Large Neighborhood Search za Capacitated Vehicle Routing Problem

Mateja Milošević 31/2021
Matija Stanković 207/2021

Projekat u okviru kursa Računarska Inteligencija

Februar 2026

Sadržaj

1 Uvod

Problemi rutiranja vozila predstavljaju jednu od centralnih tema u oblasti kombinatorne optimizacije i operacionih istraživanja, naročito zbog svoje široke primene u logistici, transportu i distribucionim sistemima. Među njima, posebno se ističe *Capacitated Vehicle Routing Problem* (CVRP) zbog svoje praktične relevantnosti i računske složenosti. CVRP spada u klasu NP-teških problema, što znači da za veće instance nije moguće pronaći optimalna rešenja u prihvatljivom vremenu korišćenjem egzaktnih metoda.

Upravo zbog toga, u realnim primenama fokus se retko stavlja na pronalaženje optimalnog rešenja, već na dobijanje dovoljno dobrih rešenja u ograničenom vremenu pomoću različitih heuristika i metaheuristika. Jedna od metaheuristika koja se izdvaja za ovu klasu problema je *Large Neighborhood Search* (LNS), koja kombinuje ideje lokalne pretrage i konstruktivnih heuristika [3].

Ovaj projekat zasniva se na postojećim metodama iz literature, pre svega na radovima Shaw [3] i Pisinger i Røpke [2], pri čemu je fokus rada na reprodukciji, implementaciji i eksperimentalnoj analizi ovih metoda na standardnim benchmark instancama iz CVRPLIB biblioteke [1].

1.1 Koncept Neighborhood (Susedstva)

U optimizacionim algoritmima, pojam *neighborhood* (susedstvo) označava skup rešenja koja se mogu dobiti iz trenutnog rešenja primenom određenih transformacija ili operatora. Lokalni pretraživači koriste ovu ideju kako bi iterativno unapređivali trenutno rešenje, ispitujući njegova susedna rešenja i prelazeći na bolja ukoliko postoje.

Klasične heuristike lokalne pretrage, poput 2-opt i 3-opt algoritama, oslanjaju se na relativno mala susedstva. Takav pristup omogućava brzo ispitivanje susednih rešenja, ali često dovodi do problema zaglavljivanja u lokalnim optimumima, jer algoritam nema mehanizam da napravi veće promene koje bi omogućile prelazak u udaljenije regione prostora pretrage.

Ograničenja malih susedstava motivisala su razvoj metoda koje koriste veće i fleksibilnije strukture susedstva, što je dovelo do nastanka Large Neighborhood Search pristupa.

2 Large Neighborhood Search (Basic)

2.1 Osnovna ideja i opšti tok algoritma

Large Neighborhood Search (LNS) je metaheuristički algoritam za rešavanje kombinatornih optimizacionih problema koji se zasniva na iterativnom unapređivanju trenutnog rešenja korišćenjem velikih susedstava. Za razliku od klasičnih lokalnih heuristika koje razmatraju mala susedstva, LNS omogućava izvođenje značajnijih strukturnih promena nad rešenjem, čime se efikasnije istražuje prostor pretrage i smanjuje verovatnoća zaglavljivanja u lokalnim optimumima [2].

Algoritam se izvršava iterativno i u svakoj iteraciji obuhvata sledeće korake:

- generisanje parcijalnog rešenja iz trenutnog rešenja,
- konstrukciju novog kandidat rešenja,
- evaluaciju i odluku o prihvatanju kandidat rešenja.

Drugim rečima, algoritam u svakoj iteraciji privremeno narušava postojeće rešenje, a zatim pokušava da ga ponovo izgradi na bolji način.

Tokom izvršavanja, algoritam održava tri tipa rešenja:

- **Trenutno rešenje** (current) - predstavlja polaznu tačku za narednu iteraciju,
- **Najbolje rešenje** (best) - odnosno globalno najbolje rešenje pronađeno do tog trenutka,
- **Kandidat rešenje** (candidate) - koje se dobija transformacijom trenutnog rešenja.

2.2 Destroy–Repair operatori

Osnovu Large Neighborhood Search algoritma čine *destroy–repair* operatori. Basic LNS koristi fiksnu kombinaciju destroy i repair operatora tokom celokupnog izvršavanja algoritma.

U fazi **destroy**, uklanja se deo trenutnog rešenja, čime se dobija parcijalno rešenje. Ova faza omogućava algoritmu da napravi velike promene u strukturi rešenja koje nisu moguće korišćenjem standardnih lokalnih operatora.

