[[1]](#footnote-1)

Dokumentacija za projektni zadatak

Miroslav Đoćoš BI55/2019, [djotoss@gmail.com](mailto:djotoss@gmail.com) ili [djocos.miroslav@uns.ac.rs](mailto:djocos.miroslav@uns.ac.rs)

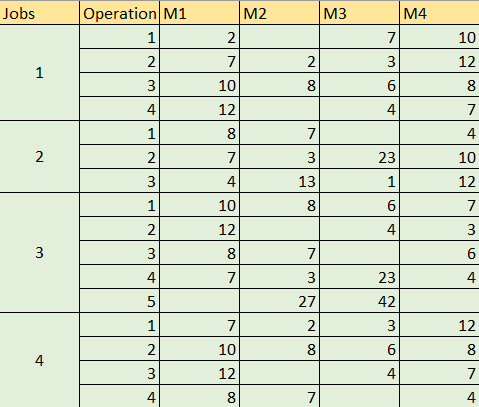
# Uvod

U današnje vreme, sve je manje postoji potreba za masovnom proizvodnjom, a sve više za proizvodnjom individualizovanih proizvoda koji se plasiraju kupcima. Job shop scheduling (JSS) je jedan primer iz problema optimizacije uređenja poslovanja modernih fabrika koje poseduju više mašina pomoću kojih se može izvršiti više različitih poslova, pri čemu svaka mašina može izvršiti istu operaciju za različito vreme, ili je čak ni ne može izvršiti. U ovom radu biće predstavljena implementacija genetskog algoritma, koji rešava ovakve JSS probleme.

# Job shop scheduling

Kao što je pomenuto u uvodu, JSS rešava problem optimizacije organizacije poslovanja fabrike, koja poseduje više mašina, gde svaka može da izvrši više zadatih poslova sastavljenih od pojedinačnih operacija. Treba napomenuti da je problem definisan i rešavan prvenstveno radi bolje organizacije fabrika, tj. zbog potrebe da se sa što manje resursa u što kraćem roku izvrše zadati zadaci, ali ne mora se odnositi samo na mašine i fabrike, već na čitavu klasu problema gde je potrebno izvršiti više zadataka za što kraće vreme.

JSS problem se najčešće zadaje pomoću tabele koja ne samo da je pregledna, već se iz nje takođe programski lako mogu učitavati podaci. Primer takve tabele prikazan je na *slici 1.*



*Slika 1: Primer tabele za zadavanje spiska poslova sastavljenih od pojedinih operacija, spiska mašina i vremena za koje pojedina mašina može da izvrši pojedinu operaciju*

Može se primetiti da u tabeli na *slici 1* postoje nepopunjena polja. Nepopunjeno polje za neku mašinu simbolizuje da je datu operaciju iz konkretnog zadatka nemoguće izvršiti. Matematički se model JSS može opisati na sledeći način: Postoji **n** poslova J = (J1 , J2 , ... , Jn ), koji se obrađuju na **m** različitih mašina M = (M1, M2, .., Mm). Svaki posao Ji sadrži **n** uzastopnih operacija koje se moraju obaviti, izraženo kao skup operacija Oi = {Oi,1 , Oi,2 , ..., Oi,ni }. **L** je ukupan broj operacija svih poslova. Svaka operacija Oi,j može biti obrađena na predefinisanom skupu mašina za obradu iz skupa svih dostupnih mašina Mi,j ∈ M. Cilj optimizacije *JSS* je da se odabere mašina Mk iz skupa mašina Mi,j za svaku operaciju Oi,j i odrediti vreme početka STi,j i vreme završetka FTi,j svake operacije tako da se minimizuje vreme potrebno za završetak rada svih mašina.. To se može zapisati matemtičkom jednačinom kao:

Prilikom implementacije, mora se vodti računa da se ispoštuje nekoliko ograničenja.

