Povzetek v slovenskem jeziku

5.1 Uvod

Ko sem prišel leta 2007 na podiplomski doktorski študij računalništva, iz Beograda v Koper, sem prišel z veliko željo, da spoznam mesto, v katerem bom preživel naslednja štiri leta. Mestno jedro mi je bilo takoj všeč predvsem zaradi številnih znamenitosti, natančno 55, ki se nahajajo tudi na turističnem zemljevidu Kopra. Ker je doktorski študij naporen in nisem imel veliko prostega časa, sem se začel spraševati kako bi bilo, če bi lahko svoj obhod po mestnih znamenitostih optimiziral na tak način, da porabim najmanj možnih korakov in tako prihranim nekaj časa. Problem sem poimenoval *problem potujočega obiskovalca* (angl. *Traveling Visitor Problem, (TVP)*).

V zgodnjih 30.letih 20.stoletja, je avstrijski matematik Karl Menger izzval takratno raziskovalno skupnost, naj z matematičnega vidika preuči, sledeč problem [98]: Glasnik želi obiskati vsako mesto s seznama na katerem je m mest natanko enkrat in se nato vrniti v svoje mesto pri tem so cene potovanj iz mesta i v mesto j znane vnaprej. Vprašanje je torej kateri obhod je najcenejši? Primer TSP je formalno definiran na polnemu grafu G, na množici vozlišč $V = \{v_1, v_2, ..., v_m\}$, za neko celo število m in s ceno povezav $c_{i,j}$ za povezavo (v_i, v_j) za vse i in j v G.

TSP lahko obravnavamo tudi kot problem permutacij. Naj bo P_m množica vseh permutacij iz množice $\{1, 2, ..., m\}$. Potem je problem trgovskega potnika poiskati $\pi = (\pi(1), \pi(2), ..., \pi(m))$ v P_m, za katero velja, da je $c_{\pi(m)\pi(1)} + \sum_{i=1}^{m-1} c_{\pi(i)\pi(i+1)}$ minimalen.

TSP je eden izmed najpomembnejših predstavnikov večje množice problemov, imenovane kombinatorični optimizacijski problemi [64]. Ker spada TSP v razred NP-težkih (angl. *NP-Hard*) problemov [74], učinkovitega algoritma za TSP ne poznamo. Natančneje, takšen algoritem obstaja če in samo če se razreda P in NP prekrivata. S praktičnega vidika to pomeni, da ne poznamo natančnega algoritma za katerikoli TSP primer z m vozlišči, ki se obnaša značilno bolje, kot algoritem, ki izračuna vseh $(m-1)!$ možnih obhodov ter vrne obhod z najmanjšo ceno.

V praksi lahko za reševanje enakega problema uporabimo tudi drugačen pristop. Določeni TSP primer, z m vozlišči ima lahko katerikoli obhod, ki poteka skozi vsa vozlišča m in predstavlja možno rešitev, ki je zgornja meja (angl. *upper bound*) za najnižjo možno ceno. Algoritem, ki v polinomskem času (angl. *polynominal time*)

konstruira možne rešitve s to zgornjo mejo se imenuje hevristika [12, 119]. Načeloma, ti algoritmi tvorijo rešitve, vendar brez zagotovila o kakovosti rešitve glede na razliko med njihovo ceno in optimalno ceno.

Poznamo dve vrsti TSP: simetrični TSP in asimetrični TSP. V simetrični obliki, znani pod imenom STSP [72, 70, 91, 95], je razdalja med vozliščema i in j enaka razdalji med vozliščema j in i. V primeru asimetričnega TSP(ATSP) [14, 16, 18, 24], takšna simetrija ne obstaja. Poleg tega obstaja še vrsta različnih variacij TSP, ki so opisane in raziskane v literaturi ter predstavljene v doktorski disertaciji v Poglavju 2. Spodaj povzemam nekatere izmed njih.

Grupirani TSP (angl. *Clustered TSP*) [62], ozko-grlni TSP (angl. *Bottleneck TSP*) [60], posplošen TSP (angl. *Generalized TSP*) [59, 77], problem glasnika (angl. *Messenger Problem*) [98], ki je znan tudi kot problem izgubljenega prodajalca (angl. *Wondering Salesman Problem*) [76], problem zamenjave (angl. *The swapping problem*) [2], problem minimalne latentnosti (angl. *Minimum Latency Problem*) [11] znan tudi kot problem dostavljalca (angl. *Delivery Man Problem*) [60] ali problem potujočega mojstra (angl. *Traveling Repairman Problem*) [50], problem seizmičnih plovil (angl. *Seismic Vessel Problem*) [58], ki je posplošitev problema skladiščnega dvigala (angl. *Stacker Crane Problem*) [25], problem potujočega turnirja (angl. *Traveling Tournament Problem*) [40], problem lokacije objekta (angl. *Facility Location Problem*) [39]. In končno problem potujočega obiskovalca, ki je podrobno opisan, raziskovan in rešen v doktorski disertaciji v Poglavju 4, za ta problem ne poznamo nobenega vira v literaturi.

