Univezitet u Beogradu

Matematički fakultet

MASTER RAD

Rešavanje dvostepenog problema instalacije neograničenih kapaciteta primenom genetskog algoritma

Stefan Mišković

1072/2010

Beograd, oktobar 2011.

Mentor:	dr Miodrag Živković Matematički fakultet, Beograd
Članovi komisije:	dr Zorica Stanimirović Matematički fakultet, Beograd
	dr Vladimir Filipović Matematički fakultet, Beograd
Datum odbrane:	_____

Rezime

U radu je prikazan genetski algoritam za rešavanje dvostepenog problema instalacije neograničenih kapaciteta, koji predstavlja jedan od centralnih problema dizajna telekomunikacionih mreža. Imajući u vidu da je problem NP-težak, velika pažnja se posvećuje heurističkim metodama. Ovde je za rešavanje prvi put upotrebljen genetski algoritam. Za ulaze manjih dimenzija algoritam daje optimalna rešenja, kao i CPLEX rešavač. Pored toga, prvi put su rešeni neki slučajevi većih dimenzija. Implementirane su dve verzije genetskog algoritma, jedna koja kao operator selekcije koristi običnu turnirsku selekciju, a druga jednu njenu modifikaciju, fino gradiranu turnirsku selekciju.

Abstract

In this paper a genetic algorithm for solving two-stage uncapacitated facility location problem (TSUFLP) is presented. TSUFLP is one of the central problems in design of telecommunication networks. It is proven that TSUFLP is NP-hard, so main attention for solving this problem is paid to heuristic methods. Here, a genetic algorithm that solves TSUFLP is presented for the first time. For small size test instances, a comparison with the CPLEX solver is given and it is shown that algorithm gives optimal solutions for this instances. Also, the results for large scale instances are presented for the first time in the literature. Two versions of genetic algorithm are implemented — one that uses tournament selection as selection operator, and another one that uses its modification, the fine grained tournament selection.

SADRŽAJ

1. UVOD

U poslednje dve decenije zabeležen je nagli razvoj telekomunikacionih mreža, koje se danas koriste za prenos podataka, zatim audio i video signala, i tako dalje. Na veliki značaj ovih mreža svakako utiče i njihova primena na internetu.

Mnoga su pitanja koja se vezuju za sâm dizajn telekomunikacionih mreža. U ovom radu je razmatran dvostepeni problem instalacije neograničenih kapaciteta, koji predstavlja jedan od ključnih problema njihovog dizajniranja. Imajući u vidu da su terminali mreže pridruženi koncentratorima, a koncentratori superkoncentratorima, u problemu se razmatra minimizacija cene kreiranja takve mreže, pri čemu su u ukupnu cenu uračunate cene pridruživanja terminala koncentratorima i koncentratora superkoncentratorima, odnosno cene instalacije izabranih koncentratora i superkoncentratora.

Razmatrani problem je NP-težak, pa su algoritmi koji nalaze optimalno rešenje u praksi, zbog njihove neefikasnosti, neprimenljivi. Zbog toga se pribegava heurističkim metodama. Rešenje koje je ovde predloženo se zasniva na primeni genetskog algoritma.

U poglavlju 2 se govori o NP-kompletnim problemima, pre svega definiciji NP-kompletnosti i nekim najpoznatijim problemima koji su NP-kompletni, a u poglavlju 3 o heurističkim metodama, značaju kombinatorne optimizacije u rešavanju problema za koje nije poznat efikasan algoritam, kao i raznim vrstama heuristika. U poglavlju 4 su navedene najbitnije činjenice koje se vezuju za pojam genetskog algoritma. U poglavlju 5 je formulisan problem koji se razmatra. U poglavljima 6 i 7 je opisan način rešavanja problema, koji se zasniva na primeni genetskog algoritma, pri čemu su, u cilju pronalaženja što boljih rezultata korišćene i pomoćne heuristike. Konačno, u poglavlju 8 dat je kratak kratak prikaz i analiza dobijenih rezultata. Doprinos rada i pravci daljeg rada razmatraju se u zaključku.

2. NP-KOMPLETNI PROBLEMI

2.1 Definicija NP-kompletnosti

Za algoritam se kaže da je efikasan ako je njegova vremenska složenost $O\big(P(n)\big)$, gde je $P(n)$ polinom od veličine problema n. Klasa svih problema koji se mogu rešiti efikasnim algoritmom označava se sa P (polinomijalno vreme). Postoji dosta problema za koje se ne zna efikasan algoritam. Za neke će se možda naći nekad polinomijalno rešenje, ali se veruje da se uopšte i ne mogu rešiti efikasno. Za te probleme se, dakle, ne zna da li se nalaze u klasi P. Posebnu grupu takvih problema čine tzv. NP-kompletni problemi. Uopšte, problem je nedeterministički polinomski, tj. u klasi NP, ako se u polinomijalnom vremenu može utvrditi da li predloženo rešenje pripada ili ne skupu rešenja datog problema. Problem je NP-kompletan ukoliko se svaki problem iz klase NP može na njega svesti.

Definicija 2.1: Deterministička Tjuringova mašina (ili samo Tjuringova mašina, DTM) je uređena sedmorka $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$, gde je:

1. Q konačan neprazan skup stanja;
2. Γ konačan neprazan skup simbola nad nekom azbukom;
3. $b \in \Gamma$ oznaka za prazan simbol;
4. $\Sigma \subseteq \Gamma \backslash \{b\}$ skup ulaznih simbola;
5. $q_0 \in Q$ početno stanje;
6. $F \subseteq Q$ skup završnih stanja;
7. $\delta: Q \backslash F \times \Gamma \to Q \times \Gamma \times \{L, R\}$ funkcija prelaza, gde je L levi, a D desni pomeraj.

Definicija 2.2: Nedeterministička Tjuringova mašina (NDTM) je uređena šestorka $M = (Q, \Gamma, i, b, F, \delta)$, gde je:

1. Q konačan neprazan skup stanja;
2. Γ konačan neprazan skup simbola nad nekom azbukom;
3. $i \in Q$ početno stanje;

4. $b \in \Gamma$ oznaka za prazan simbol;

5. $F \subseteq Q$ skup završnih stanja;

6. $\delta: Q \backslash (A \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ relacija prelaza, gde je L levi, a D desni pomeraj.

Razlika između determinističe i nedeterminističke Tjuringove mašine je u definiciji simbola δ. U slučaju determinističke, δ je funkcija, dok je u slučaju nedeterminističke relacija. Drugim rečima, kod DTM se može obavljati najviše jedna akcija u datom trenutku, dok kod NDTM to nije slučaj, odnosno dozvoljeno je da se u datom trenutku obavlja i više akcija.

Tjuringova mašina M prihvata reč $x \in \Sigma^*$ ako i samo ako, primenjena na ulaz x, daje izlaz „da“. Jezik L koji prepoznaje mašina M, u oznaci $L(M)$ se zadaje izrazom

$$L(M) = \{x \in \Sigma^* \mid M \text{ prihvata } x\}.$$

Definicija 2.3: Klasa P je skup svih jezika koji se mogu prepoznati determinističkom Tjuringovom mašinom u polinomijalnom vremenu. Drugim rečima,

$$P = \{L \mid L = L(M) \text{ za polinomijalnu determinističu Tjuringovu mašinu } M\},$$

gde je $L(M) = \{w \in \Sigma^* \mid M \text{ prihvata } w\}$ i M deterministička Tjuringova mašina za koju važe sledeća dva uslova:

1. M se zaustavlja za sve ulaze w;

2. postoji prirodan broj k takav da je $T_M(n) \in O(n^k)$, gde je

$$T_M(n) = \max\{t_M(w) \mid w \in \Sigma^*, |w| = n\}$$

i $t_M(w)$ potreban broj koraka da se mašina zaustavi za dati ulaz w.

Definicija 2.4: Klasa NP je skup jezika nad konačnom azbukom koji se mogu proveriti u polinomijalnom vremenu, gde se termin „proveriti“ definiše na sledeći način [18]. Neka je L jezik nad konačnim alfabetom Σ. Važi da je $L \in$ NP ako i samo ako postoji binarna relacija $R \subset \Sigma^* \times \Sigma^*$ i pozitivan broj k tako da važe sledeća dva uslova:

1. za sve $x \in \Sigma^*$ važi da je $x \in L$ ako i samo ako postoji $y \in \Sigma^*$ tako da važi $(x, y) \in R$ i $|y| \in O(|x|^k)$;

2. jezik $L_R = \{x\sigma y \mid (x, y) \in R\}$ nad $\Sigma \cup \{\sigma\}$ prihvata deterministička Tjuringova mašina u polinomijalnom vremenu.

Definicija 2.5: Neka su L_1 i L_2 dva jezika, podskupa za skupove ulaza U_1 i U_2. Jezik L_1 polinomijalno svodljiv na L_2 ako postoji algoritam polinomijalne vremenske složenosti koji

dati ulaz $u_1 \in U_1$ prevodi u ulaz $u_2 \in U_2$, tako da je $u_1 \in L_1$ ako i samo ako je $u_2 \in L_2$. Algoritam je polinomijalan u odnosu na veličinu ulaza u_1.

Definicija 2.6: Problem je NP-težak ako je svaki problem iz klase NP polinomijalno svodljiv na njega. Problem je NP-kompletan ako pripada klasi NP i ako je NP-težak.

Ako je neki problem u klasi P, onda trivijalno sledi da je on i u klasi NP. Ali, u klasi NP postoje neki problemi za koje se ne zna da li se nalaze i u klasi P; drugim rečima, nije poznato da li za njih postoji polinomijalni deterministički algoritam. Danas je pitanje jednakosti klasa P i NP najznačajniji problem teorijskog računarstva [19]. Ako bi bilo P = NP, onda bi to značilo da se svi problemi klasi NP mogu rešiti efikasnim algoritmom. Međutim, kako se veliki broj naučnika bavio ovim problemom, mada to formalno još niko nije pokazao, sa razlogom se veruje da je P ≠ NP. Ukoliko bi neko uspeo da dokaže ili opovrgne tvrdnju P = NP, rešio bi jedan od tzv. milenijumskih problema i osvojio nagradu od milion dolara [20].

Za sve NP-kompletne probleme je dokazano da su međusobno ekvivalentni, pa tako postoji efikasan algoritam za neki NP-kompletan problem ako i samo ako postoji efikasan algoritam za sve NP-kompletne probleme. Ako je bilo koji NP-kompletan problem u klasi P, tada bi važilo P = NP. Štaviše, to tvrđenje bi važilo i ako se za bilo koji NP-težak problem dokaže da pripada klasi P. Da bi se proverila NP-kompletnost nekog problema dovoljno je dokazati da pripada klasi NP i da se može polinomijalno svesti na neki postojeći NP-kompletan problem. Problem je NP-težak ako je „bar težak“ kao neki NP-kompletan problem.

2.2 Primeri nekih NP-kompletnih problema i načini rešavanja

Do ovog trenutka je pronađeno nekoliko hiljada NP-kompletnih problema. Njihov broj se stalno uvećava, ali još niko nije u mogućnosti da dokaže da li postoji neki polinomijalni algoritam za bilo koji od tih problema.

Problem 2.1 (problem zadovoljivosti): Neka je B Bulov izraz u konjuktivnoj normalnoj formi. Za Bulov izraz se kaže da je zadovoljiv ako postoji takvo dodeljivanje vrednosti 0 i 1 promenljivim da je njegova vrednost jednaka 1. Problem zadovoljivosti se sastoji u utvrđivanju da li je dati izraz zadovoljiv. Ovaj problem je NP-kompletan na osnovu Kukove teoreme, čiji se dokaz može pronaći u [21].

Problem 2.2 (pokrivač grana): Dat je neusmereni graf G. Pokrivač grana za G je skup čvorova takav da je svaka grana iz G susedna bar jednom od čvorova iz skupa. Za dati prirodan broj k, ovaj problem se sastoji u utvrđivanju da li postoji pokrivač grana čiji je broj čvorova manji ili jednak k [16].

Problem 2.3 (Hamiltonov put): Ustanoviti da li u grafu postoji Hamiltonov put. Hamiltonov put je put u grafu koji svaki čvor grafa sadrži tačno jednom [22].

Problem 2.4 (trodimenzionalno uparivanje): Neka su X, Y i Z tri disjunktna skupa sa po n elemenata i neka je $T \subseteq X \times Y \times Z$. Problem je ustanoviti da li postoji skup $M \subseteq T$ takav da je $|M| = n$ i za svaka dva elementa (x_1, y_1, z_1) i (x_2, y_2, z_2) iz M važi $x_1 \neq x_2$, $y_1 \neq y_2$ i $z_1 \neq z_2$ [23].

Problem 2.5[1]: Dat je kompletan graf $G = (V, E)$, funkcije $w: V \to \mathbb{N}$ i $d: E \to \mathbb{N}$, prirodan broj p i realan broj R. Odrediti da li postoji skup $S, S \subseteq V$, tako da je $|S| = p$ i za sve čvorove $v \in V$ važi da je $v \in S$ ili $(\exists s \in S) d(s, v) \cdot w(v) \leq R$ [35].