1. Sve mašine su dostupne u trenutku 0, svaka operacija se može obraditi u trenutku 0

2. U određeno vrijeme, jednu operaciju istovremeno može obraditi samo jedna mašina. Jednom kada je operacija završena, mašina se može koristiti u drugim operacijama

3. Kada obrada započne, ne može se prekinuti

4. Ne postoje ograničenja u redosledu izvršavanja operacija između različitih poslova, dok takva ograničenja postoje između operacija istog posla

5. Vreme za pripremu mašine i vreme transporta tokom rada su zanemarivi, ili su uračunati u samo trajanje operacije

6. Nema hitnih slučajeva (kao što su kvarovi mašina, nove narudžbe itd.)

# Implementacija genetskog algoritma

Za rešavanje JSS problema pogodni su algoritni dobre globalne pretrage, kao što je na primer genetski algoritam (GA). Ako bi se primenjivala metoda računanja ukupne dužine trajanja za svaku moguću kombinaciju, čak i za relativno mali problem, rešavanje može biti prilično skupo.

Za primenu GA, potrebno je prvo formirati jedinku (hromozom). U ovoj implementaciji, jedinka se formira od dva niza dužine **L**. Prvi niz predstavlja redosled izvršavanja poslova, gde je svaki član celobrojna vrednost iz skupa J. Drugi deo jedinke je takođe niz koji se sastoji od naziva mašina koje se koriste u izvršavanju operacija (npr. 'M2'). Pa tako jedan hromozom može izgledati kao što je prikazano u *tabeli 1*, gde **JO** predstavlja prvio deo hromozoma a **MA** drugi.

*Tabela 1: Primer jednog formiranog hromozoma*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **JO** | 3 | 1 | 3 | 4 | 2 | 4 | 4 | ... |
| **MA** | M3 | M1 | M4 | M2 | M2 | M3 | M4 | ... |

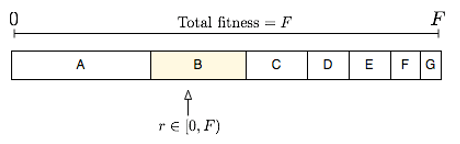
Ovako formiran hromozom se može čitati na sledeći način:

1. Izvršavanje operacije 1 iz posla 3 na mašini M3 (prva operacija zato što se 3 pojavljuje prvi put)
2. Izvršavanje operacije 1 iz posla 1 na mašini M1
3. Izvršavanje operacije 2 iz posla 3 na mašini M4 (druga operacija zato što se 3 pojavljuje drugi put)
4. ...

Nakon formiranja grupe jedinki (populacije) se vrši računanje vrednosti kriterijuma optimalnosti za svaku jedinku, dalje se one rangiraju na osnovu te vrednosti i vrši se selekcija.

Selekcija je dakle postupak kojim se na osnovu vrednosti kriterijuma optimalnosti biraju jedinke koje će učestvovati u formiranju nove generacije. Odabran metod za metodu selekcije je ruletska selekcija. Na taj način je obezbeđeno da sama vrednost kriterijuma optimalnosti direktno utiče na odabir jedinki, ali se smanjuje rizik upadanja u lokalni minimum i povećava se prostor pretrage (ne izbace se malo lošija rešenja iz opticaja odmah). Ako je vrednost kriterijuma optimalnosti i-te jedinke, verovatnoća da će ta jedinka biti odabrana za formiranje sledeće generacije dobija se kao:

gde je broj jedinki populacije. Naziv rulet slekcija je ova metoda dobila po tome što se može objasniti preko primera o ruletu u kasinu, gde taj rulet ima nejednaka polja. Verovatnoća da loptica na slučajan način padne na tačno određeno polje računa se kao odnos dimenzija tog polja i dimenzija celokupnog ruleta (zbir dimenzija svih polja). Dodatno pojašnjenje može se videti na *slici 2.* Nakon selekcije dobija se lista parova jedinki koja je spremna za operaciju ukrštanja.



