementuje metaheurestika Memetické algoritmy na optimalizovanie tohto rozmiestnenia na získaných dátach poskytnutých Dopravným podnikom mesta Žilina. Cieľom je minimalizovať počet potrebných nabíjacích bodov pri zachovaní plnej obslužnosti všetkých autobusových liniek. Aplikácia bola vypracovaná v programovacom jazyku C Sharp na platforme .Net 8.0. Aplikáciu je možné obsluhovať buď cez príkazový riadok alebo grafické rozhranie vytvorené vo Windows Presentation Foundation.

Kľúčové slová: Memetické algoritmy, metaheurestika, nabíjacie stanice, elektrobusy, C Sharp

Abstract

DOSTÁL, Juraj: Memetic algorithm for charging infrastructure location problem for electric buses. [Bachelor thesis] – University of Žilina. Faculty of Management Science and Informatics; Department of mathematical methods and operations research. – Supervisor: Ing. Maroš Janovec, PhD. – Žilina: FRI UNIZA, 2025. Number of pages: 46

This bachelor thesis deals with the design and implementation of an application to solve the problem of deploying static charging stations and points for electric buses. In this thesis, Metaheuristics Memetic Algorithms are implemented to optimize this deployment on the obtained data provided by the transport company in Žilina. The goal is to minimize the number of charging points needed while maintaining full serviceability of all bus routes. The application was developed in C Sharp programming language on .Net 8.0 platform. The application can be operated either through a command line or a graphical interface created in Windows Presentation Foundation.

Keywords: Memetic algorithm, metaheuristics, charging stations, electrobus, C Sharp

Obsah

Úvod	11
1 Analýza úlohy	12
1.1 Elektromobilita vo verejnej hromadnej doprave.....	12
1.1.1 Elektrické autobusy	12
1.1.2 Pohon elektrických autobusov	12
1.1.3 Prednosti a slabiny elektrických autobusov	12
1.2 Matematický model.....	13
1.2.1 Popis množín	13
1.2.2 Popis konštánt.....	13
1.2.3 Popis rozhodovacích premenných.....	13
1.2.4 Model	14
1.2.4.1 Účelová funkcia.....	14
1.2.4.2 Podmienky	14
2 Memetické algoritmy.....	16
2.1 História	16
2.2 Koncept Memetických algoritmov	17
2.2.1 Základ Memetického algoritmu.....	17
2.2.2 Generovanie začiatkovej populácie.....	18
2.2.3 Generovanie novej populácie	18
2.2.4 Reštartovanie populácie.....	19
2.2.5 Lokálne vyhľadávanie.....	19
2.3 Operátory	20
2.3.1 Výber.....	20
2.3.2 Kríženie.....	20
2.3.3 Mutácia	21
2.3.4 Lokálne vyhľadávanie.....	21
3 Návrh	22
3.1 Popis a spracovanie dát	22
3.2 Návrh optimalizačného algoritmu.....	22
3.2.1 Reprezentácia riešenia.....	22
3.2.2 Základ algoritmu.....	23
3.2.3 Inicializácia populácie.....	24
3.2.4 Generovanie novej populácie	24
3.2.5 Lokálne vyhľadávanie.....	25
3.2.6 Selekcia	26
3.2.7 Kríženie.....	26
3.2.8 Mutácia	27
3.3 Kontrola prípustnosti riešenia	27
3.3.1 Udalostne orientovaná simulácia.....	27
3.3.2 Udalosť	29
4 Implementácia	30
4.1 Diagram balíčkov a tried	30
4.2 Balíček Base.....	31

4.2.1	Balíček Simulation.....	33
4.2.2	Balíček Algorithm	35
4.3	Používateľské rozhranie	36
4.3.1	Rozhranie príkazového riadku.....	36
4.3.2	Grafické používateľské rozhranie	37
5	Experimenty	39
5.1	Testovanie parametrov	39
5.1.1	Parameter pravdepodobnosť mutácie	39
5.1.2	Parameter ukončovacie kritérium	40
5.1.3	Parameter veľkosť populácie.....	40
5.1.4	Parameter pravdepodobnosť lokálneho vyhľadávania.....	41
5.2	Testovacie scenáre.....	42
5.3	Vyhodnotenie testovacích scenárov	43
	Záver.....	45

Zoznam obrázkov

Obrázok 1: Pseudokód Memetického algoritmu [2].....	17
Obrázok 2: Pseudokód generuj začiatočnú populáciu [2]	18
Obrázok 3: Pseudokód generuj novú populáciu [2].....	18
Obrázok 4: Pseudokód reštart populácie [2].....	19
Obrázok 5: pseudokód lokálne vyhľadávanie [2]	19
Obrázok 6: Diagram aktivít – Základ algoritmu	23
Obrázok 7: Diagram aktivít – Generuj začiatočnú populáciu.....	24
Obrázok 8: Diagram aktivít – Generuj novú populáciu.....	25
Obrázok 9: Diagram aktivít – Lokálne vyhľadávanie	26
Obrázok 10: Diagram aktivít - Simulácia.....	28
Obrázok 11: Diagram balíčkov	30
Obrázok 12: Diagram tried	31
Obrázok 13: Diagram tried – Base balík	32
Obrázok 14: Diagram tried – Simulation balík.....	33
Obrázok 15: Diagram tried – Algorithm balík	35
Obrázok 16: Rozhranie príkazového riadku.....	36
Obrázok 17: Hlavné okno	37
Obrázok 18: Okno riešenia.....	38
Obrázok 19: Vybudované nabíjacie stanice.....	44

Zoznam tabuliek

Tabuľka 1: Pravdepodobnosť mutácie.....	40
Tabuľka 2: Ukončovacie kritérium	40
Tabuľka 3: Veľkosť populácie	41
Tabuľka 4: Lokálne vyhľadávanie	41
Tabuľka 5: Testovacie scenáre	43
Tabuľka 6: Testovacie scenáre – Bez lokálneho vyhľadávania.....	43

ÚVOD

Elektromobilita sa v súčasnosti stáva jedným z najvýznamnejších trendov v oblasti dopravy, pričom jej implementácia vo verejnej hromadnej doprave prináša množstvo ekologických a ekonomických výhod. Elektrické autobusy zohrávajú kľúčovú úlohu pri znižovaní emisií skleníkových plynov, zlepšovaní kvality ovzdušia a zvyšovaní energetickej účinnosti. Napriek týmto benefítom však ich rozšírenie naráža na viaceré výzvy, ako je optimalizácia nabíjacej infraštruktúry, vysoké investičné náklady, či obmedzený dojazd vozidiel.

Táto bakalárska práca sa zaoberá problémom optimálneho rozmiestnenia nabíjacích staníc pre elektrické autobusy, pričom využíva metaheuristický prístup Memetického algoritmu na riešenie tejto úlohy. Cieľom práce je navrhnúť a implementovať aplikáciu, ktorá umožní nájsť vhodné rozmiestnenie nabíjacích staníc s minimalizáciou nákladov. Práca vychádza z reálnych dát poskytnutých Dopravným podnikom mesta Žilina.

Bakalárska práca je rozdelená do niekoľkých tematických častí. Prvá časť práce sa zaoberá vytvorením prehľadu o Memetických algoritmoch. V ďalšej časti sa zaoberáme analýzou úlohy rozmiestňovania nabíjacej infraštruktúry a definovaním úlohy pomocou matematického modelu. Tretia časť sa zaoberá návrhom implementácie optimalizačného algoritmu na danú úlohu. Predposledná časť sa venuje konkrétnej implementácii a používateľskému rozhraniu. V poslednej časti sa zaoberáme hľadaním vhodných parametrov pre Memetický algoritmus a výsledkom rozmiestnenia nabíjacích staníc.

Výsledky tejto práce môžu prispieť k efektívnemu plánovaniu nabíjacej infraštruktúry pre elektrické autobusy, čím podporujú rozvoj udržateľnej dopravy v mestskom prostredí.

1 ANALÝZA ÚLOHY

Kapitola obsahuje analýzu elektromobility vo verejnej hromadnej doprave, v ktorej je definovaný pojem elektrický autobus a ich typy pohonov. V tejto kapitole sú tiež spomenuté kladné a záporné vlastnosti elektrických autobusov. Následne sme popísali matematický model z diplomovej práce [4], ktorý bol použitý pre túto prácu. Model definuje účelovú funkciu, parametre a podmienky pre úlohu. Pri analýze úlohy sme použili dáta z mestskej hromadnej dopravy v meste Žilina. Tieto dáta obsahujú informácie o prevádzke autobusov spoločnosťou Dopravný podnik mesta Žilina.

1.1 Elektromobilita vo verejnej hromadnej doprave

Elektrické autobusy zohrávajú kľúčovú úlohu pri znižovaní emisií v doprave. Sektor dopravy tvorí značný podiel na emisiách skleníkových plynov a z tohto dôvodu je elektrifikácia dopravného sektora kľúčovým príspevkom k boju proti globálnemu otepľovaniu. Ďalšou významnou úlohou, ktorú elektrobuses zohrávajú, je efektívnosť hospodárenia s priestorom na ekologický rozvoj miest. Mestská hromadná doprava ponúka riešenia problémov, ako sú dopravné zápchy alebo nedostatok parkovacích miest, čo priamo ovplyvní aj kvalitu života v mestách. [5]

1.1.1 Elektrické autobusy

Elektrobus alebo elektrický autobus je vozidlo, ktoré je poháňané elektrickým pohonným systémom, kde zdrojom je elektrická energia. Pri elektrických vozidlách sa stretávame s inými parametrami ako pri klasických vozidlách. Výrobcovia elektrických autobusov uvádzajú priemerný dojazd ako spotrebu elektrickej energie v kWh na jeden kilometer alebo bežný dojazd vozidla v kilometroch pri plnom nabití. [4]

1.1.2 Pohon elektrických autobusov

Pri elektrických vozidlách sa môžeme stretnúť s rôznymi typmi pohonov ako sú hybridné a plug-in hybridné autobusy, trolejbusy, batériové elektrické autobusy alebo autobusy na palivový článok. Hybridné a plug-in hybridné autobusy sú poháňané elektromotorom a spaľovacím motorom v paralelnej alebo sériovej konfigurácii, kde pri plug-in sa pridáva možnosť nabitia batérie z elektrickej siete. Ostatné typy elektrobuses poháňajú iba elektromotory, kde trolejbus je napájaný z elektrickej trakcie. Naproti tomu batériový elektrobus využíva uloženú elektrickú energiu v batériách a autobusy na vodíkové palivové články si zase vyrábajú elektrinu počas ich prevádzky. [5]

1.1.3 Prednosti a slabiny elektrických autobusov

Nasadenie elektrických autobusov má svoje klady a zápory v mestskom prostredí. V ekologickej oblasti je výhodou znižovanie emisií a možné využitie obnoviteľných zdrojov na vyrábanie elektrickej energie pre elektrické autobusy. Taktiež vie prispieť k zlepšeniu kvality ovzdušia, zníženiu hlučnosti a zvýšeniu energetickej účinnosti, čo zlepšuje kvalitu života v mestách a pôsobí kladne na sociálnu oblasť. Pri elektrických autobusoch sú prevádzkové náklady nižšie v porovnaní s klasickými autobusmi so spaľovacím motorom. Tieto nízke prevádzkové náklady sú ale sprevádzané vysokými

investičnými nákladmi, čo môžeme považovať za ich slabinu. Pri elektrických autobusoch sa tiež stretávame s obavami ako je dojazd, dlhší čas nabíjania alebo životnosť batérie. S implementáciou elektrických autobusov je spojené aj riziko súvisiace s ich zavedením do prevádzky, ako aj optimalizovanie a vybudovanie nabíjacej infraštruktúry. [5]

1.2 Matematický model

Cieľom matematického modelu je nájsť optimálne rozmiestnenie nabíjacích staníc a počet nabíjacích bodov na začiatkových a konečných autobusových zastávkach tak, aby vozidlá dokázali pokryť priradené spoje počas celého týždňa. Model bol vytvorený pre obsluhu spojov vozidiel mestskej hromadnej dopravy počas celého týždňa. Tento model je vytvorený pre prevádzku autobusových spojov na celý týždeň tak, aby zohľadňoval aj zabezpečenie spojov, ktoré sa nekonajú každý deň a tiež tie, ktoré sú realizované cez víkend. Vozidlá sú nastavené na plne nabitú batériu pred začiatkom prvého spoja každého dňa. [4]

1.2.1 Popis množín

- V množina vozidiel,
- I množina začiatkových a konečných zastávok a depa, kde môže byť vybudovaná nabíjacia stanica,
- J_{vd} množina spojov vozidla $v \in V$ v deň $d \in D$,
- Jp_{vd} množina spojov vozidla $v \in V$ v deň $d \in D$, ktoré majú rozdielnu koncovú zastávku predchádzajúceho spoja a spoja nasledujúceho,
- T množina všetkých časov,
- T_{ijvd} množina časov kedy je vozidlo $v \in V$ pri spoji $j \in J_{vd}$ na zastávke $i \in I$ v deň $d \in D$.

1.2.2 Popis konštánt

- M kapacita batérie,
- E pomer dobitia energie za časovú jednotu,
- S dostatočne veľká konštanta,
- b_{jv} energia potrebná na vykonanie spoja $j \in J_{vd}$ vozidla $v \in V$

1.2.3 Popis rozhodovacích premenných

- $y_i \in \{0,1\}$ vybudovanie nabíjacej stanice v mieste $i \in I$,
- $s_i \in \mathbb{Z}^+$ počet nabíjacích bodov v mieste $i \in I$,
- $x_{ijvt} \in \{0,1\}$ či sa bude vozidlo $v \in V$ nabíjať pred spojom $j \in J_{vd}$ na zastávke $i \in I$ počas časového intervalu $t \in T_{ijvd}$ v deň $d \in D$,
- $d_{jv} \in \mathbb{Z}^+$ kapacita batérie vozidla $v \in V$ na začiatku spoja $j \in J_{vd}$,
- $z_{jv} \in \{0,1\}$ či sa bude vozidlo $v \in V$ nabíjať na koncovej zastávke spoja $j \in J_{vd}$ alebo na začiatkovej zastávke spoja $j+1 \in J_{vd}$.

