SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Rješavanje problema trgovačkog putnika genetskim algoritmom

Gregor Pogačar
Voditelj: *Marko Đurašević*

Zagreb, lipanj 2018.

SADRŽAJ

1. Uvod

Podrijetlo problema trgovačkog putnika i nije sasvim poznato. Problem se prvi puta matematički formulirao 1850-ih godina od strane W.R.Hamiltona, Irskog matematičara koji je prvi proučavao odnose među grafovima. Njegova poznata igra '*Icosian Game*' iz 1857. godine se upravo bazira na problemu trgovačkog putnika i pronalasku najkraćeg Hamiltonovog ciklusa. Kasnije je Karl Mengar generalizirao i dobro definirao problem primjenjući iscrpnu pretragu i heuristiku najbližeg susjeda. Inspirirani problemima koji su nastajali razvojem tehnologije 1950-ih godina, znastvenici iz različitih područja su se sve više zanimali za inačice ovog problema. Do značajnijih rješenja se došlo tek sredinom 1970-ih za otprilike 2400 gradova. Kako su se u to vrijeme počele razvijati evolucijske strategije, te je istovremeno jačala računalna moć, optimalne rute su se pojavljivale sve češće za sve veći broj gradova.

U prvom djelu ovog rada je opisan genetički algoritam kao metoda koja oponaša evolucijski proces, te su objašnjeni osnovni operatori, struktura i karakteristike algoritma. U nastavku je definiran problem trgovačkog putnika implementaciji pomoću genetskog algoritma.

2. Genetski Algoritam

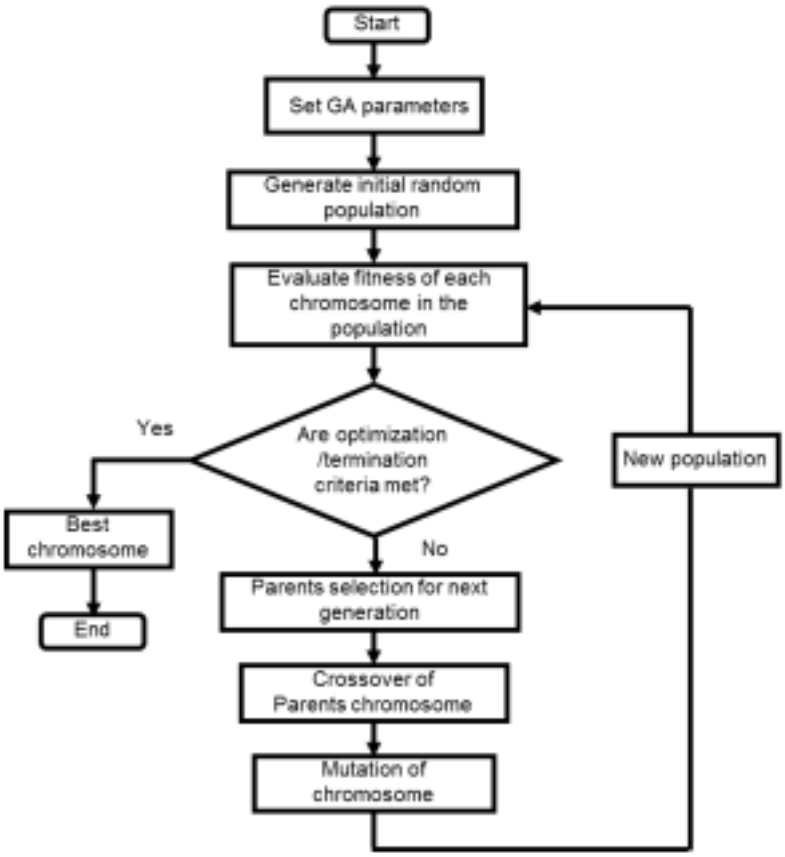
Genetički algoritam (GA) je evolucijski algoritam koji se koristi za rješavanje kompleksnih problema simulirajući prirodni evolucijski proces. GA je tehnika pretraživanja heurističkim pristupom koja se koristi u različitim područjima optimizacije. Algoritam se temelji na konceptu preživljanja najsposobnijeg 'survival of the fittest'. Genetički algoritam je prvi pomnije promatrao John Holland 1970-ih godina kada su se biolozi počeli zanimati za implementaciju genetskih karakteristika u računalne programe. Radi jednostavnosti primjene i savršeno prirodne ideje na kojoj su zasnovani, danas genetički algoritam vrlo uspješno sudjeluje u rješavanju čitavog niza optimizacijskih problema. Način djelovanja ubraja genetski algoritam u metode usmjerenog pretraživanja prostora rješenja u traženju globalnog optimuma. Cilj algoritma je dobiti najprecizniju aproksimaciju rješenja što manjem broju iteracija (generacija) ili zadovoljiti prethodno definirane kriterije. Populacije zajedno predstavljaju sva rješenja jedne generacije. Veći broj jedinki u populaciji može nositi ispravno rješenje, no to ne znači da je to najbolje rješenje. Rezultat može odstupati od najboljeg mogućeg, no ključan aspekt genetskog algoritma je što dolazi do rezultata u konačnom vremenu. Prostor rješenja genetskog algoritma se postepeno smanjuje primjenjivajući genetske operatore poput križanja, mutacije i selekcije. Dobar genetički algoritam ima svaku generaciju sveukupno bolju od prošle pri čemu su samo određene jedinke prenijele povoljne gene. Genetski materijal kojeg jedinka prenosi na svojeg potomka su kromosomi koji su najčešće pohranjeni u binarnom prikazu, kao vektor ili matrica. To je skup svojstava jedne jedinke za koji genetski algoritam kaže da bi se trebao razmjenjivati i slučajno mjenjati. U načelu jedinke koje imaju veću vjerojatnost reprodukcije su one koje predstavljaju naprikladnije rješenje za dani problem.

