ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

METÓDY PRE RIEŠENIE ÚLOHY NÁVRHU TURNUSOV PRE ELEKTROBUSY VO VEREJNEJ DOPRAVE

Dizertačná práca

28360020213006

Študijný program: Inteligentné informačné systémy
Študijný odbor: Informatika
Pracovisko: Katedra matematických metód a operačnej analýzy
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline
Školiteľ: doc. Ing Michal Koháni, PhD.

Žilina, 2021 **Ing. Maroš Janovec**

Anotácia

Typ práce:	Dizertačná práca
Akademický rok:	2020/2021
Názov práce:	Metódy pre riešenie úlohy návrhu turnusov elektrobusov vo verejnej doprave
Autor:	Ing. Maroš Janovec
Školiteľ:	doc. Ing. Michal Kohániˇ, PhD.
Jazyk:	Slovenčina
Počet strán:	130 s.
Počet obrázkov:	15
Počet tabuliek:	36
Počet referencií:	74
Kľúčové slová:	návrh turnusov, elektrické autobusy, matematické modelovanie, exaktný prístup, generovanie stĺpcov, metaheuristiky, Grouping genetic algorithm, redukcia dát

Poďakovanie

Touto cestou by som sa rád poďakoval svojmu školiteľovi doc. Ing. Michal Koháni, PhD. za jeho neustálu pomoc, časté diskusie a trpezlivé vedenie počas celého doktorandského štúdia. Zároveň by som sa chcel poďakovať všetkým kolegom na katedre za ich záujem o tému mojej dizertačnej práce a za ich nápady, ktoré mi veľmi pomohli.

Abstrakt

JANOVEC, Maroš: *Metódy pre riešenie úlohy návrhu turnusov elektrobusov vo verejnej doprave* [Dizertačná práca].

Žilinská univerzita v Žiline. Fakulta riadenia a informatiky, Katedra matematických metód a operačnej analýzy.

Školiteľ: doc. Ing. Michal Koháni, PhD.

Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, 2021. 130 s.

V poslednej dobe začínajú získavať na význame elektrické vozidlá, keďže ich prevádzkové náklady sú podstatne nižšie ako v prípade konvenčných vozidiel. Zároveň sa ich používaním znižujú emisie CO_2, čím sa prispieva k zlepšeniu životného prostredia. Jednou z oblastí, kde je možné nasadiť elektrické vozidlá, je verejná doprava. V rámci tejto oblasti je možné vymeniť konvenčný autobus za elektrický autobus, keďže ich možnosti nasadenia sú veľmi podobné. Pri nasadení elektrických autobusov je nutné riešiť viaceré optimalizačné úlohy ako návrh siete nabíjacích staníc, návrh liniek, návrh turnusov vozidiel a posádok, kde na rozdiel od riešenia týchto úloh v prípade konvenčných vozidiel je nutné brať do úvahy špecifické črty, ktorými sa líšia od konvenčných autobusov. Sú to napríklad obmedzený dojazd a dlhý čas nabíjania. V tejto práci sa budeme zaoberať riešením úlohy návrhu turnusov elektrických autobusov vo verejnej doprave. V práci sú popísané základné pojmy pre úlohu návrhu turnusov, ďalej je zadefinovaná úloha návrhu turnusov vozidiel, špecifiká súvisiace s prevádzkou elektrických autobusov a prehľad súčasného riešenia danej problematiky. V práci sa ďalej prezentuje definícia úlohy návrhu turnusov pre elektrické autobusy vo verejnej doprave, ako aj navrhnutý matematický model. Okrem riešenia pomocou matematického modelu je navrhnutých aj niekoľko metód na riešenie tejto úlohy. Prvou navrhnutou metódou je metóda redukcie vstupov s využitím riešenia pomocou matematického modelu a IP solvera, ktorá je schopná redukovať množstvo spojov ich spájaním, čím zmenší rozmer úlohy, ktorá sa následne rieši pomocou matematického modelu. Druhá metóda je založená na prístupe generovania stĺpcov, kde boli navrhnuté špecifické algoritmy na riešenie pod-problému založené na algoritme hľadania *k*-najkratších ciest. Treťou navrhnutou metódou bola implementácia metaheuristiky Grouping genetic algorithm, ktorá je špecifická pre prácu s rozdeľovaním celej množiny prvkov do skupín. Všetky navrhnuté metódy boli otestované na testovacích sadách úloh vytvorených z reálnych dát prevádzky MHD v meste Žilina. Každá metóda bola skúmaná z pohľadu získaných výsledkov ako aj času výpočtu a boli porovnané voči optimálnym výsledkom získaným pomocou riešenia matematického modelu, respektíve voči dolnej hranici riešenia v prípade úloh väčšieho rozsahu, kde nebolo možné získať optimálne riešenie. V závere práce boli všetky navrhnuté metódy porovnané a vyhodnotené.

Kľúčové slová: návrh turnusov, elektrické autobusy, matematické modelovanie, exaktný prístup, generovanie stĺpcov, metaheuristiky, Grouping genetic algorithm, redukcia dát

Abstract

JANOVEC, Maroš: *Solving methods for the electric bus scheduling problem in public transport systems* [Dissertation thesis].

University of Žilina. Faculty of Management Science and Informatics, Department of Mathematical Methods and Operations Research

Supervisor: doc. Ing. Michal Koháni, PhD.

University of Žilina, Faculty of Management Science and Informatics, 2021. 130 p.

In recent years the electric vehicles are gaining a lot of interest, due to the lower operational costs than conventional vehicles. Also, with the usage of electric vehicles, the CO_2 emissions are decreased, which has great importance in improving the living environment. One of the fields, where we can deploy electric vehicles is public transport. In this environment, the change from a conventional bus to an electric bus is possible, because of the similar deployment options. However, during the deployment of electric buses problems like the design of charging network, design of lines, design of vehicle and crew schedules need to be addressed, because of the differences of electric buses from conventional buses like limited driving range and long recharging time. In this work, we address the problem of designing the electric bus schedules in the public transport system. In the text the basic terms of vehicle scheduling problem are described. Next, the problem of vehicle scheduling and the specifics of electric buses are defined as well as the overview of current solutions on this topic. In the thesis, the definition of the electric bus scheduling problem is defined with the formulation of the mathematical model. Besides the solution via the mathematical model, we propose several methods for the solution of the designed problem. The first proposed method is a method based on the input reduction heuristic with the application of a mathematical model and IP solver. This method is able to reduce the number of service trips by grouping them. The grouping decreases the size of the problem and then the problem is solved by the mathematical model. The second method is based on the column generation approach, where we designed algorithms for solving the subproblem phase, specific for our case of electric bus scheduling based on the *k*-shortest path algorithm. The third proposed method is an implementation of a metaheuristic Grouping genetic algorithm, which is specific to work with a partitioning of a whole set of elements into groups. All the proposed methods were tested on datasets generated from real data provided by public transport system provider in the city of Žilina. Each proposed method was researched from the point of obtained results as well as the computation time and the obtained results were compared to the optimal results gained by solving the mathematical model, or alternatively to the lower bound of the solution in the case of larger datasets, where optimal results could not be obtained. At the end of the thesis, all the methods were compared and evaluated.

Key words: scheduling, electric buses, mathematical model, exact approach, column generation, metaheuristics, Grouping genetic algorithm, data reduction

Obsah

Úvod

V posledných rokoch začínajú prechádzať do popredia elektrické vozidlá. Keďže nemajú žiadne emisie CO_2 a zároveň majú nízke prevádzkové náklady, začínajú byť zaujímavou alternatívou ku konvenčným vozidlám. Tento trend sa týka aj verejnej dopravy. V blízkej dobe môžeme očakávať výmenu flotily konvenčných vozidiel za flotilu elektrických vozidiel. Vzhľadom na to, že špecifikácia elektrických vozidiel sa v niektorých ohľadoch značne líši od konvenčných vozidiel, je nutné riešiť problém návrhu dopravného systému pre elektrické vozidlá inak ako pre konvenčné vozidlá.

Doprava je súčasťou výrobného procesu, ako aj súčasť poskytovania služieb, keďže je schopná preklenúť čas aj priestor. Dôležitosť dopravy v spoločnosti je dokladovaná aj záujmom štátov i firiem na zlepšovaní jej ekonomických, ekologických a kultúrno-spoločenských vplyvov. Tieto ciele sa praxi dajú dosiahnuť dvomi prístupmi. Prvým prístupom je modernizácia technologickej časti dopravného systému. Tento prístup zahŕňa vývoj nových technológií, nasadzovanie ekonomickejších dopravných prostriedkov, inovácií a podobne. Druhým prístupom je reorganizácia štruktúry dopravného systému, čím dosiahneme zlepšenie dopravnej technológie a činnosti dopravného systému s možnosťou zapojenia výsledkov získaných prvým prístupom. Problémom je komplikovaná štruktúra a územná rozľahlosť dopravných systémov, a preto je nutné využitie sofistikovaných nástrojov na spracovanie informácií. V našej práci sme sa zamerali na oblasť verejnej dopravy v mestách.

Pri návrhu systému verejnej dopravy je nutné riešiť množstvo úloh. Sú nimi napríklad rozmiestnenie zastávok a návrh liniek, kde určujeme, kde bude začiatočná a koncová stanica linky a kadiaľ bude daná linka prechádzať, čo znamená, na ktorých zastávkach sa bude zastavovať s možnosťou nástupu a výstupu cestujúcich. Ďalšou úlohou je návrh spojov, čím určujeme ako často a kedy bude premávať vozidlo na danej linke. Pri úlohe návrhu turnusov rozhodujeme, ktoré vozidlo pôjde obslúžiť daný spoj a návrh zmien posádok vozidiel určuje, ktorá posádka bude obsluhovať dané vozidlo v danom čase. Na riešenie týchto úloh sa najčastejšie využívajú metódy matematického programovania. Nakoľko je každá z týchto úloh komplexnou úlohou z oblasti operačného výskumu týkajúcou sa strategických rozhodnutí, je problém ich riešiť pre veľké rozsahy úloh, ktoré sa často vyskytujú v praxi. Preto je ďalšou úlohou navrhnúť algoritmy, schopné riešiť úlohy

veľkého rozsahu s dobrou kvalitou riešenia v rozumnom čase, čo prispeje k úsporám pri realizácii daného dopravného systému.

Informatikou rozumieme vedu, ktorá sa zaoberá prácou s informáciami. Medzi formy práce s informáciami patrí ich zhromažďovanie, spracovanie, uloženie, transformácia a využitie v rôznych typoch systémov. Informatika obsahuje vedecké zložky skúmajúce predmet skúmania vo všeobecnej rovine, ako aj aplikačné zložky prospievajúce k aplikácii získaných poznatkov v praxi za účelom rozvoja produktov a služieb.

Informatika je používaná takmer vo všetkých oblastiach života, keďže prispieva k možnostiam lepšieho riešenia problémov. V tejto práci sa zameriame na využitie informatiky v oblasti dopravy pri riešení komplexných úloh veľkého rozsahu, ktoré nie je možné riešiť bez využitia sofistikovaných nástrojov poskytovaných práve informatikou.

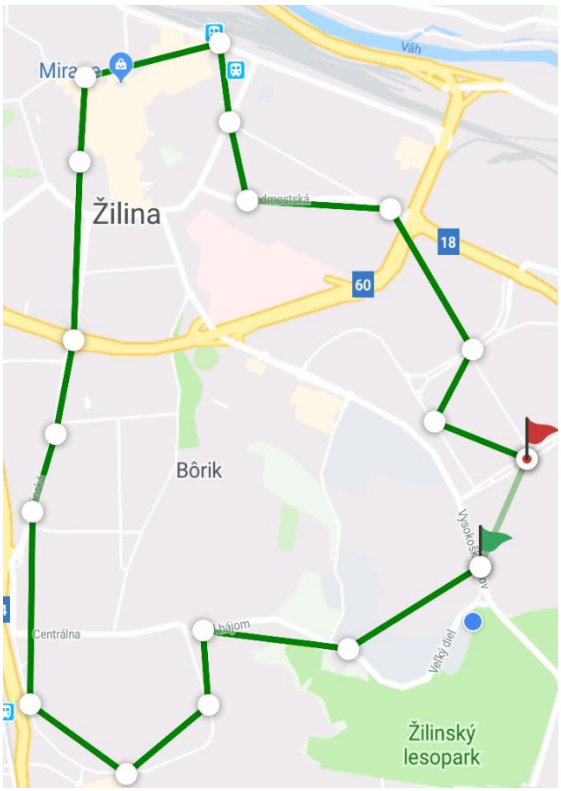
Úloha, ktorou sa budeme zaoberať v práci, je úloha návrhu turnusov elektrických vozidiel. Pri tejto úlohe však nebudeme riešiť návrh zmien posádok pre tieto vozidlá. Táto úloha pomerne je všeobecná a nakoľko rôzne typy vozidiel majú rôzne vlastnosti, budú sa líšiť aj problémy týkajúce sa konkrétneho typu vozidla. Keďže najpoužívanejším typom vozidla vo verejnej doprave je konvenčný autobus, je možné ho nahradiť elektrickým autobusom, ktorý sa konvenčnému autobusu z pohľadu použitia veľmi podobá. Verejná doprava je zároveň miestom, kde by mohli elektrické autobusy priniesť značné úspory nákladov na prevádzku a zároveň pomohli znížiť emisie a tak zlepšiť ovzdušie v mestách. Našou úlohou teda bude riešiť úlohu návrhu turnusov elektrických autobusov vo verejnej doprave.

1 Úvod do problematiky

Pohybujeme sa v oblasti verejnej dopravy a špeciálne v prostredí návrhu turnusov. Preto je dôležité si zadefinovať niekoľko základných pojmov, ktoré sa v danej problematike používajú. Pri návrhu turnusov sa pracuje s dopravnou sieťou a niekoľkými základnými pojmami, ktoré popisujú dopravný systém.

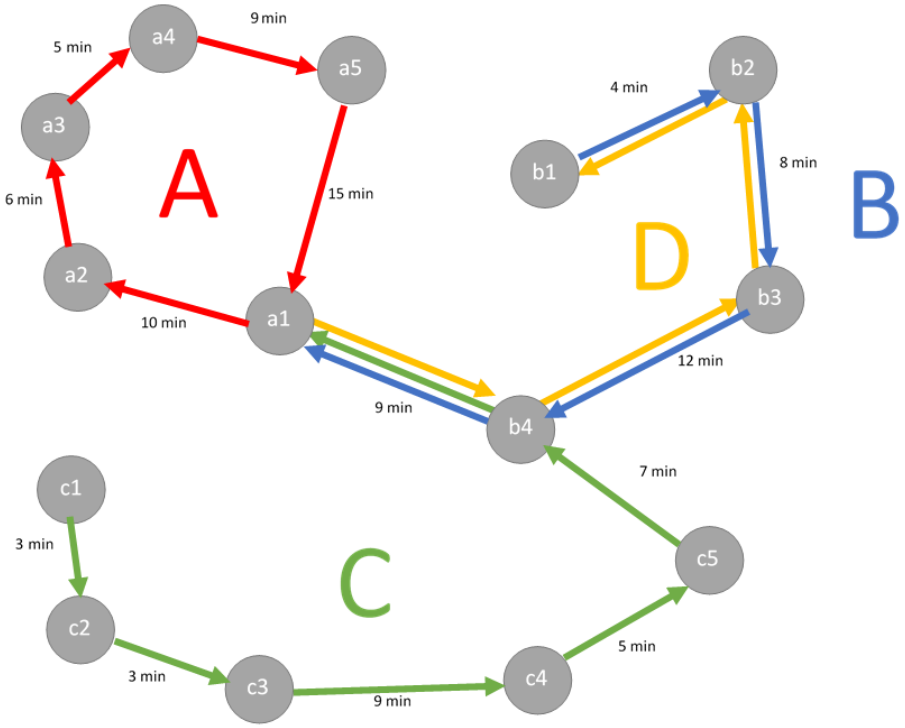
Dopravnú sieť môžeme zadefinovať ako graf, ktorého vrcholy sú tvorené uzlami cestnej siete, respektíve ďalšími významnými bodmi. V našom prípade týmito bodmi budú zastávky mestskej dopravy, prípadne depo a umiestnenia nabíjačiek. Cestné úseky budú tvoriť hrany grafu dopravnej siete, pričom môžu byť jednosmerné alebo obojsmerné. Každý cestný úsek je charakterizovaný jeho dĺžkou v metroch alebo kilometroch. Z pohľadu úloh riešených v dopravných systémoch je nutné poznať aj časové ohodnotenie úsekov, ktoré predstavuje čas, za ktorý je daný úsek možné prejsť.

Základnou jednotkou popisujúcu trasy vozidiel verejnej dopravy sú spoje. Každý spoj tvorí neprázdna konečná množina zastávok, ktoré sú rozmiestnené v dopravnej sieti a najčastejšie prislúchajú k niektorým jej uzlom, respektíve sa nachádzajú na dobre prístupných miestach pre cestujúcich. Celý spoj je možné popísať ako jednosmernú cestu definovanú v cestnej sieti, pričom vozidlo vykonáva prepravnú činnosť [41][47]. Trasa je definovaná prechodmi medzi zastávkami (obr. 1.1). Z pohľadu úlohy návrhu turnusov sú však dôležité iba počiatočná a koncová zastávka spoja, ktoré nám presne geograficky udávajú miesto začiatku spoja a miesto ukončenia spoja. Miesto začiatku a miesto konca spoja sa môžu nachádzať na tom istom mieste. Okrem trasy popísanej začiatočnou a koncovou zastávkou je nutné pri definícii spoja poznať aj ďalšie údaje. Sú nimi čas začiatku spoja a čas ukončenia spoja. Čas začiatku spoja nám hovorí, kedy sa má vozidlo dostaviť na prvú zastávku spoja. Čas konca spoja je časom, kedy sa vozidlo dostane na poslednú zastávku spoja. Po odčítaní času konca spoja od času začiatku dostávame časové trvanie spoja. Časové trvanie spoja musí zahŕňať časovú dĺžku všetkých úsekov prejdených počas spoja ako aj časy potrebné pre nástup a výstup cestujúcich. Spoj je teda z pohľadu časovej a priestorovej polohy presne určený miestom začiatočnej zastávky, miestom koncovej zastávky, časom začiatku a časom konca [41][47]. Množina všetkých spojov počas jedného dňa tvorí cestovný poriadok [1].



Obr. 1.1: Príklad spoja v dopravnej sieti v meste Žilina

Príklad grafu popisujúci spoje môžeme vidieť na obrázku 1.2. Graf $G = (V, H)$ je usporiadaná dvojica množín $V = \{a_1, \dots, a_5, b_1, \dots, b_4, c_1, \dots, c_5\}$ a $H = \{(a_1, a_2), (a_2, a_3), \dots, (a_5, a_1), (b_1, b_2), \dots, (b_4, a_1), (c_1, c_2), \dots, (c_5, b_4), (a_1, b_4), (b_4, b_3), \dots, (b_2, b_1)\}$, kde množina V reprezentuje vrcholy grafu, v našom prípade zastávky a množina H reprezentuje orientované hrany, čiže cestu vozidla medzi týmito dvomi zastávkami. V tomto grafe máme definované štyri podgrafy reprezentujúce spoje. Červenou je označený spoj $A = \{a_1, a_2, a_3, a_4, a_5, a_1\}$, modrou spoj $B = \{b_1, b_2, b_3, b_4, a_1\}$ zelenou spoj $C = \{c_1, c_2, c_3, c_4, c_5, b_4, a_1\}$ a žltou spoj $D = \{a_1, b_4, b_3, b_2, b_1\}$. Každá hrana grafu obsahuje ohodnotenie predstavujúce čas prechodu v minútach.



Obr. 1.2: Graf popisujúci spoje v dopravnej sieti s vyznačením spojov a ohodnoteniami hrán.

Turnus v cestovnom poriadku je postupnosť spojov, ktoré sú obslúžené jedným vozidlom počas jedného dňa, pričom turnus musí začínať a končiť v tom istom stredisku, ktorým je najčastejšie depo. Podmienkou pre postupnosť spojov je aj časová prípustnosť, ktorá nám hovorí, že medzi koncovým časom jedného spoja a začiatočným časom nasledujúceho spoja musí byť dostatočná časová rezerva na prekonanie presunu medzi koncovou zastávkou prvého spoja a začiatočnou zastávkou nasledujúceho spoja [41][47].

Jednou z podmienok pri zostavovaní turnusov je, aby boli obslúžené všetky spoje. Zároveň turnusy zostavujeme s určitým zámerom. Zámerom môže byť napríklad minimalizácia nákladov spojených s realizáciou turnusov. Náklady sú tvorené najčastejšie počtom použitých vozidiel, nakoľko každé ďalšie vozidlo znamená navýšenie nákladov na nákup a údržbu vozidla. Ďalšou časťou nákladov môžu byť prevádzkové náklady vo forme množstva najazdených kilometrov na presunoch medzi jednotlivými spojmi, čím sa snažíme čo najviac znížiť čas kedy vozidlo chodí naprázdno.

Okrem úlohy priradenia vozidiel je nutné riešiť aj priradenie posádok k jednotlivým vozidlám a posádkam je nutné zostaviť zmeny. Väčší počet vozidiel pri turnusoch vozidiel znamená navýšenie počtu ľudí na obsluhu vozidiel, čo je jeden z ďalších dôležitých faktorov ovplyvňujúcim kvalitu turnusov. Posádky vozidiel taktiež obmedzujú tvorbu turnusov tým, že majú zo zákona určenú maximálnu dobu jazdy, ako aj povinnosť dodržiavať zákonom

stanovené prestávky. Pri tvorbe zmien a turnusov nemusí byť splnené, že jeden turnus musí odpovedať jednej zmene, nakoľko sa môžu posádky vozidla vystriedať. Nakoľko je však zostavovanie zmien posádok ďalšou veľmi náročnou úlohou, tak sa ňou v tejto práci nebudeme zaoberať.

Pre názornejšie predstavenie si ukážeme tvorbu časopriestorového grafu spojov na ktorom sa často rieši úloha návrhu turnusov. Na vhodnejšie vyjadrenie spojov a prechodov medzi nimi si vytvoríme časopriestorový graf spojov $G´ = (V', H´)$, ktorého príklad môžeme vidieť aj na obrázku 1.3. Ten vytvoríme transformáciou grafu spojov v cestnej sieti (obr. 1.2) $G = (V, H)$ tak, že každý spoj bude reprezentovaný jedným vrcholom. Tento vrchol bude obsahovať štyri údaje definujúce spoj – miesto začiatku, miesto konca, čas začiatku a čas ukončenie spoja. Ďalej do grafu G pridáme dva špeciálne vrcholy reprezentujúce depo, z ktorého vozidlá ráno vychádzajú a večer sa do neho vracajú.

Orientovanými hranami časopriestorového grafu spojov $G´ = (V', H´)$ budú prechody medzi jednotlivými spojmi, respektíve medzi depom a spojom, kde hrana (a, b) medzi vrcholmi (spojmi) $a \in V´$ a $b \in V´$ reprezentuje prázdny prejazd medzi koncovou zastávkou spoja a a začiatočnou zastávkou spoja b. Táto hrana je ohodnotená dĺžkou a časom prechodu.

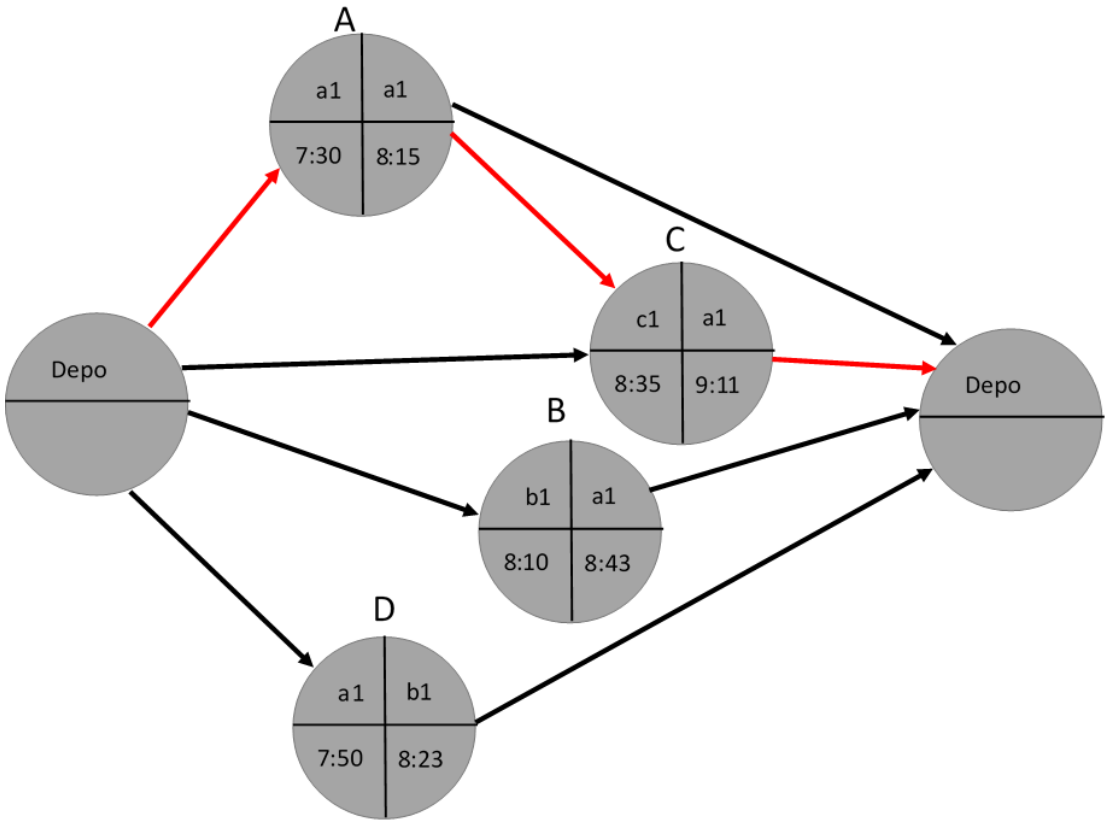
Tab. 1.1: Matica časových vzdialeností *t* medzi koncovou (k) a začiatočnou (z) zastávkou spojov

t (min)	z(A)	z(B)	z(C)	z(D)
k(A)	0	15	20	0
k(B)	0	15	20	0
k(C)	0	15	20	0
k(D)	20	0	35	20

Potom turnus bude podgrafom orientovaného grafu $G´$, pričom tento podgraf obsahuje jedinú cestu grafom definujúcu postupnosť obslúžených spojov. Toto usporiadanie musí spĺňať podmienku následnosti, čiže pri n-tici spojov v turnuse $T = (s_1, s_2, s_3, \ldots, s_n)$ platí následnosť $s_1 < s_2 < s_3 < \ldots < s_n$, čo znamená, že je možné sa časovo presunúť z cieľovej zastávky spoja s_i do počiatočnej zastávky spoja s_{i+1} pre $i = 1, \ldots, n-1$. Zároveň musí platiť, že turnus začína a končí v depe. Pre zostavenie turnusov pre celý cestovný poriadok vytvárame takú množinu podgrafov, kde sú obsiahnuté všetky vrcholy grafu G' práve jedenkrát (okrem ranného a večerného depa) a platí podmienka následnosti spojov pre každý turnus. Pokiaľ je naším optimalizačným kritériom minimalizácia počtu použitých vozidiel, hľadáme množinu podgrafov s minimálnou kardinalitou. V prípade, že

chceme minimalizovať celkovú dĺžku prázdnych prejazdov budeme hľadať podgrafy s najmenším súčtom ohodnotení hrán prázdnych prejazdov, čiže pre hrany $h \in H'$.

Pokiaľ by sme chceli vytvoriť turnusy pre spoje z obrázku 1.2, definujeme si časové ohodnotenia hrán medzi koncovými a začiatočnými zastávkami liniek pomocou matice v tabuľke 1.1. Zároveň si musíme stanoviť začiatočný čas spoja a čas jeho ukončenia. Príklad turnusu je zobrazený na obrázku 1.3 na upravenom časopriestorovom grafe, kde sú hrany časopriestorového grafu redukované iba na tie, ktoré reprezentujú možný prechod z časového hľadiska. Jedným z možných turnusov je napríklad turnus $(Depo, A, C, Depo)$ zobrazený červenou farbou.



Obr. 1.3. Príklad turnusu $(Depo, A, C, Depo)$ zobrazený na časopriestorovom grafe pre spoje definované na obrázku 1.2.

1.1 Definícia úlohy návrhu turnusov

Úloha návrhu turnusov vozidiel je intenzívne skúmaná v posledných 50-60 rokoch. Nie je to z dôvodu jej náročnosti, ale hlavne z dôvodu rôznych možností rozšírenia tejto úlohy, aby sa viac podobala realite. Tým sa však úloha stáva podstatne zložitejšou a často sa z nej stáva NP-ťažká úloha [2]. V tejto časti si uvedieme základný popis úlohy, jej základné znenie a model, ako aj možné štandardné rozšírenia úlohy. Keďže sa v našej práci budeme

venovať elektrickým vozidlám, uvedieme aj niektoré problémy, prečo nie je možné riešiť úlohu návrhu turnusov elektrických vozidiel ako štandardnú úlohu návrhu turnusov.

Pre popísanie základnej úlohy návrhu turnusov s jedným depom a jedným typom vozidiel existuje niekoľko matematických modelov v závislosti od pohľadu na daný problém. My si uvedieme jeden z nich. K tomu máme zadefinované nasledujúce množiny.

N: množina všetkých spojov

D_0: ranné depo

D_n: večerné depo

K: množina všetkých vozidiel

$F(i)$: množina možných nasledovníkov spoja $i \in N$

$B(i)$: množina možných predchodcov spoja $i \in N$

Nech máme definovanú množinu N spojov cestovného poriadku s fixnými časmi začiatku ($s_i\ ,\ i \in N$) a konca ($k_i\ ,\ i \in N$), ako aj definovanými umiestneniami začiatočnej a koncovej stanice. Ďalej máme definované časy prechodu medzi jednotlivými dvojicami staníc ($t_{ij},\ i,j \in N$) a máme k dispozícii vozidlá jedného typu. Našou úlohou je priradiť jednotlivé spoje vozidlám tak, aby boli splnené nasledujúce podmienky:

- každý spoj bol obslúžený práve jedným vozidlom,
- každé vozidlo bude obsluhovať iba prípustnú sekvenciu spojov a
- celkové náklady budú minimálne.

Prvá podmienka zabezpečuje, že obslúžime všetky spoje a na obslúženie každého spoja nám stačí jedno vozidlo. Druhá podmienka nám obmedzuje množinu možných kombinácií spojov iba na tie, ktoré sa dajú z časového hľadiska uskutočniť, čiže dva spoje $i \in N$ a $j \in N$ môžu za sebou nasledovať iba vtedy, ak platí podmienka $k_i + t_{ij} \leq s_j$, kde k_i je koniec spoja $i \in N$, t_{ij} je čas prechodu medzi koncovou stanicou spoja $i \in N$ a začiatočnou stanicou spoja $j \in N$ a s_j je začiatočný čas spoja $j \in N$. Celkové náklady sa môžu skladať z dvoch častí. Prvou časťou si fixné náklady, napríklad investícia do vozidiel a ich údržba. Druhou časťou sú variabilné alebo prevádzkové náklady, ktoré môžu byť interpretované rôznymi spôsobmi a súvisia s kvalitou samotného rozvrhu pri daných fixných nákladoch, napríklad celková prejdená vzdialenosť na prechodoch medzi spojmi, produktívny čas alebo čas čakania vozidla. V praxi sa najčastejšie snažíme minimalizovať

fixné náklady a minimalizácia variabilných nákladov sa rieši ako druhotná optimalizácia s nižšou prioritou.

Spoj j môže byť nasledovníkom spoja i , čiže patrí do množiny $F(i)$ práve vtedy, ak spĺňa podmienku možnej následnosti, čiže $k_i + t_{ij} \leq s_j$, kde k_i je koniec i-tého spoja, t_{ij} je čas prechodu medzi koncovou stanicou spoja i a začiatočnou stanicou spoja j a s_j je začiatočný čas spoja j. Podobne si môžeme zadefinovať množinu $B(i)$ možných predchodcov spoja i, kde musí byť splnená podmienka $k_j + t_{ji} \leq s_i$, aby mohol byť spoj j zaradený ako možný predchodca. Okrem týchto množín si definujeme premennú x_{ij}^k, ktorá predstavuje rozhodnutie, či obslúžime spoj j hneď po spoji i vozidlom k, alebo nie.

$$min \qquad \sum_{k \in K} \sum_{j \in F_{D_0}} x_{D_0 j}^k \tag{1.1}$$

$$z.p. \qquad \sum_{k \in K} \sum_{i \in B(j)} x_{ij}^k = 1 \qquad \forall\, j \in N \tag{1.2}$$

$$\sum_{i \in B(j)} x_{ij}^k = \sum_{l \in F(j)} x_{jl}^k \qquad \forall\, j \in N\,, k \in K \tag{1.3}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, k \in K, i \in N \cup D_0 \cup D_n, j \in F(i) \tag{1.4}$$

Účelová funkcia (1.1) vyjadruje iba fixné náklady, čiže počet vozidiel. Podmienky (1.2) zabezpečujú, že každý spoj bude obslúžený práve jedným vozidlom a podmienky (1.3) nám zabezpečujú, že po obslúžení spoja bude vozidlo pokračovať ďalej k ďalšiemu spoju, respektíve k depu. Podmienky (1.4) nám vyjadrujú binárnosť premennej x_{ij}^k.

Tento typ úlohy nepatrí medzi NP-ťažké úlohy nakoľko sa táto úloha dá previesť na úlohu hľadania maximálneho toku v sieti k čomu existuje rýchly algoritmus na nájdenie optimálneho riešenia. Avšak väčšina rozšírení tejto úlohy spôsobí, že rozšírená úloha bude NP-ťažká.

1.1.1 Varianty úlohy

Okrem základných podmienok úlohy návrhu turnusov sa v literatúre vyskytujú aj rozšírenia tejto úlohy, kde predpokladáme existenciu viacerých dep pre vozidlá, možnosť heterogénnej flotily s viacerými typmi vozidiel, kde každé nemôže obsluhovať všetky spoje alebo možnosť variabilných časov odchodu na spoj a ďalšie obmedzenia turnusov jednotlivých vozidiel. Niektoré z týchto variant si teraz popíšeme.

Viacero dep

V prípade návrhu turnusov s viacerými depami máme k dispozícii viacero dep rozmiestnených na rôznych miestach v dopravnej sieti. Zároveň sa pridáva podmienka, že sa konkrétne vozidlo musí vrátiť po skončení turnusu do toho istého depa, z ktorého ráno vyšlo. Táto podmienka spôsobí, že sa úloha návrhu turnusov stane NP-ťažkou. Pre niektoré špecifické prípady tejto úlohy existujú rýchle exaktné algoritmy, ako napríklad pre úlohu s dvomi depami založené na teórii grafov spomínané v [3].

Heterogénna flotila

Ďalšou možnosťou rozšírenia úlohy je návrh turnusov s viacerými typmi vozidiel, čiže s heterogénnou flotilou. Táto úloha sa rieši v prípade, že máme k dispozícii viacero typov vozidiel a na niektoré spoje musia byť použité práve vybrané typy, napríklad na spoje v blízkosti nemocníc musia byť použité bezbariérové vozidlá. Rôzne typy vozidiel môžu mať rôzne fixné alebo prevádzkové náklady alebo rýchlosti na prekonanie presunov medzi spojmi. Okrem toho každý typ vozidla môže mať obmedzenú kapacitu. Tento problém je opäť NP-ťažký, aj bez kombinácie s viacerými depami [2].

K tejto úlohe bolo navrhnutých množstvo spôsobov riešenia. Sú založené na exaktnom, ale aj na heuristickom prístupe. Jedným z najpoužívanejších spôsobov riešenia je vytvoriť multigraf, ktorý sa skladá z podsietí vytvorených osobitne pre každú kombináciu depa a typu vozidla. Následne sa pre každú hranu multigrafu vytvára rozhodovacia premenná.

Ak sú spoje cestovného poriadku obmedzené iba na niektoré typy vozidiel, tak dostávame ďalšie rozšírenie nazývané skupiny typov vozidiel. Toto rozšírenie sa rieši podobne ako návrh turnusov s heterogénnou flotilou.

Časové okná

Ďalším rozšírením v aplikácii návrhu turnusov vozidiel je zváženie variabilného času príchodu a odchodu vozidla nazývaného časové okná.

Štandardne sa dopravný systém skladá z dvoch rôznych typov spojov. Prvým sú pravidelné spoje, ktoré sú zahrnuté v linkách. Druhou skupinou sú nepravidelné spoje, napríklad školské autobusy alebo posilňujúce spoje počas dopravnej špičky. Časové okná sa najčastejšie aplikujú na nepravidelné spoje, pretože posun spoja neovplyvní zmenu frekvencie linky, avšak malé zmeny môžu byť aplikovateľné aj na pravidelné spoje [2].

Pri tejto úlohe teda definujeme časové okno pre spoj i cez najskorší a najneskorší možný začiatok l_i a u_i pre všetky spoje.

Vo všeobecnosti je problém návrhu turnusov vozidiel s časovými oknami NP-ťažký nakoľko aj najjednoduchšia verzia s jedným vozidlom a jedným depom môže byť prevedená na úlohu obchodného cestujúceho s časovými oknami.

Obmedzenia spojov

Ďalším praktickým rozšírením sú obmedzenia vynucujúce špeciálne vlastnosti na prípustnom rozvrhu turnusov. Asi najtypickejším príkladom tohto rozšírenia sú časové obmedzenia rozvrhu, ktoré nám obmedzujú maximálnu dobu obehu vozidla na turnuse kvôli palivu alebo údržbe. Je to jedna z alternatív ako formulovať problém návrhu turnusov elektrických autobusov vo verejnej doprave, avšak zahŕňa iba niektoré ich aspekty [2].

Elektrické vozidlá

Oblasťou, ktorá získava v posledných rokoch množstvo pozornosti je elektromobilita. Takisto narastá jej význam kvôli nízkym emisiám a lepšiemu využitiu energie. Avšak, zároveň s nástupom elektrických vozidiel prichádza aj množstvo nových problémov pri riešení úloh s ich nasadzovaním. Preto môžeme povedať, že ďalším rozšírením úlohy návrhu turnusov vozidiel je aj úloha návrhu turnusov elektrických vozidiel.

Elektrické vozidlá sa však od konvenčných vozidiel so spaľovacími motormi pomerne dosť líšia a treba počítať s ďalšími obmedzujúcimi podmienkami. Sú to napríklad obmedzený dojazd [4], ktorý je možné modelovať pomocou klasického návrhu turnusov vozidiel s obmedzeniami spojov, čím sme schopní obmedziť dobu obehu z dôvodu paliva. Problémom však je, že batérie v elektrických vozidlách je nutné dobíjať a dobitie batérie je časovo náročný proces v porovnaní s doplnením paliva konvenčného vozidla.

Riešenie tohto problému je v súčasnosti stále veľkým problémom a dokonca nie je sformulovaný ani štandardný model popisujúci túto úlohu, nakoľko je modelovanie ovplyvnené použitou technológiou nabíjania, ako aj ďalšími premennými, ktoré vplývajú na samotnú prevádzku elektrického vozidla.

1.2 Elektrické autobusy a ich obmedzenia

Keďže v poslednej dobe začínajú elektrické autobusy naberať na význame, nakoľko nemajú žiadne emisie CO_2 a zároveň majú nízke náklady na prevádzku, je nutné riešiť

problémy spojené s ich zavádzaním do praxe. Jednou z takýchto úloh je aj návrh turnusov elektrických autobusov vo verejnej doprave. Táto úloha sa skladá z dvoch častí. Prvou časťou je základná úloha návrhu turnusov vozidiel a s ňou spojené podmienky pre jednotlivé turnusy.

- Každý spoj je obslúžený práve jedným vozidlom,
- každé vozidlo bude obsluhovať iba prípustnú sekvenciu spojov.

Druhou časťou je rozšírenie základnej úlohy tak, aby brala do úvahy povahu elektrických autobusov. Preto musíme pridať ďalšie podmienky charakterizujúce elektrické autobusy a špecificky spotrebu energie a batériu.

- Kapacita batérie musí byť dostatočná na prejdenie k nabíjačke počas celého turnusu.
- Na vybraných miestach sa môže batéria nabiť. Pri nabíjaní sa však nesmie presiahnuť maximálna kapacita batérie.
- Na každej prípojke sa v jednom čase môže nabíjať iba jedno vozidlo.

Samozrejme pri návrhu turnusov je nutné brať do úvahy aj technológiu nabíjania, keďže tá podstatne ovplyvňuje samotné navrhovanie turnusov. Máme k dispozícii tri kategórie nabíjania. Prvou je pomalé nabíjanie cez noc v depe. Druhou je priebežné nabíjanie na rýchlych nabíjačkách počas prevádzky medzi jednotlivými spojmi. Poslednou alternatívou je výmena batérií. V poslednom prípade sa turnusy budú riešiť podobne ako vo všeobecnom prípade, ale je nutné optimalizovať samotné nabíjanie batérií a v prípade, že nie je k dispozícii plne nabitá batéria, musíme vybrať, ktorú batériu použiť a počítať s nižším dojazdom. My sa budeme v práci venovať úlohe, kde bude použité priebežné nabíjanie.

Ďalšou úlohou je aj výber miest na umiestnenie nabíjačky. V tomto prípade môžu vzniknúť dve rôzne úlohy. Buď máme zadané miesta a počty jednotlivých nabíjačiek alebo hľadáme aj tieto umiestnenia. Vzhľadom na to, že samotná úloha rozmiestnenia nabíjacích staníc je veľmi náročný problém, tak sa v tejto práci budeme venovať prípadu, kedy sú známe umiestnenia staníc.

Keďže v súčasnosti nie je vytvorený žiaden štandardný matematický model pre tento typ úlohy, jedným z cieľov mojej práce bude sformulovať lineárny matematický model rešpektujúci obmedzenia vyplývajúce z použitia elektrických autobusov.

Ďalšou časťou práce bude nájsť vhodnú metódu, ktorou by sa úloha popísaná vytvoreným modelom dala riešiť aj pre úlohy praktického rozsahu. K dispozícii máme niekoľko exaktných metód, pri ktorých by sme chceli zistiť možný rozsah riešiteľných úloh a zároveň získať riešenia pre úlohy malého rozsahu, čím by sme získali základ pre porovnávanie s heuristickými metódami.

Pri skúmaní heuristického prístupu riešenia sa zameriame na existujúce heuristiky a metaheuristiky, najmä na ich úpravu pre daný problém, ako aj na zistenie ich časovej náročnosti a presnosti riešenia.

1.2.1 Batéria

Prvou skupinou problémov sú problémy súvisiace s batériou. Základným obmedzením, ktoré prináša batéria je obmedzenie dojazdu vozidla. Približný dojazd súčasných elektrických autobusov sa pohybuje od 100 do 300 km [4], čo nie je dosť na pokrytie celodennej prevádzky autobusu. Samozrejme, podarilo sa už vyrobiť elektrické autobusy s podstatne väčším dojazdom, avšak testovanie prebehlo zatiaľ iba v laboratórnych podmienkach, nie pri nasadení do plnej prevádzky [5].

S dojazdom súvisí aj problém veľkosti batérie. Ak by sme chceli pridať batérie na zvýšenie dojazdu elektrického autobusu spôsobilo by to navýšenie hmotnosti elektrického autobusu a následné zvýšenie jeho spotreby energie. Podobným problémom sa treba zaoberať aj v prípade množstva pasažierov. Čím väčší počet pasažierov, tým vyššia hmotnosť a zároveň aj vyššia spotreba energie. Pri veľkosti batérie by sme mali aj zvažovať počet možných miest v elektrickom autobuse, keďže väčšie batérie zaberajú miesto cestujúcim a v dôsledku by sa mohlo stať, že budeme potrebovať väčší počet elektrických autobusov na pokrytie požiadaviek pasažierov.

Existuje aj technológia, ktorá dobíja batériu počas jazdy – rekuperácia energie počas brzdenia alebo inak povedané regeneratívne brzdy. Počas brzdenia elektrického autobusu dochádza ku tvorbe elektrickej energie, ktorá sa vracia späť do batérie.

Ďalším aspektom batérie je jej životnosť. Počas používania batérie dochádza k takzvanej degradácii, čo je proces, pri ktorom sa znižuje maximálna kapacita batérie. Tento proces trvá väčšinou niekoľko rokov, pričom sa podľa [6] zníži kapacita lítium-iónovej batérie na 70-80% jej pôvodnej kapacity počas 5-7 rokov používania [50]. Na tento proces vplýva najmä počet nabíjacích cyklov a hĺbka vybitia batérie. Počet nabíjacích cyklov znamená koľkokrát sa batéria nabila do plnej kapacity, pričom sa kumuluje dobitá energia

aj počas čiastočného nabíjania. HĺbkA vybitia batérie označuje minimálnu kapacitu batérie, ktorú má batéria pred nabíjaním. Samozrejme životnosť batérie vplýva aj na náklady spojené s jej výmenou, ktorá sa odporúča robiť približne každých 5-7 rokov. Životnosť batérie sa dá predĺžiť dodržiavaním určitých doporučení. Jedným z nich je udržiavať hladinu nabitia batérie nad 20% jej maximálnej kapacity. Podobne by hladina batérie nemala presiahnuť 80% z celkovej kapacity, čo súvisí so samotným nabíjaním, čo si uvedieme nižšie [7][48].

