VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

METÓDY OPTIMALIZÁCIE PLÁNOVANIA NÁKLADNEJ PREPRAVY

OPTIMIZATIONS METHODS FOR FREIGHT TRANSPORTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAL GABONAY**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Gabonay Michal, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Metody optimalizace plánování nákladní přepravy**
Optimizations Methods for Freight Transportation
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s problémem plánování nákladní přepravy a nastudujte různé přístupy k jeho řešení.
2. Zpracujte studii na uvedené téma zahrnující stávající techniky řešení.
3. Zvolte vhodný typ úlohy, na který se při řešení zaměříte.
4. Navrhněte vhodné algoritmy pro řešení vybrané úlohy.
5. Navržené algoritmy implementujte a proveďte sady experimentů řešících vybraný problém.
6. Navržený přístup porovnejte s existujícími metodami.
7. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího projektu.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání a demonstrace prototypu algoritmu z bodu 4 zadání.

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

Vedoucí práce: **Bidlo Michal, Ing., Ph.D.**
Konzultant: Rudová Hana, doc. Mgr., Ph.D., FI MUNI
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 25. října 2019

Abstrakt

Táto práca sa zaoberá evolučným návrhom algoritmu pre optimalizáciu plánovania nákladnej prepravy. V dnešnej dobe je dopyt po preprave nákladu stále narastajúci. Správnym plánovaním trasy sa dajú značne znížiť náklady na prepravu, hlavne ak sa jedná o rozsiahly počet obsluhujúcich zákazníkov a spoločnosť s dostatočne veľkou flotilou vozidiel.

V tejto je popísané, ako je daný problém plánovania prepravy definovaný a aké sú jeho špecifikácie a varianty. Ďalej sa práca venuje možnostiam, ako sa dá daný problém riešiť. Následne sú aplikované riešenia optimalizácie pomocou evolučných algoritmov, presnejšie genetického algoritmu a evolučnej stratégie, na konkrétny problém smerovania vozidiel s vyzdvihnutím a doručením zásielok (angl. *Vehicle routing problem with Pickup and Delivery*). Na záver sa vyhodnocujú a analyzujú výsledky sád experimentov, ktorých úlohou je zhodnotiť vhodnosť algoritmov a použitých technik pre daný problém.

Abstract

The following work concerns the study of the evolutionary algorithm, which optimizes freight transport planning. The demand for freight transport is constantly increasing nowadays and with creating, implementing and using proper route planning we are able to significantly reduce transportation costs. However, it is preferably to implement it in companies with large numbers of served customers and with a sufficiently large fleet of vehicles.

The study starts by defining what fright transport planning problem is and by characterizing its existing specifications and variants. My work proceeds to give a background of the possible solutions to the multifaceted aspects of the problem. The specific subproblem I choose to focus on is the Vehicle routing problem with Pickup and Delivery for which I apply the optimization solution. In the main body of my thesis, I will elaborate on the chosen optimization solution which encompasses the genetic algorithm and evolutionary strategy. The aim of the study is to measure the suitability of the algorithms and techniques used, for which reason the final part of my work will deal with the analysis and evaluation of the experiments.

Kľúčové slová

problém smerovania vozidiel, preprava nákladu, optimalizácia, genetický algoritmus, evolučná stratégia

Keywords

Vehicle Routing Problem, Pickup & Delivery, Optimalization, Genetic Algorithm, Evolution strategy

Citácia

GABONAY, Michal. *Metódy optimalizácie plánovania nákladnej prepravy.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Metódy optimalizácie plánovania nákladnej prepravy

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Michala Bidla Ph.D. a s použitím odbornej literatúry. Ďalšie informácie mi poskytla doc. Mgr. Hana Rudová Ph.D., ako externá konzultantka z Masarykovej Univerzity. Uviedol som všetky literárne pramene, publikácie a zdroje, z ktorých som čerpal.

.....
Michal Gabonay
3. júna 2020

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu diplomovej práce, pánovi Ing. Michalovi Bidlovi Ph.D, a externej konzultantke doc. Mgr Hane Rudovej, Ph.D. za ich pomoc a usmerňovanie. Taktiež sa chcem poďakovať za podporu svojej rodine a blízkym počas celého štúdia. Táto práca bola podporená Ministerstvom školstva, mládeže a telovýchovy z podpory Veľkých infraštruktúr pre výskum, experimentálny vývoj a inovácie v rámci projektu "IT4Innovations národní superpočítačové centrum - LM2015070".

Obsah

Kapitola 1

Úvod

V dnešnej dobe existuje stále sa zvyšujúci dopyt po preprave nákladu rôzneho druhu. Riadenie dodávateľského reťazca (angl. *supply chain managment*) sa kvôli tomuto faktu stáva dôležitejšou úlohou čoraz viac, deň za dňom. Kvalitné riadenie dodávateľského reťazca predstavuje veľkú konkurenčnú výhodu. Jednou z kľúčových častí riadenie dodávateľského reťazca je práve preprava materiálov k výrobe a preprava výsledného tovaru k zákazníkom za čo najnižšiu cenu. Vedľajšou výhodou efektívneho riadenia prepravy, môže byť tiež zníženie záťaže na životné prostredie. To je vďaka tomu, že vozidlá prepravujú tovar viac efektívne, čiže je potreba menej áut a autá prejdú menšiu vzdialenosť na to, aby rozviezli rovnaké množstvo tovaru. Riešenie na úlohy ako sú tieto, sú obsiahnuté v probléme smerovania vozidiel (angl. *Vehicle routing problem*). Tieto problémy sú najčastejším problémom logistického sektora kvôli vysokej cene prepravy, ktorú sa snažia minimalizovať. Prepravné a výrobné spoločnosti často musia riešiť takéto komplexné problémy plánovania prepravy, pričom sa často jedná o typ úloh, ku ktorým neexistujú optimálne algoritmy na riešenie. Riešenia s uspokojivými výsledkami sú obvykle získavané použitím vhodných heuristík (algoritmov "šitých na mieru" pre dané úlohy), často sú však získané riešenia nie príliš kvalitné a tiež návrh heuristík predstavuje komplexný inžiniersky problém. A preto sa v dnešnej dobe stále viac pristupuje k použitiu nekonvenčných postupov, medzi ktoré patria napríklad evolučné algoritmy. Jedná sa o tzv. metaheuristiky, ktorých konfiguráciu je ľahké prispôsobovať rôznym typom úloh (napr. reprezentácia problému, ohodnotenie riešenia) a tým pádom značne zjednodušiť návrh optimalizačnej techniky. Ďalej bolo ukázané (napr. v [3, 19]), že pomocou evolučných techník je možné získať riešenia, ktoré sú mimo dosah konvenčných metód. V rámci výskumu pokročilých techník optimalizácie nákladnej prepravy budú použité evolučné algoritmy aj v tejto práci.

Cieľom tejto práce je návrh a overenie metódy pre optimalizáciu riadenie nákladnej dopravy. Ako už bolo skôr spomínané, zvolené techniky na riešenie problému sú evolučné algoritmy, konkrétne genetický algoritmus a evolučná stratégia, v kombinácii s návrhom vhodných operátorov pre generovanie nových riešení. Dané techniky sú aplikované na riešenie úlohy problému smerovania vozidiel s vyzdvihnutím a doručením (angl. *Vehicle Routing Problem with Pick-Up and Delivery*).

Text tejto práce je rozdelený do niekoľkých častí. V kapitole 2 je podrobne popísaný riešený problém, jeho aspekty, definícia a jeho rôzne variácie. Táto kapitola taktiež obsahuje možnosti jeho riešenia, podložené štúdiami zaoberajúcimi sa danou problematikou. Tretia kapitola 3 zahŕňa teoretický prehľad evolučných algoritmov, ktorých technika bola použitá na riešenie tejto práce. Navrhnuté a použité mutačné operátory sú opísané v kapitole 4. Štyri varianty vlastného návrhu evolučného prístupu k riešeniu problému smerovania vozidiel,

ktorý bol podstatou tejto práce, je predstavený v kapitole 5. Kapitola 6 obsahuje dosiahnuté výsledky experimentov. Na záver v poslednej kapitole 7 sú zhodnotené výsledky, diskusia k danému riešeniu a tiež uvedené možné rozšírenia do budúcna.

Kapitola 2

Problém smerovania vozidiel

Problém smerovania vozidiel (angl. *Vehicle Routing Problem* - VRP) bol predstavený už pred viac než 60 rokmi, v štúdii z roku 1959 od Dantziga a Ramsera [9] (vtedy ešte pod anglickým názvom *Truck Dispatching Problem*). V tejto publikácií bol problém popisovaný pre aplikovanie do praxe na prepravu paliva medzi benzínovými stanicami. Od tej doby bola definícia viackrát upravovaná a upresňovaná a bolo predstavených množstvo algoritmov pre riešenie daného problému. VRP je dôležitým problémom v oblasti dopravy, distribúcie a logistiky. V tejto kapitole je presne popísaná definícia problému, zadefinované rôzne variácie a následne opísané možnosti, ktoré boli predstavené ako vhodné riešenia danej problematiky. Na záver sú taktiež predstavené dva open-source projekty, ktoré môžu slúžiť aj na riešenie VRP a ktoré sa využívajú v praxi.

2.1 Definícia problému

Problém smerovania vozidiel (VRP) je kombinatorický problém, ktorého riešenie je reprezentované diskrétnymi hodnotami. VRP v základnej forme je pokladaný za zovšeobecnený problém obchodného cestujúceho (angl. *Traveling Salesman Problem* - TSP). Táto skutočnosť mu dovoľuje rozšírenia do zložitejších podôb. VRP rovnako ako TSP je klasifikovaný ako NP-úplný, čiže so vzrastajúcim počtom požiadaviek na prepravu (miest) exponenciálne narastá počet potencionálnych riešení a teda aj čas potrebný pre nájdenie riešenia [21]. Obecný VRP na vstupe príma množinu požiadaviek na prepravu, čiže miesta ktoré je treba navštíviť a flotilu (množinu) vozidiel, ktoré sú k dispozícii pre rozvoz z jedného alebo viacerých diep. Úlohou je nájdenie optimálnej množiny trás s čo najnižšou cenou, za použitia čo najnižšieho počtu vozidiel s tým, že riešenie bude spĺňať všetky potrebné vstupné podmienky (niektoré z najčastejších sú popísané v podkapitole 2.2). Inak povedané, úlohou je priradiť každému vozidlu požiadavky na prepravu ktoré má obslúžiť, a určiť poradia v akom má vozidlo navštíviť všetky miesta doručenia, jeho priradených požiadavok na prepravu. To musí byť vykonané tak, aby neboli porušené žiadne zo vstupných obmedzení a pravidiel (ak konkrétna úloha nejaké má). Rôzne variácie VRP majú rôzne obmedzenia a pravidlá (viac o rozšíreniach VRP v kapitole 2.2). Grafické znázornenie základného princípu VRP je zobrazené na obrázku 2.1.

Klasický VRP je definované takto [14]:
Nech $G = (V, A)$ je graf, kde $V = \{1, ..., n\}$ je množina vrcholov reprezentujúcich miesta s depom, umiestneným vo vrchole 1, a A je množina hrán tvaru (i, j), kde $i, j \in V$. S každou hranou $(i, j), i \neq j$ je pridružená nezáporná hodnota c_{ij} z matice vzdialenosti C, pričom

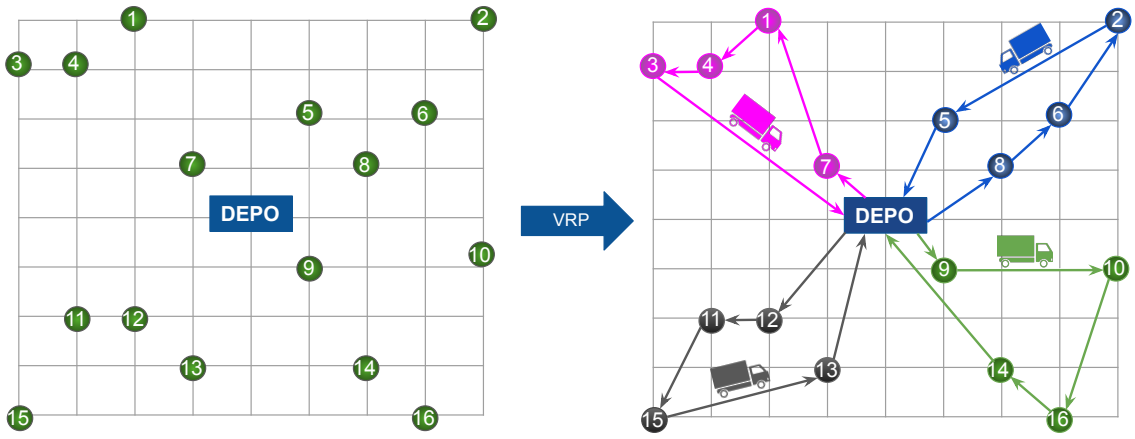
c_{ij} vyjadruje vzdialenosť medzi miestami i, j. V určitých kontextoch môže byť c_{ij} vyjadrené ako cena trasy alebo čas potrebný na prejdenie trasy. Ďalej sa predpokladá, že je k dispozícii m vozidiel umiestnených v depe. Riešenie VRP spočíva v nájdení množiny najlacnejších trás vozidiel takým spôsobom, aby:

1. Každé miesto z množiny $V\backslash\{1\}$ je navštívené práve jeden krát práve jedným vozidlom.
2. Každé vozidlo začína aj končí v depe.
3. Ďalšie dodatočné podmienky sú splnené.

Medzi najčastejšie dodatočné podmienka spadá:

1. *Obmedzená kapacita vozidiel (CVRP)*: nezáporná váha (alebo požiadavok) d_i je priradený pre každé miesto $i > 1$. Pre žiadnu trasu nesmie suma týchto, váh prekročiť kapacitu vozidla.
2. Počet miest v každej trase je zhora obmedzený hodnotou q.
3. *Obmedzenie maximálnej dĺžky (DVRP)*: dĺžka žiadnej trasy nesmie prekročiť určenú hranicu L.
4. *Časové okná (VRPTW)*: miesto i musí byť navštívené v časovom intervale $[a_i, b_i]$, pričom čakanie v meste i je dovolené.
5. *Vzťah medzi dvojicou miest*: miesto i musí byť navštívené pred mestom j.

Dodatočné podmienky sú pridávané s rozšíreniami VRP. Podmienky ktoré sú zahrnuté v úlohe riešenej v tejto práci sa bližšie venuje nasledujúca podkapitola 2.2.



Obr. 2.1: Problém smerovania vozidiel - grafické znázornenie. V ľavej časti sú rozmiestnené požiadavky na prepravu, a v pravej časti sú výsledné, optimalizované trasy pre danú úlohu.

2.2 Rozšírenia VRP

Základný VRP berie do úvahy rôzne predpoklady, podľa ktorých sa riadi, ako napríklad rovnaké vozidlá vo flotile s neobmedzenou kapacitou, jedno depo z ktorého vychádzajú všetky vozidlá, jedna cesta pre jedno vozidlo atď. Tieto predpoklady môžu byť upravené alebo odstránené pridaním rôznych pravidiel a obmedzení, čím sa základný VRP rozšíri do rôznych typov úlohy. V reálnom svete je často konkrétny problém kombináciou jednotlivých rozšírení a rôznych ďalších obmedzení prepravy podľa konkrétnych potrieb zadania. V tejto podkapitole sú popísané tie rozšírenia VRP, ktoré sú obsiahnuté v pri riešení v tejto práci.

2.2.1 VRP s maximálnou dĺžkou

Rozšírenie nazývané problém smerovania vozidiel s maximálnou dĺžkou (angl. *Distance-Constrained Vehicle routing problem* - DVRP [21]), zavádza obmedzenie, že žiadna trasa nemôže prekročiť určenú maximálnu dĺžku (alebo čas) L. Každá hrana $(i, j) \in A$ má priradenú nezápornú hodnotu l_{ij} (l_{ij} sa môže ale nemusí rovnať cene hrany c_{ij}). Suma týchto dĺžok hrán v trase musí byť menšia alebo rovná maximálnej dĺžke trasy L. Toto obmedzenie maximálnej dĺžky trasy môže byť rôzne pre každé vozidlo. V takom prípade sa maximálna dĺžka určuje ako $L_k, k = 1, ..., m$, kde m je počet vozidiel [21].

2.2.2 VRP s kapacitou vozidiel

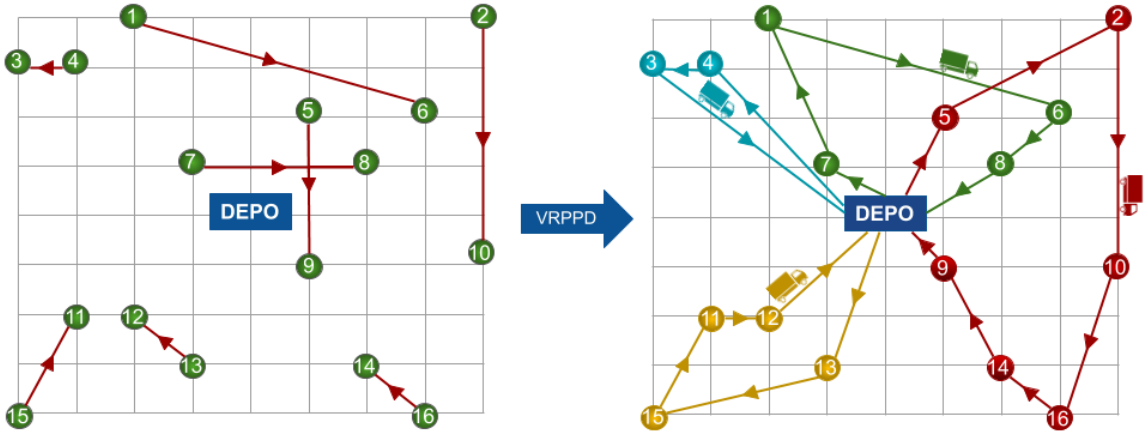
Problém smerovania vozidiel s kapacitou (angl. *Capacitated Vehicle Routing Problem* - CVRP) je rozšírenie, kde vozidlá majú len určitú konečnú kapacitu, ktorú dokážu uniesť. Formulácia ostáva rovnaká ako u základného VRP, ale s pridanou podmienkou, že pre každé vozidlo k vo flotile o veľkosti m vozidiel, nepresiahne celková suma prevážaného tovaru všetkých n zákazníkov na trase R_k, kapacitu vozidla: $\sum_{i=1}^{n} d_i \leq Q_k$, kde Q_k je kapacita vozidla k a d_i je množstvo prepravovaného tovaru pre i-tého zákazníka na trase R_k (obvykle hmotnosť alebo objem tovaru). Toto rozšírenie CVRP je väčšinou brané ako štandard a málo kedy sa rieši úloha smerovania vozidiel, bez toho aby sa brala do úvahy kapacita vozidiel. Špeciálna varianta CVRP môže pridávať opačnú podmienku pre vozidlo na trase, čiže vozidlá musia previesť minimálne určité množstvo nákladu K_{min} [21].

2.2.3 VRP s vyzdvihnutím a doručením

Jednou z dôležitých variant VRP je problém smerovania vozidiel s vyzdvihnutím a doručením (angl. *Vehicle Routing Problem with Pick-Up a Delivery* - VRPPD). VRPPD sa líši od základného VRP hlavne tým, že požiadavok na prepravu od zákazníka je rozdelený na dvojicu miest, vyzdvihnutie zásielky (*pickup*) a doručenie (*delivery*). Aby bolo riešenie problému platné, požiadavka na prepravu (vyzdvihnutie a doručenie) musí byť prevezená rovnakým vozidlom a miesto vyzdvihnutia nákladu musí byť v plánovanej trase vždy pred miestom doručenia nákladu. A keďže v tejto variante sa počas trasy tovar aj nakladá a nie len vykladá ako u základného VRP, je tiež nutné dbať na to, aby kapacita vozidla nebola presiahnutá v žiadnom momente plánovanej trasy rozvozu [21].

Jednoduchý príklad VRPPD je znázornený na obrázku 2.2. V tomto príklade je úlohou obslúžiť osem požiadaviek na prepravu, kde každý požiadavok je definovaný dvoma miestami (zelený bod s číslom) a kde smer šípky znázorňuje to, odkiaľ sa má náklad vyzdvihnúť a kam sa má náklad doručiť. Všetky vozidlá začínajú aj končia v jednom mieste (depo). Riešením tejto úlohy sú 4 trasy reprezentované poradím navštívených miest:

- Trasa 1 (modré vozidlo): DEPO → 4 → 3 → DEPO
- Trasa 2 (zelené vozidlo): DEPO → 7 → 1 → 6 → 8 → DEPO
- Trasa 3 (červené vozidlo): DEPO → 5 → 2 → 10 → 16 → 14 → 9 → DEPO
- Trasa 4 (žlté vozidlo): DEPO → 13 → 15 → 11 → 12 → DEPO



Obr. 2.2: Príklad problému smerovania vozidiel s vyzdvihnutím a doručením. V ľavej časti obrázku vidno prepojenie miest červenou šípkou, čo znázorňuje miesto vyzdvihnutia a miesto doručenia z požiadavky na prepravu.

