UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Zorica M. Dražić

MODIFIKACIJE METODE PROMENLJIVIH OKOLINA I NJIHOVE PRIMENE ZA REŠAVANJE PROBLEMA RASPOREĐIVANJA PRENOSA DATOTEKA

doktorska disertacija

Beograd, 2014.

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Zorica M. Dražić

MODIFICATIONS OF THE VARIABLE NEIGHBORHOOD SEARCH METHOD AND THEIR APPLICATIONS TO SOLVING THE FILE TRANSFER SCHEDULING PROBLEM

Doctoral Dissertation

Belgrade, 2014.

Podaci o mentoru i članovima komisije

Mentor

dr Vladimir Filipović, vanredni profesor, Matematički fakultet, Univerzitet u Beogradu

Članovi komisije

dr Dušan Tošić, redovni profesor, Matematički fakultet, Univerzitet u Beogradu

dr Miodrag Živković, redovni profesor, Matematički fakultet, Univerzitet u Beogradu

dr Dragan Urošević, viši naučni saradnik, MI SANU

dr Aleksandar Savić, docent, Matematički fakultet, Univerzitet u Beogradu

Datum odbrane:

Podaci o doktorskoj disertaciji

Naslov doktorske disertacije

Modifikacije metode promenljivih okolina i njihove primene za rešavanje problema raspoređivanja prenosa datoteka.

Rezime

Metoda promenljivih okolina se u praksi pokazala vrlo uspešnom za rešavanje problema diskretne i kontinualne optimizacije. Glavna ideja ove metode je sistematska promena okolina unutar prostora rešenja u potrazi za boljim rešenjem. Za optimizaciju funkcija više promenljivih koriste se metode koje nalaze lokalni minimum polazeći od zadate početne tačke. U slučaju kada kontinualna funkcija ima mnoštvo lokalnih minimuma, nalaženje globalnog minimuma obično nije lak zadatak jer najčešće dostignuti lokalni minimumi nisu optimalni. Kod uobičajenih implementacija sa ograničenim okolinama različitih dijametara iz proizvoljne tačke nije moguće dostići sve tačke prostora rešenja. Zbog toga je strategija korišćenja konačnog broja ograničenih okolina primenjiva na probleme kod kojih optimalno rešenje pripada nekom unapred poznatom ograničenom podskupu skupa $\mathbb{R}^n$.

U cilju prevazilaženja pomenutog ograničenja predložena je nova varijanta metode, Gausovska metoda promenljivih okolina. Umesto definisanja niza različitih okolina iz kojih će se birati slučajna tačka, u ovoj metodi se sve okoline poklapaju sa celim prostorom rešenja, a slučajne tačke se generišu korišćenjem različitih slučajnih raspodela Gausovog tipa. Na ovaj način se i tačke na većem rastojanju od tekuće tačke mogu teorijski dostići mada sa manjom verovatnoćom.

U osnovnoj verziji metode promenljivih okolina neophodno je unapred definisati sistem okolina, njihov ukupan broj i veličinu, kao i tip raspodele koja će se koristiti za odabir slučajne tačke unutar tih okolina. Gausovska metoda promenljivih okolina za razliku od osnovne verzije ima manje parametara jer su sve okoline teorijski iste veličine (jednake celom prostoru pretrage) i imaju jedinstvenu jednoparametarsku familiju raspodela – Gausovu raspodelu slučajnih brojeva sa promenljivom disperzijom.

Problem raspoređivanja prenosa datoteka (File transfer scheduling problem - FTSP) je optimizacioni problem koji svoju primenu pronalazi u mnogim oblastima

poput telekomunikacijama, LAN i WAN mrežama, raspoređivanju u okviru MIMD (multiple instruction multiple data) računarskih sistema i dr. Spada u klasu NP teških problema za čije rešavanje se uobičajeno koriste heurističke metode. Zadatak optimizacije FTSP sastoji se u traženju odgovarajućeg rasporeda pojedinačnih prenosa datoteka, tj. vremenskih trenutaka kada će svaka datoteka započeti svoj prenos tako da dužina vremenskog intervala od trenutka kada prva datoteka započne prenos do trenutka u kom poslednja završi bude što manja.

U cilju egzaktnog rešavanja datog problema razvijeni su modeli celobrojnog linearnog programiranja i dati su dokazi njihove korektnosti. Na taj način se mogu dobiti optimalna rešenja ovog problema na test primerima malih i srednjih dimenzija.

Za rešavanje test primera velikih dimenzija razvijena je metoda promenljivih okolina koja koristi "permutacijsku" reprezentaciju gde se može koristiti uobičajen sistem okolina. Osim toga, data metoda promenljivih okolina se koristi i za određivanje gornje granice rešenja koja je neophodna u formulaciji predloženih modela celobrojnog linearnog programiranja. U cilju bržeg dostizanja boljih rešenja u bliskoj okolini tekuće tačke implementirane su tri različite varijante lokalne pretrage:
2-zamena, 2-zamena susednih i metoda spusta kroz promenljive okoline.

U cilju primene metoda kontinualne optimizacije za rešavanje FTSP razvijena je težinska reprezentacija rešenja. Ovakva reprezentacija omogućava primenu metoda kontinualne optimizacije koje ne zahtevaju diferencijabilnost funkcije cilja. Kako se Gausovska metoda promenljivih okolina pokazala uspešnom u rešavanju problema kontinualne optimizacije, primenjena je i za rešavanje FTSP. Prethodno opisane metode lokalnog pretraživanja se mogu koristiti i u slučaju težinske reprezentacije rešenja.

Korišćenjem navedenih metoda pronađena su optimalna rešenja za test primere malih i srednjih dimenzija. Na instancama velikih dimenzija koje su van domašaja egzaktnih metoda dobijena su dobra rešenja metaheurističkim metodama u razumnom vremenu izvršavanja.

Ključne reči

Kontinualna optimizacija, kombinatorna optimizacija, metoda promenljivih okolina, celobrojno linearno programiranje, metaheuristike.

Naučna oblast

Računarstvo

Uža naučna oblast

Optimizacija

UDK broj

[519.863+519.852]:004.772(043.3)

Dissertation Data

Doctoral dissertation title

Modifications of the Variable neighborhood search method and their applications to solving the File transfer scheduling problem.

Abstract

The Variable neighborhood search method proved to be very successful for solving discrete and continuous optimization problems. The basic idea is a systematic change of neighborhood structures in search for the better solution. For optimization of multiple variable functions, methods for obtaining the local minimum starting from certain initial point are used. In case when the continuous function has many local minima, finding the global minimum is usually not an easy task since the obtained local minima in most cases are not optimal. In typical implementations with bounded neighborhoods of various diameters it is not possible, from arbitrary point, to reach all points in solution space. Consequently, the strategy of using the finite number of neighborhoods is suitable for problems with solutions belonging to some known bounded subset of $\mathbb{R}^n$.

In order to overcome the previously mentioned limitation the new variant of the method is proposed, Gaussian Variable neighborhood search method. Instead of defining the sequence of different neighborhoods from which the random point will be chosen, all neighborhoods coincide with the whole solution space, but with different probability distributions of Gaussian type. With this approach, from arbitrary point another more distant point is theoretically reachable, although with smaller probability.

In basic version of Variable neighborhood search method one must define in advance the neighborhood structure system, their number and size, as well as the type of random distribution to be used for obtaining the random point from it. Gaussian Variable neighborhood search method has less parameters since all the neighborhoods are theoretically the same (equal to the solution space), and uses only one distribution family - Gaussian multivariate distribution with variable dispersion.

File transfer scheduling problem (FTSP) is an optimization problem widely applicable to many areas such as Wide Area computer Networks (WAN), Local Area Ne-

tworks (LAN), telecommunications, multiprocessor scheduling in a MIMD machines, task assignments in companies, etc. As it belongs to the NP-hard class of problems, heuristic methods are usually used for solving this kind of problems. The problem is to minimize the overall time needed to transfer all files to their destinations for a given collection of various sized files in a computer network, i.e. to find the file transfer schedule with minimal length.

In order to obtain the exact solution of the FTS problem, integer linear programming formulations are proposed and their correctness is proved. In this way optimal solutions can be found for small and medium size test instances.

For large test instances, the Variable neighborhood search method is proposed using the "permutation" representation and typical neighborhood structures. Moreover, the same method is used for obtaining the upper bounds of the solutions which are used in proposed integer linear programming formulations. For obtaining better solutions in the small neighborhood of the current solution, three different local search procedures are implemented: 2-swap, 2-swap adjacent and variable neighborhood descent.

In order to apply the continuous optimization methods for solving FTSP, the weighted solution representation is developed. Such representation enables the continuous optimization methods to be used, which do not require the differentiability of objective function. Since Gauss Variable neighborhood search method proved to be successful in continuous optimization problems, it was applied to FTSP. Previously described local search procedures can also be used with weighted solution representation.

Using the proposed methods optimal solutions for all small and medium size test instances are found. For large size instances, which are beyond the reach of exact methods, metaheuristic methods obtained good solutions in reasonable time.

Keywords

continual optimization, combinatorial optimization, variable neighborhood search, integer linear programming, metaheuristics.

Scientific field

Computer Science

Scientific subfield

Optimization

UDC number

[519.863+519.852]:004.772(043.3)

Predgovor

U ovom radu razmatra se jedan diskretan problem optimizacije: problem raspoređivanja prenosa datoteka. Razvijene su različite modifikacije metode promenljivih okolina za rešavanje ovog problema, kao i tri različita modela celobrojnog linearnog programiranja. Razvijena je i nova modifikacija metode, Gausovska metoda promenljivih okolina, sa primarnim ciljem rešavanja problema kontinualne optimizacije. Proučeno je kako se ova modifikacija može primeniti na probleme diskretne optimizacije adaptacijom za konkretan problem raspoređivanja prenosa datoteka.

Rad se sastoji od pet poglavlja. U uvodnom poglavlju su prikazane osnovne informacije o diskretnoj i kontinualnoj optimizaciji, kao i kratak pregled postojećih metaheuristika za njihovo rešavanje. U ovom poglavlju dati su još kratki opisi različitih varijanti metode promenljivih okolina i pregled dosadašnje primene metode za rešavanje problema kontinualne optimizacije.

U drugom delu prikazana je nova modifikacija metode promenljivih okolina, Gausovska metoda promenljivih okolina, za rešavanje problema kontinualne optimizacije. Nakon opisa metode dato je poređenje sa drugim metodama za rešavanje odgovarajućeg problema.

Treće poglavlje posvećeno je rešavanju problema raspoređivanja prenosa datoteka. Date su dve nove reformulacije problema, kao i tri modela celobrojnog linearnog programiranja. Prezentovani su detalji modifikacija metode promenljivih okolina za rešavanje ovog problema, a zatim su prikazani eksperimentalni rezultati.

U četvrtom poglavlju su objedinjeni prethodni pristupi i izvršena je adaptacija Gausovske metode promenljivih okolina za rešavanje problema kontinualne optimizacije sa ciljem rešavanja diskretnog problema raspoređivanja prenosa datoteka. Na kraju su u petom poglavlju dati zaključak i naučni doprinos ovog rada i dobijenih rezultata.

Ovom prilikom bih zelala da se zahvalim mentoru, dr Vladimiru Filipoviću, koji

je rukovodio izradom moje doktorske disertacije, kao i članovima komisije: prof. dr Dušanu Tošiću, prof. dr Miodragu Živkoviću, prof. dr Draganu Uroševiću i doc. dr Aleksandru Saviću. Posebnu zahvalnost dugujem i dr Nenadu Mladenoviću, višem naučnom savetniku Matematičkog instituta SANU, koji mi je pružio ogromnu pomoć brojim savetima, idejama i predlozima.

Naročito se zahvaljujem svojim roditeljima, porodici i prijateljima na bezgraničnom razumevanju i ogromnoj podršci koja mi je rad na ovoj disertaciji učinila neuporedivo lakšim.

Sadržaj

Glava 1

Uvod

1.1 Problemi kontinualne i kombinatorne optimizacije

Optimizacija je postupak potrage za optimumom, tj. nalaženja najboljeg rešenja nekog problema u određenom smislu i pri određenim uslovima. U užem smislu to je grana matematike i računarstva. U opštem slučaju zadatak optimizacije se može formulisati na sledeći način [21, 12]: Odrediti

$$\min\{f(x)|x \in X, X \subseteq S\} \tag{1.1.1}$$

gde je S prostor rešenja, X skup dopustivih rešenja, x dopustivo rešenje i f realna funkcija koju nazivamo funkcija cilja. Ukoliko je S konačan ili beskonačan prebrojiv skup reč je o kombinatornoj (diskretnoj) optimizaciji, a u slučaju $S = \mathbb{R}^n$ u pitanju je kontinualna (neprekidna) optimizacija.

Tačka $x^* \in X$ se naziva globalnim minimumom funkcije f ako $f(x^*) \leq f(x)$, za svako $x \in X$. Ukoliko postoji $\epsilon > 0$ takvo da je $f(x^*) \leq f(x)$ za svako $x \in X$ i $|| \ x - x^* \ || \leq \epsilon$, onda za x^* kažemo da je lokalni minimum. Zadatak je pronaći dopustivo rešenje x^* (ili više njih) takvo da je $f(x^*) = \min_{x \in X} f(x)$. Takvo rešenje nazivamo optimalno rešenje. Optimalno rešenje može biti jedinstveno, a može postojati i više rešenja sa istom (minimalnom) vrednosčću funkcije cilja. Najčešće je dovoljno pronaći samo jedno optimalno rešenje.

Kako je $\max_{x \in X} f(x) = -\min_{x \in X}(-f(x))$, problem maksimizacije se obično svodi na problem minimizacije pa je dovoljno rešavati samo jedan od njih.

Do sada je razvijen veliki broj metoda za nalaženje globalnog minimuma. Uspeh u nalaženju rešenja prvenstveno zavisi od težine problema. Za NP-teške probleme ne može se ni od jedne metode očekivati da u razumnom vremenu dođe do tačnog rešenja. Čak i u tom slučaju je poželjno da se brzo dođe do "dovoljno dobrog" rešenja. Do globalnog minimuma može se doći nalaženjem svih lokalnih minimuma, a zatim biranjem najboljeg od njih. U praktičnim zadacima broj lokalnih minimuma moze biti jako veliki ili čak beskonačan pa je ovaj pristup nepraktičan.

Algoritmi za rešavanje optimizacionih problema se mogu podeliti u dve kategorije: egzaktne i približne (heurističke) algoritme.

Egzaktni algoritam za rešavanje nekog konkretnog problema oblika 1.1.1, ukoliko postoji, nalazi optimalno rešenje x^*, pružajući ujedno i dokaz o njegovoj optimalnosti, ili pokazuje da ne postoji dopustivo rešenje, tj. $X = \emptyset$. Egzaktne metode su najpoželjnije jer garantuju optimalnost rešenja, ali je njihova primenljivost, u slučaju teških problema, u praksi mala. Prvi razlog je činjenica da se pomoću njih mogu uspešno rešavati samo problemi malih dimenzija. Mnogi problemi kombinatorne optimizacije spadaju u klasu problema za koje nije poznat algoritam polinomske složenosti za njihovo rešavanje (NP-teški problemi). Pri rešavanju problema kombinatorne optimizacije broj dopustivih rešenja često jeste konačan, ali za veće dimenzije problema je i jako veliki pa je egzaktne metode nemoguće praktično primeniti za njihovo rešavanje. Sa druge strane, često nije moguće primeniti egzaktne metode jer ne postoji odgovarajući matematički model na koji bi se one mogle primeniti ili je model previše složen. Za takve probleme heurističke metode su najčešće jedini način na koji ih je moguće rešavati.

Sa druge strane, heuristički algoritmi (heuristike) ne garantuju optimalnost dobijenih rešenja, ali često je dovoljno naći i samo dovoljno dobro rešenje. Glavna prednost heuristika je njihova brzina, tj. relativno kratko vreme potrebno da se nađe dovoljno dobro rešenje, jer u slučaju NP-teških problema (uz pretpostavku da je $P \neq NP$) egzaktnim metodama bi u najgorem slučaju trebalo eksponencijalno vreme za njihovo rešavanje. Heuristike se najčešće projektuju i implementiraju za konkretan problem, oslanjajući se na karakteristike samog problema, što često može biti dugotrajan i naporan posao. Heuristička metoda koja je razvijena za jedan problem, često nije primenljiva za druge probleme. Sa druge strane, neke heurističke metode su razvijene na apstraktnijem nivou i mogu se, uz manje

adaptacije, primeniti za rešavanje većeg broja različitih problema. Takve heuristike zovemo metaheuristikama.

1.2 Metaheuristike

Metaheuristike se formalno mogu definisati kao iterativni procesi koji koriste približne metode, kombinujući na inteligentan način različite koncepte za pretraživanje čitavog prostora rešenja kako bi se na efikasan način našlo rešenje koje je što je moguće bliže optimalnom [82].

Neke od najpoznatijih metaheuristika su:

- Tabu pretraživanje
- Genetski algoritam
- Metoda promenljivih okolina
- Lagranžova relaksacija
- Simulirano kaljenje
- Mravlji algoritmi
- Pčelinje kolonije
- Elektromagnetizam

Literatura na temu metaheuristika je brojna. U daljem tekstu navedene su neke od poznatijih metaheuristika sa svojim osobinama, dok se više detalja o metaheuristikama može naći u [82, 86, 64, 11, 37]. Pošto će u ovom radu biti primenjivana metoda promenljivih okolina, ona će detaljnije biti opisana u narednim odeljcima.

Tabu pretraživanje (eng. Tabu search) je metaheuristika koja se zasniva na principu lokalne pretrage. Tabu pretraživanje pamti informacije o ranijim pretragama i ti podaci se koriste pri izboru narednog rešenja. Ideja je da koristeći ove informacije algoritam ne vrši pretragu u delovima koji su već pretraženi, te se uvodi zabrana povratka u pretraženu okolinu ili u neka od dobijenih rešenja u određenom broju narednih iteracija. Pretraga se ne vrši obavezno od lošijeg ka boljem rešenju, nego je dozvoljeno i kretanje od boljeg ka lošijem. Kada se naiđe na

lokalni minimum, on se isključuje iz okoline za pretraživanje i pamti u tabu listi (eng. tabu list), a pretraga se dalje nastavlja u drugom delu prostora rešenja.

Tabu pretraga se može realizovati upotrebom različitih vrsta memorija: kratkoročnih za formiranje tabu lista, srednjoročnih i dugoročnih. Kako bi se povećala efikasnost algoritma koriste se metode intenzifikacije i diverzifikacije. Srednjoročne memorije služe za intenzifikaciju kojom se pretraga koncentriše u okolinama u kojima se očekuju bolja rešenja. Jedan od načina je formiranje elitnih rešenja (eng. elite solutions) čiji se atributi, koji ih karakterišu, relativno razlikuju. Nakon pretrage kompletnog prostora obavlja se pretraga od tih elitnih rešenja pri čemu se ignorišu sve zabrane koje su postojale do trenutka pronalaženja tog rešenja. Sa druge strane, dugoročne memorije imaju ulogu da omoguće diverzifikaciju kojom se pretraga usmerava ka novim, neistraženim regionima prostora rešenja. Kako bi se to realizovalo, vodi se evidencija o nekim atributima posećenih rešenja (koliko puta se nalazio u rešenju ili koliko puta je menjao status). Ukoliko u određenom broju iteracija ne dođe do popravke rešenja, jedan od načina je da se pretraga nastavi od novog početnog rešenja koje uključuje, na primer, atribute koji nisu karakterisali već posećena rešenja.

Elitna rešenja mogu biti iskorišćena i za upotrebu tehnike povezujućih staza (eng. path relinking). Zamenom atributa formira se put od jednog do drugog elitnog rešenja na kom se može očekivati da će biti pronađeno i neko novo bolje rešenje.

Više detalja o ovoj metodi se može naći u [39, 40, 42, 41].

Genetski algoritam (eng. Genetic algorithm) je metaheuristika koja je zasnovana na evoluciji, prirodnom odabiru i ukrštanju vrsta koju je prvi predložio J. Holand u knjizi [58]. Populacija se sastoji od većeg broja digitalno kodiranih rešenja koja se nazivaju jedinkama. Na osnovu jedne populacije formira se nova tako što se iz prvobitne operatorom selekcije izdvoje jedinke koje su najbolje prilagođene (najčešće sa najboljom vrednošću funkcije cilja), izvrši ukrštanje tih jedinki čime se dobija jedna ili više novih jedinki, a zatim na nove jedinke primeni operator mutacije sa određenom verovatnoćom. Ukrštanjem dobro prilagođenih jedinki dobijaju se jedinke koje su prilagođene bar koliko i roditelji ako ne i bolje, dok se mutacijom izbegava dominacija jedinki sa istim osobinama. Više o ovoj metodi se može naći u [43, 6, 5, 31].

Lagranžova relaksacija (eng. Lagrangian relaxation) [7, 32] se zasniva na ideji da se jedan NP-težak problem transformiše u drugi problem za koji postoje efikasne i brze metode rešavanja. To pojednostavljenje se postiže tako što se nekim od uslova zadatka dodele odgovarajući faktori koji se nazivaju Lagranžovi množioci. Na ovaj način se deo ili čak svi uslovi zadatka prebacuju u funkciju cilja. Ukoliko se pogodno izaberu uslovi koji će da uđu u funikciju cilja, mogu se dobiti problemi koje je lakše rešiti, a čija rešenja su aproksimacije rešenja polaznog problema.

Simulirano kaljenje (eng. Simulated annealing) je svoj naziv dobilo od postupka kaljenja čelika gde se, u cilju dobijanja veće tvrdoće, temperatura postepeno snižava. Ideja metode je da se u svakoj iteraciji bira na slučajan način tačka u blizini trenutnog rešenja i ona se može prihvatiti za novo rešenje bez obzira da li je došlo do poboljšanja ili ne. Glavni parametar metode se obično naziva parametrom temperature i proporcionalan je verovatnoći sa kojom se prihvata i lošije rešenje kao tekuće u postupku optimizacije. Na početku parametar temperature je dosta veliki dok se vremenom polako snižava, a time i verovatnoća prihvatanja lošijih rešenja. Detalji metode se mogu naći u [63, 90].

Mravlji algoritmi (eng. Ant colony optimization) [24, 23] su algoritmi inspirisani ponašanjem mravljih kolonija. Prilikom kretanja (u potrazi za hranom) mravi luče na tlo fermone koji služe kao informacija za ostale članove kolonije kojim putem treba da se kreću. Veći broj mrava koji prođe istim putem ostavlja jači trag koji će pripadnici kolonije slediti, međutim uvek postoje i mravi koji nezavisno krenu novim putem u potrazi za hranom. Glavna ideja mraljih algoritama je da se simulira ponašanje mravljih kolonija u cilju pretrage dopustivog prostora za rešenjem.

Pčelinje kolonije (eng. Bee colony optimization) je metoda koja je inspirisana ponašanjem pčela u potrazi za hranom. Predložili su je Lučić i Todorović 2001. godine [69]. Pčele u prirodi tragaju za hranom i donose uzorke nektara u košnicu gde se ispituje njegov kvalitet. Kvalitetniji nektar je praćen plesom pčela kojim se ukazuje na pravac i daljinu odakle je donet. Svaka pčela se sa nekom verovatnoćom odlučuje ili da prati trag gde je već pronađen dobar nektar, da igrom "reklamira" mesto hrane i time "regrutuje" druge pčele da prate taj trag ili da nastavlja da traga za novim izvorom hrane. Detaljniji opis algoritama zasnovanih na ovoj ideji se može naći u [70, 71].

Elektromagnetizam (eng. Electromagnetism) je metaheuristika koja vrši pre-

tragu višedimenziog prostora rešenja primenjujući zakone elektromagnetizma. Svako rešenje vrši ili privlačenje ili odbijanje drugih potencijalnih rešenja po Kulonovom zakonu koji definiše intenzitet, pravac i smer elektrostatičke sile kojom nepokretno naelektrisanje malih dimenzija deluje na drugo. Princip algoritma se zasniva na tome da će inferiorna rešenja sprečiti kretanje pretrage u njihovom pravcu odbijanjem, dok će bolja rešenja vršiti privlačenje u svom pravcu. Algoritam su predlozili Birbil i Fang 2003 u [10], dok se više o samoj metodi i njenim primenama može naći u [72, 22, 16].

1.3 Metoda promenljivih okolina

Metodu promenljivih okolina (eng. Variable Neighborhood Search, VNS) su 1997. godine predložili Mladenović i Hansen u radu [77], a ideja je prvi put izložena 1995. godine [75]. Metoda je najpre razvijena za probleme kombinatorne optimizacije, a kasnije i za probleme kontinualne optimizacije, dok se danas uspešno primenjuje za rešavanje problema i iz jedne i iz druge oblasti.

Glavna ideja VNS-a je sistematska zamena okolina u kojima se vrši pretraga za rešenjem u cilju izbegavanja zapadanja u lokalne minimume. Metoda se zasniva na tri činjenice [48]:

- Lokalni minimum u jednoj okolini nije obavezno lokalni minimum za neku drugu okolinu.
- Globalni minimum je lokalni minimum u svim okolinama.
- Za veliki broj problema lokalni minimumi u jednoj ili više okolina su relativno blizu jedni drugima.

Poslednja činjenica ukazuje na to da lokalni minimum često sadrži informaciju o globalnom i zbog toga se javlja potreba za istraživanjem okolina lokalnih minimuma. Ove tri činjenice se mogu koristiti deterministički, stohastički ili kombinovano, čime se dobijaju različite verzije VNS-a. U daljem tekstu navedene su neke od osnovnih varijanti VNS metode, dok se više detalja može naći u [44, 46, 48, 47, 49, 51].

1.3.1 Metoda promenljivog spusta

Metoda promenljivog spusta (eng. Variable Neighborhood Descent, VND) [46, 48] je deterministička varijanta VNS-a. Zasnovana je na činjenici da jedno rešenje ne mora biti lokalni minimum za dve različite okoline.

Pri realizaciji VND-a najpre se mora u prostoru rešenja definisati konačan skup okolina N_k, $k = 1, 2, ..., k_{\max}$, gde $N_k(x)$ označava k-tu okolinu rešenja x. Zatim se na slučajan način ili primenom neke heuristike bira početno rešenje koje je tada ujedno i trenutno najbolje x^* i odakle će se vršiti lokalno pretraživanje u prvoj okolini $N_1(x^*)$. Kao rezultat tog pretraživanja dobijamo lokalni minimum x' u okolini N_1. Ukoliko sa x' nije postignuto poboljšanje rešenja, postupak lokalne pretrage se nastavlja u narednim okolinama $N_k(x^*)$, $k = 2, 3, ..k_{\max}$. Svaki put kada se u nekoj okolini $N_k(x^*)$ nađe bolje rešenje x', tj. takvo da je $f(x') < f(x^*)$, ono se proglašava za najbolje rešenje ($x^* := x'$) i pretraga se nastavlja u okolini $N_1(x^*)$. Ukoliko se pri pretrazi okoline $N_k(x^*)$, trenutno najbolje rešenje ne može popraviti, pretraga se nastavlja u narednoj okolini $N_{k+1}(x^*)$. Postupak se završava kada se x^* ne moze popraviti ni u jednoj okolini, što znači da je x^* lokalni minimum u odnosu na sve definisane okoline.

Opisan algoritam se može prikazati pseudo-šemom na Slici 1.1, gde je prikazana strategija najboljeg poboljšanja (eng. Best improvement strategy) tj. u svakoj okolini se pronalazi najbolje rešenje. Metoda se može realizovati i korišćenjem strategije prve popravke (eng. First improvement strategy) koja po pronalaženju prvog boljeg rešenja u trenutnoj okolini, prelazi u njega i pretragu nastavlja u N_1 okolini tog novog rešenja.

1.3.2 Redukovana metoda promenljivih okolina

Redukovana metoda promenljivih okolina (eng. Reduced Variable Neighborhood Search, RVNS) [46, 48] je nastala kao odgovor na pitanje kako pobeći iz dostignutog lokalnog minimuma i njegove okoline kako bi se našao neki drugi minimum. Najpre treba odrediti u kom pravcu treba tražiti bolje rešenje i najjednostavniji pristup je slučajni pravac. Drugo pitanje je koliko daleko treba ići u izabranom pravcu. Prirodno je istražiti najpre neku malu okolinu rešenja, međutim okoline lokalnih minimuma mogu biti dosta velike pa je neophodno ići dalje.

RVNS je jednostavnija od VND-a jer ne postoji proces lokalne pretrage čime se

```
/* Inicijalizacija */
Izabrati skup okolina N_k, k = 1,...,k_max koje će se koristiti za pretragu;
Na slučajan način izabrati početno rešenje x ∈ X i postaviti x* ← x;
Postaviti k ← 1;
repeat naredne korake until k > k_max
        /* Istraživanje okoline */
        Pronaći najbolje rešenje x' u okolini N_k(x*);
        /* Da li se pomeriti? */
        if f(x') < f(x*) then
                Postaviti x* ← x' i k ← 1;
        else
                Postaviti k ← k + 1;
end
```

Slika 1.1: Pseudo-šema VND algoritma

izbegava složeno i dugotrajno izvršavanje u okviru jedne VNS iteracije. Nakon izbora skupa okolina N_k, $k = 1, ...k_{\max}$ i početnog rešenja x ($x^* := x$), na slučajan način se iz prve okoline $N_1(x^*)$ bira tačka x', koja ne mora biti lokalni minimum. Ukoliko je vrednost funkcije cilja u toj tački manja od prethodno najboljeg rešenja, onda se pomeramo u tu tačku ($x^* := x'$) i postupak nastavljamo dalje iz nje. U slučaju da je $f(x') > f(x^*)$, pretraga se nastavlja u narednoj okolini. Korak u kom se na slučajan način bira tačka x' u okolini $N_k(x^*)$ naziva se razmrdavanje (eng. shaking). Nakon što se ispitaju sve okoline bez uspešnog poboljšanja, postupak se nastavlja od prve okoline, sve dok neki od kriterijuma zaustavljanja ne bude ispunjen. Kriterijum zaustavljanja može biti maksimalno vreme izvršavanja, maksimalni broj iteracija, maksimalni broj iteracija između dva poboljšanja ili neki drugi kriterijum, zavisno od konkretne primene.

Algoritam se može prikazati pseudo-šemom na Slici 1.2.

Moguća varijacija metode bi bila da se u svakom koraku razmrdavanja umesto jedne bira m slučajnih tačaka, gde je m unapred zadati parametar. Time se povećavaju šanse da se u datoj okolini dobije bolje rešenje. Ova metoda se često koristi za brzo dobijanje početnog (dovoljno dobrog) rešenja, odakle bi se detaljnijom pretragom dobilo kvalitetnije rešenje.

```
/* Inicijalizacija */
Izabrati skup okolina N_k, k = 1,...,k_max koje će se koristiti za pretragu;
Na slučajan način izabrati početno rešenje x ∈ X i postaviti x* ← x;
Odrediti kriterijum zaustavljanja;
repeat naredne korake until ispunjen je kriterijum zaustavljanja
    Postaviti k ← 1;
    repeat naredne korake until k > k_max
        /* Razmrdavanje */
        Na slučajan način izabrati tačku x' u okolini N_k(x*);
        /* Da li se pomeriti? */
        if f(x') < f(x*) then
            Postaviti x* ← x' i k ← 1;
        else
            Postaviti k ← k + 1;
    end
end
```

Slika 1.2: Pseudo-šema RVNS algoritma

1.3.3 Osnovna i opšta varijanta metode promenljivih okolina

Osnovna varijanta metode promenljivih okolina (eng. Basic Variable Neighborhood Search, BVNS) je nastala kao kombinacija prethodna dva pristupa. Algoritam ne prati unapred zadatu putanju nego istražuje različite okoline trenutno najboljeg rešenja i prelazi u novo samo u slučaju kada naiđe na bolje rešenje od trenutnog. Korišćenjem sistematske promene okolina, od najmanje ka najvećoj, pretraga se u jednoj okolini vrši tako što se najpre na slučajan način bira tačka u tekućoj okolini, a zatim se od nje primenjuje proces lokalne pretrage. Ova pretraga okolina se obavlja pomoću procedura razmrdavanja i lokalne pretrage (eng. local search).

U cilju efikasne primene metode, najpre je neophodno definisati prikladnu strukturu okolina. Neka je N_k $(k = 1, ..., k_{\max})$ konačan skup okolina takvih da je $N_k(x)$ skup svih rešenja u k-toj okolini rešenja x. Najjednostavniji i najčešći izbor strukture okolina je onaj kod kojih okoline imaju rastuću kardinalnost, tj. $|N_1(x)| < |N_2(x)| < \ldots < |N_{k_{\max}}(x)|$.

U svakom koraku algoritma, VNS započinje od nekog rešenja x^* i celog broja $k \in \{1, ..., k_{\max}\}$ koji se odnosi na trenutnu okolinu N_k. U procesu razmrdavanja na slučajan način se bira neko rešenje x' iz okoline $N_k(x^*)$. Zatim se, počevši od x', primenjuje lokalna pretraga za potencijalno boljim rešenjem x'' u nekoj okolini

```
/* Inicijalizacija */
Izabrati skup okolina N_k, k = k_1, ..., k_max koje će se koristiti za pretragu;
Na slučajan način izabrati početno rešenje x ∈ X i postaviti x* ← x, f* ← f(x);
Odrediti kriterijum zaustavljanja;
repeat naredne korake until ispunjen je kriterijum zaustavljanja
1:    Postaviti k ← 1;
      repeat naredne korake until k > k_max
            /* Razmrdavanje */
            Generisati slučajnu tačku x' ∈ N_k(x*) ;
            /* Lokalna pretraga */
            Primeniti neku metodu lokalnog pretraživanja u okolini rešenja x'
              kako bi se dobio lokalni minimum x'' problema;
            /* Da li se pomeriti? */
            if f(x'') < f* then
                  Postaviti x* ← x'', f* ← f(x'') i goto 1;
            else
                  Postaviti k ← k + 1;
      end
end
Tačka x* je aproksimativno rešenje problema.
```

Slika 1.3: Pseudo-šema BVNS algoritma

od x' (koja ne mora biti istog tipa kao i okolina za razmrdavanje). Najbolje rešenje dobijeno ovom lokalnom pretragom x'' se poredi sa x^*. Ukoliko je x'' bolje od x^*, onda x'' zauzima mesto trenutno najboljeg rešenja i proces pretrage se nastavlja za okolinu N_1 rešenja x''. U suprotnom, x^* ostaje najbolje rešenje do sad i algoritam nastavlja sa razmrdavanjem za sledeću okolinu.