1.2.2 Vonkajšie vplyvy

Táto skupina problémov ovplyvňuje opäť batériu a spotrebu vozidla. Jedným z príkladov je tvar trasy elektrického autobusu. Pri trase, ktorá je prevažne tvorená stúpaním bude spotreba určite vyššia ako pri jazde po rovnej trase. Naopak pri jazde nadol je nutné brzdiť, čo dáva možnosť na využitie regeneratívnych bŕzd na dobíjanie elektrického autobusu [8].

Ďalším veľmi dobrým príkladom vonkajšieho vplyvu je počasie a špecificky vonkajšia teplota. Bolo zistené, že pri nízkych teplotách batéria stráca kapacitu [51], čo samozrejme môže do značnej miery ovplyvniť turnusy. Takisto v zimnom období sa energia spotrebuje aj na kúrenie, a podobne v letnom období na klimatizáciu, čo zvyšuje odber energie z batérie a znižuje dojazd [8][48][49].

1.2.3 Nabíjanie

Asi najväčším problémom vyplývajúcim z nabíjania je časová náročnosť tohto procesu. Dotankovanie konvenčného autobusu trvá niekoľko minút. Naproti tomu dobitie batérie elektrického autobusu môže zabrať až niekoľko hodín v závislosti od použitej technológie nabíjania. Túto skutočnosť je nutné zvážiť aj pri návrhu turnusov elektrických autobusov. Zvolená technológia nabíjania zároveň ovplyvňuje aj degradáciu batérií, kde rýchle nabíjanie jednosmerným prúdom zvyšuje degradáciu, ale znižuje čas nabíjania [8][9].

Ďalším problémom je samotný proces nabíjania, nakoľko je tento proces nelineárny. Z výskumov vyplýva, že lineárne je ho možné aproximovať len na intervale od 20% do 80% maximálnej kapacity batérie. Pokiaľ sa však bude držať stav nabitia len v týchto hodnotách dosiahneme aj zníženie degradácie batérie [10].

Ďalej je nabíjanie obmedzené elektrickou sieťou, keďže v danom čase je možné poskytnúť iba určitý nabíjací výkon v závislosti na vyťažení siete. Táto skutočnosť môže ovplyvniť rýchlosť nabíjania elektrického autobusu. Pri nabíjaní viacerých vozidiel alebo

počas dňa sa môže výkon každej nabíjačky znížiť. Naproti tomu v noci, keď je odber zo siete menší je možné nabíjať plným výkonom aj väčšie množstvo vozidiel.

Náklady na nabíjanie ovplyvňuje aj cena elektriny. Cena sa počas dňa môže meniť. Najčastejšie je elektrina počas nočných hodín lacnejšia, čo by uprednostňovalo nabíjanie cez noc, avšak je nutné nabíjať aj cez deň. Takisto sa cena môže líšiť v závislosti od zmluvy s prevádzkovateľom, ktorý môže spoplatniť iba odobratú energiu alebo stanoviť cenu na základe maximálnej hodnoty výkonu počas odberu energie [11].

Nemalým problémom je aj počet nabíjacích staníc, ktoré sú k dispozícii, keďže vybudovanie každej stanice prináša nemalé náklady. Preto by sa malo vybudovať iba toľko nabíjacích staníc, koľko je nutné. Pokiaľ by sa však vybudovalo príliš málo staníc, mohlo by sa stať, že sa vytvorí pred nimi front elektrických autobusov čakajúcich na nabitie a v dôsledku toho by mohlo dôjsť ku kolapsu verejnej dopravy, respektíve by bolo nutné nasadiť ďalšie elektrické autobusy .

1.2.4 Rozšírenia úlohy

Takisto ako aj pri klasickom návrhu turnusov vozidiel, aj tu je možné túto úlohu rozšíriť. K týmto rozšíreniam patria napríklad heterogénna flotila elektrických autobusov, úloha s viacerými depami alebo kombinovaná úloha, kde sa riešia turnusy kombinovanej flotily elektrických a konvenčných autobusov.

Ďalšou alternatívou úlohy je úloha s nabíjaním na jednom mieste, najčastejšie v depe, alebo na viacerých miestach rozmiestnených v dopravnej sieti. S tým súvisí aj úloha rozmiestňovania nabíjacích staníc, ktorou sa však v tejto práci nebudeme zaoberať a budeme predpokladať známe umiestnenia staníc.

Možným rozšírením úlohy by mohla byť aj úloha s kombináciou technológií nabíjania, kde sa každý typ elektrického autobusu nabíja inou technológiou.

2 Súčasný stav

V tejto časti popíšeme práce iných autorov venujúcich sa problematike návrhu turnusov vozidiel ako aj problematike riešenia úlohy návrhu turnusov elektrických autobusov v rôznom prostredí a s rôznymi predpokladmi.

2.1 Úloha návrhu turnusov

Úloha návrhu turnusov je jednou z často riešených kombinatorických optimalizačných úloh. Kvôli možnostiam rozšírenia sú intenzívne skúmané možnosti riešenia jednotlivých variantov, ako aj formulácie nových rozšírení.

V práci [31] sa autori zaoberajú riešením úlohy návrhu turnusov s homogénnou flotilou a jedným depom. Predstavujú kvázi-priraďovací algoritmus založený na aukčnom algoritme pre lineárnu kvázi-priraďovaciu úlohu. Zároveň sú tu spomenuté rôzne možnosti modelovania úlohy návrhu turnusov ako aj porovnanie navrhnutej metódy s inými známymi algoritmami na riešenie danej úlohy.

Omnoho problematickejším variantom úlohy návrhu turnusov je variant s viacerými depami, o ktorej je známe, že je to NP-ťažká úloha. Autori práce [32] riešia túto úlohu pomocou exaktného algoritmu vetiev a rezov, ktorá je kombináciou metód vetiev a hraníc s metódou rezných nadrovín. Ich implementácia využíva separačný algoritmus spolu s heuristikou, aby zrýchlili konvergenciu metódy rezných nadrovín. Zároveň predstavili stratégiu vetvenia založenú na koncepte „fractionality persistency“.

Ďalšou možnosťou riešenia úlohy návrhu turnusov s viacerými depami je aj metóda generovania stĺpcov, ktorú použili v [33]. Prezentujú novú formuláciu úlohy návrhu turnusov s viacerými depami ako úlohu delenia množiny, ktorá je vhodná na riešenie pomocou metódy generovania stĺpcov. Prístup riešenia návrhu turnusov vozidiel s viacerými depami pomocou metódy generovania stĺpcov je prezentovaný aj v [34], kde je využívaný na riešenie rozsiahlych úloh obsahujúcich až 25 tisíc spojov.

Ďalším variantom úlohy návrhu turnusov je úloha návrhu turnusov s časovým obmedzením ciest, ktorej sa venujú v práci [35]. Autori prezentujú nový matematický model ako aj dve heuristické metódy. Úloha je riešená na systéme s tranzitnými autobusmi v meste Baltimore, Kanada. Podobným problémom sa zaoberali aj v práci [36], kde na riešenie úlohy

návrhu turnusov vozidiel s obmedzeniami cesty a dojazdu použili metaheuristiku Ant Colony Optimization.

Základnú úlohu návrhu turnusov riešili aj v práci [37], kde bola uvedená do špecifického prostredia transportu cukrovej repy a na jej riešenie boli použité dve metaheuristické metódy – metóda Variable Neighbourhood Search (VNS) a metóda GRASP.

2.2 Úloha návrhu turnusov elektrických autobusov

Práce v oblasti návrhu turnusov elektrických autobusov môžeme rozdeliť na dva smery, v závislosti od použitej technológie nabíjania. Sú to systémy s výmenou batérií a systémy priebežného nabíjania.

2.2.1 Systém s výmenou batérií

Výmena batérií je systém, kde máme k dispozícii elektrický autobus s vymeniteľnou batériou a zároveň je k dispozícii infraštruktúra na ich výmenu za nabité batérie. Táto technológia je pomerne drahá z pohľadu počiatočnej investície, keďže je nutné nakúpiť viacej batérií ako je elektrických autobusov. Avšak táto technológia má potenciál zjednodušiť samotné navrhovanie turnusov, keďže nabíjanie batérie sa môže odohrávať počas prevádzky elektrického autobusu.

Použiteľnosť tejto technológie bola overená v [12], kde autori testovali automatický systém výmeny batérií na elektrickom autobuse. Testovaný autobus mal batériu pripevnenú na streche autobusu a automatický systém ju mohol ľahko vybrať a nahradiť plne nabitou batériou. Prvé testovanie systému prebehlo na uzavretej trati, kde bol elektrický autobus monitorovaný. Získané dáta boli použité na upresnenie spotreby elektrického autobusu a vytvorenie simulačného modelu pre tento systém. Následne bol vykonaný test v meste Pohang v Južnej Kórei, kde testovali spotrebu v reálnych podmienkach. Bola vybraná jedna linka, kde umiestnili dve výmenné stanice a použité boli tri elektrické autobusy. Následne porovnávali simulačný model s reálnymi dátami.

Autori v [13] sa zaoberajú problémom transformácie flotily konvenčných vozidiel na flotilu elektrických vozidiel. V rámci tohto problému sa riešia úlohy zistenia počtu staníc na výmenu batérie, ďalej ich umiestnenie, nabíjacie kapacity a manažment batérií pre každú výmennú stanicu ako aj návrh turnusov a smerovanie vozidiel. V rámci smerovania vozidiel je predstavený matematický model, ktorý vytvára skupiny spojov, ktoré sú obslúžené medzi

zachádzkami na nabíjanie, a teda hľadá najkratšiu cestu na obslúženie skupiny spojov spolu so zachádzkami k staniciam na výmenu batérií. Zároveň je sformulovaný matematický model aj pre úlohu návrhu turnusov elektrických vozidiel. Tým, že predspracujú úlohu návrhu turnusov, je možné ju previesť na úlohu smerovania vozidiel so zachádzkami na výmenu batérií, avšak táto úloha je NP-ťažká a bude ju nutné riešiť pomocou heuristík.

V článku [14] sa rozoberá riešenie problému návrhu turnusov elektrických autobusov s jedným depom pre systém s výmenou batérií. Je predstavený matematický model s dvomi účelovými funkciami, a síce minimalizácia kapitálových nákladov a minimalizácia celkového dopytu po nabíjaní. Zároveň je predstavený algoritmus na riešenie tejto úlohy, ktorý je založený na myšlienke genetického algoritmu s nedominovaným usporiadaním (Non-dominated Sorting Genetic Algorithm - NSGA-II). Úloha je riešená na datasete z mesta Šanghaj, Čína, kde boli vytipované tri linky, kde je obsluhovaných 119 spojov. Stanica na výmenu batérií bola umiestnená asi 5 minút jazdy od depa. Výsledky potvrdzujú možnosť použitia pre návrh turnusov elektrických autobusov s výmenou batérií.

2.2.2 Systém s priebežným nabíjaním

Priebežné nabíjanie je systém nabíjania, kde sa elektrický autobus nabíja počas prevádzky. Toto sa môže diať v depe, na koncových staniciach alebo počas spojov na vybraných zastávkach. Zároveň je možné mať nabíjacie stanice umiestnené na viacerých miestach, alebo len na jednom. Táto technológia je lacnejšia v porovnaní s výmenným systémom z pohľadu kapitálovej investície. Avšak problém návrhu turnusov vozidiel je podstatne náročnejší, keďže musíme brať do úvahy aj stav nabitia elektrického autobusu a musíme plánovať aj čas na nabíjanie. Zároveň je nutné riešiť problém rozvrhu nabíjania, keďže na jednej nabíjačke sa môže v jednom čase nabíjať iba jedno vozidlo.

Prvou časťou problému so systémom priebežného nabíjania je problém umiestnenia nabíjacích staníc a zároveň optimalizácia veľkosti batérie. Týmto problémom sa venujú napríklad v článku [15]. V ňom predstavujú matematický model, ktorý minimalizuje počet použitých nabíjačiek, ako aj minimalizuje kapacitu použitej batérie pre každú linku. V tomto prípade sa predpokladá, že elektrický autobus obsluhuje iba jednu linku a nabíjacia stanica môže byť umiestnená na ľubovoľnej zastávke. Samozrejme, zdieľanie nabíjačiek na spoločných zastávkach medzi linkami je možné. Ďalšou časťou je simulačné overenie výsledkov, kde predstavujú simulačný model, ktorým reprezentujú spotrebu energie elektrického autobusu. Úloha na ktorej daný problém riešia je z mesta Berlín zahrňujúca 17

liniek s použitými 134 autobusmi. Zároveň sa overujú rôzne scenáre popisujúce vonkajšie podmienky a z toho odvodené spotreby elektrického autobusu.

Jedným z príkladov realizácie systému s priebežným nabíjaním je projekt z roku 2014 v meste Milton Keynes [73], kde obsluha jednej linky bola prevedená na elektrické autobusy. Použité bolo indukčné nabíjanie batérie, kde stanice na nabíjanie boli umiestnené na dvoch koncových staniciach linky, nakoľko linka prechádza mestom v oboch smeroch. Počas riešenia boli analyzované dáta o linkách z rôznych miest, pričom sa z nich určovala spotreba elektrického autobusu. Samotné navrhovanie turnusov však riešené nebolo, nakoľko sa snažili nabíjať len počas prestávok na koncových staniciach. Tento článok poskytuje pohľad na úspešnú realizáciu transformácie obsluhy linky z konvenčných naftových autobusov na elektrické autobusy.

Okrem návrhu turnusov vozidiel je nutné riešiť aj optimalizáciu nabíjania na nabíjacej stanici, keďže aj poradie nabíjania, nabíjací čas a výkon nabíjačky vplýva na náklady na nabíjanie. Tomuto problému sa venujú autori v [16]. V rámci tohto článku predstavujú matematický model kontrolovaného nabíjania, ktorý minimalizuje celkové náklady na nabíjanie tým, že sa snaží upraviť množstvo dobitej energie, ako aj nabíjací výkon nabíjačky. Na overenie výsledkov modelu boli zozbierané dáta z existujúcej nabíjacej stanice.

Jedným z možných prístupov na riešenie problému návrhu turnusov elektrických autobusov je simulácia. Príkladom je aplikácia v meste Putrajaya v Malajzii [30], kde overovali možnosť nahradenia plynových autobusov za elektrické. Na postavenie modelu využívajú modelovací nástroj EMME (Equilibre Multimodal, Multimodal Equilibrium), ktorým modelovali prevádzku autobusov, ako aj tranzitný a dopravný systém, ktorý simuloval reálne podmienky v doprave. Ďalším krokom bol zber dát o pasažieroch, použitých autobusoch a doprave. Posledným krokom bolo spustenie a vyhodnotenie simulácie podľa preddefinovaných scenárov predstavujúcich rôzny počet použitých autobusov ako aj rôzne podmienky v doprave. Výsledky simulácie boli porovnávané s reálnymi dátami plynových autobusov, čím došli k záveru, že by elektrické autobusy priniesli zníženie nákladov.

Simulačný prístup zvolili aj autori v [17]. V tomto prípade porovnávali možnosti rôznych typov nabíjania v rámci priebežného nabíjania a ich ekonomický prínos. Porovnávali nabíjanie cez noc, nabíjanie na koncových staniciach a rýchle nabíjanie na

vybraných zastávkach počas spojov. Prípadová štúdia prebehla na príklade mesta Belleville v Kanade, kde prevádzkujú 9 liniek s 237 spojmi. Zo simulácie vyplynulo, že vhodnejšie na prevádzku je nabíjanie na koncových staniciach a rýchle nabíjanie na vybraných zastávkach počas spojov, avšak tieto alternatívy nadmerne zaťažujú elektrickú sieť, zvlášť v prípade špičiek pri odbere elektrickej energie. Z toho dôvodu je nabíjanie cez noc vhodnejšie, ak chceme predĺžiť životnosť transformátorov a znížiť vyťaženosť elektrickej siete. Pri riešení úlohy sú zachované turnusy vozidiel a autori sa snažia prispôsobiť infraštruktúru pre konkrétnu technológiu.

Riešením úlohy návrhu turnusov elektrických autobusov sa zaoberali aj v článku [18], kde predstavili algoritmus založený na *k*-Greedy Algorithm. Táto úloha je riešená pre kombináciu elektrických a naftových autobusov, kde sa autori snažili, aby sa maximalizovala najazdená vzdialenosť elektrických autobusov. Autori limitujú množstvo dobitej energie maximálnym možným odberom z elektrickej siete pre daný čas. Ďalšou podmienkou, ktorú implementovali je obmedzenie maximálnej a minimálnej úrovne nabitia batérie. Počet elektrických autobusov je v tomto prípade obmedzený. Heuristika najprv vytvára turnusy pre elektrické vozidlá a následne sa dopĺňajú naftové autobusy na obslúženie spojov, ktoré neboli pokryté elektrickými autobusmi.

2.2.3 Modelovanie úlohy návrhu turnusov elektrických autobusov s priebežným nabíjaním

Keďže neexistuje jednotný matematický model úlohy návrhu turnusov elektrických autobusov, rôzni autori sa pokúšali o vytvorenie matematického modelu popisujúceho daný problém a zároveň jeho riešenie. Každý z nich zapracováva do modelu iné predpoklady, čím vzniká priestor na ďalší výskum. V nasledujúcej časti si popíšeme niekoľko modelov vytvorených rôznymi autormi, ako aj ich špecifiká a riešenia navrhnuté autormi.

V diplomovej práci [20] boli prezentované dva rôzne modely na riešenie úlohy návrhu turnusov. Prvý model je charakteristický spojitým modelovaním nabíjania, kde množstvo dobitej energie je určené modelom a predstavuje myšlienku, že nabíjanie je lineárny proces. Druhý model je schopný popísať nabíjanie aj ako nelineárny proces, a to pomocou diskretizácie stavu nabitia batérie. Pre každú nabíjaciu udalosť sa vytvára graf stavu nabitia, kde sa prezentujú všetky možné prechody medzi stavmi nabíjania. Následne sa graf následnosti spojov spolu s grafmi stavu nabitia spája do jediného grafu, na ktorom sa následne rieši úloha návrhu turnusov s diskrétnymi stavmi nabitia batérie.

V rámci modelov sú k dispozícii obmedzenia ako limitovanie maximálneho stavu batérie, ďalej limitovanie stavu nabitia na základe predchádzajúceho stavu a maximálny počet prípojok na každej nabíjačke. Účelová funkcia minimalizuje počet použitých elektrických autobusov a zároveň súhrnnú vzdialenosť prechodov medzi spojmi navzájom a medzi spojmi a nabíjačkami.

Prvý model bol riešený exaktne pomocou IP solvera, pričom bolo možné riešiť iba malý rozsah úloh, kde bol počet spojov veľmi limitovaný. Na riešenie druhého modelu s diskretizáciou stavu nabitia bola použitá metóda generovania stĺpcov, ktorá riešila problém aj pre väčšie rozsahy. V metóde bol použitý ako hlavný problém výber vhodných turnusov na obslúženie spojov a ako pod-problém sa riešila úloha nájdenia prípustného turnusu, ktorý by priniesol úsporu.

Ako prípadová štúdia boli použité turnusy v meste Eidhoven. Dáta obsahovali 7 liniek so 709 spojmi. Postupne sa riešili 3 úlohy definované datasetmi A, B a C vytvorenými z pôvodných dát. V tomto prípade spojitý model bol schopný vyriešiť iba prvé dva datasety (A, B), ktoré mali menej ako 300 spojov. Diskrétnym modelom boli vyriešené úlohy popísané všetkými datasetmi, ako aj pôvodná úloha so všetkými spojmi.

V tomto modeli sa však môžu vyskytovať problémy so zachovaním energie, keďže sa môže stať, že sa spotrebuje viac energie ako je nutné. Zároveň tento model je možné použiť iba na úlohu s jedným depom. Vzhľadom na prípadovú štúdiu nebolo nutné riešiť problém s nabíjaním na viacerých staniciach nakoľko sa predpokladá, že sa bude nabíjať iba v depe.

Podobný prístup bol prezentovaný aj v [20]. Opäť sa tu prezentujú dva lineárne modely v závislosti od povahy procesu nabíjania. Prvý model predpokladá lineárnosť nabíjania. Druhý model používa diskretizáciu stavu nabitia batérie na popis ľubovoľného typu nabíjania.

Optimalizujú sa celkové náklady, čiže suma fixných nákladov reprezentujúcich počet použitých elektrických autobusov ako súčet autobusov vychádzajúcich z ranného depa a variabilných nákladov reprezentovanými cenou hrany, čo je cena samotného spoja a prechodu nasledujúceho za ním.

Špecifikom modelov je diskretizácia času, kde celkový čas je rozdelený na intervaly rovnakej dĺžky. Toto umožňuje jednoznačne definovať, koľko elektrických autobusov sa

bude v každom čase nachádzať na nabíjačke, respektíve v depe, keďže sa predpokladá iba jedno miesto nabíjania, a to depo.

Prvý aj druhý model boli riešené exaktne pomocou IBM ILOG CPLEX 12.2. Na riešenie druhého modelu s diskretizáciou stavu nabitia bola použitá aj metóda generovania stĺpcov, kde boli riešené dve verzie modelu. Prvá verzia bol klasický zmiešaný lineárny matematický model a druhá verzia bol zmiešaný lineárny matematický model s Lagrangeoveou relaxáciou. V oboch prípadoch bol hlavným problémom úloha výberu turnusov obsluhovaných jediným vozidlom a ako pod-problém sa riešilo generovanie prípustného turnusu, ktorý prinesie úsporu.

Oba modely boli testované na dátach z mesta Leuven, čo zahŕňalo 4 linky a 543 spojov. Z týchto dát boli vytvorené 4 datasety, kde posledný obsahoval všetky spoje na všetkých linkách. Posledný dataset nemohol byť z časového hľadiska vyriešený exaktne, podarilo sa to iba pomocou metódy generovania stĺpcov. Vo väčšine prípadov metóda generovania stĺpcov našla optimálne riešenie a zároveň to bolo oveľa rýchlejšie ako pri použití exaktnej metódy riešenia.

Autori článku však berú do úvahy iba prípady, kde sa nabíjanie odohráva v depe a nie na iných miestach, zároveň je možné že bude dochádzať ku stratám energie počas nabíjania. Problémom môže byť aj situácia, keď bude musieť elektrický autobus odísť z depa hneď po príchode a bude mať na nabitie len 15 minút, pretože sa predpokladá, že pri odchode z depa je vždy plne nabitý, čo sa za 15 minút určite nedá stihnúť pri definovanej rýchlosti nabíjania.

Trochu odlišným spôsobom pri tvorbe modelu postupovali autori článku [21]. V tomto článku sa predpokladá iba lineárny proces nabíjania, ale rieši sa aj dopad na elektrickú sieť a jej maximálny výkon v danom čase. Okrem toho je modelovaná kombinovaná úloha, kde máme k dispozícii obmedzený počet elektrických vozidiel a zároveň sú k dispozícii naftové autobusy.

Model obsahuje dve účelové funkcie. Prvá je maximalizácia vzdialenosti prejdenej elektrickými autobusmi, nakoľko ich prevádzkové náklady sú podstatne nižšie ako u naftových autobusov. Druhá účelová funkcia minimalizuje náklady spojené s nabíjaním elektrických autobusov. Okrem základných obmedzení návrhu turnusov vozidiel sú v modeli zahrnuté podmienky limitujúce maximálny nabíjací výkon poskytovaný sieťou, ako aj jeho rovnomerné rozdelenie vozidlám. Zároveň musí byť splnené, že sa elektrický

autobus nabíja v rozmedzí minimálneho a maximálneho nabíjacieho výkonu. Samozrejmé je obmedzenie stavu nabitia batérie jeho minimálnou a maximálnou hranicou.

Autori v článku dokazujú, že úloha návrhu turnusov elektrických autobusov je NP-ťažká a preto sa nedá riešiť exaktne na úlohách väčšieho rozsahu. Z toho dôvodu pripájajú aj návrh dvoch heuristických algoritmov na riešenie kombinovanej úlohy návrhu turnusov elektrických a naftových autobusov. Sú to Sequential heuristic a Global heuristic.

Sequential heuristic je založená na myšlienke, že najprv vytvoríme turnus pre každý elektrický autobus a následne pre každý z nich zostavujeme rozvrh nabíjania. Počas tvorby turnusu sa využíva greedy heuristika a následne je zlepšovaná výmennou heuristikou. V tomto kroku sa snažia maximalizovať vzdialenosť najazdenú elektrickými autobusmi. Druhá časť, úloha tvorby nabíjacieho rozvrhu, sa rieši pre každý elektrický autobus zvlášť. Zároveň sa dá previesť na úlohu hľadania maximálneho toku v sieti a autori prezentujú dva algoritmy na jeho hľadanie.

Global Heuristic začína tvorbou prípustných turnusov pre všetky vozidlá. Následne sa v druhej časti rieši optimalizácia rozvrhu nabíjania, pričom sa táto rieši pre všetky elektrické autobusy zároveň. Prvý krok sa vykonáva podobne ako v prípade Sequential heuristic. V prípade Sequential heuristic máme k dispozícii využitie elektrickej siete reprezentujúce pridelený nabíjací výkon v danom čase po každom kroku, avšak pri Global Heuristic je nutné toto využitie odhadovať, nakoľko sa presné využitie elektrickej siete bude počítať až v druhom kroku. Na určenie rozvrhu nabíjania sa opäť úloha prevedie na úlohu hľadania maximálneho toku v sieti.

Na overenie heuristík a modelu boli použité dva datasety. Prvý dataset bol získaný z reálnych dát od firmy Group La Poste a zahŕňal dve úlohy so 45 a 46 linkami. Druhý dataset boli vygenerované dáta. Global heuristic bola schopná vyriešiť všetky úlohy pomerne kvalitne. Sequential heuristic v niektorých prípadoch zlyhala, ale v prípadoch, kde nezlyhala bola podstatne rýchlejšia ako Global heuristic. Pri exaktnom riešení neboli schopní vyriešiť úlohy s viac ako 40 vozidlami, zatiaľ čo heuristiky si poradili s úlohami do 200 vozidiel.

Autori uvažujú iba jedno miesto na nabíjanie – depo, avšak prezentovaný model vie obmedziť nabíjací výkon vzhľadom na vyťaženie elektrickej siete, čo je jeho výhodou pre praktickú aplikáciu.

V článku [22] sa autori pokúšajú riešiť jednak úlohu návrhu turnusov elektrických autobusov a zároveň optimalizácie rozvrhu nabíjania. V tomto prípade sa kombinujú dva typy grafov, a to graf možných nasledujúcich spojov a graf nabíjacích udalostí. Nabíjacie udalosti sa môžu odohrávať medzi spojmi alebo môžu spoje nasledovať priamo za sebou.

V rámci modelu sa minimalizujú celkové náklady, čo zahrňuje fixné náklady na použité autobusy, variabilné náklady spojené s prevádzkou vozidla, čiže náklady za každý prechod, náklady spojené s množstvom nabitej energie a nakoniec náklady na postavenie prípojky k nabíjačke. Vzhľadom na tvar modelovanej siete bolo nutné zabezpečiť časovú následnosť jednotlivých nabíjacích udalostí, nakoľko model určuje aj začiatok a koniec týchto udalostí. Hladina nabitia batérie je obmedzená zhora aj zdola, čo slúži na zníženie vplyvu degradácie batérie a ako podmienka nepresiahnutia maximálnej kapacity batérie. Zároveň tieto podmienky môžu slúžiť ako ohraničenie nabíjacieho procesu, kedy je lineárny.

Model je však nelineárny a nie je ho možné riešiť exaktným spôsobom. Preto autori predstavujú aj metódu riešenia založenú na Grouping genetic algorithm(GGA). V tejto metóde sa vytvárajú takzvané bloky spojov, ktoré predstavujú skupiny spojov, ktoré budú obslúžené za sebou bez nabíjania. Následne sa medzi tieto bloky vkladajú nabíjacie udalosti pri ktorých sa autobus vždy nabije do plnej kapacity.

Druhou časťou úlohy bolo riešenie rozvrhu nabíjania tak, aby sa minimalizoval počet použitých prípojok, nakoľko každá ďalšia prípojka znamená ďalšie náklady na jej vybudovanie. Rieši sa to spájaním všetkých nabíjacích udalostí do rozvrhu tak, aby sa neprekrývali. Ak sa vyskytnú také, čo sa v čase prekrývajú, musia sa odohrávať na odlišných prípojkách.

Modely aj algoritmus boli testované na dátach z miest Aachen s 3 linkami a 200 spojmi a Roskilde s 1 linkou a 135 spojmi. Pri testovaní prebehlo aj ladenie parametrov genetického algoritmu. Testovali sa dva rôzne typy autobusov a výsledky sa porovnávali s pôvodným nastavením systému s naftovými autobusmi. V prípade mesta Roskilde vyšlo, že počet použitých elektrických autobusov bude väčší ako počet naftových autobusov. V prípade mesta Aachen sa v niektorých prípadoch počet použitých elektrických autobusov zhodoval s počtom naftových.

Nevýhodou modelu je, že uvažuje iba jednu lokáciu pre nabíjačky a to depo a zároveň sa uvažuje, že autobus sa vždy nabíja do plnej kapacity, čo z pohľadu využitia energie nemusí byť ekonomické.

2.3 Exaktné metódy

Keďže úloha návrhu turnusov elektrických autobusov je často vyjadrovaná ako úloha zmiešaného celočíselného lineárneho programovania, dajú sa na jej riešenie využiť exaktné algoritmy na riešenie úloh tohto typu. Okrem toho sa často používa aj simplexová metóda, ktorá slúži na čiastkové získavanie výsledkov, respektíve na riešenie LP-relaxácií. Exaktnými metódami na riešenie úloh zmiešaného programovania sú napríklad metóda rezných nadrovín, metóda vetiev a hraníc a metóda generovania stĺpcov.

2.3.1 Metóda rezných nadrovín

Metóda rezných nadrovín [24] stavia na riešení LP-relaxácie modelu úlohy celočíselného programovania pomocou primárnej alebo duálnej simplexovej metódy. Táto metóda postupne pridáva k modelu LP-relaxácie úlohy podmienky (normálne rezy), ktorým vyhovujú všetky celočíselné riešenia úlohy, ale ktorým nevyhovuje napríklad súčasné optimálne neceločíselné riešenie LP-relaxácie. Tým, že pridáme takéto podmienky k modelu sa množina neceločíselných riešení zmenší. Keďže sa v prípade LP-relaxovanej úlohy jedná o úlohu spojitého lineárneho programovania, je možné použiť na riešenie primárnu alebo duálnu simplexovú metódu. Ak počas riešenia LP-relaxácie úlohy s pridanými normálnymi rezmi nájdeme krajný bod, ktorý je optimálnym riešením a zároveň spĺňa podmienky celočíselnosti, potom toto riešenie bude aj optimálnym riešením pôvodnej úlohy.

Dôležitou časťou tohto algoritmu je tvorba normálneho rezu. Bolo publikovaných veľa algoritmov na tvorbu normálnych rezov. My si uvedieme konštrukciu použitú v prvom Gomoryho algoritme použitú na riešenie úlohy celočíselného programovania [24].

Rez sa v tomto prípade vytvára pre konkrétne optimálne neceločíselné bázické riešenie $\underline{\boldsymbol{x}}$ úlohy spojitého lineárneho programovania, ktoré predstavuje LP-relaxáciu celočíselnej úlohy. Toto riešenie mohlo byť získané primárnou alebo duálnou simplexovou metódou. Aby mohol byť rez vytvorený musí byť splnený predpoklad, že všetky koeficienty účelovej funkcie budú celočíselné. Z toho vyplýva, že v prípade celočíselnosti zložiek riešenia $\underline{x}_j$, pre $j = 1, \ldots, n$, bude aj účelová funkcia reprezentovaná zložkou $\underline{x}_0$ celočíselná.

V prípade primárnej simplexovej metódy máme k dispozícii okrem optimálneho riešenia $\underline{\boldsymbol{x}} \in E_{n+1}$ minimalizujúceho účelovú funkciu aj zoznam bázických $\boldsymbol{B}^p$ a nebázických $\boldsymbol{N}^p$ indexov premenných úlohy v kanonickom tvare. Ďalej je ešte k dispozícii optimálna simplexová tabuľka $(\boldsymbol{A}^p|\boldsymbol{b}^p)$ s rozmermi $(m+1) \times (n+1)$. V tomto prípade

platí, že $\underline{x}_{B^p(i)} = b_i^p$ pre $i = 0,\ldots,m$ a $\underline{x}_{N^p(i)} = 0$ pre $i = 1,\ldots n-m$. Tvar podmienok sústavy pre $i = 0,\ldots,m$ bude:

$$\sum_{j\in N^p} a_{ij}^p x_j + x_{B^p(i)} = b_i^p \tag{2.1}$$

Pre prípad duálnej lexigografickej simplexovej metódy máme okrem optimálneho riešenia $\underline{\boldsymbol{x}} \in E_{n+1}$ maximalizujúceho účelovú funkciu aj zoznam bázických $\boldsymbol{B}^p$ a nebázických $\boldsymbol{N}^p$ indexov premenných úlohy v kanonickom tvare a optimálnu lexikograficky normálnu tabuľku$\left(\underline{\boldsymbol{b}}^p \middle| \underline{\boldsymbol{A}}^p\right)$ s rozmermi $(n+1)\times(n-m+1)$. Tu platí $\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}^p$ a tvar podmienok sústavy pre $i \in \boldsymbol{B}^p$ bude:

$$\sum_{j\in N^p} \underline{a}_{ij}^p x_j + x_i = \underline{b}_i^p \tag{2.2}$$

Pre zostrojenie rezu potrebujeme zadefinovať celú časť reálneho čísla a, ktorá bude predstavovať najväčšie celé číslo menšie alebo rovné číslu a, označovaná ako $\lfloor a \rfloor$. Ďalej zápis $\{a\}$ bude označovať zlomkovú časť čísla a definovanú ako rozdiel $a - \lfloor a \rfloor$. Z tejto definície vidíme, že pre každé reálne číslo a platí $\{a\} \in \langle 0,1)$.

Nech $(\boldsymbol{A}^p|\boldsymbol{b}^p)$ resp. $\left(\underline{\boldsymbol{b}}^p \middle| \underline{\boldsymbol{A}}^p\right)$ je optimálna tabuľka algoritmu primárnej resp. duálnej simplexovej metódy s optimálnym bázickým prípustným riešením $\underline{\boldsymbol{x}}$ a nech $\underline{x}_i$ pre $i \in \boldsymbol{B}^p$ nie je celé číslo. Potom podmienka

$$\sum_{j\in N^p} -\{a_{\mathrm{ij}}^p\} x_j \leq \{b_i^p\} \tag{2.3}$$

respektíve v prípade duálnej simplexovej metódy

$$\sum_{j\in N^p} -\{\underline{a}_{\mathrm{ij}}^p\} x_j \leq \{\underline{b}_i^p\} \tag{2.4}$$

bude normálnym rezom [24].

Ďalej si uvedieme prvý Gomoryho algoritmus.

Krok 1: Inicializácia. Pre LP-relaxáciu úlohy celočíselného programovania zostavíme lexigkograficky kladnú tabuľku lexikografickej duálnej simplexovej metódy a choď na krok 1.

Krok 2: Riešenie spojitej úlohy lineárneho programovania. V tomto kroku budeme opakovane používať pivotovú transformáciu algoritmu duálnej lexikografickej metódy, až kým nenastane jeden z prípadov:

a. V pivotovom riadku matice $\underline{\boldsymbol{A}}^p$ nie je možné vybrať pivota z dôvodu nezápornosti všetkých prvkov. V tomto prípade ukončíme algoritmus z dôvodu neexistencie celočíselného riešenia.

b. Bolo nájdené optimálne riešenie $\underline{\boldsymbol{x}}$ súčasnej úlohy s bázou $\boldsymbol{B}^p$ a tabuľkou $\left(\underline{\boldsymbol{b}}^p \middle| \underline{\boldsymbol{A}}^p\right)$. V tomto prípade prejdi na krok 2.

Krok 3: Konštrukcia rezu. Ak optimálne riešenie $\underline{\boldsymbol{x}}$ je celočíselné tak konči, pretože zložky riešenia $\underline{\boldsymbol{x}}$, ktoré odpovedajú premenným vstupnej úlohy celočíselného programovania, tvoria jej optimálne riešenie. Inak vyber najmenší index r neceločíselnej zložky $\underline{\boldsymbol{x}}$ podľa vzťahu

$$r = \min\{i = 0, \dots, n: \lfloor \underline{x}_j \rfloor \neq \underline{x}_i\} \tag{2.5}$$

Ďalej zaveď novú doplnkovú premennú x_k pre reznú podmienku v tvare

$$x_k = -\{\underline{b}_r^p\} + \sum_{j \in N^p} -\left\{\underline{a}_{r_j}^p\right\}(-x_j) \tag{2.6}$$

Krok 4: Uprav sústavu $\left(\underline{\boldsymbol{b}}^p \middle| \underline{\boldsymbol{A}}^p\right)$ pridaním riadku $n+1$ s koeficientami $-\{\underline{b}_r^p\}$ a $-\{\underline{a}_{rj}^p\}$ pre $j \in N^p$ a s rozšírenou tabuľkou pokračuj na krok 1.

2.3.2 Metóda vetiev a hraníc

Táto metóda [25] je jednou z univerzálnych metód na riešenie úloh celočíselného lineárneho programovania a dá sa použiť aj na úlohy zmiešaného celočíselného programovania. Princípom metódy je postupné spracovanie celej množiny riešení vo forme čiastočne usporiadaného stromu, kde jeho koreňom je celá množina riešení. Ďalšie vetvy stromu tvoria podmnožiny celej množiny riešení.

Algoritmus funguje tak, že postupne sa vytvára strom riešení a postupne sa spracovávajú jeho vetvy. Na začiatku algoritmu máme iba jednu vetvu, a to koreň stromu – celú množinu riešení. Prvým krokom algoritmu je výber vetvy zo zoznamu ešte nespracovaných vetiev. Táto vybraná vetva sa následne spracuje, pričom počas spracovania môžu z tejto vetvy vzniknúť nové, ešte nespracované vetvy alebo sa nájde najlepšie celočíselné v danej vetve, respektíve sa zistí, že optimálne riešenie úlohy daná vetva neobsahuje, čo vyústi do vyradenia danej vetvy z ďalšieho spracovania. Okrem toho je nutné

si značiť doposiaľ najlepšie nájdené riešenie počas celého prehľadávania, ktoré sa počas behu aktualizuje, ak sme našli lepšie riešenie počas spracovávania vetvy.

Počas spracovania vetvy je nutné určiť dve hodnoty. Sú nimi dolná a horná hranica, ktoré reprezentujú hodnotu účelovej funkcie pre doposiaľ najlepšie nájdené celočíselné riešenie a potenciálnu hodnotu účelovej funkcie pre najlepšie možné riešenie v danej vetve. Pre prípad minimalizácie je dolnou hranicou potenciálna hodnota účelovej funkcie v danej vetve a horná hranica je účelovou funkciou doposiaľ najlepšieho nájdeného celočíselného riešenia.

Ak sa počas spracovania vetvy zistí, že doposiaľ najlepšie nájdené celočíselné riešenie (horná hranica) má hodnotu účelovej funkcie menšiu alebo rovnú dolnej hranici v danej vetve, tak môže byť spracovávaná vetva z prehľadávania vylúčená, pretože najlepšie možné riešenie danej vetvy je určite horšie ako doposiaľ najlepšie nájdené celočíselné riešenie, a teda daná vetva nemôže obsahovať optimálne riešenie.

Pokiaľ je horná hranica (doposiaľ najlepšie nájdené celočíselné riešenie) menšia ako dolné hranice vo všetkých ešte nespracovaných vetvách, môžeme algoritmus ukončiť, nakoľko doposiaľ najlepšie nájdené celočíselné riešenie je aj optimálnym riešením úlohy, pretože v žiadnej vetve neexistuje lepšie riešenie.

Pokiaľ sme vetvu nevylúčili z prehľadávania, tak sa táto vetva rozdelí (vetvenie) na niekoľko podmnožín - vetiev, ktoré sú následne pridané do zoznamu ešte nespracovaných vetiev. Zároveň aktualizujeme hodnotu účelovej funkcie doposiaľ najlepšieho nájdeného celočíselného riešenia (hornú hranicu), ak sme v danej vetve našli celočíselné prípustné riešenie, ktoré má nižšiu hodnotu účelovej funkcie ako horná hranica.

Algoritmus končí v okamihu, keď sme spracovali všetky vetvy zo zoznamu ešte nespracovaných vetiev. V tomto okamihu môžeme povedať, že doposiaľ najlepšie nájdené celočíselné riešenie je aj optimálnym riešením úlohy. [25]

Na získanie konkrétnej implementácie algoritmu je potrebné špecifikovať kroky ako sú poradie spracovávania vetiev zo zoznamu ešte nespracovaných vetiev – schéma prehľadávania stromu riešení, metódu na nájdenie celočíselného prípustného riešenia v danej vetve, metódu na výpočet dolnej hranice v spracovávanej vetve a spôsob vetvenia.

Výpočet dolnej hranice

Výpočet dolnej hranice je veľmi dôležitý, pretože presnosť dolnej hranice každej vetvy môže pomôcť pri rýchlejšom rozhodovaní, či danú vetvu ďalej preskúmavať alebo v nej nie je možné nájsť lepšie riešenie a je možné ju vyradiť z prehľadávania. Pri výpočte dolnej hranice sa veľmi často používa LP-relaxácia úlohy a jej riešenie simplexovou metódou, čím dostaneme najlepšie možné riešenie v prípade neceločíselnej úlohy. Takisto je možné na zistenie dolnej hranice použiť metódu rezných nadrovín spomínanú vyššie, čím dostávame metódu vetiev a rezov.

Výpočet dolnej hranice môže závisieť aj od konkrétneho problému a využívať špecifiká úlohy na určenie presnejšej dolnej hranice. Napríklad na riešenie úlohy návrhu turnusov vozidiel s viacerými depami bol v článku [23] predstavený algoritmus, ktorý zlepšuje výpočet dolnej hranice. Riešil sa na úlohe popísanej nasledujúcim modelom.

Autori uvažujú orientovaný graf $G = (V, A)$, kde množina vrcholov obsahuje dve podmnožiny, $W = 1, \ldots, m$ kde každý vrchol predstavuje jedno depo a $N = m + 1, \ldots, m + n$, kde jeden vrchol reprezentuje jeden spoj. Každá hrana (i, j), kde $i, j \in N$ reprezentuje prechod medzi spojmi i a j a hrana (i, j), kde $i \in W$ (resp. $j \in W$) predstavuje začiatok (resp. koniec) turnusu. Ďalej je definovaná cena hrany c_{ij} pre hranu (i, j). Premenná x_{ij} bude 1, ak hrana (i, j) bude použitá, inak 0.

$$min \qquad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.7}$$

$$z.p. \quad \sum_{i \in V} x_{ij} = r_j \quad \forall\, j \in V \tag{2.8}$$

$$\sum_{j \in V} x_{ij} = r_i \quad \forall\, i \in V \tag{2.9}$$

$$\sum_{(i,j) \in P} x_{ij} \leq |P| - 1 \quad \forall\, P \in \Pi \tag{2.10}$$

$$x_{ij} \in Z^+; i, j \in V \in Z^+ \quad \forall\, i, j \in V \tag{2.11}$$

Účelová funkcia (2.7) minimalizuje celkové náklady. Podmienky (2.8) a (2.9) zabezpečujú, že každý vrchol bude navštívený nanajvýš r_i-krát, kde $r_i = 1$ pre $i \in N$. Podmienky (2.10) zakazujú nemožné turnusy, pričom Π predstavuje množinu zahrnutých minimálnych neprípustných ciest, napr. priame cesty medzi dvomi rôznymi depami.

Obligatórna podmienka (2.11) zabezpečuje celočíselnosť a nezápornosť premennej x_{ij}. Spolu s podmienkami (2.8) a (2.9) je jasné, že $x_{ij} \in (0,1)$ pre $i,j \in N$.

Pri výpočte dolnej hranice je využitá LP relaxácia úlohy len s podmienkami (2.8) a (2.9). Následne sa kvôli splneniu podmienky (2.10) spúšťajú separačné procedúry, ktoré nahradzujú splnenie podmienok (2.10). Počas každého kola separácie sa kontrolujú podmienky LPI (Lifted path inequalities) a PEC (Path elimination constraints). Metóda PEC vytvára nové podmienky pre nahradenie podmienok (2.10) tak, že na základe aktuálneho riešenia zakazuje také turnusy, ktoré začínajú a končia v rôznych depách. Samotný separačný algoritmus sa snaží čo najrýchlejšie nájsť čo najviac obmedzení, ktoré boli porušené vyprodukovaným riešením LP-relaxácie úlohy.

Metóda LPI pridáva ďalšie podmienky k LP-relaxácii úlohy na základe aktuálneho riešenia, ktoré môže byť vyjadrené ako lineárna kombinácia charakteristických vektorov prípustných turnusov. Tieto podmienky sú podmienky typu prvého Gomoryho rezu, popísaného pri metóde rezných nadrovín.

Schéma prehľadávania

Schéma prehľadávania predstavuje postup, v akom poradí budeme spracovávať ešte nespracované vetvy. Jeden zo spôsobov je usmernené prehľadávanie, kde sa vyberá najprv vetva s najnižšou hodnotou dolnej hranice.