1.2.4 Model

Model obsahuje účelovú funkciu a podmienky, ktoré sú popísané matematickou formou a aj slovné.

1.2.4.1 Účelová funkcia

$$\min \sum_{i \in I} s_i \quad (1)$$

- (1) Účelová funkcia minimalizuje počet vybudovaných nabíjacích bodov na nabíjacích staniciach.

1.2.4.2 Podmienky

$$s_i \leq S * y_i \quad \forall i \in I \quad (2)$$

$$d_{0v0} = M \quad \forall v \in V \quad (3)$$

$$\sum_{v \in V} \sum_{j \in J_v} x_{ijvtd} \leq s_i \quad \forall d \in D, t \in T, i \in I \quad (4)$$

$$d_{jvd} + E * \sum_{i \in I} \sum_{t \in T_{ijvd}} x_{ijvtd} \leq M \quad \forall d \in D, v \in V, j \in J_{vd} \quad (5)$$

$$d_{jvd} \leq d_{j-1vd} - b_{j-1vd} + E * \sum_{i \in I} \sum_{t \in T_{ij-1vd}} x_{ij-1vtd} \quad \forall d \in D, v \in V, j \in J_{vd} \quad (6)$$

$$d_{0vd} \leq d_{j-1vd} + E * \sum_{i \in I} \sum_{t \in T_{ivd}} x_{ivjtd} \quad \forall d \in D, v \in V, j \in J_{vd} \quad (7)$$

$$\sum_{i \in I} \sum_{t \in T_{ijvtd}} x_{ijvtd} \leq S * z_{jv} \quad \forall d \in D, v \in V, j \in J_{vd} \quad (8)$$

$$\sum_{i \in I} \sum_{t \in T_{ijvtd}} x_{ij+1vtd} \leq S * (1 - z_{jv}) \quad \forall d \in D, v \in V, j \in J_{pvd} \quad (9)$$

- (2) Podmienka zabezpečuje, že sa využívajú len tie nabíjacie body v mieste $i \in I$ kde sa vybudovala aj nabíjacia stanica.
- (3) Pre každé vozidlo $v \in V$ platí, že v prvý deň pred začiatkom prvého spoja má plnú kapacitu batérie.
- (4) Táto podmienka nám zabezpečí, že na zastávke $i \in I$ sa môže súčasne nabíjať len toľko vozidiel, koľko je vybudovaných nabíjacích bodov.

- (5) Kapacita batérie sa nesmie prekročiť.
- (6) Podmienka modeluje nabíjanie a vybíjanie batérie počas obsluhy spojov.
- (7) Podmienka modeluje nabíjanie batérie v depe pri prechádzaní na nový deň. Index j predstavuje číslo posledného spoja z predchádzajúceho dňa.
- (8)-(9) Podmienka (8) a (9) zabezpečuje možnosť nabíjania buď na konečnej zastávke posledného spoja, alebo na začiatkovej zastávke nasledujúceho spoja.

2 MEMETICKÉ ALGORITMY

Memetické algoritmy získavajú čoraz väčšiu pozornosť v komunite oproti všeobecným návodom na riešenie problémov, keďže boli vytvorené ako široká trieda algoritmov inšpirovaná šírením myšlienok a zložením z viacerých existujúcich operátorov miesto špecifických optimalizačných algoritmov. V posledných rokoch boli Memetické algoritmy úspešne aplikované na riešenie NP ťažkých úloh a komplexných reálnych problémov, kde vo veľa prípadoch vykazovali vysoký výkon. [1]

Memetické algoritmy sú techniky na optimalizovanie založené na kombinácii nápadov z rôznych algoritmických riešení, ako je v evolučných technikách vyhľadávanie na základe populácie a lokálne vyhľadávanie pri Gradientovej metóde. [1] V odbornej literatúre sú Memetické algoritmy definované ako populačné metaheuristiky so základom evolučného charakteru a lokálnych vyhľadávacích algoritmov, ktoré sa vykonávajú vo vonkajšom rámci generačného cyklu. [2]

2.1 História

Na konci osemdesiatych rokov minulého storočia Moscato a Noman definovali Memetické algoritmy, pričom sa inšpirovali interpretáciou ľudskej kultúrnej informácie od filozofa Richarda Davkinsa. [1]

Podľa jeho filozofickej teórie možno ľudskú kultúru rozložiť na jednoduché jednotky, nazývané mémy. Tento mém je základom poznania, ktorý môžeme v ľudských mozgoch duplikovať, modifikovať a kombinovať navzájom za účelom vytvorenia nového mému. V niektorých prípadoch vzniknú mémy, ktoré nie sú prirodzene zaujímavé pre ľudské spoločenstvo a v krátkom časovom úseku zanikajú. Naproti tomu môžu vzniknúť mémy, ktoré sú silné a vedia sa rozšíriť do celého spoločenstva. Tieto mémy prechádzajú zmenami alebo vzájomnou kombináciou, čo je dôsledkom nových mémov, ktoré majú silnejšie vlastnosti ako odolnosť a sú náchylnejšie na šírenie. [1]

Príkladom pre toto tvrdenie môže byť šírenie klebiet medzi ľuďmi. Podľa toho ako sú klebety zaujímavé, tak dlho pretrvávajú v čase a dostanú sa ku jednotlivcom spoločenstva. Aby sa tieto klebety stali zaujímavejšie, tak podliehajú úpravám takzvaným mutáciám, ktoré sú trvácnejšie a schopnejšie šírenia. Dôvodom uvedenia tohto príkladu je uvedenie si rozdielu medzi prenosom mémov a biologickým náprotivkom, t. j. génom. Gény sa počas života človeka nemenia a prenášajú sa tak, ako sú zdedené. Oproti génom sú mémy oveľa pružnejšie a do určitej miery sa držia Lamarckian-ovho modelu evolúcie, čo vplýva na rýchlejšiu adaptáciu oproti biologickým génom. [1]

Memetické algoritmy sa inšpirovali a definovali na základe tejto filozofickej teórie o ľudskej kultúre, ktoré modifikovali genetické algoritmy s využitím operátora lokálne vyhľadávanie na riešenie problému obchodného cestujúceho. Hybridné algoritmy sa v oblasti optimalizácie už používali, ale na nový vizionársky pohľad sa komunita zo začiatku pozerala skepticky. Po desiatich rokoch od definície Memetických algoritmov došlo k ich masívnemu rozšíreniu vo vedeckých prácach. Tento algoritmus sa aplikoval v oblastiach ako strojové učenie, kombinatorická optimalizácia, plánovanie, rozvrhovanie a časový harmonogram. Pričom hlavným dôvodom bolo rozšírenie teórie No Free Lunch

Theorem. Táto teória dokazuje, že neexistuje univerzálny optimalizačný algoritmus, ktorý by sme vedeli použiť na viaceré optimalizačné úlohy. Ak algoritmus dosahuje dobrý výkon na určitej triede problémov, tak potom musí nevyhnutne platiť horším výkonom pri ostatných zostávajúcich množinách problémov. Výskumníci v oblasti optimalizácie sa snažili navrhnuť algoritmy, ktoré majú lepší výkon ako ostatné algoritmy existujúce už desiatky rokov. Po rozšírení teórie No Free Lunch Theorem museli výskumníci dramaticky zmeniť pohľad na tuto tému. Aby sa vytvoril vhodný algoritmus na optimalizačné úlohy, bolo treba pochopiť vzťah medzi algoritmom a riešeným problémom. Tým pádom sa problém stal východiskom na vytvorenie adekvátneho algoritmu, čo znamená, že konkrétny optimalizačný algoritmus musí riešiť konkrétne vlastnosti problému. [1]

2.2 Koncept Memetických algoritmov

Na riešenie konkrétneho problému neexistuje konkrétny Memetický algoritmus, ale dostávame šablónu z knihy [2] ako všeobecne implementovať tento algoritmus na riešenie optimalizačného problému. Pri riešení konkrétneho optimalizačného problému je potrebné prispôbiť algoritmus. V tejto kapitole sú spomenuté pojmy ako gén a fitness funkcia. Gén u jednotlivca predstavuje jednu časť z jednotlivca. Fitness funkcia hodnotí kvalitu riešenia. Nasledujúce podkapitoly prezentujú šablónu Memetického algoritmu. [2]

2.2.1 Základ Memetického algoritmu

Na nasledujúcom pseudokóde môžeme vidieť základ pre Memetický algoritmus, kde na začiatku sa vygeneruje začiatočná populácia pre náš algoritmus a za vygenerovanie populácie je zodpovedná metóda GenerujZačiatočnúPopuláciu. Potom nasleduje cyklus,

```
MemetickýAlgoritmus() {  
    populácia = GenerujZačiatočnúPopuláciu()  
    Opakuj {  
        populáciaNová = GenerujNovúPopuláciu(populácia)  
        populácia = AktualizujPopuláciu(populácia, populáciaNová)  
        Ak (populácia skonvergovala) potom {  
            populácia = ReštartPopulácie(populácia)  
        }  
    } pokiaľ (PodmienkaUkončenia)  
}
```

Obrázok 1: Pseudokód Memetického algoritmu [2]

kde sa vytvára nová populácia, aktualizuje sa a zisťuje, či neskonvergovala. Za generovanie novej populácie je zodpovedná metóda GenerujNovúPopuláciu s parametrom populácia. Po vygenerovaní novej populácie nasleduje metóda AktualizujPopuláciu, ktorá sa používa na rekonštrukciu aktuálnej populácie pomocou starej a novo vytvorenej. Nakoniec sa rozhoduje o tom, či populácia degradovala, čo určujeme informačnou diverzitou ako napríklad Shannon entropy. Hneď ako sa populácia považuje za degenerovanú, tak nasleduje postup reštartovania súčasnej populácie. Podmienku pre ukončenie tohto cyklu, môžeme definovať ako stanovenie

limitu na celkový počet iterácií, dosiahnutie maximálneho počtu iterácií bez zlepšenia alebo vykonanie určitého počtu reštartov populácie. [2]

2.2.2 Generovanie začiatocnej populácie

Metóda GenerujZačiatočnúPopuláciu je zodpovedná za vytvorenie počiatocnej sady. Môže vzniknúť jednoduchým náhodným generovaním alebo použitím sofistikovaného mechanizmu násady, vďaka čomu sa do počiatocnej populácie dostanú kvalitné

```
GenerujZačiatočnúPopuláciu()
{
    populácia = PrázdnaPopulácia()
    Opakuj pre j od 1 do VeľkosťPopulácie {
        jednotlivec = GenerujNáhodnéhoJednotlivca()
        jednotlivec = LokálneVyhľadávanie(jednotlivec)
    }
    vrát populáciu;
}
```

Obrázok 2: Pseudokód generuj začiatočnú populáciu [2]

konfigurácie. Ďalšou možnosťou je použiť lokálne vyhľadávanie, ktoré je definované neskôr v texte. Postup tejto metódy je zobrazený v pseudokóde na obrázku 2. [2]

2.2.3 Generovanie novej populácie

Na obrázku 3 vidíme pseudokód algoritmu GenerujNovúPopuláciu, ktorá je kľúčovou časťou Memetických algoritmov. Jeho cieľom je vytvoriť novú populáciu riešení na základe existujúcej populácie. Tento algoritmus je založený na postupnom aplikovaní

```
GenerujNovúPopuláciu(populácia)
{
    zásobník[0] = populácia
    Opakuj pre j od 1 do PočetOperátorov {
        zásobník[j] = PrázdnaPopulácia()
    }
    Opakuj pre j od 1 do PočetOperátorov {
        rodičia = VyberZoZásobníka(zásobník[j-1], VeľkosťPopulácie)
        potomkovia = AplikujOperátor(operátor[j], rodičia)
        Opakuj pre z od 1 do VeľkosťPopulácie {
            potomkovia[z] vlož do zásobník[j]
        }
    }
    vrát zásobník[PočetOperátorov]
}
```

Obrázok 3: Pseudokód generuj novú populáciu [2]

operátorov. Pri genetickom algoritme je počet operátorov tri a to selekcia, rekombinácia a mutácia. Pri Memetickom algoritme sa k uvedeným operátorom pridalo lokálne vyhľadávanie. [2]

2.2.4 Reštartovanie populácie

V algoritme si treba dávať pozor na predčasnú konvergenciu k suboptimálnym riešeniam, čo v populácii znamená, že sú navzájom veľmi podobné, čo bráni prieskumu iných možností. Ak sa populácia považuje za degenerovanú, tak sa zavolá metóda na reštartovanie populácie (viď Obrázok 4). Následne sa uchová nejaká časť populácie a zvyšok sa vygeneruje znova ako pri metóde GenerujZačiatočnúPopuláciu. Tento

```
ReštartPopulácie(populácia)
{
    populáciaNová = PrázdnaPopulácia()
    početZachovať = VeľkosťPopulácie * PercentoZachovať
    Opakuj pre j od 1 do početZachovať {
        jednotlivec = VyberNajlepšiehoZPopulácie(populácia)
        jednotlivec vlož do populáciaNová
    }
    Opakuj pre j od početZachovať + 1 do VeľkosťPopulácie {
        jednotlivec = GenerujNáhodnéhoJednotlivca()
        jednotlivec = LokálneVyhľadávanie(jednotlivec)
        jednotlivec vlož do populáciaNová
    }
    vráť populáciaNová
}
```

