Rješavanje problema usmjeravanja vozila ograničenog kapaciteta metodom simuliranog kaljenja

Pavičić, Josip

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni diplomski studij matematike
smjer: Matematika i računarstvo

Rješavanje problema usmjeravanja vozila ograničenog kapaciteta metodom simuliranog kaljenja

Diplomski rad

Osijek, 2023

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni diplomski studij matematike
smjer: Matematika i računarstvo

Rješavanje problema usmjeravanja vozila ograničenog kapaciteta metodom simuliranog kaljenja

Diplomski rad

Mentor:

izv. prof. dr. sc. Domagoj Matijević

Komentor:

dr. sc. Mateja Đumić

Kandidat:

Josip Pavičić

Osijek, 2023

Sadržaj

1 | Uvod

U današnjem modernom društvu, efikasnost i optimizacija logističkih procesa igraju ključnu ulogu u ekonomiji i održivosti. Jedan od izazova koji se susreću u logistici je problem usmjeravanja vozila ograničenog kapaciteta, gdje se vozila koriste za dostavu tereta na različite lokacije. Rješenje ovog problema je pronalazak ruta vozila koje minimiziraju ukupnu prijeđenu udaljenost vozila.

U drugom poglavlju rada detaljno ćemo definirati problem usmjeravanja vozila ograničenog kapaciteta te dati matematički model koji ga opisuje.

U trećem i četvrtom poglavlju koristit ćemo simulirano kaljenje i to dvije njegove inačice, osnovnu verziju simuliranog kaljenja i empirijsko simulirano kaljenje. Prvo ćemo definirati algoritme te ih implementirati.

U petom poglavlju usmjereni smo na analizu rezultata dobivenih primjenom implementiranih algoritama. Koristeći programski jezik C++17, pronalazimo najbolja rješenja za svaku instancu problema, računamo prosječna rješenja kroz višestruko pokretanje algoritma, procjenjujemo standardnu devijaciju najboljih rješenja i prosjeka višestrukih pokretanja, te analiziramo apsolutnu i relativnu pogrešku. Dodatno, analiziramo funkcije cilja kroz iteracije kako bismo bolje razumjeli ponašanje algoritama u procesu optimizacije.

Ovaj rad ima za cilj pružiti dublji uvid u primjenu simuliranog kaljenja i empirijskog simuliranog kaljenja za rješavanje problema usmjeravanje vozila ograničenog kapaciteta.

2 | Problem usmjeravanje vozila ograničenog kapaciteta

Usmjeravanje vozila ograničenog kapaciteta (*engl. Capacitated Vehicle Routing Problem - CVRP*) je problem kombinatorne optimizacije koji se javlja u brojnim stvarnim logističkim i transportnim scenarijima. Cilj ovog problema je učinkovito rasporediti vozila kako bi se isporučila roba ili pružile usluge kupcima uz minimalne troškove prijevoza i uz poštivanje kapaciteta vozila. Ovaj problem ima iznimnu važnost u industriji i logistici jer učinkovito raspoređivanje vozila može rezultirati značajnim smanjenjem troškova i optimizacijom vremena isporuke.

2.1 Matematički model problema

U problemu je zadano n koordinata, koje predstavljaju lokacije. Prva lokacija je uvijek lokacija skladišta, a ostalih $n-1$ lokacija su lokacije kupaca. Svaki kupac j ima potražnju robe kapaciteta q_j. Dostupno je m vozila ukupnog kapaciteta Q koji kupcima dostavljaju robu iz skladišta. U inačici problema koju ćemo promatrati u ovom radu, postoji samo jedno skladište i sva vozila istog su kapaciteta. Svaki od n kupaca ima točno određeni zahtjev za količinom robe. Cilj je svim kupcima dostaviti traženu robu iz skladišta tako da duljina ukupno pređenih putova bude minimalna. Vozilo kada jednom krene iz skladišta može obilaziti gradove sve dok ne istovari svu robu koja se u njemu nalazi. Nakon što istovari robu, treba se vratiti u skladište.

Problem usmjeravanja vozila ograničenog kapaciteta formulirat ćemo kao model linearnog cjelobrojnog programiranja. Prilikom rješavanja ovog problema potrebno je minimizirati zbroj prijeđenih udaljenosti svih ruta vozila, pri čemu je potrebno zadovoljiti potražnju svih kupaca.

Binarnu varijablu x_{ijk} definiramo na sljedeći način:

$$x_{ijk} = \begin{cases} 1, & \text{ako putujemo od kupca } i \text{ do kupca } j \text{ s vozilom } k \\ 0, & \text{inače} \end{cases}$$
$$\forall k \in \{1, 2, ..., m\}, i, j \in \{1, 2, ..., n\}.$$

Pri čemu nema putovanja od kupca i do kupca i, odnosno.:

$$x_{iik} = 0 \qquad \forall k \in \{1, 2, ..., m\}, i \in \{1, 2, ..., n\}.$$

Parametar d_{ij} predstavlja udaljenost puta od kupca i do kupca j. Sad možemo definirati funkciju cilja na sljedeći način:

$$\min \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ijk}. \tag{2.1}$$

Nakon što smo definirali funkciju cilja, moramo formulirati uvjete, odnosno ograničenja problema. Takvih ograničenja ima četiri:

1. Vozilo mora napustiti kupca kojeg je posjetilo

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik} \qquad \forall j \in \{1, 2, ..., n\}, k \in \{1, 2, ..., m\}$$

2. Svaki kupac treba biti posjećen točno jednom

$$\sum_{k=1}^{m} \sum_{i=1}^{n} x_{ijk} = 1 \qquad \forall j \in \{2, 3, ..., n\}$$

   Zajedno s prvim uvjetom, osiguravamo da je svaki kupac posjećen točno jednom, istim vozilom.

3. Svako vozilo kreće iz skladišta

$$\sum_{j=2}^{n} x_{1jk} = 1 \qquad \forall k \in \{1, 2, ..., m\}$$

   Zajedno s prvim uvjetom, znamo da se svako vozilo vratilo u skladište.

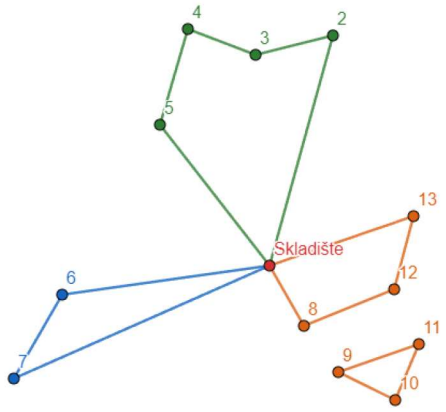
4. Ograničenje kapaciteta
   Prilikom utovara robe potrebno je pripaziti da se poštuje kapacitet vozila.

$$\sum_{i=1}^{n} \sum_{j=2}^{n} q_j x_{ijk} \leq Q \qquad \forall k \in \{1, 2, ..., m\}$$

2.2 Eliminiranje nepovezanih podruta

Uočimo da prethodno definirani problem sam po sebi nije dobro definiran. Moguće je da rješenje koje ispunjava gornje uvjete bude nedopustivo za stvaran problem. To se događa kad je ruta nepovezan graf.

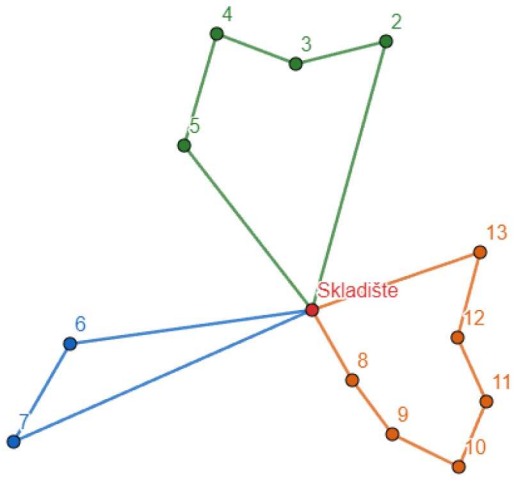
Ruta svakog pojedinog vozila je usmjereni graf gdje su vrhovi kupci i skladište, a lukovi su putovi od jednog kupca do drugog. U svaku rutu skladište je uključeno točno jednom. Ako je ruta nepovezani graf, onda sve komponente te rute koje ne sadrže skladište smatramo nepovezanim podrutama.



Slika 2.1: Prikaz dopustivog rješenja za prethodno definirani problem cjelobrojnog programiranja koje je nedopustivo za stvaran problem.

Postoje različiti načini kako eliminirati pojavljivanje podrute. Mi ćemo koristiti eksplicitnu Dantzig-Fulkerson-Johnson (DFJ) formulaciju, više možete vidjeti u [2].

Ideja DFJ formulacije je koristeći podskupove eliminirati podrute. Skup svih čvorova (kupaca i skladišta) je skup $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$ gdje jedan predstavlja skladište. Skup kupaca koji čini podrutu je $S = \{9, 10, 11\}$ i podskup je od V. S čine tri kupca, te broj putova za taj podskup je također tri, zato je taj skup podruta. Stoga trebamo maknuti jedan put, npr. maknemo put između čvorova 9 i 11.



Slika 2.2: Prikaz dopustivog rješenja

Nakon uklanjanja, moramo dodati dva puta. Možemo dodati puteve između čvorova 8 i 9 te 11 i 12 te maknuti put između čvora 8 i 12. Dobivamo rutu prikazanu na slici 2.2.

Objasnimo ovu eliminaciju podruta. Broj putova koji povezuju čvor od podskupa S i čvor od skupa $V \backslash S$ mora biti barem 2. Odnosno, broj putova između kupaca u podskupu S mora biti strogo manji od broja kupaca u tom podskupu.