Prvi koraki v reševanju TSP so bili klasični poskusi. Te metode so sestavljene iz natančnih algoritmov in hevristike. Hevristične metode, kot so presek ravnine (angl. *cutting planes*) [31], vejitev in povezovanje (angl. *branch and bound*) [31, 27], lahko optimalno rešijo relativno majhne probleme (v odvisnosti od velikosti m), med tem ko metode, kot so različne variante Lin-Kernighan algoritma [6, 44, 67, 77], Concorde tehnike [4, 3, 5] nam dajo relativno dobre rezultate, tudi za večje probleme. Posamezni algoritmi, zasnovani na požrešnih principih, kot sta najbližji sosed (angl. *nearest neigbour*) [60] in vpeto drevo (angl. *spanning tree*) [66], se prav tako uporabljajo za reševanje TSP.

Natančne metode za reševanje TSP rezultirajo z eksponentnimi računskimi kompleksnostmi, tako da so v izogib obstoječim slabostim potrebne nove metode. Te metode vključujejo različne principe optimizacijskih tehnik, naravno orientirani optimizacijski algoritmi, populacijsko orientirani optimizacijski algoritmi, ter drugi. Različna bitja in naravni sistemi, ki se razvijajo v naravi so zanimivi in dragoceni izvori navdiha, za raziskovanje in ustvarjanje novih sistemov in algoritmov za reševanje TSP, ter njegovih variacij. Nekatere od teh metod so predstavljene v doktorski disertaciji v Poglavju 2. Naj jih naštejemo nekaj.

Evolutivno računanje (angl. *Evolutionary Computation*) [124, 105, 133, 99], genetski algoritmi (angl. *Genetic Algorithms*) [41, 48, 49, 101, 108, 115, 123, 127, 131, 132, 137, 136], memetični algoritmi (angl. *Memetic Algorithms*) [59, 85, 86, 87, 103, 111], sistemi mravelj (angl. *Ant Systems*) [37], simulirano ohlajanje (angl. *Simulated Annealing*) [82], in naposled *cepljeni genetski algoritmi* (angl. *Grafted Genetic Algorithms*) [36, 34]. Slednji predstavljajo vrsto hibridnih genetskih algoritmov in so raziskani, podrobno opisani ter demonstrirani v doktorski disertaciji v Poglavju 3.

V doktorski disertaciji sta obdelani dve temi iz teoretičnega računalništva. Optimizacijsko hevristična metoda imenovana cepljeni genetski algoritmi (GGA). In kombinatorično optimizacijski problem, imenovan problem potujočega obiskovalca (TVP). Cilja doktorske disertacije sta naslednja:

Cepljeni genetski algoritmi: Cilj je pokazati kakovost dobljenih rešitev in hitrost izvajanja cepljenega genetskega algoritma, kadar se uporablja za probleme Simetričnih TSP-jev, ki so na voljo na svetovnem spletu, v obliki splošno priznanih ocenjevalnih primerov (angl. *benchmarks*), kot tudi dejanskih primerov, nastalih za problem potujočega obiskovalca.

Problem potujočega obiskovalca: Cilj je opisati in definirati problem iz realnega življenja, ustvariti realne primere za mesta v okolici in rešiti primere problemov z uporabo nove metode ter znanih metod, ki bodo skupaj prikazane v doktorski disertaciji. Raziskovalni cilj disertacije je dokazovanje spodaj navedenih Hipotez 1 in 2.
Hipoteza 1: Metoda za reševanje TSP, sestavljena iz dveh neodvisnih metod, genetskega algoritma in 2-opt hevristike, združuje kakovosti obeh metod na takšen način, da ju znatno prekaša, glede na kakovost rešitve.

Hipoteza 2: Glede na kakovost rešitve posebna metoda za reševanje problema potujočega obiskovalca, prekaša splošne algoritme za reševanje problema trgovskega potnika, ko jih uporabimo za reševanje problema potujočega obiskovalca.

