SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1670

Rješavanje problema raspoređivanja u okruženju nesrodnih strojeva korištenjem evolucijskih algoritama

Ivan Vlašić

Zagreb, srpanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2018.

DIPLOMSKI ZADATAK br. 1670

Pristupnik: **Ivan Vlašić (0036478088)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Rješavanje problema raspoređivanja u okruženju nesrodnih strojeva korištenjem evolucijskih algoritama**

Opis zadatka:

Opisati problem raspoređivanja poslova u okruženju nesrodnih strojeva. Istražiti prikaze kojima se mogu predstaviti rješenja problema raspoređivanja u okruženju nesrodnih strojeva. Osmisliti i ispitati različite načine inicijalizacije početne populacije u svrhu poboljšanja performansi i ubrzanja konvergencije postupaka evolucijskog računarstva. Osmisliti način izrade primjeraka problema raspoređivanja s dodatnim ograničenjima. Prilagoditi prikaz rješenja i genetske operatore evolucijskih algoritama u svrhu rješavanja problema raspoređivanja s dodatnim ograničenjima. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:
Prof. dr. sc. Domagoj Jakobović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Siniša Srbljić

Zahvaljujem mentoru prof. dr. sc Domagoju Jakoboviću i asistentu dr. sc. Marku Đuraseviću na pomoći i vodstvu tijekom studiranja i tijekom izrade ovog rada.

SADRŽAJ

1. Uvod

Raspoređivanje je proces kojim se određeni broj poslova dodjeljuje određenom skupu strojeva na način da se optimizira jedan ili više kriterija. Problem je prisutan u mnogim industrijama te se koristi u proizvodnji poluvodiča, raspoređivanju strojeva u tvornicama, aviona u zrakoplovnim lukama i drugim procesima [1]. U okruženju nesrodnih strojeva cilj je svaki posao dodijeliti jednom stroju na kojem se izvodi do svog završetka. Strojevi se smatraju nesrodnima ako vrijeme obavljanja posla ovisi o stroju na kojem se taj posao izvršava. Budući da raspoređivanje spada u NP-teške probleme, za njihovo rješavanje često se koriste različite metaheurističke metode. U metaheurističke metode ubrajaju se i različiti evolucijski algoritmi koji su korišteni u ovom radu.

Kvaliteta rješenja evolucijskog algoritma uvelike ovisi o načinu prikaza jedinke koja se koristi za kodiranje rješenja. Budući da svaki prikaz ima svoje prednosti i mane, potrebno je pronaći optimalni prikaz koji će postizati najbolje rezultate. Cilj ovog rada je usporediti šest prikaza rješenja za okruženje nesrodnih strojeva koji se spominju u literaturi, pri čemu će se optimizirati četiri kriterija i vrijeme izvođenja. Dodatno, u sklopu rada će se ispitati različiti načini inicijalizacije početne populacije u svrhu poboljšanja performansi i ubrzanja konvergencije postupaka evolucijskog računarstva. Na kraju će se prikaz rješenja i genetski operatori prilagoditi problemima raspoređivanja s dodatnim ograničenja u svrhu dodatnog testiranja kvalitete algoritma.

Ostatak ovog rada strukturiran je na sljedeći način. Drugo poglavlje daje kratki pregled problema raspoređivanja u okruženju nesrodnih strojeva i kriterija koji se koriste u optimizaciji. Treće poglavlje sadrži kratki pregled evolucijskih algoritama. Četvrto poglavlje opisuje prikaze rješenja koji su korišteni te uspoređuje postignute rezultate. U petom poglavlju optimalni pronađeni prikaz rješenja dodatno je testiran s različitim načinima inicijalizacije početne populacije u svrhu poboljšanja kvalitete postignutih rješenja i konvergencije postupaka. Šesto poglavlje uvodi dodatna ograničenja u problem raspoređivanja te se prikaz rješenja i genetski operatori prilagođavaju za rješavanje takvih problema. Dodatno, uspoređuje se rad različitih algoritama. U sedmom poglavlju je iznesen kratki pregled rada i zaključak o postignutim rezultatima.

2. Raspoređivanje u okruženju nesrodnih strojeva

Okruženje nesrodnih strojeva sastoji se od n poslova koje je potrebno rasporediti na m strojeva [2]. Svaki posao opisan je s nekoliko svojstava koja uključuju vrijeme obrade p_{ij} koje označava trajanje izvršavanja posla j na stroju i, vrijeme objavljivanja r_j koje označava vrijeme pojavljivanja posla u sustavu, vrijeme željenog završetka d_j koje označava točku u vremenu do koje bi posao trebao biti obavljen te težinu w_j koja označava važnost posla.

U problemu se mogu pojaviti i različita ograničenja poput ograničenja u redoslijedu izvođenja poslova, vremena postavljanja, kvarova strojeva i ograničenja na kojim strojevima se pojedini poslovi mogu izvoditi pri čemu je dodatno potrebno prilagoditi postupak rješavanja.

2.1. Mjere poslova

Za svaki od poslova definirane su sljedeće mjere: [1]

- C_j - označava vrijeme završetka pojedinog posla
- F_j - označava vrijeme koje je posao proveo u sustavu, definira se kao $F_j = C_j - r_j$
- T_j - označava vrijeme koje je posao proveo u sustavu nakon predviđenog vremena završetka, definira se kao $T_j = max(C_j - d_j, 0)$
- U_j - označava je li posao obavljen nakon predviđenog vremena završetka, definira se kao $U_j = \begin{cases} 1\text{: } \mathrm{T}_j > 0 \\ 0\text{: } \mathrm{T}_j = 0 \end{cases}$

2.2. Kriteriji testiranja

Kako bi bilo moguće usporediti različite prikaze rješenja i donijeti zaključke o njihovoj kvaliteti, potrebno je odrediti kriterije koji će biti korišteni u testiranju. Kriteriji koji će biti korišteni u ovom radu su sljedeći:

- C_{max} - označava najkasnije vrijeme završetka svih poslova, definira se kao $C_{max} = max_j(C_j)$
- Ft - označava sumu svih vremena koje su poslovi proveli u sustavu, definira se kao $Ft = \sum_j F_j$
- Twt - označava težinsku sumu kašnjenja svih poslova, definira se kao $Twt = \sum_j w_j T_j$
- Nwt - označava težinsku sumu poslova obavljenih nakon vremena završetka, definira se kao $Uwt = \sum_j w_j U_j$

2.3. Klasifikacija problema raspoređivanja

Problemi raspoređivanja mogu se klasificirati u nekoliko grupa ovisno o pojedinim svojstvima. Ovisno o dostupnosti parametara (poput informacija o svim poslovima koji će doći u sustav) problemi raspoređivanja mogu se podijeliti u grupu u kojoj su te informacije poznate od početka te grupu u kojoj informacije o pojedinom poslu postanu dostupne tek kad taj posao dođe u sustav. Raspoređivanje također može biti statičko, kod kojeg je cijeli raspored konstruiran prije izvršavanja sustava, te dinamično, kod kojeg se raspored gradi tijekom izvršavanja sustava. Uvjeti raspoređivanja u velikoj mjeri utječu na algoritme koji se mogu koristiti za rješavanje problema. Primjerice, konstruktivni algoritmi koji iterativno stvaraju rješenje se obično koriste u dinamičkim sustavima, dok se kod statičkih sustava uglavnom koriste metode heurističkog pretraživanja. Ovaj rad usredotočen je na statičko raspoređivanje kod kojeg su sve informacije u sustavu poznate unaprijed.

3. Evolucijski algoritmi

Kao što je naznačeno u uvodu, problem raspoređivanja spada u NP-teške probleme. Iz tog razloga, za njihovo rješavanje se često koriste različite heurističke metode kod kojih uglavnom nije sigurno je li pronađeno rješenje optimalno, no rezultati su obično dovoljno dobri za korištene potrebe. Jedna od takvih metoda su evolucijski algoritmi koji su korišteni u ovom radu. U navedenu skupinu spadaju genetski algoritmi, evolucijske strategije, genetsko programiranje, evolucijsko programiranje i klasifikatorski sustavi sa sposobnošću učenja.

3.1. Glavne značajke

Glavna značajka evolucijskih algoritama je simulacija procesa evolucije u prirodi. Algoritam započinje s nasumično generiranom populacijom jedinki (rješenja) koje se postupno poboljšavaju kroz generacije. Kako bi se jedinke mogle uspoređivati, potrebno je definirati funkciju dobrote koja određuje kvalitetu svake od jedinki. Na osnovu toga operatorima selekcije bolje jedinke preživljavaju i prenose svoj genetski materijal u sljedeće generacije dok slabije jedinke izumiru. Jedna od češće korištenih selekcija, koja je korištena i u ovom radu, je turnirska selekcija kod koje se iz nasumično odabranih n jedinki iz populacije odabire ona najbolja. Odabrane jedinke operatorima križanja kombiniraju svoj genetski materijal čime nastaje nova jedinka - dijete, s karakteristikama oba roditelja. Dodatno pretraživanje prostora rješenja postiže se operatorima mutacije čime se u novonastaloj jedinki uvode nasumične manje promjene. Mutacijom se dodatno postiže izbjegavanje lokalnih optimuma i postiže nadoknada genetskog materijala izgubljenog križanjem. Rezultat je nova jedinka koja u idealnom slučaju ima bolje karakteristike i dobrotu od svojih roditelja. Algoritam se provodi iterativno do ispunjavanja kriterija zaustavljanja koji, ovisno o problemu, može biti različit - proteklo vrijeme, broj generacija, broj evaluacija, postignuta dobrota i slično. Algoritam 1 prikazuje primjer jednog jednostavnog genetskog algoritma.

Budući da svaki problem ima različite karakteristike, i prikaz rješenja jedinke će

Algoritam 1: Genetski algoritam

```
inicijaliziraj populaciju s nasumičnim rješenjima;
izračunaj dobrotu svake jedinke;
dok uvjet zaustavljanja nije zadovoljen čini
    izaberi roditelje;
    obavi križanje nad roditeljima;
    mutiraj dobivene potomke;
    izračunaj dobrotu novih jedinki;
    izaberi jedinke za sljedeću generaciju;
kraj
```

se razlikovati od problema do problema, kao i korišteni operatori selekcije, križanja i mutacije. Za svaki algoritam dodatno je potrebno definirati i druge parametre poput veličine populacije, vjerojatnosti mutacije i kriterija zaustavljanja. Često nije moguće precizno odrediti optimalne parametre s kojima će algoritam postići najbolje rezultate te je potrebno istražiti više različitih prikaza rješenja kao i vrijednosti parametara algoritma te usporedbom rezultata doći do zaključka o optimalnim parametrima. Navedeno će detaljnije biti opisano u sljedećem poglavlju.

4. Usporedba prikaza

Kao što je već spomenuto, kvaliteta rješenja evolucijskog algoritma uvelike ovisi o načinu prikaza jedinke koja se koristi za kodiranje rješenja. Pri tome, nije moguće pronaći općenit optimalni prikaz, već će se rezultati razlikovati za svaki od promatranih problema.

U ovom poglavlju usporedit će se šest prikaza rješenja za okruženje nesrodnih strojeva koji se spominju u literaturi, pri čemu su optimizirana četiri kriterija.