Kada brojač okolina dostigne $k_{\max}$, pretraga se nastavlja dalje počevši od prve okoline N_1. Ovo se ponavlja dokle god se ne zadovolji neki kriterijum zaustavljanja. Najčešće su to maksimalno dozvoljeno vreme izvršavanja, maksimalni ukupan broj iteracija, maksimalni broj iteracija između dva poboljšanja, maksimalni broj ponavljanja najboljeg rešenja i sl. Koraci BVNS-a se mogu prikazati pseudo-šemom na Slici 1.3.

Ovo je najrasprostranjenija varijanta metode i postoje brojne njene modifikacije i proširenja. Jedan primer modifikacije je da se umesto razmrdavanja najpre u manjim okolinama, krene od većih. Time bi se umesto od $k \leftarrow 1$ krenulo od $k \leftarrow k_{\max}$ i umesto uvećavanja vrednosti parametra k u ovom slučaju bi se smanjivala njegova vrednost.

/* *Inicijalizacija* */
Izabrati skup okolina N_k, $k = k_1, \ldots, k_{\max}$ koje će se koristiti za razmrdavanje;
Izabrati skup okolina N_l, $l = k_1, \ldots, l_{\max}$ koje će se koristiti za lokalnu pretragu;
Na slučajan način izabrati početno rešenje $x \in X$ i postaviti $x^* \leftarrow x$, $f^* \leftarrow f(x)$;
Odrediti kriterijum zaustavljanja;
repeat naredne korake **until** ispunjen je kriterijum zaustavljanja
1: Postaviti $k \leftarrow 1$;
repeat naredne korake **until** $k > k_{\max}$
/* *Razmrdavanje* */
Generisati slučajnu tačku $x' \in N_k(x^*)$;
/* *Lokalna pretraga koristeci VND* */
Postavi $l \leftarrow 1$;
repeat naredne korake **until** $l > l_{\max}$
/* *Istraživanje okoline* */
Pronaći najbolje rešenje x'' u okolini $N_l(x')$;
/* *Da li se pomeriti?* */
if $f(x'') < f(x')$ **then**
Postaviti $x' \leftarrow x''$ i $l \leftarrow 1$;
else
Postaviti $l \leftarrow l + 1$;
/* *Da li se pomeriti?* */
if $f(x'') < f^*$ **then**
Postaviti $x^* \leftarrow x''$, $f^* \leftarrow f(x'')$ i **goto** 1;
else
Postaviti $k \leftarrow k + 1$;
end
end
Tačka x^* je aproksimativno rešenje problema.

Slika 1.4: Pseudo-šema GVNS algoritma

Ukoliko bi se procedura lokalne pretrage iz preudo-šeme 1.3 zamenila sa VND-om, dobila bi se Opšta metoda promenljivih okolina (eng. General Variable Neighborhood Search, GVNS), čiji koraci su prikazani pseudo-šemom na Slici 1.4.

1.3.4 Metoda promenljivih okolina sa dekompozicijom

Ukoliko bi se izvršilo razbijanje polaznog problema na manje potprobleme, koji su lakši za rešavanje, a zatim kombinovanjem rešenja tih potproblema dobilo rešenje polaznog, reč je o Metodi promenljivih okolina sa dekompozicijom (eng. Variable Neighborhood Decomposition Search, VNDS) [50].

Glavna razlika izmedju VNDS i BVNS algoritama je u tome što umesto primene lokalne pretrage u čitavom prostoru rešenja S, polazeći od $x' \in N_k(x^*)$, VNDS u

```
/* Inicijalizacija */
Izabrati skup okolina N_k, k = k_1, ..., k_max koje će se koristiti za pretragu;
Na slučajan način izabrati početno rešenje x i postaviti x* ← x, f* ← f(x);
Odrediti kriterijum zaustavljanja;
repeat naredne korake until ispunjen je kriterijum zaustavljanja
1:    Postaviti k ← 1;
      repeat naredne korake until k > k_max
            /* Razmrdavanje */
            Generisati tačku x' ∈ N_k(x*) ;
            Odrediti skup y atributa rešenja x' koji ne karakterišu x, y = x' \ x*;
            /* Lokalna pretraga */
            Primeniti neku metodu lokalnog pretraživanja u prostoru rešenja za y
                  i dobijeni optimum označiti sa y', a sa x'' odgovarajuće
                  rešenje za polazni problem, x'' = (x' \ x*) ∪ y';
            /* Da li se pomeriti? */
            if f(x'') < f* then
                  Postaviti x* ← x'' i goto 1;
            else
                  Postaviti k ← k + 1;
      end
end
Tačka x* je aproksimativno rešenje problema.
```

Slika 1.5: Pseudo-šema VNDS algoritma

svakoj iteraciji rešava potproblem u nekom potprostoru $V_k \subseteq N_k(x^*)$, $x' \in V_k$. Ovim postupkom se smanjuje vreme trajanja lokalne pretrage, što u nekim slučajevima (zavisno od prirode polaznog problema) može smanjiti vreme izvršavanja cele metode.

Pseudo-šema za VNDS je prikazana na slici 1.5. Neka je x^* trenutno rešenje i x' rešenje dobijeno u koraku razmrdavanja. U svakom koraku moguće je izdvojiti k atributa (npr. promenljivih) koji karakterišu rešenje x', ali ne i x^*. Neka je sa y označen skup svih takvih atributa, $y = x' \setminus x^*$. Lokalna pretraga se sada može realizovati u prostoru rešenja za y, tj. dozvoljava se jedino promena atributa iz skupa y, a zabranjuje izmena iz skupa $x^* \cap x' = x' \setminus y$. U slučaju da se lokalnom pretragom dobije poboljšanje y', onda to y' i skup atributa $x' \setminus y$ određuju novo rešenje x'' koje po prirodi treba da bude bolje od x^*.

Promenljiva k u ovom slučaju određuje okolinu u kojoj se vrši razmrdavanje, dok samo razmrdavanje određuje manji problem na koji se fokusiramo. Ako se lokalnom pretragom nije popravilo rešenje y, uvećava se k, $k \leftarrow k + 1$ ili $k \leftarrow k + k_{step}$, dok u suprotnom se postavlja $k \leftarrow 1$ ili $k \leftarrow k_{\min}$.

1.3.5 Zakošena metoda promenljivih okolina

Veliki broj problema ima grupisane lokalne minimume, međutim postoje i problemi kod kojih se lokalni minimumi nalaze vrlo daleko jedni od drugih, međusobno ograđeni prostranim "planinama". U takvim primerima razmrdavanjem i lokalnom pretragom ne može se doći do njih jer metoda dozvoljava samo prelaske iz lošijih u bolja rešenja. Ukoliko bi se koristile dosta veće okoline, koje bi obuhvatile oblasti u kojima su neki drugi lokalni minimumi, metoda bi se pretvorila u višestruko lokalno pretraživanje (eng. Multistart). Kompromisno rešenje se dobija Zakošenom metodom promenljivih okolina (eng. Skewed Variable Neighborhood Search, SVNS) [45].

U ovoj varijanti metode biće dozvoljen prelazak iz jednog lokalnog minimuma x^* u lošiji minimum x'' ukoliko se x'' dosta razlikuje od x^*. Mera različitosti dva rešenja može biti njihovo rastojanje za koje se koristi ista metrika kao i za definisanje okolina. Kriterijum za prelazak iz jednog rešenja u drugo koristi kombinaciju vrednosti funkcije cilja u tim rešenjima i njihovu udaljenost. Neka je sa $\rho(x^*, x'')$ označena udaljenost rešenja x^* od x'', npr. Hamingovo rastojanje ako su rešenja predstavljena kao vektori nula i jedinica ili Euklidsko rastojanje u slučaju kontinualnih problema. Prelazak iz rešenja x^* u x'' biće dozvoljeno ukoliko je $f(x'') - \alpha \cdot \rho(x^*, x'') < f(x^*)$, gde $f(\cdot)$ označava vrednost funkcije cilja, a α je unapred zadati parametar. Na slici 1.6 data je pseudo-šema za SVNS.

Parametar metode α se bira eksperimentalno tako da omogući pretraživanje okolina daleko od x kada je $f(x'')$ veće od $f(x)$. Na primer, da bi se izbeglo pomeranje od x ka bliskim rešenjima kada je $\rho(x, x'')$ malo, može se izabrati velika vrednost za parametar α i suprotno.

1.3.6 Paralelizacija i hibridizacija

Osnovni cilj paralelnog izvršavanja je da se ubrzaju izračunavanja deljenjem na više procesora. Strategije za paralelizaciju VNS metode opisane su u preglednom radu [84]. U radu [36] predloženo je nekoliko varijanti paralelizacije VNS metode. Jedna od mogućih strategija je paralelno izvršavanje procedure lokalne pretrage u cilju ubrzanja ovog najzahtevnijeg dela VNS metode. Druga varijanta je nezavisno izvršavanje više VNS procedura istovremeno, sa različitim početnim rešenjima i izbor najboljeg rešenja na kraju.

/* *Inicijalizacija* */
Izabrati skup okolina N_k, $k = k_1, \ldots, k_{\max}$ koje će se koristiti za pretragu;
Na slučajan način izabrati početno rešenje x i postaviti $x^* \leftarrow x$, $f^* \leftarrow f(x)$;
Odrediti kriterijum zaustavljanja i parametar α;
repeat naredne korake **until** kriterijum zaustavljanja nije ispunjen
1: Postaviti $k \leftarrow 1$;
repeat naredne korake **until** $k > k_{\max}$
/* *Razmrdavanje* */
Generisati tačku $x' \in N_k(x^*)$;
/* *Lokalna pretraga* */
Primeniti neku metodu lokalnog pretraživanja u okolini rešenja x'
kako bi se dobio lokalni minimum x'' problema;
/* *Poboljšanje ili ne?* */
if $f(x'') < f^*$ **then** postaviti $f^* \leftarrow f(x'')$ i $x^* \leftarrow x''$;
/* *Da li se pomeriti?* */
if $f(x'') - \alpha \cdot \rho(x, x'') < f(x)$ **then**
Postaviti $x \leftarrow x''$ i **goto** na 1;
else
Postaviti $k \leftarrow k + 1$;
end
end
Tačka x^* je aproksimativno rešenje problema.

Slika 1.6: Pseudo-šema SVNS algoritma

Autori rada [20] su izvršili paralelizaciju koristeći jedan nadređen i nekoliko podređenih procesora za pretraživanje više okolina. Nadređeni procesor vodi računa o kriterijumu zaustavljanja, ažuriranju rešenja, donosi odluke o usmeravanju dalje pretrage, dok podređeni procesori izvršavaju delove VNS metode počevši od procedure razmrdavanja.

U nekoliko radova [87, 8, 52] izvršena je i hibridizacija VNS metode sa drugim metaheuristikama. Međutim, hibridizaciju nije jednostavno postići a da se ne izgubi jednostavnost same VNS metode.

1.4 VNS pristup za probleme kontinualne optimizacije

Detaljan pregled različitih pristupa u rešavanju problema kontinualne optimizacije može se naći u [60, 83, 85, 92]. Metoda promenljivih okolina, iako najčešće korišćena za rešavanje problema kombinatorne optimizacije, poslednjih godina uspešno je primenjivana i na probleme kontinualne optimizacije. Višeizvorni Veber-ov problem (eng. Multisource Weber problem) drugačije nazvan i lokacijski-alokacijski problem (eng. Location-alocation problem) je bio prvi kontinualni problem za koji je predlozena primena VNS-a za njegovo rešavanje [13, 14]. Naredni problem kontinualne optimizacije koji je bio "napadnut" VNS-om bio je generalni problem bilinearnog programiranja (eng. General bilinear programming problem) [46].

Iako oba ova problema pripadaju kontinualnoj optimizaciji, glavni koraci za rešavanje su zapravo bili kombinatorne prirode. Na primer, za višeizvorni Veber-ov problem okolina neke tačke (kontinualnog rešenja) se može definisati ili premeštanjem objekata ili realokacijom potrošača. Na ovaj način ukupan broj rešenja je uvek konačan.

VNS pristup "čistoj" kontinualnoj optimizaciji je prvi put predložen u [78] gde je razmatrano rešavanje nelinearnog problema bez ograničenja za dizajn rasutog spektra polifaznog radara (eng. Spread spectrum radar polyphase code design problem). Kasnije je razvijen softverski paket GLOB za rešavanje nelinearnih problema sa intervalnim ograničenjima [76, 65], koji je usavršen u radu [25] dodavanjem više varijanti VNS-a koje korisnik ima na raspolaganju.

Za mnoge probleme kontinualne (kao i kombinatorne) optimizacije, VNS metoda

se pokazala uspešnijom od Genetskih algoritama i Tabu pretrage [48, 65, 78].

1.4.1 Glob - VNS

Posmatrajmo problem nelinerne globalne optimizacije bez ograničenja

$$\underset{x\in\mathbb{R}^n}{\text{global}\min} f(x) \tag{1.4.1}$$

gde je $f : \mathbb{R}^n \to \mathbb{R}$ neprekidna funkcija. U opštem slučaju f ne mora biti konveksna ili glatka funkcija, i može biti dobijena kao izlaz nekog numeričkog izračunavanja.

Ovakvi problemi su dosta rasprostranjeni u praksi, npr. u inženjerskim problemima dizajna proizvoda, analizi podataka, finansijskom planiranju, kontroli rizika, modelovanju u prirodnim naukama, hemiji itd. U većini slučajeva problemi ovog tipa se dosta teško rešavaju zbog postojanja velikog broja lokalnih minimuma čiji broj može eksponencijalno da raste sa povećanjem dimenzije problema. Sa druge strane, čak i za probleme manjih dimenzija za koje se rešenje može dobiti nekom od metoda koje garantuju nalaženje globalnog minimuma, teškoća je velika količina vremena koja je neophodna kako bi se došlo do tog rešenja. Stoga, razvijene su različite (meta)heuristike koje se oslanjaju na snagu računara. Detaljni pregled različith (meta)heuristika može se naći u [37].

Ideja korišćenja različitih geometrija za strukturu okoline i različitih raspodela slučajnih brojeva u koraku razmrdavanja dovela je do razvoja Glob-VNS [79], [25]. Pokazalo se da je Glob-VNS dosta efikasniji u odnosu na verijantu sa fiksiranim geometrijama i raspodelama.

Koraci Glob-VNS-a se mogu prikazati pseudo-šemom na slici 1.7.

Pre nego što se primeni Glob-VNS algoritam, mora se doneti odluka o nekoliko parametara metoda. Prvo, neophodan je izbor *geometrije* $\mathcal{G}$ strukture okolina $N_k(x)$, $k = 1, \ldots, k_{\max}$, $x \in \mathbb{R}^n$. Najčešći izbori su:

$$N_k(x) = \{y|\ \rho(x, y) \leq \rho_k\}, \tag{1.4.2}$$

ili

$$N_k(x) = \{y|\ \rho_{k-1} < \rho(x, y) \leq \rho_k\}. \tag{1.4.3}$$

/* *Inicijalizacija* */
Izabrati parove $(\mathcal{G}_l, \mathcal{P}_l),\ l = 1, \ldots, m$, tj. geometrije okolina i raspodele
  i postaviti poluprečnike $\rho_i,\ i = 1, \ldots, k_{\max}$;
Na slučajan način izabrati početno rešenje $x \in S$ i postaviti
  $x^* \leftarrow x$, $f^* \leftarrow f(x)$;
repeat naredne korake **until** ispunjen je kriterijum zaustavljanja
1: Postaviti $l \leftarrow 1$;
repeat naredne korake **until** $l > m$
Formirati okoline N_k, $k = 1, \ldots, k_{\max}$ koristeći
geometrijsku strukturu $\mathcal{G}_l$ i poluprečnik ρ_k;
Postaviti $k \leftarrow 1$;
repeat naredne korake **until** $k > k_{\max}$
/* *Razmrdavanje* */
Generisati slučajnu tačku $y \in N_k(x^*)$ koristeći
slučajnu raspodelu $\mathcal{P}_l$;
/* *Lokalna pretraga* */
Primeniti neku metodu lokalnog pretraživanja u okolini
rešenja y kako bi se dobio lokalni minimum y';
/* *Da li se pomeriti?* */
if $f(y') < f^*$ **then**
Postaviti $x^* \leftarrow y'$, $f^* \leftarrow f(y')$ i **goto** 1;
end
Postaviti $k \leftarrow k + 1$;
end
Postaviti $l \leftarrow l + 1$;
end
end
Tačka x^* je aproksimativno rešenje problema.

Slika 1.7: Pseudo-šema Glob-VNS algoritma

Metrika $\rho(\cdot)$ je obično neko ℓ_p rastojanje između dve tačke, $1 \leq p \leq \infty$, ([25], [68], [79], [78]). Najčešće se uzima da je $p = 1, 2$, ili ∞. Geometrija okolina $\mathcal{G}$ je na ovaj način određena izborom metrike $\rho(\cdot)$, a $N_k(x)$ je određeno sa $\mathcal{G}$ i ρ_k. U radovima [25] i [68] koriste se okoline definisane sa (1.4.3). U radu [68] koristi se ℓ_∞ norma, dok se u [25] izbor metrike prepušta korisniku ili se one menjaju automatski po nekom unapred definisanom redosledu. Radijusi ρ_k se menjaju monotono uvećavajući k čije vrednosti su ili unapred definisane od strane korisnika ili se automatski izračunavaju u procesu optimizacije.

Sa druge strane, neophodno je i definisati *raspodelu* slučajnih brojeva $\mathcal{P}$ koja će se koristiti za nalaženje slučajne tačke y iz okoline $N_k(x)$ u koraku razmrdavanja. Najčešće se koristi uniformna raspodela. Složenost izračunavanja za generisanje tačke na slučajan način u $N_k(x)$, korišćenjem uniformne raspodele, zavisi od geometrije ovog skupa. Određivanje slučajne tačke sa uniformnom raspodelom u okolini definisanoj l_∞ normom (n-dimenziona kocka) je trivijalno, ali u slučaju da se koristi neka od drugih normi ovaj proces se komplikuje. Na primer, za generisanje slučajnih tačaka uniformno raspoređenih u okviru jedinične kugle B mogu se koristiti različiti algoritmi. Jedan način je metoda probe i greške (eng. acceptance-rejection method) koja najpre generiše slučajnu tačku, uniformno raspoređenu u kocki $Q = [-1, 1]^n$, a zatim se vrši provera da li se tačka nalazi unutar ili izvan B. U slučaju kada je tačka van B, ona se odbacuje i proces se ponavlja sve dok se ne dobije tačka koja će biti unutar kugle B. Ova metoda je jednostavna za implementaciju, ali za veće dimenzije je dosta neefikasna. Kako sa porastom dimenzije odnos zapremina B i Q teži nuli, ovaj pristup nije pogodan za velike dimenzije. Alternativno, mogu se koristiti sferne koordinate kako bi se prevazišao ovaj problem, ali u tom slučaju algoritam koristi trigonometrijske funkcije čije izračunavanje značajno usporava algoritam.

Drugi, efikasniji način za generisanje slučajnih brojeva korišćenjem uniformne raspodele može se konstruisati u dva koraka:

- Arens - Diter algoritam ([1], [33]) se koristi za brzo generisanje jednodimenzione normalne slučajne promenljive. Generisanjem n-dimenzionog slučajnog vektora na ovaj način, nakon normiranja, dobija se tačka uniformno raspodeljena na jediničnoj sferi.

- Slučajan poluprečnik r se generiše uzimajući u obzir da je funkcija gustine za

r proporcionalna površini sfere poluprečnika r, tj. Cr^{n-1}. Funkcija raspodele $F(x)$ i njen inverz su jednostavni za računanje, te se r može dobiti kao $r = F^{-1}(u)$, gde je $u \in [0, 1]$ uniformno dobijena slučajna veličina.

Adekvatnom modifikacijom drugog koraka može se na efikasan način generisati i uniformno raspoređena tačka iz okoline tipa 1.4.3.

U Glob-VNS implementaciji, korisnik može da izabere geometrijske strukture $\mathcal{G}_l$ indukovane ℓ_1, ℓ_2 ili ℓ_∞ normom u (1.4.2) i (1.4.3), kao i za $\mathcal{P}_l$ uniformnu ili hipergeometrijsku raspodelu. Takođe, korisnik može da proizvoljno definiše broj i redosled kombinacija $(\mathcal{G}_l, \mathcal{P}_l)$, $l = 1, \ldots, m$.

U slučaju procedure lokalne pretrage, u literaturi se može naći više predloga. U radu [68] predložen je komercijalni solver SNOPT [38]. U [9] predložen je metod oblasti poverenja (eng. trust region), dok u Glob-VNS paketu [25] korisnik ima izbor šest različitih metoda lokalne pretrage.

Glava 2

Gausovska metoda promenljivih okolina za probleme kontinualne optimizacije

Iz razmatranja prethodne glave može se videti da su u velikom broju radova koji se sreću u literaturi postignuti veoma obećavajući rezultati primenom heuristika zasnovanih na VNS-u. S obzirom da je broj $k_{\max}$ različitih okolina koje se koriste konačan, nije moguće polazeći od trenutno najbolje tačke dostići svaku tačku iz dopustivog skupa (koji je podskup od $\mathbb{R}^n$). Zbog ovoga, postoji mogućnost da se ne dostigne do oblasti gde se može nalaziti globalni minimum. Drugim rečima, ova strategija je primenljiva kada problem koji se razmatra nije bez ograničenja (1.4.1), nego je sa intervalnim ograničenjima oblika:

$$\operatorname*{global\,min}_{x \in S} f(x), \tag{2.0.1}$$

gde je $S = \{x \in \mathbb{R}^n \mid a_i \leq x_i \leq b_i,\ i = 1, 2, \ldots, n\}$.

U slučaju problema nelinearne globalne optimizacije bez ograničenja (1.4.1), donja i gornja granica za promenljive su postavljene na proizvoljno velike negativne i pozitivne brojeve. U tom slučaju, poluprečnici ρ_k se biraju na taj način da niz okolina $N_k(x)$, $1 \leq k \leq k_{\max}$ omogućava da se dostigne bilo koja tačka iz ograničene oblasti S, pretpostavljajući da se optimalno rešenje problema nalazi u toj oblasti. Ovo predstavlja ozbiljnu manu postojećih verzija VNS-a. U slučaju da želimo veće okoline kako bismo dostigli do bilo koje tačke u S, moramo izabrati veće poluprečnike okolina. Najčešće za okolinu $k_{\max}$ se uzima poluprečnik

takav da okolina obuhvata ceo prostor pretrage S. Tako dobijeni poluprečnici mogu prouzrokovati manju efikasnost metode.

U ovom odeljku će biti opisana nova varijanta VNS-a koja prevazilazi ovo ograničenje. Umesto definisanja niza različitih fizičkih okolina $N_1(x), \ldots, N_{k_{\max}}(x)$ i procesa razmrdavanja u okolini $N_k(x)$, može se uzeti da su sve okoline jednake celom prostoru rešenja i definisati niz *raspodela* $\mathcal{P}_1(x), \ldots, \mathcal{P}_{k_{\max}}(x)$ koje će se koristiti u postupku razmrdavanja. Jednostavnosti radi, može se uzeti za svaku $\mathcal{P}_k(x)$ da je n-dimenziona normalna raspodela centrirana u x. Ovakva varijanta VNS-a naziva se *Gausovska metoda promenljivih okolina, Gaus-VNS* i prvi put je predložena u radu E. Carrizosa, M. Dražić, Z. Dražić, N. Mladenović[15].

Koristeći ovaj pristup, teorijki je moguće "skočiti" iz tekućeg najboljeg rešenja u bilo koju tačku celog prostora rešenja. Time je moguće dostići do oblasti gde se nalazi globalni minimum polazeći od bilo koje tačke. Štaviše, sa adekvatnim izborom matrica kovarijansi, veći prioritet se može dati pravcima pretrage za koje je veća verovatnoća da se tu nalazi globalni ili lokalni optimum. Uzimajući za matricu kovarijansi dijagonalnu matricu svi pravci se tretiraju ravnopravno.

2.1 Opis Gaus-VNS

Glavna ideja Gaus-VNS-a je da se okoline tačke x, $\{N_k(x)\}_{1\leq k\leq k_{\max}}$, zamene raspodelama verovatnoća $\{\mathcal{P}_k(x)\}_{1\leq k\leq k_{\max}}$. U koraku razmrdavanja sledeća slučajna tačka će se birati koristeći raspodelu verovatnoća $\mathcal{P}_k(x)$.

Ukoliko bi se za $\mathcal{P}_k(x)$ izabrala uniformna raspodela sa nosačem $N_k(x)$, tada bi se dobio klasičan pristup VNS-a. Za raspodele sa neograničenim nosačima prirodan izbor je višedimenziona Gausova raspodela slučajnih brojeva. Pretpostavimo u daljem tekstu da je $\mathcal{P}_k(x)$ višedimenziona Gausova raspodela sa očekivanjem x i matricom kovarijansi Σ_k. Drugim rečima, slučajna tačka u procesu razmrdavanja se generiše n-dimenzionim slučajnim vektorom y sa funkcijom gustine:

$$\varphi(y) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(y-x)^\top \Sigma_k^{-1}(y-x)}.$$

Posmatrano iz ugla koji se odnosi na implementaciju, veoma je važno da se koristi efikasan generator slučajnih brojeva sa Gausovom raspodelom. Slučajne veličine sa gustinom raspodele $\mathcal{P}_k(x)$ se mogu lako dobiti od n nezavisnih vrednosti $z_1, ..., z_n$

dobijenih Gausovom raspodelom sa matematičkim očekivanjem 0 i disperzijom 1, za koju postoje generatori pseudoslučajnih brojeva. Ako je $\Sigma_k = L_k L_k^\top$ Čoleski dekompozicija simetrične i pozitivno definitne matrice Σ_k, onda je slučajan vektor dobijen kao $y = x + L_k z$, gde je $z = (z_1, ..., z_n)$, vektor sa nezavisnim slučajnim veličinama sa raspodelom $N(0, 1)$. Odavde, generisanje slučajnog vektora, korišćenjem $\mathcal{P}_k(x)$, se svodi na računanje Čoleski dekompozicije matrice Σ_k, a zatim generisanja n nezavisnih brojeva korišćenjem normalne raspodele sa očekivanjem 0 i disperzijom 1. Postupak se dodatno olakšava ukoliko se za matricu kovarijansi Σ_k uzme umnožak jedinične matrice I, tj.

$$\Sigma_k = \sigma_k^2 I, \qquad k = 1, 2, \ldots, k_{\max}, \tag{2.1.1}$$

jer je u ovom slučaju Čoleski dekompozicija jednostavno $\Sigma_k = (\sigma_k I)(\sigma_k I)^\top$. Tada su koordinate z_i slučajnog vektora z jednodimenione nezavisne Gausove slučajne veličine sa očekivanjem 0 i disperzijom σ_k.

Nakon razmrdavanja i na opisan način dobijanja slučajne tačke iz celog prostora pretrage (S ili $\mathbb{R}^n$), primenjuje se lokalna pretraga u cilju dobijanja poboljšanog rešenja. Za dobijanje lokalnog minimuma koristi se neka od sledećih metoda poznatih iz literature:

- Nelder - Mid (NM) metoda [80] spada u klasu metoda direktne pretrage koje ne koriste izvode funkcije. U te svrhe se koristi simpleks, poliedar određen sa $n + 1$ tačkom. U svakoj od iteracija, jedna od njegovih tačaka se odbacuje i zamenjuje novom dobijajući na taj način novi simpleks. NM koristi tri probna koraka: refleksiju, ekspanziju i kontrakciju. Ovim koracima se formiraju tri tačke od kojih jedna od njih zamenjuje najlošiju tačku simpleksa. Ukoliko ni jedna od ovih novih tačaka ne zaslužuje da zameni najlošiju, najlošijih n tačaka se pomera ka najboljoj. Ovaj postupak se ponavlja dok simpleks ne postane dovoljno mali ili se ne ispuni neki drugi kriterijum zaustavljanja.

- Huk - Dževis (HJ) metoda [59] sastoji se od dva koraka. Ideja je da se u svakoj iteraciji odrede smerovi u kojima grafik funkcije ima "brda" i "doline", a zatim se "doline" slede do nove tačke. Ovo se postiže tako što se iz trenutne tačke u svakoj iteraciji naprave "koraci" duž svake od osa. Korak je uspešan ako se vrednost funkcije smanjila, a neuspešan ako se vrednost funkcije povećala ili

ostala ista.

- Rozenbrok (RO) metoda [89] u svakoj iteraciji vrši pretraživanje duž svakog od n ortogonalnih pravaca, gde je n broj promenljivih u funkciji cilja. Pomeranje tačke duž jedne ose vrši se za unapred zadati korak. Ukoliko funkcija cilja ima manju vrednost od vrednosti u prethodnoj tački, u sledećem pretraživanju duž istog pravca vrednost za koju se pomeramo se množi sa poznatim koeficijentom $\alpha > 1$. U slučaju da poboljšanje nije postignuto, u sledećem pretraživanju duž tog pravca se vrednost za koju se pomeramo množi sa $-\beta,\ 0 < \beta < 1$. Ova procedura se nastavlja sve dok se u svakom pravcu ne pojavi jedno uspešno i jedno neuspešno pretraživanje.

- Metoda najstrmijeg spusta (eng. Steepest Descent - SD) polažeći od proizvoljne početne tačke X_0 najpre određuje vrednost gradijenta u toj tački $\nabla F(X_0)$. Gradijent pokazuje smer u kom funkcija najbrže raste u okolini te tačke. Kako je u ovom slučaju zadatak da se odredi minimum (a ne maksimum) funkcije, smer najbržeg opadanja je određen antigradijentom $-\nabla F(X_0)$. Funkcija se pretražuje u tom smeru, tj. rešava se problem jednodimenzionalne optimizacije. Dobijeno rešenje se prosleđuje kao inicijalno za narednu iteraciju.

- Flečer - Pauel (FP) metoda [35] sastoji se od toga da kada se u ciklusu dođe do tacke X_k, najpre se određuje u njoj vrednost gradijenta i Hesijan matrice H_k. Zatim se vrsi jednodimenziono pretraživanje kako bi se odredili skalari S_k kao minimum duž pravca $-H_k \cdot \nabla F(X_k)$. Primenom obrasca $X_{k+1} = X_k - S_k H_k \cdot \nabla F(X_k),\ k = 0, 1, ...$ nalazi se sledeća tačka. Matrica H_k određuje se iz jednakosti koju su predložili Flečer i Pauel: $H_{k+1} = H_k + \frac{P_k P_k^T}{<P_k, y_k>} - \frac{H_k y_k (H_k y_k)^T}{<y_k, H_k y_k>},\ k = 0, 1, ...$ gde je H_0 proizvoljna simetrična pozitivno definitna matrica, $P_k = X_{k+1} - X_k$ i $y_k = \nabla F(X_{k+1}) - \nabla F(X_k)$.

- Flečer - Rivs (FR) metoda [34, 95] pripada klasi metoda konjugovanih pravaca u kojoj se ne koristi Hesijan (matrica drugih izvoda) već samo gradijent funkcije. Ova se metoda može shvatiti i kao specijalan slučaj metode promenljive metrike.

Prve tri metode lokalne pretrage ne zahtevaju gradijent funkcije i mogu se koristiti i kada funkcija cilja nije glatka. Za preostale tri metode neophodan je podatak o gradijentu funkcije koji može biti ili zadat od strane korisnika ili izračunat

```
/* Inicijalizacija */
Izabrati skup matrica kovarijansi Σ_k, k = 1,...,k_max ;
Na slučajan način izabrati početno rešenje x ∈ S ;
Postaviti x* ← x, f* ← f(x);
repeat naredne korake until ispunjen je kriterijum zaustavljanja
1:    Postaviti k ← 1;
      repeat naredne korake until k > k_max
            /* Razmrdavanje */
            Generisati y koristeći Gausovu raspodelu slučajnih brojeva
                 sa očekivanjem x* i matricom kovarijansi Σ_k;
            /* Lokalna pretraga */
            Primeniti neku metodu lokalnog pretraživanja u okolini
             rešenja y kako bi se dobio lokalni minimum y';
            /* Da li se pomeriti? */
            if f(y') < f* then
                  Postaviti x* ← y', f* ← f(y') i goto 1;
            end
            Postaviti k ← k + 1;
      end
end
Tačka x* je aproksimativno rešenje problema.
```

Slika 2.1: Pseudo-šema Gaus-VNS algoritma

korišćenjem neke od numeričkih metoda. Za efikasnu implementaciju navedenih metoda lokalne pretrage korišćeni su algoritmi preuzeti iz [66] i [95].

Postupak lokalne pretrage se završava kada se ispuni neki od sledećih uslova: broj iteracija dostigne maksimum (npr. u slučaju NM), tačke dobijene u dve uzastopne iteracije su na rastojanju manjem od neke unapred zadate vrednosti ls_eps, razlika vrednosti funkcije u tačkama dobijenim u dve uzastopne iteracije je manja od unapred zadate vrednosti ls_fun_eps ili norma gradijenta je manja od unapred zadate vrednosti ls_grad_eps.