U fazi **repair**, parcijalno rešenje se ponovo proširuje do validnog kompletnog rešenja. Ova faza se realizuje primenom konstruktivnih heuristika koje ubacuju uklonjene elemente nazad u rešenje na način koji minimizuje pogoršanje funkcije cilja.

Kombinacija destroy i repair operatora implicitno definiše susedstvo trenutnog rešenja i omogućava istraživanje znatno većeg dela prostora rešenja u odnosu na klasične metode lokalne pretrage [3].

2.3 Prihvatanje rešenja i uloga metaheuristika

Nakon generisanja kandidat rešenja, primenjuje se **funkcija prihvatanja (accept funkcija)** koja odlučuje da li kandidat postaje novo trenutno rešenje. Za razliku od strogo pohlepnih pristupa, LNS često dozvoljava prihvatanje i lošijih rešenja. Na ovaj način algoritam izbegava situacije u kojima prerano ostaje zaglavljen u lošem rešenju samo zato što su sva njegova neposredna susedstva lošija.

U praksi se često koriste kriterijumi zasnovani na Simulated Annealing principu, gde se verovatnoća prihvatanja lošijeg rešenja smanjuje tokom vremena, čime se algoritam postepeno usmerava ka stabilnijim i kvalitetnijim rešenjima.

2.4 Prednosti i nedostaci Basic LNS-a

Prednosti:

- **Jednostavna implementacija**
- **Brže izvršavanje po iteraciji**
- **Predvidljivo ponašanje**

Nedostaci:

- **Nema adaptivnosti**
- **Zavisnost od izbora operatora**
- **Manja robusnost**

3 Adaptive Large Neighborhood Search

3.1 Osnovna ideja adaptivnog pristupa

Intuitivno, adaptivni pristup pokušava da odgovori na pitanje koje se strategije u datom trenutku pokazuju kao korisne, umesto da se unapred osloni na jednu fiksnu kombinaciju operatora. Ideja adaptivnog pristupa je da algoritam uči iz prethodnih iteracija i češće koristi one operatore koji su se pokazali uspešnijim u pronalaženju kvalitetnih rešenja [2].

Na ovaj način, ALNS kombinuje više različitih strategija uništavanja i popravke rešenja, pri čemu se njihov uticaj menja u zavisnosti od faze pretrage i karakteristika instance. Time se postiže robusnije ponašanje algoritma u poređenju sa Basic LNS pristupom.

3.2 Mehanizam izbora operatora

Adaptivni algoritam u svakoj iteraciji bira jedan destroy i jedan repair operator na osnovu njihovih trenutnih težina. Operatori sa većim težinama imaju veću verovatnoću izbora, ali nijedan operator nije potpuno isključen, čime se zadržava raznovrsnost pretrage.

Roulette Wheel Selection

Izbor operatora se vrši probabilistički primenom *roulette wheel* metode. Svaki operator dobija verovatnoću proporcionalnu svojoj težini, tako da uspešniji operatori imaju veće šanse da budu izabrani, ali i slabiji operatori zadržavaju mogućnost izbora.

3.3 Evaluacija i bodovanje operatora

Nakon primene izabranih operatora i generisanja kandidat rešenja, njihov učinak se ocenjuje kroz sistem bodovanja.

Sistem bodovanja

U zavisnosti od kvaliteta kandidat rešenja, operatorima se dodeljuju sledeći bodovi:

- ω_1 – pronađeno novo globalno najbolje rešenje,
- ω_2 – kandidat rešenje je bolje od trenutnog rešenja,
- ω_3 – prihvaćeno lošije rešenje,
- ω_4 – kandidat rešenje je odbijeno.

Pri čemu važi $\omega_4 < \omega_3 < \omega_2 < \omega_1$.

Ovakav sistem omogućava da operatori koji doprinose poboljšanjima budu nagrađeni, dok se manje uspešni operatori koriste ređe, ali se nikada ne eliminišu potpuno.

3.4 Ažuriranje težina

Na osnovu dodeljenih bodova, težine operatora se ažuriraju kako bi se odrazila njihova trenutna efikasnost u pretrazi.

Ažuriranje težina

Ažuriranje težina vrši se prema sledećoj formuli:

$$w_{new} = (1 - \lambda) \cdot w_{old} + \lambda \cdot score$$

gde je λ faktor reakcije koji određuje brzinu prilagođavanja težina. Manje vrednosti parametra λ dovode do sporijeg, ali stabilnijeg učenja, dok veće vrednosti omogućavaju bržu reakciju algoritma na promene u ponašanju operatora.