Kako bi eliminirali sve moguće podrute, prethodno trebamo primijeniti na svaki podskup koji može postati podruta. Podskup koji sadrži skladište ne može biti podruta. Prazan podskup ili podskup s jednim elementom također ne može biti podruta. Stoga, prethodno treba biti primijenjeno na svaki mogući podskup od V koji ima barem dva elementa i ne sadrži skladište.

Sad možemo dodati još jedan uvjet kako bi problem bio dobro definiran:

5. Eliminiranje podruta

$$\sum_{i \in S, j \notin S} x_{ijk} \geq 2 \qquad S \subset V \backslash \{1\},\ 2 \leq |S| \leq n-2$$

Primijetimo da broj svih podskupova eksponencijalno raste. Broj podskupova od 0 ili svih čvorova je 2, a broj podskupova koji sadrže jedan čvor (bez skladišta) je $n-1$. Stoga broj generiranih podskupova jednak je izrazu $2^n - 2 - (n-1)$.

2.3 Metode rješavanja

Problem usmjeravanje vozila ograničenog kapaciteta (*engl. Capacitated Vehicle Routing Problem - CVRP*) poznati je kombinatorički problem koji pripada kategoriji NP teških problema. U njegovom rješavanju možemo koristiti egzaktne metode ili (meta)heurističke pristupe. Christofides, Mingozii i Toth [1] koristili su egzaktnu metodu grananja i ograničavanja (*engl. branch and bound*) s dinamičkim programiranjem. Korištenjem egzaktnih metoda možemo dobiti globalno optimalno rješenje, ali potrebno vrijeme izvršavanja je eksponencijalno za dovoljno veliku dimenziju problema, vidi [10].

Na ovaj problem primijenjeni su razni heuristički algoritmi. Obaid [6] uspješno koristi tabu pretraživanje (*engl. tabu search*) na problemu CVRP. Tabu pretraživanje je metaheuristička metoda optimizacije koja se koristi za rješavanje različitih kombinatoričkih problema tako što održava popis zabranjenih rješenja kako bi se izbjegli ponavljajući koraci u pretrazi i pronašao bolji rezultat.

U radu [5] pokazano je da još jedan metaheuristički algoritam ostvaruje kompetitivne rezultate u odnosu na ostale heuristike (simulirano kaljenje, tabu pretraživanje, razne varijante Clarke i Wright algoritma štednje). Radi se o algoritmu kolonije mrava (*engl. ant colony optimization*). Taj algoritam simulira ponašanje mrava u prirodi kako bi rješavao optimizacijske probleme, koristeći princip traganja za najkraćim putem temeljenim na feromonima.

Heuristički algoritmi, kao što su simulirano kaljenje, često se koriste za rješavanje optimizacijskih problema. Ti algoritmi ne pretražuju cijeli prostor dopustivih

rješenja, već pametnim odlukama pretražuju dio prostora. Iako ne mogu garantirati pronalazak globalnog optimalnog rješenja, njihova snaga leži u pronalaženju dovoljno dobrih rješenja u razumnom vremenskom okviru.

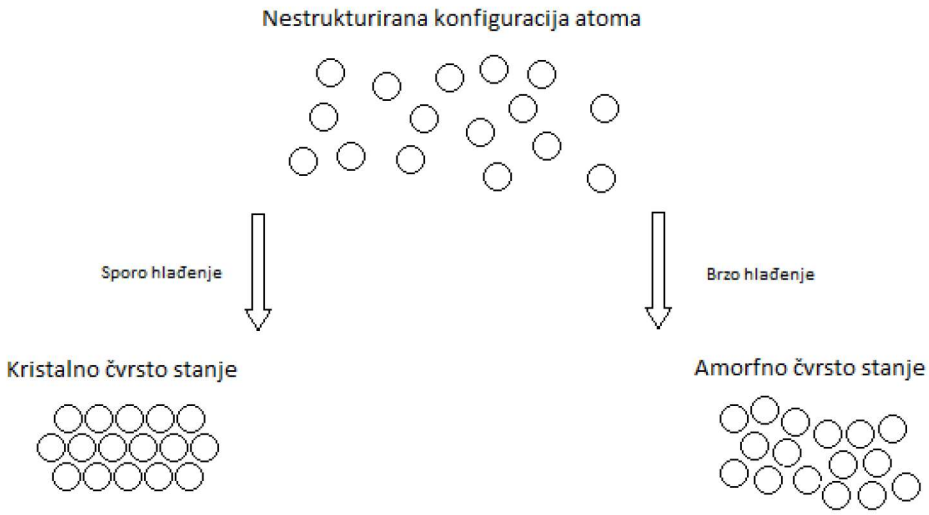
Kombiniranjem pretrage, odlučivanja i često stohastičkih postupaka, heuristike izbjegavaju pretraživanje cijelog područja dopustivih rješenja, što smanjuje vrijeme potrebno za rješavanje problema.

3 | Simulirano kaljenje

3.1 Osnovni algoritam simuliranog kaljenja

Ideja algoritma simuliranog kaljenja inspirirana je metalurgijom, odnosno zagrijavanjem materijala na visoku temperaturu, te postupnog hlađenja materijala kako bi dobili čvršći materijal. Kako bi bolje shvatili ovaj fenomen, pogledajmo sliku 3.1. Na slici vidimo kristalno čvrsto stanje koje je u stanju minimalne energije materijala i najveće tvrdoće, te je to globalni optimum. Takvo stanje dobijemo sporim hlađenjem materijala, dok bržim hlađenjem materijala dobijemo amorfno čvrsto stanje koje postiže lokalni optimum.

Metodom simuliranog kaljenja imitiramo proces kaljenja tako da je funkcija cilja optimizacijskog problema, slična energiji materijala koju minimiziramo uvođenjem "fiktivne" temperature i načina njenog hlađenja.



Slika 3.1: Prikaz stanja materije nakon sporog (lijevo) i brzog (desno) hlađenja.

Simulirano kaljenje je stohastički optimizacijski algoritam. Algoritam pamti samo jedno rješenje te njega unaprjeđuje tako da nasumično odabere kandidata za novo rješenje iz dopustivog područja. Ako je kandidat bolji od trenutnog rješenja, zamjenjuje ga. U suprotnom, ako kandidat uzrokuje povećanje funkcije cilja za

neki ΔE, vjerojatnost prihvaćanja je:

$$p(X) = e^{\frac{-\Delta E(X)}{T}} \tag{3.1}$$

gdje je T imitacija temperature (u praksi, ovu vjerojatnost imitiramo na sljedeći način: izaberemo nasumično cijeli broj iz intervala [0,1] te ako je taj broj manji od $e^{\frac{-\Delta E}{T}}$ onda prihvatimo kandidata kao novo trenutno rješenje, u suprotnom ga odbacimo).

Kad je temperatura T visoka, vjerojatnost prihvaćanja je blizu 1, odnosno za $T >> \Delta E$ izraz $\frac{-\Delta E}{T}$ je blizu nuli. Rezultat toga je da se na početku algoritma, kad je temperatura najviša, većina loših kandidata se prihvaća. To rezultira dobrom diverzifikacijom, pretrage dopustivog područja. Početak algoritma sličan je slučajnoj šetnji dopustivim područjem. No, na niskoj temperaturi, jaka je intenzifikacija i sve više i više se odbacuju loši kandidati. Na srednjoj temperaturi, algoritam povremeno prihvaća kandidate koji povećavaju funkciju cilja. Taj mehanizam omogućuje algoritmu izlazak iz lokalnog optimuma.

Kada algoritam postigne termodinamičku ravnotežu na određenoj temperaturi (u praksi kad postigne predefinirani broj prihvaćanja novih kandidata na određenoj temperaturi), temperatura se dalje smanjuje. Smanjenje temperature omogućuje algoritmu da se približi i konvergira prema optimalnom rješenju. Na nižim temperaturama, algoritam se usredotočuje na fina podešavanja i poboljšanja rješenja kako bi se postigla optimalnost.

Uvjet zaustavljanja određuje trenutak kada se pretraga prostora rješenja zaustavlja i algoritam završava svoje izvođenje. Uzimajući u obzir različite faktore, kao što su dostignuta temperatura, broj iteracija ili vremensko ograničenje, odabir ispravnog uvjeta zaustavljanja ključan je za postizanje željenih rezultata.
Algoritam simuliranog kaljenja se sastoji od četiri glavna dijela:

- početna temperatura
- broj prihvaćanja novih kandidata na temperaturi T
- pravilo hlađenja
- uvjet zaustavljanja

Radi nedostatka teoretskih rezultata, moramo navedene parametre u algoritmu postaviti empirijski [3].

Početna temperatura

Postavljanje prikladne početne temperature može biti izazov jer utječe na ravnotežu između intenzifikacije (usredotočenosti na trenutno najbolje rješenje) i diverzifikacije (istraživanje šireg prostora rješenja). Previsoka početna temperatura može rezultirati prekomjernom diverzifikacijom i sporom konvergencijom, dok

preniska početna temperatura može dovesti do previše intenzifikacije i zaglavljivanja u lokalnim optimumima.

Postoji nekoliko pristupa za određivanje početne temperature. Mi ćemo koristiti jedan od najčešćih pristupa u kojem se početna temperatura postavlja na visoku vrijednost, najčešće neku fiksnu, koja je dovoljno velika da omogući široku pretragu prostora rješenja. Nakon toga, tijekom pretrage, temperatura se postupno smanjuje prema unaprijed definiranom pravilu hlađenja.

Pravilo hlađenja

Dobro pravilo hlađenja trebalo bi smanjivati temperaturu tako da se u što manje iteracija dođe do zadovoljavajućeg rješenja. Postoje više pravila hlađenja, ali u ovom radu koristit ćemo najpopularniji, geometrijski zakon hlađenja:

$$T_{k+1} = \alpha \cdot T_k$$

gdje je k trenutna iteracija, a α konstanta između $[0.8, 0.99]$.