Tijekom izrade jednostavnog genetskog algoritma potrebno je sadržavati 5 osnovnih komponenti:

1. Kromosomska reprezentacija rješenja problema.
2. Populaciju inicijalnih rješenja.
3. Genetske operatore za evoluciju populacije
4. Funkciju koja predstavlja kvalitetu rješenja problema.
5. Parametre koji specificiraju vjerojatnosti genetskih operatora

Ono što uistinu optimira genetički algoritam je funkcija dobrote. *Fitness* funkcija ili funkcija dobrote je funkcija koja određuje sposobnost svake jedinke. Postupkom evaluacije, jedinke se međusobno odjeljuju i razlikuju prema načinu dodjeljene kvalitete, odnosno sposobnosti. Kriteriji dobrote su ti koji čine cijelu stvar tako prirodnom. Način na koji će pojedine jedinke bit bolje od ostalih je striktno vezan za prirodu problema. Stoga fitness funkcija predstavlja osnovicu za poboljšanje u populaciji.

Na slici 2.1 je prikazano odvijanje pojedinih operacija uobičajenog genetskog algoritma.



Slika 2.1: Shema genetskog algoritma

2.1. Genetski Operatori

Operatori se koriste nad jedinkama neke populacije kako bi se ubrzao proces evolucije. Jedinku koju odabiremo za genetski proces nazivamo roditelj, a onu koja nastaje kao produkt križanja dva roditelja zovemo dijete.

2.1.1. Selekcija

Proces selekcije ima važnu ulogu u pretraživanju boljih jedinki i u održavanju raznolikosti genotipa populacije. Pomoću selekcije, dobra svojsta se prenose na sljedeće generacije. Kako gene prenosimo generacijama, potrebno je očuvati dobar genetski materijal i često se događa da se materijal izgubi ako se lošijim jedinkama ne pruža prilika da ih prenesu. Genetski algoritam prema vrsti selekcije može biti generacijski ili eliminacijski. Generacijskom selekcijom se odabire kromosom iz trenutne populacije za uključivanje u sljedeću populaciju. Kromosomi tada mogu ali i ne moraju prolaziti proces reprodukcije ili mutacije. Ideja iza eliminacijske selekcije je u tome što veći dio kromosoma preživi do sljedeće generacije. Određenim operatorom selekcije se odabiru kromosomi koji će biti izbačeni. Postoji više vrsta selekcijskih operatera, a ovdje ćemo spomuti samo njih nekoliko.

2.1.1.1. Turnirska Selekcija

Vjerojatno najpoznatija selekcija radi visoke efikasnosti i lagane implementacije. U turnirskoj selekciji, nasumice je odabrano k jedinki iz veće populacije. Jedinke se nakon toga natječu međusobno. Oni sa najboljom dobrotom bivaju izabrani kao začetnici nove populacije neke sljedeće generacije. Broj jedinki koje će sudjelovati u turniru se često naziva kao 'veličina turnira'. Turnirska selekcija dozvoljava svim jedinkama da se natječu, što uvelike održava raznolikost populacije. No valja napomenuti da će konstantna raznolikost utjecati na brzinu konvergencije rješenja. Slika (2.2) ilustrira strategiju turnirske selekcije.

U primjeru sa Slike 2.3, veličina turnira je postavljena na tri, što znači da se 3 kromosoma natječu međusobno i pobjeđuje najsposobniji. U turnirskoj selekciji, veće turnirske veličine su sklone većem gubitku raznolikosti. Posljedice turnirske veličine vode ka pohlepnoj pretrazi rješenja.