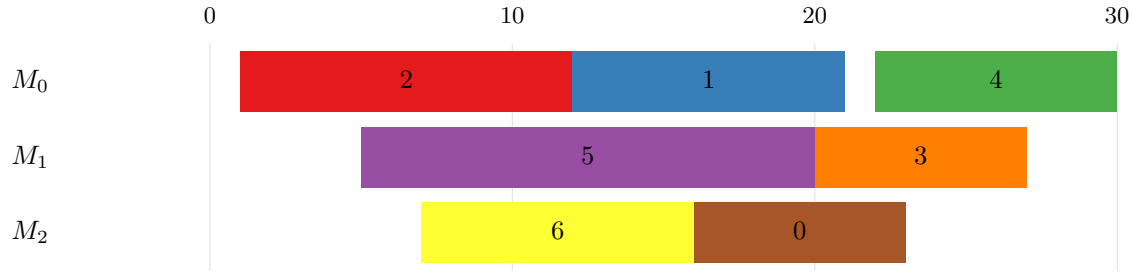
Kako bi se bolje označile razlike, za svaki od prikaza bit će prikazano rješenje jednostavnog problema raspoređivanja. U tablici 4.1 opisana su svojstva problema dok je na slici 4.1 prikazano rješenje koje će se kodirati svakim od prikaza. Problem se sastoji od sedam poslova koje je potrebno rasporediti na tri stroja.

Tablica 4.1: Svojstva problema korištenog za analizu

Indeks posla j	r_j	d_j	w_j	p_{0j}	p_{1j}	p_{2j}
0	15	30	0.9	10	7	7
1	10	23	0.07	9	9	8
2	1	13	0.87	11	13	12
3	20	37	0.06	9	6	11
4	22	35	0.06	10	14	11
5	5	25	0.26	11	15	16
6	7	20	0.78	15	12	9

4.1. Prikazi rješenja za problem nesrodnih strojeva

U ovom poglavlju opisani su prikazi koji su korišteni za rješavanje problema raspoređivanja u okruženju nesrodnih strojeva. Prikazi koji će biti korišteni su kodiranje permutacijskim nizom, kodiranje permutacijskim nizom s listom strojeva, kodiranje



Slika 4.1: Primjer rješenja problema raspoređivanja

listom realnih brojeva, kodiranje slučajnim nizom, kodiranje matricom i kodiranje grupama poslova. Svaki od prikaza koristi prilagođene operatore križanja i mutacije koji osiguravaju da se u svakom trenutku dobiju valjana rješenja.

4.1.1. Kodiranje permutacijskim nizom

Kodiranje permutacijskim nizom (engl. *permutation encoding*, PE) jedan je od najjednostavnijih prikaza [3]. Prikaz koristi listu cijelih brojeva čija veličina odgovara broju poslova n tako da brojevi u listi odgovaraju indeksima pojedinih poslova. Poslovi će biti raspoređeni na strojeve ovisno o pozicijama u listi na način da će poslovi na početku liste biti raspoređeni s većim prioritetom, prije strojeva na kraju. Međutim, prikaz ne sadrži informacije o tome na koji stroj će svaki od poslova biti raspoređen. U ovom radu, poslovi su raspoređeni na onaj stroj na kojem će se najranije izvršiti.

Na slici 4.2 prikazano je rješenje korištenjem ovog prikaza. Prvi posao kojeg je potrebno rasporediti je posao J_6. Posao će biti raspoređen na stroj M_2 budući da na navedenom stroju postiže najranije vrijeme izvođenja. Sljedeći posao koji je potrebno rasporediti je J_2. Za navedeni posao vrijeme završetka na stroju M_0 iznosi 12, a na stroju M_1 iznosi 14. Budući da već postoje poslovi koji su raspoređeni na stroju M_2, potrebno je prvo izračunati njihovo vrijeme završetka što u ovom slučaju iznosi 13. Tada bi vrijeme završetka posla J_2 na stroju M_2 iznosilo 25. Posao će biti raspoređen na stroj M_0 budući da će na u tom slučaju najranije završiti s izvođenjem. Ostali poslovi raspoređeni su na isti način.

6	2	5	1	0	4	3

Slika 4.2: Rješenje kodirano permutacijskim nizom

4.1.2. Kodiranje permutacijskim nizom s listom strojeva

Kodiranje permutacijskim nizom s listom strojeva (engl. *permutation encoding with machine list*, PEML) slično je prikazu običnog kodiranja permutacijskim nizom uz dodatak da prikaz sadrži dodatnu listu čije vrijednosti odgovaraju indeksima strojeva u sustavu na koji odgovarajući posao treba biti raspoređen [4]. Na ovaj način, prikaz sadrži sve informacije potrebne za raspoređivanje te nije potrebno uvoditi dodatne izračune.

Na slici 4.3 prikazano je rješenje korištenjem ovog prikaza. Prvi posao kojeg je potrebno rasporediti je J_6. Posao će biti raspoređen na stroj M_2, budući da se indeks navedenog stroja nalazi na odgovarajućoj poziciji u listi strojeva. Sljedeći posao je J_2, koji će biti raspoređen na stroj M_0. Ostali poslovi raspoređeni su na isti način.

6	2	5	1	0	4	3
2	0	1	0	2	0	1

Slika 4.3: Rješenje kodirano permutacijskim nizom s listom strojeva

4.1.3. Kodiranjem slučajnim nizom

Kodiranje slučajnim nizom (engl. *random key encoding*, RKE) koristi listu realnih brojeva čije su vrijednosti u intervalu $[0, m\rangle$ gdje m predstavlja broj strojeva [5]. U ovom prikazu dužina liste jednaka je broju poslova tako da je svako polje indeksom povezano s odgovarajućim poslom. Cijeli dio broja određuje stroj na kojem će se odgovarajući posao izvršiti, dok realni dio određuje njegov prioritet. Prioritet poslova za pojedini stroj određuje se na način da će oni poslovi s manjom vrijednosti biti raspoređeni ranije.

Na slici 4.4 prikazano je rješenje korištenjem ovog prikaza. Budući da primjer sadrži tri stroja mogući interval je $[0, 3\rangle$. Prvo polje u listi koristi se za raspoređivanje posla J_0. Iz cijelog dijela broja vidljivo je da će posao biti raspoređen na stroj M_1. Na navedeni stroj bit će raspoređeni i poslovi J_3 i J_6. Po realnom dijelu broja vidljivo je da najveći prioritet ima posao J_3, a najmanji J_6. Ostali poslovi raspoređeni su na isti način.

1.56	0.46	2.11	1.13	2.86	0.54	1.79

Slika 4.4: Rješenje kodirano slučajnim nizom

4.1.4. Kodiranje listom realnih brojeva

Kodiranje listom realnih brojeva (engl. *floating point encoding*, FPE) koristi listu realnih brojeva čije su vrijednosti u intervalu $[0, 1]$ [4]. Kako bi se utvrdio stroj na kojem će se pojedini posao izvršiti, prvo je interval $[0, 1]$ potrebno podijeliti na m jednakih podintervala. Posao će biti raspoređen na stroj ovisno o tome kojem podintervalu stroja njegova vrijednost pripada. Redoslijed kojim će se poslovi izvoditi na pojedinom stroju određuje se na način da će poslovi koji imaju manju vrijednost prioriteta biti raspoređeni ranije.

Na slici 4.5 prikazano je rješenje korištenjem ovog prikaza. Budući da primjer sadrži tri stroja, intervali će biti $[0, 0.33\rangle$ za stroj M_0, $[0, 33, 0.66\rangle$ za stroj M_1 i $[0.66, 1]$ za stroj M_2. Korištenjem ovih intervala, može se zaključiti da će na stroju M_0 biti izvršeni poslovi J_0, J_2 i J_6. Prvi će se izvršiti posao J_3 budući da mu je dodijeljena najmanja vrijednost, potom slijedi J_0 te J_6. Ostali poslovi raspoređeni su na isti način.

0.23	0.46	0.11	0.79	0.86	0.54	0.31

Slika 4.5: Rješenje kodirano listom realnih brojeva

4.1.5. Kodiranje matricom

Kodiranje matricom (engl. *matrix encoding*, ME) koristi matricu realnih brojeva čiji retci odgovaraju strojevima, a stupci poslovima [6]. U sustavu s m strojeva i n poslova, matrica će imati m redaka i n stupaca. Svako polje matrice sadrži vrijednost iz intervala $[0, 1]$. Ako polje sadrži vrijednost 0, posao s indeksom koji odgovara indeksu stupca neće se izvršiti na stroju s indeksom koji odgovara indeksu retka. Ako polje sadrži vrijednost veću od nule, vrijednost odgovara prioritetu koji taj posao ima na odgovarajućem stroju. Budući da se svaki posao može izvršiti na samo jednom stroju, potrebno se pobrinuti da svaki stupac ima vrijednost veću od nule u samo jednom od redaka. Navedenom ograničenju je potrebno prilagoditi operatore križanja i mutacije.

Na slici 4.6 prikazano je rješenje korištenjem ovog prikaza. Na stroju M_0 bit će raspoređeni poslovi J_0, J_3 i J_5 budući da u navedenom retku stupci koji odgovaraju

njihovim indeksima imaju vrijednosti veće od 0. Prvi posao koji će se izvršiti na stroju je J_5 budući da ima najveći prioritet, nakon toga slijedi J_0 pa J_3. Ostali poslovi raspoređeni su na isti način.

$$\begin{bmatrix} 0.21 & 0 & 0 & 0.72 & 0 & 0.1 & 0 \\ 0 & 0.48 & 0 & 0 & 0 & 0 & 0.35 \\ 0 & 0 & 0.9 & 0 & 0.24 & 0 & 0 \end{bmatrix}$$

Slika 4.6: Rješenje kodirano matricom

4.1.6. Kodiranje grupama poslova

Kodiranje grupama poslova (engl. *machine list encoding*, MLE) jednostavan je prikaz koji za svaki od strojeva sadrži listu u kojoj su navedeni poslovi koji će se izvršiti na tom stroju redoslijedom kojim su navedeni u listi [7]. Kod ovog prikaza potrebno je osigurati da se svaki od poslova nalazi u samo jednoj listi budući da se svaki posao može izvršiti na samo jednom stroju.

Na slici 4.7 prikazano je rješenje korištenjem ovog prikaza. Budući da prva lista pripada prvom stroju, iz priloženog se može zaključiti da će se na stroju M_0 izvršiti poslovi J_1 i J_4. Ostali poslovi raspoređeni su na isti način.

1	4

6	0	2

5	3

Slika 4.7: Rješenje kodirano grupama poslova

4.2. Postavke eksperimenta

U analizi prikaza korišteni su kriteriji Ft, Twt, Nwt, C_{max} koji su prethodno opisani. Svaki od kriterija optimiziran je samostalno i neovisno od drugih na uzorku od 60 problema s ukupnom dobrotom izračunatom kao sumom vrijednosti kriterija najboljeg rješenja za svaki od problema. Korišteni problemi imaju raznovrsne karakteristike

kako bi se algoritam i prikazi mogli testirati što detaljnije. Problemi sadržavaju između 12 i 100 poslova, te 3 i 10 strojeva.

Kako bi se postigli što bolji rezultati i olakšala usporedba, parametri svakog od prikaza su prethodno optimizirani. U tablici 4.2 sadržani su svi parametri korišteni za pojedine prikaze. U svim primjerima korištena je veličina populacije od 30 jedinki uz uvjet zaustavljanja od 1 000 000 evaluacija. Za selekciju je korištena eliminacijska turnirska selekcija s veličinom turnira od 3 jedinke, dok su ostali operatori, kao i vjerojatnost mutacije, prilagođeni pojedinom prikazu.