Ďalšou alternatívou je prehľadávanie do hĺbky. V tomto prípade sa určí nejaké elementárne pravidlo podľa ktorého sa nájde prioritná vetva na spracovanie. Príkladom by mohlo byť pravidlo, že vždy budeme prehľadávať ľavú vetvu, až kým nedosiahneme odrezanie vetvy z dôvodu neexistencie riešenia LP-relaxácie alebo z dôvodu, že dolná hranica vetvy je väčšia alebo rovná hornej hranici úlohy. Podmienky pre vetvy budeme ukladať do zásobníka. V prípade odrezania vetvy sa vrátime o úroveň vyššie, čiže vyberieme poslednú podmienku zo zásobníka a pridáme podmienku pre pravú vetvu a pokračujeme vo vykonávaní algoritmu, pričom sa vraciame k elementárnej podmienke [25].

Výpočet hornej hranice

Výpočet hornej hranice zabezpečuje nájdenie celočíselného riešenia pre danú vetvu, ktoré sa môže stať hornou hranicou celej úlohy, ak je lepšie než doteraz najlepšie nájdené celočíselné riešenie. Čím kvalitnejšia metóda je použitá na nájdenie celočíselného riešenia,

tým skôr sa môže stretnúť dolná a horná hranica vetvy, čím ju môžeme vylúčiť z prehľadávania.

Na túto úlohu sa veľmi často využívajú heuristické metódy, ktoré vedia získať pomerne dobré riešenie v krátkom čase. Napríklad na riešenie úlohy popísanej modelom (2.7)-(2.11) sa použila heuristika zložená z konštrukčnej heuristiky založenej na hľadaní najkratšej cesty s predefinovanými cenami hrán nasledovaná zlepšovacími procedúrami [23].

Vetvenie

Systém vetvenia popisuje, akým spôsobom sa bude strom riešení vetviť. Najčastejšie sa používa vetvenie na dve rôzne vetvy, rozdelené podmienkou pridanou k LP-relaxácii modelu na základe aktuálneho riešenia. Pri celočíselných úlohách sa vetvy delia na základe premenných, ktoré nemajú celočíselnú hodnotu. V prípade úloh, kde premenná môže nadobúdať iba dve hodnoty sa môže pridať podmienka, kde sa vybraná premenná fixuje na jednu z možných hodnôt (Kolesárov algoritmus).

Samozrejme podmienka na výber premennej podľa ktorej sa bude vetviť môže byť pomerne komplexná. Napríklad v prípade [23] sa toto kritérium skladá z piatich kritérií. Prvým je že sa vyberá premenná reprezentujúca hranu (a, b), ktorá nadobúda hodnoty v intervale $x_{ab} \in (0{,}4; 0{,}6)$. Druhé kritérium vyberie najprv tú premennú x_{ab}, ktorá nebola celočíselná ani v predchádzajúcej nadradenej vetve. Tretie kritérium vyberie tú premennú x_{ab}, kde vrchol a v hrane (a, b) je možné navštíviť z depa po hranách, ktoré majú v riešení hodnotu 1. Štvrté kritérium je podobné tretiemu, ale tentokrát v smere od vrcholu b k depu. Piatym kritériom je výber takej hrany (a, b), ktorá je často vyberaná počas výpočtu hornej hranice pomocou heuristiky.

2.3.3 Metóda generovania stĺpcov

Ďalšou z metód na riešenie úloh zmiešaného programovania je metóda generovania stĺpcov [27]. Táto metóda sa používa na riešenie úloh lineárneho programovania. Základnou myšlienkou je, že väčšina úloh lineárneho programovania je príliš veľká na to, aby sme explicitne zahrnuli všetky možné rozhodovacie premenné. Predpokladom na použitie je, že väčšina premenných bude nebázických a preto bude mať hodnotu nula v optimálnom riešení. Vďaka tomu budeme potrebovať na riešenie problému iba malú množinu premenných. Základom metódy generovania stĺpcov je generovanie iba tých premenných, ktoré majú potenciál zlepšiť riešenie, čiže účelovú funkciu. Budeme teda generovať premenné (stĺpce

v simplexovej matici), ktoré majú záporné redukované ceny, za predpokladu že riešime minimalizačnú úlohu.

Počas riešenia sa úloha rozdelí na dve úlohy - obmedzený hlavný problém (restricted master problem - RMP) a pod-problém. V obmedzenom hlavnom probléme sa rieši pôvodná úloha iba na malej množine možných riešení. Pod-problém je úloha nájdenia vhodnej premennej, ktorá by po pridaní k hlavnému problému mohla zlepšiť účelovú funkciu. Pre účelovú funkciu pod-problému platí, že je to redukovaná cena novej premennej, pričom musíme brať do úvahy aktuálne duálne premenné a zároveň musíme splniť prirodzené obmedzenia riešenia reprezentované premennou.

Metóda generovania stĺpcov pracuje iteratívne, až kým nenájde optimálne riešenie LP-relaxácie hlavného problému, respektíve nie je zastavená iným kritériom, napríklad čas na výpočet alebo nájdenie dostatočne dobrého riešenia.

Prvým krokom algoritmu je vyriešenie obmedzeného hlavného problému. Ďalej z tohto riešenia získame hodnoty duálnych cien pre každé obmedzenie hlavej úlohy. Tieto informácie použijeme pri tvorbe účelovej funkcie pod-problému. Následne sa vyrieši pod-problém a pokiaľ je jeho účelová funkcia záporná, existuje premenná, ktorá by mohla zlepšiť účelovú funkciu hlavného problému. Následne je táto premenná pridaná k hlavnému problému a proces opäť pokračuje riešením hlavnej úlohy s pridanou premennou. Po vyriešení dostávame novú sadu duálnych hodnôt a pokračuje sa v riešení pod-problému. Tento proces sa opakuje až kým v pod-probléme neidentifikujeme novú premennú, čo nastane, ak redukovaná cena bude nezáporná. Pokiaľ sme dosiahli tento stav, tak žiadna premenná už nemá potenciál zlepšiť účelovú funkciu a môžeme konštatovať, že riešenie obmedzeného hlavného problému je optimálnym riešením LP-relaxácie pôvodnej úlohy.

Pre lepšie pochopenie si ukážeme príklad riešenia základnej úlohy návrhu turnusov pomocou metódy generovania stĺpcov.

Máme teda množinu spojov N, ktoré je potrebné obslúžiť. Našou úlohou je navrhnúť turnusy pre jednotlivé použité vozidlá s cieľom minimalizovať počet použitých vozidiel. Pri metóde generovania stĺpcov však nebudeme vytvárať samotné turnusy ako v prípade matematického modelu (1.1)-(1.4), ale využijeme inú perspektívu na problém, kde budeme vyberať, ktoré turnusy z množiny všetkých turnusov sa majú použiť, aby bola splnená podmienka, že budú všetky spoje obslúžené.

Majme teda množinu S, ktorá obsahuje všetky možné prípustné turnusy, pričom prípustný turnus je taký, kde nie sú v rámci turnusu obsluhované dva paralelné spoje. Zároveň musí byť splnená podmienka, že medzi dvomi za sebou nasledujúcimi spojmi musí byť dostatočný čas na to, aby sa vozidlo presunulo medzi koncovou zastávkou prvého spoja a začiatočnou zastávkou druhého spoja.

Premenná x_s reprezentuje rozhodnutie, či sa daný turnus s použije v celkovom návrhu turnusov. Konštanta a_{si} reprezentuje samotný turnus s, pričom hovorí, či bol spoj i počas turnusu s obslúžený. Našou úlohou je zabezpečiť, aby bol každý spoj obslúžený. Zároveň minimalizujeme počet použitých vozidiel tým, že minimalizujeme počet vybraných turnusov, keďže jeden vybraný turnus zodpovedá použitiu jedného vozidla na jeho obsluhu.

$$min \qquad \sum_{s \in S} x_s \tag{2.12}$$

$$z.p. \qquad \sum_{s \in S} x_s a_{si} = 1 \qquad \forall i \in N \tag{2.13}$$

$$x_s \in \{0,1\} \qquad \forall s \in S \tag{2.14}$$

Keď sa pozrieme na simplexovú maticu pri riešení úlohy matematickým modelom (2.12)-(2.14), tak každý stĺpec tejto matice predstavuje jeden turnus. Avšak počet všetkých prípustných turnusov je obrovský. Preto na riešenie použijeme metódu generovania stĺpcov.

V prvom kroku si vytvoríme redukovaný hlavný problém, čiže budeme riešiť pôvodnú úlohu na redukovanej množine turnusov P tak, aby bol problém riešiteľný, a zároveň relaxujeme podmienku binárnej premennej x_s. Dostávame teda:

$$min \qquad \sum_{s \in P} x_s \tag{2.15}$$

$$z.p. \qquad \sum_{s \in P} x_s a_{si} = 1 \qquad \forall i \in N \tag{2.16}$$

$$x_s \geq 0 \qquad \forall s \in P \subseteq S \tag{2.17}$$

K nej duálna úloha s duálnymi premennými π_i bude:

$$max \qquad \sum_{i \in N} \pi_i \tag{2.18}$$

$$z.p. \qquad \sum_{i \in N} \pi_i a_{si} \leq 1 \quad \forall s \in P \tag{2.19}$$

$$\pi_i \in \mathrm{R} \qquad \forall i \in \mathrm{P} \tag{2.20}$$

Po vyriešení tejto úlohy dostávame jej optimálne riešenie. Ďalším krokom je nájdenie takého ďalšieho turnusu, ktorý by mohol zlepšiť riešenie. Táto úloha je naším pod-problémom. S ohľadom na optimálne riešenie duálnej úlohy $\bar{\boldsymbol{\pi}}$, redukovaná cena nového pridaného stĺpca (turnusu) $r \in S$ a $r \notin P$ bude

$$c_r = 1 - \sum_{i \in N} a_{ri} \bar{\pi}_i \,. \tag{2.21}$$

Čiže, budeme hľadať taký stĺpec (turnus), ktorý bude mať zápornú redukovanú cenu $c_r < 0$. Redukovaná cena turnusu sa dá vyjadriť aj ako súčet ohodnotení použitých hrán turnusu predstavujúcich prejazdy medzi jednotlivými spojmi. Čiže

$$c_r = 1 - \sum_{i \in N} a_{ri} \bar{\pi}_i = 1 - \sum_{i \in N} \sum_{j \in N} y_{ij} \bar{\pi}_i \,. \tag{2.22}$$

Pri minimalizácii redukovanej ceny turnusu budeme teda hľadať najkratšiu cestu medzi ranným a večerným depom, pričom ceny hrán sú všeobecné. V rámci pod-problému riešime teda úlohu popísanú nasledujúcim matematickým modelom:

Môžeme ju hľadať napríklad pomocou modelu:

$$min \qquad 1 - \sum_{i \in N} \sum_{j \in F(i)} y_{ij} \bar{\pi}_i \tag{2.23}$$

$$z.p. \qquad \sum_{j \in F(D_0)} y_{D_0 j} = 1 \tag{2.24}$$

$$\sum_{i \in B(j)} y_{ij} = \sum_{l \in F(j)} y_{jl} \qquad \forall\, j \in N \tag{2.25}$$

$$z.p. \qquad y_{ij} \in \{0,1\} \qquad \forall i \in N, j \in F(i) \tag{2.26}$$

Kde y_{ij} je rozhodnutie, či po obslúžení spoja i sa vozidlo presunie na obsluhu spoja j. Množina N je množina spojov. Množina $F(i)$ obsahuje spoje nasledujúce po spoji i, pričom platí, že tieto spoje je možné obslúžiť po spoji i vzhľadom na časovú následnosť a patrí k nim aj večerné depo. Množina $B(i)$ je množina predchádzajúcich spojov k spoju i, to znamená, že je možné obslúžiť spoj i po spoji z tejto množiny vzhľadom na časovú

následnosť a patrí k nim aj ranné depo. Podmienka (2.24) nám definuje, že z ranného depa D_0 vyrazí práve jedno vozidlo a podmienky (2.25) zabezpečujú, že ak vozidlo príde obslúžiť spoj j, tak aj zo spoja j odíde na ďalší spoj, prípadne do večerného depa.

Po vyriešení úlohy pod-problému zostáva už iba transformovať riešenie pod-problému $\boldsymbol{y} = (y_{11}, y_{12}, \dots, y_{1n}, \dots, y_{nn})$ na reprezentáciu stĺpca (turnusu) $\boldsymbol{a_r} = (a_{r1}, a_{r2}, \dots, a_{rn})$. Dosiahneme to tak, že pre každý spoj i, ktorý bol navštívený počas turnusu reprezentovaným riešením $\boldsymbol{y}$ priradíme v stĺpci $\boldsymbol{a_r}$ číslo 1, a pre nenavštívené číslo 0. Inak povedané

$$a_{ri} = \sum_{j \in B(i)} y_{ji} \qquad \forall i \in N \tag{2.27}$$

Riešenie pod-problému, nový turnus, pridáme do množiny P obmedzenej hlavnej úlohy a opakujeme proces riešenia. Ak sa počas riešenia pod-problému dostaneme do situácie, že nový turnus bude mať nezápornú redukovanú cenu, znamená to, že už ďalší turnus, ktorý by zlepšil riešenie neexistuje a riešenie obmedzeného hlavného problému môžeme prehlásiť za optimálne riešenie LP-relaxácie pôvodnej úlohy. Následne už zostáva iba vyriešiť pôvodnú úlohu na obmedzenej množine turnusov P, čím získame konečné riešenie úlohy návrhu tunosov.

Je však potrebné si uvedomiť, že napriek optimálnemu riešeniu jednotlivých úloh obmedzeného hlavného problému a pod-problému, metóda generovania stĺpcov nezaručuje získanie optimálneho riešenia úlohy návrhu turnusov definovanej hlavným problémom riešeným na obmedzenej množine turnusov P.

2.4 Heuristické metódy

Heuristika je metóda hľadania riešenia, ktorá nie je presná, ani nezaručuje nájdenie optimálneho riešenia, ale nemusí byť až tak časovo náročná ako exaktné metódy. Je založená na nejakých logických predpokladoch ohľadom danej úlohy a snaží sa priblížiť k optimálnemu riešeniu, aj keď ho zrejme nikdy nedosiahne, dostane sa len k sub-optimálnemu riešeniu (približujúcemu sa k optimálnemu riešeniu). Samozrejme sa dá tento prístup využiť na riešenie úlohy návrhu turnusov elektrických autobusov ako bolo predstavené v [21][22][70].

2.4.1 Prosté heuristiky

Heuristiky môžeme rozdeliť na primárne a duálne. Primárne heuristiky začínajú v prípustnom riešení a prechádzajú vždy k ďalšiemu prípustnému riešeniu, pričom hodnota lokálneho kritéria (obyčajne zhodná s účelovou funkciou) nového riešenia bude menšia ako hodnota lokálneho kritéria v predchádzajúcom riešení. Končí v prípade, keď sa už nedá prejsť k riešeniu s lepšou hodnotou lokálneho kritéria [25].

Duálna heuristika naopak začína v neprípustnom riešení a prechádza k riešeniu s menšou mierou neprípustnosti s predpokladom, že sa hodnota lokálneho kritéria zvýšila čo najmenej. Končí v momente, keď už sa nedá prejsť k riešeniu s menšou mierou neprípustnosti alebo prejdením k prípustnému riešeniu [25].

Pri konštrukciách heuristických metód sa zvyčajne oba postupy kombinujú tak, že duálna heuristika poskytne východiskové riešenie a následne primárna heuristika toto riešenie ďalej zlepšuje.

Heuristiky napriek svojej rýchlosti nedokážu prehľadávať širšiu množinu riešení. Po ukončení skončia v nejakom lokálnom minime a toto minimum už nedokážu zlepšiť, pretože z princípu heuristiky vyplýva, že musí riešenie iba zlepšovať, čo vylučuje zhoršenie lokálneho kritéria za účelom prehľadania širšej množiny riešení. Tento problém riešia metódy nazývané metaheuristiky.

2.4.2 Metaheuristiky

Metaheuristiky [29] sú metódy na optimalizáciu, ktoré využívajú princípy prostých heuristík, ale za určitých okolností môžu prejsť aj k horšiemu riešeniu, čím môžu preskúmať širšiu časť množiny prípustných riešení a tým dosiahnuť iné lokálne minimá, ktoré môžu viesť k zlepšeniu riešenia, aj keď sú to stále len sub-optimálne riešenia.

Každá metaheuristika je definovaná krokom a zoznamom prípustných operácií, ktoré popisujú prechod od jedného riešenia k druhému. Tieto operácie sa líšia medzi jednotlivými metaheuristikami a zároveň často závisia od typu riešenej úlohy.

Poznáme niekoľko prístupov v metaheuristikách. Jedným je postupné prehľadávanie okolia súčasného riešenia, čím sa vytvára reťaz riešení. Príkladom sú metaheuristiky simulated annealing a tabu search. Druhým používaným prístupom sú metaheuristiky založené na princípe spracovávania populácie, ktoré vytvárajú množiny riešení a pracujú s celou populáciou. Asi najznámejším príkladom je genetický algoritmu, ktorý si berie za vzor evolúciu a kríženie génov.

2.4.3 Metaheuristiky založené na spracovávaní okolia

Tieto metaheuristiky pracujú s tzv. okolím súčasného riešenia. Okolie je množina riešení, ku ktorým môžeme zo súčasného riešenia prejsť jediným krokom, čiže vykonaním jednej operácie. Je teda určená súčasným riešením a množinou povolených operácií danej metaheuristiky. Povolená operácia je napríklad vloženie nejakého prvku do riešenia alebo výmena dvoch prvkov v rámci riešenia [39].

Ako aj názov skupiny metaheuristík napovedá, sú špecifické tým, že spracovávajú okolie súčasného riešenia. Spôsob, akým je okolie spracovávané, sa líši na základe heuristiky a určuje poradie v akom sa preberú jednotlivé riešenia z okolia, respektíve, či ich prebrať všetky [39].

Simulated annealing

Táto metaheuristika [29] prehľadáva okolie súčasného riešenia stratégiou prvý vhodný, čiže postupne prechádza okolie a prejde k nasledujúcemu riešeniu, ak je hodnota účelovej funkcie nasledujúceho riešenia menšia ako hodnota účelovej funkcie aktuálneho riešenia. Počas spracovávania okolia sa však riešenia s horšou účelovou funkciou nezahadzujú, ale vykonáva sa náhodný pokus s určitou pravdepodobnosťou, ktorý rozhodne o tom, či prejdeme k horšiemu riešeniu.

Pravdepodobnosť prechodu musí spĺňať podmienku, že je tým menšia, čím je väčšie zhoršenie účelovej funkcie. Najčastejšie sa na výpočet pravdepodobnosti prechodu využíva vzťah $p = e^{-(f(x)-f(x^c))/t}$, kde x^c je súčasné riešenie, x je nové riešenie ku ktorému chceme prejsť, $f(x)$ a $f(x^c)$ sú hodnoty účelových funkcií nového a súčasného riešenia a t je parameter metaheuristiky [39][40].

Parameter $t > 0$ sa nazýva teplota. Na začiatku je tento parameter nastavený na hodnotu t^{max}, ktorá je dostatočne vysoká. Tento parameter sa snaží zvýšiť pravdepodobnosť prechodu, čím podporuje opustenie lokálneho minima. Zároveň sa počas behu algoritmu zvyčajne po určitom počte krokov znižuje podľa zadefinovaného vzťahu. Teplota je znižovaná z dôvodu postupného ustaľovania riešenia v nejakom minime. Proces metaheuristiky končí v momente dosiahnutia niektorého kritéria zastavenia [39][40].

Počas behu heuristiky sa uchováva najlepšie nájdené riešenie, ktoré metaheuristika po skončení prezentuje ako výstup.

Takisto je možné túto metaheuristiku rozšíriť o tzv. zahrievanie, čo je proces pri ktorom sa teplota opätovne nastaví na maximálnu hodnotu t^{max}, čo podporí šancu opustiť aktuálne lokálne minimum.

2.4.4 Metaheuristiky založené na spracovávaní populácie

Metaheuristiky spracovávajúce populáciu nepoznajú pojem ako „súčasné riešenie“ používaný v prostých minimalizačných heuristikách alebo metaheuristikách založených na spracovávaní okolia. Tieto metaheuristiky pracujú súčasne s množinou riešení nazývanou populácia. Neprechádzajú od jedného riešenia k ďalšiemu, ale súčasná populácia sa mení na nasledujúcu a po zmene sa aktualizuje najlepšie nájdené riešenie. Populačné metaheuristiky sa často inšpirujú prírodou a jej zákonitosťami.

Genetický algoritmus

Genetický algoritmus (GA) je populačnou metaheuristikou, ktorá je inšpirovaná evolúciou organizmov. GA pracuje s populáciou riešení (chromozómy) pomocou dvoch operácií, ktoré reprezentujú zmeny v organizmoch počas ich vývoja. Prvou operáciou je kríženie, ktoré reprezentuje rozmnožovanie jedincov, pri ktorom vznikajú nové jedince. Operácia teda vyberá niektoré chromozómy ako rodičov a z nich sa následne vytvárajú potomkovia. Druhou operáciou je mutácia, ktorá reprezentuje jednoduché zmeny v jedincoch. Pri mutácii sa chromozóm nejakou operáciou čiastočne upraví, pričom vznikne nové riešenie [29][53]-[55].

Celkovo algoritmus funguje na princípe striedania populácií, čiže v jednom okamihu máme k dispozícii len jednu populáciu, z ktorej sa vytvorí nasledujúca populácia, ktorá ju nahradí. Zároveň sa počas celého behu algoritmu udržiava informácia o najlepšom nájdenom riešení, ktorá sa priebežne aktualizuje, ako sa vytvárajú nové riešenia a vymieňajú populácie.

Počiatočná fáza algoritmu – inicializácia, zahŕňa vytvorenie prvotnej populácie pomocou nejakého algoritmu do veľkosti populácie *PopSize*. Táto veľkosť populácie sa počas behu algoritmu nemení.

Nasleduje fáza, v ktorej sa vytvára nová populácia. Tá sa vytvára postupnou tvorbou nových riešení pomocou metódy kríženia a následne sa upraví pomocou metódy mutácie. Tieto operácie sa opakujú, až kým nedosiahneme vytvorenie dostatočného množstva riešení reprezentujúcich novú populáciu.

Kríženie je operácia ktorá pozostáva z výberu rodičov zo súčasnej populácie, z ktorých sa následne vytvoria potomkovia do novej populácie. Výber rodičov, alebo selekcia, sa implementuje rôznymi spôsobmi. Môže sa robiť náhodným výberom alebo špecifickými algoritmami ako napríklad turnajový algoritmus alebo metóda rankingu, pri ktorých majú kvalitnejšie riešenia vyššiu šancu stať sa rodičmi. Druhým krokom je samotné kríženie, čiže operácia, ktorá skombinuje niekoľko (zvyčajne dve) riešenia za účelom vytvorenia nového riešenia, prípadne niekoľkých riešení. Samozrejme je nutné, aby novovytvorené riešenia spĺňali všetky podmienky spojené s prípustnosťou riešenia. Preto je táto operácia veľmi závislá od typu úlohy, ktorá sa rieši genetickým algoritmom.

Mutácia je ďalším krokom v tvorbe novej populácie, kde sa mení riešenie vytvorené pomocou kríženia. Zmena, ktorú vykonáva mutácia, nemusí úplne zmeniť charakter riešenia. Stačí, ak čiastočne pozmení riešenie, čo spôsobí malú zmenu a presunie sa do okolia riešenia vstupujúceho do mutácie.

Po vytvorení dostatočného počtu nových riešení pomocou kríženia a následnej mutácie máme k dispozícii novú populáciu, ktorá nahradí tú aktuálnu. Tento postup sa opakuje, až kým nenastane podmienka zastavenia.

3 Ciele a metodika práce

Témou dizertačnej práce je riešenie úlohy návrhu turnusov elektrických autobusov vo verejnej doprave. Táto úloha je špecifickým prípadom úlohy návrhu turnusov vozidiel, pričom musíme brať ohľad na obmedzený dojazd elektrického autobusu, ako aj na dĺžku času nabíjania. Pri riešení tejto úlohy budú zadané presné umiestnenia nabíjacích staníc, čiže nebudeme riešiť ich umiestňovanie. Pre nabíjanie elektrických autobusov bola zvolená technológia priebežného nabíjania, kedy sa vozidlo môže nabíjať len medzi jednotlivými spojmi na nabíjačkách rozmiestnených v dopravnej sieti.

Výsledkom riešenia danej úlohy je vytvorenie turnusov elektrických autobusov, ktoré budú určovať, ktoré spoje budú obsluhované daným elektrickým autobusom. Okrem toho bude nutné určiť aj rozvrh nabíjania elektrických autobusov na nabíjačkách a množstvo elektrickej energie, ktorá bude počas nabíjacej udalosti doplnená.

Prvým cieľom dizertačnej práce je formulácia lineárneho matematického modelu, nakoľko neexistuje štandardná formulácia pre riešenie úlohy návrhu turnusov elektrických autobusov.

Druhým cieľom je overenie riešiteľnosti navrhnutého modelu pomocou štandardného IP solvera.

Tretím cieľom je návrh vlastných metód založených na exaktných a heuristických prístupoch. V prípade návrhu vlastnej exaktnej metódy budeme využívať princípy na ktorých je postavená metóda vetiev a hraníc, respektíve metóda generovania stĺpcov. V prípade heuristických metód budeme využívať princípy existujúcich heuristických a metaheuristických metód.

Pre každú z metód bude potrebné

- navrhnúť niekoľko možných algoritmov a implementovať ich,
- preskúmať správanie z hľadiska presnosti a efektivity vzhľadom k presnému riešeniu, resp. k dolnému odhadu riešenia, ako aj časovú náročnosť konkrétnej metódy,
- preskúmať správanie v závislosti na nastavení parametrov algoritmu a na základe analýzy navrhnúť odporúčania pre nastavenie parametrov algoritmu pre čo najefektívnejšie riešenie úlohy.

Pre otestovanie navrhnutých algoritmov k jednotlivým metódam sa musia vykonať experimenty na dostatočne veľkej množine referenčných úloh, ktoré budú prebrané z reálnych dát. Experimenty poslúžia na zistenie kvality poskytovaných riešení, časovú náročnosť algoritmu, ako aj na zistenie maximálneho rozsahu úloh riešiteľných danou metódou.

3.1 Metodika a metódy skúmania

V nasledujúcej kapitole sa budeme podrobnejšie venovať jednotlivým cieľom práce. Samotnú metodiku súvisiacu s riešenými problémami si rozoberieme pri konkrétnych metódach, ktoré sú vedené v jednotlivých podkapitolách. Zároveň pri každej metóde uvedieme nielen metodiku, ale aj výsledky a vyhodnotenie experimentov.

V cieľoch práce sme naznačili, ktoré metódy a prístupy je nutné preskúmať pri riešení úlohy návrhu turnusov elektrických autobusov. Prvou časťou je modelovanie samotného problému, kde stanovíme vzťahy medzi potrebnými rozhodnutiami a ich dôsledkami na modelovaný systém, ako aj samotnú štruktúru modelovaného systému. Následne prepíšeme tento model systému do formy matematického modelu, ktorý je schopný kvantifikovať výsledné rozhodnutia. Ďalšou časťou bude návrh metód riešenia úlohy popísanej navrhnutým matematickým modelom. Ďalej sa vykonajú numerické experimenty s navrhnutým matematickým modelom a navrhnutými metódami na špecifikovaných testovacích úlohách. Na základe výsledkov sa jednotlivé metódy vyhodnotia, ako sú schopné riešiť úlohy rôznych rozmerov a pri rôznom nastavení systému reprezentovanom vo forme scenárov.

V prípade exaktných metód sa bude sledovať čas, za ktorý sa daná úloha vyrieši. Pri experimentoch budeme sledovať aj rozmer úloh riešiteľných pomocou skúmaných exaktných metód. V prípade, že exaktná metóda prekročí časový limit na výpočet sa bude sledovať najlepší získaný výsledok v porovnaní s dolnou hranicou riešenia.

V prípade heuristických metód sa bude vyhodnocovať hlavne kvalita riešenia v porovnaní s exaktným riešením úlohy, respektíve v prípade rozsiahlejších úloh s dolnou hranicou riešenia úlohy. Takisto sa bude sledovať výpočtový čas algoritmu.

V kapitole 4.1 sú popísané dáta, z ktorých sa vytvorili testovacie úlohy, definícia scenárov ako aj nástroje použité na implementáciu riešiacich metód.

V kapitole 4.2 sa budeme venovať modelovaniu problému a jeho exaktnému riešeniu pomocou IP solvera, ktorý využíva metódu vetiev a hraníc na získanie exaktného riešenia. Následné experimenty nám vypovedia o rozsahu riešiteľnosti úloh ako aj poskytnú výsledky, respektíve spodnú hranicu riešenia, pre následné porovnávanie s ostatnými navrhnutými metódami.

V kapitole 4.3 bude predstavená metóda redukcie vstupov s následným riešením pomocou matematického modelu a IP solvera. V tejto časti bude predstavená heuristika na redukciu vstupov založená na spájaní dvoch spojov do jedného formou zreťazenia. Výsledky experimentov pred a po redukcii pri rôznom nastavení parametrov budú porovnané a vyhodnotené.

V kapitole 4.4 predstavíme implementáciu metódy založenej na metóde generovania stĺpcov. V implementácii budú prezentované metódy použité na riešenie jednotlivých častí metódy, ako aj niekoľko variantov riešenia pod-problému. Výsledky riešenia pomocou jednotlivé variantov riešenia budú porovnané a zhodnotené, pričom sa zameriame na možnosti, ktoré jednotlivé metódy riešenia priniesli. Zároveň budú výsledky porovnané aj s exaktnými riešeniami úloh.

V kapitole 4.5 sa budeme venovať úprave heuristickej metódy založenej na genetickom algoritme na riešenie úlohy návrhu turnusov elektrických autobusov. Budú predstavené jednotlivé časti algoritmu, ako aj ich implementácia. Pri experimentoch budeme sledovať kvalitu riešení poskytnutých navrhnutou heuristickou metódou, ako aj vplyv parametrov metódy na kvalitu a stabilitu riešenia v rámci niekoľkých replikácií.

V záverečnej časti uvedieme analýzu výsledkov poskytnutých jednotlivými prístupmi, ich vzájomné porovnanie z pohľadu presnosti dosiahnutých výsledkov a časovej náročnosti výpočtu, ako aj odporúčania ohľadom nastavenia jednotlivých metód.

4 Výsledky práce

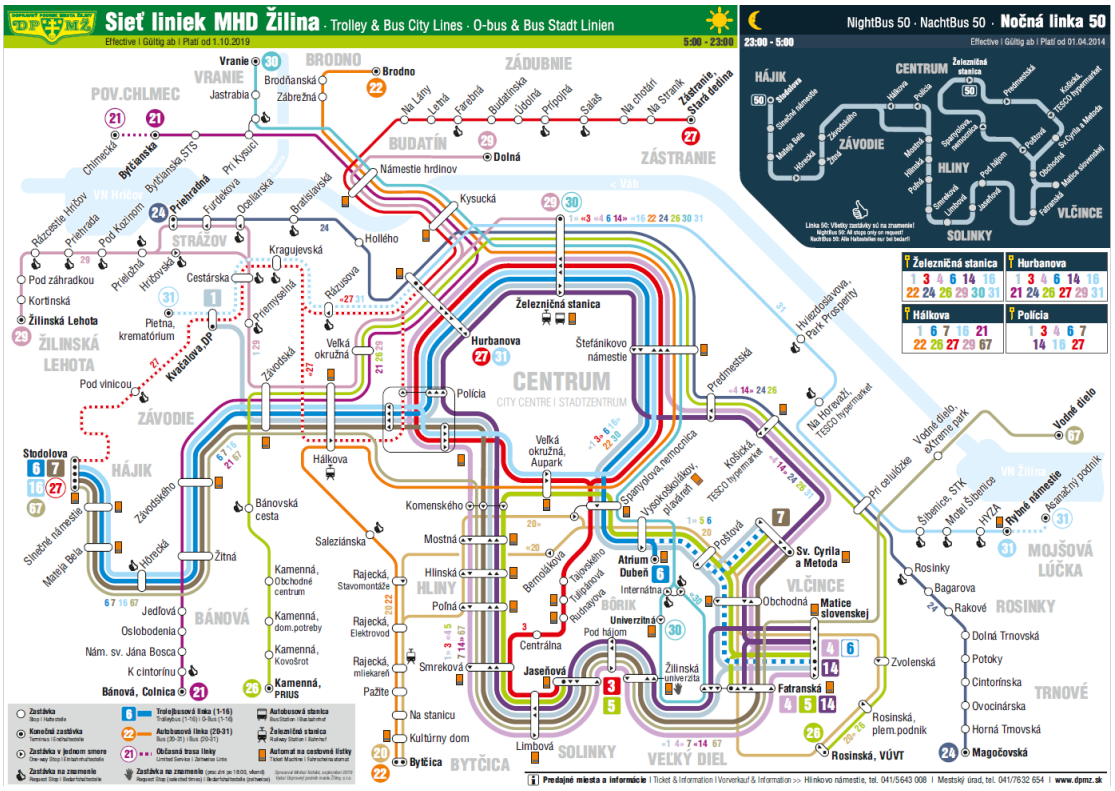
V tejto časti práce si predstavíme nami navrhnutý matematický model úlohy na riešenie úlohy návrhu turnusov elektrických autobusov, ďalej metódu redukcie vstupov, metódu založenú na metóde generovania stĺpcov a heuristickú metódu založenú na genetickom algoritme. V každej časti sú uvedené aj výsledky experimentov pre konkrétnu metódu riešenia, ako aj ich zhodnotenie. Zároveň sú tu uvedené špecifikácie stroja a dát použitých pri experimentoch.

4.1 Testovacie dáta a použité softvérové nástroje

Na otestovanie navrhnutých metód a ich porovnanie sú použité datasety vygenerované z dát poskytnutých prevádzkovateľom verejnej dopravy DPMŽ v meste Žilina. Poskytnuté dáta obsahujú informácie o spojoch obsluhovaných naftovými autobusmi a trolejbusmi počas jedného pracovného dňa. Schematický plán liniek mesta môžeme vidieť na obrázku 4.1.

Z poskytnutých dát sme vytvorili 10 úloh vybraním spojov niektorých liniek, ktoré obsahujú rôzne počty obsluhovaných spojov. Špecifikácia datasetov je nasledovná:

- DS1 – 49 spojov na linke 26
- DS2 – 77 spojov na linke 27
- DS3 – 83 spojov na linkách 26 a 29
- DS4 – 105 spojov na linke 20, 29, 30 a 31
- DS5 – 135 spojov na linke 20, 26, 29 a 30
- DS6 – 160 spojov na linke 26, 27 a 29
- DS7 – 245 spojov na linke 21, 22 a 27
- DS8 – 415 spojov na všetkých autobusových linkách
- DS9 – 494 spojov na všetkých trolejbusových linkách
- DS10 – 927 spojov na všetkých autobusových a trolejbusových linkách



Obr. 4.1. Schéma liniek MHD mesta Žilina [42]

Dôležité je umiestnenie depa, odkiaľ budú elektrické autobusy ráno odchádzať a večer doňho prichádzať. Pri každom datasete sa ako ranné aj večerné depo berie autobusové depo v meste Žilina, ktoré je vyznačené ako červený bod na mape na obrázku 4.2.

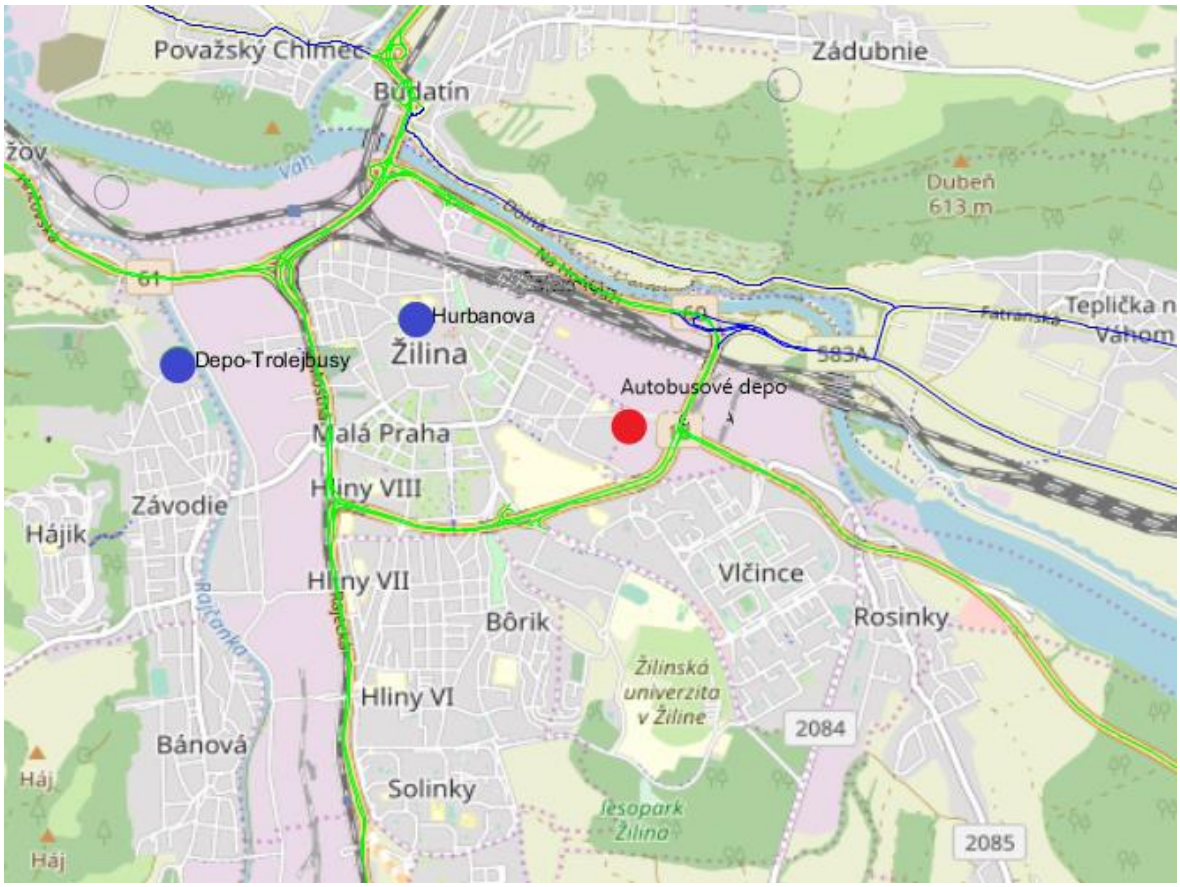
Na simuláciu rôznych podmienok počas prevádzky, ktoré sa týkajú maximálnej kapacity batérie a spotreby energie, sme špecifikovali tri rôzne scenáre. Prvý scenár (jarný) simuluje základné podmienky prevádzky a vzťahuje sa k jarnému a jesennému obdobiu roka. V dnešnej dobe sa kapacita batérií elektrických autobusov pohybuje od 20kWh do 160kWh, čo dokladuje projekt ZeEUS [4]. Avšak, väčšina elektrických autobusov má kapacitu batérií medzi 120kWh a 160kWh. My sme teda stanovili maximálnu kapacitu batérie na 140kWh. Podobne sme stanovili aj priemernú spotrebu energie elektrického autobusu na 0,8kWh/km. Druhým scenárom je letný scenár. V tomto období sa kvôli vyšším vonkajším teplotám často spúšťa klimatizácia autobusu. Keďže v elektrickom autobuse je aj klimatizácia napájaná z batérie má to za následok zvýšenie spotreby elektrickej energie. V našom prípade sa táto spotreba zvýši o 35% na 1,08kWh/km. V poslednom treťom scenári, ktorý reprezentuje zimné obdobie sa opätovne zvyšuje spotreba elektrickej energie o 35%, kvôli nutnosti kúrenia v podmienkach nižších vonkajších teplôt. Zároveň sa vplyvom

nižších vonkajších teplôt znižuje kapacita batérie. My sme v našom scenári definovali zníženie o 25% na 105kWh. V tabuľke 4.1 môžeme vidieť sumár všetkých definovaných scenárov.

Tabuľka 4.1. Sezónne scenáre

Scenár	***Spotreba energie***	***Maximálna kapacita batérie***
Jar	0,8 kWh/km	140 kWh
Leto	1,08 kWh/km (+ 35%)	140 kWh
Zima	1,08 kWh/km (+ 35%)	105 kWh (-25%)

Ďalšou podstatnou časťou experimentov sú špecifikácie nabíjacích staníc. Prvým atribútom nabíjačky je jej rýchlosť nabíjania. Tá sa odvíja od výkonu nabíjačky. Na základe špecifikácií výrobcov nabíjacej infraštruktúry [44]-[46] sa nabíjací výkon nabíjačiek pre elektrické autobusy pohybuje v rozmedzí od 30 kW do 150 kW. My sme vybrali štandardnú rýchlo-nabíjačku o výkone 80 kW. To znamená, že jej rýchlosť nabíjania bude 1,33 kWh/min. Takisto je nutné špecifikovať umiestnenie nabíjačiek, keďže aj pri presune k nabíjačke a následne na spoj sa spotrebúva energia. My sme sa rozhodli pre použitie jediného scenára pre umiestnenie nabíjačiek. Ich pozícia je znázornené na mapke na obrázku 4.2, kde vyznačené modré body reprezentujú lokácie nabíjačiek. Prvé dve nabíjačky sa nachádzajú v trolejbusovom depe a jedna nabíjačka je dostupná v centre mesta pri zastávke Hurbanova. Tieto umiestnenia reprezentujú aktuálne rozmiestnenie nabíjačiek v meste, keďže prevádzkovateľ mestskej dopravy testuje niekoľko elektrických autobusov v prevádzke.



Obr. 4.2. Umiestnenie nabíjačiek (modré body) a depo (červený bod) v meste Žilina

4.1.1 Použité softvérové nástroje

Pre návrh modelu a jeho následné testovanie sme používali štandardný IP solver Xpress IVE 7.3. Tento softvér obsahuje modelovací jazyk Mosel, do ktorého sme prepísali náš matematický model a následne vykonali experimenty.

Na implementáciu jednotlivých navrhnutých metód sme využívali programovací jazyk Java. Pre výpočet LP-relaxácií a úloh definovaných matematickým modelom sme využili knižnice pre jazyk Java ponúkané IP solverom Xpress IVE.

Všetky experimenty prebiehali na stroji s procesorom Intel Core i5-7200U 2,5Ghz a veľkosťou operačnej pamäte RAM 16GB.

4.2 Exaktné metódy na riešenie úlohy návrhu turnusov elektrických autobusov

Medzi základné metódy riešenia optimalizačných úloh patria metódy založené na modelovaní problému a jeho následnom riešení pomocou exaktných optimalizačných metód, akou je napríklad metóda vetiev a hraníc.

Základnou časťou exaktných metód je teda popísanie problému pomocou matematického modelu. V súčasnosti neexistuje jednoznačná štandardná formulácia úlohy návrhu turnusov elektrických autobusov vo verejnej doprave. Všetky matematické modely, ktoré túto úlohu popisujú stavajú na rôznych predpokladoch v závislosti od špecifického charakteru skúmaného variantu úlohy. Ako sme už spomínali v predchádzajúcich kapitolách, existujú rôzne technológie nabíjania, rôzne prístupy k nabíjaniu a rôzne predpoklady týkajúce sa nabíjacej infraštruktúry. Zároveň závisí od autorov, akú mieru abstrakcie úlohy použijú.

Preto sme sa rozhodli vytvoriť vlastný matematický model úlohy návrhu turnusov elektrických autobusov vo verejnej doprave, ktoré budú zodpovedať našej predstave úlohy, štandardným podmienkam úlohy návrhu turnusov ako aj predpokladom týkajúcich sa nabíjacej infraštruktúry a technológie nabíjania.

Základné podmienky úlohy návrhu turnusov sú nasledovné.

- Obslúženie každého spoja jediným vozidlom.
- Zabezpečenie, že každé vozidlo obsluhuje iba možnú sekvenciu spojov - v jednom čase obsluhuje iba jeden spoj (vylúčenie paralelnej obsluhy spojov).

Ďalšie podmienky špecifické pre elektrické autobusy sú nasledovné.

- Kapacita batérie musí byť dostatočná na prejdenie k nabíjačke počas celého turnusu.
- Na vybraných miestach sa môže batéria nabiť. Pri nabíjaní sa však nesmie presiahnuť maximálna kapacita batérie.
- Na každej prípojke sa v jednom čase môže nabíjať iba jedno vozidlo.

Poslednou časťou sú podmienky týkajúce sa nabíjacej infraštruktúry, ktoré vychádzajú z nášho návrhu úlohy.

- Nabíjačky sú rozmiestnené v dopravnej sieti na rôznych miestach.
- Každá nabíjačka má iba jedinú prípojku.
- Proces nabíjania je lineárny.
- Počet použitých elektrických autobusov musí byť minimálny.

Ďalšou časťou po modelovaní je riešenie úloh pomocou matematického modelu. Jednou z možností na riešenie matematického modelu je využitie štandardných IP solverov, ktoré majú zabudovanú širokú škálu metód na riešenie úloh popísaných lineárnym matematickým modelom. My sme si vybrali solver Xpress IVE, ktorý je nástrojom

dlhoročne používaný na našej katedre. V rámci tohto solveru sa zvyčajne riešia úlohy zmiešaného lineárneho programovania pomocou metódy vetiev a hraníc, pričom sa využíva aj simplexová metóda na riešenie LP-relaxácií.

V nasledujúcej časti si formálne zadefinujeme úlohu návrhu turnusov elektrických autobusov vo verejnej doprave a popíšeme nami navrhnutý matematický model.

