SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6387

OCJENA UČIKOVITOSTI PRIKAZA RJEŠENJA U RASPOREĐIVANJU NA NESRODNIM STROJEVIMA

Mile Čarić

Zagreb, lipanj 2019.

Zahvaljujem mentoru prof. dr. sc Domagoju Jakobovića

i asistentu dr. sc. Marku Đuraseviću na pomoći i vodstvu tijekom izrade ovog rada.

Sadržaj

Uvod

Raspoređivanje je proces kojim se dodjeljuje određen broj poslova određenom broju strojeva s ciljem optimiranja jednog ili više kriterija. Problem je sveprisutan i može se vidjeti u raspoređivanju aviona u zrakoplovnim lukama, u raspoređivanju radnika u brojnim djelatnostima, regulaciji semafora u prometu, raspoređivanju na nesrodnim strojevima i drugim procesima. Cilj raspoređivanja je dobiti raspored koji definira kada će se i na kojem stroju obavljati točno određeni posao. Raspored je moguće vrednovati po različitim kriterijima poput ukupnog vremena izvođenja ili po broju zakašnjelih aktivnosti.

Razvojem tehnologije raste i složenost raspoređivanja, velik dio takvih problema je vrlo težak za rješavanje. Često nije moguće upotrijebiti rješenje koje jamči pronalazak optimalnog rješenja, zbog vremenskog ograničenja ili nepostojanja algoritma koji to omogućava. Zbog navedenog problema se razvijaju metaheuristički algoritmi kojima se ne pronalazi nužno najbolje rješenje već 'dovoljno dobro' rješenje uz zadovoljavanje ograničenja za pronalazak rješenja. U metaheurističke algoritme ubrajaju se i genetski algoritmi korišteni u ovom radu.

Strojeve smatramo nesrodnima ako vrijeme obavljanja pojedinog posla ovisi o stroju na kojem se izvodi. Kod raspoređivanja na nesrodnim strojevima posao se dodjeljuje jednom stroju na kojem se izvodi u cijelosti. Cilj ovog rada je implementirati novi način prikaza rješenja raspoređivanja na nesrodnim strojevima i usporediti njegovu učinkovitost u odnosu na već postojeća rješenja. Za potrebe implementacije prikaza rješenja bit će napravljena inicijalizacija početne populacije i dodat će se novi genetski operatori križanja i mutacije. Dodatno prilagodit će se evaluacija kako bi uspješno ocijenila implementirano rješenje i testirat će se različiti parametri da bi se vidjelo kako se mijenja ocjena rješenja u ovisnosti o parametrima.

Ostatak ovog rada je organiziran u dva dijela. Prvi dio rada će ukratko objasniti način rada genetskih algoritama, detaljno opisati prikaz rješenja raspoređivanja i opisati dodane genetske operatore križanja i mutacije. Drugi dio rada će prikazati kriterije po kojima je obavljeno testiranja i detaljan prikaz dobivenih rezultata u usporedbi sa već postojećim rješenjima, na kraju će bit iznesen pregled rada i zaključak.

1. Genetski algoritmi

Ideja iza genetskih algoritama može se naći u mehanizmu prirodnog odabira. U prirodi preživljavaju najbolje prilagođene jedinke situaciji u okolišu. Najsposobniji preživljava i dobiva priliku da se reproducira i time prenosi svoj genetski materijal. Genetski algoritmi imitiraju prirodni proces evolucije. Potencijalno rješenje predstavlja jedinku, a situaciju u okolišu predstavlja funkcija cilja.

```
Genetski_algoritam
{
   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 u P(t);
        mutiraj jedinke iz P(t);
   }
   ispiši rješenje;
}
```

Slika 1 Pseudokod genetskog algoritma

Svaka jedinka je predstavljena jednakom podatkovnom strukturom i moguće ju je dekodirati kako bi se ocijenila prihvatljivost rješenja koje nudi ta jedinka. Prihvatljivost rješenja pojedine jedinke ocjenjuje funkcija dobrote. Jedinku nazivamo kromosom, a elemente jedinke genima.

Ideja genetskih algoritama je kreirati početnu populaciju sačinjenu od određenog broja jedinki, tj. kromosoma. Zatim se nad tim jedinkama vrši selekcija, obavlja križanje i mutacija sve dok uvjet zaustavljanja nije zadovoljen.

Cilj selekcije sačuvati i prenijeti dobra svojstva populacije na sljedeću generaciju. Selekcijom se odabiru jedinke koje će sudjelovati u reprodukciji i na taj način sačuvati 'dobar genetski materijal'. Selekcijom osiguravamo veću vjerojatnost preživljavanja i reprodukcije boljim jedinkama, ali je potrebno i određenu vjerojatnost preživljavanja i reprodukcije ostaviti gorim jedinkama kako bi se sačuvali potencijalno dobri dijelovi genetskog materijala koje sadrže gore ocjenjene jedinke.

Proces križanja (engl. crossover) je binarni genetski operator kojim od dvije postojeće jedinke stvaramo jednu novu jedinku koja nasljeđuje svojstva roditelja. Najopćenitije, križanjem nazivamo stvaranje novog kromosoma kod kojeg su pojedini geni preuzeti od jednog roditelja, a ostali od drugog.

Mutacija je genetski operator koji vrši promjenu nad jednim kromosomom. Mutacijom se odabire jedan ili više gena unutar kromosoma nad kojima se vrši promjena. Odabrani gen se zamjenjuje novom vrijednošću iz domene.

Uvjet zaustavljanja definira do kad će se genetski algoritam izvršavat. Uvjet može biti određeno vrijeme izvršavanja, stvaranje određenog broja kromosoma ili generacija, postizanje neke ocjene.

Svim genetskim algoritmima potrebno je definirati parametre u koje ubrajamo veličinu populacije, broj generacija, vjerojatnosti križanja i mutacije. Ovisno o problemu moguće je definirati i dodatna ograničenja u vidu parametara.