Poredeći Gaus-VNS sa Glob-VNS-om, Gaus-VNS ima manje parametara koje treba unapred definisati. Za Glob-VNS korisnik mora unapred da odredi broj i kombinaciju geometrije i okoline, tj. parove $(\mathcal{G}_l, \mathcal{P}_l)$, $l = 1, \ldots, m$, kao i poluprečnike ρ_k, $k = 1, \ldots, k_{\max}$. U slučaju Gaus-VNS-a sve okoline su iste, jednake S ili $\mathbb{R}^n$ i koristi se samo jedna familija raspodela - Gausova raspodela slučajnih brojeva. Sa očiglednim izborom $\Sigma_k = \sigma_k^2 I$ neophodno je odrediti samo varijanse σ_k, $k = 1, \ldots, k_{\max}$. Koraci opisanog algoritma dati su pseudo-šemom na slici 2.1.

2.2 Eksperimentalni rezultati

U ovom odeljku predstavljena su poređenja rezultata dobijenih Gaus-VNS-om sa rezultatima dobijenim prethodnim heuristikama zasnovanim na VNS-u za rešavanje problema globalne optimizacije. Najpre se vrše poređenja rezultata sa uspešnim VNS heuristikama iz literature, a zatim sa rezultatima drugih metaheuristika.

Sva testiranja su izvršena na Intel Core i3-2350M CPU 2.30GHz sa 4GB RAM memorije pod Windows 7 operativnim sistemom. Algoritam je kodiran u programskom jeziku C.

Pri rešavanju nelinearnih problema globalne optimizacije najčešća mera kvaliteta metode se zasniva na ukupnom broju računanja vrednosti funkcije do trenutka kada se optimalno (ili najbolje) rešenje dostiglo. Međutim, računska složenost algoritma zavisi i od broja ulaznih parametara poput tolerancije za kriterijum zaustavljanja ili od izbora metode lokalne pretrage. Ukupno vreme izvršavanja je postavljeno tako da globalni minimum bude pronađen u svakom izvršavanju.

Ulazni parametri koji su dali najbolje rezultate za Glob-VNS dati su u [25]. Radi korektnosti poređenja isti parametri su korišćeni i za Gaus-VNS. Parametar $k_{\max}$ je i za Glob-VNS i Gaus-VNS podešen na $k_{\max} = 5$ i sve tolerancije za procedure lokalne pretrage su postavljene na 10^{-4}. Poluprečnici okolina ρ_k kod Glob-VNS-a, odnosno disperzije σ_k kod Gaus-VNS-a su usklađivani za svaku instancu pojedinačno. Na primer, neki od izbora za σ_k, $k = 1, ..., 5$, su (0.1, 0.2, 0.3, 0.4, 0.5) ili (0.1, 0.5, 1.0, 3.0, 5.0). Usklađivanje disperzija σ_k za svaku instancu pojedinačno nije neophodno, međutim ukoliko se one prilagode problemu koji se rešava može se očekivati brža konvergencija. U slučaju kada je problem takav da funkcija ima gusto raspoređene lokalne minimume pogodnije su manje vrednosti za σ_k, dok u slučaju kada su lokalni minimumi na većoj udaljenosti jedni od drugih pogodnije su veće vrednosti. Što se tiče izbora metoda za lokalnu pretragu korišćene su one metode koje su u Glob-VNS-u dale najbolje rezultate.

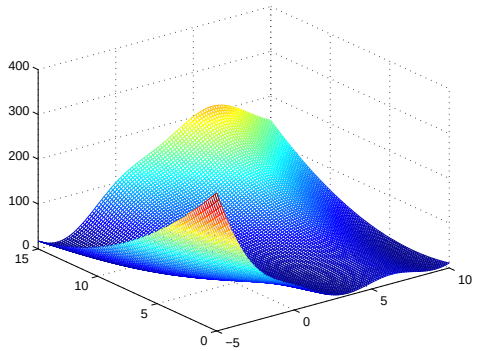
2.2.1 Standardne test funkcije

Za prvi deo testiranja korišćene su standardne test funkcije iz literature [9, 54]. Ove funkcije su izabrane iz dva razloga. Prvi je što su iste funkcije u literaturi korišćene za testiranje i upoređivanje rezultata. Drugi razlog, prema autorima [54], karakteristike

ovih test funkcija su dovoljno raznolike da pokrivaju veliki broj problema koji se javljaju u globalnoj optimizaciji. Osobine ovih funkcija su raznolike: posmatraju se funkcije koje imaju nekoliko izolovanih lokalnih minimuma poput Goldštajn i Prajs funkcije, funkcije sa mnogo nagomilanih lokalnih minimuma poput Šubert funkcije, funkcije čiji globalni minimum leži u veoma uskom otvoru (poput bunara) kao što je slučaj sa Isom funkcijom ili funkcije koje imaju oblik vrlo uzane doline (poput kanjona) kao što je slučaj sa Rozenbrok funckcijom.

Branin RCOS funkcija (RC)

- Broj promenljivih: $n = 2$.
- Definicija: $RC(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$.
- Opseg promenljivih: $-5 < x_1 < 10,\ 0 < x_2 < 15$.
- Broj lokalnih minimuma: osim globalnih nema drugih lokalnih minimuma.
- Globalni minimum: $(x_1, x_2)^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$.
- Vrednost funkcije u globalnom minimumu: $RC((x_1, x_2)^*) = 0.397887$.

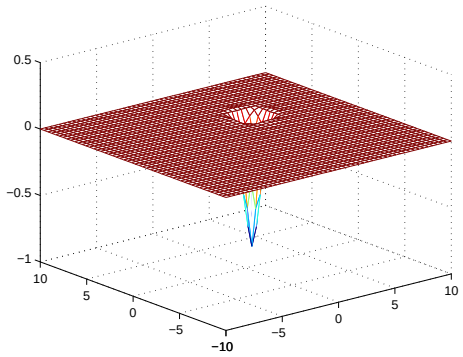


Slika 2.2: Branin funkcija na $[-5, 10] \times [0, 15]$

Isom funkcija (ES)

- Broj promenljivih: $n = 2$.
- Definicija: $ES(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$.
- Opseg promenljivih: $-10 < x_j < 10,\ j = 1, 2$.

- Broj lokalnih minimuma: više lokalnih minimuma.
- Globalni minimum: $(x_1, x_2)^* = (\pi, \pi)$.
- Vrednost funkcije u globalnom minimumu: $ES((x_1, x_2)^*) = -1$.



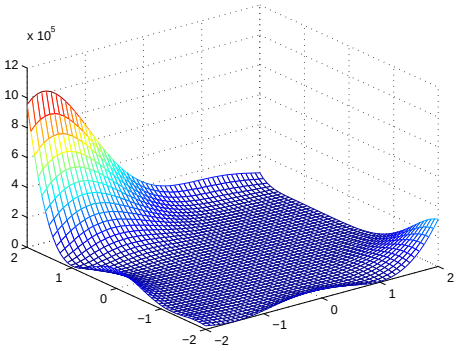
Slika 2.3: Isom funkcija na $[-10, 10] \times [-10, 10]$

Goldštajn i Prajs funkcija (GP)

- Broj promenljivih: $n = 2$.
- Definicija: $GP(x_1, x_2) = u * v$, gde je:
  $u = 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)$,
  $v = 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)$.
- Opseg promenljivih: $-2 < x_j < 2,\ j = 1, 2$.
- Broj lokalnih minimuma: 4 lokalna minimuma.
- Globalni minimum: $(x_1, x_2)^* = (0, -1)$.
- Vrednost funkcije u globalnom minimumu: $GP((x_1, x_2)^*) = 3$.

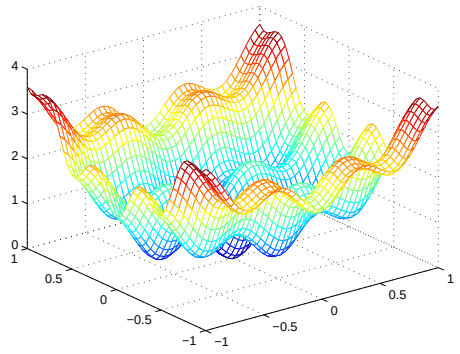
Rastrigin funkcija (RT)

- Broj promenljivih: $n = 2$.
- Definicija: $RT(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$.
- Opseg promenljivih: $-1 < x_j < 1,\ j = 1, 2$.



Slika 2.4: Goldštajn i Prajs funkcija na $[-2, 2] \times [-2, 2]$

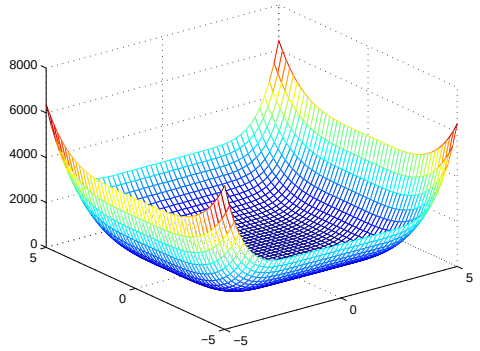
- Broj lokalnih minimuma: mnogo lokalnih minimuma.
- Globalni minimum: $(x_1, x_2)^* = (0, 0)$.
- Vrednost funkcije u globalnom minimumu: $RT((x_1, x_2)^*) = 0$.



Slika 2.5: Rastrigin funkcija na $[-1, 1] \times [-1, 1]$

Hamp funkcija (HM)

- Broj promenljivih: $n = 2$.
- Definicija: $HM(x_1, x_2) = 1.0316285 + 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$.
- Opseg promenljivih: $-5 < x_j < 5,\ j = 1, 2$.
- Broj lokalnih minimuma: osim globalnih nema drugih lokalnih minimuma.
- Globalni minimum: $(x_1, x_2)^* = (0.0898, -0.7126), (-0.0898, 0.7126)$.

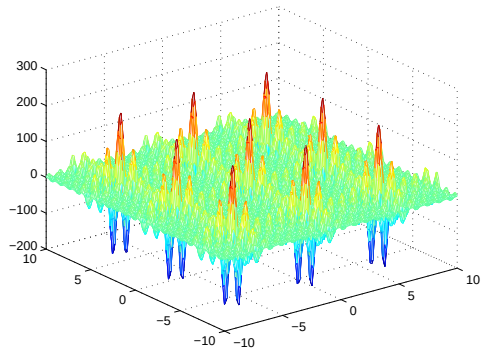


Slika 2.6: Hamp funkcija na $[-5, 5] \times [-5, 5]$

- Vrednost funkcije u globalnom minimumu: $HM((x_1, x_2)^*) = 0$.

Šubert funkcija (SH)

- Broj promenljivih: $n = 2$.
- Definicija: $SH(x_1, x_2) = (\sum_{j=1}^{5} j \cos((j+1)x_1 + j))(\sum_{j=1}^{5} j \cos((j+1)x_2 + j))$.
- Opseg promenljivih: $-10 < x_j < 10,\ j = 1, 2$.
- Broj lokalnih minimuma: 760 lokalnih minimuma.
- Globalni minimum: 18 globalnih minimuma
- Vrednost funkcije u globalnom minimumu: $SH((x_1, x_2)^*) = -186.7309$.



Slika 2.7: Šubert funkcija na $[-10, 10] \times [-10, 10]$

De Džung funkcija (DJ)

- Broj promenljivih: $n = 3$.
- Definicija: $DJ(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$.
- Opseg promenljivih: $-5 < x_j < 5,\ j = 1, 2, 3$.
- Broj lokalnih minimuma: osim globalnog nema drugih lokalnih minimuma.
- Globalni minimum: $(x_1, x_2, x_3)^* = (0, 0, 0)$.
- Vrednost funkcije u globalnom minimumu: $DJ((x_1, x_2, x_3)^*) = 0$.

Hartman funkcija ($H_{3,4}$)

- Broj promenljivih: $n = 3$.
- Definicija: $H_{3,4}(x_1, x_2, x_3) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2]$

Tabela 2.1: Parametri za Hartman funkciju.

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0381	0.5743	0.8828

- Opseg promenljivih: $0 < x_j < 1,\ j = 1, 2, 3$.
- Broj lokalnih minimuma: 4 lokalna minimuma.
- Globalni minimum: $(x_1, x_2, x_3)^* = (0.114614, 0.555649, 0.852547)$.
- Vrednost funkcije u globalnom minimumu: $H_{3,4}((x_1, x_2, x_3)^*) = -3.86278$.

Kolvil funkcija (CV)

- Broj promenljivih: $n = 4$.
- Definicija: $CV(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$.

- Opseg promenljivih: $-10 < x_j < 10,\ j = 1, 2, 3, 4.$
- Broj lokalnih minimuma: osim globalnog nema drugih lokalnih minimuma.
- Globalni minimum: $\mathbf{x}^* = (1, 1, 1, 1)$.
- Vrednost funkcije u globalnom minimumu: $CV(\mathbf{x}^*) = 0$.

Šekel funkcije ($S_{4,m}$)

- Broj promenljivih: $n = 4$.
- Definicija: $S_{4,m}(\mathbf{x}) = -\sum_{i=1}^{m}[\sum_{j=1}^{4}(x_j - a_{ij})^2 + c_i]^{-1}$

Tabela 2.2: Parametri za Šekel funkcije.

i	a_{ij}				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

- Opseg promenljivih: $0 < x_j < 10,\ j = 1, 2, 3, 4.$
- Broj lokalnih minimuma: m lokalnih minimuma.
- Globalni minimum: isti za $S_{4,5}, S_{4,7}, S_{4,10}$, $\mathbf{x}^* = (4, 4, 4, 4)$.
- Vrednosti funkcija u globalnom minimumu: $S_{4,5}(\mathbf{x}^*) = -10.1532$, $S_{4,7}(\mathbf{x}^*) = -10.4029$, $S_{4,10}(\mathbf{x}^*) = -10.5364$.

Hartman funkcija ($H_{6,4}$)

- Broj promenljivih: $n = 6$.

Tabela 2.3: Parametri za Hartman funkciju.

i	a_{ij}						c_i
1	10.00	3.00	17.00	3.50	1.70	8.00	1.0
2	0.05	10.00	17.00	0.10	8.00	14.00	1.2
3	3.00	3.50	1.70	10.0	17.00	8.00	3.0
4	17.00	8.00	0.05	10.00	0.10	14.00	3.2

i	p_{ij}					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

- Definicija: $H_{6,4}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]$

- Opseg promenljivih: $0 < x_j < 1,\ j = 1, ..., 6.$

- Broj lokalnih minimuma: 6 lokalnih minimuma.

- Globalni minimum:
  $\mathbf{x}^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300).$

- Vrednost funkcije u globalnom minimumu: $H_{6,4}(\mathbf{x}^*) = -3.32237.$

Grivank funkcija (GR)

- Broj promenljivih: $n = 6.$

- Definicija: $GR(\mathbf{x}) = \sum_{j=1}^{6} \frac{x_j^2}{4000} - \prod_{j=1}^{6} \cos(\frac{x_j}{\sqrt{j}}) + 1.$

- Opseg promenljivih: $-1 < x_j < 1,\ j = 1, ..., 6.$

- Broj lokalnih minimuma: mnogo lokalnih minimuma.

- Globalni minimum: $\mathbf{x}^* = (0, 0, 0, 0, 0, 0).$

- Vrednost funkcije u globalnom minimumu: $GR(\mathbf{x}^*) = 0.$

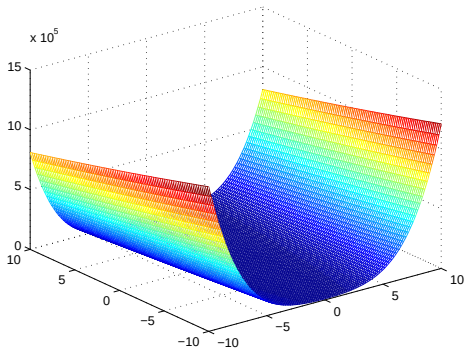
Dikson funkcija (DX)

- Broj promenljivih: $n = 10.$

- Definicija: $DX(\mathbf{x}) = (1 - x_1)^2 + (1 - x_{10})^2 + \sum_{j=1}^{9}(x_i^2 - x_{i+1})^2$.
- Opseg promenljivih: $-10 < x_j < 10,\ j = 1, ..., 10$.
- Broj lokalnih minimuma: osim globalnog nema drugih lokalnih minimuma.
- Globalni minimum: $\mathbf{x}^* = (1, ..., 1)$.
- Vrednost funkcije u globalnom minimumu: $DX(\mathbf{x}^*) = 0$.

Rozenbrok funkcije (R_n)

- Broj promenljivih: $n = 2, 5, 10$.
- Definicija: $R_n(\mathbf{x}) = \sum_{j=1}^{n-1}[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$.
- Opseg promenljivih: $-5 < x_j < 10,\ j = 1, 2, ..., n$.
- Broj lokalnih minimuma: osim globalnog nema drugih lokalnih minimuma.
- Globalni minimum: $\mathbf{x}^* = (1, ..., 1)$.
- Vrednost funkcije u globalnom minimumu: $R_n(\mathbf{x}^*) = 0$.

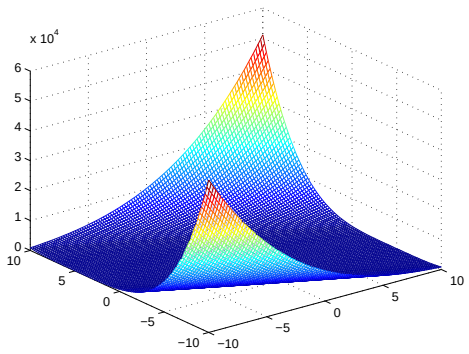


Slika 2.8: Rozenbrok funkcija za n=2 na $[-10, 10] \times [-10, 10]$

Zaharov funkcije (Z_n)

- Broj promenljivih: $n = 2, 5, 10$.
- Definicija: $Z_n(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 + (\sum_{j=1}^{n} 0.5jx_j)^2 + (\sum_{j=1}^{n} 0.5jx_j)^4$.

- Opseg promenljivih: $-5 < x_j < 10,\ j = 1, 2, ..., n.$
- Broj lokalnih minimuma: osim globalnog nema drugih lokalnih minimuma.
- Globalni minimum: $\mathbf{x}^* = (0, ..., 0)$.
- Vrednost funkcije u globalnom minimumu: $Z_n(\mathbf{x}^*) = 0$.



Slika 2.9: Zaharov funkcija za n=2 na $[-10, 10] \times [-10, 10]$

2.2.2 Eksperimentalni rezultati i poređenja na standardnim test funkcijama

Za pregled rezultata korišćena je računska složenost, tj. ukupan broj računanja vrednosti funkcije plus n puta broj računanja vrednosti gradijenta sve dok algoritam ne pronađe globalni optimum. Program je izvršavan po 10 puta za svaki primer i za krajnji rezultat su uzete prosečne vrednosti dobijene u svih 10 izvršavanja.

Rezultati su predstavljeni u Tabeli 2.4 koja je organizovana na sledeći način:

- U prve dve kolone nalazi se ime funkcije i odgovarajuća oznaka funkcije.
- U trećoj koloni je dimenzija problema tj. ukupan broj promenljivih.
- Naredne dve kolone se odnose na rezultate poznate iz literature: $VNS-1$ [9] i $VNS-2$ [91]. Rezultati su predstavljeni računskom složenošću za dobijanje optimalnog rešenja. Treba napomenuti da ovi rezultati nisu direktno uporedivi sa Gaus-VNS-om zbog drugačijeg kriterijuma zaustavljanja i tolerancije korišćene u određenim delovima programa.

- Kolona označena sa *lokal minim.* sadrži informaciju o tome koja metoda lokalne pretrage je korišćena za Glob-VNS i Gaus-VNS.
- Naredne dve kolone sadrže prosečnu računsku složenost algoritama Glob-VNS i Gaus-VNS za dobijanje optimalnih rešenja u 10 izvršavanja.
- Poslednja kolona sadrži odnos rezultata iz prethodne dve kolone, izračunatu korišćenjem formule $\frac{f_{Glob-VNS}-f_{Gaus-VNS}}{f_{Gaus-VNS}} \cdot 100\%$

Najbolji rezultat, tj. najmanji broj u tabeli, koji predstavlja računsku složenost za dobijanje optimalnog rešenja, je prikazan podebljano.

Tabela 2.4: Eksperimentalni rezultati za standardne test funkcije

funkcija		n	Računska složenost VNS-1	Računska složenost VNS-2	lokal minim.	Računska složenost Glob-VNS	Računska složenost Gaus-VNS	% poboljšanje
Branin	RC	2	153	308	FR	131	**112**	16.96%
Isom	ES	2	167	–	HJ	163	**148**	10.14%
Goldštajn i Prajs	GP	2	–	206	NM	260	**116**	124.14%
Rastrigin	RA	2	246	–	RO	206	**199**	3.52%
Hamp	HM	2	335	–	NM	160	**80**	100.00%
Šubert	SH	2	**366**	–	FR	382	591	-35.36%
De Džung	DJ	3	104	–	FP	38	**26**	46.45%
Hartman	H3,4	3	249	521	NM	246	**223**	10.31%
Hartman	H6,4	6	735	1244	HJ	**397**	448	-11.38%
Kolvil	CV	4	854	–	NM	669	**497**	34.61%
Šekel	S4,10	4	590	988	SD	599	**399**	50.13%
Grivank	GR	6	807	–	SD	135	**126**	7.14%
Dikson	DX	10	2148	–	FP	1640	**1576**	4.06%
Rozenbrok	R2	2	556	–	NM	158	**125**	26.40%
Rozenbrok	R5	5	**1120**	–	NM	1286	1308	-1.68%
Rozenbrok	R10	10	2653	–	FP	**2357**	2561	-7.97%
Rozenbrok	R50	50	**11934**	–	FR	38621	37901	1.90%
Rozenbrok	R100	100	**30165**	–	FR	147274	122446	20.28%
Zaharov	Z2	2	251	–	FR	179	**133**	34.59%
Zaharov	Z5	5	837	–	FR	728	**461**	57.92%
Zaharov	Z10	10	1705	–	FR	1142	**1010**	13.07%
Zaharov	Z50	50	17932	–	FR	**4304**	5302	-17.28%
							Prosek	22.17%

Značajno bolji rezultat algoritma $VNS-1$ za funkcije R50 i R100 su posledica korišćenja bolje metode lokalne pretrage, tj. skraćenog (eng. truncated) algori-

tama konjugovanih gradijenata za problem oblasti poverenja (eng. trust region). Međutim, Rozenbrok funkcije imaju samo jedan lokalni (ujedno i globalni) minimum i stoga nisu najbolji primer za procenu efikasnosti algoritama za globalnu optimizaciju.

Kao što se može videti iz Tabele 2.4, Gaus-VNS je u većini slučajeva dobio bolje rezultate od Glob-VNS-a. Cilj ovih testova je bio direktno poređenje navedenih algoritama, koristeći iste parametre, radi dobijanja realističnije slike. Iz tog razloga, navedene rezultate dobijene Gaus-VNS-om ne treba smatrati najboljima, jer bi se dodatnim podešavanjima parametara mogli postići i bolji rezultati.

2.2.3 Eksperimentalni rezultati i poređenja na test funkcijama velikih dimenzija

Dimenzije standardnih test funkcija iz Tabele 2.4 su dosta male, dok VNS može da postigne rezultate i za primere dosta većih dimenzija. Iz tog razloga, ponašanje Gaus-VNS-a je testirano i na problemima većih dimenzija, takođe preuzetih iz literature.

Rastrigin funkcija (RAn)

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right),$$

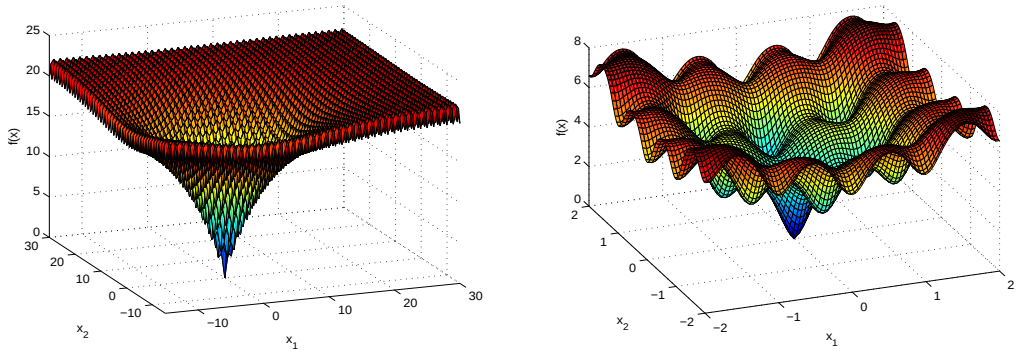
$$-5.12 \leq x_i \leq 5.12, \quad i = 1, \ldots, n, \quad f_{\min} = 0.$$

Funkcija potencijalne energije molekula (eng. Molecular potencial energy function) ($MPEn$) [67, 26]

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \left(1 + \cos 3x_i + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682\cos x_i}}\right), \\ & 0 \leq x_i \leq 5, \quad i = 1, \ldots, n, \quad f_{\min} = -0.0411183034 \cdot n. \end{aligned}$$

Ekli funkcija (ACn) [62, 61]

$$\begin{aligned} f(x) &= 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\cos(2\pi x_i)\right), \\ & -15 \leq x_i \leq 30, \quad i = 1, \ldots, n, \quad f_{\min} = 0. \end{aligned}$$



Slika 2.10: Ekli funkcija na [-15,30]×[-15,30] i $[-2, 2] \times [-2, 2]$.

Navedene funkcije imaju eksponencijalni broj lokalnih minimuma: Rastrigin funkcija ima 11^n lokalnih minimuma, funkcija potencijalne energije molekula 3^n, a Ekli funkcija čak 45^n. Poređenja rezultata koje je dao Gaus-VNS sa rezultatima Glob-VNS-a su dati u Tabelama 2.5, 2.6, 2.7. Tabele su organizovane slično kao i Tabela 2.4, osim što se sada poređenja vrše isključivo sa rezultatima koje je dao Glob-VNS. Zbog složenosti funkcija i veće dimenzije prostora, za ove primere $k_{\max}$ je povećan na 10 i 15, što je prikazano u tabelama u četvrtoj koloni. Parametri koji se odnose na toleranciju su za ova izračunavanja postavljeni na 10^{-5}.

Tabela 2.5: Eksperimentalni rezultati za Rastrigin funkciju

		lokal		Računska složenost		
funkcija	n	minim.	$k_{\max}$	`Glob-VNS`	`Gaus-VNS`	% poboljšanje
RA10	10	SD	15	**52471**	85589	-38.69%
RA20	20	SD	15	**213597**	287075	-25.60%
RA30	30	SD	15	**366950**	599635	-38.80%
RA40	40	SD	15	**697160**	1115923	-37.53%
RA50	50	SD	15	**1334842**	1504701	-11.29%
RA100	100	SD	15	**5388075**	6248753	-13.77%
RA150	150	SD	15	**11007093**	13678014	-19.53%
RA200	200	SD	15	**24026456**	31639001	-24.06%
					Prosek	-26.16%

Iz Tabele 2.5 se može videti da Gaus-VNS daje lošije rezultate od Glob-VNS-a za Rastrigin funkciju u slučaju svih dimenzija. Međutim, vrednosti iz Tabela 2.6 i 2.7 ukazuju na to da Gaus-VNS daje u većini slučajeva bolje rezultate za druge dve funkcije. U slučaju funkcije potencijalne energije molekula Gaus-VNS je bolji od Glob-VNS-a u 6 od 8 slučajeva, dok za Ekli funkciju dobijeni rezultati su bolji u

Tabela 2.6: Eksperimentalni rezultati za funkciju potencijalne energije molekula

funkcija	n	lokal minim.	$k_{\max}$	Računska složenost		% poboljšanje
				`Glob-VNS`	`Gaus-VNS`	
MPE10	10	SD	10	8102	**5015**	61.56%
MPE20	20	SD	10	26647	**21172**	25.86%
MPE30	30	SD	10	66441	**49162**	35.15%
MPE40	40	SD	10	118006	**109468**	7.80%
MPE50	50	SD	10	202280	**143309**	41.15%
MPE100	100	SD	10	**830343**	1183873	-29.86%
MPE150	150	SD	10	**2353315**	2802372	-16.02%
MPE200	200	SD	10	7683209	**5859705**	31.12%
					Prosek	19.59%

Tabela 2.7: Eksperimentalni rezultati za Ekli funkciju

funkcija	n	lokal minim.	$k_{\max}$	Računska složenost		% poboljšanje
				`Glob-VNS`	`Gaus-VNS`	
AC10	10	SD	10	188670	**50149**	276.22%
AC20	20	SD	10	433194	**158412**	173.46%
AC30	30	SD	10	909918	**304825**	198.51%
AC40	40	SD	10	1577138	**528718**	198.30%
AC50	50	SD	10	4791075	**1143721**	318.90%
AC60	60	SD	10	7820247	**2315178**	237.78%
AC70	70	SD	10	36641634	**4255533**	761.04%
AC80	80	SD	10	212944367	**17180658**	1139.44%
					Prosek	412.96%

svih 8 slučajeva. Analizirajući osobine testiranih funkcija, može se doći do zaključka da Glob-VNS radi bolje na primerima koji imaju lokalne minimume pravougaono raspoređene, poput Rastrigin funkcije, dok Gaus-VNS se bolje ponaša na primerima funkcija radijalnog oblika (koje na istom poluprečniku imaju iste vrednosti). U slučaju Ekli funkcije, za velike dimenzije Gaus-VNS daje drastično bolje rezultate, čak i 1139% bolji rezultat za $n = 80$.

2.2.4 Poređenja sa rezultatima drugih metaheuristika

U ovom odeljku prikazano je poređenje rezultata Gaus-VNS metode sa nekoliko drugih dobro poznatih metaheuristika kao što su tabu pretraga, genetski algoritmi, mravlji algoritmi, raštrkana pretraga (eng. scatter search), inteligencija zasnovana

na rojevima (eng. swarm intelligence) i njihovi hibridi. Izabrane metode, koje su davale najbolje rezultate u literaturi, predstavljene su u Tabeli 2.8. U prve dve kolone se nalaze imena metoda i njihove oznake. U trećoj koloni su date reference na literaturu iz kojih su preuzeti rezultati, dok u poslednjoj koloni su navedene oznake korišćenih kriterijuma zaustavljanja za svaku metodu.

Tabela 2.8: Metaheuristike za rešavanje problema globalne optimizacije

Metaheuristika	Oznaka	Referenca	Zaustavljanje
Genetic and Nelder-Mead	GNM	Chelouah and Siarry [17]	(2.2.2)
Continuous Reactive Tabu Search	CRTS*	Battiti and Tecchiolli [4]	(2.2.1)
Swarm with Nelder-Mead	SNM	Fan et al. [30]	(2.2.2)
Continuous scatter search	CSS	Herrera *et al.* [56]	(2.2.1)
Restarted modified Nelder-Mead	RMNM	Zhao *et al.* [94]	(2.2.1)
Hybrid Continuous Interacting Ants	HCIAC	Dreo and Siarry[29]	(2.2.3)
Directed Tabu Search with adaptive pattern search	DTS	Hedar and Fukushima [55]	(2.2.1)
Gauss Variable neighborhood search	Gauss-VNS	[15], ovaj rad	(2.2.1)

*izabrana najbolja rešenja dobijena varijantama CRTSmin i CRTSave.

Zbog toga što ne postoji ustaljena terminologija za imena metaheuristika na srpskom jeziku, one su u tabeli navedene u originalu.

Izvršiti direktno poređenje ovih metoda je jako teško realizovati iz više razloga. Jedan je već spomenuti kriterijum zaustavljanja: različite metode koriste različite formule za merenje uspešnosti metode, kao i kraj procesa pretrage za rešenjem. Korišćeni kriterijumi zaustavljanja, čije oznake su u poslednjoj koloni Tabele 2.8, su:

$$| \tilde{f} - f_{min} | < 10^{-4} \cdot | f_{min} | + 10^{-6} \tag{2.2.1}$$

$$| \tilde{f} - f_{min} | < 10^{-4} \cdot | f_{init} | + 10^{-6} \tag{2.2.2}$$

$$| \tilde{f} - f_{min} | < 10^{-4} \cdot | f_{min} | + 10^{-4} \tag{2.2.3}$$

U ovim formulama, $\tilde{f}$ označava vrednost funkcije cilja dobijenu primenom odgovarajuće metode, f_{min} je unapred poznata vrednost globalnog minimuma, a f_{init} je prosečna vrednost funkcije cilja dobijena polazeći od 100 slučajno izabranih dopustivih tačaka. Kriterijum (2.2.1) je najstrožiji pa je za njegovo dostizanje potrebno izvršiti veći broj izračunavanja vrednosti funkcije cilja. Gaus-VNS koristi upravo ovaj kriterijum zaustavljanja.

U Tabeli 2.9 dat je pregled rezultata svih 8 metoda. Poređenje je vršeno na osnovu prosečnog broja izračunavanja vrednosti funkcije cilja, polazeći od 100 različitih početnih tačaka dobijenih na slučajan način, do trenutka kada se poznato optimalno rešenje f_{min} nije dostiglo. Neke od metoda nisu uspele da dostignu f_{min} u svih 100 poziva. U tim slučajevima, u zagradi je zapisan ukupan broj uspešnih izvršavanja. Na primer, prosečan broj računanja vrednosti funkcije cilja za RMNM metod pri rešavanju GP instance je 69, ali uzimajući u obzir samo 80 izvršavanja, jer se u preostaliih 20 nije dostiglo optimalno rešenje.