Svaki od problema optimiziran je ukupno 30 puta u svrhu točnije analize. Za svaki od izračuna spremljena je najbolja vrijednost te je na osnovu dobivenih vrijednosti izračunata minimalna i maksimalna vrijednost dobrote te njihov medijan. Dodatno, uz navedene vrijednosti, u obzir se uzima i $Tmin$, što predstavlja sveukupno najbolje rješenje dobiveno kombinacijom najboljih rezultata iz svih 30 pokretanja. Kao što je ranije spomenuto, u analizi rezultata koristi se i vrijeme izvođenja svakog od prikaza.

Tablica 4.2: Parametri korišteni za prikaze

Parametar	PE	PEML	RKE	FPE	ME	MLE
Veličina populacije	30	30	30	30	30	30
Vjerojatnost mutacije	0.7	0.7	0.9	0.3	0.9	0.9
Selekcija	Eliminacijska turnirska selekcija s veličinom od 3 jedinke					
Uvjet zaustavljanja	1 000 000 evaluacija					

4.3. Rezultati

Dobiveni rezultati prikazani su u tablici 4.3. Najbolje rješenje u svakom retku označeno je sivom bojom. Dodatno, na slici 4.8 prikazani su boxplotovi rezultata u svrhu detaljnije usporedbe. Iz tablice je vidljivo da PE prikaz postiže najbolje rezultate za sve kriterije. PE dodatno postiže i veliku stabilnost što je vidljivo iz boxplotova jer navedeni prikaz postiže najmanje raspršene rezultate za svaki od kriterija. S druge strane, najlošiji rezultati dobiveni su za PEML prikaz koji je postigao najgore rezultate za sve kriterije te se ujedno pokazao iznimno nestabilnim zbog čega se rezultati svakog od izračuna najčešće u velikoj mjeri razlikuju.

Prikazi koji koriste listu realnih brojeva postižu lošije rezultate od PE prikaza, ali puno bolje rezultate od PEML prikaza. Iz tablice je vidljivo da FPE prikaz obično

postiže lošije rezultate od svih ostalih prikaza, osim PEML prikaza, što je vidljivo kod optimiziranja kriterija Twt i C_{max}. Rezultati dobiveni RKE prikazom su dosta bolji, ali i kod ovog prikaza je vidljivo da ne postiže jednako dobre rezultate za sve kriterije. Za kriterije Ft i Nwt dobiveni rezultati su treći po kvaliteti s vrlo malo razlike s drugo plasiranim prikazom, dok su rezultati ostalih kriterija nešto lošiji, pogotovo za C_{max} kriterij.

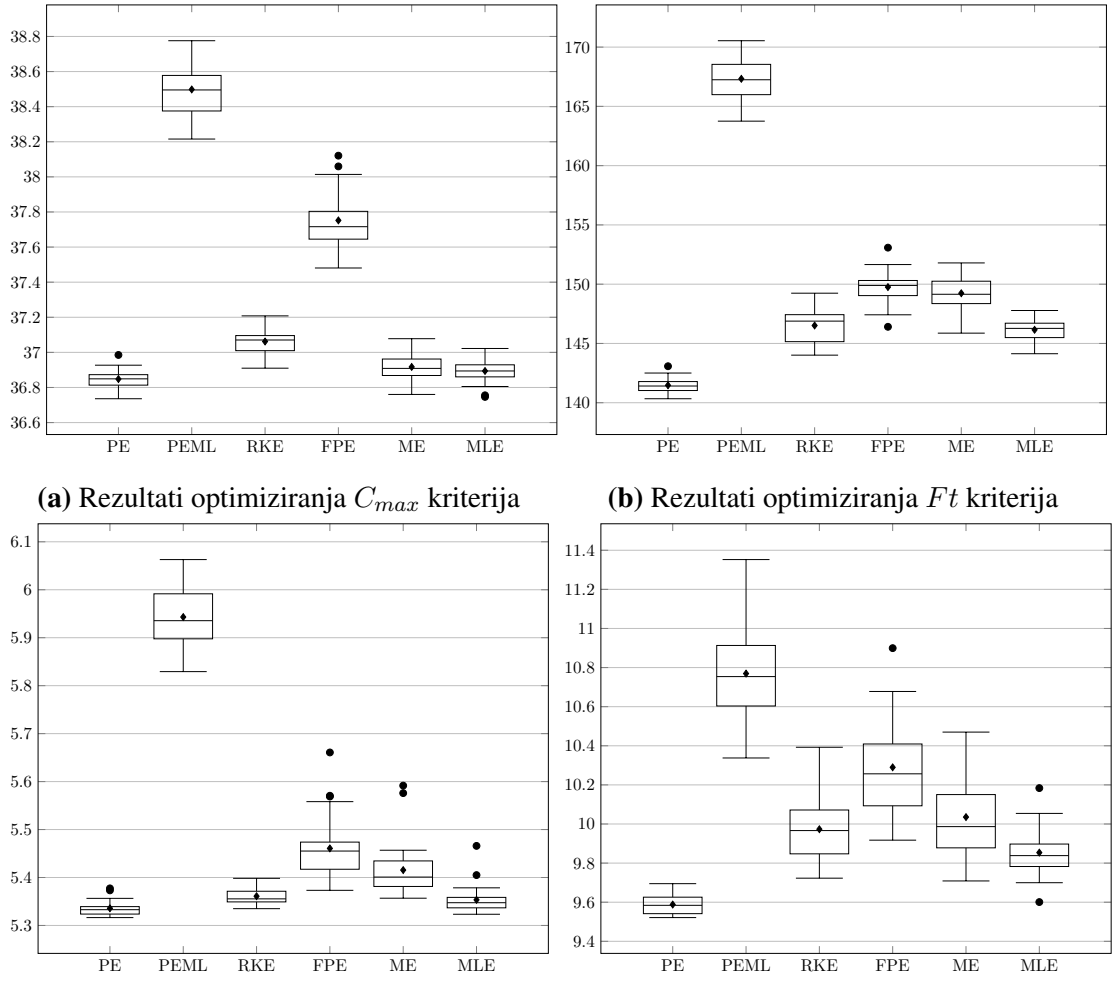
Kod ME prikaza, optimiziranje kriterija C_{max} je postiglo rezultate koji su bili samo malo lošiji od onih dobivenih prikazima PE i MLE, dok su za ostale kriterije rezultati dobiveni prikazima PE, RKE i MLE bili dosta bolji. Zadnji prikaz, MLE, konzistentno je postizao dobre rezultate za sve kriterije. Dodatna prednost kod ovog prikaza je veća stabilnost.

Tablica 4.3: Rezultati dobiveni testiranjem prikaza rješenja

Kriterij		Prikaz					
		PE	PEML	RKE	FPE	ME	MLE
Vrijeme izvođenja		681.0	508.1	436.8	339.1	489.5	505.4
C_{max}	Tmin	36.59	37.14	36.55	36.79	36.50	36.52
	Min	36.74	38.22	36.91	37.48	36.76	36.75
	Med	36.85	38.50	37.07	37.72	36.91	36.90
	Max	36.99	38.78	37.21	38.12	37.08	37.02
Ft	Tmin	138.9	151.2	139.7	140.8	141.0	139.7
	Min	140.3	163.8	144.0	146.4	145.9	144.1
	Med	141.4	167.3	146.9	149.9	149.2	146.3
	Max	143.1	170.5	149.2	153.1	151.8	147.8
Nwt	Tmin	5.316	5.615	5.334	5.340	5.317	5.318
	Min	5.316	5.829	5.335	5.373	5.357	5.323
	Med	5.333	5.938	5.356	5.455	5.403	5.348
	Max	5.377	6.063	5.398	5.661	5.592	5.466
Twt	Tmin	9.452	9.725	9.516	9.533	9.525	9.499
	Min	9.521	10.34	9.723	9.917	9.709	9.601
	Med	9.584	10.76	9.968	10.27	9.987	9.846
	Max	9.695	11.35	10.39	10.90	10.47	10.18

Dodatni kriterij koji se razmatra pri usporedbi je vrijeme izvođenja. U tablici 4.3

je vidljivo da se vremena uvelike razlikuju kod svih prikaza. Iako prikaz PE postiže najbolje rezultate, ujedno ima i najduže vrijeme izvođenja, dok FPE, iako nije postigao jednako dobre rezultate, ima najkraće vrijeme izvođenja koje je skoro duplo manje od onog za PE prikaz. Vremena izvođenja ostalih prikaza su između dva navedena i bitno se ne razlikuju. Iz navedenog se može zaključiti da prikazi u velikoj mjeri utječu na vremena izvođenja. Dok prikazi koji koriste listu realnih brojeva poput RKE i FPE prikaza postižu najkraća vremena, oni prikazi koji koriste permutacijske liste poput PE, PEML i MLE prikaza, imaju dosta duža vremena izvođenja.



(a) Rezultati optimiziranja C_{max} kriterija

(b) Rezultati optimiziranja Ft kriterija

(c) Rezultati optimiziranja Nwt kriterija

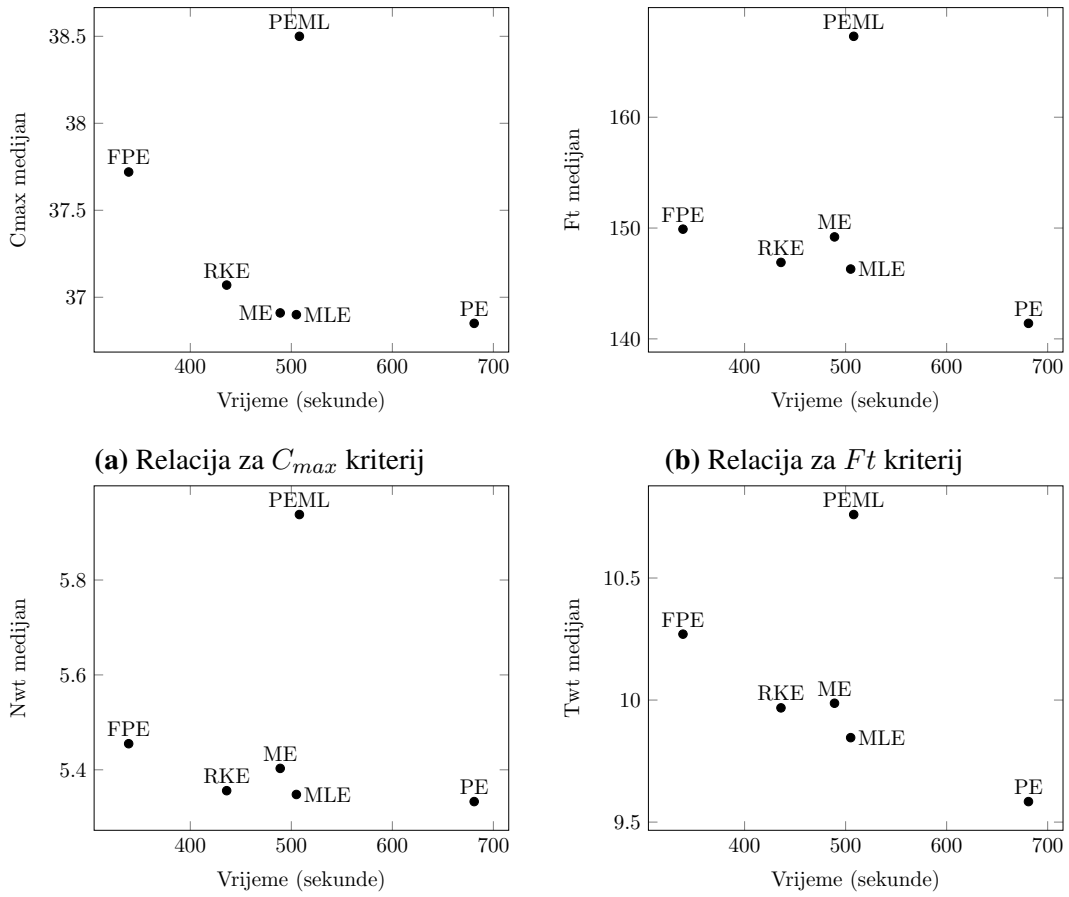
(d) Rezultati optimiziranja Twt kriterija

Slika 4.8: Usporedba rezultata dobivenih različitim prikazima rješenja

Na slici 4.9 prikazana je relacija medijana vrijednosti dobivenih različitim prikazima za sve kriterije i njihovih vremena izvođenja. Iz navedenog je vidljivo da svi kriteriji postižu najbolje rezultate, ali i najduže vrijeme izvođenja za prikaz PE, dok prikaz FPE postiže najbrže vrijeme izvođenja, ali ne i najlošije rezultate. Ostali prikazi

imaju različite kompromise između postignutih rezultata i vremena izvođenja. Vidljivo je da prikazi koji postižu bolje rezultate obično imaju dulje vrijeme izvođenja. Iznimka je prikaz PEML koji, iako ima dugačko vrijeme izvođenja, postiže najlošije rezultate.

