SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVOD ZA ELEKTRONIKU, MIKROELEKTRONIKU, RAČUNALNE I INTELIGENTNE SUSTAVE

SEMINARSKI RAD

PRIMJENA GENETSKIH ALGORITAMA U MREŽNOM PLANIRANJU

Toni Frankola

Mentor: doc. dr. sc. Marin Golub

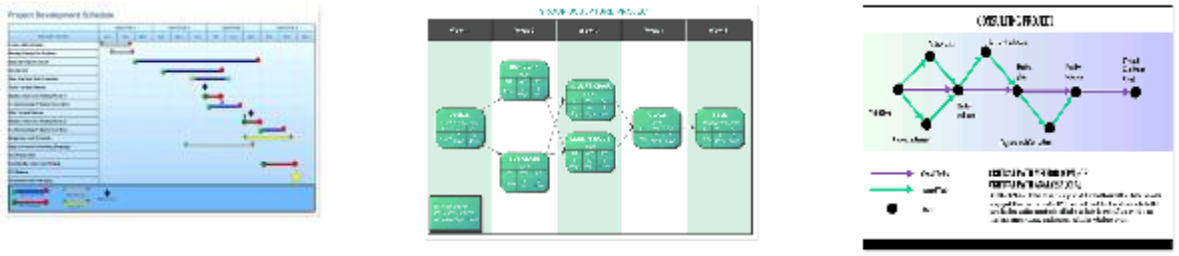
Zagreb, 31. kolovoz 2005.

Sadržaj

1. Uvod

U svakodnevnom životu suočeni smo sa stalnom potrebom optimalnog korištenja računalnih sredstava i smanjenja projektnih troškova. Optimalno korištenje može se postići samo dobrim planiranjem projektnih aktivnosti. Planiranje složenih projekta s mnoštvom aktivnosti vrlo je teško bez pomoći računala.

Planiranje aktivnosti se u projektnim planovima najčešće modelira odgovarajućim grafom tj. mrežnim planom. Mrežni plan prikazuje slijed kojim se aktivnosti obavljaju, njihove zavisnosti i njihova trajanja. Neki od oblika mrežnih planova su Gantt, Pert i CPM. Njihovi dijagrami su prikazani su na slici 1.1.



(a) (b) (c)

Slika 1.1. – Osnovni oblici mrežnih dijagrama u projektnom menadžmentu. (a) – Gantt, (b) – PERT i (c) CPM

Projektni menadžer će se tijekom izrade projektnog plana najčešće susretati sa sljedećim problemima:

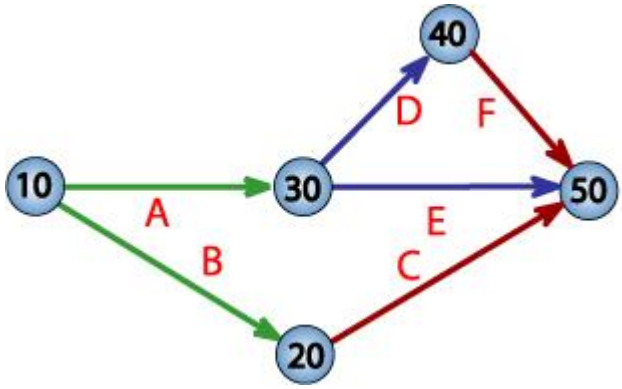
- struktura mrežnog plana, to jest kako poredati aktivnosti na projektu i kako modelirati njihove zavisnosti; koje će se aktivnosti izvoditi paralelno, a koje serijski;
- koje resurse koristiti i kada ih koristiti;
- koji su planirani datumi za dovršenje aktivnosti u projektnom planu;
- koji je kritični put koji usporava čitav projekt.

U ovom će seminarskom radu biti opisani mogući načini primjene genetskih algoritama za optimiranje projektnih planova ali i nekih drugih problema koji se mogu formulirati mrežnim planom (primjerice transportni problem). Osnova za izgradnju genetskih algoritama su rješenja tih istih problema linearnim programiranjem koji su većim dijelom opisana u [Kal96]. Pri optimiranju mrežnih planova optimiranje će se provoditi samo za količinu aktivnosti koje su pridružene pojedinim granama mrežnog plana. Struktura mrežnih planova bila je unaprijed zadana i nepromjenjiva.

2. Mrežno planiranje

Kvalitetno upravljanje složenim projektima uvelike ovisi o pažljivom planiranju svih aktivnosti tijekom projekta. Tijekom jednog projekta aktivnosti mogu biti brojne, a početci/završetci pojedinih aktivnosti mogu imati različite odnose (neke aktivnosti čekaju kraj prethodne aktivnosti da bi započele, a druge aktivnosti započinju istovremeno s tom aktivnošću). Aktivnosti se, dakle, mogu odvijati paralelno i serijski.

Mrežni plan se prikazuje kao usmjereni graf. Primjer jednog jednostavnog mrežnog plana dan je na Slici 2.1. Pri modeliranju mrežnih planova aktivnosti se mogu prikazivati u granama ili u čvorovima. Osim jednostavnih informacija o slijedu pojedinih aktivnosti i njihovom trajanju, na grafovima se mogu upisivati i dodatne informacije o vremenima početka i završetka ove aktivnosti, najranijem početku i završetku i slično.



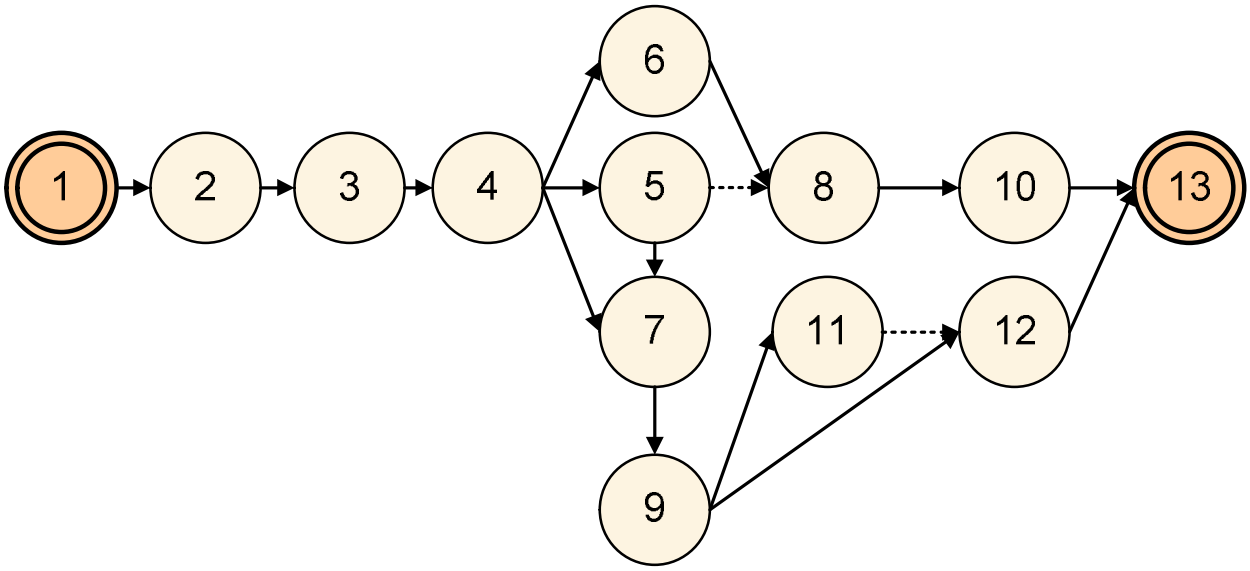
Slika 2.1. – Jednostavni mrežni plan sa aktivnostima u granama. U čvorovima je upisano vrijeme završetka pojedine aktivnosti

Krajem pedesetih godina prošlog stoljeća razvijene su metode mrežnog planiranja pomoću kojih je moguće lakše i pouzdanije pratiti odvijanje projekta, predvidjeti moguće izvore problema i koordinirati aktivnosti na projektu tako da se izbjegne kašnjenje. Od mnogih metoda najpoznatije i najčešće korištene su metode PERT (Program Evaluation and Review Technique) i CPM (Critical Path Method) [Kal96]. Obje metode pomažu nam u boljoj identifikaciji trajanja pojedine aktivnosti kao i mogućim načinima skraćenja trajanja cijelog projekta.

2.1. Optimiranje jednostavnog mrežnog problema linearnim programiranjem

Optimiranje jednostavnih mrežnih problema, ako pretpostavimo linearne zavisnosti, može se jednostavno optimirati odgovarajućim linearnim modelom [Kal96]. Jednostavno optimiranje može kao cilj imati ukupno skraćenje trajanja projekta angažiranjem dodatnih resursa (npr. uvođenje druge smjene ili više strojeva). Najčešće će se „ubrzavati“ one aktivnosti koje se nalaze na kritičnom putu koji usporava odvijanje svih ostalih aktivnost. Uvođenje dodatnih resursa automatski povlači i dodatne troškove. Zbog toga treba naći ono optimalno rješenje kod kojeg će dobrobiti (prihodi) od skraćenja ukupnog trajanja biti veće nego cijena dodatnih resursa. Skraćenje vremena trajanja projekta vrlo je kompleksno jer se ne smije gledati samo skraćenje vremena na jednoj

grani nego se mora gledati na projektnom planu u cjelini. S obzirom da su elementi projektnog plana međusobno isprepleteni, samo se optimiranjem cijelog plana može pronaći optimalno rješenje. Na slici 2.2 prikazan je primjer jednog projekta čije su aktivnosti prikazane grafom.



Slika 2.2. – Mrežni problem prikazan grafom. Bridovi grafa koji su označeni punim linijama predstavljaju stvarne aktivnosti. Bridovi grafa označeni isprekidanom linijom predstavljaju fiktivne aktivnosti.

U tablici 2.1 dani su podatci za ovaj mrežni plan. Podatci uključuju: trajanje pojedine aktivnosti, mogućnosti skraćenja i trošak smanjenja po jedinici vremena. Trošak projekta je 200 novčanih jedinica po jedinici vremena.

Cilj optimiranja je pronaći ono rješenje koji za zadani mrežni plan daje najmanje troškove.