2.3 Súčasný stav a existujúce riešenia

Problémy smerovania vozidiel a problém obchodného cestujúceho sú prakticky už od dôb ich zavedenia rozsiahlo študované a rozoberané v literatúrach zaoberajúcich sa optimalizáciou a to hlavne kvôli rozsiahlym možnostiam aplikovania do praxe. Podrobne o technikách riešenia VRP pojednáva napríklad [23].

2.3.1 Presné metódy

Presné metódy sú založené na prístupe skúšania všetkých možností. Garantujú nájdenie optimálneho riešenia, ale negarantujú nájdenie riešenia v prijateľnom čase, hlavne pre veľké problémy. VRP je NP-úplný problém, čiže tieto metódy niesu úplne vhodné pre riešenie tohto typu problému, hlavne ak je v danom probléme množstvo zákazníkov. Algoritmy presnej metódy na riešenie VRP sú založené na prechádzaní stavového priestoru všetkých riešení pre úlohu. Priechod je realizovaný na stromovej štruktúre riešení s tým, že ak je v nejakom uzle stromu neplatné/nevhodné riešenie, celá nadväzujúca vetva tohto uzlu je neplatná/nevhodná. Najčastejšie využívané algoritmy sú metóda vetvenia a hraníc (angl.

Branch and Bound), metóda vetvenia a rezov (angl. *Branch and Cut*) a metóda vetvenia a ceny (angl. *Branch and Price*). Napríklad autori v [20] navrhujú prístup presných metód riešenia pre CVRP pomocou metódy vetvenia a hraníc, kde dosahujú výnimočne nie až tak zlé výsledky pre menší rozsah problému (cca 10-50) avšak za cenu väčšieho výpočtového času. Prekvapivo dobré výsledky sú však dosahované v práci [4], kde v prijateľnom čase dokážu nájsť optimálne riešenie pre problém VRPPDTW aj pre niektoré inštancie až do stoviek miest (nie všetky a nie vždy).

2.3.2 Heuristiky

Heuristické metódy vykonávajú relatívne obmedzené skúmanie stavového priestoru a zvyčajne poskytujú kvalitné riešenia v prijateľných výpočtových časoch. To nasvedčuje aj tomu, že je množstvo dobre navrhnutých a predstavených spôsobov. Často však slúžia len ako doplnok k inej metóde riešenia, zväčša nejakej metaheuristike, obvykle pre vytvorenie inicializačného riešenia. Niektoré zaujímavé heuristiky vyberiem a predstavím.

Jedna z osvedčených heuristík pre riešenie VRP je prístup "Najprv zhluk, potom trasa" (angl. *cluster-first, route-second*), ktorá bola predstavená v [10]. Ako prvé sa umiestnia body snažiac sa čo najviac priblížiť ťažisku zhlukovaných miest. Ďalej sa okolo každého takto vytvoreného bodu vytvorí zhluk tak, aby sa minimalizovala suma vzdialeností miest k jednotlivým bodom. Následne sa vyhodnotí trasa pre každý zhluk vyriešením TSP.

V ďalšej práci [16] autor predstavil heuristiku ktorý vytvára trasu pomocou adaptívnej paralelnej schémy. Na základe experimentov práca vyhodnotila danú heuristiku ako prínosnú pre vytvorenie inicializačného riešenie pre mnohé metaheuristiky, ktorým sľubuje výrazné zlepšenie a konvergenciu.

Autori v práci [8] navrhli heuritiku, v ktorej na začiatku vytvárajú trasy $(0, i, 0)$, kde $(i = 1, ..., n), n$ je počet miest v úlohe. Následne vytvorené trasy postupne spájajú za účelom čo najväčších úspor na cene spojenej trasy. Spôsob spájania dvoch trás vyzerá tak, že sa trasa $(0, ..., i, 0)$ a trasa $(0, j, ..., 0)$ spojí do jednej trasy $(0, ..., i, j, ..., 0)$, pričom výsledná úspora zo spojenia je $s_{ij} = c_{i0} + c_{0j} - c_{ij}$.

2.3.3 Metaheuristiky

Metaheuristiky sa pre riešenie VRP osvedčili ako rozumná voľba pre riešenie VRP. Mnoho z najúspešnejších metaheuristík pre veľké inštancie VRP sú založené na nejakej forme paralelných výpočtov. VRP je možné riešiť mnohými typmi metaheuristík, tu je pár najzaujímavejších príkladov:

- **Mravčie algoritmy (angl. *Ant Algorithms*)** - prvý krát bolo predstavené riešenie pre CVRP pomocou mravčích algoritmov už v roku 1999 [7]. V nasledujúcich rokoch vzniklo mnoho rozšírení a vylepšení aj pre komplikovanejšie typy VRP, napríklad [5], kde naviac použili viacero mravčích kolónií pre dosiahnutie čo najlepších výsledkov. Základ algoritmu je v dvoch jednoduchých krokoch, vytvorenie trás pre vozidlá a aktualizácia trás pre vozidlá.

- **Simulované žíhanie (angl. *Simulated Annealing*)** - V článku [1] je predstavená metóda riešenia VRPTW pomocou paralelizovateľného systému založeného na metaheuristike nazývanej simulované žíhania. Systém sa skladá z dvoch optimalizačných fáz, globálnej a lokálnej. Tie bežia súbežne s tým, že si medzi sebou vymieňajú informácie, čo vedie k lepším výsledkom.

- **Tabu prehľadávanie (angl. *Tabu Search* - TS)** - TS sa ukázala ako veľmi efektívna metóda na riešenie VRP. Často za využíva v kombinácií s nejakou inou metaheuristikou, napríklad s genetickými algoritmami alebo s inými heuristikami (tzv. hybridné metódy). Napríklad v článku [15] predstavili metódu na riešenie m-VRPTW[1] pomocou TS založenom na zozname neobslúžených zákazníkov a mechanizme vyžadujúcom plné využitie vozidiel. V ich práci umožňujú porušovanie časových okien, ale za určitú penalizáciu. Ďalej napríklad P. Toth a D. Vigo navrhli vo svojej práci [22] použitie *Granular Tabu Search* (GTS) pre riešenie VRP. Hlavná myšlienka, ktorá stojí za GTS, vyplýva z pozorovania, že dlhšie hrany grafu majú menšiu pravdepodobnosť, že patria do optimálneho riešenia.

- **Genetické algoritmy (angl. *Genetic Algorithms* - GA)** - Na riešenie VRP pomocou GA je obvyklé reprezentovať každého jednotlivca iba jedným chromozómom, ktorý je reťazcom celých čísel, pričom každý z nich predstavuje zákazníka v poradí v akom sa majú obslúžiť. Jednotlivé trasy v chromozóme sú oddelené od seba pomocou identifikátora depa alebo identifikátora vozidla.

  S aplikovaním GA na VRP sa začalo krátko po prvých úspešných riešeniach VRP pomocou metaheuristík (simulované žíhanie a tabu prehľadávanie), avšak výsledky boli veľmi zmiešané a podpriemerné až do práce [19], kde sa autorovi podarilo dosiahnuť podobných výsledkov ako v tej dobe dosahovali iné algoritmy. Od tej doby sa vývoj riešení VRP pomocou GA stále vylepšoval a GA sa stali populárnou metódou na riešenie VRP, často doprevádzané s pomocou iných metaheuristík. Táto kombinácia sa ukázala ako veľká výhoda algoritmov, čo napríklad dokazuje práca [3], ktorá jasne ukázala, že GA rozšírené o prehľadávanie susedov výrazne zlepšilo výsledky. GA sa ukázali ako použiteľné aj na VRP variantu s vyzdvihnutím a doručením, napríklad v článku [11] autori predviedli funkčný evolučný prístup na riešenie VRPPD s časovými oknami.

2.3.4 Open-source nástroje zamerané na riešenie VRP

Za zmienku tiež stoja open-source hotové projekty pre riešenie VRP, ktoré je možné prispôsobiť svojim potrebám a použiť v praxi na reálnych problémoch. Najrozšírenejšie a najlepšie spracované sú **VROOM**[2] a **OR-Tools**[3]. VROOM je optimalizačný nástroj zameraný na VRP, napísaný v C++. Je schopný riešiť mnohé variácie VRP v rozumnom čase, funguje na princípe klient-server rozhrania. Ďalším nástrojom je OR-Tools vyvíjaný Googlom. Jedná sa o open-source software pre riešenie širokej škály optimalizačných úloh, medzi ktoré patrí aj VRP. Podporuje viaceré možnosti pridávania podmienok a obmedzení pre riešenie VRP. Funguje tak, že ako prvé sa namodeluje problém, ktorý sa má riešiť. Na to je možné použiť viacero programovacích jazykov ako C++, Python, Java alebo C#. Následne po namodelovaní problému je možné vybrať z pomedzí viacero riešiteľov ako napríklad komerčný CPLEX[4], alebo tiež open-source SCIP[5] alebo CP-SAT[6]. Modelovanie a riešenie sú samostatne oddelené časti, a je dokonca možné využiť aj vlastný riešiteľ.

[1] Problém smerovania vozidiel s časovými oknami a obmedzeným počtom vozidiel - niektorý zákazníci musia byť vylúčený z riešenie - neobslúžený

[2] https://github.com/VROOM-Project/vroom

[3] https://developers.google.com/optimization

[4] https://www.ibm.com/analytics/cplex-optimizer

[5] https://scip.zib.de/

[6] https://developers.google.com/optimization/cp/cp_solver

2.4 Zameranie tejto práce

Táto práca sa zameriava na riešenie optimalizácie variácie VRPPD s obmedzením na kapacitu a počet vozidiel a s obmedzením na maximálnu dĺžku trasy. Všetky vozidlá začínajú aj končia svoje trasy v jednom bode (depo). Úloha na vstup berie množinu požiadaviek na prepravu, kde každý takýto požiadavok je zadaný dvojicou miest, pričom jedno z miest je miesto kde sa náklad vyzdvihuje a druhé je miesto kde sa náklad doručuje. Ďalej má každý požiadavok informáciu o množstve prepravovaného tovaru. Druhým vstupom je flotila homogénnych vozidiel (rovnaké vlastnosti a kapacity), určená počtom vozidiel, ktoré sú k dispozícii na prepravu, kapacitou týchto vozidiel, miestom odkiaľ vozidlá začínajú (depo) a maximálnou dĺžkou, ktorú môže vozidlo prejsť. Cieľom práce je vytvoriť program, za pomoci evolučných algoritmov, taký čo z vyššie spomínaných vstupov nájde čo najlepšie riešenie pre danú úlohu. Riešením úlohy sa myslí navrhnuté platné trasy vozidiel obsluhujúc všetky požiadavky na prepravy zo vstupu.

Kapitola 3

Evolučné algoritmy

V 19. storočí vedci ako Mendel alebo Darwin priniesli myšlienku dedičnosti vlastností z rodičov na potomkov v prírode [2]. Práve týmito myšlienkami a teóriami vzniku nových druhov a evolúciou sa v 20. storočí nechali inšpirovať mnohí vedci z oblasti modernej počítačovej vedy pri návrhu evolučných algoritmov. Na rôznych školách a inštitúciach sa vyvíjali a študovali nezávisle na sebe rôzne typy evolučných algoritmov, ako evolučné stratégie, evolučné programovanie, genetické programovanie a genetické algoritmy. Každý z týchto typov zvolil odlišný prístup, ale všetky vychádzajú a sú inšpirované na princípoch evolúcie z prírody [2].

Evolučné algoritmy sa radia do kategórie metaheuristických optimalizačných algoritmoch, založených na populácii. Základ pre všetky evolučné algoritmy je postavený na podobných princípoch opierajúcich sa o simulovaní evolúcie jedincov v populácii, nazývanej tiež populáciou kandidátnych riešení. Každý z jedincov v populácii predstavuje zakódovanú verziu možného riešenia problému. Každému jedincovi je priradená tzv. fitness hodnota daná vyhodnotením určitej účelovej funkcie alebo tiež fitness funkcie. Táto fitness hodnota určuje vhodnosť jedinca (riešenia) pre konkrétny zadaný problém. V každej iterácii (generácii) prebieha výber jedincov, ktorí sa stanú rodičmi pre novú generáciu. Jedinci s lepšou fitness hodnotou majú väčšiu pravdepodobnosť, že budú vybraní. Následne prebieha reprodukcia vybraných jedincov pomocou rôznych variačných (genetických) operácií (napríklad kríženie alebo mutácia), čím sa vygenerujú nový potomkovia. Nakoniec prebieha nahradzovanie jedincov, čiže tvorba novej populácie, kde sa rozhoduje kto z rodičov a potomkov prežije a postúpi do novej generácie. Tieto kroky sa opakujú až kým nie je splnená podmienka ukončenia, čo môže byť napríklad vopred zadaný maximálny počet generácií, nájdenie dostatočne vyhovujúceho riešenia (hranica fitness hodnoty) alebo sa najlepšie nájdené riešenie nezlepšilo po daný počet iterácii. Štruktúru evolučného algoritmu popisuje algoritmus 1.

Algoritmus 1: Základná štruktúra evolučného algoritmu

Result: najlepšie nájdené riešenie
čas $t = 0$;
inicializácia počiatočnú populácie $P(t)$;
vypočítaj vhodnosť každého jedinca v $P(t)$;
repeat
  $t = t + 1$;
  vytvor novú populáciu $P(t)$ z predošlej populácie $P(t-1)$;
  vypočítaj vhodnosť jedincov novej populácie $P(t)$;
until splnená podmienka ukončenia;

Do rodiny evolučných algoritmov spadá viacero typov algoritmov, štyri najznámejšie a najpoužívanejšie sú evolučné stratégie, evolučné programovanie, genetické programovanie a genetické algoritmy. Táto kapitola sa podrobne zaoberá dvoma z týchto typov evolučných algoritmov, a to evolučnými stratégiami a genetickými algoritmami, keďže tieto dva prístupy boli vybrané, použité a odskúšané pri návrhu algoritmu pre optimalizáciu nákladnej prepravy. Informácie zhrnuté v kapitole sú čerpané hlavne z kníh [13, 6], ktoré sa do hĺbky zaoberajú evolučným algoritmom.

3.1 Genetické algoritmy

Genetické algoritmy (GA) publikoval v roku 1975 profesor J. Holland [12] ako štúdiu schopností adaptačných procesov u zložitých systémoch v prírode. GA simulujú evolúciu objektov schopných prispôsobovať sa podmienkam svojho okolia. V súčasnosti sú GA jedny z najpopulárnejších evolučných algoritmov. Silne zakladá na odlišnosti genotypu a fenotypu. Genotyp reprezentuje zakódované riešenie a fenotyp predstavuje objekt tohoto riešenia. To znamená, že fenotyp sa získa dekódovaním genotypu. Genetické operácie prebiehajú na úrovni genotypu zatiaľ čo vyhodnocovanie fitness funkcie jedincov je vykonávané nad fenotypom.

Začiatočným bodom pre GA je reprezentácia riešení, čo sa často označuje aj ako chromozóm alebo jedinec. Samotný chromozóm pozostáva z génov, ktorých poradie a hodnoty kódujú kandidátne riešenie. Hlavné rysy GA vychádzajú zo základných princípov evolučných algoritmov. Populácia sa zvyčajne skladá z desiatok až stoviek jedincov, kde každý jedinec predstavuje kandidátne riešenie. Evolučný cyklus prebieha tak, že sa po (typicky náhodnom) vygenerovaní počiatočnej populácie jedincov vykonáva ohodnotenie jedincov, selekcia rodičov, tvorba potomkov a nahradenie predchádzajúcej populácie, pričom sa tieto kroky opakujú, kým nie je splnené ukončujúce kritérium.

GA je stochastický algoritmus, náhodnosť hrá základnú rolu v algoritme. Selekcia rovnako ako reprodukcia by sa bez náhodnosti nezaobišli. To, že GA je založený na práci s populáciou, čiže sa v každej iterácii v pamäti uchováva niekoľko možných riešení problému, prináša hneď niekoľko výhod. Algoritmus môže kombinovať rôzne riešenia, čím môže využívať dobré častí jednotlivých riešení a spojiť ich do ešte lepšieho riešenia. Ďalšou výhodou je, že tento typ algoritmu je vhodný na paralelizáciu. Robustnosť[1] GA tiež prispela k jeho popularite a úspešnosti. Vďaka týmto výhodám sa z GA stal silný nastroj na riešenie optimalizačných úloh [18].

3.1.1 Genetické operátory

Genetické algoritmy obsahujú tri základné genetické operátori, ktorými sú selekcia, kríženie a mutácia.

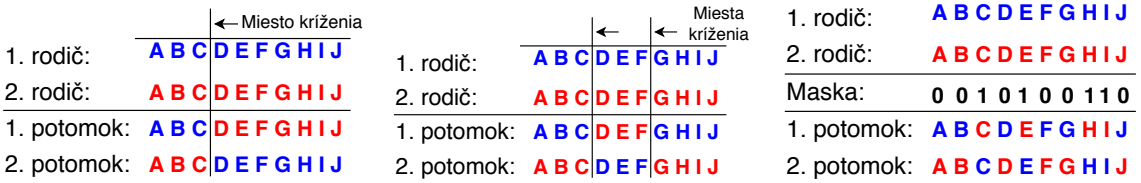
Selekcia alebo výber rodičov, predstavuje postup ktorým sú z aktuálnej populácie vyberaný jedinci k tvorbe novej populácie. Výber spočíva v určení pravdepodobnosti výberu jedincov za rodičov na základe hodnôt fitness funkcie. Pravdepodobnosť sa určuje buď úmerne k fitness (angl. *proportional assignment*) alebo podľa poradí (angl. *ranked-based assignment*). Samotný výber jedincov je následne implementovaný pomocou techník ako vážená ruleta, stochastické univerzálne vzorkovanie, turnaj a iné. Techniky selekcie sú často

[1] Robustnosťou GA sa myslí schopnosť aplikovania na širokú škálu typov problémov. Samotná podstata GA nie je viazaná na žiadne počiatočné podmienky problému, pred použitím samotného GA [18].

doplnené o tzv. **elitizmus**, u ktorého sa vyberie niekoľko najlepších jedincov, ktorý vždy prežijú do ďalšej generácie [18]. Najbežnejšie techniky selekcie sú nasledujúce. **Ruleta** u ktorej veľkosti výsekov kola sú úmerne prepočítané podľa hodnôt fitness každého jedinca. Ruleta sa roztočí a na kom sa zastaví je vybraný ako rodič. Koleso rulety sa roztočí toľko krát koľko je potrebné vybrať rodičov. **Turnaj** spočíva v tom, že je z populácie vybraný náhodne určitý počet jedincov do turnaja, kde medzi sebou súperia. Najlepšie ohodnotený jedinec so skupiny je vybraný ako rodič. Proces sa opakuje kým nie je vybraný potrebný počet rodičov.

Kríženie (angl. *crossover/recombination*) je spôsob tvorby nových jedincov z informácií obdržaných z génu rodičov. Kríženie spočíva vo vzájomnej výmene časti chromozómov medzi rodičmi. Konkrétne metódy kríženia závisia na použitej reprezentácii chromozómov. Parameter pravdepodobnosti kríženia P_C, určuje s akou pravdepodobnosť kríženie prebehne, ak kríženie neprebehne, potomkovia sú totožní ako rodičia. Základné operátory kríženia sú:

- **Jednobodové kríženie** - Určí sa náhodne miesto kríženia - pre n génov v chromozóme hodnota $1..n-1$ a časti chromozómov rodičov za na tomto mieste zmenia, ako ukazuje obr. 3.1a.
- **Dvojbodové kríženie** - Určia sa náhodne dva rôzne miesta kríženia - pre n génov v chromozóme hodnota $1..n-1$ a časti chromozómov rodičov za na týchto miestach zmenia, ako ukazuje obr. 3.1b.
- **Uniformné kríženie** - Každý gén sa prenesie s pravdepodobnosťou p (obvykle 0.5) medzi rodičmi. Vygenerovaná maska určuje to, z ktorého rodiča sa gény presúvajú, ako ukazuje obr. 3.1c.