Tabela 2.9: Poređenja rezultata 8 metaheuristika iz literature

Test funkcije	GNM	CRTS	SNM	CSS	RMNM	HCIAC	DTS	Gaus VNS
BR	295	**38**	230	65	*60*	–	212	122
GP	259	171	304	**108**	69 (80)	34533	230	*126*
HT_3	492	513	436	–	**67**	–	438	*233*
HT_6	930	*750*	–	–	398 (50)	–	1787 (83)	**459**
SB	**345**	–	753	762	275 (40)	–	274 (92)	*601*
RO_2	459	–	440	292	*224*	18747	254	**151**
RO_{10}	14563 (83)	–	*3303*	5847 (75)	5946 (95)	–	9037 (85)	**2623**
SH_5	698 (85)	**664**	*850*	1197	912 (90)	17761	819 (75)	942
SH_{10}	635 (85)	*693*	–	–	318 (75)	–	828 (52)	**412**

Eksperimentalni rezultati pokazuju da Gaus-VNS u proseku postiže bolje performanse u odnosu na ostalih sedam metoda. Podebljani rezultati u Tabeli 2.9 označavaju najmanje prosečne računske složenosti, dok iskošene vrednosti označavaju druge po kvalitetu. U obzir se uzimaju samo metode koje su u svih 100 pokretanja došle do rezultata. Gaus-VNS-u je u 4 od 9 slučaja trebalo najmanje prosečnog računanja vrednosti funkcije cilja, dok je drugi po kvalitetu u 3 slučaja.

Glava 3

Problem raspoređivanja prenosa datoteka

U današnje vreme, kada većina korisnika Interneta i/ili umreženih računara ima potrebu za slanjem i preuzimanjem datoteka sa udaljenih servera ili drugih računara, postoji potreba da se obezbedi da se taj sadržaj brzo i lako prenese. Česta je pojava da računar sa kog će se preuzimati sadržaj nije u mogućnosti da opsluži sve klijente u istom trenutku, kao i da je računar koji treba da preuzme više sadržaja ograničen brojem istovremenih prenosa u kojima može da učestvuje. Posledica toga je da i pošaljioci i primaoci moraju da čekaju da se neki od već aktivnih prenosa završe kako bi započeli svoje slanje/ preuzimanje što se izuzetno nepovoljno odražava na performanse.

Za unapred poznate zahteve koje datoteke treba da se razmene između kojih računara, cilj je da se obezbedi efikasnost ovih prenosa u smislu da se ukupno vreme koje će računari čekati da započnu prenos i sam prenos podatka minimizuje. Jedna od mogućnosti je da se unapred odredi raspored po kom će započinjati prenosi datoteka tako da ukupno vreme koje je neophodno da celokupan prenos bude završen bude što je moguće kraće.

Ovaj problem predstavlja jedan od osnovnih problema u oblastima poput telekomunikacija, WAN (eng. Wide Area Networks) i LAN (eng. Local Area Networks) mrežama, multiprocesorskom raspoređivanju na MIMD (više instrukcija, više podataka, eng. Multiple Instruction Multiple Data) sistemima. Svoju primenu može naći i u dodeljuvanju zadataka u okviru kompanija itd.

3.1 Pregled i definicija problema raspoređivanja prenosa datoteka

Problem raspoređivanja prenosa datoteka (eng. File Transfer Scheduling Problem, FTSP) se odnosi na prenošenje kolekcija datoteka različitih veličina između računara u okviru mreže. U inicijalnom trenutku, za svaku od datoteka poznato je između koja dva računara u okviru mreže je neophodno da se izvrši njen prenos. Datoteke mogu biti različitih veličina, te je za svaku od njih unapred poznato i vreme neophodno da se izvrši njen prenos između dva računara. Prenos je dozvoljen isključivo direktno od računara na kom se nalazi datoteka na početku do njenog krajnjeg odredišta, tj. bez prosleđivanja. Na primer, ukoliko je potrebno da se datoteka prenese sa računara A na računar B, nije dozvoljena upotreba računara C kao posrednika. Prekidi u procesu prenosa nisu dozvoljeni, tj. kada započne prenos datoteke on se nastavlja bez prekida dok se celokupna datoteka ne prenese. Pored navedenih ograničenja, ovaj problem otežava i činjenica da svaki od računara ima svoj kapacitet koji se odnosi na broj slobodnih portova za komunikaciju sa drugim računarima. Ovo ograničenje se odnosi na ukupan broj istovremenih prenosa u kojima svaki od računara može da učestvuje u datom trenutku bilo da se radi o slanju ili primanju datoteke.

FTSP je predstavljen prvi put u literaturi u radu [18] gde je dokazano da pripada klasi NP-kompletnih problema. Problem je moguće definisati pomoću neusmerenog multigrafa $G = (V, E)$ koji će se na dalje nazivati *graf prenosa*. Čvorovi grafa $v \in V$ predstavljaju računare u okviru mreže. Parametar $p(v)$ označava kapacitet računara.

Grane grafa $e \in E$ označavaju datoteke koje treba da budu prenešene između dva računara koji su predstavljeni čvorovima na njihovim krajevima. Celobrojne vrednosti $L(e)$ označavaju neophodno vreme prenosa datoteke e izraženo u nekim vremenskim jedinicama. Problem se sastoji u tome da se odredi skup početnih vremena prenosa za svaku od datoteka čime se formira traženi *raspored* prenosa datoteka, takav da dužina vremenskog intervala od trenutka kada započne prenos prve datoteke do trenutka kada se završi prenos poslednje bude što manja.

Za zadati graf prenosa $G = (V, E)$ raspored prenosa datoteka se može predstaviti funkcijom $s : E \to [0, \infty)$ koja svakoj grani $e \in E$ dodeljuje početni vremenski

trenutak prenosa $s(e)$, takav da za svaki čvor $v \in V$ i vremenski trenutak $t \geq 0$ vazi:

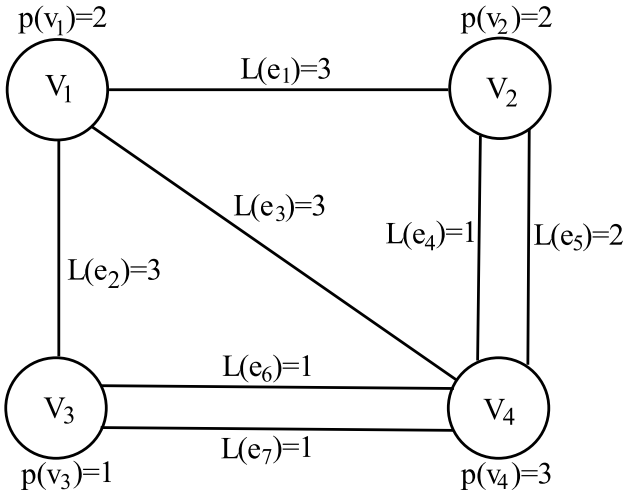
$$|\{e : v \text{ je jedan od krajeva grane } e \text{ i } s(e) \leq t < s(e) + L(e)\}| \leq p(v)\,. \quad (3.1.1)$$

Za zadati raspored s, dužina ukupnog vremena neophodnog za prenos svih datoteka definiše se kao najduže vreme izvršavanja, tj. maksimum $s(e) + L(e)$ za sve grane $e \in E$. Zadatak je da se odredi raspored prenosa sa najkraćim vremenskim intervalom u kom će sve datoteke biti prenešene, tj. da se odredi s takvo da $Obj(s) = \max_{e \in E}\{s(e) + L(e)\}$ bude minimalno.

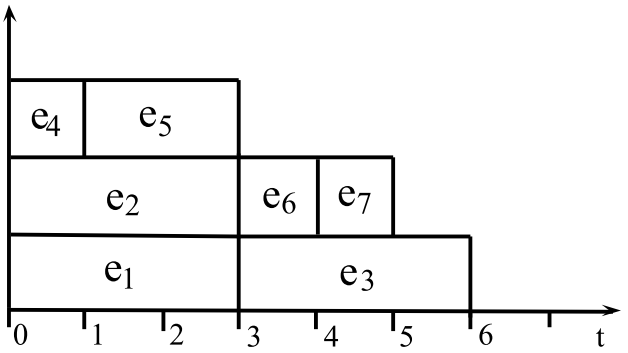
Primer 3.1. *Na slici 3.1 naveden je primer jednog grafa prenosa i odgovarajući vremenski dijagram koji odgovara jednom od dopustivih rasporeda prenosa datoteka za taj graf. Graf sadrži četiri čvora* $\{v_1, v_2, v_3, v_4\}$ *i sedam grana* $\{e_1, .., e_7\}$*, što je ekvivalentno mreži od četiri računara i sedam datoteka koje treba da se prenesu između njih. U trenutku* $t = 0$ *započinje se prenos datoteka koje su predstavljene granama* $e_1, e_2,$ *i* e_4*. Za preostale grane u ovom trenutku ne postoji mogućnost da postanu aktivne, jer bar jedan od njihovih krajnjih čvorova nema više slobodnih portova. Kako je za prenos datoteke, predstavljene granom e4, neophodna jedna vremenska jedinica* $(L(e_4) = 1)$*, u vremenskom trenutku* $t = 1$*, kada se njen prenos završi, oslobađa se po jedan port za čvorove* v_2 *i* v_4 *i neka od preostalih grana čiji je krajni čvor neki od ova dva može da postane aktivna. U ovom slučaju, to je grana* e_5*. U trenutku* $t = 3$*, kada se završe prenosi granama* e_1*,* e_2 *i* e_5*, može da otpočne prenos samo grane* e_3 *i* e_6*, dok grana* e_7 *mora da čeka dokle god se ne oslobode portovi na njenim krajnjim čvorovima.*

U radu [18] autori su predložili nekoliko algoritama za računanje minimalnog vremena prenosa svih datoteka koji se izvršavaju za polinomsko vreme. Međutim, predloženi algoritmi su se odnosili na vrlo usku klasu grafova sa specifičim ograničenjima:

- Bipartitivni grafovi u kojima sve grane imaju istu dužinu;
- Ne sadrže proste cikluse osim u trivijalnom slučaju ciklusa koji se sastoje od 2 čvora i više grana istih dužina između njih;
- Multigrafovi sa parnim ciklusom (ciklus sa parnim brojem čvorova) i jednakim dužinama fajlova;



(a)



(b)

Slika 3.1: Graf transfera (a) i odgovarajući vremenski dijagram rasporeda prenosa datoteka (b)

- Multigrafovi sa neparnim ciklusom (ciklus sa neparnim brojem čvorova), jednakim dužinama fajlova i svi kapaciteti čvorova su jednaki 1.

Metaheuristički pristup za rešavanje FTS problema, zasnovan na neuronskim mrežama, može se naći u [2]. Eksperimentalna testiranja su vršena samo na instancama manjih dimenzija: grafovi sa najviše 39 čvorova i 100 grana. Međutim, izbor instanci je bio takav da su sva optimalna rešenja bila jednaka donjoj granici problema (o kojoj će više reči biti u narednom odeljku).

U literaturi [73] se razmatra i usmereni problem raspoređivanja prenosa datoteka (eng. Directed File Transfer Scheduling Problem). Iako je i taj problem NP-težak, u praksi se pokazao daleko lakšim od osnovne varijante problema pošto postoji algoritam polinomske složenosti za njegovo rešavanje u slučaju kada sve grane imaju istu dužinu.

Ako bi se u obzir uzimali samo potpuno povezani grafovi, sa po 2 grane između svaka dva čvora (jedna za slanje, druga za primanje) i uz dozvoljeno prosleđivanje, dobija se problem koji je razmatran u [88, 93, 74]. U ovom slučaju zadatak je naći odgovarajuću rutu za datoteke i odgovarajući raspored njihovih prenosa, kako bi za najkraće vreme sve bile dostavljene na svoja odredišta.

Problem sa dozvoljenim prosleđivanjem fajlova razmatran je i u [57]. U [3] je predložena varijanta problema gde se pored prosleđivanja koristi i pretpostavka kako se svaka datoteka može izdeliti na više manjih datoteka istih dužina, a zatim se umesto polaznih vršiti prenos tih delova polazne datoteke koji su iste dužine.

Još jedna varijanta FTS problema se odnosi na slučaj kada zahtevi za prenosima datoteka nisu svi unapred poznati. U tom slučaju zahtevi dolaze jedan za drugim i odluka kada koji transfer treba da se započne se donosi bez znanja koji dodatni zahtevi mogu da se jave naknadno. Ovaj problem su razmatrali autori u [53] kao i halapljive (eng. greedy) algoritame za njegovo rešavanje.

3.2 Donja granica problema

Autori rada [18] su opisali postupak dobijanja elementarne donje granice rešenja problema. Neka E_u označava skup svih datoteka koje treba da se pošalju ili prime preko čvora u, tj. skup svih grana takvih da im je jedan od krajnjih čvorova čvor u. Stepen čvora u označava se sa $d_u = |E_u|$. Neka $E_{u,v}$ predstavlja $E_u \bigcap E_v$,

tj. skup datoteka čiji prenos treba da se obavi između čvorova u i v. Neka je $\sum_u = \sum_{e \in E_u} L(e)$ i $\sum_{u,v} = \sum_{e \in E_{u,v}} L(e)$. Kao i do sada, $p(u)$ se koristi da označi kapacitet čvora u. Vreme neophodno da se izvrše svi prenosi koji uključuju čvor u je najmanje $\left\lceil \frac{\sum_u}{p(u)} \right\rceil$, gde $\lceil x \rceil$ označava najmanji ceo broj veći ili jednak od x.

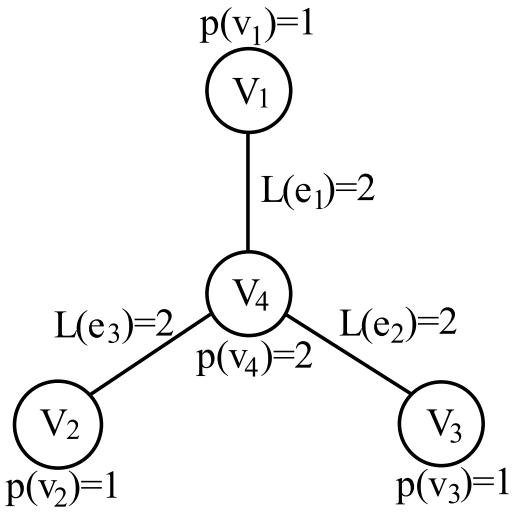
Lema 3.2.1. *([18]) Optimalno vreme celokupnog prenosa datoteka $OPT(G)$ za bilo koji graf G mora da zadovoljava $OPT(G) \geq \max_u \left\lceil \frac{\sum_u}{p(u)} \right\rceil$.*

Definicija 3.2.1. *Veličina $\max_u \left\lceil \frac{\sum_u}{p(u)} \right\rceil$ naziva se elementarnom donjom granicom FTS problema.*

Kao što se može videti iz sledeća dva primera, $OPT(G)$ je često veće od elementarne donje granice.

Primer 3.2. *Na slici 3.2 je prikazan graf sa četiri čvora i tri datoteke koje treba da budu prenešene između njih. Za svaki čvor na jednostavan način se može izračunati:*
$\sum_{v_1} = L(e_1) = 2,$
$\sum_{v_2} = L(e_3) = 2,$
$\sum_{v_3} = L(e_2) = 2,$
$\sum_{v_4} = L(e_1) + L(e_2) + L(e_3) = 2 + 2 + 2 = 6.$
Elementarna donja granica se dobija kao:
$\max\{\left\lceil \frac{\sum_{v_1}}{p(v_1)} \right\rceil, \left\lceil \frac{\sum_{v_2}}{p(v_2)} \right\rceil, \left\lceil \frac{\sum_{v_3}}{p(v_3)} \right\rceil, \left\lceil \frac{\sum_{v_4}}{p(v_4)} \right\rceil\} =$
$\max\{\left\lceil \frac{2}{1} \right\rceil, \left\lceil \frac{2}{1} \right\rceil, \left\lceil \frac{2}{1} \right\rceil, \left\lceil \frac{6}{2} \right\rceil\} = \max\{2, 2, 2, 3\} = 3.$
Jednostavna analiza problema pokazuje da ukupno vreme od 3 vremenske jedinice kako bi se obavili svi prenosi nije moguće. Kako bi se izvršio prenos sve tri datoteke preko čvora v_4 ovaj čvor, čiji je kapacitet jednak 2, mora da učestvuje u prenosu dve datoteke istovremeno. Prenos treće datoteke ne može da započne dok se ne oslobodi jedan od portova čvora v_4. Pošto su za prenos sve tri datoteke neophodne po dve vremenske jedinice, treća datoteka mora da sačeka i da njen prenos otpočne tek kada prve dve aktivne završe. Kako su i za nju potrebne dve vremenske jedinice, ukupno vreme potrebno da se završi prenos sve tri datoteke je 4. Odavde, očigledno je da je $OPT(G) = 4 > 3$.

Primer 3.3. *Posmatrajmo još jednom primer sa slike 3.1. Na sličan način može se izračunati:*
$\sum_{v_1} = L(e_1) + L(e_2) + L(e_3) = 3 + 3 + 3 = 9,$



Slika 3.2: Primer kada je $OPT(G) \geq \max_u \lceil \frac{\sum_u}{p(u)} \rceil$

$\sum_{v_2} = L(e_1) + L(e_4) + L(e_5) = 3 + 1 + 2 = 6,$
$\sum_{v_3} = L(e_2) + L(e_6) + L(e_7) = 3 + 1 + 1 = 5,$
$\sum_{v_4} = L(e_3) + L(e_4) + L(e_5) + L(e_6) + L(e_7) = 3 + 1 + 2 + 1 + 1 = 8.$
Odavde, vrednost elementarne donje granice je:
$\max\{\left\lceil \frac{\sum_{v_1}}{p(v_1)} \right\rceil, \left\lceil \frac{\sum_{v_2}}{p(v_2)} \right\rceil, \left\lceil \frac{\sum_{v_3}}{p(v_3)} \right\rceil, \left\lceil \frac{\sum_{v_4}}{p(v_4)} \right\rceil\} =$
$\max\{\lceil \frac{9}{2} \rceil, \lceil \frac{6}{2} \rceil, \lceil \frac{5}{1} \rceil, \lceil \frac{8}{3} \rceil\} = \max\{5, 3, 5, 3\} = 5.$
Posmatrajmo čvor v_1 koji ima tri datoteke u čijim prenosima treba da učestvuje. Njegov kapacitet je 2 što znači da istovremeno može da učestvuje u prenosu najviše dve od ove tri datoteke. Za to vreme, treća datoteka mora da čeka. Pošto za njihov prenos treba po tri vremenske jedinice, treća datoteka može započeti sa prenosom tek kada prve dve završe. Za njen prenos je neophodno jos tri vremenske jedinice, pa je za celokupan prenos sve tri datoteke preko čvora v_1 neophodno ukupno 6 vremenskih jedinica, tj. nije moguće dostići elementarnu donju granicu koja je jednaka 5.

3.3 Modeli celobrojnog linearnog programiranja za FTSP

U ovom odeljku biće date tri formulacije FTSP kao problema celobrojnog linearnog programiranja (eng. Integer Linear Programming, ILP). Treba napomenuti da su ovo prve takve formulacije poznate u literaturi. Deo rezultata prezentovanih u ovom odeljku je prikazan i u radu Z. Dražić, A. Savić, V. Filipović [28].

3.3.1 Reformulacije problema

U cilju formulacije i dokazivanja validnosti modela ILP, najpre će problem, opisan u odeljku 3.1, u nastavku teksta u oznaci $P1$, biti reformulisan.

Vreme neophodno da se izvrši prenos svake datoteke izraženo je u broju vremenskih jedinica, tj. celobrojnim vrednostima. Iz tog razloga, jednostavnije je koristiti celobrojne vremenske intervale umesto kontinualnog vremena.

Definicija 3.3.1. *Celobrojnu vrednost $\tau \in Z^+$ ćemo zvati indeksom vremenskog intervala od $\tau - 1$ do τ.*

Lema 3.3.1. *Neka je s' neko dopustivo rešenje FTS problema. Postoji funkcija $s'' : E \to Z^+$ takva da $Obj_{P1}(s'') \leq Obj_{P1}(s')$.*

Dokaz Neka je $m(s') = \min\{s'(e)|e \in E\}$, tj. najmanji vremenski trenutak kada započinje prvi prenos po rasporedu s'. Ako je $m(s') > 0$, onda je $s^*(e) = s'(e) - m(s')$ takođe dopustivo rešenje FTS problema sa manjom vrednošću funkcije cilja. Odavde, s^* je rešenje u kojem bar jedan prenos datoteke započinje u vremenskom trenutku $t = 0$.

Definišimo $s''(e) = \lfloor s'(e) \rfloor$. Očigledno je da je $Obj_{P1}(s'') \leq Obj_{P1}(s')$. Dokažimo sada da je s'' dopustivo rešenje.

Neka su $e_1, e_2 \in E$ i $s'(e_1) + L(e_1) \leq s'(e_2)$, tj. ne postoje preklapanja u prenosu datoteka predstavljenih sa e_1 i e_2. U ovom slučaju $s'(e_2) - s'(e_1) \geq L(e_1)$. Iz $s'(e) = \lfloor s'(e) \rfloor + R,\ 0 \leq R < 1$ i $s''(e) = \lfloor s'(e) \rfloor$, dobija se da je $s'(e) = s''(e) + R,\ 0 \leq R < 1$. Za $e_1, e_2 \in E$ važi $s'(e_1) = s''(e_1) + R_1,\ 0 \leq R_1 < 1$ i $s'(e_2) = s''(e_2) + R_2,\ 0 \leq R_2 < 1$. Oduzimanjem, dobija se $s''(e_1) - s''(e_2) = s'(e_1) - s'(e_2) + (R_2 - R_1)$ gde $-1 < R_2 - R_1 < 1$, tj. $s''(e_1) - s''(e_2) = \lfloor s'(e_1) - s'(e_2) \rfloor + R_3 + (R_2 - R_1)$ gde $0 \leq R_3 < 1,\ -1 < R_2 - R_1 < 1$. Odavde, $R_3 + (R_2 - R_1) \in (-1, 2)$, a kako je leva strana prethodne jednakosti celobrojna vrednost onda i $R_3 + (R_2 - R_1)$ mora biti ceo broj iz intervala $(-1, 2)$, tj. $R_3 + (R_2 - R_1) = 0$ ili $R_3 + (R_2 - R_1) = 1$. Na osnovu ovoga, razlikujemo dva slučaja:
Slučaj 1: $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor$.
Slučaj 2: $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor + 1$.

U slučaju 1, kako je $\lfloor s'(e_2) - s'(e_1) \rfloor \geq L(e_1)$, to povlači $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor \geq L(e_1)$. U slučaju 2, $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor + 1 \geq$

$\lfloor L(e) \rfloor + 1 = L(e) + 1$.

Ovim je dokazano da će svaki par prenosa koji nema preklapanja u s' ostati takav i u s'', što znači da je s'' takodje dopustivo rešenje. □

Sada je moguće definisati *aktivan* prenos datoteke.

Definicija 3.3.2. *Grana $e \in E$ u rasporedu prenosa $s : E \to Z^+$ je aktivna u vremenskom intervalu sa indeksom τ ako je $s(e) < \tau$ i $s(e) + L(e) > \tau - 1$.*

Definišimo problem $P2$.

Problem P2: Za raspored $s : E \to Z^+$, za svako $\tau \in Z^+$ i $v \in V$ neka važi:
$|\{e : v$ je jedan od krajeva grane e i e je aktivna grana u vremenskom intervalu sa indeksom $\tau\}| \leq p(v)$.

Dužina ukupnog vremena neophodnog za prenos svih datoteka određenog rasporedom prenosa s, tj. $Obj_{P2}(s)$, definisano je kao najveći indeks vremenskog intervala τ u smislu Definicije 3.3.1. Cilj je naći raspored s sa najmanjim ukupnim vremenom prenosa, u oznaci $Opt_{P2}(G)$.

Teorema 3.3.2. *Za zadat graf G, problemi $P1$ i $P2$ imaju iste optimalne vrednosti funkcije cilja.*

Dokaz ($\Rightarrow$) Neka je s^* optimalno rešenje problema $P1$. Iz Leme 3.3.1, postoji dopustivo rešenje problema $P2$ sa celobrojnim vrednostima trenutaka kada datoteke započinju svoj prenos takvo da njegova vrednost funkcije cilja nije veće od početne vrednosti funkcije cilja, tj. $Opt_{P2}(G) \leq Opt_{P1}(G)$.

($\Leftarrow$) Sa druge strane, svako dopustivo rešenje problema $P2$ je ujedno i dopustivo rešenje problema $P1$ pošto je $Z^+ \subset R$, pa optimalno rešenje problema $P1$ ne može biti veće od optimalnog rešenja problema $P2$, tj. $Opt_{P2}(G) \geq Opt_{P1}(G)$. □

Definišimo sada i problem $P3$.

Problem P3: Za raspored $s : E \to \{1, 2, ..., T_{\max}\}$, za svako $\tau \in \{1, 2, ..., T_{\max}\}$ i $v \in V$ neka važi:
$|\{e : v$ je krajnji čvor grane e i e je aktivna u vremenskom intervalu sa indeksom $\tau\}| \leq p(v)$.

Dužina ukupnog vremena neophodnog za prenos svih datoteka određenog rasporedom prenosa s, tj. $Obj_{P3}(s)$, definisano je kao najveći indeks vremenskog intervala τ u smislu Definicije 3.3.1.

Cilj je naći raspored s sa najmanjim ukupnim vremenom prenosa, u oznaci $Opt_{P3}(G)$.

Teorema 3.3.3. *Za zadati graf G i $T_{\max} \geq \sum_{e_i \in E} L(e_i)$, problemi $P2$ i $P3$ imaju iste optimalne vrednosti funkcije cilja.*

Dokaz ($\Rightarrow$) Možemo formirati raspored u kom prvi prenos e_1 započinje na samom početku, drugi prenos u $L(e_1)$, k-ti prenos e_k počinje u $\sum_{i=1}^{k-1} L(e_i)$. Poslednji prenos će završiti u $\sum_{e_i \in E} L(e_i)$. Kako je $T_{\max} \geq \sum_{e_i \in E} L(e_i)$ odatle sledi $s(e)+L(e) \leq T_{\max}$ za sve e. S obzirom da je vrednost $s(e)$ ceo broj za problem $P2$ onda $s(e) \in \{1, ..., T_{\max}\}$. Odatle, $Opt_{P2}(G) \geq Opt_{P3}(G)$.

($\Leftarrow$) Kako je $\{1, ..., T_{\max}\} \subset Z^+$ odatle sledi $Opt_{P2}(G) \leq Opt_{P3}(G)$. □

3.3.2 Modeli ILP za FTS problem

Nakon završenih teoretskih razmatranja u prethodnom odekljku 3.3.1, svi uslovi za formiranje novog modela ILP, u daljoj oznaci ILP1, za FTS problem su ispunjeni.

Posmatrajući problem $P3$, definišimo indikatorske promenljive

$$X_{e\tau} = \begin{cases} 1, & \text{prenos granom } e \text{ je aktivan u intervalu sa indeksom } \tau \\ 0, & \text{u suprotnom.} \end{cases} \tag{3.3.1}$$

Takodje, neka je $T_{\max}$ gornja granica optimalnog rešenja problema $P3$. Jedna takva gornja granica je definisana Teoremom 3.3.3, međutim dosta preciznija gornja granica se može dobiti korišćenjem neke metaheuristike, na primer metode promenljivih okolina. Ovo omogućava formulisanje modela ILP za FTS problem na sledeći način:

Odrediti

$$\min z \tag{3.3.2}$$

pri uslovima:

$$\sum_{\tau=1}^{T_{\max}} X_{e\tau} = L(e), \qquad e \in E \tag{3.3.3}$$

$$\sum_{e\in E, e\ni v} X_{e\tau} \leq p(v), \qquad v \in V,\ 1 \leq \tau \leq T_{\max} \tag{3.3.4}$$

$$z \geq \tau \cdot X_{e\tau}, \qquad e \in E,\ 1 \leq \tau \leq T_{\max} \tag{3.3.5}$$

$$X_{ej} \geq X_{ei} + X_{ek} - 1, \qquad e \in E,\ 1 \leq i < j < k \leq T_{\max} \tag{3.3.6}$$

$$X_{e\tau} \in \{0, 1\}, \qquad e \in E,\ 1 \leq \tau \leq T_{\max} \tag{3.3.7}$$

$$z \in Z^{+}. \tag{3.3.8}$$

Uslovi (3.3.3) obezbeđuju da je cela datoteka predstavljena granom e prenešena na svoje odredište i da je njen prenos završen do trenutka $T_{\max}$. Sa (3.3.4) ograničen je maksimalni broj istovremenih prenosa za svaki čvor njegovim kapacitetom. Ograničenja (3.3.5) označavaju da nema aktivnih prenosa nakon vremenskog trenutka z. Uslovi (3.3.6) obezbeđuju da, ukoliko je prenos datoteke aktivan u vremenskim trenucima i i k, onda on mora biti aktivan i u svim vremenskim trenucima j takvim da $i < j < k$, tj. prenosi se izvršavaju kontinualno bez prekida. U slučaju da prenos prekine svoje izvršavanje, ono ne može da se nastavi za istu datoteku u nekom narednom vremenskom trenutku.

Broj promenljivih u ovom modelu je $|e| \cdot T_{\max} + 1$. Uslova (3.3.3) ima $|e|$, uslova (3.3.4) ima $|v| \cdot T_{\max}$, dok uslova (3.3.5) ima $|e| \cdot T_{\max}$. Uslova (3.3.6) ima najviše i njihov broj je $|e| \cdot \binom{T_{\max}}{3}$. Uslova (3.3.7) i (3.3.8) ima kao i promenljivih i oni se u literaturi ne broje jer se u komercijalnim rešavačima zadaju kao granice promenljivih a ne kao uslovi. Ukupan broj uslova je $|e| + |v| \cdot T_{\max} + |e| \cdot T_{\max} + |e| \cdot \binom{T_{\max}}{3} =$

$O(|e| \cdot \binom{T_{\max}}{3})$.

Dokažimo da je rešenje dobijeno iz ILP1 formulacije ujedno i rešenje problema $P3$.

Teorema 3.3.4. *Neka je s^* optimalno rešenje problema $P3$. Za optimalno rešenje $(z^*, X^*_{e\tau})$ ILP1 formulacije koje zadovoljava uslove (3.3.3) - (3.3.8) vazi $z^* = Obj_{P3}(s^*)$.*

Dokaz ($\Rightarrow$)

Neka je s^* optimalno rešenje problema $P3$ i $Obj_{P3}(s^*) = A$. Očigledno je da je $A \leq T_{\max}$ s obzirom da je $T_{\max}$ gornja granica jednog već poznatog dopustivog rešenja. Za dato optimalno rešenje s^*, moguće je na jednostavan način odrediti vremenske intervale u kojima je svaka grana e aktivna. Nakon utvrđivnja vremenskih intervala, poznat nam je celokupan prenos datoteka pa je moguće odrediti sve vrednosti $X^*_{e\tau}$. Postavimo $z^* = A$. Dokažimo da je $(z^*, X^*_{e\tau})$ dopustivo rešenje ILP1 formulacije.

Kako je $\sum_{\tau=1}^{T_{\max}} X_{e\tau}$ jednako broju aktivnih intervala prenosa granom e, uslov (3.3.3) je zadovoljen.

Za fiksirano $v \in V$ i $\tau \in \{1, 2, ..., T_{\max}\}$, suma u (3.3.4) je jednaka broju aktivnih prenosa u vremenskom intervalu sa indeksom τ, koji započinju ili završavaju u čvoru v. Kako je ovaj broj ograničen kapacitetom čvora $p(v)$, onda važi nejednakost (3.3.4).

Da bi se dokazala nejednakost (3.3.5) razlikujemo dva slučaja:

i. Ako je $X_{e\tau} = 0$, uslov (3.3.5) je zadovoljen jer je $z > 0$ po definiciji.

ii. Ukoliko je $X_{e\tau} = 1$, onda je u vremenskom intervalu sa indeksom τ bar jedan prenos aktivan (prenos granom e) i zbog toga važi $Obj(s^*) = A = z^* \geq \tau$, što povlači da je uslov (3.3.5) zadovoljen i u ovom slučaju.

Da bismo dokazali nejednakost iz (3.3.6) zapišimo je najpre na drugačiji način: $X_{ei} + X_{ek} \leq 1 + X_{ej}$ za svako $i < j < k$. Jasno je da je ova nejednakost zadovoljena u slučaju da je $X_{ei} = 0 \vee X_{ek} = 0$. U suprotnom, kada je $X_{ei} = X_{ek} = 1$, nejednakost će da važi samo za $X_{ej} = 1$. S ozbirom da je transfer e kontinualan, tj. bez prekida, ne postoje vremenski intervali $i < j < k$ takvi da je on aktivan u intervalima i i k a neaktivan u intervalu j. Ovo dokazuje zadovoljenost uslova (3.3.6).

Iz definicije promenljivih $X_{e\tau}$ i z očigledno je da su i uslovi (3.3.7) i (3.3.8), koji opisuju prirodu promenljivih, zadovoljeni.

Ovim je pokazano da je $(z^*, X^*_{e\tau})$ dopustivo rešenje ILP1 formulacije problema, pa vrednost optimalnog rešenja neće biti veća od $z^* = A = Obj(s^*)$.

($\Leftarrow$) Dokažimo i drugi smer. Pretpostavimo da je $(z^*, X^*_{e\tau})$ optimalno rešenje ILP1 formulacije koje zadovoljava uslove (3.3.3) - (3.3.8).