Pored rezultata dobivenih uz fiksan broj evaluacija, zanimljivo je usporediti i rezultate dobivene uz vremenski limit. Na slici 4.10 prikazan je tijek promjene dobrote pri optimizaciji svih kriterija uz vremenski limit od 5 sekundi. Iz priloženog je još jednom vidljivo da PEML prikaz postiže daleko najgore rezultate. Kod PE prikaza, za koji je dosad zaključeno da postiže najbolje rezultate, vidljivo je da su u početku, osim za kriterij C_{max}, dobiveni rezultati lošiji od ostala četiri prikaza. Međutim, vrijednost dobrote navedenog prikaza se smanjuje brže od ostalih što na kraju rezultira boljim rezultatima. Kod ostalih prikaza nisu vidljive toliko značajne razlike između početnih rješenja i rezultata. Iako FPE obično kreće s boljim rezultatima nego RKE prikaz, RKE postiže bolju konvergenciju i bolje rezultate. Za MLE prikaz je vidljivo da u većini slučajeva započinje s najboljim rezultatima što utječe i na krajnje vrijednosti.



(a) Relacija za C_{max} kriterij

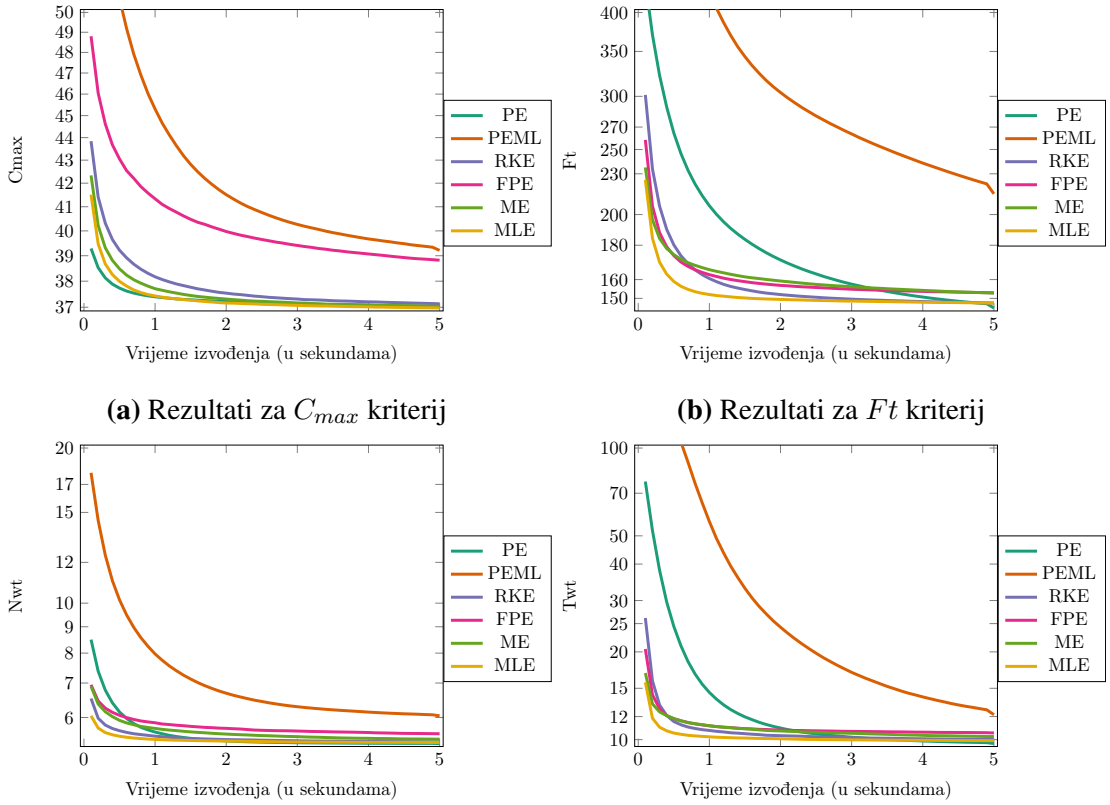
(b) Relacija za Ft kriterij

(c) Relacija za Nwt kriterij

(d) Relacija za Twt kriterij

Slika 4.9: Relacija između vremena izvođenja i učinka algoritma korištenjem različitih prikaza

FPE prikaz za C_{max} kriterij postiže rezultate koji su samo malo bolji od najlošijeg prikaza PEML, ali za sve ostale kriterije postiže zadovoljavajuće rezultate koji se ne razlikuju u velikoj mjeri od ostalih prikaza.



(a) Rezultati za C_{max} kriterij

(b) Rezultati za Ft kriterij

(c) Rezultati za Nwt kriterij

(d) Rezultati za Twt kriterij

Slika 4.10: Promjene vrijednosti dobrote tijekom izvođenja algoritma

Uzimajući u obzir samo dobivene rezultate, može se doći do zaključka da je PE superiorniji od ostalih prikaza. Kodiranje permutacijskim nizom koji odlučuje samo o prioritetu kojim se poslovi raspoređuju na strojeve se pokazala kao dobra odluka budući da algoritam treba samo pronaći dovoljno dobar poredak poslova, dok se raspoređivanje poslova na strojeve prepušta heuristici. Nedostatak ovog prikaza su lošiji rezultati na početku izračuna budući da algoritmu treba neko vrijeme da pronađe prikladne permutacije. Dodatni nedostatak je vrijeme izračuna gdje se prikaz pokazao kao najsporiji, kao i činjenica da dva različita prikaza mogu rezultirati istim rasporedom budući da se za raspoređivanje koristi heuristika koja za više različitih prioriteta poslova može dovesti do istog rasporeda. S druge strane, PEML prikaz je za sve kriterije postigao najlošije rezultate što dovodi da zaključka da korištenje dodatnog genotipa rezultira lošijom kvalitetom prikaza. Budući da se operatori primjenjuju neovisno na genotipima, moguće su situacije u kojima je postignuta dobra permutacija poslova, ali

je dobiveni raspored poslova na strojeve dovoljno loš da negativno utječe na dobivene rezultate rješenja. Kod prikaza koji koriste listu realnih brojeva pokazano je da RKE prikaz postiže puno bolje rezultate od FPE prikaza koji se pokazao kao bolji samo od najlošijeg prikaza. Prednost kod ovih prikaza je vrijeme izvođenja kao i mala memorijska složenost budući da koriste samo jednu listu realnih brojeva. Prikaz ME je za sve kriterije osim C_{max} postigao prosječna rješenja koja se ne ističu previše od ostalih prikaza. Nedostatak je najveća memorijska složenost od svih ostalih prikaza budući da koristi matricu veličine $m * n$, kao i činjenica da je potrebno definirati nove operatore koji su posebno prilagođeni navedenom prikazu te potreba izbjegavanja nedopuštenih rješenja. Zadnji prikaz, MLE, je postigao druge po redu rezultate za svaki od kriterija. Prednost ovog prikaza nad PE prikazom je vrijeme izvođenja koje je primjetno bolje, ali uz zamjenu za malo veću memorijsku složenost. Dodatna prednost ovog prikaza je jednostavnost interpretacije. Uzimajući sve u obzir, prikaz koji je korišten u daljnjim izračunima je MLE budući da je postigao dovoljno dobar omjer između dobivenih rezultata i vremena izvođenja. Navedeni prikaz također omogućuje prilagođavanje različitim ograničenjima što je bilo posebno korisno u nastavku rada.

5. Inicijalizacija početne populacije upotrebom pravila raspoređivanja

Kao što je u prošlom poglavlju pokazano, pogotovo na primjeru MLE prikaza, početna populacije može u velikoj mjeri utjecati na dobivene rezultate. Prikaz je uglavnom započinjao s boljom početnom populacijom što je rezultiralo i boljom prosječnom dobrotom krajnjih rješenja. Navedeno dovodi do zaključka da odabir prikaza i operatora nisu jedini način poboljšavanja algoritma. U literaturi se na puno mjesta opisuju različite heuristike konstruiranja rasporeda i utjecaj koji imaju na kvalitetu rezultata [8] [9] [10] [11]. Cilj ovog poglavlja je istražiti neke od heuristika korištenih za rješavanje problema u okruženju nesrodnih strojeva i iskoristiti ih za inicijalizaciju početne populacije evolucijskih algoritama te dobivene rezultate usporediti s onima dobivenim algoritmom u kojem se koristi nasumično inicijalizirana početna populacija.

5.1. Pravila raspoređivanja

Pravila raspoređivanja su jednostavne heuristike koje inkrementalno grade raspored. Rade na način dodjeljivanja prioriteta poslovima i strojevima tako da tijekom svakog oslobađanja stroja na njega rasporede sljedeći neraspoređeni posao s najvećim prioritetom. Primjer takvog pravila raspoređivanja je funkcija $p_i = 1/p_{ij}$ koja bi prvo raspoređivala poslove s kraćim trajanjem. Prednost ovakvog načina konstruiranja rješenja je brzina izvođenja, primjenjivost u dinamičkim okruženjima i mogućnost brzog prilagođavanja promjenjivim uvjetima raspoređivanja. S druge strane, budući da pravila raspoređivanja raspored grade iterativno, postižu lošije rezultate od nekih sofisticiranijih algoritama koji se koriste za rješavanje problema raspoređivanja. Također, kvalitetna pravila zahtijevaju puno uloženog vremena i truda kao i znanje stručnjaka u navedenom području. Dodati nedostatak je taj što performanse pravila raspoređivanja uvelike ovise o problemu koji se rješava, što znači da niti jedno pravilo neće biti prikladno za rješavanje svih problema, već je potrebno utvrditi najprikladnije.

5.2. Postavke eksperimenta

Kao što je ranije spomenuto, u ovom i sljedećem poglavlju korišten je MLE prikaz budući da je postigao dobar omjer dobivenih rezultata i vremena izvođenja. Dodatna pogodnost korištena u inicijalizaciji je jednostavno konstruiranje jedinke iz rješenja dobivenog heuristikama. Prikaz će biti testiran različitim načinima konstruiranja početnih rješenja kao i veličina populacije. Dodatno, umjesto korištenja svih dosad korištenih kriterija, korišten je samo kriterij Twt u svrhu smanjivanja složenosti analize rezultata.