3.5 Prednosti i nedostaci Adaptive LNS u odnosu na Basic LNS

Adaptive Large Neighborhood Search uvodi dodatni nivo fleksibilnosti u odnosu na Basic LNS kroz dinamički izbor destroy i repair operatora. Iako oba algoritma dele isti osnovni princip rada, njihove performanse i ponašanje mogu značajno da se razlikuju u praksi.

Prednosti Adaptive LNS

Glavne prednosti adaptivnog pristupa su:

- automatsko prilagođavanje strategije tokom izvršavanja algoritma,
- smanjena zavisnost od unapred izabranih operatora,
- robusnije ponašanje na različitim instancama problema,
- mogućnost kombinovanja više različitih destroy i repair strategija,
- bolja ravnoteža između istraživanja novih rešenja i unapređivanja postojećih.

Nedostaci Adaptive LNS

Sa druge strane, adaptivni pristup ima i određene nedostatke:

- složenija implementacija u poređenju sa Basic LNS,
- dodatno računarsko opterećenje zbog selekcije i ažuriranja težina operatora,
- veća osetljivost na izbor parametara (faktor reakcije, sistem bodovanja),
- sporije izvršavanje po iteraciji u odnosu na Basic LNS.

U situacijama kada je unapred poznata dobra kombinacija operatora ili kada je brzina izvršavanja kritična, Basic LNS može predstavljati jednostavnije i efikasnije rešenje.

4 Capacitated Vehicle Routing Problem

Capacitated Vehicle Routing Problem (CVRP) predstavlja jedan od osnovnih i najznačajnijih problema u oblasti rutiranja vozila. Cilj CVRP-a je da se odredi skup ruta za vozila tako da se minimizuje ukupna pređena distanca, uz poštovanje kapaciteta vozila i zahteva korisnika.

Problem se sastoji od centralnog depoa i skupa korisnika sa poznatim lokacijama i zahtevima. Svaki korisnik mora biti posećen tačno jednom, dok svaka ruta mora započeti i završiti se u depou. Ukupna potražnja korisnika na jednoj ruti ne sme premašiti kapacitet vozila.

CVRP spada u klasu NP-teških problema, što znači da se sa porastom broja korisnika prostor mogućih rešenja eksponencijalno povećava, čineći egzaktne metode nepraktičnim za veće instance. Zbog toga se u praksi najčešće primenjuju heuristički i metaheuristički pristupi, kao što je Large Neighborhood Search.

4.1 Formalna definicija problema

Formalno, CVRP je u ovom radu definisan na sledeći način:

- dat je skup čvorova $V = \{0, 1, 2, \ldots, n\}$, gde čvor 0 predstavlja depo,
- svaki korisnik $i \in V \setminus \{0\}$ ima zahtev d_i,
- vozila imaju identičan kapacitet Q,
- data je matrica rastojanja između svih parova čvorova.

Cilj je pronaći skup ruta koje zadovoljavaju sledeće uslove:

- svaka ruta počinje i završava se u depo-u,
- svaki korisnik je uključen tačno u jednu rutu,
- zbir zahteva korisnika na svakoj ruti ne prelazi kapacitet vozila,
- ukupna dužina svih ruta je minimalna.

4.2 Set A instance

U ovom radu koriste se **Set A** instance iz CVRPLIB biblioteke [1]. Ove instance predstavljaju standardni benchmark skup za evaluaciju algoritama za CVRP i sadrže probleme različitih veličina, sa poznatim optimalnim rešenjima.

Set A instance su posebno pogodne za upoređivanje kvaliteta heurističkih i metaheurističkih pristupa, jer omogućavaju direktno merenje odstupanja dobijenih rešenja u odnosu na optimalnu vrednost funkcije cilja.

5 Način rešavanja i implementacija algoritma

5.1 Reprezentacija CVRP problema i rešenja

Osnovna reprezentacija problema implementirana je u modulu `src/cvrp/cvrp_problem.py`, kroz klasu `CVRPProblem`. Ova klasa sadrži sve informacije o instanci problema, a to su skup čvorova, zahteve korisnika, kapacitet vozila, depo, kao i naziv instance koji se koristi tokom eksperimentalne analize.