Važno je napomenuti da pri odabiru geometrijskog zakona hlađenja treba uzeti u obzir da odlučujuća brzina hlađenja može značajno utjecati na uspjeh algoritma. Ako je hlađenje prebrzo, postoji rizik da će algoritam brzo završiti u lokalnom optimumu, prije nego što ima priliku istražiti šire područje potencijalnih rješenja. S druge strane, ako je hlađenje presporo, algoritam će trajati predugo ili u slučaju ograničenog vremena može završiti prije nego što dođe do globalnog optimuma.

Stoga, pravilno podešavanje brzine hlađenja igra ključnu ulogu u postizanju dobrih rezultata simuliranog kaljenja, te je važno pažljivo prilagoditi ovaj parametar ovisno o konkretnim zahtjevima problema kako bi se postigla ravnoteža između brzine konvergencije i ispitivanja šireg prostora rješenja.

Algoritmom 1 dan je osnovni algoritam simuliranog kaljenja. Ulaz algoritma je početna temperatura T, minimalna temperatura T_{min}, konstanta α za pravilo hlađenja, broj vozila K, kapacitet vozila Q te lista kupaca $lista\_kupaca$. Treba napomenuti da $lista\_kupaca$ sadrži broj kupca, koordinate kupca i potražnju, dok S sadrži samo broj kupaca. Funkcija f u koraku 5. je funkcija cilja definirana u potpoglavlju 2.1.

Algoritam 1 Simulirano kaljenje

Ulaz: T, T_{min}, α, K, Q, $lista\_kupaca$

1: $S \leftarrow$ *Generiranje početnog rješenja*($lista\_kupaca$, K, Q)
2: **dok je** $T \geq T_{min}$ **činiti**
3: **za** i:=1 **do** 100 **korak** 1 **činiti**
4: $S' \leftarrow$ *Generiranje kandidata*(S)
5: $\Delta E \leftarrow f(S') - f(S)$
6: **ako je** $\Delta E \leq 0$ **onda**
7: $S \leftarrow S'$
8: **inače**
9: prihvati S′ s vjerojatnošću $e^{\frac{-\Delta E}{T}}$
10: $T \leftarrow \alpha T$
11: **vrati** S

3.2 Empirijsko simulirano kaljenje

Empirijsko simulirano kaljenje razlikuje se od običnog simuliranog kaljena u vjerojatnosti prihvaćanja loših rješenja. Glavna ideja ove metode se temelji na tome da distribucija prihvaćanja loših rješenja ima drugačiju distribuciju od eksponencijalne, definirane u jednadžbi 3.1. Eksponencijalna distribucija previše je restriktivna u pretrazi područja dopustivih rješenja. Vjerojatnost prihvaćanja novog lošijeg rješenja prirodno je povezana s distribucijom prethodno prihvaćenih lošijih. Stoga ćemo koristiti gama funkciju gustoće koja je kontrolirana s dva parametra, parametar oblika k i parametar skaliranja θ. Prema tome, vjerojatnost prihvaćanja unutar empirijskog simuliranog kaljenja temeljit će se na empirijski prilagođenoj gama funkciji distribucije, tj. parametri funkcije empirijski su određeni na temelju prethodno prihvaćenih lošijih rješenja. Ovaj način računanja vjerojatnosti prihvaćanja lošijih rješenja omogućuje nam ispravljanje distribucije prihvaćanja lošijih rješenja. Time namjeravamo postići da je prihvaćanje lošijih rješenja efikasnije.

Neka je $Z = \frac{\Delta E(X)}{T}$ slučajna varijabla. Slučajna varijabla Z ima gama distribuciju te je njena funkcija gustoće g dana s:

$$g(z;\theta,k) = \begin{cases} \frac{1}{\theta^k \Gamma(k)} z^{k-1} e^{-\frac{z}{\theta}}, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{3.2}$$

gdje je $k > 0$ parametar oblika, a $\theta > 0$ parametar skaliranja distribucije.
$\Gamma(k)$ je gama funkcija definirana:

$$\Gamma(k) = \int_0^\infty e^{-t} t^{k-1}\, dt.$$

Uočimo da je gama distribucija za $k = 1$ i $\theta = 1$ upravo eksponencijalna.

Metodom maksimalne vjerodostojnosti (*engl. Maximum Likelihood*) procjenjujemo parametre gama distribucije (k i θ) za dani skup prihvaćenih loših rješenja. Radi toga, empirijsko simulirano kaljenje djeluje fleksibilnije u potrazi za globalnim optimumom.

Vjerojatnosna funkcija gama distribucije za N nezavisnih i jednako distribuiranih događaja $(z_1, z_2, ..., z_N)$ dana je s:

$$L(k,\theta) = \prod_{i=1}^{N} g(z_i;k;\theta). \tag{3.3}$$

Korištenjem definicije funkcije gustoće 3.2 dobivamo:

$$L(k,\theta) = \prod_{i=1}^{N} \left(\frac{1}{\theta^k \Gamma(k)} z_i^{k-1} e^{-\frac{z_i}{\theta}} \right).$$

Zatim izraz logaritmiramo kako bismo lakše pronašli maksimum po θ:

$$\begin{aligned}\ell(k,\theta) &= \ln\left(\prod_{i=1}^{N}\left(\frac{1}{\theta^k\Gamma(k)}z_i^{k-1}e^{-\frac{z_i}{\theta}}\right)\right)\\ &= \ln\left(\frac{1}{\theta^{Nk}(\Gamma(k))^N}\cdot\prod_{i=1}^{N}\left(z_i^{k-1}\right)\cdot e^{-\sum_{i=1}^{N}\frac{z_i}{\theta}}\right)\\ &= -Nk\ln(\theta) - N\ln(\Gamma(k)) + (k-1)\sum_{i=1}^{N}\ln(z_i) - \frac{1}{\theta}\sum_{i=1}^{N}z_i.\end{aligned}$$

Prethodni izraz parcijalno deriviramo po θ:

$$\frac{\partial\ell}{\partial\theta} = -\frac{Nk}{\theta} + \frac{1}{\theta^2}\sum_{i=1}^{N}z_i.$$

Zatim parcijalnu derivaciju izjednačimo s nulom:

$$-\frac{Nk}{\theta} + \frac{1}{\theta^2}\sum_{i=1}^{N}z_i = 0.$$

Kad pomnožimo izraz s θ^2 dobijemo $\hat{\theta}$ u kojoj vjerojatnosna distribucija za vrijednost toga parametra postiže maksimum:

$$\hat{\theta} = \frac{1}{kN}\sum_{i=1}^{N}z_i. \tag{3.4}$$

Analogno tražimo k za koji 3.3 postiže maksimum:

$$\begin{aligned}\frac{\partial\ell}{\partial k} &= \sum_{i=1}^{N}\ln(z_i) - N\ln(\theta) - \frac{N}{\Gamma(k)}\Gamma'(k) = 0\\ \sum_{i=1}^{N}\ln(z_i) - N\ln\left(\frac{1}{kN}\sum_{i=1}^{N}z_i\right) - N\psi(k) &= 0\\ \sum_{i=1}^{N}\ln(z_i) + N\ln(k) - N\ln\left(\frac{1}{N}\sum_{i=1}^{N}z_i\right) - N\psi(k) &= 0\\ \ln(k) - \psi(k) &= \ln\left(\frac{1}{N}\sum_{i=1}^{N}z_i\right) - \frac{1}{N}\sum_{i=1}^{N}\ln(z_i)\end{aligned}$$

gdje je $\psi : (0,+\infty) \to \mathbb{R}$ digama funkcija. Ta funkcija je strogo konkavna i strogo monotono rastuća. Definirana je kao derivacija prirodnog logaritma od gama funkcije:

$$\psi(z) = \frac{\partial}{\partial z}\ln\Gamma(z) = \frac{\Gamma'(z)}{\Gamma(z)}.$$

Pronaći aproksimaciju parametra k složen je problem, stoga ćemo koristiti aproksimaciju danu u [7].

$$\hat{k} \approx \frac{3 - q + \sqrt{(q-3)^2 + 24q}}{12q}, \tag{3.5}$$

gdje je q:

$$q = \ln\left(\frac{1}{N}\sum_{i=1}^{N} z_i\right) - \frac{1}{N}\sum_{i=1}^{N}\ln(z_i).$$

Algoritam 2 Empirijsko simulirano kaljenje

Ulaz: T, T_{min}, α, K, Q, $lista\_kupaca$, k, θ
$S \leftarrow$ *Generiranje početnog rješenja*($lista\_kupaca$, K, Q)
dok je $T \geq T_{min}$ **činiti**
    za i:=0 **do** 100 **činiti**
        $S' \leftarrow$ *Generiranje kandidata*(S)
        $\Delta E \leftarrow f(S') - f(S)$
        ako je $\Delta E \leq 0$ **onda**
            S = S'
        inače
            prihvati S' s vjerojatnošću $g(\frac{\Delta E}{T}; \theta, k)$
            dodaj $\frac{\Delta E}{T}$ skupu z
            izračunaj vrijednosti θ, k i ažuriraj ih
    $T \leftarrow \alpha T$
vrati S

Algoritmom 2 dan je pseudokod empirijskog simuliranog kaljenja. Ulaz algoritma je početna temperatura T, minimalna temperatura T_{min}, konstanta α za pravilo hlađenja, broj vozila K, kapacitet vozila Q, lista kupaca $lista\_kupaca$ te parametri gama distribucije k i θ.

Računanje i ažuriranje parametara gama funkcije g definirana jednadžbom 3.2 vršimo funkcijom 3.4 za parametar θ i funkcijom 3.5 za parametar k.

Uočimo da radimo 100 iteracija na jednoj temperaturnoj razini, zatim ju smanjujemo. Više o parametrima algoritma pogledajte u [7].