2. Problem raspoređivanja poslova na nesrodnim strojevima

2.1. Definicija raspoređivanja

Raspoređivanje je u širem smislu postupak izrade bilo kakvog rasporeda. U različitim okruženjima postupak izrade rasporeda može se podijeliti na više stupnjeva, ovisno o algoritmu i problemu. U problemima koji uključuju samo jedno sredstvo, raspoređivanje se sastoji od određivanja redoslijeda aktivnosti na tom sredstvu. Ako se koristi više odvojenih sredstava, prije izrade redoslijeda potrebno je odrediti koje aktivnosti će se odvijati na kojem sredstvu. U oba slučaja, postupak izrade cjelokupnog rasporeda nazivat će se raspoređivanjem. U problemima s više sredstava, postupak dodjeljivanja aktivnosti sredstvima nazivat će se pridruživanjem (engl. matching), a postupak izrade redoslijeda na pojedinom sredstvu uređivanjem (engl. sequencing). Ovisno o postupku raspoređivanja, te dvije radnje mogu biti odvojene ili isprepletene.[1]

Cilj raspoređivanja je upariti sredstva i radnje u određenom vremenskom periodu. Radnje koje se izvode na sredstvima nazivamo poslovima. Poslovi mogu biti nedjeljivi ili sastojati se od manjih cjelina kao što su zadaci ili operacije. Svaki podao zahtijeva određeno sredstvo u određenoj količini vremena. Sredstva u kontekstu raspoređivanja nazivamo strojevima. Uređeni odnos poslova i strojeva kroz vrijeme naziva se raspored. Kod izrade rasporeda često postoje ograničenja. Primjerice, pojedini posao se mora izvoditi na točno određenom striju, ili prije izvršenja nekog drugog posla. Kod izrade rasporeda moraju se uvažiti sva ograničenja.

Raspoređivanje se može podijeliti na statičko i dinamičko. Kod statičkog raspoređivanja raspored se gradi prije izvršavanja. Kod dinamičkog raspoređivanja raspored se gradi tokom izvršavanja. U ovom radu se koristi statičko raspoređivanje, a sve informacije o strojevima i poslovima su poznate unaprijed.

2.2. Definicija posla

Objekt čije je izvršavanje osigurano postupkom raspoređivanja nazivamo posao (engl. *Job*). Posao može biti nedjeljiva cjelina ili sastavljen od više aktivnosti ili operacija. U ovom radu poslovi će biti predstavljeni kao nedjeljiva cjelina. Skup svih poslova se obično označava sa *J,* a pojedini posao sa J_j . Svakom poslu su pridružena svojstva navedena u nastavku:

- $p_{i\,j}$ - označava trajanje izvođenja određenog posla na određenom stroju. Indeks *i* označava stroj na kojemu se posao izvodi, a indeks *j* označava posao.
- r_j - naziva se vrijeme pripravnosti. Označava vrijeme pojavljivanja posla u sustav, prvi trenutak u kojem je posao moguće izvesti.
- d_j - vrijeme željenog završetka. Trenutak do kojeg očekujemo da će posao biti završen.
- w_j - težina posla. Predstavlja mjeru koja se pridjeljuje određenom poslu kako bi se odredio prioritet izvođenja pojedinog posla u odnosu na drugi. Veća težina predstavlja viši prioritet.
- C_j - definira se kao vrijeme završetka posla *j*.
- F_j - vrijeme provedeno u sustavu. Definira se kao $F_j = C_j - r_j$.
- T_j - vrijeme provedeno u sustavu nakon predviđenog vremena. Definira se kao $T_j = max(C_j - d_j, 0)$.
- U_j - oznaka koja predstavlja je li posao obavljen u predviđenom vremenu. Definira se kao $U_j = \begin{cases} 1, & T_j > 0 \\ 0, & T_j = 0 \end{cases}$.

2.3. Raspoređivanje na nesrodnim strojevima

Okruženje nesrodnih strojeva sastoji se od *n* poslova koje je potrebno rasporediti na *m* strojeva [1]. Svaki posao se moze izvoditi na bilo kojem stroju, ali vrijeme izvršavanja posla ovisi o stroju. Potrebno je definirati $p_{i\,j}$, vrijeme obrade posla *j* na stroju *i,* za svaku kombinaciju posla i stroja. U ovom radu posao se u cijelosti izvodi na jednom stroju.

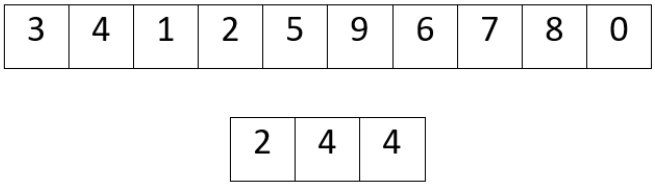
3. Implementacija

Kako bi se implementiralo rješenje problema raspoređivanja u postojeći sustav ECF, namijenjen izradi genetskih algoritama, potrebno je promijeniti inicijalizaciju početne populacije, stvoriti nove genetske operatore križanja i mutacije, prilagoditi evaluaciju i parametre. U ovom poglavlju će se objasniti prikaz jedinke i promjene koje su načinjene kako bi sustav uspješno radio za ovaj problem raspoređivanja na nesrodnim strojevima.

3.1. Prikaz jedinki

Za prikaz jedinki koristio se 'dvojni genotip' sačinjen od permutacijskog niza (engl. *permutation encoding*) i niza prirodnih brojeva (s oznakom 'IntGenotype'). Prikaz permutacijskim nizom se koristi za prikaz poslova koje je potrebno rasporediti u sustavu. Veličina permutacijskog niza odgovara broju poslova, tako ako je u sustavu *n* poslova veličina permutacijskog niza je *n*. Poslovi se određuju jedinstvenim brojem, a taj broj u nizu predstavlja taj posao, dakle permutacijski niz je sačinjen od niza jedinstvenih cijelih brojeva koji odgovaraju indeksu posla kojeg je potrebno rasporediti, a potom i izvršiti. Broj poslova, *n*, sustavu se predaje u parametarskoj datoteci.

Budući da se za prikaz strojeva i poslova koji su im dodijeljeni moraju uvažiti dodatna ograničenja, odabran je niz prirodnih brojeva. Veličina niza odgovara broju strojeva, *m*, a suma brojeva u nizu odgovara broju poslova, *n*. Broj strojeva kao i broj poslova se sustavu se predaje putem argumenata.



Slika 2, izgled jedinke

```
<Genotype>
    <Permutation>
        <Entry key="size">10</Entry>
    </Permutation>
    <IntGenotype>
        <Entry key="size">3</Entry>
        <Entry key="lbound">0</Entry>
        <Entry key="ubound">10</Entry>
    </IntGenotype>
</Genotype>
```

Slika 3, argumenti

Na slikama 2 i 3 dan je konkretan primjer jedinke i potrebni parametri da bi sustav stvorio prikazanu jedinku. U konkretnom primjeru broj potrebno je rasporediti deset poslova (*n=10*) na tri stroja (*m=3*).

Poslovi 3 i 4 se izvode na stroju M_0, poslovi 1, 2, 5 i 9 se izvode na stroju M_1, a poslovi 6, 7, 8 i 0 na stroju M_2, navedenim redoslijedom.

3.2. Inicijalizacija

Prilikom inicijalizacije početne populacije u permutacijskom nizu se raspoređuju brojevi od 0 do *n -1*, a svaki uneseni broj jednoznačno određuje posao.