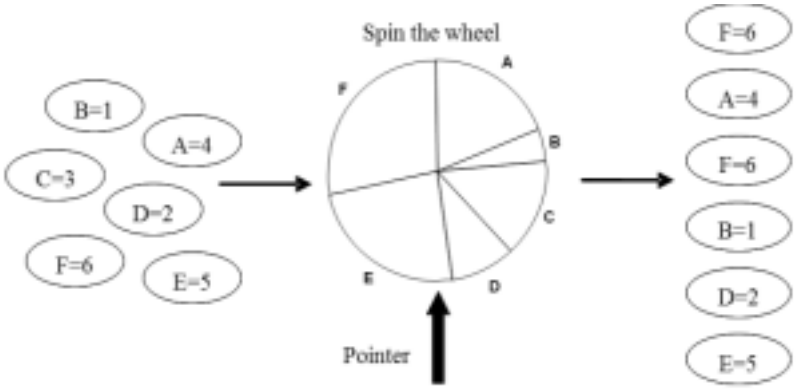
Slika 2.2: Strategija turnirske selekcije

Dva su razloga gubitka raznolikosti; neke jedinke nisu uopće odabrane kako bi sudjelovale u turniru ili su neke jedinke jednostavno prerano izgubile pa ne mogu sudjelovati u daljnjim populacijama.

2.1.1.2. Roulette Wheel Selekcija

U jednostavnog proporcionalnoj selekciji, jedinke su izabrane na način da je vjerojatnost direktno proporcionalna sa njihovom dobrotom. Vjerodostojni prikaz odnosa dobrota između jedinki bi bio '*pie chart*' koji podsjeća na kolo ruleta gdje je svaki isječak vjerojatnost da se ta jednika izabere. Očigledno je da će one jedinke sa boljom dobrotom imati veću vjerojatnost da budu izabrane jer će zauzimati više prostora na kolu. Opseg kola predstavlja sumu svih dobrota svih jedinki. Kada se kolo zavrti na slici 2.3, pokazivač će nakon zaustavljanja najvjerojatnije pokazivati na onaj najveći segment. No u svakom slučaju se može reći da svaki segment ima šansu sa vjerojatnošću širine isječka segmenta. Stoga je vidljivo da će raznolikost populacije biti očuvana. Ako označimo dobrote različitih jedinki (1, 2, 3, ... , n) sa $f_1, f_2, f_3, ..., f_n$ onda je vjerojatnost odabira pi i-te jedinke definiran kao,

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{2.1}$$



Slika 2.3: Shema Roulette Selekcije

2.1.1.3. Eliminacijska selekcija

Kao suprotnost jednostavnoj selekciji, eliminacijska selekcija bira loše kromosome koje zatim eliminira i zamjeni novima putem reprodukcije. Radi mogućnosti duplikata jedinki u jednoj populaciji uvodi se i poboljšanje eliminacijske selekcije. Nakon svake reprodukcije se provjerava da li novonastali kromosom već postoji. Ako postoji, postupak se ponavlja sve dok se ne generira jedinka bez duplikata. Ovaj postupak je veoma pristupačan radi njegove jednostavnosti pri implementaciji.
Elitizam je mehanizam koji se pojavljuje kad je potrebno zaštiti dobru jedinku neke generacije. Javlja se mogućnost eliminacije te jedinke silnom upotrebom genetskih operatora tijekom niza iteracija. Ovom procedurom se svakom generacijom približavamo rješenju problema, odnosno globalnom optimumu, jer štitimo važne jedinke. Budući da se u svakom koraku mora pronaći najbolja jedinka kako bismo je zaštitili od eliminacije, genetički algoritam se može znatno usporiti tom monotonom pretragom.

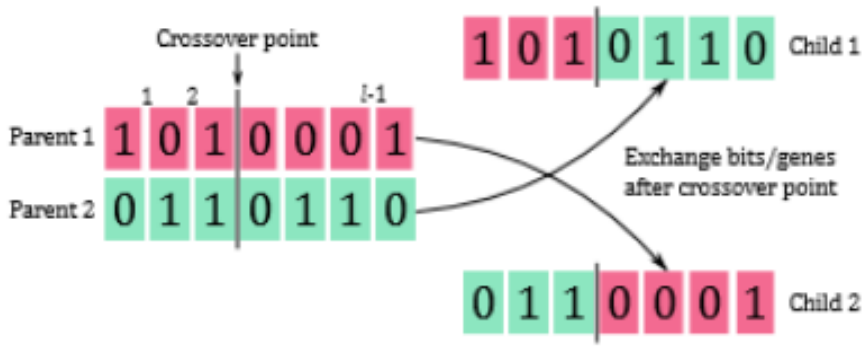
2.1.2. Križanje

Križanje (*eng. Crossover*) je genetski operator koji služi za izmjenu genetskog materijala između populacije kroz određen broj generacija. U procesu sudjeluju roditelji koji prenose svoje gene na novonastalu djecu (*offspring*). Odabir jedinki čiji će kromosomi sudjelovati u križanju najčešće ovisi o funkciji evaluacije. Veća je vjerojatnost da će križanjem dobrih rješenja nastati djeca barem s takvim genima, stoga loše jedinke često nemaju prilike sudjelovati u procesu. Dijete nakon procesa križanja posjeduje isti broj kromosoma kao i roditelji. Dobivena populacija je mogla nastati nasumice

odabranom izmjenom kromosoma ali se koriste neke utjecajnije metode odabira.

