Vehicle Routing Problem

Projekat iz oblasti računarske inteligencije

Anja Cvetković
Stefan Jevtić

Školska 2023/2024 godina

1 Uvod

Vehicle Routing Problem (skraćeno VRP) je termin koji se koristi za probleme uspostavljanja ruta sa skup vozila na osnovu jednog ili više skladišta i mušterija do kojih treba da se izvrši dostava dobara. Ovo je veliki problem u distribuciji i logistici i danas, a prvi put je opisan 1959. godine, od strane Danciga i Ramsera. Može se posmatrati kao generalizacijom problema putujućeg trgovca, gde se najčešće vrši minimizacija pređenog puta, a u današnje vreme vrši se i minimizacija potrošnje goriva ili vremena.

2 Klasični VRP

Neka je $X = \{x_i | i = 1, ..., N\}$ skup od N mušterija i neka je x_0 skladište. Skup $V = \{v_j | j = 1, ..., M\}$ je skup vozila pozicioniranih u skladištu. Posmatrani vremenski period je T.

Svaki x_i može imati naredne zahteve:

1. Količinu q_i proizvoda za dostavu
2. Vreme u_i za koje se zahteva da vozilo isporuči robu
3. Prioritet δ_i

Vozilo v_j ima naredne karakteristike:

1. Maksimalni kapacitet tovara O_j
2. Period rada od trenutka T_j^s to trenutka T_j^k

3. Fiksirana cena C_k

Pretpostavićemo da je data cena najkraćeg puta između mušterija x_i i x_j sa c_{ij} i korespondirajuće vreme t_{ij}.

Cilj

Dizajnirati rute najmanje cene za svako vozilo tako da:

1. Svaka mušterija biva posećena samo jednom i to od strane tačno jednog vozila
2. Svaka ruta počinje i završava se u skladištu
3. Zadovoljavanje dodatnih ograničenja

3 Vehicle Routing Problem With Time Windows

VRPTW definišemo skupom vozila V) i skupom musterija C, $|C| = n$. U ovom radu skup vozila se tretira kao homogen - vozila su identična sa zadatim kapacitetom q. Svaku mušteriju i karakteriše količina traženih proizvoda d_i) kao i vremenski interval $[a_i, b_i]$ prijema dobara. Vozilo mora da poseti mušteriju pre b_i, a u slučaju dolaska pre a_i mora da čeka.

Svaka ruta počinje i završava se u istom depou, tačnije u čvoru 0 i čvoru $n + 1$ koji su ekvivalentni, a njena cena je ekvivalenta vremenu da se ona obiđe uz dodatna uračunata zadržavanja u svakom čvoru. Vreme da se od mušterije i dođe do mušterije j, u oznaci t_{ij} ekvivalento je euklidoskom rastojanju izmedju ta dva čvora - što je u skladu sa Solomonovim skupom podataka koji su i korišćeni u ovom radu, te ćemo uvesti oznaku c_{ij} kao cenu puta između čvorova i i j.

3.1 Matematički model

Definisimo promenljivu x_{ijk} za par čvorova i i j, gde je $i \neq j$, $i \neq n + 1$, $j \neq 0$, i vozilo k kao:

$$x_{ijk} = \begin{cases} 1, \textit{ ako vozilo k ide direktno od čvora i do j}, \\ \\ 0, \textit{ inače} \end{cases}$$

Promenljiva s_{ik} se definiše za svako vozilo k i svaki čvor c_i kao vreme kada je vozilo k počelo da servisira mušteriju u čvoru i. U slučaju da vozilo k ne obilazi čvor i onda je ova vrednost nerelevantna.

$$\min \sum_{k \in V} \sum_{i \in C} \sum_{j \in C} c_{ij} x_{ijk} \tag{1}$$

$$\sum_{k \in V} \sum_{j \in C} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{i \in C} d_i \sum_{j \in C} x_{ijk} \leq q \qquad \forall k \in V \tag{3}$$

$$\sum_{j \in C} x_{0jk} = 1 \qquad \forall k \in V \tag{4}$$

$$\sum_{i \in C} x_{ihk} - \sum_{j \in C} x_{hjk} = 0 \qquad \forall h \in C, \forall k \in V \tag{5}$$

$$\sum_{i \in C} x_{i(n+1)k} = 1 \qquad \forall k \in V \tag{6}$$

$$x_{ijk}(s_{ik} + t_{ij} - s_{jk}) \leq 0 \qquad \forall i, j \in C, \forall k \in V \tag{7}$$