Obrázok 4: Pseudokód reštart populácie [2]

postup je známy ako stratégia náhodný prisťahovalec (random-immigrant strategy). Ďalšou možnosťou je použitie operátora silnej alebo ťažkej mutácie s úlohou zmeniť oblasť populácie z aktuálnej. Pri mutácii si treba zvoliť kompromis medzi príliš silnou mutáciou, ktorá zničí všetky informácie obsiahnuté v populácii alebo takou, ktorá spôsobí opätovné konvergovanie populácie po niekoľkých iteráciách. Pri použití stratégie náhodným prisťahovalcom je nutné dbať na opatrnosť, aby sa predišlo tomu, že pôvodná populácia znova ovládne riešenie. [2]

2.2.5 Lokálne vyhľadávanie

Algoritmus lokálneho vyhľadávania (Local Search) sa vyznačuje tým, že v danom čase udržiava iba jednu konfiguráciu, ako môžeme vidieť aj na nasledujúcom obrázku 5. Lokálne vyhľadávanie začína určitou aktuálnou konfiguráciou, pričom je táto konfigurácia

```
LokálneVyhľadávanie(aktuálny)
{
    Opakuj {
        nový = GenerujSuseda(aktuálny)
        Ak (nový je lepší ako aktuálny) potom {
            aktuálny = nový
        }
    } pokiaľ (PodmienkaUkončenia)
    vráť aktuálny
}
```

Obrázok 5: pseudokód lokálne vyhľadávanie [2]

vytvorená náhodne alebo nejakým iným algoritmom. Následne sa hľadá v iteratívnom prístupe lepšia konfigurácia než je aktuálna. Ak sa taká nájde, tak sa lepšia zmení na aktuálnu a pokračuje sa v hľadaní ďalej. Tento algoritmus končí podľa zvoleného ukončovacieho kritéria. Takýmto pravidlom môže byť vopred určený počet iterácií od posledného zlepšenia alebo mechanizmus na odhad pravdepodobnosti, či sa nachádza v lokálnom optime. Takýto prístup sa metaforicky volá aj „Lezenie do kopca“, práve kvôli týmto vlastnostiam. Spôsob na hľadanie lepšej konfigurácie závisí od špecifickosti úlohy a zvolenej reprezentácie. Preto neexistuje presný návrh tohto algoritmu pre hľadanie lepšej konfigurácie. [2]

2.3 Operátory

Pri šablóne pre Memetický algoritmus sa stretávame s názvom operátor. Pod pojmom operátor si môžeme predstaviť metódu, ktorá pracuje s aktuálnym riešením. Memetické algoritmy kombinujú populačný prístup genetických algoritmov s lokálnym vyhľadávaním. Preto využívajú operátory z genetického algoritmu ako je selekcia, kríženie, mutácia a navyše obsahujú operátor lokálneho vyhľadávania, ktorý sa aplikuje na jednotlivé riešenia v populácii s cieľom zlepšiť ich kvalitu. [1]

2.3.1 Výber

Výber alebo selekcia (Selection) je operátor, ktorý nemení riešenie priamo ale určuje, ktoré riešenia majú väčšiu šancu prispieť svojimi črtami k vytvoreniu nových riešení. Tento operátor je založený na myšlienke, že vyberá jednotlivcov podľa hodnoty založenej z fitness funkcie a jeho známa implementácia je metóda rulety (roulette-wheel method). Na pravdepodobnostné rozdelenia sa používa výber, v ktorom je pravdepodobnosť výberu úmerná jeho fitness. Cieľom selekcie je uprednostniť kvalitnejšie riešenia. [2]

Namiesto výberu metódy rulety môžeme použiť poradie (Ranking) alebo turnajový výber (Tournament Selection). Poradie je založené na zoradení riešení podľa fitness funkcií. Tento spôsob je jednoduchší a efektívnejší, ale strácajú sa niektoré informácie. Druhou alternatívou je turnajový výber, kde sa vyberie a porovná súbor riešení a vyberie sa ten najlepší. [3]

2.3.2 Kríženie

Kríženie (Crossover) je operátor, kde nahradíme niektoré gény u jedného rodiča za gény od druhého rodiča. Operátor pracuje na dvoch alebo viacerých rodičovských riešeniach s cieľom vytvoriť jedného alebo viacero nových potomkov riešení. Hlavným cieľom kríženia je preskúmať priestor riešení kombinovaním takzvaných dobrých vlastností nájdených v rôznych rodičovských riešeniach, čím sa potenciálne vytvárajú ešte lepšie riešenia. [3]

Na implementovanie tohto operátora máme viaceré možnosti, ako jednobodové kríženie, dvojbodové kríženie, nelineárne kríženie alebo kríženie úplnou náhodou, ktoré nazývame aj ako jednotné kríženie (uniform crossover) a mnoho ďalších. Pri jednobodovom krížení sa vyberie jeden bod kríženia a potomkovia vzniknú výmenou častí rodičov za týmto bodom. Pri dvojbodovom krížení to funguje podobne,

kde sa vymenia gény medzi týmito krížovými bodmi. Pri jednotnom krížení sa vytvorí maska z 0 a 1 pre každý gén samostatne, kde pri 0 sa gén dedí od prvého rodiča a pri 1 je gén od druhého rodiča. [3]

2.3.3 Mutácia

Predtým ako sa rozhodneme použiť mutáciu (mutation), sa musíme rozhodnúť, či bude použitá. Ak sa rozhodneme ju použiť, tak chceme v riešení zmeniť nejaký gén náhodne. Hlavným účelom operátora mutácia je zaviesť do populácie náhodné zmeny, čím sa predchádza predčasnej konvergencii k lokálnym optimám. Na rozdiel od kríženia, ktoré kombinuje existujúce vlastnosti, mutácia zavádza nové genetické informácie do populácie. Mutácia má do istej miery sekundárnu funkciu a to takú, že pomáha zachovať primeranú úroveň populačnej diverzity. Je to poistka, ktorá umožňuje procesu uniknúť so suboptimálnych oblastí priestoru riešenia. [3]

2.3.4 Lokálne vyhľadávanie

Pri Memetických algoritmoch sa ku operátorom pridáva aj lokálne vyhľadávanie, ktoré už bolo bližšie spomenuté v texte v kapitole 2.2.5 . Operátor lokálneho vyhľadávania je mechanizmus, ktorým hľadáme lepšie riešenie, ako je aktuálne. Hľadanie lepšieho riešenia funguje tak, že operátor má nejaké riešenie, kde upravuje toto riešenie a zisťuje, či je lepšie ako to aktuálne, z ktorého vzniklo. [2]

3 NÁVRH

V tejto kapitole o návrhu je bližšie popísaná práca s poskytnutými dátami a návrh na implementáciu Memetického algoritmu pre danú úlohu.

3.1 Popis a spracovanie dát

Pre aplikovanie optimalizačného algoritmu je potrebné mať reálne dáta, aby sme dostali reálne výsledky. Dáta z Dopravného podniku mesta Žilina spracované v tejto bakalárskej práci sú vo formáte CSV, CVR a v textových súboroch.

V šiestich súboroch sa nachádzajú dáta, ako napríklad v súbore Zastavky.csv je zoznam zastávok s identifikačným číslom, názvom a geografickou lokáciou. Vzdialenosť medzi týmito zastávkami je definovaná v súbore Useky.txt. Súbor TurnusyZoznam.csv obsahuje zoznam všetkých turnusov, ktoré sa uskutočňujú a súbor TurnusyTyzden.cvr je zoznam turnusov, ktoré sa majú vykonať v daný deň počas celého týždňa, pričom vždy je definovaný autobus, ktorý zabezpečuje priradené turnusy. Jeden autobus môže vykonávať viac turnusov počas dňa. V súbore Spoje.txt sú definované spoje, ktoré sa majú vykonať pre daný turnus autobusu počas dňa. V Poslednom súbore ZasSpoje.txt je definovaná postupnosť zastávok s časom odchodu pre spoje.

Pri načítavaní dát, ich treba správne a efektívne ukladať. Preto sa vytvorili objekty ako zastávka, úsek, elektrobús a linka. Každý autobus si pamätá svoj harmonogram liniek, ktoré má vykonať a linka si pamätá postupnosť zastávok s časom odchodu. Každému autobusu sa pridá čas s odchodom z depa a príchodom do depa pre každý deň.

3.2 Návrh optimalizačného algoritmu

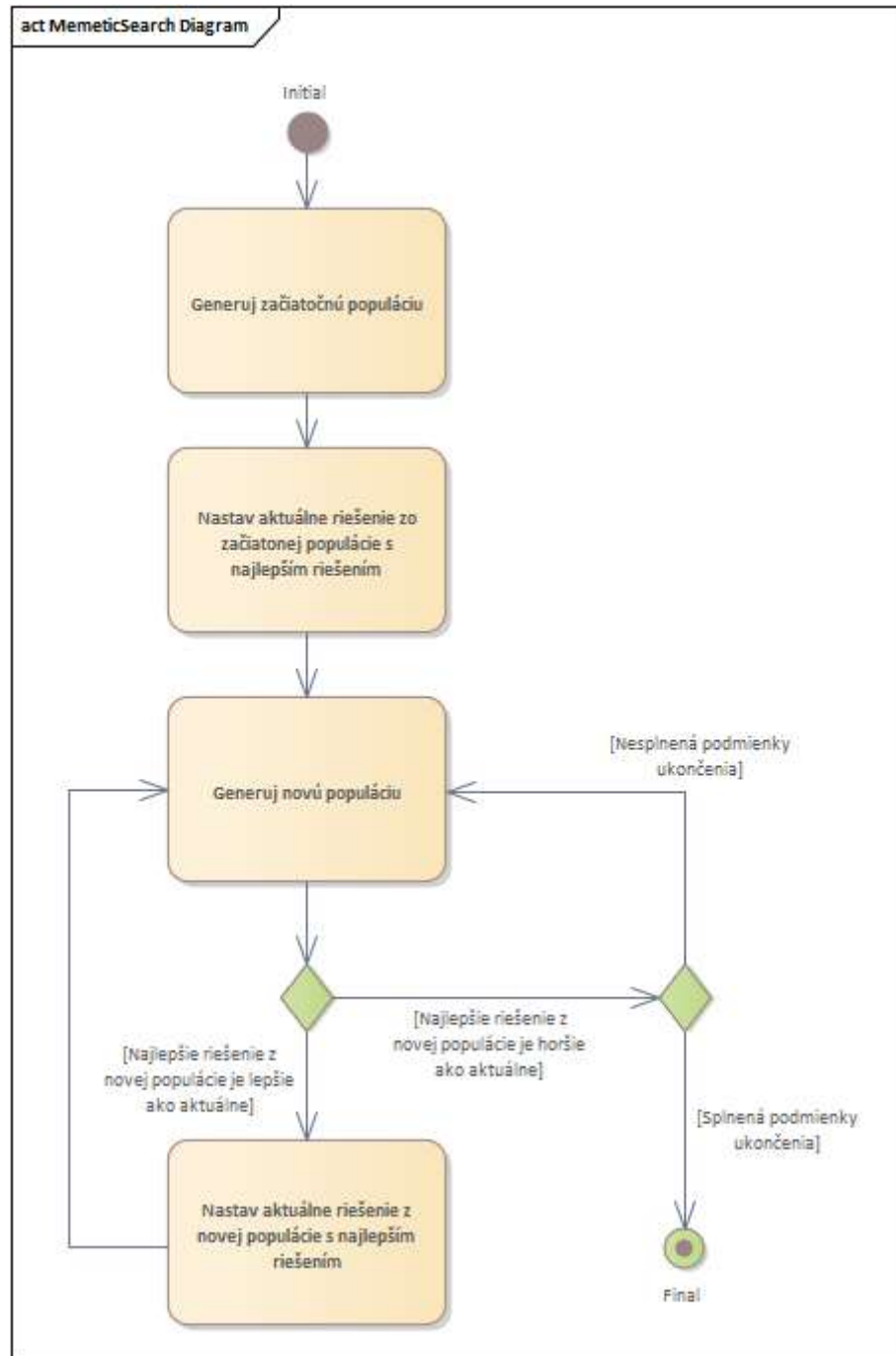
Pre hľadanie rozmiestnenia elektrických nabíjacích staníc sme zvolili Memetické algoritmy, ktoré boli popísané v kapitole 2. V tejto kapitole je opísaný návrh algoritmu pre našu lokačnú úlohu a tiež navrhnuté objekty jednotlivcov a populácia.

3.2.1 Reprezentácia riešenia

Pri našom návrhu riešenia sa stretávame s pojmami ako jednotlivci a populácia, ktoré sú inšpirované pojmami z populačných algoritmov. Populácia sa skladá z viacerých jednotlivcov a predstavuje jednu generáciu, z ktorej sa vytvára ďalšia populácia. Pojem jednotlivec, predstavuje jedno riešenie pre našu úlohu. Riešenie predstavuje vektor, ktorý obsahuje počet vybudovaných nabíjacích bodov na zastávke, kde autobus končí svoj spoj alebo začína. Pri samotnom výsledku určujeme, aké riešenie je dobré podľa fitness funkcie. Fitness funkcia sa skladá z účelovej funkcie a pokutovej funkcie. Účelová funkcia je celková cena za vybudovanie navrhnutého riešenia. Pri jednotlivcoch nám môžu vzniknúť aj nevhodné riešenia, pri ktorých autobus nie je schopný prejsť svoj priradený turnus. Preto tento druh riešenia sme sa rozhodli pokutovať, kde pokuta je násobok z ceny nabíjacej stanice.

