Metaheurističke metode 5.nedelja

November 19, 2024

Simulirano kaljenje
Simulated annealing

- S-metaheuristika inspirisana procesom kaljenja čelika.
- Zasnovana na lokalnom pretraživanju koja dozvoljava posećivanje lošijih rešenja u cilju nalaženja globalnog optimuma.

Simulirano kaljenje - Simulated annealing

- Metoda započinje izborom početnog rešenja (na slučajan način ili nekom heuristikom) i postavljanjem parametra koji predstavlja temperaturu na relativno visoku vrednost T_0
- U svakoj iteraciji na slučajan način se bira sused x_0 iz (unapred odredjene) okoline tekućeg rešenja x. Ako je izabrano rešenje x_0 bolje od x (u pogledu vrednosti funkcije cilja), ono postaje tekuće i nastavlja se sa pretraživanjem njegove okoline. Ukoliko je lošije od tekućeg, x_0 se prihvata sa odredjenom verovatnoćom.
- Verovatnoća prihvatanja lošijeg rešenja obično zavisi od temperature, kao i stepena degradacije lošijeg rešenja, i opada sa povećanjem broja iteracija algoritma.
- Pri kraju izvrsavanja verovatnoća prihvatanja lošijeg resenja je minimalna, tj. vrlo verovatno se neće prihavtiti lošije rešenje jer se smatra da je već dostignut odredjen kvalitet rešenja i na ovaj način se izbegava moguće pogoršavanje tekućeg rešenja. Jedan od mogućih načina definisanja verovatnoće je: $\rho(T) = e^{-\delta/T},\ \delta = f(x') - f(x)$
- Za svaku vrednost temperature, izvršava se nekoliko iteracija. Nakon odredjenog broja iteracija vrši se snižavanje temperature, čime se smanjuje verovatnoća prihvatanja lošijeg rešenja kako pretraga napreduje.

Simulirano kaljenje -Simulated annealing

Algorithm 1 SA

$x = x_0$ generisanje inicijalnog rešenja
$x\_best = x_0$
$T = T_0$
while $T > T_f$ **do**
  for $i = 1 : iter$ **do**
    $x^* =$ Nacumično resenje u okolini telućeg x
    $\delta = f(x^*) - f(x)$
    if $\delta <= 0$ **then**
      $x = x^*$
    else
      x^* se prihvata sa verovatnoćom $\rho(T)$
    end if
    if $f(x^*) < f(x\_best)$ **then**
      $x\_best = x^*$
    end if
  end for
  $T = T * \alpha$
end while

Tabu pretraga-Tabu Search

- S-metaheuristika zasnovana na lokalnom pretraživanju
- Koristi memoriju procesa pretrage u cilju izbegavanja zamke lokalnog optimuma
- Tabu lista- kratkoročna memorija procesa pretrage u kojoj se skladište informacije o lokalnim optimumima, ili o potezima koji vode u nedavno posećene lokalne optimume
- Dužina tabu liste je ključni parametar ove metaheuristike i najčešće je konstantne dužine.

Tabu pretraga-Tabu Search

- Osnovna varijanta tabu pretraživanja započinje generisanjem početnog rešenja i lokalnom pretragom u njegovoj okolini.
- Nakon toga pronadjeni lokalni optimum se ubacuje u tabu listu i u odredjenom broju narednih iteracija biva isključen iz okoline
- U medjuvremenu se nastavlja sa lokalnim pretraživanjem tako modifikovane okoline i daljim kreiranjem tabu liste.
- Svaki element tabu liste se nakon odredjenog broja iteracija (tzv. tabu vreme) oslobadja iz tabu liste i ponovo uključuje u proces pretrage.

TS Algoritam

Algorithm 2 TS

$x = x_0$ generisanje inicijalnog rešenja
$x\_best = x_0$
$TL = \{\}$
while nije ispunjen kriterijum zaustavljanja **do**
    $x' =$ Naći najbolje rešenje u okolini telućeg (x, TL)
    if $f(x') < f(x\_best)$ **then**
        $x\_best = x'$
    end if
    $TL = TL \cup x'$
    if $|TL| > r$ **then**
        $TL = TL \setminus x_r$
    end if
    $x = x'$
end while
return x_best

TS

- Čuvanja celokupnog rešenja je memorijski zahtevno, pa se umesto toga često čuvaju potezi koji dovode do zabranjenih rešenja.
- Medjutim, na ovaj način se dešava da budu zabranjena sva rešenja koja imaju istu osobinu kao i ono rešenje koje je zaista zabranjeno, čime se onemogućava pretraživanje čitavog prostora.
- Loš izbor dužine tabu liste može dovesti do pojačane diverzifikacije (predugačka tabu lista) i pojave ciklusa (suviše mala tabu lista).
- Zato se uvode liste promenljive dužine kao i aspiracijske funkcije kojom se proverava prag značajnosti nekog rešenja. Najčešće korišćen aspiracijski kriterijum ukida tabu status onih poteza koji vode do rešenja koja imaju bolju vrednost funkcije cilja od vrednosti funkcije cilja najboljeg posećenog rešenja.
- TS metoda ima veliki broj parametara koje treba adekvatno izabrati.
- Modifikacija: osim kratkoročne memorije (Tabu Lista) koriste se i srednjeročna, odnosno dugoročna memorija