Metod `initial_solution()` se koristi za generisanje validnog početnog rešenja. Korisnici se nasumično raspoređuju u rute uz poštovanje ograničenja kapaciteta.

Reprezentacija rešenja implementirana je u fajlu `src/cvrp/cvrp_solution.py`, kroz klasu `CVRPSolution`.

Ova klasa omogućava:

- čuvanje strukture ruta,
- izračunavanje ukupne vrednosti funkcije cilja (evaluate funkcija),
- kopiranje rešenja radi bezbedne manipulacije tokom pretrage (konstruktor kopije),
- praćenje uklonjenih korisnika u destroy fazi (removed_customers),
- beleženje istorije vrednosti rešenja radi analize konvergencije (history).

5.2 Basic Large Neighborhood Search

Osnovna varijanta algoritma implementirana je u modulu `src/lns/basic_lns.py`, kroz klasu `BasicLNS`, koja nasleđuje apstraktnu klasu `BaseLNS` definisanu u fajlu `src/lns/base.py`.

U Basic LNS pristupu koristi se fiksna kombinacija destroy i repair (RandomDestroy i GreedyRepair) operatora tokom celokupnog izvršavanja algoritma.

Tok algoritma implementiran je pomoću metoda run i u svakoj iteraciji može se opisati sledećim koracima:

- primena destroy operatora na trenutno rešenje,
- popravka parcijalnog rešenja pomoću repair operatora,
- evaluacija kandidat rešenja,
- odluka o prihvatanju rešenja korišćenjem accept funkcije.

5.3 Adaptive Large Neighborhood Search

Adaptivna varijanta algoritma implementirana je u fajlu `src/lns/adaptive_lns.py`, kroz klasu `AdaptiveLNS`. Ova verzija algoritma uvodi mehanizam učenja koji omogućava dinamički izbor destroy i repair operatora tokom izvršavanja.

Za svaki destroy i repair operator održavaju se odgovarajuće težine (destroy_weights i repair_weights), koje predstavljaju njihovu relativnu uspešnost. Operatori se biraju probabilistički pomoću *roulette wheel* selekcije (metod roulette_wheel), gde operatori sa većom težinom imaju veću verovatnoću izbora.

Sistem bodovanja zasniva se na sledećim kriterijumima:

- ω_1 – pronađeno novo globalno najbolje rešenje,
- ω_2 – poboljšanje trenutnog rešenja,
- ω_3 – prihvaćeno lošije rešenje,
- ω_4 – odbijeno rešenje.

Ažuriranje težina operatora vrši se u metodu update_weights po formuli:

$$w_{\text{new}} = (1 - \lambda) \cdot w_{\text{old}} + \lambda \cdot \text{score},$$

gde parametar λ (reaction_factor) predstavlja faktor reakcije.

5.4 Destroy i Repair operatori

Destroy operatori

Destroy operatori su implementirani u direktorijumu `src/destroy/` i zaduženi su za uklanjanje dela trenutnog rešenja, čime se dobija parcijalno rešenje koje se kasnije popravlja. Implementirane su sledeće strategije:

- **Random Destroy** (`random_destroy.py`) – nasumično uklanja određeni procenat korisnika iz rešenja, bez dodatnih heurističkih kriterijuma.
- **Worst Destroy** (`worst_destroy.py`) – uklanja korisnike čije izbacivanje najviše smanjuje ukupnu cenu rešenja.
- **Related Destroy** (`related_destroy.py`) – uklanja međusobno slične korisnike, pri čemu se sličnost određuje na osnovu udaljenosti, zahteva i pripadnosti istoj ruti.
- **Worst Route Destroy** (`worst_route_destroy.py`) – identifikuje najskuplju rutu u rešenju i delimično je uništava uklanjanjem korisnika iz te rute.

Repair operatori

Repair operatori su implementirani u direktorijumu `src/repair/` i zaduženi su za ponovno ubacivanje uklonjenih korisnika u parcijalno rešenje, uz poštovanje ograničenja kapaciteta vozila. U okviru projekta implementirane su sledeće strategije:

- **Greedy Repair** (`greedy_repair.py`) – svaki uklonjeni korisnik se ubacuje na poziciju koja najmanje povećava ukupnu cenu rešenja.
- **Regret Repair** (`regret_repair.py`) – prioritet daje korisnicima kod kojih bi nepravovremeno ubacivanje imalo najveće negativne posledice po kvalitet rešenja.