5.2 Raziskava

5.2.1 Cepljeni Genetski Algoritmi

Botanično cepljenje je postopek, ko je tkivo prve rastline pritrjeno na tkivo druge rastline. Cepljenje lahko zmanjša čas cvetenja in skrajša čas rejskega programa. Lokalni iskalec je razširitev konvencionalnega genetskega algoritma, saj ne obstaja potreba za uporabo komponent genetskega algoritma. To omogoča optimizacijo posameznih genomov, izven evolucijskega procesa. V našem algoritmu po izvedeni rekombinaciji (vrstica 7 v Algoritmu 9), se lokalni iskalec uporablja za optimizacijo vsakega genoma potomcev (vrstica 8 v Algoritmu 9). Zaradi uporabe omenjene zunanje optimizacijske metode, genetski algoritem ni več čist, zato govorimo o cepljenem genetskem algoritmu [34], [36]. Omenjena oblika optimizacije se izvaja lokalno ter spreminja genom s pomočjo hevrističnega spreminjanja rešitve. 2-opt lokalni iskalec 2.2.3 je lokalna optimizacijska metoda za TSP, ki je bila vcepljena v standardni genetski algoritem (vrstica 8 v Algoritmu 9). Ta lokalna optimizacijska metoda, izvaja 2-opt hevristiko, ki izmenjuje povezave grafa z namenom zmanjšanja dolžine obhoda. Postopek izmenjave je sestavljen iz odstranjevanja dveh povezav iz trenutnega obhoda in ponovnega povezovanja na najboljši možen način Figura 3.1.

V opravljenem poskusu smo testirali vpliv cepljenja algoritma lokalnega iskalca z genetskim algoritmom za reševanje problema trgovskega potnika. Za testiranje naše strategije in primerjave le te z ostalimi rešitvami, smo uporabili primere simetričnega problema trgovskega potnika, ki so na voljo na spletu, v knjižnici TSPLIB [120]. Uporablili smo 20 primerov, z različno kompleksnostjo in obsegom od 14 do 150 mest, Tabela 3.1. Primerjali smo našo metodo(Cepljeni Genetski Algoritem), ki je sestavljena iz dveh algoritmov (GGAemc in GGAdpc), s štirimi drugami metodami. Za zgornjo mejo kakovosti rešitve smo uporabili požrešno hevristiko (angl. *Greedy Heuristic* 2.2.2), za spodnjo mejo smo uporabili globalni minimum, pridobljen s Concorde 2.2.5. Nato smo primerjali našo cepljeno metodo z 2-opt in genetskim algoritmom.

Rezultati eksperimenta so predstavljeni v Tabeli 3.1. Šesti stolpec v tabeli predstavlja rešitve našega cepljenega algoritma, ki je bil programiran s pomočjo križanja povezav (angl. *edge map crossover* 2.2.4), kot operator za rekombinacijo (GGAemc). V sedemnajstih od dvajset obravnavanih primerov je bila najdena optimalna rešitev, preostali trije primeri odstopajo od optimalne rešitve za 0.01, 0.10 in 0.22 odstotka. Rešitve so bile najdene hitro, porabili smo med 0.6 in 15.2 sekund ter v relativno majhnem številu generacij. Sedmi stolpec v Tabeli 3.1, pripada našemu cepljenemu genetskemu algoritmu, ki vsebuje križanje ohranjanja razdalje (angl. *distance preserving crossover* 2.2.4), kot operator za rekombinacijo (GGAdpc). V enajstih od dvajset obravnavanih primerov je bila najdena optimalna rešitev, v preostalih devetih primerih so odstopanja od optimalne rešitve od 0.13 do 0.32 odstotka. V primerjavi z GGAemc, sta čas izvajanja in število generacij GGAdpc nekoliko manjša, posebej v nižjem predelu tabele, ki predstavlja kompleksnejše primere.

Kvantitativni rezultati na testnih primerih iz TSPLIB kažejo, da imata cepljena algoritma, GGAemc in GGAdpc prednosti. Kljub številnim pomankljivostim njunih komponent, se njune kombinacije cepljenja zelo dobro obnesejo. Rezultati primerov iz TSPLIB kažejo, da omenjena metoda cepljenja združuje dobre lastnosti iz obeh uporabljenih metod in občutno prekaša vsako izmed njiju.

5.2.2 Problem Potujocega Obiskovalca

Obiskovalci so prispeli v hotel v nekem novem mestu z željo, da obiščejo vse zanimivosti mesta natanko enkrat in se po ogledu vrnejo v hotel. Na ogled mesta se odpravijo peš - po ulicah, sprehajalnih zonah in poteh za pešce. Cilj je skrajšati pot obiskovalca.