4 | Implementacija metoda

4.1 Prikaz rješenja

Prikaz rješenja treba moći reprezentirati sva moguća rješenja problema. Između svaka dva rješenja u prostoru rješenja mora postojati put, odnosno iz bilo kojeg rješenja se mora moći doći do bilo kojeg drugog rješenja u konačno mnogo permutacija. Mi ćemo koristiti linearni prikaz rješenja, tj. niz znakova zadane abecede. Točnije koristit ćemo vektor diskretnih varijabli. Vektor diskretnih varijabli S za problem koji ima k vozila i n-1 kupaca je duljine:

$$|S| = (n-1) + (k-1) + 2.$$

Dakle, vektor sadrži sve indekse kupaca kojih ima n - 1. Mora imati indeks skladišta(0) na prvoj i posljednjoj poziciji, te $(k-1)$ indeksa skladišta koji dijele vektor na rute. Ruta počinje i završava skladištem, a svi indeksi između su kupci te rute. Pogledajmo prikaz rješenja:

0	7	6	0	5	4	3	2	0	8	9	10	11	12	13	0

Slika 4.1: Prikaz rješenja problema vizualiziranog na slici 2.2

Iz ove slike možemo iščitati rute vozila. Prvo vozilo kreće iz skladišta, ide do kupca broj 7, pa do kupca broj 6 te se vraća u skladište. Uočimo da rutu prvog vozila čitamo redom s lijeva na desno. Analogno čitamo rute i za vozilo broj 2 i vozilo broj 3.

4.2 Početno rješenje

Oba optimizacijska algoritma, algoritmi 1 i 2, počinju s početnim rješenjem. Početno rješenje možemo odrediti konstruktivnim pristupom (npr. pohlepni pristup, lokalno pretraživanje...) ili nasumičnim pristupom. Konstruktivni pristup gradi početno rješenje korak po korak, počnemo od praznog rješenja i postupno dodajemo komponente (kupce) dok se ne stvori konačno rješenje. Zbog toga ima značajno duže vrijeme izvršavanja, stoga ćemo koristiti nasumičan pristup generiranja početnog rješenja. Početno rješenje generiramo na sljedeći način:

1. **Nasumično miješanje poretka kupaca**: Učitanom redoslijedu posjeta kupaca iz instance problema nasumično promijenimo poredak. Ovo miješanje sprječava algoritam da svaki put krene iz iste početne konfiguracije. Pokretanje algoritma više puta na istoj instanci rezultira različitim početnim konfiguracijama, poboljšavajući istraživanje prostora dopustivih rješenja.

2. **Iterativno dodjeljivanje kupaca vozilima**: Krenemo od nasumično permutiranog redoslijeda kupaca, redom za svakog kupca radimo sljedeći postupak: ako zahtjevi tog kupca stanu unutar kapaciteta prvog vozila, dodijelimo ga tom vozilu. Ako to nije moguće, prelazimo na sljedećeg vozila i tako nastavljamo sve dok svaki kupac ne bude dodijeljen jednom vozilu.

Primijetimo da ovakav način generiranja početnog rješenja može naići na određene situacije u kojima neće biti primjenjiv. Pogledajmo sljedeći primjer:

Primjer 1. *Pretpostavimo da imamo 4 vozila, svaki s kapacitetom 10, te ukupno 8 kupaca* (i, d) *gdje je* i *broj kupca, a* d *zahtjevi kupca. Recimo da smo nasumičnim miješanjem dobili poredak:* $(4, 5), (2, 2), (8, 4), (1, 1), (3, 3), (6, 4), (5, 9), (7, 7)$. *Iterativnim dodjeljivanjem kupaca, prvo vozilo obilazi kupce 4, 2, 1. Drugo vozilo obilazi kupce 8, 3. Treće vozilo obilazi kupca 6 dok zadnje vozilo obilazi kupca 5. Kupca broj 7 ne obilazi nijedno vozilo, jer nijedno vozilo nema dovoljno slobodnog kapaciteta.*

Kako bismo pokušali izbjeći prethodni problem, početni poredak kupaca miješamo sve dok ne dobijemo početno dopustivo rješenje ili dok ne pređemo broj od 100 miješanja.

Generiranje početnog rješenja dan je pseudokodom:

Algoritam 3 Generiranje početnog rješenja

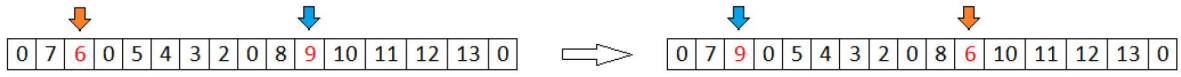
Ulaz: lista_kupaca, K (broj vozila), Q (kapacitet vozila)

```
za i:=1 do 100 činiti
    neka kapaciteti_vozila bude lista duljine K s elementima Q
    neka kupci bude kopija liste lista_kupaca
    nasumično promiješaj listu kupci
    neka S bude lista duljine K čiji su elementi prazne liste - rute
    za svakog kupca iz liste kupci činiti
        za j:=0 do K-1 činiti
            ako je kapaciteti_vozila[j] >= potražnja kupca onda
                dodaj kupca na kraj rute S[j]
                oduzmi zahtjeve kupca od kapaciteti_vozila[j]
                prekini petlju
pretvori S u prikaz rješenja prethodno spomenuti
vrati S
```

4.3 Generiranje kandidata

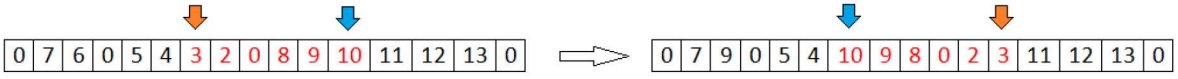
U svakoj iteraciji generiramo kandidata kojeg ćemo možda prihvatiti kao novo rješenje. Kako bismo generirali kandidata iz trenutnog rješenja koristit ćemo se akcijama zamjene, obrtanja i umetanja. Navedene operacije mogu generirati nedopustivog kandidata, stoga nakon odrađene operacije, moramo provjeriti valjanost. Ako je generirani kandidat iz dopustivog područja rješenja, onda donosimo odluku o prihvaćanju.

Zamjena se izvršava tako da izaberemo dva nasumična indeksa vektora prikaza pod uvjetom da nisu prvi i zadnji indeks vektora, te elemente (kupac ili skladište) na tim indeksima zamijenimo.



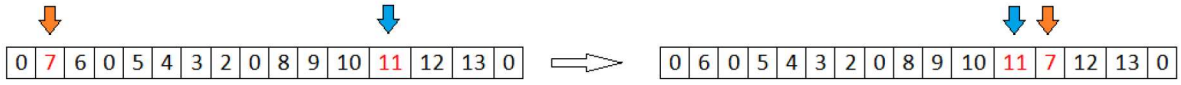
Slika 4.2: Zamjena kupaca na indeksima 2 i 10

U obrtanju, isto kao u zamjeni, izabiremo dva nasumična indeksa koja nisu ni prvi ni zadnji. Zatim obrćemo dio vektora između ta dva indeksa (uključujući te indekse) od većeg indeksa k manjem indeksu.



Slika 4.3: Obrtanje kupaca i skladišta između indeksa 6 i 11

U umetanju, kao i u prethodnim operatorima, na isti način izaberemo nasumična dva indeksa. Umetnemo kupca (ili skladište) s prvog indeksa odmah iza drugog indeksa.



Slika 4.4: Umetanje kupca s indeks 1 iza indeksa 12

U algoritmu 4, kojim je dan pseudokod generiranja kandidata, nasumičan odabir akcija "umetni","zamijeni","obrni" uniformno je distribuiran, tj. svaka akcija ima jednaku vjerojatnost biti odabrana u procesu nasumičnog odabira. Algoritam se izvodi 1000 puta i vrati nepromijenjeno rješenje, ili se zaustavi kad nađe dopustivog kandidata te ga vrati.

Algoritam 4 Generiranje kandidata

Ulaz: S

1: *akcije* ← [*"umetni"*, *"zamijeni"*, *"obrni"*]
2: **za** *i:=1* **do** *1000* **činiti**
3: N := S
4: *akcija* ← nasumično odaberi element liste *akcije*
5: nasumično odaberi dva međusobno različita indeksa *idx1* i *idx2*
6: **ako je** *akcija = "zamjeni"* **onda**
7: zamijeni $S[idx1]$ i $S[idx2]$
8: **inače ako je** *akcija = "obrni"* **onda**
9: obrni redoslijed elemenata u S od *idx1* do *idx2*
10: **inače**
11: $elem \leftarrow S[idx1]$
12: izbriši element S[idx1]
13: umetni element *elem* u na index *idx2 + 1*
14: **ako je** *N dopustivo rješenje* **onda**
15: **vrati** N
16: **vrati** S

4.4 Parametri algoritama

U tablici 4.1 navedeni su parametri algoritma simuliranog kaljenja koje koristimo u ovom radu, dok su u tablici 4.2 navedeni parametri za algoritam simuliranog kaljenja.

Parametar	Vrijednost
T	2000
T_{min}	0.001
α	0.95

Tablica 4.1: Parametri algoritma simuliranog kaljenja

Parametar	Vrijednost
T	2000
T_{min}	0.001
α	0.95
k	1
θ	1

Tablica 4.2: Parametri algoritma empirijskog simuliranog kaljenja

Uočimo kad su k i θ jednaki jedan, imamo eksponencijalnu distribuciju.