U slučaju ulaznih veličina manjih dimenzija, na rešavanje nekog NP-kompletnog problema može se primeniti egzaktan algoritam. Međutim, kako je već za nešto veće ulazne veličine vreme izvršavanja algoritma izuzetno veliko, rešavanje se zasniva na približnim tehnikama. Neki od pristupa su: parametrizacija, randomizacija, aproksimacija, heuristike, i tako dalje. Parametrizacija se zasniva na činjenici da, ukoliko su neki od parametara problema fiksirani, često postoji efikasan algoritam. Randomizacija radi pospešivanja vremena izvršavanja koristi slučajne brojeve, što u opštem slučaju ne garantuje da će algoritam uspeti. Obično se uspešnost izvršavanja algoritma procenjuje nekom verovatnoćom. Aproksimacijom se smanjuje prostor pretrage mogućih rešenja na taj način što se ne traži baš optimalno rešenje, već ono koje mu je veoma blisko.

[1] Engleski termin: p-center problem

3. HEURISTIČKE METODE

3.1 Kombinatorna optimizacija

Kombinatorna optimizacija je matematička disciplina koja proučava probleme nalaženja optimalnih vrednosti neke funkcije na konačnom skupu. Predstavlja jednu granu matematičke optimizacije i usko je povezana sa operacionim istraživanjima i teorijom algoritama. Zauzima značajno mesto u raznim oblastima među kojima su telekomunikacije, veštačka inteligencija, softversko inženjerstvo, matematika, itd., i predstavlja značajno sredstvo u analizi telekomunikaconih mreža, mreža računara, elektroenergetskih i raznih drugih vrsta mreža.

Problem kombinatorne optimizacije se definiše na sledeći način. Dat je najviše prebrojiv[1] skup X i funkcija $f: X \to \mathbb{R}$. Naći minimum funkcije f na skupu X, drugim rečima odrediti

$$\min_{x \in X} f(x).$$

Analogno, ovaj problem se može definisati kao problem traženja maksimuma funkcije $f: X \to \mathbb{R}$ na najviše prebrojivom skupu X, budući da važi jednakost

$$\max_{x \in X} f(x) = -\min_{x \in X}\big(-f(x)\big).$$

Često se za skup X pretpostavlja i da je konačan, pa se u literaturi sreće i definicija gde se činjenica da je skup X prebrojivo beskonačan zanemaruje.

Ako je $X \subseteq \mathbb{Z}^n$, tada se govori o celobrojnom programiranju. Ovde je $\mathbb{Z}^n$ skup svih vektora oblika $(x_1, x_2, \dots, x_n)$, gde $x_i \in \mathbb{Z},\ 1 \leq i \leq n$. Njegov specijalan slučaj je problem linearnog celobrojnog programiranja, koji je oblika

$$\min_{x \in X} c^T x,\ \ X = \{x \in \mathbb{Z}^n \mid Ax \leq b\},$$

gde je A celobrojna matrica, a c i b celobrojni vektori, svi odgovarajućih dimenzija. Uopštenje prethodnog problema, kada X može da sadrži i realne vektore je problem linearnog programiranja,

$$\min_{x \in X} c^T x,\ \ X = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

[1] Konačan ili prebrojivo breskonačan

Jedan specijalan slučaj celobrojnog programiranja je 0 – 1 programiranje u kome su vektori koje sadrži skup X iz skupa $\{0,1\}^n$. Drugim rečima, problem 0 – 1 programiranja je oblika

$$\min_{x \in X} c^T x, \quad X = \{x \in T^n \mid Ax \leq b\},$$

gde je $T = \{0,1\}$.

Naredni problemi su tipični problemi koji se rešavaju metodama kombinatorne optimizacije.

Problem 3.1 (problem trgovačkog putnika): Ako je dat određen broj gradova, cene putovanja od bilo kog grada do bilo kog drugog, odrediti koja je najjeftinija ruta koja obilazi svaki grad tačno jednom i vraća se u početni.

Problem 3.2 (problem n dama): Rasporediti n dama na šahovsku tablu $n \times n$ tako da ne napadaju jedna drugu.

Problem 3.3 (problem ranca): Dat je ranac zapremine n i m predmeta. Za svaki predmet poznata je njegova vrednost v_i i zapremina z_i, $1 \leq i \leq m$. Potrebno je napuniti ranac najvrednijim sadržajem, drugim rečima odrediti koje predmete treba staviti u ranac, a koje ne, tako da njihova ukupna zapremina ne prevazilazi zapreminu ranca, a da je njihova cena najveća moguća.

3.2 Heuristike

Većina problema kombinatorne optimizacije su NP-teški, pa vreme izvršavanja algoritama koji nalaze njihova optimalna rešenja može postati nedopustivo dugačko. Zbog toga se često odustaje od njihovog rešavanja egzakntim algoritmom, već se pribegava tzv. heuristikama. One ne garantuju nalaženje samog rešenja, ali u većini slučajeva daju suboptimalna rešenja koja su bliska optimalnom.

Postoje mnogobrojne specijalizovane heurističke metode koje su dizajnirane za rešavanje specifičnih problema. Zbog toga su one samo na takve probleme i primenljive. Takva je, na primer, heuristika za rešavanje problema trgovačkog putnika (videti problem 5). Sa druge strane, u naglom razvoju su i opšte heurističke metode za rešavanje opšteg problema kombinatorne optimizacije

$$\min_{x \in X} f(x), \quad f: X \to \mathbb{R},$$

pri čemu se u ovom slučaju pretpostavlja da je skup X konačan. Primeri takvih heuristika su bazično lokalno pretraživanje, simulirano kaljenje, tabu pretraga, genetski algoritmi i druge.

3.3 Heuristike zasnovane na lokalnom pretraživanju

Za heuristike koje su zasnovane na principu lokalnog pretraživanja važi da se svakom $x \in X$, gde je X dopustivi skup, prudružuje okolina $U(x)$, takva da važi $U(x) \subset X$ i $x \notin U(x)$. Elementi skupa $U(x)$ se nazivaju susedi od x. Ako se krene od početne tačke iz X, u svakoj iteraciji se generiše, prema utvrđenom pravilu, neka nova tačka iz okoline rešenja iz prehodne iteracije, koja predstavlja novo potencijalno rešenje. Ako u nekom koraku izvršavanja takvo rešenje ne postoji, za rešenje se uzima ono čija je vrednost funkcije cilja najmanja. Opšti princip lokalnog pretraživanja se može prikazati pseudokodom sa slike 3.1.

```
x_1: x_1 ∈ X;
x* = x_1;
f* = f(x_1);
n = 1;
while (kriterijum_odabira(x_n))
{
  x_{n+1}: x_{n+1} ∈ U(x_n);
  if (f(x_{n+1}) < f(x*))
  {
    x* = x_{n+1};
    f* = f(x_{n+1});
  }
}
return x*;
```

Slika 3.1: Princip lokalnog pretraživanja

Simulirano kaljenje [24] je jedna od heuristika koja se zasniva na principu lokalnog pretraživanja. Simulira proces kaljenja nekog raspopljenog materijala do dostizanja čvrstog stanja. Ako se unutrašnja energija smanjuje, menja se i prihvata nova konfiguracija atoma, dok se u slučaju povećavanja energije, ta energija prihvata sa određenom verovatnoćom

prema Bolcmanovom[1] termodinamičkom zakonu. Od početne konfiguracije atoma, ovaj proces se ponavlja, pri čemu se postepeno smanjuje energija. Ako materijal dostigne stanje minimalne energije, onda će on i očvrsnuti. Kod heuristike koja se na tom principu zasniva trenutnoj konfiguraciji atoma odgovara jedno dopustivo rešenje, unutrašnjoj energiji vrednost funkcije cilja, a promeni energetskog stanja odgovara prelaz s prethodnog na naredno rešenje.

Tabu pretraživanje [25] je heuristika koja je takođe zasnovana na principu lokalnog pretraživanja. Ovde se okolina tačke x kreira i u zavisnosti od prethodne istorije, u oznaci H, i na taj način se kreira skup $U(x, H)$. Zatim se on, ukoliko je veliki (u smislu kardinalnosti), sužava na manji podskup $U'(x)$. Naredna tačka se bira tako da bude najbolja među tačkama iz tog skupa. Nekad se dopušta da se ne izabere najbolja već ona koja je gora od trenutno posmatrane, čime se izbegava konvergencija algoritma u lokalnom minimumu.

Kod metode promenljivih okolina [37] okolina definiše se nekoliko okolina, pa se trenutno najboljem rešenju na slučajan način odredi sused u nekoj od tih okolina. Modifikacije se sprovode sistematski, poštujući unapred definisan redosled okolina. Izvršavanje metode se ponavlja sve do zadovoljenja nekog kriterijuma zaustavljanja. Do tada pronađeno najbolje rešenje usvaja se kao konačno.

3.4 Primeri ostalih heuristika

Optimizacija mravljim kolonijama [6] je nastala na osnovu načina kretanja mrava u potrazi za hranom. Mravi se u početku do hrane kreću slučajno, a nakon što je pronađu, vraćajući se u mravinjak, za sobom ostavljaju tragove feromona. Nakon toga, ako neki drugi mrav zapazi trag feromona, on se neće kretati slučajno kao njegov prethodnik, već po tom utvrđenom tragu, čime je povećana verovatnoća da će doći do hrane.

Optimizacija na osnovu roja čestica [7]. Prema analogiji sa jatom ptica, optimizacija se zasniva na grupi čestica, ili ptica, koje lete po slobodnom prostoru u potrazi za najboljom lokacijom. Svaka ptica odgovara jednostavnom elementu koji se kreće duž multidimenzionalnog prostora određujudi vrednosti kriterijumske funkcije u različitim tačkama. Kretanje svake čestice je diktirano brzinom koja se neprestano menja u potrazi za sopstvenom najboljom pozicijom ili najboljom pozicijom za ostatak čestica iz roja. Performansa svake čestice se meri unapred definisanom funkcijom za koju je zahtevano da sadrži karakteristike optimizacionog problema.

[1] Ludwig Eduard Boltzmann (1884–1906), austrijski fizičar

Optimizacija rojem pčela [36]. Kada se pčela vrati u košnicu sa nektarom koji je dovoljno hranljiv da bi ostale pčele otišle do njegovog izvora, ona izvede karakterističan ples na zidu košnice kako bi podelila informaciju o lokaciji tog izvora sa ostalim pčelama. Ovaj ples može sadržati dve bitne informacije, o pravcu i o udaljenosti izvora hrane. Osnovni algoritam za optimizaciju koji se zasniva na ovakvom ponašanju pčela vrši slučajnu pretragu u kombinaciji sa pretragom okoline. Algoritam se oslanja na koncept centralnog plesnog podijuma kako bi se odredila najpogodnija lokacija. Regrutacija može biti izvršena na osnovu fitnes funkcije svake lokacije ili vrednosti ove funkcije mogu određivati verovatnoću da pčela bude regrutovana.

Često se koriste i hibridne metode koje predstavljaju kombinaciju dve ili više heurističkih (ili egzaktnih metoda). Takvi su, na primer, memetički algoritmi, koji se baziraju na činjenici da se termin evolucije može sagledati u širem kontekstu od evolucije bioloških sistema, kao evolucija bilo kog kompleksnog sistema zasnovanog na principima nasleđivanja, varijacije i selekcije. Termin „meme“ se definiše kao osnovna jedinica prenosa ili imitacije pri evoluciji nekog sistema [39].

4. GENETSKI ALGORITMI

4.1 Biološka osnova

Svi živi organizmi se sastoje od ćelija. U svakoj ćeliji postoji isti skup hromozoma koji sačinjavaju lance DNK i služe kao model za ceo organizam. Hromozomi se sastoje od gena, blokova DNK. Svaki gen kodira određeni protein. U osnovi se može reći da svaki gen kodira neku osobinu, na primer boju očiju, boju kose i sl. Različite mogućnosti za svako svojstvo, (na primer braon ili plava boja očiju) se nazivaju aleli. Kompletan skup celokupnog genetskog materijala (svih hromozoma) se naziva genom. Posebno, skup svih gena u hromozomu se naziva genotip. Sa kasnijim razvojem, genotip je osnova fenotipa organizma, njegovih fizičkih i mentalnih karakteristika, kao što su boja očiju, inteligencija, i tako dalje. Tokom reprodukcije, najpre se vrši rekombinacija (tj. ukrštanje). Tako geni roditelja na određen način formiraju sasvim nov hromozom. Nakon toga može doći i do mutacije, što znači da će se eventualno neki geni DNK promeniti. Ove promene nisu toliko česte i obično su uzrokovane greškama koje se javljaju pri kopiranju gena svojih roditelja za dato dete. Prilagođenost organizma predstavlja zapravo verovatnoću njegovog opstanka i mogućnost da se on dalje reprodukuje i u narednoj generaciji ostavi svoje potmostvo.

Osnova za genetske algoritme nalazi se u teoriji evolucije nastale krajem 19. veka. Oni imitiraju proces prilagođavanja i na taj način pokušavaju da reše dati problem. Osnovna konstrukcija je populacija jedinki koja predstavlja skup svih hromozoma u datom trenutku. One jedinke iz populacije koje su u većoj meri prilagođene okruženju se međusobno dalje reprodukuju, pri čemu se stvara nova generacija koja je prilagođenija od svoje prethodne. Evolucija se postiže mutacijama i ukrštanjem različitih jedinki, pri čemu u kreiranju naredne imaju pravo da učestvuju najbolje jedinke iz trenutne generacije. Kao idejni tvorac genetskih algoritama uzima se Džon Holand [8].