3.2.2 Základ algoritmu

Základ algoritmu tvorí inicializácia populácie a cyklus s generovaním ďalšej populácie, pričom cyklus končí, keď sa nenájde lepší jednotlivec po definovanom počte opakovaní. Na nasledujúcom diagrame aktivít (obrázok 6) vidíme základnú architektúru algoritmu. Pri prvom vetvení sa rozhoduje, či je najlepšie riešenie z novej populácie lepšie ako

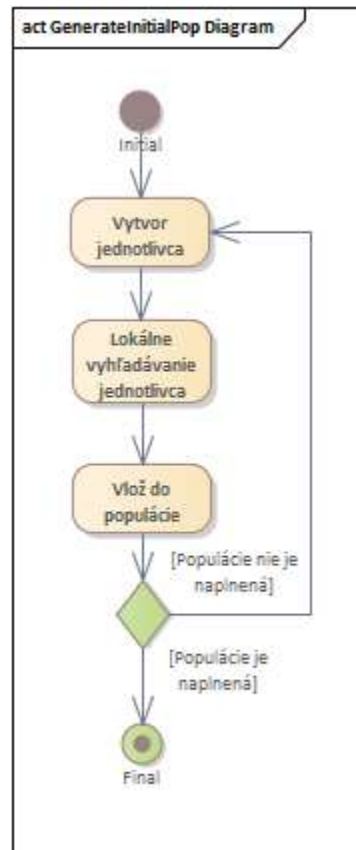


Obrázok 6: Diagram aktivít – Základ algoritmu

aktuálne pomocou fitness funkcie. Podmienkou ukončenia je vopred definovaný počet opakovaní generovania novej populácie, pričom sa nezlepšilo najlepšie nájdené riešenie. Algoritmus sa ukončí v okamihu keď je podmienka splnená.

3.2.3 Inicializácia populácie

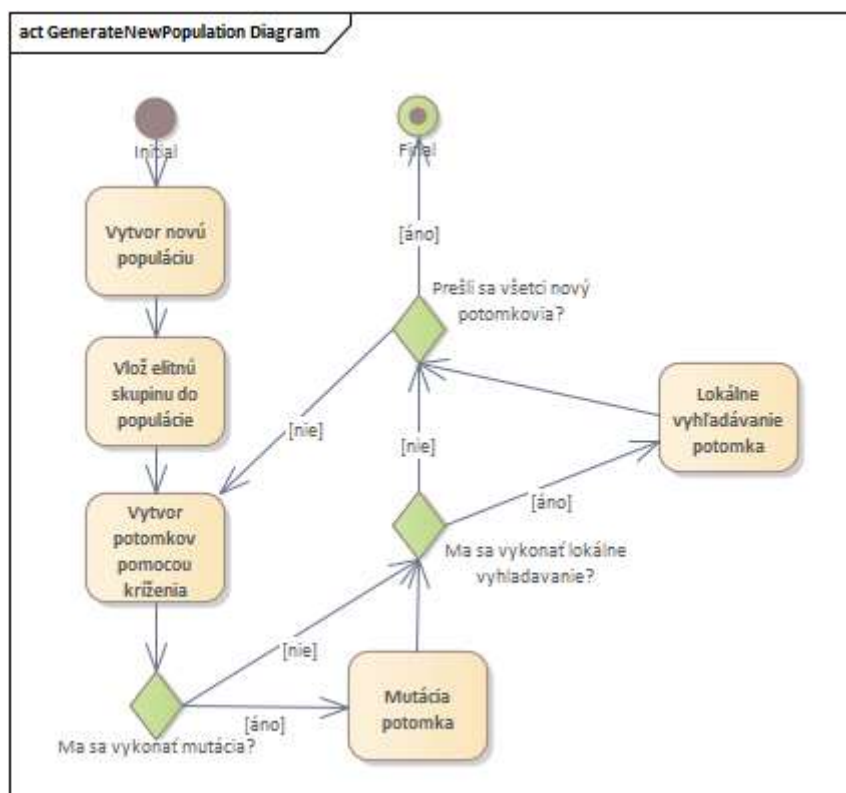
Algoritmus na vytváranie začiatkovej populácie je zobrazený na obrázku 7. V diagrame, je vidieť priebeh tohto algoritmu, kde sa vytvorí jednotlivec, potom sa spustí algoritmus lokálneho vyhľadávania nad týmto jednotlivcom. Na záver sa vloží do populácie a táto postupnosť sa opakuje pokiaľ populácia nie je naplnená.



Obrázok 7: Diagram aktivít – Generuj začiatkovú populáciu

3.2.4 Generovanie novej populácie

Nová populácia nám vznikne s pomocou aktuálnej. Algoritmus na novú populáciu vidíme na nasledujúcom diagrame. Aby sme nestratili informáciu o najlepšom riešení, tak desať percent z veľkosti populácie zachováame ako elitnú skupinu v novej populácii. Potom z aktuálnej populácie vytvoríme potomkov, inak povedané jednotlivcov, ktorí vzniknú pomocou operátorov výberu a kríženia. Potomok môže prejsť mutáciou a lokálnym

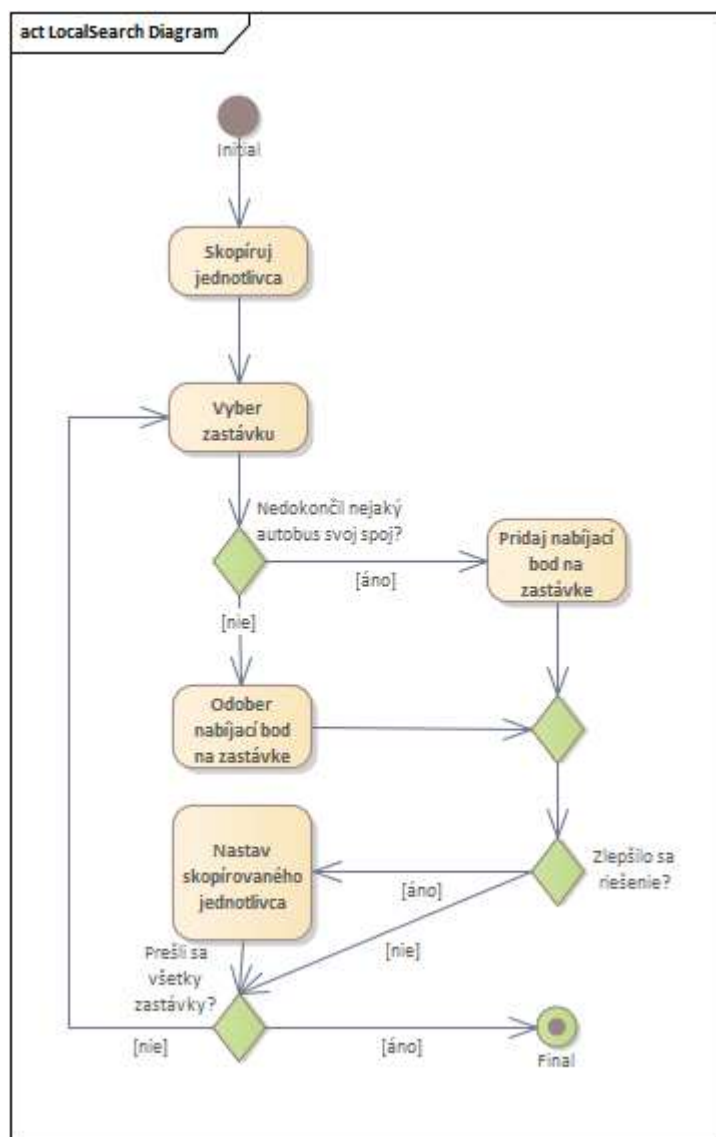


Obrázok 8: Diagram aktivít – Generuj novú populáciu

vyhľadávaním a vloží sa do populácie. Tento proces sa vykoná nad každým potomkom. O tom či sa aplikuje mutácia alebo lokálne vyhľadávanie, rozhoduje náhodný generátor.

3.2.5 Lokálne vyhľadávanie

Na obrázku 9 je zobrazený diagram aktivít pre lokálne vyhľadávanie, kde pracujeme s jednotlivcom, z ktorého si urobíme hlbokú kópiu a túto kópiu upravujeme. V riešení prechádzame autobusové zastávky, ktorým pridávame alebo ubierame nabíjacie body. Rozhodnutie o tom, či sa pridajú, je založené na tom, či autobusy dokončili svoje spoje alebo nie. Keď sa nájde lepšie riešenie jednotlivca ako je aktuálne, tak sa nastaví za aktuálneho a proces pokračuje. Ak sa prešli všetky zastávky, tak táto metóda končí.



Obrázok 9: Diagram aktivít – Lokálne vyhľadávanie

3.2.6 Selekcia

Výber riešení funguje na metóde rulety, ktorá bola spomínaná v kapitole 2.3.1. Populácia vytvorí pravdepodobnostné pole. Keďže našim cieľom je minimalizovať účelovú funkciu, preto chceme, aby riešenia s menšou hodnotou mali väčšiu pravdepodobnosť vybratia. Z tohto dôvodu používame prevrátenú hodnotu z fitness funkcie, kde následne tieto hodnoty z každého jednotlivca dávame do percentuálneho pomeru. Pomocou vygenerovania náhodnej hodnoty sa vyberú dvaja rozdielny jednotlivci, ktorí sa použijú na kríženie.

3.2.7 Kríženie

V algoritme na generovanie novej populácie sme novo vytvorenú populáciu už naplnili s elitnou skupinou, ale je potrebné ešte doplniť populáciu a to pomocou kríženia. Najprv sa získajú pomocou selekcie dvaja rodičia, ktorými vytvoríme nového potomka. Nový potomok získava vlastnosti rodičov, kde vytvorená maska určuje, ktoré vlastnosti

sa dedia od jedného rodiča a ktoré od druhého. Masky je nula jednotkový vektor, kde nula určuje dedenie od prvého rodiča a jednotka od druhého. Táto maska je pseudonáhodne generovaná generátorom na náhodné čísla.

3.2.8 Mutácia

Pri mutácii jednotlivca je vybraná jedna autobusová zastávka, ktorej nabíjací bod je pridaný alebo odobratý. Pridáva sa v momente, keď neobsahuje žiadny nabíjací bod, inak sa uberajú.

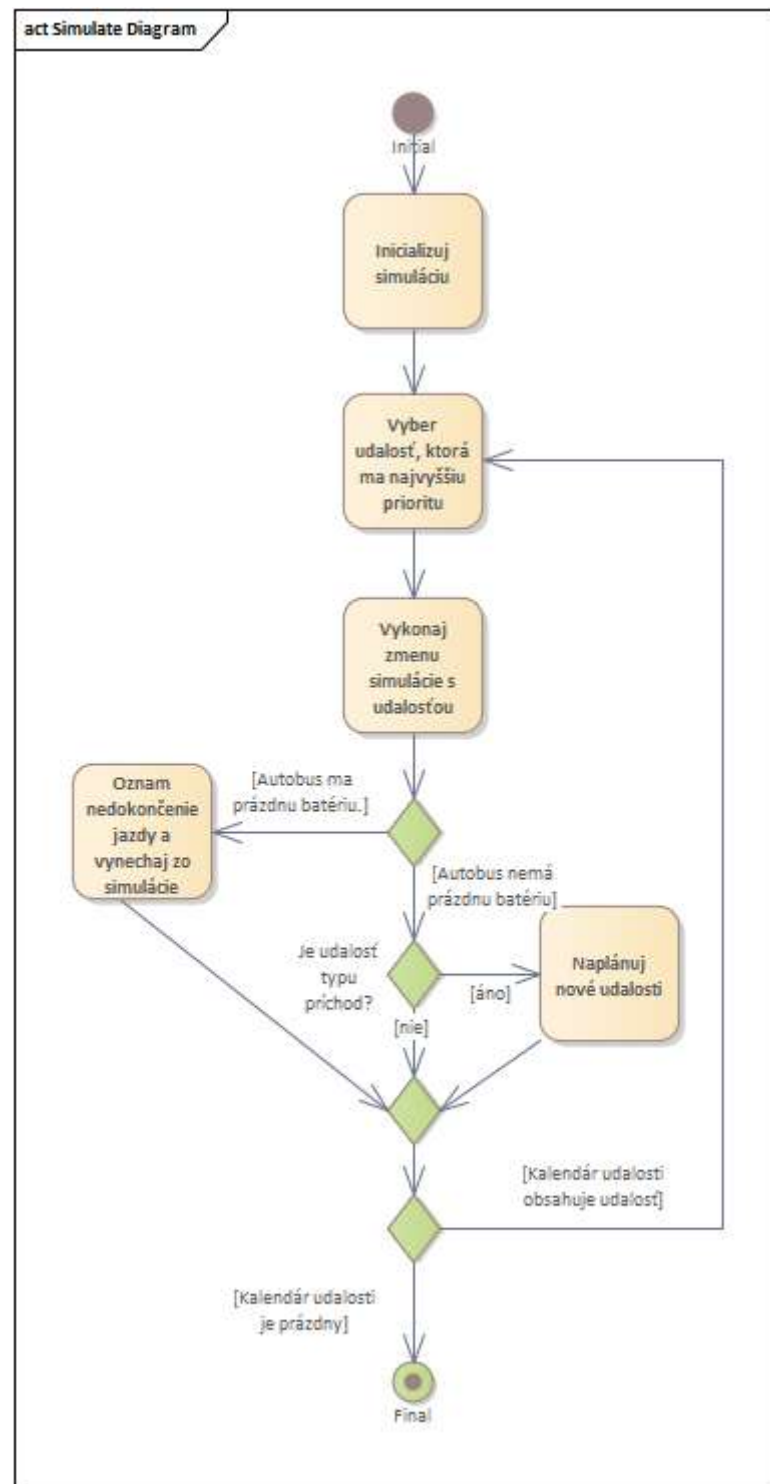
3.3 Kontrola prípustnosti riešenia

Aby sme overili správnosť nášho riešenia, potrebujeme simulovať prevádzku autobusov počas celého týždňa. Preto je potrebné navrhnuť, ako tieto výsledky overíme. Na toto overenie prípustnosti riešenia sme sa rozhodli použiť udalostne orientovanú simuláciu, ktorá sa skladá z udalostí. Pri udalostne orientovanej simulácii sa do kalendára udalostí ukladajú udalosti, ktoré sa postupne vyberajú v poradí času ukončenia a vyvolá sa zmena stavu systému.