5.5 Funkcija prihvatanja rešenja

Kriterijum prihvatanja rešenja implementiran je u modulu `src/accept simulated_annealing_accept.py`. Ovaj pristup dozvoljava prihvatanje lošijih rešenja sa određenom verovatnoćom, naročito u ranim fazama pretrage, čime se smanjuje verovatnoća zaglavljivanja u lokalnim optimumima.

U implementaciji je korišćena geometrijska šema hlađenja, pri čemu je početna temperatura postavljena na vrednost $T_0 = 1000$. Temperatura se u svakoj iteraciji ažurira prema izrazu

$$T \leftarrow \alpha \cdot T,$$

gde je faktor hlađenja $\alpha = 0.95$.

Minimalna dozvoljena vrednost temperature iznosi $T_{\min} = 10^{-6}$, nakon čega se lošija rešenja više ne prihvataju.

5.6 Grafički korisnički interfejs

Pored implementacije algoritama, u okviru projekta je razvijen i jednostavan grafički korisnički interfejs (GUI) koji omogućava interaktivno eksperimentisanje sa Large Neighborhood Search algoritmom.

GUI omogućava korisniku da:

- izabere instancu CVRP problema iz dostupnog skupa test primera,
- odabere algoritam (*Basic LNS* ili *Adaptive LNS*),
- odabere destroy i repair funkcije koje će se koristiti tokom pretrage,
- pokrene algoritam i prati njegovo izvršavanje.

Nakon završetka izvršavanja, interfejs omogućava vizuelizaciju dobijenog rešenja u vidu grafičkog prikaza ruta vozila, kao i prikaz ukupne cene rešenja, što je ostvareno pomoću metoda plot u klasi cvrp_solution.

5.7 Eksperimentalna analiza

Eksperimentalna analiza realizovana je u okviru Jupyter notebook fajla `analytics/analytics.ipynb`. O rezultatitma analize biće više reči u narednom poglavlju.

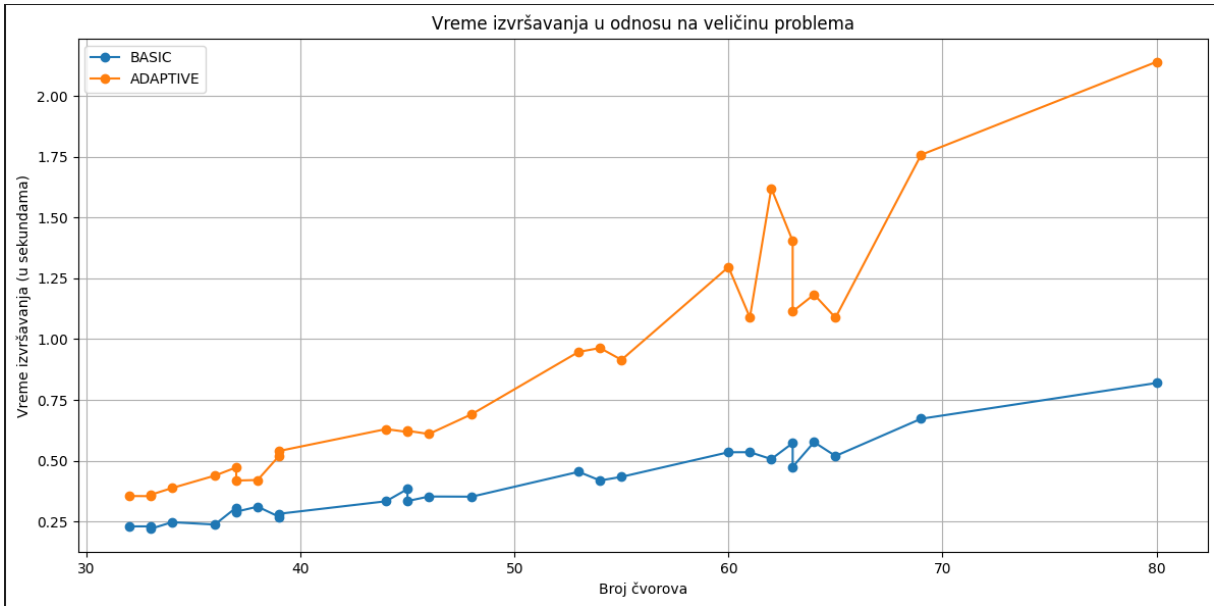
6 Eksperimentalni rezultati i diskusija

Svi eksperimenti su izvršeni na računaru sa operativnim sistemom Linux, korišćenjem Python programskog jezika. Implementacija je realizovana bez korišćenja paralelizacije, a svi algoritmi su testirani pod istim uslovima kako bi poređenje bilo fer i konzistentno.