Prilikom inicijalizacije niza prirodnih brojeva postoji više ograničenja. Veličina niza odgovara broju strojeva na kojima se poslovi izvode, a brojevi unutar niza odgovaraju broju poslova dodijeljenih strojevima. Ukupan zbroj poslova unutar niza mora odgovarati ukupnom broju poslova. Broj strojeva i broj poslova se učitavaju iz parametarske datoteke, a zatim se na slučajan način odabere pozicija od koje će se kružno popunjavati niz. Broj unutar niza, odnosno broj poslova dodijeljen stroju, se odabire kao slučajan broj između nula i ukupnog broja poslova. Nakon odabira broja poslova varijabla u koju je spremljen parametar s ukupnim brojem poslova se umanji za odabrani iznos. Taj postupak se ponavlja sve do zadnjeg elementa niza, odnosno zadnjeg stroja kojemu je potrebno dodijeliti broj poslova. Njemu se dodjeljuje preostali iznos čime je osigurano da ukupna suma elemenata niza odgovara ukupnom broju poslova.

```
intValues.resize(nSize_);
int rand= state->getRandomizer()->getRandomInteger(0, nSize_);

// randomly create each dimension
if (rand != 0) {
    for (uint i = rand; i < nSize_; i++){//cirkularno
        intValues[i] = state->getRandomizer()->getRandomInteger(0, maxValue_);
        maxValue_ -= intValues[i];
    }
    for (uint i = 0; i < rand - 1; i++) {
        intValues[i] = state->getRandomizer()->getRandomInteger(0, maxValue_);
        maxValue_ -= intValues[i];
    }
    intValues[rand - 1] = maxValue_;
}
else
{
    for (uint i = rand; i < nSize_-1; i++) {//cirkularno
        intValues[i] = state->getRandomizer()->getRandomInteger(0, maxValue_);
        maxValue_ -= intValues[i];
    }
    intValues[nSize_ - 1] = maxValue_;
```

Slika 4, dio koda inicijalizacije niza cijelih brojeva

3.3. Genetski operatori

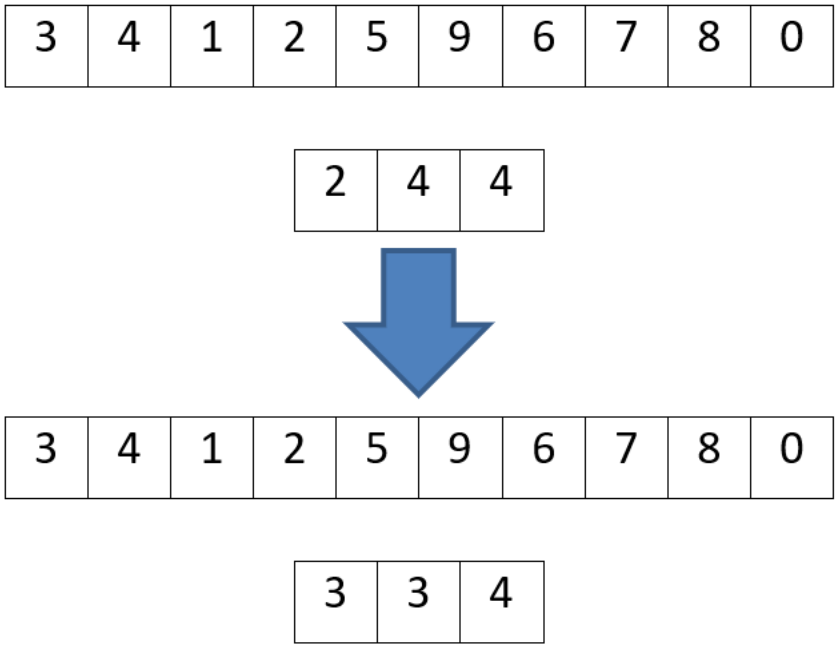
Kako bi se zadovoljila već navedena ograničenja, dodani su posebni genetski operatori križanja i mutacije za niz cijelih brojeva koji stvaraju nove jedinke koje zadovoljavaju te kriterije.

3.3.1. Operatori mutacije

Po uzoru na prirodni postupak mutacije, unutar genotipa na određenom genu ili više njih se događa promjena. Unutar genetskih algoritama potrebno je operatore mutacije prilagoditi problemu i dodijeliti im vjerojatnost. Za ovaj problem stvoreno je deset operatora mutacije, koji se dijele u dvije grupe ovisno o tome smanjuju li element niz (broj poslova na stroju) za 1 ili za slučajan iznos. Nakon odabira stroja kojemu se smanjuje broj poslova potrebno je odabrati stroj kojemu će se za taj isti iznos povećati broj poslova kako bi ukupna suma poslova ostala nepromijenjena. U nastavku će se objasniti svi dodani operatori mutacije.

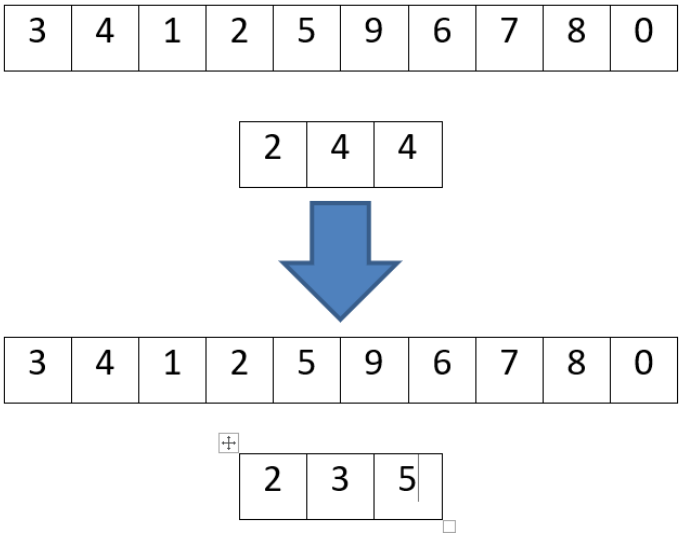
Prva dva dodana operator mutacije su 'leveling' i 'levelingR'. Navedeni operatori obavljaju mutaciju na način da unutar genotipa pronađu stroj koji ima naviše dodijeljenih poslova i stroj koji ima najmanje dodijeljenih poslova. Stroju koji ima najviše poslova se smanji broj poslova, a stroju koji ima najmanje poslova se poveća broj poslova. Operator 'leveling'

mijenja broj poslova za jedan, dok operator 'levelingR' mijenja broj poslova za slučajno odabran broj iz intervala [0, ukupan broj poslova na stroju koji ima najviše poslova].



