SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Prikazi rješenja i operatori lokalnog pretraživanja u genetskim algoritmima za problem raspoređivanja u okruženju nesrodnih strojeva

Hrvoje Backović
Voditelj: *izv. prof. dr. sc. Domagoj Jakobović*

Zagreb, svibanj 2016.

SADRŽAJ

1. Uvod **1**

2. Raspoređivanje u okruženju nesrodnih strojeva **3**

2.1. Kriteriji raspoređivanja 3

2.2. Pristupi rješavanju problema raspoređivanja 5

3. Genetski algoritmi za probleme raspoređivanja **6**

3.1. Prikazi rješenja 7

3.1.1. Permutacijski prikaz s dva niza 7

3.1.2. Prikaz s nizom decimalnih brojeva 8

3.1.3. Permutacijski prikaz s jednim nizom 9

3.2. Lokalni operatori pretraživanja 9

3.2.1. Zamjena dva posla na istom stroju 10

3.2.2. Zamjena dva posla na različitim strojevima 11

3.2.3. Prebacivanje posla s jednog stroja na drugi 11

4. Rezultati i analiza **12**

4.1. Okruženje za testiranje i implementacija 12

4.2. Postavljanje parametara 13

4.3. Usporedba rezultata 13

5. Zaključak **17**

6. Literatura **18**

7. Sažetak **19**

1. Uvod

Raspoređivanje se može definirati kao optimizacijski problem gdje je skup poslova potrebno rasporediti na dostupni skup resursa, pri čemu se minimiziraju određeni kriteriji (npr. kriterij da svi poslovi budu što prije gotovi) [2] [3]. Problem se može dodatno zakomplicirati uvođenjem raznih svojstava i ograničenja sustava (npr. ograničenje u redoslijedu izvođenja poslova) čime se problem raspoređivanja dodatno otežava. Većina poznatih problema raspoređivanja spadaju u NP-teške probleme, što znači da ne postoje efikasni algoritmi za njihovo izračunavanje. S obzirom na tu činjenicu, u praksi se vrlo često koriste razne heuristike koje pronalaze rješenje koje nije optimalno, ali je prihvatljivo. Te heuristike se prema načinu na koji pronalaze rješenje mogu podijeliti u one koje pretražuju prostor rješenja i one koje iterativno grade rješenje. U ovom radu fokus je stavljen na genetske algoritme kao predstavnike heuristika koje pretražuju prostor rješenja, ali se pri testiranju uspoređuju rezultati i sa heuristikama koje iterativno grade rješenje.

Kod pretraživanja prostora rješenja vrlo je bitno odabrati dobar prikaz rješenja jer je pokazano da prikaz jako utječe na ukupnu uspješnost samog algoritma [1]. Iz tog razloga u ovom radu opisana su dva postojeća prikaza rješenja i uveden je jedan novi prikaz s ciljem postizanja još boljih rezultata. Od postojećih prikaza korišteni su permutacijski prikaz koji koristi dva niza i prikaz sa nizom decimalnih brojeva, a uvodimo novi prikaz koji će koristiti samo jedan permutacijski niz.

Osim prikaza, vrlo su bitni i operatori kojima se pretražuje prostor rješenja, odnosno operatori koji iz postojećih rješenja daju nova, potencijalno još neotkrivena rješenja. Kod genetskih algoritama postoje već definirani brojni operatori križanja i mutacije. Oni su dobri, ali su vrlo općeniti i nisu prilagođeni za konkretni problem koji se optimira. Iz tog razloga definiramo lokalne operatore pretraživanja koji će lokalnim promjenama nad rješenjem dati novo rješenje koje je slično i potencijalno bolje te time osigurati veću stabilnost algoritma. U ovom radu isprobana su tri lokalna operatora pretraživanja: zamjena dva posla na istom stroju, zamjena dva posla na različitim stro-

jevima te prebacivanje posla s jednog stroja na drugi.

Rad je organiziran na sljedeći način: prvo se detaljno opisuje problem raspoređivanja u okruženju nesrodnih strojeva nakon čega se opisuje pristup rješavanju tog problema koji koristi genetske algoritme. Pri tome se opisuju navedeni prikazi rješenja i lokalni operatori pretraživanja. Nakon toga se opisuje okruženje u kojem se provodilo testiranje te su izneseni i analizirani rezultati. Na samom kraju se daje kratak zaključak koji predlaže i moguća poboljšanja ovog pristupa.