4.2 Osnovni pojmovi

Neka je dat konačan skup X i bilo kakva funkcija $f: X \to \mathbb{R}$. Neka je kod problema kombinatorne optimizacije, gde je potrebno naći minimum funkcije f (funkcija cilja) na skupu X (prostor dopustivih rešenja), svakom rešenju iz skupa X dodeljen niz konačne dužine nad nekom konačnom azbukom simbola. Taj se naziva kôd tog rešenja. Skup svih

kodova rešenja iz X čini tzv. prostor kodiranih rešenja, X'. Jedno rešenje iz X se nazova jedinka, a njemu pridruženi kôd hromozom te jedinke. Tada, u svakoj iteraciji, genetski algoritam generiše jedan podskup prostora kodiranih rešenja, koji se naziva populacija. Preciznije, neka je $X = \{x_1, x_2, \dots, x_n\}$ i $k: X \to X'$ funkcija kodiranja. Tada je

$$X' = \{k(x_i) | 1 \leq i \leq n\} = \{k(x_1), k(x_2), \dots, k(x_n)\} = \{x'_1, x'_2, \dots, x'_n\}.$$

Dalje, neka je u m-toj iteraciji generisana populacija $P_m = \{x'_{1,m}, x'_{2,m}, \dots, x'_{r,m}\}$, gde je $x'_{i,m} \in X', 1 \leq i \leq r$. Broj r se naziva još i veličina populacije. Za svaku tačku iz P_m vrši se određivanje njene prilagođenosti (pogodnosti, kvaliteta) pomoću funkcije prilagođenosti, $F: X' \to \mathbb{R}$. Što je kompleksniji problem u pitanju, kompleksnija je i funkcija F. Kada se radi o numeričkim podacima, funkcija F se jednostavno određuje iz prirode samog problema, a kad su u pitanju nenumerički podaci, funkcije prilagođenosti su znatno komplikovanije i u obzir treba uzeti neku metriku za izračunavanje rešenja. Obično se uzima

$$F(x) = -f\big(d(x)\big) = -f(k^{-1}(x)),$$

gde je f funkcija cilja, a $d: X' \to X$ funkcija dekodiranja[1]. Sada se iz skupa P_m biraju one jedinke koje imaju bolju prilagođenost, pa se na njih dalje deluje operatorima koji su po uzoru na genetske operatore iz prirode. Na taj način se dobija nova generacija koja se sastoji od naredne populacije $P_{m+1} = \{x'_{1,m+1}, x'_{2,m+1}, \dots, x'_{r,m+1}\}$.

Na početku izvršavanja algoritma potrebno je inicijalizovati početnu populaciju. Ona se može u potpunosti dobiti na slučajan način ili pak korišćenjem neke heuristike. Pri tome bi ona trebalo da sadrži kodove što raznovrsnijih rešenja. U svakoj iteraciji algoritma uzastopno se primenjuju genetski operatori. Nakon određenog broja iteracija algoritam se zaustavlja. Različiti su kriterijumi zaustavljanja. U osnovnoj verziji algoritma, zaustavljanje se vrši kada se dostigne određeni broj iteracija, ali se koriste i drugi kriterijumi zaustavljanja, kao i njihova kombinacija. Osnovna verzija genetskog algoritma prikazana je na slici 4.1.

4.3 Kodiranje

Kodiranje predstavlja funkciju k, koja preslikava prostor dopustivih rešenja, X, u prostor kodiranih rešenja, X'. Izbor funkcije k u opštem slučaju zavisi od prirode problema. Ako k preslikava skup X čiji su elementi unapred zadate dužine nad azbukom $\{0,1\}$, tada se govori

[1] U opštem slučaju, funkcija kodiranja ne mora da bude bijekcija, pa funkcija dekodiranja ne mora ni da postoji

$X = \{x_i | 1 \leq i \leq n\}$;
$X' = \{k(x_i) | 1 \leq i \leq n\}$;
$F: X' \to \mathbb{R}$;
$P_1 = \{x'_{i,1} | 1 \leq i \leq r\}: P_1 \subseteq X'$;
(x^*, f^*):
{
  $f^* = \min_{1 \leq i \leq r}\{f(d(x'_{i,1}))\}$;
  $f^* = f(d(x^*))$;
}
$m = 1$;
while (!kriterijum_zaustavljanja())
{
  for $(x'_{i,m}: x'_{i,m} \in P_m = \{x'_{i,m} | 1 \leq i \leq r\})$
    $F(x'_{i,m})$;
  selekcija();
  ukrštanje();
  mutacija();
  $P_{m+1} = \{x'_{i,m+1} | 1 \leq i \leq r\}$;
  (x^*, f^*):
  {
    $f^* = \min(f^*, \min\limits_{1 \leq i \leq r} f(d(x'_{i,m+1})))$;
    $f^* = f(d(x^*))$;
}
return x^*;

Slika 4.1: Osnovna verzija genetskog algoritma

o binarnom kodiranju. Ovo kodiranje se često koristi kod problemima kombinatorne optimizacije. Ovim se, čak i sa malim brojem bitova, omogućava generisanje velikog skupa potencijalnih hromozoma. Sa druge strane, u nekoj od iteracija genetskog algoritma, nakon selekcije i mutacije, može doći to pojave eventualnih grešaka, pa je potrebno izvršiti određene korekcije.

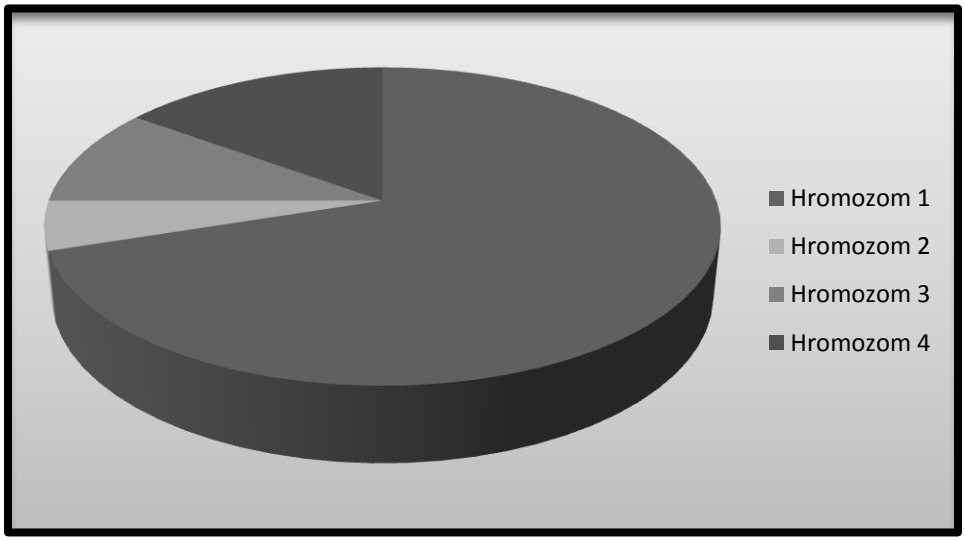
Neka je, na primer, dat ranac u kome se nalaze predmeti različite veličine i cene. Za datu vrednost veličine ranca neka je potrebno maksimizovati cenu predmeta u njemu, ali tako da njihova ukupna veličina ne prelazi veličinu ranca. Dakle, potrebno je odabrati određene predmete (iz skupa svih predmeta) koje treba staviti u ranac (problem 3.3). Ovde se prirodno nameće binarno kodiranje, čiji su hromozomi dužine broja predmeta u rancu i bit na i-tom mestu ima vrednost 1 ako je predmet uključen u ranac, a 0 inače.

Pri kodiranju se može koristiti i niz prirodnih brojeva, koji se najčešće prikazuje pogodnim za upotrebu u slučaju permutacionih problema, kao na primer kod problema trgovačkog

putnika, gde je za dati skup gradova i njihovih međusobnih rastojanja potrebno posetiti sve gradove tačno jednom, ali tako da dužina ukupnog puta bude najkraća moguća. Ako, na primer, ima n gradova, svaki hromozom u ovom slučaju predstavlja jednu permutaciju skupa $\{1, 2, \dots, n\}$. Ponekad su za zadati problem obe ove vrste kodiranja teško primenljive, pa tako u opštem slučaju jednom hromozomu možemo dodeliti i niz realnih vrednosti, znakova ili čak nešto komplikovanijih objekata. Primeri takvih hromozoma su $\{2.1, 3.22, \sqrt{2}, 9.7764, \pi\}$, $\{(desno), (desno), (dole), (gore), (desno)\}$, $ABGDRZJHDAXVF$, i slično.

4.4 Selekcija

Selekcija predstavlja izbor jedinki iz trenutne populacije P_m koje će biti korišćenje za dobijanje naredne generacije. Prema Darvinovoj teoriji, one jedinke koje su najbolje prilagođene sredini bi trebalo da prežive i formiraju nove jedinke. Postoji mnogo načina za odabir takvih hromozoma. U osnovnoj verziji algoritma se bira paran broj s, gde je $s \leq r$ i r veličina populacije. Zatim se s puta vrši odabir jedinki iz populacije u zavisnosti od njihove funkcije prilagođenosti. Pri tom, jedna jedinka može biti izabrana i više puta.

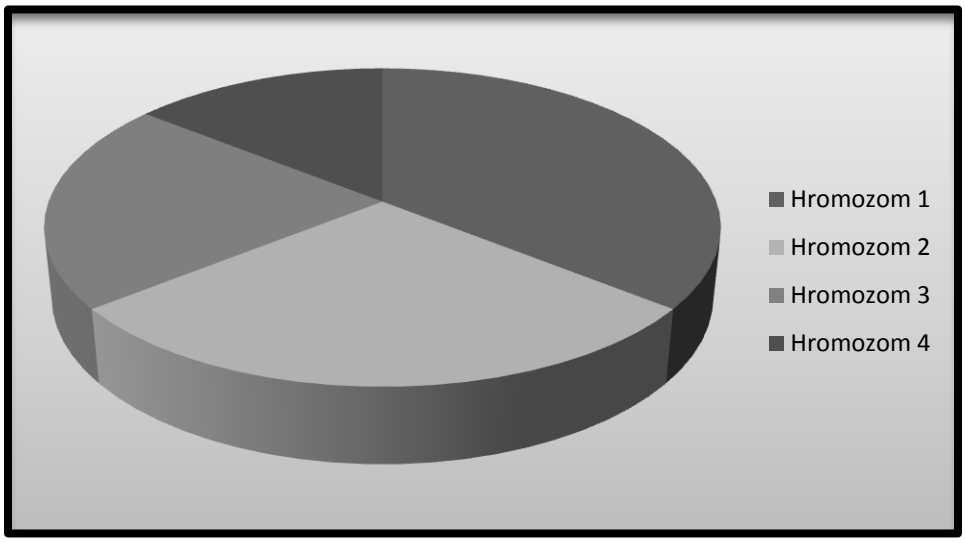


Slika 4.2: Selekcija zasnovana na principu ruletskog točka

Često se pri odabiru jedinki za dobijanje nove generacije koristi tzv. princip ruletskog točka [9], kod koga se na ruletskom točku se nalaze hromozomi date populacije, od kojih će svaki, u zavisnosti od vrednosti njegove funkcije prilagođenosti, zauzimati određeni deo (slika 4.2). Ako se kuglica baci na točak, pri čemu se ona sigurno zaustavlja u nekom njegovom

delu, očekuje se da će se ona zaustaviti na onim hromozomima čija je veća funkcija prilagođenosti. Na taj način će i jedinke koje imaju bolju pogodnost biti i izabrane.

Međutim, u ovom slučaju može doći do pojave da je neki hromozom, proporcionalno njegovoj funkciji prilagođenosti, zauzima, na primer, i do 70% kruga (kao što je slučaj na slici 4.2), a potreba je da se on bira ređe od 70% slučajeva. Ovaj problem se može rešiti tako što se hromozomi rangiraju i to na taj način da hromozom koji ima najgoru funkciju prilagođenosti imati pogodnost 1, a hromozom koji ima najbolju funkciju prilagođenosti imati pogodnost n. Na taj način se dobija ruletski točak prikazan na slici 4.3.



Slika 4.3: Selekcija zasnovana na principu rangiranja

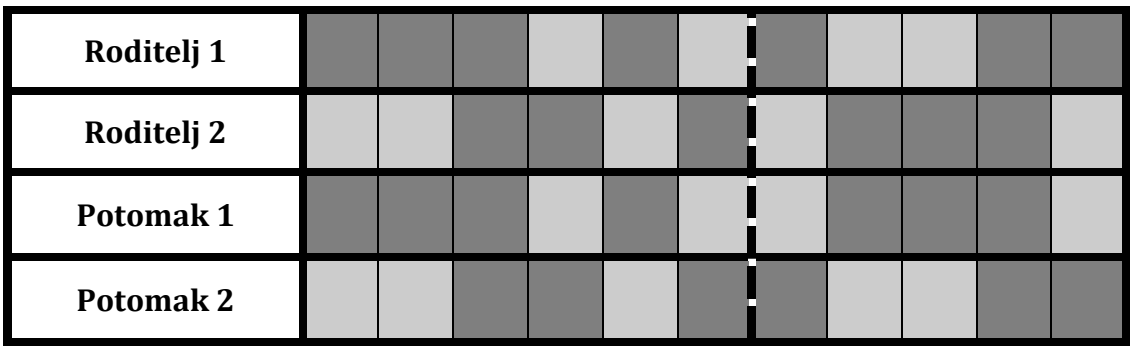
4.5 Elitizam

Kod prostog genetskog algoritma se u svakoj generaciji zameni čitava populacija novim jedinkama. Iako je pretpostavka da će kvalitetni geni sa velikom verovatnoćom proći u narednu generaciju, to ne mora uvek da bude slučaj. Tako se može desiti da neki potomci budu lošiji od roditelja ili čak i da najbolja jedinka ne prođe u narednu generaciju. Jedan od najlošijih ishoda je da se u nekom trenutku dostigne rešenje koje je dobrog kvaliteta, a da ga se zatim izgubi primenom operatora ukrštanja i mutacije. Zato se često primenjuje princip elitizma, koji se zasniva na ideji da se najbolji ili više najboljih hromozoma prenesu direktno u narednu generaciju. Ovim se znatno pobošljava rezultat dobijen primenom algoritma u celini, jer se sa sigurnošću može tvrditi da će najbolje rešenje do koga se došlo u datom trenutku izračunavanja uvek uzimati u obzir.