U poglavlju su korišteni sljedeći načini inicijalizacije početne populacije:

- potpuno nasumična inicijalizacija (RND)
- inicijalizacija rješenjima generiranim korištenjem 26 pravila raspoređivanja prikladnih za raspoređivanje u dinamičkim okruženjima izrađena od strane raznih stručnjaka (DEX)
- inicijalizacija rješenjima generiranim korištenjem 50 pravila raspoređivanja prikladnih za raspoređivanje u dinamičkim okruženjima izrađena automatski korištenjem genetskog programiranja (DGP)
- inicijalizacija rješenjima generiranim korištenjem 30 pravila raspoređivanja prikladnih za raspoređivanje u statičkim okruženjima izrađena automatski korištenjem genetskog programiranja (SGP)
- inicijalizacija rješenjima generiranim korištenjem 106 pravila raspoređivanja dobivena kombiniranjem prethodno spomenutih heuristika (CMB)

Umjesto do sad korištene početne populacije od 30 jedinki, u ovom poglavlju upotrebljene su različite veličine kako bi usporedila uspješnost različitih kombinacija inicijalizacije početne populacije. U populacijama većim od broja jedinki dobivenih heuristikama, ostatak populacije popunjen je nasumično dobivenim jedinkama. Korištene su veličine populacija od 150, 200 i 500 jedinki, kao i populacije dobivene samo upotrebom heuristika bez dodatnih jedinki dobivenih nasumičnom inicijalizacijom. Vjerojatnost mutacije u svim izračunima iznosi 0.9, kao i u prošlom poglavlju.

Svaki od problema optimiziran je ukupno 30 puta u svrhu točnije analize. Za svaki od izračuna spremljena je najbolja vrijednost te je na osnovu dobivenih vrijednosti izračunata minimalna i maksimalna vrijednost dobrote te njihov medijan. Dodatno, uz navedene vrijednosti, u obzir se uzima i $Tmin$, što predstavlja sveukupno najbolje rješenje dobiveno kombinacijom najboljih rezultata iz svih 30 pokretanja.

5.3. Rezultati

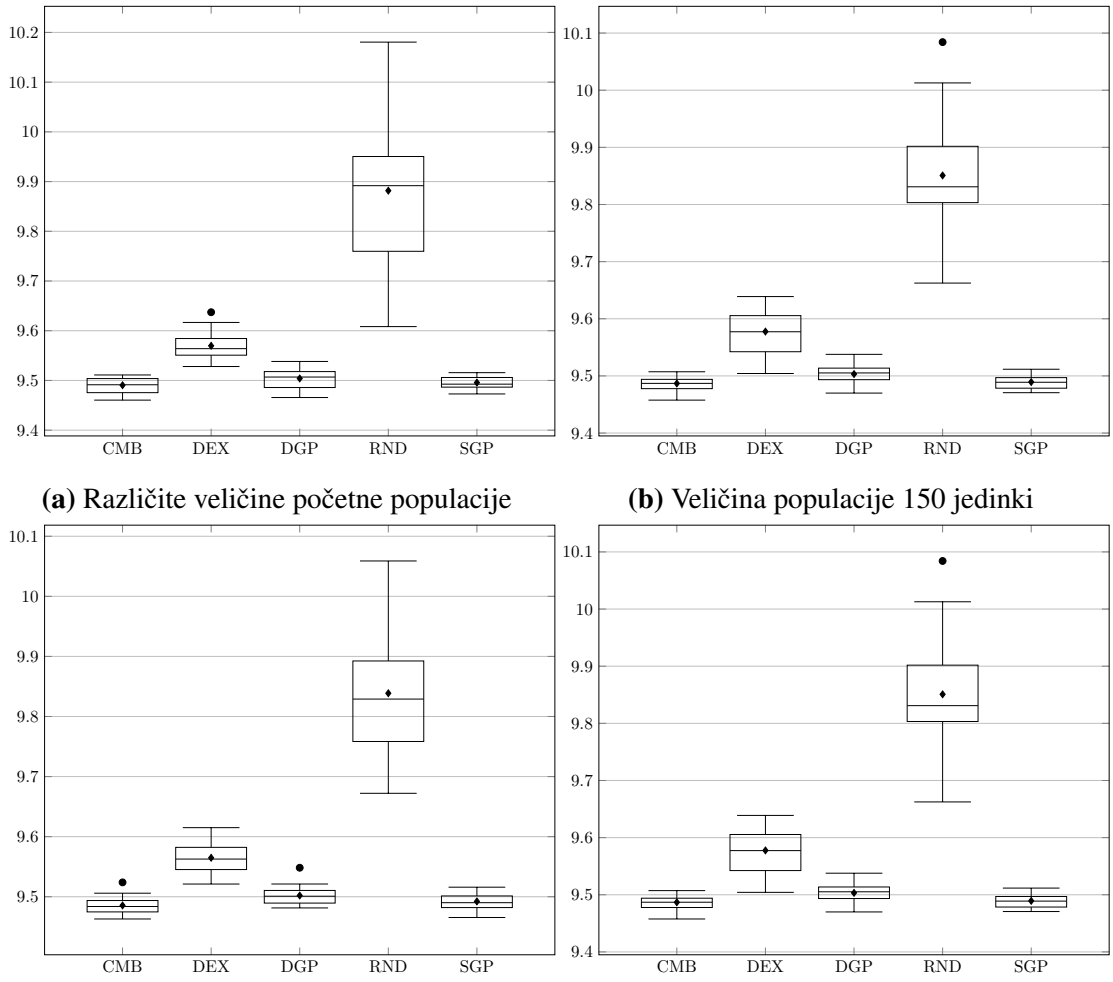
Dobiveni rezultati prikazani su u tablici 5.1. Kao što je već spomenuto, korištene su početne populacije veličine 150, 200 i 500 jedinki, kao i skupina koja koristi minimalne veličine populacije za sve načine inicijalizacije, točnije populacija veličine 26 za DEX način inicijalizacije, 50 za DGP, 30 za SGP, 106 za CMB te 30 za RND. Najbolje rješenje u svakom retku označeno je sivom bojom. Dodatno, na slici 5.1 prikazani su boxplotovi rezultata u svrhu detaljnije usporedbe. Iz tablice je vidljivo da su najbolji rezultati dobiveni CMB načinom inicijalizacije. Navedeno je moguće objasniti činjenicom da CMB način koristi više različitih heuristika, kao i jedinke dobivene nasumičnom inicijalizacijom, što mu poboljšava raznolikost genetskog materijala i povećava prostor pretraživanja.

Tablica 5.1: Rezultati dobiveni testiranjem različitih načina inicijalizacije

Veličina populacije		Način inicijalizacije				
		CMB	DEX	DGP	RND	SGP
~	Tmin	9.439	9.499	9.445	9.485	9.451
	Min	9.46	9.528	9.466	9.608	9.473
	Med	9.492	9.566	9.509	9.9	9.493
	Max	9.511	9.637	9.538	10.18	9.516
150	Tmin	9.435	9.462	9.441	9.475	9.456
	Min	9.458	9.504	9.47	9.662	9.471
	Med	9.488	9.579	9.506	9.834	9.491
	Max	9.507	9.639	9.538	10.084	9.512
200	Tmin	9.431	9.472	9.439	9.475	9.453
	Min	9.463	9.521	9.481	9.672	9.465
	Med	9.485	9.563	9.502	9.831	9.491
	Max	9.524	9.615	9.548	10.059	9.516
500	Tmin	9.43	9.45	9.438	9.492	9.453
	Min	9.457	9.502	9.456	9.605	9.466
	Med	9.485	9.545	9.49	9.821	9.494
	Max	9.511	9.633	9.527	9.942	9.514

Slični rezultati dobiveni su i pomoću SGP načina inicijalizacije koji koristi pravila

raspoređivanja prikladna za statička okruženja. U većini slučajeva dobiveni rezultati su samo malo lošiji od onih dobivenih CMB načinom, što dovodi do zaključka da, gledajući načine inicijalizacije pojedinačno, pravila raspoređivanja prikladna za raspoređivanje u statičkim okruženjima izrađena automatski upotrebom genetskog programiranja postižu najbolje rezultate. Najgori rezultati dobiveni su nasumičnom inicijalizacijom što se može objasniti lošom početnom populacijom koja onemogućava bržu konvergenciju poput ostalih prikaza. Iako je DEX način postigao bolje rezultate od nasumično dobivene početne populacije, rezultati su ipak lošiji od onih dobivenih ostalim načinima. DGP način je postigao dobre rezultate, međutim lošije od onih dobivenih CMB i SGP načinom. Iz boxpolotova na slici 5.1 vidljivo je da je RND način inicijalizacije rezultirao većom raspršenosti rezultata. Ostali prikazi su stabilniji i postižu manje raspršene rezultate.



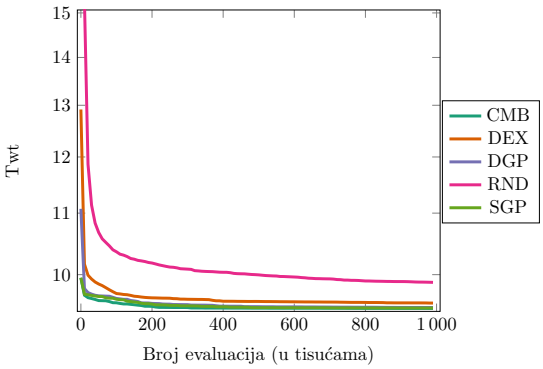
(a) Različite veličine početne populacije

(b) Veličina populacije 150 jedinki

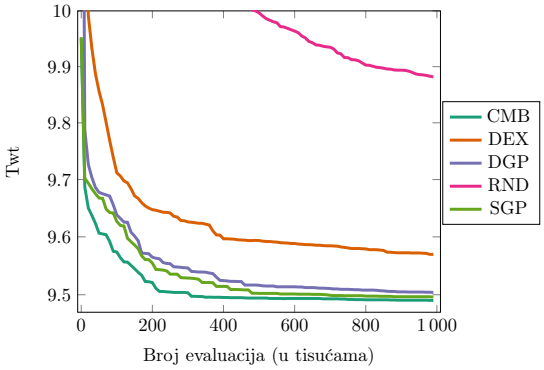
(c) Veličina populacije 200 jedinki

(d) Veličina populacije 500 jedinki

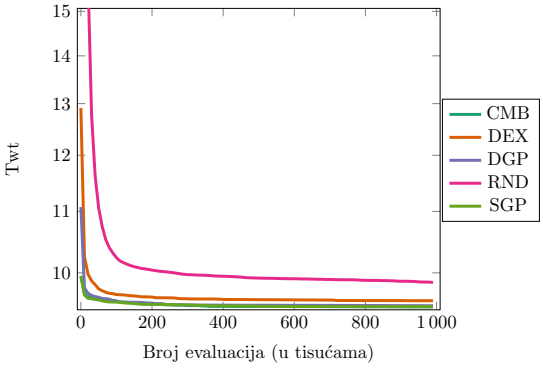
Slika 5.1: Usporedba rezultata dobivenih različitim veličinama populacije