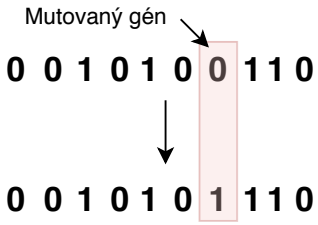


Obr. 3.1: (a) jednobodové kríženie, (b) dvojbodové kríženie, (c) uniformné kríženie

Mutácie sú veľmi dôležitým genetickým operátorom, pri ktorom dochádza s určitou pravdepodobnosťou (obvykle malou) k umelej zmene hodnoty náhodne vybraného génu (génov) u jedinca. Mutácie ako jediné z genetických operátorov majú schopnosť prinášať nové vlastnosti jedincov, ktoré sa v populácii ešte nevyskytli a nie je možné k ním dôjsť len pomocou kríženia. Má vplyv na to, aby algoritmus neuviazol v lokálnom minime. Ukážka najjednoduchšej možnej mutácie je na obrázku 3.2. Aplikovanie operátora mutácie je znázornené na chromozóme vo forme binárneho reťazca. Hodnota náhodného génu v chromozóme sa zmení na opačnú hodnotu.

3.1.2 Priebeh základného genetického algoritmu

Základný najjednoduchší GA, tiež nazývaný kanonický genetický algoritmus, možno opísať ako algoritmus, ktorý premení jednu populáciu kandidátnych riešení na inú populáciu,



Obr. 3.2: Ukážka princípu operácie mutácie.

pomocou použitia genetických operátorov. Výber využíva informácie o súčasnej populácii a uprednostňuje riešenia s lepšou fitness hodnotou. To zodpovedá myšlienke prežitia najsilnejšieho. Kríženie a mutácie upravujú tieto riešenia v snahe odhaliť ešte lepšie riešenia. Mutáciou sa dosahuje zavedenie nových génových vlastností do populácie, zatiaľ čo kríženie umožňuje prepájanie častí existujúcich riešení medzi sebou. Algoritmus 2 uvádza kľúčové kroky v kanonickom genetickom algoritme. [6]

Algoritmus 2: Kanonický genetický algoritmus [6]

Result: najlepšie nájdené riešenie
definuj ako sa má riešenie zakódovať do genotypu a definuj funkciu fitness ;
vytvor počiatočnú populáciu genotypov ;
dekóduj každý genotyp na riešenie a vypočítaj fitness hodnotu každého jedinca ;
repeat
  vyberte n členov z aktuálnej populácie (rodičia), aby ste vytvorili skupinu pre párenie ;
  repeat
    vyber dvoch rodičov zo skupiny párenia ;
    s pravdepodobnosťou p_{cross} vykonaj operáciu kríženia na vybraných rodičoch, čím sa vytvorí dvojica potomkov ;
    s pravdepodobnosťou p_{mut} vykonaj operáciu mutácie na potomkoch ;
  until je vytvorených n nových potomkov;
  nahraď starú populáciu novo vytvorenou (nová generácia) ;
until splnená podmienka ukončenia;

3.2 Evolučné stratégie

Evolučné stratégie (ES) sú najstarším typom z rodiny evolučných algoritmov a celkovo patria medzi prvé úspešné stochastické algoritmy. ES boli vytvorené Rechenbergom a Schwefelom v 60-tych rokoch v Nemecku v kontexte technickej inžinierskej optimalizácie (aerodynamika telies) [17]. Rovnako ako všetky evolučné algoritmy, ES boli inšpirované evolúciou v prírode a prirodzeným výberom, ale nezachádzajú v tejto myšlienke do takej hĺbky ako napríklad genetické algoritmy. V Dnešnej dobe patria ES medzi kvalitné a rozvinuté optimalizačné metódy, používané predovšetkým v oblasti technických aplikácií v inžinierskej oblasti [13].

Základný ES je definovaný formulou $x(t+1) = x(t)+N(0,\sigma)$, kde x je vektor n reálnych čísel, $x \in R^n$ a $N(0,\sigma)$ je vektor náhodných čísel s nulovou stredovou hodnotou a štandardnou odchýlkou σ. Táto formula vyjadruje generovanie nových hodnôt z aktuálneho riešenia $x(t)$ na nové riešenie $x(t+1)$. Smerodajná odchýlka σ reprezentuje mieru rozptylu generovaných hodnôt. Väčšia hodnota σ znamená väčší rozdiel medzi rodičom a potomkom. Hodnoty vo vektore $N(0,\sigma)$ sú náhodne rozložené, napríklad podľa gaussovskej distribúcie. Selekcia nových riešení v ES je striktne deterministická, v závislosti na fitness hodnote jedinca (silnejší vždy víťazia)[13].

3.2.1 Základne stratégie reprodukcie a prežitia u ES

Existuje viacero variant ES, ktoré sa označujú podľa počtu rodičov μ a počtu potomkov λ. ES ktorá pracuje len s jedným rodičom a jedným potomkom sa tiež označuje ako **(1 + 1)** evolučná stratégia. V tejto variante algoritmu teda tvorí celú populáciu len jeden jedinec (rodič), z ktorého sa generuje len jeden jedinec (potomok). Následne medzi sebou súperia (rodič a potomok), kto z nich je lepším a tým pádom postúpi do novej generácie.

Viac sofistikovanou verziou ES sú mnohočlenné stratégie, čiže také, kde je viacero rodičov v populácii a viacero generovaných potomkov. Dva reprezentanti mnohočlenných ES sú tzv. "PLUS" ES označované ako $(\mu + \lambda)$ a "ČIARKA" ES označované ako (μ, λ) [13]. V prípade stratégie $(\mu+\lambda)$ sa na začiatku náhodne inicializuje počiatočná populácia o μ jedincoch a vyhodnotí sa ich fitness hodnota. Následne sa z μ rodičov vygeneruje λ potomkov a s pravdepodobnosťou p sa na nich aplikuje operátor mutácie. Takto novo vygenerovaný potomkovia sa spoja s pôvodnými rodičmi a všetci dohromady sa zoradia od najviac vhodného riešenia po najmenej vhodné riešenie. Prvých μ jedincov postupuje do novej populácie a stanú sa novými rodičmi v ďalšej generácii. Tieto kroky sa opakujú, kým nie je splnené ukončujúce kritérium. Základný priebeh $(\mu + \lambda)$-ES, ako tu bol popísaný, je znázornený v Algoritme 3. U stratégii (μ, λ) je priebeh podobný, ale s tým rozdielom, že sa vyberá μ jedincov, ktorý postúpia do novej generácie iba z novo vygenerovaných λ potomkov. Predošlý rodičia týchto potomkov vymierajú, to znamená, že život každého jedinca je obmedzený len na jednu generáciu. Pri oboch metódach je možné využiť okrem operácie mutácie aj operáciu kríženia [13].

Algoritmus 3: Základný algoritmus evolučných stratégie $(\mu + \lambda)$ [6].

```
Result: najlepšie nájdený riešenie
inicializácia populácie o veľkosti μ jedincov ;
vyhodnotenie všetkých μ jedincov (fitness hodnota) ;
repeat
    repeat
        náhodný výber rodiča z μ ;
        vygeneruj potomka aplikovaním operácie mutácie na vybraného rodiča ;
    until vygenerovaných λ potomkov;
    vyhodnotenie λ potomkov ;
    zotrieď μ + λ rodičov a potomkov od najlepšieho po najhorší ;
    vyber μ najlepších jedincov z rodičov a potomkov, ktorý postúpia do ďalšej
     generácie ;
until splnená podmienka ukončenia;
```

Kapitola 4

Mutačné operátory pre riešenie VRPPD

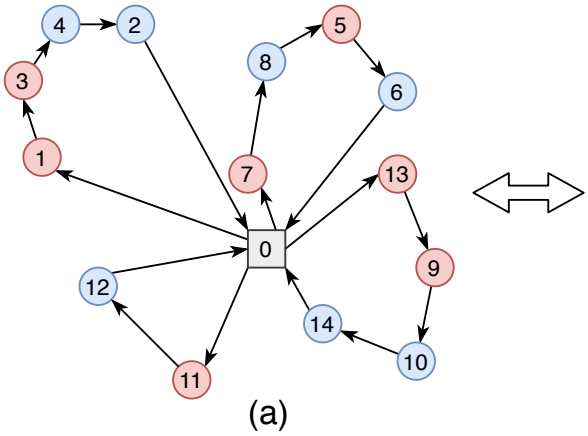
Pre potreby evolučnej optimalizácie problému VRPPD v rámci tejto DP bolo navrhnuté používanie niekoľkých operátorov mutácie. Hľadanie riešenia VRPPD je kombinatorický problém, čiže chromozóm je permutácia jednotlivých génov, ktoré sú reprezentované miestami umiestnenými v trasách. To znamená, že každé miesto sa v riešení nachádza práve jeden krát a jeho presné umiestnenie v rámci chromozómu je nevyhnutným aspektom riešenia. Dôležitou vlastnosťou riešenej úlohy je fakt, že nie každá permutácia riešenia je platná (podmienky platnosti riešenia pre úlohu v tejto práci sú v kapitole 2).

Vzhľadom na charakteristiku problému VRP existujú dve základné typy manipulácie s jedincom. Za prvé, zmena vozidla, ktoré má na starosti prevoz požiadavky na prepravu, čiže to, v ktorej trasa sa daný požiadavok na prepravu nachádza. Presun požiadaviek na prepravu medzi trasami sa taktiež odráža na výslednom počte použitých vozidiel vo výsledku. Ak sa v trase nachádza len jeden požiadavok na prepravu a tento požiadavok na prepravu je vybraný na presun, zníži sa tak počet použitých vozidiel. Opačný prípad, čiže vkladanie požiadavky na prepravu do prázdnej trasy počet použitých vozidiel navýši. Druhý typ manipulácie s jedincom je zmena poradia, v akom sú jednotlivé miesta navštívené vozidlom.

V tejto kapitole je na úvod predstavené zakódovanie reprezentácie riešenia VRPPD a následne sú tu postupne popísané všetky mutačné operátory použité pre účely tejto diplomovej práce.

4.1 Zakódovanie riešenia

Výsledkom VRPPD, rovnako ako aj základného VRP, je plán trasovania rozvozu. To znamená, zoznam trás pre jednotlivé vozidlá, kde každá trasa pozostáva zo zoznamu miest, v poradí, v akom majú byť postupne navštevované. To znamená, že na zakódovanie riešenia bol použitý zoznam zoznamov. Trasa je teda zakódovaná v celočíselnom lineárnom zozname miest s presne daným usporiadaním. Každý požiadavok na prepravu sa skladá z dvojice miest, vyzdvihnutia a doručenia. Vyzdvihnutie p_i je zakódované ako nepárne celé číslo a k nemu pridružené doručenie je zakódované ako nasledujúce párne číslo $d_i = p_i + 1$. Depo je zakódovaný ako číslo 0. Princíp reprezentácie riešenia a chromozómu je znázornený na obrázku 4.1, celá tabuľka reprezentuje jedného jedinca v populácii (chromozóm).



trasa 0:	0	1	3	4	2	0
trasa 1:	0	7	8	5	6	0
trasa 2:	0	13	9	10	14	0
trasa 3:	0	11	12	0		

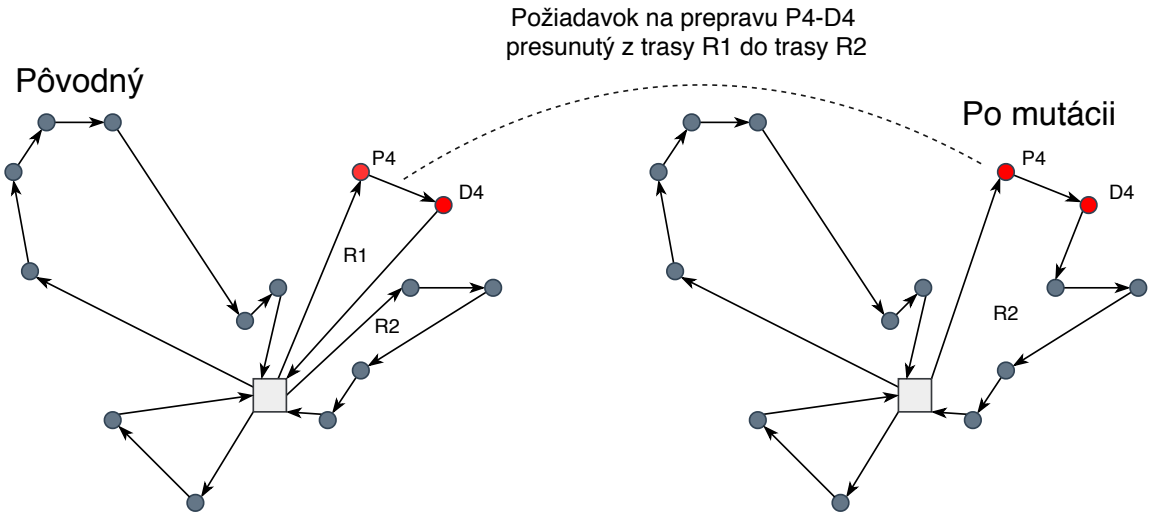
Obr. 4.1: Princíp kódovania riešenia. Časť (a) znázorňuje reálnu reprezentáciu daného riešenia. Miesto depo je zobrazené ako šedý štvorec s nulou. Miesta vyzdvihnutia sú vyobrazené červenými bodmi, miesta doručenia sú vyobrazené modrými bodmi. Pridruženým miestom doručenia je označené nasledovným číslom k miestu vyzdvihnutia, čiže napríklad k miestu vyzdvihnutiu 1 prislúcha miesto doručenia 2, ku 3 je 4 atď. V časti (b) je vyobrazená forma zakódovania riešenia vo forme tabuľky, ktorá reprezentuje zoznam trás v riešení, kde každá trasa je zoznam miest, v poradí, v akom sú navštívené. Príklad, vozidlo začne v depe, označeným 0, vyzdvihne tovar v mieste 1, pokračuje do miesta 3, následne v mieste 4 vyloží naložený tovar z miesta 3, ďalej vyloží tovar v mieste 2 a vracia sa do depa 0.

4.2 Náhodný presun medzi dvoma trasami

U tohto mutačného operátora (`randomTransfer`) sa jedná o to, že sa presunie požiadavok na prepravu medzi dvoma trasami. Priebeh operácie je taký, že sa na začiatok náhodne vyberie jedna trasa, ktorá obsahuje aspoň jeden požiadavok na prepravu a následne sa náhodne zvolí druhá, odlišná, trasa. V prvej zvolenej trase sa náhodne zvolí miesto, ku ktorému sa nájde pridružený pár (ak bolo zvolené doručenie, nájde sa k nemu vyzdvihnutie; ak bolo zvolené vyzdvihnutie nájde sa k nemu doručenie). Následne sa celý požiadavok na prepravu odstráni z prvej trasy a vloží sa do druhej trasy na náhodnú pozíciu spôsobom, že vyzdvihnutie a doručenie sú v trase hneď za sebou. Princíp presunu požiadavky na prepravu medzi dvoma trasami je znázornený na obrázku 4.2.

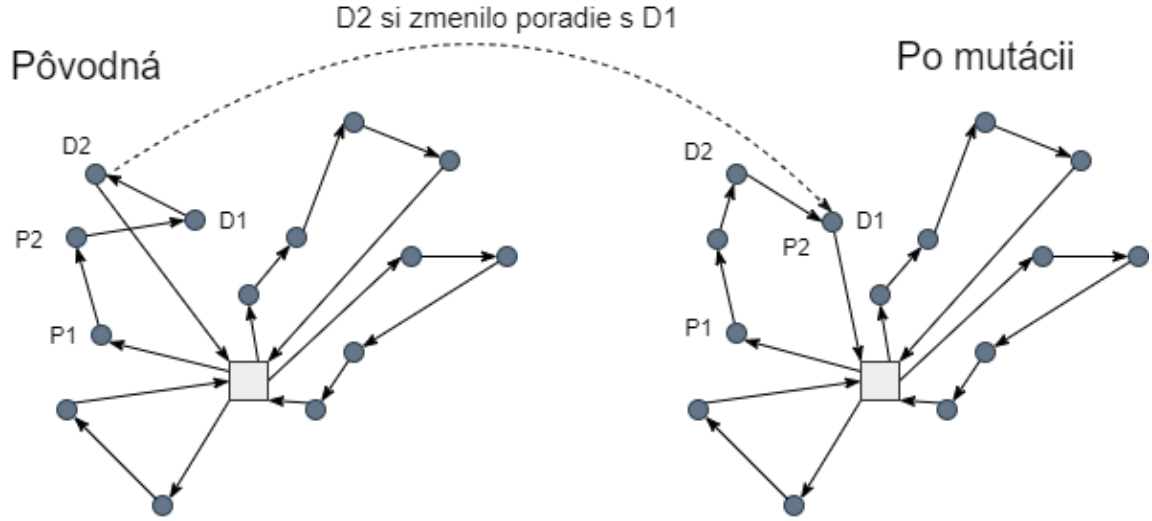
4.3 Náhodná výmena pozícií dvoch miest v trase

Táto operácia (`randomSwitch`) pracuje len nad jednou trasou, u ktorej mení poradie navštívených miest. Zámena poradia dvoch miest v trase prebieha tak, že sa náhodne vyberie trasa, ktorá obsahuje aspoň dve požiadavky na prepravu (4 miesta). V danej trase následne prebehne niekoľkokrát výmena dvoch miest. Počet vykonaných výmen miest je náhodný, v rozsahu $(1, \texttt{MUT\_PER\_ROUTE})$. Vo vybranej trasa sa následne vyberie jedno miesto takým spôsobom, že čím väčšie má miesto vstupné a výstupné hrany, tým väčšia je pravdepodobnosť výberu tohto miesta. Následne sa zvolí druhé miesto na výmenu tak, aby nebola porušená podmienka poradia vyzdvihnutia a doručenia prvého vybraného miesta. To sa docieli tak, že sa druhé miesto volí len z miest na pozíciách možných pre výmenu s prvým vybraným miestom. Napríklad, ak je ako prvé miesto zvolené vyzdvihnutie, ktorého



Obr. 4.2: Operátor mutácie presunu požiadavky na prepravu medzi dvoma trasami. Červene zvýraznený požiadavok na prepravu je presunutý z trasy R1 do trasa R2, čím trasa R1 zanikla (R1 je prázdna).

príslušné doručenie je poradím piatim miesto v trase, poradie druhého zvoleného miesto je v intervale 1 až 5. Výmenou týchto miest avšak stále môže nastať to, že trasa nebude platná (prekročená kapacita vozidla, prekročená maximálna dĺžka trasy alebo porušenie poradia vyzdvihnutia-doručenia u druhého miesta). V takom prípade sa zopakuje výber druhej lokácie. Ak sa nenájde splniteľná dvojica pre výmenu v piatich pokusoch, po výmene miest sa vykoná oprava neplatnej trasy. Opravou trasy sa rozumie to, že sa s čo najmenším počtom úprav presunú miesta v trase tak, aby neboli porušené žiadne tvrdé podmienky splniteľnosti riešenia. Ukážka operácie výmeny poradia dvoch miest v trase je v príklade na obrázku 4.3.

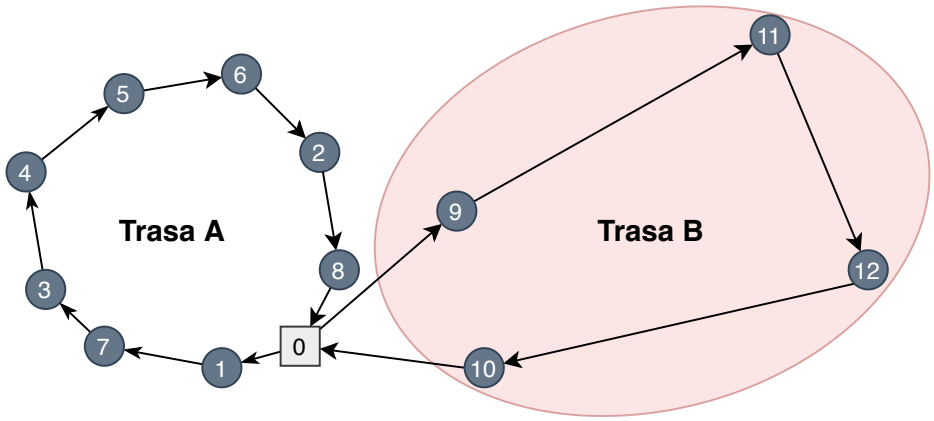


Obr. 4.3: Operátor mutácie zmeny dvoch miest v trase. Poradie miesta D2 bolo zamenené s poradím miestom D1, čím sa trasa zmenila z 0-P1-P2-D1-D2-0 na trasu 0-P1-P2-D2-D1-0.