Aktivnost (i-j)	Normalno trajanje	Skraćeno trajanje	Jedinični trošak skraćenja
1-2	2	1	300
2-3	4	3	200
3-4	10	7	300
4-5	4	2	350
4-6	6	4	150
4-7	7	5	100
5-7	5	3	150
5-8	-	-	-

Aktivnost (i-j)	Normalno trajanje	Skraćeno trajanje	Jedinični trošak skraćenja
6-8	7	4	200
7-9	8	6	100
8-10	9	5	100
9-11	4	3	100
9-12	5	3	150
10-13	2	1	100
11-12	-	-	-
12-13	6	3	133.3

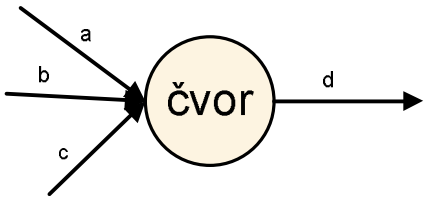
Tablica 2.1. – Aktivnosti jednostavnog mrežnog plana

Cilj optimiranja je završiti projekt u što kraćem roku sa što manjim troškovima. U tom je slučaju funkcija cilja:

$$\min Troskovi = f \cdot t_n + \sum_{i,j} c_{ij}(n_{ij} - t_{ij}) \qquad (2.1)$$

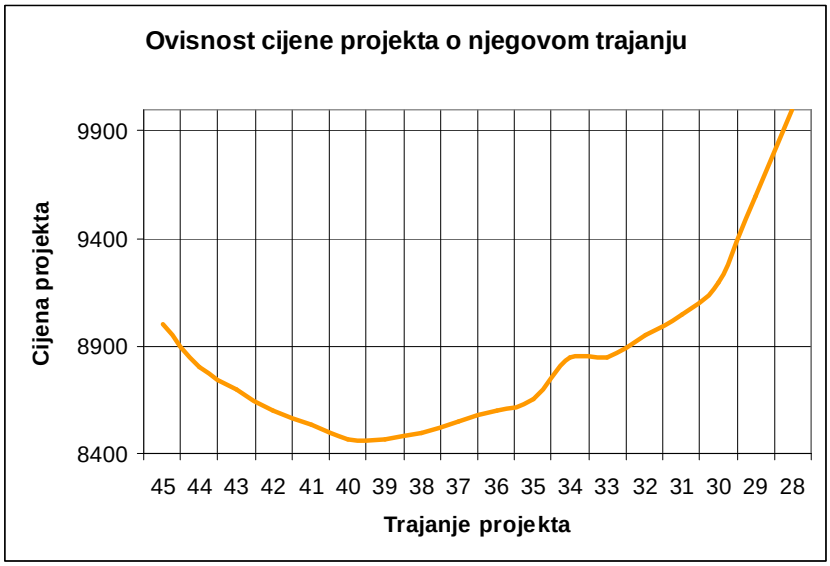
Za opisani problem ograničenja su sljedeća:

1. Vrijeme trajanja pojedine aktivnost je u intervalu [Skraćeno trajanje, Normalno trajanje]
2. Aktivnost koja slijedi nakon čvora ne može početi prije nego završe sve aktivnosti koje ulaze u taj čvor. (Slika 2.3.)



Slika 2.3. – Aktivnost d ne može početi prije nego što završe aktivnosti a, b i c

Ako se pretpostavi linearna zavisnost između skraćenja trajanja i povećanja troškova. tada je problem moguće riješiti linearnim programiranjem. Pritom se dobivaju rezultati prikazani na slici 2.4.



Slika 2.4. – Prikaz zavisnosti troškova projekta i trajanja projekta

Troškovi bi bili najmanji ako projekt traje između 39 i 40 vremenskih jedinica.

3. Genetski algoritmi

Genetski algoritam je heuristička metoda optimiranja koja je zasnovana na evoluciji živih bića. Genetski algoritmi predloženi su od strane Johna H. Hollanda još u ranim sedamdesetim godinama prošlog stoljeća. Osnovna je ideja ovih algoritama kopiranje rada prirodnog evolucijskog procesa.

Problemi čija optimalna rješenja pokušavamo pronaći nije moguće riješiti klasičnim metodama pretraživanja jer je prostor stanja problema najčešće prevelik da bi računalo moglo pretražiti sva rješenja u nekom razumnom vremenu.

Ako promatramo evoluciju kao „algoritam“ koji traži optimalno rješenje (cilj je stvoriti što naprednji organizam), onda možemo reći da je prirodna evolucija vrlo uspješan „algoritam“. Nakon 4.5 milijarde godina izvođenja algoritam i dalje producira kvalitetna rješenja. Živa bića danas rješavaju najveći broj problema, uključujući mnoge probleme koje današnja računala ne mogu riješiti.

Upravo snaga prirodne evolucije potakla je brojne autore da u prirodi potraže inspiraciju za osmišljanje novih optimizacijskih algoritama. Osim genetskih algoritam, tako su nastale evolucijske strategije i simulirano kaljenje, ali i brojne druge metode umjetne inteligencije.

3.1. Evolucija u prirodi

Evoluciju u prirodi prvi je započeo proučavati britanski znanstvenik Charles Darwin u 19. stoljeću. On je u svom dijelu „Porijeklo vrsta“ postavio temelje današnje znanosti o biološkoj evoluciji.

Darwin je u svojoj knjizi predstavio 5 ključnih značajki evolucijskog procesa:

1. Sve vrste produciraju više potomaka od trenutnog broja jedinki u jednog generaciji
2. Tijekom vremena prosječna veličina populacije neke vrste ostaje uvijek jednaka (postoji samo blagi trend rasta) neovisno o velikom broju potomaka
3. Dostupna hrana (resursi) je ograničena, ali njena količina je većinom konstantna kroz određeni period.

Iz prethodne tri tvrdnje vidljivo je da se jedinka pojedine vrste mora boriti kako bi preživjela.

4. Kod vrsta koje se razmnožavaju spolno, općenito gledajući, nema dvije jedinke koje su jednake: jedinke nasljeđuju svojstva od svojih roditelja uz poneku varijaciju (mutaciju).
5. Varijacije su često nasljedne.

Svaka se jedinka mora „boriti“ da bi preživjela. Jedinka sa najboljim svojstvima će biti u mogućnosti da preživi. Dobra svojstva nastaviti će se prenositi dalje na njeno potomstvo. Tijekom vremena dobra svojstva će postati dominantna i proširit će se populacijom.

Evolucija je vođena dvama osnovnim procesima:

- Prirodna selekcija
  Selekcija je proces kojim se vrši odabir jedinki iz populacije koje će preživjeti i koje će se dalje reproducirati.
- Spolna reprodukcija
  Proces reprodukcije osigurava da svaka nova jedinka od roditelja naslijedi neka svojstva. Tijekom toga procesa događaju se i mutacije koje pogoduju stvaranju različitosti genetskog materijala. Evolucijski proces znatno je kvalitetniji kada u reprodukciji sudjeluju dva roditelja nego kada je samo jedan, kao kod nespolnog razmnožavanja. Evolucijske procese kod organizama koji se razmnožavaju nespolno moguće je vršiti samo mutacijama.

3.2. Struktura jednostavnog genetskog algoritma

Algoritam jednostavnog genetskog algoritma prikazan je na slici 3.1.

```
Genetski_algoritam(Velicina_Populacije, t, pm, pc, M)
{
   t = 0;
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
      t = t + 1;
      selektiraj P'(t) iz P(t-1);
      križaj jedinke iz P'(t) i djecu spremi P(t);
      mutiraj jedinke P(t);
   }
   ispiši rješenje;
}
```

Slika 3.1. – Struktura jednostavnog genetskog algoritma

Parametri su genetskog algoritma:

- Veličina populacije (broj jedinki u jednog generaciji)
- t (Uvjet završetka genetskog algoritma. Najčešće se iskazuje kao broj iteracija algoritma, vrijeme trajanja ili broj iteracija u kojima nije došlo do promjene najbolje jedinke.)
- p_m (vjerojatnost mutacije)
- p_c (vjerojatnost križanja)
- M (mortalitet)

3.3. Proces evolucije kao motivacija za genetski algoritam

U ovom poglavlju uspoređeni su osnovni mehanizmi koji se koriste u procesu evolucije sa onima koji se koriste u genetskim algoritmima.

Osnovni parametri koji određuju evoluciju, pa tako i genetske algoritme, su: način kodiranja svojstva jedne jedinke, način evaluacije dobrote pojedine jedinke, način selekcije, križanja i mutacije.

3.3.1. Kodiranja svojstava jedinke

Kodiranje u prirodi

Danas se pretpostavlja da su sva svojstva živog bića pohranjena u kromosomima. Kromosomi su lančaste tvorevine koje se nalaze u jezgri svake stanice. Skup informacija, tj. djelić kromosoma u kojem je pohranjeno jedno svojstvo, naziva se gen.

Kemijska struktura koja je prisutna u kromosomima je molekula DNA (eng. deoxyribonucleic acid – deoksiribonukleinska kiselina). To je dvolančana molekula u kojoj je pohranjen „kod“ (upute) kako izgraditi organizam. Između dva lanca te molekule nalaze se četiri komplementarne kiseline: adenin(A) + timin(T) i citozin(C) + guanin(G). Pomoću tih kiselina moguće je ostvariti sistem kodiranja zasnovan na četverobrojnom sustavu. “Brojevi” su A+T, T+A, C+G i G+C.

Pojednostavljen prikaz DNA molekule je na slici 3.2.

Slika 3.2. – Pojednostavljeni prikaz molekule DNA

Da bi se neka informacija dekodirala dovoljno je promatrati samo jedan lanac, jer su kiseline koje se nalaze u drugom lancu uvijek komplementarne s kiselinama u prvom. Postojanje komplementarnih lanaca omogućava dupliciranje molekule DNA.

Točno određeni dijelovi molekule dekodiraju se u gen koji opisuje neko svojstvo. Istraživanjem nizova dušičnih kiselina utvrđene su kombinacije od četiri kiseline na jednom lancu koje odgovaraju jednom od 20 proteina od kojih su izgrađena živa bića. Smatra se da je tek dio kiselina u lancu DNA zaslužan za kodiranje korisnih informacija, a prema sadašnjim istraživanjima, veliki dio informacija koji je pohranjen u DNA predstavlja neki oblik redundantnih informacija.

Kodiranje u genetskom algoritmu

Da bi se neki problem mogao rješavati genetskim algoritmima potrebno je odabrati način prikaza jedne jedinke u računalu. O način prikaza jedne jedinke uvelike ovisi kvaliteta genetskog algoritma i njegova mogućnost da pronađe najbolje rješenje.

Postoji mnoštvo načina kako kodirati kromosome u računalu. Najčešći je prikaz kromosoma kao slijeda binarnih brojeva. U praksi se pokazalo da upravo ovaj prikaz daje najbolje rezultate.

Kod binarnog prikaza kromosom je prikazan kao binarni vektor $x \in [dg,gg]$. Pritom vektor B=000…00 predstavlja donju granicu, a vektor B=111…11 gornju granicu. Dužina vektora n, definira broj rješenja koja se mogu prikazati. Pretvorba brojeva vrši se sljedećim izrazima [Gol04]:

$$x = dg + \frac{b}{\left(2^n - 1\right)}(gg - dg) \qquad b = \frac{x - dg}{gg - dg}(2^n - 1) \qquad (3.1)(3.2)$$

Osim binarnog kodiranja, kod optimiranja problema sa realnim brojevima moguće je koristiti i prikaze kao što je prikaz realnog broja sa pomičnom točkom (npr. prema IEEE 754-1985 standardu).

Za probleme u rasporedu moguće je koristiti i nizove cijelih brojeva kao prikaz kromosoma. Primjerice za rješavanje problema trgovačkog putnika kao kromosom može se odabrati niz cijelih brojeva koji predstavljaju slijed niza gradova koje putnik mora obići.

Osim ovdje navedenih postoje i mnoge druge reprezentacije, kao npr. reprezentacije zasnovane na poljima, stablima, listama i slično.

Odabir prikaza kromosoma uvjetuje izvedbu genetskih operatora kao što su križanje i mutacija.

