FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVOD ZA TELEKOMUNIKACIJE

HEURISTIČKE METODE OPTIMIZACIJA

Parkiranje vozila u spremištu javnog prijevoznika

Tomislav Božurić

Marin Vernier

Siječanj, 2019

1 Opis problema

U svrhu ovog projekta riješavan je problem parkiranja vozila u spremištima kompanije za javni prijevoz. Svako od vozila se smješta na prometne trake i pritom moramo imati na umu vrijeme polaska vozila iz spremišta, kako bi svako vozilo na vrijeme moglo izaći iz spremišta. Sam problem definira određena ograničenja koja je bilo potrebno zadovoljiti:

- Vozilo se parkira na točno jednu traku.
- Na svaku traku se parkiraju vozila isključivo jedne serije.
- Svako vozilo se može parkirati isključivo na neku od traka koja pruža potrebnu infrastrukturu.
- Suma duljina vozila na traci ne smije premašiti kapcitet trake.
- Vozilu se može dodijeliti samo jedna pozicija u traci.
- Pozicija u redu u traci može biti dodjeljena samo jednom vozilu.
- Polazak bilo kojeg vozila u traci mora biti prije polaska vozila koje slijedi u traci.
- Polazak svih vozila u blokirajućoj traci mora biti prije prvog vozila u blokiranoj traci.

  Sam problem ima dva globalna cilja, prvi globalni cilj jest **minimizirati** broj različitih serija vozila u susjednim trakama, broj korištenih traka i neiskorišten prostor na korištenim trakama. Taj globalni cilj ignorira vremensku komponentu problema te ga opsijemo kao:

  $$\max p_1 f_1 + p_2 f_2 + p_3 f_3.$$

  Drugi globalni cilj jest **maksimizirati** broj vozila s istim tipom rasporeda unutar trake, broj vozila s istim tipom rasporeda u susjednim trakama, te maksimizirati dorbu praksu glede vremenskog razmaka između dva izlaska vozila iz spremišta (idealno svaki izlazak odvojen između $[15, 20]$ minuta, najmanje je prihvaljivo da su izlazci razmaknuti manje od 10 minuta). Ovaj maksimizacijski problem možemo opisati sljedećom funkcijom:

  $$\max r_1 g_1 + r_2 g_2 + r_3 g_3$$

Cilj	*Vrijednost*	*Težinski faktor*
f_1	broj parova susjednih korištenih traka koje se razlikuju u parkiranoj seriji vozila	$\frac{1}{p_1}$ = broj korištenih traka -1
f_2	broj korištenih traka	$\frac{1}{p_2}$ = ukupni broj traka
f_3	suma neiskorištenog kapacitena na korištenim trakama	$\frac{1}{p_3}$ = ukupni kapacitet svih traka - ukupna duljina svih vozila
g_1	broj parova susjednih vozila u traci s istim tipom rasporeda	$\frac{1}{r_1}$ = ukupni broj vozila u svim trakama - broj korištenih traka
g_2	broj susjednih korištenih traka za koje vrijedi da zadnje vozilo u traci ima raspored istog tipa kao prvo vozilo u sljedećoj traci	$\frac{1}{r_2}$ = broj korištenih traka -1
g_3	suma nagrada i penala za vremenski razmak između polazaka, za sva susjedna vozila, u svim trakama $$n = \begin{cases} 15, & 10 \leq vr \leq 20 \\ 10, & vr > 20 \\ 4 \cdot (10 - vr), & vr < 10 \end{cases} \quad (1)$$	$\frac{1}{r_3}$ = 15 $\cdot$ broj evaluiranih parova (susjeda iz traka)