4.4 Cielený presun medzi trasami podľa ceny

Operátor mutácie (`guidedTransfer`), ktorý cielene presúva požiadavok na prepravu medzi dvoma trasami, sa od operátoru mutácie náhodného presunu líši v troch bodoch. Spôsobe výberu upravovanej trasy, vo výbere presúvanej požiadavky na prepravu a v metóde vkladania požiadavky na prepravu do druhej trasy.

Na začiatku sa vyberie trasa r_i z ktorej sa požiadavok na prepravu presúva. Určenie, ktorú trasu je potrebné upraviť, prebieha tak, že všetkým neprázdnym trasám v riešení je priradená váha výberu. Táto váha výberu je závislá na pomere celkovej ceny trasy a počtu požiadaviek na prepravu, ktoré sú v danej trase obsluhované. To znamená, že trasa s veľkou cenou a malým počtom požiadaviek na prepravu je vybraná s väčšou pravdepodobnosťou. Príklad je znázornený na obrázku 4.4, kde trasa B má väčšiu váhu výberu. Samotný výber následne prebieha pomocou *diskrétnej distribúcii pravdepodobnosti*[1], podľa váhy výberu trás. V zvolenej trase sa následne vyberie jeden požiadavok na prepravu. Ten sa vyberá tak, že sa priradí váha všetkým požiadavkám na prepravu v trase, podľa cien vstupných a výstupných hrán oboch miest z danej požiadavky na prepravu. Čím väčšia cena hrán, tým väčšia pravdepodobnosť výberu. Takto sa potom pomocou diskrétnej distribúcii pravdepodobnosti vyberie jeden požiadavok na prepravu, ktorý bude presunutý. Druhá trasa r_j, do ktorej je požiadavok na prepravu presunutý, je vybraná náhodne.



Obr. 4.4: Príklad toho, akým spôsobom sa určuje výber trasy, ktorá sa má upravovať. Trasa B má menší pomer obsluhovaných požiadaviek na prepravu k celkovej cene trasy, čiže trasa B má väčšiu pravdepodobnosť, že bude vybraná na úpravu.

Následne sa požiadavok na prepravu odstráni z trasy r_i a vloží sa do trasy r_j. Pri vkladaní požiadavky na prepravu do trasy je kladený dôraz na dve kritéria. Za prvé, výsledná trasa po vložení musí byť platná, čiže nesmie porušovať žiadne pravidlá splniteľnosti, ktoré sú kapacita vozidla, maximálna dĺžka trasy a poradie vyzdvihnutia a doručenia. Za druhé, obidve miesta z požiadavky na prepravu sú do trasy vložené na čo najvhodnejšiu pozíciu v poradí. To je dosiahnuté tak, že ako prvé sa hľadajú a ohodnotia všetky splniteľné pozície pre miesto vyzdvihnutia a následne sa k všetkým splniteľným miestam vyzdvihnutia vy-

[1]Rozdelenie pravdepodobnosti produkuje náhodné celé číslo v intervale $[0, n)$ takým spôsobom, že každý i-ty prvok má pravdepodobnosť výberu w_i/S, kde w_i je váha i-teho prvku a S je suma váh všetkých prvkov.

tvorí pár najlepšej možnej pozície pre umiestnenie miesta doručenia v trase. Pár pozícií pre vloženie miesta vyzdvihnutia a doručenia, ktorý navýši cenu trasy čo najmenej, je vybraný pre vloženie miest do trasy.

4.5 Cielený presun medzi trasami podľa ťažiska

Táto operácia mutácie (`centroidTransfer`), pri ktorej sa presunie požiadavok na prepravu z jednej trasy do druhej, je založená na ťažisku ("centroide") trasy. Každá trasa má predpočítané svoje ťažisko, ktoré vychádza z rozmiestnenia všetkých miest v trase (vrátane depa). Výpočet ťažiska trasy je ukázaný v rovnici 4.1. Operácia prebieha tak, že sa ako prvé určí trasa r_i, z ktorej sa požiadavok na prepravu vyjme. Tento výber je náhodný s tým, že vybraná trasa nesmie byť prázdna. Vo vybranej trasa sa následne vyberie jeden požiadavok. Postup tohoto výberu je taký, že je každej požiadavke na prepravu priradená váha, ktorá určuje to, ako veľmi je daný požiadavok na prepravu vzdialený od ťažiska trasy. Čím viac je miesto vzdialené od ťažiska, tým väčšia pravdepodobnosť, že bude vybrané. Výpočet hodnoty vzdialenosti miesta od ťažiska je vo vzorci 4.2. Pre zachovanie náhodnosti je opäť pre samotný výber použitá diskrétnej distribúcie pravdepodobnosti na základe priradených váh každej požiadavke na prepravu. Základný princíp výberu požiadavky na prepravu na základe ťažiska trasy je na obrázku 4.5. Ďalej sa zvolí trasa r_j, podľa umiestnenia ťažiska trasy vzhľadom k miestam premiestňovanej požiadavky na prepravu. Čím menšia vzdialenosť miest presúvanej požiadavky na prepravu od ťažiska trasy, tým väčšia pravdepodobnosť, že bude vybraná práve táto trasa. Potom sa požiadavok na prepravu odstráni z pôvodnej trasy r_i a vloží oba miesta na čo najlepšiu pozíciu v trase r_j (postup vkladania požiadavky na prepravu do trasy je rovnaký ako u mutačného operátora `guidedTransfer` z podkapitoly 4.4).

Ťažisko trasy je vypočítané ako aritmetický priemer súradníc všetkých miest. Ťažiskom trasy je bod so súradnicami c_x a c_y, súradnice jednotlivých miest v trase sú (x_i, y_i):

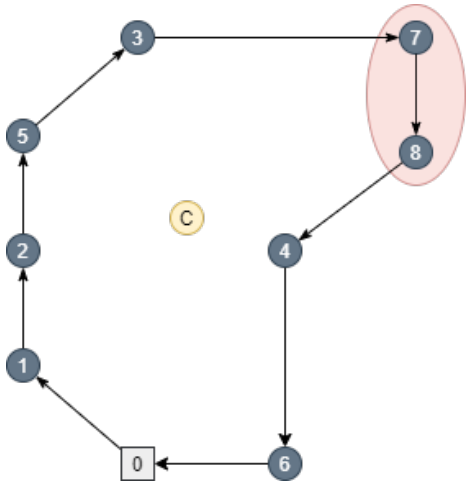
$$c_x = \frac{1}{n}\sum_{n=0}^{n-1} x_i \quad c_y = \frac{1}{n}\sum_{n=0}^{n-1} y_i \tag{4.1}$$

Výpočet hodnoty, ktorá určuje ako je miesto vzdialené od ťažiska trasy, kde (x, y) sú súradnice miesta a (c_x, c_y) sú súradnice ťažiska trasy:.

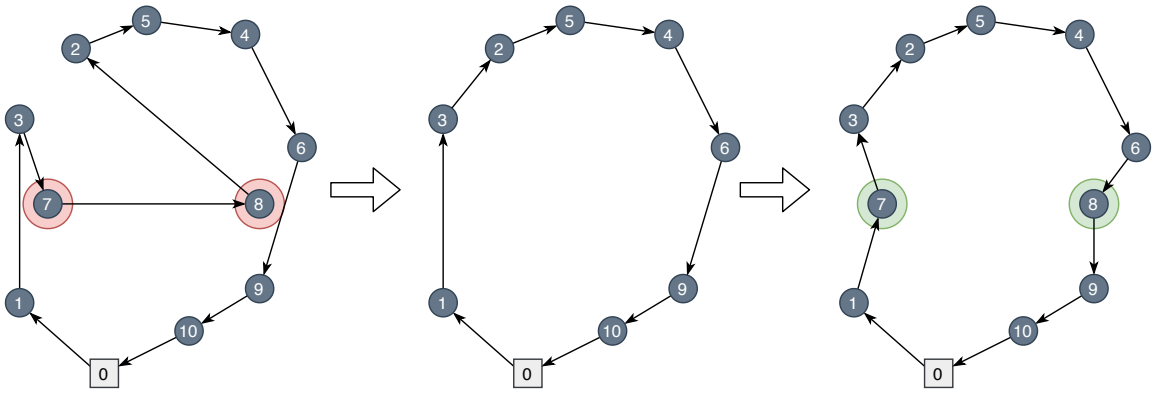
$$(x - c_x)^2 + (y - c_y)^2 \tag{4.2}$$

4.6 Presunutie požiadavky na prepravu v rámci jednej trasy

Presunutie požiadavky na prepravu v trase (`realocate`) je posledným mutačným operátorom. Ako prvé sa zvolí trasa (obsahujúca aspoň dve požiadavky na prepravu), podľa pomeru ceny trasy a počtu miest umiestnených v trase tak, že dlhšie trasy s menším počtom miest sú vybrané s väčšou pravdepodobnosťou. Ukážka na obrázku 4.4. Následne sa vyberie požiadavok na prepravu podľa príslušných hrán obsiahnutých miest. Obidva miesta z vybranej požiadavky na prepravu sú potom odstránené zo zvolenej trasy a následne sú oba miesta vložené na čo najlepšie možné pozície v trase (postup vkladania požiadavky na prepravu do trasy je rovnaký ako u mutačného operátora `guidedTransfer` z podkapitoly 4.4). Ukážka priebehu tejto operácie je v príklade na obrázku 4.6.



Obr. 4.5: Vizualizácie použitia ťažiska trasy (označený ako C), v mutačnom operátore `centroidTransfer`, pre výber požiadavky na prepravu, ktorý má byť vyňatý z trasy.



Obr. 4.6: Operátor mutácie úpravy pozície požiadavky na prepravu - `realocate`. Na začiatku je požiadavok na prepravu 7-8 vyhodnotený ako najhorší v trase. Následne sa odstráni z trasy. Nakoniec sa pre obe miesta 7 a 8 nájdu najvhodnejšie pozície, do ktorých sa miesta v trase vložia.

Kapitola 5

Evolučné riešenie problému VRPPD

Vrámci tejto diplomovej práce boli navrhnuté štyri varianty evolučných algoritmov pre riešenie problému VRPPD. U jednej z týchto variánt je použitý genetický algoritmus a u zvyšných troch bola použitá evolučná stratégia. Všetky varianty zdieľajú medzi sebou niektoré rysy, ako napríklad reprezentáciu riešenia, počiatočnú inicializáciu populácie, vyhodnocovanie kvality riešenia alebo niektoré použité mutačné operátory. Základ evolučnej stratégie, u všetkých troch variánt je takmer rovnaký, tieto varianty sa líšia hlavne v kroku mutácie. V každej navrhnutej variante je použitá reprezentácia riešení tak, ako bola popísaná v podkapitole 4.1.

Nastavenie parametrov evolučných algoritmov je vidno v tabuľke 5.1. Hodnoty konfigurácii parametrov boli získané ladením evolučných algoritmov. Parameter určujúci maximálny počet generácií (`GENERATIONS`) je premenlivý, v závislosti na veľkosti a charakteristike vstupnej úlohy pre vyriešenie.

V tejto kapitole je na začiatku popísaný princíp inicializácie populácie a vyhodnocovanie kvality riešenia. Následne postupne v podkapitolách sú opísané charakteristiky a nastavenia jednotlivých navrhnutých variánt evolučných algoritmov pre riešenie VRPPD. V závere kapitoly sú stručne opísané základné informácie k implementácii daných algoritmov.

parameter	hodnota - GA	hodnota - ES
PMUT	50	50
TOUR	4	-
POPSIZE	30	-
λ	-	30
μ	-	3
MUTAGENE_PER_ROUTE	2	3
MUTAGENES	2	4
ELITISM	TRUE	FALSE
ES_PLUS	-	TRUE

Tabuľka 5.1: Nastavenie hlavných evolučných parametrov.

5.1 Inicializácia populácie

Iniciálna populácia je generovaná náhode s cieľom čo maximálneho pokrytia prehľadávaného priestoru. Populácia o veľkosti μ riešení, vo variantách založených na evolučnej stratégii a o veľkosti POPSIZE u variante založenej na genetickom algoritme. Každý jedinec je náhodne vygenerované platné riešenie, zostrojené nasledovne. Postupne sa za radom umiestňujú miesta do náhodne vygenerovanej trasy. Ak už nie je možné miesto pridať do trasy (porušenie platnosti riešenia), vloží sa do inej náhodne zvolenej trasy. Ak je na rade d_i, ktorého p_i nie je v aktuálnej trase, d_i sa vloží na koniec trasy v ktorej je pridružené p_i.

5.2 Výpočet kvality riešenia

V každej generácii evolúcie je vyhodnotená fitness hodnota F pre každé riešenie v populácii. Každý jedinec má svoju fitness hodnotu F na základe ktorej sa potom odvíja priebeh evolučného procesu. Fitness hodnota F závisí na sume ceny všetkých trás v danom riešení. Čím menšia suma cien trás, tým lepšia fitness hodnota jedinca. Výpočet ceny jednej trasy je podľa matematickej rovnice 5.1.

Cenu cesty c_{ij}, medzi dvojicou miest, vypočítame ako euklidovskú vzdialenosť dvoch bodov:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{5.1}$$

Majme trasu $r_k = \langle u_1^k, ..., u_{N_k}^k \rangle$ reprezentujúcu postupnosť N_k zastávok u miest, v k-tej trase, kde u_i^k je i-ta zastávka trasy k. Potom celkovú cenu pre vozidlo k možno získať ako:

$$C_k = \sum_{i=0}^{N_k} c_{u_i^k u_{i+1}^k} \tag{5.2}$$

Fitness funkcia je obrátená hodnota sumy všetkých trás R v riešení z rovnice 5.2 (za účelom udržania fitnes hodnoty v intervale (0, 1) u väčšiny prípadov), čiže:

$$F = \frac{1}{\sum_{k \in R} C_k} \tag{5.3}$$

5.3 Základná evolučná stratégia

U tejto varianty (ďalej v práci označovaná ako ES) algoritmu bola použitá evolučná stratégia, konkrétne $(\mu + \lambda)$-ES. Nastavenie evolučných parametrov je v tabuľke 5.1. Priebeh algoritmu je nasledovný. Na začiatok sa inicializuje počiatočná populácia o veľkosti μ, tak ako to bolo popísané v podkapitole 5.1. Následne prebieha evolučný cyklus, až kým nieje splnená podmienka ukončenia, ktorou je dosiahnutie určitého počtu generácii. Evolučný cyklus začína generovaním λ potomkov. Potom sa tvorí spôsobom, že sa náhodne vyberie jeden z rodičov, na ktorého je následne aplikovaná mutácia.

Operácia mutácie sa aplikuje jeden až `MUTAGENES`-krát, rotujúc pri generovaní potomkov tak, že $nMut = (i \mod \texttt{MUTAGENES}) + 1$, kde $nMut$ je počet volaných mutácií pri generovaní jedného potomka a i poradové číslo aktuálne generovaného potomka v rozsahu $(0, \lambda)$. Pri každej operácii mutácie sa s pravdepodobnosťou `PMUT` aplikuje jeden z dvoch mutačných operátorov. Mutačné operátory použité v tejto variante sú `randomTransfer` popísaný v podkapitole 4.2 a `randomSwitch` popísaný v sekcii 4.3. Zvolenie, ktorý z týchto dvoch operátorov sa použije, je náhodne určené s pravdepodobnosťou 50% pre oba operátory.

Ďalší postup algoritmu je klasický $(\mu + \lambda)$-ES, to znamená, z rodičia a potomkovia sa zoradia od najlepšie po najhoršie, podľa fitnes hodnoty popísanej v podkapitole 5.2. Z nich sa následne vyberie μ najlepších jedincov, ktorý idú do ďalšej generácie.

5.4 Evolučná stratégia s použitím cielených presunov

Varianta algoritmu evolučná stratégia s použitím cielených presunov (ES-CM) má priebeh podobný ako predošlá varianta ES z podkapitoly 5.3. Základ je $(\mu + \lambda)$-ES, počiatočná inicializácie populácie (sekcia 5.1), a následne evolučný cyklus. Generovanie potomkov a tvorba novej populácie je taktiež totožná. Rozdiel prichádza hlavne v mutovaní jedincov pri tvorbe nových riešení. Počet aplikovaní mutácie je opäť rotujúci v rozsahu jeden až `MUTAGENES` krát, pričom každé použitie je ešte limitované s pravdepodobnosťou použitia PMUT. Pri tejto variante sa využívajú tri mutačné operátory, `guidedTransfer` (sekcia 4.4), `randomSwitch` (sekcia 4.3) a `realocate` (sekcia 4.6). Každý z týchto operátorov má rovnakú pravdepodobnosť výberu. Ak je zvolený operátor `guidedTransfer`, dodatočne sa následne s pravdepodobnosťou `PMUT` aplikuje operátor `realocate` na trasa, do ktorej sa presúval požiadavok na prepravu. Výber jedincov postupujúcich do ďalšej generácie je opäť rovnaký pre všetky navrhnuté varianty, založené na evolučnej stratégii, popísané v podkapitole 5.3.

5.5 Evolučná stratégia s použitím cielených presunov a ťažiskom trasy

Ako už bolo spomenuté v predošlých podkapitolách, základ evolučnej stratégie je aj u tejto variácie evolučná stratégia s použitím cielených presunov a ťažiskom trasy (ES-CMC) rovnaký. Rozdiel je v kroku mutácie, kde si líši aplikovanie a použité mutačné operátory. Táto varianta používa opäť tri mutačné operátory, `randomSwitch` (sekcia 4.3), `centroidTransfer` (sekcia 4.5) a `realocate` (sekcia 4.6). Výber toho, ktorý operátor mutácie sa použije je náhodne, pričom každý operátor má rovnakú pravdepodobnosť, že bude zvolený. Nastavenie evolučných parametrov je spoločné, ako v tabuľke 5.1.

5.6 Genetický algoritmus s použitím cielených presunov a ťažiskom trasy

Táto varianta navrhnutého algoritmu je založená na genetickom algoritme a využití cielených presunov a ťažiska trasy (označovaná ako GA-CMC). Nastavenie evolučných parametrov sú v tabuľke 5.1 v druhom stĺpci. Algoritmus začína s vytvorením počiatočnej populácie o veľkosti `POPSIZE`, podľa postupu napísaného v podkapitole 5.1.

Evolučný cyklus u tejto varianty prebieha nasledovne. Ako prvé sa vyhodnotí kvalita všetkých jedincov, tak ako to je popísané v podkapitole 5.2. Následne prebieha tvorba novej populácie. Algoritmus využíva elitizmus, čiže najlepší rodič postupuje ako prvý jedinec do novej generácie. Druhým zvoleným jedincom postupujúcim do ďalšej generácie je najlepší jedinec po tom, ako bola na ňom aplikovaná operácie mutácie. Zvyšných `POPSIZE` $-$ 2 jedincov je vyberaných metódou turnaja. Do turnaja vstupujú náhodne vybraný jedinci z rodičov, o počte TOUR. Dvaja najlepší jedinci z turnaja postupujú do ďalšieho kroku, kde je následne na obdivoch jedincoch aplikovaná operácie mutácie. Aplikovanie mutačných operátorov je vykonaných 1 až `MUTAGENES` krát, náhodne určené, pričom každé aplikovanie je

následne ešte vykonané s pravdepodobnosťou `PMUT`. Táto varianta používa rovnaké mutačné operátory ako varianta ES-CMC, `randomSwitch` (sekcia 4.3), `centroidTransfer` (sekcia 4.5) a `realocate` (sekcia 4.6). Turnaj sa opakuje až dokým neje vytvorená nová populácia o veľkosti `POPSIZE`. Počet evolučných cyklov je obmedzený hodnotou `GENERATIONS`.

5.7 Implementácia

Táto podkapitola sa zameriava na priblíženie obecných techník a metód použitých pri implementácie programu pre riešenie VRPPD. Program je implementovaný v programovacom jazyku C++. Vývoj prebiehal v prostredí operačného systému Linux. Implementácia využíva len štandardné knihovne jazyka C++.

Na implementáciu boli použité charakteristiky objektovo orientovaného programovania. Program obsahuje päť tried. Trieda `Config` reprezentuje nastavenie evolučných parametrov. Ďalšou triedou je `Task`, ktorou úlohou je spracovanie vstupných dát úlohy, ktorá sa má riešiť. Trieda pripraví dátovú štruktúru, s ktorou sa následne pracuje pri riešení VRPPD a predpočíta maticu vzdialeností medzi všetkými miestami v úlohe. Trieda `Individual` reprezentuje jedinca v populácie, respektívne predstavuje jedno z možných riešení pre úlohu VRPPD. Jej úlohou je definovanie jedinca a manipulácia s jedincom. Nasledujúca trieda `Route` predstavuje jednu trasu jedného vozidla a obsahuje metódy, ktorú manipulujú s trasou. Trieda `Solver` je základ programu, kde sa spája všetko dohromady a jej metóda `Solve()` obsahuje samotnú implementáciu algoritmu pre riešenie VRPPD. Reprezentácia populácie je teda vektor Objektov z triedy `Individual`, kde každý objekt `Individual` má svoj vektor objektov z triedy `Route`.