2.1.2.1. Križanje točkom prekida

Najjednostavnije i najintuitivnije je križanje s jednom točkom prekida (Slika 2.4). Odabrana su dva roditelja čija je kromosomska reprezentacija binarni niz. Točka prekida se bira u granicama duljine kromosoma, te se nakon toga dio prije prekida uzima od jednog roditelja, a drugi dio od drugog roditelja. Na ovaj način se geni cjepkaju na segmentu i zadržava se njihov inicijalni poredak unutar tog segmenta. Križanje sa dvije ili više točaka prekida funkcionira na sličan način.



Slika 2.4: Križanje točkom prekida [2]

2.1.2.2. Uniformno križanje

Uniformno križanje koristi definirani omjer križanja (*mixing ration*) između dva roditelja. Križanje prekidnim točkama sa $n - 1$ prekida gdje je je n broj bitova se naziva uniformno. Pokazalo se da je omjer križanja tipično 0.5 i smatra se da je takav pristup više stohastički nastrojen. *"Uniformno križanje je najlakše i najjednostavnije implementirati kao logičku operaciju, a i najbrže je, jer su logičke operacije sastavni dio strojnog jezika računala."* [6]. Budući da su kromosomi prikazani kao niz binarnih brojeva, nad njima se izvršavaju kombinacije binarnih logičkih operacija XOR, AND i OR. Za razliku od križanja u jednoj ili više točaka, operacije omogućuju parcijalan odabir gena roditelja.

Valja spomenuti i *p-uniformno* križanje, kod kojeg svaki gen dodatno ima vjerojatnost *p* koja odlučuje kolika je mogućnost da ga dijete dobije od pojedinog roditelja. Više o operatorima križanja koji se koriste u našem problemu kasnije.

2.1.3. Mutacija

Prilikom pretrage rješenja genetičkim algoritmom može se dogoditi sljedeći scenarij. U određenom broju generacija se mogu pojaviti rješenja koja se više ne razlikuju od onih dobivenih u ranijim generacijama. Razlozi tome su različiti, ali glavni odgovor na ovaj problem daje mutacija gena. Mutacija prije svega služi za unošenje i održavanje raznovrsnosti u populaciji. Procesom mutacije se u kromosomu jedinke mjenjaju određeni geni. Pozitivan utjecaj mutacije se može uočiti u situaciji kada algoritam zapne u lokalnom optimumu, no mjenjanjem pojedinih gena se mogu generirati rješenja kakva nisu do tada postojala. Vjerojatnost mutacije (pm) je parametar koji određuje kolikim će udjelom pojedini geni sudjelovati u mutaciji. Ako mutiramo gene s malom vjerojatnošću mutacije ($pm \approx 0$), postupak optimiranja bi mogao brzo konvergirati i zapeti u lokalnoj točki minimuma. S druge stanje preveliki faktor mutacije ($pm \approx 1$) alogitam svodi na potpuno nasumičnu metodu pretrage sa značajnim gubitkom genetičkog materijala.

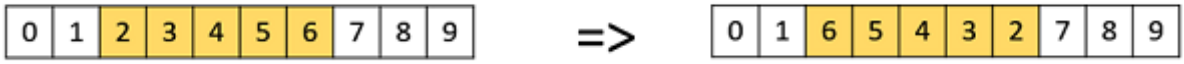
2.1.3.1. Bit-flip mutacija

Ova mutacija je jednostavna, od l bitova kromosoma, bit na $j - tom$ mjestu se pretvara u komplementarnu vrijednost (0 ili 1). Koristi se kada je rješenje genetičkog algoritma implementirano kao birani niz. Navedena mutacija je prikazana na slici 2.5.

Slika 2.5: Bit-flip Mutacija

2.1.3.2. Inverzna mutacija

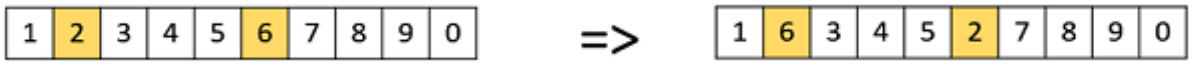
Kako izgleda mutacija kada rješenje nije niz binarnih brojeva već permutacija cijelih brojeva, dano je na slici 2.8. Nasumice odabiremo interval kromosoma i zatim ga zrcalimo.



Slika 2.6: Inverzna mutacija

2.1.3.3. Mutacija zamjenom

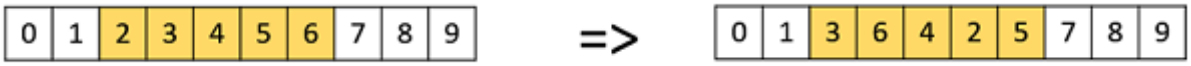
Nasumice odabiremo dva gena u kromosomu i zamjenjujemo ih međusobno. Klasični način rješavanja u mnogim permutacijskim problemima.