Formirajmo najpre rešenje s^* problema $P3$. Za svaku granu $e \in E$, uzmimo da je $s^*(e) = -1 + \min\{\tau | X_{e\tau} = 1\}$. Za fiksirano e, uslov (3.3.6) obezbeđuje da je prenos granom e aktivan samo u uzastopnim vremenskim intervalima, što znači da će taj transfer biti neprekidan. Dužina trajanja prenosa granom e je jednaka $L(e)$ iz uslova (3.3.3). Kako je $\tau \leq T_{\max}$, svaki prenos se završava najkasnije do $T_{\max}$. Iz (3.3.4), u svakom vremenskom intervalu sa indeksom τ broj prenosa koji započinju i završavaju u čvoru v je ograničen sa $p(v)$. Uzimajući sve ovo u obzir, zaključujemo da je s^* dopustivo rešenje problema $P3$.

Neka je τ^* indeks poslednjeg vremenskog intervala za koji je bar jedan $X_{e\tau^*} = 1$. Kako je $\tau^* \leq T_{\max}$ to znači da takav indeks postoji. Iz (3.3.5) se dobija $z^* \geq \tau^* \cdot X_{e\tau^*} = \tau^* \cdot 1 = \tau^*$. U rešenju s^*, u vremenskom intervalu sa indeksom τ^* bar jedan prenos je aktivan i u svim narednim vremenskim intervalima, nakon ovog sa indeksom τ^*, ne postoji ni jedan više aktivan prenos, pa je $Obj(s^*) = \tau^* = z^*$. Kako je ovo dopustivo rešenje, vrednost optimalnog rešenja problema $P3$ nije veća od z^*. □

Uslov (3.3.6) iz ILP1 formulacije koji se odnosi na neprekidnost prenosa pojedinačnih datoteka se može zameniti uslovom:

$$X_{ei} + X_{ek} \leq 1, \qquad e \in E,\ 1 \leq i,\ i + L(e) \leq j \leq T_{max} \tag{3.3.9}$$

koji zajedno sa uslovima (3.3.3) - (3.3.5) i (3.3.7) - (3.3.8) definiše drugu ILP formulaciju problema, u daljoj oznaci ILP2. Broj promenljivih je ponovo $|e| \cdot T_{\max} + 1$, dok broj uslova (3.3.9) je $|e| \cdot \binom{T_{\max} - L(e) + 1}{2}$, stoga ukupan broj uslova za ILP2 formulaciju je $O(|e| \cdot \binom{T_{\max} - L(e) + 1}{3})$.

Dokažimo da je rešenje dobijeno iz ILP2 formulacije takođe rešenje problema P3.

Teorema 3.3.5. *Neka je s^* optimalno rešenje problema $P3$. Za optimalno rešenje $(z^*, X^*_{e\tau})$ ILP2 formulacije koje zadovoljava uslove (3.3.3) - (3.3.5), (3.3.9) i (3.3.7) - (3.3.8) važi $z^* = Obj_{P3}(s^*)$.*

Dokaz Kako su uslovi (3.3.3) - (3.3.5) i (3.3.7) - (3.3.8) isti kao i u ILP1 formulaciji problema, dokaz u oba smera koji se odnosi na ovaj deo formulacije je isti kao i u Teoremi 3.3.4. Dokažimo preostali uslov (3.3.9).

($\Rightarrow$)

Da bismo pokazali da važi uslov (3.3.9) analizirajmo tri slučaja:

i. Ako prenos granom e u intervalu sa indeksom i nije aktivan, tj. $X_{ei} = 0$, onda nejednakost iz (3.3.9) sledi direktno jer je X_{ej} po definiciji ili 0 ili 1.

ii. Ako prenos granom e u intervalu sa indeksom j nije aktivan, tj. $X_{ej} = 0$, nejednakost iz (3.3.9) sledi jer je X_{ej} takođe po definiciji ili 0 ili 1.

iii. Ako je $X_{ei} = 1$ i $X_{ej} = 1$ tj. prenos je počeo ili u intervalu sa indeksom i ili pre toga, on se morao završiti ili u intervalu sa indeksom $i + L(e)$ ili pre njega, te je zbog $i + L(e) \leq j \leq T_{max}$ u intervalu sa indeksom j sigurno završen. Kako prenos ne može nakon prekida započeti ponovo, odatle X_{ej} mora biti 0, što je kontradikcija sa polaznom pretpostavkom da je $X_{ej} = 1$.

($\Leftarrow$)

Pretpostavimo da važi (3.3.9) i pokažimo da se na osnovu ovoga obezbeđuje neprekidnost prenosa datoteka. Definišimo nove promenljive: τ_1 kao indeks intervala kada je počeo prenos datoteke e i τ_2 kao indeks poslednjeg intervala kada je bio taj prenos aktivan:

$$\tau_1 = \min\{\tau | X_{e\tau} = 1\},$$

$$\tau_2 = \max\{\tau | X_{e\tau} = 1\}.$$

Sabiranjem $X_{e\tau_1}$ i $X_{e\tau_2}$, dobija se $X_{e\tau_1} + X_{e\tau_2} = 2$, što iz (3.3.9) direktno povlači da za indekse τ_1 i τ_2 ovako definisanih indeksa vremenskih itervala mora da važi $\tau_1 + L(e) > \tau_2$.

Pretpostavimo da je transfer prekinut negde između intervala sa indeksima τ_1 i τ_2, tj. da $\exists k \in [\tau_1, \tau_2]$ takvo da $X_{ek} = 0$. Tada, ako bi razvili sumu $\sum_{\tau=1}^{T_{\max}} X_{e\tau}$, dobijamo sledeći izraz:

$$\sum_{\tau=1}^{T_{\max}} X_{e\tau} = \sum_{\tau=1}^{\tau_1-1} X_{e\tau} + \sum_{\tau=\tau_1}^{k-1} X_{e\tau} + X_{ek} + \sum_{\tau=k+1}^{\tau_2} X_{e\tau} + \sum_{\tau=\tau_2+1}^{T_{\max}} X_{e\tau}.$$

Prvi, treći i peti sabirak su jednaki nuli, tj. $\sum_{\tau=1}^{\tau_1-1} X_{e\tau} = X_{ek} = \sum_{\tau=\tau_2+1}^{T_{\max}} X_{e\tau} = 0$ zbog toga što je prenos počeo tek u intervalu sa indeksom τ_1, završio se u intervalu sa inteksom τ_2 i po pretpostavci nije aktivan u intervalu sa indeksom k. Za drugi i četvrti sabirak važiće: $\sum_{\tau=\tau_1}^{k-1} X_{e\tau} \leq k-1-\tau_1+1$ i $\sum_{\tau=k+1}^{\tau_2} X_{e\tau} \leq \tau_2-(k+1)+1$. Sabiranjem ove dve sume dobija se da je :

$$\sum_{\tau=\tau_1}^{k-1} X_{e\tau} + \sum_{\tau=k+1}^{\tau_2} X_{e\tau} \leq k-1-\tau_1+1+\tau_2-(k+1)+1 = \tau_2-\tau_1 = L(e)$$

što je kontradikcija, jer važi da je $\tau_2 - \tau_1 < L(e)$. □

Definišimo dodatne indikatorske promenljive

$$Y_{e\tau} = \begin{cases} 1, & \text{prenos granom } e \text{ počinje u intervalu sa indeksom } \tau \\ 0, & \text{u suprotnom.} \end{cases} \tag{3.3.10}$$

Tada se neprekidnost prenosa može predstaviti narednim uslovima koji bi u formulacijama ILP1 i ILP2 zamenili uslov (3.3.6) i (3.3.9), respektivno.

$$\sum_{\tau=1}^{T_{\max}} Y_{e\tau} = 1, \qquad e \in E \tag{3.3.11}$$

$$X_{e\tau} - X_{e(\tau-1)} \leq Y_{e\tau}, \qquad e \in E,\ 2 \leq \tau \leq T_{\max} \tag{3.3.12}$$

$$X_{e1} = Y_{e1}, \qquad e \in E. \tag{3.3.13}$$

Broj promenljivih u ILP3 formulaciji je $2 \cdot |e| \cdot T_{\max} + 1$, dok je broj uslova od (3.3.11) - (3.3.13) jednak $|e| \cdot (T_{\max} + 1)$. Ukupan broj uslova u ILP3 formulaciji je $O(|e| \cdot T_{\max})$.

Dokažimo da je rešenje dobijeno i iz ove formulacije (3.3.3) - (3.3.5),(3.3.7) - (3.3.8) i (3.3.11) - (3.3.13), u daljem tekstu ILP3, rešenje problema P3.

Teorema 3.3.6. *Neka je s^* optimalno rešenje problema P3. Za optimalno rešenje $(z^*, X^*_{e\tau})$ ILP3 formulacije koje zadovoljava uslove (3.3.3) - (3.3.5),(3.3.7) - (3.3.8) i (3.3.11) - (3.3.13) važi $z^* = Obj_{P3}(s^*)$.*

Dokaz Slično kao i u dokazu Teoreme 3.3.5 dovoljno je pokazati samo ekvivalenciju između neprekidnosti prenosa datoteka i uslova (3.3.11) - (3.3.13).

($\Rightarrow$)

Po definiciji problema P3 svaka datoteka započinje svoj prenos samo jednom, tj. samo jedno $Y_{e\tau}$ će biti jednako 1, pa odatle direktno sledi uslov (3.3.11).

Da bismo dokazali uslov (3.3.12) posmatrajmo tri slučaja:

i. Kada je stanje prenosa isto u vremenskim intervalima sa indeksom τ i $\tau - 1$. Tada je $X_{e\tau} = X_{e(\tau-1)} = 1$ ukoliko je prenos u oba intervala aktivan, odnosno $X_{e\tau} = X_{e(\tau-1)} = 0$ ako je neaktivan. Nejednakost iz uslova (3.3.12) će svakako važiti jer je leva strana jednaka 0, što je svakako ne veće od binarnog $Y_{e\tau}$.

ii. Ako je prenos aktivan u vremenskom intervalu sa indeksom $\tau - 1$, ali nije aktivan u intervalu sa indeksom τ. Tada je $X_{e(\tau-1)} = 1$, $X_{e\tau} = 0$ i $X_{e(\tau)} - X_{e(\tau-1)} = -1$ pa nejednakost iz (3.3.12) svakako važi.

iii. Ako je prenos aktivan u vremenskom intervalu sa indeksom τ ($X_{e\tau} = 1$), ali nije aktivan u intervalu sa indeksom $\tau - 1$ ($X_{e(\tau-1)} = 0$). To znači da je prenos počeo u intervalu sa indeksom τ pa je $Y_{e\tau} = 1$. Na osnovu ovoga, zbog binarne prirode promenljivih nejednakost iz (3.3.12) i u ovom slučaju važi.

Po definiciji problema, ni jedan prenos ne moze početi pre intervala sa indeksom $\tau = 1$. Ako je prenos počeo baš tada, onda je $Y_{e1} = 1$ i prenos je svakako aktivan kada započne prenos pa je i $X_{e1} = 1$. Ako transfer nije počeo u intervalu sa indeksom $\tau = 1$, tada nije ni aktivan, pa je $Y_{e1} = X_{e1} = 0$ čime je zadovoljen i uslov (3.3.13).

($\Leftarrow$)

Iz uslova (3.3.11) sledi da će za svaku granu e samo jedan $Y_{e\tau}$ biti jednak jedinici, što znači da za svaku granu postoji jedinstveni interval sa indeksom τ kada ona započinje svoj prenos.

Razdvojimo dva slučaja:

i. Neka je prenos počeo u vremenskom intervalu sa indeksom $\tau = 1$. Tada iz uslova (3.3.13) sledi da prenos počinje i traje u tom intervalu, tj. $X_{e1} = Y_{e1} = 1$. Kako za neko $t \geq 2$ mora biti $Y_{et} = 0$ onda iz (3.3.12) važi da je $X_{et} \leq X_{e(t-1)}$, što znači da kad jednom X_{et} postane 0, za sve naredne vremenske intervale ova promenljiva će takođe imati vrednost nula, tj. nakon prekida jednog prenosa nema njegovog novog započinjanja.

ii. Neka je prenos počeo u vremenskom intervalu sa indeksom $\tau \geq 2$. To znači da je $Y_{e1} = 0$ pa iz uslova (3.3.13) mora biti $X_{e1} = 0$. Dokle god je $Y_{et} = 0$, zbog nejednakosti (3.3.12) biće i $X_{et} = 0$, za $t < \tau$, tj. $X_{e1} = X_{e2} = ... = X_{e(\tau-1)} = 0$. Ako je $X_{e\tau} = 0$, onda će iz istog razloga i svaki naredni X_{et}, $t > \tau$ biti jednak nuli, tj. $X_{e(\tau+1)} = X_{e(\tau+2)} = ... = X_{eT_{\max})} = 0$, pa je u tom slučaju $\sum_{t=1}^{T_{\max}} X_{et} = 0$ što je kontradikcija zbog uslova (3.3.3). Sa druge strane, ako je $X_{e\tau} = 1$ neprekidnost datog prenosa se translacijom početka prenosa sa intervala sa indeksom τ na interval sa indeksom 1 dokazuje na isti način kao i u prvom slučaju.

□

Iz izloženog se može videti da ILP2 formulacija ima za red veličine manji broj uslova od ILP1 formulacije uz isti broj promenljivih. ILP3 formulacija ima za jedan red veličine manji broj uslova od ILP2 formulacije iako je praktično dupliran broj promenljivih. Broj uslova koji određuje neprekidnost prenosa je u ILP3 formulaciji istog reda veličine kao i broj uslova koji obezbeđuju da nema aktivnih prenosa nakon završnog vremenskog trenutka. Iz literature se može videti da dopustiv skup relaksiranog problema ponekad često može biti manji ukoliko je broj uslova u ILP formulaciji veći. Zbog toga formulacije ILP1 i ILP2 mogu biti korisne pri iznalaženju odsecajućih ravni za dati problem, što može biti jedan od pravaca daljeg istraživanja.

3.4 Metoda promenljivih okolina za rešavanje FTSP

U ovom odeljku je opisana metoda promenljivih okolina za rešavanje FTSP sa tri različite implementacije postupka lokalne pretrage. Deo rezultata prezentovanih u ovom odeljku prikazani su u samostalnom radu Z. Dražić [27].

3.4.1 Opis algoritma

Kako bi se implementirao VNS za FTS problem, najpre treba definisati funkciju cilja $f(x)$ i skup dopustivih rešenja X. Neka je $X = \{0, ..., n-1\}^n$ skup n-dimenzionalnih vektora sa celobrojnim vrednostima, $0 \leq x_k \leq n-1$, gde je n ukupan broj grana za FTS problem. Za proizvoljan vektor $x \in X$ vrednost funkcije cilja $f(x)$ se može

odrediti na sledeći način. Elementi x_k vektora x predstavljaju prioritete dodeljene svakoj grani e_k grafa. Sa zadatim prioritetima grana, može se formirati jedinstveni raspored prenosa datoteka po sledećem algoritmu:

1) Sve grane se na jedinstven način sortiraju u skladu sa njihovim prioritetom u opadajućem redosledu. U slučaju da dve grane imaju istu vrednost prioriteta, prednost ima grana sa većom dužinom. U slučaju da su i prioriteti i dužine grana isti, prednost ima grana sa manjim indeksom.

2) Sva pojedinačna vremena prenosa datoteka $L(e_k)$ su celobrojne vrednosti, tako da u optimalnom rasporedu prenosa sve grane započinju svoj prenos u celobrojnim vremenskim trenucima. Počevši od trenutka $t = 0$ algoritam pokušava da u sortiranom nizu grana pronađe prvu granu koja može započeti svoj prenos tako da se ne naruši ograničenje koje se odnosi na kapacitet čvorova. Kada se nađe takva grana, njoj se dodeli početno vreme transfera t. Postupak se ponavlja za sve ostale neraspoređene grane. Ukoliko ni jedna grana ne može da započne prenos u trenutku t, uvećavamo t za 1 i ponavljamo ceo postupak sve dok svim granama e_k ne budu dodeljena početni vremenski trenuci t_k.

Za raspored prenosa, formiran na ovaj način, vrednost funkcije cilja $f(x)$ je difinisana kao najveća vrednost t za koju grane završavaju sa svojim prenosom, tj. $f(x) = max_{1 \leq k \leq n}(t_k + L(e_k))$.

Ulazni parametri za VNS su: najmanji i najveći indeks okolina koje treba pretraživati ($k_{\min}$ i $k_{\max}$), maksimalni broj iteracija i parametar p koji predstavlja verovatnoću prelaska iz jednog u drugo rešenje sa istom vrednošću funkcije cilja.

Inicijalizacija se vrši na slučajan način. Na početku VNS procedure za svaki graf početno rešenje x se formira kao slučajna permutacija celobrojnih vrednosti $0, 1, ..., n - 1$.

Ključni element VNS-a predstavlja struktura okolina jer od njihovog izbora zavise performanse same metode. Najveća poteškoća je pronaći balans između efektivnosti i mogućnosti da se izađe iz zamki lokalnog minimuma. Za dato k, okolina trenutnog rešenja $N_k(x)$ će sadržati sve vektore prioriteta koji se mogu razlikovati od vektora prioriteta rešenja x u najviše k vrednosti.

U procesu razmrdavanja, VNS generiše novi vektor $x' \in N_k(x)$ prateći sledeći algoritam:

1) U prvom koraku, formira se vektor od k različitih celih brojeva iz skupa $\{0, 1, ..., n-1\}$. Ovako formiran vektor, $(i_1, ..., i_k)$, predstavlja indekse elemenata iz vektora rešenja koji će biti modifikovani.

2) Koristeći vrednosti $i_1, ..., i_k$, dobijene u prethodnom koraku, formira se vektor $(p_1, ..., p_k)$ kao slučajna permutacija vrednosti elemenata vektora x sa indeksima $i_1, ..., i_k$. Elementi vektora formiranog u ovom koraku predstavljaju nove vrednosti prioriteta koje će se koristiti za formiranje vektora x'.

3) Vektor x' se formira najpre kao kopija trenutnog vektora x, a zatim mu se samo k elemenata modifikuje na sledeći način: $x'(i_j) = p_j, j = 1, 2, ..., k$, tj. vrednosti prioriteta grana na pozicijama i_j menjamo sa novim prioritetima p_j.

Rešenje x' koje se dobija razmrdavanjem često nije čak ni lokalni minimum, tako da je u cilju nalaženja boljeg rešenja potrebno primeniti neki metod lokalnog pretraživanja u okolini rešenja x'. Lokalna pretraga istražuje neku okolinu rešenja x' u cilju da pronađe novo, bolje rešenje sa manjom vrednošću funkcije cilja. Glavna komponenta lokalnog pretraživanja je definisanje adekvatne okoline rešenja x'. Okoline datog rešenja sadržaće sva rešenja koja se mogu dobiti od polaznog menjajući ga na neki način. Mala okolina rešenja x', koja će se pretraživati radi boljeg rešenja, sadržaće sve vektore koji se mogu dobiti od vektora x' zamenom dva njegova proizvoljna elementa. Koristiće se strategija prvog poboljšanja (eng. first improvement strategy). Ova strategija iterativno pretražuje okolinu rešenja x' i čim nađe prvo poboljšanje (rešenje sa manjom vrednošću funkcije cilja), lokalna pretraga se zaustavlja i sa novodobijenim poboljšanjem ponovo pokušava da se poboljša lokalnim pretraživanjem, ponavljajući algoritam u okolini tog novog rešenja. Algoritam se ponavlja sve dok ima poboljšanja. U slučaju da uopšte nema poboljšanja, rešenje x' se ne menja, odnosno $x'' = x'$. Lokalna pretraga se završava kada se rešenje više ne može popraviti i najbolje rešenje dobijeno lokalnom pretragom označavamo sa x''.

Nakon lokalne pretrage treba doneti odluku da li se prebaciti u novo rešenje x'' i da li nastavak glavnog algoritma VNS metode nastaviti od njega ili ne. Bazični VNS algoritam prelazi iz rešenja x u rešenje x'' samo u slučaju da je vrednost funkcije cilja rešenja x'' strogo manja od vrednosti funkcije cilja rešenja x. U slučajevima kada funkcija ima dosta lokalnih minimuma sa istom vrednošću funkcije cilja, prelazak iz jednog u drugo rešenje može da proširi pretragu i poveća šanse za pronalazak još

boljeg rešenja. Sa druge strane, ukoliko se ovaj prelazak iz jednog u drugo rešenje vrši svaki put kada se nađe rešenje sa istom vrednošću funkcije cilja, postoji velika verovatnoća da se "vrtimo u krug". Da bismo izbegli ove dve ekstreme situacije, u slučaju kada je $f(x) = f(x'')$, algoritam će koristiti parametar p koji se odnosi na verovatnoću prelaska iz rešenja x u rešenje x''.

Uzimajući prethodno razmatranje u obzir, nakon lokalne pretrage, postoje tri mogućnosti:

- U slučaju kada je rešenje x'' bolje od x, tj. $f(x'') < f(x)$, postaviti $x := x''$ i nastaviti pretragu u istoj okolini N_k;
- Ukoliko je $f(x'') > f(x)$, tada ponoviti pretragu sa istim rešenjem x i narednom okolinom N_{k+1};
- Ukoliko je $f(x'') = f(x)$, tada sa verovatnoćom p postaviti $x := x''$ i nastaviti pretragu u istoj okolini N_k, a sa verovatnoćom $1-p$ nastaviti pretragu sa istim rešenjem x u narednoj okolini N_{k+1}.

Kriterijum zaustavljanja za predložen VNS algoritam će biti dostizanje maksimalnog broja iteracija. Jedna iteracija se odnosi na proces nalaženja jednog novog rešenja (razmrdavanje) i proces lokalne pretrage u okolini tog rešenja.

Opisani algoritam se može predstaviti psudo-šemom na slici 3.3.

3.4.2 Metoda spusta kroz promenljive okoline za FTSP

U ovom odeljku je prikazana modifikacija metode promenljivih okolina u kojoj će se postupak lokalne pretrage realizovati kao spust kroz promenljive okoline.

Nakon dobijanja rešenja x' postupkom razmrdavanja, postupak lokalne pretrage opisan u odeljku 3.4 vrši pretragu zamenom svaka dva elementa vektora x' u potrazi za boljim rešenjem. Nailaskom na prvo bolje rešenje, premeštamo se u njega i nastavljamo dalju pretragu u njegovoj okolini. Jedna ideja da se opisani postupak lokalne pretrage ubrza je da se koristi kombinacija okolina, tj. takva modifikacija loklane pretrage da okolinu u kojoj tražimo poboljšanje za x' izdelimo na skup više manjih okolina za čiju je pretragu složenost dosta manja i na taj način realizujemo VND.

Definišimo nove okoline za lokalnu pretragu tako što će se okolina lokalne pretrage definisana zamenom svaka dva elementa vektora x' izdeliti na više manjih okolina:

/* *Inicijalizacija* */
Izabrati skup okolina N_k, $k = k_{\min}, \ldots, k_{\max}$ koje će se koristiti za pretragu;
Za dato k, okolina $N_k(x)$ će sadržati sve vektore iste dužine kao x koji se mogu razlikovati od njega u najviše k vrednosti;
Na slučajan način izabrati početno rešenje $x \in X$ kao permutaciju celobrojnih vrednosti $0, 1, ..., n-1$;
Postaviti $x^* \leftarrow x$, $f^* \leftarrow f(x)$;
Odrediti kriterijum zaustavljanja;
repeat naredne korake **until** ispunjen je kriterijum zaustavljanja
    Postaviti $k \leftarrow k_{\min}$;
1: **repeat** naredne korake **until** $k > k_{\max}$
        /* *Razmrdavanje* */
        Formirati vektor $(i_1, ..., i_k)$ od k različitih celih brojeva iz skupa $\{0, 1, ..., n-1\}$;
        Formirati vektor $(p_1, ..., p_k)$ kao slučajnu permutaciju vrednosti elemenata vektora x sa indeksima $(i_1, ..., i_k)$;
        $x' := x$;
        $x'(i_j) := p_j, j = 1, 2, ..., k$;
        /* *Lokalna pretraga* */
        Zamenom svaka dva proizvoljna elementa vektora x', naći lokalni minimum x'' problema;
        Koristiti strategiju prvog poboljšanja;
        /* *Da li se pomeriti?* */
        if $f(x'') < f^*$ **then**
            Postaviti $x^* \leftarrow x''$, $f^* \leftarrow f(x'')$ i **goto** 1;
        elseif $f(x'') > f^*$ **then**
            Postaviti $k \leftarrow k + 1$;
        else
            Sa verovatnoćom p postaviti $x^* \leftarrow x''$, $f^* \leftarrow f(x'')$ i **goto** 1, a sa verovatnoćom $p - 1$ postaviti $k \leftarrow k + 1$;
    end
end
Tačka x^* je aproksimativno rešenje problema.

Slika 3.3: Pseudo-šema VNS algoritma za FTSP

- Okolinu $\mathcal{N}_1(x')$ čine sva rešenja koja se mogu dobiti od polaznog x' zamenom svaka dva njegova uzastopna elementa, tj. sa razmakom 0.

- Okolinu $\mathcal{N}_2(x')$ čine sva rešenja koja se mogu dobiti od polaznog x' zamenom svaka dva njegova elementa koja imaju razmak od jednog elementa između sebe, tj. zamenom svaka dva $x'(i)$ i $x'(i+2)$, $i = 0, ..., n-3$.

- Okolinu $\mathcal{N}_l(x'), l = 1, ..., n-1$ čine sva rešenja koja se mogu dobiti od polaznog x' zamenom dva njegova elementa koja imaju razmak od $l-1$ elementa između sebe.

Istraživanje svake pojedinačne okoline $\mathcal{N}_l(x'),\ l = 1, ..., n-1$ je brže od istraživanja okoline iz prethodnog odeljka. Unija svih okolina $\mathcal{N}_1, ..., \mathcal{N}_{n-1}$ čini, zapravo, okolinu iz postupka loklane pretrage definisane u prethodnom odeljku što nam obezbeđuje da u pretrazi nećemo propustiti neki lokalni optimum. Treba napomenuti da je sistem okolina u VND potpuno nezavisan u odnosu na sistem okolina iz postupka razmrdavanja.

U okviru spusta po promenljivim okolinama vrši se pretraživanje kroz navedenih $n-1$ okolina upravo u navedenom redosledu: $\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_{n-1}$. To znači da će se lokalna pretraga sastojati od sledećih nekoliko koraka:

- Postupak započinje pretragom okoline $\mathcal{N}_1(x')$ za boljim rešenjem.

- Ukoliko bolje rešenje u nekoj okolini $\mathcal{N}_k(x'), k \leq n-2$ nije nađeno, pretraga se nastavlja u okolini $\mathcal{N}_{k+1}(x')$.

- Ukoliko je pretragom pronađeno bolje rešenje, ono postaje tekuće rešenje i ceo postupak pretrage okolina započinje ponovo od okoline N_1 ove nove tačke.

- Ukoliko bolje rešenje nije pronađeno ni u poslednjoj okolini $\mathcal{N}_{n-1}(x')$, postupak pretrage se završava.

Ideja ove modifikacije je da se prvo pretraže sva rešenja koja su "najbliža" rešenju x' (tj. koja se dobijaju samo zamenom njegova dva uzastopna elementa) i da se na taj način brzo dobiju značajnije popravke polaznog rešenja. Ova strategija se bazira na uverenju da u reprezentaciji rešenja datog VNS-a za FTSP, bliski elementi vektora prioriteta imaju veću međuzavsnost nego oni koji su na većoj udaljenosti. Na ovaj način najviše puta će se realizovati pretraga kroz okolinu $\mathcal{N}_1$, jer svaka popravka

izvedena u bilo kojoj okolini $\mathcal{N}_l$, $l = 2, ..., n-1$ proizvodi još (bar) jednu pretragu kroz okolinu $\mathcal{N}_1$. Sa druge strane, pretraga kroz okolinu $\mathcal{N}_l$ se izvodi samo onda kada smo stigli do lokalnog minimuma u odnosu na sve okoline $\mathcal{N}_1, ..., \mathcal{N}_{l-1}$, što ne mora biti tako često.

Opisani algoritam se može predstaviti psudo-šemom na slici 3.4.

Primer 3.4. *Posmatrajmo jednostavan primer toka lokalne pretrage. Neka se vektor x', dobijen nakon razmrdavanja, sastoji od 5 elemenata: $x' = (1, 2, 3, 4, 5)$. Neka su vektori $p1 = (1, 2, 4, 3, 5)$ i $p2 = (1, 4, 2, 3, 5)$ jedina rešenja koja imaju manju vrednost funkcije cilja od vektora x' takva da važi $f(x') > f(p1) > f(p2)$. Postupak lokalne pretrage definisane u odeljku 3.4 bi se sastojao od sledećeg niza zamena:* $\mathbf{x'} = (\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5})$, $(2, 1, 3, 4, 5)$, $(3, 2, 1, 4, 5)$, $(4, 2, 3, 1, 5)$, $(5, 2, 3, 4, 1)$, $(1, 3, 2, 4, 5)$, $(1, 4, 3, 2, 5)$, $(1, 5, 3, 4, 2)$, $\mathbf{p1} = (\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{3}, \mathbf{5})$. *Nakon prvog poboljšanja, pretraga se nastavlja od početka u njegovoj okolini:* $(2, 1, 4, 3, 5)$, $(4, 2, 1, 3, 5)$, $(3, 2, 4, 1, 5)$, $(5, 2, 4, 3, 1)$, $\mathbf{p2} = (\mathbf{1}, \mathbf{4}, \mathbf{2}, \mathbf{3}, \mathbf{5})$. *Ukupan broj zamena je 13.*

Sa druge strane, postupak lokalne pretrage, definisane u ovom odeljku, sastoji se od sledećih zamena: $\mathbf{x'} = (\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5})$, $(2, 1, 3, 4, 5)$, $(1, 3, 2, 4, 5)$, $\mathbf{p1} = (\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{3}, \mathbf{5})$, $(2, 1, 4, 3, 5)$, $\mathbf{p2} = (\mathbf{1}, \mathbf{4}, \mathbf{2}, \mathbf{3}, \mathbf{5})$. *Ukupan broj zamena u ovom slučaju je 5.*

Kao što se može videti, VND u ovom slučaju dosta brže stigne do željenog rešenja. U oba slučaja, nakon dobijanja vektora $(1, 4, 2, 3, 5)$, *izvršiće se isti broj zamena do kraja procedure lokalne pretrage u (za ovaj primer neuspešnoj) potrazi za boljim rešenjem.*

3.4.3 Ubrzanje lokalnog pretraživanja

Procedure lokalne pretrage opisane u odeljcima 3.4 i 3.4.2 su dosta temeljne jer ispituju sve elemente iz okolina datog rešenja x' koje se sastoje iz zamene svaka dva elementa, samo drugačijim redosledom. Međutim, njihova glavna mana je što pretraga u oba slučaja traje relativno dugo. Jedan mogući način da se značajno ubrza lokalna pretraga je da se smanji okolina koja će se pretraživati. Time bi se manje vremena gubilo na dugotrajne zamene elemenata vektora x' kao i računanje vrednosti funkcije cilja za svaki novodobijeni vektor.

Neka je jedina okolina lokalne pretrage, koja će se koristiti u ovoj modifikaciji metode, $\mathcal{N}_1$ okolina iz odeljka 3.4.2. Dakle, u ovoj varijanti okolinu $\mathcal{N}_1(x')$ čine sva rešenja koja se mogu dobiti od polaznog x' samo zamenom svaka dva njegova

/* *Inicijalizacija* */
Izabrati skup okolina N_k, $k = k_{\min}, \ldots, k_{\max}$ koje će se koristiti za pretragu;
Za dato k, okolina $N_k(x)$ će sadržati sve vektore iste dužine kao x koji se mogu razlikovati od njega u najviše k vrednosti;
Izabrati skup okolina $\mathcal{N}_l$, $l = 1, \ldots, n-1$ koje će se koristiti za lokalnu pretragu;
Za dato l, okolina $\mathcal{N}_l(x')$ će sadržati sve vektore iste dužine kao x' koji se mogu dobiti od njega zamenom dva njegova elementa koji imaju razmak od $l-1$ elemenata između sebe;
Na slučajan način izabrati početno rešenje $x \in X$ kao permutaciju celobrojnih vrednosti $0, 1, ..., n-1$;
Postaviti $x^* \leftarrow x$, $f^* \leftarrow f(x)$;
Odrediti kriterijum zaustavljanja;
repeat naredne korake **until** ispunjen je kriterijum zaustavljanja
    Postaviti $k \leftarrow 1$;
1: **repeat** naredne korake **until** $k > k_{\max}$
        /* *Razmrdavanje* */
        Generisati slučajnu tačku $x' \in N_k(x^*)$;
        /* *Lokalna pretraga koristeći VND* */
        Postavi $l \leftarrow 1$;
        repeat naredne korake **until** $l > n-1$
            /* *Istraživanje okoline* */
            Pronaći najbolje rešenje x'' u okolini $\mathcal{N}_l(x')$;
            /* *Da li se pomeriti?* */
            if $f(x'') < f(x')$ **then**
                Postaviti $x' \leftarrow x''$ i $l \leftarrow 1$;
            else
                Postaviti $l \leftarrow l + 1$;
        end
        /* *Da li se pomeriti?* */
        if $f(x'') < f^*$ **then**
            Postaviti $x^* \leftarrow x''$, $f^* \leftarrow f(x'')$ i **goto** 1;
        if $f(x'') > f^*$ **then**
            Postaviti $k \leftarrow k + 1$;
        else
            Sa verovatnoćom p postaviti $x^* \leftarrow x''$, $f^* \leftarrow f(x'')$ i **goto** 1,
            a sa verovatnoćom $p-1$ postaviti $k \leftarrow k + 1$;
    end
end
Tačka x^* je aproksimativno rešenje problema.

Slika 3.4: Pseudo-šema VNS-VND algoritma za FTSP

uzastopna elementa. Lokalna pretraga sada bi se sastojala samo od provere da li u okolini $\mathcal{N}_1(x')$ postoji bolje rešenje. U slučaju pronalaska boljeg rešenja, premeštamo se u njega i pretragu nastavljamo u istoj okolini novog rešenja, a u slučaju da popravke nema proces lokalne pretrage je završen.