3.3.2. Evaluacija dobrote pojedine jedinke

Evaluacija dobrote u prirodi

Evaluacija dobrote u prirodi ne postoji kao točno definirani matematički izraz. Ne možemo reći da je jedna jedinka u nekoj populaciji u potpunosti bolja od druge. Prirodno okruženje je dinamično i ono stalno evaluira jedinke različitim „izazovima“.

Evaluacija dobrote u genetskom algoritmu

Da bismo olakšali simulaciju prirodnog okruženja potrebno je prilikom implementacije genetskog algoritama ispravno odabrati način kako ćemo ocjenjivati dobrotu pojedine jedinke. Dobrota neke jedinke je mjera kvalitete toga rješenja u zadanom prostoru rješenja. Kod jednostavnih problema, kao što je pronalaženje minimuma neke funkcije, dobrotu možemo prikazati kao vrijednost koju ta funkcija ima za zadani kromosom. Kod složenijih problema dobrota se može izraziti kao npr. vrijeme trajanja projekta, vrijeme koje je potrebno da se obavi simulacija sustava koji optimiramo i slično.

3.3.3. Način selekcije

Selekcija u prirodi

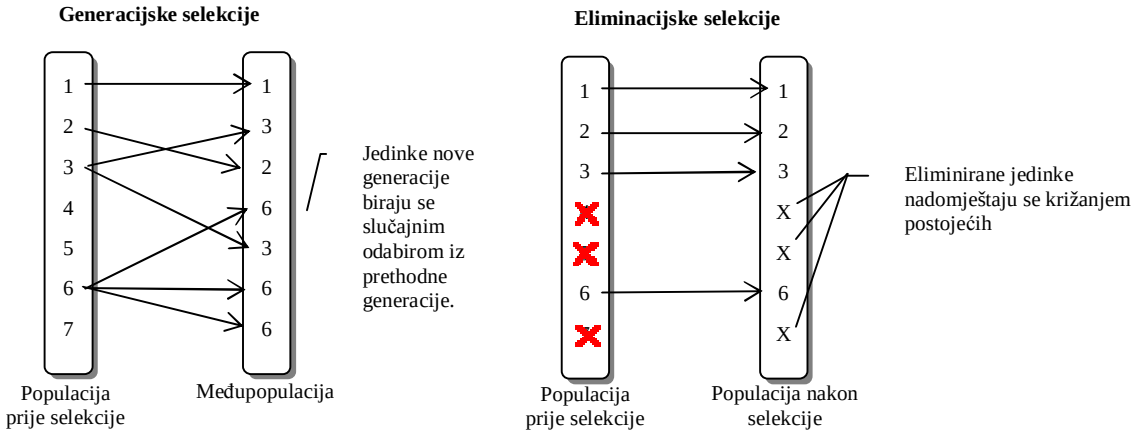
Selekcija se u prirodi obavlja evaluacijom fenotipa jedinke. Fenotip je skup značajki koje su nastale na temelju genotipa neke jedinke i kao posljedica interakcije jedinke sa činiocima iz okoline. Prirodno okruženje odabire one jedinke koje imaju najkvalitetniji genetski materijal. Takve, najbolje jedinke, najčešće sudjeluju u reprodukciji tako da se najveći dio toga dobrog genetskog materijala prenosi na njihove potomke. S vremenom prosječna kvaliteta genetskog materijala postaje sve bolja i bolja. U prirodi taj proces ipak nije strogo definiran i može se dogoditi da neka vrsta nakon dugog vremena poboljšanja nastupi period u kojem je prosječna kvaliteta genetskog materijala sve lošija. Prirodna selekcija je mehanizam koji se stalno mijenja. U jednoj „iteraciji“ prirodne selekcije moguće je da će jedinke sa određenim svojstvima biti najbolje, a u nekoj drugoj „iteraciji“ evolucija će cijeniti neka posve druga svojstva. U prirodnoj selekciji postoji i doza slučajnosti jer nije moguće sa sigurnošću utvrditi na koji način se odabiru pojedine jedinke kao najbolje.

Selekcija u računalu

Selekcija je proces kojim se osigurava prenošenje boljeg genetskog materijala iz generacije u generaciju. Postupci selekcije međusobno se razlikuju po načinu odabira jedinki koje će se prenijeti u sljedeću generaciju.

Prema načinu prenošenja genetskog materijala selekcije se dijele na (slika 3.3):

- Generacijske selekcije
  Proces odabire najbolje jedinke i od njih kreira novu generaciju.
- Eliminacijske selekcije
  Proces selekcije eliminira najgore jedinke iz te generacije.



Slika 3.3. – Podjela selekcija prema načinu prenošenja genetskog materijala

Generacijske selekcije djeluju tako da iz populacije odabire određen broj najboljih jedinki i od njih se stvara nova međupopulacija. Broj jedinki koji preživi selekciju je manji od veličine popu-

lacije pa se jedinke koje nedostaju nadomještaju kopiranjem već selektiranih jedinki. Nedostatak ovoga načina selekcije je istovremeno postojanje populacije i međupopulacije u istom spremniku. Drugi nedostatak je proces višestrukog kopiranja istih jedinki, što rezultira nekvalitetnim (vrlo sličnim) genetskim materijalom.

Kod eliminacijskih selekcija nema stroge granice između jedne i druge generacije. Tijekom procesa eliminacije eliminiraju se slučajno odabrane jedinke (lošije jedinke imaju veću vjerojatnost eliminacije). Eliminirane jedinke nadomještaju se križanjem postojećih.

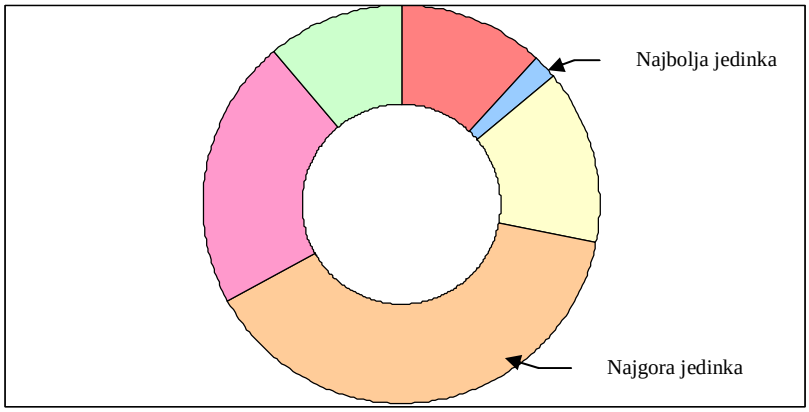
Selekcije možemo dijeliti i prema načinu odabira pojedinih jedinki [Gol02]. Selekcije u tom slučaju dijelimo na:

- Proporcionalne selekcije
  - Jednostavna proporcionalna
  - Stohastička univerzalna
- Rangirajuće selekcije
  - Sortirajuće selekcije
    - Selekcije najboljih
      - (μ,λ) selekcija
      - $(\mu+\lambda)$ selekcija
      - Krnja selekcija
    - Linearno sortirajuća selekcija
  - Turnirske selekcije
    - K-turnirska selekcija
    - Jednostavna turnirska selekcija

Prilikom izrade ovoga seminara korištene su Jednostavna proporcionalna i K-Turnirska selekcija.

Jednostavna proporcionalna selekcija

Jednostavnu proporcionalnu selekciju najslikovitije možemo prikazati kao kotač ruleta sa različitim veličinama za svaki djelić kotača. Za svaku jedinku naše populacije definirati ćemo dobrotu, a obrnuto proporcionalnu vrijednost ćemo unijeti na kotač za tu jedinku. Tako će najbolja jedinka imati najmanji odsječak kotača (postoji najmanja šansa da će upravo ona biti eliminirana), a najlošija jedinka najveći odsječak kotača (Slika 3.4.).

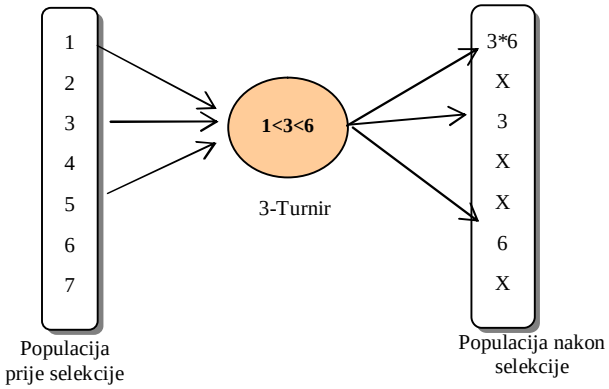


Slika 3.4. – Primjer jedinki raspoređenih po kotaču ruleta

K-turnirska selekcija

U K-turnirskim selekcijama odabire se k članova populacije koji sudjeluju u turniru. Tijekom turnira uspoređuju se dobrote svih k jedinki.

Primjer 3-turnirske selekcije prikazan je na slici 3.5. Slučajnim odabirom odabrane su jedinke 1, 3 i 6 za sudjelovanje u turniru. Jedinka 1 ima najmanju dobrotu i ona se eliminira, a jedinke 3 i 6 se križaju i produciraju novu jedinku „3*6“ koja se dodaje u populaciju.

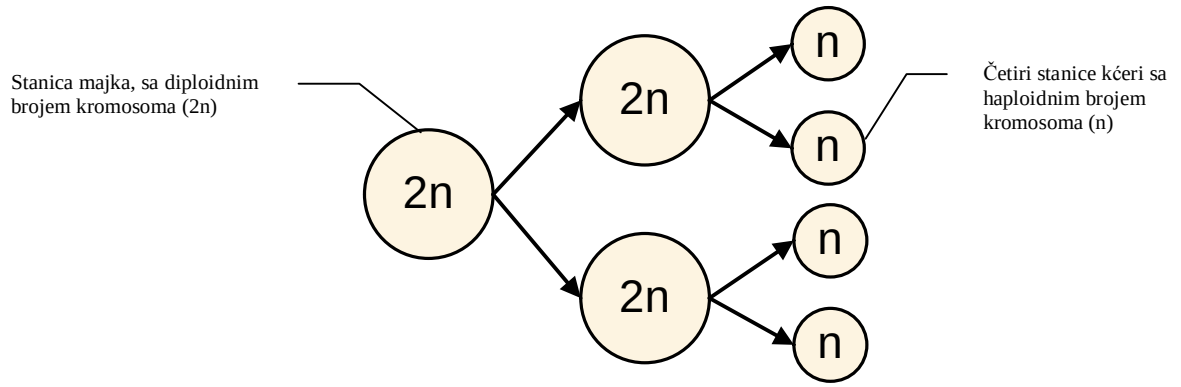


Slika 3.5. – Shema 3-Turnirske selekcije

3.3.4. Križanje

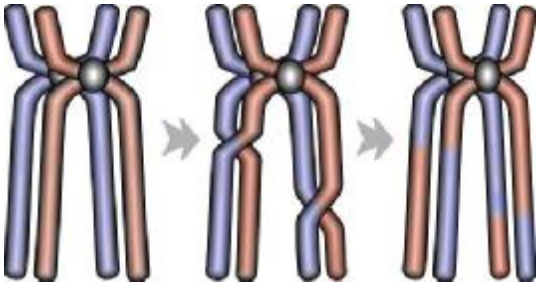
Križanje u prirodi

Križanje u prirodi je vrlo složen proces u kojem najveću ulogu imaju kromosomi. Broj kromosoma jedne jedinke je paran, dakle 2n (čovjek ima 46 kromosoma). Tijekom procesa mejoze (obavlja se jedino u spolnim stanicama), nizom složenih procesa, nastaju četiri nove stanice sa haploidnim (polovičnim) brojem kromosoma (Slika 3.6.).