Napomena:
Kako se hlađenjem stvara geometrijski niz, možemo eksplicitno izračunati broj

iteracija algoritma. Formula n-tog člana geometrijskog niza je:

$$T_n = T_{start} \cdot \alpha^{n-1},$$

gdje je T_n temperatura n-te temperaturne razine, $T_{start} = 2000$ početna temperatura, $\alpha = 0.95$ koeficijent hlađenja. Uvjet zaustavljanja je kad temperatura padne ispod $T_{min} = 0.001$, stoga imamo izraz:

$$\begin{aligned}
T_n &> T_{min} \\
T_{start} \cdot \alpha^{n-1} &> T_{min} \\
\alpha^{n-1} &> \frac{T_{min}}{T_{start}} \\
\ln\left(\alpha^{n-1}\right) &> \ln\left(\frac{T_{min}}{T_{start}}\right) \\
(n-1) \cdot \ln(\alpha) &> \ln\left(\frac{T_{min}}{T_{start}}\right) \\
n &= \left\lfloor \frac{\ln\left(\frac{T_{min}}{T_{start}}\right)}{\ln(\alpha)} + 1 \right\rfloor
\end{aligned}$$

Uvrštavanjem prethodnih vrijednosti dobivamo $n = 283$ temperaturnih razina. Na svakoj temperaturnoj razini generiramo novo rješenje 100 puta, stoga ukupan broj iteracija algoritama 1 i 2 je 28300.

5 | Rezultati i analiza

U ovom poglavlju provodimo analizu rezultata usporedbe empirijskog simuliranog kaljenja i standardnog simuliranog kaljenja. Kako bismo postigli što kvalitetnije rezultate i zaključke, koristili smo raznovrsne skupove podataka koji se koriste u literaturi za ispitivanje metoda prilikom rješavanja problem usmjeravanja vozila ograničenog kapaciteta.

Za evaluaciju performansi oba algoritma, koristili smo sljedeće skupove podataka: skupovi podataka A, B i P preuzeti su iz studije Augert et al. (1995.). Ovi skupovi podataka predstavljaju izazovne primjere problema usmjeravanja vozila ograničenog kapaciteta i često se koriste za usporedbu različitih algoritama optimizacije. Skupovi podataka variraju u veličini i složenosti, pružajući opsežan spektar problema za evaluaciju performansi algoritama.

Za izvršavanje algoritama koristilo se računalo sa sljedećim specifikacijama: Intel(R) Core(TM) i5 12600k, 16Gb rama, te 64 bitni operativni sustav Windows 11.

Algoritmi su bili implementirani prvotno u programskom jeziku Python, verzija 3.10.11, s višestrukim pokretanjem (100 puta). Zbog sporosti izvršavanja algoritam napisanih u programskom jeziku Python, odlučili smo koristiti[1] C++17 te 10 od ukupno 16 niti za paralelno računanje. Svaka nit je pokretala algoritam 10 puta, što rezultira sa 100 višestrukih pokretanja s različitim početnim rješenjima.

Za prikazivanje rezultata grafički koristimo Python, odnosno biblioteku matplotlib.

U narednim odjeljcima detaljno ćemo analizirati rezultate eksperimenta, uspoređujući rezultate i performanse empirijskog simuliranog kaljenja i običnog kaljenja na različitim skupovima podataka.

5.1 Skupovi podataka

Navedeni skupovi podataka A, B i P dostupni su na [11]. Instance iz skupova podataka su dane u *TSPLIB*95 formatu. To je tekstualni format za pohranjivanje grafova koji se koristi u proučavanju slučajeva problema trgovačkog putnika i srodnih problema, kao što je problem usmjeravanja vozila ograničenog kapaciteta, vidi više na [8].

Pogledajmo primjer instance s 5 kupaca na slici 5.1. U prvom redu koji počinje s "NAME" vidimo naziv problema, koji u sebi sadrži i broj kupaca i skladišta n te

[1] Implementacija dostupna na https://github.com/pavicicjosip/Simulated-Annealing

broj vozila *k*. U retku "COMMENT" nalazi se komentar instance, broj vozila i optimalna vrijednost, ako je izračunata i dokazano optimalna. U redu "CAPACITY" se nalazi kapacitet pojedinog vozila. Koordinate kupaca i skladišta su dane u djelu "NOODE_COORD_SECTION", a potražnja kupaca i skladišta je dana u djelu "DEMAND_SECTION". Indeks skladišta je dan u djelu "DEPOT_SECTION" nakon kojeg slijedi red s -1 te red s EOF. Uočimo da je potražnja skladišta nula.

```
NAME : A-n6-k2
COMMENT : (No of trucks: 2, Optimal value: OPT)
TYPE : CVRP
DIMENSION : 6
EDGE_WEIGHT_TYPE : EUC_2D
CAPACITY : 50
NODE_COORD_SECTION
 1 82 76
 2 96 44
 3 50 5
 4 49 8
 5 13 7
 6 29 89
DEMAND_SECTION
1 0
2 19
3 21
4 6
5 19
6 7
DEPOT_SECTION
 1
 -1
EOF
```

Slika 5.1: Instanca s 5 kupaca i 2 vozila kapaciteta 40.

5.2 Rezultati i analiza

U ovom poglavlju usporedit ćemo rješenja algoritama na svakom skupu podataka. U tablici ćemo prikazati udaljenosti dobivene empirijskim simuliranim kaljenjem(ESK) i običnim simuliranim kaljenjem(SK).

5.2.1 Rezultati simuliranog kaljenja

U tablici 5.1 prikazani su rezultati simuliranog kaljenja za instance skupa podataka A, B i P. Tablica sadrži relevantne podatke o usporedbi optimalnih rješenja s rješenjima dobivenim simuliranim kaljenjem. U većini problema dobili smo zadovoljavajuće rješenje, dok smo za 10 instanci postigli optimalna rješenja. Prosjek apsolutnih pogrešaka (u tablici 5.1) skupa A je 24.78, skupa B je 20.47, skupa P je 20.9. Prosječna relativna pogreška skupa A je 1.94%, skupa B je 1.91%, skupa P je 3.04%. Prosječno vrijeme računanja skupa A je 336.75, skupa B 345.21, skupa P 314.85 sekundi.

Inst.	Q	Opt.	S naj.	S pros.	ΔS^*	δS^*	σ	Vrijeme(s)
A-n32-k5	100	784	787.08	822.30	3.08	0.39	22.79	192.83
A-n33-k5	100	661	662.11	684.09	1.11	0.17	16.30	225.06
A-n33-k6	100	742	742	759.57	0	0	18.98	246.38
A-n34-k5	100	778	780.94	803.23	2.94	0.38	17.46	249.54
A-n36-k5	100	799	802.13	841.80	3.13	0.39	20.48	248.89
A-n37-k5	100	669	672.47	696.82	3.47	0.52	15.74	248.59
A-n37-k6	100	949	949	1001.94	0	0	24.33	273.41
A-n38-k5	100	730	735.05	784.73	5.05	0.69	29.01	268.66
A-n39-k5	100	822	829.76	879.28	7.76	0.94	30.49	266.02
A-n39-k6	100	831	835.25	860.70	4.25	0.51	19.86	255.02
A-n44-k6	100	937	938.84	1010.75	1.84	0.20	30.52	295.42
A-n45-k6	100	944	948.55	1092.52	4.55	0.48	81.56	317.69
A-n45-k7	100	1146	1151.34	1219.26	5.34	0.47	26.84	302.17
A-n46-k7	100	914	918.13	976.83	4.13	0.45	26.22	304.04
A-n48-k7	100	1073	1096.19	1150.13	23.19	2.16	26.13	321.47
A-n53-k7	100	1010	1035.19	1101.54	25.19	2.49	35.21	350.05
A-n54-k7	100	1167	1190.16	1275.53	23.16	1.98	39.37	353.33
A-n55-k9	100	1073	1077.40	1156.99	4.40	0.41	37.98	365.06
A-n60-k9	100	1354	1383.84	1465.91	29.84	2.20	34.28	400.42
A-n61-k9	100	1034	1070.52	1160.78	36.52	3.53	65.63	439.53
A-n62-k8	100	1288	1332.53	1414.32	44.53	3.46	40.13	391.63
A-n63-k9	100	1616	1694.50	1786.03	78.50	4.86	43.48	452.57
A-n63-k10	100	1314	1363.14	1428.84	49.14	3.74	35.32	425.86
A-n64-k9	100	1401	1467.53	1539.58	66.53	4.75	34.49	432.41
A-n65-k9	100	1174	1253.20	1395.63	79.20	6.75	63.96	451.90
A-n69-k9	100	1159	1198.49	1280.51	39.49	3.41	36.32	467.02
A-n80-k10	100	1763	1885.71	1985.39	122.71	6.96	45.31	547.29
B-n31-k5	100	672	676.09	685.16	4.09	0.61	6.20	182.62
B-n34-k5	100	788	788	795.95	0	0	7.16	208.98
B-n35-k5	100	955	955	979.70	0	0	12.63	227.87
B-n38-k6	100	805	808.70	821.20	3.70	0.46	10.98	239.43
B-n39-k5	100	549	553.16	573.66	4.16	0.76	17.65	241.79
B-n41-k6	100	829	835.90	877.67	6.90	0.83	28.31	280.29
B-n43-k6	100	742	746.98	764.37	4.98	0.67	12.90	274.20
B-n44-k7	100	909	929.39	970.64	20.39	2.24	32.73	299.58
B-n45-k5	100	751	755.00	823.65	4.00	0.53	43.34	299.02
B-n45-k6	100	678	684.74	764.44	6.74	0.99	45.66	315.45
B-n50-k7	100	741	744.23	757.71	3.23	0.44	20.08	316.01
B-n50-k8	100	1312	1322.85	1355.92	10.85	0.83	27.67	330.58
B-n51-k7	100	1032	1041.20	1151.02	9.20	0.89	66.01	367.18
B-n52-k7	100	747	752.81	785.20	5.81	0.78	32.97	329.44
B-n56-k7	100	707	719.16	749.48	12.16	1.72	25.33	344.59
B-n57-k7	100	1153	1166.06	1406.35	13.06	1.13	132.26	393.44
B-n57-k9	100	1598	1616.63	1680.05	18.63	1.17	33.02	399.98
B-n63-k10	100	1496	1555.68	1621.54	59.68	3.99	36.09	452.36