4.6 Ukrštanje

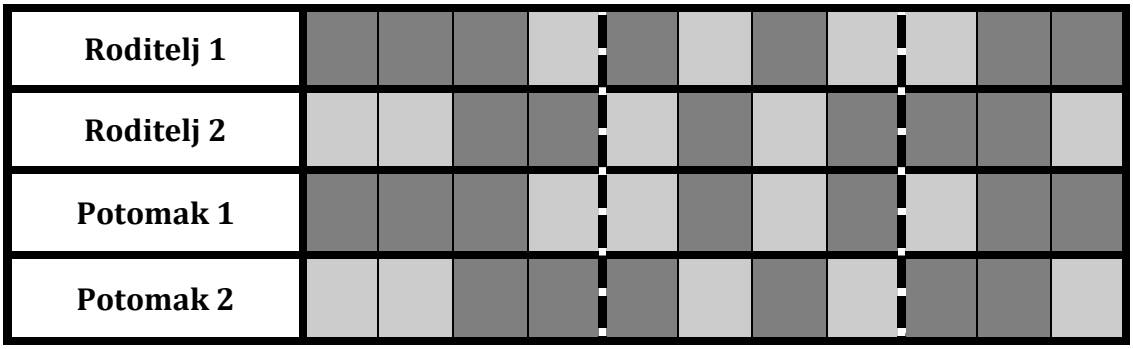
U opštem slučaju, ukrštanje je postupak kojim se na slučajan način razmenjuju geni dva roditelja, pri čemu se dobijaju dve nove jedinke, potomci tih roditelja. Može se očekivati da, ako su roditelji imali visoku funkciju prilagođenosti, to biti slučaj i sa njihovom potmocima, tj. da će i oni biti dobro prilagođene jedinke.

Sledeće vrste ukrštanja su prikazane na primeru binarnog kodiranja, a analogno važi i za ostale slučajeve kodiranja. Kod jednopozicionog ukrštanja uniformno se bira jedna tačka ukrštanja (broj t takav da je $0 \leq t \leq r$, gde je r veličina populacije). Ovde prvi potomak uzima pre tačke ukrštanja gene od jednog roditelja, a nakon nje od drugog, suprotno drugom potomku (slika 4.4).



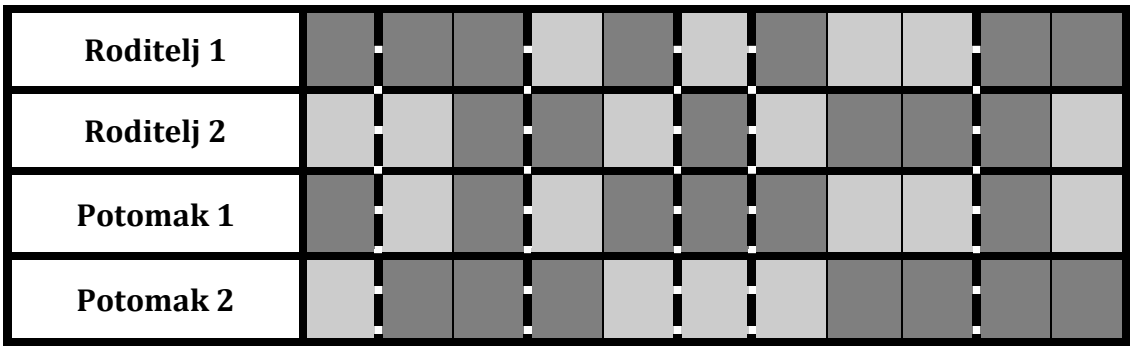
Slika 4.4: Jednopoziciono ukrštanje

Kod dvopozicionog ukrštanja slučajno se biraju dve pozicije t_1 i t_2, $0 \leq t_1, t_2 \leq r$, nakon čega se geni roditelja razmenjuju na način prikazan na slici 4.5. U tom slučaju jednan potomak od prvog roditelja dobija gene pre tačke t_1 i nakon tačke t_2, a između tih tačaka od drugog roditelja; obrnut je slučaj sa drugim potomkom.



Slika 4.5: Dvopoziciono ukrštanje

Kod uniformnog ukrštanja, za svaki gen, odabir roditelja za prenošenje datog gena na potomstvo je slučajan (slika 4.6). Ukrštanje može biti i aritmetičko, gde se za obrazovanje potomstva koriste neke aritmetičke operacije.



Slika 4.6: Uniformno ukrštanje

4.7 Mutacija

Mutacija predstavlja promenu sadržaja hromozoma neke jedinke slučajnom zamenom pojedinih simbola tog koda nekim drugim iz iste azbuke simbola. Ona se na svaku jedinku primenjuje sa verovatnoćom p_m. Za veličinu p_m se koristi obično veoma mali broj, budući da su u prirodi mutacije ređe i javljaju. Međutim taj broj ne sme biti ni mnogo mali, jer se može desiti da će populacija iz generacije u generaciju postajati vrlo slična, što dovodi do pojave lokalnog optimuma. Dakle, ukoliko se mutacija uopšte ne primenjuje, izražena je mogućnost lokalne konvergencije, a to je neželjen efekat. Sa druge strane, ako je vrednost p_m velika, genetski algoritam će prostor rešenja pretraživati bez jasnog fokusa.

U slučaju binarnog kodiranja, mutacija predstavlja invertovanje jednog bita, tj. ako je taj bit imao vrednost 1, nakon mutacije će sa verovatnoćom p_m imati vrednost 0, i obratno. Na primer, ukoliko se mutacija izvršila na trećem bitu niske 100110, rezultat će biti 101110. Ukoliko je kôd predstavljen mutacijom datog skupa elemenata, mutacija bi mogla da predstavlja zamenu mesta za dva prirodna broja. Na primer, hromozom 123456897 se mutira u 183456297. Ukoliko je hromozom predstavljen nizom realnih brojeva, mutacija bi mogla da predstavlja dodavanje realnog broja koji je mali po apsolutnoj vrednosti na neke od realnih veličina. Na primer, od hromozoma (1.33 2.54 7.47 11.3 1.28 3.22) može se dobiti hromozom (1.33 2.54 7.35 11.3 1.28 3.43).

4.8 Parametri

Pri implementaciji genetskih algoritama, veoma važnu ulogu igra odabir parametara. Ako se parametri podese pre izvršavanja samog genetskog algoritma, tada se govori o fiksnom

postavljanju vrednosti parametara. Inače, govori se o adaptivnoj promeni. U tom slučaju se parametri automatski menjaju na osnovu prethodne uspešnosti. Veličina parametara se obično prilagođava prirodi problema, pa za različite grupe problema može i da znatno varira. U opštem slučaju se i ne može govoriti o fiksnoj vrednosti parametara, mada se, bar u osnovnoj verziji algoritma, ponekad mogu približno odrediti neke preporučene vrednosti. Na primer, za verovatnoću mutacije treba uzeti mali broj, dok verovatnoća ukrštanja treba da bude znatno veća (blizu 1) [8]. Neke od vrednosti parametara koje se preporučuju su:

1. verovatnoća mutacije: između 0.005 i 0.01;
2. verovatnoća ukrštanja: između 0.8 i 0.95;
3. veličina populacije: varira, nekad u intervalu 20 – 30 jedinki, a nekad 50 – 150.

4.9 Efikasnost

Genetski algoritmi se često koriste za rešavanje problema za koje je egzaktnim metodama potrebno neprihvatljivo mnogo vremena da dođu do tačnog rešenja. Primeri takvih problema su NP-teški problemi za koje se ne zna da li postoji polinomijalni algoritam, pa su vremenske složenosti poznatih algoritama koji rešavaju ovu grupu problema znatno veće od polinomijalne. Genetski algoritmi su veoma korisni zbog mogućnosti njihove paralelizacije, što se naglim razvojem paralelnog programiranja poslednjih godina pokazalo veoma korisnim. Takođe, veoma su jednostavni i za implementaciju. No, njihovo vreme izvršavanja nekad može da bude problem, jer su sporiji u poređenu sa većinom poznatih heuristika. Ali, to i nije toliki problem s obzirom na brzinu današnjih procesora. Često se za ubrzavanje genetskog algoritma koristi keširanje.

5. FORMULACIJA PROBLEMA

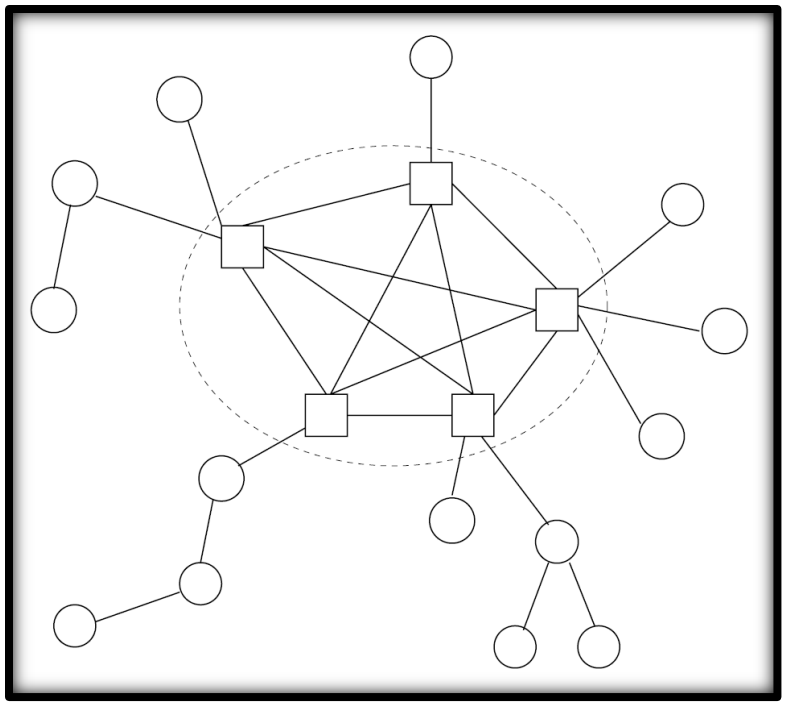
5.1 Dizajn telekomunikacione mreže

Do pre dve decenije telekomunikacione mreže su se uglavnom vezivale za prenos telefonskih signala, no danas je sasvim drugačija slika — pored zvuka, vrši se prenos, između ostalog, i podataka i video zapisa. Posebno, na značaj ovih mreža utiče i njihova primena na internetu. Zbog njihovog ubrzanog razvoja u poslednje vreme, javljaju se i novi problemi koje treba rešiti, pri čemu i neki stari postaju još značajniji. Otuda i veliki interes za nalaženje efikasnih algoritama za njihovo rešavanje.

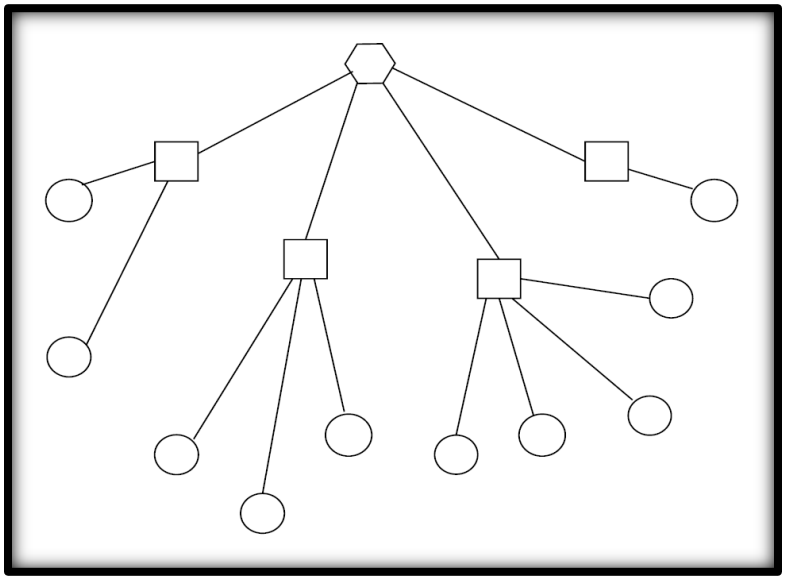
U tipičnoj telekomunikacionoj mreži saobraćaj se prikuplja iz velikog broja izvora, a zatim preusmerava na odgovarajuće odredište, pa zbog toga i mnoge mreže imaju hijerarhijsku strukturu. Na nižim nivoima se prikuplja sav saobraćaj, a nakon toga šalje na viši nivo. Iako su telekomunikacione mreže (bilo da se radi o prenosu zvuka, videa ili podataka, bilo da se radi o nacionalnoj ili internacionalnoj mreži) dizajnirane tako da se sastoje od ogromnog broja takvih nivoa.