SA vs TS

- TS koristi memorijske strukture u procesu pretraživanja, sto je odsutno kod SA metode
- SA metoda u svakoj iteraciji nasumično bira rešenje iz okoline tekućeg rešenja, dok TS pretražuje celu okolinu (ili veći deo) da bi identifikovao rešenja visokog kvaliteta
- SA lako nalazi kvalitetne delove pretraživačkog prostora, često se koristi hibridizacija u kojoj SA generiše početno rešenje TS metode

Problem rutiranja vozila sa ograničenim kapacitetima (Capacitated Vehicle Routing Problem - CVRP)

- Dato je n klijenata koje treba uslužiti. Svaki klijent ima potražnju $d_i, i = 1, 2, ..., n$ za odredjenom robom koja se nalazi u skladištu s.
- Na raspolaganju je m vozila jednakih kapaciteta Q.
- Zadatak je odrediti rutu svakog vozila tako da ukupni transportni trošak bude minimalan, poštujući sledeća ograničenja:
  1. Svaki klijent mora biti uslužen od strane tačno jednog vozila.
  2. Suma potreba svih korisnika koje uslužuje isto vozilo ne sme biti veća od Q.
  3. Svako vozilo svoju rutu započinje i završva u skladištu s.

CVRP Primer dopustivog rešenja

Na slici je dat je primer dopustivog rešenja instance koja ima 7 korisnika, 3 vozila i kapacitet svakog vozila je 50. Skladište je locirano u nultom čvoru i ono predstavlja početnu i krajnju tačku svake rute. Broj koji se nalazi pored svakog čvora (korisnika) označava njegovu potrebu.

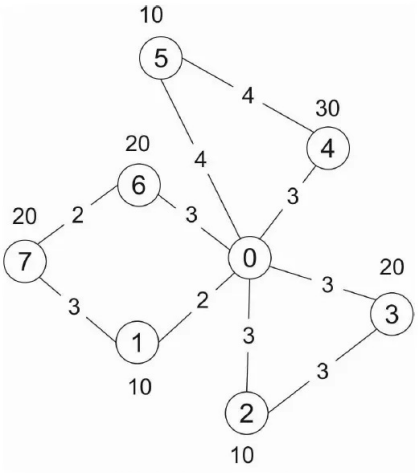


Figure 1: $n = 7, Q = 50$

CVRP Model

- $G = (V, E)$ je kompletan usmeren graf sa skupom čvorova $V = \{0, 1, 2, ..., n\}$ i skupom grana $E = \{(i, j) : i, j \in V, i \neq j\}$.
- c_{ij} je nenegativna težina koja predstavlja rastojanje (transportni trošak) od korisnika i do korisnika j.
- Matrica rastojanja je simetrična, tj. važi $c_{ij} = c_{ji},\ \forall i, j \in V, i \neq j$ i zadovoljava nejednakost trougla, $c_{ij} + c_{jk} \geq c_{ik}, \forall i, j, k \in V$.
- Skladište je predstavljeno nultim čvorom.
- Svakom korisniku $i \in V$ pridružena je težina d_i koja predstvalja njegovu potrebu za odredjenom robom. Budući da nulti čvor predstavlja skaldište ($d_0 = 0$).
- Na raspolaganju je p vozila kapaciteta Q
- Cilj je naći optimalnu rutu svakog vozila, tj. m ruta kojima se minimizuje ukupni transportni trošak pri čemu uslovi 1,2. i 3. moraju biti ispunjeni.
- Uvodimo promenljive:

$$x_{rij} = \begin{cases} 1, \text{grana } (i, j) \text{ je deo rute vozila } r \\ 0, \text{inače} \end{cases}$$

CVRP Model

$$\min \sum_{r=1}^{p} \sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} c_{ij} x_{rij},$$

$$\sum_{r=1}^{p} \sum_{i=0, i\neq j}^{n} x_{rij} = 1, \ j = \overline{1,n},$$ (Svakog korisnika posećuje tačno jedno vozilo.)

$$\sum_{j=1}^{n} x_{r0j} = 1, \ r = \overline{1,p},$$ (Svakom vozilu dozvoljeno je da samo jednom napusti skladište.)