Kapitola 6

Experimentálne výsledky

V tejto kapitole sú popísané dosiahnuté výsledky experimentov na implementácii navrhnutých algoritmov v tejto práci. Popis dátovej sady, nad ktorou boli experimenty vykonávané, je v podkapitole 6.1. Experimenty sú rozdelené do dvoch častí. Prvá časť sa zaoberá porovnaním štyroch navrhnutých variánt evolučného prístupu k riešenie VRPPD, ktoré boli predstavené v tejto práci. V podkapitole 6.2 je vysvetlený princíp experimentu, následne zhrnutie dosiahnutých výsledkov a nakoniec zhodnotenie a diskusia nad dosiahnutými výsledky pre tento experiment. Druhý typ experimentu sa zameriava na porovnanie navrhnutého algoritmu ES-CMC v tejto práci (ktorý dosahoval najlepších výsledkov z pomedzi štyroch navrhnutých variánt), s existujúcou metódou pre riešenie problému. Konkrétne sa jedná o porovnanie s konvenčným nástrojom OR-Tools (predstavený v sekcii 2.3.4), ktorý je často využívaný v praxi. Podkapitola 6.3 popisuje experiment, demonštruje dosiahnuté výsledky experimentu, následne rozoberá, analyzuje a zhodnocuje nadobudnuté výsledky tohto experimentu.

Všetky spúšťania programu v rámci výsledných experimentov boli vykonané na počítačovom zhluku Anselm, národného superpočítačového centra IT4Innovations[1]. Pre každé spustenie sa vždy alokuje šesť 16-jadrových uzlov, pričom na každom z uzlov sa spustí program, čiže sa vždy spúšťa 96 (6 x 16) behov programu súbežne (v prípade, že všetkých 6 uzlov je už voľných a priradených).

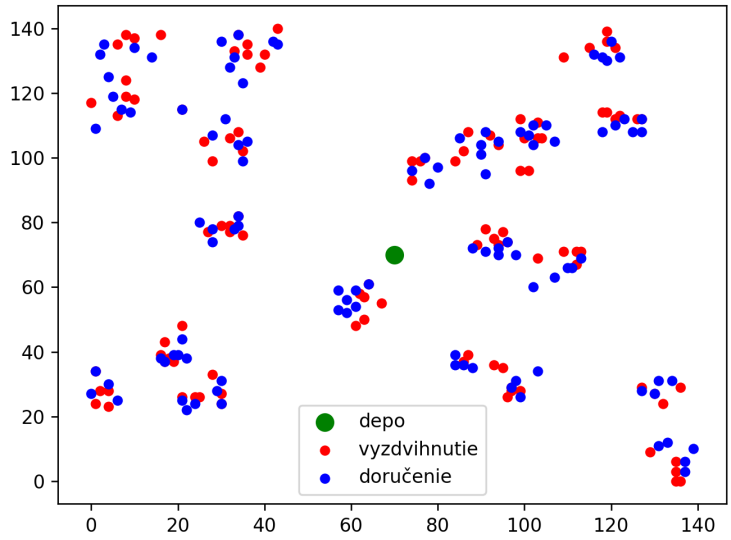
6.1 Dátová sada použitá pre účely experimentov

Použitá dátová sada pre experimenty bola čerpaná z *Li & Lim benchmark*[2]. Táto dátová sada je určená primárne pre problém VRPPDTW (problém smerovania vozidiel s vyzdvihnutím a doručením a s časovými oknami). Pre potreby tejto diplomovej práce bola však riešená o trochu iná variácia VRP. Na rozdiel od VRPPDTW je v tejto práci riešený problém bez časových okien, ale s tým, že je pridané obmedzenie maximálnej dĺžky trasy. Aj napriek týmto odlišnostiam v riešenom probléme, je dátová sada dostačujúca a použiteľná pre overenie funkčnosti navrhnutého algoritmu v tejto práci.

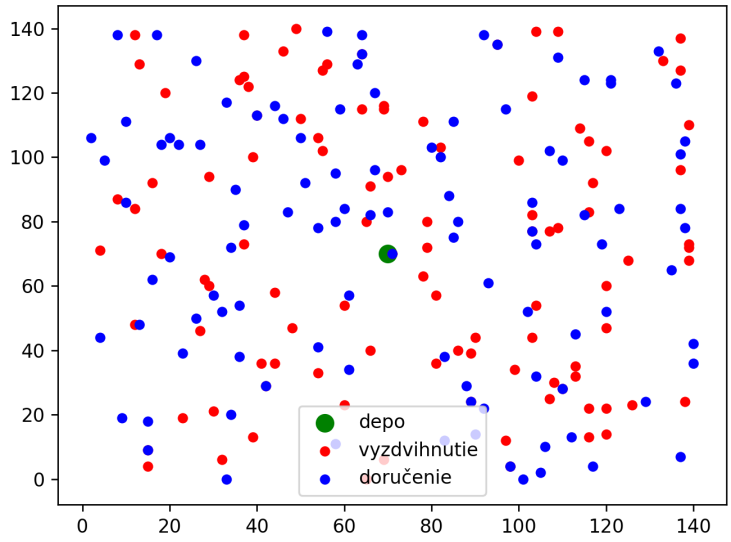
Dátová sada obsahuje inštancie rôznych veľkostí (počtu miest v úlohe) - 100, 200, 400, 600, 800 a 1000. V dátovej sade je obsiahnutých niekoľko charakteristík problém. Hlavné tri charakteristiky sú miesta umiestnené v zhlukoch (*clusters* - C), náhodne rozmiestnené miesta vyzdvihnutia a doručenia (*random* - R) a kombinácia týchto charakteristík, miesta

[1] https://www.it4i.cz/
[2] https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/

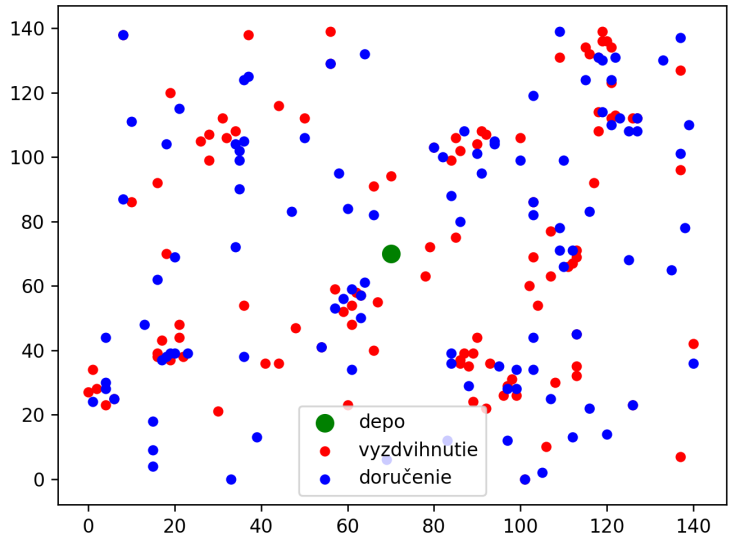
sú rozmiestnené náhodne, ale časť požiadaviek na prepravu je sformovaná do zhlukov spôsobom, že v jednom zhluku je viacero vyzdvihnutí, ktorých miesta doručenia sú zoskupené do iného zhluku (*random and clusters* - RC). Každá z týchto charakteristík má dve variácie zamerania sa na iný aspekt, variácia 1 má menšiu kapacitu vozidiel a užšie časové okná v miestach (čo v prípade tejto práce nehrá žiadnu rolu, pretože časové okná niesu brané v úvahu), zatiaľ čo variácia 2 má väčšiu kapacitu vozidiel a väčšie časové okná. Názov jednotlivých inštancií je zložený v tvare L⟨ch⟩⟨z⟩_⟨n⟩_⟨x⟩, kde `ch` je charakteristika (C, R alebo RC), `z` je zameranie (1 - menšia kapacita vozidiel/užšie časové okná, 2 - väčšia kapacita vozidiel/širšie časové okná), `n` reprezentuje počet miest v inštancii (2 pre veľkosť 200 miest, 4 pre 400 ...) a `x` je poradové číslo inštancie. Na obrázkoch 6.1, 6.2 a 6.3 sú príklady jednotlivých charakteristík úloh z dátovej sady. Keďže dátová sada je tvorená syntetickými dátami, výsledné vzdialenosti sú považované za univerzálnu jednotku vzdialenosti. Vzdialenosti medzi dvoma miestami sú brané ako euklidovská vzdialenosť dvoch bodov (vzorec 5.1).



Obr. 6.1: Vizualizácia charakteristiky problému z dátovej sady: charakter zhlukov - C (pre vstupný počet miest 200). Na obrázku vidno, že miesta sú zoskupené okolo seba, čím vytvárajú zhluky.



Obr. 6.2: Vizualizácia charakteristiky problému z dátovej sady: náhodný charakter - R (pre vstupný počet miest 200). Rozmiestnenie miest (modré a červené body) sú v priestore rozmiestnené náhodným spôsobom.



Obr. 6.3: Vizualizácia charakteristiky problému z dátovej sady: náhodne a zhlukovanie - RC (pre vstupný počet miest 200). Na tomto obrázku je vidieť kombinácia dvoch charakteristík. Niektoré miesta sú zoskupené v zhlukoch, niektoré sú rozmiestnené náhodne v priestore.

6.2 Experiment - porovnanie navrhnutých typov evolučných techník

Predmetom výskumu experimentu popísaného v tejto podkapitole, bolo použitie a porovnanie štyroch variánt evolučných algoritmov na riešenie problému VRPPD. Konkrétne sa jedná o varianty navrhnuté v tejto práci, popísané v kapitole 5. Jednou s variánt je evolučná stratégia - ES, s nastavením popísaným v podkapitole 5.3. Ďalšou je evolučná stratégia s použitím cielených presunov - ES-CM, tak ako bola popísaná v podkapitole 5.4. Ďalej evolučná stratégia s použitím cielených mutácií a ťažiskom trasy - ES-CMC, zo sekcie 5.5. Poslednou z variánt je genetický algoritmus s použitím cielených mutácií a ťažiskom trasy - GA-CMC, ktorý bol opísaný v podkapitole 5.6. Výsledkom experimentu je štatistická analýza evolučných behov programu pre všetky varianty, konvergenčná analýza použitých evolučných algoritmov a analýza výsledných riešení pre VRPPD.

Hodnota maximálneho počtu generácii `GENERATIONS` bola pre všetky spustenia v experimente nastavená na 100000. Nastavenie parametrov evolučných algoritmov je to, ktoré je uvedené v úvode kapitoly 5 v tabuľke 5.1. Parameter maximálnej dĺžky trasy vozidla, u tejto sady experimentov, bol nastavený vzhľadom na to, aby počet použitých vozidiel a hodnota celkovej vzdialenosti všetkých trás boli podobné z najlepšími dosiahnutými výsledkami pre dané inštancie (aj napriek tomu, že tie výsledky boli pre riešenie problému VRPPDTW). Tým pádom bola zvolená hodnota maximálnej dĺžky 250 pre inštancie L*1_2_*, zatiaľ čo pre inštancie L*2_2_* bola vybraná hodnota 300.

Jedným z výstupov experimentu je zhrnutie celkových vzdialenosti optimalizovaných trás, všetkých vozidiel, z 96 behov programu, pre uvedené varianty evolučných algoritmov na rôznych dátových sadách. Porovnanie výsledkov je vyobrazené v tabuľke 6.1. Hodnoty v tabuľke predstavujú univerzálnu jednotku vzdialenosti. Ako bolo spomenuté v úvode kapitoly, pre každé spustenie kombinácie inštancia-varianta EA, bol program spustený 96 krát. V tabuľke má každá bunka inštancie 3 riadky. Prvý riadok predstavuje najlepšie dosiahnuté riešenie zo všetkých 96 behov programu. Druhý riadok je hodnota mediánu všetkých behov programu, kde σ je smerodajná odchýlka. Tretí riadok bunky je hodnota najhoršieho výsledku riešenia spomedzi 96 behov programu.

Iný spôsob vizualizácie štatistických výsledkov experimentu je vo forme grafov, s dátami vyobrazenými pomocou boxplotov, zobrazených v súbore obrázkov 6.4. Opäť sú to vynesené dáta pre všetkých 96 behov programu. Y-ová osa grafov reprezentuje celkovú vzdialenosť všetkých trás, X-ová osa je použitá varianta EA a na vrchu grafu je názov inštancie.

Na grafoch z obrázku 6.5 je znázornený vývoj celkovej vzdialenosti trás v riešení najlepšieho jedinca v priebehu evolúcie, pre všetkých 96 behov. Silne oranžovou čiarou v grafe je zvýraznený beh programu, ktorý dosiahol najlepší výsledok zo všetkých behov, po 100000 generáciách. Vybraných pre zobrazenie v tejto práci je len niekoľko inštancií, pre ostatné inštancie bol priebeh obdobný.

V rámci experimentu bola taktiež uskutočnená podrobná analýza získaných najlepších riešení. Niektoré z týchto analýz najlepších riešení sú v tabuľkách 6.2, 6.3 a 6.4. V jednej tabuľke sú výsledky z jednej inštancie, pre štyri použité techniky EA. Každý riadok vyjadruje jednu trasu pre vozidlo, s hodnotou počtu navštívených miest je v danej trase a v zátvore prejdenú vzdialenosť vozidla v trase. Priama vizualizácie výsledných, optimalizovaných trás riešenia pre tieto prípady sú vyobrazené na obrázkoch 6.6, 6.7 a 6.8.

inštancie	ES	ES-CM	GA-CMC	ES-CMC
	2012	1704	1583	1536
LC1_2_1	2389, σ=132	1999, σ=100	1790, σ=90	1705, σ=96
	2670	2283	1972	1905
	2254	1846	1749	1692
LC1_2_2	2538, σ=150	2131, σ=100	1916, σ=83	1894, σ=89
	3027	2374	2103	2108
	2969	2392	2351	2237
LC2_2_1	3262, σ=124	2673, σ=112	2477, σ=72	2372, σ=69
	3610	2952	2671	2608
	2876	2230	2162	2139
LC2_2_2	3198, σ=125	2618, σ=115	2368, σ=84	2275, σ=68
	3511	2929	2544	2508
	3748	3215	3284	3203
LR1_2_1	4165, σ=137	3589, σ=127	3548, σ=109	3437, σ=100
	4504	3940	3815	3790
	3643	3064	3158	3058
LR1_2_2	4034, σ=143	3393, σ=131	3395, σ=140	3272, σ=105
	4356	3759	4385	3595
	4423	3622	3777	3676
LR2_2_1	4822, σ=138	4002, σ=127	4027, σ=115	3866, σ=111
	5090	4323	4311	4112
	5510	4590	4693	4584
LR2_2_2	5940, σ=164	4998, σ=133	5007, σ=162	4869, σ=154
	6249	5288	5431	5301
	3000	2524	2449	2356
LRC1_2_1	3289, σ=130	2748, σ=112	2617, σ=80	2542, σ=70
	3613	3035	2941	2724
	2877	2333	2228	2194
LRC1_2_2	3147, σ=127	2600, σ=111	2429, σ=91	2403, σ=92
	3450	2874	2652	2638
	3339	2734	2718	2595
LRC2_2_1	3621, σ=125	3030, σ=125	2939, σ=95	2814, σ=93
	3943	3298	3140	3079
	3636	2893	2826	2749
LRC2_2_2	3944, σ=127	3199, σ=124	3114, σ=110	3030, σ=112
	4283	3509	3396	3268

Tabuľka 6.1: Celkové vzdialenosti optimalizovaných trás z 96 nezávislých behov uvedených variánt EA na rôznych dátových sadách. Hodnoty sú bližšie nešpecifikované jednotky vzdialenosti. Každá bunka obsahuje tri hodnoty, prvá hodnota je celková trasa najlepšieho behu. Druhá hodnota je medián z 96 behov, pričom σ je hodnota smerodajnej odchýlky. Na tretom riadku je najhoršie nájdené riešenie zo všetkých behov pre variantu.

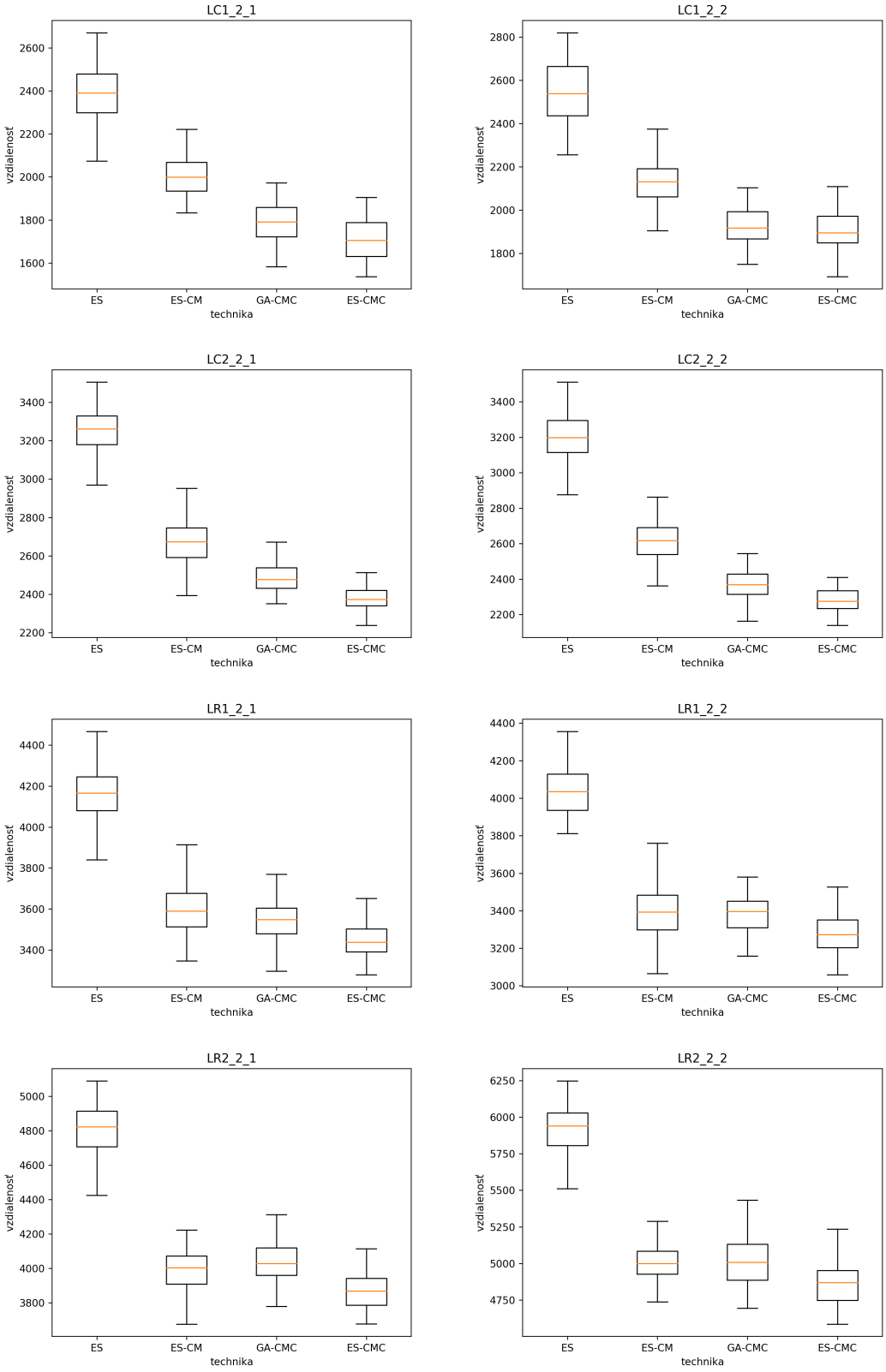
6.2.1 Zhodnotenie výsledkov experimentu

Z výsledkov experimentu v tabuľke 6.1 a v grafoch na obrázku 6.4, je zreteľné zlepšenie medzi variantou ES a variantami, ktoré využívajú mutácie s cielenými presunmi `guidedTransfer` (varianta ES-CM, GA-CMC a ES-CMC). Pri použití ES-CM sa zlepšenie vzhľadom k ES, pohybuje v rozsahu 18-24%. Varianty ES-CMC a GA-CMC, ktoré používajú mutáciu využívajúcu vlastnosti ťažiska trasy (`centroidTransfer`), vykazujú lepšie výsledky ako varianty nevyužívajúcej ťažiska trasy, čiže ES-CM a ES. Väčšie zlepšenia je dosiahnuté u inštancií s miestami rozmiestnenými v zhlukoch (inštancie LC*). V týchto prípadoch sa zlepšenie v metóde ES-CMC od ES-CM zlepšilo o 5-11%. Pri úlohách s náhodne rozmiestnenými miestami je zlepšenie stále viditeľné, ale pohybuje sa len na hranici okolo 1-5%. Z týchto pozorovaní sa dá usúdiť, že aplikovaním použitých techník pri mutácii (`guidedTransfer` a `centroidTransfer`) je algoritmus schopný nájsť kvalitnejšie riešenia. Ďalším pozorovaným poznatkom je to, že použitím vlastnosti ťažiska trasy u varianty ES-CMC a varianty GA-CMC, je hlavne prínosné v prípadoch, keď sú miesta zoskupené viac pri sebe. Malá zmena tiež prichádza v použití odlišných druhov evolučných algoritmov (genetický algoritmus/evolučná stratégia), čiže medzi variantou GA-CMC a ES-CMC, aj napriek tomu, že obe varianty používajú rovnaké mutačné operátory. Použitie genetického algoritmu GA-CMC je o 3-5% menej efektívne pri hľadaní kvalitných riešení, oproti použitiu evolučnej stratégie ES-CMC. Z analýzy porovnania výsledkov štyroch variánt návrhu evolučných algoritmov z tejto práce, možno usúdiť, že varianta evolučnej stratégie s použitím cielených presunov a ťažiskom trasy (ES-CMC) dosahuje najlepšie výsledky pri riešení VRPPD.