Eksperimentalna evaluacija Basic i Adaptive Large Neighborhood Search algoritama sprovedena je na instancama iz *Set A* kolekcije CVRPLIB biblioteke. Svi eksperimenti su izvršeni sa fiksnim brojem od **3000 iteracija**.

Za svaku instancu, oba algoritma su pokretana 10 puta, a kao konačan rezultat uzimano je najbolje pronađeno rešenje (u smislu minimalne ukupne cene). U nastavku su prikazane i diskutovane tri grupe rezultata.

6.1 Vreme izvršavanja u odnosu na veličinu instance



Slika 1: Vreme izvršavanja u odnosu na veličinu problema za Basic i Adaptive LNS

Na slici 1 je prikazana zavisnost vremena izvršavanja algoritama od broja čvorova u instanci. Može se uočiti da vreme izvršavanja raste sa povećanjem veličine problema kod oba algoritma, što je očekivano ponašanje.

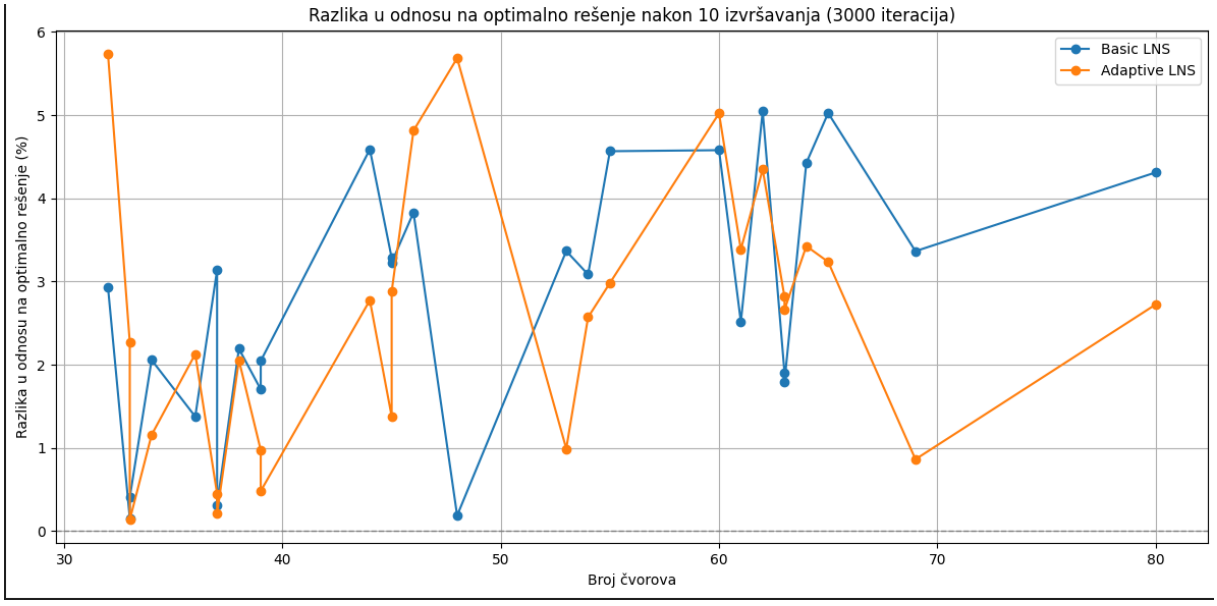
Basic LNS pokazuje stabilno i niže vreme izvršavanja u odnosu na Adaptive LNS za sve testirane instance. Razlog za to je jednostavnija struktura algoritma, jer se u svakom koraku koristi fiksna kombinacija destroy i repair operatora, bez dodatnog računanja vezanog za adaptivnu selekciju.

Adaptive LNS zahteva dodatno vreme zbog mehanizma učenja, selekcije operatora i ažuriranja težina, što se posebno primećuje kod većih instanci. Ipak, i pored tog dodatnog troška, vreme izvršavanja ostaje u prihvatljivim granicama.

6.2 Odstupanje od optimalnog rešenja

Slika 2 prikazuje razliku između najboljeg pronađenog rešenja i optimalne vrednosti poznate iz literature, izraženu u procentima. Za svaku instancu prikazan je rezultat dobijen nakon 10 nezavisnih izvršavanja algoritama, pri čemu je uziman najbolji rezultat.