2. Raspoređivanje u okruženju nesrodnih strojeva

Problem raspoređivanja u okruženju nesrodnih strojeva definiran je s n poslova i m strojeva, pri čemu je za svaki posao potrebno odrediti na koji će se stroj rasporediti, kao i redoslijed izvođenja poslova na svakom od strojeva [2]. Svakom poslu dodijeljen je indeks iz intervala $[0, n-1]$, dok je svakom stroju dodijeljen indeks iz intervala $[0, m-1]$. Prilikom raspoređivanja na raspolaganju su svojstva specifična za sam problem i okruženje. Ta svojstva su nabrojana i opisana u tablici 2.1. Kada se određuje raspored poslova po strojevima, s obzirom na navedena svojstva, optimizira se jedan od kriterija raspoređivanja opisanih u nastavku.

Oznaka	Opis
p_{ij}	vrijeme izvođenja posla j na stroju i
r_j	vremenski trenutak kada posao j dolazi u sustav
d_j	rok, vremenski trenutak kada posao j treba biti gotov
w_j	težina, važnost posla

Tablica 2.1: Svojstva problema raspoređivanja

2.1. Kriteriji raspoređivanja

Kvalitetu nekog rasporeda možemo definirati preko brojnih kriterija raspoređivanja. Navedeni kriteriji se zatim mogu koristiti kao funkcija dobrote (engl. *fitness function*) u genetskom algoritmu i po njima se može optimirati rješenje. Kako bi se lakše definirali kriteriji za čitav raspored, u tablici 2.2 su definirani kriteriji za pojedinačne poslove, koji se koriste pri definiranju kriterija za cjelokupni raspored [3]. U tablici 2.3 navedena su 4 glavna kriterija koji će se optimirati u ovom radu.

Oznaka	Opis
C_j	vremenski trenutak kada posao j završi s izvođenjem
F_j	tok, vrijeme koje je posao j proveo u sustavu. Definira se kao $F_j = C_j - r_j$
T_j	zakašnjelost, vrijeme koliko se posao j izvršavao nakon roka $T_j = max(C_j - d_j, 0)$
U_j	je li posao j zakasnio, $U_j = \begin{cases} 1 & \text{if } T_j > 0 \\ 0 & \text{if } T_j = 0 \end{cases}$

Tablica 2.2: Kriteriji za pojedinačne poslove

Oznaka	Opis
$Cmax$	maksimalno vrijeme završetka izvođenja. $Cmax = max(Cj)$
Ft	ukupni tok. $Ft = \sum_{j=1}^{n} F_j$
Twt	ukupna težinska zakašnjelost. $Twt = \sum_{j=1}^{n} w_j T_j$
Uwt	težinski broj zakašnjelih poslova. $Uwt = \sum_{j=1}^{n} w_j U_j$

Tablica 2.3: Kriteriji po kojima se provodi optimizacija

2.2. Pristupi rješavanju problema raspoređivanja

Problem raspoređivanja može se podijeliti u nekoliko klasa s obzirom na pojedina svojstva problema. Na primjer, ako su sve informacije o problemu poznate odmah na početku izvođenja, onda se takav problem naziva *offline* raspoređivanje. S druge strane, problem se može zadati na način da se informacije o pojedinom poslu otkrivaju tek kada taj posao uđe u sustav. U tom slučaju se taj problem naziva *online* raspoređivanje. Nadalje, problem raspoređivanja možemo podijeliti i po načinu na koji se konstruira rješenje. Ako se cjelokupno rješenje konstruira prije početka izvođenja sustava, onda se to naziva statičko raspoređivanje. Kada se rješenje konstruira paralelno sa izvođenjem sustava, tada se to naziva dinamičko raspoređivanje.

U ovom radu koncentriramo se na *offline* statičko raspoređivanje. Osim korištenja genetskih algoritama, postoje heuristike za rješavanje problema raspoređivanja koje inkrementalno grade rješenje problema. Svaki put kada neki stroj postane slobodan, te heuristike donose odluku o tome koji posao će rasporediti na njega. Razlika između pojedinih heuristika je samo u tome na koji način donose tu odluku. U ovom radu se ćemo se osvrnuti na rezultate koje daju min-min, max-min, sufferage i min-max heuristike [1].

3. Genetski algoritmi za probleme raspoređivanja

Genetski algoritam je stohastički optimizacijski postupak koji se može primijeniti na razne optimizacijske probleme te se u današnje vrijeme intenzivno koristi [4]. Algoritam simulira način na koji funkcionira evolucija u prirodi kako bi pronašao što bolje rješenje nekog problema. Svako potencijalno rješenje problema predstavi se kao jedinka u populaciji, pri čemu svakoj jedinci pridijelimo realan broj koji odražava kvalitetu jedinke, odnosno potencijalnog rješenja. Funkcija koja obavlja to pridijeljivanje se naziva funkcija dobrote. Za funkciju dobrote se kod problema raspoređivanja koriste kriteriji koji su navedeni u prethodnom poglavlju. Algoritam počinje s početnom populacijom slučajnih rješenja koja zatim prolazi kroz generacije gdje bolja rješenja imaju veću šansu da napreduju, dok slabija rješenja otpadaju. Kod svake generacije se nad pojedinim jedinkama provodi selekcija, križanje i mutacija.