*Slika 2: Nasumično izabran hromozom na osnovu slučajno izgenerisanog r iz opsega [0, F)*

Proces ukrštanja kod rešavanja JSS problema primenom GA predstavlja najkompleksniji deo ovog zadatka. Naime, kompleksan je zbog toga što se mora voditi računa da se ne prekrši ni jedno od ograničenja navedenih u prethodnom poglavlju. Nakon ukrštanja jednog para koji se sastoji od dva roditelja, nastaju dva deteta hromozoma, a kada se ukrste svi parovi, nastane nova generacija koja je iste veličine kao generacija roditelja. Odabrana metoda za ukrštanje je ukrštanje u jednoj tački **JO** dela hromozoma. Inicijalno je implementirano da to bude u tački sa čije leve strane se nalazi 60% **JO** dela hromozoma. U *tabeli 2*  je prikazano kako bi izgledala podela dva hromozoma gde je prvi hromozom obeležen sa crvenim slovima, dok je drugi obeležen sa ljubičastim.

*Tabela 2: Ilustracija ukrštanja hromozoma, roditelji*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **JO** | **3** | **2** | **4** | **3** | **1** | **2** | **4** | **1** | **3** | **3** |
| **MA** | **M3** | **M1** | **M4** | **M2** | **M2** | **M3** | **M4** | **M1** | **M1** | **M4** |
| **JO** | **1** | **3** | **3** | **1** | **4** | **3** | **4** | **3** | **2** | **2** |
| **MA** | **M2** | **M2** | **M3** | **M4** | **M1** | **M4** | **M2** | **M2** | **M3** | **M4** |

Proces ukrštanja se odvija na sledeći način:

1. **JO** deo hromozoma *dete 1* se formira tako što se prvih 60% prepiše iz roditelja 1 (zeleni deo tabele)
2. Ostalih 40% **JO** dela se formira tako što se iterira redom kroz vrednosti iz **JO** roditelja 2, ali tako da se za svaku vrednost (naziv posla) proverava da li ta vrednost fali u *detetu 1* da bi se održao isti broj operacija za svaki posao. Ako fali, ta vrednost (naziv posla) se upisuje na prvo slobodno mesto.
3. Što se tiče **MA** dela, za *dete 1* se prvih 60% takođe prepisuje od roditelja 1.
4. Za preostalih 40% prati se sa kojih se pozicija prepisuju vrednosti iz **JO** od roditelja 2 i sa tih pozicija iz **MA** dela od roditelja 2 se prepisuju *detetu 1.*
5. Ponoviti postupak i za *dete 2*

U *tabeli 3* može se videti kako izgledaju deca, tj. jedinke nastale ukrštanjem.

*Tabela 3: Prikaz dece nastale ukrštanjem*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **JO** | **3** | **2** | **4** | **3** | **1** | **2** | **1** | **3** | **3** | **4** |
| **MA** | **M3** | **M1** | **M4** | **M2** | **M2** | **M3** | **M2** | **M2** | **M3** | **M1** |
| **JO** | **1** | **3** | **3** | **1** | **4** | **3** | **3** | **2** | **4** | **2** |
| **MA** | **M2** | **M2** | **M3** | **M4** | **M1** | **M4** | **M3** | **M1** | **M4** | **M3** |

Ovako formiran algoritam ima tendenciju da grupiše rešenja oko nekoliko tačaka, a te tačke mogu bit lokalni minimumi. Takva pojava onemogućuje uspešni nastavak pretrage jer su jedinke dosta slične. Iz tog razloga uvodi se mutacija.

Dakle, prilikom formiranja svakog deteta, postoji predefinisana verovatnoća da se desi mutacija. Ako jedinka upadne u postupak mutacije, postoje dva ishoda koja su podjednako verovatna i biraju se na slučajan način.

1. Mutacija **JO**
2. Mutacija **MA**

Za slučaj mutacije **JO** odabrana metoda za mutaciju je swap metoda, dakle izvrši se zamena pozicija dva susedna elementa u **JO** delu hromozoma a na njihovim pozicijama u delu **MA** se mašine biraju od ponuđenih validnih (mašine koje mogu da izvrše datu operaciju o datom poslu). Kod **MA** mutacije se na mestu mutacije na slučajan način izvrši odabir nove mašine iz liste mogućih mašina za datu operaciju (ne isključuje se mogućnost ponovnog odabira iste našine). Mesto mutaicije se takođe bira na slučajan način.