Table 1: Opis komponenti funkcija

2 Opis primijenjenog algoritma

U svrhu rješavanja ovog projekta korištena su tri algoritma. Pohlepni algoritam, eliminacijski genetski algoritam s turnirskom selekcijom, te taboo pretraživanje u svrhu lokalnog pretraživanje nakon pronađenog rješenja koje zadovoljava gore navedena ograničenja. Na početku pomoću pohlepnog algoritma opisanog u idućem odjeljku pokušavamo razmjestiti što je više moguće vozila u trake, zatim na temleju tako dobivenog rješenja pomoću genetskog algoritma gradimo populaciju jedinku jednostavnim permutacijama, te nakon što genetskim algortmom nađemo rješenje koje zadovoljava sva ograničenja, taboo pretraživanjem lokalno pretražujemo za još bolja rješenja. Korišten je genetski eliminacijski algoritam, kao prikaz rješenja koristi se vektor permutacija cijelih nenegativnih brojeva koji označavaju parkirnu traku na koje je pojedino vozilo parkirano. Primjerice moramo li parkirati 50 vozila u spremište javnog prijevoznika, taj vektor ima 50 elemenata i svaki elemenet onačava na kojoj traci je parkirano vozilo koje je označeno indeksom u vektoru. Kao selekcija koristi se jednostavna 3-turnirska selekcija gdje se najgora jedinka eliminira iz populacije, a preostale dvije se križaju te se naposljetku dijete mutira i dodaje u populaciju. Odabrana je 3-turnirska selekcija jer se pokazala da se dobro ponaša tijekom postupka traženja rješenja i omjer preživljavanja dobrih i loših jedinki je zadovoljavjuć. Kao metode križanja implementirane su: križanje s jednom točkom prekida, križanje s više točaka prekida, te višesturko križanje kojem preko parametra predamo željeni broj elemenata, i na temelju tog broja nasumično generiramo indekse i prilikom kreiranja nove jednike na tim indeksima uzimamo vrijednosti od drugog roditelja, dok na preostalim indeksima elemente prvog roditelja. Za mutaciju se koristi jednostavna mutacija gdje s vjerojatnossću mutacije jedinke p_m biramo nasumično element jedinke, i mijenjamo ga nekim drugom elementom, odnosno nekom drugom parkirnom trakom. Dodatno, implementirana je i metoda unifomrne mutacije, gdje iteriramo po svim elementima jedinke i s određenom vjerojatnošću uzimamo element prvog roditelja ili uzimamo element drugog roditelja. U samome radu koriste se obje vrste mutacije i svaku biramo s vjerojatnošću 0.5. Ekvivalentno, za križanje koristimo višestruko križanje s različitim brojem elemenata koje želimo nasljediti od drugog roditelja. Kad genetski algoritam naše validno rješenje, on ga predaje taboo algoritmu, koji ga koristi kao početno rješenje. Početno rješenje postavljamo kao najbolje i kao trenutno rješnje. Za trenutno rješenje generiramo susjedstvo, na način da radimo permutacije svakog vozila sa svakim. Dodatno, susjedstvno čine i rasporedi gdje premještamo vozila u prazne trake. U

svakoj iteraciji evaluiramo sve tako generirane susjede i tražimo najboljeg poboljšavajućeg susjeda. Kad nađemo boljeg susjeda odabiremo njega kao trenutno rješenje i pamtimo tu izmjenu u taboo listu. Algoritam staje nakon određenog broja iteracija. Detalji oko pojedinih heurističkih odosno evolucijskih algoritama dani su u nastavku u tablicama.

Table 2: Taboo pretraživanje

Prikaz rješenja	mapa u kojoj za svaku traku imamo listu vozila
Funkcija cilja	evaluira se prema zadanim funkcijama cilja
Funkcija prikladnosti	evaluira se prema zadanim funkcijama cilja
Početno rješenje	rješenje koje je našao genetski algoritam
Trajanje tabua	8 iteracija
Generiranje susjedstva	sve permutacije vozila(uz popunjavanje praznih traka)
Kriterij zaustavljanja	50 iteracija