$$a_i \leq s_{ik} \leq b_i \qquad \forall i \in C, \forall k \in V \tag{8}$$

Funkcija cilja (1) minimizuje ukupnu cenu puta. Ograničenje (2) obezbeđuje da svaki čvor bude posećen tačno jednom i (3) označava da tovar svakog vozila ne prelazi maksimalan kapasitet q. Dalje, (4), (5) i (6) obezbeđuju da svaka ruta počinje i završava se u depou i nakon istovara robe u jednom čvoru, vozilo mora da nastavi ka sledećem. Nejednakost (7) govori o redosledu obilaska rute, a (8) potvrđuje poštovanje time window ograničenja. Uočimo da se vozilo koje je neiskorišćeno može modelovati rutom (0, n+1).
Može se uvesti i gornja granica broja vozila:

$$\sum_{k \in V} \sum_{j \in C} x_{0jk} \leq |V| \tag{9}$$

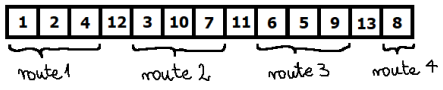
4 Genetski algoritam

Principi genetskih algoritama (GA) su već dobro poznati. Održava se populacija rešenja, na osnovu kojih se pomoću operacija selekcija odabiraju jedinke koje će ostaviti potomstvo. Potomci imaju osobine

oba roditelja. Fitness funkcija odogovara funkciji cilja, što je u ovom slučaju ukupno vreme da se obiđu sve rute. Analogno biološkim procesima, jedinke koje su bolje prilagođene imaju veće šanse da prežive i ostave potomstvo, sa ciljem da se kroz imitaciju evolucije pronađu jedinke sa boljom vrednošću fitness funkcije.

4.1 Reprezentacija hromozoma

Bazična struktura jedne rute može se konstruisati kao lista celobrojnih vrednosti koja počinje i završava se nulom (depo) a između se navode čvoroi koji se obilaze u zadatom redosledu. Strukture svih ruta možemo nadovezati da bi dobilni finalnu strukturu, a nule se mogu izbaciti radi povećanja čitljivosti. Kao graničnike između pojedinalnih ruta mogu da se umetnu oznake vozila. Naime, ako čvorove koji predstavljaju mušterije označimo vrednostima od 1 do n, a pritom imamo m vozila na raspolaganju, vozila ćemo označavati vrednostima od n+1 do n+m, s tim što se oznaka vozila za poslednju rutu može izostaviti.



Slika 1: Reprezentacija hromozoma

4.2 Inicijalna populacija

Prvi korak genetskog algoritma jeste generisanje inicijalne populacije dopustivih rešenja. Korišćen je modifikovana nearest neighbor heuristika sa paramterom randomizacije, koji omogućava kreiranje i nasumičnih dopustivih rešenja.

Algorithm 1 Inicijalizacija hromozoma

```
1: Create remaining customers list from available customers
2: routes = []
3: last visited = Depot
4: current route = []
5: while Remaining customers list is not empty do
6:     Create list of feasible customers and sort it according to distance from
   last visited customer
7:     if List of feasible customers is empty then
8:         last visited = Depot
9:         Append current route to routes list
10:        current route = []
11:        go to 5
12:    end if
13:    if random[0, 1] ≤ randomizing parameter then
14:        Choose a random feasible customer and add it to current route
15:    else
16:        Choose a nearest customer from feasible customer list and add it to
   current route
17:    end if
18: end while
```

4.3 Operatori selekcije

Operator selekcije bira jedinke iz populacije koje će ostaviti potomstvo. Navešćemo algoritme koji su korišćeni.

Random selekcija predstavlja odabir nasumičnog pojedinca iz populacije.

Turnirska selekcija bira uzorak iz populacije, a zatim se pojedinac sa najboljom vrednosšću fitness funkcije bira iz tog uzorka.

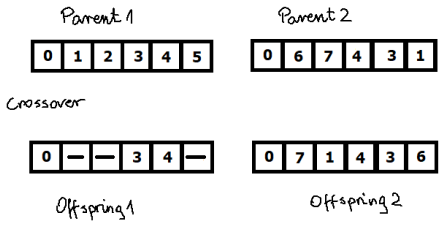
Ruletska selekcija podrazumeva izračunavanje proporcije fitness vrednosti za svakog pojedinca, a zatim se odabira pojedinac na osnovu tih proporcija.

Rang selekcija je slučna ruletskoj, ali umesto da zasniva verovatnoću odabira na osnovu fitness vrednosti, zasniva se na rangiranju pojedinaca.

4.4 Operatori ukrštanja

Order Crossover (OX) je specifično dizajniran za korišćenje nad listom bez duplikata. Prvo, biraju se dve nasumične pozicije iz roditelja i kopira se odgovarajuća podniska u dete, između izabranih

pozicija. Nakon toga, prolaskom kroz drugog roditelja kopiramo odgovarajuće elemente u dete, po redu, ako već nisu prisutni.