Slika 3.6. – Pojednostavljena shema procesa mejoze

Tijekom procesa mejoze osim stvaranja novih spolnih stanica dolazi i do izmjene genetskog materijala. To je proces u kojem dva kromosoma formirana tijekom rane faze mejoze izmjenjuju dijelove kromosomskih niti, kao što je to prikazano na slici 3.7.

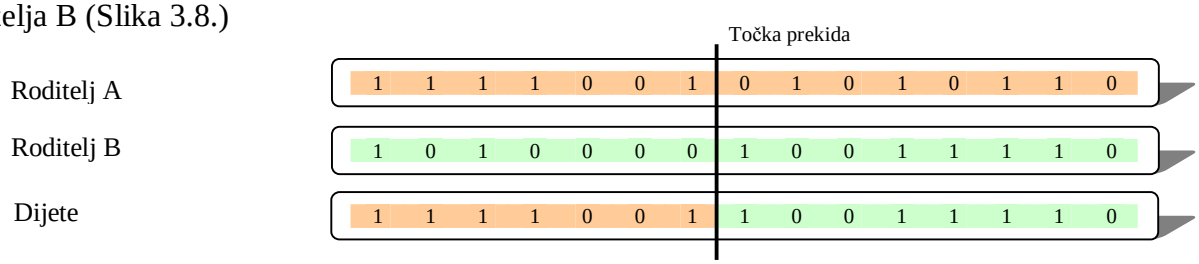


Slika 3.7. – Shema procesa križanja prilikom kojeg dolazi do zamjene dijelova kromosomskih niti

Križanje u genetskom algoritmu

Križanje je proces u kojem se od dva roditelja, križanjem njihovih gena, dobiju jedna ili dvije jedinke koje predstavljaju njihovo potomstvo. Izvedba križanja u računalu uvelike ovisi o načinu kako su kromosomi prikazani. U slučaju kada je kromosom prikazan kao vektor bitova može se primjenjivati mnogo načina križanja. Neki od najčešćih načina su:

1. Križanje s jednom točkom prekida
   Slučajnim odabirom određuje se točka prekida i vektori obaju roditelja se prekidaju na tom mjestu. Do točke prekida dijete nasljeđuje svojstva roditelja A, a od točke prekida roditelja B (Slika 3.8.)



Slika 3.8. – Križanje s jednom točkom prekida

2. Križanje sa dvije točke prekida
   Ovo je križanje slično križanju sa jednom točkom prekida, sa jedinom razlikom što se nakon svake točke genetski materijal naizmjence preuzima od drugog roditelja.

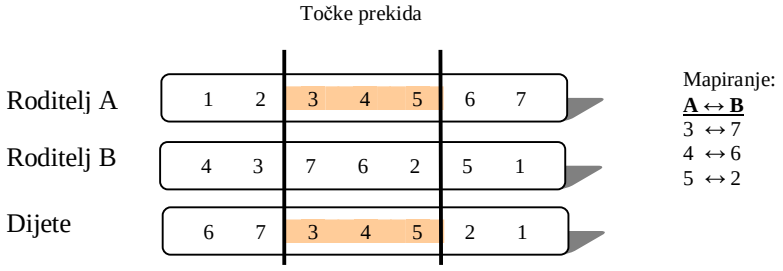
3. Uniformno križanje
   Provodi se kao logička operacija nad bitovima sa logičkim operatorima I, ILI i XOR. Izraz za uniformno križanje:

   $$DIJETE = AB + R\,(A \oplus B) \qquad (2.X)$$

   Pri čemu su A i B roditelji, a R je slučajno generirani vektor jednake duljine.

   Korištenje uniformnog križanja osigurava prijenos onih svojstava koja su kod oba roditelja jednaka.

Ako je naš kromosom kodiran kao niz cijelih brojeva (bez ponavljanja), onda koristimo neke druge načine križanja. U [Mic96] predložen je PMX kao metoda križanja kromosoma koji su prikazani kao nizovi cijelih brojeva. Metoda funkcionira kao križanje sa dvije točke prekida. Između prekidnih točaka uzima se kromosom roditelja A, a ostatak se nadomješta kromosomom roditelja B. U slučaju da se gen koji se kopira već postoji u djetetu, koristi se mapiranje kao na slici 3.9.



Slika 3.9. – Prikaz PMX metode križanja za kromosome koji su predstavljeni kao nizovi cijelih brojeva

3.3.5. Mutacije

Mutacije u prirodi

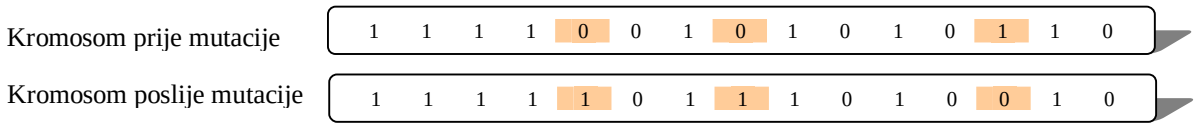
Pod pojmom mutacija u prirodi se podrazumijeva bilo koja promjena genetskog materijala, a najčešće nastaje kao greška u kopiranju prilikom procesa mitoze (dioba stanica) i mejoze (dioba stanica koja rezultira stanicama za haploidnim brojem kromosoma). Najveći broj mutacija koje nastanu se pokazuju kao štetne za organizam. Međutim, da bi proces evolucije bio potpun, potrebne su stalne promjene. Dobre mutacije ugraditi će se u organizam i nastaviti prenositi na nove jedinke u sljedećim generacijama. Iznimno loše mutacije mogu dovesti i do smrti organizma (primjerice kod pojave tumora).

Mutacije u genetskom algoritmu

Mutacije uvelike pomažu izbjegavanju lokalnih optimuma funkcije cilja koju optimiramo. Primjenom mutacija postiže se raznolikost genetskog materijala i omogućava pretraživanje novih – potencijalno najboljih rješenja. Mutacije također obnavljaju genetski materijal. Kad nam sve jedinke na jednom dijelu kromosomu imaju istu vrijednost, ta se vrijednost tijekom križanja nikad neće mijenjati. Tijekom procesa mutacije postoji mogućnost da se upravo taj dio kromosoma promjeni.

Vjerojatnost mutacije određuje se kao jedan od parametara genetskog algoritma. Vjerojatnost mutacije jednog bita obično se odabire u intervalu 0.001 - 0.01. Primjerice, ako je $p_m = 0.005$, to znači da mutira 5 bitova na 1000.

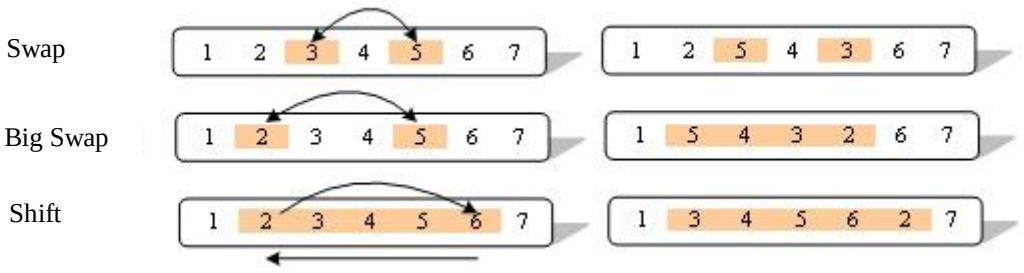
U slučaju kad je kromosom kodiran binarno, primjer jednostavne mutacije prikazan je na slici 3.10.



Slika 3.10. – Prikaz mutacija kod binarno kodiranog kromsoma. (Bitovi kromosoma koji su mutirali označeni su narančastom bojom)

Osim jednostavne mutacije, mogući su i drugi načini mutiranja binarnog kromosoma, kao što su: miješajuća mutacija, potpuna miješajuća mutacija i invertirajuća mutacija.

U slučaju kad se koriste neke druge reprezentacije kromosoma, mutacije su nešto drugaičije. Ako kromosom kodiramo kao niz cijelih brojeva onda su neke od mogućih mutacija: Swap, Big Swap i Shift. Ove mutacije prikazane su na slici 3.11.



Slika 3.11. - Operatori mutacije za kromosome koji su kodirani kao nizovi cijelih brojeva

3.4. Jednostavni genetski algoritam

U velikom broju slučajeva većinu problema moguće je riješiti koristeći samo jednostavni genetski algoritam. Jednostavni genetski algoritam sastoji se od:

1. kromosoma kodiranog binarno,
2. jednostavne selekcije,
3. jednostavnog križanja sa jednom točkom prekida i
4. jednostavne mutacije

3.5. Evaluacija upotrebljivosti genetskog algoritma

Genetske algoritme koji su realizirani za rješavanje mrežnih problema, a opisani su u sljedećim poglavljima, potrebno je evaluirati kako bi se utvrdila njihova upotrebljivost za rješavanje zadanih problema. Evaluacija se može provoditi na dva načina:

1. Usporedbom pronađenih optimalnih rješenja sa rješenjima koja su dobiveni linearnim programiranjem
2. Usporedba efikasnosti pronalaženja rješenja sa efikasnošću Monte Carlo simulacije

Prvi je postupak vrlo jednostavan. Ako genetski algoritam sa podešenim parametrima pronalazi isto optimalno rješenje koje je pronašao i linearni program tada možemo zaključiti da realizirani genetski algoritam daje dobra rješenja za problem koji smo mu zadali.

Monte Carlo simulacija je metoda slučajnog pretraživanja prostora rješenja. Algoritmi Monte Carlo koriste se sa simulaciju različitih fizičkih i matematičkih sistema. Rješenja pronalaze slučajnim generiranjem elementa iz skupa mogućih rješenja. Ako realizirani genetski algoritam pronalazi rješenje brže (u manjem broju iteracija i/ili u kraćem trajanju), tada je realizirani genetski algoritam dobar jer daje rješenja brže od Monte Carlo metode.