Opisani algoritam se može predstaviti psudo-šemom na slici 3.5.

Na ovaj način sužavamo pretragu čime sa jedne strane značajno ubrzavamo sam algoritam. Sa druge strane, oslanjajući se samo na činjenicu da su lokalni minimumi blizu jedan drugom, pa je moguće da se u prethodnim modifikacijama dosta vremena gubilo na bezuspešnu pretragu za boljim minimumom nakon što je on već na početku pronađen, postoji mogućnost da izgubimo neko dobro rešenje.

Uprkos ovom navodnom nedostatku, praktični eksperimenti pokazuju da se na ovaj način može u pretrazi ipak otići u potpuno drugačijem smeru. Drugim rečima, može se desiti da sa ovakvom pretragom dođemo do rešenja do kog u slučaju algoritma iz odeljka 3.4, uprkos temeljnijoj pretrazi, to nije bilo izvodljivo. Sledeći jednostavan primer upravo pokazuje ovo zapažanje.

Primer 3.5. *Neka se vektor x' dobijen nakon razmrdavanja sastoji od 5 elemenata: $x' = (1, 2, 3, 4, 5)$. Neka su naredne permutacije označene sa $p1 = (1, 2, 4, 3, 5)$, $p2 = (4, 2, 1, 3, 5)$ i $p3 = (1, 4, 2, 3, 5)$ jedina rešenja koja imaju manju vrednost funkcije cilja od rešenja x' i neka važi $f(x') > f(p1) > f(p2) > f(p3)$, gde je $f(\cdot)$ vrednost funkcije cilja.*

Postupak lokalne pretrage, definisane u odeljku 3.4, bi imao sledeći tok: $\mathbf{x' = (1, 2, 3, 4, 5)}$, $(2, 1, 3, 4, 5)$, $(3, 2, 1, 4, 5)$, $(4, 2, 3, 1, 5)$, $(5, 2, 3, 4, 1)$, $(1, 3, 2, 4, 5)$, $(1, 4, 3, 2, 5)$, $(1, 5, 3, 4, 2)$, $\mathbf{p1 = (1, 2, 4, 3, 5)}$. *Nakon ponovnog pokretanja lokalne pretrage od $p1$ dobijamo:* $(2, 1, 4, 3, 5)$, $\mathbf{p2 = (4, 2, 1, 3, 5)}$.

Sa druge strane, primenom lokalne pretrage opisane u ovom odeljku dobijamo: $\mathbf{x' = (1, 2, 3, 4, 5)}$, $(2, 1, 3, 4, 5)$, $(1, 3, 2, 4, 5)$, $\mathbf{p1 = (1, 2, 4, 3, 5)}$, $(2, 1, 4, 3, 5)$, $\mathbf{p3 = (1, 4, 2, 3, 5)}$.

Rešenja $p2 = (4, 2, 1, 3, 5)$ i $p3 = (1, 4, 2, 3, 5)$ se razlikuju na ukupno 3 pozicije i jasno je da će zamenom bilo koja dva elementa rešenja $p2$ dobijenog algoritmom lokalne pretrage iz odeljka 3.4 ne može dobiti $p3$ koje u ovom primeru ima manju vrednost funkcije cilja.

Poredeći algoritam lokalne pretrage opisan u ovom odeljku i VND iz odeljka 3.4.2 jasno je da VND može doći do nekog boljeg rešenja lokalnim pretragama u

```
/* Inicijalizacija */
Izabrati skup okolina N_k, k = k_min, ..., k_max koje će se koristiti za pretragu;
Za dato k, okolina N_k(x) će sadržati sve vektore iste dužine kao x koji se mogu
  razlikovati od njega u najviše k vrednosti;
Izabrati okolinu N_1, koja će se koristiti za lokalnu pretragu;
Okolina N_1(x') će sadržati sve vektore iste dužine kao x' koji se mogu
  dobiti od njega zamenom dva njegova uzastopna elementa;
Na slučajan način izabrati početno rešenje x ∈ X kao permutaciju
  celobrojnih vrednosti 0, 1, ..., n − 1;
Postaviti x* ← x, f* ← f(x);
Odrediti kriterijum zaustavljanja;
repeat naredne korake until ispunjen je kriterijum zaustavljanja
      Postaviti k ← 1;
1:    repeat naredne korake until k > k_max
            /* Razmrdavanje */
            Generisati slučajnu tačku x' ∈ N_k(x*) ;
            /* Lokalna pretraga */
            Postavi l ← 1;
            repeat naredne korake until l > 1
                  /* Istraživanje okoline */
                  Pronaći najbolje rešenje x'' u okolini N_l(x');
                  /* Da li se pomeriti? */
                  if f(x'') < f(x') then
                        Postaviti x' ← x'' i l ← 1;
                  else
                        Postaviti l ← l + 1;
            end
            /* Da li se pomeriti? */
            if f(x'') < f* then
                  Postaviti x* ← x'', f* ← f(x'') i goto na 1;
            elseif f(x'') > f* then
                  Postaviti k ← k + 1;
            else
                  Sa verovatnoćom p postaviti x* ← x'', f* ← f(x'') i goto 1,
                    a sa verovatnoćom p − 1 postaviti k ← k + 1;
      end
end
Tačka x* je aproksimativno rešenje problema.
```

Slika 3.5: Pseudo-šema VNS-LS1 algoritma za FTSP

okolinama $\mathcal{N}_2$,...,$\mathcal{N}_{n-1}$ koje su u ovom slučaju nedostupne. U tom slučaju postupak razmrdavanja u narednoj iteraciji VNS algoritma očigledno će birati slučajne tačke iz okolina različitih rešenja, te i u ovom slučaju pretraga opet odlazi u drugačijem smeru.

3.5 Eksperimentalni rezultati

U ovom odeljku su predstavljeni eksperimentalni rezultati koji pokazuju efikasnost opisanih ILP formulacija i VNS algoritama. Sva izračunavanja koja se odnose na ILP formulacije su vršena na Intel Core i3-2350M CPU 2.30GHz sa 4GB RAM memorije pod Windows 7 operativnim sistemom, dok izračunavanja koja se odnose na VNS implementacije na Intel Core 2 Duo 2.67 GHz PC računaru sa 4 GB RAM memorije pod Windows XP operativnim sistemom.

3.5.1 Slučajno generisani primeri grafova problema

Uvidom u odgovarajuću literaturu nisu pronađeni javno dostupni primeri grafova za FTSP. Iz tog razloga, za eksperimentalna testiranja su korišćene nove instance generisane na slučajan način. Procedura za njihovo generisanje je detaljno opisana u nastavku, a nalaze se dostupne preko interneta na adresi http://poincare.matf.bg.ac.rs/∼zdrazic/ftsp.

Generisane instance uključuju različit broj čvorova grafa ($|V|$ = 5, 10, 30, 50, 100) i različit broj grana ($|E|$=10, 50, 100, 200, 500). Na početku procesa generisanja instanci, za svaki čvor se na slučajan način bira ceo broj iz intervala $[1, v_{\max}]$ koji će označavati njegov kapacitet. Za vrednost parametra $v_{\max}$ su korišćene vrednosti $v_{\max}$ = 8, 15, 50, 80, 150 u zavisnosti od broja čvorova u grafu. Na sličan način su generisane i dužine svake od grana, birajući na slučajan način ceo broj iz intervala $[1, e_{\max}]$, gde je $e_{\max}$ = 8, 15, 50, 80, 150. Nakon ovoga, formirana je matrica povezanosti grafa na slučajan način, ne dozvoljavajući da grane imaju početak i kraj u istom čvoru. Uzimaju se u obzir samo povezani grafovi. Izbegavanjem grana koje za svoj početak i kraj imaju isti čvor kao i nepovezanih grafova postignuto je da ovako generisane instance budu bliže realnim problemima. Kako bi se postigla raznolikost, za svaki par $|V|$ i $|E|$, generisano je po deset različitih instanci korišćenjem različite početne vrednosti generatora slučajnih brojeva.

3.5.2 Određivanje optimalnog rešenja

Za testiranje ILP formulacija i dobijanje optimalnog rešenja FTS problema, korišćena su dva rešavača: CPLEX 12.4 [19] i Gurobi 5.0.2. [81]. Za oba rešavača ukupno vreme izvršavanja je bilo ograničeno na 7200 sekundi.

Pored testiranja ILP formulacija, za instance malih dimenzija korišćen je i program zasnovan na totalnoj enumeraciji. Ova metoda najpre generiše sve permutacije skupa $\{1, ..., |E|\}$ a zatim od svih njih traži onu sa najmanjom vrednošću funkcije cilja koja se računa na isti način kao što je opisano u odeljku 3.4. Na ovaj način sve moguće permutacije su proverene, te je dobijeno rešenje optimalno. Kako broj permutacija drastično raste sa povećanjem broja grana, to je već za $|E| = 10$ ukupan broj permunacija $10! = 3628800$ pa je ova metoda primenjena samo za instance kod kojih je broj grana manji ili jednak 10.

U Tabeli 3.1 su prikazani eksperimentalni rezultati dobijeni korišćenjem oba rešavača za sva tri modela ILP, kao i rezultati dobijeni totalnom enumeracijom. Tabela je organizovana na sledeći način:

- U prvoj koloni se nalaze imena instanci. Ime instance sadrži podatak o njenom broju čvorova i broju grana. Na primer, instanca ftsp_50_200_0 sadrži 50 čvorova i 200 grana. Broj 0 na kraju označava redni broj instance tog tipa.

- Druga kolona, označena sa LB, sadrži vrednost elementarne donje granice za svaku od instanci dobijene kao što je opisano u odeljku 3.2.

- U trećoj koloni, T_{max}, se nalazi vrednost gornje granice za svaku od instanci koja je neophodna za sva tri modela. Za ove vrednosti su uzeti najbolji rezultati dobijeni nekom od modifikacija metode promenljivih okolina, opisanih u ovoj i narednoj glavi.

- Četvrta kolona, TE, sadrži vrednosti optimalnih rešenja do kojih se došlo primenom algoritma totalne enumeracije.

- Peta kolona, opt, sadrži vrednost optimalnog rešenja za datu instancu ili oznaku '-' ukoliko optimalnu vrednost nije bilo moguće dobiti ni na jedan od navedenih načina.

- Šesta i sedma kolona, CPLEX i Gurobi, se odnose na rezultate dobijene testiranjem prve ILP1 formulacije FTS problema (3.3.3) - (3.3.8) korišćenjem

CPLEX i Gurobi rešavača. U slučaju da je odgovarajući ILP rešavač uspeo da pronađe optimalno rešenje za manje od 7200 sekundi, u ovim kolonama upisano je vreme izvršavanja u sekundama. U slučaju da za 7200 sekundi nije pronađeno optimalno rešenje, prikazana je oznaka '-'.

- Naredne dve kolone su organizovane isto kao šesta i sedma kolona samo za drugu ILP2 formulaciju FTS problema.
- Poslednje dve kolone se odnose na treću ILP3 formulaciju FTS problema.

Dobijeni rezultati pokazuju da se za manje instance na ovaj način mogu dobiti optimalna rešenja za veoma kratko vreme. Iz Tabele 3.1 se vidi da su za instance malih dimenzija i CPLEX i Gurobi rešavači pronašli optimalna rešenja za sve instance za sve tri ILP formulacije. Poredeći vremena izvršavanja CPLEX i Gurobi rešavača za formulaciju ILP1, ukupno vreme za dobijanje optimalnih rešenja za svih 20 instanci za CPLEX je 6.175 sekundi, dok je za Gurobi 69.795 sekindi. Za formulaciju ILP2, CPLEX rešavaču je trebalo 1.400 sekunde, a za Gurobi 7.281 sekunde, dok je za formulaciju ILP3, CPLEX pronašao sva optimalna rešenja za ukupno 2.136 sekunde, a Gurobi za 5.949 sekunde. Iz ovoga se može videti da CPLEX za instance malih dimenzija dolazi do optimalnih rešenja brže nego Gurobi.

U Tabelama 3.2 i 3.3 su prikazani rezultati za instance srednjih i velikih dimenzija, respektivno. Obe tabele su organizovane na sličan način kao i Tabela 3.1 sa izuzetkom kolone TE jer totalna enumeracija za ove instance nije mogla da završi sa radom i dođe do rešenja.

Eksperimenti pokazuju da se vreme izvršavanja i memorijski zahtevi CPLEX i Gurobi rešavača vrlo brzo povećavaju sa povećavanjem dimenzije problema. Iz Tabele 3.2 se može uočiti da su za instance srednjih dimenzija koristeći formulaciju ILP1, i CPLEX i Gurobi rešavač dostigli optimalna rešenja za samo, ukupno u zbiru, tri instance od 20, i to CPLEX za instance $ftsp\_10\_50\_1$ i $ftsp\_10\_50\_4$, dok Gurobi za $ftsp\_10\_50\_4$ i $ftsp\_10\_50\_5$. U slučaju formulacije ILP2, CPLEX dostiže optimalna rešenja za 18 od 20 instanci, dok Gurobi za 15 od 20. U slučaju svih 15 instanci gde su oba rešavača došla do optimalnih rešenja, CPLEX je bio znatno brži od Gurobi-a, dok ukupno vreme izvršavanja CPLEX-a za tih 15 instanci je 1708.126 sekundi, a za Gurobi 8205.660 sekundi. Formulacija ILP3 se pokazala najuspešnijom, gde su oba rešavača pronašla optimalna rešenja za svih 20 instanci.

Tabela 3.1: Optimalna rešenja za instance malih dimenzija

Inst.	LB	T_{max}	TE	opt	ILP1		ILP2		ILP3	
					CPLEX	Gurobi	CPLEX	Gurobi	CPLEX	Gurobi
ftsp_5_10_0	14	14	14	14	1.461	5.441	0.397	0.830	0.349	1.061
ftsp_5_10_1	10	15	15	15	0.247	1.257	0.050	0.237	0.056	0.218
ftsp_5_10_2	12	12	12	12	0.243	0.936	0.049	0.163	0.122	0.214
ftsp_5_10_3	14	14	14	14	0.238	3.713	0.092	0.808	0.286	0.580
ftsp_5_10_4	10	10	10	10	0.102	0.645	0.054	0.073	0.044	0.153
ftsp_5_10_5	6	13	13	13	0.533	1.864	0.040	0.043	0.047	0.301
ftsp_5_10_6	12	13	13	13	0.921	2.116	0.060	0.155	0.057	0.448
ftsp_5_10_7	8	8	8	8	0.022	0.014	0.037	0.007	0.013	0.011
ftsp_5_10_8	11	11	11	11	0.083	1.416	0.024	0.062	0.051	0.139
ftsp_5_10_9	26	26	26	26	0.984	40.966	0.125	4.548	0.558	2.146
ftsp_10_10_0	12	15	15	15	0.054	0.044	0.027	0.010	0.032	0.011
ftsp_10_10_1	12	12	12	12	0.065	0.030	0.077	0.015	0.070	0.017
ftsp_10_10_2	5	14	14	14	0.082	0.076	0.037	0.015	0.056	0.017
ftsp_10_10_3	19	19	19	19	0.645	10.764	0.084	0.167	0.143	0.519
ftsp_10_10_4	17	17	17	17	0.194	0.264	0.058	0.037	0.047	0.036
ftsp_10_10_5	10	15	15	15	0.066	0.045	0.036	0.018	0.040	0.018
ftsp_10_10_6	3	15	15	15	0.063	0.090	0.030	0.015	0.026	0.017
ftsp_10_10_7	11	14	14	14	0.046	0.045	0.045	0.038	0.048	0.023
ftsp_10_10_8	11	14	14	14	0.049	0.036	0.053	0.014	0.062	0.010
ftsp_10_10_9	2	14	14	14	0.077	0.033	0.025	0.026	0.029	0.010

Tabela 3.2: Optimalna rešenja za instance srednjih dimenzija

Inst.	LB	T_{max}	opt	ILP1		ILP2		ILP3	
				CPLEX	Gurobi	CPLEX	Gurobi	CPLEX	Gurobi
ftsp_10_50_0	26	92	92	-	-	12.049	864.247	182.411	428.012
ftsp_10_50_1	17	81	81	38.070	-	75.002	301.976	61.758	286.511
ftsp_10_50_2	40	42	42	-	-	14.478	251.179	31.712	60.536
ftsp_10_50_3	31	57	57	-	-	5.113	142.688	9.509	28.379
ftsp_10_50_4	17	20	20	84.103	213.489	0.579	1.390	2.272	11.209
ftsp_10_50_5	32	32	32	-	1216.878	4.311	87.372	8.049	34.128
ftsp_10_50_6	36	36	36	-	-	15.484	193.194	67.375	90.172
ftsp_10_50_7	85	85	85	-	-	311.254	1657.113	270.304	374.590
ftsp_10_50_8	85	85	85	-	-	168.025	816.382	19.046	293.097
ftsp_10_50_9	44	44	44	-	-	8.983	262.177	85.624	86.147
ftsp_30_100_0	199	199	199	-	-	103.406	-	690.726	1687.984
ftsp_30_100_1	45	68	68	-	-	3.723	5.976	25.507	45.858
ftsp_30_100_2	29	108	108	-	-	1027.879	3438.606	419.317	1156.148
ftsp_30_100_3	108	108	108	-	-	844.046	-	833.258	1007.004
ftsp_30_100_4	31	67	67	-	-	6.704	24.405	39.845	19.379
ftsp_30_100_5	163	163	163	-	-	74.745	-	146.850	559.970
ftsp_30_100_6	30	50	50	-	-	0.672	0.741	0.481	0.623
ftsp_30_100_7	177	177	177	-	-	-	-	462.554	725.661
ftsp_30_100_8	45	96	96	-	-	53.870	158.214	81.909	2800.482
ftsp_30_100_9	270	270	270	-	-	-	-	2481.737	3195.945

Tabela 3.3: Optimalna rešenja za instance velikih dimenzija

Inst.	LB	T_{max}	opt	ILP1		ILP2		ILP3	
				CPLEX	Gurobi	CPLEX	Gurobi	CPLEX	Gurobi
ftsp_30_500_0	167	167	-	-	-	-	-	-	-
ftsp_30_500_1	718	718	-	-	-	-	-	-	-
ftsp_30_500_2	269	269	-	-	-	-	-	-	-
ftsp_30_500_3	117	126	-	-	-	-	-	-	-
ftsp_30_500_4	528	528	-	-	-	-	-	-	-
ftsp_50_200_0	73	567	-	-	-	-	-	-	-
ftsp_50_200_1	89	116	116	-	-	117.996	-	200.537	321.738
ftsp_50_200_2	423	423	423	-	-	-	-	-	7139.659
ftsp_50_200_3	102	109	109	-	-	101.565	704.355	357.902	425.117
ftsp_50_200_4	59	344	344	-	-	-	-	-	4110.832
ftsp_100_500_0	208	578	-	-	-	-	-	-	-
ftsp_100_500_1	197	224	-	-	-	-	-	-	-
ftsp_100_500_2	300	300	-	-	-	-	-	-	-
ftsp_100_500_3	794	830	-	-	-	-	-	-	-
ftsp_100_500_4	154	224	-	-	-	-	-	-	-

Poredeći vremena izvršavanja CPLEX i Gurobi rešavača za formulaciju ILP3, CPLEX ima manje vreme izvršavanja od Gurobi-a za 19 od 20 instanci. Samo za instancu *ftsp*_30_100_4 Gurobi je brže pronašao optimalno rešenje od CPLEX-a. Ukupno vreme rada CPLEX-a za svih 20 instanci je u ovom slučaju bilo 5920.244 sekundi, dok za Gurobi 12891.835 sekundi.

Rezultati prikazani u Tabeli 3.3 pokazuju da ni CPLEX ni Gurobi nisu imali uspeha sa formulacijom ILP1, gde ni za jednu od 15 instanci nisu dobili optimalno rešenje. Što se tiče formulacije ILP2, CPLEX je uspeo da dobije rešenje za 2 od 15 instanci, dok Gurobi za samo jednu instancu. Formulacija ILP3 se i kod instanci velikih dimenzija pokazala najuspešnijom, gde je CPLEX uspeo da dođe do rešenja za 2 od 15 instanci, a Gurobi za 4 od 15 instanci. Vremena izvršavanja CPLEX-a, za sve instance gde su oba rešavača došla do rešenja, su manja u odnosu na Gurobi, međutim CPLEX se pokazuje lošijim u odnosu na Gurobi u smislu broja instanci gde je dostignuto optimalno rešenje.

Kao što je i bilo očekivano na osnovu analize iz odeljka 3.3.2, uzimajući u obzir rezultate iz sve tri Tabele 3.1 - 3.3, ILP3 formulacija se pokazala najuspešnijom. ILP rešavači se mogu koristiti za pronalaženje optimalnih rešenja za instance malih i srednjih dimenzija. Za te instance, CPLEX se pokazao bržim u odnosu na Gurobi. Za velike instance, ovaj pristup se nije pokazao toliko uspešnim, jer su optimalna rešenja pronađena za samo 4 instance od 15, i u ovom slučaju Gurobi je imao više uspeha od CPLEX-a.

3.5.3 Rezultati VNS algoritama

Sve tri varijante metode promenljivih okolina, opisane u odeljcima 3.4, 3.4.2 i 3.4.3 su programirane u programskom jeziku C. Kriterijum zaustavljanja za sva tri VNS algoritma je postavljen na maksimalni broj iteracija $itermax = 100$. Vrednosti ostalih parametara su: $k_{\min} = 2$, $k_{\max} = 20$, $p = 0.4$. Zbog stohasticke prirode VNS-a, algoritmi su izvršavani po 20 puta na svakoj instanci. U Tabelama 3.4 - 3.6 je dat pregled rezultata izvršavanja VNS algoritama iz odeljka 3.4 na svim generisanim instancama. Tabela 3.4 predstavlja rezultate dobijene na instancama malih dimenzija ($v_{\max} \leq 10$ i $e_{\max} \leq 10$), dok se Tabele 3.5 i 3.6 odnose na rezultate dobijene za instance srednjih dimenzija ($v_{\max}$=10, 30 i $e_{\max} = 50, 100$) i velikih dimenzija ($v_{\max}$=30, 50, 100 i $e_{\max} = 200, 500$), respektivno. Zbog velikog ukupnog

vremena izvršavanja 100 iteracija za instance iz Tabele 3.6 predstavljeni su rezultati samo za po pet instanci svakog tipa, umesto deset, kao što je to slučaj u Tabelama 3.4 i 3.5.

Tabela 3.4 je organizovana na sledeći način:

- U prvoj i drugoj koloni se nalaze ime instance i vrednost elementarne donje granice, predstavljene na isti način kao i u Tabelama 3.1-3.3.
- U trećoj koloni su vrednosti optimalnih rešenja do kojih se došlo u prethodnom odeljku, koristeći CPLEX ili Gurobi za navedene ILP formulacije.
- U četvrtoj koloni, označenoj sa VNS_{naj}, su prikazane najbolje vrednosti funkcije cilja dobijene VNS-om u 20 izvršavanja algoritma. Ukoliko algoritam dostiže optimalno rešenje koje se nalazi u trećoj koloni, umesto vrednosti funkcije cilja upisana je oznaka "*opt*".
- Naredne dve kolone se odnose na prosečno vreme izvršavanja t potrebno da bi se odredilo najbolje rešenje VNS-om i prosečno ukupno vreme izvršavanja t_{tot} potrebno da algoritam završi sa radom.
- Poslednje dve kolone ($avnsp$ i σ) sadrže informacije o kvalitetu rešenja dobijenih u svih 20 izvršavanja. Procentualna relativna greška rešenja $avnsp$ se računa po formuli $avnsp = \frac{1}{20}\sum_{i=1}^{20} vnsp_i$, gde je $vnsp_i = 100 * \frac{VNS_i - VNS_{naj}}{VNS_{naj}}$. VNS_i predstavlja rešenje dobijeno VNS-om u i-tom izvršavanju, $i = 1, ..., 20$. Standardna devijacija σ prosečnog odstupanja rešenja $vnsp_i, i = 1, ..., 20$ u odnosu na VNS_{naj} se dobija po formuli $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(vnsp_i - avnsp)^2}$.

Kao što se može videti iz Tabele 3.4 VNS je pokazao odlične rezultate na ovim instancama, dostigavši optimalna rešenja u svim slučajevima jako brzo. Najduže prosečno vreme izvršavanja je manje od 0.021 sekunde. Posmatrajući poslednje dve kolone može se videti da je za svaku instancu VNS algoritam dostigao optimalno rešenje u svih 20 izvršavanja, što ukazuje na njegovu efikasnost.

Podaci u Tabelama 3.5 i 3.6 su predstavljeni na sličan način kao i u Tabeli 3.4. Za instance za koje nije poznata vrednost optimalnog rešenja, elementarna donja granica problema je poznata i lako se može izračunati. U tim slučajevima, ukoliko su rešenja dobijena VNS-om jednaka elementarnoj donjoj granici, onda ona jesu optimalna. Međutim, ukoliko su rešenja dobijena VNS-om veća od vrednosti

Tabela 3.4: Rezultati VNS algoritma na malim instancama

Inst.	LB	opt	VNS_{naj}	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.018	0.021	0.000	0.000
ftsp_5_10_1	10	15	opt	0.020	0.021	0.000	0.000
ftsp_5_10_2	12	12	opt	0.015	0.017	0.000	0.000
ftsp_5_10_3	14	14	opt	0.018	0.020	0.000	0.000
ftsp_5_10_4	10	10	opt	0.012	0.014	0.000	0.000
ftsp_5_10_5	6	13	opt	0.014	0.018	0.000	0.000
ftsp_5_10_6	12	13	opt	0.017	0.020	0.000	0.000
ftsp_5_10_7	8	8	opt	0.018	0.019	0.000	0.000
ftsp_5_10_8	11	11	opt	0.013	0.014	0.000	0.000
ftsp_5_10_9	26	26	opt	0.019	0.020	0.000	0.000
ftsp_10_10_0	12	15	opt	0.013	0.019	0.000	0.000
ftsp_10_10_1	12	12	opt	0.007	0.009	0.000	0.000
ftsp_10_10_2	5	14	opt	0.010	0.012	0.000	0.000
ftsp_10_10_3	19	19	opt	0.019	0.020	0.000	0.000
ftsp_10_10_4	17	17	opt	0.013	0.015	0.000	0.000
ftsp_10_10_5	10	15	opt	0.013	0.014	0.000	0.000
ftsp_10_10_6	3	15	opt	0.011	0.013	0.000	0.000
ftsp_10_10_7	11	14	opt	0.011	0.012	0.000	0.000
ftsp_10_10_8	11	14	opt	0.012	0.013	0.000	0.000
ftsp_10_10_9	2	14	opt	0.010	0.011	0.000	0.000

elementarne donje granice, ne može se utvrditi da li su te vrednosti optimalne ili ne. Uzimajući ovo u obzir, u trećoj koloni VNS_{naj} u slučajevima kada je VNS dostigao elementarnu donju granicu stoji oznaka "$LBopt$".

VNS algoritam dostiže sva poznata optimalna rešenja, dok čak u tri slučaja gde optimalno rešenje nije poznato za instance $ftsp$_30_500_1, $ftsp$_30_500_2 i $ftsp$_30_500_4, optimalnost dobijenog VNS rešenja proizilazi iz poklapanja dobijene vrednosti sa elementarnom donjom granicom.

Implementirana procedura lokalne pretrage, sa jedne strane, vrši efikasno pretraživanje okoline rešenja jer menjajući svaka dva njegova elementa pretražuje veliku okolinu. Za instance malih dimenzija ovaj postupak je dosta brz. Međutim, za instance velikih dimenzija ovaj postupak, praćen sortiranjem koje se izvršava prilikom računanja vrednosti funkcije cilja, dosta usporava algoritam. Zbog toga su vremena izvršavanja VNS-a za neke od instanci iz Tabele 3.6 veoma velika. Takođe se može uočiti da različite instance istih dimenzija imaju značajno drugačija prosečna vremena izvršavanja. Razlog tome je upravo različita efikasnost lokalne pretrage za

Tabela 3.5: Rezultati VNS algoritma na srednjim instancama

Inst.	LB	opt	VNS_{naj}	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	5.681	5.775	0.000	0.000
ftsp_10_50_1	17	81	opt	4.733	4.811	0.000	0.000
ftsp_10_50_2	40	42	opt	3.371	3.532	0.000	0.000
ftsp_10_50_3	31	57	opt	4.147	4.212	0.000	0.000
ftsp_10_50_4	17	20	opt	1.903	2.025	0.000	0.000
ftsp_10_50_5	32	32	opt	2.493	2.725	1.406	1.817
ftsp_10_50_6	36	36	opt	2.962	3.328	0.000	0.000
ftsp_10_50_7	85	85	opt	6.015	6.114	0.000	0.000
ftsp_10_50_8	85	85	opt	5.141	5.225	0.000	0.000
ftsp_10_50_9	44	44	opt	3.206	3.525	0.000	0.000
ftsp_30_100_0	199	199	opt	72.225	73.568	0.000	0.000
ftsp_30_100_1	45	68	opt	26.481	27.229	0.000	0.000
ftsp_30_100_2	29	108	opt	42.427	44.350	0.093	0.278
ftsp_30_100_3	108	108	opt	50.615	52.523	0.000	0.000
ftsp_30_100_4	31	67	opt	26.903	27.724	0.970	4.188
ftsp_30_100_5	163	163	opt	59.277	60.400	0.000	0.000
ftsp_30_100_6	30	50	opt	7.918	8.070	0.000	0.000
ftsp_30_100_7	177	177	opt	71.007	72.350	0.000	0.000
ftsp_30_100_8	45	96	opt	42.533	43.724	0.000	0.000
ftsp_30_100_9	270	270	opt	98.240	100.092	0.000	0.000

Tabela 3.6: Rezultati VNS algoritma na velikim instancama

Inst.	LB	opt	VNS_{naj}	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	168	9348.251	9705.179	3.075	1.746
ftsp_30_500_1	718	-	$LBopt$	34507.249	34860.305	0.000	0.000
ftsp_30_500_2	269	-	$LBopt$	11505.293	12513.212	0.074	0.189
ftsp_30_500_3	117	-	129	6187.198	6739.084	5.271	2.671
ftsp_30_500_4	528	-	$LBopt$	20450.232	20823.454	0.000	0.000
ftsp_50_200_0	73	-	567	1409.227	1430.328	0.000	0.000
ftsp_50_200_1	89	116	opt	358.705	368.036	1.164	2.560
ftsp_50_200_2	423	423	opt	1033.391	1048.931	0.000	0.000
ftsp_50_200_3	102	109	opt	322.114	326.639	5.872	4.132
ftsp_50_200_4	59	344	opt	851.106	863.951	0.000	0.000
ftsp_100_500_0	208	-	578	22430.227	22846.668	0.000	0.000
ftsp_100_500_1	197	-	227	9998.633	10234.793	5.132	2.608
ftsp_100_500_2	300	-	301	13189.866	13547.214	3.090	1.929
ftsp_100_500_3	794	-	830	31584.480	32155.459	0.000	0.000
ftsp_100_500_4	154	-	224	10114.071	10383.340	1.674	4.057

Tabela 3.7: Rezultati VNS algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	VNS_{naj}	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	171	5.393	2.716
ftsp_30_500_1	718	-	*LBopt*	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	2.034	1.732
ftsp_30_500_3	117	-	135	4.327	3.738
ftsp_30_500_4	528	-	*LBopt*	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	3.165	2.150
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	237	9.008	4.292
ftsp_100_500_2	300	-	306	10.327	5.433
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	11.964	9.504

različite primere, tj. u nekim slučajevima lokalna pretraga mnogo brže dođe do poboljšanja rešenja nego u drugim primerima.

Za veliki broj instanci u Tabelama 3.5 i 3.6 dobijeno je isto rešenje u svih 20 izvršavanja VNS-a (17 od 20 za srednje i 7 od 15 za velike) što se može videti iz poslednje dve kolone gde su i *avnsp* i σ jednaki 0.

Kao što je već napomenuto, za instance velikih dimenzija isti broj iteracija VNS-a se izvršavao za značajno različito ukupno vreme. Kako bi se izbegla ova velika vremena izvršavanja, izvršeno je dodatno testiranje sa drugačijim kriterijom zaustavljanja. Za svako izvršavanje postavljen je kriterijum zaustavljanja na ukupno vreme izvršavanja $t_{tot} = 1000s$ umesto 100 iteracija. Pregled rezultata dobijenih VNS-om sa ovim vremenskim kriterijumom zaustavljanja za velike instance dat je u Tabeli 3.7. Tabela je organizovana na sličan način kao i prethodne, sa izuzetkom izostavljenih kolona koje se odnose na vreme jer je $t_{tot} = 1000s$ za sve instance.

Poredeći rezultate u Tabelama 3.6 i 3.7, u slučajevima kada je u svih 20 izvršavanja za jednu instancu dobijeno isto rešenje nakon 100 iteracija (*avnsp* i σ imaju vrednost 0), isti rezultat je dostignut i nakon 1000s. U većini slučajeva 1000s je bilo dovoljno kako bi se pronašao isti minimum kao i u 100 iteracija, ali vrednosti *avnsp* i σ su dosta veće, kao što je bilo i za očekivati. Ova činjenica ukazuje na to da je za neka pokretanja VNS-u trebalo više vremena kako bi dostigao vrednost jednaku ili

blisku VNS_{naj}. Izuzeci su instance $ftsp\_50\_200\_1$ i $ftsp\_50\_200\_3$ gde se 100 iteracija završilo za manje od 1000s pa je nakon produžavanja ukupnog vremena rada VNS-a u 20 izvršavanja se dostigla rešenja bliža pronađenom minimumu. Za četiri instance u Tabeli 3.7 najbolji rezultat u svih 20 poziva ima veću vrednost nego u Tabeli 3.6.