Rezultati pokazuju da Adaptive LNS u većini slučajeva ostvaruje manju razliku u odnosu na optimalno rešenje u poređenju sa Basic LNS algoritmom. Ova prednost je naročito izražena kod srednjih i većih instanci, gde adaptivni mehanizam omogućava efikasnije kombinovanje različitih



Slika 2: Odstupanje u odnosu na optimalno rešenje za Basic i Adaptive LNS izraženo u procentima

destroy i repair operatora.

Basic LNS u pojedinim manjim instancama postiže konkurentne, pa čak i bolje rezultate, ali njegova performansa značajno zavisi od izabrane kombinacije operatora, što ga čini manje robusnim u opštem slučaju.

6.3 Konvergencija algoritama

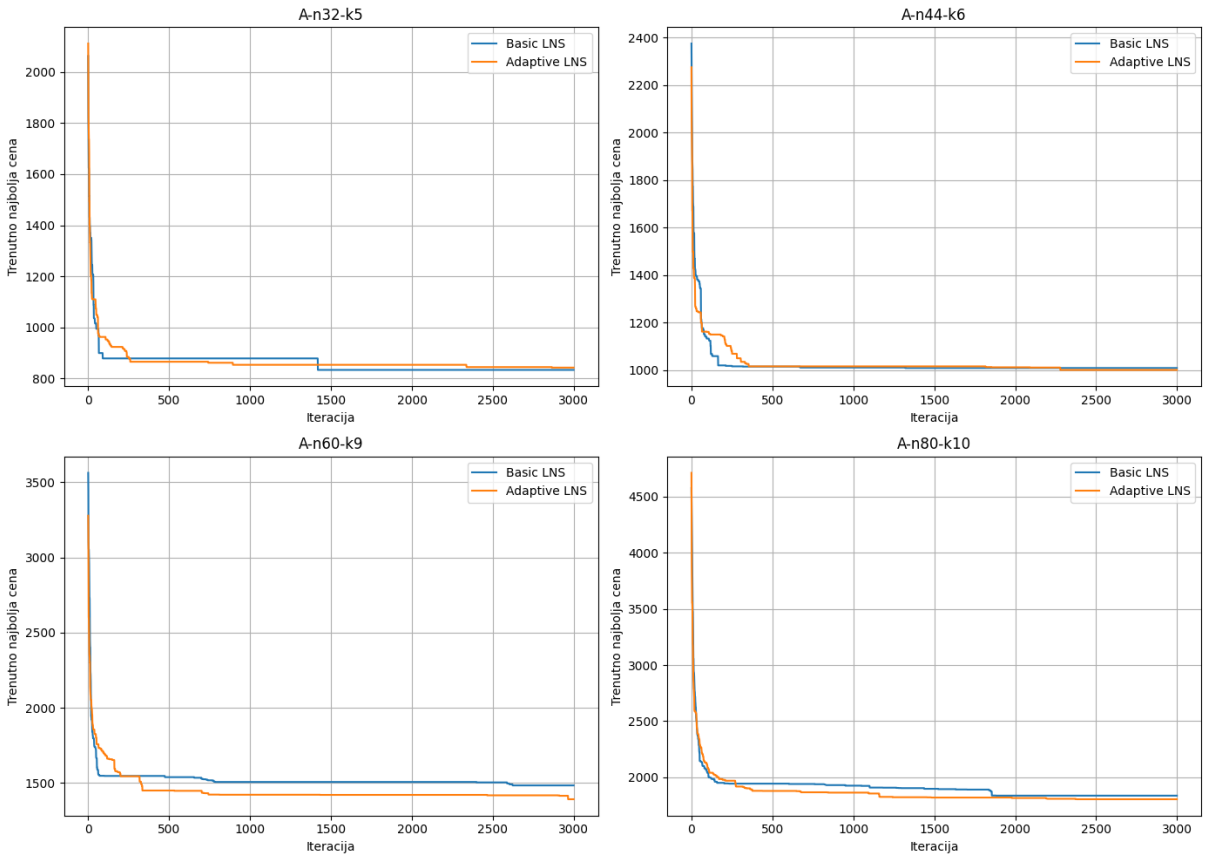
Na slici 3 prikazana je konvergencija algoritama za četiri reprezentativne instance različitih veličina: A-n32-k5, A-n44-k6, A-n60-k9 i A-n80-k10. Na grafikonima je prikazana vrednost najbolje cene pronađene do svake iteracije tokom 3000 iteracija.