4. Primjena genetskih algoritama u optimiranju mrežnih planova

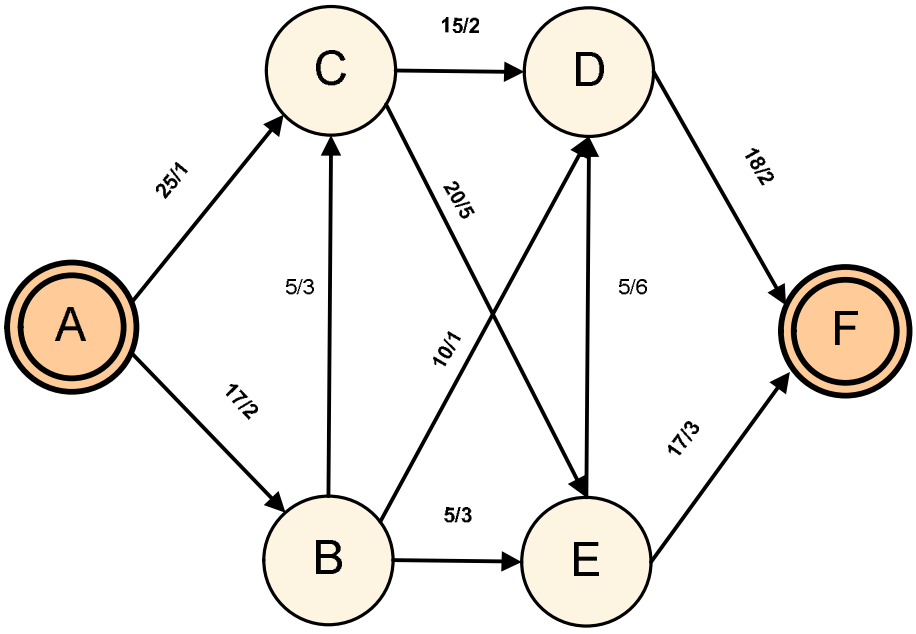
4.1. Jednostavni transportni problem

Neka je zadan sljedeći problem, definiran u [Kal96]: U nekoj prometnoj mreži moguće su sljedeće veze, kojima se definirani kapaciteti i jedinični troškovi. Potrebno je transportirati 30 jedinica nekog proizvoda od čvora A do čvora F uz najmanji trošak. Stvarni podatci prikazani su u tablici 4.1. Dijagram mrežnog plana prikazan je na slici 4.1.

Veza	Kapacitet	Jedinični trošak
A—B	17	2
A—C	25	1
B—C	5	3
B—D	10	1
B—E	5	3

Veza	Kapacitet	Jedinični trošak
C—D	15	2
C—E	20	5
E—D	5	6
D—F	18	2
E—F	17	3

Tablica 4.1. – Jednostavni transportni problem

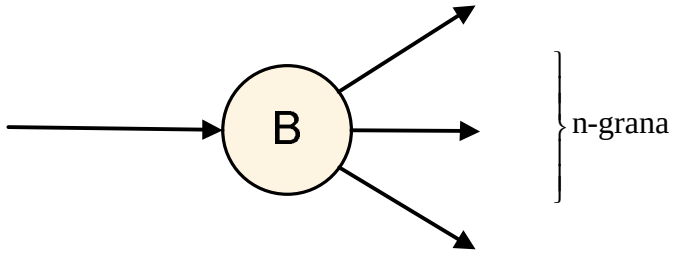


Slika 4.1. Dijagram zadane prometne mreže u obliku mrežnog plana

4.1.1. Pojednostavljenje mrežnog plana

Da bi se ovaj problem mogao uspješno rješavati pomoću genetskih algoritama, potrebno je definirati neka pojednostavljenja samog mrežnog plana.

Pretpostavimo da iz nekog čvora B slijedi n grana koje vode prema drugim čvorovima, kao što je to prikazano na slici 4.2.

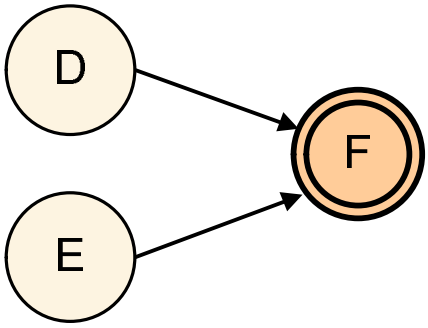


Slika 4.2. - Prikaz čvora sa n izlaznih grana

Da bi genetski algoritam uspješno mogao optimirati ovakav problem, potrebno je smanjiti dimenziju problema. GA će u ovom slučaju optimirati samo količinu prometa na n-1 grani, a količina prometa na n-toj grani računat će se prema sljedećem izrazu:

$$promet_n = ulazni\_promet\_cvoraA - \sum_{i=1}^{n-1} promet_i \tag{4.1}$$

Optimiranje ulaznog prometa za posljednji čvor (Slika 4.3) se ne provodi. Promet posljednje čvora izračunava se po izrazu (4.2).



$$prometF = izlazni\_prometD + izlazni\_prometE \tag{4.2}$$

Slika 4.3. – Krajnji dio mreže za zadani problem

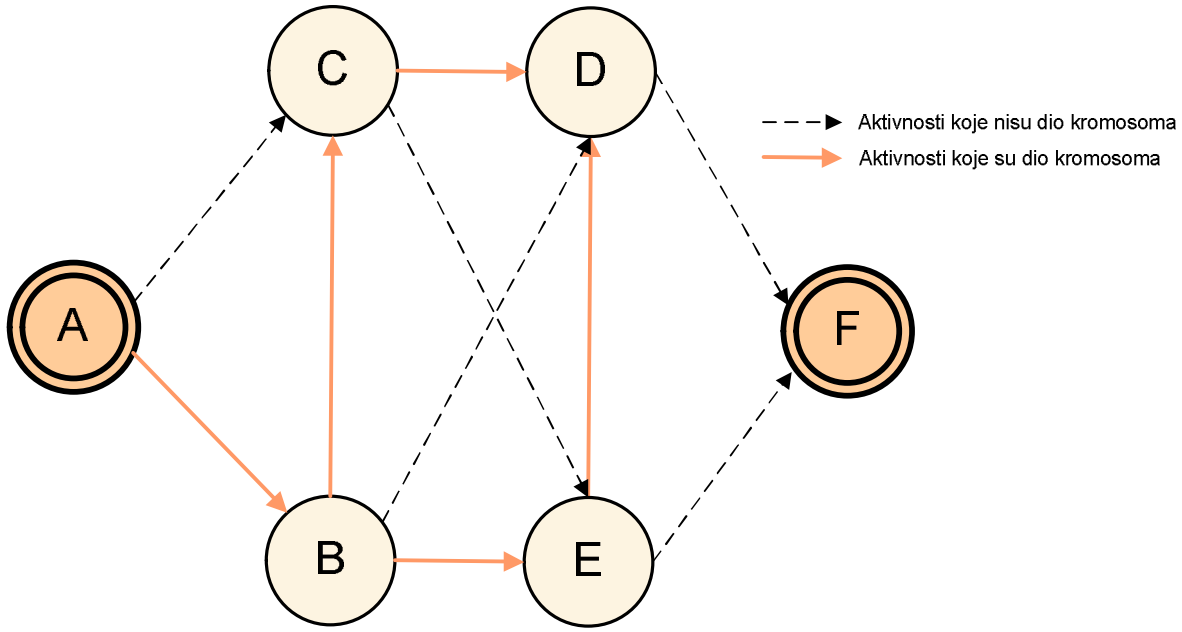
4.1.2. Izvedba genetskog algoritma

U ovom poglavlju je opisan izrađeni genetski algoritam koji je namijenjen rješavanju ovoga problema.

Prikaz kromosoma

Za rješavanje ovoga problema korištena je binarna reprezentacija kromosoma. Kromosom je sastavljen od 5 gena, a svaki gen predstavlja aktivnost u jednog grani. Pritom je kao dio kromosoma korišteno samo n-1 izlaznih grana iz nekog čvora kako bi se smanjio prostor stanja koje algoritam mora pretražiti. Izračun za preostale grane napravljen je prema izrazima (4.1) i (4.2).

Svaki gen kodiran je sa po 5 bita. Time je postignuta dovoljna preciznost od 32 cijela broja za svaki gen. Na slici 4.4. prikazana je shema mreže sa prikazanim aktivnostima koje se kodiraju u kromosom i aktivnostima koje se ne kodiraju u kromosom.



Slika 4.4. – Prikaz zadanog mrežnog plana sa označenim aktivnostima koje se kodiraju u kromosome

Funkcija cilja

Funkcija cilja za ovaj genetskih algoritam dana je u samoj formulaciji problema. Potrebno je naći onaj prometni raspored čiji su troškovi najmanji.

Kazna jedinke

Kazna jedinke je trošak transporta svih 30 jedinica proizvoda od početne do krajnje točke.

Prilikom djelovanja operatora križanja i mutacije postoji vjerojatnost da će novo-generirane jedinke biti nemoguće. Neki od problema nemogućih rješenja su:

- Prijenos većeg broje jedinica kroz jednu granu nego što je to dopušteno njenim kapacitetom
- Ulazni promet jednog čvora nije jednak izlaznom prometu
- Broj jedinica koji je izašao iz početnog čvora ne odgovara ukupnom broju jedinica koje je potrebno prevesti

- Broj jedinica koji je došao do krajnjeg čvora ne odgovara ukupnom broju jedinica koje je potrebno prevesti

U slučajevima kad se prilikom evaluacije detektira rješenje koje ne zadovoljava sva ograničenja, tada se takvo rješenje pokuša popraviti. U slučaju kada to nije moguće, kazna jedinke se povećava kako jedinka ne bi postala najbolje rješenje.

Jedinka se kažnjava po sljedećem izrazu:

$$kazna = kazna * 2^{broj\_ogranicenja\_koja\_nisu\_zadovoljena} \tag{4.3}$$

Dodatno kažnjavanje jedinke osigurava da će takva jedinka biti eliminirana u nekoj od sljedećih iteracija.

Selekcija
U ovom genetskom algoritmu selekcija se vrši 3-turnirskim odabirom.

Križanje
Križanje između dva roditelja je uniformno. Roditelji produciraju samo jedno dijete.

Mutacije
Mutacija je jednostavna, s obzirom na to da se koristi binarnim kromosomski prikaz, pa tijekom mutiranja dolazi do negacije vrijednosti slučajno odabranih bitova kromosoma.