4.2.1 Matematický model úlohy návrhu turnusov elektrických autobusov

Majme množinu spojov *N*, ktoré treba obslúžiť. Ďalej potrebujeme reprezentovať ranné a večerné depo. Ranné depo je reprezentované vrcholom D_0. Večerné depo je reprezentované množinou vrcholov označených ako D_n, ktorá obsahuje jeden vrchol pre každý obsluhovaný spoj. Túto úpravu sme boli nútení urobiť z toho dôvodu, že ak by sme pridali iba jeden vrchol reprezentujúci depo, tak by mali všetky elektrické autobusy pri príchode do večerného depa rovnaký stav nabitia batérie.

Ďalej definujeme množinu R reprezentujúcu všetky dostupné nabíjačky, pričom každá z nich má iba jednu nabíjaciu prípojku. Na každej nabíjačke $r \in R$ máme k dispozícii množinu všetkých nabíjacích udalostí, ktoré reprezentujú časový interval, počas ktorého je možné nabíjať elektrický autobus. V našom prípade je každá nabíjacia udalosť odvodená od konca korešpondujúceho spoja, a preto majú nabíjacie udalosti rôznu dĺžku. Hlavnou myšlienkou na zostrojenie takéhoto typu intervalov je, že autobus na nabíjačku môže prísť iba vtedy, keď ukončí obsluhu spoja predchádzajúceho nabíjaniu. Takisto je zakázaný prechod medzi rôznymi nabíjačkami, čiže autobus sa môže nabíjať medzi dvojicou spojov iba na jedinej nabíjačke. Množina nabíjacích udalostí prislúchajúcich nabíjačke $r \in R$ je označená T^r. Množina T^r je usporiadaná podľa času začiatku nabíjacích udalostí.

Na charakteristiku každého spoja spoj $i \in N$ je potrebných niekoľko atribútov. Prvým atribútom je čas začiatku spoja s_i a čas trvania spoja t_i. V čase trvania sú zahrnuté časy prejazdov medzi zastávkami spoja ako aj časy pre nástup a výstup pasažierov. Keďže budeme používať elektrické autobusy, je nutné vedieť, koľko energie sa spotrebuje počas obsluhy spoja. Táto spotreba je označená ako konštanta c_i.

Každá nabíjačka $r \in R$ je charakterizovaná rýchlosťou nabíjania q_r, ktorá definuje, koľko energie sa nabije počas jednej jednotky času (v našom prípade uvažujeme minúty). Rýchlosť nabíjania každej nabíjačky môže byť rôzna, čo umožňuje nášmu modelu simulovať aj kombináciu nabíjania cez noc a príležitostného nabíjania medzi spojmi.

Ako sme predtým spomenuli, na každej nabíjačke $r \in R$ máme k dispozícii množinu nabíjacích udalostí T^r. Takisto aj nabíjacia udalosť má niekoľko dôležitých atribútov. Prvým je čas začiatku s_{rt} nabíjacej udalosti $t \in T^r$ na nabíjačke $r \in R$. Ten je odvodený z korešpondujúceho spoja založenom na vzťahu medzi koncom korešpondujúceho spoja a časom prejazdu medzi koncovou stanicou spoja a umiestnením nabíjačky. Tento vzťah je definovaný ako $s_{rt} = s_i + t_i + t_{ir}$, pričom s_i je začiatok korešpondujúceho spoja, t_i je jeho trvanie a t_{ir} je čas prejazdu z koncovej stanice spoja k nabíjačke. Koniec nabíjacej udalosti je naviazaný na začiatok nasledujúcej, čiže nabíjacia udalosť končí vtedy, keď začne nasledujúca nabíjacia udalosť.

Ďalšou podstatnou časťou problému sú dĺžka trvania prejazdu a spotreba energie počas prejazdu. Čas prejazdu medzi koncovou stanicou spoja i a začiatočnou stanicou spoja j je reprezentovaný konštantou t_{ij}. Ďalej je definovaná spotreba na tomto prejazde ako konštanta c_{ij}. Čas prejazdu medzi koncovou stanicou spoja i a umiestnením nabíjačky r je označený ako t_{ir} a spotreba na tomto prejazde ako c_{ir}. Umiestnenie nabíjačky je definované aj časom prejazdu t_{rj} od nabíjačky r k začiatočnej stanici spoja j a spotrebou na tomto prejazde c_{rj}.

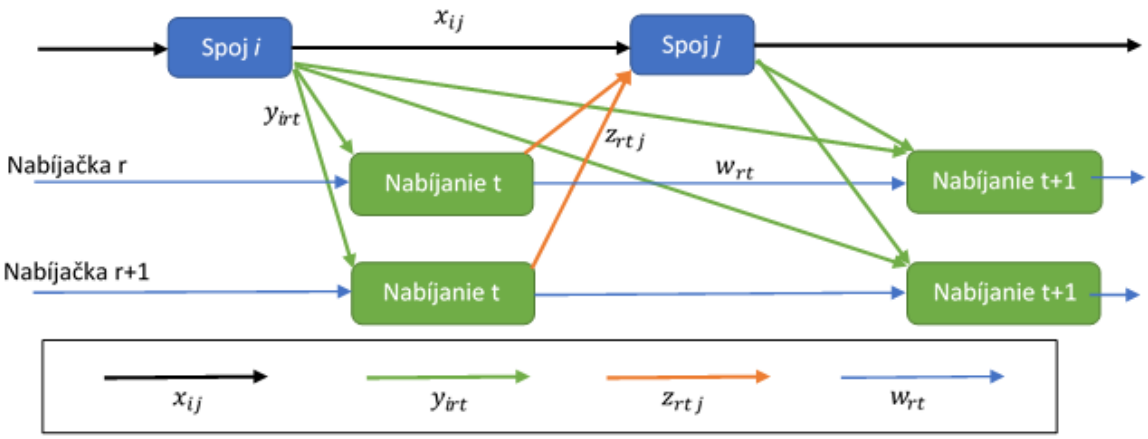
Posledné konštanty, ktoré je potrebné definovať sú atribúty charakterizujúce použitý typ elektrického autobusu a jeho batérie. Maximálna kapacita batérie je reprezentovaná konštantou SoC_{max}, ktorá môže byť chápaná aj ako horná hranica nabitia batérie v prípade, že nechceme nabíjať batériu vždy do plna, alebo simulujeme iba lineárne nabíjanie, ktoré je možné reprezentovať iba medzi 20-80% kapacity batérie [10]. Podobne si zadefinujeme aj konštantu SoC_{min} ako spodnú hranicu nabitia batérie, ktorá môže byť chápaná aj ako minimálna rezerva energie, čiže počas prevádzky by energia nemala klesnúť pod túto hodnotu.

Ďalej si zadefinujeme množiny reprezentujúce vzťahy medzi spojmi a nabíjacími udalosťami. Prvou množinou je množina F_i, ktorá obsahuje spoje, ktoré môžu z časovej perspektívy nasledovať za spojom i. Môžeme teda povedať, že spoj j môže nasledovať za spojom i, ak čas začiatku spoja j je neskôr ako čas ukončenia spoja i spolu s časom prejazdu medzi týmito dvoma spojmi. Z matematického hľadiska môže byť spoj j v množine F_i práve vtedy, keď platí podmienka $s_j \geq s_i + t_i + t_{ij}$. Podobne je zadefinovaná aj množina B_i, ktorá je množinou možných prechádzajúcich spojov spoja i. Pre túto množinu musí platiť

podmienka $s_i \geq s_j + t_j + t_{ji}$, kde s_i je začiatočný čas spoja i, s_j je začiatočný čas spoja j, t_j je trvanie spoja j a t_{ji} je čas prejazdu medzi spojmi j a i.

Druhým vzťahom je vzťah medzi spojom a nabíjacími udalosťami. Tu si definujeme množinu Fc_{ri}, ktorá reprezentuje možné nasledujúce nabíjacie udalosti na nabíjačke r k spoju i. Nabíjacia udalosť t môže byť nasledovníkom spoja i len v prípade, že je čas začiatku nabíjacej udalosti neskôr ako je koniec spoja spolu s časom prejazdu medzi koncovou stanicou spoja i a nabíjačkou r, matematicky $s_{rt} \geq s_i + t_i + t_{ir}$. Podobne si zadefinujeme aj množinu Bc_{ri}, ktorá je množinou všetkých možných predchádzajúcich nabíjacích udalostí na nabíjačke r k spoju i. V tomto prípade môže nabíjacia udalosť t predchádzať spoj i len ak je začiatok spoja väčší alebo rovný ako je začiatok nabíjacej udalosti navýšený o čas prejazdu medzi nabíjačkou r a začiatočnou stanicou spoja i, matematicky $s_{rt} + t_{ri} \leq s_i$.

Pre každú nabíjaciu udalosť t na nabíjačke r definujeme dve množiny Fi_{rt} a Bi_{rt}. Tieto množiny reprezentujú možné nasledujúce spoje k nabíjacej udalosti t na nabíjačke r, respektíve možné predchádzajúce spoje nabíjacej udalosti t na nabíjačke r. Spoj i môže byť nasledovníkom nabíjacej udalosti t iba ak platí podmienka $s_{rt} + t_{ri} \leq s_i$. Podobne musí platiť podmienka $s_{rt} \geq s_i + t_i + t_{ir}$, aby mohol byť spoj i predchodcom nabíjacej udalosti t na nabíjačke r.



Obr. 4.3: Popis rozhodovacích premenných

Poslednou časťou, ktorá musí byť zadefinovaná sú rozhodovacie premenné. Prvou rozhodovacou premennou je x_{ij}, ktorá rozhoduje, či spoj j bude obsluhovaný ihneď po obslúžení spoja i. Ďalšou rozhodovacou premennou je y_{irt} a reprezentuje rozhodnutie o presune na nabíjaciu udalosť t na nabíjačke r ihneď po obslúžení spoja i. Rozhodovacia

premenná z_{rtj} má opačný charakter a rozhoduje o presune na obsluhu spoja j po nabíjaní počas nabíjacej udalosti t na nabíjačke r. Posledná rozhodovacia premenná umožňuje prechod medzi dvomi za sebou nasledujúcimi nabíjacími udalosťami na jednej nabíjačke. V praxi to znamená, že elektrický autobus môže pokračovať v nabíjaní počas nasledujúcej nabíjacej udalosti na tej istej nabíjačke. Preto premenná w_{rt} nadobúda hodnotu 1, ak elektrický autobus bude pokračovať v nabíjaní počas nabíjacej udalosti $t+1$ po absolvovaní nabíjania počas nabíjacej udalosti t na nabíjačke r. Všetky doteraz spomenuté rozhodovacie premenné sú binárne a na obrázku 4.3 môžeme vidieť ich vizualizáciu na malom výreze z celého digrafu tvoreného všetkými spojmi a nabíjacími udalosťami.

Keďže sa v našej úlohe pracuje s energiou ako palivom a potrebujeme vedieť stav nabitia batérie elektrického autobusu vo vybraných momentoch, zaviedli sme prídavné rozhodovacie premenné e_i a ε_{rt}. Premenná e_i reprezentuje stav nabitia batérie elektrického autobusu tesne pred začatím obsluhy spoja i. Premenná ε_{rt} podobne reprezentuje stav nabitia batérie autobusu tesne pred začatím nabíjania počas nabíjacej udalosti t na nabíjačke r.

Účelová funkcia

$$min \sum_{j \in F_{D_0}} x_{D_0 j} + \sum_{r \in R} \sum_{t \in Fc_{rD_0}} y_{D_0 rt} \tag{4.1}$$

Účelová funkcia (4.1) sa skladá z dvoch súm, ktoré minimalizujú počet použitých elektrických autobusov. Prvá suma obsahuje elektrické autobusy, ktoré vychádzajú z ranného depa a idú priamo obslúžiť spoje. Druhá suma reprezentuje elektrické autobusy, ktoré idú z depa najprv na nabíjačku.

Podmienky návrhu turnusov

$$z.p. \qquad \sum_{i \in B_j} x_{ij} + \sum_{r \in R} \sum_{t \in Bc_{rj}} z_{rtj} = 1 \; \forall j \in N \tag{4.2}$$

$$\sum_{j \in Bi_{rt}} y_{jrt} + w_{rt-1} \leq 1 \; \forall r \in R, t \in T^r \tag{4.3}$$

$$\sum_{i \in B_j} x_{ij} + \sum_{r \in R} \sum_{t \in Bc_{rj}} z_{rtj} = \sum_{l \in F_j} x_{jl} + \sum_{r \in R} \sum_{t \in Fc_{rj}} y_{jrt} \; \forall j \in N \tag{4.4}$$

$$\sum_{i \in Bi_{rt}} y_{irt} + w_{rt-1} = \sum_{j \in Fi_{rt}} z_{rtj} + w_{rt} \; \forall r \in R, t \in T^r \tag{4.5}$$

Podmienky (4.2) zaručujú, že každý spoj bude obslúžený práve jedným vozidlom. Podmienky (4.3) stanovujú, že každá nabíjacia udalosť bude navštívená maximálne jedenkrát, čo znamená, že iba jeden elektrický autobus môže byť nabíjaný v jednom čase na jednej nabíjačke. Podmienky (4.4) sú štandardnými podmienkami zachovania toku pre spoje, pričom zaručujú, že ak bol spoj obslúžený elektrickým autobusom, tak ten bude po ňom pokračovať na ďalší spoj, nabíjačku alebo do depa. Ako podmienky zachovania toku pre každú nabíjaciu udalosť slúžia podmienky (4.5), čo znamená, že elektrický autobus po nabíjaní bude pokračovať v nabíjaní alebo pôjde obslúžiť spoj, prípadne pôjde do depa.

Podmienky spotreby energie

$$e_{D_0} = SoC_{max} \tag{4.6}$$

$$e_i \geq SoC_{min} + c_i + \sum_{j \in F_i} x_{ij} c_{ij} + \sum_{r \in R} \sum_{t \in Fc_{ri}} y_{irt} c_{ir} \; \forall i \in N \tag{4.7}$$

$$e_j + c_{rj} + Mq_r\left(1 - z_{rtj}\right) \geq SoC_{min} + z_{rtj} c_{rj} \forall r \in R, t \in T^r, j \in Fi_{rt} \tag{4.8}$$

$$e_j \leq e_i - x_{ij}\left(c_i + c_{ij}\right) + SoC_{max}\left(1 - x_{ij}\right) \forall j \in N, i \in B_j \tag{4.9}$$

$$e_j \geq e_i - x_{ij}\left(c_i + c_{ij}\right) - SoC_{max}\left(1 - x_{ij}\right) \forall j \in N, i \in B_j \tag{4.10}$$

$$\varepsilon_{rt} \leq e_i - y_{irt}(c_i + c_{ir}) + SoC_{max}(1 - y_{irt}) \forall r \in R, t \in T^r, i \in Bi_{rt} \tag{4.11}$$

$$\varepsilon_{rt} \geq e_i - y_{irt}(c_i + c_{ir}) - SoC_{max}(1 - y_{irt}) \forall r \in R, t \in T^r, i \in Bi_{rt} \tag{4.12}$$

Podmienky (4.6) slúžia na inicializáciu stavu nabitia batérie elektrického autobusu na maximálnu kapacitu na začiatku pracovného dňa. Ďalšie podmienky (4.7) zabezpečujú, že elektrický autobus bude mať dosť energie na prejdenie spoja a nasledujúceho presunu k nasledujúcemu spoju alebo nabíjačke. Táto podmienka tiež zahrňuje aj minimálnu rezervu energie SoC_{min}. Podobne, podmienky (4.8) zaručujú, že elektrický autobus bude mať dosť energie po nabíjaní na to, aby bol schopný prejsť k nasledujúcemu spoju.

Podmienky (4.9)-(4.12) sa spájajú so zachovaním energie. Tieto podmienky by sme mohli interpretovať aj ako väzbu medzi dvomi za sebou nasledujúcimi spojmi, respektíve spojom a nabíjacou udalosťou. V prípade, že za sebou nasledujú dva spoje za sebou musia platiť podmienky (4.9) a (4.10). Ak elektrický autobus absolvuje prejazd medzi spojmi i a j,

potom je energia zo začiatku spoja i znížená o energiu spotrebovanú na obsluhu spoja i a energiu potrebnú na prejazd medzi spojmi i a j. V prípade, že elektrický autobus neabsolvuje spoj j po spoji i, budú podmienky (4.9) a (4.10) uvoľnené. Podmienky (4.11) a (4.12) budú použité v prípade, že zo spoja i prejde elektrický autobus k nabíjačke r kvôli nabíjaniu počas nabíjacej udalosti t. Stav nabitia batérie tesne pred začatím nabíjania bude stavom nabitia batérie pred začiatkom spoja i znížený o energiu spotrebovanú na prejdenie spoja i ako aj na prejazd medzi koncovou stanicou spoja i a nabíjačkou r. V opačnom prípade budú podmienky (4.11) a (4.12) uvoľnené.

Podmienky nabíjania

$$e_j + c_{rj} - \varepsilon_{rt} + SoC_{max}\left(1 - z_{rtj}\right) \geq 0 \quad \forall r \in R, t \in T^r, j \in Fi_{rt} \tag{4.13}$$

$$\varepsilon_{rt+1} - \varepsilon_{rt} + SoC_{max}(1 - w_{rt}) \geq 0 \quad \forall r \in R, t \in T^r \tag{4.14}$$

$$e_j + c_{rj} - Mq_r\left(1 - z_{rtj}\right) \leq SoC_{max} \quad \forall r \in R, t \in T^r, j \in Fi_{rt} \tag{4.15}$$

$$\varepsilon_{rt+1} - Mq_r(1 - w_{rt}) \leq SoC_{max} \quad \forall r \in R, t \in T^r \tag{4.16}$$

$$e_j \leq \varepsilon_{rt} + z_{rtj}\left(\left(s_j - t_{rj} - s_{rt}\right)q_r - c_{rj}\right) + SoC_{max}\left(1 - z_{rtj}\right) \quad \forall j \in N, r \in R, t \in Bc_{rj} \tag{4.17}$$

$$\varepsilon_{rt+1} \leq \varepsilon_{rt} + w_{rt}(s_{rt+1} - s_{rt})q_r + SoC_{max}(1 - w_{rt}) \quad \forall r \in R, t \in T^r \tag{4.18}$$

$$e_j + c_{rj} - \varepsilon_{rt} - SoC_{max}\left(1 - z_{rtj}\right) \leq (s_{rt+1} - s_{rt})q_r \quad \forall r \in R, t \in T^r, j \in Fi_{rt} \tag{4.19}$$

V tejto časti sú zadefinované podmienky, ktoré sa vzťahujú k minimálnej a maximálnej hranici nabitej energie počas nabíjacích udalostí. Podmienky (4.13) a (4.14) definujú, že množstvo nabitej energie musí byť nezáporné, inak by mohla byť počas nabíjania energia spotrebovávaná, čo nie je žiadúce. Podmienky (4.13) sú viazané k prípadu, kde elektrický autobus pokračuje obsluhou spoja po ukončení nabíjania. Podmienky (4.14) popisujú prípad, keď elektrický autobus pokračuje v nabíjaní počas nasledujúcej nabíjacej udalosti. Ďalšie podmienky (4.15) a (4.16) zaručujú, že po nabíjaní nebude prekročená maximálna kapacita batérie definovaná konštantou SoC_{max}. K prípadu, keď po nabíjaní nasleduje spoj sa viažu podmienky (4.15) a k prípadu, keď sa pokračuje v nabíjaní sa viažu podmienky (4.16).

Podmienky (4.17), (4.18) a (4.19) limitujú maximálne množstvo dobitej energie založené na dostupnom čase pre nabíjanie počas nabíjacej udalosti. Ako sme spomenuli pri predchádzajúcich podmienkach, máme dva prípady. V prvom elektrický autobus po nabíjaní

pokračuje v turnuse obsluhou spoja. Vtedy je dostupný čas pre nabíjanie obmedzený časom začiatku nasledujúceho spoja, čo je popísané podmienkami (4.17). Zároveň je v tomto prípade dostupný nabíjací čas obmedzený aj začiatkom nasledujúcej nabíjacej udalosti, čo je reprezentované podmienkami (4.19). Druhý prípad popisuje situáciu, keď po ukončení nabíjania počas jednej nabíjacej udalosti pokračuje elektrický autobus v nabíjaní počas nasledujúcej nabíjacej udalosti. V tomto prípade je nutné obmedziť dostupný čas nabíjania počas danej nabíjacej udalosti časom začiatku nasledujúcej nabíjacej udalosti, čo je popísané podmienkami (4.18).

Obligatórne podmienky

$$x_{ij} \in \{0,1\} \quad \forall i \in N \cup D_0 \cup D_n, j \in F_i \tag{4.20}$$

$$y_{irt} \in \{0,1\} \quad \forall i \in N, r \in R, t \in Fc_{ri} \tag{4.21}$$

$$z_{rtj} \in \{0,1\} \quad \forall r \in R, t \in T_r, j \in Fi_{rt} \tag{4.22}$$

$$w_{rt} \in \{0,1\} \quad \forall r \in R, t \in T_r \tag{4.23}$$

$$e_j \geq 0 \quad \forall i \in N \tag{4.24}$$

$$\varepsilon_{rt} \geq 0 \quad \forall r \in R, t \in T_r \tag{4.25}$$

Rozhodovacie premenné x_{ij}, y_{irt}, z_{rtj} and w_{rt} sú binárneho charakteru, čo je definované podmienkami (4.20), (4.21), (4.22) and (4.23). Prídavné rozhodovacie premenné e_i a ε_{rt}, ktoré udržiavajú informáciu o aktuálnom stave nabitia batérie počas turnusu sú definované podmienkami (4.24) a (4.25) ako nezáporné.

4.2.2 Výsledky experimentov

Na overenie a otestovanie navrhnutého matematického modelu sme sa rozhodli použiť štandardný IP solver Xpress IVE. Tento solver obsahuje modelovací jazyk Mosel, pomocou ktorého vieme jednoducho zadať náš matematický model do solvera. Tento solver využíva na riešenie zadanej úlohy rôzne metódy. V prípade úlohy zmiešaného matematického programovania, čo je aj naša úloha, využíva na výpočet hlavne metódu vetiev a hraníc.

Okrem otestovania matematického modelu na rôznych datasetoch sme získané výsledky porovnali aj s výsledkom riešenia štandardnej úlohy návrhu turnusov (VSP) reprezentovanej matematickým modelom (1.1)-(1.4). Zároveň sme zisťovali aj časovú náročnosť výpočtu. Časový limit na výpočet problému bol stanovený na 16 hodín, keďže pri

väčších úlohách sa po 16 hodinách výpočtu začali prejavovať problémy spojené s nedostatkom pamäte RAM.

Dosiahnuté výsledky sa nachádzajú v tabuľkách 4.2, 4.3 a 4.4, kde každá tabuľka je venovaná jednému scenáru. Stĺpec *Rieš* predstavuje najlepšie nájdené riešenie počas výpočtu modelu, čo predstavuje počet použitých elektrických autobusov na pokrytie všetkých spojov turnusmi. Stĺpec *BB* predstavuje najlepšiu spodnú hranicu riešenia a stĺpec *Čas* predstavuje čas výpočtu modelu v sekundách. Posledný stĺpec *VSP* slúži na porovnanie výsledkov nášho matematického modelu voči prípadu, keď flotilu dostupných vozidiel budú tvoriť konvenčné autobusy. Samozrejme, je možné, že výsledok v prípade VSP bude nižší ako v prípade návrhu turnusov elektrických autobusov, keďže úloha VSP neuvažuje s obmedzeným dojazdom vozidiel, ako ani s potrebou doplniť palivo (v prípade elektrických autobusov je to nabíjanie elektrickej energie).

Tab. 4.2. Výsledky experimentov s matematickým modelom v prípade jarného scenára

Dataset	***Rieš.***	***BB***	***Čas (s)***	***VSP***
DS1	4	4	3	4
DS2	4	4	12,2	4
DS3	5	5	35,6	5
DS4	6	6	29,7	6
DS5	8	8	169,2	8
DS6	9	9	281,6	9
DS7	13	13	13414	13
DS8	33	26	57600	26
DS9	-	28	57600	28
DS10	-	49	57600	49

V tabuľke 4.2 môžeme nájsť výsledky pre jarný scenár. Ako vidíme na výsledkoch pre datasety DS1-DS7, tak sa riešenie úlohy VSP a návrhu turnusov pre elektrické autobusy rovná. Tento fakt nám hovorí to, že pre úlohy definované týmito datasetmi je možné nahradiť flotilu konvenčných autobusov za flotilu elektrických autobusov v pomere 1:1. Takisto si môžeme všimnúť, že pre datasety DS1-DS7 bolo riešenie pomocou matematického modelu schopné dosiahnuť optimálne riešenie. Avšak pre datasety DS8-DS10 to pre prípad jarného scenára už nebolo možné z časového hľadiska, keďže sme dosiahli časový limit experimentu.

V tabuľke 4.3 môžeme vidieť výsledky pre prípad letného scenára. Opätovne je možné postrehnúť, že výsledky v prípade datasetov DS1-DS4 sú identické s riešením úlohy VSP. Čiže aj v prípade letného scenára so zvýšenou spotrebou je možné pre tieto úlohy

vymeniť flotilu v pomere jeden elektrický autobus za jeden konvenčný autobus. Ďalej si môžeme všimnúť, že pri väčších datasetoch DS5-DS10 dochádza k problému s riešiteľnosťou modelu. Môžeme teda povedať, že na riešiteľnosť modelu značne vplýva zvýšenie spotreby elektrického autobusu.

Tab. 4.3. Výsledky experimentov s matematickým modelom v prípade letného scenára

Dataset	***Rieš.***	***BB***	***Čas (s)***	***VSP***
DS1	4	4	3,6	4
DS2	4	4	117,5	4
DS3	5	5	43	5
DS4	6	6	712,2	6
DS5	9	8	57600	8
DS6	10	9	57600	9
DS7	27	13	57600	13
DS8	-	26	57600	26
DS9	-	28	57600	28
DS10	-	49	57600	49

Tab. 4.4. Výsledky experimentov s matematickým modelom v prípade zimného scenára

Dataset	***Rieš***	***BB***	***Čas (s)***	***VSP***
DS1	4	4	10,4	4
DS2	4	4	40	4
DS3	5	5	629,3	5
DS4	6	6	1237	6
DS5	10	8	57600	8
DS6	11	9	57600	9
DS7	-	13	57600	13
DS8	-	26	57600	26
DS9	-	28	57600	28
DS10	-	49	57600	49

Výsledky pre posledný, zimný scenár nájdeme v tabuľke 4.4. V prípade zimného scenára sa okrem zvýšenej spotreby aj znížila maximálna kapacita batérie. Môžeme vidieť, že matematický model začína mať problémy s riešením pri väčších datasetoch, podobne ako v prípade letného scenára. Keďže sa tento problém ukázal aj pri zimnom scenári a navyše sa zhoršili najlepšie nájdené riešenia v prípade datasetov DS5 a DS6, môžeme povedať, že problém s riešiteľnosťou modelu ovplyvňuje aj maximálna kapacita batérie elektrického autobusu.

4.2.3 Zhodnotenie výsledkov

Keď zoberieme do úvahy všetky experimenty, vidíme, že nami navrhnutý matematický model je schopný riešiť úlohy menšieho rozsahu špecifikované datasetmi DS1-DS4. Pri úlohách väčšieho rozsahu dosahuje horšie výsledky, nakoľko čas výpočtu sa značne predĺžil. Avšak riešiteľnosť modelu závisí aj od aplikovaného scenára. V prípade jarného scenára sa výsledky začínajú zhoršovať až pri datasetoch DS7-DS10. V letnom a zimnom scenári sa problémy prejavujú už pri datasete DS5. Zároveň si môžeme všimnúť, že čím väčšiu úlohu riešime, tým viac sa zväčšuje rozdiel medzi najlepším dosiahnutým výsledkom a dolnou hranicou riešenia.

Pri porovnaní výsledkov s riešením úlohy štandardného návrhu turnusov (VSP) vidíme, že v prípade jarného scenára a datasetov DS1-DS7 sa optimálne riešenie úlohy návrhu turnusov elektrických autobusov zhoduje. Toto nám napovedá, že v týchto prípadoch je možné nahradiť flotilu konvenčných autobusov za flotilu elektrických autobusov v rovnakom počte. Samozrejme, pri znižujúcej sa maximálnej kapacite batérie, respektíve pri zvýšenej spotrebe energie, môžeme očakávať, že sa počet použitých elektrických autobusov v porovnaní s počtom konvenčných autobusov zvýši. Je to z dôvodu vzrastu potreby nabíjať elektrické autobusy, čím dosiahneme vyššiu mieru obsadenosti nabíjačiek, hlavne v čase špičiek. To spôsobí nedostatok nabíjacej kapacity a dosiahneme stav, keď by autobusy museli čakať na nabíjanie. Tento stav nie je vhodný, keďže nie je možné obslúžiť spoje a preto by sa museli nasadiť do prevádzky ďalšie elektrické autobusy.

Ako môžeme vidieť vo všetkých troch tabuľkách 4.2, 4.3 a 4.4, veľkosť úlohy, reprezentovaná počtom spojov, značne zvyšuje náročnosť výpočtu, čo sa prejavuje nárastom času výpočtu. Ďalším faktorom, ktorý ovplyvňuje čas výpočtu, je spotreba energie elektrického autobusu. Tento fakt je potvrdený výsledkami v tabuľke 4.3, kde pre úlohy, ktoré boli vyriešené v prípade jarného scenára k optimu, v letnom scenári bol značne predĺžený čas výpočtu. Je to viditeľné už pri datasetoch DS4, DS5 a DS6. Podobne aj maximálna kapacita batérie značne ovplyvňuje čas výpočtu, čo je dokumentované výsledkami v tabuľke 4.4 pre zimný scenár. Ako vidíme pri porovnaní časov medzi letným a zimným scenárom, už pri datasetoch DS1-DS4 je značný nárast času výpočtu. Zároveň v prípade datasetov DS5 a DS6 vidíme zhoršenie najlepšieho nájdeného riešenia.

V tejto časti sme predstavili nový matematický model úlohy návrhu turnusov elektrických autobusov vo verejnej doprave. Experimenty s modelom overili matematický model a preukázali použiteľnosť modelu v reálnych úlohách. Takisto sme boli schopní

definovať obmedzenia veľkosti úloh, ktoré je možné pomocou matematického modelu a štandardného IP solvera riešiť. Popri tom sme našli aj faktory, ktoré najviac ovplyvňujú riešiteľnosť úlohy.

V praxi sa však vyskytujú úlohy podstatne väčšieho rozsahu, ako boli definované v experimentoch. Na riešenie týchto úloh však nie je možné aplikovať náš matematický model v spojení so štandardným IP solverom, nakoľko je tento prístup veľmi časovo a pamäťovo náročný, čo ukázali aj experimenty s väčšími datasetmi. Kvôli týmto obmedzeniam je nutné nájsť iné metódy riešenia úlohy založené na rýchlejších metódach, ako napríklad heuristiky a metaheuristiky.

4.3 Metóda redukcie vstupov s využitím riešenia pomocou IP solvera

Riešenie úlohy návrhu turnusov pomocou matematického modelu sa ukazuje ako veľmi náročné na čas, hlavne v závislosti od zväčšujúceho sa rozmeru úlohy. Preto sme sa rozhodli zamerať na algoritmus, ktorý je schopný redukovať vstupy [56] – [58], čiže v našom prípade množstvo spojov. Pri aplikácii tohto algoritmu sme schopný zredukovať počet vstupných spojov do matematického modelu, čím budeme schopný skrátiť čas výpočtu a v prípade väčších úloh dokonca nájsť veľmi dobré riešenie, ktoré by sme pri pôvodnej úlohe neboli schopní nájsť.

Algoritmus, ktorý si predstavíme, je založený na myšlienke spájania dvoch spojov do jedného tým, že ich zreťazíme a budeme sa k nim správať ako k jedinému spoju. Samozrejme, tento prístup môže zapríčiniť stratu optimality riešenia úlohy, ale bude nám schopný poskytnúť riešenie blízke optimálnemu, je to teda heuristický algoritmus.

Hlavnou myšlienkou pri algoritme spájania spojov je fakt, že v mestách sa často vyskytujú autobusové linky, ktoré premávajú z centra mesta do okrajových štvrtí. V tomto prípade bude teda koncový terminál linky veľmi ďaleko od nabíjačiek, ktoré budú zrejme umiestnené v centre mesta. Zároveň sa aj v praxi vyskytujú často spoje, ktoré vedú presne opačným smerom do centra mesta a nasledujú krátko po ukončení spoja do okrajovej časti. Tento fakt nás vedie na myšlienku spojenia takýchto dvoch spojov, čím vytvoríme jediný spoj, ktorý bude začínať a končiť v centre mesta v blízkosti nabíjačiek.

Náš návrh algoritmu spájania spojov sa skladá z dvoch fáz. V prvej fáze si vytvoríme možné dvojice spojov na spájanie. V druhej fáze sa zameriame na výber najvhodnejších dvojíc z možných dvojíc vytvorených v prvej fáze, ktoré majú byť spojené.

4.3.1 Výber potenciálnych dvojíc spojov na spojenie

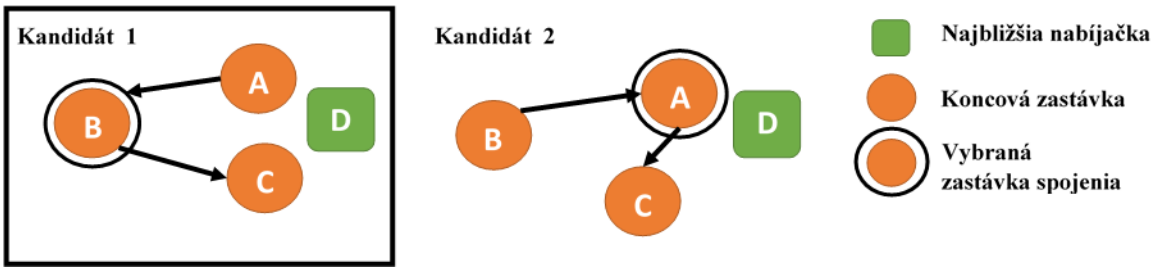
V prvej fáze vyberáme potenciálne dvojice, ktoré by mohli byť spojené na základe niekoľkých kritérií. Prvým kritériom je výber na základe koncových zastávok. Tieto koncové zastávky, v ktorých dôjde k spojeniu, musia byť umiestnené v okrajových častiach mesta. Na zistenie, ktoré zastávky to sú, využijeme graf spojení medzi zastávkami. Množinu jeho vrcholov budú tvoriť jednotlivé zastávky. Množinu hrán budú tvoriť spojenia medzi zastávkami. Hrana sa medzi dvomi zastávkami bude nachádzať iba vtedy, ak za sebou tieto zastávky nasledujú v nejakom spoji. Následne musíme identifikovať zastávky, ktoré spĺňajú vlastnosť, že vedú len do okrajovej časti. Na tento účel sme zvolili kritérium, že vyberieme vrcholy grafu zastávok, ktoré majú stupeň jedna. Tento fakt nám hovorí to, že takáto zastávka je koncová a zároveň v okrajovej časti, keďže koncové zastávky v centre mesta sú súčasne koncovými zastávkami niekoľkých liniek a teda budú mať stupeň vrcholu väčší ako jedna.

Ďalej si musíme definovať kritérium na výber vhodných kandidátov (dvojíc spojov) na spojenie. Prvým kritériom je že prvý z dvojice spojov musí končiť na zastávke nachádzajúcej sa v okrajovej časti. Zároveň druhý spoj z dvojice musí začínať na tej istej zastávke ako prvý skončil. Druhé kritérium na spojenie je založené na myšlienke, že dva spoje nasledujú v krátkom časovom slede za sebou. Veľkosť tohto časového intervalu je definovaná parametrom *MaxTime*.

Po aplikácii oboch kritérií výberu dostávame zoznam potenciálnych dvojíc spojov na spojenie. Samozrejme, niektoré spoje sa môžu nachádzať vo viacerých dvojiciach. Preto je nutné vybrať tú dvojicu, ktorá bude podľa určitého kritéria najvhodnejšia. Táto operácia sa odohráva v druhej fáze algoritmu.

4.3.2 Výber najvhodnejšej dvojice spojov na spojenie

V druhej fáze algoritmu na spájanie spojov sa zameriame na výber najvhodnejšej dvojice spojov na spojenie spomedzi dvojíc získaných v prvej fáze. Potrebujeme teda definovať kritérium, na základe ktorého vyberieme najvhodnejšiu dvojicu. Zároveň sa dbá na to, aby každý spoj bol v redukovanej úlohe zahrnutý iba jedenkrát. Preto sa po výbere najvhodnejšieho kandidáta musia prejsť všetci ostatní kandidáti a odstrániť tí kandidáti, ktorí obsahovali spoje z vybraného kandidáta.



Obr. 4.4: Príklad výberu kandidáta podľa kritéria BCC2

Prvým navrhnutým kritériom je vybrať tú dvojicu za sebou nasledujúcich spojov, ktorá má medzi sebou najkratší čas. Myšlienkou pri tomto kritériu je výber tých dvojíc, kde nie je dlhý čas čakania na začiatok druhého spoja. Toto kritérium sme označili ako BCC1 (Best candidate criterion).

Myšlienka druhého navrhnutého kritéria je spojená s elektrickými autobusmi a ich vzdialenosťou od nabíjacích staníc. Podľa druhého kritéria (BCC2) si najskôr vyberáme tie dvojice spojov, ktorých bod spojenia (zastávka) je čo najďalej od najbližšej nabíjačky. V praxi to znamená to, že sa snažíme uprednostňovať spojenia spojov, pri ktorých budú začiatočná a koncová zastávka blízko nabíjačky. Príkladom rozhodovania môže byť situácia na obrázku 4.4, kde máme k dispozícii dvoch kandidátov na spojenie a máme z nich vybrať toho vhodnejšieho na základe kritéria BCC2. V prípade kandidáta 1 sa bod spojenia nachádza pomerne ďaleko od najbližšej nabíjačky. Kandidát 2 má bod spojenia veľmi blízko nabíjačky. Ak aplikujeme kritérium BCC2, tak vyberieme kandidáta 1, keďže jeho bod spojenia je veľmi ďaleko od najbližšej nabíjačky. Tým získame začiatočnú aj koncovú zastávku spoja blízko nabíjačky a sme schopný ušetriť čas aj energiu potrebnú na prípadné presuny k nabíjačke a z nabíjačky.

Výstupom druhej fázy heuristiky na redukciu spojov je redukovaný zoznam spojov. Tento zoznam tvoria spojené spoje. Avšak nie všetky spoje sa spracovali pomocou spájania. Preto sa po konci výberu kandidátov sa spúšťa procedúra na doplnenie zoznamu spojov o tie spoje, ktoré sa v zozname po redukcii nenachádzajú.

4.3.3 Výsledky experimentov

Pri experimentoch sme testovali vplyv redukcie na výpočtový čas a výsledky riešenia úlohy pomocou matematického modelu v porovnaní s riešením pôvodnej (neredukovanej) úlohy. Testovali sme niekoľko rôznych redukcií. Pre parameter *MaxTime*, viažuci sa k výberu kandidátov v prvej fáze, sme testovali hodnoty 5 a 15 minút maximálneho času

medzi spojmi. Ďalej sme testovali obe kritériá výberu kandidátov BCC1 a BCC2 pre obe hodnoty parametra *MaxTime*.

V tabuľke 4.5 nájdeme výsledky redukcie spojov pre jednotlivé parametre reprezentované stĺpcami tabuľky, ako aj stĺpec *Original*, ktorý obsahuje počet spojov pôvodnej úlohy. Každá redukcia je označená názvom reprezentujúcim veľkosť parametra *MaxTime*, za ktorým nasleduje typ aplikovaného kritéria pre výber najvhodnejšieho kandidáta. Ako môžeme vidieť, tak prvé dve kritériá 5_BCC1 a 5_BCC2 majú rovnakú mieru redukcie. Pri menších datasetoch sa dokonca ich spojené spoje zhodovali, ale pri väčších datasetoch boli niektoré spojené dvojice spojov rozdielne. Podstatne väčšiu redukciu ponúkajú redukcie 15_BCC1 a 15_BCC2. Je to spôsobené parametrom *MaxTime*, ktorý bol v tomto prípade väčší. Celkovo najväčšiu redukciu dosiahla redukcia 15_BCC2, a to v rozmedzí od 17% do 44%. Jediným datasetom, kde ani jedna redukcia neuspela je dataset DS9. Tento dataset obsahuje iba trolejbusové spoje, ktoré sa pohybujú najmä v centre mesta a často sú to cyklické spoje, čiže nie je splnený predpoklad heuristiky, že spoje premávajú iba do okrajových častí a práve v nich začínajú alebo končia.

Tab. 4.5. Výsledky redukcie vstupov pre jednotlivé datasety

Dataset	***Redukcia vstupov***				***Original***
	5_BCC1	**5_BCC2**	**15_BCC1**	**15_BCC2**	
DS1	42	42	31	31	49
DS2	60	60	44	43	77
DS3	63	63	52	51	83
DS4	123	123	96	94	160
DS5	81	81	73	71	105
DS6	102	102	85	83	133
DS7	196	196	155	149	245
DS8	329	329	273	266	415
DS9	494	494	494	494	494
DS10	839	839	780	773	927

Po redukcii sme pristúpili k testovaniu výpočtu redukovaných úloh pomocou matematického modelu a IP solvera. V tabuľkách 4.6, 4.7 a 4.8 môžeme vidieť výsledky týchto experimentov pre jednotlivé scenáre. Prvý stĺpec *Rieš./BB* pri každej redukcii obsahuje najlepšie nájdené riešenie a spodnú hranicu riešenia. Druhý stĺpec reprezentuje čas výpočtu v sekundách. Posledný stĺpec *Original* sú výsledky experimentov pre pôvodnú úlohu pred redukciou.

V tabuľke 4.6 vidíme, že v prípade datasetov DS1-DS7 sa pri redukciách 5_BCC1 a 5_BCC2 podarilo nájsť rovnako dobré výsledky ako pri pôvodnom riešení. Zároveň sa čas potrebný na výpočet značne znížil. V prípade datasetu DS8 vyšla o niečo lepšie redukcia 5_BCC2, ktorá našla optimálne riešenie v časovom limite na rozdiel od redukcie 5_BCC1. Redukcie 15_BCC1 a 15_BCC2 takisto dosiahli veľmi dobré výsledky a podarilo sa im ešte výraznejšie zredukovať čas výpočtu. Avšak pri redukcii 15_BCC1 sa v niektorých úlohách zhoršila spodná hranica riešenia oproti pôvodnej úlohe. Preto sa ukazuje, že kritérium BCC2 je vhodnejšie na použitie. Zároveň si môžeme všimnúť, že redukcia 15_BCC2 dosahuje vo väčšine prípadov aj najlepší čas výpočtu. V prípade jarného scenáru sa pri použití redukcie 15_BCC2 priemerne ušetrilo až 57% času v porovnaní s riešením pôvodnej úlohy.

Tab. 4.6. Výsledky experimentov pre jednotlivé redukcie pre jarný scenár s najlepším nájdeným riešením a spodnou hranicou (*Rieš./BB*) a časom výpočtu (*Čas*) v sekundách

Data set	***5_BCC1***		***5_BCC2***		***15_BCC1***		***15_BCC2***		***Original***	
	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)
DS1	4/4	1,4	4/4	1,4	4/4	2	4/4	1,7	4/4	3
DS2	4/4	9,5	4/4	9,5	5/5	3	4/4	4,6	4/4	12,2
DS3	5/5	7,1	5/5	6,8	5/5	4,9	5/5	9,6	5/5	35,6
DS4	6/6	32,8	6/6	32,8	6/6	13,6	6/6	15,4	6/6	29,7
DS5	8/8	66,8	8/8	66,8	8/8	43,6	8/8	49,1	8/8	169,2
DS6	9/9	128,2	9/9	129,7	10/10	53,6	9/9	58,4	9/9	281,6
DS7	13/13	865,9	13/13	771	14/14	358	13/13	295	13/13	13414
DS8	27/26	57600	26/26	56528	26/26	42694	26/26	2083	33/26	57600
DS9	-/28	57600	-/28	57600	-/28	57600	-/28	57600	-/28	57600
DS10	-/49	57600	-/49	57600	-/49	57600	-/49	57600	-/49	57600

Tab. 4.7. Výsledky experimentov pre jednotlivé redukcie pre letný scenár s najlepším nájdeným riešením a spodnou hranicou (*Rieš./BB*) a časom výpočtu (*Čas*) v sekundách

Data set	***5_BCC1***		***5_BCC2***		***15_BCC1***		***15_BCC2***		***Original***	
	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)
DS1	4/4	1,5	4/4	1,5	4/4	2,2	4/4	1,7	4/4	3,6
DS2	4/4	10,5	4/4	9,6	5/5	4,9	4/4	4,6	4/4	117,5
DS3	5/5	23,7	5/5	21,6	5/5	6,9	5/5	9,7	5/5	43
DS4	6/6	43,2	6/6	43,9	6/6	13,5	6/6	18,5	6/6	712,2
DS5	8/8	85,2	8/8	91,5	8/8	49,7	8/8	53,2	9/8	57600
DS6	9/9	177,7	9/9	177,1	10/10	63,8	9/9	74,8	10/9	57600
DS7	13/13	787,9	13/13	646	14/14	365,8	13/13	397,4	27/13	57600
DS8	27/26	57600	26/26	56000	26/26	10718	26/26	10625	-/26	57600
DS9	-/28	57600	-/28	57600	-/28	57600	-/28	57600	-/28	57600
DS10	-/49	57600	-/49	57600	-/49	57600	-/49	57600	-/49	57600

Tabuľka 4.7 obsahuje výsledky pre letný scenár. Podobne ako aj pri jarnom scenári sa podarilo pre malé scenáre nájsť rovnako dobré výsledky ako v prípade pôvodnej úlohy, avšak v podstatne kratšom čase. Pre prípady datasetov DS5-DS8 sa dokonca podarilo nájsť optimálne riešenia, pričom v pôvodnej úlohe sa to nepodarilo. Opäť sa ukazuje, že kritérium BCC2 je vhodnejšie ako kritérium BCC1, keďže v prípade redukcií 5_BCC1 a 5_BCC2 sa pri datasete DS8 podarilo nájsť lepšie riešenie redukcii 5_BCC2 a v prípade redukcie 15_BCC1 sa pri niektorých datasetoch zhoršila spodná hranica riešenia. Z pohľadu ušetreného času opäť vychádza najlepšie redukcia 15_BCC2, ktorá ušetrila v priemere 70% času.