U opštem slučaju, telekomunikaciona mreža se sastoji od pristupnih mreža koje spajaju terminale (korisničke čvorove) sa koncetratorima (prekidači ili multiplekseri) i bazne mreže koja spaja ove koncetratore sa centralnom jedinicom (koren). Različite vrste ovih mreža se razlikuju u dizajnu pristupnih i baznih mreža. Terminali mogu biti povezani sa koncetratorima direktno, što daje zvezdastu strukturu, ili indirektno, kada imaju strukturu drveta ili magistrale. Takođe, postoje razne strukture za baznu mrežu — ona može biti potpuno povezana ili mrežasta. Ako postoji centralna jedinica, tada je ona zvezdastog oblika, oblika drveta ili magistrale. Kod nekih modela, terminali se mogu i direktno povezati na centralnu jedinicu. Na slikama 5.1, 5.2 i 5.3 su prikazani neki primeri takvih mreža.

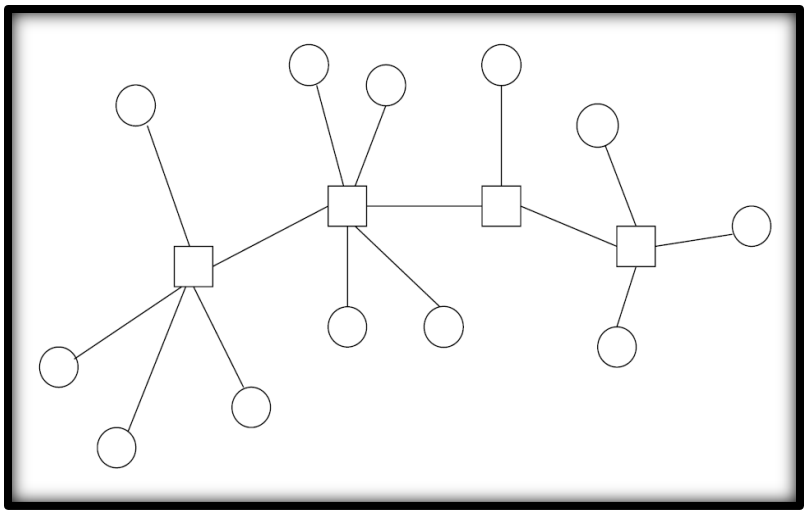
Kako je gotovo nemoguće nositi se sa svim problemima dizajna istovremeno, određeni problemi prilikom dizajnara telekomunikacionih mreža raščlanjeni. Tako se odvojeno tretiraju problem broja lokacija za koncentratore i načina povezivanja sa terminalima, problem dizajna bazne i problem dizajna pristupne mreže, zatim određivanje cene instaliranja mreže, cene njenog operisanja, pouzdanost i sposobnost da u razumnom vremenu odgovori na zahteve terminalâ. Postoje i modeli koji dozvoljavaju i povremeno proširivanje mreže novim koncetratorima. Takvi modeli se nazivaju dinamičkim.



Slika 5.1: Telekomunikaciona mreža kod koje je bazna mreža u potpunosti povezana, a pristupne mreže imaju strukturu drveta. Kvadratima su označeni koncentratori, krugovima terminali, a bazna mreža je uokvirena isprekidanom linijom



Slika 5.2: Telekomunikaciona mreža sa centralnom jedinicom kod koje i bazna i pristupne mreže imaju zvezdastu strukturu. Centralna jedinica je označena pravilnim šestouglom, koncetratori kvadratima, a terminali krugovima



Slika 11: Telekomunikaciona mreža kod koje bazna mreža ima strukturu magistrale, dok su pristupne zvezdaste. Kvadratima su označeni koncentratori, a krugovima terminali

5.2 Formulacija problema koji se rešava

Modeli telekomunikacionih mreža neograničenih kapaciteta se odnose na raspoređivanje koncentratora i pridruživanje terminala tim koncetratorima pod pretpostavkom da za koncentratore nema ogrančenja kapaciteta, pa se samim tim i ne uzima u obzir potražnja terminalâ. Neki problemi ovog tipa takođe razmatraju i povezanost bazne i pristupnih mreža.

Problem 5.1 (dvostepeni problem instalacije neograničenih kapaciteta): Neka je N skup terminala, M skup mogućih lokacija za koncentratore i K skup mogućih lokacija za superkoncentratore. Za svaki od terminala data je cena pridruživanja tog terminala svakoj lokaciji koncentratora, za svaku od lokacija koncentratora data je cena instalacije koncentratora na toj lokaciji i cena pridruživanja svakom superkoncentratoru, a za sve lokacije superkoncentratora data je cena instalacije koncentratora na tom mestu. Cilj je minimizovati cenu instaliranja koncentratora i pridruživanja terminala koncentratorima, tako da svaki terminal bude pridružen nekom koncentratoru, koji će dalje biti pridružen nekom od superkoncentratora [10].

U ovom radu koristi se formulacija problema predložena u [38], koja je zasnovana na celobrojnom programiranju. Neka je C_{ij} $(i \in N, j \in M)$ cena pridruživanja terminala i koncetratoru na lokaciji j (u koncentrator j), B_{jk} $(j \in M, k \in K)$ suma cene postavljanja koncentratora j i cene njegovog pridruživanja superkoncentratoru na lokaciji k (u nastavku

superkoncentratorom k), a F_k $(k \in K)$ cena postavljanja superkoncentratora k. Za sve $i \in N, j \in M$ i $k \in K$, neka su binarne promenljive x_{ij}, y_{jk} i z_k definisane na sledeći način:

$$x_{ij} = \begin{cases} 1, \text{terminal } i \text{ je dodeljen koncetratoru } j, \\ 0, \text{inače,} \end{cases}$$

$$y_{jk} = \begin{cases} 1, \text{koncentrator } j \text{ je dodeljen superkoncetratoru } k, \\ 0, \text{inače,} \end{cases}$$

$$z_k = \begin{cases} 1, \text{superkoncentrator } k \text{ je instaliran,} \\ 0, \text{inače.} \end{cases}$$

Imajući u vidu gornju notaciju, problem se moze matematički zapisati na sledeci način. Odrediti

$$\min \sum_{i \in N} \sum_{j \in M} C_{ij} x_{ij} + \sum_{j \in M} \sum_{k \in K} B_{jk} y_{jk} + \sum_{k \in K} F_k \, z_k,$$

pri sledećim ograničenjima:

$$\sum_{j \in M} x_{ij} = 1, \qquad \forall i \in N, \tag{5.1}$$

$$x_{ij} \leq \sum_{k \in K} y_{jk}, \qquad \forall i \in N, \ \ \forall j \in M, \tag{5.2}$$

$$y_{jk} \leq z_k, \qquad \forall j \in M, \ \ \forall k \in K, \tag{5.3}$$

$$\sum_{k \in K} y_{jk} \leq 1, \qquad \forall j \in M, \tag{5.4}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in N, \ \ \forall j \in M, \tag{5.5}$$

$$y_{jk} \in \{0, 1\}, \qquad \forall j \in M, \ \ \forall k \in K, \tag{5.6}$$

$$z_k \in \{0, 1\}, \qquad \forall k \in K. \tag{5.7}$$

Funkcija cilja ovde minimizuje ukupnu sumu cene pridruživanja terminala koncentratorima ($\sum_{i\in N}\sum_{j\in M} C_{ij}x_{ij}$), cene instalacije koncetratora i njihovog pridruživanja superkoncentratorima ($\sum_{j\in M}\sum_{k\in K} B_{jk}y_{jk}$) i cene instalacije superkoncentratora ($\sum_{k\in K} F_k\, z_k$). Značenja uslova (5.1) – (5.7) su sledeća:

(5.1) svaki terminal se pridružuje tačno jednom koncentratoru;

(5.2) ako je dati terminal pridružen koncentratoru, taj koncentrator mora biti instaliran i pridružen jednom superkoncentratoru;

(5.3) ako je koncentrator pridružen superkoncentratoru, taj superkoncentrator mora biti instaliran;

(5.4) svaki koncentrator može biti dodeljen najviše jednom superkoncetratoru;

(5.5), (5.6), (5.7) odgovarajuće promenljive su binarne, tj. mogu uzimati vrednosti iz skupa {0,1}.

Dvostepeni problem instalacije neograničenih kapaciteta je NP-težak [11].

5.3 Formulacije sličnih problema

Dvostepeni problem instalacije neograničenih kapaciteta se često u literaturi povezuje sa nekoliko njemu srodnih problema, pomenutih u nastavku. Za sve njih je poznato da spadaju u grupu NP-teških problema.

Problem 5.2 (jednostepeni problem instalacije neograničenih kapaciteta): Za dati skup terminala N i koncetratora M, data je cena pridruživanja svakog terminala svakom koncentratoru i cena instalacije svakog koncentratora. Potrebno je minimizovati cenu pridruživanja terminala nekim koncetratorima i instalacije tih koncetratora, pri čemu svaki terminal mora biti pridružen bar jednom koncentratoru.

Po definicji ovog problema celobrojnim programiranjem [10] (uz oznake iz problema 5.1), potrebno je odrediti

$$\min \sum_{i\in N}\sum_{j\in M} C_{ij}x_{ij} + \sum_{k\in K} F_j y_j,$$

imajući u vidu sledeći skup ograničenja:

$$\sum_{j \in M} x_{ij} = 1, \qquad \forall i \in N, \tag{5.8}$$

$$x_{ij} \leq y_j, \qquad \forall i \in N, \ \forall j \in M \tag{5.9}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in N, \ \forall j \in M, \tag{5.10}$$

$$y_j \in \{0, 1\}, \qquad \forall j \in M. \tag{5.11}$$

Uslov (5.8) znači da se svaki terminal pridružuje koncentratoru na tačno jednoj lokaciji, uslov (5.9) znači da terminal može biti dodeljen koncetratoru samo ako je ovaj instaliran, dok uslovi (5.10) i (5.11) označavaju da su odgovarajuće promenljive binarne, tj. da uzimaju vrednosti iz skupa $\{0, 1\}$.

I jednostepeni i dvostepeni problem imaju i svoju „ograničenu“ varijantu, koja se odnosi na ograničavanje kapaciteta koncetratorâ i definisanje potražnje terminalâ.

Problem 5.3 (jednostepeni problem instalacije ograničenih kapaciteta): Za dati skup terminala N sa pridruženim potražnjama d_i $(i \in N)$ i skup koncetratora M sa pridruženim kapacitetima q_j $(j \in M)$, potrebno je instalirati koncetratore na nekim lokacijama i pridružiti svaki terminal tačno jednom koncentratoru tako da je cena pridruživanja minimalna, a kapaciteti koncentratora dovoljni da zadovolje potažnju pridruženih terminala.

Definicija ovog problema zasnovana na celobrojnom programiranju [10] se, uz prethodne oznake, odnosi na određivanje

$$\min \sum_{i \in N} \sum_{j \in M} C_{ij} x_{ij} + \sum_{k \in K} F_j y_j,$$

imajući u vidu sledeće pretpostavke:

$$\sum_{j \in M} x_{ij} = 1, \qquad \forall i \in N, \tag{5.12}$$

$$\sum_{i \in N} d_i x_{ij} \leq q_j y_j, \qquad \forall j \in M \tag{5.13}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in N, \ \forall j \in M, \tag{5.14}$$

$$y_j \in \{0,1\}, \qquad \forall j \in M. \qquad (5.15)$$

Analogno bi se mogao formulisati i dvostepeni problem instalacije ograničenih kapaciteta. Slično, postoje i višestepeni slučajevi, pa se tako može govoriti i o n-to stepenim problemima instalacije ograničenih, odnosno neograničenih kapaciteta za proizvoljno $n \in \mathbb{N}$.

5.4 Dosadašnji rezultati

Među prvima jednostepeni problem instalacije neograničenih kapaciteta razmatraju autori u [30]. U [4] se razmatra problem kod koga su koncentratori spojeni na centralnu jedinicu, pri čemu je svakom koncentratoru pridružena vrednost pridruživanja centralnoj jedinici. U [5] je razvijen model u kome su svi koncentratori spojeni međusobno, dok je svakom koncentratoru pridružen svaki terminal, pri čemu su u obzir uzete cene pridruživanja terminalâ koncentratorima, cene instaliranja koncentratora i njihovog međusobnog spajanja. U [2] autor ograničava broj mogućih koncentratora koji se instaliraju. Predloženo rešenje koristi heuristiku koja se bazira pre svega na idejama pohlepnog algoritma. U [3] autori koriste dve heuristike bazirane na Langranžovoj relaksaciji za rešavanje varijante problema opisane u [5]. Slična ideja za rešavanje se koristi i u [31] i [32], gde se razmatra problem kod koga su svi koncentratori međusobno spojeni minimalnim drvetom razapinjanja. U [26] je razmotren upravo dvostepeni problem instalacije neograničenih kapaciteta za čije rešavanje je korišćeno simulirano kaljenje. Tu je uzeto da skupovi N, M i K imaju isti broj elemenata. U [33] i [34] je razmatra jednostepeni problem instalacije ograničenih kapaciteta, gde su u obzir uzete potražnje terminala i kapaciteti koncentratora. U [27] je rešavan dvostepeni problem ograničenih kapaciteta i za rešavanje je korišćenja Lagranžova relaksacija u kombinaciji sa metodom grananja i ograničavanja. Za isti problem je u [1] predloženo rešenje koje koristi tabu pretragu. Do sada u literaturi za dvostepeni problem instalacije neograničenih kapaciteta nikada nije predloženo rešenje genetskim algoritmom.