Slika 2.7: Mutacija zamjenom

2.1.3.4. Mutacija mješanjem

Sličan princip kao i kod inverzne mutacije, odabiremo nasumice podset gena kromosoma i izmješamo slučajanim poredakom. (Slika 2.8) [3]



Slika 2.8: Mutacija mješanjem

3. Problem trgovačkog putnika

Problem trgovačkog putnika (*Travelling Salesman Problem*) jedan je od najpoznatijih problema računarne znanosti. Definicija problema je prilično jednostavna i intuitivna: *Za zadanu listu gradova i udaljenosti između svaka dva grada, koja je najkraća moguća ruta koja posjećuje svaki grad točno jednom i vraća se u početni grad.* Drugačija interpretacija ovog problema je matematičkom formulacijom.
Zadan je težinski graf $G = (V, E)$ u kojem je težina t_{ij} između vrhova i i j pozitivna vrijednost. Vrhovi grafa predstavljaju gradove, a bridovi puteve između dva grada. Tražimo Hamiltonov ciklus minimalne težine. Hamiltonov put (ciklus) u grafu G je put koji sadrži sve vrhove grafa. Graf je Hamiltonov ako sadrži Hamiltonov ciklus. Stoga je očito da skup gradova koji obilazimo mora biti Hamiltonov graf. Dokazivanje da je graf Hamiltonov kao i pronalazak Hamiltonovog ciklusa, odnosno rješavanje TSP-a spada u kategoriju teških algoritamskih zadataka. Ono što dodatno otežava ovaj problem je to što nije nužno simetričan. Težina iz grada i u grad j ne mora biti jednaka težini u suprotnome smjeru, te se tu problem razlaže na mnogo podvrsta koje ovise o različitim resursima. Pretpostavimo da je graf koji predstavlja mrežu gradova potpun, odnosno postoji konekcija između bilo koja dva grada. U takvom potpunom grafu s n vrhova postoji $(n-1)!/2$ različitih hamiltonovskih ciklusa. Prvi vrh možemo fiksirati jer kroz njega sigurno prolazi ciklus [7]. Može se lako pokazati da zahtjev da se vratimo u početni grad ne mijenja kompleksnost ovog problema. Budući da imamo $(n-1)!$ mogućnosti, kada n teži u beskonačnost, složenost teži prema $O(n!)$.

3.1. Varijante problema

Kako problem nije sasvim jednoznačno određen, postoje mnoge varijacije na temu. U stvarnosti se javljaju raznorazne prepreke koje pridonose ukupnoj težini nekog puta. Najlakše je zamisliti situaciju u kojoj je put AB između gradova A i B jednosmjeran.

Tada se pri implementaciji za težinu brida BA postavlja velika udaljenost ($t_{BA} => \infty$). Osim udaljenosti, za izračun težine brida uzimaju se troškovi poput goriva, cestarine, najma vozila i sličnih. Valja uočiti da radi sume troškova neki bridovi mogu biti negativne težine, što se opet može tretirati ogromna udaljenost. Ovdje su prikazane neke od najčešće korištenih varijanti problema trgovačkog putnika.

3.1.1. Simetrični TSP

Najjednostavniji oblik u kojem je mapa gradova predstavljena pomoću potpunog neusmjerenog težinskog grafa.

3.1.2. Asimetrični TSP

Asimetrična varijanta TSP-a podrazumijeva barem jedan par gradova A i B u kojem je smjer obilaska bitan, odnosno $t_{AB} \neq t_{BA}$. Ovakav prikaz se ostvaruje korištenjem usmjerenog grafa.

3.1.3. Metrički TSP

Ovu varijantu zadovoljavaju samo metrike koje zadovoljavaju svojstva metrike (nenegativnost, identitet, simetrija, nejednakost trokuta). Posebno nas zanima svojstvo nejednakosti trokuta koje vrijedi za udaljenost između gradova 3.1.

$$\forall A, B, C \quad d(A, C) \leqslant d(A, B) + d(B, C) \tag{3.1}$$

3.1.4. Višestruki TSP

Graf G je podijeljen na m podgrafova koji imaju zajednički vrh P (početni vrh). Iz vrha P kreće m trgovačkih putnika od kojih svaki obilazi svoj podgraf i vraća se u vrh P. Najkraća udaljenost je zbroj prevaljenih udaljenosti svih m putnika.