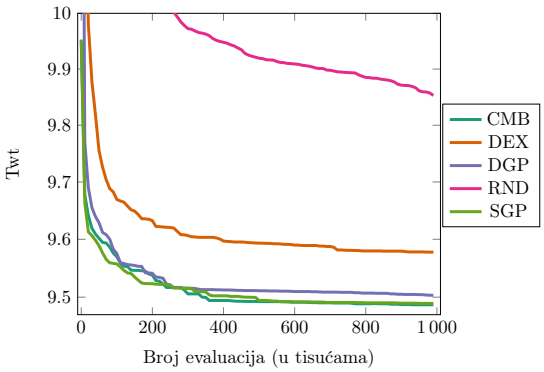
(a) Različite veličine populacije



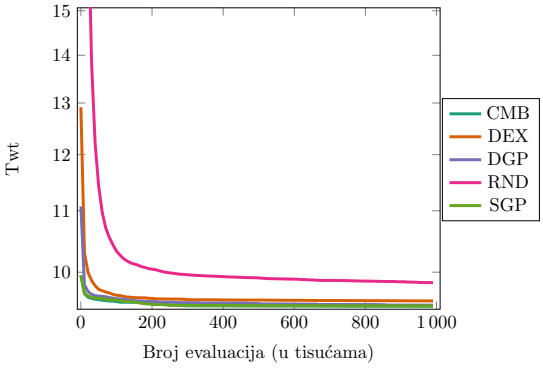
(b) Različite veličine populacije (uvećano)



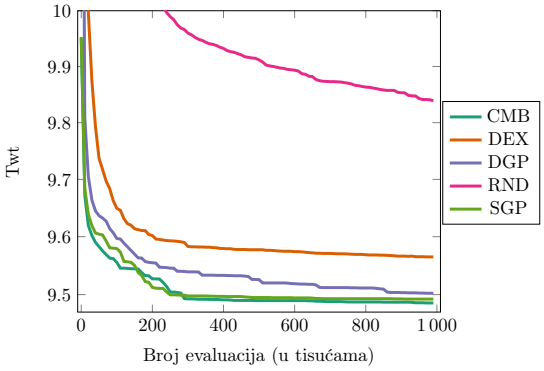
(c) Veličina populacije 150 jedinki



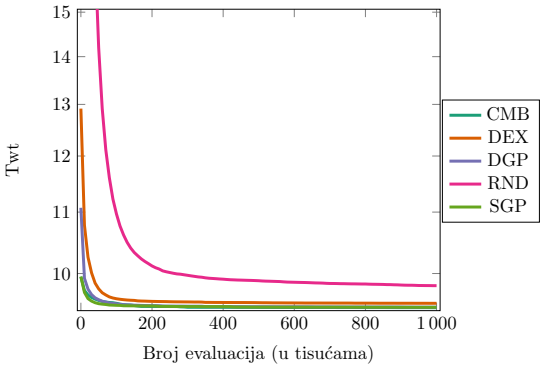
(d) Veličina populacije 150 jedinki (uvećano)



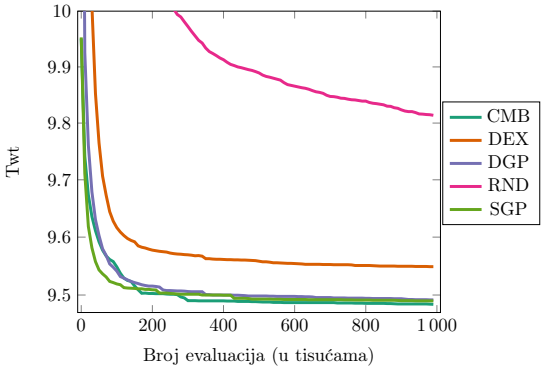
(e) Veličina populacije 200 jedinki



(f) Veličina populacije 200 jedinki (uvećano)



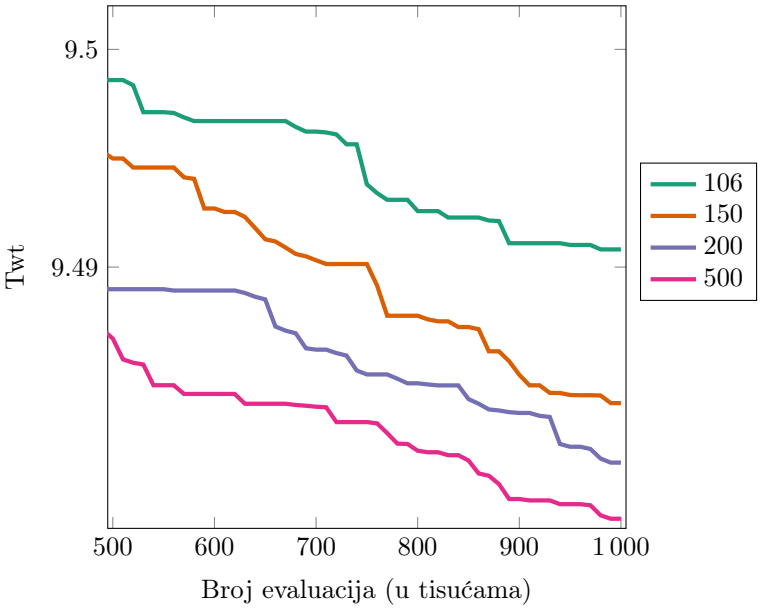
(g) Veličina populacije 500 jedinki



(h) Veličina populacije 500 jedinki (uvećano)

Slika 5.2: Promjene minimalne vrijednosti dobrote tijekom izvođenja za različite veličine populacije

Navedeni zaključci vidljivi su i iz grafova na slici 5.2. Početna populacija dobivena RND načinom inicijalizacije započinje s lošijom prosječnom dobrotom i sporije konvergira od ostalih načina. DEX način također konvergira sporije, ali započinje s kvalitetnijom početnom populacijom. Ostala tri načina inicijalizacije postižu slične rezultate, pogotovo CMB i SGP koji se razlikuju u manjoj mjeri.



Slika 5.3: Usporedba minimalnih vrijednosti dobrote dobivenih različitim veličinama populacije za CMB način inicijalizacije

Na slici 5.3 nalazi se kretanje minimalne dobrote dobiveno CMB načinom inicijalizacije za različite veličine populacije. Iz grafa je vidljivo da povećanje veličine populacije dovodi i do bolje dobrote iako se radi o iznimno malim poboljšanjima. Početna populacija dobivena samo pomoću heuristika postiže najlošije rezultate. Povećanje broja jedinki dobivenih nasumičnom inicijalizacijom uzrokuje poboljšavanje dobrote. Navedeno se može objasniti većom raznolikosti genetskog materijala dobivenom jedinkama koje su inicijalizirane nasumično.

Uzimajući u obzir sve dobivene rezultate, vidljivo je da način inicijalizacije početne populacije može utjecati na rezultate. Iako je nasumična inicijalizacija u prošlom poglavlju postizala dobre rezultate, u ovom poglavlju je pokazano da upotreba pravila raspoređivanja dovodi do kvalitetnijih jedinki i boljih rezultata. Iako su ručno dobivene heuristike pozitivno utjecale na konačna rješenja, vidljivo je da se ipak ne mogu mjeriti s onima dobivenim automatski upotrebom različitih algoritama. Razlog tomu je što i automatski izrađena pravila raspoređivanja postižu bolje rezultate od onih izrađenih ručno. Upotreba pravila raspoređivanja ne usporava algoritam budući da do

rezultata dolaze u kratkom vremenu koje u prosjeku iznosi 0.1 sekunde po pravilu. Navedeno dovodi do zaključka da ih se isplati iskoristiti u inicijalizaciji početne populacije evolucijskih algoritama budući da mogu pozitivno pridonijeti konačnoj kvaliteti rezultata.

6. Rješavanje problema raspoređivanja s dodatnim ograničenjima

Kako bi se algoritam i korišteni MLE prikaz dodatno analizirali, u ovom poglavlju ispituju se različita ograničenja u rješavanju problema raspoređivanja koja se često spominju u literaturi [12] [13]. Dodatno, uspoređen je rad nekoliko algoritama kako bi se utvrdilo koji postiže najbolje rezultate s korištenim prikazom jedinki i dobivenim ograničenjima.

6.1. Korištena ograničenja

Ograničenja koja su korištena u ovom radu su vremena postavljanja, kvarovi strojeva, ograničenje na kojim se strojevima pojedini poslovi mogu izvoditi i ograničenje u redoslijedu izvođenja poslova. U nastavku je detaljnije opisano kako se koje od ograničenja koristi.

6.1.1. Vrijeme postavljanja

Ograničenje vremena postavljanja (OVP) označava vrijeme utrošeno na zamjenu poslova, odnosno prilagodbu stroja na izvođenje idućeg posla. U ovom radu vrijeme se generira za svaki par poslova uniformnom razdiobom u intervalu $[0, 5]$, uz iznimku da prvi posao koji se izvodi na stroju nema vrijeme postavljanja. Cilj je provjeriti sposobnost prilagodbe algoritma dodatnom parametru zbog kojeg može doći do povećanog kašnjenja poslova, a samim time i lošijih rezultata.

6.1.2. Kvarovi strojeva

Ograničenje kvarova strojeva (OKS) označava razdoblja u kojima strojevi nisu dostupni i tijekom kojih ne dolazi do izvršavanja poslova - poslovi moraju čekati popravak stroja kako bi mogli nastaviti izvođenje. Glavna značajka ovog ograničenja je atomičnost izvršavanja posla, to jest poslovi se ne mogu započeti izvršavati prije kvara i nastaviti nakon, već je potrebno pričekati interval vremena u kojem je stroj dostupan dovoljno dugo za izvršenje cjelokupnog posla.

Budući da se kvarovi generiraju nasumično, potrebno je osigurati da se dobiveni intervali nalaze unutar ukupnog vremena izvođenja poslova na svakom od strojeva. U tu svrhu, za svaki od problema izračunata je aproksimacija vremena izvođenja kao

$$SP_i = \frac{\sum_i^m \sum_j^n p_{ij}}{m^2}$$

gdje p_{ij} predstavlja vrijeme izvođenja posla j na stroju i, n broj poslova, a m broj strojeva. Vremena početaka kvarova za svaki od problema generiraju se iz intervala $[0, SP_i]$. Broj prekida i interval mogućeg trajanja zadani su parametrima ograničenja, a ovise o broju strojeva korištenih u primjerima tako da manji broj strojeva rezultira manjim brojem kvarova s dužim trajanjem, a veći broj strojeva rezultira većim brojem kvarova čija je dužina trajanja raznolikija.

6.1.3. Izvođenje poslova

Ograničenje izvođenja poslova (OIP) predstavlja ograničenje koje za svaki stroj definira one poslove koji se na njemu mogu izvoditi. Na taj način simulira se postojanje specifičnih poslova koji se mogu izvoditi samo na podskupu danih resursa, čime se smanjuje mogući prostor pretraživanja.

U svrhu korištenja ovog ograničenja, potrebno je prilagoditi operatore križanja i mutacije te način stvaranja početnih jedinki kako se ne bi generirala rješenja koja ne zadovoljavaju navedeno ograničenje, to jest, potrebno je pobrinuti se da se kod dodjele posla stroj odabire samo iz dozvoljenog skupa. Dodatno, potrebno je pobrinuti se da za svaki posao postoji barem jedan stroj na kojem je dopušteno njegovo izvršavanje kako bi dobiveni rasporedi mogli biti potpuni.

Lista poslova koji se mogu izvršiti na svakom od strojeva generira se nasumično, a njihov broj ovisi o postotku poslova zadanom za svaki od strojeva parametrima generatora. U problemima s manje strojeva ta vrijednost iznosi $[0.7, 0.9]$, a za probleme s više strojeva iznosi $[0.5, 1]$.

6.1.4. Redoslijed izvođenja

Ograničenje redoslijeda izvođenja (ORI) predstavlja ograničenje koje za svaki posao definira prethodnike, to jest poslove koji se moraju izvršiti prije njega.