$$\sum_{i=0, i\neq j}^{n} x_{rij} = \sum_{i=0}^{n} x_{rji}, \ \forall j = \overline{1,n}, r = \overline{1,p},$$ (Broj vozila koji ulazi u čvor jedank broju vozila koji izlazi iz nj

$$\sum_{i=0}^{n} \sum_{j=1, j\neq i} d_j x_{rij} \leq Q, \ r = \overline{1,p},$$ (Kapacitet vozila ne sme biti prekoračen.)

$$\sum_{r=1}^{p} \sum_{i\in S} \sum_{j\in S, i\neq j} x_{rij} \leq |S| - 1, \ \forall S \subseteq \{1, \ldots, n\},$$ (Eleminišu se sve podture koje ne obilaze skladište.)

$$x_{rij} \in \{0, 1\}, \ r = \overline{1,p}, \ j = \overline{1,n}, \ i \neq j$$

CVRP-Reprezentacija rešenja

Rešenje je predstvaljeno vektorom dužina $n + p$ oblika:

$$[r_1, k_{i1}, k_{i2}, ..., k_{ij}, r_2, k_{ij+1}, k_{ij+2}, ..., r_p, ...k_{in}]$$

Nakon svakog vozila (r) sledi niz korisnika koje to vozilo treba posetiti. U implementaciji se na mestu vozila nalazi 0 i ona označava početak rute. Koristeći ovu reprezentaciju, dopustivo rešenje sa slike bilo bi predstavljeno vektorom $[0\ 4\ 5\ 0\ 6\ 7\ 1\ 0\ 2\ 3]$.

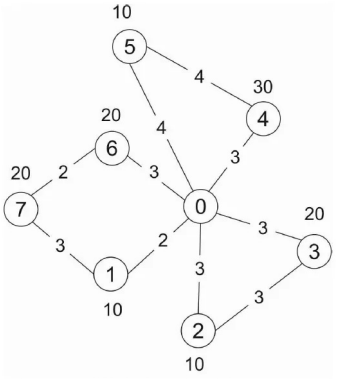


Figure 1: $n = 7, Q = 50$

CVRP Generisanje inicijalnog rešenje

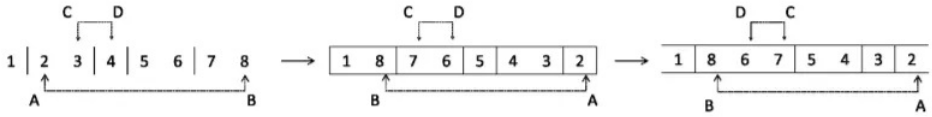
Algorithm 3 CVRP Initial solution

rute=zeros(p,n);
$L = sort\_index(d)$
for i=1:p **do**
    Dodaj korisnika $L(i)$ u rutu i
end for
for j=p+1:n **do**
    for i=1:p **do**
        if kapacitet rute i + kapacitet korisnika $j \leq Q$ **then**
            dodaj korisnika j u rutu i
            break
        end if
    end for
end for
x = transformisi_matricu_u_vektor(rute);

Okolina za SA

Za pretraživanje prostora korišćena je SIMPGEN transformacija rešenja. U pitanju je transformacija koja se sprovodi na sledeći način:

1. Izabrati dve spoljne pozicije A i B,
2. Invertovati podlistu od A do B,
3. Izabrati dve unutrašnje pozicije C i D (izmedju A i B),
4. Invertovati podlistu od C do D.



U svakoj iteraciji SA metode, nasumično se generiše rešenje u SIMGEN okolini tekućeg, tj. nasumično su odredjene 4 pozicije, A,B,C i D, i primenjena je SIMPGEN transformacija na tekuće rešenje.

Okolina za TS

Specijalni slučajevi SIMPGEN trasnformacije:

1. Umetanje - A $=$ C , D i B su na uzastopnim pozicijama,
2. Zamena - A, C i D, B su na uzastopnim pozicijama,
3. Invertovanje C $=$ D, C $\neq$ A, D $\neq$ B.

Budući da TS metoda pretražuje celu okolinu tekućeg rešenja (ili njen veći deo), umesto SIMPGEN okoline u TS metodi je korišćena njena restrikcija na okoline definisane operatorima umetanja/zamene/invertovanja, tj. korisniku je omogućeno da bira koji će od ova tri operatora koristiti.

Instance

Instance na kojima testiramo algoritme su u formatu:

$$
\begin{array}{l}
n \\
p \\
Q \\
1, x_1, y_1 \\
2, x_2, y_2 \\
\vdots \\
n, x_n, y_n \\
1, 0 \\
2, d_2 \\
\vdots \\
n, d_n
\end{array}
$$

gde (x_i, y_i) predstavlja lokaciju korisnika i, a d_i njegovu potrebu. Prva lokacija predstavlja lokaciju skladišta.

Questions?