Tab. 4.8. Výsledky experimentov pre jednotlivé redukcie pre zimný scenár s najlepším nájdeným riešením a spodnou hranicou (*Rieš./BB*) a časom výpočtu (*Čas*) v sekundách

Data set	***5_BCC1***		***5_BCC2***		***15_BCC1***		***15_BCC2***		***Original***	
	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)	Rieš. /BB	Čas (s)
DS1	4/4	6,4	4/4	6,1	4/4	3	4/4	2,5	4/4	10,4
DS2	4/4	19,7	4/4	18,9	5/5	5,9	4/4	4,1	4/4	40
DS3	5/5	69,2	5/5	69,6	5/5	44,9	5/5	32,7	5/5	629,3
DS4	6/6	556,2	6/6	528,1	7/6	57600	6/6	117	6/6	1237
DS5	9/9	57600	9/8	57600	9/8	57600	9/8	57600	10/8	57600
DS6	10/9	57600	10/9	57600	10/10	1329	9/9	716,2	11/9	57600
DS7	-/13	57600	-/13	57600	19/14	57600	17/13	57600	-/13	57600
DS8	-/26	57600	-/26	57600	-/26	57600	-/26	57600	-/26	57600
DS9	-/28	57600	-/28	57600	-/28	57600	-/28	57600	-/28	57600
DS10	-/49	57600	-/49	57600	-/49	57600	-/49	57600	-/49	57600

V tabuľke 4.8 sú uvedené výsledky pre zimný scenár. Pri zimnom scenári boli najväčšie problémy pri pôvodnej úlohe a tento problém sa preniesol aj na redukované úlohy. Dokazuje to veľkosť úloh, ktoré sme schopní dopočítať po redukcii. Znova však musíme podotknúť, že kritérium BCC2 je vhodnejšie, keďže v prípade redukcií 5_BCC1 a 15_BCC1 sa zhoršili spodné hranice riešení pri niektorých úlohách. Najlepšie výsledky pre zimný scenár opäť poskytuje redukcia 15_BCC2, pričom dosahuje aj najlepšie časy výpočtu. Priemerný ušetrený čas za však znížil na 45%, avšak aj to je značná úspora času oproti pôvodnej úlohe.

Môžeme si však všimnúť, že v prípade datasetov DS9 a DS10 sa v prípade všetkých scenárov nepodarilo nájsť riešenie po žiadnej redukcii. Je to spôsobené hlavne tým, že tieto datasety obsahujú trolejbusové spoje, ktoré sa nepodarilo zredukovať a tak zostala veľkosť úlohy stále veľmi veľká.

4.3.4 Zhodnotenie výsledkov

V tejto časti sme predstavili heuristickú metódu na redukciu vstupov úlohy návrhu turnusov elektrických autobusov do matematického modelu založenú na spájaní dvoch spojov do jediného spoja ich zreťazením. Táto metóda teda redukuje rozsah úlohy, čím sme schopní nájsť dobré riešenia úlohy aj na úlohách väčšieho rozsahu, ktoré sme v pôvodnom tvare neboli schopní riešiť. Zároveň táto metóda značne znižuje aj čas výpočtu potrebný na riešenie úlohy.

Redukcia vstupov sa skladá z dvoch fáz, kde v prvej fáze vyberáme kandidátov (dvojice spojov) na spojenie, pričom uvažujeme iba o spájaní spojov vedúcich do okrajových častí miest a takých spojov, ktoré nasledujú krátko za sebou. Pri experimentoch sa ukázalo, že parameter *MaxTime* ovplyvňujúci maximálny čas medzi spojmi na spojenie do veľkej miery ovplyvňuje pomer redukcie voči pôvodnej úlohe. Druhá fáza metódy vyberá najvhodnejších kandidátov na spojenie, pričom sme testovali dve rôzne kritériá: BCC1 (najkratší čas medzi spojmi) a BCC2 (bod spojenia je najďalej od najbližšej nabíjačky). Z týchto dvoch kritérií sa pri experimentoch ukázalo ako lepšie kritérium BCC2, keďže nezhoršovalo spodnú hranicu úlohy a často dosahovalo aj lepšie riešenia v kratšom čase.

Takisto sa ukázalo, že datasety založené na trolejbusových spojoch nie je možné riešiť pomocou metódy na redukciu vstupov v tom tvare ako sme ju navrhli, keďže základným predpokladom bolo spájanie spojov vedúcich do okrajových častí miest a sú to jednosmerné jazdy. Trolejbusové spoje sú väčšinou cyklickým typom spojov, čiže nie sme schopný aplikovať redukčnú heuristiku.

Metóda na redukciu spojov sa v konečnom dôsledku ukázala ako veľmi vhodná na použitie, keďže značne skrátila čas výpočtu. Zároveň pri vhodnom nastavení kritérií výberu je možné získať riešenia veľmi blízke optimálnym. Pre širšie použitie však bude nutné navrhnúť iné kritériá výberu kandidátov, ktoré by boli schopné vhodne vybrať spoje aj v prípade cyklického typu spojov. Ďalším možným zlepšením pomeru redukcie by mohlo byť spájanie nielen dvoch ale aj viacerých spojov. Navyše je možné túto metódu využiť aj pri iných heuristických metódach, kde by mohla slúžiť na predspracovanie úlohy a tým zmenšiť jej rozmer a zrýchliť výpočet.

4.4 Metóda založená na metóde generovania stĺpcov

Matematický model, ktorý sme predstavili v prechádzajúcej časti, je komplexný a jeho použitie spolu so štandardným IP solverom a algoritmom vetiev a hraníc je v prípade

veľkých datasetov obmedzené. Je to hlavne z dôvodu časovej náročnosti ako aj pamäťovej náročnosti. Pomocou modelu sme teda schopný riešiť úlohy len približne o 100-150 spojoch, čo nie je vôbec dostatočné pre praktické použitie.

Preto sme sa rozhodli vyskúšať iný typ prístupu, a síce prístup na základe metódy generovania stĺpcov. Pri tejto metóde sa na problém návrhu turnusov elektrických autobusov pozeráme z iného uhla pohľadu. V prípade metódy generovania stĺpcov nebudeme matematickým modelom priamo vytvárať turnusy, ale budeme vyberať zo všetkých možných turnusov tie, ktoré sa použijú na pokrytie spojov.

Môžeme teda transformovať problém návrhu turnusov na problém, kde sa bude rozhodovať, ktoré turnusy zo všetkých dostupných prípustných turnusov musia byť použité na pokrytie turnusov tak, aby boli splnené podmienky pre návrh turnusov. Sú to podmienky, že každý spoj bude obslúžený a počas nabíjania sa na nabíjačke môže nabíjať iba jedno vozidlo v jednom čase. Túto úlohu budeme nazývať hlavný problém, respektíve master problem.

Definujeme množinu S všetkých prípustných turnusov. Prípustný turnus je taký, ktorý spĺňa všetky podmienky spotreby energie a nabíjania. Potom náš hlavný problém môže byť popísaný nasledujúcim matematickým modelom, kde binárna rozhodovacia premenná λ_s predstavuje rozhodnutie, či turnus $s \in S$ bude použitý v riešení alebo nie.

$$min \qquad \sum_{s \in S} \lambda_s \tag{4.26}$$

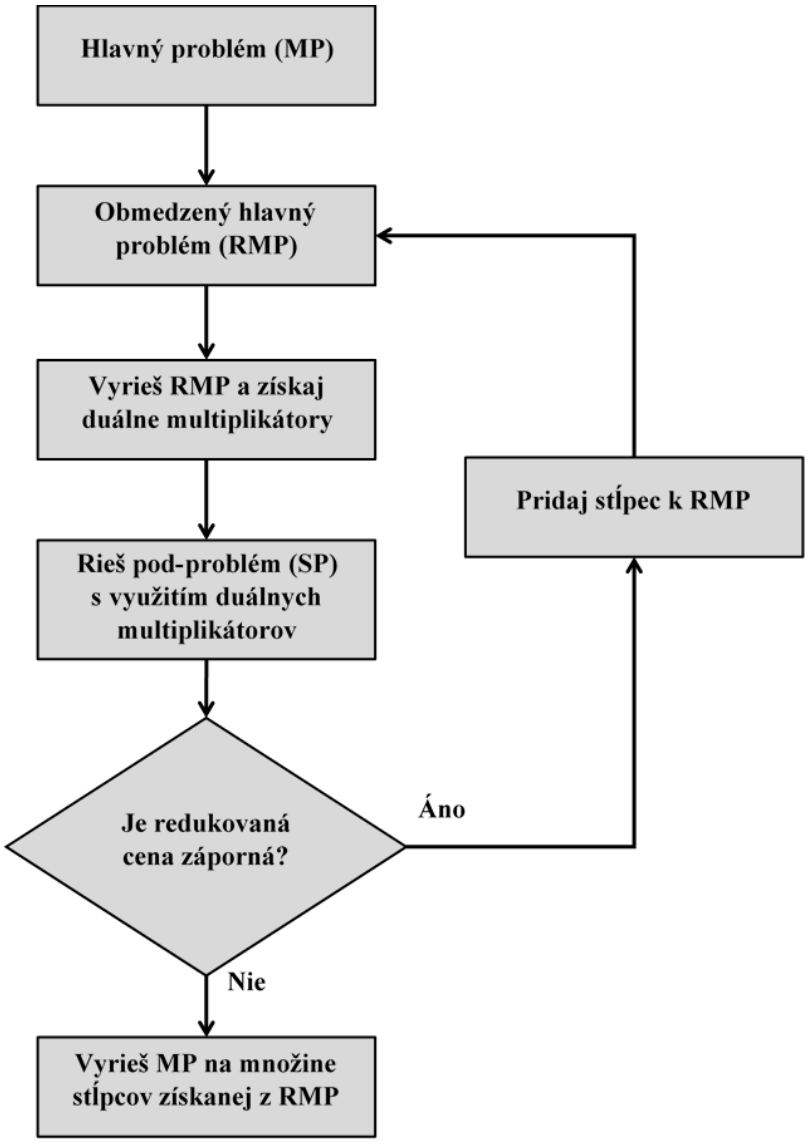
$$z.p. \qquad \sum_{s \in S} \lambda_s \left(\sum_{i \in N} a_{sij} + \sum_{r \in R} \sum_{t \in T^r} c_{srtj} \right) = 1 \quad \forall j \in N \tag{4.27}$$

$$\sum_{s \in S} \lambda_s \left(\sum_{i \in N} b_{sirt} + d_{sr(t-1)} \right) \leq 1 \quad \forall r \in R, t \in T^r \tag{4.28}$$

$$\lambda_s \in \{0,1\} \quad \forall s \in S \tag{4.29}$$

Účelová funkcia (4.26) minimalizuje počet použitých turnusov, čo zodpovedá minimalizácii počtu použitých elektrických autobusov, nakoľko jeden turnus je obsluhovaný jedným elektrickým autobusom. Podmienka (4.27) slúži ako záruka, že každý spoj bude obslúžený. Konštanty a_{sij} sú 1 v prípade, že spoj j je obslúžený hneď po spoji i počas turnusu s a konštanty c_{srtj} sú 1, ak je spoj j obslúžený hneď po nabíjaní počas nabíjacej udalosti t na nabíjačke r počas turnusu s. Podmienka (4.28) reprezentuje obmedzenie, že

počas jednej nabíjacej udalosti sa môže nabíjať iba jeden elektrický autobus, čo zodpovedá podmienke, že na nabíjačke sa v jednom čase môže nabíjať iba jediný elektrický autobus. Konštanty b_{sirt} v tejto podmienke predstavujú prechod medzi spojom i a nabíjacou udalosťou t na nabíjačke r počas turnusu s. Konštanty $d_{sr(t-1)}$ sú pokračovaním nabíjania medzi nabíjacou udalosťou t a predchádzajúcou nabíjacou udalosťou $t-1$ na rovnakej nabíjačke r počas turnusu s. Podmienky (4.29) sú podmienkami binárnosti pre každú premennú λ_s.



Obr. 4.5. Schéma metódy generovania stĺpcov pre úlohu návrhu turnusov elektrických autobusov

Tento pohľad na problém je špecifický pre použitie metódy generovania stĺpcov. Keďže metóda generovania stĺpcov je iteratívna metóda, ktorá pozostáva z opakovaného riešenia dvoch problémov musíme zadefinovať dve úlohy a síce: hlavný problém a pod-

problém. Následne bude celkové riešenie úlohy prebiehať podľa schémy metódy generovania stĺpcov, ktorú môžeme vidieť na obrázku 4.5.

4.4.1 Riešenie hlavného problému

Hlavný problém je definovaný matematickým modelom (4.26)-(4.29). Avšak, počet všetkých prípustných turnusov je obrovský a aj pre malé úlohy je časovo náročné nájsť všetky prípustné turnusy. Preto sa hlavný problém transformuje na obmedzený hlavný problém popísaný matematickým modelom (4.30)-(4.33), ktorý je identický s hlavným problémom s tým rozdielom, že úlohu rieši len na obmedzenej množine turnusov P, ktorá je podmnožinou všetkých prípustných turnusov S.

$$min \qquad \sum_{s \in P} \lambda_s \tag{4.30}$$

$$z.p. \qquad \sum_{s \in P} \lambda_s \left(\sum_{i \in N} a_{sij} + \sum_{r \in R} \sum_{t \in T^r} c_{srtj} \right) \geq 1 \quad \forall j \in N \mid \pi_j \geq 0 \tag{4.31}$$

$$\sum_{s \in P} \lambda_s \left(\sum_{i \in N} b_{sirt} + d_{sr(t-1)} \right) \leq 1 \quad \forall r \in R, t \in T^r \mid \sigma_{rt} \leq 0 \tag{4.32}$$

$$\lambda_s \in \{0,1\} \quad \forall s \in P \subset S \tag{4.33}$$

V modeli sme zamenili podmienku (4.27) za (4.31), v ktorej sme relaxovali podmienku, že každý spoj musí byť obslúžený práve jedným elektrickým autobusom na podmienku, že každý spoj bude obslúžený aspoň jedným vozidlom. V praxi je táto úprava často používaná, nakoľko je splnená podmienka, že každý spoj je obslúžený. Zároveň je jednoduché upraviť turnusy získané riešením, tak aby sa v dvoch turnusoch neobsluhoval ten istý spoj. Dá sa to urobiť tak, že z jedného turnusu tento spoj odstránime a upravíme turnus podľa nutných obmedzení a podmienok. V našom prípade sa turnus upraví tak, že sa upraví rozvrh nabíjania, pričom vieme, že po odstránení spoja z turnusu sa bude nabíjať menej, keďže na spoji sa zvyčajne spotrebuje podstatne viac energie ako na prejazde medzi spojmi.

Pri riešení obmedzeného hlavného problému musíme začať prípustným riešením úlohy návrhu turnusov elektrických autobusov. Pre počiatočné riešenie sme zvolili kyvadlové jazdy, kde je každý spoj obslúžený iným elektrickým autobusom a každý turnus tvorí iba obsluha jediného spoja. V tejto fáze riešime LP-relaxáciu obmedzeného hlavného problému, kde namiesto podmienky celočíselnosti (4.33) vyžadujeme len $\lambda_s \geq 0$. Riešením

LP-relaxácie obmedzeného hlavného problému zároveň získame duálne multiplikátory π_j pre každý spoj a σ_{rt} pre každú nabíjaciu udalosť.

Štandardne je možné riešiť hlavný problém ako aj obmedzený hlavný problém pomocou štandardného IP solvera. Ako bude ukázané aj pri experimentoch, tak časová a výpočtová náročnosť obmedzeného hlavného problému nie je veľmi vysoká, čiže sme sa rozhodli použiť na riešenie obmedzeného hlavného problému štandardný IP solver Xpress IVE s použitím simplexovej metódy pre riešenie LP-relaxácie a metódy vetiev a hraníc pre riešenie celočíselnej úlohy hlavného problému.

4.4.2 Riešenie pod-problému

Druhý problém, nazývaný pod-problém, sa používa na vygenerovanie nových prípustných turnusov založených na riešení obmedzeného hlavného problému, ktoré by zlepšili riešenie obmedzeného hlavného problému, ak ich pridáme do množiny, s ktorou pracuje obmedzený hlavný problém. V pod-probléme potrebujeme nájsť prípustnú sekvenciu spojov a nabíjacích udalostí, ktoré by neporušili podmienky spotreby energie a nabíjania batérie.

Inak povedané, v pod-probléme sa snažíme nájsť turnus, ktorý bude mať zápornú redukovanú cenu po pridaní do podmnožiny turnusov v obmedzenom hlavnom probléme, a tak by mohol zlepšiť účelovú funkciu obmedzeného hlavného problému. Redukovaná cena turnusu (stĺpca) je našom prípade nasledovná:

$$c_s^* = 1 - \sum_{j \in N} \pi_j \left(\sum_{i \in N} a_{sij} + \sum_{r \in R} \sum_{t \in T^r} c_{srtj} \right) - \sum_{r \in R} \sum_{t \in T^r} \sigma_{rt} \left(\sum_{i \in N} b_{sirt} + d_{sr(t-1)} \right) \quad (4.34)$$

Čiže v pod-probléme sa snažíme minimalizovať redukovanú cenu turnusu. Táto úloha môže byť vyjadrená modelom s účelovou funkciou (4.35) a prídavnou podmienkou (4.36), ktorá zabezpečuje, že z ranného depa vyjde iba jediné vozidlo. Zároveň musíme zahrnúť aj podmienky toku vozidla, podmienky spotreby energie ako aj podmienky nabíjania (4.4)-(4.25). Tieto podmienky môžeme použiť z predchádzajúceho matematického modelu, pričom aj definícia rozhodovacích premenných je identická.

$$\begin{aligned} min \quad & 1 - \sum_{i \in N} \sum_{j \in F_j} \pi_j x_{ij} - \sum_{i \in N} \sum_{r \in R} \sum_{t \in F c_{ri}} \pi_j y_{irt} \\ & - \sum_{j \in N} \sum_{r \in R} \sum_{t \in B c_{rj}} \sigma_{rt} z_{rtj} - \sum_{r \in R} \sum_{t \in T^r} \sigma_{rt} w_{r(t-1)} \end{aligned} \quad (4.35)$$

$$z.p. \qquad \sum_{j \in F_{D_0}} x_{D_0 j} + \sum_{r \in R} \sum_{t \in Fc_{rD_0}} y_{D_0 rt} = 1 \qquad (4.36)$$

Pod-problém sme najprv riešili pomocou matematického modelu (4.4)-(4.25) a (4.34)-(4.35). Túto metódu nazveme SP-mod. Je to prvá verzia riešenia pod-problému. Ako si však môžeme všimnúť, tento model je veľmi podobný matematickému modelu úlohy návrhu turnusov, ktorý sme predstavili v predchádzajúcej kapitole. Experimenty s pôvodným modelom ukázali, že tento problém je veľmi náročný na výpočtový čas, čo na riešenie pod-problému v rámci metódy generovania stĺpcov nie je ideálne. Preto by bolo vhodné implementovať iný typ metódy na riešenie pod-problému alebo aspoň minimalizovať počet behov, počas ktorých sa pod-problém bude riešiť pomocou matematického modelu. Táto metóda nemusí byť exaktnou metódou, z dôvodu, že metóda generovania stĺpcov nevyžaduje nutne exaktnú metódu na riešenie pod-problému, ale metódu, ktorá poskytne riešenie so zápornou redukovanou cenou. Zároveň by mala byť táto metóda veľmi rýchla, nakoľko metóda generovania stĺpcov je iteratívny proces, čiže sa bude spúšťať veľa krát.

Metóda SP-Yen

Keď sa pozrieme na definíciu pod-problému, zistíme, že táto úloha môže byť transformovaná na úlohu hľadania najkratšej prípustnej cesty na acyklickom digrafe so všeobecnou cenou hrany, pričom prípustnosť cesty je obmedzená podmienkami pre spotrebu energie a nabíjanie. Vrcholy digrafu zodpovedajú spojom a nabíjacím udalostiam. Hrany digrafu sú tvorené možnými prechodmi medzi jednotlivými spojmi a nabíjacími udalosti, pričom cena hrany je určená duálnym multiplikátorom získaným v obmedzenom hlavnom probléme. Pri účelovej funkcii (4.35) vidíme, že minimalizujeme pre každú premennú záporne vzatú hodnotu duálnych multiplikátorov a tieto záporne vzaté hodnoty budú použité ako cena orientovanej hrany vstupujúcej do vrcholu. V prípade spojov j to bude cena definovaná multiplikátorom π_j a v prípade nabíjacej udalosti t na nabíjačke r to bude multiplikátor σ_{rt}. Keď to zhrnieme, v pod-probléme budeme hľadať sekvenciu spojov a nabíjacích udalostí, ktorá má minimálnu cenu a spĺňa obmedzenia nabíjania počas nabíjacích udalostí a obmedzenia spotreby energie počas spojov a presunov.

Pre hľadanie najkratšej cesty existuje niekoľko polynomiálnych algoritmov ako napríklad Dijkstrov algoritmus a A* algoritmus. Avšak tieto algoritmy neuvažujú ďalšie obmedzenia cesty. S touto základnou myšlienkou na použitie algoritmu najkratšej cesty sme

navrhli prvú metódu na riešenie pod-problému. Nazvali sme ju SP-Yen. Táto metóda je popísaná algoritmom 1.

Algorimus 1: *SP-Yen*

Vstup: Graf G spojení medzi spojmi a nabíjacími udalosťami

$k := 1$
Repeat
    cesta:=Nájdi *k*-tu najkratšiu cestu (turnus)
    If *cesta* je prípustná: stop
    else $k := k + 1$
Until $k < kMax$ alebo *cesta* je prípustný turnus

Výstup: prípustný turnus

Základnou myšlienkou na hľadanie najkratšej prípustnej cesty je generovanie najkratších ciest z ranného do večerného depa a následné otestovanie, či sú prípustné. Na generovanie najkratších ciest sme sa rozhodli použiť algoritmus na hľadanie *k*-najkratších ciest. Pre implementáciu algoritmu sme použili Yenov algorimus [74], pretože hľadá najkratšie cesty iteratívnym spôsobom. Nájdenie najkratšej cesty je síce rýchle, avšak nájdenie *k*-tej najkratšej cesty môže byť na veľmi veľkých grafoch časovo náročné, zvlášť v prípade, že k bude vysoké. Preto sme sa rozhodli obmedziť maximálny počet najkratších ciest, ktoré sa budú hľadať, parametrom *kMax*. Ak je *k*-ta najkratšia cesta prípustná, tak je pridaná k množine v obmedzenom hlavnom probléme. V prípade, že po otestovaní všetkých *kMax*-najkratších ciest nenájdeme ani jednu prípustnú cestu, tak sa spustí matematický model na nájdenie najkratšej prípustnej cesty popísaný metódou SP-mod. Na základe matematického modelu sa potom rozhodne, či sme našli najkratšiu prípustnú cestu a metóda bude pokračovať riešením obmedzeného hlavného problému, alebo sa algoritmus skončí.

Na otestovanie či je najkratšia cesta prípustná je použitá heuristika. V tejto heuristike sa postupne prechádza cez vrcholy cesty a vyhodnocujú sa podmienky spotreby a nabíjania. Na začiatku turnusu je batéria plne nabitá. Pri prejazde sa hladina energie zníži o príslušnú hodnotu. Takisto pri prechode spojom sa aktuálna hodnota energie zníži o energiu potrebnú na prejdenie spoja. V prípade nabíjacej udalosti sa energia dobíja, čiže aktuálna hladina energie zvýši. V heuristike je použitá stratégia, že vždy sa nabíja toľko energie koľko je možné. V praxi to znamená, že sa vyhodnotí, koľko času je k dispozícii na nabíjanie (obmedzené začiatkom nasledujúcej úlohy) a následne sa bude nabíjať počas celého dostupného času. V prípade, že by sa mala prekročiť maximálna kapacita, tak sa batéria iba nabije do plnej kapacity, zvyšný čas sa nenabíja.

Metóda SP-YenM

Inšpirovaný prácami [20] a [26] sme navrhli druhú metódu na riešenie pod-problému nazvanú SP-YenM, ktorá sa snaží využiť všetky tieňové ceny (duálne multiplikátory) získané riešením obmedzeného hlavného problému. Táto metóda je popísaná algoritmom 2.

Metóda SP-YenM funguje podobne ako predchádzajúca metóda SP-Yen s tým rozdielom, že vyhľadáva ďalšie cesty po nájdení najkratšej prípustnej cesty. Čiže, metóda SP-YenM hľadá najkratšiu prípustnú cestu opäť pomocou algoritmu na nájdenie *k*-najkratších ciest spolu s overovaním prípustnosti pomocou heuristiky. Avšak po nájdení najkratšej prípustnej cesty sa spúšťa funkcia *Multi* popísaná algoritmom 3.

Algoritmus 2: *SP-YenM*

Vstup: Graf G spojení medzi spojmi a nabíjacími udalosťami

$k := 1$
Repeat
    cesta:=Nájdi *k*-tu najkratšiu cestu (turnus)
    If *cesta* je prípustná:
        Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
        Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
        stop
    else $k := k + 1$
Until $k < kMax$ alebo *cesta* je prípustný turnus

Výstup: zoznam prípustných turnusov

Prídavná funkcia *Multi* sa snaží nájsť ďalšie turnusy, ktoré sú komplementárnymi k prvému nájdenému turnusu. Metóda funguje nasledovne. Po nájdení prvého prípustného turnusu pomocou Yenovho algoritmu pre *k*-najkratších ciest je vytvorený nový redukovaný digraf. Tento vznikne tak, že odstránime všetky vrcholy použité v prvom turnuse z digrafu aj so všetkými hranami incidentnými s týmito vrcholmi. Následne sa na tomto redukovanom digrafe opätovne spustí algoritmus na nájdenie najkratšej prípustnej cesty pomocou Yenovho algoritmu na hľadanie *k*-najkratších ciest a kontroly prípustnosti pomocou heuristiky. Tento proces je ukončený v prípade, že dosiahneme parameter *kMax* alebo vygenerovaný prípustný turnus nemá zápornú redukovanú cenu. Po nájdení druhého turnusu sa opakuje ten istý scenár, čiže opäť zredukujeme graf a hľadám ďalší prípustný turnus so zápornou redukovanou cenou. Po skončení metódy *Multi* je k dispozícii skupina komplementárnych turnusov, ktoré sa pridajú do podmnožiny v obmedzenom hlavnom probléme. Samozrejme, ak sa v prvom kroku nepodarí nájsť prípustný turnus, tak sa spustí matematický model na získanie prípustného turnusu.

Algoritmus 3: Funkcia *Multi*

Vstup: Redukovaný graf G' spojení medzi spojmi a nabíjacími udalosťami

$k := 1$
Repeat
cesta:=Nájdi *k*-tu najkratšiu cestu (turnus) na grafe G'
If *cesta* je prípustná a má negatívnu redukovanú cenu:
Z grafu G' odstráň vrcholy použité v *cesta*
Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
stop
else $k := k + 1$
Until $k < kMax$ alebo *cesta* je prípustný turnus

Výstup: zoznam prípustných turnusov

Konvergencia riešenia pomocou tejto metódy sa ukazuje byť omnoho rýchlejšia ako v prípade vygenerovania len jediného turnusu. Je to hlavne z dôvodu, že komplementárne turnusy tvoria dohromady časť dobrého riešenia a preto vedia účelovú funkciu obmedzeného problému rýchlo stlačiť nadol.

Metóda SP-TYenM

Tretia metóda SP-TYenM je založená na analýze matematických modelov pre obmedzený hlavný problém a pod-problém spolu s problémom hľadania najkratšej cesty. Keď sme používali prvú a druhú metódu, všimli sme si, že Yenov algoritmus je založený na hľadaní ďalších najkratších ciest v okolí prvej najkratšej cesty, a teda sú tieto ďalšie najkratšie cesty veľmi podobné prvej najkratšej ceste.

Algoritmus 4: *SP-TYenM*

Vstup: Graf G spojení medzi spojmi

$k := 1$
Repeat
cesta:=Nájdi *k*-tu najkratšiu cestu (turnus)
If *cesta* je prípustná:
Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
stop
else *cesta*:=Pridaj nabíjacie udalosti do *cesta*
If *cesta* je prípustná:
Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
stop
else $k := k + 1$
Until $k < kMax$ alebo *cesta* je prípustný turnus

Výstup: zoznam prípustných turnusov

V našom prípade sa často najkratšie cesty líšili iba v navštívených nabíjacích udalostiach. Čo sa týka spojov, tak tie boli väčšinou rovnaké pre veľké množstvo

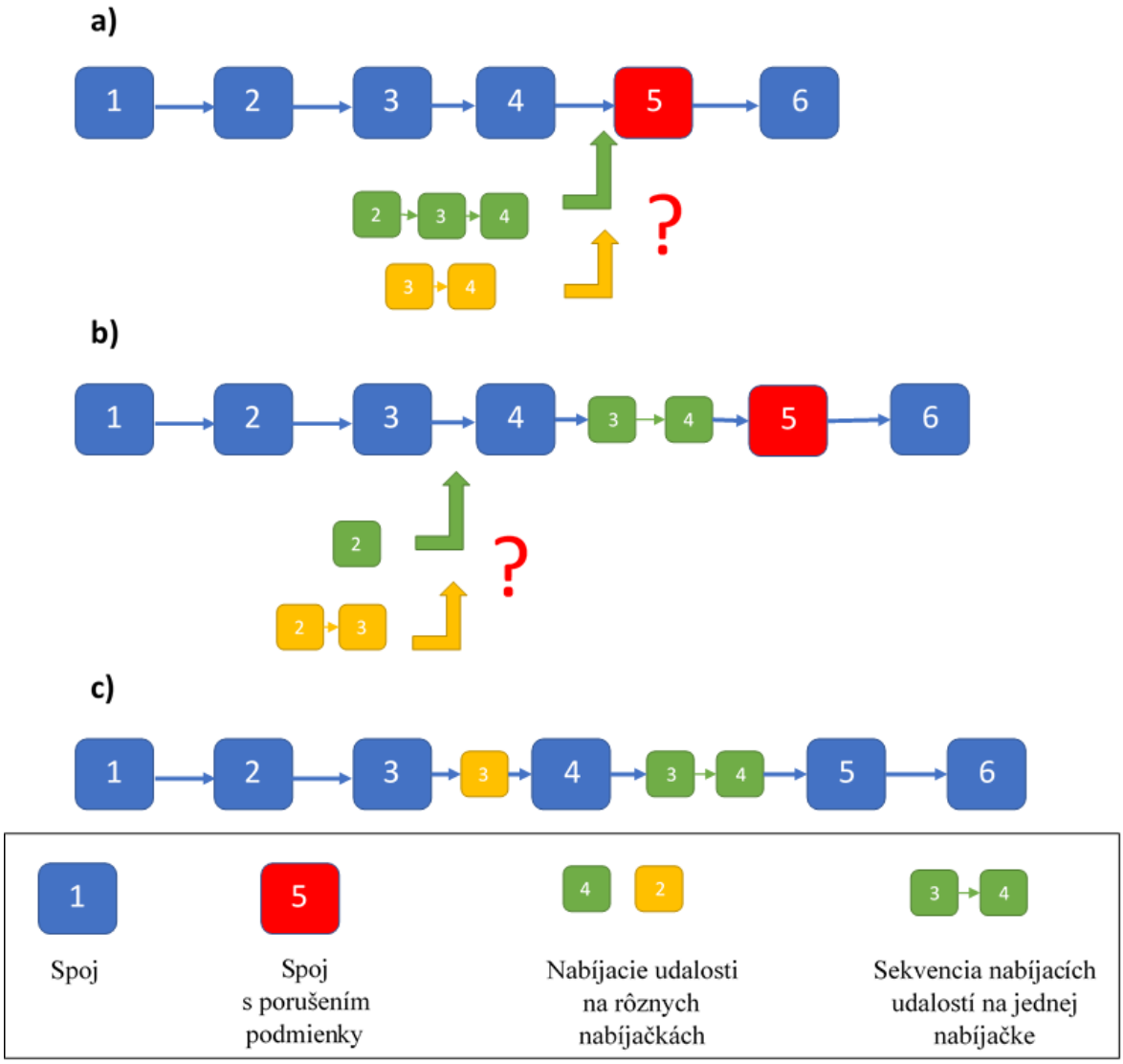
vygenerovaných najkratších ciest (turnusov). Keď sme analyzovali účelovú funkciu pod-problému, zistili sme, že kvalita vygenerovaného turnusu (cena cesty) môže byť zlepšená iba pridávaním spojov. Toto je založené na tom, že v účelovej funkcii sa každá tieňová cena spoja π_j, pre každý spoj j, berie ako záporná hodnota kvôli podmienke $\pi_j \geq 0$, ktorými je definovaná tieňová cena v obmedzenom hlavnom probléme. Z definície tieňovej ceny $\sigma_{rt} \leq 0$ pre každú nabíjacie udalosť t na nabíjačke r a účelovej funkcie pod-problému vidíme, že zahrnutie nabíjacej udalosti do turnusu nezlepší celkovú cenu turnusu. Navyše, v prípade $\sigma_{rt} < 0$ bude tieňová cena v účelovej funkcii kladná a pri použití nabíjacej udalosti zhorší účelovú funkciu pod-problému.

Z týchto vedomostí sme navrhli heuristický algoritmus ako tretiu metódu na nájdenie najkratšej prípustnej cesty, ktorá je ukázaná v algoritme 4. Táto heuristika funguje veľmi podobne ako prvá metóda a skladá sa z dvoch fáz. V prvej fáze sa hľadá najkratšia cesta len na digrafe tvorenom spojmi a ranným a večerným depom, čiže sa z digrafu úplne odstránili vrcholy reprezentujúce nabíjacie udalosti aj s ich incidentnými hranami. Po nájdení najkratšej cesty nastupuje druhá fáza, v ktorej sa snažíme vytvoriť prípustný turnus z tejto najkratšej cesty pomocou pridávania nabíjacích udalostí do turnusu. Samozrejme, ak najkratšia cesta spĺňa podmienky pre spotrebu energie, tak nie je nutné pridávanie nabíjacích udalostí.

Druhá fáza sa vykonáva pomocou heuristiky zobrazenej na obrázku 4.5 a pracuje nasledovne. Najprv je nutné identifikovať, ktorý spoj porušuje podmienku spotreby energie (obr. 4.5a). Na tento účel je použitá tá istá funkcia ako funkcia z prvej metódy, ktorá zisťuje, či je turnus prípustný. Keď sme našli problematický spoj, budeme postupovať smerom od tohto spoja k začiatku turnusu. Zoberieme dva za sebou nasledujúce spoje turnusu a vyhľadáme všetky možné sekvencie nabíjacích udalostí medzi týmito dvomi spojmi na všetkých nabíjačkách.

Možná sekvencia sa skladá z nabíjacích udalostí prebiehajúcich na rovnakej nabíjačke, na ktoré je možné sa dostať počas času medzi dvomi vybranými spojmi (zelená a žltá sekvencia na obr. 4.5a). Pre všetky vybrané nabíjacie udalosti musí platiť podmienka, že sú nasledovníkmi prvého spoja a predchodcami druhého spoja. Navyše musí platiť, že je možné tieto nabíjacie udalosti zreťaziť, inak povedané nasledujú v čase jedna za druhou. Keď získame všetky nabíjacie udalosti spĺňajúce predchádzajúce podmienky, začneme

vytvárať sekvencie nabíjacích udalostí sekvenčným spracovaním získaním nabíjacích udalostí.



Obr. 4.6. Príklad heuristiky na pridávanie nabíjacích udalostí, kde riešime problém so spojom 5.

Pracujeme vždy po nabíjačkách, pridávame nabíjacie udalosti do sekvencie, až kým nenarazíme na nabíjaciu udalosť, ktorá má tieňovú cenu $\sigma_{rt} < 0$. Túto nabíjaciu udalosť odstránime zo sekvencie a spracovania, pretože jej tieňová cena indikuje, že môže zhoršiť redukovanú cenu turnusu. Keďže by pridaním ďalšej nabíjacej udalosti došlo k porušeniu podmienky, že nabíjacie udalosti nasledujú v čase za sebou, ukončíme vytváranie sekvencie. Ak ešte zostali na danej nabíjačke nespracované nabíjacie udalosti, budeme pokračovať vytváraním ďalšej sekvencie. Zároveň, ak sa pri spracovaní nabíjacích udalostí stretneme s nabíjacou udalosťou, ktorá už bola použitá, tak sa odstráni zo sekvencie ako aj spracovávania podobne ako v prípade zápornej tieňovej ceny. Toto sa môže stať v prípade,

že generujeme niekoľko turnusov počas jedného pod-problému ako v prípade metódy SP-YenM (Algoritmus 3). Keď sme skončili so spracovávanie nabíjacích udalostí na jednej nabíjačke, pokračujeme vytváraním sekvencií na ďalších nabíjačkách.

Potom, čo sme vytvorili zoznam všetkých možných sekvencií (žltá a zelená sekvencia na obr. 4.6a), začneme s pridávaním nabíjacej sekvencie do turnusu a testovaním, či pridanie danej sekvencie pomohlo vyriešiť problém s porušením podmienky spotreby energie (obr. 4.6a). Ak pridanie sekvencie pomohlo, pokračujeme s testovaním celého turnusu, či je prípustný alebo sa vyskytol ďalší problém s porušením podmienky spotreby energie pri nasledujúcich spojoch. Ak opäť objavíme problém, budeme ho riešiť rovnakým spôsobom ako ten prechádzajúci.

Ak sa nepodarilo vyriešiť problém, otestujeme ďalšiu sekvenciu. Ak sa nepodarilo vyriešiť problém ani po otestovaní poslednej sekvencie, tak sa vyberie sekvencia, pri ktorej sa nabije najviac energie (obr. 4.6b), tá sa pridá do turnusu a pokračuje sa ďalšou dvojicou spojov, ktoré predchádzali skúmanej dvojici spojov. Pre túto novú dvojicu sa opätovne opakuje proces pridávania sekvencie nabíjacích udalostí. Týmto spôsobom sa pokračuje až kým nedosiahneme prípustný turnus (obr. 4.6c) alebo nedosiahneme začiatok turnusu, čo značí, že nie je možné daný turnus upraviť na prípustný turnus pomocou pridávania nabíjacích udalostí.

Ak heuristika nie je schopná získať prípustný turnus z najkratšej cesty, vrátime sa späť k Yenovmu algoritmu na hľadanie *k*-najkratších ciest a nájdeme ďalšiu najkratšiu cestu na redukovanom grafe zo spojov. Ďalšiu najkratšiu cestu opätovne otestujeme, či ju je možné upraviť na prípustnú pomocou heuristiky na pridávanie nabíjacích udalostí do turnusu. V prípade, že dosiahneme hodnotu parametra *kMax* pre hľadanie *k*-najkratších ciest a nepodarilo sa nám nájsť prípustný turnus, pokračujeme hľadaním turnusu so zápornou redukovanou cenou pomocou matematického modelu.

Tretia metóda na nájdenie turnusu so zápornou redukovanou cenou (SP-TYenM) využíva tiež rovnakú myšlienku ako druhá metóda, a síce hľadanie viacerých turnusov počas riešenia jedného pod-problému. Podobne ako v prípade druhej metódy, ak sa hľadá zlepšujúci turnus pomocou matematického modelu, tak sa nehľadá viacero turnusov v jednom pod-probléme.

Metóda SP-TYenMD

Táto posledná metóda nazvaná SP-TYenMD vychádza z tretej metódy, s tým rozdielom, že je do nej pridaná heuristika na odstraňovanie spojov z vygenerovaného turnusu (Algoritmus 5). Heuristika na odstraňovanie spojov je aplikovaná iba v prípade, že je heuristika na pridávanie nabíjacích udalostí neúspešná pri generovaní prípustného turnusu pre všetkých *kMax*-najkratších ciest. Rovnako ako aj pri ostatných metódach, v prípade, že je heuristika na odstraňovanie spojov neúspešná, aplikuje sa hľadanie pomocou matematického modelu.

Základná myšlienka, za ktorou je odstraňovanie je, že po odstránení spoja vznikne v turnuse časová medzera v dĺžke odstráneného spoja, čo poskytne možnosť na nabíjanie elektrického autobusu, ktorý bude môcť následne pokračovať v turnuse.

Algoritmus 5: *SP-TYenMD1*

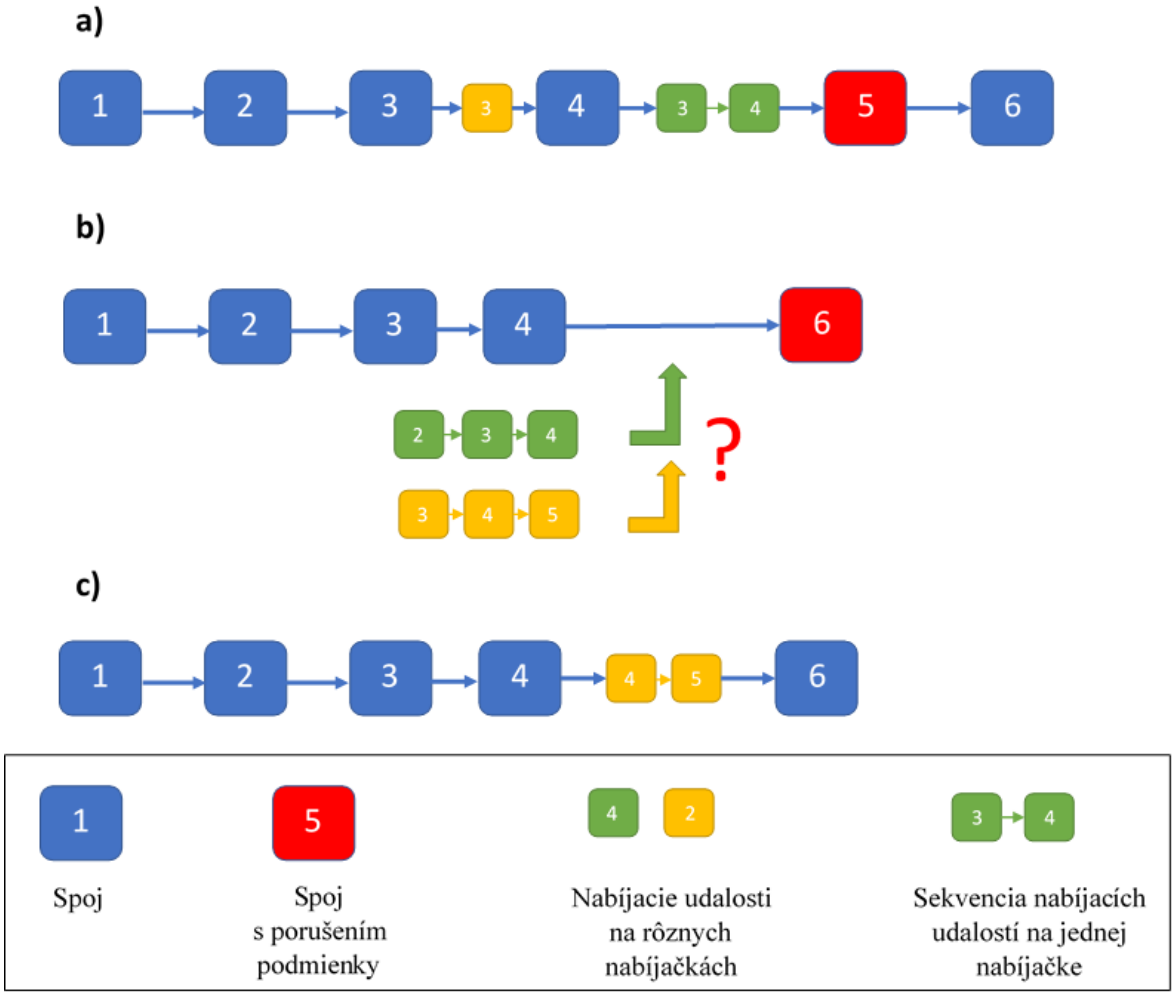
Vstup: Graf G spojení medzi spojmi

$k := 1$
Repeat
  cesta:=Nájdi *k*-tu najkratšiu cestu (turnus)
  If *cesta* je prípustná:
    Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
    Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
    stop
  else *cesta*:=Pridaj nabíjacie udalosti do *cesta*
  If *cesta* je prípustná:
    Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
    Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
    stop
  else $k := k + 1$
Until $k < kMax$ alebo *cesta* je prípustný turnus

$k := 1$
Repeat
  $d := 1$
  repeat
    cesta:= Uprav $cesta$ odstránením d spojov
    If *cesta* je prípustná:
      Vytvor redukovaný graf $G' ->$ odstráň vrcholy použité v *cesta*
      Funkcia *Multi* – nájdi ďalší prípustný turnus na grafe G'
      stop
    Else $d := 1$
  until $d < dMax$ or *cesta* je prípustný turnus
  $k := k + 1$
Until $k < kMax$ *cesta* je prípustný turnus

Výstup: zoznam prípustných turnusov

Heuristika na odstraňovanie spojov pracuje podobne ako heuristika na pridávanie nabíjacích udalostí (obr. 4.7). Najprv heuristika aplikuje heuristiku na pridávanie nabíjacích udalostí a vytvorí turnus obsahujúci nabíjacie udalosti, aj keď tento turnus je neprípustný. Ďalej v tomto turnuse nájde spoj, ktorý je problematický a spôsobuje porušenie podmienky pre spotrebu energie (obr. 4.7a). Tento problematický spoj je následne odstránený z čistého turnusu bez nabíjacích udalostí, ktorý bol vygenerovaný algoritmom na vyhľadanie *k*-najkratších ciest (obr. 4.7b). Na záver je opätovne aplikovaná heuristika na pridávanie nabíjacích udalostí (obr. 4.7c). Ak bolo odstránenie spoja spolu s pridaním nabíjacích udalostí úspešné, heuristika vyprodukovala prípustný turnus. Ak odstránenie spoja nepomohlo na vygenerovanie prípustného turnusu, tak odstraňovanie bude pokračovať na turnuse s už odstráneným spojom. Maximálny počet odstránených spojov je ohraničený parametrom *dMax*.