Pseudokod izrađenog genetskog algoritma
Pseudokod genetskog algoritma za rješavanje ovoga problema prikazan je slici 4.5

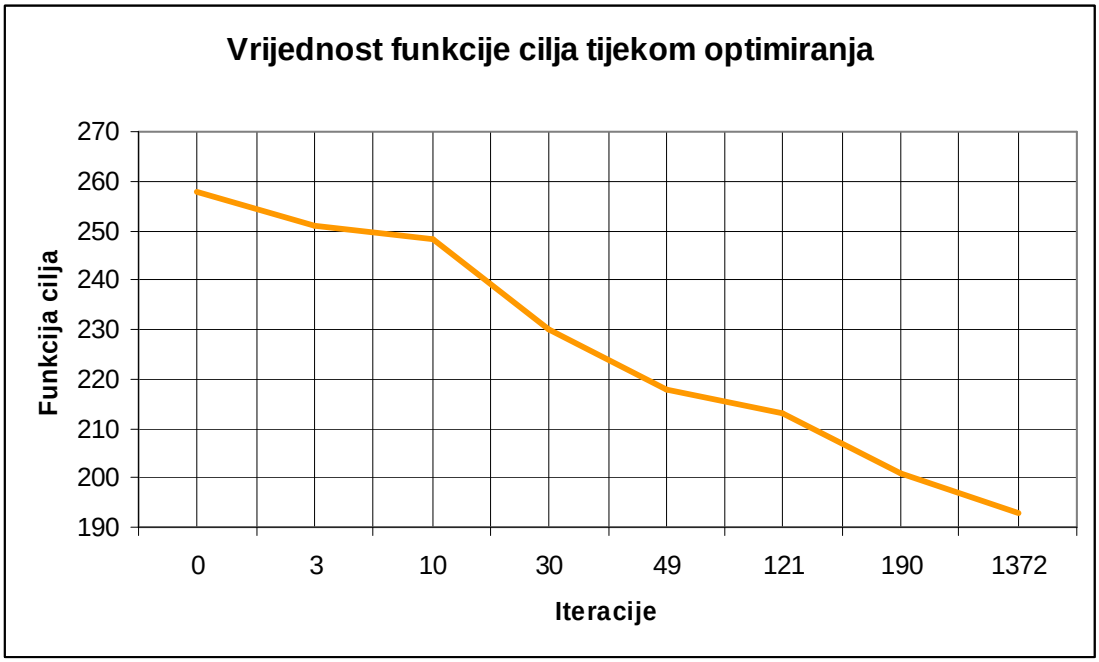
```
Genetski_algoritam(Velicina_Populacije, t, pm, pc, M)
{
   t = 0;
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
      t = t + 1;
      selektiraj tri jedinke P1'(t), P2'(t) i P3'(t) iz P(t-1);
      eliminiraj najlošiju jedinku od selektirane tri
      križaj preostale dvije jedinke;
      mutiraj novu jedinku jedinke Pdijete(t);
      provjeri jedinku Pdijete(t), ako jedinka predstavlja nemoguće rješenje popravi ju
      spremi jedinku Pdijete(t) u populaciju P(t)
   }
   ispiši rješenje;
}
```

Slika 4.5. – Pseudokod ostvarenog genetskog algoritma

4.1.3. Rezultati

Ostvareni genetski algoritam u više od 85% slučajeva (rezultati su prikazani tablici 4.2) pronalazi rješenje koje je istovjetno rezultatima koji su dobiveni linearnim programiranjem, a opisani su u [Dil05].

Na slici 4.5. prikazan je tijek optimiranja tijekom jednog pokretanja genetskog algoritma (Veličina populacije 50, broj iteracija do zaustavljanja 5000).



Slika 4.5. – Vrijednost funkcije cilja tijekom jednog pokretanja genetskog algoritma

Podešavanje parametara

Da bi se utvrdilo za koje parametre genetski algoritam daje najbolje rješenje, provedeno je testiranje sa različitim kombinacijama veličine populacije i uvjeta zaustavljanja. Veličine populacije koje su se koristile su: 25, 50, 100, 200, 500, 1000 i 2000 jedinki. Uvjet zaustavljanja je broj iteracija bez poboljšanja najbolje jedinke (mogući broj iteracija: 100, 200, 500, 1000 i 2000). Genetski je algoritam pokrenut po 50 puta za svaku kombinaciju veličine populacije i broja iteracija.

U tablici 4.2. prikazane su najbolje vrijednosti koje su dobivene prilikom pokretanja za sve kombinacije veličina populacije i broja iteracija za zaustavljanje. Vidljivo je da većina kombinacija daje optimalno rješenje tj. pronalazi minimum koji iznosi 193 što je jednako minimumu koji se dobije linearnim rješavanjem ovoga problema.

Broj iteracija	Veličina populacije						
	25	50	100	200	500	1000	2000
100	201	201	193	197	193	193	197
200	201	193	193	193	193	193	197
500	193	193	193	193	193	193	193
1000	193	193	193	193	193	193	193
2000	193	193	193	193	193	193	193

Tablica 4.2. – Najbolje rješenje koje je dobio genetski algoritam u ovisnosti o veličini populacije i broju iteracija do zaustavljanja

U tablici 4.3. prikazane su prosječne vrijednosti koje su se dobile kao rješenja prilikom svih 50 pokretanja genetskog algoritma. Vidljivo je da genetski algoritam daje dosta stabilna rješenja koja su u najvećem broju slučajeva vrlo bliska minimumu kad je broj iteracija za zaustavljanje veći ili jednak 1000 iteracija. Veličina populacije ne utječe toliko na kvalitetu rješenja.

Broj iteracija	Veličina populacije						
	25	50	100	200	500	1000	2000
100	207,98	202,98	199,86	204,34	200,86	200,92	201,52
200	209	200,6	200,44	201,34	200,68	200,48	200,78
500	200,48	199,16	198,38	200,54	200,36	200,52	200,36
1000	200,36	199,82	197,36	200,2	199,1	198,6	196,52
2000	200,22	193	199,58	198,84	196,22	194,12	196,44

Tablica 4.3. – Prosječne vrijednosti funkcije cilja koje su dobivene za 50 pokretanja genetskog algoritma u ovisnosti o veličini populacije i broju iteracija do zaustavljanja

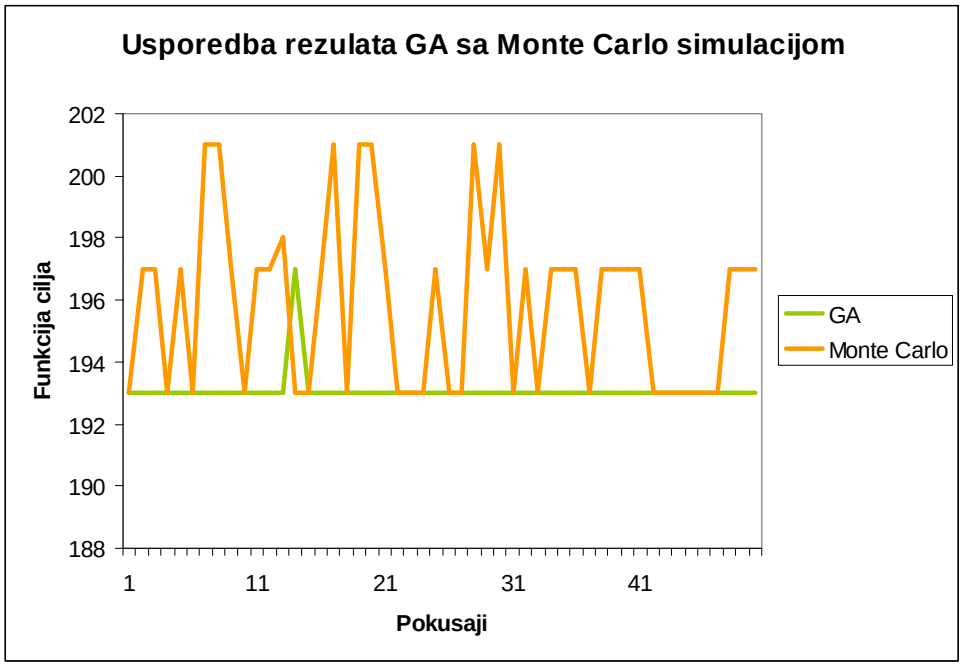
Kako bi ova analiza bila potpuna, u tablici 4.4. prikazana su prosječna vremena izvođenja (u sekundama) u ovisnosti o veličini populacije i broju iteracija. Vidljivo je da genetski algoritam može pronaći rješenje u vrlo kratkom vremenu. Njegovo vrijeme izvođenja ipak je znatno dulje nego vrijeme izvođenja linearnog programa, koji se izvršava gotovo trenutno.

Broj iteracija	Veličina populacije						
	25	50	100	200	500	1000	2000
100	0	0,06	0,1	1,46	5,26	10,8	22,98
200	0	0,26	1,08	3,68	9,26	18,12	35,46
500	0,28	1,1	2,5	7,66	20,48	39,2	30,08
1000	1,04	2,2	4,9	14,06	36,24	15,44	39,72
2000	2,42	6,46	11,48	27,96	16,24	31,8	13,02

Tablica 4.4. – Prosječno vrijeme izvođenja genetskog algoritma u 50 pokretanja. Vrijeme je izraženo u sekundama

Evaluacija upotrebljivosti ostvarenog genetskog algoritma

Opisani rezultati, ostvareni ovim genetskim algoritmom, uspoređeni su sa rezultatima koji su dobiveni Monte Carlo simulacijom za isti problem. Za svako pokretanje genetskog algoritma izračunat je broj jedinki koje su činile sve generacije, a taj isti broj jedinki generiran je i Monte Carlo simulacijom. Rezultati su pokazali da GA daje stabilnija rješenja koja su monotonije raspoređena u blizini globalnog optimuma. Kvaliteta optimuma Monte Carlo simulacije poprilično varira (Slika 4.6), Monte Carlo simulacija dostiže optimum ali samo u malom broju slučajeva, dok GA optimum postiže u 98% pokušaja.



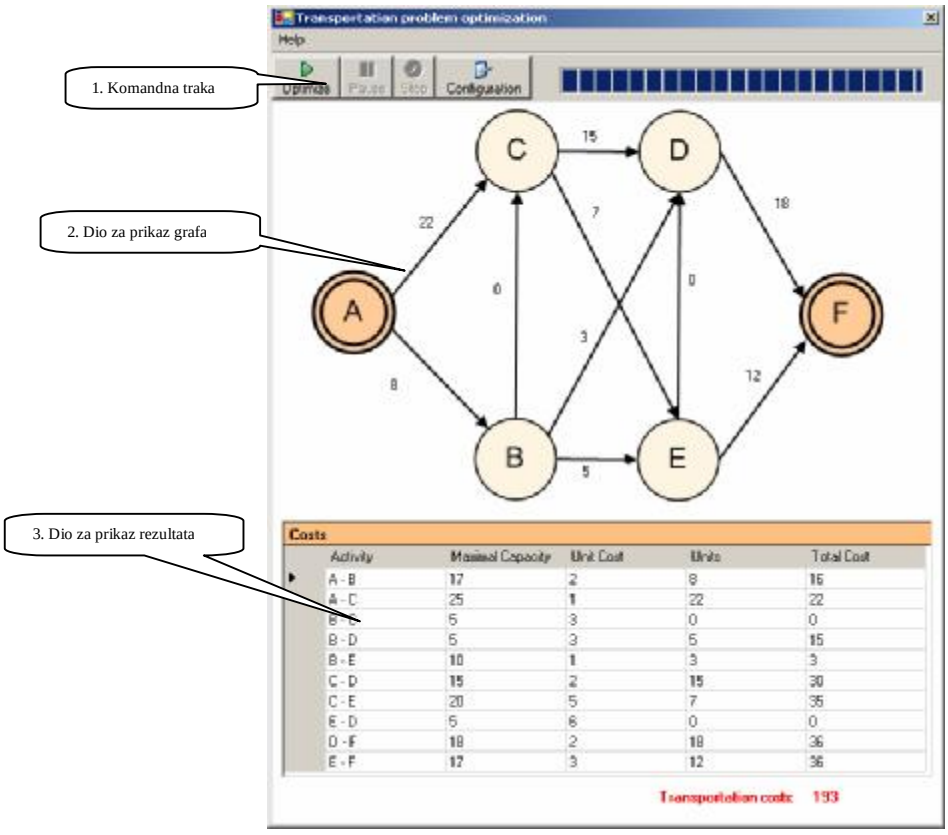
Slika 4.6. – Usporedba rezultata GA i rezultata dobivenih Monte Carlo simulacijom

Opsežnim testiranjima sa različitim parametrima pokazalo se da su rješenja koja su dobivena sa GA prosječno samo 1% - 5% bolja od onih koja su postignuta Monte Carlo simulacijom. Uzrok tome je vrlo mali prostor potencijalnih rješenja jer su sve vrijednosti su mali cijeli brojevi. Kada bi se prostor proširio na npr. realne brojeve GA bi postizao još bolje rezultate u odnosu na Monte Carlo simulaciju.