3.5.4 Rezultati VNS-VND i VNS-LS1 algoritama

U ovom odeljku su prikazani eksperimentalni rezultati VNS-VND algoritma opisanog u odeljku 3.4.2 i VNS algoritma sa ubrzanom lokalnom pretragom, u daljem tekstu u oznaci VNS-LS1, iz odeljka 3.4.3. Skup instanci na kojima su vršena testiranja je isti kao i u prethodnim odeljcima. Radi lakšeg i realističnijeg poređenja dobijenih rezultata, za obe implementacije su korišćeni isti parametri kao i za VNS iz prethodnog odeljka: $k_{\min} = 2$, $k_{\max} = 20$, $p = 0.4$. Algoritmi su izvršavani po 20 puta na svakoj instanci. Rezultati VNS-VND algoritma su dati u Tabelama 3.8-3.10, dok su rezultati VNS-LS1 algoritma u tabelama 3.11-3.14. Sve tabele su organizovane na isti način kao i tabele iz prethodnog odeljka. Kriterijum zaustavljanja VNS-VND algoritma za instance malih i srednjih dimenzija je postavljen na ukupan broj od 100 iteracija. Zbog velikog ukupnog vremena izvršavanja VNS-VND algoritma za instance velikih dimenzija, za ove instance je korišćen samo kriterijum zaustavljanja od 1000s.

Iz Tabela 3.8 i 3.9 se lako uočava da je VNS-VND algoritam dostigao unapred poznata optimalna rešenja za sve instance malih i srednjih dimenzija u svih 20 izvršavanja. Poredeći dobijene rezultate sa rezultatima VNS-a iz Tabele 3.5 može se uočiti da VNS-VND daje kvalitetnije rezultate za instance srednjih dimenzija. Kao što je već pomenuto, VNS algoritam je dobio isto (optimalno) rešenje u svih 20 izvršavanja za 17 od 20 instanci. Bolji kvalitet rešenja dobijenih VNS-VND-om se odnosi na 3 instance gde VNS nije dobio isto rešenje u svih 20 izvršavanja ($ftsp\_10\_50\_5$, $ftsp\_30\_100\_2$ i $ftsp\_30\_100\_4$), što nije slučaj i sa VNS-VND-om. Upoređujući vremena izvršavanja obe metaheuristike za instance malih i srednjih dimenzija, VNS zanemarljivo brže pronalazi rešenja, ali VNS-VND dostiže rešenja boljeg kvaliteta u svih 20 izvršavanja za instance srednjih dimenzija.

U Tabeli 3.10 su predstavljeni rezultati VNS-VND algoritma za intance velikih dimenzija sa kriterijumom zaustavljanja od 1000 sekundi. Zbog velikog vremena

Tabela 3.8: Rezultati VNS-VND algoritma na malim instancama

Inst.	LB	opt	$VNS-VND_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.025	0.032	0.000	0.000
ftsp_5_10_1	10	15	opt	0.022	0.025	0.000	0.000
ftsp_5_10_2	12	12	opt	0.015	0.017	0.000	0.000
ftsp_5_10_3	14	14	opt	0.018	0.019	0.000	0.000
ftsp_5_10_4	10	10	opt	0.014	0.015	0.000	0.000
ftsp_5_10_5	6	13	opt	0.018	0.020	0.000	0.000
ftsp_5_10_6	12	13	opt	0.021	0.022	0.000	0.000
ftsp_5_10_7	8	8	opt	0.013	0.014	0.000	0.000
ftsp_5_10_8	11	11	opt	0.015	0.020	0.000	0.000
ftsp_5_10_9	26	26	opt	0.025	0.028	0.000	0.000
ftsp_10_10_0	12	15	opt	0.011	0.012	0.000	0.000
ftsp_10_10_1	12	12	opt	0.014	0.014	0.000	0.000
ftsp_10_10_2	5	14	opt	0.012	0.012	0.000	0.000
ftsp_10_10_3	19	19	opt	0.020	0.021	0.000	0.000
ftsp_10_10_4	17	17	opt	0.014	0.016	0.000	0.000
ftsp_10_10_5	10	15	opt	0.014	0.017	0.000	0.000
ftsp_10_10_6	3	15	opt	0.012	0.013	0.000	0.000
ftsp_10_10_7	11	14	opt	0.012	0.014	0.000	0.000
ftsp_10_10_8	11	14	opt	0.011	0.014	0.000	0.000
ftsp_10_10_9	2	14	opt	0.010	0.011	0.000	0.000

izvršavanja algoritma koje bi bilo posledica kriterijuma zaustavljanja od 100 iteracija, samo je vremenski kriterijum uzet u obzir. Poredeći dobijene rezultate sa rezultatima VNS-a iz Tabele 3.7, dolazi se do zaključka da VNS-VND daje znatno bolja rešenja od VNS-a. VNS algoritam je dostigao 7 optimalnih rešenja od 15 instanci, gde od tih 7 rešenja optimalnost 3 rešenja proizilazi iz poklapanja sa elementarnom donjom granicom. Sa druge strane, VNS-VND algoritam je dostigao 9 optimalnih rešenja od 15 instanci: svih 4 poznatih iz odeljka 3.5.2 i 5 vrednosti čija optimalnost sledi iz poklapanja sa elementarnom donjom granicom. Pored instanci $ftsp$_30_500_0 i $ftsp$_100_500_2 gde VNS-VND dobija bolja rešenja od VNS-a, koja su ujedno i optimalna, VNS-VND dobija bolja rešenja od VNS-a i za instance $ftsp$_30_500_3 i $ftsp$_100_500_1 za koje se ne može utvrditi optimalnost.

Poredeći kvalitete dobijenih rešenja u Tabelama 3.7 i 3.10, VNS je dostigao isto rešenje u svih 20 izvršavanja za 8 od 15 instanci, dok se VNS-VND i ovom slučaju pokazao boljim sa 11 od 15 instanci. Za preostale 4 instance gde su vrednosti *avnsp* i σ različiti od nule, ovi parametri imaju manje vrednosti za VNS-VND nego za

Tabela 3.9: Rezultati VNS-VND algoritma na srednjim instancama

Inst.	LB	opt	$VNS-VND_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	5.900	5.999	0.000	0.000
ftsp_10_50_1	17	81	opt	4.884	4.964	0.000	0.000
ftsp_10_50_2	40	42	opt	3.695	3.842	0.000	0.000
ftsp_10_50_3	31	57	opt	4.343	4.413	0.000	0.000
ftsp_10_50_4	17	20	opt	2.259	2.314	0.000	0.000
ftsp_10_50_5	32	32	opt	3.125	3.216	0.000	0.000
ftsp_10_50_6	36	36	opt	3.583	3.702	0.000	0.000
ftsp_10_50_7	85	85	opt	6.406	6.511	0.000	0.000
ftsp_10_50_8	85	85	opt	5.274	5.361	0.000	0.000
ftsp_10_50_9	44	44	opt	3.791	3.878	0.000	0.000
ftsp_30_100_0	199	199	opt	86.978	88.528	0.000	0.000
ftsp_30_100_1	45	68	opt	29.147	29.497	0.000	0.000
ftsp_30_100_2	29	108	opt	46.443	47.143	0.000	0.000
ftsp_30_100_3	108	108	opt	54.489	56.208	0.000	0.000
ftsp_30_100_4	31	67	opt	33.880	34.293	0.000	0.000
ftsp_30_100_5	163	163	opt	78.232	79.685	0.000	0.000
ftsp_30_100_6	30	50	opt	11.180	11.401	0.000	0.000
ftsp_30_100_7	177	177	opt	87.588	89.286	0.000	0.000
ftsp_30_100_8	45	96	opt	45.654	46.357	0.000	0.000
ftsp_30_100_9	270	270	opt	99.061	100.980	0.000	0.000

Tabela 3.10: Rezultati VNS-VND algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	$VNS-VND_{naj}$	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	$LBopt$	0.180	0.274
ftsp_30_500_1	718	-	$LBopt$	0.000	0.000
ftsp_30_500_2	269	-	$LBopt$	0.000	0.000
ftsp_30_500_3	117	-	127	0.630	0.878
ftsp_30_500_4	528	-	$LBopt$	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	0.000	0.000
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	224	3.482	3.080
ftsp_100_500_2	300	-	$LBopt$	0.217	0.190
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	0.000	0.000

Tabela 3.11: Rezultati VNS-LS1 algoritma na malim instancama

Inst.	LB	opt	$VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.015	0.019	0.000	0.000
ftsp_5_10_1	10	15	opt	0.009	0.011	0.000	0.000
ftsp_5_10_2	12	12	opt	0.007	0.009	0.000	0.000
ftsp_5_10_3	14	14	opt	0.008	0.010	0.000	0.000
ftsp_5_10_4	10	10	opt	0.009	0.011	0.000	0.000
ftsp_5_10_5	6	13	opt	0.014	0.015	0.000	0.000
ftsp_5_10_6	12	13	opt	0.009	0.012	0.000	0.000
ftsp_5_10_7	8	8	opt	0.011	0.012	0.000	0.000
ftsp_5_10_8	11	11	opt	0.007	0.009	0.000	0.000
ftsp_5_10_9	26	26	opt	0.009	0.011	0.000	0.000
ftsp_10_10_0	12	15	opt	0.007	0.008	0.000	0.000
ftsp_10_10_1	12	12	opt	0.005	0.007	0.000	0.000
ftsp_10_10_2	5	14	opt	0.007	0.008	0.000	0.000
ftsp_10_10_3	19	19	opt	0.011	0.012	0.000	0.000
ftsp_10_10_4	17	17	opt	0.007	0.009	0.000	0.000
ftsp_10_10_5	10	15	opt	0.008	0.008	0.000	0.000
ftsp_10_10_6	3	15	opt	0.007	0.009	0.000	0.000
ftsp_10_10_7	11	14	opt	0.009	0.009	0.000	0.000
ftsp_10_10_8	11	14	opt	0.007	0.007	0.000	0.000
ftsp_10_10_9	2	14	opt	0.005	0.007	0.000	0.000

VNS, što ukazuje na veću pouzdanost rešenja dobijenih VNS-VND-om nego VNS-om.

Iz Tabela 3.11 - 3.13 može se videti da VNS-LS1 algoritam dostiže identične vrednosti rešenja kao i VNS-VND algoritam za sve instance. Zbog značajno ubrzane lokalne pretrage, kriterijum zaustavljanja za ovu metodu je za sve instance najpre bio 100 iteracija. Poredeći vremena izvršavanja VNS-LS1 sa VNS i VNS-VND algoritmima, VNS-LS1 znatno brže dolazi do rešenja. Najduže prosečno vreme izvršavanja za instance malih dimenzija je 0.019 sekunde, dok je za instance srednjih dimenzija 1.982 sekunde što je drastično manje u poređenju sa vremenom koje je bilo potrebno za VNS i VNS-VND. U slučaju velikih instanci, VNS-LS1 dolazi do istih vrednosti rešenja kao i VNS-VND za dosta manje vremena. Najduže prosečno vreme izvršavanja 100 iteracija je 191.594 sekunde.

Što se tiče kvaliteta dobijenih rešenja u svih 20 izvršavanja, VNS-LS1 je za nijansu lošiji od VNS-VND-a, ali bolji od VNS-a: za instance malih dimenzija isto rešenje se dostize za svih 20 instanci, dok za srednje instance za 19 od 20, a za velike

Tabela 3.12: Rezultati VNS-LS1 algoritma na srednjim instancama

Inst.	LB	opt	$VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	0.223	0.226	0.000	0.000
ftsp_10_50_1	17	81	opt	0.186	0.189	0.000	0.000
ftsp_10_50_2	40	42	opt	0.178	0.185	0.000	0.000
ftsp_10_50_3	31	57	opt	0.163	0.167	0.000	0.000
ftsp_10_50_4	17	20	opt	0.107	0.111	0.000	0.000
ftsp_10_50_5	32	32	opt	0.141	0.158	0.469	1.111
ftsp_10_50_6	36	36	opt	0.168	0.179	0.000	0.000
ftsp_10_50_7	85	85	opt	0.238	0.244	0.000	0.000
ftsp_10_50_8	85	85	opt	0.203	0.208	0.000	0.000
ftsp_10_50_9	44	44	opt	0.181	0.189	0.000	0.000
ftsp_30_100_0	199	199	opt	1.497	1.527	0.000	0.000
ftsp_30_100_1	45	68	opt	0.569	0.582	0.000	0.000
ftsp_30_100_2	29	108	opt	1.142	1.168	0.000	0.000
ftsp_30_100_3	108	108	opt	1.364	1.393	0.000	0.000
ftsp_30_100_4	31	67	opt	0.648	0.658	0.000	0.000
ftsp_30_100_5	163	163	opt	1.215	1.239	0.000	0.000
ftsp_30_100_6	30	50	opt	0.176	0.178	0.000	0.000
ftsp_30_100_7	177	177	opt	1.471	1.499	0.000	0.000
ftsp_30_100_8	45	96	opt	0.901	0.919	0.000	0.000
ftsp_30_100_9	270	270	opt	1.945	1.982	0.000	0.000

instance za 10 od 15 instanci.

U Tabeli 3.14 dat je prikaz rezultata VNS-LS1 algoritma na velikim instancama sa kriterijumom zaustavljanja od 1000s. Poredeći rezultate sva tri algoritma na velikim instancama, sa ovim vremenskim kriterijumom zaustavljanja, VNS-LS1 je dostigao ista rešenja kao i VNS-VND, koja su bolja od VNS algoritma. Poredeći rezultate dobijene u svih 20 izvršavanja VNS-LS1 je za sve instance osim jedne ($ftsp\_30\_500\_3$) dostigao iste vrednosti rešenja, dok su kod VNS-VND algoritma vrednosti *avnsp* i σ jednake nuli za 11 od 15 instanci.

Tabela 3.13: Rezultati VNS-LS1 algoritma na velikim instancama

Inst.	LB	opt	$VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	*LBopt*	62.196	65.943	0.060	0.261
ftsp_30_500_1	718	-	*LBopt*	179.368	182.375	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	72.111	73.915	0.000	0.000
ftsp_30_500_3	117	-	127	52.790	55.528	0.433	0.390
ftsp_30_500_4	528	-	*LBopt*	108.849	110.796	0.000	0.000
ftsp_50_200_0	73	-	567	14.921	15.150	0.000	0.000
ftsp_50_200_1	89	116	opt	3.998	4.099	0.000	0.000
ftsp_50_200_2	423	423	opt	11.461	11.633	0.000	0.000
ftsp_50_200_3	102	109	opt	4.173	4.284	0.367	0.731
ftsp_50_200_4	59	344	opt	9.793	9.937	0.000	0.000
ftsp_100_500_0	208	-	578	96.278	98.000	0.000	0.000
ftsp_100_500_1	197	-	224	72.896	74.163	1.429	2.034
ftsp_100_500_2	300	-	*LBopt*	96.949	99.778	0.233	0.213
ftsp_100_500_3	794	-	830	188.306	191.594	0.000	0.000
ftsp_100_500_4	154	-	224	65.394	66.930	0.000	0.000

Tabela 3.14: Rezultati VNS-LS1 algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	$VNS-LS1_{naj}$	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	*LBopt*	0.000	0.000
ftsp_30_500_1	718	-	*LBopt*	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	0.000	0.000
ftsp_30_500_3	117	-	127	0.278	0.377
ftsp_30_500_4	528	-	*LBopt*	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	0.000	0.000
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	224	0.000	0.000
ftsp_100_500_2	300	-	*LBopt*	0.000	0.000
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	0.000	0.000

Glava 4

Gaus-VNS za problem raspoređivanja prenosa datoteka

Pošto je Gaus-VNS postigao odlične rezultate za rešavanje problema kontinualne optimizacije, prirodno je bilo postaviti pitanje da li ga je moguće primeniti na probleme diskretne optimizacije i sa kakvim rezultatima. Zbog toga će u ovoj glavi biti opisana implementacija Gaus-VNS metode za rešavanje FTSP.

4.1 Opis algoritma

U odeljku 3.4 opisan je VNS algoritam za rešavanje FTSP. Elementi x_k vektora $x \in X$ su bili celi brojevi koji su označavali prioritete dodeljene svakoj grani. Na osnovu ovog vektora, na jedinstven način se mogao formirati raspored prenosa datoteka i time izračunati vrednost funkcije cilja. Jedan od osnovnih koraka VNS algoritma, razmrdavanje, koristio je okoline N_k takve da za dato k, okolina trenutnog rešenja $N_k(x)$ sadrži sve vektore prioriteta koji se mogu razlikovati od vektora prioriteta rešenja x u najviše k vrednosti.

Adaptacijom Gaus-VNS metode za FTSP više ne bi postojale okoline rešenja $N_k(x)$ u kojima se vrši pretraga. Postojaće samo jedna okolina i ona je jednaka celom prostoru pretrage. Problem pretrage udaljenih regiona bi bio na ovaj način rešen jer u svakom trenutku u samo jednom koraku možemo doći do bilo kog rešenja. Sa ovom izmenom, ostaje problem kako izbeći da se algoritam ne pretvori u slučajnu pretragu celog prostora. U postupku razmrdavanja treba izabrati jedno rešenje iz skupa $X = \{1, ...n\}^n$ (n je broj grana). Birajući nasumično vektore iz skupa

$X = \{1, ...n\}^n$ moguće je da bi drastično "odskakali" od trenutno najboljeg rešenja i time sprečili pronalazak optimalnog rešenja koje je bilo možda blizu prethodnog. Korišćenjem upravo normalne raspodele za biranje vektora prioriteta, rešen je i ovaj problem, tj. pronađen je balans između toga da pretragu vršimo u blizini trenutno najboljeg rešenja, ali da sa nekom verovatnoćom u nekom trenutku ipak možemo da "odskočimo" dosta dalje od njega i time izbegnemo klopke lokalnih minimuma.

Kod globalne optimizacije skup dopustivih rešenja problema je neprebrojiv, dok kod diskretnog FTS problema pretraga se vrši po konačnom skupu. Kako bi se metoda za rešavanje problema kontinualne optimizacije mogla adaptirati za rešavanje diskretnih problema neophodno je izvršiti preslikavanje iz neprebrojivog skupa u konačan skup.

Kako bi se realizovalo razmrdavanje iz Gaus-VNS metode nailazi se na teškoću pri reprezentaciji rešenja problema. Naime, za rešavanje FTSP-a korišćene su samo celobrojne vrednosti, dok ako želimo da koristimo normalnu raspodelu, rezultati će biti realne vrednosti. Kako se formiranje jedinstvenog rasporeda prenosa vrši tako što se porede vrednosti prioriteta, tj. grana koja ima veći prioritet započinje transfer pre one koja ima niži prioritet, neophodno je samo da se uskladi način kako će se sada predstavljati prioriteti. Umesto reprezentacije rešenja opisane u odeljku 3.4 može se koristiti težinska reprezentacija, tj. celobrojne vrednosti mogu se zameniti sa realnim vrednostima koje predstavljaju prioritete grana. Princip ostaje isti, veći realni broj, predstavljajući prioritet grane, ima prednost u odnosu na manji realan broj. Nakon dobijanja rasporeda prenosa, dalje se sva izračunavanja vrše nad tim rasporedima bez korišćenja težina.

Koristeći ovakvu reprezentaciju rešenja, sa realnim vrednostima prioriteta, jasno je da će više realnih reprezentacija rešenja odgovarati jednom istom rasporedu. Na primer, vektori prioriteta $x_1 = (2.8, 3.4, 1.1)$ i $x_2 = (2.8, 3.3, 1.1)$ imaju isti raspored prenosa: najpre grana e_2 sa prioritetom 3.4, odnosno 3.3, a zatim grane e_1 pa e_3 koje u oba slučaja imaju manje prioritete. Jasno je da će na ovaj način svakom rasporedu prenosa za FTS problem odgovarati neprebrojivo mnogo vektora prioriteta.

Ovakva realizacija preslikavanja neprebrojivog skupa dopustivih rešenja u konačan skup čini ovu primenu interesantnijom. Iako se Gaus-VNS pokazao uspešnim za rešavanje problema kontinualne optimizacije otvara se pitanje da li će zbog ovako realizovanog preslikavanja njegova efikasnost biti oslabljena za probleme diskretne

optimizacije. Drugim rečima, može se desiti da se kroz iteracije menjaju vektori prioriteta, ali da zapravo ni u jednom koraku ne dolazi do promene samog rasporeda prenosa.

Uzimajući u obzir prethodna razmatranja i izmene, Gaus-VNS algoritam za FTSP se može realizovati na sledeći način. Skup dopustivih rešenja X sada je skup svih n-dimenzionih vektora koji sadrže realne vrednosti. Kao i do sada, vrednosti vektora $x \in X$ predstavljaju prioritete grana s tim što x više neće biti predstavljen kao u prethodnoj glavi sa elementima koji su celobrojne vrednosti, nego će biti vektor od n realnih vrednosti. Jedinstveni raspored prenosa na osnovu ovih prioriteta i vrednost funkcije cilja se dobijaju na isti način kao što je opisano u odeljku 3.4, sa jedinom izmenom što poredimo realne brojeve umesto celih pri određivanju koja grana ima prednost u započinjanju svog prenosa.

Inicijalizacija se vrši na slučajan način. Na početku Gaus-VNS procedure za svaki graf početno rešenje x se formira kao niz slučajnih n realnih brojeva na intervalu od 0 do $n-1$.

Kako će se u procesu razmrdavanja birati slučajan vektor prioriteta koristeći normalnu raspodelu sa očekivanjem x (x je trenutno najbolje rešenje) i nekom disperzijom σ_k, nephodno je unapred definisati disperzije koje će se koristiti u tom koraku. Parametar k, $k_{\min} \leq k \leq k_{\max}$, sada ima ulogu indeksa koji označava koji element iz niza unapred definisanih disperzija $\sigma_{k_{\min}}, ..., \sigma_{k_{\max}}$ će se koristiti u datoj iteraciji. S ozbirom da k predstavlja indeks, prirodno je uzeti za $k_{\min} = 1$, pa je neophodno definisati samo ceo broj $k_{\max}$ i isti broj realnih vrednosti koji će se koristiti kao disperzije σ_k.

U procesu razmrdavanja, za zadato k, Gaus-VNS generiše novi vektor prioriteta x' prateći sledeći algoritam:

- Formiramo n nezavisnih vrednosti $z_1, ..., z_n$ dobijenih Gausovom raspodelom sa očekivanjem 0 i disperzijom 1.
- Svaki od ovih brojeva z_i pomnozimo sa σ_k.
- Vektor x' se formira najpre kao kopija trenutnog vektora x, a zatim se svakom elementu doda vrednost dobijena u prethodnom koraku sa iste pozicije, tj. $x' = x + z \cdot \sigma_k$.

Za generisanje slučajnih veličina z_i u okviru postupka razmrdavanja korišćen je Arens - Diter ([1], [33]) algoritam. Koristimo matricu kovarijansi kao u 2.1.1.

Nakon dobijanja x' za algoritam lokalne pretrage možemo iskoristiti bilo koji od algoritama korišćenih u sekcijama 3.4, 3.4.2 i 3.4.3. Na taj način dobijamo tri nove modifikacije metode:

- Gaus-VNS - okoline datog rešenja koje se pretražuju u okviru lokalne pretrage sadrže sva rešenja koja se mogu dobiti od polaznog x' zamenom proizvoljna dva njegova elementa.

- Gaus-VNS-VND - koristeći metodu spusta kroz promenljive okoline. Definišemo više manjih okolina, kao u odeljku 3.4.2 i pretraga se vrši kroz svaku od njih.

- Gaus-VNS-LS1 - okolinu čine sva rešenja koja se mogu dobiti od x' zamenom svaka dva njegova uzastopna elementa.

U sve tri varijante korišćena je strategija prvog poboljšanja. Donošenje odluke da li preći u novo rešenje i dalju pretragu vršiti od njega ili ne realizovano je na isti način kao i u glavi 3.4.

U opštem slučaju nije jednostavno izvršiti diskretizaciju Gaus-VNS metode odnosno realizovati na uspešan način reprezentaciju samih rešenja. Reprezentacija rešenja za VNS pristup iz prethodne glave predstavlja dobru osnovu za realizaciju Gaus-VNS pristupa. Adaptacija celobrojne podloge u prethodnoj glavi koju je bilo moguće zameniti sa realnom, omogućila je da se na uspešan način realizuje neophodno preslikavanje iz skupa dopustivih rešenja problema kontinualne optimizacije u skup dopustivih rešenja FTS problema. Međutim, za ostale probleme diskretne optimizacije za koje postoji razvijen VNS pristup, nije očigledno da li bi on mogao da vodi direktno ka Gaus-VNS pristupu.

4.2 Eksperimentalni rezultati

U ovom odeljku prikazani su rezultati priloženih metoda Gaus-VNS, Gaus-VNS-VND i Gaus-VNS-LS1 iz prethodnog odeljka za rešavanje FTS problema. Sva tri algoritma su testirana na istom skupu instanci kao i VNS, VNS-VND i VNS-LS1

algoritmi iz prethodne glave. Na svakoj instanci problema algoritmi su pokretani po 20 puta, kao i ranije.

Kako broj k više ne predstavlja veličinu okoline, nego indeks elementa iz niza disperzija za normalnu raspodelu, $k_{\max}$ takođe više nema istu ulogu kao i kod običnog VNS-a. U ovoj implementaciji korišćeno je $k_{\max} = 10$ i niz disperzija $\sigma = 0.01,\ 0.05,\ 0.10,\ 0.20,\ 0.50,\ 1.00,\ 2.00,\ 5.00,\ 10.00,\ 50.00$.

Tabele 4.1-4.3 daju potpun pregled rezultata primene Gaus-VNS algoritma na ovim instancama sa kriterijumom zaustavljanja od 100 iteracija. Tabela 4.4 sadrži rezultate Gaus-VNS algoritma za instance velikih dimenzija sa vremenskim kriterijumom zaustavljanja od 1000 sekundi.

Tabela 4.1: Rezultati Gaus-VNS algoritma na malim instancama

Inst.	LB	opt	$Gaus-VNS_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.015	0.017	0.000	0.000
ftsp_5_10_1	10	15	opt	0.015	0.015	0.000	0.000
ftsp_5_10_2	12	12	opt	0.011	0.014	0.000	0.000
ftsp_5_10_3	14	14	opt	0.015	0.016	0.000	0.000
ftsp_5_10_4	10	10	opt	0.011	0.012	0.000	0.000
ftsp_5_10_5	6	13	opt	0.014	0.015	0.000	0.000
ftsp_5_10_6	12	13	opt	0.014	0.015	0.000	0.000
ftsp_5_10_7	8	8	opt	0.010	0.012	0.000	0.000
ftsp_5_10_8	11	11	opt	0.013	0.014	0.000	0.000
ftsp_5_10_9	26	26	opt	0.021	0.021	0.000	0.000
ftsp_10_10_0	12	15	opt	0.007	0.008	0.000	0.000
ftsp_10_10_1	12	12	opt	0.009	0.009	0.000	0.000
ftsp_10_10_2	5	14	opt	0.006	0.009	0.000	0.000
ftsp_10_10_3	19	19	opt	0.009	0.015	0.000	0.000
ftsp_10_10_4	17	17	opt	0.012	0.013	0.000	0.000
ftsp_10_10_5	10	15	opt	0.009	0.011	0.000	0.000
ftsp_10_10_6	3	15	opt	0.007	0.008	0.000	0.000
ftsp_10_10_7	11	14	opt	0.012	0.013	0.000	0.000
ftsp_10_10_8	11	14	opt	0.011	0.013	0.000	0.000
ftsp_10_10_9	2	14	opt	0.008	0.009	0.000	0.000

U Tabelama 4.5 i 4.6 dati su rezultati Gaus-VNS-VND algoritma za instance malih i srednjih dimenzija sa kriterijumom zaustavljanja od 100 iteracija, dok u Tabeli 4.7 su predstavljeni rezultati dobijeni primenom istog algoritma na velike instance nakon 1000 sekundi.

Pregled rezultata Gaus-VNS-LS1 algoritma dat je na sličan način u Tabelama

Tabela 4.2: Rezultati Gaus-VNS algoritma na srednjim instancama

Inst.	LB	opt	$Gaus-VNS_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	5.896	5.975	0.000	0.000
ftsp_10_50_1	17	81	opt	4.831	4.899	0.000	0.000
ftsp_10_50_2	40	42	opt	3.428	3.760	0.000	0.000
ftsp_10_50_3	31	57	opt	4.257	4.314	0.000	0.000
ftsp_10_50_4	17	20	opt	2.107	2.182	0.000	0.000
ftsp_10_50_5	32	32	opt	2.213	3.113	0.313	1.358
ftsp_10_50_6	36	36	opt	3.486	3.582	0.000	0.000
ftsp_10_50_7	85	85	opt	6.236	6.321	0.000	0.000
ftsp_10_50_8	85	85	opt	5.060	5.132	0.000	0.000
ftsp_10_50_9	44	44	opt	3.600	3.684	0.000	0.000
ftsp_30_100_0	199	199	opt	78.051	79.682	0.000	0.000
ftsp_30_100_1	45	68	opt	29.585	30.211	0.000	0.000
ftsp_30_100_2	29	108	opt	55.522	56.661	0.000	0.000
ftsp_30_100_3	108	108	opt	50.053	54.346	0.000	0.000
ftsp_30_100_4	31	67	opt	29.157	29.809	0.000	0.000
ftsp_30_100_5	163	163	opt	64.701	66.052	0.000	0.000
ftsp_30_100_6	30	50	opt	8.739	8.919	0.000	0.000
ftsp_30_100_7	177	177	opt	76.274	77.841	0.000	0.000
ftsp_30_100_8	45	96	opt	47.032	48.008	0.000	0.000
ftsp_30_100_9	270	270	opt	98.661	100.680	0.000	0.000

Tabela 4.3: Rezultati Gaus-VNS algoritma na velikim instancama

Inst.	LB	opt	$Gaus-VNS_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	$LBopt$	9256.248	10754.811	0.000	0.000
ftsp_30_500_1	718	-	$LBopt$	34432.246	35435.953	0.000	0.000
ftsp_30_500_2	269	-	$LBopt$	14530.049	15190.678	0.000	0.000
ftsp_30_500_3	117	-	126	7100.111	10277.810	0.714	0.343
ftsp_30_500_4	528	-	$LBopt$	26234.841	26812.561	0.000	0.000
ftsp_50_200_0	73	-	567	1402.965	1427.285	0.000	0.000
ftsp_50_200_1	89	116	opt	351.347	356.683	0.000	0.000
ftsp_50_200_2	423	423	opt	1070.945	1089.185	0.000	0.000
ftsp_50_200_3	102	109	opt	358.732	363.818	0.000	0.000
ftsp_50_200_4	59	344	opt	854.417	869.325	0.000	0.000
ftsp_100_500_0	208	-	578	22421.654	22905.427	0.000	0.000
ftsp_100_500_1	197	-	224	9983.966	10373.470	0.246	1.068
ftsp_100_500_2	300	-	$LBopt$	12856.352	14798.843	0.033	0.100
ftsp_100_500_3	794	-	830	31655.646	32335.513	0.000	0.000
ftsp_100_500_4	154	-	224	10494.780	10706.632	0.000	0.000

Tabela 4.4: Rezultati Gaus-VNS algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	$Gaus-VNS_{naj}$	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	*LBopt*	1.198	1.183
ftsp_30_500_1	718	-	*LBopt*	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	0.372	0.483
ftsp_30_500_3	117	-	128	2.500	2.723
ftsp_30_500_4	528	-	*LBopt*	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	0.000	0.000
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	224	2.946	2.424
ftsp_100_500_2	300	-	*LBopt*	0.350	0.267
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	0.000	0.000

4.8-4.10, gde je za sve instance kriterijum zaustavljanja bio 100 iteracija. U Tabeli 4.11 predstavljeni su rezultati ovog algoritma na velikim instancama sa kriterijumom zaustavljanja od 1000 sekundi.

Značenje svake od kolona u svim tabelama u ovom odeljku je isti kao i u odeljku sa eksperimentalnim rezultatima u prethodnoj glavi.

Rezultati dobijeni korišćenjem sva tri algoritma, Gaus-VNS, Gaus-VNS-VND i Gaus-VNS-LS1 za male instance su očekivani: dobijena su sva optimalna rešenja, u svih 20 izvršavanja i to jako brzo.

Efikasnost algoritama opisanih u ovog glavi se najrealnije može prikazati ukoliko ih uporedimo sa algoritmima iz prethodne glave koji imaju istu proceduru lokalne pretrage. Tako, rezultate Gaus-VNS algoritma poredimo sa VNS algoritmom iz odeljka 3.4. Za instance srednjih dimenzija, poredeći rezultate VNS-a iz Tabele 3.5 i Gaus-VNS-a iz Tabele 4.2, oba algoritma dostižu sve optimalne vrednosti. Kvalitet dobijenih rešenja u 20 izvršavanja po instanci je bolji u slučaju Gaus-VNS-a koji dostiže istu optimalnu vrednost za 19 od 20 instanci, dok je VNS dostigao za 17 od 20 instanci.