Obr. 4.7: Príklad heuristiky na odstránenie spojov z turnusu, pri problematickom spoji 5, ktorý sa odstráni

V našej implementácii sme testovali dva rozdielne prístupy k odstraňovaniu spojov z turnusov vygenerovaných algoritmom na vyhľadávanie k-najkratších ciest. Prvou alternatívou je sekvenčná aplikácia heuristiky na odstraňovanie spojov, označovaná SP-TYenMD1. V tejto verzii spracúvame vygenerované najkratšie cesty (turnusy) od najkratšej a pre každú z nich sa vykonáva odstraňovanie postupne až do maxima $dMax$ spojov.

Druhou alternatívou je cyklická aplikácia heuristiky na odstraňovanie spojov, označovaná ako SP-TYenMD2. Táto verzia aplikuje heuristiku cyklicky, čo znamená, že heuristika na odstraňovanie spojov najprv odstraňuje po jednom spoji zo všetkých turnusov vygenerovaných algoritmom na hľadanie k-najkratších ciest. V prípade, že odstránenie jedného spoja z turnusu nie je dostatočné ani pre jeden vygenerovaný turnus, heuristika zvýši počet odstraňovaných spojov o jeden a opakuje proces odstraňovania postupne na všetkých vygenerovaných turnusoch do nového maximálneho počtu odstránených spojov. Heuristika končí neúspešne vtedy, keď maximálny počet odstránených spojov dosiahne parameter $dMax$. V prípade, že heuristika počas vykonávania nájde prípustný turnus, je ukončená.

4.4.3 Výsledky experimentov

V prípade predstavených metód založených na metóde generovaní stĺpcov sa pri experimentoch testovala výkonnosť metód vzhľadom na kvalitu riešenia a časovú náročnosť. Experimenty prebehli na všetkých datasetoch a so všetkými scenármi pre obdobie jari, leta a zimy.

Pre spustenie experimentov bolo nutné nastaviť niektoré parametre definované pri metódach. Čas pre výpočet jedného experimentu bol stanovený na 24 hodín. Ďalej pre výpočet jedného pod-problému bol stanovený maximálny čas tri hodiny, v prípade, že sa použije výpočet pomocou matematického modelu. Parameter $kMax$, ktorý definuje maximálny počet najkratších ciest vygenerovaných algoritmom k-najkratších ciest vo všetkých metódach kde sa používa, bol zafixovaný na hodnotu $kMax = 300$. Nakoniec, parameter $dMax$, ktorý definuje maximálny počet spojov odstraňovaných v metódach SP-TYenMD1 a SP-TYenMD2, bol stanovený na $dMax = 5$ pre datasety DS1-DS8. Pre datasety DS9 a DS10 bol parameter stanovený na $dMax = 7$, kvôli veľkosti úlohy.

Experimenty sa riešili v dvoch fázach. V prvej fáze sa testovali všetky metódy na datasetoch DS1-DS6. Keďže niektoré z navrhnutých metód sa pri týchto experimentoch ukázali ako nie veľmi vhodné na aplikáciu pri väčších úlohách, tak sme ich následne vylúčili z druhej fázy experimentov, kde sa metódy testovali na väčších datasetoch DS7-DS10.

Tabuľka 4.9: Výsledky experimentov s pri jarnom scenári

Dataset	***Najl. výsledok /BB***	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	4/4	4	5	5	5	5	5
DS2	4/4	5	5	5	5	5	5
DS3	5/5	8	6	6	6	6	6
DS4	6/6	9	8	7	7	7	7
DS5	8/8	13	12	9	9	9	9
DS6	9/9	16	14	10	10	10	10

Riešenia získané pomocou prezentovaných metód môžeme vidieť v tabuľkách 4.9, 4.10 a 4.11. Tabuľka 4.9 obsahuje hodnoty výslednej účelovej funkcie pre prípad jarného scenára, ako aj porovnanie s najlepšími nájdenými výsledkami získanými matematickým modelom (4.1)-(4.25), ktoré boli prezentované v prechádzajúcej kapitole. Tieto hodnoty sú obsiahnuté v stĺpci *Najl. výsledok/BB* aj so spodnou hranicou riešenia. Ostatné stĺpce obsahujú riešenia získané jednotlivými metódami. Ako môžeme vidieť, metóda SP-mod je z pohľadu výsledkov takmer vždy najhoršia spomedzi všetkých navrhovaných metód. Všetky metódy, ktoré využívajú generovanie viacerých turnusov počas jedného pod-problému majú v prípade jarného scenára identické výsledky. Takisto vidíme, že metóda SP-Yen nedosahuje až také dobré výsledky v porovnaní s ostatnými metódami, avšak je lepšia ako riešenie pod-problému iba pomocou matematického modelu a IP solvera.

Tabuľka 4.10: Výsledky experimentov s pri lenom scenári

Dataset	***Najl. výsledok /BB***	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	4/4	5	5	5	5	5	5
DS2	4/4	9	7	5	4	5	5
DS3	5/5	10	8	8	6	6	6
DS4	6/6	16	11	9	7	7	7
DS5	9/8	30	14	12	9	10	10
DS6	10/9	123	22	20	14	10	10

V tabuľke 4.10 nájdeme výsledky pre letný scenár. Opätovne, stĺpec *Najl. výsledok/BB* obsahuje najlepšie nájdené riešenie pomocou matematického modelu a spodnú hranicu riešenia. Podobne ako v prípade jarného scenára vidíme, že postupná aplikácia jednotlivých metód zlepšuje získané výsledky. Celkovo najlepšie výsledky poskytli metódy SP-TYenMD1 a SP-TYenMD2. Zároveň si môžeme všimnúť, že metóda SP-TYenM tiež získala veľmi dobré výsledky, v prípade datasetov DS2 a DS5 bola dokonca lepšia ako

metódy SP-TYenMD1 a SP-TYenMD2. Môžeme si všimnúť, že so zväčšujúcou sa veľkosťou úlohy sa gap medzi spodnou hranicou riešenia a získanými riešeniami pomocou väčšiny metód značne zhoršuje. Výnimkou sú metódy SP-TYenMD1 a SP-TYenMD2, ktoré pracujú pomerne stabilne aj pri väčších úlohách.

Tabuľka 4.11: Výsledky experimentov s pri zimnom scenári

Dataset	***Najl. výsledok /BB***	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	4/4	5	5	5	5	5	5
DS2	4/4	10	7	8	4	4	4
DS3	5/5	14	10	10	6	6	6
DS4	6/6	23	18	14	6	7	7
DS5	10/8	79	79	79	13	9	9
DS6	11/9	123	123	123	102	10	10

Tabuľka 4.11 obsahuje hodnoty účelovej funkcie v prípade zimného scenára. V prípade datasetov DS5 a DS6 vidíme, že pôvodný matematický model nebol schopný počas časového limitu 24 hodín nájsť optimálne riešenie a výsledky výpočtu sú zachytené v stĺpci *Najl. výsledok/BB*, kde máme najprv najlepšie nájdené riešenie pomocou modelu a za oddeľovačom spodnú hranicu riešenia získanú matematickým modelom v časovom limite. Vidíme, že metódy SP-mod a SP-Yen začínajú mať značné problémy nájsť dobré riešenia v prípade zimného scenára v závislosti od zväčšujúceho sa rozmeru úlohy. Na druhú stranu metódy SP-TYenM a obe metódy SP-YenMD ukazujú dobré výsledky pri datasetoch DS1-DS4. Výnimku tvoria datasety DS5 a DS6, kde metóda SP-TYenM dosiahla síce dobré výsledky, ale už začína zaostávať za metódami SP-YenMD1 a SP-YenMD2 a zároveň dosiahnuté výsledky sú v tomto prípade horšie ako riešenie iba pomocou pôvodného matematického modelu. Metódy SP-YenMD1 a SP-YenMD2 dosahujú pri všetkých datasetoch výborné výsledky a v prípade datasetov DS5 a DS6 dokonca dosiahli lepší výsledok ako pôvodný matematický model.

Tabuľka 4.12. Čas výpočtu experimentov pri jarnom scenári

Dataset	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	16,9	25,04	16,3	15,3	0,87	0,98
DS2	2432	433	756	396	9,1	9,5
DS3	2601	688	551	261,7	6,5	8,2
DS4	6652	800	1133	375,9	5,5	5,25
DS5	24510	4997	1420	541,1	7,5	7,2
DS6	49083	4691	7537	2737	21,8	21,7

Tabuľky 4.12, 4.13 a 4.14 obsahujú časy výpočtu pre jednotlivé experimenty. V tabuľke 4.12 sa nachádzajú časy výpočtu pre jarný scenár. Môžeme vidieť, že veľkosť datasetu značne ovplyvňuje čas výpočtu. Navrhnuté metódy tento čas podstatne znižujú v porovnaní s metódou, kde sa každý pod-problém rieši matematickým modelom (SP-mod). Keď porovnáme jednotlivé metódy, tak sa metóda SP-TYenMD1 ukazuje ako najrýchlejšia vo väčšine prípadov, zvlášť s rastúcim rozmerom úlohy. Čo je zaujímavé je, že metóda SP-YenM je najpomalšia zo všetkých navrhovaných metód, čo je spôsobené generovaním viacerých turnusov v jednom pod-probléme v porovnaní s metódou SP-Yen, ktorá generuje vždy iba jeden turnus z jedného pod-problému. Metódy pracujúce iba s redukovaným digrafom spojov (SP-TYenM, SP-TYenMD1 a SP-TYenMD2) sa ukazujú ako omnoho rýchlejšie v porovnaní s ostatnými dvomi navrhnutými metódami.

Tabuľka 4.13. Čas výpočtu experimentov pri letnom scenári

Dataset	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	171,3	76,12	71	19,7	3,6	1,33
DS2	34597	5147	7848	25,3	41,9	38,3
DS3	8780	4282	4096	276,7	12,4	11,1
DS4	18712	8452	3040	9401	7,8	6,8
DS5	86605	21016	25920	2538	18,1	15,8
DS6	91368	69527	107058	16590	60,4	62

V tabuľke 4.13 nájdeme dĺžku výpočtu pre prípad letného scenára. Podobne ako pri jarnom scenári si môžeme všimnúť, že výpočtový čas najviac ovplyvňuje veľkosť úlohy. Zároveň je zrejmé, že každá navrhnutá metóda zlepšila čas potrebný na výpočet. Jedinou výnimkou je metóda SP-YenM, ktorá v porovnaní s metódou SP-Yen zhoršila výpočtový čas, keďže generuje niekoľko turnusov počas jedného pod-problému. Avšak, z pohľadu účelovej funkcie dosahuje lepšie výsledky. Opätovne sa ukazuje že metódy SP-TYenMD1 a SP-TYenMD2 sú najlepšie aj z pohľadu výpočtového času. Pri porovnaní výpočtového času jarného a letného scenára si môžeme všimnúť nárast výpočtového času v prípade letného scenára, čo je spôsobené zvýšenou spotrebou elektrickej energie. Môžeme teda konštatovať, že ďalším faktorom, ktorý vplýva na rýchlosť výpočtu je aj spotreba elektrickej energie.

Tabuľka 4.14 obsahuje výsledky experimentov z pohľadu výpočtového času pre zimný scenár. Výsledky sú podobné jarnému a letnému scenáru a môžeme vidieť, že metódy SP-TYenMD1 a SP-TYenMD2 značne prekonávajú ostatné metódy z pohľadu rýchlosti

riešenia. Takisto vidíme, že ostatné metódy začínajú mať veľké problémy s väčším rozsahom úloh v prípade zimného scenára. Keď porovnáme výpočtové časy na väčších datasetoch medzi jednotlivými scenármi, tak sa ukazuje, že zvolený scenár podstatne ovplyvňuje čas výpočtu. Toto je spôsobené zvýšenou spotrebou energie a znížením maximálnej kapacity batérie, čo komplikuje riešenie modelu ako aj vytvorenie prípustného návrhu turnusov pomocou navrhnutých heuristík.

Tabuľka 4.14. Čas výpočtu experimentov pri zimnom scenári

Dataset	***SP-mod***	***SP-Yen***	***SP-YenM***	***SP-TYenM***	***SP-TYenMD1***	***SP-TYenMD2***
DS1	315	422	286	274	3,3	4,6
DS2	49564	33941	16696	24,6	1,2	1,2
DS3	40531	71526	9880	3095	28,6	23,7
DS4	97162	96286	26533	2237	29,7	38
DS5	92507	92848	92841	40765	116	120
DS6	65441	66923	66938	87626	151,6	146

Tabuľky 4.15, 4.16 a 4.17 obsahujú detailný pohľad na riešenie pod-problémov v prípade jarného, letného aj zimného scenára. Pre každý dataset sú uvedené celkové časy strávené počítaním ako aj priemerný čas výpočtu jedného pod-problému. Poslednou časťou je počet pod-problémov počítaných pomocou matematického modelu k celkovému počtu pod-problémov rátaných počas experimentu.

Tabuľka 4.15. Detaily riešenia pod-problému pri jarnom scenári – celkový čas (*SP čas*), priemerný čas riešenia pod-problému (*SP priemer*) a počet pod-problémov riešených matematickým modelom(*SP-mod/Num.*)

Dataset		***DS1***	***DS2***	***DS3***	***DS4***	***DS5***	***DS6***
SP-mod	SP čas (s)	13,57	2251	2429	6214,5	23285	47047
	SP priemer (s)	1,357	16,55	21,49	42,8	125,8	236,4
	SP-mod/Num.	10/10	136/136	113/113	145/145	185/185	199/199
SP-Yen	SP čas (s)	0,39	150,9	381,5	373,9	3684,7	2049
	SP priemer (s)	0,005	0,79	2,13	2,5	16,9	7,97
	SP-mod/Num.	0/73	8/189	16/179	8/149	18/218	14/256
SP-YenM	SP čas (s)	0,609	401	284	760	839	4760
	SP priemer (s)	0,013	2,28	2,35	7,24	9,75	27,36
	SP-mod/Num.	0/44	18/176	11/121	14/105	9/86	29/174
SP-TYenM	SP čas (s)	0,156	0,55	0,075	0,156	0,343	1,064
	SP priemer (s)	0,003	0,002	0,0	0,001	0,004	0,006
	SP-mod/Num.	0/44	0/183	0/117	0/103	0/81	0/174
SP-TYenMD1	SP čas (s)	0,22	0,33	0,32	0,28	0,3	0,94
	SP priemer (s)	0,005	0,001	0,002	0,002	0,003	0,006
	SP-mod/Num.	0/44	0/198	0/136	0/101	0/90	0/155
SP-TYenMD2	SP čas (s)	0,18	0,36	0,75	0,22	0,34	0,86
	SP priemer (s)	0,004	0,001	0,005	0,002	0,003	0,005
	SP-mod/Num.	0/44	0/198	0/136	0/101	0/90	0/155

V tabuľkách môžeme vidieť, že celkový aj priemerný čas výpočtu pod-problému je hlavne ovplyvnený počtom, koľkokrát sa pod-problém rátal pomocou matematického modelu. Preto sa navrhnuté metódy snažili čo najviac znížiť počet pod-problémov rátaných pomocou modelu na minimum, čo znamená, že sa zníži aj celkový a priemerný čas výpočtu pod-problému ako aj celkovej úlohy. Tým sa zároveň zvýši aplikovateľnosť tejto metódy na úlohy s väčšou veľkosťou. Z tabuliek môžeme vidieť, že metóda SP-Yen je v niektorých prípadoch lepšia ako metóda SP-YenM. Avšak metóda SP-YenM dosahuje vo väčšine prípadov lepší výsledok z pohľadu účelovej funkcie. Metódy SP-TYenM, SP-TYenMD1 a SP-TYenMD2, ktoré pracujú s redukovaným digrafom ukazujú značné zlepšenie času výpočtu pod-problému v porovnaní s ostatnými metódami. To je spôsobené hlavne tým, že počet riešení pod-problému matematickým modelom bol zredukovaný na minimum v prípade jarného a letného scenára.

Tabuľka 4.16. Detaily riešenia pod-problému pri letnom scenári – celkový čas (*SP čas*), priemerný čas riešenia pod-problému (*SP priemer*) a počet pod-problémov riešených matematickým modelom(*SP-mod/Num.*)

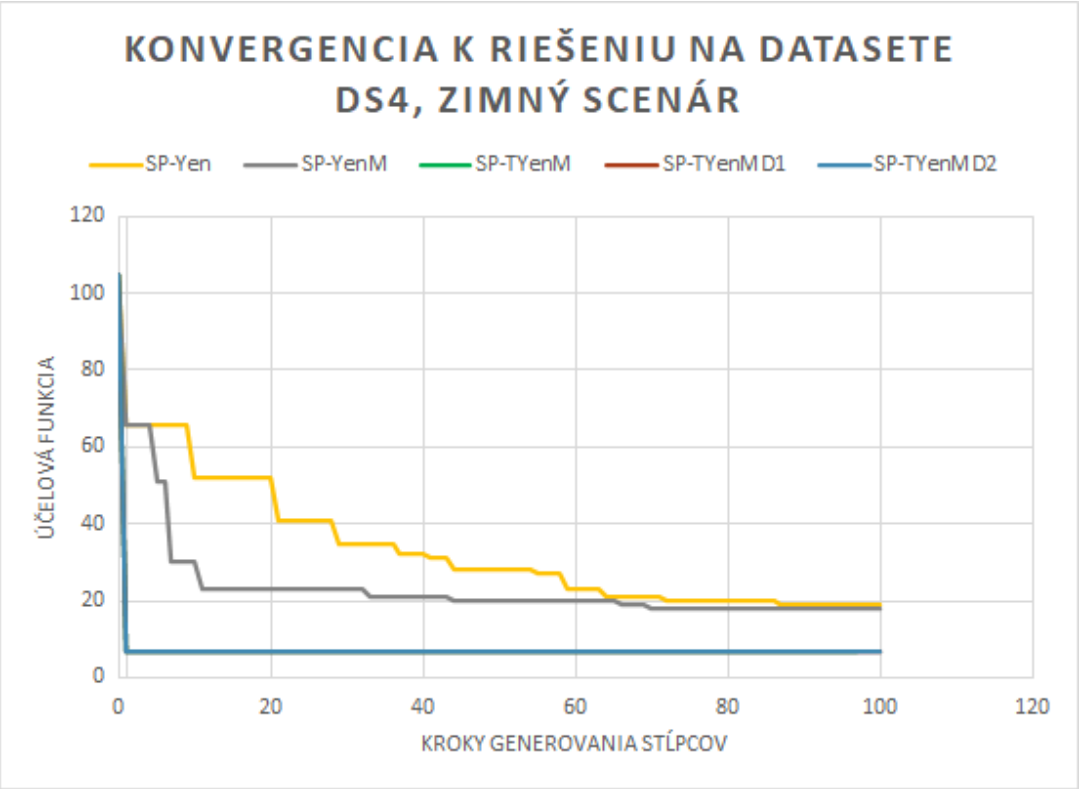
Dataset		***DS1***	***DS2***	***DS3***	***DS4***	***DS5***	***DS6***
SP-mod	SP čas (s)	148,7	24299	8619	18353	85806	91013
	SP priemer (s)	2,9	230,8	74,3	136,9	568,2	8273
	SP-mod/Num.	51/51	149/149	116/116	134/134	151/151	11/11
SP-Yen	SP čas (s)	56,7	4864	4007	7960	19801	67036
	SP priemer (s)	0,93	25,87	24,3	48,5	98	288,9
	SP-mod/Num.	2/61	59/188	40/165	42/164	44/202	84/232
SP-YenM	SP čas (s)	55,98	7458	3854	2658	25160	104109
	SP priemer (s)	1,27	40,5	32,1	25,8	135	601
	SP-mod/Num.	2/44	56/184	34/120	32/103	40/107	102/173
SP-TYenM	SP čas (s)	0,25	0,203	2,24	9010	1807	14459
	SP priemer (s)	0,004	0,008	0,018	85,8	19,02	120,5
	SP-mod/Num.	0/51	0/24	0/123	5/105	1/95	8/120
SP-TYenMD1	SP čas (s)	0,32	30,2	6	1,45	7,75	34,7
	SP priemer (s)	0,006	0,156	0,05	0,014	0,1	0,27
	SP-mod/Num.	0/54	0/193	0/128	0/101	0/74	0/128
SP-TYenMD2	SP čas (s)	0,33	26,5	5,5	1,36	6,7	37,2
	SP priemer (s)	0,006	0,14	0,04	0,013	0,09	0,29
	SP-mod/Num.	0/54	0/193	0/128	0/101	0/74	0/128

V zimnom scenári metóda SP-TYenM začína mať problémy pri väčších datasetoch. Je to spôsobené tým, že vygenerované turnusy nie je možné upraviť na prípustný tvar iba pridávaním nabíjacích udalostí, keďže turnusy sa predlžujú a čas medzi spojmi je vo väčšine prípadov veľmi krátky, a teda nie je možné dostatočne dobiť batériu, aby bolo možné pokračovať obsluhou ďalšieho spoja. Tento problém riešia metódy SP-TYenMD1 a SP-TYenMD2, ktoré pridávajú možnosť odstrániť spoje a tým vytvárajú čas na dobitie batérie. Z tohto dôvodu sa ukazujú metódy SP-TYenMD1 a SP-TYenMD2 ako najlepšie z pohľadu výpočtového času aj výsledkov. Môžeme vidieť v tabuľkách, že pri použití týchto metód sa pod-problém ani raz nerátal pomocou matematického modelu. Aj toto dopomohlo týmto metódam prekonať ostatné navrhované metódy. Zároveň dosiahli výborné výsledky v prípade zimného scenára a datasetov DS5 a DS6, kde ostatné metódy už začali zlyhávať.

Tabuľka 4.17. Detaily riešenia pod-problému pri zimnom scenári – celkový čas (*SP čas*), priemerný čas riešenia pod-problému (*SP priemer*) a počet pod-problémov riešených matematickým modelom(SP-mod/Num.)

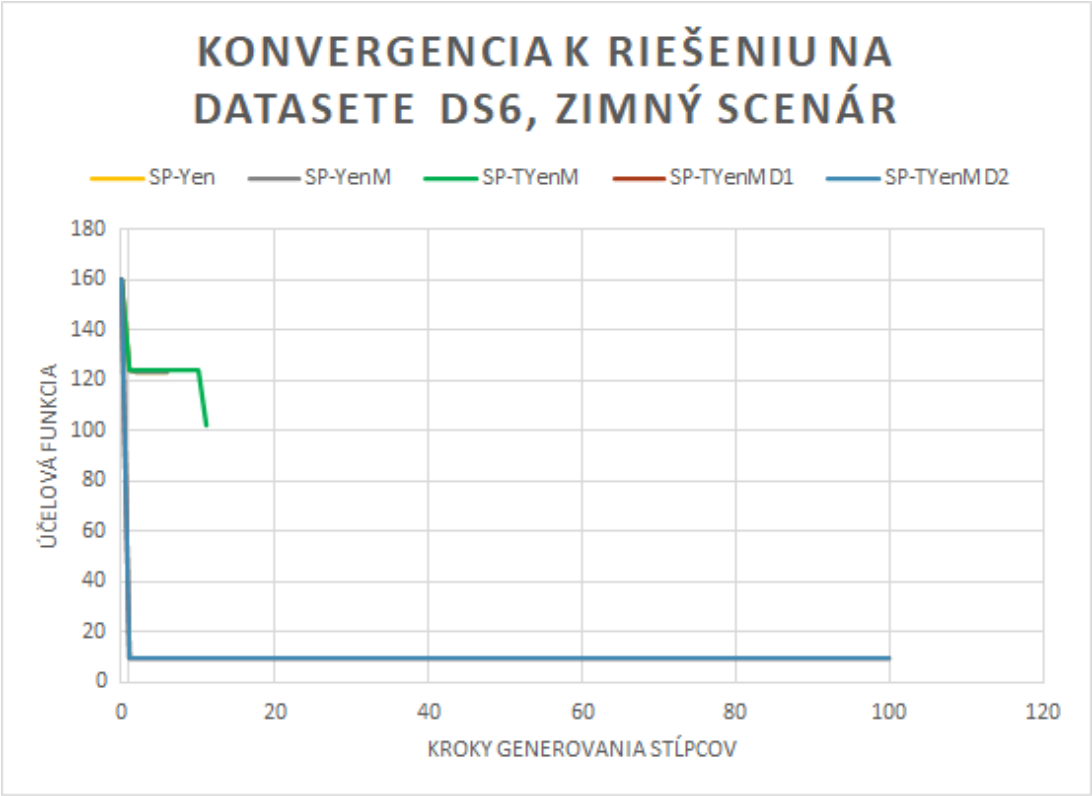
Dataset		***DS1***	***DS2***	***DS3***	***DS4***	***DS5***	***DS6***
SP-mod	SP čas (s)	196,5	49254	40262	96813	92388	65302
	SP priemer (s)	4,8	313,7	314,5	939,9	7699	9328
	SP-mod/Num.	62/62	157/157	128/128	103/103	12/12	7/7
SP-Yen	SP čas (s)	388	33619	71247	95983	92749	66640
	SP priemer (s)	5,48	205	456,7	897	7729	9520
	SP-mod/Num.	3/71	56/164	51/156	53/107	12/12	7/7
SP-YenM	SP čas (s)	265	16430	9710	26076	92781	66695
	SP priemer (s)	5,2	111,7	103,3	208,6	7731	9527
	SP-mod/Num.	4/51	66/147	38/94	62/125	12/12	7/7
SP-TYenM	SP čas (s)	255	0,61	3612	1870	39860	87398
	SP priemer (s)	5,1	0,023	28,2	19,2	383,3	7945
	SP-mod/Num.	2/50	0/26	2/128	2/97	22/104	8/11
SP-TYenMD1	SP čas (s)	2,3	0,8	17,3	24,2	101,3	120
	SP priemer (s)	0,04	0,04	0,13	0,24	0,66	0,83
	SP-mod/Num.	0/57	0/20	0/136	0/103	0/152	0/146
SP-TYenMD2	SP čas (s)	2,9	0,75	17,8	30,5	100	113
	SP priemer (s)	0,03	0,03	0,13	0,27	0,57	0,72
	SP-mod/Num.	0/92	0/20	0/136	0/111	0/151	0/156

Aby sme trochu lepšie preskúmali výkonnosť všetkých navrhnutých metód, použili sme výsledky z datasetov DS4 a DS6 pri zimnom scenári a vytvorili dva grafy konvergencie riešenia. Grafy sú zobrazené na obr. 4.8 a 4.9. V prípade datasetu DS4 (obr. 4.8) vidíme, že prvé dve navrhnuté metódy konvergujú k riešeniu postupne s rastúcim počtom riešených pod-problémov. Na grafe takisto vidíme, že metóda SP-YenM konverguje rýchlejšie ako metóda SP-Yen. Preto môžeme tvrdiť, že metóda SP-YenM je lepšia ako metóda SP-Yen, aj keď jej celkový čas behu je vo väčšine prípadov dlhší. Metódy SP-TYenM, SP-TYenMD1 a SP-TYenMD2 ukazujú rovnakú rýchlosť konvergencie v prípade datasetu DS4 a v porovnaní s prvými dvomi metódami konvergujú veľmi rýchlo už po prvých riešeniach pod-problému.



Obrázok 4.8. Konvergencia navrhnutých metód v prípade datasetu DS4 pri zimnom scenári

Na obrázku 4.9 môžeme vidieť konvergenciu všetkých metód v prípade datasetu DS6. Vidíme, že metódy SP-Yen a SP-YenM skončia po niekoľkých riešeniach pod-problému a nie sú schopné získať dobré riešenie. Toto potvrdzuje, že tieto prvé dve metódy nie sú vhodné na použitie pri rozsiahlejších úlohách, keďže sa pod-problém bude skoro vždy riešiť pomocou matematického modelu, čo sa ukázalo aj pri iných datasetoch (Tab. 4.15, 4.16 a 4.17). Ďalšie tri metódy taktiež ukazujú svoju kvalitu vcelku dobre. Metóda SP-TYenM ukázala podobnú výkonnosť ako obe metódy s odstraňovacou heuristikou na menších úlohách, ale v tomto prípade je jasne vidno na grafe, že metódy SP-TYenMD1 a SP-TYenMD2 konvergujú omnoho rýchlejšie ako metóda SP-TYenM. Navyše, metódy SP-TYenMD1 a SP-TYenMD2 ukazujú podobný výkon ako v prípade datasetu DS4 a síce, že skonvergovali k dobrému riešeniu v priebehu niekoľkých behov riešenia pod-problému. Môžeme teda tvrdiť, že tieto dve metódy sú vhodné aj na riešenie rozsiahlych úloh, pričom ich konvergencia je rýchla a získajú dobré riešenie po niekoľkých behoch riešenia pomocou metódy generovania stĺpcov.



Obrázok 4.9. Konvergencia navrhnutých metód v prípade datasetu DS6 pri zimnom scenári

Kvôli lepšiemu otestovaniu výkonnosti metód SP-TYenMD1 a SP-TYenMD2 sme sa rozhodli vykonať experimenty aj na väčších datasetoch DS7-DS10 pre prípad všetkých troch scenárov. Tieto experimenty takisto lepšie ukazujú rozdiely medzi dvomi odlišnými stratégiami odstraňovania spojov z turnusu.

Tabuľka 4.18. Výsledky, čas výpočtu a detaily riešenia pod-problému pri použití metód SP-TYenMD1 a SP-TYenMD2 pri jarnom scenári

Dataset		***DS7***	***DS8***	***DS9***	***DS10***
Najl. výsledok/BB		13/13	33/26	-/28	-/49
SP-TYenMD1	Riešenie	14	27	30	54
	Celkový čas (s)	51,9	73	325,9	1778
	SP čas (s)	9,6	52,1	251,6	1055
	SP priemer (s)	0,065	1,1	3,7	11,4
	SP-mod/Num,	0/147	0/46	0/67	0/92
SP-TYenMD2	Riešenie	14	27	29	54
	Celkový čas (s)	56,7	73,2	291,4	1724
	SP čas (s)	13	52	249,7	1018
	SP priemer (s)	0,08	1,13	4,5	11
	SP-mod/Num,	0/147	0/46	0/56	0/92

V tabuľke 4.18 sú výsledky experimentov s jarným scenárom. Môžeme vidieť, že obe skúmané metódy dosahujú v tomto prípade veľmi dobré výsledky pre všetky datasety

v porovnaní s dolnou hranicou riešenia získanou pôvodným modelom. V prípade datasetu DS9 metóda SP-TYenMD2 dosiahla lepšie riešenie ako metóda SP-TYenMD1. V ostatných prípadoch dosiahli obe metódy zhodné riešenia a aj čas výpočtu bol porovnateľný.

Tabuľka 4.19. Výsledky, čas výpočtu a detaily riešenia pod-problému pri použití metód SP-TYenMD1 a SP-TYenMD2 pri letnom scenári

Dataset		***DS7***	***DS8***	***DS9***	***DS10***
Najl. výsledok/BB		27/13	-/26	-/28	-/49
SP-TYenMD1	Riešenie	15	27	34	126
	Celkový čas (s)	182,1	211,6	11920	21802
	SP čas (s)	139	187,8	1049,5	10436
	SP priemer (s)	1,13	3,8	14,3	100,3
	SP-mod/Num.	0/124	0/49	0/73	0/104
SP-TYenMD2	Riešenie	15	28	33	55
	Celkový čas (s)	212	246	2061	18315
	SP čas (s)	167	223,5	1042	13177
	SP priemer (s)	1,5	5,3	13,9	134,5
	SP-mod/Num.	0/110	0/42	0/75	0/98

Výsledky pre letný scenár sa nachádzajú v tabuľke 4.19. Podobne ako pri jarnom scenári môžeme vidieť, že kvalita dosiahnutých výsledkov je vysoká. V prípade datasetu DS8 dosiahla lepší výsledok metóda SP-TYenMD1, avšak v prípade datasetov DS9 a DS10 to bola metóda SP-TYenMD2. Z pohľadu času výpočtu sú metódy ekvivalentné, okrem prípadu datasetov DS9 a DS10, kde cyklická metóda SP-TYenMD2 dosiahla značne lepšie časy.

Tabuľka 4.20. Výsledky, čas výpočtu a detaily riešenia pod-problému pri použití metód SP-TYenMD1 a SP-TYenMD2 pri zimnom scenári

Dataset		***DS7***	***DS8***	***DS9***	***DS10***
Najl. výsledok/BB		-/13	-/26	-/28	-/49
SP-TYenMD1	Riešenie	15	27	31	56
	Celkový čas (s)	399,6	361	2558	31409
	SP čas (s)	353	341	2178	19882
	SP priemer (s)	3,3	8,1	26,5	179
	SP-mod/Num.	0/108	0/42	0/82	0/111
SP-TYenMD2	Riešenie	14	27	31	123
	Celkový čas (s)	358	394	2674	45105
	SP čas (s)	320	374	2511	33701
	SP priemer (s)	2,5	8,9	33	333
	SP-mod/Num.	0/127	0/42	0/76	0/101

Tabuľka 4.20 obsahuje výsledky a výkon oboch metód pre zimný scenár. V tomto prípade sa ukázali viaceré rozdiely medzi dvomi skúmanými metódami. V prípade datasetu

DS7 dosiahla metóda SP-TYenMD2 lepšie riešenie, avšak v prípade datasetu DS10 bolo získané riešenie veľmi ďaleko od spodnej hranice riešenia. Metóda SP-TYenMD1 naproti tomu v prípade datasetu DS10 dosiahla kvalitné riešenie, pričom aj čas výpočtu bol omnoho lepší, čo je presný opak situácie pri letnom scenári. Preto môžeme povedať, že obe metódy SP-TYenMD1 aj SP-TYenMD2 sú vhodné na aplikáciu do prostredia s väčšími obmedzeniami. Zároveň by bolo nutné preskúmať vplyv parametra *dMax* na získaný výsledok, ako aj výpočtový čas.

Ak sa pozrieme na výkon metód SP-TYenMD1 a SP-TYenMD2 v prípade všetkých datasetov DS1-DS10, môžeme pozorovať zaujímavý fakt, keď sa na riešenie pod-problému nepoužije matematický model ani raz a zároveň pre väčšie datasety sa znížil celkový počet riešení pod-problému. Toto môže byť spôsobené efektivitou navrhnutých metód SP-TYenMD1 a SP-TYenMD2 pri hľadaní dobrých riešení pod-problému.

4.4.4 Zhodnotenie výsledkov

V tejto časti práce sme navrhli využitie metódy generovania stĺpcov na riešenie úlohy návrhu turnusov vo verejnej doprave. Prvou metódou bola metóda SP-mod, ktorá bola základnou implementáciou metódy generovania stĺpcov pomocou matematických modelov a riešenia pod-problému pomocou štandardného IP solvera. Táto metóda bola použitá za účelom porovnania ďalších navrhnutých metód s metódou využívajúcou iba matematické modely. Z pohľadu riešení, táto metóda nedosiahla dobré výsledky, špeciálne so zväčšujúcou sa veľkosťou úlohy. Takisto čas výpočtu bol vysoký, hlavne z dôvodu využitia komplexného matematického modelu na riešenie pod-problému, ktorý je takmer identický s matematickým modelom navrhnutým na riešenie celkového návrhu turnusov elektrických autobusov. V prípade zimného scenára táto metóda nebola schopná dosiahnuť dobré výsledky takmer pri žiadnom experimente a čas výpočtu sa drasticky zvýšil v porovnaní s jarným scenárom.

Prvou navrhnutou metódou bola metóda SP-Yen, ktorá využíva na riešenie pod-problému algoritmus na hľadanie *k*-najkratších ciest. V tabuľkách 4.8-4.16 môžeme vidieť zlepšenie v porovnaní so základnou metódou SP-mod. Výsledky boli o niečo lepšie a čas výpočtu sa výrazne zredukoval vo väčšine prípadov. Avšak pri datasetoch DS3-DS6 pri zimnom scenári sa začali ukazovať problémy s aplikáciu tejto metódy, pretože algoritmus na hľadanie *k*-najkratších ciest nebol schopný nájsť prípustné riešenie pod-problému a musel byť aplikovaný matematický model. Toto spôsobilo nárast výpočtového času a v prípade

datasetov DS5 a DS6 dokonca skonvergovanie na metódu SP-mod, čím sme dosiahli rovnaké riešenie ako pri použití samotnej metódy SP-mod.

Ďalšia metóda SP-YenM je prvá, ktorá využíva algoritmus generovania viacerých turnusov z jedného pod-problému. Táto metóda ukázala nádejné výsledky v porovnaní s metódou SP-Yen, zvlášť v prípade datasetov DS5 a DS6 pri jarnom a letnom scenári, čo môžeme vidieť v tabuľkách 4.9 a 4.10. Pri zimnom scenári (tabuľka 4.11) boli výsledky taktiež lepšie pre väčšinu datasetov. Metóda SP-YenM síce nezlepšila výpočtový čas v porovnaní s metódou SP-Yen, avšak toto zhoršenie je odôvodnené zlepšenými výsledkami. V prípade zimného scenára a datasetov DS5 a DS6 sa ale ukázalo, že aj metóda SP-YenM má problémy s väčšími datasetmi, zvlášť v prípade zhoršených podmienok v podobe zvýšenej spotreby energie a zníženej kapacity batérie.

Metóda SP-TYenM využíva redukovaný digraf spolu s heuristikou na pridávanie nabíjacích udalostí do turnusu. Použitie tejto metódy poskytlo značné zníženie času výpočtu a zároveň výsledky, ktoré metóda dosiahla pri každom datasete boli lepšie alebo rovnaké ako pri použití predchádzajúcich metód. Počet pod-problémov riešených matematickým modelom sa tiež znížil, viditeľné v tabuľkách 4.15 - 4.17, čo podstatne znížilo celkový aj priemerný čas výpočtu pod-problému. Táto metóda fungovala veľmi dobre aj v zimnom scenári, čo môžeme vidieť na čase výpočtu ako aj na dosiahnutých výsledkoch. V prípade datasetov DS5 a DS6 sa síce nepodarilo dosiahnuť veľmi dobré výsledky, ale oproti predchádzajúcim metódam sa riešenia zlepšili.

Posledné dve metódy SP-TYenMD1 a SP-TYenMD2 pracujú rovnako ako metóda SP-TYenM, ale je pridaná heuristika na odstránenie spojov z turnusu. V metóde SP-TYenMD1 je táto heuristika aplikovaná sekvenčne a v metóde SP-TYenMD2 je aplikovaná cyklicky. Obe metódy ukázali dominanciu na poli riešení ako aj na poli času výpočtu, v porovnaní s ostatnými navrhnutými metódami. Táto skutočnosť je spôsobená využitím heuristiky na odstránenie spojov z turnusu, keďže táto heuristika redukuje počet aplikácií matematického modelu na riešenie pod-problému na nulu vo všetkých experimentoch (Tab. 4.15 - 4.17). Preto môžeme tvrdiť, že tieto dve metódy sú najlepšie z navrhovaných metód na riešenie pod-problému. Navyše, tieto metódy majú potenciál byť použité aj na rozsiahle úlohy aj v prostredí s nízkou maximálnou kapacitou batérie a vysokou spotrebou.

Keďže posledné dve metódy boli veľmi podobné z pohľadu riešení aj času výpočtu, vykonali sa ďalšie experimenty s týmito metódami na datasetoch väčšieho rozsahu DS7-

DS10. Po vykonaní experimentov môžeme povedať, že obe metódy SP-TYenMD1 a SP-TYenMD2 sú porovnateľné. V prípade najväčšieho datasetu DS10 sa pri letnom scenári ukázala ako lepšia metóda s cyklickým odstraňovaním, avšak pri zimnom scenári táto metóda zlyhala. Naproti tomu sekvenčný prístup v aplikácii heuristiky odstraňovania spojov z turnusu metódy SP-TYenMD1 sa ukázal ako lepší v prípade zimného scenára, ale zlyhal pri letnom scenári. Obe metódy však ukázali, že nemajú problém vysporiadať sa s obomi typmi spojov, a síce cyklickými spojmi (trolejbusové linky) aj jednosmernými spojmi (autobusové linky).

Inovácie v tejto časti, ako aj zistenia založené na experimentoch, by sme mohli zosumarizovať nasledovne. Navrhli sme niekoľko rôznych metód, ktorými sme zlepšili metódu generovania stĺpcov pre riešenie úlohy návrhu turnusov elektrických autobusov. Zvlášť riešenie pod-problému, kvôli jeho zložitosti a vysokému času výpočtu pre riešenie pod-problému. Predstavili sme metódy využívajúce algoritmus na hľadanie *k*-najkratších ciest pre nájdenie najkratšej prípustnej cesty so zápornou redukovanou cenou v rámci pod-problému, ako aj metódy, ktoré vylepšili tento prístup. V prvých metódach, sa hľadanie *k*-najkratších ciest riešilo na celom digrafe spojení medzi spojmi a nabíjacími udalosťami. V ďalších nami navrhnutých metódach SP-TYenM a SP-TYenMD sa využíva redukcia celého digrafu spojení na digraf spojení len medzi spojmi. Zároveň sme predstavili heuristiku na pridávanie nabíjacích udalostí do turnusu za účelom úpravy turnusu na prípustný turnus. Posledné metódy SP-TYenMD1 a SP-TYenMD2 na tvorbu prípustného turnusu heuristiku na pridávanie nabíjacích udalostí aj heuristiku na odstraňovanie spojov z turnusu.

Okrem toho sme uviedli aj porovnanie jednotlivých navrhnutých metód vzhľadom na základnú verziu metódy generovania stĺpcov riešenú iba pomocou matematických modelov. Experimenty ukázali, že metódy SP-TYenMD1 a SP-TYenMD2 sú dominantné v oblasti výsledkov aj času výpočtu v porovnaní so základnou verziou metódy ako aj ostatnými navrhnutými metódami. Avšak, aj ostatné navrhnuté metódy ukázali aplikovateľnosť na úlohy menšieho rozsahu. Takisto, skúsenosti získané počas experimentovania s prvými metódami kulminovali konštrukciou poslednej metódy SP-YenMD. Aj keď posledné navrhnuté metódy SP-TYenMD1 a SP-TYenMD2 vo väčšine prípadov nedosiahli optimálne výsledky, ich nízky čas výpočtu aj pri väčších úlohách ukazuje ich aplikovateľnosť na úlohy reálneho rozsahu. Takisto aj rozdiel medzi získaným riešením a dolnou hranicou riešenia úlohy nebol veľký, zvlášť v prípade experimentov

s väčšími úlohami. Porovnanie oboch typov metódy SP-YenMD bolo vykonané na experimentoch s úlohami väčšieho rozsahu, kde sa však neukázala dominancia jedného z dvoch prístupov k odstraňovaniu spojov Dokladom je dataset DS10, kde pri letnom scenári bola lepšia cyklická metóda, ale pri zimnom scenári bol úspešnejší sekvenčný typ heuristiky na odstraňovanie spojov. Preto nevieme jednoznačne odporúčiť, ktorú z metód SP-TYenMD1 a SP-TYenMD2 je vhodnejšie použiť na riešenie návrhu turnusov elektrických autobusov na úlohách z praxe.

V budúcnosti by mal byť výskum tejto metódy orientovaný na zlepšenie metód na riešenie pod-problému ako aj testovanie algoritmu na iných datasetoch. Časti, ktoré by mali byť skúmané by mohli zahŕňať testovanie rôznych stratégií pre pridávanie nabíjacích udalostí do turnusu v heuristike na pridávanie nabíjacích udalostí a stratégie odstraňovania spojov v heuristike na odstraňovania spojov. Takisto by sa mala preskúmať možnosť inicializácie metódy inými riešeniami ako len kyvadlovými turnusmi.

4.5 Riešenie pomocou algoritmu Grouping Genetic Algorithm (GGA)

Ako sme ukázali pri návrhu nového matematického modelu a jeho následnom otestovaní s využitím štandardného IP solvera, ktorý využíval metódu vetiev a hraníc, nie je možné riešenie pomocou tejto exaktnej metódy v prípade rozsiahlejších úloh. Jednou z možností ako riešiť komplexné úlohy v rozumnom čase je využitie heuristických, respektíve metaheuristických metód. Ako sme spomínali vyššie, tieto metódy sú síce veľmi rýchle, ale nezaručujú získanie optimálneho riešenia. Avšak pri dobrom návrhu heuristickej metódy, sú schopné heuristické metódy poskytovať veľmi dobré riešenia a zároveň v krátkom čase.