Za selekciju se u ovom radu koristi k-turnirska selekcija, pri čemu se koristi vrijednost parametra $k = 3$. Pseudokod k-turnirske selekcije dan je na slici 3.1. Križanje i mutacija su operatori koji omogućuju pretraživanje prostora rješenja. Križanje kombinira dvije (roditeljske) jedinke u novu jedinku koja sadrži svojstva oba roditelja i potencijalno je veće dobrote. Mutacija provodi slučajne promjene nad nekom jedinkom te joj pritom mijenja i dobrotu. Za križanje i mutaciju koristimo nekoliko različitih operatora gdje se svaki put slučajno odabire jedan od njih koji će biti proveden.

U nastavku opisujemo nekoliko načina pomoću kojih je moguće prikazati rješenje problema raspoređivanja kao jedinku u populaciji. Također uvodimo i pojam lokalnih operatora pretraživanja koji uz križanje i mutaciju pretražuju prostor rješenja, ali s promjenama specifičnima za problem koji se rješava.

```
k-turnirska selekcija
{
        izaberi slučajno k jedinki;
        ukloni najgoreg od k izabranih;
        dijete = križaj(najbolja dva od k izabranih);
        s određenom vjerojatnošću mutiraj dijete;
        ubaci dijete u populaciju;
}
```

Slika 3.1: k-turnirska selekcija - pseudokod

3.1. Prikazi rješenja

Kako bi uspješno primijenili genetski algoritam na problemu raspoređivanja, potrebno je definirati kvalitetan prikaz rješenja koji će služiti kao jedinka u populaciji u genetskom algoritmu. Taj prikaz također mora biti povoljan i prilagođen postojećim operatorima križanja i mutacije. Ovdje ćemo spomenuti i opisati dva postojeća prikaza rješenja za problem raspoređivanja te ćemo predložiti jedan novi prikaz. Postojeći prikazi će poslužiti za usporedbu uz pomoć koje ćemo moći ocijeniti kvalitetu novog prikaza. Uz sam opis prikaza, osvrnut ćemo se i na njihove prednosti i nedostatke.

3.1.1. Permutacijski prikaz s dva niza

Proučavajući raznu literaturu možemo uočiti kako se za problem raspoređivanja često koristi permutacijski prikaz. Navedeni prikaz često se koristi pri rješavanju raznih kombinatoričkih problema zbog svoje jednostavnosti i intuitivnosti. Iako postoje brojne varijante permutacijskog prikaza za problem raspoređivanja, u ovom radu odlučili smo se koristiti permutacijski prikaz koji je predložen u [1].

Navedeni prikaz sastoji se od dva niza, pri čemu oba niza imaju veličinu koja odgovara broju poslova u trenutnoj instanci problema. Prvi niz predstavlja permutaciju indeksa poslova gdje će se indeks svakog posla pojaviti točno jednom u tom nizu. Taj niz određuje kojim redoslijedom će se poslovi raspoređivati. Kako bi znali točno na koji stroj će se određeni posao rasporediti, potrebno je uvesti dodatni niz prirodnih brojeva gdje svaki broj označava indeks stroja na kojem će se neki posao izvršiti. Na slici 3.2 možemo vidjeti primjer jednog takvog prikaza za problem raspoređivanja 6 poslova na 3 stroja. Iz njega možemo vidjeti da će se poslovi s indeksima 1 i 3 izvršavati (tim

redoslijedom) na stroju s indeksom 0, poslovi s indeksima 5 i 2 će se izvršavati na stroju s indeksom 1, dok će se na stroju s indeksom 2 izvršavati poslovi s indeksima 4 i 0.

4	5	2	1	0	3
2	1	1	0	2	0

Slika 3.2: Permutacijski prikaz s dva niza

Primijetimo kako u prikazu nije bitan poredak između poslova koje raspoređujemo na različite strojeve jer se oni neovisno jedan o drugome raspoređuju te tako može postojati više prikaza koji su ekvivalentni, odnosno predstavljaju ekvivalentna rješenja problema. Prednost ovog prikaza je u tome što je on intuitivan te iz njega možemo vrlo jednostavno i efikasno izvući informaciju o rješenju. S druge strane, on je memorijski zahtjevniji i ima dva niza različite vrste što može biti nezgodno zbog toga što je potrebno primijenjivati više različitih operatora križanja i mutacije.