Da bi se osiguralo napredovanje algoritma samo ka boljem rešenju, uveden je elitizam od 10%. Elitizam predstavlja postupak čuvanja određenog procenta najuspešnijih roditelja, a isti taj procenat najneuspešnije dece se eliminiše.

Ceo prethodni postupak se ponavlja do konvergencije najboljeg rešenja, tj. u ovom slučaju 20 uzastopnih istih rezultata. Ovaj broj je moguće promeniti u zavisnosti od zahteva konkretnog problema.

# Rezultati

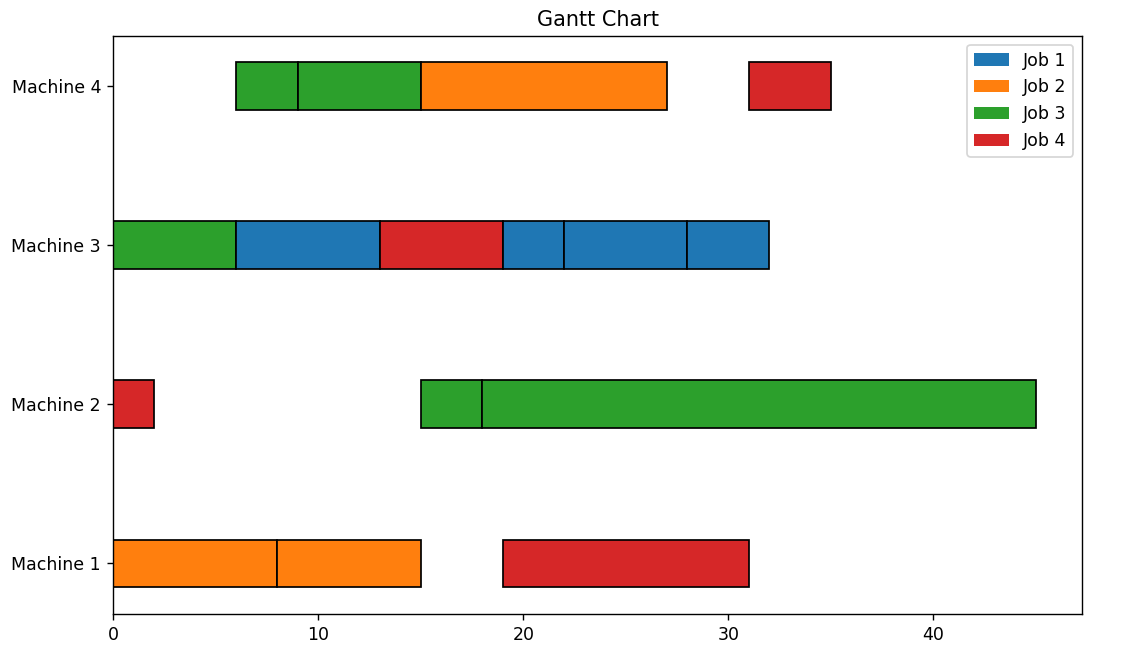
Ovako formiran algoritam rešava zahtevne probleme u roku od nekoliko sekundi, i uz nekoliko pokretanja pronalazi optimalno rešenje. Za problem prikazan na *slici 1* dobijeni su sledeći rezultati:

* Rešenje se dobije za 22 generacije (20 zahteva samo konvergencija).
* Vrednost kriterijuma optimalnosti je 45

Hromozom za koji su ovi rezultati dobijeni je:

[[4, 3, 2, 2, 1, 3, 4, 4, 1, 3, 4, 3, 2, 1, 3, 1], ['M2', 'M3', 'M1', 'M1', 'M3', 'M4', 'M3', 'M1', 'M3', 'M4', 'M4', 'M2', 'M4', 'M3', 'M2', 'M3']]

Na *slici 3* prikazan je Gantt Chart koji pokazuje kako su se operacije rasporedile po vremenu i po mašinama.



*Slika 3: Prikaz rezultata jednog od pokretanja algoritma*

1. [↑](#footnote-ref-1)