Jednou z najpoužívanejších metaheuristických metód je genetický algoritmus. Patrí medzi algoritmy, ktoré pracujú s populáciou riešení (chromozómy). Táto populácia sa v priebehu algoritmu mení, pričom sa zo starej populácie vytvorí nová na základe niekoľkých operácií, ktoré z riešení v pôvodnej populácii vytvoria nové riešenia. Veľkosť populácie je štandardne daná parametrom a počas behu algoritmu sa nemení.

Genetický algoritmus obsahuje dve špecifické metódy, ktoré vytvárajú nové riešenia – kríženie a mutácia. Kríženie je metóda, ktorá najprv vyberie z predchádzajúcej populácie niekoľko riešení (zvyčajne 2), ktoré sú nazývané aj ako rodičia a následne tieto vybrané riešenia nejakým spôsobom skombinuje, čím vytvára nové riešenia nazývané aj potomkami.

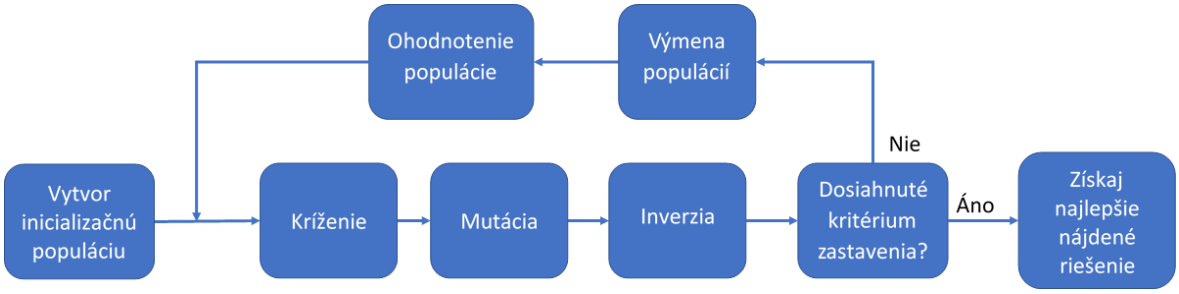
Druhou metódou je mutácia, ktorá má za úlohu čiastočne pozmeniť aktuálne riešenie, čím sa získa nové riešenie.

Úloha návrhu turnusov elektrických autobusov je jednou z typu úloh, ktoré sa snažíme rozdeliť množinu prvkov na disjunktné skupiny, pričom pre skupiny musia byť splnené špecifické podmienky. V našom prípade rozdeľujeme množinu všetkých úloh (spoje a nabíjanie) do skupín – turnusov, kde je každý turnus obslúžený práve jedným elektrickým autobusom a musí spĺňať podmienky pre nabíjanie a spotrebu energie. Keďže naša úloha má tento špecifický charakter, rozhodli sme sa teda použiť špecifický variant genetického algoritmu, ktorý je prispôsobený na riešenie úloh, kde sa množina prvkov rozdeľuje do skupín. Týmto variantom je Grouping Genetic Algorithm (GGA).

Grouping Genetic Algorithm pracuje rovnakým spôsobom ako genetický algoritmus, čo je popísané na obrázku 4.10. Na základe [41] sa Grouping Genetic Algorithm vyznačuje špecifickou reprezentáciou riešenia, keďže riešenie má reprezentovať celú množinu prvkov a zároveň aj jednotlivé skupiny – turnusy. Ďalej obsahuje tri rôzne operátory, ktoré vytvárajú novú populáciu. Sú to kríženie, mutácia a inverzia. Operátory kríženia a mutácie sú v princípe úplne rovnaké ako pri štandardnom genetickom algoritme, avšak pri implementácii týchto operátorov treba brať do úvahy špecifickú reprezentáciu riešenia. Jediným operátorom, ktorý sa v genetickom algoritme nevyskytoval je inverzia, ktorá však nemení samotné riešenie, iba jeho štruktúru.

Ako môžeme vidieť na schéme metaheuristiky na obrázku 4.10, priebeh algoritmu sa skladá z niekoľkých fáz. Prvou fázou je inicializácia počiatočnej populácie, kde sa vygenerujú počiatočné riešenia. Druhou fázou je cyklus, pri ktorom sa vygeneruje nová populácia pomocou operácie kríženia, ďalej sa operáciou mutácie nová populácia môže čiastočne zmeniť a na záver operácia inverzie môže zmeniť štruktúru riešení, pričom sa samotné riešenia nezmenia. V ďalšej fáze sa testuje kritérium zastavenia. My sme sa rozhodli pre dve kritériá zastavenia a algoritmus skončí práve vtedy, keď nastane aspoň jedno z týchto kritérií. Prvým kritériom je maximálny čas výpočtu a druhým kritériom zastavenia je počet výmen populácie bezo zmeny najlepšieho nájdeného riešenia. Ak sme ešte nedosiahli ani jedno z kritérií pre zastavenie tak dôjde k výmene populácií, čiže na miesto starej populácie sa dostane nová, ktorá bola vygenerovaná počas predchádzajúcich fáz. Následne sa nová populácia vyhodnotí a aktualizuje sa najlepšie nájdené riešenie. Samozrejme aktualizácia najlepšieho nájdeného riešenia sa odohráva vždy, keď sa vytvorí nové riešenie, čiže po vytvorení potomka v operácii kríženia, ako aj po zmene riešenia pri

mutácii. Týmto končí jeden cyklus metaheuristiky Grouping Genetic Algorithm. V prípade, že dosiahneme jedno z kritérií zastavenia, metaheuristika končí a vydá najlepšie nájdené riešenie.



Obr. 4.10. Schéma Grouping Genetic Algorithm

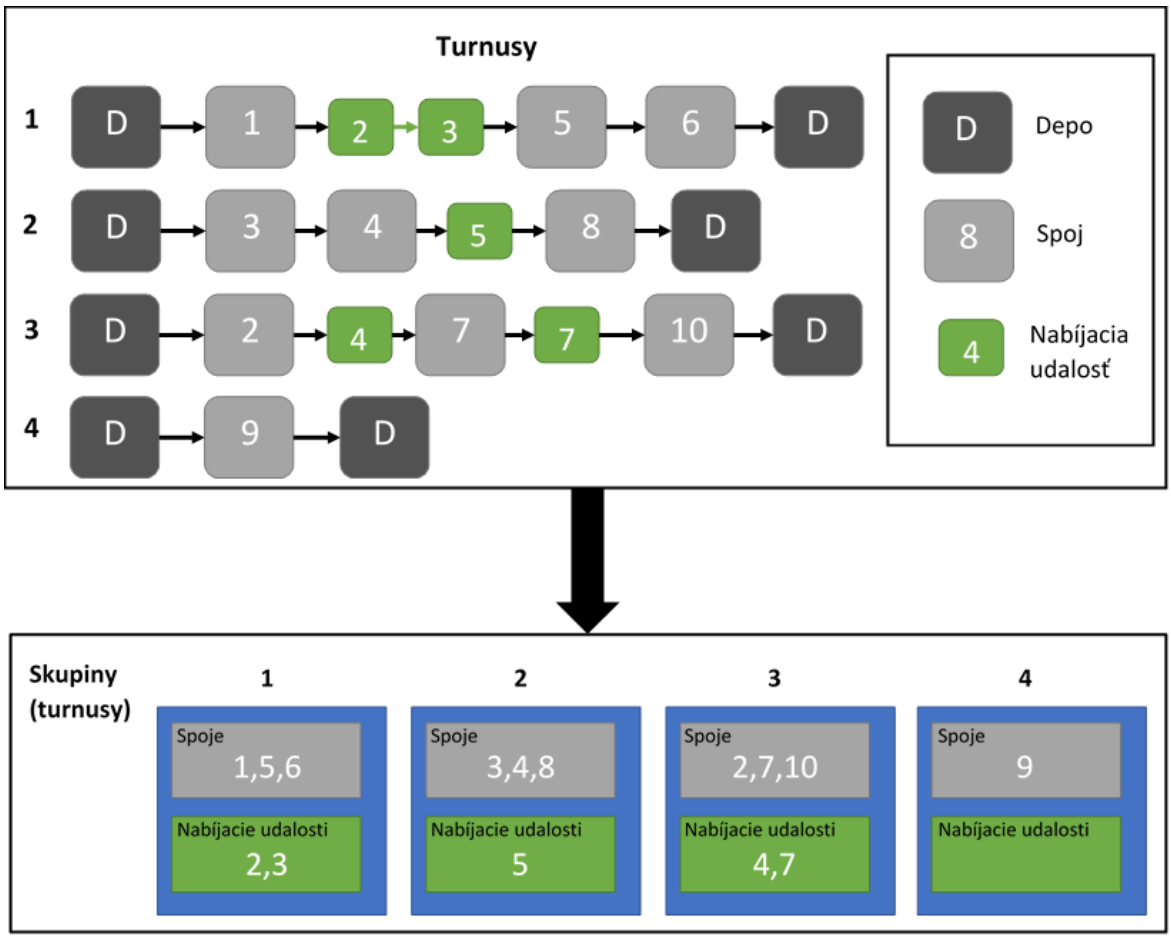
4.5.1 Reprezentácia riešenia

Reprezentácia riešenia je jednou z najdôležitejších častí algoritmu, keďže reprezentácia značne ovplyvňuje návrh operátorov a ich výkon. Preto je nutné venovať návrhu reprezentácie riešenia veľkú pozornosť. My sme sa rozhodli využiť návrhy na reprezentáciu riešenia spomínané v [41], ktoré umožňujú jednoduchšiu implementáciu operátorov špecifických pre Grouping Genetic Algorithm.

V našom prípade riešenie (chromozóm) predstavuje kompletný návrh turnusov pre elektrické autobusy. Základnú myšlienku návrhu predstavuje obrázok 4.11. Riešenie sa skladá z turnusov pre jednotlivé elektrické autobusy. Každý turnus teda predstavuje skupinu (gén), ktorú tvoria obsluhované spoje a navštívené nabíjačky. Zároveň musí platiť, že každý turnus v riešení je prípustný, čiže musí spĺňať podmienky pre nabíjanie a spotrebu energie.

V každej skupine sa nachádzajú dva zoznamy. Prvý zoznam obsahuje spoje obsluhované počas turnusu. Druhým zoznamom, je zoznam využitých nabíjacích udalostí počas daného turnusu reprezentovaného skupinou. Jednou z podmienok pri úlohe návrhu turnusov je, že každý spoj musí byť obslúžený. Táto podmienka musí byť splnená aj v našom prípade, čiže riešenie musí obsahovať všetky spoje. Zároveň musí platiť, že zoznamy spojov medzi skupinami sú disjunktné, čo predstavuje podmienku, že každý spoj je obslúžený práve jedným elektrickým autobusom. Niečo podobné platí aj pre nabíjacie udalosti. Každá nabíjacia udalosť môže byť v rámci riešenia využitá iba jedenkrát, čo zabezpečuje, že sa na nabíjačke v jednom čase nebudú nachádzať dva elektrické autobusy. Čiže aj pre zoznamy použitých nabíjacích udalostí v skupinách platí ich disjunktnosť. Rozdielom pre zoznamy nabíjacích udalostí oproti zoznamom spojov je, že nemusia byť v rámci riešenia využité

všetky nabíjacie udalosti, keďže návšteva nabíjacej udalosti nie je pre elektrický autobus povinná úloha.



Obr. 4.11. Schéma reprezentácie riešenia pre Grouping Genetic Algorithm

Pri návrhu reprezentácie riešenia je dôležité navrhnúť aj „fitness“ riešenia, čo predstavuje, akým spôsobom bude hodnotená kvalita riešenia. V našom prípade používame ako fitness hodnotu účelovej funkcie riešenia, čo predstavuje počet turnusov riešenia.

4.5.2 Inicializácia

Inicializácia počiatočnej populácie je prvou fázou Grouping Genetic Algorithm. V závislosti od kvality riešení v počiatočnej populácii môže náš algoritmus rýchlejšie skonvergovať k dobrému riešeniu. Naproti tomu viac diverzifikovaná populácia riešení poskytuje možnosť prehľadať širšiu množinu riešení a tak získať kvalitnejšie riešenie. Preto je dôležité inicializovať beh algoritmu aj dobrými riešeniami aj diverzifikovanými riešeniami. Dôležitým parametrom metaheuristiky je parameter *PopSize*, ktorý predstavuje veľkosť populácie. Pri inicializácii teda potrebujeme naplniť populáciu počtom prvkov veľkosti *PopSize*.

V našej implementácii používame na inicializáciu populácie jednoduchú vkladaciu heuristiku. Heuristika pracuje tak, že sekvenčne sa snaží pridávať spoje do jednotlivých turnusov, pričom vyhodnocuje, či je pridanie spoja do turnusu možné aj s ohľadom na podmienky nabíjania a spotreby energie. Ak je možné pridať spoj do turnusu, tak tento spoj do turnusu vloží aj s potrebnými nabíjacími udalosťami. Pri testovaní pridania spoja využíva heuristiku na pridávanie nabíjacích udalostí, ktorá bola spomínaná pri metóde generovania stĺpcov s tým, že neuvažuje s nabíjacími udalosťami použitými v ostatných turnusoch. Ak spoj nie je možné do turnusu pridať, pokračuje sa testovaním vloženia do ďalšieho turnusu. Ak nie je možné spoj vložiť do žiadneho turnusu, tak sa vytvorí nový prázdny turnus, do ktorého sa vloží daný spoj.

Predstavená vkladacia heuristika vždy vygeneruje iba jedno riešenie a je závislá od poradia vkladaných prvkov. Na vytvorenie počiatočnej populácie potrebujeme túto heuristiku volať vždy s inou permutáciou spojov, čím sa dosiahne to, že zakaždým nám heuristika vygeneruje iné riešenie. Túto metódu používame teda opakovane, až kým nevygenerujeme dostatočný počet riešení charakterizovaný parametrom *PopSize*.

4.5.3 Selekcia

Selekcia je operácia vykonávaná počas operácie kríženia za účelom výberu riešení, ktoré sa stanú rodičmi, čiže z ktorých sa bude operáciou kríženia vytvárať nová populácia. Zvyčajne sa počas selekcie snažíme vybrať pre kríženie dobré riešenia. Základnou myšlienkou selekcie je, že pri krížení dvoch dobrých riešení je vyššia šanca získať kvalitnejšie nové riešenie. Preto je dôležité, aby metóda selekcie pre kríženie bola definovaná ako pravdepodobnostná funkcia, kde lepšie riešenia budú mať vyššiu pravdepodobnosť výberu ako horšie riešenia.

V našej implementácii sme sa rozhodli použiť pre selekciu metódu Roulette-Wheel-Selection. Pri použití tejto metódy transformujeme fitness f_i riešenia i na pravdepodobnosť p_i pre každé riešenie pomocou vzorca (4.37), pričom platí že suma všetkých pravdepodobností riešení sa rovná jednej.

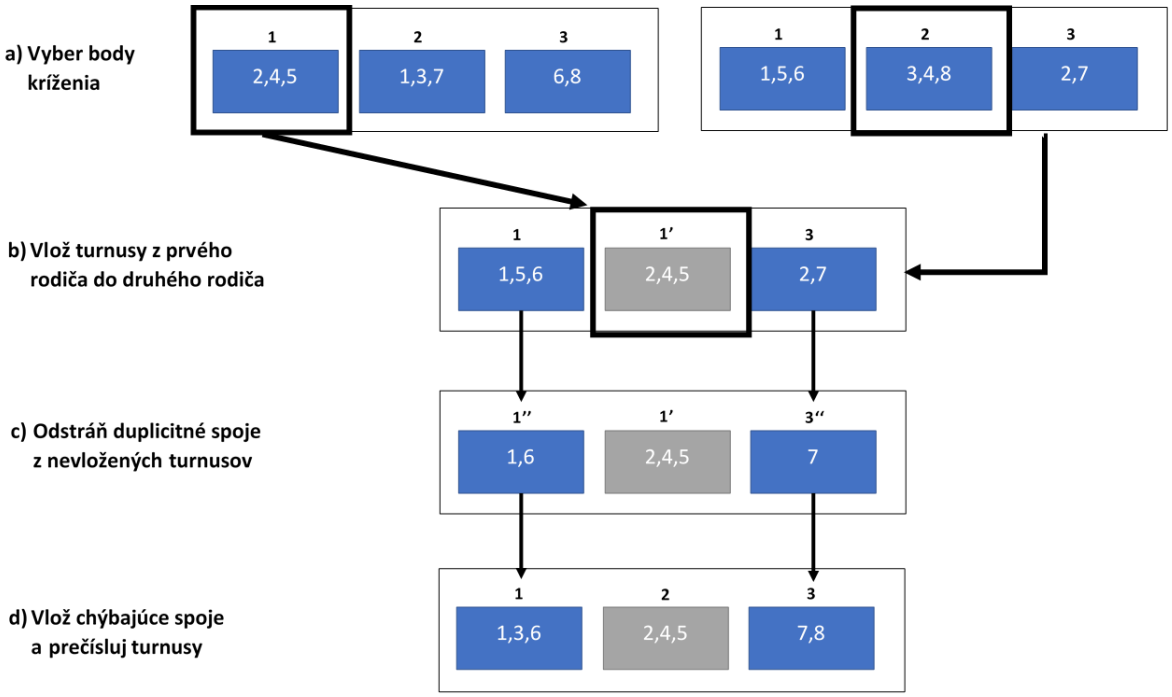
$$p_i = \frac{\frac{1}{f_i}}{\sum_{\forall j=1..popSize} \frac{1}{f_j}} \tag{4.37}$$

Ďalej sa získané pravdepodobnosti prenesú na interval predstavujúci veľkosť pravdepodobnosti v poradí v akom sú riešenia uložené v zozname riešení. Ďalej tieto

intervaly sekvenčne ukladáme do intervalu 0-1, čím pokryjeme celý interval 0-1 intervalmi pre jednotlivé riešenia. Následne sa náhodne vygeneruje číslo z intervalu 0-1 a na základe jeho hodnoty sa pozrieme do intervalu riešenia kam vygenerované číslo padlo a vyberieme toto riešenie pre kríženie.

4.5.4 Kríženie

Kríženie je operácia, pri ktorej skombinujeme niekoľko riešení za účelom získania nového riešenia. Vo väčšine prípadov sa vyberajú práve dve riešenia nazývané aj ako rodičia. Po aplikácii selekcie máme teda vybrané dve riešenia. Na kombináciu riešení existuje niekoľko metód. My sme sa rozhodli pre metódu uvedenú v [41], keďže táto bola špeciálne zadefinovaná pre úlohy so zadeľovaním prvkov do skupín.



Obr. 4.12. Schéma kríženia

Základný princíp heuristiky je možné popísať v nasledujúcich krokoch, ktoré sú zobrazené aj na obrázku 4.12. Najprv sa pre oboch rodičov (riešenia) náhodne vyberú dva body kríženia (obr. 4.12a). Následne je vybraná sekcia z prvého rodiča 1' vložená do druhého rodiča namiesto vybranej sekcie 2 druhého rodiča (obr. 4.12b), čím vzniká nový potomok. V potomkovi sa môže stať, že obsahuje duplicity spojov a niektoré spoje neobsahuje vôbec. Tento problém je riešený v ďalších krokoch. Duplicity sú z potomka odstránené, pričom sa vyhýbame vloženej sekcii. Počas tohto procesu sa môže stať, že sa niektoré gény modifikujú - z génu 1 sa teda stane gén 1'' a z génu 3 sa stane gén 3'' (obr.

4.12c). Zároveň je aplikovaná heuristika na vkladanie nabíjacích udalostí, ktorá nám zabezpečí, že aj po odobratí spojov budú všetky turnusy prípustné. V ďalšom kroku sa potomok upraví tak, aby vyhovoval podmienkam úlohy a dosiahli sme tak prípustné riešenie. Môžeme si všimnúť, že niektoré spoje nám v riešení chýbajú (obr. 4.12c), čiže ich musíme opätovne vložiť do niektorých génov (obr. 4.12d) – do génu 1″ sme vložili spoj číslo 3 a do génu 3″ spoj číslo 8. Tento krok sa realizuje vkladacou heuristikou, ktorú sme použili aj na inicializáciu počiatočnej populácie, pričom sa vyhýbame vkladaniu spojov do vloženej časti. Ak je to potrebné môžeme vytvoriť aj nové turnusy.

Pre rýchlejšie fungovanie, je možné vytvoriť z danej dvojice rodičov aj druhého potomka. Docielime to tak, že vymeníme roly oboch rodičov a opakujeme proces s vložením vybraných turnusov z druhého rodiča do prvého, následným vylúčením duplicitných spojov a vložením spojov, ktoré nie sú obslúžené.

Celý proces tvorby potomkov od výberu rodičov až po samotné kríženie a opravu potomka následne pokračuje, až kým nie je vytvorený dostatočný počet potomkov definovaný veľkosťou populácie *PopSize*. Títo novovytvorení potomkovia budú tvoriť novú populáciu.

4.5.5 Mutácia

Operátor mutácie pracuje iba s jediným riešením. Táto operácia je charakteristická tým, že aplikuje malú zmenu v riešení, čo spôsobí vytvorenie nového riešenia. Metóda má slúžiť hlavne na diverzifikáciu riešenia, ale dá sa použiť aj na jeho zlepšenie. V našej implementácii sme vytvorili výmennú heuristiku, ktorá vymieňa sekvenciu spojov medzi dvomi vybranými turnusmi.

Na začiatku heuristika vyberie náhodne dva turnusy z riešenia. Následne sa z prvého turnusu náhodne vyberie sekvencia spojov, ktoré nasledujú za sebou, a tieto budú vložené do druhého vybraného turnusu.

Aby sme mohli vybranú sekvenciu z prvého turnusu vložiť do druhého, musíme najprv nájsť miesto, kam ich vložiť. V tomto bode heuristika určí spoje v druhom turnuse, ktoré sú paralelné k tým, ktoré chceme do turnusu vložiť, čím vytvoríme druhú sekvenciu spojov. Táto sekvencia z druhého turnusu je následne odstránená a nahradená sekvenciou z prvého turnusu.

Nakoniec je odstránená sekvencia z druhého turnusu vložená do prvého turnusu na miesto odkiaľ sme vybrali sekvenciu v prvom turnuse. Samozrejme, pri každej výmene sa

kontroluje, či je táto výmena možná z pohľadu spotreby energie a nabíjania. Po výmene sa opätovne na novovzniknuté riešenia aplikuje heuristika na vkladanie nabíjacích udalostí, aby sa zabezpečilo, že vytvorené turnusy budú prípustné. Pre vkladanie nabíjacích udalostí sa používajú iba tie nabíjacie udalosti, ktoré neboli použité v žiadnom inom turnuse alebo boli použité v dvoch vybraných turnusoch, na ktorých sa vykonala výmena. Ak nie je možné vykonať výmenu sekvencií spojov, mutácia sa neuskutoční.

Operácia mutácie sa vo všeobecnosti nemusí aplikovať vždy, respektíve môže sa aplikovať viackrát. Za týmto účelom sme zvolili dva parametre pre mutáciu, ktoré tieto vlastnosti modelujú. Parameter *MutNum* určuje koľkokrát sa má skúsiť aplikovať mutácia na dané riešenie a parameter *MutProb* určuje s akou pravdepodobnosťou sa mutácia má vykonať pri danom pokuse. Týmto teda zabezpečíme, že na každom riešení sa vykoná rôzny počet mutácií, čo by malo zvýšiť diverzitu riešení.

4.5.6 Inverzia

Inverzia je špecifický operátor metódy Grouping Genetic Algorithm. Hlavnou myšlienkou inverzie je zlepšiť operáciu kríženia tak, že sa zmení poradie skupín, v našom prípade turnusov.

Pri operácii kríženia sa skupiny z jedného riešenia vkladajú do druhého riešenia. Avšak tým, že máme iba dva body kríženia v riešení, zoberieme na presun vždy iba skupiny, ktoré sa nachádzajú vedľa seba v riešení. Preto sa aplikuje operácia inverzie, ktorá zmení poradie skupín v riešení, čo má za následok vyššiu pravdepodobnosť prenesenia inej kombinácie skupín. Tento fakt má za následok to, že pri krížení sa môžu vytvoriť úplne nové riešenia, čím sa preskúma podstatne širšia množina riešení a poskytuje to možnosť nájsť kvalitnejšie riešenie.

My sme sa rozhodli implementovať túto operáciu ako jednoduchý algoritmus, ktorý niekoľkokrát za sebou vykoná výmenu dvoch náhodne vybraných skupín (turnusov).

Podobne ako v prípade mutácie aj operácia inverzie je ovplyvňovaná dvomi parametrami, ktoré stanovujú ako často a koľkokrát sa inverzia vykoná. Parameter *InvNum* určuje počet pokusov o inverziu a parameter *InvProb* stanovuje pravdepodobnosť s akou sa inverzia vykoná.

4.5.7 Výsledky experimentov

Pre otestovanie navrhnutej metaheuristiky a ohodnotenie jej výkonnosti sme vykonali niekoľko experimentov so základným nastavením parametrov. Tieto parametre

sme zvolili na základe predchádzajúcich skúseností s genetickým algoritmom. Základné nastavenie parametrov je nasledovné:

- *PopSize* = 100
- *MutNum* = 10
- *MutProb* = 0,2
- *InvNum* = 10
- *InvProb* = 0,2

Zároveň sa pri experimentovaní s metaheuristikami stretávame s tým, že sa spúšťa niekoľko replikácií, nakoľko je pri heuristických metódach využívaný generátor náhodných čísel, ktorý môže značne ovplyvniť získaný výsledok. Zároveň sa týmto spôsobom dá zistiť aj stabilita algoritmu a ako často sa mu podarí dosiahnuť dobré riešenia. My sme s našou implementáciou vykonávali pri základnom nastavení parametrov po 10 replikácií. Pre každú replikáciu bol stanovený časový limit 1200 sekúnd, v prípade datasetu DS10 to bolo 1800 sekúnd. Pre prípad druhého kritéria zastavenia bola zvolená hranica 200 výmen populácií bezo zmeny najlepšieho nájdeného riešenia.

Tabuľka 4.21. Výsledky metaheuristiky GGA pre jarný scenár so základnými nastaveniami

Dataset	***Najl. výsledok /BB***	***Grouping Genetic Algorithm***					
		Najlepšie riešenie	Najhoršie riešenie	Počet najlepších v repl.	Najlepší čas (s)	Najhorší čas(s)	Priemerný čas (s)
DS1	4/4	4	4	10	0,36	1,47	0,58
DS2	4/4	4	4	10	1,20	1,50	1,28
DS3	5/5	5	5	10	1,23	1,33	1,26
DS4	6/6	6	6	10	1,97	2,09	2,02
DS5	8/8	8	9	7	2,61	5,05	3,48
DS6	9/9	9	9	10	4,69	7,61	5,64
DS7	13/13	13	14	1	14,29	15,86	15,06
DS8	33/26	26	26	10	52,28	61,66	56,42
DS9	-/28	29	29	10	53,72	60,16	55,74
DS10	-/49	51	52	8	471,00	836,69	601,68

Tak ako aj pri ostatných navrhovaných metódach sa testovali všetky tri scenáre. Výsledky experimentov nájdeme v tabuľkách 4.21, 4.22 a 4.23. Opäť je v tabuľkách prítomný stĺpec *Najl. výsledok/BB*, ktorý obsahuje najlepšie nájdené riešenie pomocou modelu a spodnú hranicu riešenia. Stĺpec slúži na porovnanie výsledkov získaných pomocou metaheuristiky Grouping Genetic Algorithm s výsledkami získanými pomocou matematického modelu. Ďalšie stĺpce sa týkajú už samotnej metaheuristiky. V prvom stĺpci

sa nachádza najlepšie získané riešenie spomedzi všetkých replikácií, druhý stĺpec obsahuje najhoršie získané riešenie. V treťom stĺpci sa nachádza počet replikácií, kedy bolo dosiahnuté najlepšie riešenie. Posledné tri stĺpce sa týkajú času výpočtu a obsahujú najkratší, najdlhší a priemerný čas výpočtu v sekundách.

V tabuľke 4.21 vidíme výsledky s aplikáciou metaheuristiky Grouping Genetic Algorithm pri jarnom scenári. Z pohľadu výsledkov vidíme, že metaheuristika dosiahla v datasetoch DS1-DS8 optimálne výsledky pri 10 replikáciách. Zároveň aj najhoršie výsledky boli veľmi blízke optimu. V prípade datasetov DS1-DS4, DS6 a DS8 metaheuristika dosiahla optimum pri každej replikácii. V prípade datasetov DS9 a DS10 sa podarilo dosiahnuť výsledky veľmi blízke spodnej hranici riešenia. Ako môžeme vidieť aj v stĺpci *Počet najlepších v repl.* takmer pri všetkých prípadoch sme dosiahli vysoký podiel replikácií, kde sa dosiahlo najlepšie riešenie. Výnimku tvorí dataset DS7, kde sa podarilo najlepší výsledok dosiahnuť iba jedenkrát. Z pohľadu času výpočtu vidíme, že so zväčšujúcim sa datasetom rastie aj čas výpočtu. Avšak v prípade jarného scenára je veľmi nízky. Je to spôsobené tým, že sa v priebehu niekoľkých výmen populácií nájdu kvalitné riešenia, čo spôsobí rýchle ukončenie algoritmu na základe kritéria maximálneho počtu výmen populácií bezo zmeny najlepšieho nájdeného riešenia. Pri pohľade na minimálne, maximálne a priemerné časy vidíme, že sa líšia len nepatrne a rozdiely rastú hlavne pri väčších datasetoch.

Tabuľka 4.22. Výsledky metaheuristiky GGA pre letný scenár so základnými nastaveniami

Dataset	***Najl. výsledok /BB***	***Grouping Genetic Algorithm***					
		Najlepšie riešenie	Najhoršie riešenie	Počet najlepších v repl.	Najlepší čas (s)	Najhorší čas(s)	Priemerný čas (s)
DS1	4/4	4	4	10	0,59	2,62	0,87
DS2	4/4	4	4	10	2,48	2,61	2,53
DS3	5/5	5	5	10	2,27	2,55	2,35
DS4	6/6	6	6	10	3,94	4,48	4,13
DS5	8/8	9	9	10	5,56	5,78	5,64
DS6	9/9	9	10	9	9,59	15,81	12,08
DS7	-/13	13	14	1	29,84	46,33	33,52
DS8	-/26	26	26	10	117,22	140,17	124,56
DS9	-/28	30	31	2	246,10	481,71	288,07
DS10	-/49	54	55	6	1582,47	1808,36	1736,77

Tabuľka 4.22 obsahuje výsledky pre letný scenár. Pri prvých štyroch datasetoch nevidíme takmer nijaké zmeny oproti jarnému scenáru, Jedinou zmenou je mierne zvýšenie

času výpočtu. Pre datasety DS5-DS10 sa už však v niektorých prípadoch stretneme so zhoršením najlepšieho nájdeného riešenia oproti dolnej hranici v porovnaní s jarným scenárom. Najväčšiu zmenu pozorujeme v prípade datasetov DS9 a DS10. Zároveň sa mení aj stabilita algoritmu, kde sme pri replikáciách dostali väčšiu variabilitu vo výsledkoch, ako aj nižšie počty prípadov spomedzi všetkých replikácií, kedy sme dosiahli najlepšie nájdené riešenie. Takisto si môžeme všimnúť aj narastajúci čas výpočtu v porovnaní s jarným scenárom. Značné navýšenie času výpočtu sa začína pri datasete DS6, kde sa priemerný čas výpočtu v porovnaní s jarným scenárom zdvojnásobil. Niečo podobné pozorujeme aj pri datasetoch DS7-DS10 s rôznym pomerom nárastu. Je to spôsobené náročnejším procesom tvorby prípustného turnusu, nakoľko je v letnom scenári zvýšená spotreba energie, čo spôsobí nutnosť častejšieho nabíjania.

Tabuľka 4.23. Výsledky metaheuristiky GGA pre zimný scenár so základnými nastaveniami

Dataset	***Najl. výsledok /BB***	***Grouping Genetic Algorithm***					
		Najlepšie riešenie	Najhoršie riešenie	Počet najlepších v repl.	Najlepší čas (s)	Najhorší čas(s)	Priemerný čas (s)
DS1	4/4	4	4	10	0,96	2,10	1,26
DS2	4/4	4	4	10	3,39	3,85	3,61
DS3	5/5	5	5	10	3,22	5,80	3,60
DS4	6/6	6	6	10	5,58	7,90	6,13
DS5	10/8	8	9	2	8,18	17,10	10,18
DS6	11/9	9	10	6	13,11	20,80	15,40
DS7	-/13	13	14	1	40,28	68,33	47,61
DS8	-/26	26	26	10	161,98	187,31	177,41
DS9	-/28	31	32	8	381,86	674,88	499,15
DS10	-/49	54	57	1	1801,52	1815,46	1808,52

Výsledky pre zimný scenár nájdeme v tabuľke 4.23. Podobne ako v prípade jarného aj letného scenára sú riešenia v prípade datasetov DS1-DS4 identické s optimálnym riešením vo všetkých replikáciách. Vidíme však rozdiely v ostatných datasetoch v porovnaní s letným scenárom. Pre výsledky DS5-DS7 sa znovu zvýšila ich variabilita, respektíve, nedosahujeme najlepší nájdený výsledok až tak často. V prípade datasetov DS9 a DS10 sa dokonca zvýšila hodnota najlepšieho nájdeného výsledku v porovnaní s jarným a letným scenárom. Toto zvýšenie sa vzťahuje k náročnejším podmienkam fungovania, keďže v zimnom scenári je zvýšená spotreba energie a zároveň nižšia maximálna kapacita batérie. Čas výpočtu sa opätovne zvýšil aj v tomto prípade. Vidíme to pri všetkých datasetoch na priemernom čase výpočtu. Najkrajšou ukážkou, že čas výpočtu je značne ovplyvnený

maximálnou kapacitou batérie, môžeme vidieť v prípade datasetu DS10, kde sa metaheuristika ukončila z dôvodu vyčerpania času a nie z dôvodu dosiahnutia maximálneho počtu výmen populácií bezo zmeny najlepšieho nájdeného riešenia.

Druhou fázou experimentov bolo nastavenie parametrov metaheuristiky, za účelom zlepšenia riešenia a zvýšenia stability algoritmu. Keďže testovanie parametrov na všetkých scenároch by bolo komplikované, rozhodli sme sa testovať nastavenie parametrov iba pre prípad zimného scenára, kde sa vyskytovali najväčšie odchýlky od spodnej hranice riešenia. Pri nastavovaní parametrov sme postupovali metódou zafixovania parametrov na základných hodnotách a následne sme skúmali vplyv iba jedného parametra, ktorý sme menili. Z testovania sme vyradili parametre *MutNum* a *InvNum*, ktoré sú značne ovplyvňované parametrami *MutProb* a *InvProb*. Je to z dôvodu previazanosti medzi nimi. Parameter *MutNum* vyjadrujúci počet vykonaných pokusov o mutáciu je ovplyvnený parametrom *MutProb*, ktorý upravuje pravdepodobnosť vykonania mutácie. Čiže zmenou parametra *MutProb* docielime aj zmenu priemerného počtu vykonaných mutácií. Čo veľmi úzko súvisí s parametrom *MutNum*, ktorý vyjadruje počet pokusov o vykonanie mutácie.

Zafixovali sme parameter *MutNum* na hodnotu 10. Rovnako aj parameter *InvNum* bol zafixovaný na hodnotu 10. Počet vykonaných replikácií bol 10 a prvé kritérium zastavenia, počet výmen generácií bezo zmeny najlepšieho nájdeného riešenia, zostalo na hodnote 200. Zároveň bol maximálny časový limit na výpočet jednej replikácie stanovený na 600 sekúnd. Pre jednotlivé parametre metaheuristiky sme testovali nasledujúce hodnoty parametrov:

- *PopSize* = {25, 50, 100, 150, 200}
- *MutProb* = {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}
- *InvProb* = {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}

V tabuľkách 4.24 a 4.25 sa nachádzajú výsledky experimentov pre parameter *PopSize*, reprezentujúci veľkosť populácie. V tabuľke 4.24 je uvedený najlepší získaný výsledok, ako aj počet koľkokrát sa podarilo tento výsledok dosiahnuť v rámci replikácií. Ako môžeme vidieť, tak v prípade takmer všetkých hodnôt parametra sa nijak nezlepšili, ale ani nezhoršili výsledky pre datasety DS1-DS4 a DS8. Preto sa zameriame najmä na ostatné datasety.

Pri zvyšovaní parametra *PopSize* si môžeme všimnúť, že sa zlepšujú výsledky pre niektoré datasety a zároveň sa zvyšuje počet, koľkokrát sme dosiahli najlepší výsledok.

Najlepšie to asi vidno v prípade datasetov DS6, DS7 a DS10. Z pohľadu výsledkov a stability sa ukazuje, že najvhodnejšia hodnota parametra *PopSize* je 150, pri ktorej sa dosiahli asi najstabilnejšie výsledky. Zároveň sa ukazuje tendencia zlepšovania výsledkov so zvyšujúcou sa veľkosťou populácie.

Tabuľka 4.24. Najlepší nájdený výsledok (*Najl Výs*) a počet dosiahnutí najlepšieho výsledku v replikáciách (*Poč najl*) metaheuristiky GGA pre zimný scenár pre parameter *PopSize*

PopSize	***25***		***50***		***100***		***150***		***200***	
Dataset	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl
DS1	4	10	4	10	4	10	4	10	4	10
DS2	4	10	4	10	4	10	4	10	4	10
DS3	5	10	5	10	5	10	5	10	5	10
DS4	6	9	6	10	6	10	6	10	6	10
DS5	9	10	8	1	8	1	8	2	8	1
DS6	9	3	9	5	9	8	9	10	9	8
DS7	14	10	14	10	14	10	13	1	13	1
DS8	26	10	26	10	26	10	26	10	26	10
DS9	31	4	31	7	30	2	30	2	31	9
DS10	54	1	55	2	55	4	55	6	55	2

Tabuľka 4.25. Priemerný výpočtový čas (s) na jednu replikáciu metaheuristiky GGA pre zimný scenár pre parameter *PopSize*

PopSize	***25***	***50***	***100***	***150***	***200***
Dataset	Priem. čas	Priem. čas	Priem. čas	Priem. čas	Priem. čas
DS1	0,5	0,8	1,2	1,8	2,2
DS2	1,1	1,7	3,2	5,1	6,8
DS3	1,1	1,6	2,9	4,9	6,4
DS4	2,0	2,9	5,1	8,7	11,1
DS5	2,5	3,9	7,5	12,5	16,3
DS6	4,2	7,4	15,6	27,2	32,4
DS7	12,4	22,9	39,9	68,1	79,7
DS8	50,9	87,5	167,7	263,1	335,0
DS9	102,6	247,8	450,1	598,4	602,8
DS10	602,2	604,1	606,2	612,9	611,7

V tabuľke 4.25 sú uvedené priemerné časy výpočtu pre jednotlivé hodnoty parametra *PopSize*. Ako vidíme v tabuľke, tak sa zväčšujúca veľkosť populácie jednoznačne odráža na priemernej dĺžke výpočtu jednej replikácie. Je to samozrejme spôsobené nutnosťou spracovať väčšiu populáciu, čo zaberá viac času. Spoločne s faktom, že sa aplikuje kritérium ukončenia závislé na počte výmen je zrejmé, že s narastajúcou veľkosťou populácie sa bude

zvyšovať priemerný čas výpočtu približne rovnakou rýchlosťou ako rastie veľkosť populácie.

Druhým testovaným parametrom je pravdepodobnosť mutácie – *MutProb*. Výsledky a výpočtový čas na replikáciu pre tento parameter sú uvedené v tabuľkách 4.26 a 4.27. Keď sa pozrieme na kvalitu a stabilitu získaných výsledkov v tabuľke 4.26, tak môžeme vidieť, že v prípade datasetov DS1-DS4 nedošlo k žiadnej zmene. Preto sa zameriame na ostatné datasety. V prípade, keď bola pravdepodobnosť mutácie nulová, vidíme vysokú stabilitu dosahovaných výsledkov, avšak v niektorých prípadoch (DS5, DS7 a DS10) nedosiahla v porovnaní s ostatnými hodnotami parametra *MutProb* najlepšie výsledky. Ďalej vidíme, že zvýšenie parametra *MutProb* nám prinieslo mierne zlepšenie z pohľadu výsledkov, avšak nie sú dosahované veľmi často. Najviac zlomov v správaní sa odohráva medzi hodnotami 0,4 a 0,6. V prípade datasetu DS6 nastáva zlom v stabilite, kde nižšia hodnota parametra dáva stabilnejšie výsledky. Pri datasete DS7 sa však vyššia hodnota parametra podpísala na kvalitnejšom výsledku. Pri datasete DS10 je opätovne vyššia stabilita pri hodnote 0,4.

Tabuľka 4.26. Najlepší nájdený výsledok (*Najl Výs*) a počet dosiahnutí najlepšieho výsledku v replikáciách (*Poč najl*) metaheuristiky GGA pre zimný scenár pre parameter *MutProb*

Mut Prob	***0.0***		***0.2***		***0.4***		***0.6***		***0.8***		***1.0***	
Data set	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl
DS1	4	10	4	10	4	10	4	10	4	10	4	10
DS2	4	10	4	10	4	10	4	10	4	10	4	10
DS3	5	10	5	10	5	10	5	10	5	10	5	10
DS4	6	10	6	10	6	10	6	10	6	10	6	10
DS5	9	10	8	1	9	10	8	1	9	10	8	1
DS6	9	10	9	8	9	8	9	6	9	5	9	5
DS7	14	10	14	10	14	10	13	4	13	1	14	10
DS8	26	10	26	10	26	10	26	10	26	10	26	10
DS9	30	1	31	8	30	1	31	8	31	7	31	9
DS10	55	5	54	1	54	3	54	1	54	1	54	1

Tabuľka 4.27 nám ponúka pohľad na priemernú rýchlosť výpočtu jednej replikácie pre meniace sa hodnoty parametra *MutProb*. Pri detailnom pohľade na jednotlivé datasety je zrejmé, že rýchlosť výpočtu nie je ovplyvňovaná nastavením parametra *MutProb*, keďže po zvýšení parametra sa v niektorých prípadoch čas výpočtu skrátil a v niektorých predĺžil. Zároveň, pre každý dataset je toto správanie iné. Napríklad v prípade datasetu DS7 sa medzi hodnotami 0,4 a 0,6 čas výpočtu predĺžil, avšak v prípade datasetu DS9 je to presne naopak.

Môžeme teda povedať, že zmena parametra *MutProb* nevplýva jasným spôsobom na čas výpočtu replikácie. Z tohto môžeme usudzovať, že operácia mutácie je pomerne rýchla v porovnaní s operáciou kríženia, a teda jej vplyv na rýchlosť výpočtu je nízky.

Tabuľka 4.27. Priemerný výpočtový čas (s) na jednu replikáciu metaheuristiky GGA pre zimný scenár pre parameter *MutProb*

MutProb	***0.0***	***0.2***	***0.4***	***0.6***	***0.8***	***1.0***
Dataset	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)
DS1	1,2	1,1	1,2	1,1	1,4	1,7
DS2	3,1	3,2	3,2	3,3	4,4	4,2
DS3	2,9	3,1	3,2	3,2	4,0	4,2
DS4	4,9	5,4	5,7	6,4	7,7	9,6
DS5	7,6	8,0	7,6	8,1	10,1	11,4
DS6	15,9	16,1	15,1	16,2	18,5	16,6
DS7	40,9	42,7	40,2	52,9	50,1	42,1
DS8	166,4	182,1	176,1	159,8	176,3	165,5
DS9	451,3	479,8	490,6	426,8	458,7	504,6
DS10	608,2	605,6	606,7	605,2	607,4	608,0

Keďže nastavenie parametra neovplyvňuje jednoznačným spôsobom rýchlosť výpočtu, je možné parameter nastaviť na ľubovoľnú hodnotu. Avšak po zosumarizovaní výsledkov s parametrom *MutProb* odporúčame držať pravdepodobnosť mutácie v rozmedzí hodnôt 0,4 a 0,6. Pri týchto hodnotách sa pre väčšinu datasetov zvýšila stabilita výsledkov a zároveň sa pri niektorých podarilo zlepšiť dosiahnuté výsledky.

Posledným testovaným parametrom je parameter *InvProb*, čiže pravdepodobnosť vykonania inverzie. Výsledky pre tento parameter nájdeme v tabuľkách 4.28 a 4.29. Parameter *InvProb* ovplyvňuje iba pravdepodobnosť reorganizácie riešenia, nie jeho zmenu, avšak súvisí s operáciou kríženia, keďže upravuje štruktúru vstupu do tejto metódy. Získané výsledky pre jednotlivé hodnoty parametra nájdeme v tabuľke 4.28. Pri výsledkoch pre jednotlivé scenáre opäť môžeme zanedbať datasety DS1-DS4, keďže z pohľadu výsledkov sa nepodarilo zaznamenať žiadnu zmenu. Pri ostatných datasetoch si môžeme všimnúť mierne zmeny v stabilite, kde pri vyšších hodnotách sa podarilo dosiahnuť lepšiu stabilitu viditeľnú najmä v prípade datasetu DS6. V prípade datasetu DS5 vyššie hodnoty parametra zabezpečili kvalitnejšie výsledky. Pri datasete DS10 sa opätovne podarilo získavať lepšie výsledky a hlavne s vyššou stabilitou pri hodnotách parametra 0,6 a 0,8. Tieto fakty naznačujú, že by bolo vhodné nastaviť parameter *InvProb* na hodnoty vyššie ako 0,6.