3.3.1 Udalostne orientovaná simulácia

Udalosť je diskretná aktivita, ktorá mení stav simulácie. Výskyt udalostí sa plánuje dopredu a udržiavajú sa v kalendári udalostí, kde sa postupne spracúvajú. Kalendár udalostí sme implementovali prioritným frontom, kde priorita je čas ukončenia udalosti.

Na začiatku dňa sa pre každý autobus naplánuje prvý spoj a ak je treba, tak aj presun na zastávku. Po inicializácii simulácie sa začne cyklus, ktorý vyberá udalosti z kalendára udalostí a vyberá sa dovedty, kým nie je prázdny. Keď je tento kalendár prázdny, tak sa



Obrázok 10: Diagram aktivít - Simulácia

končí simulácia dňa. Keď sa vyberie udalosť, tak sa spustí metóda, ktorá zmení stav simulácie. Ak má elektrický autobus prázdnu batériu, tak sa ukončí jazda tohto autobusu a táto informácia sa zaznačí. Ak udalosť je typu príchod autobusu, tak sa naplánujú ďalšie udalosti. Naplánuje sa udalosť príchod pre ďalší spoj. Ak je k dispozícii nabíjací

bod, tak sa aj nabije a ak je potrebné, tak nasleduje premiestnenie autobusu z konečnej na začiatočnú stanicu nasledujúceho spoja.

3.3.2 Udalosť

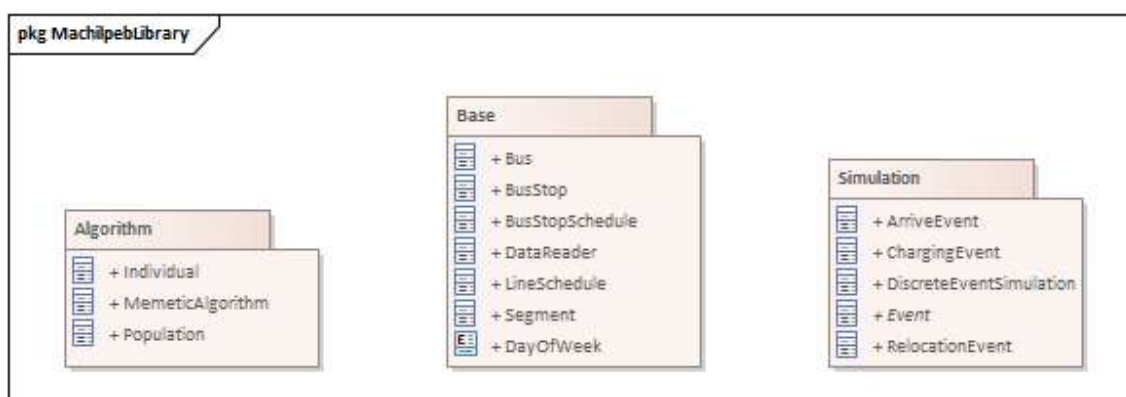
Pri tvorbe simulácie prevádzky autobusov v meste Žilina využívame tri udalosti. Prvou udalosťou je príchod, ktorá predstavuje príchod autobusu na konečnú zastávku práve obsluhovanému spoju. Ďalšia udalosť je premiestnenie, ktorá rieši presun autobusu z alebo do depa a presun na ďalší spoj. Pri týchto dvoch udalostiach sa na elektrobuse aplikuje spotrebovanie elektrickej energie v batérii. Posledná udalosť je nabíjanie, počas ktorého sa elektrický autobus nabíja. Pri udalosti nabíjanie sa doplní stav batérie podľa toho, ako dlho trvalo nabíjanie.

4 IMPLEMENTÁCIA

V tejto kapitole si popíšeme aplikáciu, ktorá rieši rozmiestnenie nabíjajúcich staníc pre elektrické autobusy s využitím Memetického algoritmu. Aplikácia bola implementovaná v programovacom jazyku C Sharp (C#), kde bola použitá platforma .Net Core verzie 8.0. Program bol vyvíjaný objektovo orientovaným prístupom v prostredí Visual Studio 2022 Professional.

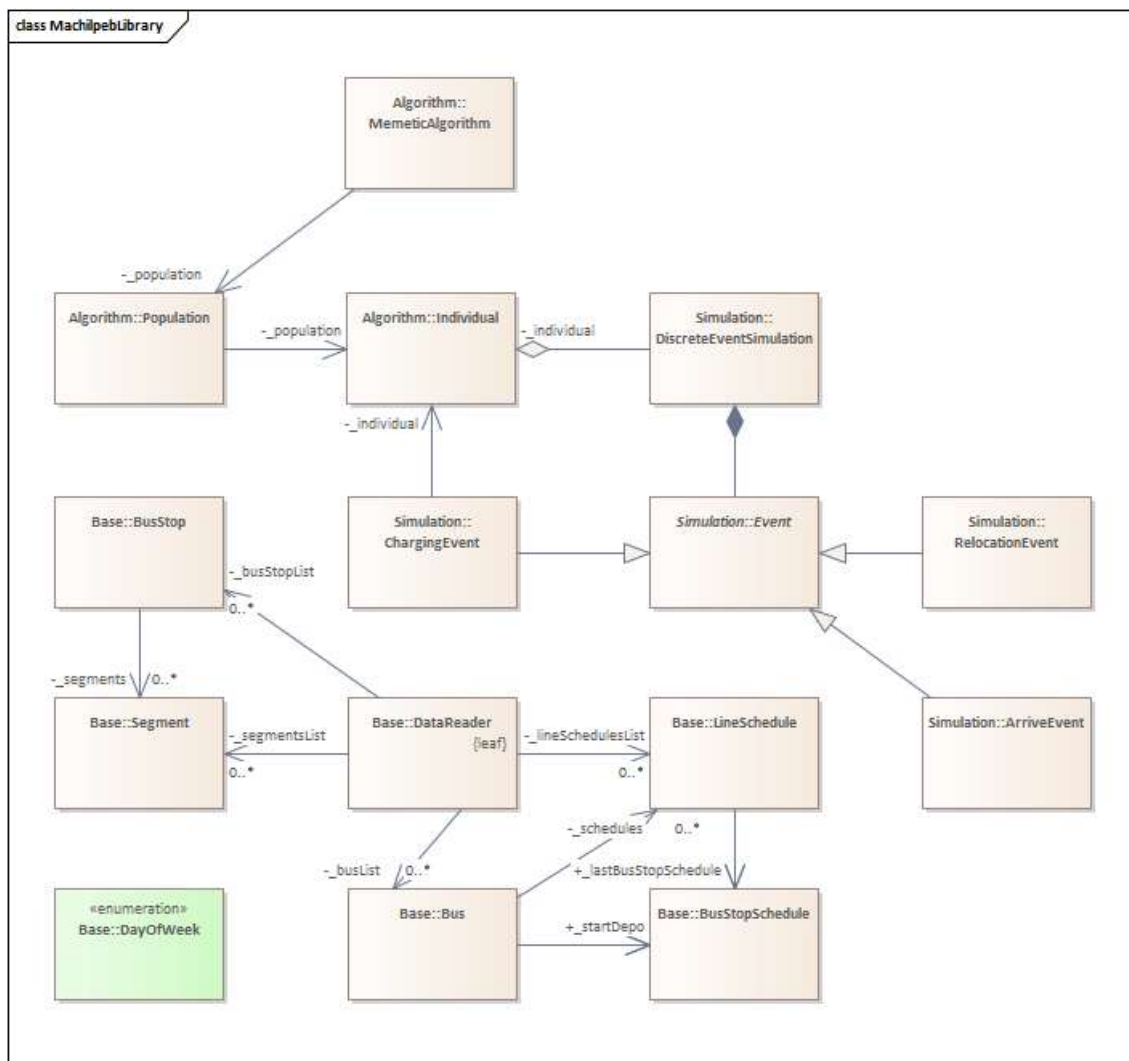
4.1 Diagram balíčkov a tried

Na nasledujúcich obrázkoch je implementácia knižnice na riešenie úlohy prostredníctvom nástroja UML. Obrázok 11 a 12 zobrazuje diagram balíčkov a diagram tried.



Obrázok 11: Diagram balíčkov

Na obrázku 11 je zobrazený diagram balíčkov, ktorý obsahuje tri balíčky Base, Algorithm a posledný Simulation. V každom balíčku vidíme triedy, ktoré patria do jednotlivých balíčkov a sú rozdelené do logických skupín. Balíček Base obsahuje základne triedy ku riešeniu našej úlohy ako autobus, zastávka a iné. Triedy, ktoré sa nachádzajú v balíčku Algorithm sa využívajú pri Memetickom algoritme. Balíček Simulation slúži pre triedy, ktoré sa používajú výlučne pri simulácii.



Obrázok 12: Diagram tried

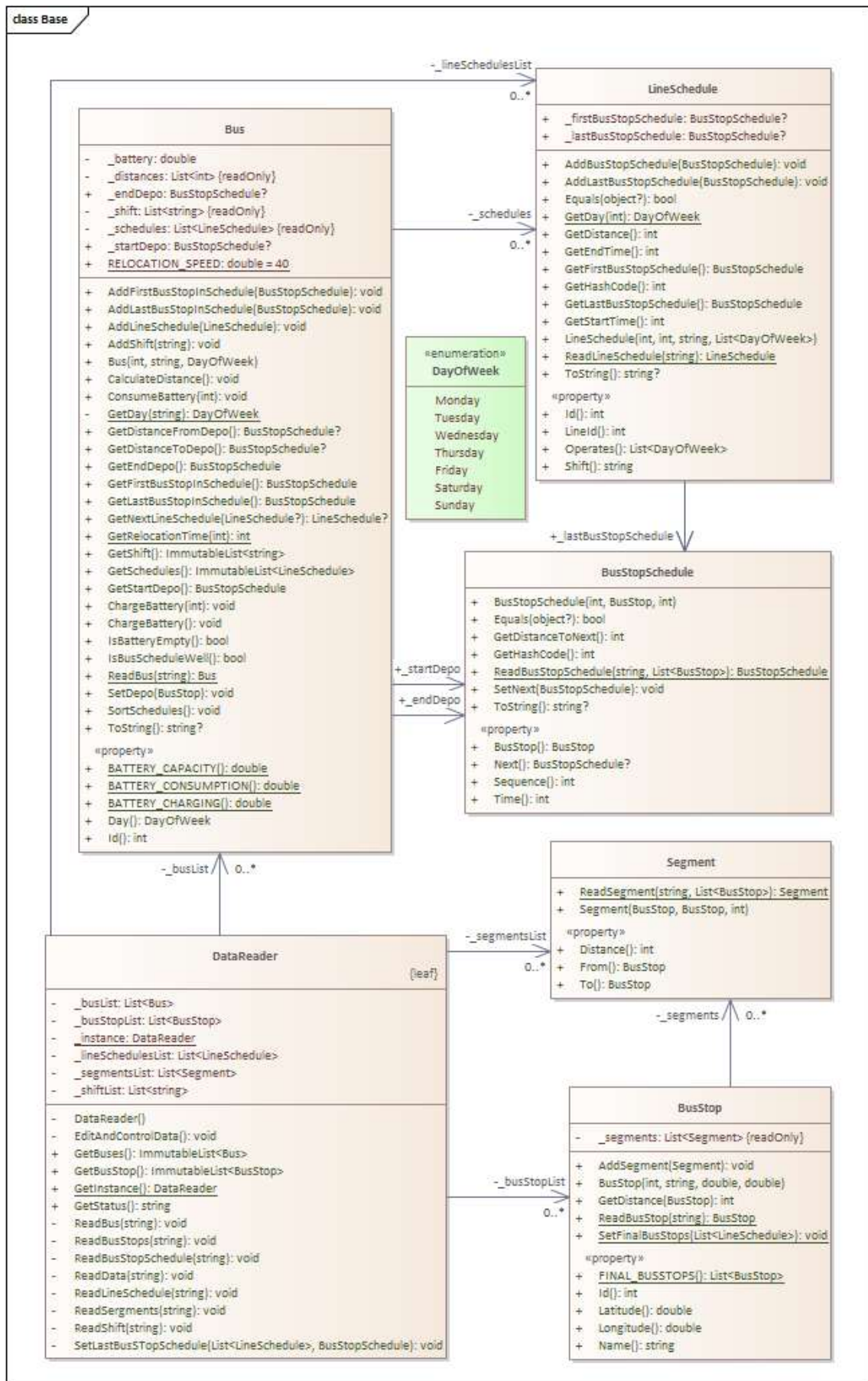
Na ďalšom obrázku vidíme diagram tried, kde sú zobrazené všetky triedy z knižnice a ich vzťahy medzi sebou.