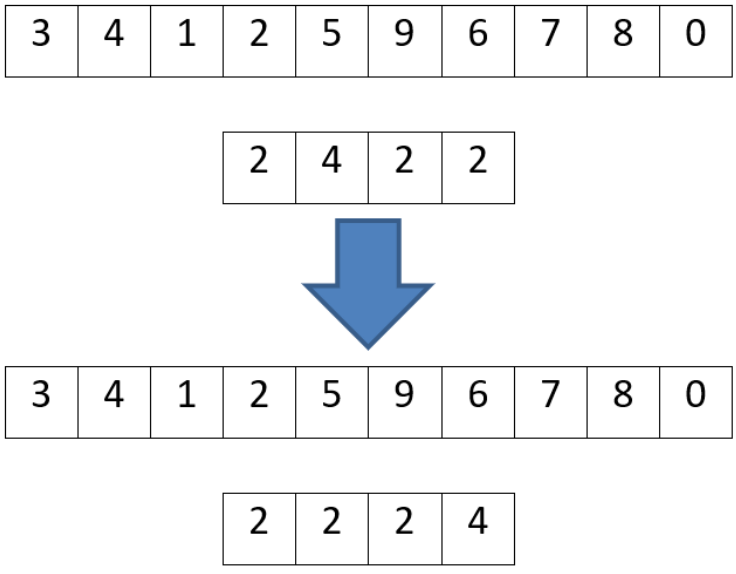
Slika 5, primjer 'leveling' mutacije

Operatori mutacije 'maxR' i 'maxRR' obavljaju mutaciju jedinke tako da pronađu unutar genotipa najveći element, odnosno stroj s najviše pridijeljenih poslova. Zatim tom elementu smanje vrijednost, a razliku dodijele elementu desno od najvećeg elementa. Ukoliko se najveći element nalazi na kraju niza, razlika se dodjeljuje elementu sa lijeve strane. Razlika između operatora 'maxR' i 'maxRR' je u tome što 'maxR' iznos najvećeg elementa smanjuje za jedan, dok operator 'maxRR' mijenja element za slučajno odabran broj iz intervala [0, iznos najvećeg elementa].



Slika 6, primjer 'maxR' mutacije

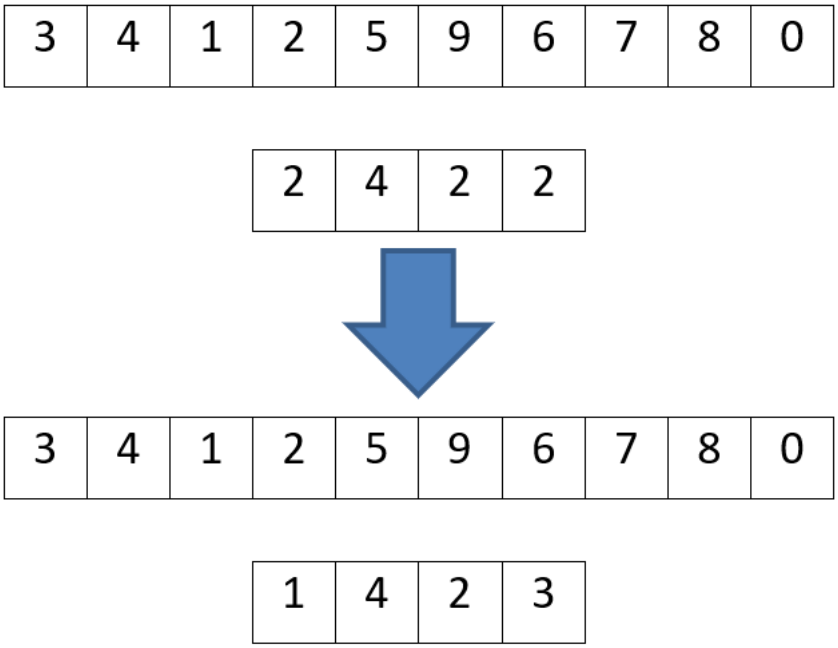
'maxRandom' i 'maxRandomR' operatori mutacije odabiru najveći element niza prirodnih brojeva smanjuju mi iznos a razliku pridjeljuju slučajno odabranom elementu unutar genotipa. Razlia između ta dva operatora je u tome što 'maxRandom' iznos najvećeg element smanji za jedan, dok operator 'maxRandomR' mijenja element za slučajno odabran broj iz intervala [0, iznos najvećeg elementa].



Slika 7, primjer 'maxRandomR' mutacije

Operatori 'randomx2' i 'randomx2R' nasumično odabiru dva različita elementa unutar niza prirodnih brojeva. Jedan element se smanjuje a drugi se uvećava za iznos za koji se smanjio

prvi. Razlia između ta dva operatora je u tome što 'randomx2' iznos odabranog element smanji za jedan, dok operator 'randomx2R' mijenja element za slučajno odabran broj iz intervala [0, iznos odabranog elementa].