3.1.2. Prikaz s nizom decimalnih brojeva

Prikaz s nizom decimalnih brojeva se u praksi vrlo često koristi u genetskim algoritmima te za njega postoje brojni poznati operatori križanja i mutacije. Prikaz se sastoji od niza decimalnih brojeva iz intervala $[0, 1]$ te veličina niza odgovara broju poslova u instanci problema. Poslu s indeksom i će biti pridružen decimalni broj s indeksom i u nizu pa će tako na primjer poslu s indeksom 2 biti pridružen decimalni broj na indeksu 2. Svaki decimalni broj određuje stroj na koji ćemo neki posao rasporediti te njegov prioritet na tom stroju. Ako interval $[0, 1]$ podijelimo na jednake podintervale kojih ima jednako koliko i strojeva u instanci problema, posao ćemo rasporediti na onaj stroj u čiji podinterval njegov decimalni broj upada pri čemu će posao s manjim brojem ići prije nekog drugog posla s većim brojem koji također upada u isti podinterval (na isti stroj). Na slici 3.3 možemo vidjeti prikaz s nizom decimalnih brojeva koji je ekvivalentan permutacijskom prikazu sa slike 3.2.

0.85	0.21	0.57	0.26	0.73	0.42

Slika 3.3: Prikaz s nizom decimalnih brojeva

Primijetimo kako i kod ovog prikaza mogu postojati različiti prikazi koji predstavljaju ekvivalentna rješenja. Ovaj prikaz je manji od permutacijskog s dva niza, ali za dekodiranje rješenja je potrebno nešto više vremena.

3.1.3. Permutacijski prikaz s jednim nizom

U pokušaju da smanjimo prethodni permutacijski prikaz s dva niza, uklonit ćemo drugi niz koji je određivao stroj na koji će se pojedini posao rasporediti. Permutacijski niz će nam određivati redoslijed raspoređivanja poslova, no da bi mogli odrediti na koji stroj ćemo rasporediti koji posao, uvodimo pohlepnu heuristiku koja će svaki posao rasporediti na stroj na kojemu će najprije završiti. Ova heuristika će stoga koristiti svojstva specifična za problem kao što su vrijeme dolaska posla u sustav, vrijeme izvođenja posla na određenom stroju i sl. Kada heuristika određuje za posao hoće li ga rasporediti na neki stroj, ona uzima u obzir vremenski trenutak kada je zadnji posao na tom stroju završio s radom, vrijeme čekanja da posao dođe u sustav, ako još nije stigao te vrijeme izvršavanja posla na tom stroju. Ako će posao završiti najranije na tom stroju u odnosu na ostale strojeve, on će biti raspoređen na njega. Ova heuristika ima složenost $O(nm)$ gdje je n broj poslova, a m broj strojeva.

Primijetimo da ovim prikazom nije moguće predstaviti sva moguća rješenja za neki problem raspoređivanja, već samo jedan određeni podskup. Iz tog razloga postoji mogućnost da se optimalno rješenje uopće ne može prikazati u ovom prikazu, ali ipak se nadamo da će se u smanjenom prostoru pretraživanja uspjeti doći do boljih rješenja od onih dobivenih s prethodna dva prikaza.

Iako je ovaj prikaz vremenski složeniji od ostalih te koristimo heuristiku koja je pohlepna, pokazat će se na rezultatima da se unatoč tome dobivaju rezultati koji nadmašuju dosadašnja dva prikaza.

3.2. Lokalni operatori pretraživanja

Operatori križanja i mutacije koje koristimo za navedene prikaze rješenja su vrlo općeniti i mogu se primijeniti na bilo koji drugi problem koji koristi sličan prikaz. Ta općenitost dolazi od toga što ti operatori promatraju rješenje na razini niza bitova ili brojeva te provode promjene bez da imaju ikakav uvid u problematiku koja se skriva iza tog rješenja. To može biti nepraktično u nekim slučajevima gdje promjene ponekad mogu biti besmislene na razini problema kojeg rješavamo. Zbog toga uvodimo lokalne operatore pretraživanja koji će nad jedinkom provoditi promjene koje su prilagođene problemu raspoređivanja.

Lokalni operatori pretraživanja su u literaturi dosta poznati te već postoji nekoliko tak-

vih operatora za problem raspoređivanja [6]. U ovom radu isprobat ćemo tri različita lokalna operatora: zamjena redoslijeda dva posla koji se nalaze na istom stroju, zamjena dva posla koji se nalaze na različitim strojevima te prebacivanje posla s jednog stroja na drugi. Primijetimo da operator ne mora nužno dati bolje rješenje. Moguće je oblikovati operatore tako da provjere sve moguće zamjene i onda uzmu najbolju od njih, ali to iziskuje relativno veliku vremensku složenost. Ovdje smo se odlučili da operatori provode slučajne zamjene i to ponavljaju sve dok dobivamo bolju jedinku. Navedene operatore primijenjujemo na postojećim prikazima u nadi da ćemo uz njihovu pomoć dobiti još bolje rezultate.