4.2 Balíček Base

Balíček base obsahuje základne triedy na riešenie úlohy ako je autobus, zastávka, úsek a ďalšie triedy. Diagram tried balíčka Base môžeme vidieť na nasledujúcom obrázku 13.

Trieda **Segment** reprezentuje úsek cesty medzi dvoma autobusovými zastávkami. Má atribúty From, To a Distance, ktoré reprezentujú začiatočnú a koncovú zastávku a vzdialenosť medzi nimi. Trieda obsahuje metódu ReadSegment, ktorá načíta segment z textového riadku a zo zoznamu autobusových zastávok.

Trieda **BusStop** reprezentuje autobusovú zastávku. Obsahuje atribúty ako Id, Name, Latitude a Longitude, ktoré reprezentujú identifikátor zastávky, jej meno a geografickú polohu. Trieda obsahuje metódu na získavanie vzdialeností medzi zastávkami.



Obrázok 13: Diagram tried – Base balík

Sequence, BusStop, Next a Time, ktoré reprezentujú poradie zastávky, samotnú zastávku, nasledujúcu zastávku a čas odchodu zo zastávky.

Trieda **LineSchedule** reprezentuje časový harmonogram spoja. Obsahuje atribúty ako Id, LinelId, Shift a Operates, ktoré reprezentujú identifikátor harmonogramu, identifikátor linky, zmenu a dni, počas ktorých harmonogram operuje. Obsahuje tiež metódy na pridávanie harmonogramu zastávok a zistenie dĺžky spoja.

Trieda **Bus** reprezentuje autobus a obsahuje metódy a atribúty potrebné pre simuláciu, ako sú kapacita batérie, spotreba batérie a rýchlosť nabíjania. Obsahuje metódy na správu harmonogramov a výpočty vzdialeností.

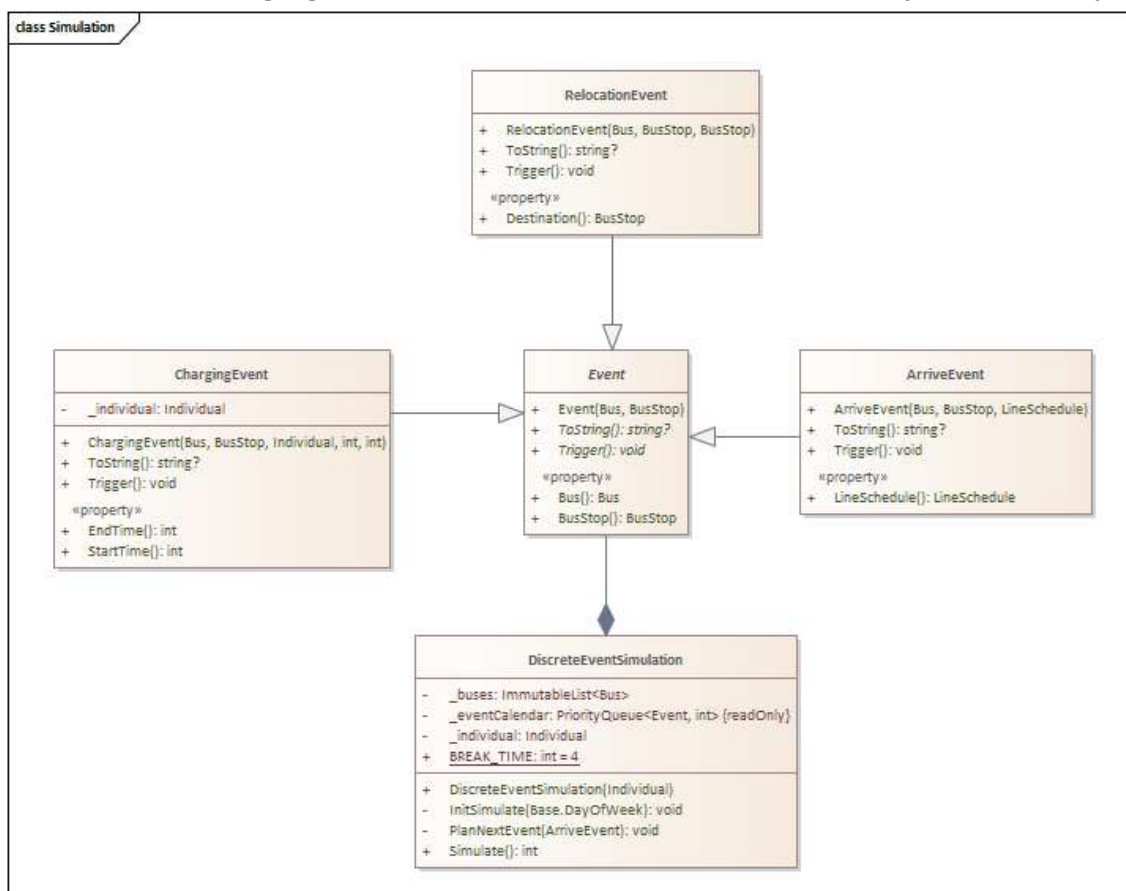
Trieda **DataReader** je zodpovedná za načítanie a správu dát potrebných pre simuláciu. Je implementovaná ako singleton a obsahuje metódy na načítanie autobusov, zastávok, segmentov a harmonogramov.

Enum trieda **DayOfWeek** reprezentuje dni v týždni. Obsahuje hodnoty pondelok, utorok, streda, štvrtok, piatok, sobota a nedeľa.

4.2.1 Balíček Simulation

Balíček Simulation slúži ako logický priestor pre triedy, ktoré sa používajú výlučne pri simulácii. Obsahuje jedného abstraktného rodiča a troch potomkov (Obrázok 14).

Abstraktná trieda **Event** reprezentuje udalosť v simulácii. Má potomkov ako **ArriveEvent**, **ChargingEvent** a **RelocationEvent**, ktorí reprezentujú špecifické typy



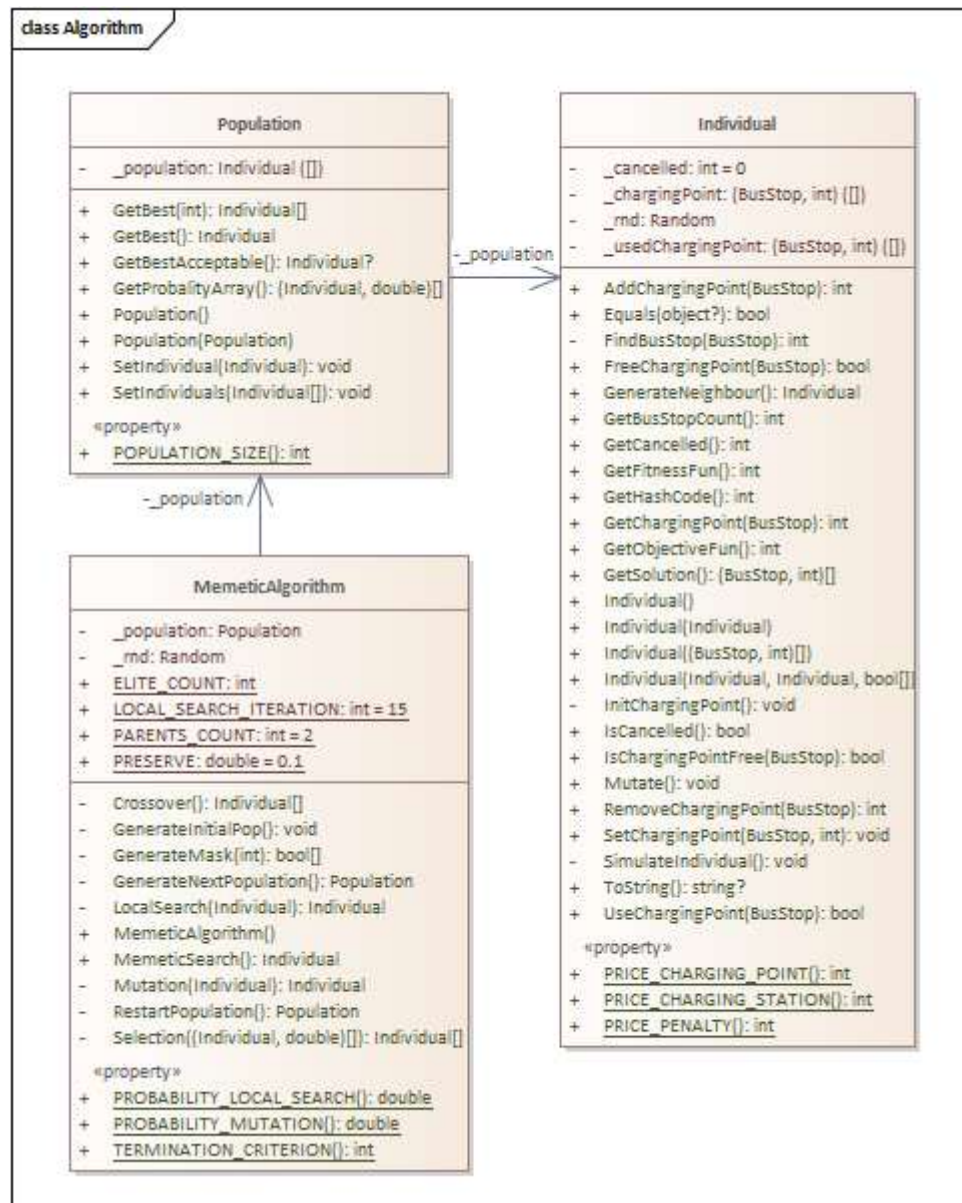
Obrázok 14: Diagram tried – Simulation balík

udalostí ako príchod autobusu na zastávku, ukončenie nabíjania a presun autobusu na ďalšiu zastávku. Metóda `Trigger` nemá implementáciu v abstraktnej triede `Event`, pretože je to abstraktná metóda, ktorú musia implementovať všetci potomkovia. Táto metóda slúži na zmenenie stavu simulácie po skončení udalosti. Triedy `ArriveEvent`, `ChargingEvent` a `RelocationEvent` implementujú metódu `Trigger` rôznymi spôsobmi v závislosti od špecifických potrieb udalosti, ktorú reprezentujú. Pri `ChargingEvent` je implementácia metódy, ktorá volá metódu `ChargeBattery` na autobuse s použitím rozdielu medzi koncovým a začiatočným časom. Vďaka tejto metóde sa pridá elektrická energia do batérie. Ďalší `Event` potomkovia implementujú metódu, ktorá sa volá `ConsumeBattery` na autobuse s použitím vzdialenosti, ktorá spotrebuje elektrickú energiu v batérii.

Trieda **`DiscreteEventSimulation`** reprezentuje diskrétnu simuláciu udalostí pre autobusy. Obsahuje metódu `Simulate`, ktorá simuluje celý týždeň prevádzky autobusových spojov a vracia počet nedokončených turnusov. Tiež obsahuje metódy na inicializáciu simulácie a plánovanie nasledujúcich udalostí.

4.2.2 Balíček Algorithm

Tento balíček obsahuje všetky triedy, ktoré sú potrebné pre náš optimalizačný algoritmus. Obsahuje triedy Population, Individual a MemeticAlgorithm, ktorý rieši celú logiku Memetického algoritmu.



Obrázok 15: Diagram tried – Algorithm balík

Trieda **Population** reprezentuje populáciu jedincov v Memetickom algoritme. Obsahuje metódy na prácu s populáciou, ako je nastavovanie jednotlivcov, získavanie pravdepodobnostného poľa a výber najlepších jedincov.

Trieda **Individual** reprezentuje jedinca v Memetickom algoritme. Obsahuje atribúty a metódy potrebné pre simuláciu jedinca a jeho hodnotenie v rámci algoritmu.

Trieda **MemeticAlgorithm** implementuje Memetický algoritmus pre optimalizáciu umiestnenia nabíjajúcich staníc pre elektrické autobusy. Obsahuje metódy

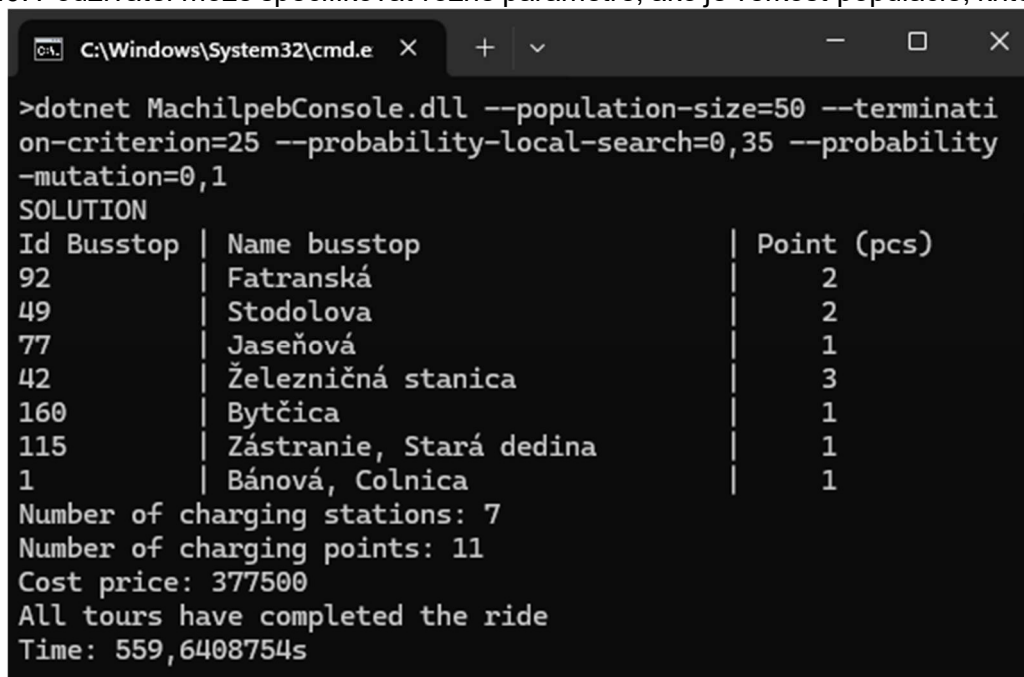
na generovanie počiatočnej populácie, generovanie novej populácie a hlavnú metódu MemeticSearch. Taktiež trieda obsahuje implementáciu operátorov selekcie, kríženia, mutácie a lokálneho vyhľadávania.