3.2. Metode rješavanja problema

Ako tražimo točno jedno određeno rješenje onda govorimo o egzaktnim algoritmima. Njih prepoznajemo po tome što je naš ishod sigurno najbolje rješenje. Problem je u njihovoj vremenskoj složenosti. Većini takvih algoritama je vrijeme izvođenja preveliko za efikasnu uporabu. Pretragom grube sile (*Brute-force search*) bi pronašli sve moguće permutacije rješenja i odabrali minimalno među njima. Metodom granjanja i ograničavanja (*Brach and bound*) bi pomoću prethodno određenih granica odbacivali loša rješenja iz ogromnog prostora rješenja i u najgorem slučaju imali složenost $O(n!)$ kao i kod pretrage grubom silom. U potrazi za bržim vremenom izvođenja prelazimo na aproksimaciju optimalnog riješenja nekim drugim metodama.

3.2.1. Aproksimativne metode

Budući da ne postoji rješenje koje će u polinomijalnom vremenu dati odgovarajuće rješenje, dok se ne pronađe takvo rješenje (ako uopće postoji) zadovoljit ćemo se dobrim aprokcimacijama u konačnom vremenu.

3.2.2. Pohlepni algoritam

Počinjemo sa jednostavnom logikom, odaberi grad i promatraj bridove. U ovog postupku se put stvara tako da se uvijek dodaje najkraći mogući brid. Pokazuje se da ovaj postupak nije baš efikasan. Može se dogoditi da se ciklus prerano zatvori ili da neki vrh nije stupnja 1 kao što bi prema pravilima trebao biti [5]. Složenost ovog algoritma za n gradova je $O(n^2 log_2(n))$.

3.2.3. Umetanje gradova

Potreban nam je najmanji brid u grafu. Njega povezuju dva grada pa njih označavamo kao početno rješenje. Grad koji sljedeći dadajemo u obilazak je onaj najdalje udaljen od skupa gradova koji već posjedujemo. On se dodaje na optimalno mjesto u obilasku. Algoritam može početi s više gradova u početnom rješenju udaljenim najkraćim bridovima. Složenost je $O(n^2)$ [5].

3.2.4. Kolonija mrava

Ova metoda je proučavana početkom 1990-ih ponajviše u kontekstu umjetne inteligencije. Prateći ponašanje kolonije mrava, znanstvenici su ovu heuristiku našli vrlo pogodnom za pojedine probleme. Kolonija mrava funkcionira sustavno kao organizam, ostavljajući pritom feromone gdje god nađu nove resurse. Algoritam započinje tako da se grupacija mrava rasprši po gradovima. Ne smiju se vratiti u početni grad sve dok ne obiđu sve druge gradove. Ključan segment je ostavljanje najintenzivnijeg traga feromona od strane onog mrava koji je našao najkraći put. Neodlučni mravi tada prate taj trag sve dok se ne nađe najkraći obilazak.

3.3. Evolucijski pristup

Pokazuje se da je pristup ovakvom i sličnim problemima učinkovitiji pomoću evolucijskih algoritama. Naime, nije potrebno pretraživati cijeli prostor rješenja, nego se procesom evolucije postepeno generacijama približavamo rješenju. Na taj način algoritam puno brže konvergira k optimalnom rješenju.

Radi jednostavnosti, za prikaz kromosoma jedinke koristiti ćemo polje prirodnih brojeva. Prije svega se n gradova numerira brojevima od 1 do n. Prvi element u polju sadrži broj predhodno numeriranog grada od kojeg počinjemo. Svako sljedeće polje sadrži broj grada kojeg idućeg posjećujemo. Prikaz permutacija navedene strukture za npr. 7 gradova:

$$[1, 3, 2, 4, 5, 6, 7] \quad \text{ili} \quad [5, 3, 2, 1, 4, 7, 6]$$

Ovakav prikaz ima svoje prednosti i mane. Svaki broj u polju se mora pojaviti isključivo jednom, stoga treba pripaziti na genetske operacije koje mogu povećati broj posjećenosti nekog grada. Polje gradova je sortirano prema načinu slijednog povezivanja hamiltonovog ciklusa. Tu se javlja mogućnost komplementarnog prikaza koji pokazuje najveću udaljenost između gradova. Kad bolje promotrimo, nije uopće bitno iz kojeg grada krećemo. Ako je dana međusobna udaljenost gradova, najkraći ciklus je unaprijed određena ruta koja sadrži sve gradove i ukupnu težinu obilaska. Valja zamisliti kako najkraćem razapinjajućem ciklusu postepeno uklanjamo vrhove povezane jednim bridom (listove), i dalje prikazujemo najkraći ciklus između preostalih gradova. Ovo se također može primjetiti kada bi naše brojeve (gradove) rotirali unutar

polja. Ukupna udaljenost između svih gradova bi i dalje bila minimalna.