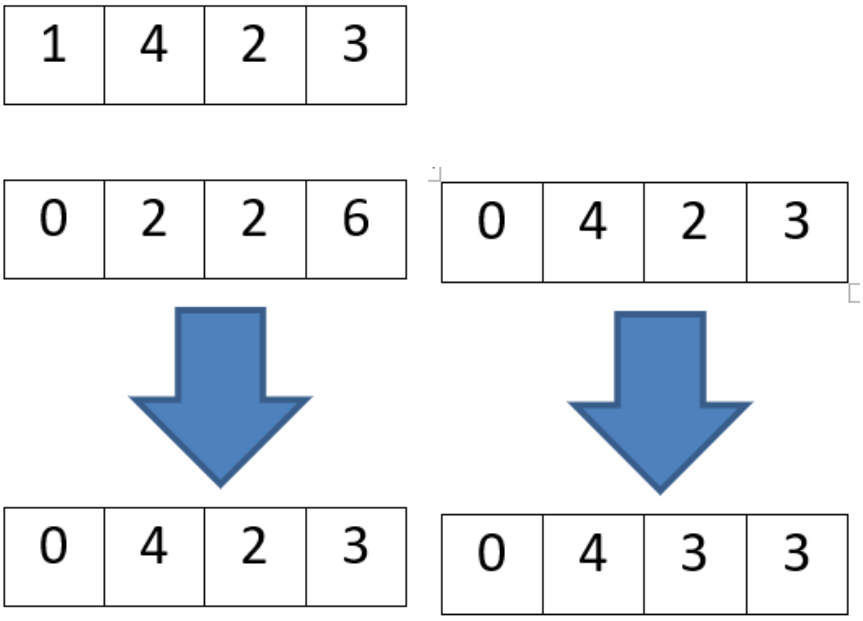


Slika 8, primjer 'randomx2' mutacije

3.3.2. Operatori križanja

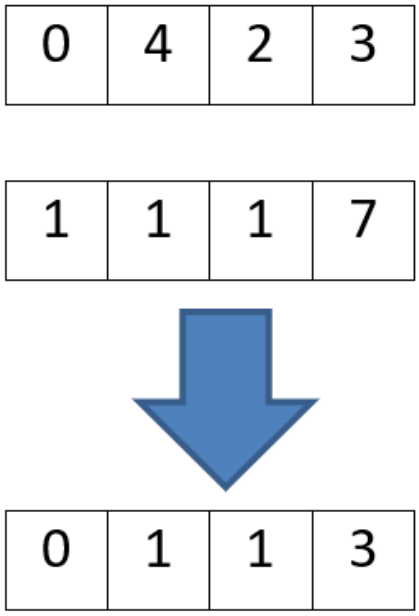
Operatori križanja (engl. crossover) od dvije postojeće jedinke (roditelja) stvaraju novu jedinku (dijete). Zbog ograničenja u ovom konkretnom primjeru operatore križanja niza prirodnih brojeva trebalo je dodatno prilagoditi. Tako se nakon svakog križanja jedinki obavlja korekcija jedinke da bi ukupna suma poslova ostala nepromijenjena. Dodana su dva operatora križanja.

Prvi operator križanja je 'crx1x'. Nakon što se odaberu dvije jedinke za križanje odabire se proizvoljno mjesto unutar genotipa. Jedinka dijete se stvara na način da lijevo od odabranog mjesta dobije gene (elemente) od jedne jedinke roditelja, a desno od odabranog mjesta gene od drugog roditelja. Nakon što se stvori jedinka djeteta provjerava se ukupna suma elemenata unutar djeteta i uspoređuje s ukupnim brojem poslova u sustavu. Sve dok suma ne odgovara ukupnom broju poslova na slučajan se način odabire jedan element polja i uvećava ili smanjuje za jedan ovisno predznaku razlike sume elemenata genotipa djeteta i ukupnog broja poslova.



Slika 9, primjer 'crx1x' operatora Slika 10, korekcija djeteta

Drugi operator križana je 'crx01'. Operator 'crx01' stvara novi genotip djeteta tako da svaki element niza vrši slučajan odabir vrijednosti 0 ili 1. Ukoliko je odabrana vrijednost 0 u odgovarajući element niza se upisuje vrijednost elementa niza prvog roditelja sa jednakim indeksom, a ukoliko je odabrana vrijednost 1 upisuje se vrijednost elementa niza drugog roditelja. Korekcija se vrši na jednak način kao i kod prethodnog operatora križanja.



Slika 11, primjer 'crx01' operatora

4. Rezultati

Kako bi bilo moguće ocijeniti uspješnost implementacije, potrebno je definirati kriterije po kojima će se vrednovati uspješnost implementacije. Kriteriji koji će se koristiti za testiranje su sljedeći:

- *Twt* - težinska suma kašnjenja svih poslova, računa se kao $Twt = \sum_{j=0}^{n} w_j T_j$
- C_{max} - najkasnije vrijeme završetka svih poslova, računa se kao $C_{max} = max_j(C_j)$
- *Ft* – suma svih vremena koje su poslovi proveli u sustavu, računa se kao $Ft = \sum_{j=0}^{n} F_j$
- *Nwt* - težinska suma poslova obavljenih nakon vremena završetka, računa se kao $Uwt = \sum_{j=0}^{n} w_j U_j$

4.1. Prilagodba parametara

Prije konačnog izvršavanja eksperimenta, ispitan je utjecaj pojedinih parametara kako bi se postigli što bolji rezultati. Ispitan je utjecaj veličine populacije i utjecaj vjerojatnosti mutacije. Optimizacija se vršila po *Twt* kriteriju. Ispitivanje se vršilo na uzorku od 60 različitih problema koji sadrže između 12 i 100 poslova, te 3 i 10 strojeva. Za selekciju je korištena eliminacijska turnirska selekcija s veličinom turnira od 3 jedinke, a uvjet zaustavljanja je 1000000 evaluacija. Svaki od 60 problema izvodio se po 30 puta kako bi se dobila što preciznija očitanja. Za svaki od problema iz dobivenih očitanja se računa minimalna vrijednost dobrote, maksimalna vrijednost dobrote, njihov medijan i *Tmin*, što predstavlja sveukupno najbolje rješenje dobiveno kombinacijom rezultata iz svih 30 pokretanja.

Najprije se ispitivao utjecaj veličine populacije na učinkovitost programskog rješenja. Ispitane su veličine populacije od 30, 100, 200, 500, 1000 jedinki. U tablici 1 su prikazani dobiveni rezultati. Budući da je iznos medijana dobrote najmanji za veličinu populacije od 1000 jedinki (10,82), upravo je ta veličina odabrana za daljnja ispitivanja.

Tablica 1, rezultati ovisno o veličini populacije

POPULACIJA	30	100	200	500	1000
TMIN	9,852	**9,710**	9,814	9,627	9,727
MIN	10,831	10,635	10,509	10,473	**10,215**
MED	11,554	11,217	11,230	10,929	**10,820**
MAX	12,319	12,170	12,170	**11,450**	11,487

Dalje se ispitivao utjecaj vjerovatnosti mutacije na dobrotu rješenja. Pokus se provodio za vjerojatnosti mutacije u iznosu od: 30%, 50%, 70% i 90%. U tablici 2 su prikazani rezultati eksperimenta. Najmanja medijan vrijednost dobrote se dobiva u slučaju kad je vjerojatnost mutacije postavljena na 70%. Za daljnja ispitivanja se koristio upravo taj parametar.

Tablica 2, rezultati ovisno o vjerojatnosti mutacije

VJEROJATNOST	30%	50%	70%	90%
TMIN	9,727	9,589	**9,585**	9,630
MIN	10,2115	10,125	**9,902**	9,991
MED	10,820	10,507	**10,240**	10,307
MAX	11,487	10,922	**10,612**	10,797

Dodatno se ispitivao utjecaj pojedinog operatora mutacije i križanja. Pokus se provodio na istom skupu kroblema s najboljim parametrima iz prethodna dva postupka. Prilikom ispitivanja učikovitosti operatora mutacije sustavu je omogućeno da koristi samo taj operator mutacije i oba operatora križanja. Prilikom ispitivanja učinkovitosti operatora križanja sustav je podešen na način da koristi sve dostupne operatore mutacije i samo jedan od operatora križanja. U tablici 3 su prikazane vrijednosti Tmin, minimalna, medijan i maksimalna vrijednost dobrote ovisno o korištenom genetskom parametru u eksperimentu.