B-n64-k9	100	861	932.75	1060.27	71.75	8.33	66.50	476.59
B-n66-k9	100	1316	1358.55	1431.19	42.55	3.23	47.06	470.44
B-n67-k10	100	1032	1075.56	1133.62	43.56	4.22	36.99	471.17
B-n68-k9	100	1272	1305.85	1385.98	33.85	2.66	44.26	474.24
B-n78-k10	100	1221	1312.40	1385.12	91.40	7.49	43.72	546.58
P-n16-k8	35	450	450	459.92	0	0	8.92	183.89
P-n19-k2	160	212	212	223.72	0	0	8.09	142.52
P-n20-k2	160	216	216	223.18	0	0	7.52	142.81
P-n21-k2	160	211	211	214.05	0	0	3.96	135.98
P-n22-k2	160	216	216	220.35	0	0	6.49	145.45
P-n22-k8	3000	603	615.12	887.50	12.12	2.01	49.28	206.57
P-n23-k8	40	529	529	574.70	0	0	27.33	212.17
P-n40-k5	140	458	461.73	479.30	3.73	0.81	13.18	246.36
P-n45-k5	150	510	512.79	538.85	2.79	0.55	15.58	280.19
P-n50-k7	150	554	562.47	590.01	8.47	1.53	13.37	322.29
P-n50-k8	120	631	644.95	680.81	13.95	2.21	29.52	351.32
P-n50-k10	100	696	705.17	746.51	9.17	1.32	20.03	372.31
P-n51-k10	80	741	765.29	822.47	24.29	3.28	45.04	387.50
P-n55-k7	170	568	580.72	603.38	12.72	2.24	12.93	338.97
P-n55-k8	160	588	593.51	615.94	5.51	0.94	15.95	338.24
P-n55-k10	115	694	707.73	733.24	13.73	1.98	14.15	372.92
P-n55-k15	70	989	1011.59	1067.88	22.59	2.28	45.27	423.34
P-n60-k10	120	744	761.92	806.84	17.92	2.41	17.21	414.63
P-n60-k15	80	968	991.77	1035.14	23.77	2.46	19.17	459.02
P-n65-k10	130	792	823.46	859.75	31.46	3.97	21.49	439.80
P-n70-k10	135	827	877.04	951.17	50.04	6.05	34.55	483.55
P-n76-k4	350	593	656.82	716.04	63.82	10.76	28.22	450.79
P-n76-k5	280	627	705.19	760.89	78.19	12.47	31.41	476.09
P-n101-k4	400	681	788.43	857.53	107.43	15.78	32.77	568.53

Tablica 5.1: Algoritam simuliranog kaljenja 1 na instancama skupova podataka A, B, P.

5.2.2 Rezultati empirijskog simuliranog kaljenja

U tablici 5.2 prikazani su rezultati empirijskog simuliranog kaljenja za instance skupa podataka A, B i P. Tablica sadrži relevantne podatke o usporedbi optimalnih rješenja s rješenjima dobivenim simuliranim kaljenjem. U većini problema dobili smo zadovoljavajuće rješenje, dok smo za 11 instanci postigli optimalna rješenja. Prosjek apsolutnih pogrešaka (u tablici 5.2) skupa A je 24.50, skupa B je 19.11, skupa P je 16.43. Prosječna relativna pogreška skupa A je 1.91%, skupa B je 1.79%, skupa P je 2.39%. Prosječno vrijeme računanja skupa A je 278.97, skupa B 280.85, skupa P 282.3 sekundi.

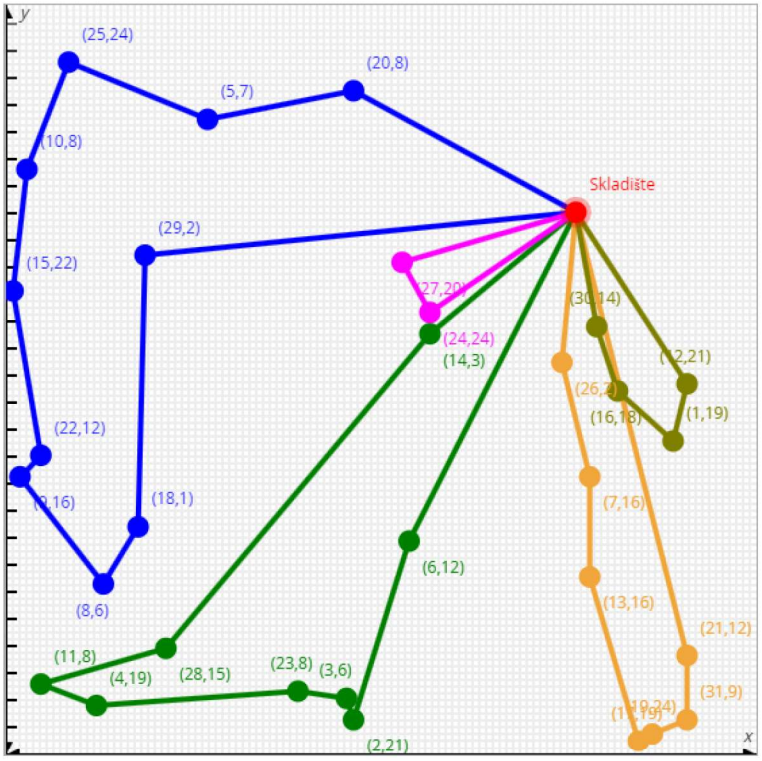
Inst.	Q	Opt.	S naj.	S pros.	ΔS^*	δS^*	σ	Vrijeme(s)
A-n32-k5	100	784	787.08	845.37	3.08	0.39	37.00	171.84
A-n33-k5	100	661	662.26	701.41	1.26	0.19	26.20	190.96
A-n33-k6	100	742	742	789.69	0	0	31.92	198.04
A-n34-k5	100	778	780.94	821.47	2.94	0.38	26.03	191.62
A-n36-k5	100	799	812.98	859.24	13.98	1.75	25.32	201.00
A-n37-k5	100	669	672.47	713.19	3.47	0.52	28.82	198.45
A-n37-k6	100	949	949	1024.23	0	0	41.00	224.79
A-n38-k5	100	730	735.05	813.60	5.05	0.69	52.33	226.88
A-n39-k5	100	822	837.65	905.94	15.65	1.90	33.07	224.98
A-n39-k6	100	831	835.25	883.96	4.25	0.51	31.76	215.51
A-n44-k6	100	937	960.69	1035.24	23.69	2.53	38.82	247.60
A-n45-k6	100	944	962.21	1160.76	18.21	1.93	119.48	265.76
A-n45-k7	100	1146	1166.83	1234.35	20.83	1.82	32.07	248.99
A-n46-k7	100	914	918.13	985.73	4.13	0.45	32.25	252.45
A-n48-k7	100	1073	1074.34	1162.76	1.34	0.12	32.82	266.24
A-n53-k7	100	1010	1021.79	1122.80	11.79	1.17	46.74	297.56
A-n54-k7	100	1167	1197.31	1289.36	30.31	2.60	52.13	300.03
A-n55-k9	100	1073	1082.54	1167.00	9.54	0.89	43.94	305.33
A-n60-k9	100	1354	1378.54	1476.38	24.54	1.81	41.45	326.45
A-n61-k9	100	1034	1064.11	1150.30	30.11	2.91	45.84	362.84
A-n62-k8	100	1288	1330.97	1425.95	42.97	3.34	44.72	333.01
A-n63-k9	100	1616	1695.59	1802.03	79.59	4.92	48.63	356.90
A-n63-k10	100	1314	1344.06	1433.39	30.06	2.29	42.23	351.99
A-n64-k9	100	1401	1465.87	1533.28	64.87	4.63	37.00	353.32
A-n65-k9	100	1174	1233.03	1392.67	59.03	5.03	80.97	381.59
A-n69-k9	100	1159	1202.95	1268.07	43.95	3.79	34.82	374.68
A-n80-k10	100	1763	1879.77	1968.49	116.77	6.62	42.93	433.45
B-n31-k5	100	672	676.09	692.05	4.09	0.61	13.99	154.11
B-n34-k5	100	788	788	808.60	0	0	24.11	170.09
B-n35-k5	100	955	955	986.91	0	0	29.56	194.52
B-n38-k6	100	805	808.70	831.02	3.70	0.46	16.78	208.73
B-n39-k5	100	549	553.16	583.45	4.16	0.76	20.77	209.65
B-n41-k6	100	829	836.79	903.52	7.79	0.94	55.34	238.82
B-n43-k6	100	742	746.98	770.83	4.98	0.67	18.54	226.63
B-n44-k7	100	909	929.39	991.84	20.39	2.24	34.32	248.68
B-n45-k5	100	751	754.67	866.30	3.67	0.49	64.78	248.50
B-n45-k6	100	678	684.21	808.74	6.21	0.92	62.35	272.44
B-n50-k7	100	741	744.23	759.06	3.23	0.44	24.10	267.11
B-n50-k8	100	1312	1329.78	1382.66	17.78	1.36	41.76	283.07
B-n51-k7	100	1032	1039.52	1224.23	7.52	0.73	98.27	304.56
B-n52-k7	100	747	750.14	794.24	3.14	0.42	34.80	266.57
B-n56-k7	100	707	714.72	757.74	7.72	1.09	31.35	286.47
B-n57-k7	100	1153	1164.58	1375.73	11.58	1.00	101.65	330.47
B-n57-k9	100	1598	1627.85	1696.50	29.85	1.87	38.42	316.11
B-n63-k10	100	1496	1562.63	1621.83	66.63	4.45	30.66	351.89