Problem potujočega obiskovalca je izpeljan iz *problema trgovskega potnika* (angl. *Traveling Salesman Problem, (TSP)*) [60, 73, 5, 52, 57, 63, 66, 90, 97], pri čemer velja pravilo, da obiskovalec izbira samo med potmi, ki jih je možno prehoditi. To pomeni, da so evklidske razdalje [54, 111], kot jih poznamo v TSP, v našem primeru napačne. Obiskovalci uporabljajo sprehajalne poti in območja za pešce, ki so različno dolge. Te omejitve določajo težo povezav, ki povezujejo vozlišča v grafu.

Definicija TVP je sledeča: Imamo graf $G = (V, E, c)$, kjer je množica vozlišč $V = S \cup X$ in $S \cap X = \emptyset$, kjer sta S zanimivosti mesta in X križišča, E niz povezav ter c cena povezav. Cilj je poiskati najkrajši zaprt sprehod skozi vsa vozlišča S (glede

na c) v grafu G, pri čemer se lahko sprehodimo skozi X.

Prva predlagana metoda za reševanje problema potujočega obiskovalca je Naivni algoritem (angl. *Naïve Algorithm*), prikazan v Algoritmu 10. V prvi vrstici psevdokode, lahko razberemo naslednje parametre: S pripada zanimivim lokacijam v mestu, X pripada križiščem, E pripada množici povezav, W pa predstavlja matriko povezav grafa G, $(S \cup X \times S \cup X)$. V prvem koraku algoritma je problem potujočega obiskovalca rešen kot primer problema trgovskega potnika. V naslednjem koraku, iz matrike povezav W izdelamo matriko povezav Z $(S \times S)$, ki predstavlja rešitev problema najkrajše poti vseh parov (angl. *APSP*). V zanki (od vrstice 6 do 8) je prikazana rešitev za TVP, ki poišče najkrajše poti od Z do T. Druga predlagana metoda za reševanje problema potujočega obiskovalca je Algoritem Koper (angl. *Koper Algorithm*), prikazan v Algoritmu 11. Prva vrstica psevdokode vsebuje enake parametre kot naivni algoritem. V prvem koraku poiščemo najkrajšo pot vseh parov v grafu G. Vhodno matriko razdalj označimo z W in izhodno matriko razdalj z Z. V naslednjem koraku, rešimo problem trgovskega potnika za matriko razdalj Z. Poleg tega smo dobili T, ki je rešitev za problem potujočega obiskovalca.

Za testiranje naše strategije smo uporabili primere problema potujočega obiskovalca, ki smo jih izdelali iz uradnih turističnih zemljevidov za mesta Koper, Beograd in Benetke. V primeru Beograda smo izdelali dva primera, ki se razlikujeta po velikosti problema, oziroma po številu vozlišč v grafu. Iz javno dostopne knjižnice, TSPLIB, smo si izbrali, spremenili in testirali dva primera problema simetričnega trgovskega potnika. Izvedli smo poskuse za 5 primerov, z različnimi velikostmi, ki so od 120 do 1002 vozlišč za posamezen primer.

Za reševanje Problema potujočega obiskovalca smo primerjali dve metodi. Prva metoda je gore omenjeni naivni algoritem, prikazan v Algoritmu 10. Druga metoda je algoritem koper, prikazan v Algoritmu 11. Za reševanje TSP, ki je eden izmed korakov pri obema algoritmoma, smo uporabili algoritem Concorde, ki je predstavljen v poglavju 2.2.5. Za reševanje problema najkrajše poti vseh parov smo uporabili prilagojeni Floyd-Warshallov algoritem, ki je bil predstavljen v poglavju 4.2.2.

Rezultati poskusa so predstavljeni v Tabeli 4.1. Peti stolpec Tabele 4.1, predstavlja dolžino sprehoda (cena rešitve, ki smo jo dobili pri poskusu). Stolpec se imenuje cena sprehoda (angl. *tour cost*) in v vseh šestih primerih so najkrajši sprehodi pridobljeni z algoritmom Koper. Zadnji stolpec v Tabeli 4.1, predstavlja razliko med uporabljenimi metodami, prikazano v odstotkih. Prva metoda, naivni algoritem, se je odrezal veliko slabše od algoritma Koper. Kakovost rešitev varira v intervalu od 6.52 odstotka, v primeru Belgrade163, do 354.46 odstotka, v primeru pr1002.