4.3 Používateľské rozhranie

Používateľské prostredie je dôležitou súčasťou aplikácie, ktorá ovplyvňuje interakciu užívateľa so systémom. Pre komunikáciu s aplikáciou boli vytvorené dve používateľské rozhrania. Prvou je komunikácia cez príkazový riadok a druhou grafické používateľské rozhranie, ktoré ponúka vizuálne intuitívnu interakciu s aplikáciou.

4.3.1 Rozhranie príkazového riadku

Konzolová aplikácia je implementovaná v MachilpebConsole súbore Program.cs. Táto aplikácia slúži na vykonávanie Memetického algoritmu pre optimalizáciu umiestnenia nabíjajúcich staníc pre elektrické autobusy za použitia rozhrania príkazového riadka. Pre spracovanie argumentov príkazového riadku bol použitý NuGet DragonFruit verzie 0.4.0. Používateľ môže špecifikovať rôzne parametre, ako je veľkosť populácie, kritérium



```
>dotnet MachilpebConsole.dll --population-size=50 --termination-criterion=25 --probability-local-search=0,35 --probability-mutation=0,1
SOLUTION
Id Busstop | Name busstop | Point (pcs)
92          | Fatranská    | 2
49          | Stodolova    | 2
77          | Jaseňová     | 1
42          | Železničná stanica | 3
160         | Bytčica      | 1
115         | Zástranie, Stará dedina | 1
1           | Bánová, Colnica | 1
Number of charging stations: 7
Number of charging points: 11
Cost price: 377500
All tours have completed the ride
Time: 559,6408754s
```

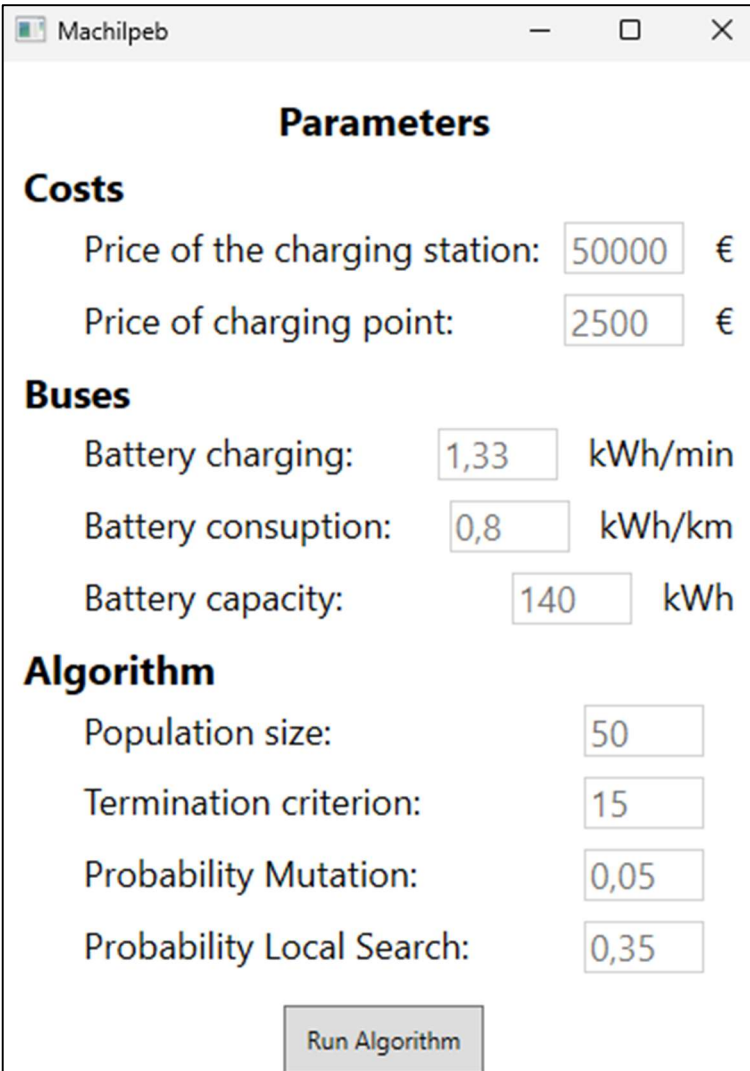
Obrázok 16: Rozhranie príkazového riadku

ukončenia, pravdepodobnosť lokálneho vyhľadávania, pravdepodobnosť mutácie, rýchlosť nabíjania batérie, spotreba batérie, kapacita batérie a náklady na vybudovanie nabíjajúcich staníc a bodov. Všetky parametre, ktoré je možné zadať sa dajú zobrazit príkazom "--help"

Po spustení algoritmu aplikácia zobrazí najlepšie riešenie, kde vypíše vybudované nabíjacie stanice s počtom vybudovaných bodov, celkový počet vybudovaných nabíjajúcich staníc a bodov, celkové náklady, počet nedokončených turnusov a čas potrebný na vykonanie algoritmu.

4.3.2 Grafické používateľské rozhranie

Grafické používateľské rozhranie je implementované pomocou Windows Presentation Foundation (WPF) a obsahuje hlavné okno aplikácie a okno riešenia.



The screenshot shows a window titled 'Machilpeb' with a standard Windows title bar (minimize, maximize, close buttons). The main content area is titled 'Parameters' and is organized into four sections: 'Costs', 'Buses', 'Algorithm', and a 'Run Algorithm' button at the bottom.

Section	Parameter	Value	Unit
Costs	Price of the charging station:	50000	€
	Price of charging point:	2500	€
Buses	Battery charging:	1,33	kWh/min
	Battery consumption:	0,8	kWh/km
	Battery capacity:	140	kWh
Algorithm	Population size:	50	
	Termination criterion:	15	
	Probability Mutation:	0,05	
	Probability Local Search:	0,35	

Run Algorithm

Obrázok 17: Hlavné okno

Hlavné okno môžeme vidieť na obrázku 17. Toto grafické užívateľské rozhranie umožňuje používateľovi zadať rôzne parametre pre algoritmus, ako sú náklady na nabíjaciu stanicu, náklady na nabíjací bod, veľkosť populácie, kritérium ukončenia, pravdepodobnosť lokálneho vyhľadávania a pravdepodobnosť mutácie. Po nastavení parametrov môže používateľ kliknúť na tlačidlo "Run Algorithm", čo spustí Memetický algoritmus a otvorí okno riešenia.

Okno riešenia (Obrázok 18) zobrazuje výsledok Memetického algoritmu pre riešenie rozmiestňovania nabíjacích staníc pre elektrické autobusy, kde sa zobrazí najlepšie riešenie s výsledkami. Zobrazí počet vybudovaných nabíjacích staníc a bodov, celkové náklady a čas potrebný na vykonanie algoritmu. Zoznam autobusových zastávok, kde sa postavili nabíjacie stanice, je zobrazený v tabuľke, ktorá obsahuje identifikátory autobusových zastávok, ich názvy a počet nabíjacích bodov.

Best Solution		
Charger Location		
Id BusStop	Name BusStop	Point (pcs)
92	Fatranská	2
49	Stodolova	2
77	Jaseňová	1
42	Železničná stanica	3
160	Bytčica	1
115	Zástranie, Stará dedina	1
1	Bánová, Colnica	1

Number of built charging stations: 7 pcs
 Number of built charging point: 11 pcs
 Total Costs: 377500 €
 Number of unfinished shift: 0 pcs
 Duration of Calculation: 378,3108503 s

Obrázok 18: Okno riešenia

5 EXPERIMENTY

Fázu experimentovania môžeme rozdeliť do dvoch častí. Prvá časť sa zaoberá hľadaním vhodnej kombinácie parametrov Memetického algoritmu a cieľom druhej časti je nájsť riešenie pre zadanú úlohu. Experimenty boli vykonávané na počítači, ktorého parametre sú nasledovné:

- Procesor: Intel® Core™ i7-4790 CPU @ 3.6 GHz
- Operačná pamäť: DDR3 16GB 3200 MHz
- Operačný systém: Ubuntu 22.04.5 LTS

5.1 Testovanie parametrov

V tejto časti práce sa zaoberáme hľadaním vhodných parametrov pre Memetický algoritmus. Optimalizačný algoritmus obsahuje niekoľko parametrov, ktoré môžeme meniť a to: veľkosť populácie, ukončovacie kritérium, pravdepodobnosť lokálneho vyhľadávania a pravdepodobnosť mutácie.

Parameter veľkosť populácie určuje, koľko jednotlivcov bude obsahovať populácia. Ďalším parametrom je ukončovacie kritérium, ktoré určuje, kedy sa algoritmus skončí a vyjadruje počet vytvorených generácií bez zlepšenia. Pri parametroch pravdepodobnosť lokálneho vyhľadávania a pravdepodobnosť mutácie nastavujeme pravdepodobnosť vykonania daného operátora pri generovaní novej populácie.

Ostatné parametre, ktoré menia vlastnosti elektrického autobusu ako kapacita batérie, spotreba energie a rýchlosť nabíjania alebo cena nabíjacej stanice a bodu sa v tejto fáze experimentovania nemenili. Tieto parametre sme nastavili nasledovne:

- Kapacita batérie: 140 (kWh)
- Spotreba energie: 0.8 (kWh/km)
- Rýchlosť nabíjania: 1.33 (kWh/min)
- Cena nabíjacej stanice: 50 000 (€)
- Cena nabíjacieho bodu: 2 500 (€)

Pre každú konfiguráciu parametrov sme vykonali desať replikácií. Hľadanie parametrov sa vykonávalo na jednom pracovnom dni. Hľadanie vhodných parametrov sme vykonávali postupom sweep search, kde jeden parameter sa mení a ostatné zostávajú rovnaké. Počas experimentov boli sledované priemerné časy výpočtu, priemerný počet nedokončených jázd a priemerná cena vybudovaných staníc a bodov pre riešenie.

5.1.1 Parameter pravdepodobnosť mutácie

Prvý testovaný parameter bol pravdepodobnosť mutácie, pri ktorej hľadáme vhodnú pravdepodobnosť. Testovali sme pravdepodobnosti mutácie v rozsahu od 0 do 0,4, ktoré sú vypísané v tabuľke 1. Ostatné parametre boli nastavené takto:

- veľkosť populácie: 30

- ukončovacie kritérium: 25
- pravdepodobnosť lokálneho vyhľadávania: 0,35

Tabuľka 1: Pravdepodobnosť mutácie

Pravdepodobnosť mutácie	Čas výpočtu		Priemerný počet nedokončených jázd	cena	
	Najlepší	priemer		najnižšia	priemer
0	12.98521	14.36991919	0.1	327500	375000
0.05	12.56541	14.59513496	0.1	327500	360750
0.1	12.3029	14.64699767	0.2	320000	355000
0.15	12.63533	13.81406059	0	327500	366250
0.2	12.82211	14.7653327	0	327500	361000
0.4	13.56648	15.31460743	0	327500	356500

Výsledky experimentov môžeme nájsť v tabuľke 1. Môžeme si všimnúť, že zmenou pravdepodobnosti mutácie sa nemení dĺžka výpočtu alebo kvalita riešenia.

5.1.2 Parameter ukončovacie kritérium

V nasledujúcom experimente sa hľadal najvhodnejší počet opakovaní od posledného zlepšenia. Zvolili sme hodnoty ako 10, 15, 25, 50, 100. Ostatné parametre sme nastavili:

- veľkosť populácie: 30
- pravdepodobnosť lokálneho vyhľadávania: 0,35
- pravdepodobnosť mutácie: 0,15

Tabuľka 2: Ukončovacie kritérium

Ukončovacie kritérium	Čas výpočtu		Priemerný počet nedokončených jázd	cena	
	najkratší	priemerný		najnižšia	priemerná
10	7.29352	8.607474	0.1	327500	351250
15	8.658947	10.45318	0.1	320000	360500
25	11.50594	13.81038	0	327500	375500
50	23.09948	25.50916	0.1	327500	360500
100	41.98306	45.84763	0.1	327500	350500

Z výsledkov (Tabuľka 2) môžeme vyčítať, že pridávaním počtu iterácií sa predlžoval priemerný čas výpočtu. Na priemernú cenu to malo iba nepatrný vplyv.

5.1.3 Parameter veľkosť populácie

Pri ďalšom experimente sme skúšali meniť veľkosť populácie od 30 až po 150. Ďalšie parametre sme nastavili nasledovne:

- ukončovacie kritérium: 25

- pravdepodobnosť lokálneho vyhľadávania: 0,35
- pravdepodobnosť mutácie: 0,15

Tabuľka 3: Veľkosť populácie

Veľkosť populácie	Čas výpočtu		Priemerný počet nedokončených jász	cena	
	najkratší	priemerný		najnižšia	priemerná
30	13.10531	14.71856	0	327500	351750
50	22.34874	23.53282	0	327500	356000
75	33.4054	35.17829	0.1	272500	327000
100	41.59216	45.86692	0.1	272500	336250
150	64.98312	67.7551	0	327500	327500

Po otestovaní vplyvu zmeny veľkosti populácie na výsledky (Tabuľka 3) môžeme vidieť, že s narastajúcou populáciou sa zvyšuje aj časová náročnosť. Pri počte nedokončených jász a cene vidíme klesajúci trend, čo nám zvyšuje šance na kvalitnejšie riešenie.