U svrhu korištenja ovog ograničenja, potrebno je prilagoditi operatore inicijalizacije, križanja i mutacije kako se ne bi generirala rješenja koja ne zadovoljavaju navedeno ograničenje. Potrebno je kod razmještaja posla pobrinuti se da prije njega na istom stroju nije raspoređen nijedan sljedbenik, a iza njega nijedan prethodnik. Dodatno, potrebno je prilagoditi način evaluacije jedinke da u ukupno vrijeme izvršavanja pojedinog posla uračuna i vrijeme koje je posao proveo čekajući da svi njegovi prethodnici završe s izvođenjem.

Ograničenje određuje postotak poslova koji će imati prethodnike. Navedeni parametar za sve primjere iznosi $[0.2, 0.3]$. Ostali korišteni parametri su maksimalni broj prethodnika te maksimalni broj sljedbenika koji rastu s porastom broja poslova u problemima, a iznose između 2 i 10 poslova.

6.2. Postavke eksperimenta

Kao što je i spomenuto, u ovom poglavlju koristi se MLE prikaz budući da je postigao dobar omjer dobivenih rezultata i vremena izvođenja. Dodatna pogodnost korištena u ovom poglavlju je jednostavna prilagodba operatora kako bi dobiveni rezultati zadovoljavali sva navedena ograničenja. Prikaz je testiran s različitim algoritmima i ograničenjima. Dodatno, umjesto korištenja svih kriterija, korišten je se samo kriterij Twt u svrhu smanjivanja složenosti analize rezultata.

Algoritmi koji su uspoređeni u ovom poglavlju su genetski algoritam s turnirskom selekcijom (GAT) koji je korišten u dosadašnjem radu, genetski algoritam s proporcionalnom selekcijom (GAP) i genetsko kaljenje (GKA). Parametri korišteni za GAP su 0.5 za faktor križanja i 10 za selekcijski pritisak, dok je za GKA faktor hlađenja iznosio 0.9 uz energiju vrijednosti 100 i uključen elitizam. U svim izračunima korištena je veličina populacije od 30 jedinki te vjerojatnost mutacije 0.9.

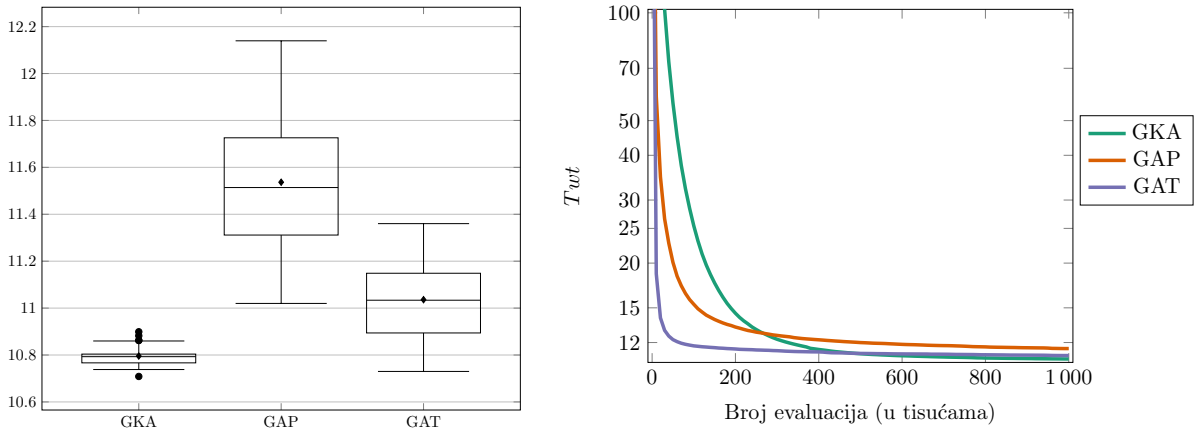
U izračunima je korišten isti skup problema na kojima su nezavisno dodana ograničenja. Svaki od problema optimiziran je ukupno 30 puta u svrhu točnije analize. Za svaki od izračuna spremljena je najbolja vrijednost te je na osnovu dobivenih vrijednosti izračunata minimalna i maksimalna vrijednost dobrote te njihov medijan. Dodatno, uz navedene vrijednosti, u obzir se uzima i $Tmin$ što predstavlja sveukupno najbolje rješenje dobiveno kombinacijom najboljih rezultata iz svih 30 pokretanja. Za

korištena ograničenja izgrađen je jednostavan generator koji ograničenja konstruira po zadanim parametrima.

6.3. Rezultati

Dobiveni rezultati prikazani su u tablici 6.1. Kao što je već naglašeno, korištena su četiri ograničenja te tri algoritma. Najbolje rješenje u svakom retku označeno je sivom bojom. Dodatno, na slikama 6.1, 6.2, 6.3 i 6.4 prikazani su detaljniji rezultati za svako od ograničenja u obliku boxplotova i grafova koji pokazuju kretanje minimalne dobrote kroz broj evaluacija.

Iz tablice je vidljivo da su najbolji rezultati dobiveni za algoritme genetskog kaljenja i genetskog algoritma s turnirskom selekcijom. Genetski algoritam s turnirskom selekcijom u većini slučajeva postiže bolje rezultate kod vrijednosti $Tmin$ što pokazuje da se, pojedinačno promatrano, najbolja rješenja mogu pronaći u rezultatima tog algoritma. Budući da postiže veću raspršenost rezultata, ostale promatrane vrijednosti su bolje u slučaju genetskog kaljenja kod kojeg je raspršenost manja i prosječna kvaliteta rezultata bolja. Do istog zaključka može se doći promatrajući boxplotove svakog od ograničenja.



(a) Rezultati optimiziranja Twt kriterija **(b)** Kretanje minimalne dobrote za Twt kriterij

Slika 6.1: Rezultati dobiveni za ograničenje vremena postavljanja

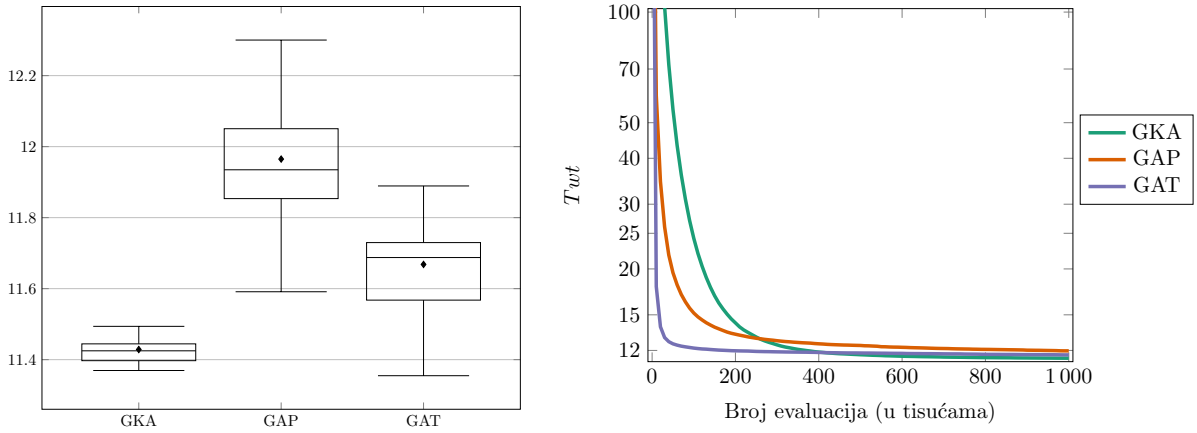
Za razliku od njih, genetski algoritam s proporcionalnom selekcijom postiže lošije rezultate i manju stabilnost. Iz grafova kretanja dobrote može se doći do više zaključaka vezanih uz same algoritme. Genetsko kaljenje započinje s najlošijim rezultatima. Navedeno je rezultat većeg faktora hlađenja što rezultira sporijim hlađenjem i povećanjem prostora pretraživanja čime algoritam sporije konvergira ali na kraju postiže

Tablica 6.1: Rezultati dobiveni testiranjem različitih ograničenja

Ograničenje		Korišteni algoritam		
		GAT	GAP	GKA
Bez ograničenja	Tmin	9.499	9.592	9.511
	Min	9.601	9.883	9.583
	Med	9.846	10.072	9.63
	Max	10.18	10.402	9.726
OVP	Tmin	10.398	10.632	10.527
	Min	10.73	11.02	10.709
	Med	11.036	11.514	10.793
	Max	11.36	12.139	10.899
OKS	Tmin	11.192	11.254	11.242
	Min	11.355	11.591	11.369
	Med	11.689	11.941	11.425
	Max	11.889	12.3	11.494
OIP	Tmin	14.283	14.513	14.305
	Min	14.579	14.91	14.483
	Med	15.146	15.759	14.612
	Max	15.938	16.58	14.756
ORI	Tmin	13.489	14.003	13.614
	Min	14.288	15.08	14.153
	Med	15.736	16.772	14.36
	Max	16.807	18.091	14.615

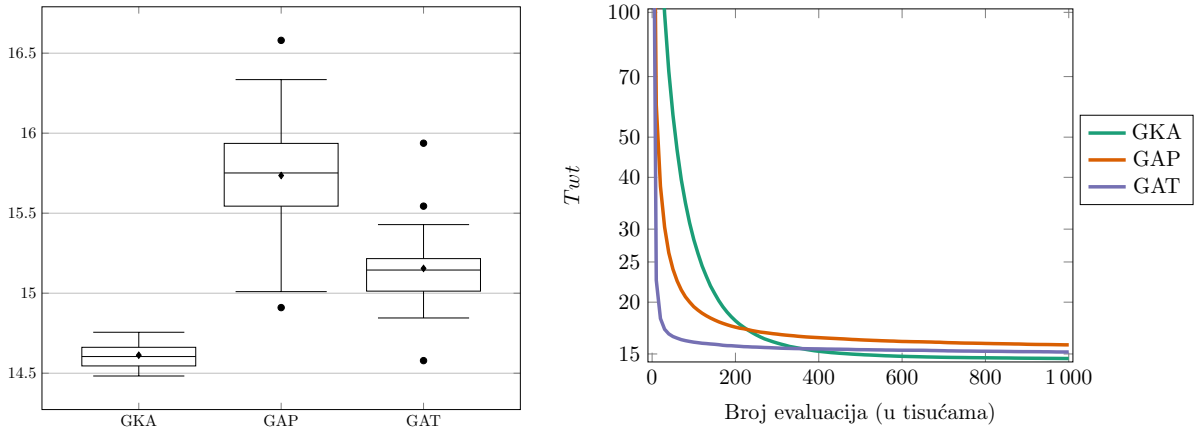
bolje rezultate. Testiranje s nižim faktorom hlađenja dovodilo je do lošijih rezultata budući da s takvim parametrima algoritam brže konvergira i lakše zapinje u lokalnim optimumima. Za razliku od genetskog kaljenja, genetski algoritam s proporcionalnom selekcijom nije uspio dati bolje rezultate ni uz optimizaciju korištenih parametara. Kod genetskog algoritma s turnirskom selekcijom može se primijetiti da brzo dolazi do kvalitetnih rezultata koji se daljnjim računanjem ne poboljšavaju puno.