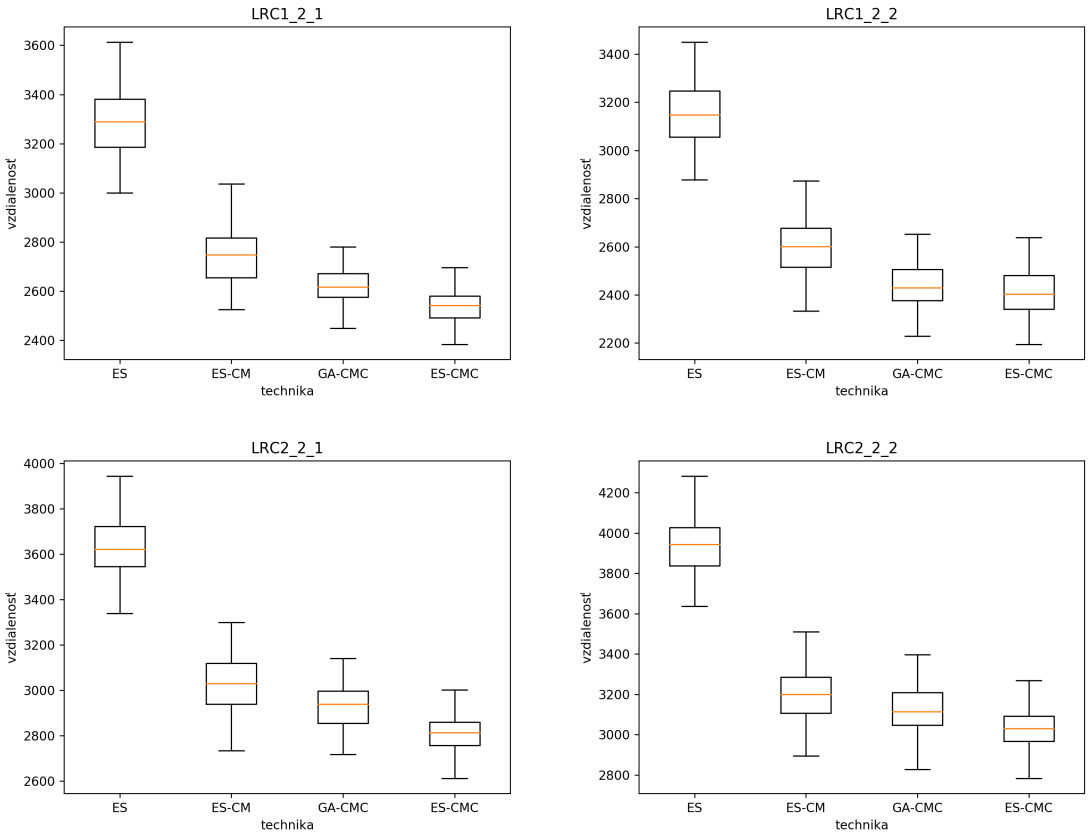
Z konvergenčnej analýzy[3] z obrázkov 6.5, je možné usúdiť, že hlavné zlepšenie u navrhnutých algoritmoch pri hľadaní riešení na VRPPD, sa pohybuje v prvých 5000 generáciách. V nasledovných generáciách medzi 5000-20000 je zlepšovanie značne pomalšie, ale stále prítomné pomerne dosť často. Pri použití techniky cielených presunov v algoritme (ES-CM, ES-CMC a GA-CMC) je kvalitné riešenie nájdené omnoho rýchlejšie, avšak je tam vidieť väčšia náchylnosť na uviaznutie v lokálnom minime. Na druhú stranu, je možné spozorovať fakt, že aj napriek uviaznutiu v lokálnom minime je algoritmus schopný v určitých momentoch z tohto lokálneho minima ujsť a priniesť ďalšie zlepšenie v riešení.

Analýzou konkrétnych výsledkov z tabuliek 6.2, 6.3 a 6.4 možno opäť pozorovať účinnosť zavedení mutačných operátorov `guidedTransfer` a `centroidTransfer`. Varianta ES, na rozdiel od ostatných variánt, potrebuje na vyriešenie tej istej úlohy väčší počet vozidiel. Z obrázkov 6.6 a 6.8 je vidno, že, niektoré výsledné optimalizované trasy, sú u variánt ES-CM, GA-CMC a ES-CMC rovnaké, aj napriek tomu, že spustenia optimalizácie boli nezávislé od seba a s rôznymi použitými technikami.

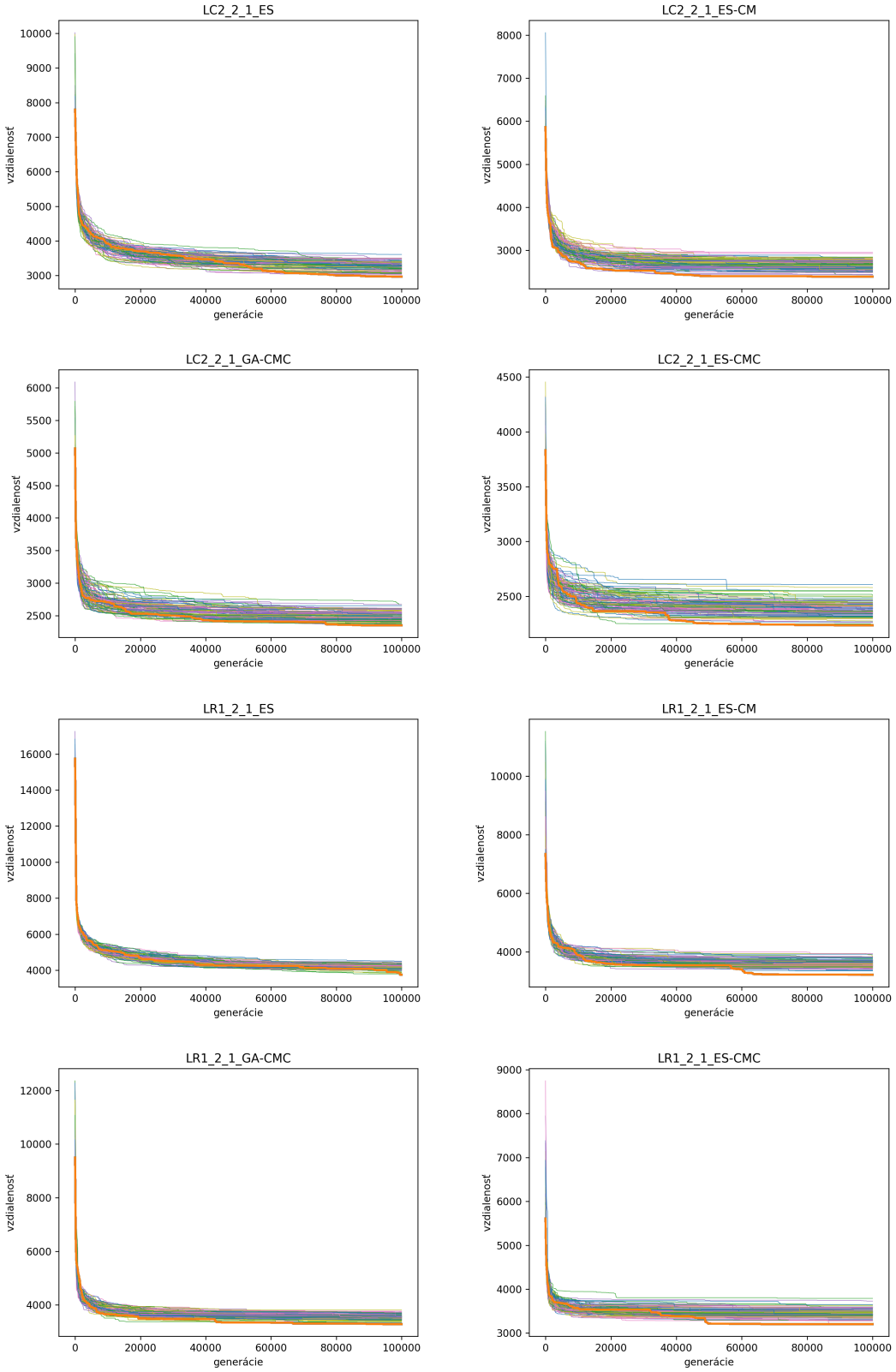
[3]Konvergenčnou analýzou sa myslí sledovanie aspektu správania algoritmu, ako sa postupne v priebehu evolúcie zlepšujú/konvergujú riešenia.



Obr. 6.4: Štatistické vyhodnotenie výsledkov evolúcie z pohľadu celkovej vzdialenosti nájdených trás pre skúmané nastavenia evolučných algoritmov. V grafoch je možné sledovať najlepšie, najhoršie a priemerné nájdené riešenie z 96 nezávislých behov (časť 1).



Obr. 6.4: Štatistické vyhodnotenie výsledkov evolúcie z pohľadu celkovej vzdialenosti nájdených trás pre skúmané nastavenia evolučných algoritmov. V grafoch je možné sledovať najlepšie, najhoršie a priemerné nájdené riešenie z 96 nezávislých behov (časť 2).



Obr. 6.5: Vývoj celkovej vzdialenosti trás v riešení najlepšieho jedinca generácie v priebehu evolúcie, pre 96 behov. Silne oranžovou čiarou v grafe je zvýraznený beh programu, ktorý na konci (generácia 100000) skončil s najlepším výsledkom.

LC2_2_2		ES	ES-CM	GA-CMC	ES-CMC
počet miest (vzdialenosť) - pre jednotlivé vozidlá	vozidlo 1	20 (257)	22 (274)	26 (266)	28 (293)
	vozidlo 2	14 (221)	22 (231)	12 (75)	36 (283)
	vozidlo 3	16 (199)	24 (258)	20 (238)	20 (226)
	vozidlo 4	18 (205)	22 (130)	30 (278)	12 (87)
	vozidlo 5	18 (230)	26 (283)	36 (283)	28 (280)
	vozidlo 6	20 (280)	28 (271)	28 (273)	28 (268)
	vozidlo 7	20 (217)	26 (279)	30 (294)	18 (199)
	vozidlo 8	18 (156)	30 (270)	22 (221)	28 (252)
	vozidlo 9	16 (244)	22 (230)	18 (232)	24 (246)
	vozidlo 10	12 (153)			
	vozidlo 11	22 (273)			
	vozidlo 12	20 (232)			
	vozidlo 13	16 (204)			
	celková vzdialenosť	2876	2230	2162	2139
	použité vozidlá	13	9	9	9
	priemerný počet miest na vozidlo	17.6	24.6	24.6	24.6
	priemerná vzdialenosť na vozidlo	220.8	247.3	240.0	237.1

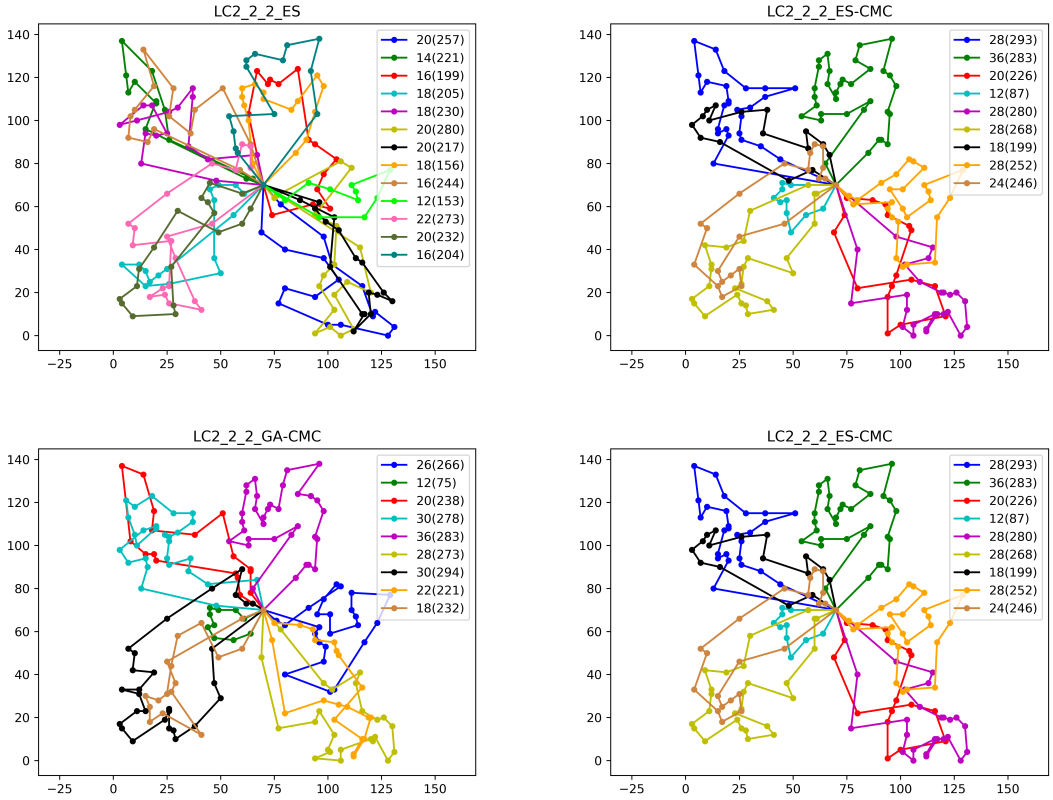
Tabuľka 6.2: Riešenia najlepších behov programu pre dátovú inštanciu LC2_2_2 u štyroch variánt EA. Hodnota u každého vozidla značí počet navštívených miest v trase pričom v zátvorke je hodnota vzdialenosti prejdenej v danej trase.

LR2_2_2		ES	ES-CM	GA-CMC	ES-CMC
počet miest (vzdialenosť) - pre jednotlivé vozidlá	vozidlo 1	14 (282)	14 (298)	18 (292)	20 (292)
	vozidlo 2	10 (268)	4 (297)	14 (265)	14 (265)
	vozidlo 3	14 (254)	14 (292)	14 (287)	10 (292)
	vozidlo 4	16 (269)	14 (292)	12 (281)	16 (297)
	vozidlo 5	14 (287)	12 (281)	8 (238)	16 (294)
	vozidlo 6	10 (200)	18 (296)	18 (283)	18 (295)
	vozidlo 7	10 (190)	16 (298)	4 (297)	20 (292)
	vozidlo 8	12 (286)	20 (291)	8 (262)	12 (288)
	vozidlo 9	14 (187)	10 (216)	12 (274)	12 (280)
	vozidlo 10	14 (211)	10 (293)	16 (249)	16 (233)
	vozidlo 11	10 (278)	20 (294)	18 (297)	16 (298)
	vozidlo 12	12 (280)	28 (298)	14 (292)	22 (272)
	vozidlo 13	8 (299)	10 (252)	16 (276)	12 (289)
	vozidlo 14	14 (222)	10 (297)	14 (296)	12 (294)
	vozidlo 15	8 (271)	18 (290)	14 (224)	4 (297)
	vozidlo 16	14 (292)	16 (299)	16 (288)	14 (298)
	vozidlo 17	14 (291)		20 (284)	
	vozidlo 18	12 (253)			
	vozidlo 19	8 (297)			
	vozidlo 20	10 (284)			
	vozidlo 21	6 (298)			
	celková vzdialenosť	5510	4590	4693	4584
	použité vozidlá	21	16	17	16
	priemerný počet miest na vozidlo	11.6	14.6	13.8	14.6
	priemerná vzdialenosť na vozidlo	261.8	286.5	275.5	286.0

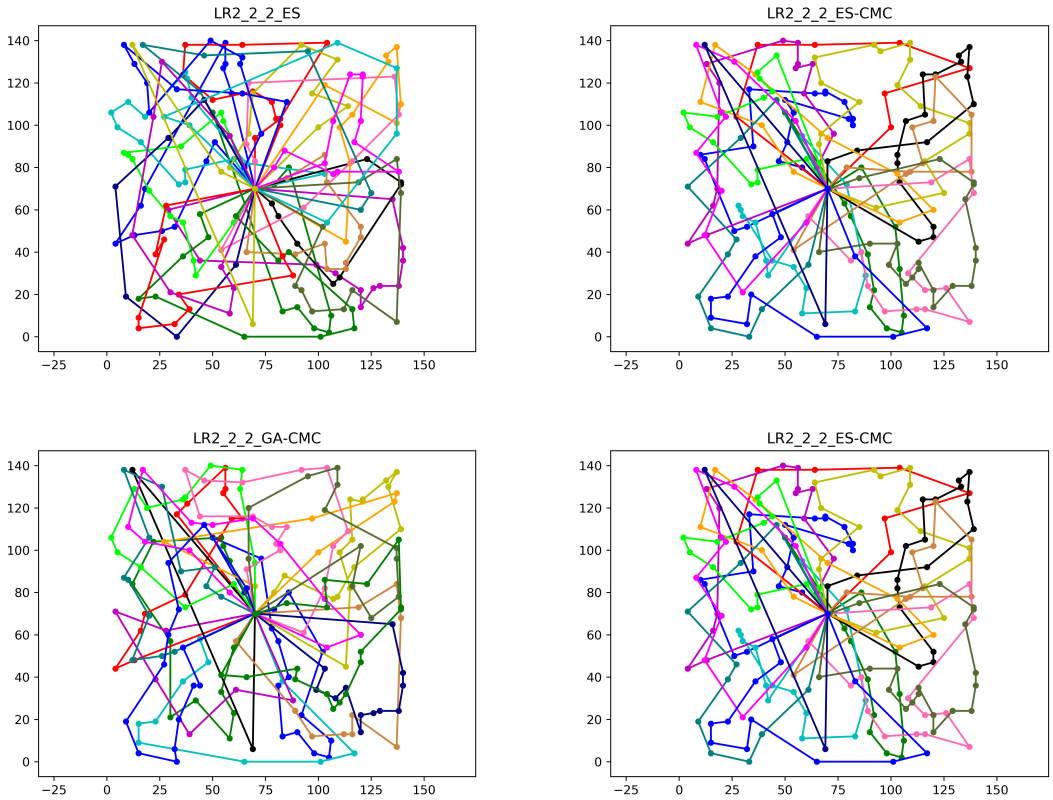
Tabuľka 6.3: Riešenia najlepších behov programu pre dátovú inštanciu LR2_2_2 u štyroch variánt EA. Hodnota u každého vozidla značí počet navštívených miest v trase pričom v zátvorke je hodnota vzdialenosti prejdenej v danej trase.