Iz razloga što se kod permutacijskog prikaza s jednim nizom raspored određuje uz pomoć heuristike, primjena ovih operatora na tom prikazu je vrlo nepraktična te u ovom radu to neće biti razmatrano, već će se navedeni operatori isprobati na preostala dva prikaza opisana u radu. U nastavku opisujemo svaki od navedenih lokalnih operatora pretraživanja te navodimo kako se primijenjuju na svakom prikazu.

3.2.1. Zamjena dva posla na istom stroju

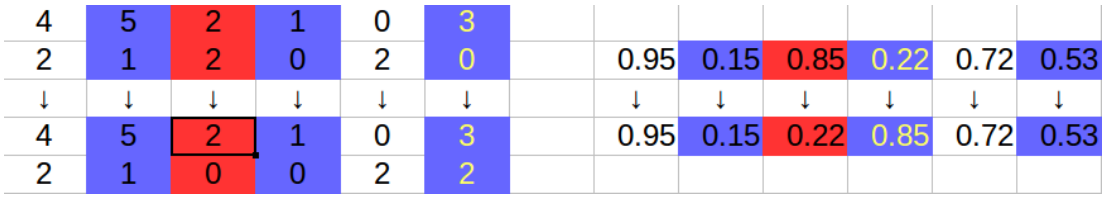
Kod zamjene dva posla koji se raspoređuju na isti stroj, prvo slučajno odaberemo jedan posao. Zatim se identificiraju ostali poslovi koji se trebaju rasporediti na isti stroj te nakon toga slučajnim odabirom uzimamo jedan od tih poslova i zamjenjujemo ga sa prvobitno odabranim poslom. Slika 3.4 pokazuje kako se navedeni operator provodi nad permutacijskim prikazom s dva niza (lijevo) i prikazom s decimalnim brojevima (desno). Crvenom bojom je označen prvobitno slučajno odabrani posao, plavom bojom su označeni potencijalni kandidati za zamjenu, dok je žutom označen slučajno odabrani kandidat za zamjenu. Primijetimo da se kod permutacijskog prikaza s dva niza zamjenjuju samo vrijednosti iz prvog niza.

4	5	2	1	0	3								
2	1	2	0	2	0		0.95	0.15	0.85	0.22	0.72	0.53
↓	↓	↓	↓	↓	↓		↓	↓	↓	↓	↓	↓
4	5	0	1	2	3		0.85	0.15	0.95	0.22	0.72	0.53
2	1	2	0	2	0							

Slika 3.4: Zamjena dva posla na istom stroju

3.2.2. Zamjena dva posla na različitim strojevima

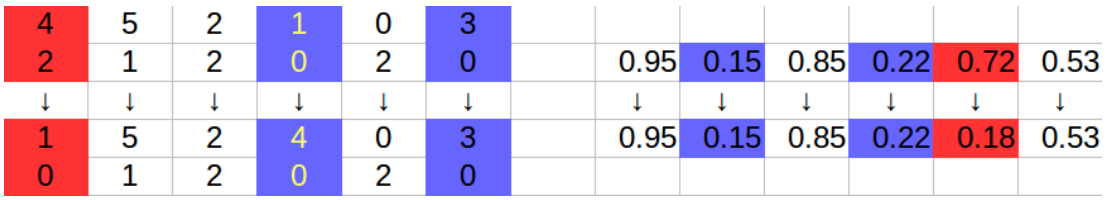
Zamjena dva posla koji se trebaju rasporediti na različite strojeve se provodi gotovo jednako kao i kod zamjene poslova na istim strojevima. Prvo slučajno odabiremo jedan posao te identificiramo potencijalne kandidate za zamjenu, no te kandidate odabiremo na način da uzimamo poslove koji se trebaju rasporediti na stroj koji nije jednak stroju od prvobitno odabranog posla. Na kraju slučajno odabiremo kandidata iz skupa kandidata i zamjenjujemo ga sa prvobitno odabranim poslom. Slika 3.5 pokazuje postupak provođenja ovog operatora, na isti način kao i kod prethodnog operatora.