Tablica 3, testiranje operatora

OPERATOR	TMIN	MIN	MED	MAX
CRX1X	9,586	9,901	10,245	10,513
CRX01	9,558	9,934	**10,200**	10,607
RANDOMX2	9,957	10,001	10,279	10,839
RANDOMX2R	9,569	10,013	10,325	10,852
LEVELING	9,965	10,383	**10,746**	11,449
LEVELINGR	9,716	10,304	10,716	11,179
MAXL	9,631	10,116	10,502	11,007
MAXLR	9,544	10,175	10,529	10,936
MAXR	9,640	10,258	10,551	10,940
MAXRR	9,700	10,140	10,494	11,084
MAXRANDOM	9,556	9,957	10,307	10,617
MAXRANDOMR	9,615	10,066	10,381	10,936

Iz tablice vidimo da se najbolja medijan vrijednosti dobrote postiže prilikom korištenja Crx1x operatora križanja, uz sve operatore mutacije, ali najmanja i najveća vrijednost postignute dobrote prilikom korištenja samo tog operatora, pokazuju gore vrijednosti u odnosu na operator Crx1x. Zbog toga u daljnjim ispitivanjima koristili su se svi dostupni operatori.

4.2. Rezultati eksperimenta i usporedba s postojećim prikazima

Nakon provedene optimizacije parametara eksperiment je pokrenut kako bi se provela optimizacija po prethodno opisanim *Ft, Twt, Nwt i* C_{max} kriterijima. Kriteriji su optimizirani samostalno na prethodno opisanom nizu od 60 problema, a za svaki problem optimizacija se

vršila 30 puta. Uvjet zaustavljanja je postavljen na 1000000 evaluacija i korištena je eliminacijska turnirska selekcija. Veličina populacije je postavljena na 1000 jedinki, vjerojatnost mutacije na 70% i korišteni su svi genetski operatori križanja i mutacije. Medijan vrijednost dobrote za *twt* kriterij optimizacije iznosi 10,240. Medijan vrijednost za C_{max} kriterij optimizacije iznosi 37,624, za *Ft* kriterij iznosi 165,721 i za *Nwt* kriterij iznosi 5,480.

Tmin vrijednost dobrote, minimalna vrijednost dobrote, maksimalna vrijednost dobrote i medijan vrijednosti dobrote za sve kriterije optimizacije prikazani su u tablici 3. Dodatno je prikazano prosječno vrijeme izvođenja programskog rješenja.

Tablica 4, rezultati optimizacije

KRITERIJ	TWT	C_{MAX}	FT	NWT
TMIN	9,585	36,642	150,744	5,364
MIN	9,902	37,474	163,659	5,429
MED	**10,240**	**37,624**	**165,721**	**5,480**
MAX	10,612	37,917	168,924	5,694
AVG TIME	485,66	486,06	473,04	484,06

U tablicama 5, 6, 7 i 8 su prikazana rješenja dobivena već postojećim prikazima jedinke. Implementiranom prikazu je dodijeljena oznaka PERMINT. Postojeći prikazi su se testirali na jednakom skupu problema kako bi rezultati bili usporedivi sa prethodno prikazanim rezultatima eksperimenta. Korištenjem opisanog prikaza uz *Twt* optimizacijski kriteriji postižu se 6,84% gora vrijednost medijana dobrote u odnosu na najbolji prikaz jedinke, a 5,1% bolji iznos u odnosu na najgore rješenje. Uz C_{max} optimizacijski kriterij postiže se 2,1% gore rješenje u odnosu na najbolje, a 2,3% bolje rješenje u odnosu na najgore. Uz *Ft* optimizacijski kriterij postiže se 17,2% gore rješenje u odnosu na najbolje i 1% bolje rješenje u odnosu na najlošije. Uz korištenje *Nwt* optimizacijskog kriterija postiže se 2,7% gore rješenje od najboljeg i 8,3% bolje rješenje od najgoreg.

Tablica 5, iznos dobrote uz *Twt* optimizacijski kriterij

	PE	PEML	RKE	FPE	ME	MLE	PERMINT
TMIN	**9,452**	9,725	9,516	9,533	9,525	9,499	**9,585**
MIN	**9,521**	10,34	9,723	9,917	9,709	9,601	**9,902**
MED	**9,584**	10,76	9,968	10,27	9,987	9,846	**10,240**
MAX	**9,695**	11,35	10,39	10,90	10,47	10,18	**10,612**

Tablica 6, iznos dobrote uz Cmax optimizacijski kriterij

	PE	PEML	RKE	FPE	ME	MLE	PERMINT
TMIN	36,59	37,14	36,55	36,79	**36,50**	36,50	**36,642**
MIN	**36,74**	38,22	36,91	37,48	36,76	36,76	**37,474**
MED	**36,85**	38,50	37,07	37,72	36,91	36,91	**37,624**
MAX	**36,99**	37,78	37,21	38,12	37,08	37,08	**37,917**

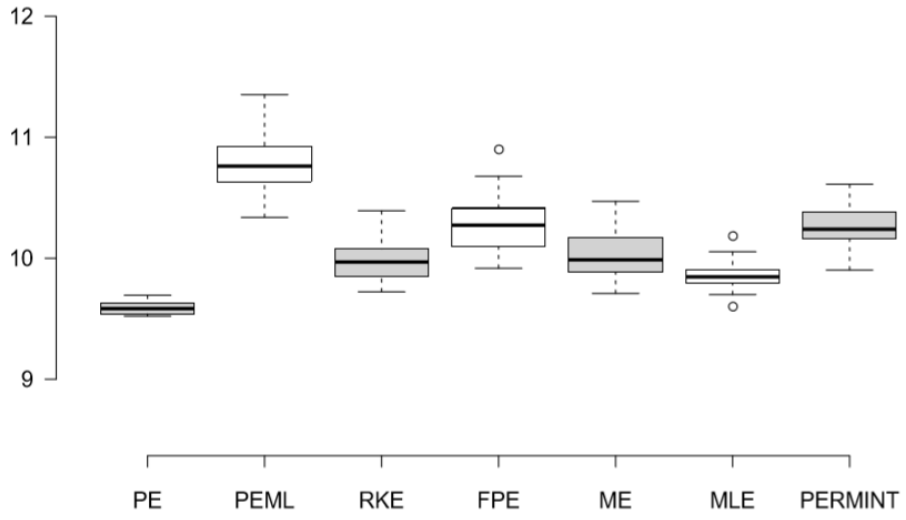
Tablica 7, iznos dobrote uz Ft optimizacijski kriterij

	PE	PEML	RKE	FPE	ME	MLE	PERMINT
TMIN	**138,9**	151,2	139,7	140,8	141,0	139,7	**150,744**
MIN	**140,3**	163,8	144,0	146,4	145,9	144,1	**163,659**
MED	**141,4**	167,3	146,9	149,9	149,2	146,3	**165,721**
MAX	**143,1**	170,5	149,2	153,1	151,8	147,8	**168,924**