B-n64-k9	100	861	946.72	1095.90	85.72	9.96	64.70	371.34
B-n66-k9	100	1316	1342.94	1435.06	26.94	2.05	49.06	359.87
B-n67-k10	100	1032	1073.29	1143.69	41.29	4.00	45.41	357.86
B-n68-k9	100	1272	1303.81	1400.94	31.81	2.50	44.02	371.01
B-n78-k10	100	1221	1272.37	1369.82	51.37	4.21	43.24	420.90
P-n16-k8	35	450	450	476.92	0	0	14.86	162.77
P-n19-k2	160	212	212	236.68	0	0	9.96	122.28
P-n20-k2	160	216	216	238.39	0	0	12.93	122.10
P-n21-k2	160	211	211	219.49	0	0	8.95	118.77
P-n22-k2	160	216	216	230.25	0	0	11.78	128.29
P-n22-k8	3000	603	617.74	643.77	14.74	2.44	32.74	183.53
P-n23-k8	40	529	529	561.22	0	0	25.08	191.08
P-n40-k5	140	458	461.73	486.27	3.73	0.81	15.53	211.15
P-n45-k5	150	510	513.42	542.08	3.42	0.67	18.05	238.15
P-n50-k7	150	554	564.06	597.44	10.06	1.82	20.57	275.32
P-n50-k8	120	631	640.96	699.68	9.96	1.58	46.16	334.11
P-n50-k10	100	696	709.37	756.01	13.37	1.92	25.18	312.28
P-n51-k10	80	741	752.99	858.29	11.99	1.62	46.48	341.27
P-n55-k7	170	568	582.49	602.37	14.49	2.55	12.23	291.42
P-n55-k8	160	588	595.21	620.31	7.21	1.12	18.35	294.36
P-n55-k10	115	694	709.85	732.07	15.85	2.28	13.69	323.49
P-n55-k15	70	989	1002.23	1097.30	13.23	1.34	65.04	362.64
P-n60-k10	120	744	762.69	808.56	18.69	2.51	22.39	355.56
P-n60-k15	80	968	980.62	1041.26	12.62	1.30	28.02	391.67
P-n65-k10	130	792	824.74	861.25	32.74	4.13	22.98	376.76
P-n70-k10	135	827	881.63	950.62	54.63	6.61	37.55	421.18
P-n76-k4	350	593	640.00	704.29	47.00	7.93	29.06	383.33
P-n76-k5	280	627	661.86	744.12	34.86	5.56	32.25	401.20
P-n101-k4	400	681	756.71	813.84	75.71	11.12	31.14	432.58

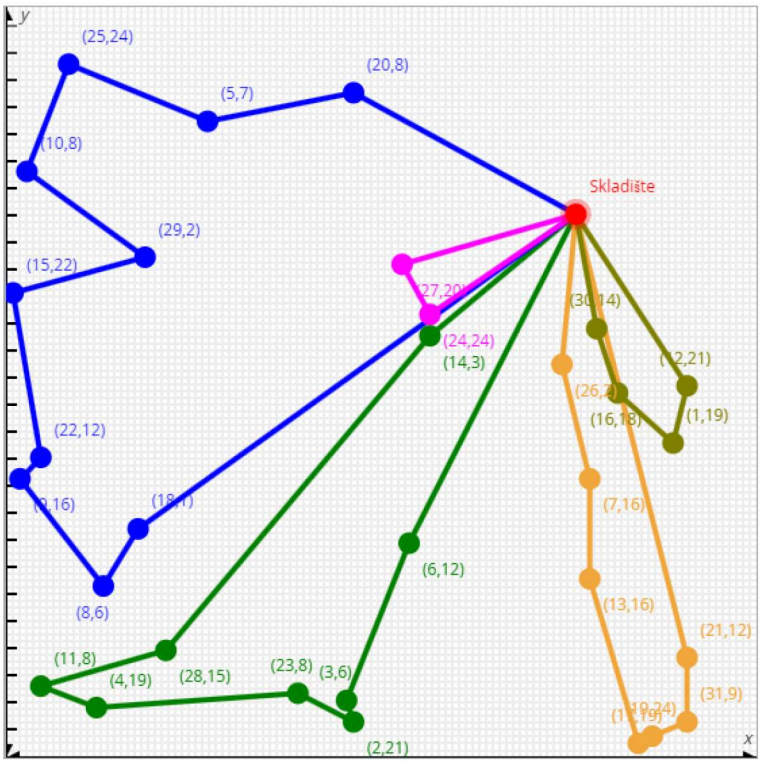
Tablica 5.2: Algoritam empirijskog simuliranog kaljenja 2 na instancama skupova podataka A, B, P.

5.2.3 Analiza rješenja

Kako bismo bolje razumjeli razlike između optimalnog rješenja i najboljeg rješenja dobivenim simuliranim kaljenjem, pogledajmo dvije slike koje vizualiziraju rute instance A-32n-k5, dane na slikama 5.2 i 5.3. Gledajući sliku vidimo da se raspored obilaska gradova plave i zelene rute razlikuju. Ako gledamo plavu rutu, najbrže postižemo optimum operacijom umetanja tako što kupca s brojem 29 umetnemo desno od kupca s brojem 18. Slično za zelenu rutu, kojom god taktikom se koristili, ako odaberemo kupca s brojem 2 i kupca s brojem 3, postižemo optimalnu rutu.



Slika 5.2: Optimalno rješenje instanca A-n32-k5.

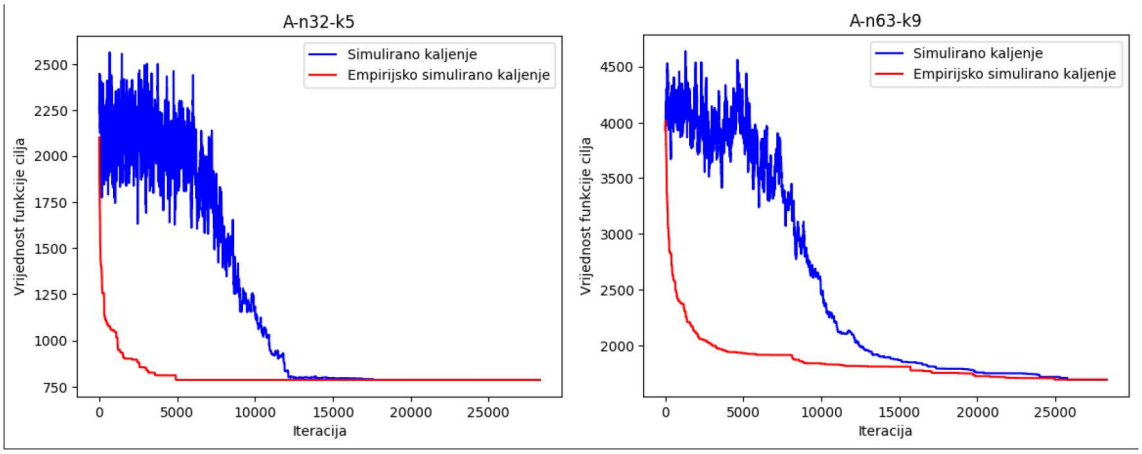


Slika 5.3: Rješenje instance A-n32-k5 dobiveno simuliranim kaljenjem i empirijskim simuliranim kaljenjem.

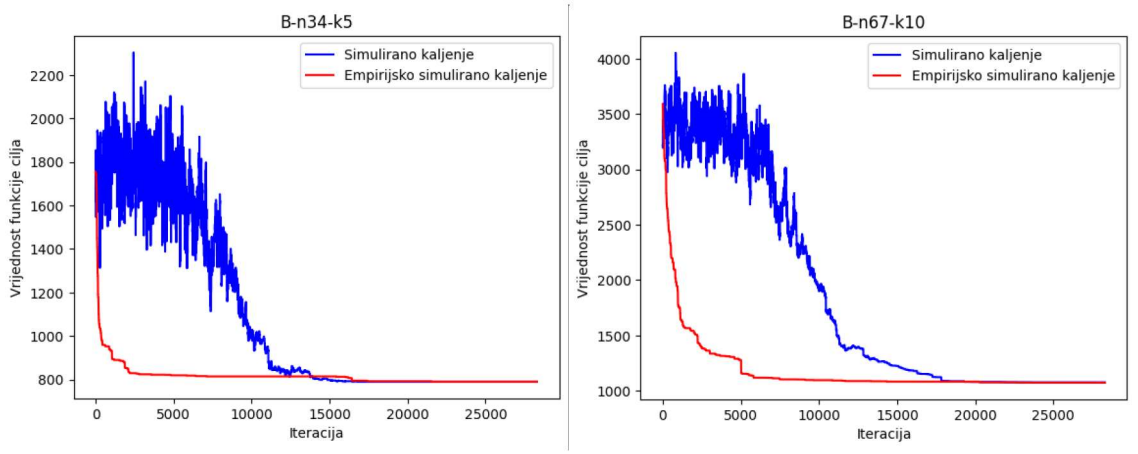
U analizi funkcije cilja usporedili smo brzinu približavanja optimumu između algoritma simuliranog kaljenja i empirijskog simuliranog kaljenja na nekoliko primjera (slike 5.4, 5.5, 5.6).

Primjećujemo da se funkcija cilja ESK puno brže približava optimalnom rješenju nego funkcija cilja SK. Ovo je vidljivo po strmijem padu krivulje funkcije cilja ESK, dok krivulja funkcije cilja SK pokazuje sporiji pad.