Slika 3.5: Zamjena dva posla na različitim strojevima

3.2.3. Prebacivanje posla s jednog stroja na drugi

Kod ovog operatora također prvo slučajnim odabirom izabiremo jedan posao. Nakon toga slučajno odabiremo jedan stroj koji je različit od stroja na kojem se nalazi odabrani posao. Sada je potrebno slučajno odabrati poziciju na stroju gdje želimo ubaciti naš posao. Kod prikaza s nizom decimalnih brojeva, poslu dodijelimo slučajan decimalni broj koji upada u interval stroja na koji ga želimo prebaciti jer će se tako posao ubaciti na slučajno mjesto između postojećih poslova na tom stroju. Kod permutacijskog prikaza s dva niza je situacija kompliciranija jer nakon što odredimo poziciju između koja dva posla će se ubaciti naš novi posao, potrebno je posmaknuti prethodne poslove u permutacijskom nizu kako bi se redoslijed promijenio. Slika 3.6 slikovito prikazuje kako se ovaj operator primijenjuje na oba prikaza. Na slici se posao 4 prebacuje na stroj 0 između dva posla koji su već na tom stroju.



Slika 3.6: Prebacivanje posla s jednog stroja na drugi

4. Rezultati i analiza

U ovom poglavlju opisano je okruženje u kojem se provodilo testiranje te na koji način su se odabrali parametri algoritama kako bi dobili što bolje rezultate. Nakon toga iznosimo rezultate testiranja nad algoritmima koji koriste spomenute prikaze i lokalne operatore. Uvodimo oznake GA/PERM2 za permutacijski prikaz s dva niza, GA/FP za prikaz s nizom decimalnih brojeva te GA/PERM za permutacijski prikaz s jednim nizom. Kod testiranja lokalnih operatora u rezultatima se navodi vjerojatnost da se lokalni operator primijeni na jedinku što označavamo sa p_{ls}. Za heuristike koje smo spomenuli u 2. poglavlju te prikaze GA/PERM2 i GA/FP referiramo se na rezultate iz [1] te na njih dodajemo nove rezultate za GA/PERM i lokalne operatore.

4.1. Okruženje za testiranje i implementacija

S obzirom da se u ovom radu uvelike oslanjamo na rezultate iz [1], testiranje provodimo na jednakom skupu primjera i to na isti način. Skup primjera za testiranje se sastoji od 60 različitih problema raspoređivanja u kojem broj strojeva varira od 3 do 10 dok poslova ima između 12 i 100. Testiranje se provodi tako da se svaki primjer izvršava 30 puta te se uzima najbolji rezultat, osim testiranja lokalnih operatora gdje se svaki primjer izvršavao 10 puta zbog velikog broja eksperimenata koje je bilo potrebno obaviti. Ukupni rezultat se dobije tako da se zbroje najbolji rezultati za svaki primjer. Implementacija samih algoritama napisana je u jeziku C++ koristeći programsko okruženje ECF (Evolutionary Computation Framework [5]) koje se koristi za implementaciju raznih postupaka evolucijskog računanja među koje spadaju i genetski algoritmi.

4.2. Postavljanje parametara

Kvaliteta rezultata koje daje genetski algoritam jako ovisi o parametrima samog algoritma pa je zato potrebno odabrati optimalne parametre. Optimalni parametri se obično određuju tako da se isproba nekoliko različitih kombinacija od kojih se uzme najbolja. U ovom radu se osvrćemo na rezultate iz [1] te koristimo identične parametre. To se odnosi na maksimalni broj iteracija genetskog algoritma te veličinu populacije i vjerojatnost mutacije za svaki prikaz. Za maksimalni broj iteracija uzimamo 1000000, za permutacijski prikaz s dva niza koristimo vjerojatnost mutacije 0.7 i veličinu populacije 1000 dok za prikaz s nizom decimalnih brojeva koristimo vjerojatnost mutacije 0.5 i veličinu populacije 30. Za permutacijski prikaz s jednim nizom kojeg smo uveli u ovom radu koristimo jednake parametre kao i za permutacijski prikaz s dva niza. Pokazalo se da su ti parametri povoljni za ovaj prikaz te daju vrlo dobre rezultate.

Kod testiranja lokalnih operatora, kako bi se odredili optimalni parametri, isprobane su različite vrijednosti p_{ls}. Također se testirala i mogućnost da lokalni operatori potpuno zamjene mutaciju, pri čemu se onda u genetskom algoritmu postavlja vjerojatnost mutacije 0.