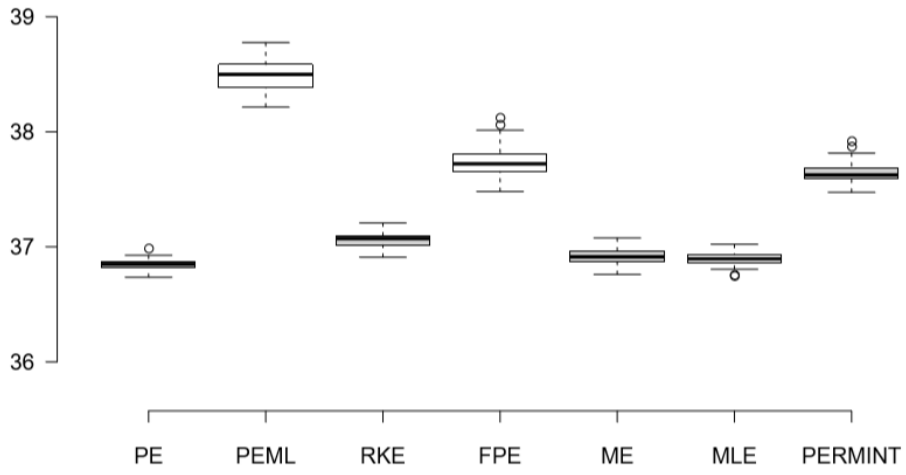
Tablica 8, iznos dobrote uz Nwt optimizacijski kriterij

	PE	PEML	RKE	FPE	ME	MLE	PERMINT
TMIN	**5,316**	5,615	5,5,334	5,340	5,317	5,318	**5,364**
MIN	**5,316**	5,829	5,335	5,373	5,357	5,323	**5,429**
MED	**5,333**	5,938	5,356	5,455	5,403	5,348	**5,480**
MAX	**5,377**	6,063	5,398	5,661	5,592	5,466	**5,694**

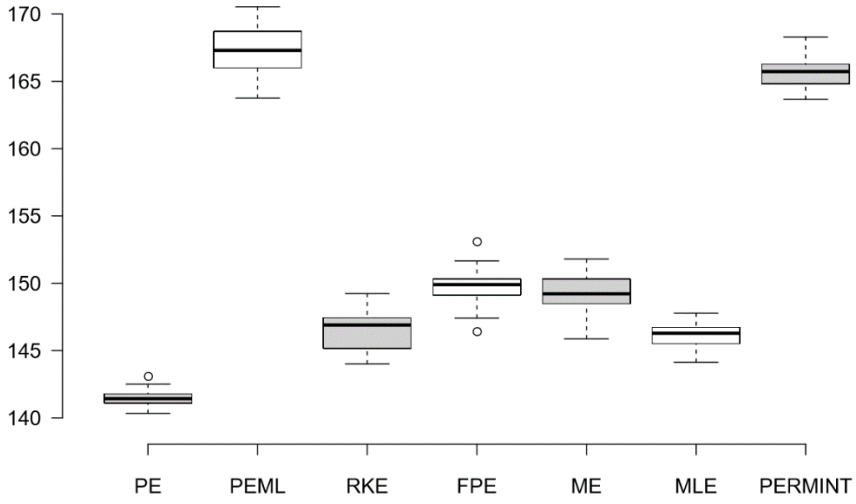
U svrhu detaljnije usporedbe, na slikama 12, 13, 14, 15 prikazani su boxplotovi rezultata optimizacije po svim navedenim kriterijima i sa oblicima izgleda jedinke.



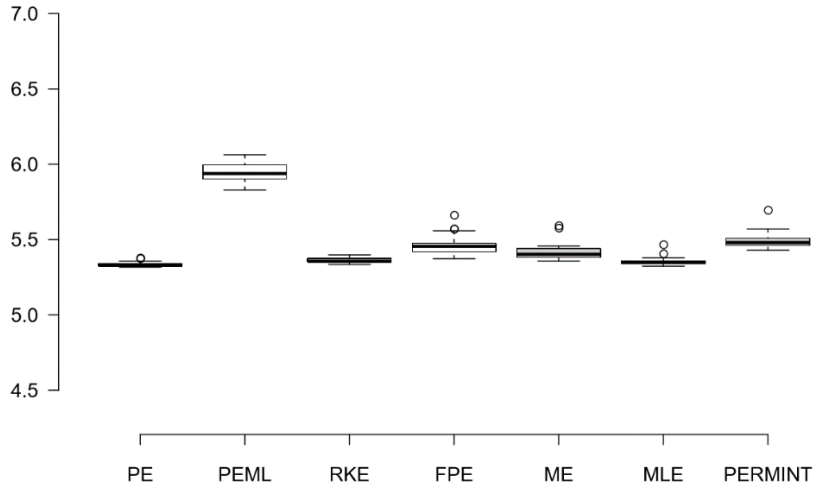
Slika 12, rezultati uz korištenje Twt optimizacijskog kriterija