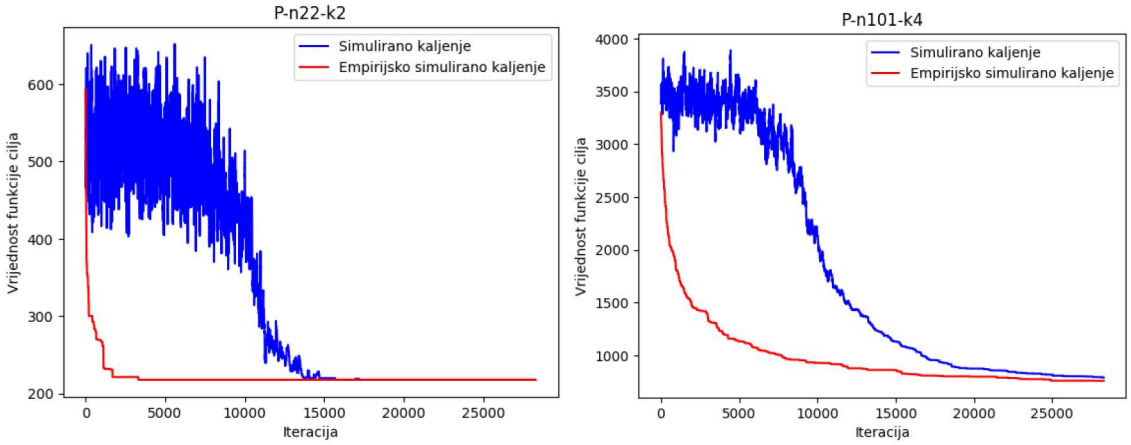
Uočimo da SK prihvaća puno više lošijih rješenja s velikim odstupanjem od trenutnog rješenja, nego što to radi ESK. Ovo ukazuje na veću efikasnost algoritma ESK u pronalaženju boljih rješenja u usporedbi s algoritmom SK. Uz veću efikasnost ESK algoritma, u prosjeku dobivamo i bolja rješenja nego od SK algoritma. Također ESK algoritam je brži u izvršavanju za isti broj iteracija, nego SK algoritam. Jedina prednost SK algoritma u odnosu na ESK je ta što za većinu instanci problema ima manju standardnu devijaciju. Razlog tomu je što graf funkcije cilja empirijskog simuliranog kaljenja podsjeća na graf lokalnog pretraživanja, tj. algoritma u kojem se ne prihvaćaju lošija rješenja. Tek kad se zumira u graf funkcije cilja EKS algoritma, mogu se vidjeti prihvaćanja lošijih rješenja.



Slika 5.4: Slike funkcije cilja algoritama ESK i SK za dvije instance skupa A.



Slika 5.5: Slike funkcije cilja algoritama ESK i SK za dvije instance skupa B.



Slika 5.6: Slike funkcije cilja algoritama ESK i SK za dvije instance skupa P.

6 | Zaključak

U ovom istraživanju detaljno smo proučavali problem usmjeravanja vozila ograničenog kapaciteta i primijenili algoritme simuliranog kaljenja (SK) i empirijskog simuliranog kaljenja (ESK) kako bismo pronašli optimalna rješenja za različite instance problema. Kroz analizu rezultata i funkcija cilja, istraživali smo kako se ovi algoritmi ponašaju i uspoređivali njihovu učinkovitost.

Analizirajući apsolutne i relativne pogreške, primijetili smo da su rezultati vrlo blizu optimalnih rješenja, što ukazuje na visoku preciznost algoritama. Osim toga, vrijeme računanja za oba algoritma ostalo je u prihvatljivim granicama, što je važno za praktičnu primjenu u stvarnim logističkim sustavima.

Najznačajnije uočavanje proizlazi iz analize funkcije cilja. Na temelju usporedbe brzine približavanja optimumu između algoritma SK i ESK (vidljive na slikama 3.4, 3.5 i 3.6), zaključujemo da se funkcija cilja ESK puno brže približava optimalnom rješenju nego funkcija cilja SK. ESK algoritam pokazuje strmiji pad krivulje funkcije cilja, što ukazuje na njegovu veću efikasnost u bržem pronalaženju boljih rješenja. Osim toga, ESK algoritam u prosjeku dobiva bolja rješenja od SK algoritma.

U zaključku, ovaj rad pruža dublji uvid u primjenu simuliranog kaljenja i empirijskog simuliranog kaljenja za rješavanje problema usmjeravanja vozila ograničenog kapaciteta. ESK se pokazao kao efikasniji i brži algoritam u procesu optimizacije, nudeći bolja rješenja u kraćem vremenu.

Literatura

[1] N. Christofides, A. Mingozzi, P. Toth, *Exact algorithm for the vehicle routing problem, based on spanning tree and shortest path relaxations*, Mathematical Programming, 20, 255–282. (1981)

[2] G. Dantzig, R. Fulkerson, S. Johnson, *Solution of a Large-Scale Traveling-Salesman Problem*, Journal of the Operations Research Society of America 2(4):393-410 (1954).

[3] J. Dréo, A. Pétrowski, P. Siarry, R. Taillard, *Metaheuristics for Hard Optimization*, Springer-Verlag Berlin Heidelberg, 2006

[4] S. Kurepa, H. Kraljević, *Matematička analiza, četvrti dio,*Tehnička knjiga, Zareb, 1986.

[5] S. Mazzeo, I. Loiseau, *An Ant Colony Algorithm for the Capacitated Vehicle Routing*, Electronic Notes in Discrete Mathematics 18 (2004) 181–186

[6] O. I. Obaid, *Solving Capacitated Vehicle Routing Problem (CVRP) Using Tabu Search Algorithm (TSA)*, AL-Iraqia University, 2018

[7] B. Rabbouch, F. Saâdoui, R. Mraihi, *Empirical-type simulated annealing for solving the capacitated vehicle routing problem*, Journal of Experimental & Theoretical Artificial Intelligence (2019)

[8] G. Reinelt, *TSPLIB 95*, Heidelberg University Institute for Applied Mathematics

[9] N. Sarapa, *Teorija vjerojatnosti*,Školska knjiga, Zareb, 2002.

[10] S.-W. Lin, Z.-J. Lee, K.-C. Ying, C.-Y. Lee, *Applying hybrid meta-heuristics for capacitated vehicle routing problem*, Expert Systems with Applications 36 (2009) 1505–1512.

[11] Web izvor dostupan na `\http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`.

Sažetak

U današnjem modernom društvu, optimizacija logističkih procesa igra ključnu ulogu u ekonomiji i održivosti. Jedan od izazova u logistici je problem usmjeravanja vozila ograničenog kapaciteta, gdje se vozila koriste za dostavu tereta na različite lokacije. Ovaj problem zahtijeva optimalno raspoređivanje vozila kupcima kako bi se minimizirala prijeđena udaljenost vozila.

U radu se prvo definira problem raspoređivanja vozila s kapacitetom i daje matematički model problema. Zatim se predstavljaju algoritmi simuliranog kaljenja (SK) i empirijskog simuliranog kaljenja (ESK) kao metode za rješavanje ovog problema. SK koristi eksponencijalnu distribuciju prihvaćanja lošijih rješenja, dok ESK koristi gama distribuciju.

Nakon toga, detaljno se opisuje implementacija ovih algoritama i njihova primjena na problem usmjeravanja vozila ograničenog kapaciteta. Prikazane su različite komponente implementacije, uključujući generiranje početnih rješenja i kandidata te sam algoritam simuliranog kaljenja.

U petom dijelu rada analiziraju se rezultati dobiveni primjenom ovih algoritama na različite skupove podataka (A, B, P). Uspoređuju se optimalna rješenja s rješenjima dobivenim pomoću SK i ESK. Rezultati pokazuju da su oba algoritma sposobna pružiti zadovoljavajuća rješenja, pri čemu su za nekoliko instanci postignuta optimalna rješenja. Analizom apsolutnih i relativnih pogrešaka utvrđena je visoka preciznost algoritama.

Najvažniji zaključak proizlazi iz analize funkcija cilja, gdje se pokazalo da se funkcija cilja ESK brže približava optimalnom rješenju u usporedbi s funkcijom cilja SK. ESK se također pokazao efikasnijim u bržem pronalaženju boljih rješenja i pružio bolja rješenja u prosjeku.

Ovaj rad pridonosi razumijevanju učinkovitosti SK i ESK algoritama u optimizaciji problema raspoređivanja vozila s kapacitetom.

Ključne riječi

simulirano kaljenje, empirijsko simulirano kaljenje, problem usmjeravanja vozila ograničenog kapaciteta

Solving capacitated vehicle routing problem using simulated annealing

Summary

In today's modern society, optimizing logistic processes plays a pivotal role in the economy and sustainability. One of the challenges in logistics is the capacitated vehicle routing problem, where trucks are used to deliver cargo to various locations. This problem requires the optimal allocation of trucks to customers to minimize the traveled distance of vehicles.

The paper begins by defining the capacited vehicle routing problem and provides a mathematical model for the problem. It then introduces the simulated annealing (SA) algorithm and the empirical-type simulated annealing (ETSA) algorithm as methods for solving this problem. SA employs an exponential distribution for accepting worse solutions, while ETSA utilizes a gamma distribution.

Subsequently, the paper extensively describes the implementation of these algorithms and their application to the capacited vehicle routing problem. Different components of the implementation are presented, including the generation of initial solutions and candidates, as well as the SA algorithm itself.

In the third part of the paper, the results obtained by applying these algorithms to various datasets (A, B, P) are analyzed. Optimal solutions are compared with solutions obtained using SA and ETSA. The results demonstrate that both algorithms are capable of providing satisfactory solutions, with several instances achieving optimal solutions. The analysis of absolute and relative errors reveals a high level of precision in the algorithms.

The most significant conclusion arises from the analysis of the objective functions, showing that the objective function of ETSA converges more quickly to the optimal solution compared to that of SA. ETSA has also proven to be more efficient in finding better solutions more rapidly and has provided better solutions on average.

This paper contributes to the understanding of the effectiveness of SA and ETSA algorithms in optimizing the vehicle routing problem with capacity constraints.

Keywords

simulated annealing, empirical-type simulated annealing, capacitated vehicle routing problem

Životopis

Rođen sam 17. svibnja 1999. godine u Osijeku. Pohađao sam Osnovnu školu Miroslava Krleže u Čepinu. Nakon osnove škole upisujem III. gimnaziju u Osijeku. Nakon završene srednje škole, odlučio sam se za preddiplomski studij Matematika i računarstvo na Odjelu za matematiku u Osijeku. Godine 2021. stječem naziv sveučilišni prvostupnik matematike i računarstva. Iste godine se upisujem, isto na Odjelu za matematiku u Osijeku, na sveučilišni diplomski studij matematike, smjer matematika i računarstvo.