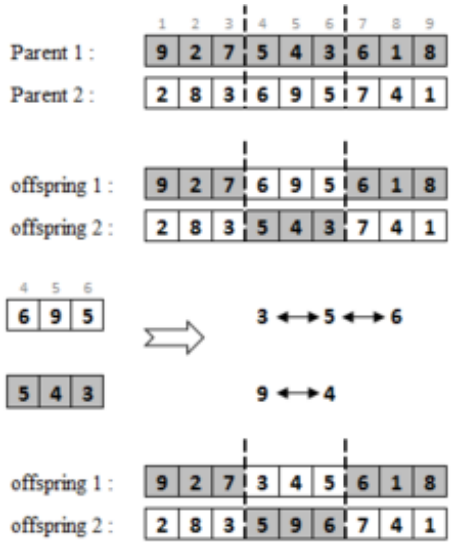
U konačnici genetskim algoritmom pouspješujemo neke od adaptivnih algoritama posebnim genetskim operatorima specifičnima za ovaj problem.

3.3.1. GX križanje

Pohlepno križanje je vrlo jednostavno. Uzima se jedan grad jednog roditelja i usporedbom određuje najkraća udaljenost od tog grada do ostalih gradova oba roditelja. Nakon toga se taj grad pridodaje djetetu. Ukoliko se odaberu gradovi koji su se već pojavili, koristi se slučajan odabir grada sve dok nisu svi gradovi odabrani.

3.3.2. PMX križanje

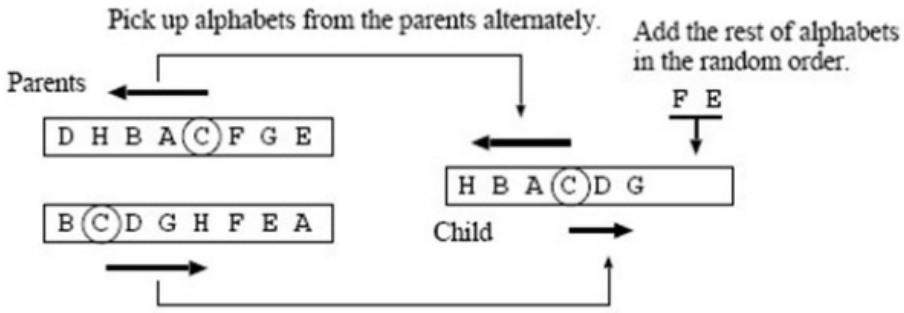
Veoma slično već spomenutom križanju s k- prekidnih točaka opisano u 2. poglavlju. Ideja je izbaciti višestruko pojavljivanje gradova koje se može dogoditi pri razmjeni gena dvaju roditelja. Na slici 3.1 se nalazi primjer križanja sa dvije prekidne točke. U prvom koraku se izmjeni sadržaj podijeljen na tri djela. Očito su gradovi 6 i 9 u 1. djetetu i gradovi 3 i 4 u 2. djetetu posjećeni dvaput. Problem se rješava tako da se ponavljajući gradovi zamjene onima koji su bili na istom mjestu kod drugog roditelja prije križanja. U našem slučaju prije križanja su na indeksu 4 bili gradovi 5 i 6, na indeksu 6 gradovi 3 i 5. Stoga se grad 6 1. djeteta mjenja u 5, pa zatim radi duplikata na indeksu 6 u grad s brojem 3. Na sličan način se odvijaju postupci za drugo dijete.



Slika 3.1: Partially Matched Crossover

3.3.3. GSX križanje

Najučinkovitije križanje koje za razliku od PMX-a znatno bolje čuva genetski materijal. Slično je pohlepnom algoritmu jer se odabire neki grad i vrši pretraga kod oba roditelja. Traži se što dulji podskup najkraće udaljenih gradova povezanih sa odabranim gradom kao što je prikazano na slici 3.2. Nakon preslikavanja nasumičnog gena djetetu, lijevo i desno od tog gena se nadodaju geni roditelja sve dok se ne naiđe na kraj ili neki gen koji dijete već posjeduje. Tada se dijete slijedno popunjava preostalim genima odnosno gradovima jednog od roditelja.



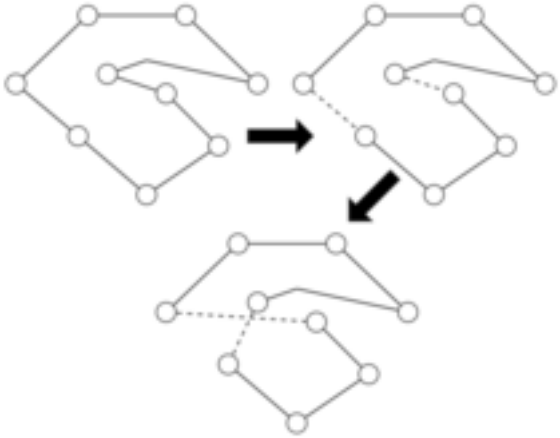
Slika 3.2: *Greedy Sub-tour Crossover*

3.3.4. GSM mutacija

Pohlepna mutacija zamjenom (Greedy-swap mutation) je mutacija u kojoj su odabrana dva grada jednog kromosoma. Ako se gradovi zamjene i zadovoljavaju minimalni uvjet obilaska, mutacija se manifestira nad tom jedinkom. Ova mutacija je nastala kao spoj mutacije zamjenom i pohlepnog alogoritma.