Tabuľka 4.28. Najlepší nájdený výsledok (*Najl Výs*) a počet dosiahnutí najlepšieho výsledku v replikáciách (*Poč najl*) metaheuristiky GGA pre zimný scenár pre parameter *InvProb*

Inv Prob	***0.0***		***0.2***		***0.4***		***0.6***		***0.8***		***1.0***	
Data set	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl	Najl Výs	Poč najl
DS1	4	10	4	10	4	10	4	10	4	10	4	10
DS2	4	10	4	10	4	10	4	10	4	10	4	10
DS3	5	10	5	10	5	10	5	10	5	10	5	10
DS4	6	10	6	10	6	10	6	10	6	10	6	10
DS5	9	10	9	10	9	10	8	2	9	10	8	2
DS6	9	6	9	7	9	5	9	6	9	8	9	9
DS7	14	10	13	1	13	1	14	10	14	10	13	1
DS8	26	10	26	10	26	10	26	10	26	10	26	10
DS9	31	10	30	1	31	9	31	7	31	7	30	1
DS10	55	2	53	1	55	2	54	1	53	1	55	2

Tabuľka 4.29 obsahuje priemerný čas výpočtu replikácie pre jednotlivé hodnoty parametra *InvProb*. Ako vidno v tabuľke, tak pre väčšinu datasetov sa po zvýšení parametra *InvProb* nad hodnotu 0,4 mierne znížil čas výpočtu. Je to zrejme spôsobené tým, že reorganizácia riešenia pomocou inverzie pomáha rýchlejšej aplikácii kríženia. Čiže ak chceme znížiť čas výpočtu, je vhodné tento parameter zvýšiť.

Tabuľka 4.29. Priemerný výpočtový čas (s) na jednu replikáciu metaheuristiky GGA pre zimný scenár pre parameter *InvProb*

InvProb	***0.0***	***0.2***	***0.4***	***0.6***	***0.8***	***1.0***
Dataset	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)	Priem. čas(s)
DS1	1,0	1,1	1,5	1,0	1,1	1,0
DS2	3,4	3,5	3,8	3,0	2,9	2,8
DS3	3,3	3,3	3,8	2,8	2,8	2,8
DS4	5,7	6,4	5,5	4,9	5,1	4,7
DS5	8,0	8,1	8,1	8,1	7,1	7,4
DS6	16,4	17,9	17,4	14,0	16,5	14,5
DS7	41,5	43,1	44,4	35,9	38,6	39,9
DS8	156,1	172,7	171,9	164,7	159,3	153,0
DS9	399,3	460,5	507,8	445,1	462,6	444,5
DS10	606,8	608,1	607,0	607,7	608,5	608,3

Na základe experimentov môžeme odporúčať zvýšenie parametra *InvProb* na hodnoty v blízkosti 0,8. Pri týchto hodnotách bol mierne znížený čas výpočtu a zároveň sa podarilo zvýšiť stabilitu získavaných výsledkov. V niektorých prípadov sa dokonca podarilo dosiahnuť aj lepšie výsledky.

Tabuľka 4.30. Nastavenie parametrov metahueristiky GGA

Parameter	***Základné nastavenie***	***Upravené nastavenie***
PopSize	100	150
MutProb	0,2	0,5
MutNum	10	10
InvProb	0,2	0,8
InvNum	10	10

Poslednou časťou experimentov je porovnanie výkonnosti algoritmu pri základnom nastavení parametrov a po nastavení parametrov. Z dôvodu testovania parametrov iba na zimnom scenári a zároveň z dôvodu, že pri zimnom scenári sa prejavili najväčšie rozdiely sme sa rozhodli porovnanie realizovať len na zimnom scenári pre všetky datasety. Zároveň sme zvýšili počet replikácií na 20 a čas výpočtu jednej replikácie bol stanovený na 1800 sekúnd. Druhé kritérium zastavenia, počet výmen generácií bezo zmeny najlepšieho nájdeného riešenia, zostal nastavený na hodnotu 200. Ostatné parametre boli nastavené podľa tabuľky 4.30.

Tabuľka 4.31. Výsledky metaheuristiky GGA pre zimný scenár so základnými nastaveniami

Dataset	***Grouping Genetic Algorithm – základné nastavenie***					
	Najlepší výsledok	Najhorší výsledok	Počet najlepších v repl.	Najlepší čas (s)	Najhorší čas(s)	Priemerný čas (s)
DS1	4	4	20	0,89	2,10	1,18
DS2	4	4	20	3,36	3,85	3,60
DS3	5	5	20	3,22	5,80	3,48
DS4	6	6	20	5,58	9,55	6,28
DS5	8	9	3	8,10	17,10	9,24
DS6	9	10	11	13,11	23,24	15,69
DS7	13	14	1	40,28	68,33	46,77
DS8	26	26	20	161,98	194,68	175,90
DS9	31	32	14	373,28	674,88	475,94
DS10	54	57	2	1801,52	1815,72	1808,78

V tabuľke 4.31 nájdeme výsledky pre experimenty so základným nastavením parametrov na zvýšenom počte replikácií. Tabuľka 4.32 obsahuje výsledky pre upravené nastavenie parametrov. Ako môžeme vidieť pri ich porovnaní, tak pri základnom nastavení parametrov dosiahla metóda GGA podstatne nižšie časy výpočtu. Naproti tomu, nastavenie parametrov pomohlo zlepšiť stabilitu výsledkov, ako aj ich zlepšenie v prípade datasetu DS9. Ako mieru stability využívame počet replikácií, pri ktorých metaheuristika dosiahla najlepší výsledok. S výnimkou datasetov DS1-DS4 a DS8, kde obe nastavenia parametrov

dosiahli najlepší výsledok vo všetkých replikáciách, je zrejmé, že nastavenie parametrov zvyšuje počet replikácií, kedy sa dosiahne najlepší výsledok. V prípade datasetu DS5 sa tento počet nezvýšil, ale v prípade datasetov DS6, DS7 a DS10 je nárast viditeľný. V prípade datasetu DS9 je síce počet replikácií s najlepším riešením len 1, ale toto riešenie je lepšie ako v prípade základného nastavenia parametrov. Ak by sme zanedbali toto nájdené riešenie, tak by sa počet replikácií s najlepším riešením pri upravenom nastavení zvýšil na 17. Čiže aj v tomto prípade môžeme pozorovať zvýšenie stability výsledku.

Tabuľka 4.32. Výsledky metaheuristiky GGA pre zimný scenár s upravenými nastaveniami

Dataset	***Grouping Genetic Algorithm – upravené nastavenie***					
	Najlepší výsledok	Najhorší výsledok	Počet najlepších v repl.	Najlepší čas (s)	Najhorší čas(s)	Priemerný čas (s)
DS1	4	4	20	1,27	2,45	1,37
DS2	4	4	20	4,75	4,97	4,82
DS3	5	5	20	4,51	4,98	4,70
DS4	6	6	20	7,64	10,67	8,37
DS5	8	9	2	11,09	18,34	11,70
DS6	9	10	14	17,32	33,82	21,99
DS7	13	14	5	53,47	99,05	62,72
DS8	26	26	20	209,44	258,99	222,68
DS9	30	32	1	629,84	1251,93	944,56
DS10	54	56	5	1801,01	1819,99	1806,66

4.5.8 Zhodnotenie výsledkov

V tejto časti sme predstavili návrh metaheuristiky Grouping Genetic Algorithm na riešenie úlohy návrhu turnusov elektrických autobusov vo verejnej doprave. Pre implementáciu danej metaheuristiky sme museli navrhnúť jednotlivé operátory kríženia, mutácie a inverzie. Zároveň sme otestovali výkonnosť metaheuristiky z pohľadu výsledkov aj času výpočtu.

Z prvých experimentov so základným nastavením parametrov je jasné, že návrh operátorov veľmi ovplyvňuje kvalitu získaných riešení. V našom prípade sa podarilo navrhnúť dobré operátory, o čom svedčia dosiahnuté výsledky, kde sme pre menšie datasety dosiahli optimálne výsledky vo všetkých replikáciách pri všetkých scenároch. Pre stredne veľké a rozsiahlejšie úlohy sa nie vždy podarilo dosiahnuť optimálne výsledky, ale vo väčšine prípadov bol rozdiel medzi spodnou hranicou a získaným výsledkom minimálny.

Takisto sme zistili, že aj výkonnosť našej metaheuristiky je značne ovplyvňovaná scenárom, pre ktorý sa úloha rieši, keďže čas výpočtu narástol niekoľkonásobne pri prechode

z jarného scenára na letný, kde bola zvýšená spotreba energie. Opätovný niekoľkonásobný nárast času sme pozorovali aj v prípade prechodu z letného scenára na zimný, ktorý je ovplyvnený poklesom maximálnej kapacity batérie.

Druhou časťou experimentov bolo zistenie vplyvu nastavenia parametrov na beh metaheuristiky. Skúmali sme vplyv troch parametrov: *PopSize*, *MutProb* a *InvProb*. Každý parameter ovplyvňoval metaheuristiku iným spôsobom. Zvýšenie parametra *PopSize* reprezentujúceho veľkosť populácie zvyšuje čas výpočtu, ale zároveň zlepšilo výsledky a stabilitu algoritmu. V našom prípade odporúčame hodnoty veľkosti populácie medzi 100 až 200 jedincov. Druhý parameter *MutProb* reprezentujúci pravdepodobnosť mutácie neovplyvňoval rýchlosť výpočtu, ani kvalitu výsledkov, ale hodnoty v okolí 0,5 zvýšili stabilitu získavaných výsledkov. Posledný parameter *InvProb* reprezentujúci pravdepodobnosť inverzie takisto zlepšil stabilitu výsledkov, ak sa držal v hodnotách okolo 0,8.

Poslednou časťou experimentov bolo porovnanie výsledkov metaheuristiky GGA na zimnom scenári pre základné a upravené nastavenie parametrov metódy. Pri týchto experimentoch sa ukázalo, že úprava parametrov síce zvýšila časovú náročnosť metódy, ale zároveň poskytla zvýšenú stabilitu poskytovaných výsledkov reprezentovanú počtom replikácií, kedy bolo dosiahnuté najlepšie nájdené riešenie.

4.6 Porovnanie metód

V tejto kapitole sme zhromaždili najlepšie dosiahnuté výsledky pre všetky datasety a scenáre pre predstavené metódy za účelom porovnania riešení medzi jednotlivými metódami. V každej tabuľke sa nachádza stĺpec pre každú metódu s najlepším dosiahnutým riešením a časom výpočtu. Stĺpec *Matematický model* predstavuje riešenie pomocou matematického modelu a IP solvera. Stĺpec *Redukcia vstupov* obsahuje výsledky pre metódu redukcie vstupov s následnou aplikáciou riešenia pomocou matematického modelu. Predposledný stĺpec *CG* predstavuje metódu generovania stĺpcov a posledný stĺpec tabuľky *GGA* reprezentuje riešenie pomocou metódy Grouping genetic algorithm.

Tabuľka 4.33 obsahuje výsledky pre jarný scenár. V tomto prípade dosiahli pri menších úlohách najlepšie výsledky matematický model, redukcia vstupov a metóda GGA. Z pohľadu výpočtového času najlepšie vychádza z porovnania metóda GGA. Za ňou nasleduje metóda CG, ktorá síce nedosiahla najlepšie výsledky, ale bola schopná dosiahnuť veľmi dobré výsledky v čase podstatne kratšom, ako bolo riešenie pomocou matematického

modelu a redukcie vstupov. Pre väčšie datasety ukazujú najlepšie výsledky heuristické metódy CG a GGA, ktoré sú schopné riešiť aj úlohy veľkého rozsahu na rozdiel od prvých dvoch metód.

Tabuľka 4.33. Porovnanie riešenia pomocou metód Matematický model, Redukcia vstupov, Generovanie stĺpcov (CG) a Grouping genetic algorithm (GGA) na jarnom scenári

Dataset	*BB*	*Matematický model*		*Redukcia vstupov*		*CG*		*GGA*	
		Rieš.	Čas	Rieš	Čas	Rieš	čas	Rieš	čas
DS1	4	4	3	4	1,7	5	1,0	4	0,6
DS2	4	4	12,2	4	4,6	5	9,5	4	1,3
DS3	5	5	35,6	5	9,6	6	8,2	5	1,3
DS4	6	6	29,7	6	15,4	7	5,3	6	2,0
DS5	8	8	169,2	8	49,1	9	7,2	8	3,5
DS6	9	9	281,6	9	58,4	10	21,7	9	5,6
DS7	13	13	13414	13	295	14	56,7	13	15,1
DS8	26	33	57600	26	2083	27	73,2	26	56,4
DS9	28	-	57600	-	57600	29	291,4	29	55,7
DS10	49	-	57600	-	57600	54	1724	51	601,7

Tabuľka 4.34. Porovnanie riešenia pomocou metód Matematický model, Redukcia vstupov, Generovanie stĺpcov (CG) a Grouping genetic algorithm (GGA) na letnom scenári

Dataset	*BB*	*Matematický model*		*Redukcia vstupov*		*CG*		*GGA*	
		Rieš.	Čas	Rieš	Čas	Rieš	čas	Rieš	čas
DS1	4	4	3,6	4	1,7	5	1,3	4	0,9
DS2	4	4	117,5	4	4,6	5	38,3	4	2,5
DS3	5	5	43,0	5	9,7	6	11,1	5	2,4
DS4	6	6	712,2	6	18,5	7	6,8	6	4,1
DS5	8	9	57600	8	53,2	10	15,8	9	5,6
DS6	9	10	57600	9	74,8	10	62,0	9	12,1
DS7	13	27	57600	13	397,4	15	182,1	13	33,5
DS8	26	-	57600	26	10625	27	211,6	26	124,6
DS9	28	-	57600	-	57600	33	2061,0	30	288,1
DS10	49	-	57600	-	57600	55	18315	54	1736,8

V tabuľke 4.34 sú uvedené výsledky jednotlivých metód pri letnom scenári. Pre datasety DS1-DS4 sa metóde *Matematický model* podarilo dosiahnuť optimálne výsledky, avšak pri väčších datasetoch začala mať problémy. Metóda redukcie vstupov bola schopná zlepšiť riešiteľnosť pomocou matematického modelu, čo dokazujú výsledky v tabuľke 4.34, kde bola schopná nájsť optimálne riešenia až po dataset DS8. Pre najväčšie datasety DS9 a DS10 sa však podarilo nájsť dobré výsledky iba metódam CG a GGA. Zároveň vidno, že heuristické metódy nachádzajú veľmi dobré výsledky v podstatne kratšom čase.

Tabuľka 4.35. Porovnanie riešenia pomocou metód Matematický model, Redukcia vstupov, Generovanie stĺpcov (CG) a Grouping genetic algorithm (GGA) na zimnom scenári

Dataset	*BB*	*Matematický model*		*Redukcia vstupov*		*CG*		*GGA*	
		Rieš.	Čas	Rieš	Čas	Rieš	čas	Rieš	čas
DS1	4	4	10,4	4	2,5	5	4,6	4	1,4
DS2	4	4	40,0	4	4,1	4	1,2	4	4,8
DS3	5	5	629,3	5	32,7	6	23,7	5	4,7
DS4	6	6	1237	6	117	7	38,0	6	8,4
DS5	8	10	57600	9	57600	9	120,0	8	11,7
DS6	9	11	57600	9	716,2	10	146,0	9	22,0
DS7	13	-	57600	17	57600	14	358,0	13	62,7
DS8	26	-	57600	-	57600	27	394,0	26	222,7
DS9	28	-	57600	-	57600	31	2674,0	30	944,5
DS10	49	-	57600	-	57600	56	31409	54	1806

Výsledky pre zimný scenár sa nachádzajú v tabuľke 4.35. Aj v prípade zimného scenára začína metóda *Matematický model* zlyhávať už pri menších datasetoch. Zároveň si môžeme všimnúť aj problémy pri metóde *Redukcia vstupov*, ktorá zlyhala už v prípade datasetu DS7. Avšak bola schopná zlepšiť riešiteľnosť úlohy pomocou matematického modelu a zároveň znížila čas potrebný na výpočet. Metóda *CG* dosiahla dobré výsledky aj pre najväčšie datasety v pomerne dobrom čase. Najlepšie z pohľadu času vyšla metóda *GGA*, ktorá dosiahla aj pri najväčších datasetoch najlepšie výsledky.

5 Záver

V tejto práci sme sa venovali úlohe návrhu turnusov elektrických autobusov vo verejnej doprave, ktorá je variantom tradičnej úlohy návrhu turnusov. Keďže sa v úlohe využívajú elektrické autobusy, je nutné ju upraviť a zahrnúť špecifiká spojené s elektrickými autobusmi, ktorými sú obmedzený dojazd a nutnosť nabíjania, ktoré trvá dlhší časový úsek. Nabíjačky, na ktorých je možné nabíjanie, sú rozmiestnené v dopravnej sieti. Riešením tejto úlohy riešime jeden z problémov pri transformácii flotily konvenčných autobusov na flotilu elektrických autobusov, pričom táto transformácia sa bude zrejme vyžadovať vo viacerých mestách kvôli trendu znižovania emisií. Zároveň prevádzka elektrických autobusov prinesie zníženie operačných nákladov, keďže cena elektriny je momentálne omnoho nižšia ako cena palív.

Na riešenie tejto úlohy je možné využiť niekoľko prístupov, ktoré sme uviedli v kapitole o stave danej problematiky. V súlade s vytýčenými cieľmi práce sme sa venovali metódam na riešenie úlohy návrhu turnusov elektrických autobusov, ktoré zahŕňajú exaktné ako aj heuristické metódy riešenia.

Na modelovanie úlohy sme formulovali matematický model špecifický pre naše zadanie, čiže spĺňal všetky potrebné podmienky týkajúce sa spotreby energie a nabíjania, ale aj štandardné podmienky návrhu turnusov. Následne sme s týmto modelom vykonali numerické experimenty, pričom sme testovali exaktné riešenie úlohy pomocou štandardného IP solvera Xpress IVE. Na základe experimentov sme zistili exaktné riešenia testovaných úloh, respektíve spodnú hranicu riešenia, čo nám poslúžilo ako porovnávacie kritérium pri ostatných skúmaných metódach. Zároveň sme boli schopný stanoviť veľkosť úloh riešiteľných exaktným spôsobom.

V ďalších častiach sme sa venovali návrhu metód na riešenie zadefinovanej úlohy. Prvou skúmanou metódou bola metóda redukcie vstupov s využitím riešenia úlohy pomocou matematického modelu a IP solvera. Táto metóda predspracuje vstupné dáta heuristickým spôsobom založeným na spájaní spojov, čím je schopná zredukovať množstvo vstupov odosielaných do matematického modelu. Týmto spôsobom sa podarilo riešiť úlohy väčšieho rozsahu a zároveň znížiť čas výpočtu potrebný na riešenie úloh.

Druhou skúmanou metódou bola metóda založená na metóde generovania stĺpcov, ktorá pracuje s iným pohľadom na problém a na riešenie častí problému využíva exaktné

metódy. Táto metóda však vo všeobecnosti nezaručuje nájdenie exaktného riešenia, ale je schopná riešiť exaktne LP-relaxáciu úlohy. Pri tejto metóde sme sa stretli s problémom riešenia pod-problému, kde sa generujú nové prípustné turnusy. Pre rýchlejšie riešenie pod-problému sme navrhli niekoľko metód. Vo všetkých z nich sa využíva algoritmus na hľadanie *k*-najkratších ciest. Pri niektorých metódach sme využili charakter problému a redukovali tak digraf, na ktorom sa hľadali najkratšie cesty, pričom sme ďalej pre tvorbu prípustných turnusov využili heuristické metódy na pridávanie nabíjacích udalostí a odstraňovanie spojov z turnusu. Pre porovnanie výkonnosti jednotlivých metód riešenia pod-problému sa vykonali numerické experimenty, ktoré ukazujú na využitie poslednej navrhnutej metódy. Pri experimentoch sa ukázalo, že metóda generovania stĺpcov je schopná dosiahnuť dobré výsledky aj na úlohách reálneho rozsahu, pričom čas výpočtu bol výrazne nižší ako v prípade použitia exaktných metód.

V ďalšej časti práce sme sa venovali heuristickému prístupu, a síce metóde založenej na algoritme Grouping Genetic Algorithm. Táto metóda je špecifickou odnožou genetického algoritmu, ktorá je prispôsobená na prácu s úlohami, kde množinu prvkov zadeľujeme do skupín. V našej implementácii sme predstavili operátory kríženia, mutácie a inverzie, ktoré sú prispôsobené na prácu s úlohou návrhu turnusov pre elektrické autobusy. Metóda bola testovaná na niekoľkých datasetoch a porovnaná s exaktnými výsledkami. Zároveň boli vypracované odporúčania pre nastavenie parametrov tejto metódy na základe vykonaných experimentov. Poslednou časťou testovania metódy bolo overenie zlepšenia správania sa algoritmu po nastavení parametrov. Ukázalo sa, že nastavenie parametrov značne ovplyvňuje stabilitu algoritmu a rýchlosť výpočtu niektorých fáz algoritmu. Metóda sa ukázala ako veľmi vhodná na použitie pri reálnych úlohách, nielen z pohľadu získaných výsledkov, ale aj z pohľadu výpočtového času.

V záverečnej časti práce sme sa venovali porovnaniu výsledkov jednotlivých metód, pričom sa ukázalo, že z pohľadu výsledkov dominujú metódy založené na výpočte matematického modelu, avšak nie je možné pomocou nich riešiť úlohy väčšieho rozsahu. V prípade heuristických metód generovania stĺpcov a Grouping genetic algorithm sa ukazuje ich sila najmä z pohľadu výpočtového času a ich možnosť aplikácie na úlohy veľkého rozsahu.

5.1 Prínos dizertačnej práce pre rozvoj vedného oboru

Z pohľadu vedného odboru sa táto práca zaoberá riešením jednej zo štandardných úloh operačného výskumu, ktorou je úloha návrhu turnusov. Táto úloha má veľké množstvo variantov, medzi ktoré patrí aj úloha návrhu turnusov pre elektrické autobusy.

Jedným z prínosov práce pre rozvoj vedného odboru je vytvorenie prehľadu o úlohe návrhu turnusov a špeciálne úlohe návrhu turnusov pre elektrické autobusy. Jedným z prínosov práce je vytvorenie matematického modelu úlohy, pri ktorej sa uvažuje rozmiestnenie nabíjacích staníc v dopravnej sieti, keďže väčšina autorov zaoberajúcich sa touto témou zvažuje iba umiestnenie nabíjačiek na jednom mieste (zvyčajne v depe). My v našej práci uvažujeme variant, kde sú nabíjačky rozmiestnené v dopravnej sieti a nie je nutné nabíjať batériu vždy do plnej kapacity. Ďalším prínosom dizertačnej práce je návrh inovatívnych metód založených na informatických princípoch na riešenie úlohy založených na rôznych prístupoch k riešeniu. Sú to metóda redukcie vstupov s využitím exaktného riešenia redukovanej úlohy, ďalej metóda generovania stĺpcov s návrhom heuristík pre riešenie úlohy hľadania zlepšujúceho turnusu a metóda založená na genetickom algoritme špecifickom pre úlohy delenia množiny prvkov na skupiny Grouping genetic algorithm.

5.2 Prínos dizertačnej práce pre prax

Keďže v posledných rokoch začínajú naberať na význame elektrické vozidlá, je nutné riešiť problémy s ich nasadzovaním do prevádzky. V prípade mestskej verejnej dopravy sú to elektrické autobusy. Avšak tie sa od konvenčných autobusov líšia a pri ich nasadzovaní do prevádzky je nutné počítať s ich špecifikami, najmä s obmedzeným dojazdom a dlhým časom nabíjania.

V blízkom čase môžeme očakávať transformáciu flotily konvenčných autobusov na flotilu elektrických autobusov. Pri tejto transformácii je nutné riešiť aj úlohu návrhu turnusov pre elektrické autobusy, čomu sa venujeme práve v tejto práci. Asi za najväčší prínos tejto dizertačnej práce pre praktické využitie považujeme návrh matematického modelu a riešiacich metód využiteľných v systémoch pre podporu rozhodovania v oblasti verejnej dopravy. Aplikáciou metód je možné uľahčiť prechod z flotily konvenčných na flotilu elektrických autobusov vo verejnej doprave. Metódy sú veľmi dobre škálovateľné a preto využiteľné nielen v mestskej doprave, na ktorej boli testované, ale aj pre prímestskú dopravu a samozrejme pre súkromné prepravné spoločnosti, ktoré prevádzkujú pravidelné zásobovanie zákazníkov tovarom a majú záujem využívať elektrické vozidlá.

5.3 Odporúčania pre budúci výskum

V tejto práci sme sa venovali úlohe návrhu turnusov pre elektrické autobusy iba v prostredí verejnej dopravy. Túto úlohu by bolo možné aplikovať aj na prostredie prímestskej dopravy, čo však vyžaduje ďalšie testovanie navrhnutých metód. Preto by sa budúci výskum v tejto oblasti mohol zamerať na otestovanie navrhnutých metód na iných typoch datasetov, ako aj širšie testovanie vplyvu niektorých parametrov metód na kvalitu riešenia a výpočtový čas.

Ďalším smerovaním by mohli byť úpravy modelu za účelom jeho zjednodušenia, ako aj možnosť rozšírenia modelu o rozhodovanie o využití nabíjacích staníc s cieľom minimalizovať ich počet. V tomto smere je možná aj úprava účelovej funkcie a následná úprava navrhnutých metód, aby boli schopné vysporiadať sa s touto zmenou.

Literatúra

[1] Petr Cenek, Valent Klima, and Jaroslav Janáček. *Optimalizace dopravních a spojových procesů*. Vysoká škola dopravy a spojov v Žiline. Edičné stredisko VŠDS, 1994. ISBN 80-7100-197-X.

[2] Bunte, Stefan & Kliewer, Natalia. (2010). *An overview on vehicle scheduling models*. Public Transport. 1. 299-317. 10.1007/s12469-010-0018-5.

[3] P. Czimmermann. *On a certain transport scheduling problem for heterogeneous bus fleet*. Communications, no. 3, pp. 17–18, 2006.

[4] ZeEUS project. (2016) *Zeeus ebus report: An overview of electric buses in Europe*. [Online]. Available at: http://zeeus.eu/uploads/publications/documents/zeeus-ebus-reportinternet.pdf

[5] An Electric Bus Just Broke the World Record for Distance Traveled on a Single Charge. Earth & Energy https://futurism.com/an-electric-bus-just-broke-the-world-record-for-distance-traveled-on-a-single-charge (2019-08-21)

[6] A. Hoke, A. Brissette, D. Maksimović, A. Pratt and K. Smith. *Electric vehicle charge optimization including effects of lithium-ion battery degradation. 2011 IEEE Vehicle Power and Propulsion Conference*, Chicago, IL, 2011, pp. 1-8. doi: 10.1109/VPPC.2011.6043046

[7] Wood, E., Neubauer, J., Brooker, A.D., Gonder, J., & Smith, K. (2012). *Variability of Battery Wear in Light Duty Plug-In Electric Vehicles Subject to Ambient Temperature, Battery Size, and Consumer Usage: Preprint.*. NREL Report No. CP-5400-53953. http://www.nrel.gov/docs/fy12osti/53953.pdf

[8] Mikko Pihlatie, VTT . *Planning of electric bus systems* . Presentation. Presented on Latin American webinar: Centro Mario Molina Chile & UNEP. 2017. [Online] http://movelatam.org/wp-content/uploads/2017/09/VTT_electric_bus_system_planning.pdf

[9] National Academies of Sciences, Engineering, and Medicine. 2018. *Battery Electric Buses—State of the Practice*. Washington, DC: The National Academies Press. https://doi.org/10.17226/25061.

[10] Rogge, Matthias; Wollny, Sebastian; Sauer, Dirk U. 2015. *Fast Charging Battery Buses for the Electrification of Urban Public Transport—A Feasibility Study Focusing on Charging Infrastructure and Energy Storage Requirements*. Energies 8, no. 5: 4587-4606.

[11] California air resources board. *Electricity Costs for Battery Electric Bus Operation*, Draft disscussion document. [Online] https://ww3.arb.ca.gov/msprog/bus/ratesanddemand.pdf

[12] Kim, Jeongyong; Song, Inho; Choi, Woongchul. 2015. *An Electric Bus with a Battery Exchange System*. Energies 8, no. 7: 6806-6819.

[13] Mirchandani, Pitu ; Madsen, Oli B.G. ; Adler, Jonathan. 2012. *Scheduling and location issues in transforming service fleet vehicles to electric vehicles*. Paper presented at 12th International Conference on Advanced Systems for Public Transport, Santiago, Chile.16 p.

[14] Zhu Chao, Chen Xiaohong, *Optimizing Battery Electric Bus Transit Vehicle Scheduling with Battery Exchanging: Model and Case Study*, Procedia - Social and Behavioral Sciences, Volume 96, 2013, Pages 2725-2736, ISSN 1877-0428, https://doi.org/10.1016/j.sbspro.2013.08.306.

[15] Alexander Kunith, Roman Mendelevitch & Dietmar Goehlich (2017) *Electrification of a city bus network—An optimization model for cost-effective placing of charging infrastructure and battery sizing of fast-charging electric bus systems*, International Journal of Sustainable Transportation, 11:10, 707-720, DOI: 10.1080/15568318.2017.1310962

[16] Leou, R.-C & Hung, J.-J. (2017). *Optimal Charging Schedule Planning and Economic Analysis for Electric Bus Charging Stations*. Energies. 10. 10.3390/en10040483.

[17] Moataz Mohamed, Hany Farag, Nader El-Taweel, Mark Ferguson. *Simulation of electric buses on a full transit network: Operational feasibility and grid impact analysis*. Electric Power Systems Research, Volume 142, 2017, Pages 163-175, ISSN 0378-7796, https://doi.org/10.1016/j.epsr.2016.09.032.

[18] T. Paul and H. Yamada. *Operation and charging scheduling of electric buses in a city bus route network*. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp. 2780-2786. doi: 10.1109/ITSC.2014.6958135

[19] Posthoorn, C. *Vehicle Scheduling of Electric City Buses: A Column Generation Approach.* (Master thesis). 2016. Faculty of Electrical Engineering, Mathematics and Computer Science.Technische Universiteit Delft. http://resolver.tudelft.nl/uuid:0a0b2596-e908-475c-b8d7-48e4b6bbb37d

[20] van Kooten Niekerk, Marcel & Akker, J.M. & Hoogeveen, J.A.. (2017). *Scheduling electric vehicles*. Public Transport. 9. 10.1007/s12469-017-0164-0.

[21] Sassi, Ons & Oulamara, Ammar. (2014). *Electric Vehicle Scheduling and Optimal Charging Problem: Complexity, Exact and Heuristic Approaches*. International Journal of Production Research. 55. 10.1080/00207543.2016.1192695.

[22] Matthias Rogge, Evelien van der Hurk, Allan Larsen, Dirk Uwe Sauer. *Electric bus fleet size and mix problem with optimization of charging infrastructure*. Applied Energy, Volume 211, 2018, Pages 282-295, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2017.11.051.

[23] Carpaneto, G & Dell'Amico, Mauro & Fischetti, Matteo & Toth, Paolo. (1989). *A branch and bound algorithm for the multiple depot vehicle scheduling problem*. Networks. 19. 531 - 548. 10.1002/net.3230190505.

[24] Janáček, J.: *Matematické programování*. EDIS, ŽU-Žilina, 1999, 225 s, ISBN 80-8070-054-0.

[25] Janáček, J., Kohání, M., Szendreyová, A., Buzna, Ľ.: *Diskrétna optimalizácia*. EDIS, Žilina, 2015, 316 s, ISBN 978-80-554-1052-4.

[26] Jaroslav Janacek, Michal Kohani, Matyas Koniorczyk, Peter Marton, *Optimization of periodic crew schedules with application of column generation method*, Transportation Research Part C: Emerging Technologies, Volume 83, 2017, Pages 165-178, ISSN 0968-090X, https://doi.org/10.1016/j.trc.2017.07.008.

[27] Desrosiers J., Lübbecke M.E. (2005) *A Primer in Column Generation*. In: Desaulniers G., Desrosiers J., Solomon M.M. (eds) Column Generation. Springer, Boston, MA. doi: 10.1007/0-387-25486-2_1.

[28] Freling, R., Huisman, D. & Wagelmans, A.P.M. *Journal of Scheduling* (2003) 6:63. https://doi.org/10.1023/A:1022287504028

[29] Gendreau, M. and Potvin, J.Y.: *Handbook of Metaheuristics*, Springer US, 2010, 648 s., ISBN 978-14-419-1665-5

[30] Teoh, Lay Eng & Khoo, Hooi Ling & Yoke Goh, Siew & Mun Chong, Lai. (2017). *Scenario-based Electric Bus Operation: A Case Study of Putrajaya, Malaysia*. International Journal of Transportation Science and Technology. 7. 10.1016/j.ijtst.2017.09.002.

[31] Freling, R. & Paixão, José & Wagelmans, Albert. (1995). *Models and Algorithms for Vehicle Scheduling*. Erasmus University Rotterdam, Econometric Institute, Econometric Institute Report.

[32] Fischetti, Matteo & Lodi, Andrea & Toth, Paolo. (1970). A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem.

[33] Ribeiro, C. and F. Soumis. (1994). *A Column Generation Approach to theMulti-Depot Vehicle Scheduling Problem*. Operations Research 42, 41–52.

[34] Löbel A (1999) *Solving large-scale multiple-depot vehicle scheduling problems*. Wilson NHM, ed. Computer-Aided Transit Scheduling (Springer, Berlin), 193–220.

[35] Ali Haghani, Mohamadreza Banihashemi, *Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints.* Transportation Research Part A: Policy and Practice, Volume 36, Issue 4, 2002, Pages 309-333, ISSN 0965-8564, https://doi.org/10.1016/S0965-8564(01)00004-0.

[36] Haixing Wang, Jinsheng Shen. *Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints.* Applied Mathematics and Computation, Volume 190, Issue 2, 2007, Pages 1237-1249, ISSN 0096-3003, https://doi.org/10.1016/j.amc.2007.02.141.

[37] Anokić, A., Stanimirović, Z., Stakić, Đ. *et al.* Metaheuristic approaches to a vehicle scheduling problem in sugar beet transportation. *Oper Res Int J* (2019). https://doi.org/10.1007/s12351-019-00495-z

[38] Ceder, A. (2007). Public Transit Planning and Operation (1st ed.). CRC Press. https://doi.org/10.1201/b12853

[39] Janáček, J., Janáčková, M., Szendreyová, A., Gábrišová, L., Kohániová, M., & Jánošíková, Ľ. (2010). *Navrhovanie územne rozľahlých obslužných systémov* (1. vyd.). V Žiline: Žilinská univerzita.

[40] Janáček, Jaroslav (2006). *Optimalizace na dopravních sítích*. 2. preprac. vyd. V Žiline, Žilinská univerzita, 248 s., ISBN 80-8070-586-0.

[41] Michael Mutingi, Charles Mbohwa. (2017) *Grouping Genetic Algorithms - Advances and Applications.* Studies in Computational Intelligence 666, Springer, ISBN 978-3-319-44393-5

[42] Schéma liniek mesta Žilina http://www.dpmz.sk/dokumenty/siet_liniek_mhd_zilina.pdf

[43] ABB EV Infrastructure. (2018), *Electric vehicle infrastructure - overnight charging for electric buses and trucks*. [Online]. Available at: https://searchext.abb.com/library/Download.aspx?DocumentID=9AKK107045A5070&-LanguageCode=en&DocumentPartId=&Action=Launch

[44] Siemens AG. (2013), *Electric bus for airports*. [Online]. Available at: https://assets.new.siemens.com/siemens/assets/public.-1494324771.52495109232b1176df25668b35daa19243bf8b58.electricbus-for-airports-en.pdf

[45] Proterra. (2018), *CatalystR : 35 foot busperformance specifications*. [Online]. Available at: https://www.proterra.com/wpcontent/uploads/2018/05/Proterra-Catalyst-35-ft-spec-sheet-Sept-2018.pdf

[46] Palúch, S. (1998) *Simultaneous Bus and Crew Scheduling*. In: Proceedings of International Scientific Conference of Mathematics, University of Žilina, June 30 - July 3 1998

[47] Languang Lu, Xuebing Han, Jianqiu Li, Jianfeng Hua, Minggao Ouyang. (2013) *A review on the key issues for lithium-ion battery management in electric vehicles.* Journal of Power Sources, Volume 226, p. 272-288, ISSN 0378-7753, https://doi.org/10.1016/j.jpowsour.2012.10.060.

[48] An, Kai & Barai, Pallab & Smith, Kandler & Mukherjee, Partha. (2014). *Probing the Thermal Implications in Mechanical Degradation of Lithium-Ion Battery Electrodes.* Journal of the Electrochemical Society. 161. A1058-A1070. 10.1149/2.069406jes.

[49] Millner, Alan. (2010). *Modeling Lithium Ion battery degradation in electric vehicles.* 349 - 356. 10.1109/CITRES.2010.5619782.

[50] Hasan, Md & Pourmousavi Kani, Seyyed Ali & Bai, Feifei & Saha, Tapan. (2017). *The impact of temperature on battery degradation for large-scale BESS in PV plant*. 1-6. 10.1109/AUPEC.2017.8282448.

[51] H. Abidi, K. Hassine and F. Mguis. (2018) *Genetic Algorithm for Solving a Dynamic Vehicle Routing Problem with Time Windows*. 2018 International Conference on High

Performance Computing & Simulation (HPCS), Orleans, France, pp. 782-788, doi: 10.1109/HPCS.2018.00126.

[52] Skok, M., Škrlec, D. & Krajcar, S. (2001) *The Genetic Algorithm Scheduling of Vehicles from Multiple Depots to a Number of Delivery Points*. In: Advances in Fuzzy Systems and Evolutionary Computation. New York, World Scientific nad Engineering Society Press.

[53] Surekha, P., & Sumathi, S. (2011). *Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms.* TI Journals. World Applied Programming.

[54] Tucker A, Crampton J, Swift S. (2005) *RGFGA: an efficient representation and crossover for grouping genetic algorithms*. Evolutionary Computation.13(4):477-499. DOI: 10.1162/106365605774666903.

[55] Fishkin, A.V., Jansen, K. & Mastrolilli, M. (2008) *Grouping Techniques for Scheduling Problems: Simpler and Faster*. Algorithmica 51, p. 183–199. https://doi.org/10.1007/s00453-007-9086-6

[56] Zeng, Yong & Liu, Da-Cheng & Li, Ju-Xuan & Hou, Xiang-Yu. (2013). *A Study of Grouping Heuristics on Vehicle Scheduling Problem Based on Changeable Expenditure Coefficient Model*. TELKOMNIKA Indonesian Journal of Electrical Engineering. 11. 10.11591/telkomnika.v11i1.1913.

[57] Claire Hanen, Zdenek Hanzalek, (2020) *Grouping tasks to save energy in a cyclic scheduling problem: A complexity study*, European Journal of Operational Research, Volume 284, Issue 2, p. 445-459, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2020.01.005.

[58] Y. Hou, Y. Yang, Z. Liu and L. Sun. (2016) *A discontinuous coordinated charging strategy for electric vehicles.* 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), Hefei, China, pp. 1099-1102, doi: 10.1109/ICIEA.2016.7603746.

[59] Houbbadi, Adnane & Trigui, Rochdi & Pelissier, Serge & Redondo-Iglesias, Eduardo & Bouton, Tanguy. (2019). *Optimal Scheduling to Manage an Electric Bus Fleet Overnight Charging.* Energies. 12. 2727. 10.3390/en12142727.

[60] Mohamed, Moataz & Garnett, Ryan & Ferguson, Mark & Kanaroglou, Pavlos. (2016). *Electric Buses: A Review of Alternative Powertrains*. Renewable and Sustainable Energy Reviews. 62. 10.1016/j.rser.2016.05.019.

[61] Sassi, Ons & Ramdane Cherif-Khettaf, Wahiba & Oulamara, Ammar. (2014). *Vehicle Routing Problem with Mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs.* World Academy of Science, Engineering and Technology, Open Science Index 99, International Journal of Mathematical and Computational Sciences, 9(3), 171 - 181.

[62] Constantino, Ademir & Calvi, Rogerio & Mendonça, Candido & De Araujo, Silvio & Landa-Silva, Dario & Flausino dos Santos, Allainclair. (2017). *Solving a Large Real-*

world Bus Driver Scheduling Problem with a Multi-assignment based Heuristic. JOURNAL OF UNIVERSAL COMPUTER SCIENCE. 23. 479-504.

[63] Reuer, Josephine & Kliewer, Natalia & Wolbeck, Lena. (2015). *The Electric Vehicle Scheduling Problem - A study on time-space network based and heuristic solution approaches*. Conference on Advanced Systems in Public Transport(CASPT), Rotterdam, The Netherlands.

[64] Uvaraja, Vikneswary and Lee, Lai Soon (2017) *Metaheuristic approaches for urban transit scheduling problem: a review.* Journal of Advanced Review on Scientific Research, 34 (1). pp. 11-25. ISSN 2289-7887

[65] Olsen, Nils & Kliewer, Natalia & Wolbeck, Lena. (2020). *A study on flow decomposition methods for scheduling of electric buses in public transport based on aggregated time–space network models*. Central European Journal of Operations Research. 10.1007/s10100-020-00705-6.

[66] Ceder, Avishai. (2002). *Urban Transit Scheduling: Framework, Review and Examples. Journal of Urban Planning and Development.* Journal of Urban Planning and Development. 128. 10.1061/(ASCE)0733-9488(2002)128:4(225).

[67] Chen, Xiaopan & Kong, Yunfeng & Dang, Lanxue & Hou, Yane & ye, Xinyue. (2016). *Exact and Metaheuristic Approaches for a Bi-Objective School Bus Scheduling Problem*. PloS one. 11. e0153614. 10.1371/journal.pone.0153614.

[68] Barco Jiménez, John & Guerra, Andres & Muñoz, Luis & Quijano, Nicanor. (2013). *Optimal Routing and Scheduling of Charge for Electric Vehicles: A Case Study*. Mathematical Problems in Engineering. 2017. 10.1155/2017/8509783.

[69] Wen, Min & Linde, E. & Ropke, Stefan & Mirchandani, P. & Larsen, Allan. (2016). *An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem*. Computers & Operations Research. 76. 10.1016/j.cor.2016.06.013.

[70] Vaquerizo, Ma & Baruque, Bruno & Corchado, Emilio. (2012). *Combining Metaheuristic Algorithms to Solve a Scheduling Problem*. 381-391. 10.1007/978-3-642-28931-6_37.

[71] Yao, Enjian & Liu, Tong & Lu, Tianwei & Yang, Yang. (2019). *Optimization of Electric Vehicle Scheduling with Multiple Vehicle Types in Public Transport*. Sustainable Cities and Society. 52. 101862. 10.1016/j.scs.2019.101862.

[72] Zhou, Guangjing & Xie, Dongfan & Zhao, Xiao-Mei & Lu, Chaoru. (2020). *Collaborative Optimization of Vehicle and Charging Scheduling for a Bus Fleet Mixed With Electric and Traditional Buses*. IEEE Access, vol. 8, pp. 8056-8072, doi: 10.1109/ACCESS.2020.2964391.

[73] Miles, John & Potter, Stephen. (2014). Developing a viable electric bus service: The Milton Keynes demonstration project. Research in Transportation Economics. 48. 10.1016/j.retrec.2014.09.063.

[74] Yen, J. (1971). Finding the K Shortest Loopless Paths in a Network. *Management Science, 17*(11), 712-716. https://doi.org/10.1287/mnsc.17.11.712

Vlastné publikácie

[1] Maroš Janovec. *Exaktný prístup k turnusovaniu elektrických autobusov*. In: Využitie kvantitatívnych metód vo vedeckovýskumnej činnosti a v praxi. Zborník príspevkov z medzinárodného vedeckého seminára. 1. vyd. Bratislava : Vydavateľstvo EKONÓM, 2019. s. 74-82. ISBN 978-80-225-4617-1

[2] Maroš Janovec, Michal Kohání (2019). *Exact approach to the electric bus fleet scheduling*. Transportation Research Procedia, Volume 40, Pages 1380-1387, ISSN 2352-1465, https://doi.org/10.1016/j.trpro.2019.07.191.

[3] M. Janovec and M. Kohání (2019). *Battery Degradation impact on the Electric Bus Fleet Scheduling.* International Conference on Information and Digital Technologies (IDT), Zilina, Slovakia, pp. 190-197. doi: https://doi.org/10.1109/DT.2019.8813693

[4] Janovec, M. and Kohani, M. (2020). *Comparison of Continuous and Discontinuous Charging Models for the Electric Bus Scheduling Problem*. In Proceedings of the 9th International Conference on Operations Research and Enterprise Systems - Volume 1: ICORES, ISBN 978-989-758-396-4, pages 179-186. DOI: https://doi.org/10.5220/0008962901790186

[5] Janovec M., Kohani M. (2020) *Data Reduction Algorithm for the Electric Bus Scheduling Problem*. In: Neufeld J.S., Buscher U., Lasch R., Möst D., Schönberger J. (eds) Operations Research Proceedings 2019. Operations Research Proceedings (GOR (Gesellschaft für Operations Research e.V.)). Springer, Cham. https://doi.org/10.1007/978-3-030-48439-2_98