4.1.4. Aplikacija za grafičko prikazivanje rezultata

Kako bi se olakšala analiza rada genetskog algoritma, izgrađena je aplikacija za vizualno predočavanje rezultata. Aplikacija omogućuje višestruka pokretanja procesa optimiranja zadanog problema. Tijekom izvođenja optimiranja moguće ju je prekinuti kako bi se pregledali rezultati optimiranja. Na ekranu se u trenutku prekida i u trenutku dovršenja simulacije prikazuju rezultati optimiranja.

Osnovni dijelovi aplikacije



Slika 4.7. – Aplikacija za grafički prikaz rezultata optimiranja transportnog problema

Na slici 4.7 prikazani su osnovni dijelovi aplikacije za simulaciju rezultata. Aplikacija se sastoji od tri glavna dijela:

1. Komandne traka
2. Dijela za prikaz grafa
3. Dijela za prikaz rezultata

Komandna traka
Omogućuju pokretanje i zaustavljanje simulacije. Na traci se nalaze četiri naredbe opisane u tablici 4.5..

Naredba	Opis
Start	Započinje proces rješavanja zadanog problema genetskim algoritmom. Generira novu populaciju i pokreće optimiranje. Ako je simulacija bila prethodno zaustavljena, nastavlja sa optimizacijom.
Zaustavi	Privremeno zaustavlja trenutno aktivnu simulaciju. Na grafu i u tablici za prikaz rezultata prikazuje trenutno ostvarene rezultate.
Prekini	Zaustavlja trenutnu simulaciju. Rezultati se prikazuju na grafu i u tablici.
Postavke	Otvara dijalog koji omogućuje upravljanje postavkama programa.

Tablica 4.5. – Naredbe komandne trake

Prikaz grafa
Na grafu se prikazuje trenutni raspored transportiranih jedinica po granama grafa. Graf se osvježava samo u trenutku kad se optimiranje zaustavi ili kad je optimiranje gotovo.

Prikaz rezultata
Rezultati optimiranje prikazani su u tablici koja sadržava sljedeće vrijednosti (Tablica 4.6):

Vrijednost	Opis
Aktivnost	Naziv aktivnosti u obliku Čvor1-Čvor2.
Maksimalni kapacitet	Maksimalni broj jedinica koje je moguće transportirati od Čvora1 do Čvora2
Jedinični trošak transporta	Cijena transporta jedne jedince
Broj prevezenih jedinica	Broj prevezenih jedinica koje je postavio genetski algoritam
Ukupan trošak	Ukupan trošak prijevoza svih jedinica na ovom bridu grafa

Tablica 4.6. – Opis tablice koja prikazuje rezultate optimiranja

4.2. Skraćenje vremena trajanja projekta

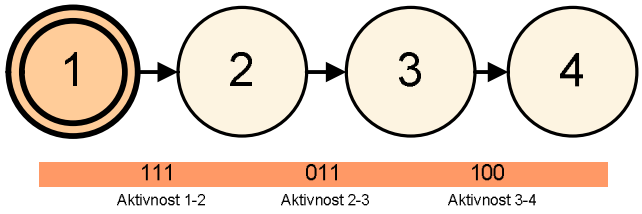
Za projekt koji je opisan u poglavlju 2.1. bit će opisan genetski algoritam koji pronalazi optimalni projektni plan sa minimalnim trajanjem i minimalnim troškovima. Cilj je usporediti može li se genetskim algoritmom postići jednako dobro rješenje kao ono koje je postignuto linearnim programom.

4.2.1. Izvedba genetskog algoritma

Ovaj genetski algoritam vrlo je jednostavan jer je i zadani mrežni problem vrlo jednostavan.

Prikaz kromosoma

Kromosom je za ovaj genetski algoritam kodiran binarno. Pritom se kromosom sastoji od više gena. Svaki gen predstavlja jednu granu (tj. aktivnost) projektnog plana (Slika 4.8.). Geni su kodirani sa 3 bita. Unutar jednog bita pohranjena je informacija za koliko je vremenskih jedinica potrebno smanjiti aktivnost te grane (normalna trajanja pojedine aktivnosti su unaprijed zadana). Njihov ukupan broj je 14. Ukupna duljina kromosoma je 42 bita. Prilikom generiranja novih jedinki vodi se računa o tome da nove jedinke zadovoljavaju dana ograničenja.



Slika 4.8. – Nekoliko aktivnosti iz mrežnog dijagrama i njihovi geni kodirani binarno.

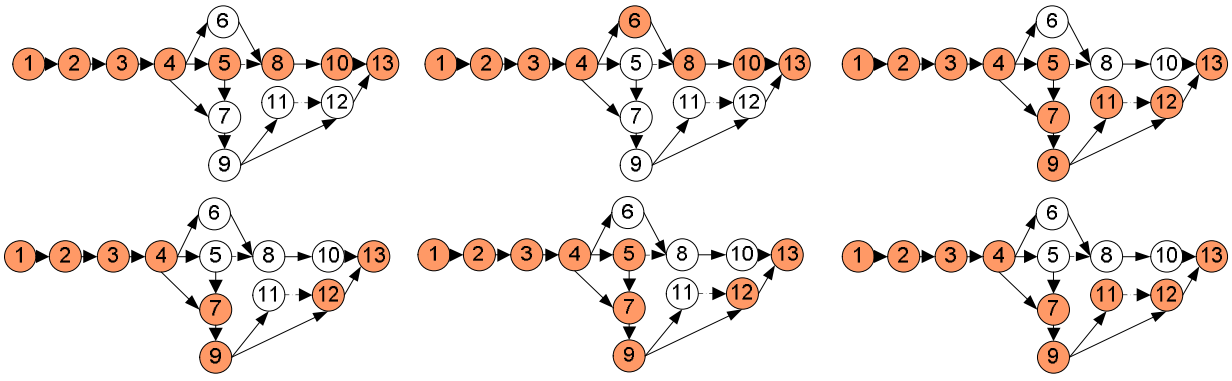
Funkcija cilja

Kao što je već opisano u definiciji problema, potrebno je pronaći onaj skraćeni projektni plan čiji su troškovi završetka projekta najmanji.

Kazna jedinke

Kazna jedinke izračunava se u skladu sa funkcijom cilja. Kazna jedinke je cijena završetka projekta. Pri izračunu te vrijednosti potrebno je pronaći najdulji put kroz mrežni plan tzv. kritični put tj. onaj put čije je vrijeme trajanja najdulje u odnosu na sve ostale putove.

Vrijeme trajanja projekta na kritičnom putu je jednako ukupnom vremenu trajanja projekta tj. kao kazna te jedinke. Svi kritični putovi za zadani mrežni plan prikazani su na slici 4.9.



Slika 4.9. – Mogući kritični putovi zadanog mrežnog dijagrama

Selekcija

Odabrana selekcija za ovaj problem je 3 turnirska selekcija. U jednom turniru sudjeluju nasumično odabrane tri jedinke. Jedinka sa najskupljim izvođenjem projekta se eliminira iz populacije, a preostale dvije križanjem produciraju novu jedinku.

Križanje

Križanje dviju jedinki realizirano je kako uniformno križanje. Dva roditelja prilikom križanja produciraju samo jedno dijete.

Mutacije

Realizirana je jednostavna mutacija. Mutacija se provodi nakon križanja dvaju roditelja, sa zadanom vjerojatnošću.

Pseudokod izrađenog genetskog algoritma

Pseudokod genetskog algoritma za rješavanje ovoga problema prikazan je slici 4.10

```
Genetski_algoritam(Velicina_Populacije, t, pm, pc, M)
{
   t = 0;
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
      t = t + 1;
      selektiraj tri jedinke P1'(t), P2'(t) i P3'(t) iz P(t-1);
      eliminiraj najlošiju jedinku od selektirane tri
      križaj preostale dvije jedinke;
      mutiraj novu jedinku jedinke Pdijete(t);
      provjeri jedinku Pdijete(t), ako jedinka predstavlja nemoguće rješenje popravi ju
      spremi jedinku Pdijete(t) u populaciju P(t)
   }
   ispiši rješenje;
}
```

Slika 4.10. – Pseudokod ostvarenog genetskog algoritma

4.2.2. Rezultati

S obzirom da se radi o vrlo jednostavnom problemu genetski algoritam uspješno pronalazi rješenja koja su usporediva sa rezultatima koji dobiveni linearnim programiranjem, a opisani su u poglavlju 2.1.

Testiranje genetskog algoritma obavljeno je sa nekoliko veličina populacije (10, 50, 100, 200 i 500), a genetski je algoritam pokrenut po 50 puta. Također se varirao i uvjet zaustavljanja (broj iteracija bez promjene najbolje jedinke).

U tablici 4.7. prikazani su najbolji od 50 rezultata testiranja za svaku kombinaciju veličine populacije i broja iteracija. Pri većini simulacija genetski algoritam zaglavi u lokalnom optimumu 8499,9, a tek nekoliko simulacija vraća stvarnu minimalnu vrijednost 8466,6. S obzirom da se

radi o vrlo malenoj pogrešci, jer su koraci u sustavu diskretni, čak je i rješenje 8499,9 zadovoljavajuće. Povećanjem broja jedinki u populaciji i broja iteracija može se postići globalni optimum, ali proces njegova pronalaženja traje predugo.