6. GENETSKI ALGORITAM ZA REŠAVANJE PROBLEMA

Naglim razvojem telekomunikacionih mreža i njihove široke primene u poslednje dve decenije, javlja se i potreba za konstrukcijom efikasnih algoritama koji rešavaju probleme njihovog dizajna. S obzirom na značaj razmatranog problema u dizajniranju telekomunikacionih mreža, potreba je razvijati efikasne algoritme koji će davati zadovoljavajuća rešenja za ulazne veličine velikih dimenzija. Otuda i motivacijaja za primenu genetskih algoritama za rešavanje dvostepenog problema instalacije neograničenih kapaciteta.

6.1 Kodiranje

Genetski algoritam koji je predložen za rešavanje dvostepenog problema instalacije neograničenih kapaciteta se bazira na binarnom kodiranju, što je i donekle intuitivno, imajući u vidu definiciju problema preko celobrojnog programiranja. Jedna implementacija bi se sastojala u sledećem: hromozom date jedinke bi trebalo da se sadrži od niza bitova u kome su redom poređane binarne promenljive x_{ij} $(i \in N, j \in M)$, y_{jk} $(j \in M, k \in K)$ i z_k $(k \in K)$, gde na odgovarajućoj poziciji x_{ij} uzima vrednost 1 ako i samo ako je i-ti terminal pridružen j-tom koncentratoru, y_{jk} uzima vrednost 1 ako i samo je j-ti koncentrator pridružen k-tom superkoncentratoru i z_k uzima vrednost 1 ako i samo ako je uspostavljen superkoncetrator na lokaciji k. Međutim, ovakva implementacija nije uzeta pre svega zbog manje efikasnosti izvršavanja algoritma.

U ovoj imlementaciji se koristi sledeći način kodiranja. Neka je N skup terminala, M koncentratora, K skup superkoncentratora i neka važi $|N| = n,\ |M| = m,\ |K| = k$. Prvi deo hromozoma će se sastojati od k bitova, gde će i-ti bit $(1 \leq i \leq k)$ uzeti vrednost 1 ako je superkoncentrator uspostavljen na lokaciji i, a inače će imati vrednost 0. Neka je uspostavljeno p superkoncentratora, $1 \leq p \leq k$. Nakon toga, u genetskom kodu se nalazi niz od m gena, gde svaki gen opisuje jedan od koncetratora i predstavlja niz bitova. Prvi bit svakog gena označava da li je dati koncentrator uspostavljen i u zavisnosti od toga uzima vrednost 0 ili 1. Drugi deo gena predstavlja niz bitova dužine $\lceil \log_2 k \rceil$, gde $\lceil x \rceil$ označava

najmanji ceo broj koji je veći ili jednak x. Pri tom je dužina tih bitova jednaka dužini binarnog zapisa broja k. Taj broj označava kom superkoncentratoru je pridruženi dati koncetrator na sledeći način. Broj koji se dobija pretvaranjem datog niza bitova u dekadnu vrednost je neki ceo broj u interavlu $[1, p]$. Neka su superkoncetratori sortirani rastuće prema vrednosti lokacija na kojima su uspostavljeni. Njih ima p. Ako je vrednost drugog dela gena za dati koncetrator jednaka j ($1 \leq j \leq p$), posmatra se j-ta vrednost u sortiranom nizu i pridružuje joj se superkoncetrator na lokaciji koja ima tu vrednost. Ukoliko se pri računanju kod ukrštanja ili inicijalizacije populacije na način koji će biti opisan dobije broj veći od p, vrednost se računa po modulu p, pri čemu, ukoliko je ona jednaka 0, ažurira se na p. Na ovaj način ne dolazi do pojave nekorektnih jedinki. U nastavku genetskog koda nalazi se n gena, po jedan za svaki terminal. Svaki od njih je dužine $\lceil \log_2 m \rceil$ i predstavlja niz bitova koji su binarni zapis koncetratora kome je taj terminal pridružen. Pretpostavlja se da su prethodno svi koncetratori sortirani rastuće po vrednosti lokacija na kojima su uspostavljeni, a zatim se posmatra odgovarajući član sortiranog niza, pri čemu je taj niz dužine s, gde je s broj uspostavljenih koncentratora. Postupa se na analogan način na koji se koncetratorima pridružuju superkoncetratori.

Na primer, neka je $K = \{1, 2, \dots, k\}$, $M = \{1, 2, \dots, m\}$ i $N = \{1, 2, \dots, n\}$, gde je $k = |K| = 6$, $m = |M| = 5$ i $n = |N| = 4$. Genetski kod neke jedinke može biti definisan kao niz bitova

[101011] [(0|101) (1|001) (1|010) (0|010) (1|011)] [(010) (001) (011) (011)],

gde su, zbog preglednosti, celine gena, koje se odnose na superkoncetratore, koncetratore i terminale, respektivno, razdvojene uglastim zagradama „[]“, pojedinačni geni koji označavaju koncetratore i terminale razdvojeni malim zagradama „()“, a delovi gena koncetratorâ koji se odnose na to da li je koncetrator uspostavljen i kom superkoncetratoru je pridružen, razdvojeni uspravnom crtom „|“. Ovde deo [101011] označava da postoji $k = 6$ lokacija za superkoncetratore od kojih je $p = 4$ uspostavljeno. Nakon sortiranja rastuće uspostavljenih koncetratora dobija se niz $P = [1, 3, 5, 6]$, $|P| = p = 4$. U nastavku, deo hromozoma koji se odnosi na koncetratore označava da ima ukupno $m = 5$ potencijalnih lokacija za njih. Tako je, na primer, trećem koncentratoru pridružen gen oblika (1|010). Prvi bit 1 označava da je on uspostavljen. Drugi deo gena, 010, je dužine $\lceil \log_2 6 \rceil = 3$, njegova dekadna vrednost je $(010)_2 = 2$ i označava da je datom koncetratoru pridružen onaj superkoncetrator koji je drugi u nizu P, tj. onaj koji se nalazi na lokaciji 3. Svi ovi geni su ukupne dužine 4, a binarna vrednost, koja predstavlja drugi deo odgovarajućeg gena, posmatrana dekadno je ceo broj u intervalu $[1, p] = [1, 5]$. U nastavku su $n = 4$ gena koji opisuju terminale od kojih je svaki dužine $\lceil \log_2 m \rceil = \lceil \log_2 5 \rceil = 3$. Neka je $s = 3$ broj uspostavljenih koncentratora. Svaki gen terminala predstavlja kom koncetratoru je on pridružen. Nakon sortiranja uspostavljenih koncetratora rastuće po lokaciji dobijamo niz $S = [2, 3, 5]$, $|S| = s = 3$. Na primer, gen prvog terminala ima vrednost (010) i, nakon prevođenja u dekadni sistem, se dobija $(010)_2 = 2$, pa sledi da je

prvi terminal pridružen koncetratoru prvog reda koji je na lokaciji čija je vrednost druga po redu u nizu S, tj. lokaciji 3. Slično važi za ostale terminale. Dekadna vrednost svakog gena terminala je ceo broj u intervalu $[1, s] = [1, 3]$.

6.2 Selekcija

Za operator selekcije su odabrane turnirska selekcija i njena modifikacija, fino gradirana turnirska selekcija [12]. Tako su dobijene i istestirane dve verzije algoritma.

Turnirska selekcija [13] je jedan od najpopularnijih načina selekcije. Razvitkom paralelnih genetskih algoritama, njena popularnost je u poslednjem periodu još više izražena. U turnirskoj selekciji, svaki element populacije se bira za prelazak u narednu generaciju ukoliko ima bolju funkciju prilagođenosti od nekoliko ostalih proizvoljno odabranih elemenata populacije. Parametar selekcije je veličina turnira, N. Važi da je $N \in \mathbb{N}$. Algoritam se može prikazati pseudokodom na slici 6.1. U primenjenom algoritmu je uzeto $N = 5$.

```
br_pop = length(stara_populacija);
for (i = 0; i < br_pop; i++)
{
   for (j = 0; j < N; j++)
   {
      k = random(0, br_pop);
      turnir[j] = stara_populacija(k);
   }
   for (j = 0; j < N; j++)
      if F(turnir[j]) > f(nova_populacija[i])
         nova_populacija[i] = turnir[j];
}
return nova_populacija;
```

Slika 6.1: Turnirska selekcija

Vremenska složenost turnitske selekcije je $O(Nn)$, gde je n broj elemenata populacije. Međutim, obično je N veoma malo u odnosu na n, tako da je složenost linearna i iznosi $O(n)$. Vidimo da kod turnirske selekcije nije potrebno sortiranje, koje je obično prisutno kod nekih od načina selekcije. Zbog nedostatka potrebe da se sortira cela populacija, velika je njena popularnost u paralelnim implementacijama. Međutim, izvršavanjem algoritma se često dešava da se sâm proces odvija veoma sporo ili pak dolazi do prerane konvergencije.

Ovo se može nadomestiti korišćenjem tzv. fino gradirane turnirske selekcije, koja predstavlja uopštenje prethodnog algoritma [12].

Umesto celobrojnog parametra N (veličina turnira) koristi se realan parametar F — željena prosečna veličina turnira. Od ovog parametra zavisi proces selekcije, pa bi prosečna veličina turnira u populaciji trebalo da bude što bliža njegovoj vrednosti. Baš kao i kod obične turnirske selekcije, element populacije je izabran ako ima bolju funkciju prilagođenosti od svojih proizvnoljno odabranih konkurenata. Međutim, u ovom slučaju, tokom jednog koraka selekcije postoje različite vrednosti turnira. Na slici 6.2 je prikazan pseudokod za ovaj algoritam ako su razlike veličinâ turnirâ manje ili jednake 1 i izabrane su tako da je njihova prosečna veličina što bliža parametru $F \in \mathbb{R}$. Ovaj način rešavanja je uzet za konstrukciju genetskog algoritma za rešavanje jednostrukog problema instalacije neograničenih kapaciteta u [14]. U ovom algoritmu je uzeto $F = 5.5$, a odabir jedinke se vrši na osnovu 3, 5, 6 i 8 uzoraka iz populacije.

```
F_floor = trunc(F);
br_pop = length(stara_populacija);
sr_pop = trunc(br_pop * (1 – frac(F)));
for (i = 0; i < sr_pop; i++)
{
   for (j = 0; j < F_floor; j++)
   {
      k = random(0, br_pop);
      turnir[j] = stara_populacija(k);
   }
   for (j = 0; j < F_floor; j++)
      if F(turnir[j]) > f(nova_populacija[i])
         nova_populacija[i] = turnir[j];
}
for (i = sr_pop; i < br_pop; i++)
{
   for (j = 0; j < F_floor + 1; j++)
   {
      k = random(0, br_pop);
      turnir[j] = stara_populacija(k);
   }
   for (j = 0; j < F_floor + 1; j++)
      if F(turnir[j]) > f(nova_populacija[i])
         nova_populacija[i] = turnir[j];
}
return A';
```

Slika 6.2: Fino gradirana turnirska selekcija gde je razlika veličinâ turnirâ najviše 1

6.3 Ukrštanje

U ovoj implementaciji genetskog algoritma odabrano je trostruko jednopoziciono ukrštanje. Za verovatnoću ukrštanja uzet je parametar $p = 0.85$, što znači da će se u 85% slučajeva ukrštanje izvršiti. U preostalih 15% slučajeva se odmah prelazi na operator mutacije.

Trostruko jednopoziciono ukrštanje se može pojasniti na sledećem primeru. Neka je $K = \{1, 2, \dots, k\}, M = \{1, 2, \dots, m\}, N = \{1, 2, \dots, n\}$, $k = |K| = 5$, $m = |M| = 3$ i $n = |N| = 4$. Dalje, neka su data dva roditelja (za zapis je korišćena ista notacija kao u odeljku 6.1)

[111 || 01] [(0|010) (1|001) || (1|100)] [(01) (01) || (10) (10)],

[010 || 10] [(1|001) (1|010) || (0|001)] [(01) (10) || (01) (10)],

pri čemu je tačka ukrštanja označena sa ||. Najpre se za deo hromozoma koji se vezuje za superkoncetratore razmeni genetski materijal nakon tačke ukrštanja, nakon čega će ti geni za jedno dete biti oblika [11110], a za drugo [01001]. Zatim se u delu za koncentratore razmeni odgovarajući genetski materijal te će odgovarajući geni prvog i drugog deteta biti

[(0|010) (1|001) || (0|001)],

[(1|001) (1|010) || (1|100)],

respektivno. Za sada je novodobijena prva jedinka korektno definisana, dok druga nije. Problem kod drugog potomka je u genu koji opisuje poslednji koncentrator, (1|100), budući da je $(100)_2 = 4 > 3 = p$, gde je p broj uspostavljenih superkoncentratora. To bi značilo da bi u nizu sortiranih superkoncetratora odgovarajućem koncetratoru bio pridružen četvrti po redu, što je nemoguće, budući da taj niz ima 3 člana. Zato se taj broj posmatra po modulu 3, pri čemu je dobijena vrednost 1. Ako bi dobijena vrednost bila 0, na rezultat bi se dodalo 3. Na taj način oba potomka, za sada, imaju dobro definisan genetski materijal, a deo genetskog materijala koji se vezuje za koncetratore će biti oblika

[(0|010) (1|001) || (0|001)],

[(1|001) (1|010) || (1|001)].