Iz Tabele 4.3 se vidi da što se tiče instanci velikih dimenzija sa kriterijumom zaustavljanja od 100 iteracija i tu je Gaus-VNS pokazao poboljšanje po nekoliko kriterijuma u odnosu na VNS. Za četiri instance ($ftsp\_30\_500\_0$, $ftsp\_30\_500\_3$,

Tabela 4.5: Rezultati Gaus-VNS-VND algoritma na malim instancama

Inst.	LB	opt	$Gaus-VNS-VND_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.016	0.017	0.000	0.000
ftsp_5_10_1	10	15	opt	0.016	0.017	0.000	0.000
ftsp_5_10_2	12	12	opt	0.014	0.015	0.000	0.000
ftsp_5_10_3	14	14	opt	0.015	0.017	0.000	0.000
ftsp_5_10_4	10	10	opt	0.012	0.012	0.000	0.000
ftsp_5_10_5	6	13	opt	0.014	0.014	0.000	0.000
ftsp_5_10_6	12	13	opt	0.013	0.017	0.000	0.000
ftsp_5_10_7	8	8	opt	0.009	0.012	0.000	0.000
ftsp_5_10_8	11	11	opt	0.012	0.014	0.000	0.000
ftsp_5_10_9	26	26	opt	0.017	0.021	0.000	0.000
ftsp_10_10_0	12	15	opt	0.008	0.009	0.000	0.000
ftsp_10_10_1	12	12	opt	0.007	0.008	0.000	0.000
ftsp_10_10_2	5	14	opt	0.007	0.007	0.000	0.000
ftsp_10_10_3	19	19	opt	0.014	0.015	0.000	0.000
ftsp_10_10_4	17	17	opt	0.012	0.013	0.000	0.000
ftsp_10_10_5	10	15	opt	0.010	0.012	0.000	0.000
ftsp_10_10_6	3	15	opt	0.008	0.008	0.000	0.000
ftsp_10_10_7	11	14	opt	0.010	0.012	0.000	0.000
ftsp_10_10_8	11	14	opt	0.009	0.011	0.000	0.000
ftsp_10_10_9	2	14	opt	0.008	0.009	0.000	0.000

$ftsp$_100_500_1 i $ftsp$_100_500_2) Gaus-VNS dostiže bolja rešenja od VNS-a. Rešenje dobijeno za instancu $ftsp$_30_500_3 je čak manje od rešenja dobijenih sa sve tri modifikacije VNS-a iz glave 3. Poredeći kvalitet rešenja dobijenih u svih 20 izvršavanja po instanci, Gaus-VNS se i tu pokazao boljim od VNS-a, gde je isto rešenje dobijao u svih 20 poziva za 12 od 15 instanci, dok je kod VNS-a to bio slučaj sa 7 od 15 instanci.

Sa promenjenim kriterijumom zaustavljanja na ukupno vreme izvršavanja od 1000 sekundi, Gaus-VNS se i u tom slučaju pokazao boljim od VNS-a dostigavši manje vrednosti za 4 od ukupno 15 instanci.

Gaus-VNS-VND algoritam, za instance srednjih dimenzija, dostiže sva optimalna rešenja u svih 20 izvršavanja, kao i VNS-VND algoritam iz odeljka 3.4.2. U slučaju instanci velikih dimenzija, oba algoritma za 1000 sekundi dostižu iste vrednosti. Slično je i sa algoritmima Gaus-VNS-LS1 i VNS-LS1 iz odeljka 3.4.3: za instance srednjih i velikih dimenzija dostižu se iste vrednosti, u relativno sličnom vremenu sa bliskim kvalitetom rešenja.

Tabela 4.6: Rezultati Gaus-VNS-VND algoritma na srednjim instancama

Inst.	LB	opt	$Gaus-VNS-VND_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	5.619	5.697	0.000	0.000
ftsp_10_50_1	17	81	opt	4.626	4.687	0.000	0.000
ftsp_10_50_2	40	42	opt	3.342	3.550	0.000	0.000
ftsp_10_50_3	31	57	opt	4.078	4.136	0.000	0.000
ftsp_10_50_4	17	20	opt	2.024	2.072	0.000	0.000
ftsp_10_50_5	32	32	opt	2.773	2.842	0.000	0.000
ftsp_10_50_6	36	36	opt	3.229	3.404	0.000	0.000
ftsp_10_50_7	85	85	opt	6.021	6.103	0.000	0.000
ftsp_10_50_8	85	85	opt	4.896	4.965	0.000	0.000
ftsp_10_50_9	44	44	opt	3.361	3.447	0.000	0.000
ftsp_30_100_0	199	199	opt	73.495	75.041	0.000	0.000
ftsp_30_100_1	45	68	opt	27.237	27.820	0.000	0.000
ftsp_30_100_2	29	108	opt	43.768	44.649	0.000	0.000
ftsp_30_100_3	108	108	opt	46.230	52.853	0.000	0.000
ftsp_30_100_4	31	67	opt	28.703	29.344	0.000	0.000
ftsp_30_100_5	163	163	opt	62.379	63.645	0.000	0.000
ftsp_30_100_6	30	50	opt	8.331	8.501	0.000	0.000
ftsp_30_100_7	177	177	opt	70.811	72.246	0.000	0.000
ftsp_30_100_8	45	96	opt	40.986	41.850	0.000	0.000
ftsp_30_100_9	270	270	opt	92.165	94.019	0.000	0.000

Tabela 4.7: Rezultati Gaus-VNS-VND algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	$Gaus-VNS-VND_{naj}$	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	$LBopt$	1.078	1.409
ftsp_30_500_1	718	-	$LBopt$	0.000	0.000
ftsp_30_500_2	269	-	$LBopt$	0.446	0.904
ftsp_30_500_3	117	-	127	1.969	1.629
ftsp_30_500_4	528	-	$LBopt$	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	0.000	0.000
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	224	4.107	2.954
ftsp_100_500_2	300	-	$LBopt$	0.283	0.217
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	0.000	0.000

Tabela 4.8: Rezultati Gaus-VNS-LS1 algoritma na malim instancama

Inst.	LB	opt	$Gaus-VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_5_10_0	14	14	opt	0.006	0.007	0.000	0.000
ftsp_5_10_1	10	15	opt	0.007	0.007	0.000	0.000
ftsp_5_10_2	12	12	opt	0.002	0.006	0.000	0.000
ftsp_5_10_3	14	14	opt	0.007	0.007	0.000	0.000
ftsp_5_10_4	10	10	opt	0.007	0.009	0.000	0.000
ftsp_5_10_5	6	13	opt	0.002	0.007	0.000	0.000
ftsp_5_10_6	12	13	opt	0.005	0.007	0.000	0.000
ftsp_5_10_7	8	8	opt	0.002	0.004	0.000	0.000
ftsp_5_10_8	11	11	opt	0.004	0.006	0.000	0.000
ftsp_5_10_9	26	26	opt	0.006	0.007	0.000	0.000
ftsp_10_10_0	12	15	opt	0.004	0.004	0.000	0.000
ftsp_10_10_1	12	12	opt	0.003	0.004	0.000	0.000
ftsp_10_10_2	5	14	opt	0.004	0.004	0.000	0.000
ftsp_10_10_3	19	19	opt	0.005	0.006	0.000	0.000
ftsp_10_10_4	17	17	opt	0.004	0.007	0.000	0.000
ftsp_10_10_5	10	15	opt	0.003	0.005	0.000	0.000
ftsp_10_10_6	3	15	opt	0.004	0.005	0.000	0.000
ftsp_10_10_7	11	14	opt	0.004	0.007	0.000	0.000
ftsp_10_10_8	11	14	opt	0.005	0.005	0.000	0.000
ftsp_10_10_9	2	14	opt	0.004	0.004	0.000	0.000

Tabela 4.9: Rezultati Gaus-VNS-LS1 algoritma na srednjim instancama

Inst.	LB	opt	$Gaus-VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_10_50_0	26	92	opt	0.227	0.230	0.000	0.000
ftsp_10_50_1	17	81	opt	0.201	0.205	0.000	0.000
ftsp_10_50_2	40	42	opt	0.152	0.185	0.000	0.000
ftsp_10_50_3	31	57	opt	0.181	0.184	0.000	0.000
ftsp_10_50_4	17	20	opt	0.098	0.106	0.000	0.000
ftsp_10_50_5	32	32	opt	0.096	0.194	1.875	1.503
ftsp_10_50_6	36	36	opt	0.155	0.182	0.000	0.000
ftsp_10_50_7	85	85	opt	0.264	0.268	0.000	0.000
ftsp_10_50_8	85	85	opt	0.203	0.207	0.000	0.000
ftsp_10_50_9	44	44	opt	0.150	0.181	0.000	0.000
ftsp_30_100_0	199	199	opt	1.525	1.562	0.000	0.000
ftsp_30_100_1	45	68	opt	0.583	0.603	0.000	0.000
ftsp_30_100_2	29	108	opt	1.208	1.250	0.000	0.000
ftsp_30_100_3	108	108	opt	1.278	1.348	0.000	0.000
ftsp_30_100_4	31	67	opt	0.656	0.695	0.000	0.000
ftsp_30_100_5	163	163	opt	1.301	1.329	0.000	0.000
ftsp_30_100_6	30	50	opt	0.172	0.173	0.000	0.000
ftsp_30_100_7	177	177	opt	1.488	1.522	0.000	0.000
ftsp_30_100_8	45	96	opt	0.836	0.914	0.000	0.000
ftsp_30_100_9	270	270	opt	1.939	1.978	0.000	0.000

Tabela 4.10: Rezultati Gaus-VNS-LS1 algoritma na velikim instancama

Inst.	LB	opt	$Gaus-VNS-LS1_{naj}$	t (sec)	t_{tot} (sec)	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	*LBopt*	44.611	68.336	0.299	0.400
ftsp_30_500_1	718	-	*LBopt*	182.750	186.766	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	84.679	89.610	0.000	0.000
ftsp_30_500_3	117	-	127	33.591	65.497	1.102	0.794
ftsp_30_500_4	528	-	*LBopt*	139.973	142.988	0.000	0.000
ftsp_50_200_0	73	-	567	16.795	17.087	0.000	0.000
ftsp_50_200_1	89	116	opt	4.408	4.463	0.000	0.000
ftsp_50_200_2	423	423	opt	12.625	12.841	0.000	0.000
ftsp_50_200_3	102	109	opt	4.026	5.967	0.505	1.136
ftsp_50_200_4	59	344	opt	10.197	10.378	0.000	0.000
ftsp_100_500_0	208	-	578	105.646	107.932	0.000	0.000
ftsp_100_500_1	197	-	224	50.340	59.714	1.696	1.946
ftsp_100_500_2	300	-	*LBopt*	71.226	98.522	0.283	0.263
ftsp_100_500_3	794	-	830	148.025	151.218	0.000	0.000
ftsp_100_500_4	154	-	224	50.689	51.892	0.000	0.000

Tabela 4.11: Rezultati Gaus-VNS-LS1 algoritma na velikim instancama sa $t_{tot} = 1000s$

Inst.	LB	opt	$Gaus-VNS-LS1_{naj}$	avnsp (%)	σ (%)
ftsp_30_500_0	167	-	*LBopt*	0.000	0.000
ftsp_30_500_1	718	-	*LBopt*	0.000	0.000
ftsp_30_500_2	269	-	*LBopt*	0.000	0.000
ftsp_30_500_3	117	-	127	0.236	0.360
ftsp_30_500_4	528	-	*LBopt*	0.000	0.000
ftsp_50_200_0	73	-	567	0.000	0.000
ftsp_50_200_1	89	116	opt	0.000	0.000
ftsp_50_200_2	423	423	opt	0.000	0.000
ftsp_50_200_3	102	109	opt	0.000	0.000
ftsp_50_200_4	59	344	opt	0.000	0.000
ftsp_100_500_0	208	-	578	0.000	0.000
ftsp_100_500_1	197	-	224	0.000	0.000
ftsp_100_500_2	300	-	*LBopt*	0.000	0.000
ftsp_100_500_3	794	-	830	0.000	0.000
ftsp_100_500_4	154	-	224	0.000	0.000

Glava 5

Zaključak

U ovom radu su prikazane modifikacije metode promenljivih okolina i njihove primene za problem rasporedjivanja prenosa datoteka. Razvijena je i nova modifikacija metode promenljihih okolina za rešavanje problema kontinualne optimizacije, Gaus-VNS (Glava 2). U okviru VNS metaheuristike diverzifikacija rešenja je ključan element koji bitno utiče na efikasnost metode. Diverzifikacija ostvarena korišćenjem višedimenzione Gausove raspodele se pokazala uspešnijom u odnosu na prethodno korišćene metode. Lokalno poboljšavanje rešenja postignuto je korišćenjem efikasnih standardnih algoritama. U većini slučajeva Gaus-VNS je nadmašio druge metaheuristike za rešavanje problema kontinualne optimizacije poznate u literaturi. Ovakva varijanta VNS-a, sa prezentovanim rezultatima, prvi put je predložena u radu u vrhunskom međunarodnom časopisu E. Carrizosa, M. Dražić, Z. Dražić, N. Mladenović [15].

Drugi deo disertacije je posvećen rešavanju diskretnog problema raspoređivanja prenosa datoteka. Dati problem je po prvi put uspešno modeliran kao celobrojni linearni program (Odeljak 3.3). Ovo je postignuto čak na tri različita načina, korišćenjem dve nove reformulacije problema. Ekvivalencija svih modela i reformulacija sa polaznim problemom je matematički dokazana. Korišćenjem predloženih modela za sve test primere manjih i srednjih dimenzija pronađena su optimalna rešenja. Deo rezultata prikazan je u radu koji je prihvaćen za publikovanje u međunarodnom časopisu Z. Dražić, A. Savić, V. Filipović [28].

Za dobijanje rešenja na test primerima velikih dimenzija koji su bili van domašaja egzaktnih metoda, razvijena je metoda promenljivih okolina (Odeljak 3.4). Originalni način kodiranja problema korišćenjem "permutacijske" reprezentacije rešenja

je omogućilo korišćenje standardnog sistema okolina. Tri različite varijante lokalnog pretraživanja su se pokazale kao uspešne u dostizanju lokalnih i globalnih optimuma. Deo prezentovanih rezultata objavljen je u samostalnom radu u naučnom časopisu Z. Dražić [27].

U cilju objedinjavanja prethodnih pristupa i radi poređenja, primenjena je Gaus-VNS kao metoda kontinualne optimizacije na dati problem diskretne optimizacije (Glava 4). Uspešno je realizovano preslikavanje iz neprebrojivog skupa svih dopustivih rešenja problema kontinualne optimizavije u konačan skup dopustivih rešenja problema diskretne optimizacije. Zahvaljujući tome razvijena je težinska reprezentacija rešenja problema raspoređivanja prenosa datoteka koja je omogućila primenu sve tri metode lokalnog pretraživanja realizovane za metodu promenljivih okolina.

Neki od pravaca daljeg razvoja i unapređenja dobijenih rezultata mogu biti:

- Automatizacija određivanja parametara višedimenzione Gausove raspodele;
- Primena Gaus-VNS algoritma na probleme kontinualne optimizacije sa ograničenjima;
- Upotreba drugih raspodela slučajnih brojeva;
- Rešavanje FTSP uz pomoć neke druge metaheuristike ili hibridizacija;
- Razvijanje egzaktne metode za FTSP na osnovu predloženih ILP formulacija;
- Modifikacija opisanih metaheuristika za rešavanje sličnih problema diskretne optimizacije;
- Paralelizacija i izvršavanje na višeprocesorkim računarima.

5.1 Naučni doprinos rada

Najvažniji naučni doprinosi dobijeni istraživanjem prikazanim u ovom radu su:

- Tri nove formulacije celobrojnog linearnog programiranja za problem raspoređivanja prenosa datoteka sa dokazom njihove korektnosti, kao i dve nove reformulacije samog problema;
- Razvoj sistema okolina i metoda lokalnog pretraživanja u okviru metode promenljivih okolina za problem raspoređivanja prenosa datoteka;

- Modifikacije metode promenljivih okolina korišćenjem Gausove raspodele slučajnih brojeva koju je moguće primeniti i na neograničene oblasti;
- Primena modifikacije metode promenljivih okolina, korišćenjem Gausove raspodele slučajnih brojeva, za rešavanje problema raspoređivanja prenosa datoteka;
- Efektivno smanjenje broja parametara metode promenljivih okolina pri rešavanju problema kontinualne optimizacije;

Rešenja prikazana u ovom radu uspešno se mogu primeniti na rešavanje opisanih i sličnih problema. Naučno istraživanje opisano u ovom radu daje doprinos oblastima kontinualne i kombinatorne optimizacije, kao i metodi promenljivih okolina i problemima raspoređivanja. Deo dobijenih rezultata je već objavljen u međunarodnim časopisima i časopisima od nacionalnog značaja, dok su ostali delovi u pripremi za objavljivanje ili u fazi recenzije.

Literatura

[1] J. H. Ahrens and U. Dieter. "Efficient table-free sampling methods for the exponential, Cauchy, and normal distributions". *Communications of the ACM* 31(11) (1988), pp. 1330–1337.

[2] M. K. Akbari, S. M. Hosseini-nejad, and M. Kalantari. "A Neural Network Realization of File Transfer Scheduling". *The CSI Journal of Computer Science and Engineering* 2 (2004), pp. 19–29.

[3] E. Anderson, J. Hall, J. Hartline, M. Hobbes, A. Karlin, J. Saia, R. Swaminathan, and J. Wilkes. "Algorithms for data migration". *Algorithmica* 57(2) (2010), pp. 349–380.

[4] R. Battiti and G. Tecchiolli. "The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization". *Annals of Operations Research* 63(2) (1996), pp. 151–188.

[5] D Beaslev, D. R. Bull, and R. R. Martin. "An overview of genetic algorithms: Part 2, research topics". *University computing* 15(4) (1993).

[6] D. Beasley, R. Martin, and D. Bull. "An overview of genetic algorithms: Part 1. Fundamentals". *University computing* 15 (1993), pp. 58–58.

[7] J. E. Beasley. "Lagrangian relaxation" in *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc. 1993, pp. 243–303.

[8] J. D. Beltrán, J. E. Calderón, R. J. Cabrera, J. A. Moreno-Pérez, and J. M. Moreno-Vega. "GRASP-VNS hybrid for the Strip Packing Problem." *Hybrid metaheuristics* 2004 (2004), pp. 79–90.

[9] M. Bierlaire, M. Thémans, and N. Zufferey. "A heuristic for nonlinear global optimization". *INFORMS Journal on Computing* 22(1) (2010), pp. 59–70.

[10] Ş. İ. Birbil and S.-C. Fang. "An electromagnetism-like mechanism for global optimization". *Journal of global optimization* 25(3) (2003), pp. 263–282.

[11] C. Blum and A. Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". *ACM Computing Surveys (CSUR)* 35(3) (2003), pp. 268–308.

[12] J Brimberg, P Hansen, and N Mladenovic. "Continuous Optimization by Variable NeighborhoodSearch–A Chapter for Encyclopedia ofOperations Research and Management Science (EORMS)" (2013).

[13] J. Brimberg and N. Mladenovic. "A variable neighbourhood algorithm for solving the continuous location-allocation problem". *Cahiers du GERAD* (1995).

[14] J. Brimberg, P. Hansen, N. Mladenović, and E. D. Taillard. "Improvement and comparison of heuristics for solving the uncapacitated multisource Weber problem". *Operations Research* 48(3) (2000), pp. 444–460.

[15] E. Carrizosa, M. Dražić, Z. Dražić, and N. Mladenović. "Gaussian variable neighborhood search for continuous optimization". *Computers & Operations Research* 39(9) (2012), pp. 2206–2213.

[16] P.-C. Chang, S.-H. Chen, and C.-Y. Fan. "A hybrid electromagnetism-like algorithm for single machine scheduling problem". *Expert Systems with Applications* 36(2) (2009), pp. 1259–1267.

[17] R. Chelouah and P. Siarry. "Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions". *European Journal of Operational Research* 148(2) (2003), pp. 335–348.

[18] E. G. Coffman Jr, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. "Scheduling file transfers". *SIAM Journal on Computing* 14(3) (1985), pp. 744–780.

[19] *CPLEX solver, IBM company, www.ibm.com/software/integration/optimization/cplex-optimizer.*

[20] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenović. "Cooperative parallel variable neighborhood search for the p-median". *Journal of Heuristics* 10(3) (2004), pp. 293–314.

[21] D. Cvetković, M. Čangalović, Đ. Dugošija, V. Kovačević-Vujčić, S. Simić, and J. Vuleta. *Kombinatorna optimizacija: Matematička teorija i algoritmi.* Društvo operacionih istraživača Jugoslavije, 1996.

[22] D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke. "A hybrid scatter search/electromagnetism meta-heuristic for project scheduling". *European Journal of Operational Research* 169(2) (2006), pp. 638–653.

[23] M. Dorigo. *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*. Vol. 4150. Springer-Verlag New York Incorporated, 2006.

[24] M. Dorigo, V. Maniezzo, and A. Colorni. "Ant system: optimization by a colony of cooperating agents". *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 26(1) (1996), pp. 29–41.

[25] M. Drazić, V. Kovacevic-Vujcić, M. Cangalović, and N. Mladenović. "Glob—a new VNS-based software for global optimization" in *Global Optimization*. Springer, 2006, pp. 135–154.

[26] M. Dražić, C. Lavor, N. Maculan, and N. Mladenović. "A continuous variable neighborhood search heuristic for finding the three-dimensional structure of a molecule". *European Journal of Operational Research* 185(3) (2008), pp. 1265–1273.

[27] Z. Dražić. "Variable Neighborhood Search for the File Transfer Scheduling Problem". *Serdica Journal of Computing* 6(3) (2012), 333p–348p.

[28] Z. Dražić, A. Savić, and V. Filipović. "An integer linear formulation for the file transfer scheduling problem". *TOP, DOI:10.1007/s11750-013-0312-x* (2014).

[29] J. Dréo and P. Siarry. "Hybrid continuous interacting ant colony aimed at enhanced global optimization". *Algorithmic Operations Research* 2(1) (2007).

[30] S.-K. S. Fan, Y.-C. Liang, and E. Zahara. "A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search". *Computers & Industrial Engineering* 50(4) (2006), pp. 401–425.

[31] V Filipović. "Operatori selekcije i migracije i web servisi kod paralelnih evolutivnih algoritama, doktorska disertacija" (2006).

[32] M. L. Fisher. "The Lagrangian relaxation method for solving integer programming problems". *Management science* 50(12 supplement) (2004), pp. 1861–1871.

[33] G. S. Fishman. *Monte Carlo: concepts, algorithms, and applications*. Vol. 1196. Springer New York, 1996.

[34] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients". *The computer journal* 7(2) (1964), pp. 149–154.

[35] R. Fletcher and M. J. Powell. "A rapidly convergent descent method for minimization". *The Computer Journal* 6(2) (1963), pp. 163–168.

[36] F. García-López, B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega. "The parallel variable neighborhood search for the p-median problem". *Journal of Heuristics* 8(3) (2002), pp. 375–388.

[37] M. Gendreau and J.-Y. Potvin. *Handbook of metaheuristics*. Vol. 146. Springer, 2010.

[38] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". *SIAM review* 47(1) (2005), pp. 99–131.

[39] F. Glover. "Tabu search—part I". *ORSA Journal on computing* 1(3) (1989), pp. 190–206.

[40] F. Glover. "Tabu search—part II". *ORSA Journal on computing* 2(1) (1990), pp. 4–32.

[41] F. Glover, M. Laguna, et al. *Tabu search*. Vol. 22. Springer, 1997.

[42] F. Glover, M. Laguna, E Taillard, and D De Werra. *Tabu search*. Baltzer Basel, 1993.

[43] D. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989" ().

[44] P. Hansen and N Mladenovic. "An introduction to variable neighborhood search" in *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, (S. Voss et al. eds.)* 1999, pp. 433–458.

[45] P Hansen, B Jaumard, N Mladenovic, and A Parreira. "Variable neighborhood search for weighted maximum satisfiability problem". *Les Cahiers du GERAD* (2000), pp. 2000–62.

[46] P. Hansen and N. Mladenović. "Variable neighborhood search: Principles and applications". *European journal of operational research* 130(3) (2001), pp. 449–467.

[47] P. Hansen and N. Mladenović. "Variable neighborhood search" in *Handbook of metaheuristics*. Springer, 2003, pp. 145–184.

[48] P. Hansen, N. Mladenovic, and Q. Groupe d'études et de recherche en analyse des décisions Montréal. "A tutorial on variable neighborhood search" (2003).

[49] P. Hansen, N. Mladenović, and J. A. M. Pérez. "Variable neighbourhood search: methods and applications". *4OR* 6(4) (2008), pp. 319–360.

[50] P. Hansen, N. Mladenović, and D. Perez-Britos. "Variable neighborhood decomposition search". *Journal of Heuristics* 7(4) (2001), pp. 335–350.

[51] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez. "Variable neighborhood search" in *Handbook of Metaheuristics*. Springer, 2010, pp. 61–86.

[52] G. Harm and P. Hentenryck. "A multistart variable neighborhood search for uncapacitated facility location" in *Proceedings of MIC*. 2005.

[53] J. T. Havill and W. Mao. "Greedy On-Line File Transfer Routing." in *Euro-PDS*. 1997, pp. 225–230.

[54] A.-R. Hedar and M. Fukushima. "Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization". *Optimization Methods and Software* 17(5) (2002), pp. 891–912.

[55] A.-R. Hedar and M. Fukushima. "Tabu search directed by direct search methods for nonlinear global optimization". *European Journal of Operational Research* 170(2) (2006), pp. 329–349.

[56] F. Herrera, M. Lozano, and D. Molina. "Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies". *European Journal of Operational Research* 169(2) (2006), pp. 450–476.

[57] D. Higuero, J. M. Tirado, F. Isaila, and J. Carretero. "Enhancing file transfer scheduling and server utilization in data distribution infrastructures" in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*. IEEE. 2012, pp. 431–438.

[58] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[59] R. Hooke and T. A. Jeeves. ""Direct Search"Solution of Numerical and Statistical Problems". *Journal of the ACM (JACM)* 8(2) (1961), pp. 212–229.

[60] R. Horst, P. M. Pardalos, and H. E. Romeijn. *Handbook of global optimization*. Vol. 2. Springer, 2002.

[61] *http://tracer.lcc.uma.es/problems/ackley/ackley.html.*

[62] *http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page295.htm.*

[63] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi. "Optimization by simmulated annealing". *science* 220(4598) (1983), pp. 671–680.

[64] G. A. Kochenberger et al. *Handbook in Metaheuristics*. Springer, 2003.

[65] V Kovacevic-Vujcic, M Cangalovic, M Drazic, and N Mladenovic. "VNS-based heuristics for continuous global optimization". *Le Thi Hoai An, Pham Dinh Thao (Eds.), Modelling, Computation and Optimization in Information Systems and Management Sciences. Hermes Science Publishing* (2004), pp. 215–222.

[66] S. Krčevinac, M. Čupić, J. Petrić, and I. Nikolić. *Algoritmi i programi iz operacionih istraživanja*. Naučna knjiga, 1989.

[67] C. Lavor and N. Maculan. "A function to test methods applied to global minimization of potential energy of molecules". *Numerical Algorithms* 35(2-4) (2004), pp. 287–300.

[68] L. Liberti and M. Drazic. "Variable neighbourhood search for the global optimization of constrained NLPs" in *Proceedings of GO Workshop, Almeria, Spain*. Vol. 2005. 2005.

[69] P. Lucic and D. Teodorovic. "Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence" in *Preprints of the TRISTAN IV triennial symposium on transportation analysis*. 2001, pp. 441–445.

[70] P. Lucic and D. Teodorovic. "Transportation modeling: an artificial life approach" in *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on*. IEEE. 2002, pp. 216–223.

[71] P. Lučić and D. Teodorović. "Computing with bees: attacking complex transportation engineering problems". *International Journal on Artificial Intelligence Tools* 12(03) (2003), pp. 375–394.

[72] B. Maenhout and M. Vanhoucke. "An electromagnetic meta-heuristic for the nurse scheduling problem". *Journal of Heuristics* 13(4) (2007), pp. 359–385.

[73] W. Mao. "Directed file transfer scheduling" in *ACM 31st Annual Southeast Conference (ACM-SE 1993)*. 1993, pp. 199–203.

[74] W. Mao and R. Simha. "Routing and scheduling file transfers in packetswitched networks". *Journal of Computing and Information* 1(1) (1994), pp. 559–574.

[75] N. Mladenović. "A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization applications" in *In Optimization Days, page 112, Montreal*. 1995.

[76] N. Mladenović, M Drazić, M. Cangalović, and V Kovacevic-Vujcic. "Variable neighborhood search in global optimization" in *In: Mladenović, N., Dugosija, D. (Eds.), Proceedings of XXX Symposium on Operations Research, Herceg Novi.* 2003, pp. 327–330.

[77] N. Mladenović and P. Hansen. "Variable neighborhood search". *Computers & Operations Research* 24(11) (1997), pp. 1097–1100.

[78] N. Mladenović, J Petrović, V Kovačević-Vujčić, and M Čangalović. "Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search". *European Journal of Operational Research* 151(2) (2003), pp. 389–399.

[79] N. Mladenović, M. Dražić, V. Kovačevic-Vujčić, and M. Čangalović. "General variable neighborhood search for the continuous optimization". *European Journal of Operational Research* 191(3) (2008), pp. 753–770.

[80] J. A. Nelder and R. Mead. "A simplex method for function minimization". *The computer journal* 7(4) (1965), pp. 308–313.

[81] G. Optimization. "Gurobi optimizer reference manual". *URL: http://www. gurobi. com* (2012).

[82] I. H. Osman and G. Laporte. "Metaheuristics: A bibliography". *Annals of Operations Research* 63(5) (1996), pp. 511–623.

[83] P. M. Pardalos and J. B. Rosen. *Constrained global optimization: algorithms and applications.* Springer-Verlag New York, Inc., 1987.

[84] J. A. M. Pérez, P. Hansen, and N. Mladenovic. *Parallel variable neighborhood search.* Groupe d'études et de recherche en analyse des décisions, 2004.

[85] J. Pintér. "Continuous global optimization software: A brief review". *Optima* 52 (1996), pp. 1–8.

[86] C. Ribeiro and P. Hansen. *Essays and surveys in metaheuristics.* Kluwer Academic Publishers, 2002.

[87] C. C. Ribeiro, E. Uchoa, and R. F. Werneck. "A hybrid GRASP with perturbations for the Steiner problem in graphs". *INFORMS Journal on Computing* 14(3) (2002), pp. 228–246.

[88] P. I. Rivera-Vega, R. Varadarajan, and S. B. Navathe. "Scheduling file transfers in fully connected networks". *Networks* 22(6) (1992), pp. 563–588.

[89] H. Rosenbrock. "An automatic method for finding the greatest or least value of a function". *The Computer Journal* 3(3) (1960), pp. 175–184.

[90] B. Suman and P. Kumar. "A survey of simulated annealing as a tool for single and multiobjective optimization". *Journal of the operational research society* 57(10) (2005), pp. 1143–1160.

[91] M. D. Toksarı and E. Güner. "Solving the unconstrained optimization problem by a variable neighborhood search". *Journal of Mathematical Analysis and Applications* 328(2) (2007), pp. 1178–1187.

[92] A. Torn and A. Zilinskas. *Global optimization*. Springer-Verlag New York, Inc., 1989.

[93] R. Varadarajan and P. I. Rivera-Vega. "An efficient approximation algorithm for the file redistribution scheduling problem in fully connected networks" in *Problem in Fully Connected Networks." www. cise. ufl. edu/tech-reports/tech-reports/tr92-abstracts. shtml*. Citeseer. 1992.

[94] Q. H. Zhao, D. Urosević, N. Mladenović, and P. Hansen. "A restarted and modified simplex search for unconstrained optimization". *Computers & Operations Research* 36(12) (2009), pp. 3263–3271.

[95] S. Zlobec and J. J. Petrić. *Nelinearno programiranje*. Naučna knjiga, 1989.

Biografija autora

Zorica Dražić je rođena 28. avgusta 1983. godine u Beogradu. Završila je osnovnu školu "Vuk Stefanović Karadžić" i Petu beogradsku gimnaziju u Beogradu. Diplomirala je 2008. godine na Matematičkom fakultetu u Beogradu, smer Računarstvo i informatika. Iste godine upisala je doktorske studije na istom fakultetu na Katedri za računarstvo i informatiku. Sve ispite na doktorskim studijama je položila sa prosečnom ocenom 10.

Od oktobra 2009. godine radi na Matematičkom fakultetu u Beogradu, na Katedri za numeričku matematiku i optimizaciju, prvo u zvanju saradnika u nastavi, a zatim od 2011. godine u zvanju asistenta. Tokom tog vremena držala je vežbe iz predmeta: Uvod u numeričku matematiku, Numeričke metode, Numerička analiza 1A i 1B, Matematičko programiranje i optimizacija, Matematika za studente hemije.

Angažovana je od 2009. godine na projektima Ministarstva prosvete, nauke i tehnološkog razvoja: 144007 Matematički modeli i metode optimizacije sa primenama (2009-2010) i 174010 Matematički modeli i metode optimizacije velikih sistema (2011-) pod rukovodstvom prof. dr Nenada Mladenovića. Učestvovala je sa radovima na tri međunarodne konferencije: Mini VNS 2012, EURO 2013 i BalCOR 2013. Držala je predavanja na Seminaru za računarstvo i primenjenu matematiku MI SANU i Seminaru Katedre za računarstvo i informatiku Matematičkog fakulteta.

Прилог 1.

Изјава о ауторству

Потписани -а Зорица Дражић

број индекса 2025/2009

Изјављујем

да је докторска дисертација под насловом

Модификације методе променљивих околина и њихове примене за решавање проблема распоређивања преноса датотека

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 10.06.2014.

Зорица Дражић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Зорица Дражић

Број индекса 2025/2009

Студијски програм Информатика

Наслов рада Модификације методе променљивих околина и њихове примене за решавање проблема распоређивања преноса датотека

Ментор проф. др Владимир Филиповић

Потписани/а Зорица Дражић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 10.06.2014.

Зорица Дражић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Модификације методе променљивих околина и њихове примене за решавање проблема распоређивања преноса датотека

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

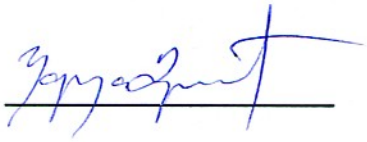
5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 10.06.2014.



1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.