4.3. Usporedba rezultata

Tablica 4.1 prikazuje rezultate koje daju konstruktivne heuristike i genetski algoritmi za sva tri spomenuta prikaza. Iako novi prikaz koji je opisan u ovom radu ne pretražuje čitav prostor rješenja, kao ostala dva prikaza, iz rezultata je vidljivo kako je navedeni prikaz (GA/PERM) uspio nadmašiti ostala dva prikaza (GA/PERM2 i GA/FP) u svim testiranim kriterijima. S druge strane, GA/PERM je sporiji od ostalih zato što koristi heuristiku koja mora obaviti dodatne izračune kako bi interpretirala prikaz kao rješenje problema. Također primijetimo da genetski algoritmi općenito daju puno bolje rezultate od konstruktivnih heuristika, ali imaju puno veće vrijeme izvođenja [1]. Ovdje se može uočiti kompromis koji radimo između vremena izvođenja i kvalitete rezultata.

Algoritam	Kriterij			
	Twt	Nwt	Ft	Cmax
Heuristike				
Min-min	16.71	**7.143**	**157.2**	38.31
Max-min	22.06	8.138	195.8	38.83
Min-max	17.49	7.793	167.3	38.06
Sufferage	**16.65**	7.194	160.9	**37.92**
Genetski algoritmi				
GA/PERM2	9.725	5.615	151.2	37.14
GA/FP	9.533	5.340	140.8	36.79
GA/PERM	**9.441**	**5.306**	**137.8**	**36.499**

Tablica 4.1: Rezultati koje daju heuristike i genetski algoritmi bez lokalnih operatora

Prije testiranja lokalnih operatora na svim kriterijima, isprobane su različite kombinacije lokalnih operatora za kriterij Twt kako bi se utvrdila optimalna kombinacija s kojom bi zatim provodili daljnja testiranja. U tablici 4.2. prikazani su rezultati za sve kombinacije korištenja lokalnih operatora. Različite kombinacije operatora prikazane su u tablici sa binarnim nizom od 3 bita, pri čemu najlijeviji bit označava korištenje operatora zamjene poslova na istim strojevima, središnji bit označava korištenje operatora zamjene poslova na različitim strojevima, dok najdesniji bit označava korištenje operatora prebacivanja poslova s jednog stroja na drugi. Iz rezultata je vidljivo da najbolje rezultate daje kombinacija gdje se koristi samo operator zamjene dva posla na različitim strojevima pa ćemo u nastavku testiranja koristiti upravo tu kombinaciju.

Operatori	Prikaz	p_{ls}		
		0.05	0.1	0.2
001	GA/PERM2	12.31	13.53	18.22
	GA/FP	12.00	13.33	16.27
010	GA/PERM2	**10.92**	**12.78**	**15.85**
	GA/FP	**10.45**	**11.93**	**14.09**
011	GA/PERM2	11.33	12.94	16.82
	GA/FP	10.91	12.45	14.20
100	GA/PERM2	12.52	13.87	18.77
	GA/FP	12.22	13.80	16.71
101	GA/PERM2	12.88	14.03	19.01
	GA/FP	12.61	13.78	17.00
110	GA/PERM2	11.84	13.29	17.13
	GA/FP	11.53	12.90	14.61
111	GA/PERM2	12.09	13.42	17.88
	GA/FP	11.77	13.14	16.00

Tablica 4.2: Rezultati dobiveni korištenjem različitih kombinacija lokalnih operatora za Twt kriterij

U tablici 4.3 su prikazani rezultati testiranja lokalnih operatora. Treći stupac označava je li se koristila mutacija u algoritmu. Suprotno očekivanjima, lokalni operatori nisu uspijeli pridonijeti kvaliteti rezultata. Kada se koristi mutacija, lokalni operatori pogoršavaju rezultat što ih se češće koristi. S druge strane, kada ne koristimo mutaciju, lokalni operatori donekle nadomještaju operatore mutacije, ali nedovoljno za dobivanje kvalitetnih rješenja. Može se vidjeti da se u tom slučaju kvaliteta rješenja poboljšava s porastom p_{ls} do otprilike 0.3, nakon čega kvaliteta rješenja ponovno počinje opadati. Moguće objašnjenje za to je da se lokalni operatori ovdje koriste previše slučajno i zapravo u određenoj mjeri rade isto što i operator mutacije. Npr. za GA/FP prikaz, jedan od operatora mutacije koji se koristi je promjena jednog broja što je ekvivalentno prebacivanju posla s jednog stroja na drugi. Moguće je da bi se boljim usmjeravanjem pretrage u navedenim operatorima ipak mogli postići bolji rezultati.