3.3.5. K-opt Mutacija

$K - opt$ mutacija bira nasumično k nesusjednih bridova u ciklusu koje zatim uklanja. Nastaju dva puta koja se zatim spajaju sa k novih bridova. Ukoliko ukupan novi ciklus nije kraći od prijašnjeg, vraćaju se stari bridovi. Najpoznatija je $2 - opt$ heuristika koja se koristi u različitim genetskim algoritmima. $3 - opt$ algoritam uklanja 3 brida te dodaje 3 brida na 2 moguća načina. Općenito, $k - opt$ mutacija uz t fiksnih točaka ima $(k - t)$ mogućih rekombinacija od kojih se uzima najkraća. Slika 3.3 prikazuje jednostavnu primjer $2 - opt$ mutacije.



Slika 3.3: 2-opt Mutacija

Poznata Lin-Keringhan metoda se upravo temelji na k-opt algoritmu. U svakoj iteraciji se mijenja vrijednost k, te ispituje da li to pridonosi kraćim obilaskom [5]. Glavna ideja je ograničiti broj k-opt koraka i svesti ih na 2-opt korake jer je njihova složenost najmanja. Ova mutacija se treba pomno birati sa križanjem kako algoritam nebi prerano konvergirao. Radi konstante potrebe za izborom k bridova složenost ove metode je veća nego kod 2-opt zamjene.

3.4. Primjene u svakodnevnici

Radi svoje jednostavnosti i sveobujnosti, koncept trgovačkog putnika uvelike sudjeluje u našem životu. Rute kojima čovjek prolazi svakodnevno mogu itekako biti minimizirane. Težine tada više nisu nužno samo udaljenosti. U svakom projektiranju ideje se javljaju sitni sukobi vlastitih ili tuđih interesa koji se mogu poistovjetiti sa ponovnim posjećivanjem vrhova. Tako je razvojem prometa određivanje rute postalo problemom. U minimalnom vremenu treba n dostavljača raspršiti po točno određenim lokacijama kako bi podijelili svoje pakete što brže i s manje troškova. Spajanje komponenti matične ploče računala te raspored poslova koje obavlja radna jedinica također podsjećaju na trgovačkog putnika. U konačnici se svaki naš korak se može pametnije organizirati koristeći neku od metoda rješavanja problema trgovačkog putnika.

4. Zaključak

Čini se kako genetički algoritam doista pronalazi dobra rješenja za problem trgovačkog putnika. Razlike u rješenjima ponajviše ovise o izboru križanja i mutacije. Kroz niz godina, razvilo se mnogo metoda čiji se hidbridi i dalje stvaraju. Izgleda kako je najveći problem genetskog algoritma kod trgovačkog putnika upravo konzistentnost legalnih rješenja budući da se nekim genetskim operatorima narušavaju pravila obilaska. Genetski algoritam je dobar način pronalaska rješenja u konačnom vremenu na velikom prostoru. Pokazuje se kako korištenje genetskih operatora u kombinaciji s već poznatom metodom pretrage rezultira uspješnim algoritmom. Rezultate koje daje genetski algoritam obično odstupaju za svega par posto od globalnog optimuma što ih čini više nego dobrom zamjenom za kojekakve probleme.

5. Literatura

[1] http://www.math.uwaterloo.ca/tsp/history.

[2] http://www.dataminingapps.com/2017/03/solving-the-knapsack-problem-with-a-simple-genetic-algorithm, .

[3] https://www.tutorialspoint.com/genetic_algorithms, .

[4] http://www.zemris.fer.hr/~golub/ga/studenti/prokopec/diplomski.htm.

[5] https://en.wikipedia.org/wiki/Travelling_salesman_problem#Special_cases_of_the_TSP.

[6] Marin Golub. Genetski algoritam 1.dio. http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf, Listopad 1997.

[7] Mario-Osvin Pavčević. Uvod u teoriju grafova, 2006.

[8] Tanja Vojković Velga Bosančić, Anka Golemac. Kako pomoći trgovačkom putniku, Listopad 2012.

6. Sažetak

Problem trgovačkog putnika je jedan od najpoznatijih problema kombinatorne optimizacije u kojem putnik treba obići sve gradove jednom i vratiti se u početni. U ovom radu je opisan genetski algoritam i njegova primjena na problemu trgovačkog putnika. Genetski algoritmi su heurističke metode inspirirane procesom evolucije. Navedeni su neki od poznatijih operatora korisnih za problematiku trgovačkog putnika, te su objašnjene veze koje čine baš te operatore pogodnima. Pokazano je kako je odabir genetskih operatora krajnje bitan kako bi se izbjegli lokalni optimumi, te je spomenuta konkretna primjena problema u svakodnevnici.