Iako rezultati ograničenja u određenoj mjeri ovise o korištenim parametrima njihovog generiranja, ograničenje vremena postavljanja ne utječe puno na dobivene rezultate, budući da su vrijednosti vremena generirane iz malog intervala. Uz ograničenje



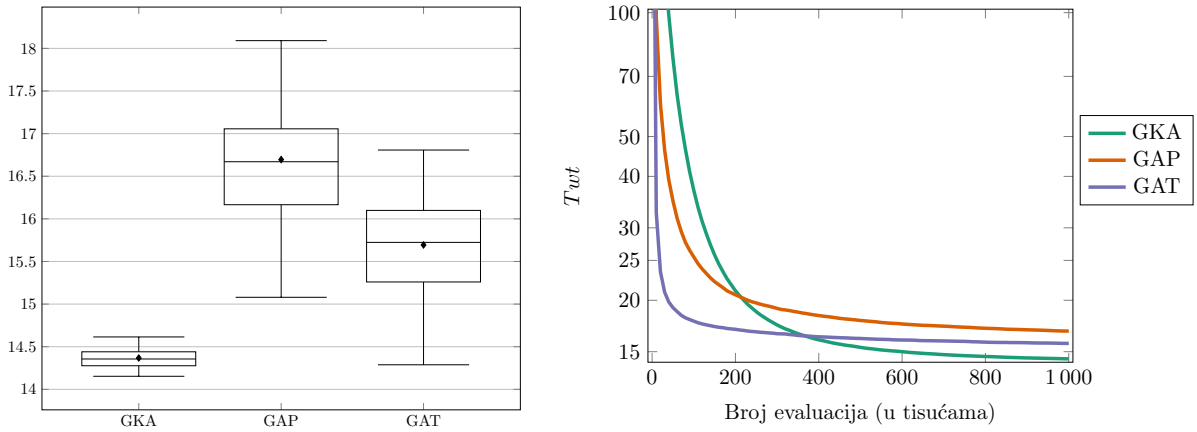
(a) Rezultati optimiziranja Twt kriterija **(b)** Kretanje minimalne dobrote za Twt kriterij

Slika 6.2: Rezultati dobiveni za ograničenje kvarova strojeva



(a) Rezultati optimiziranja Twt kriterija **(b)** Kretanje minimalne dobrote za Twt kriterij

Slika 6.3: Rezultati dobiveni za ograničenje izvođenja poslova



(a) Rezultati optimiziranja Twt kriterija **(b)** Kretanje minimalne dobrote za Twt kriterij

Slika 6.4: Rezultati dobiveni za ograničenje redoslijeda izvođenja

kvarova strojeva postižu se veće vrijednosti za optimirani kriterij. Dodatna nepogodnost optimiziranja ovakvog problema je što u slučajevima prekida, izvođenje posla ne može započeti do pojavljivanja dovoljno dugog intervala za njegovo izvođenje. Navedeno ponekad rezultira dužim vremenom čekanja jer se mogu pojaviti intervali u kojima ne postoji kvar na stroju, ali se posao svejedno ne može započeti izvršavati. Ograničenje izvođenja poslova postiže još veće vrijednosti kriterija, budući da u velikom broju slučajeva nije moguće postići optimalni raspored jer strojevi imaju ograničenja izvođenja poslova. Ovime se smanjuje broj mogućih kombinacija rasporeda što često uključuje i one rasporede koji bi postigli bolje rezultate.

Sličan problem prisutan je i kod ograničenja redoslijeda izvođenja gdje se može primijetiti da su dobiveni rezultati slični kao i kod prethodnog ograničenja. Budući da poslovi imaju prethodnike koji moraju biti zadovoljeni prije njih, smanjuje se broj mogućih kombinacija rasporeda. Dodatan problem kod ovog ograničenja je dugo vrijeme izvršavanja budući da se često moraju provjeravati uvjeti što zahtijeva složenije računske operacije.

7. Zaključak

Cilj ovog rada bio je istražiti problem raspoređivanja u okruženju nesrodnih strojeva korištenjem evolucijskih algoritama. Budući da kvaliteta rješenja evolucijskih algoritama uvelike ovisi o načinu prikaza jedinke koja se koristi za kodiranje rješenje, u prvom dijelu rada cilj je bio implementirati različite prikaze koji se spominju u literaturi i usporediti dobivene rezultate. Pokazano je da prikaz PE koji koristi kodiranje permutacijskim nizom postiže najbolje rezultate za sva četiri promatrana kriterija, ali ujedno i najduže vrijeme izvođenja. Za razliku od njega, prikaz PEML postiže najgore rezultate budući da koristi kodiranje permutacijskim nizom s listom strojeva, zbog čega zahtijeva dodatni genotip. Budući da se genotipi optimiraju neovisno jedan od drugog, algoritmu je teže doći do kvalitetnih rezultata. Za daljnji rad izabran je prikaz MLE koji koristi kodiranje grupama poslova. Navedeni prikaz je postigao dobar omjer rezultata i vremena izvođenja, i ujedno omogućuje jednostavnu prilagodbu operatora potrebnih za daljnji rad.

U drugom dijelu uspoređeni su različiti načini inicijalizacije početne populacije u svrhu poboljšavanja rada algoritma korištenjem bolje početne populacije. Pokazano je da početna populacija koja koristi kombinaciju različitih načina inicijalizacije postiže najbolje rezultate budući da tako dolazi do veće raznolikosti genetskog materijala i povećavanja prostora pretraživanja. Navedene inicijalizacije omogućile su bržu konvergenciju i bolju kvalitetu dobivenih rezultata.

U trećem dijelu cilj je bio uvesti različita ograničenja kako bi se problem i korišteni prikaz dodatno analizirali, ali ujedno usporediti rad nekoliko različitih algoritama kako bi se utvrdilo koji postiže najbolje rezultate. Pokazano je da genetsko kaljenje rezultira najkvalitetnijim rezultatima u slučajevima korištenja većeg faktora hlađenja, što mu omogućuje širi prostor pretraživanja. U navedenim uvjetima algoritam započinje rad s jedinkama lošije kvalitete te sporije konvergira, ali na kraju postiže bolje rezultate. Pokazano je da se algoritam može dobro prilagoditi navedenim ograničenjima, no nedostatak je potreba prilagodbe operatora što za pojedina ograničenja može biti zahtjevno. Najmanji problem predstavljalo je ograničenje vremena postavljanja,

dok su najgori rezultati dobiveni za ograničenje izvođenja poslova budući da navedeno ograničenje smanjuje broj mogućih kombinacija rasporeda što često uključuje i one rasporede koji bi postigli bolje rezultate.

U radu je pokazano da je problem raspoređivanja složen problem za koji nije moguće odrediti optimalni algoritam i parametre koji bi postizali najbolje rezultate za sve promatrane probleme. Iako problem raspoređivanja spada u NP-teške probleme, odnosno probleme za koje nije moguće u polinomijalnoj složenosti pronaći optimalno rješenje, u radu je dokazano da se upotrebom različitih algoritama i poboljšanja može doći do rješenja koja su dovoljno dobra za većinu upotreba. Ujedno je pokazano da su navedeni algoritmi dovoljno prilagodljivi kako bi ih se moglo primijeniti na široki spektar problema raspoređivanja i time omogućiti brže i lakše postizanje kvalitetnih rezultata.

LITERATURA

[1] M. L. Pinedo, Scheduling: Theory, algorithms, and systems: Fourth edition, Vol. 9781461423614, Springer US, Boston, MA, 2012.

[2] J. Y.-T. Leung, Handbook of scheduling : algorithms, models, and performance analysis, Chapman & Hall/CRC, Boca Raton, Fla., 2004.

[3] A. Costa, F. A. Cappadonna, S. Fichera, A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times, The International Journal of Advanced Manufacturing Technology 69 (9-12) (2013) 2799–2817. doi:10.1007/s00170-013-5221-5.

[4] M. Đurasević, D. Jakobović, Comparison of solution representations for scheduling in the unrelated machines environment, in: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2016, pp. 1336–1342. doi:10.1109/MIPRO.2016.7522347.

[5] J. C. Bean, Genetic algorithms and random keys for sequencing and optimization, ORSA Journal on Computing 6 (2) (1994) 154–160. doi:10.1287/ijoc.6.2.154.

[6] S. Balin, Non-identical parallel machine scheduling using genetic algorithm, Expert Systems with Applications 38 (6) (2011) 6814–6821. doi:10.1016/j.eswa.2010.12.064.

[7] E. Vallada, R. Ruiz, A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times, European Journal of Operational Research 211 (3) (2011) 612–622. doi:10.1016/j.ejor.2011.01.011.

[8] A. Kuczapski, M. Micea, L. Maniu, V. Cretu, Efficient generation of near optimal inital populations to enhance genetic algorithms for job-shop scheduling, Information technology and control 39 (1) (2010) 32–37.

[9] L. Fanjul-Peyro, R. Ruiz, Iterated greedy local search methods for unrelated parallel machine scheduling, European Journal of Operational Research 207 (1) (2010) 55–69.

[10] L. Fanjul-Peyro, R. Ruiz, Size-reduction heuristics for the unrelated parallel machines scheduling problem, Computers & Operations Research 38 (1) (2011) 301–309.

[11] L. Perdigão Cota, M. Nohra Haddad, M. Jamilson Freitas Souza, V. Nazário Coelho, Hybrid GRASP Heuristics to Solve an Unrelated Parallel Machine Scheduling Problem with Earliness and Tardiness Penaltie, Electrical Notes in Theoretical Computer Science 302 (1) (2014) 53–72.

[12] W.-J. Yin, M. Liu, C. Wu, Learning Single-Machine Scheduling Heuristics Subject to Machine Breakdowns with Genetic Programming, Department of Automation, Tsinghua University, Beijing.

[13] M. Pfund, J. W. Fowler, A. Gadkari, Y. Chen, Scheduling jobs on parallel machines with setup times and ready times, ScienceDirect 54 (2007) 764–782.

Rješavanje problema raspoređivanja u okruženju nesrodnih strojeva korištenjem evolucijskih algoritama

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. Jedna od metoda rješavanja tog problema je upotreba evolucijskih algoritama. U radu se opisuje primjena evolucijskih algoritama na rješavanje problema raspoređivanja u okruženju nesrodnih strojeva. Uspoređuju se različiti prikazi jedinki korišteni u literaturi kako bi se pronašao onaj koji postiže najbolje rezultate za promatrane kriterije. Dodatno, upotrebom različitih pravila raspoređivanja u inicijalizaciji početne populacije pokušava se postići poboljšavanje rezultata i njihova brža konvergencija. Konačno, različiti algoritmi su primijenjeni za rješavanje problema raspoređivanja s dodatnim ograničenjima te je analiziran utjecaj svakog ograničenja na dobivene rezultate.

Ključne riječi: raspoređivanje, okruženje nesrodnih strojeva, evolucijski algoritmi, pravila raspoređivanja, ograničenja u raspoređivanju

Solving Scheduling Problems in the Unrelated Machines Environment by Using Evolutionary Algorithms

Abstract

Scheduling is a well known NP-hard optimization problem, which means an algorithm that finds the optimal solution within reasonable time constraints does not exist. One of the methods for solving this problem is using evolutionary algorithms. This thesis describes the application of evolutionary algorithms in solving the problem of scheduling in the unrelated machines environment. Different solution representations used in the literature are compared to find the one that achieves the best results for the observed criteria. In addition, various dispatching rules in initialization of the population are tried to achieve improved results and their faster convergence. Finally, various algorithms have been applied to solve the problem of scheduling with constraints and the impact of each constraint on the obtained results is analyzed.

Keywords: scheduling, unrelated machines environment, evolutionary algorithms, dispatching rules, scheduling constraints