LRC2_2_2		ES	ES-CM	GA-CMC	ES-CMC
počet miest (vzdialenosť) - pre jednotlivé vozidlá	vozidlo 1	12 (231)	22 (290)	20 (197)	38 (287)
	vozidlo 2	16 (167)	24 (279)	20 (276)	18 (262)
	vozidlo 3	12 (247)	14 (180)	24 (260)	16 (179)
	vozidlo 4	14 (225)	24 (243)	20 (273)	18 (208)
	vozidlo 5	8 (109)	26 (290)	20 (265)	22 (255)
	vozidlo 6	18 (220)	22 (273)	28 (275)	24 (272)
	vozidlo 7	18 (256)	18 (290)	20 (240)	12 (198)
	vozidlo 8	14 (206)	18 (284)	22 (243)	20 (241)
	vozidlo 9	16 (269)	24 (246)	20 (295)	14 (276)
	vozidlo 10	12 (132)	20 (263)	14 (206)	24 (294)
	vozidlo 11	14 (202)	14 (249)	18 (293)	20 (272)
	vozidlo 12	14 (214)			
	vozidlo 13	14 (212)			
	vozidlo 14	8 (161)			
	vozidlo 15	18 (294)			
	vozidlo 16	14 (262)			
	vozidlo 17	16 (222)			
	celková vzdialenosť	3636	2893	2826	2749
	použité vozidlá	17	11	11	11
	priemerný počet miest na vozidlo	14.0	20.5	20.5	20.5
	priemerná vzdialenosť na vozidlo	213.4	262.4	256.6	249.4

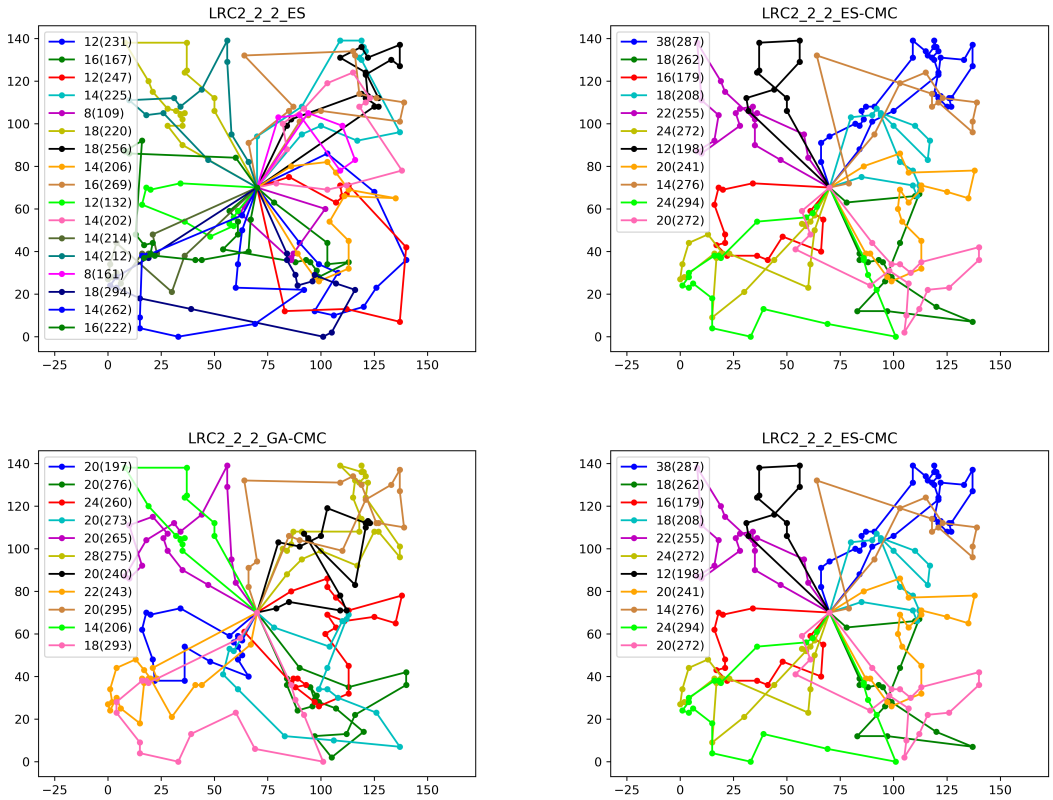
Tabuľka 6.4: Riešenia najlepších behov programu pre dátovú inštanciu LRC2_2_2 u štyroch variánt EA. Hodnota u každého vozidla značí počet navštívených miest v trase pričom v zátvorke je hodnota vzdialenosti prejdenej v danej trase.



Obr. 6.6: Vizualizácia optimalizovaných trás, najlepšieho nájdeného riešenia z 96 behov pre každú variantu EA, na vstupnej inštancii LC2_2_2. Hodnoty v legende vyjadrujú pre každú trasu počet navštívených miest a v zátvorke celkovú prejdenú vzdialenosť vozidla na trase.



Obr. 6.7: Vizualizácia optimalizovaných trás, najlepšieho nájdeného riešenia z 96 behov pre každú variantu EA, na vstupnej inštancii LR2_2_2.



Obr. 6.8: Vizualizácia optimalizovaných trás, najlepšieho nájdeného riešenia z 96 behov pre každú variantu EA, na vstupnej inštancii LRC2_2_2. Hodnoty v legende vyjadrujú pre každú trasu počet navštívených miest a v zátvorke celkovú prejdenú vzdialenosť vozidla na trase.

6.3 Experiment - porovnanie môjho riešenie s OR-Tools

Zameranie tohto experimentu pojednáva o overení funkčnosti navrhnutého algoritmu pre riešenie VRPPD. V experimente sa porovnávajú dosiahnuté výsledky získané použitím algoritmu ES-CMC (opísaného v podkapitole 5.5) s výsledkami získanými nástrojom OR-Tools[4] (nástroj pre riešenie rôznych optimalizačných úloh, využívaný v praxi, predstavený v sekcii 2.3.4). OR-Tools bol nastavený podľa potreby tejto práce, bol v ňom namodelovaný problém VRPPD s kapacitou vozidiel a obmedzením maximálnej dĺžky trasy pre jedno vozidlo. OR-Tools využíva riešiteľ CP-SAT, ktorý je predvolený a doporučovaný. Pre každú kombináciu inštancie a maximálnej dĺžky trasy bol ES-CMC spustený 96 krát, ako to je písané v úvode kapitoly 6. Beh OR-Tools je deterministický pretože hodnota seedu riešiteľa CP-SAT je nastavená na predvolenú hodnotu (oficiálne doporučované). Dôsledkom toho, viaceré spustenie OR-Tools s rovnakým vstupom, vracia rovnaký výsledok.

V tabuľkách 6.5 a 6.6 je zobrazené porovnanie riešení vyjadrených v podobe celkovej prejdenej vzdialenosti v nájdených trasách (v bližšie nešpecifikovaných jednotkách dĺžky), s konvenčným nástrojom OR-Tools. Vyhodnotenie bolo uskutočnené pomocou algoritmu ES-CMC, ktorý sa v porovnaní s ostatnými metódami (porovnanie z podkapitoly 6.2) ukázal ako najvhodnejší. Tabuľka 6.5 vyhodnocuje výsledky nad dátovou sadou popísanou v podkapitole 6.1, pre veľkosti problému s 100, 200, 400 a 600 mestami. V tabuľke 6.6 sú vyhodnotené inštancie s väčším počtom miest - 800 a 1000. Pre potreby porovnania ES-CMC s OR-Tools bolo ďalej zavedené kritériom maximálnej dĺžky trasy. Porovnávajú sa spustenia pre rôzne hodnoty maximálnej dĺžky trasy. Pre dátové sady s nižším počtom miest (100-600) sú porovnávané výsledky pre hodnoty maximálnej dĺžky trasy 300, 400 a 500. Pre inštancie s väčším počtom miest (800 a 1000) sú hodnoty maximálnej dĺžky trasy nastavené na 600, 700 a 800. Každá bunka v tabuľke pre výsledky algoritmu ES-CMC predstavuje najlepšie nájdené riešenie, medián riešení a najhoršie nájdené riešene, spomedzi 96 nezávislých behov programu. Hodnota výsledku riešenia je vyjadrená počtom použitých vozidiel (hodnota v zátvorke) a sumou všetkých trás v riešení (v bližšie nešpecifikovanej jednotke vzdialenosti). V časti tabuľky s výsledkami OR-Tools je v každej bunke počet použitých vozidiel (v zátvorke) a vzdialenosť všetkých trás. Bunka tiež obsahuje percentuálnu odlišnosť najlepšieho nájdeného riešenia metódou ES-CMC a riešenia získaného pomocou OR-Tools. Zelené hodnoty percent znamenajú, že technika ES-CMC našla lepší výsledok, naopak červené hodnoty znamenajú, že lepší výsledok bol získaný pomocou nástroja OR-Tools. Štatistické vyhodnotenie výsledkov, získanými pomocou ES-CMC sú tiež zobrazené vo forme grafov, v súbore obrázkov 6.9. Do grafov je ešte vynesená hodnota získaná pomocou OR-Tools, vyznačená modrým bodom. Nastavenie maximálneho počtu generácii zavisí od veľkosti danej inštancie od 100000 až po 180000.

6.3.1 Zhodnotenie výsledkov experimentu

Výsledky experimentu ukazujú, že navrhnutý spôsob ES-CMC na riešenie VRPPD s maximálnou dĺžkou trasy, dokáže pre niektoré vstupné úlohy vrátiť kvalitné riešenia. Z tabuľky 6.5 je jasné, že niektoré experimenty v tomto ohľade prekonávajú výsledky získané pomocou nástroja OR-Tools. Inštancie so zadanou maximálnou dĺžkou trasy, u ktorých je percentuálne vyjadrený rozdiel negatívny (zelený) znamená, že metódou navrhnutou v tejto práci sa získalo lepšie riešenie. V prípade inštancie LR1_2_1, bola metóda ES-CMC schopná nájsť lepšie riešenie pre všetky zadané maximálne hodnoty trasy. V experimente

[4] https://developers.google.com/optimization

s inštanciou LRC1_2_1 so zadanou maximálnou dĺžkou trasy 300, bol algoritmus ES-CMC schopný nájsť lepšie riešenie ako OR-Tools vo všetkých 96 behov programu a najlepšie nájdený výsledok je až o 6,2% lepší ako výsledok získany pomocou OR-Tools. Najväčší rozdiel týchto dvoch použitých metód, v prospech ES-CMC bol dosiahnutý u inštancie LRC1_4_1 s maximálnou dĺžkou trasy 300. Kvalita riešenia je v tomto prípade až o vyše 10% lepšia.

Na druhú stranu, s niektorými úlohami si však navrhnutá metóda ES-CMC poradila nie až tak dobre. Treba tiež podotknúť negatívnu skutočnosť, že na získanie výsledných riešení potreboval algoritmus ES-CM 5-15 krát väčší výpočetný čas s porovnaním od OR-Tools a pre každý experiment bol program spustený až 96 krát. Ak by program bežal kratšiu dobu, dosiahnuté výsledky by boli o niečo menej kvalitné. Ako je vidieť z konvergenčnej analýzy na obrázku 6.5, hlavné zlepšenie síce prichádza na začiatku, v krátkom čase, ale pre dosiahnutie porovnateľných výsledkov, zlepšenie často prichádza až neskôr. Navrhnutý algoritmus si s porovnaním z OR-Tools viedol lepšie hlavne v úlohách s menším počtom miest, ktoré boli rozmiestnené náhodným spôsobom. Nedostatky riešenia sú zreteľné hlavne v úlohách s charakterom zhlukovaných miest (LC* inštancie). Aj napriek zavedeniu techniky využívajúcej ťažisko trás na určovanie vhodného výberu a vkladanie požiadaviek na prepravu, má algoritmus problém s úlohami takéhoto charakteru, respektívne OR-Tools si s úlohami tohto charakteru radí veľmi dobre. Horšie výsledky v týchto úlohách sú záskané aj pri malích inštanciách, a u veľký dosahujú horšie výsledke na rozdiel od OR-Tools aj o 27-40% (u najvačšej inštancii LC1_10_1). Jedným z dôvodom, prečo algoritmus ES-CMC zaostáva v týchto problémoch s charakterom zhlukovaných miest je to, že pri vytváraní nových riešení sa vykonáva príliš veľká zmena z rodiča na potomka. Tieto úlohy potrebujú presné a malé úpravy poradí trás v rámci jednej trás. Riešením by mohlo byť to, že by sa v poslednej generácie, najlepší jedinec ešte doladil. Napríklad spôsobom, že by sa samostatne optimalizovala každá trasa nejakou inou metaheuristikou (napríklad tabu prehľadávaním).

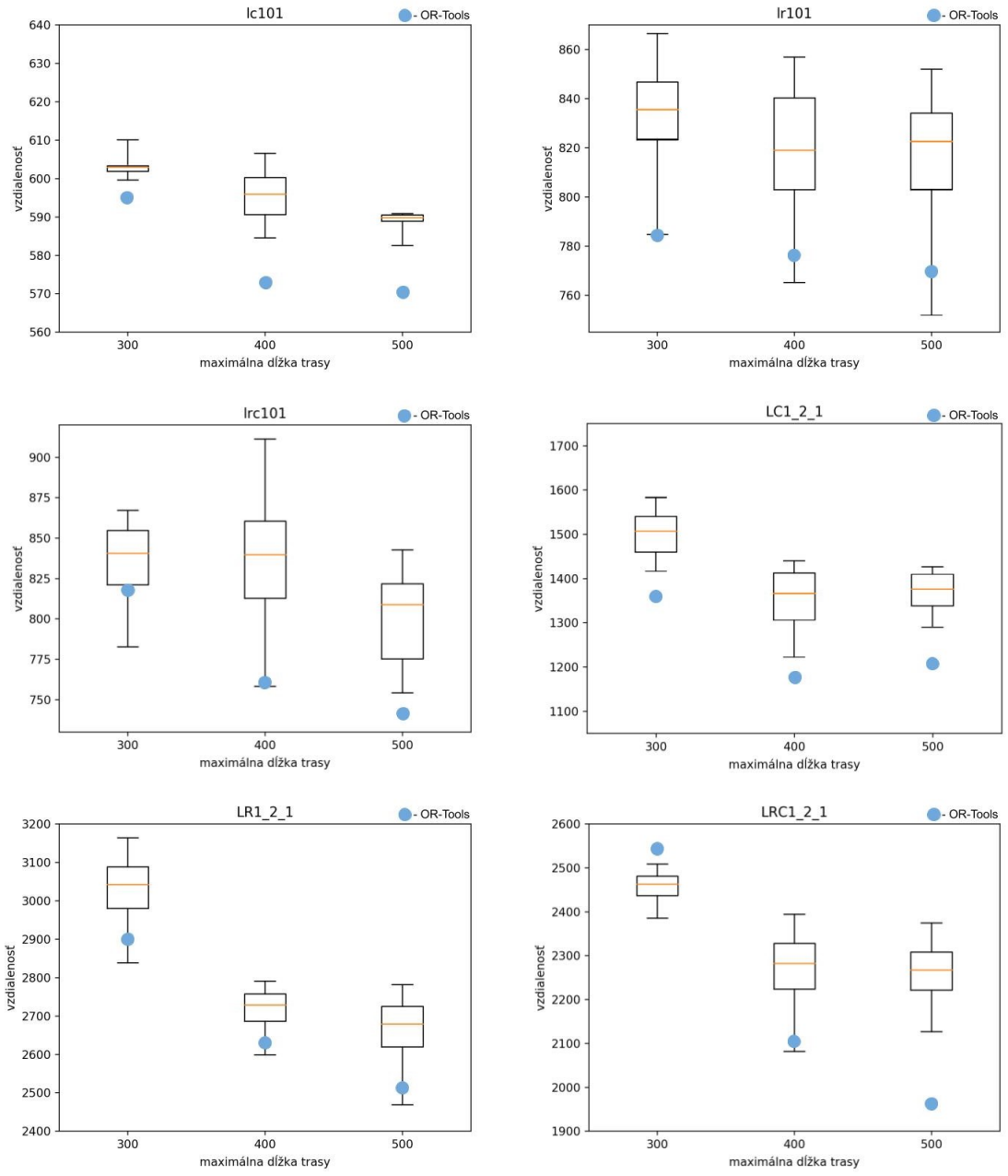
Zaujímavým vypozorovaným poznatkom z experimentov je to, že navrhnutý algoritmus ES-CMC funguje lepšie pri väčšom obmedzení na maximálnu dĺžku trasy. Dokonca v prípade inštancie LRC_1_2_1, bol algoritmus ES-CMC schopný nájsť lepšie riešenie pri väčšom obmedzení na maximálnu trasy. Zavedením prísnejšieho obmedzenia sa problém navonok javí, že sa jeho náročnosť zvyšuje, ale pravdou je, že sa tým prehľadávaný priestor zmenšuje, čiže je menej platných riešení na danú úlohu. S menším prehľadávaným priestorom je aj menšia pravdepodobnosť uviaznutia v lokálnom minime.

	algoritmus ES-CMC			OR-Tools		
inštancie	**300**	**400**	**500**	**300**	**400**	**500**
lc101	(3) 599 (3) 604 (3) 614	(2) 584 (3) 597 (3) 609	(2) 581 (2) 589 (2) 591	(3) 595 0,60%	(2) 573 1,90%	(2) 570 1,90%
lr101	(3) 784 (5) 836 (6) 866	(3) 765 (4) 819 (5) 856	(2) 752 (3) 824 (4) 851	(3) 784 0,00%	(3) 778 -1,70%	(2) 770 -2,40%
lrc101	(4) 782 (4) 841 (4) 867	(3) 758 (4) 839 (5) 911	(3) 754 (4) 809 (5) 842	(3) 819 -4,60%	(3) 761 -0,40%	(2) 743 1,40%
LC1_2_1	(7) 1430 (9) 1504 (9) 1584	(5) 1237 (8) 1370 (9) 1449	(6) 1316 (7) 1380 (8) 1440	(5) 1362 4,90%	(4) 1188 4,10%	(4) 1208 8,90%
LR1_2_1	(11) 2838 (13) 3042 (14) 3164	(8) 2517 (10) 2728 (10) 2790	(7) 2443 (9) 2679 (10) 2781	(12) 2899 -2,20%	(8) 2634 -4,50%	(7) 2526 -3,30%
LRC1_2_1	(10) 2385 (10) 2462 (10) 2508	(7) 2081 (8) 2282 (9) 2394	(7) 2126 (8) 2267 (9) 2374	(10) 2542 -6,20%	(7) 2103 -1,10%	(5) 1976 7,50%
LC1_4_1	(16) 3909 (19) 4255 (20) 5724	(12) 3282 (15) 3790 (16) 3930	(10) 3063 (14) 3417 (15) 3523	(14) 3728 4,80%	(10) 3049 7,60%	(6) 2537 20,70%
LR1_4_1	(27) 7394 (28) 7721 (29) 7947	(18) 5822 (20) 6244 (22) 6709	(14) 5449 (16) 5718 (17) 5834	(28) 7396 -0,10%	(17) 5577 4,30%	(14) 5257 3,60%
LRC1_4_1	(22) 6068 (25) 6608 (26) 6803	(16) 5100 (18) 5406 (20) 5708	(15) 4864 (16) 5144 (17) 5282	(26) 6751 -10,20%	(14) 4742 7,50%	(11) 4452 9,20%
LC1_6_1	- - -	(23) 7340 (26) 7831 (27) 8058	(19) 6400 (22) 6806 (23) 7106	- -	(19) 6813 7,70%	(13) 5262 21,60%
LR1_6_1	- - -	- - -	(29) 12867 (31) 13627 (32) 13816	- -	- -	(31) 13164 -2,30%
LRC1_6_1	- - -	- - -	(28) 10859 (29) 11424 (31) 11628	- -	- -	(24) 10062 7,90%