Slično, kod genetskog materijala koji se odnosi na terminale nakon tačke ukrštanja se izvrši zamena genetskog materijala. Taj deo hromozoma će za dva novodobijena potmoka imati oblik

[(01) (01) || (01) (10)],

[(01) (10) || (10) (10)].

Budući da je u slučaju prvog deteta uspostavljen jedan koncentrator, a u slučaju drugog dva, vidi se da je prvo dete sada nekorektno definisano, pa je potrebno izvršiti korekciju. Time, analognim postupkom kao u prethodnom slučaju, poslednji terminal, (10), postaje (01), nakon čega su obe jedinke koje označavaju dva potomka korektno definisane i imaju oblik

[11110] [(0|010) (1|001) || (0|001)] [(01) (01) || (01) (01)],

[01001] [(1|001) (1|010) || (1|001)] [(01) (10) || (10) (10)].

Na osnovu ovog primera jasno se vidi kako se vrši trostruko jednopoziciono ukrštanje u opštem slučaju. Treba još napomenuti da se, ukoliko se dobije potomak kod koga nije uspotavljen nijedan koncentrator, odnosno super koncentrator, vrši uspostavljenje proizvnoljno izabranog koncetratora, odnosno superkoncentratora. Nakon obavljenog postupka ukrštanja su sve jedinke uvek korektno definisane.

6.4 Mutacija

U ovom algoritmu je primenjena prosta mutacija sa zaleđenim bitovima. Drugim rečima, ukoliko se desi da na nekom genu sve jedinke populacije imaju isti kôd, povećava se verovatnoća mutacije na tom bitu, budući da se prostor pretrage u tom slučaju smanjuje za jednu celu dimenziju. Time se ovde reaguje radikalnije i prostor pretrage se vraća u okvire normale. Ako je n dužina hromozoma u bitovima, uzeto je da je verovatnoća mutacije jednaka $p_m = 0.05$, dok na zaleđenim bitovima iznosi 0.25. Ukoliko mutacijom jedinka postane nekoretna, postupa se kao u slučaju selekcije, tj. odgovarajući gen se svodi po modulu boja pridruženih objekata.

6.5 Ostali apekti genetskog algoritma

Inicijalizacije populacije je izvršena na slučajan način. U cilju dobijanja jedinki boljeg kvaliteta u pocetnoj populaciji, superkoncetratoru se dodeljuje vrednost 1 sa verovatnoćom 0.35. Slično, uzeta je ista verovatnoća za uspostavljanje koncetratora, pa u 35% slučajeva bitovi dobijaju vrednost 1 na odgovarajućem delu gena. Za određivanje kojim superkoncetratorima su pridruženi koncentratori, uniformno se u skupu prirodnih brojeva u intervalu $[1, 10^9]$ bira broj i posmatra se njegov ostatak po modulu p, gde je p broj uspostavljenih superkoncetratora. Ako je rezultat 0, njegova vrednost se menja u p. Slično

se postupa i za pridruživanje terminala koncentratorima, s tim što se u ovom slučaju posmatra po modulu broja s, gde je s broj uspostavljenih koncentratora.

Kao kriterijum zaustavljanja uzeto je ispunjenje bar jednog od sledeća dva uslova:

1. prekoračen je broj generacija, gde je najveći dozvoljeni broj 10000;
2. u poslednjih 250 generacija uočeno je ponavljanje najboljih jedinki.

Takođe, u algoritmu je primenjen elitizam [15]. U svakoj generaciji se direktno prenosi 1/2 najboljih jedinki. Ovo, sa jedne strane, doprinosi efikasnosti algoritma, a sa druge čuva dobre karatkeristike prethodne generacije. Broj jedinki unutar jedne populacije iznosi 160.

7. POMOĆNE HEURISTIKE

Predloženi genetski algoritam za neke instance ne daje rešenje dovoljno blisko optimalnom, zbog čega je poboljšan primenom dve dodatne heuristike na sve jedinke poslednje populacije.

7.1 Prva pomoćna heuristika

Neka je uspostavljeno p, $1 \leq p \leq k$, superkoncentratora. Najpre se sortiraju opadajuće po ceni $F_i, 1 \leq i \leq p$. Tako se dobija niz dužine p, $k_1, k_2, \dots, k_p$. Taj niz predstavlja zapravo neku permutaciju skupa uspostavljenih superkoncetratora. Zatim, za svaki koncentrator j, $1 \leq j \leq m$, posmatraju se sve moguće cene pridruživanja tog koncentratora superkoncentratorima i sortiraju rastuće, nakon čega se dobija niz $B_{jk_{j1}}, B_{jk_{j2}}, \dots, B_{jk_{jp}}$. Slično, za svaki terminal $i, 1 \leq i \leq n$, cene njegovog pridruživanja svakom koncentratoru se sortiraju rastuće pri čemu se dobija niz $C_{ij_{i1}}, C_{ij_{i2}}, \dots, C_{ij_{is}}$, gde je s broj uspostavljenih koncentratora. Dalje, neka je skup uspostavljenih koncetratora $\{j_1, j_2, \dots, j_s\}$. Svaki član familije skupova

$$\left\{\{k_{j1}, k_{j2}, \dots, k_{jp}\} \middle| 1 \leq j \leq m\right\}$$

predstavlja permutaciju skupa $\{k_1, k_2, \dots, k_p\}$, dok svaki član familije skupova

$$\left\{\{j_{i1}, j_{i2}, \dots, j_{is}\} \middle| 1 \leq i \leq n\right\}$$

predstavlja permutaciju skupa $\{j_1, j_2, \dots, j_s\}$.

Nakon obavljenih sortiranja, postupa se na sledeći način. U i-toj iteraciji $(1 \leq i \leq p)$ pokušava se sa izbacivanjem superkoncentratora k_i iz niza $k_1, k_2, \dots, k_p$ zamenom, za svaki koncetrator, prvim sa njegove liste koja se sastoji sortiranih superkoncentratorâ, pri čemu ukoliko je taj superkoncentrator upravo k_i, pokušava se sa sledećim. Na kraju se uporede funkcije cilja sa i bez superkoncentratora k_i i bira bolja opcija. Ukoliko je druga opcija bolja, izvrše se potrebna pridruživanja.

Dalje, na sličan način se pokušava sa izbacivanjem nekog od koncentratora iz skupa $\{j_1, j_2, \dots, j_s\}$, pa se i-toj iteraciji $(1 \leq i \leq s)$ vrši pokušaj zamene koncentratora j_i, za svaki

terminal, prvim sa liste terminala koja se sastoji sortiranih koncentratorâ, pri čemu ukoliko je taj koncentrator upravo j_i, pokušava se sa sledećim. Na kraju se uporede funkcije cilja sa i bez koncentratora j_i i izabere bolja opcija. Ukoliko je druga opcija bolja, izvrše se potrebna pridruživanja.

Ova heuristika služi da bi se optimizovao broj uspostavljenih superkoncentratora i koncentratora.

7.2 Druga pomoćna heuristika

Neka je uspostavljeno p superkoncentratora i s koncentratora. Najpre se sortira svaki od uspostavljenih superkoncentratora opadajuće po ceni njihove instalacije, a niz neuspostavljenih superkoncentratura rastuće. Tako se dobijaju skupovi $\{u_1, u_2, \dots, u_p\}$ i $\{\overline{u_1}, \overline{u_2}, \dots, \overline{u_{k-p}}\} = K \backslash \{u_1, u_2, \dots, u_p\}$. U i-toj iteraciji, gde je $1 \leq i \leq \min\{p,\ k-p\}$, pokušava se sa zamenom superkoncetratora u_i sa neuspostavljenim $\overline{u_i}$ i, ukoliko novodobijena jedinka ima bolje karakteristike, izvrše se potrebna ažuriranja. Slično, prave se skupovi uspostavljenih i neuspostavljenih koncentratora, $\{v_1, v_2, \dots, v_p\}$ i $\{\overline{v_1}, \overline{v_2}, \dots, \overline{v_{m-p}}\} = M \backslash \{v_1, v_2, \dots, v_p\}$. U j-toj iteraciji, gde je $1 \leq j \leq \min\{p,\ m-p\}$, pokušava se sa zamenom koncetratora u_j sa neuspostavljenim $\overline{u_j}$ i, ukoliko novodobjena jedinka ima bolje karakteristike, izvrše se potrebna ažuriranja.

7.3 Šema predloženog genetskog algoritma sa pomoćnim heuristikama

Na osnovu opisa, šema celokupnog genetskog algoritma sa pomoćnim heuristikama se može prikazati pseudokodom na slikama 7.1 i 7.2. Najpre se izvši inicijalizacija, gde se jedinke prve populacije generišu proizvoljno. Zatim se, sve dok nije ispunjen kriterijum zaustavljanja, vrši primena genetskih operatora nad trenutnom populacijom. Najpre se izvrši selekcija, gde se u prvoj varijanti algoritma koristi obična turnirska, a u drugoj fino gradirana turnirska selekcija, a nakon toga trostruko jednopoziciono ukrštanje i prosta mutacija sa saleđenim bitovima. Kriterijumi zaustavljanja su prekoračen maksimalan broj generacija i ponavljanje najboljih jedinki u dovoljnom broju generacija. Nakon samog algoritma, sledi primena dve heuristike, jedne za drugom.

```
inicijalizacija();
while (!kriterijum_zaustavljanja())
{
   turnirska_selekcija();
   trostruko_jednopoziciono_ukrštanje();
   prosta_mutacija_sa_zaleđenim_bitovima();
}
prva_heuristika();
druga_heuristika();
```

Slika 7.1: Kratak pregled prve varijante genetskog algoritma

```
inicijalizacija();
while (!kriterijum_zaustavljanja())
{
   fino_gradirana_turnirska_selekcija();
   trostruko_jednopoziciono_ukrštanje();
   prosta_mutacija_sa_zaleđenim_bitovima();
}
prva_heuristika();
druga_heuristika();
```

Slika 7.2: Kratak pregled druge varijante genetskog algoritma

8. ANALIZA REZULTATA

8.1 Implementacija i test instance

Za rešavanje dagot problema napravljena su tri programa. Prva implementacija je dobijena ugradnjom paketa IBM ILOG CPLEX u projekat sa glavnim programom na jeziku C. Ova implementacija rešava problem formulisan u obliku celobrojnog programiranja. Druge dve implementacije su dve varijante genetskog algoritma koje se razlikuju u izboru operatora selekcije. U prvom slučaju je implementirana obična, a u drugom fino gradirana turnirska selekcija. Ove varijante genetskog algoritma implementirane su na programskom jeziku C i kompajlirane na platformi Visual Studio 2010 Ultimate. Svi eksperimenti su izvršeni na personalnom računaru koji koristi procesor Intel na radnom taktu 3GHz pod Windows XP operativnim sistemom.

Kako test instance za ovaj problem nisu bile dostupne na internetu, one su generisane. Pod veličinom instance podrazumeva se broj elemenata skupa N, tj. broj terminala. Generisano je ukupno 40 test instanci, od kojih je za prvih 20 ispunjeno

$$|N| = |M| = |K| \leq 15,$$

budući da veće instance CPLEX ne može da reši. Na tim instancama svaka od dve varijante genetskog algoritma je upoređena sa CPLEX-om. Za svaku test instancu i implementaciju genetskog algoritma program je izvršavan po 10 puta, i odgovarajuće srednje vrednosti su prikazane u tabelama.

Od pomenutih 40, 30 test instanci je generisano na način koji je predložen u [26]. Neka $\lceil x \rceil$ označava najmanji ceo broj veći ili jednak od x, i neka $\mathbb{U}[a, b]$ označava uniformnu raspodelu na intervalu $[a, b]$. Koordinate terminala generišu se nezavisno sa uniformnom raspodelom verovatnoća

$$x_i, y_j \in \mathbb{U}[0, 10000]\,, \quad 1 \leq i, j \leq |N|,$$

a ostali parametri određeni su koordinatama terminala

$$c_{ij} = \left\lceil \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right\rceil, \quad 1 \leq i, j \leq |N|,$$

$$d_{jk} = 1000 + c_{jk}, \quad 1 \leq j, k \leq |N|,$$

$$f_k = 1000 + c_{k1}, \ \ 1 \leq k \leq |N|,$$

Uzeto je da važi

$$1 \leq |N| = |M| = |K| \leq 500,$$

pri čemu instance veličine preko 150 do sada nisu rešavane u literaturi.

Generisano je i 10 test instanci za koje je

$$500 \geq |N| > |M| > |K| > 15,$$

imajući u vidu hijerarhijsku strukturu telekomunikacionih mreža u praksi, gde je obično broj terminala veći od broja koncentratora, a broj koncentratora veći od broja superkoncentratora.

8.2 Rezultati na manjim test instancama

Rezultati za 20 malih test instanci prikazani su u tabeli 8.1. Za ove instance rešavač CPLEX i genetski algoritmi dali su iste, optimalne rezultate, koji su zajedno sa vremenima izvršavanja prikazani u tabeli u koloni rešenja.