Table 3: Genetski algoritam

Prikaz rješenja	vektor permutacija parkirnih traka
Funkcija cilja	suma svih pogrešaka koje dijele rješenje od validnog
Funkcija prikladnosti	suma svih pogrešaka koje dijele rješenje od validnog
Početno rješenje	pohlepni algoritam i permutacije
Selekcija	3-turnirska selekcija
Križanje	uzimanje n elemenata druge jedinke
Mutacija	uniformna mutacija i zamjena parkirne trake vozilu
Vjerojatnost mutacije	$p_m = 0.03$
Veličina populacije	10
Kriterij zaustavljanja	vremensko ograničenje

3 Pseudokod

Pseudokod pohlepnog algoritma dan je u nastavku:

1. Grupiraj vozila po seriji kojoj pripadaju.
2. Za svaku seriju vozila izračunaj prosječan broj traka na koje se automobili iz te serije mogu parkirati.
3. Sortiraj rastuće serije vozila po prethodno izračunatoj mjeri.
4. Za svaku seriju iz prethodno sortirane liste dohvati vozila iz te serije i sortiraj ih rastuće po: vremenu polaska, broju parkinih traka na koje vozilo može parkirati, tipu rasporeda te zatim po identifikatoru.
5. Za svako tako sortirano vozilo dohvati parkirne trake na koje se vozilo može parkirati i sortiraj parkirne trake rastuće po broju blokirajućih traka, pa po slobodnom prostoru na pojedinoj parkirnoj traci.
6. Za tako sortiranu listu parkinih traka na svaku od njih pokušaj parkirati vozilo ako parkiran traka nije puna i ako se na njoj ne nalaze vozila neke druge serije.

Pseudokod genetskog eliminacijskog algoritma i taboo algoritma je klasični pristup, pa ga ovim putem ne navodim.

4 Rezultati

Iz rezultata prikazanih u nastavku možemo vidjeti kako algoritam uspješno pronalazi rješenja te parkira vozila u garažu bez kršenja ograničenja. Zbog same probabilističke prirode genetskih algoritama, vrijeme pronalska rješenja varira u razumljivim intervalima. Rad genetskog algoritma isproban je na mnogo parametara se pokazalo da najbolja rješenja pronalazi za malu vjerojatnost mutacije (otprilike $1 - 3\%$). Također se pokazalo da za veće veličine populacije nismo dobili na kvaliteti rješenja, pa samim time zbog većeg broj operacija koje je potrebno izvesti kod veće populacije, korištena je manja populacija koja je davala rješenja jednake ili bolje kvalitete u kraćem vremenu. Bilo je potrebno dosta pripaziti na operatore križanja i mutacije zbog vrlo ograničenog problema, pa samim time je razumljivo da vjerojatnost mutacije se očekuje da bude malena (kao što je i uobičajeno), a da operatorom križanja ne izgubimo dobar genetski materijal od roditelja

ili u drugoj krajnosti da nasumično pretražujemo. Također, zbog zadanih ograničenja genetskom algoritmu treba nešto više iteracija da evaluira k dobrim rješenjima, pa je bilo potrebno u svakom koraku pamtiti što je više prethodno izračunatih operacija kako nebi bespotrebno trošili procesorsko vrijeme.

Table 4: Rezultati

Broj minuta	nakon 1 minute	nakon 5 minuta	beskonačno
f_1			
f_2			
f_3			
g_1			
g_2			
g_3			
$cilj_1$			
$cilj_2$			
$omjer$			

5 Zaključak

Projekt se pokazao kao dosta izazovan i bilo je zanimljivo vidjeti kako u praksi izgleda rješavanje nekog tako ograničavajućeg problema. S obzirom da genetski algoritam se može vrlo različito ponašati za male promjene u parametrima, kao prijedlog jednog od daljnih poboljšanja bi bilo provođenje statistike za različite paremetre genetskog algoritma i uzimanjem najbolje prosječne vrijednosti odrediti optimalne parametre incijalno implementiranog genetskog algoritma. U svrhu dodatnog poboljšanja vremena potrebnog za pronalazak rješenja mogli bismo u više dretvi pokrenuti više instanci odnosno napraviti *thread-poll* te pratiti u određenim vremenskim razmacima napredovanje pojedine instance i onu najbolju ostaviti aktivnom.