5.1.4 Parameter pravdepodobnosť lokálneho vyhľadávania

Posledným experimentom v tejto fáze bolo nájdenie pravdepodobnosti pre vykonanie lokálneho vyhľadávania. Začali sme od 0, kde vlastne nepoužívame lokálne vyhľadávanie pri novom jedincovi a až po 1, kde vždy použijeme lokálne vyhľadávanie. Ostatné parametre sme nastavili nasledovne:

- veľkosť populácie: 50
- ukončovacie kritérium: 25
- pravdepodobnosť mutácie: 0,15

Tabuľka 4: Lokálne vyhľadávanie

Pravdepodobnosť lokálneho vyhľadávania	Čas výpočtu		Priemerný počet nedokončených jász	cena	
	najkratší	priemerný		najnižšia	priemerná
0	4.747805	4.937989	0.9	322500	360000
0.2	12.93772	15.66894	0	327500	365750
0.35	21.49937	24.45082	0	327500	346750
0.5	28.48969	30.47784	0	327500	360750
0.75	41.16358	42.97036	0	327500	341750
1	51.57857	54.46624	0	327500	337000

Výsledky experimentu nájdeme v tabuľke 4. Zvyšujúcou sa pravdepodobnosťou, sa predlžuje čas výpočtu, ale zlepšujú sa riešenia. Avšak od hodnoty 0,5 už zlepšenie riešenia nevzniká.

5.2 Testovacie scenáre

Pre druhú časť experimentov bolo navrhnutých viacero scenárov, ktoré odrážajú prevádzku elektrobusu v rôznych ročných obdobiach, nakoľko nízka alebo vysoká priemerná vonkajšia teplota môže mať negatívny vplyv na výkon elektrobusu. Pre otestovanie vhodného rozmiestnenia elektrických nabíjacích staníc a zastávok v ročných obdobiach, sme nastavili parametre optimalizačného algoritmu nasledovne:

- veľkosť populácie: 50
- ukončovacie kritérium: 25
- pravdepodobnosť lokálneho vyhľadávania: 0,35
- pravdepodobnosť mutácie: 0,15

Parametre ceny vybudovania nabíjacej stanice a nabíjacieho bodu sme nastavili nasledovne:

- Cena nabíjacej stanice: 50 000 (€)
- Cena nabíjacieho bodu: 2 500 (€)

Základný scenár sa týka jarých a jesenných mesiacov, kedy nie sú výrazné vplyvy tepla, ani chladu a preto boli parametre eklektického autobusu nastavené podľa hodnoty udanej výrobcom a to nasledovne:

- Kapacita batérie: 140 (kWh)
- Spotreba energie: 0.8 (kWh/km)
- Rýchlosť nabíjania: 1.33 (kWh/min)

Zimný scenár simuluje prevádzku elektrobusu počas zimného obdobia, kedy vplyv nízkych teplôt znižuje kapacitu batérie. Okrem toho je potrebné vykurovať interiér autobusu, čo vedie k zvýšenej spotrebe energie na kilometer. Preto sa parametre nastavili nasledovne:

- Kapacita batérie: 112 (kWh)
- Spotreba energie: 1 (kWh/km)
- Rýchlosť nabíjania: 1.33 (kWh/min)

Posledný scenár sa zameriava na letné mesiace, kedy prevádzka klimatizácie elektrických autobusov zvyšuje spotrebu batérie a preto bola spotreba energie na kilometer zvýšená. Tento scenár voláme letný a jeho parametre sme nastavili nasledovne:

- Kapacita batérie: 140 (kWh)
- Spotreba energie: 1 (kWh/km)
- Rýchlosť nabíjania: 1.33 (kWh/min)

5.3 Vyhodnotenie testovacích scenárov

Pre testovacie scenáre, ktoré sme navrhli v predchádzajúcej podkapitole, sme vykonali výpočtové experimenty s cieľom nájsť vhodné riešenie. Pre porovnanie výsledkov sme sa rozhodli experimenty pustiť aj na algoritme bez použitia lokálneho vyhľadávania s rovnakými parametrami. Tento algoritmus môžeme nazvať aj Genetický algoritmus. Výsledky Memetického algoritmu vidíme v tabuľke 5 a v tabuľke 6 sú zobrazené výsledky bez použitia lokálneho vyhľadávania.

Tabuľka 5: Testovacie scenáre

Scenár	Čas výpočtu		Priemerný počet nedokončených jász	Cena	
	najkratší	priemerný		najnižšia	priemerná
základný	125.8424	128.6513744	0	327500	366250
letný	160.2452	176.3634238	4	537500	579500
zimný	154.1283	168.9444439	5	657500	784500

Tabuľka 6: Testovacie scenáre – Bez lokálneho vyhľadávania

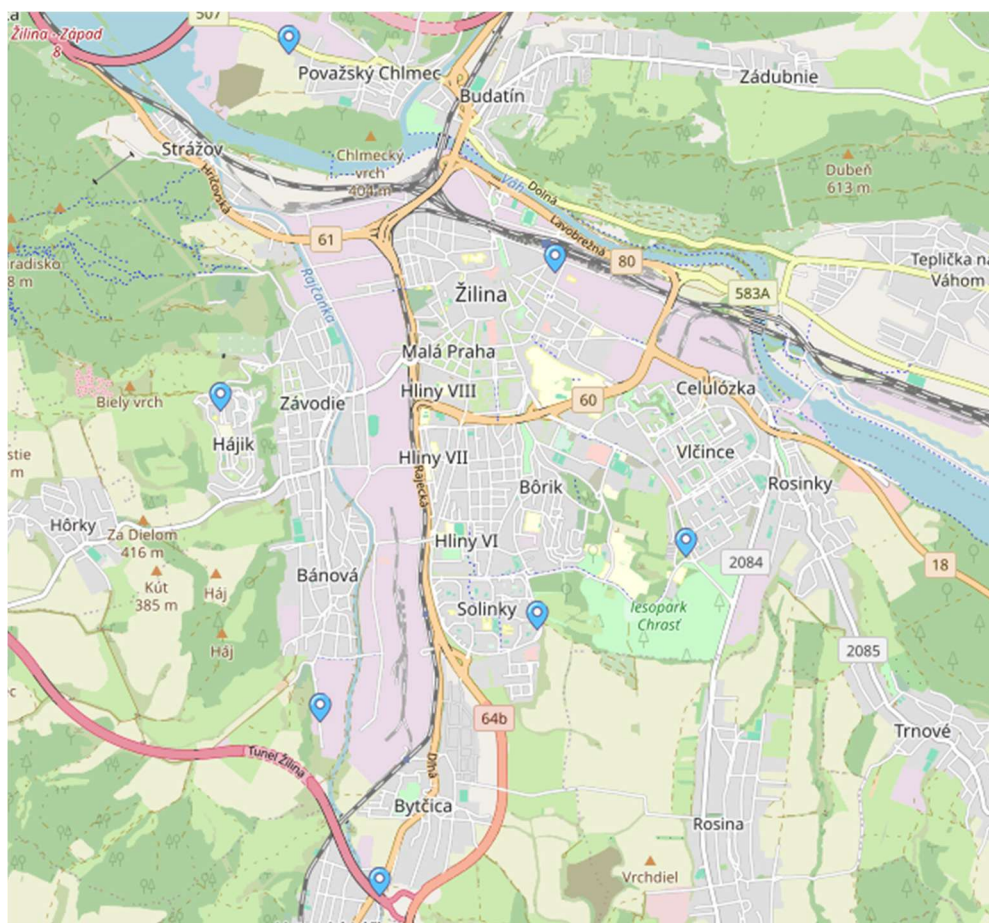
Scenár	Čas výpočtu		Priemerný počet nedokončených jász	Cena	
	najkratší	priemerný		najnižšia	priemerná
základný	7.668336	8.141362	0	485000	650250
letný	7.642503	8.141524	4	647500	947750
zimný	7.647733	8.108902	10.1	915000	1047250

Algoritmus bez lokálneho vyhľadávania je výrazne rýchlejší ako Memetický algoritmus, ale výsledky cien sú znateľne lepšie pri algoritme s lokálnym vyhľadávaním. Pri letnom a zimnom scenári si môžeme všimnúť nedokončenie jász autobusov. Tento jav vzniká z dôvodu, že optimalizačný algoritmus stavia nabíjacie stanice iba na konečné zastávky a je nedostatočná kapacita batérie na vzdialenosť a spotrebu elektrického autobusu. Na vyriešenie tohto problému by sme museli upraviť algoritmus tak, aby budoval nabíjacie stanice a body aj na zastávkach, ktoré nie sú konečné pre spoje.

Na nasledujúcom obrázku 19 vidíme vyznačené miesta, kde algoritmus navrhol vybudovanie šiestich nabíjacích staníc a jedenásť nabíjacích bodov pre základný scenár. Na obrázku je vyznačených sedem zastávok, pretože zastávky Bánova, Colnica a Bytčianska sa vo výsledku striedali, pričom obidve varianty dávali rovnako dobré riešenie. Zoznam vybudovaných nabíjacích bodov na zastávke vyzerá nasledovne:

- Fatranská: 2
- Stodolova: 3
- Jaseňová: 1
- Železničná stanica: 3
- Bytčica: 1

- Bánova, Colnica/Bytčianska: 1



Obrázok 19: Vybudované nabíjacie stanice

Nabíjacie stanice boli rozmiestnené rovnomerne po území mesta Žilina, pričom väčšina z nich sa nachádza na jeho okrajoch, s výnimkou zastávky Železničná stanica, ktorá sa nachádza v centre mesta. Takéto rozloženie pravdepodobne súvisí s tým, že ide o zastávky, na ktorých jednotlivé linky mestskej hromadnej dopravy začínajú alebo končia svoju trasu.

Zastávky Fatranská, Stodolová a Železničná stanica boli vybudované s väčším počtom nabíjajúcich bodov, čo naznačuje ich vyššie zaťaženie hromadnou dopravou. Je možné, že na týchto zastávkach dochádza k častejšiemu výskytu elektrických autobusov. Je to z dôvodu vyššej frekvencie spojov danej linky alebo sú to zastávky, kde začína a končí veľa spojov z rôznych liniek, a preto je potrebné zabezpečiť vyššiu dostupnosť nabíjajúcich bodov pre elektrické autobusy práve v týchto lokáciách.

ZÁVER

Bakalárska práca sa úspešne zaoberala návrhom a implementáciou aplikácie na optimalizáciu rozmiestnenia nabíjacích staníc pre elektrické autobusy pomocou Memetického algoritmu. Cieľom algoritmu bolo minimalizovať počet potrebných nabíjacích bodov a tým aj celkové náklady, pričom sa bral ohľad na zabezpečenie plnej obslužnosti všetkých autobusových liniek počas týždňa.

Riešenie bolo overené na reálnych dátach z Dopravného podniku mesta Žilina. Na základe výsledkov experimentov možno konštatovať, že Memetický algoritmus preukázal schopnosť nájsť efektívne riešenia pri rozumnej výpočtovej náročnosti. Výsledky ukazujú, že správna voľba parametrov algoritmu má výrazný vplyv na kvalitu riešení, čo bolo potvrdené sériou experimentov zameraných na ich ladenie.

Výsledky práce potvrdili význam využitia Memetických algoritmov pri plánovaní nabíjacej infraštruktúry pre elektrické autobusy. Táto práca predstavuje prínos nielen pre teoretický výskum v oblasti optimalizačných algoritmov, ale aj pre praktickú implementáciu udržateľných riešení vo verejnej doprave.

V budúcnosti môže byť aplikácia ďalej rozšírená o nové funkcionality, ako je integrácia parciálnych autobusov a využitie trolejového vedenia alebo implementovanie paralelizmu pre rýchlejšie výpočty aplikácie.

Zoznam použitej literatúry

- [1] **Neri, Ferrante a Cotta, Carlos.** *Memetic algorithms and Memetic Computing Optimization: A literature review.* 2, 2012, Swarm and Evolutionary Computation, s. 1-14.
- [2] **Moscato, Pablo a Cotta, Carlos.** A Modern Introduction to Memetic Algorithms. [aut. knihy] Michel Gendreau a Jean-Yves Potvin. *Handbook of Metaheuristics.* s.l. : Springer, 2010, s. 141-183.
- [3] **Reeves, Colin R.** Genetic Algorithms. [aut. knihy] Michel Gendreau a Jean-Yves Potvin. *Handbook of Metaheuristics.* s.l. : Springer, 2010, s. 109-139.
- [4] **Vasilovský, Patrik.** *Návrh nabíjacej infraštruktúry pre elekrobusy v mestskej hromadnej doprave.* Žilina : s.n., 2018. s. 53. Diplomová práca. reg. číslo: 85/2017.
- [5] **Manzoli, Jônatas Augusto, Trovão, João Pedro a Antunes, Carlos Henggeler.** *A review of electric bus vehicles research topics – Methods and trends.* 2022, Renewable and Sustainable Energy Review, Zv. 159

Prílohy

Zoznam príloh

Príloha A | DVD49

Príloha A | DVD

Priložená DVD obsahuje:

- zdrojové kódy implementovanej aplikácie,
- skompilovanú aplikáciu,
- vstupné dáta,
- výsledky vykonaných experimentov,
- bakalársku prácu v elektronickej podobe.