Kod manjih instanci oba algoritma brzo konvergiraju ka kvalitetnim rešenjima, pri čemu su razlike u brzini i kvalitetu manje izražene. Kako veličina instance raste, Adaptive LNS pokazuje stabilniju i postepeniju konvergenciju, često dostižući bolje konačne vrednosti.

Adaptive LNS u ranim fazama pretrage pokazuje izraženije proširenje prostora rešenja, dok se u kasnijim iteracijama ponašanje algoritma stabilizuje i fokusira na eksploataciju najboljih pronađenih rešenja.

6.4 Diskusija

Rezultati eksperimentalne analize potvrđuju očekivane osobine oba algoritma. Basic LNS je brži i jednostavniji za implementaciju, ali je osetljiv na izbor operatora. Adaptive LNS, iako sporiji po iteraciji, pokazuje veću robusnost i u proseku postiže kvalitetnija rešenja, posebno kod složenijih instanci.



Slika 3: Konvergencija rešenja za Basic i Adaptive LNS

Na osnovu dobijenih rezultata može se zaključiti da Adaptive LNS predstavlja pogodniji izbor za rešavanje raznovrsnih i većih CVRP instanci, dok Basic LNS može biti efikasan u situacijama gde je unapred poznata dobra kombinacija operatora.

7 Zaključak

7.1 Rezime rada

U ovom radu razmatran je *Capacitated Vehicle Routing Problem*, koji predstavlja jedan od ključnih problema u oblasti kombinatorne optimizacije i logistike. Zbog svoje NP-teške prirode, CVRP se u praksi najčešće rešava primenom heurističkih i metaheurističkih metoda, koje omogućavaju dobijanje kvalitetnih aproksimativnih rešenja u prihvatljivom vremenu.

Fokus rada bio je na implementaciji i eksperimentalnoj analizi *Large Neighborhood Search* algoritma, kroz njegove *Basic* i *Adaptive* varijante.

Eksperimentalna evaluacija sprovedena je na standardnim *Set A* instancama iz CVRPLIB biblioteke, uz poređenje dobijenih rešenja sa poznatim optimalnim vrednostima. Rezultati su pokazali da oba algoritma uspešno pronalaze kvalitetna rešenja, pri čemu Adaptive LNS u proseku ostvaruje manja odstupanja od optimalnih vrednosti, posebno kod srednjih i većih instanci problema.

Sa druge strane, Basic LNS se pokazao kao brži i jednostavniji za implementaciju, ali osetljiviji na izbor konkretnih operatora.

Na osnovu dobijenih rezultata može se zaključiti da Large Neighborhood Search predstavlja efikasan i fleksibilan pristup za rešavanje CVRP problema, dok adaptivna varijanta dodatno povećava robusnost algoritma na različite instance.

7.2 Ograničenja i moguća unapređenja

Iako su dobijeni rezultati prilično dobri, rad ima i određena ograničenja koja treba uzeti u obzir. Pre svega, eksperimentalna analiza je ograničena na instance iz jednog benchmark skupa (Set A), što ne omogućava potpunu generalizaciju zaključaka na sve varijante CVRP problema. Takođe, parametri algoritama (broj iteracija, faktor reakcije, sistem bodovanja) birani su empirijski i nisu dodatno optimizovani.

Mogući pravci daljeg unapređenja uključuju proširenje eksperimentalne analize na druge skupove instanci, kao i razmatranje dodatnih varijanti VRP problema, poput problema sa vremenskim prozorima (VRPTW). Pored toga, moguće unapredjenje rada predstavlja uvođenje novih destroy i repair operatora, kao i detaljnija analiza uticaja parametara adaptivnog mehanizma na brzinu konvergencije i kvalitet rešenja.

Za kraj, dodatna optimizacija implementacije i eventualno korišćenje paralelizacije mogli bi doprineti smanjenju vremena izvršavanja, čime bi algoritam bio pogodniji za primene na većim i realističnijim instancama problema.

Literatura

[1] CVRPLIB. Capacitated vehicle routing problem library. `https://galgos.inf.puc-rio.br/cvrplib`. Accessed: 2026-02.

[2] David Pisinger and Stefan Røpke. A general heuristic for vehicle routing problems. *Transportation Science*, 41(1), 2007.

[3] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP-98)*, 1998.