Broj iteracija	Veličina populacije				
	10	50	100	200	500
100	8499,9	8499,9	8499,9	8499,9	8466,6
500	8499,9	8499,9	8499,9	8499,9	8499,9
1000	8499,9	8499,9	8499,9	8499,9	8499,9
2000	8499,9	8499,9	8499,9	8499,9	8466,6
5000	8499,9	8499,9	8499,9	8466,6	8499,9

Tablica 4.7. – Najbolja vrijednost rješenja koju je pronašao genetski algoritam za određenu veličinu populacije i broj iteracija do zaustavljanja

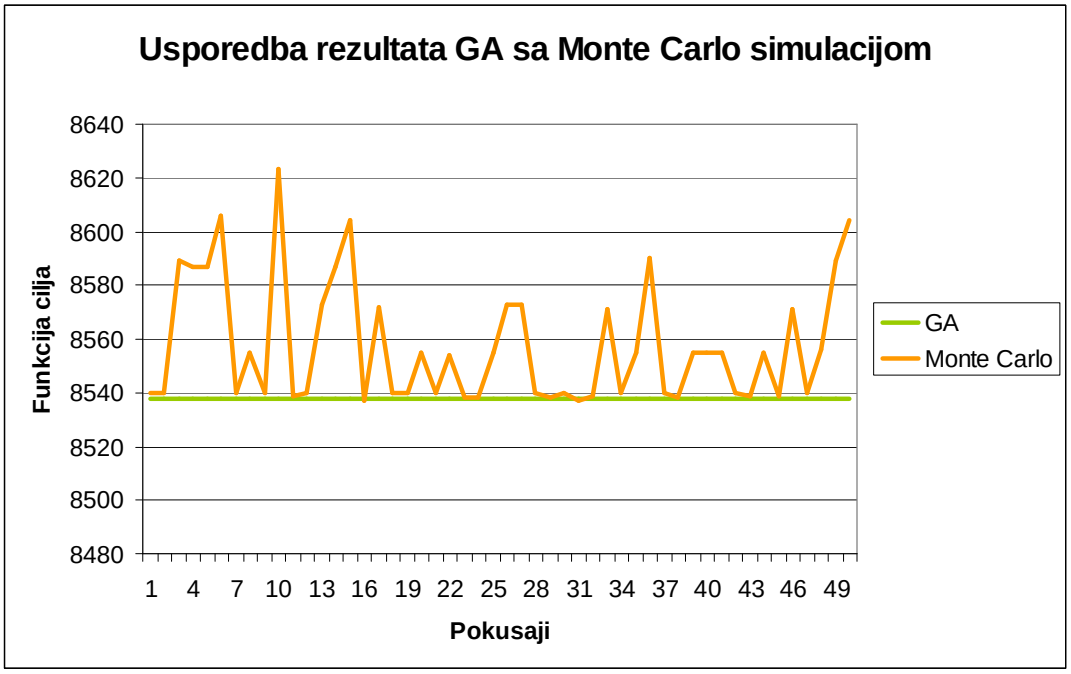
U tablici 4.8. prikazane su prosječne vrijednosti koje dobiva genetski algoritam u svih 50 pokretanja. Vidljivo je da rješenja dosta variraju kada su populacije male, a da već kod populacija 100 i 200 jedinki te pri velikom broju iteracija imamo dosta stabilno rješenje u svakom pokretanju. Populacija 500 je stabilna već i za mali broj iteracija do zaustavljanja.

Broj iteracija	Veličina populacije				
	10	50	100	200	500
100	8601,9	8574,9	8557,9	8531,2	8511,2
500	8585,9	8561,9	8549,9	8511,9	8499,9
1000	8587,9	8549,9	8519,9	8509,9	8499,9
2000	8581,9	8545,9	8505,9	8501,9	8499,2
5000	8569,9	8507,9	8501,9	8499,2	8499,9

Tablica 4.8. – Prosječna vrijednost rješenja za određenu veličinu populacije i broj iteracija do zaustavljanja. Za svaku kombinaciju veličine populacije i broj iteracija genetski je algoritam pokrenut 50 puta

Evaluacija upotrebljivosti ostvarenog genetskog algoritma

Kao što je vidljivo iz tablice 4.7 realizirani GA vrlo lako pronalazi rješenja zadanog problema. Već slučajnim odabirom moguće je da GA pronađe dobro rješenje. Usporedbom sa Monte Carlo simulacijom pokazalo se da se i slučajnim pretraživanjem prostora stanja nalaze relativno zadovoljavajuća rješenja. Prosječna rješenja dobivena sa Monte Carlo simulacijom, sa različitim brojem jedinki, bila su 0,1‰-10‰ lošija od onih koje je dobio GA. Skup rješenja kod Monte Carlo simulacije je više raspršen oko optimuma dok GA daje stalno isto, vrlo dobro i stabilno rješenje. (Slika 4.11)

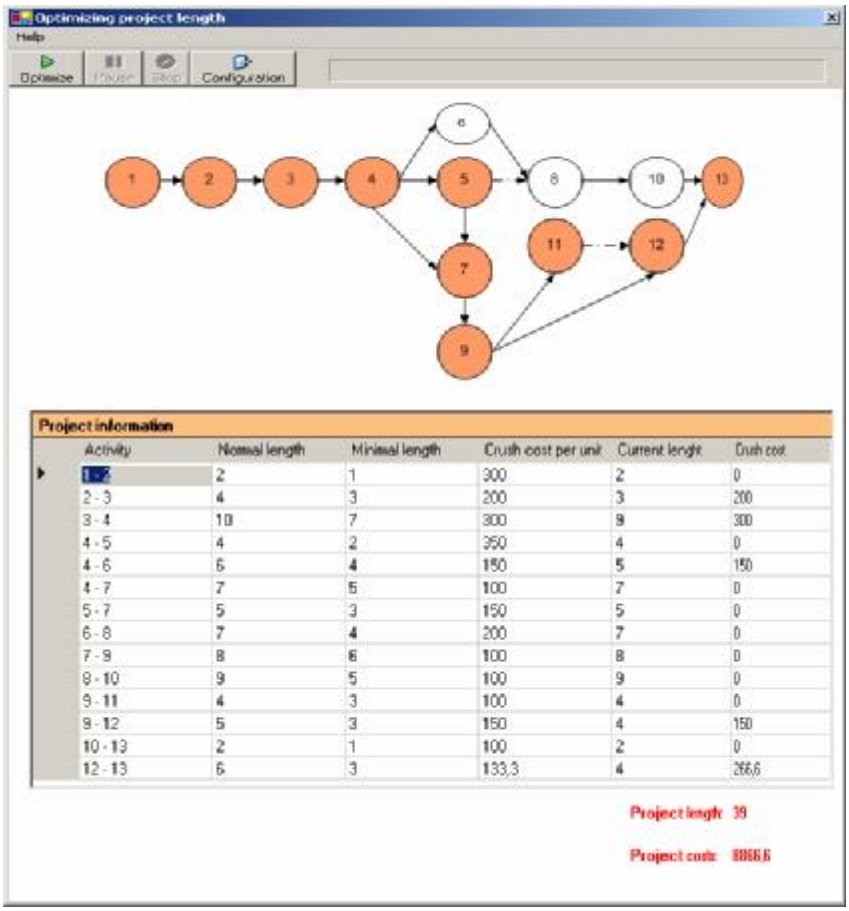


Slika 4.11. – Usporedba rezultata GA i rezultata dobivenih Monte Carlo simulacijom

Ova usporedba GA i Monte Carlo simulacije pokazuje da je upotreba GA na zadanom problemu nije posve opravdana, prostor mogućih rješenja premalen je da bismo na njemu efikasno izvršavali GA. Bolja rješenja postigla bi se kada bi prostor rješenja uključivo više rješenja.

4.2.3. Aplikacija za grafički prikaz rezultata

Radi olakšane analize rada genetskog izrađena je aplikacija koja vizualno prikazuje rezultate optimiranja. Aplikacija je prikazana na slici 4.12.



Slika 4.12. – Aplikacija za grafički prikaz rezultata skraćenja projekta

Aplikacija omogućuje prikaz detalja o svim aktivnostima projektnog plana, što olakšava analizu rezultata. Optimiranje se može pokretati više puta, a tijekom procesa optimiranja program je moguće zaustaviti kako bi se vidio trenutni rezultat. Detalji o upotrebi aplikacije su opisani u poglavlju 4.1.4.

5. Zaključak

Genetski algoritmi, koji su izrađeni za ovdje opisane probleme u mrežnom planiranju s velikom uspješnošću pronalaze rješenja koja su posve jednaka onim optimumima koji se dobivaju linearnim programiranjem.

Svi mrežni problemi koji su opisani u ovom radu pretpostavljali su linearnu zavisnost kako bi bilo moguće usporediti rezultate koje dobivaju GA sa rezultatima linearnih programa. Prednost GA nad linearnim programiranjem u mrežnom planiraju je u tome što GA ne postavlja nikakve zahtjeve nad problemom koji rješava (mrežni problem može biti i nelinearan). Osim toga kod složenijih mrežnih problema bilo bi jako teško izgraditi sustav linearnih jednadžbi koji bi opisivali taj problem.

U izrađenim simulacijama GA se pokazao kao vrlo dobra metoda optimiranja upravo na zadanim mrežnim problemima. Rezultati su bili stabilniji i kvalitetniji od onih koji su dobiveni sa Monte Carlo simulacijom. Zbog malog prostora rješenja, ove je probleme bilo moguće riješiti i pretraživanjem prostora stanja. Međutim, kod kompleksnijih problema sa većim prostorom rješenja, GA bi polučio bolje rezultate od Monte Carlo simulacije.

Budući rad

Genetski algoritmi opisani u ovom radu bazirani su na kromosomskom prikazu. Zbog specifičnosti zapisa o trajanju projekta i njihovih zavisnosti, za ovaj genetski algoritam trebalo bi analizirati moguće alternativne zapise kromosoma. Neke od mogućih alternativa mogle bi biti matrice ili liste kao reprezentacije kromosoma.

Genetske algoritme koji su izrađeni za ovaj seminar trebalo bi dodatno podešavati na različitim i većim strukturama mrežnih planova. U tu svrhu trebalo bi napraviti generator koji može generirati takav usmjereni graf koji u sebi nema ciklusa, u kojem postoji put od početne do krajnje točke i koji osigurava dovoljan protok od početne do krajne točke. Uz upotrebu ovakvog generatora bilo bi moguće kreirati dovoljno veliki broj testnih uzoraka pomoću kojih bi bilo moguće dodatno podesiti izrađeni genetski algoritam.

Skraćenje projekta interesantan je način optimiranja projekata. Međutim u stvarnim životnim uvjetima bilo bi vrlo teško pronaći projekt koji odgovara uvjetima koji su opisani u ovom radu. Da bismo bili u mogućnosti optimirati stvarne projekte u stvarnim projektnima okruženjima ovaj bi genetski algoritam trebalo proširiti mnogim svojstvima kao što su resursi na projektu, vremena početka i završetka, rokovi projekta, nelinearne zavisnosti i slično. Tek kad se u igru uključe svi ti ostali faktori i kada se izrađeni GA proširi tim svojostima, on će biti primjenjiv u stvarnim okruženjima.

6. Literatura

[Gol04] Golub M. *Genetski algoritmi – Prvi dio,* 2004. Fakultet elektrotehnike i računarstva. Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave. Dostupno na Internet adresi: http://www.zemris.fer.hr/~golub/ga/ga.html

[Gol02] Golub M. *Genetski algoritmi – Prvi dio,* 2002. Fakultet elektrotehnike i računarstva. Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave. Dostupno na Internet adresi: http://www.zemris.fer.hr/~golub/ga/ga.html

[Jak05] Jakobović D. *Genetski algoritmi – „Predavanja iz kolegija Analiza i projektiranje računalom“*, 2005. Fakultet elektrotehnike i računarstva. Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave. Dostupno na Internet adresi: http://www.zemris.fer.hr/predmeti/apr/

[Kal96] Kalpić D., Mornar V., *Operacijska istraživanja*. 1996. Biblioteka informacijsko društvo.

[Sav00] Savaux M., Dauzere-Peres S., *Building a Genetic Algorithm for a Single Machine Scheduling Problem.* 2000.

[Mic94] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1994.

[Dil05] Dilber, H., *Operacijska istraživanja*, Fakultet elektrotehnike i računarstva. Zavod za primjenjenu matematiku