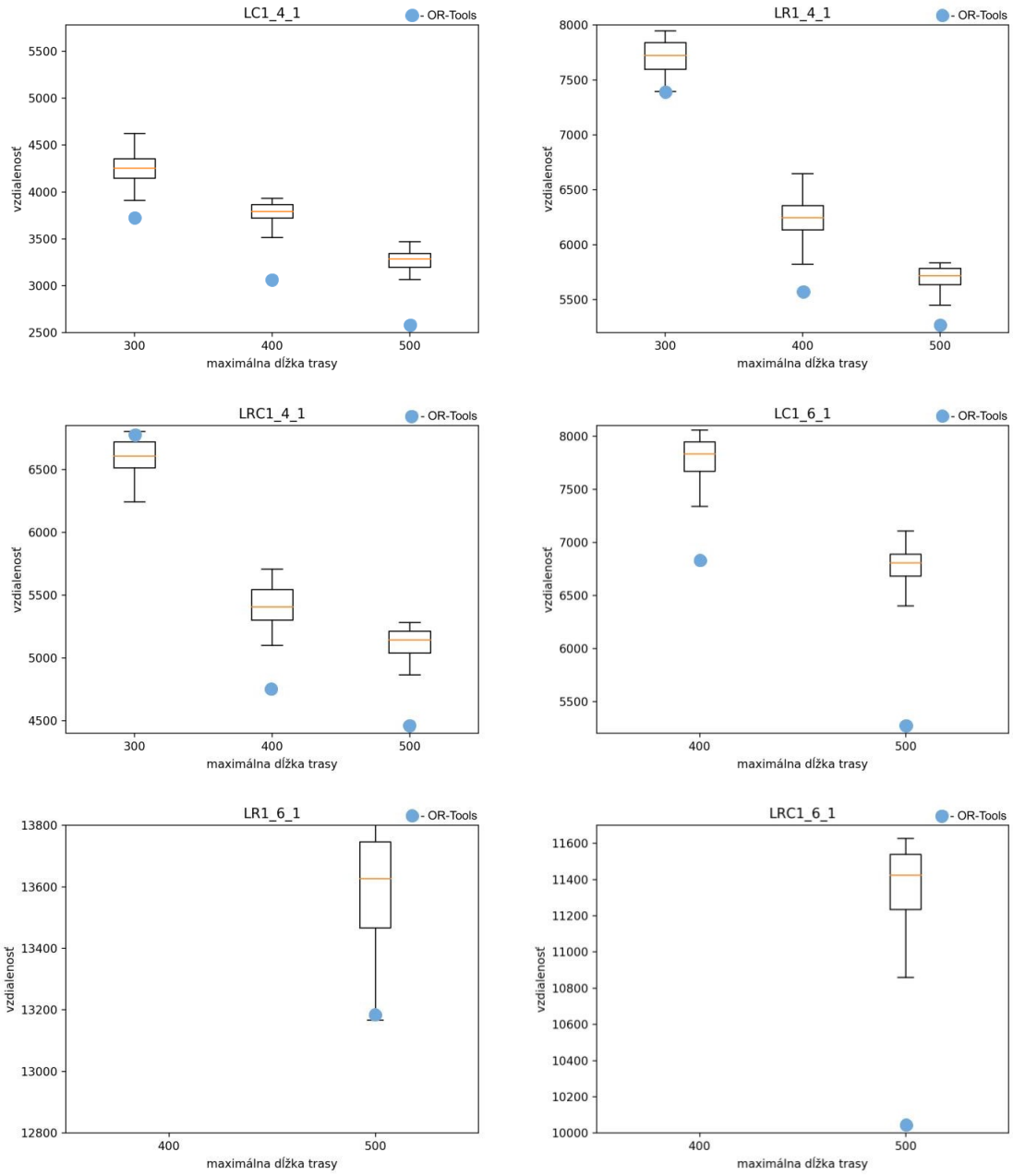
Tabuľka 6.5: Porovnanie výsledkov získanými použitím algoritmu ES-CMC a s použitím nástroja OR-Tools. Výsledky sú vyjadrené počtom použitých vozidiel (hodnota v zátvorke) a sumou všetkých trias v riešení (hodnota v jednotke vzdialenosti). V časti tabuľky s výsledkami od OR-Tools je tiež zobrazená percentuálna odlišnosť najlepšieho nájdeného riešenia metódou ES-CMC a riešenia získaného pomocou OR-Tools. Zelené hodnoty znamenajú lepší výsledok od ES-CMC, červené hodnoty znamenajú lepší výsledok nájdený nástrojom OR-Tools. Bunky ktoré neobsahujú žiadne hodnoty značia, že pre dané nastavenie nebolo nájdené riešenie (časť 1).

	algoritmus ES-CMC			**OR-Tools**		
inštancie	**600**	**700**	**800**	**600**	**700**	**800**
LC1_8_1	(28) 11700 (32) 12673 (33) 13350	(24) 9138 (26) 10005 (29) 11135	(20) 8950 (24) 9926 (27) 10524	(17) 9196 27,20%	(11) 7160 27,60%	(10) 6881 30,00%
LR1_8_1	(46) 25771 (49) 26937 (50) 27578	(36) 21723 (38) 22559 (40) 22923	(35) 20648 (36) 21721 (39) 22182	(46) 24596 4,70%	(36) 20660 5,10%	(28) 18484 11,70%
LRC1_8_1	(40) 21140 (42) 21907 (42) 22439	(31) 17941 (35) 18837 (37) 19791	(27) 16180 (31) 17754 (34) 18560	(34) 18719 12,90%	(30) 17165 4,50%	(25) 15709 2,90%
LC1_10_1	- - -	(39) 20315 (43) 22888 (44) 24888	(33) 17301 (36) 18458 (39) 19259	- -	(25) 16022 26,70%	(17) 12342 40,10%
LR1_10_1	- - -	- - -	(45) 31862 (48) 33747 (49) 34367	- -	- -	(42) 29324 8,60%
LRC1_10_1	- - -	- - -	(42) 28528 (46) 29838 (48) 30483	- -	- -	(39) 26541 7,40%

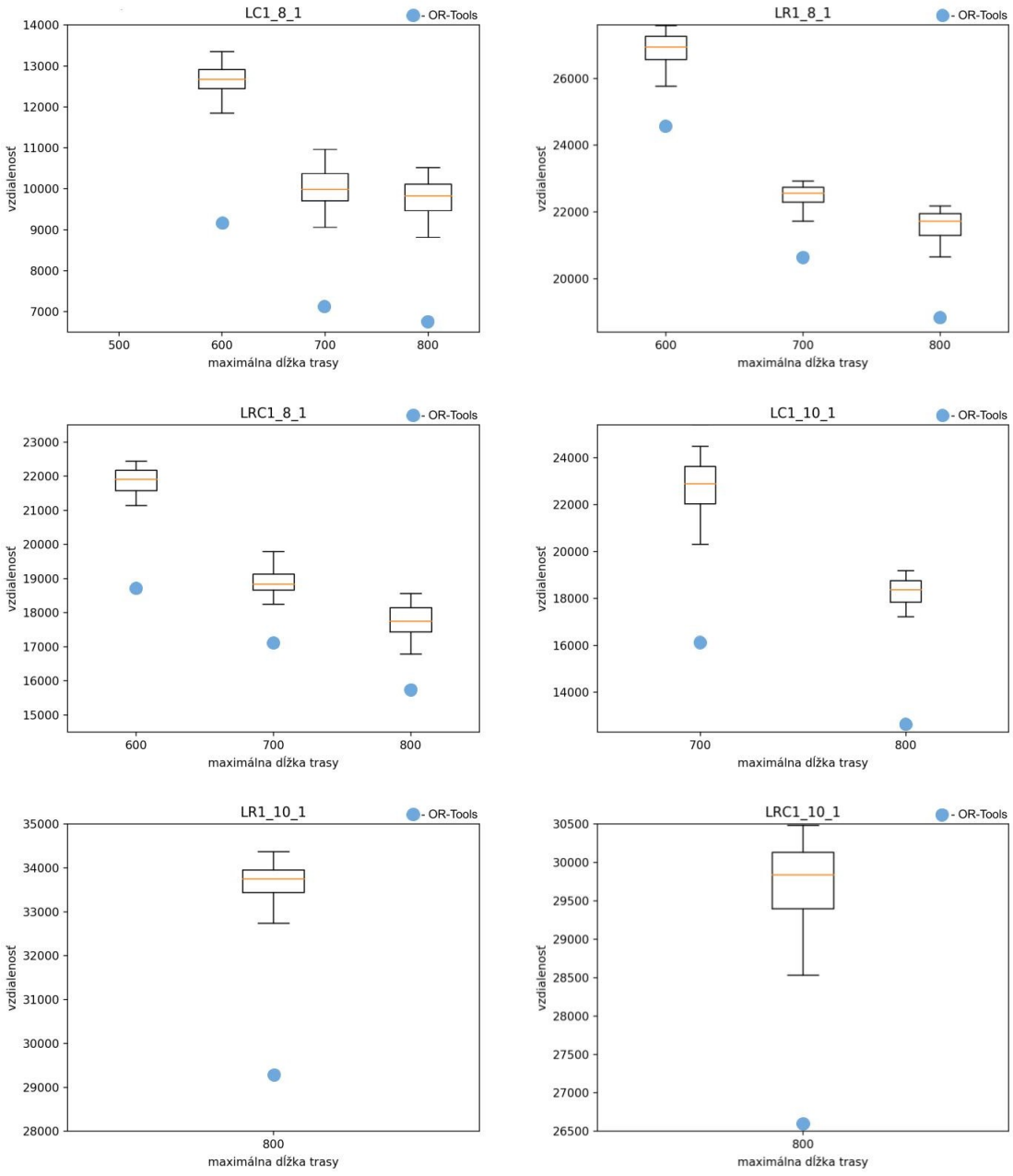
Tabuľka 6.6: Porovnanie výsledkov získanými použitím algoritmu ES-CMC a s použitím nástroja OR-Tools. Výsledky sú vyjadrené počtom použitých vozidiel (hodnota v zátvorke) a sumou všetkých trias v riešení (hodnota v jednotke vzdialenosti). V časti tabuľky s výsledkami od OR-Tools je tiež zobrazená percentuálna odlišnosť najlepšieho nájdeného riešenia metódou ES-CMC a riešenia získaného pomocou OR-Tools. Zelené hodnoty znamenajú lepší výsledok od ES-CMC, červené hodnoty znamenajú lepší výsledok nájdený nástrojom OR-Tools. Bunky ktoré neobsahujú žiadne hodnoty značia, že pre dané nastavenie nebolo nájdené riešenie (časť 2).



Obr. 6.9: Zobrazenie štatistického vyhodnotenia výsledkov, získaných pomocou algoritmu ES-CMC a OR-Tools, z pohľadu celkovej vzdialenosti všetkých trás (os Y) v riešení na rôznych dátových sadách pre viacero nastavení maximálnej dĺžky trasy (os X). Hodnota získaná pomocou nástroja OR-Tools je vyznačená modrou bodkou (časť 1).



Obr. 6.9: Zobrazenie štatistického vyhodnotenia výsledkov, získaných pomocou algoritmu ES-CMC a OR-Tools, z pohľadu celkovej vzdialenosti všetkých trás (os Y) v riešení na rôznych dátových sadách pre viacero nastavení maximálnej dĺžky trasy (os X). Hodnota získaná pomocou nástroja OR-Tools je vyznačená modrou bodkou (časť 2).



Obr. 6.9: Zobrazenie štatistického vyhodnotenia výsledkov, získaných pomocou algoritmu ES-CMC a OR-Tools, z pohľadu celkovej vzdialenosti všetkých trás (os Y) v riešení na rôznych dátových sadách pre viacero nastavení maximálnej dĺžky trasy (os X). Hodnota získaná pomocou nástroja OR-Tools je vyznačená modrou bodkou (časť 3).

Kapitola 7

Záver

Táto práca sa zaoberá optimalizačnou problematikou plánovania nákladnej prepravy. Konkrétne bol na riešenie vybraný problém smerovania vozidiel s vyzdvihnutím a doručením. Ako spôsob riešenia problému bol zvolený evolučný prístup. Podarilo sa navrhnúť reprezentáciu VRPPD, navrhnúť štruktúru evolučného algoritmu pre riešenie danej optimalizačnej úlohy a následne daný algoritmus implementovať. Motiváciou pre vypracovanie tejto práce bol fakt, že problém smerovania vozidiel je veľmi rozšírený optimalizačný problém s veľkou mierov aplikovateľnosti do praxe. Snaha bola navrhnúť novú metódu na riešenie daného problému a vniesť nové myšlienky ako sa na úlohu pozerať. Celkovo bola v práci na riešenie úlohy odskúšaná dvojica evolučných algoritmov, genetický algoritmus a evolučná stratégia.

7.1 Zhodnotenie výsledkov práce

Kvalita dosiahnutých výsledkov aplikovania navrhnutých metód na problém je pomerne dosť zmiešaná. V určitých prípadoch vie navrhnutý algoritmus nájsť veľmi kvalitné riešenia, ale vo niektorých prípadoch je kvalita riešení podpriemerná. Analýzou výsledkov experimentov je možné zhodnotiť, že použité metódy sú schopné veľmi rýchlo zlepšovať nájdené riešenie počas behu, ale zlepšovanie je príliš priamočiare, kvôli čomu riešenie často uviazne v lokálnom minime. Na druhú stranu, aj lokálne minimum môže predstavovať veľmi kvalitný výsledok, ktorý je postačujúci.

Problémom je, že navrhnuté metódy sa snažia pri jednom kroku mutácie nájsť čo najlepšie možné riešenie, čo má za následok, že zmena z rodiča na potomka môže byť príliš veľká a tým pádom zanedbáva množstvo riešení, ktoré môžu viesť k lepším výsledkom. Práve zle nastavená rovnováha cieleného hľadania nových riešení a ponechania "hlúpej" náhodnej zmeny chromozómu pri generovaní nových riešení má za následok opomínanie riešení, ktoré môžu viesť k lepšiemu výsledku. Práve táto nerovnováha nechcene vedie algoritmus do uviaznutia v lokálnom minime.

Riešením na tieto problémy by mohlo byť pridanie určitých aspektov adaptivity algoritmu, čím by sa regulovali evolučné parametre algoritmu podľa aktuálnej situácie riešení. Tak by sa mohla regulovať veľkosť kroku zmeny rodiča na potomka. Ďalším nevyhnutným zlepšením, čo navrhnutá metóda potrebuje, je zavedenie lepších praktík diverzifikácie riešení, čiže aby boli jednotliví jedinci v populácii od seba odlišní, čo by malo za následok rýchlejšie prehľadávanie stavového priestoru a tiež by to predstavovalo lepšiu prevenciu pred uviaznutím v lokálnom minime. Aplikovanie operátoru kríženia by taktiež mohlo pomôcť s problémom priamočiarosti prehľadávania riešení, ktoré sa často "pozerá len jedným sme-

rom". Operátor kríženia by mohol (ale nemusel) priniesť schopnosť "skoku" prehľadávaného okolia riešenia v stavovom priestore z jedného miesta na druhé, čo by opäť mohlo viesť k lepšej vlastnosti úniku z lokálneho minima.

Myšlienky predstavené v tejto práci ukázali, že majú potenciál, avšak pre ich správne fungovanie im chýba lepšie zasadenie do väčšieho kontextu lepšej realizácie návrhu. Za zváženie by stálo aplikovať myšlienky z práce, rozšírené o spomenuté návrhy na opravu ich nedostatkov, do iného druhu metaheuristiky, napríklad tabu prehľadávania.

Literatúra

[1] Arbelaitz, O., Rodriguez, C. a Zamakola, I. Low cost parallel solutions for the VRPTW optimization problem. In: IEEE. *Proceedings International Conference on Parallel Processing Workshops*. 2001, s. 176–181.

[2] Bäck, T., Fogel, D. B. a Michalewicz, Z. *Handbook of evolutionary computation*. CRC Press, 1997.

[3] Baker, B. M. a Ayechew, M. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*. Elsevier. 2003, zv. 30, č. 5, s. 787–800.

[4] Baldacci, R., Bartolini, E. a Mingozzi, A. An exact algorithm for the pickup and delivery problem with time windows. *Operations research*. INFORMS. 2011, zv. 59, č. 2, s. 414–426.

[5] Bell, J. E. a McMullen, P. R. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*. 2004, zv. 18, s. 41–48.

[6] Brabazon, A., O'Neill, M. a McGarraghy, S. *Natural computing algorithms*. Springer, 2015.

[7] Bullnheimer, B., Hartl, R. F. a Strauss, C. Applying the ant system to the vehicle routing problem. In: *Meta-heuristics*. Springer, 1999, s. 285–296.

[8] Clarke, G. a Wright, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*. Informs. 1964, zv. 12, č. 4, s. 568–581.

[9] Dantzig, G. B. a Ramser, J. H. The truck dispatching problem. *Management science*. Informs. 1959, zv. 6, č. 1, s. 80–91.

[10] Fisher, M. L. a Jaikumar, R. A generalized assignment heuristic for vehicle routing. *Networks*. Wiley Online Library. 1981, zv. 11, č. 2, s. 109–124.

[11] Garcia Najera, A. a Gutiérrez Andrade, M. Á. An evolutionary approach to the multi-objective pickup and delivery problem with time windows. In: IEEE. *2013 IEEE Congress on Evolutionary Computation*. 2013, s. 997–1004.

[12] Holland, J. H. et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[13] Kvasnička, V., Pospíchal, J. a Tiňo, P. *Evolučné algoritmy*. Slovenská technická univerzita, 2000.

[14] Laporte, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*. Elsevier. 1992, zv. 59, č. 3, s. 345–358.

[15] Lau, H. C., Sim, M. a Teo, K. M. Vehicle routing problem with time windows and a limited number of vehicles. *European journal of operational research*. Elsevier. 2003, zv. 148, č. 3, s. 559–569.

[16] Pang, K.-W. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Systems with Applications*. Elsevier. 2011, zv. 38, č. 9, s. 11939–11946.

[17] Rechenberg, I. Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. In:. 1973.

[18] Sivanandam, S. a Deepa, S. *Introduction to Genetic Algorithms*. Springer, 2008. ISBN 978-3-540-73190-0.

[19] Thangiah, S. R. *Vehicle routing with time windows using genetic algorithms*. Citeseer, 1993.

[20] Toth, P. a Vigo, D. Branch-And-Bound Algorithms for the Capacitated VRP. In: *The Vehicle Routing Problem*. 2001.

[21] Toth, P. a Vigo, D. *The vehicle routing problem*. SIAM, 2002.

[22] Toth, P. a Vigo, D. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*. INFORMS. 2003, zv. 15, č. 4, s. 333–346.

[23] Toth, P. a Vigo, D. *Vehicle routing: problems, methods, and applications*. SIAM, 2014. ISBN 978-1-61197-358-7.

Príloha A

Obsah priloženého pamäťového média

Pamäťové médium priložené k tejto práci obsahuje zdrojové kódy implementácie navrhnutých algoritmov,časť Li & Lim datasetu a tento text práce aj s jeho zdrojovým kódom.

Obsah média obsahuje nasledovné súbory a zložky:

- `dataset/` - obsahuje časť datsetu Li & Lim
- `src/` - obsahuje zdrojové kódy práce
- `text/` - obsahuje text práce
  - `source/` - zdrojový kód textu tejto práce, vytvoreného v LaTeX
  - `thesis.pdf` - text tejto práce
- `Makefile` - súbor slúžiaci na preloženie zdrojových kódov
- `pdp_xgabon00` - binárny súbor spustiteľného programu
- `README.md` - stručný manuál na preloženie a spustenie programu

Príloha B

Manuál

Stručný návod ako preložiť a používať program, za predpokladu používania prostredia s operačným systémom Linux, a nástrojom umožňujúcim preklad zdrojového kódu C++:

- Program sa preloží jednoduchým príkazom `make`. Program využíva len štandardné knihovne C++.
- Po preložení sa program spúšťa príkazom `make run` alebo `./pdp_xgabon00`.
- Konfigurácia spustenia programu je v súbore `src/config.cc`. Pre ukážkové spustenie nie je potreba meniť. V prípade potreby zmeniť niečo, je tu priložený stručný popis parametrov:
  - `INPUT_FILE` - vstupný súbor
  - `MAX_ROUTE_DURATION` - maximálna dĺžka trasy pre jedno vozidlo
  - `CONFIG_DEBUG` - na štandartný výstup je zaznamenané generácia vždy keď nastane zlepšenie, po skončení sa vypíše najlepšie nájdené riešenie vo forme:
    vehicle (počet miest): ID_miesta[naloženie][prejdená vzdialenosť], ...
    distance: celková vzdialenosť trasy
  - `CONFIG_GENERATIONS_PRINT` - každých 100 generácií vypíše fitness hodnotu najlepšieho jedinca
  - `CONFIG_RESULT_SUMMARY` - výstup programu, oddelený bodkočiarkami
  - `CONFIG_EVOLUTION_TYPE` - hodnota "ES" pre použitie evolučnej stratégie alebo hodnota "GA" pre použitie genetického algoritmu
  - `CONFIG_USE_GUIDED_MUTS` - pri mutáciách sa pracuje s cielenými úpravami trias
  - `CONFIG_USE_CENTROIDS` - pri mutáciách sa pracuje s ťažiskom trás
  - `CONFIG_GENERATIONS` - počet generácií
  - `CONFIG_TOUR` - počet vstupujúcich jedincov do turnaja (pri použití GA)
  - `CONFIG_POPSIZE` - veľkosť populácie (pri použití GA)
  - `CONFIG_PMUT` - pravdepodobnosť mutovania génu
  - `CONFIG_LAMBDA` - hodnota λ (pri použití ES)
  - `CONFIG_MI` - hodnota μ (pri použití ES)
  - `CONFIG_ES_PLUS` - ak je nastavený na `true`, je použitý typ evolučnej stratégie $(\mu + \lambda)$, v opačnom prípade je použitý typ (μ, λ)