Slika 13, rezultati uz korištenje Cmax optimizacijskog kriterija

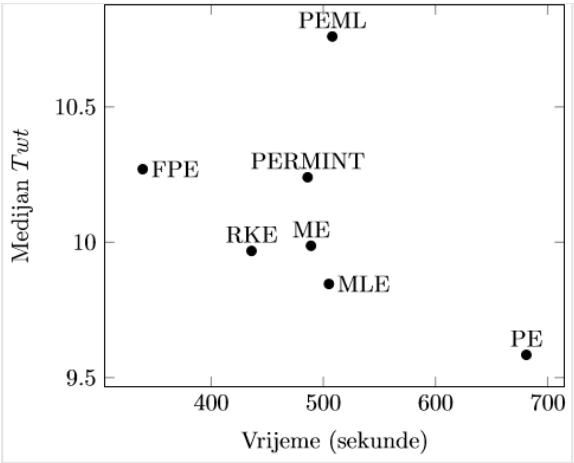


Slika 14, rezultati uz korištenje Ft optimizacijskog kriterija

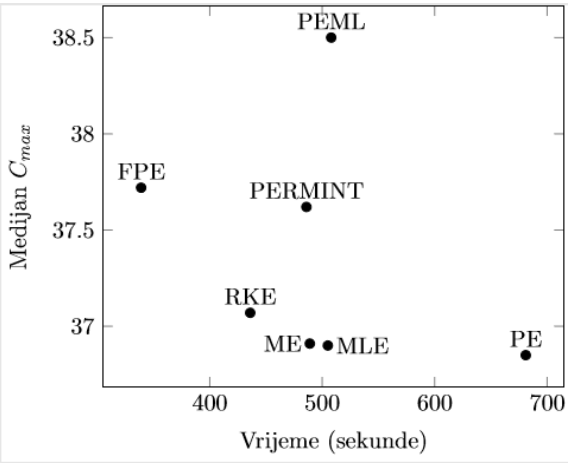


Slika 15, rezultati uz korištenje Nwt optimizacijskog kriterija

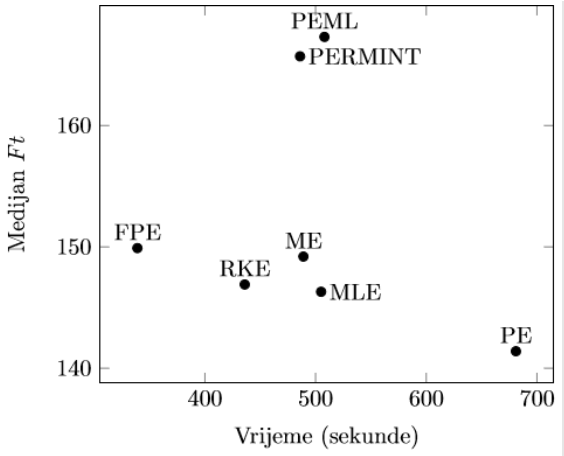
Ukoliko se kao kriterij uzme u obzir vrijeme izvođenja, implementirani prikaz postiže 42,2% gore vrijeme izvođenja od najbržeg i 41,2% bolje vrijeme izvođenja od najsporijeg. Iako PE prikaz postiže najbolje rezultate prilikom optimizacije, taj prikaz ima najdulje vrijeme izvođenja. Ostali prikazi imaju različit omjer kvalitete rješenja i vremena izvođenja. U tablici 9 prikazano je vrijeme izvođenja za postojeće implementacije, a na slikama 16, 17, 18 i 19 odnos medijana dobrote i vremena izvođenja za svaki optimizacijski kriterij. Za FPE i RKE prikaze potrebno je manje vrijeme izvođenja. Ostali prikazi imaju dulje vrijeme izvođenja u odnosu na PERMINT prikaz.



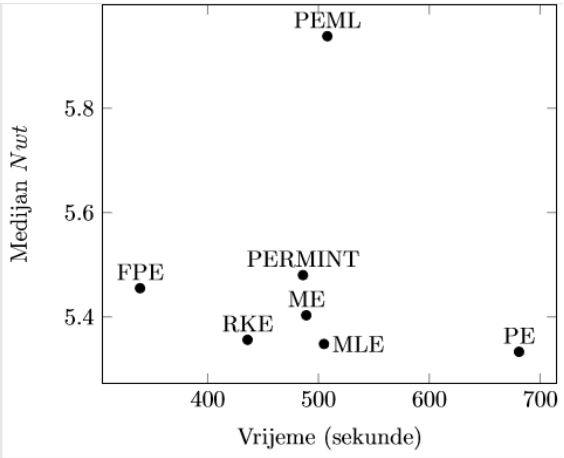
Slika 16, Twt i vrijeme izvođenja



Slika 17, Cmax i vrijeme izvođenja



Slika 18, Ft i vrijeme izvođenja



Slika 19, Nwt i vrijeme izvođenja

Tablica 9, vrijeme izvođenja u sekundama

	PE	**PEML**	**RKE**	**FPE**	**ME**	**MLE**	**PERMINT**
VRIJEME	681,0	508,1	436,8	339,1	489,5	505,4	**482,2**

Zaključak

Cilj ovog rada bio je istražiti problem raspoređivanja u okruženju nesrodnih strojeva korištenjem genetskih algoritama, implementirati novi način prikaza jedinke, dodati nove genetske operatore mutacije i križanja, ispitati učinkovitost programskog rješenja na problem raspoređivanja i usporediti ga sa već postojećim rješenjima.

Nakon provedenog ispitivanja na nizu od 60 različitih problema i optimizaciji po *Twt*, C_{max}, *Ft i Nwt* kriterijima, implementirano rješenje daje usporedive rezultate sa već postojećim rješenjima. Iako PE prikaz postiže najbolje rezultate prilikom optimizacije, vrijeme izvođenja kod tog prikaza je značajno dulje u odnosu na ostale. FPE prikaz postiže najbolje vrijeme izvođenja ali ne najbolju vrijednost dobrote. Ukoliko se u obzir uzme omjer vremena izvođenja i vrijednosti dobrote implementirani PERMINT prikaz postiže dobre rezultate. U usporedbi sa PEML prikazom, koji je također 'dvojni genotip', PERMINT se po svakom kriteriju testiranja pokazuje kao bolje rješenje.

Literatura

[1] JAKOBOVIĆ, D., Raspoređivanje zasnovano na prilagodljivim pravilima, doktorska disertacija, Fakultet elektrotehnike i računarstva, 2005.

[2] ĐURASEVIĆ, JAKOBOVIĆ, KNEŽEVIĆ, Adaptive scheduling on unrelated machines with genetic programming. Sveučilište u Zagrebu, 2015.

[3] PINEDO, M.L. Scheduling Theory, Algorithms, and Systems, New York University, 2012.

Sažetak

Ocjena učinkovitosti prikaza rješenja u raspoređivanju na nesrodnim strojevima

Zbog složenosti problema raspoređivanja ne postoji efikasan algoritam koji pronalazi optimalno rješenje unutar razumnog vremenskog ograničenja. Kako bi se taj problem riješio često se koriste genetski algoritmi. U radu je opisano okruženje nesrodnih strojeva i prikazano rješenje problema raspoređivanja u takvoj okolini korištenjem genetskih algoritama. Kako bi se došlo do rješenja bilo je potrebno prikazati jedinku, prilagoditi inicijalizaciju populacije i stvoriti nove genetske operatore križanja i mutacije. Nakon implementacije, novi prikaz je testiran i uspoređen sa već postojećim programskim rješenjima.

Ključne riječi: raspoređivanje, okruženje nesrodnih strojeva, genetski algoritmi, genetski operatori, ECF.

Summary

Evaluating the efficiency of solution representations for unrelated machines scheduling

Because of the complexity of scheduling problems, there is no efficient algorithm that finds an optimal solution within a reasonable time limit. Genetic algorithms are often used to solve this problem. The paper describes the environment of unrelated machines and shows the solution of the problem of scheduling in such environment using genetic algorithms. In order to reach the solution, it was necessary to create solution representation, adapt the initialization of the population and create new genetic crossover and mutation operators. After the implementation, the new solution representation was tested and compared with the existing software solutions.

Keywords: scheduling, unrelated machines environment, genetic algorithms, genetic operators, ECF.