Kriterij	Prikaz	Mutacija	p_{ls}						
			0.01	0.05	0.1	0.2	0.3	0.5	0.7
Twt	GA/PERM2	DA	**10.05**	10.92	12.78	15.85	18.21	26.56	35.13
		NE	22.31	18.39	16.66	15.23	14.18	16.71	21.21
	GA/FP	DA	**9.75**	10.45	11.93	14.09	17.05	25.46	32.12
		NE	20.62	18.34	16.24	14.99	14.01	17.35	19.98
Nwt	GA/PERM2	DA	**5.99**	6.41	6.83	7.28	7.91	8.88	9.33
		NE	7.80	7.24	6.92	6.86	6.34	7.36	8.56
	GA/FP	DA	**5.82**	6.27	6.76	7.29	8.01	8.81	9.29
		NE	7.67	7.11	6.70	6.26	6.10	7.22	8.32
Ft	GA/PERM2	DA	**153.0**	155.1	159.0	162.4	165.6	169.0	173.8
		NE	164.8	163.1	162.0	159.9	157.3	159.5	163.4
	GA/FP	DA	**142.2**	144.5	148.0	151.9	153.4	158.5	163.7
		NE	158.3	154.0	153.5	151.8	149.9	154.1	157.0
Cmax	GA/PERM2	DA	**38.01**	38.51	39.11	39.83	40.13	40.82	42.20
		NE	40.79	40.08	39.79	39.15	38.62	39.06	40.00
	GA/FP	DA	**37.89**	38.28	38.97	39.45	39.91	40.50	42.01
		NE	40.61	40.01	39.68	39.52	38.98	39.65	40.88

Tablica 4.3: Rezultati dobiveni korištenjem lokalnih operatora

5. Zaključak

U ovom radu opisan je novi prikaz rješenja za problem raspoređivanja u okruženju nesrodnih strojeva. Novi prikaz se usporedio sa postojeća dva prikaza te se pokazalo da iako postojeći prikazi daju vrlo dobre rezultate, novi prikaz ih je uspio nadmašiti u svim kriterijima raspoređivanja. Iz rezultata se može primijetiti kompromis između vremena izvođenja i kvalitete rezultata jer je novi prikaz najsporiji, no daje najbolje rezultate. S druge strane, konstruktivne heuristike za rješavanje problema raspoređivanja najbrže dolaze do rješenja, ali daju najlošije rezultate.

Osim prikaza rješenja, u ovom radu opisani su i lokalni operatori pretraživanja koji su se pokušali koristiti u kombinaciji sa nabrojanim prikazima s ciljem poboljšavanja rezultata. Suprotno očekivanjima, primjenom lokalnih operatora dobiveni su lošiji rezultati. Navedene lokalne operatore moguće bi bilo poboljšati (npr. prebacivanje posla sa jednog stroja na drugi se može preoblikovati da prebacuje poslove sa strojeva koji imaju predviđeno najdulje vrijeme trajanja na stroj koji ima predviđeno najkraće vrijeme trajanja).

Problem raspoređivanja ima široku primjenu (npr. raspoređivanje dretvi na procesore u operacijskim sustavima) te smatram da je važno istraživati nove pristupe koji bi doveli do boljih rezultata jer, kao što je vidljivo iz rezultata, heuristike koje se najčešće koriste ne daju dobre rezultate. Također, genetski algoritmi koji se koriste u ovom radu vrlo su koristan alat za rješavanje raznih problema i mogu dati vrlo dobre rezultate pa se zato nadam da će ovo istraživanje potaknuti druge da ih koriste.

6. Literatura

[1] M. Đurasević, D. Jakobović, "Comparison of solution representations for scheduling in the unrelated machines environment", Zagreb, 2016.

[2] J. Y-T. Leung, “Handbook of Scheduling: Algorthms, Models, and Performance Analysis”, Chapman & Hall/CRC, 2004.

[3] M. P. Pinedo, “Scheduling: theory, algorithms, and systems" , Springer Science & Business Media, 2012.

[4] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolutionary Programs”, Springer-Verlag, Berlin, 1992.

[5] "ECF - Evolutionary Computation Framework", URL: http://ecf.zemris.fer.hr/, 15.5.2016.

[6] J. Behnamian, M. Zandieh, S. M. T. Fatemi Ghomi, "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm", Elsevier Ltd., 2008.

7. Sažetak

Problem raspoređivanja je vrlo poznat i spada u klasu NP teških problema, što znači da ne postoje efikasni algoritmi koji pronalaze optimalno rješenje unutar razumnih vremenskih ograničenja. U ovom radu opisuje se pristup rješavanja tog problema korištenjem genetskih algoritama. Obrađuju se postojeći prikazi rješenja spomenutog problema nakon čega se uvodi novi prikaz koji se testira i uspoređuje s ostalima. Nadalje se opisuju operatori lokalnog pretraživanja i na koji način oni mogu poboljšati rad genetskog algoritma. Tri odabrana operatora se testiraju u kombinaciji s prethodno spomenutim rješenjima te se analizira njihova efikasnost. Osim usporedbe kvalitete rezultata, analizira se i vremenska i prostorna složenost različitih pristupa.