Namen tega raziskovanja je bil opis in rešitev novega problema v teoriji grafov, imenovanega problem potujočega obiskovalca (TVP). Čeprav je nov problem podoben dobro znanemu problemu trgovskega potnika, ko jo poskušamo rešiti z naivnim algoritmom, so rezultati daleč od optimalnih. V vseh testiranih primerih problema potujočega obiskovalca, algoritem Koper občutno prekaša Naivni algoritem.

5.3 Doprinos k znanosti

Doprinos k znanosti predstavljajo naslednji rezultati:

- Izdelava cepljenega genetskega algoritma za reševanje problema trgovskega potnika in problema potujočega obiskovalca.
- Potrditev, da problem trgovskega potnika lahko uspešno rešimo z uporabo cepljenega genetskega algoritma.
- Izdelava posebne metode za reševanje problema potujočega obiskovalca.
- Izdelava realnih primerov problema potujočega obiskovalca za mesta Koper, Beograd in Benetke.
- Potrditev da vsak primer problema potujočega obiskovalca, ki je rešen z posebno metodo, predstavlja zelo zadovoljivo rešitev.

Rezultati doktorske disertacije predstavljajo doprinos k premoščanju razlike med teoretičnim računalništvom in njegovo uporabo v praksi: boljšemu razumevanju in modeliranju realnih problemov iz gospodarstva, predstavljenih kot NP-polni problemi v teoriji grafov, kot tudi doprinos optimizacijskim metodam za reševanje omenjenih problemov.

5.4 Metodologija

Glavno orodje za opis in definicijo problema potujočega obiskovalca je teorija grafov. Za realistično predstavitev vozlišč našega grafa, kot tudi povezav ter cen je uporabljen geografski informacijski sistem "Google Earth", čigar podatkovo bazo smo uporabili za pridobitev mestnih znamenitosti, križišč in sprehajalnih poti. Pomembno vlogo za dokazovanje hipoteze 1 ima platforma za raziskovanje genetskih algoritmov "EA Visualizer" [15], aplikacija napisana v programskem jeziku Java.

Za dokazovanje hipoteze 2 smo uporabili in ustrezno nadgradili Floyd-Warshallov algoritem [28], za iskanje najkrajših poti med vsemi pari vozlišč grafa $G = (V, E, c)$. Uporabili smo tudi znane optimizacijske metode za reševanje simetričnega in asimetričnega problema trgovskega potnika, presek-ravnine [113, 114], na čigar osnovi temeljijo metode Concorde [4, 3, 5] in hevristične metode Lin-Kernighan algoritma [6, 44, 67, 77]. Obe aplikaciji sta pisani v programskem jeziku AnsiC. Poleg tega za dokazovanje hipoteze 2 smo uporabili znanje iz matematičnih modelov znanih kot problem linearnega programiranja (angl. *linear programming problems*) [23, 129], kot tudi simpleks metode (angl. *simplex methods*) [31].

5.5 Predvidena vsebina disertacije

Doktorska disertacija po Pravilniku o pripravi in zagovoru doktorske disertacije na Univerzi na Primorskem, vsebuje naslednja poglavja:

- Zahvala
- Povzetek
- Kazalo vsebine
- Poglavje 1 - Uvod

- Poglavje 2 - Ozadje
  - 2.1 Problem Trgovskega Potnika
  - 2.2 Optimizacijski Algoritmi
- Poglavje 3 - Cepljeni Genetski Algoritmi
- Poglavje 4 - Problem Potujočega Obiskovalca
- Zaključek
- Literatura
- Kazalo
- Izjava

V poglavju *Ozadje* smo predstavili osnovne pojme trgovskega potnika in optimizacijskih algoritmov, s pomočjo katerih so lahko nadaljnja poglavja disertacije postala razumljiva tudi širšemu krogu bralcev. Preostali poglavji so namenjeni predstavitvi doseženih ciljev disertacije. Poglavja so razdeljena na več podpoglavij.

Naj omenimo še, da so rezultati disertacije objavljeni v naslednjih znanstvenih člankih:

- M. Djordjevic, Influence of Grafting a Hybrid Searcher Into the Evolutionary Algorithm, *Proceedings of the Seventeenth International Electrotechnical and Computer Science Conference.* (2008), 115-118.
- M. Djordjevic, M. Tuba, and A. Brodnik Impact of Grafting a 2-opt Algorithm Based Local Searcher Into the Genetic Algorithm, *Proceedings of the 9th WSEAS International Conference on Applied Informatics and Communications.* (2009), 489-490.
- M. Djordjevic, and A. Brodnik, Quantitative Analysis of Separate and Combined Performance of Local Searcher and Genetic Algorithm, *Book of Abstract of International Conference on Operations Research, OR2011.* (2011), 130.