Veličine instanci variraju — prvih 6 su veličine 5, narednih 7 veličine 10, a poslednjih 7 veličine 15. Kolone $t(\mathrm{GA}_1)$ i $t(\mathrm{GA}_2)$ sadrže vremena izvršavanja genetskog algoritma koji koristi običnu, odnosno fino gradiranu turnirsku selekciju. Vreme nalaženja rešenja CPLEX-om je označeno sa $t(\mathrm{CPLEX})$.

Primena CPLEX-a je ovde bitna, u cilju upoređivanja rezultata sa genetskim algoritmom. Obe varijante genetskog algoritma za sve ove instance pronalaze optimalne rezultate za mnogo kraće vreme nego CPLEX.

Ulaz	N	M	K	Rešenje	t(CPLEX)	$t(GA_1)$	$t(GA_2)$
ulaz_1_01	5	5	5	4306	3.213	0.623	0.617
ulaz_1_02	5	5	5	5208	3.117	0.551	0.566
ulaz_1_03	5	5	5	4994	3.338	0.778	0.721
ulaz_1_04	5	5	5	5123	3.117	0.714	0.751
ulaz_1_05	5	5	5	5691	3.400	0.633	0.634
ulaz_1_06	5	5	5	5770	3.102	0.716	0.669
ulaz_1_07	10	10	10	11362	6.446	1.517	1.521
ulaz_1_08	10	10	10	10306	7.009	1.221	1.113
ulaz_1_09	10	10	10	9508	6.431	1.094	0.984
ulaz_1_10	10	10	10	9223	6.212	1.212	1.117
ulaz_1_11	10	10	10	8994	6.993	1.443	1.484
ulaz_1_12	10	10	10	10813	7.330	1.627	1.556
ulaz_1_13	10	10	10	11253	6.139	1.119	1.024
ulaz_1_14	15	15	15	13840	12.517	2.133	2.021
ulaz_1_15	15	15	15	15625	11.496	2.006	2.140
ulaz_1_16	15	15	15	14536	13.088	1.889	1.743
ulaz_1_17	15	15	15	14903	13.964	1.603	1.587
ulaz_1_18	15	15	15	15117	12.273	1.704	1.742
ulaz_1_19	15	15	15	13276	14.043	2.124	2.572
ulaz_1_20	15	15	15	15224	12.535	1.273	1.664

Tabela 8.1: Rezultati na manjim test instancama (CPLEX, GA_1 i GA_2)

8.3 Rezultati na većim test instancama

Test instanci sa većim vrednostima ulaznih veličina ima takođe 20. Za 15 od tih 20 instanci važi $|N| = |M| = |K|$, dok je za preostalih 5 ispunjeno $|N| \geq |M| > |K|$. U tabeli 8.2 su prikazane vrednosti dobijene testiranjem odgovarajućih genetskih algoritma, kao i vremena izvršavanja. Instance ove veličine CPLEX ne uspeva da reši. Dve implementirane verzije genetskog algoritma su dakle upoređene za instance veličine do 500. Za prvih 7 test (za koje važi $|N| \leq 150$) instanci dobijeni rezultati su slični rezultatima dobijenim u [26][1], gde je primenjen isti princip njihovog generisanja.

[1] Na ovaj način, rezultati su mogli biti upoređeni sa onima u [26], budući da test instance za ovaj problem nisu javno dostpune

Veličine instanci variraju od 20 do 500. Kolone GA_1 i GA_2 sadrže rezultate dobijene primenom genetskog algoritma koji koristi običnu, odnosno fino gradiranu turnirsku selekciju, dok su $t(GA_1)$ i $t(GA_2)$ odgovarajuća vremena izvršavanja. Iz tabele se može uočiti da dve varijante genetskog algoritma daju približno iste rezultate, pri čemu je bolje rezultate u nekim slučajevima davala obična, a u drugim fino gradirana turnirska selekcija.

Ulaz	N	M	K	GA_1	$t(GA_1)$	GA_2	$t(GA_2)$
ulaz_2_01	20	20	20	18733	4.301	18733	4.006
ulaz_2_02	50	50	50	50384	16.311	49942	15.743
ulaz_2_03	50	50	50	52388	15.994	53004	16.301
ulaz_2_04	100	100	100	104416	32.007	102103	33.640
ulaz_2_05	100	100	100	99383	31.045	100779	29.994
ulaz_2_06	150	150	150	156250	44.360	150336	43.214
ulaz_2_07	150	150	150	138413	42.221	135201	38.113
ulaz_2_08	200	200	200	208333	60.223	206986	62.419
ulaz_2_09	200	200	200	196223	65.021	190788	66.228
ulaz_2_10	300	300	300	312553	88.090	319403	90.330
ulaz_2_11	300	300	300	303813	82.211	299410	92.188
ulaz_2_12	400	400	400	416738	131.084	414556	125.513
ulaz_2_13	400	400	400	402113	117.883	398994	122.697
ulaz_2_14	500	500	500	520086	166.220	510339	173.228
ulaz_215	500	500	500	491693	155.011	503220	160.106
ulaz_2_16	150	100	50	97926	32.981	93513	34.292
ulaz_2_17	200	100	50	116320	35.598	110516	38.220
ulaz_2_18	300	150	50	147313	51.270	156251	56.334
ulaz_2_19	400	200	100	220833	74.336	209318	82.216
ulaz_2_20	500	300	100	296310	91.073	287554	97.730

Tabela 8.2: Rezultati na većim test instancama (GA_1 i GA_2)

9. ZAKLJUČAK

U ovom radu prikazana je implementacija genetskog algoritma za dvostepeni problem instalacije neograničenih kapaciteta. Dat je opis problema, opis implementacije genetskog algoritma i analiza rezultata. Prikazane su dve vrste genetskog algoritma, obe bazirane na odgovarajućim tipovima selekcije — običnoj i fino gradiranoj turnirskoj selekciji. Korišćena je prosta mutacija i trostruko jednopoziciono ukrštanje. Upotrebljeno je binarno kodiranje, implementirano na specifičan način, koji je prilagođen ovom problemu.

Naučni doprinos rada se ogleda u činjenici da je prvi put za dvostepeni problem instalacije neograničenih kapaciteta upotrebljen genetski algoritam. Pod veličinom instance podrazumeva se broj elemenata skupa N, tj. broj terminala. Za male instance koje rešavač CPLEX uspeva da reši, obe verzije genetskog algoritma pronalaze optimalne rezultate. Za deo većih instanci, koje su veličine do 150 i generisane na način opisan u [26], dobijeni rezultati su istog reda veličine kao rezultati dobijeni u [26]. Prvi put su testirane instance veličine veće od 150 (do 500).

Dalji rad se može odvijati u više pravaca, među kojima su:

1. paralelizacija predloženog genetskog algoritma što olakšava činjenica da je korišćena turnirska selekcija;
2. konstruisanje drugih tipova genetskih algoritama za rešavanje ovog problema;
3. rešavanje srodnih problema, kao što su
   - jednostepeni problem instalacije neograničenih kapaciteta,
   - jednostepeni problem instalacije ograničenih kapaciteta,
   - dvostepeni problem instalacije ograničenih kapaciteta,
   - višestepeni problem instalacije neograničenih kapaciteta,
   - višestepeni problem instalacije ograničenih kapaciteta,

   predloženim genetskim algoritmom, bez obzira da li su ili nisu rešavani nekim genetskim algoritmom;
4. rešavanje razmatranog problema i njemu srodnih nekom drugom heuristikom ili kombinacijom genetskog algoritma i neke druge heuristike.

LITERATURA

[1] M. Boccia, T. G. Crainic, A. Sforza, C. Sterle: *A Metaheuristic for a Two Echelon Location-Rounting Problem*, Procceding of SEA2010. Lecture Notes in Computer Science, 6049/2010, 288-301, 2010.

[2] G. R. Matheus, F. R .B. Cruz, H. P. L. Luna: *An Algorithm for Hierarchical Network Design*, Location Science, (2), 149-164, 1994.

[3] J. Current, H. Pirkul: *The Hierarchical Network Design Problem with Transshipment Facilities*, Eorupoean Journal of Operation Research, (52), 338-347, 1991.

[4] M. P. Helme, T. L. Magnanti: *Designing Satellite Communication Networks by Zero-One Quadratic Programming*, Networks, (19), 427-450, 1989.

[5] S. Chung, Y. Myung, D. Tcha: *Optimal Design of a Distributed Network with a Two-Level Hierarchical Structure*, European Journal of Operations Research, (62), 105-115, 1992.

[6] A. Colorni, M. Dorigo, V. Maniezzo: *Distributed Optimization by Ant Colonies*, Actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[7] J. Kennedy, R. Eberhart: *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948., 1995.

[8] J. H. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press., 1975.

[9] A. Lipowski, D. Lipowska: *Roulette-wheel selection via stochastic acceptance*, Adam Mickiewicz University, Poland, 2011.

[10] E. Gourfin, M. Labbe, H. Yaman: *Telecommunication and location*, Universitete de Bruxelles, Belgium, 2001.

[11] M. Landete, A. Marín: *New facets for the two-stage uncapacitated facility location polytope*, Computational Optimization and Applications 44, 487-519, 2009.

[12] V. Filipović: *Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama*, magistarski rad, Matematički fakultet, Beograd, 1998.

[13] B. L. Miller, D. E. Goldberg: *Genetic algorithms, Tournament Selection, and the Effects of Noise*, University of Illinois, 1995.

[14] V. Filipović, J. Kratica, D. Tošić, I. Ljubić: *Fine Grained Tournament Selection for the Sim-ple Plant Location Problem*, Proceedings of the 5th Online World Conference on Soft Computing in Indust-rial Applications WSC5, pp. 152-158, 2000.

[15] J. M. Y. Leung, T. L. Magnanti: *Valid Inequalities and Facets of the Capacitated Plant Location Problem*, Mathematical Programming, (44), 271-291, 1989.

[16] M. Živković: *Algoritmi*, Matematički fakultet, Beograd, 2000.

[17] D. Adnađević, Z. Kadelburg: *Matematička analiza 1*, Matematički fakultet, Beograd, 2004.

[18] S. Arora, B. Barak: *Computational Complexity: A Modern Approach,* Princeton University, 2007.

[19] L. Fortnow: *The status of the P versus NP problem*, Communications of the ACM 52, no. 9, pp. 78–86., 2009.

[20] A. M. Jaffe: *The Millennium Grand Challenge in Mathematics*, Clay Mathematics Institute, 2004.

[21] Z. Ognjanović, N. Krdžavac: *Uvod u teorijsko računarstvo*, Beograd, 2004.

[22] E. Demaine, S. Goldwasser: *Introduction to Algorithms*, Massachusetts Institute of Technology, 2004.

[23] A. Christina: *The 3-Dimensional Matching Problem*, Athens University of Economics, 2009.

[24] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi: *Optimization by Simulated Annealing*, Science 220 (4598): 671-680, 1983.

[25] F. Glover, M. Laguna: *Tabu Search*, Kluwer, Norwell, MA, 1987.

[26] P. Chardaire, A. Suttet, M.C. Costa: *Solving the Dynamic Facility Location Problem*, Networks, (28), 117-124, 1996.

[27] B. L. Wildbore: *Theoretical and Computational Analysis of the Two–Stage Capacitated Location Problem*, PhD thesis, Massey University, Palmerston North, New Zeland, 2008.

[28] J. Kratica: *Parallelization of the Genetic Algorithms for Solving Some NP-complete Problems*, PhD thesis, University of Belgrade, Faculty of Mathema-tics, Department of Computer Science, 2000.

[29] D. Cvetković, M. Čangalović, Đ. Dugošija, V. Kovačević-Vujčić, S. Simić, J. Vuleta: *Kombinatorna optimizacija*, Beograd, 1996.

[30] J. Krarup, P. M. Pruzan: *The Simple Plant Location Problem: Survey and Synthesis*, Europeran Journal of Operations Research, (12), 36-81, 1983.

[31] Y. Lee, B. H. Lim, J. S. Park: *A Hub Location Problem in Desinging Digital Data Service Networks: Langrangian Relaxation Approach*, Location Science, (4), 185-194, 1996.

[32] H. Pirkul, V. Nagarajan: *Location Concentrators in Centralized Computer Networks*, Annals of Operations Research, (36), 247-262, 1992.

[33] A. Mirzian: *Langrangian Relaxation for the Star-Star Concetrator Location Problem: Approximation algorithm and Bounds,* Networks, (15), 1-20, 1985.

[34] J. G. Klincewicz, H. Luss: *A Lagrangian Relaxation Heuristic for Capacitated Facility Location with Single Source Constraints*, Journal of Operational Research Society, (37), 495-500, 1986.

[35] O. Kariv, S. Hakimi: *An algorithmic approach to network location problems, Part I: thepcenters*, Unpublished Manuscript, 1976.

[36] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi: *The Bees Algorithm. Technical Note*, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[37] O. Braysy: *Local Search and Variable Neighborhood Search Algorithms for the Vehicle Routing with Time Windows.* Acta Wasaensia 87, Universitas Wasaenis, Vaasa, 2000.

[38] P. Chardaire, J. L. Luton, A. Sutter: Upper and lower bounds for the two-level simple plant location problem, Annals of Operation Research, (86), 117-140, 1999.

[39] R. Dawkins: *The Selfish Gene*, Oxford University Press, 1989.