Slika 2: Order Crossover

Partially Mapped Crossover (PMX) deca su prvobitno duboke kopije svojih roditelja. Dve pozicije se nasumično biraju i odgovarajuće podniske se razmenjuju. Elementi na odgovarajućim pozicijama se mapiraju, uklanjajući cikluse i tranzitivnost u mapiranjima. Na primer, mapiranja $1 \rightarrow 2$ i $2 \rightarrow 1$ se ignorišu, a $1 \rightarrow 2$ i $2 \rightarrow 3$ se posmatraju kao $1 \rightarrow 3$. Nakon normalizacije mapiranja, elementi iz dece se razmenjuju prema mapiranjima. Na primer, ako je a $\rightarrow$ b, element a iz prvog deteta se zamenjuje elementom b iz drugog deteta, van izabrane podniske.

Best Route Better Adjustment (BRBAX) kopira polovinu najboljih ruta iz prvog roditelja u jednog potomka i popunjava preostale rute iz drugog roditelja, zadržavajući pozicije vozila fiksne unutar deteta.

4.5 Operatori mutacije

Swap mutacija nasumično bira dve pozicije u jedinki i zamenjuje elemente na tim pozicijama.

Mutacija inverzije nasumično bira podnisku unutar jedinke i menja redosled elemenata u toj podniski.

Shaking mutacija nasumično bira podnisku unutar jedinke i meša redosled elemenata u toj podniski.

4.6 Rezultati

Uz kombinovanje različitih operatora, nad Solomonovom R101 skupu podataka, ustanovljenoj je da najbolju vrednost fitness funk-

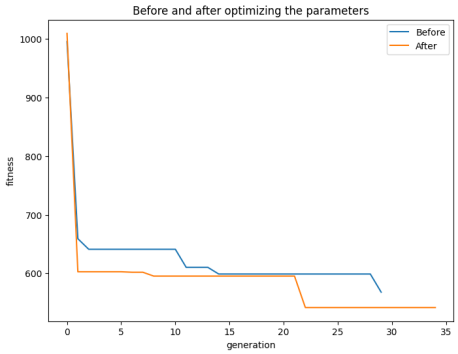
cije daju naredni operatori:

$$\text{Operator selekcije} = \text{Random selekcija}$$
$$\text{Operator ukrštanja} = \text{Best Route Better Adjustment}$$
$$\text{Operator mutacije} = \text{Swap mutacija}$$

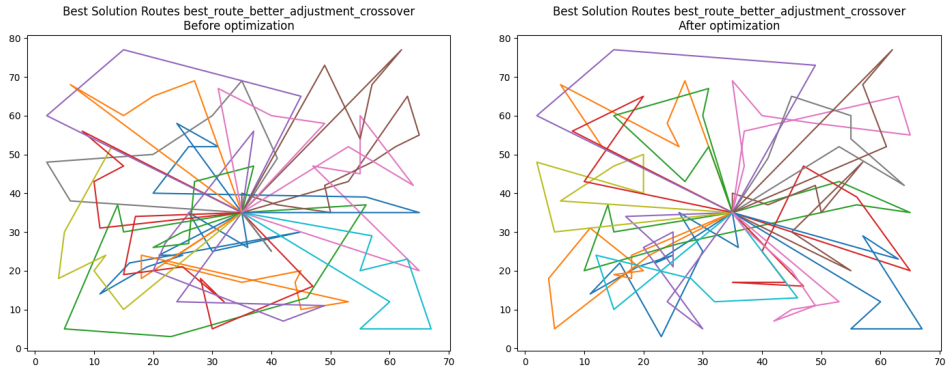
Za ovu kombinaciju operatora pronađene su najbolje vrednosti parametara genetskog algoritma:

$$\text{Veličina populacije} = 700$$
$$\text{Broj generacija} = 35$$
$$\text{Veličina elitizma} = 70$$

Za ovakve parametre dobijena najbolja vrednost normalizovane fitness funkcije u odnosu na dužinu pojedinačnih ruta je 541.695.



Slika 3: Fitness najbolje jedinke kroz generacije



Slika 4: Rute pre i posle optimizacije parametara

5 Reference

Kallehauge, B., Larsen, J., Madsen, O.B., Solomon, M.M. (2005). Vehicle Routing Problem with Time Windows. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds) Column Generation. Springer, Boston, MA.

Selected Genetic Algorithms for Vehicle Routing Problem Solving - Scientific Figure on ResearchGate.

VRPTW Benchmark Problems: http://web.cba.neu.edu/ msolomon/problems.htm