SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1967

Automatski razvoj pravila raspoređivanja za probleme raspoređivanja s ograničenjima

Kristijan Jaklinović

Zagreb, lipanj 2019.

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

Zagreb, 8. ožujka 2019.

DIPLOMSKI ZADATAK br. 1967

Pristupnik: **Kristijan Jaklinović (0036479316)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Automatski razvoj pravila raspoređivanja za probleme raspoređivanja s ograničenjima**

Opis zadatka:

Proučiti problem raspoređivanja u okruženju nesrodnih strojeva i dodatna ograničenja nad problemom koja se mogu javiti. Prilagoditi postojeće ručno izrađeno pravilo raspoređivanja za rješavanje problema s dodatnim ograničenjima. Prilagoditi hiperheurustički model za rješavanje problema raspoređivanja uz sljedeća ograničenja: trajanje postavljanja, prekid rada strojeva, ograničenje redoslijeda izvođenja poslova i ograničenje izvođenja poslova po strojevima. Isprobati različite inačice prilagodbe za zadani problem definiranjem novih terminalnih čvorova ili algoritama za izradu rasporeda. Usporediti učinkovitost ostvarenih postupaka s postojećim rješenjima iz literature. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 28. lipnja 2019.

Mentor:
Prof. dr. sc. Domagoj Jakobović

Djelovođa:
Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:
Doc. dr. sc. Marko Čupić

Zahvaljujem se svojoj obitelji na podršci tijekom studiranja i roditeljima na svemu što su mi pružili u životu.

Hvala mentoru prof. dr. sc. Domagoju Jakobovića na pružanoj pomoći tijekom godina studiranja te hvala dr. sc. Marku Đuraseviću na savjetima, konzultacijama, trudu i uloženom vremenu kako bi me usmjerio na pravi put tijekom izrade ovog rada.

SADRŽAJ

1. Uvod

Raspoređivanje je optimizacijski problem u kojem je skup poslova potrebno rasporediti na skup resursa na način da minimizira neki kriterij kao što je npr. kašnjenje poslova u sustavu [5]. Odmah primjećujemo da je takav problem sveprisutan, od velikih industrija [4], školskih rasporeda [9] pa do raspoređivanja aviona po pistama [2; 1]. Cilj je svaki posao u sustavu dodijeliti jednom stroju na kojem će se taj posao izvesti do kraja. U okruženju nesrodnih strojeva trajanje obavljanja posla ovisi o stroju na kojem se taj posao izvršava.

Većina problema raspoređivanja su NP-teški problemi što znači da ih nije moguće optimalno riješiti u polinomijalnom vremenu. Za rješavanje takvih problema najčešće se koriste aproksimativne metode, konkretnije, poboljšavajuće metaheuristike. Pod metaheurističke metode spadaju razni evolucijski algoritmi, a u ovom radu bit će opisan pristup problemu raspoređivanja koristeći genetsko programiranje kao poseban slučaj evolucijskog algoritma. Glavni nedostatak metaheurističkih postupaka je to što najčešće nisu primjenjivi kada je raspored potrebno brzo izgraditi i kada nisu dostupne sve informacije o problemu. Zato koristimo pravila raspoređivanja, a iz razloga što ih je teško ručno izraditi koristimo genetsko programiranje kako bi taj proces automatizirali. Ovaj rad biti će fokusiran na izradu pravila raspoređivanja uz različita ograničenja kao što su ograničenje vremenom postavljanja, privremenim prekidom rada stroja, prikladnim strojevima te redoslijedom izvođenja poslova.

Rad je strukturiran na način da je prvo objašnjen sam problem koji rješavamo te različiti pristupi rješavanja problema. Zatim, u trećem poglavlju slijedi uvid u način rada genetskog programiranja te kako se primjenjuje na problem raspoređivanja. Četvrto poglavlje definira ograničenja koja mogu biti prisutna u sustavu koja dodatno otežavaju problem raspoređivanja te daje ocjenu prikladnosti automatske izrade pravila raspoređivanja za razmatrana ograničenja. U posljednjem, petom poglavlju, iznesen je kratki pregled rada te zaključak na temelju postignutih rezultata.

2. Raspoređivanje u okruženju nesrodnih strojeva

Da bi se riješio problem raspoređivanja u okruženju nesrodnih strojeva potrebno je za svaki od *n* poslova odrediti jedan od *m* strojeva na kojem će se posao izvoditi kao i redoslijed izvođenja poslova na svakom stroju. Bitno je napomenuti da se posao izvršava u potpunosti na stroju kojem je pridodijeljen te se istovremeno na stroju može izvoditi samo jedan posao. Svaki posao *i* koji se izvodi na stroju *j* opisan je s nekoliko svojstava navedenih u **tablici 2.1**.

OZNAKA	OPIS
p_{ij}	vrijeme izvođenja posla j na stroju i
r_j	vrijeme kada posao j ulazi u sustav
d_j	vremensko ograničenje završetka izvršavanja posla j
w_j	težinska važnost posla j

Tablica 2.1: Svojstva problema raspoređivanja

Nakon izrade rasporeda za svaki posao mogu se izračunati određeni kriteriji (**tablica 2.2**).

OZNAKA	OPIS
C_j	vrijeme kada posao j završi s izvođenjem
F_j	vrijeme koje je posao j proveo u sustavu ($F_j = C_j - r_j$)
T_j	vrijeme izvršavanja posla j nakon vremenskog ograničenja d_j
U_j	da li je 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

Kako bi se mogla usporediti različita rješenja na temelju njihove kvalitete definirani

su neki kriteriji testiranja (**tablica 2.3**).

OZNAKA	OPIS
C_{max}	najkasnije vrijeme završetka svih poslova, $C_{max} = max_j(C_j)$
Ft	suma svih vremena koje su poslovi proveli u sustavu, $Ft = \sum_{j=1}^{n} F_j$
Twt	težinska suma kašnjenja svih poslova, $Twt = \sum_{j=1}^{n} w_j T_j$
Uwt	težinska suma poslova koji kasne, $Uwt = \sum_{j=1}^{n} w_j U_j$

Tablica 2.3: Kriteriji testiranja

Kriterij testiranja korišten u ovom radu je Twt (težinska suma kašnjenja svih poslova).

2.1. Klasifikacija problema raspoređivanja

Problem raspoređivanja može se podijeliti u nekoliko klasa s obzirom na njegova svojstva. Prva podjela je u ovisnosti o dostupnosti parametara problema. Razlikujemo probleme u kojima su informacije dostupne od početka (npr. informacije o svim poslovima koji će doći u sustav) te probleme gdje su te informacije dostupne tek kada posao dođe u sustav. Sljedeća podjela je s obzirom na vrijeme konstruiranja rasporeda. Ako je raspored konstruiran prije izvršavanja sustava tada je riječ o statičkom raspoređivanju. Ako se raspored gradi tijekom izvršavanja sustava riječ je o dinamičkom raspoređivanju. Ovaj rad bit će usredotočen na dinamičko raspoređivanje gdje informacija o sustavu nisu poznate unaprijed te se raspored gradi paralelno s radom sustava.

2.2. Ograničenja

Problem također može definirati i dodatna ograničenja u sustavu. Neka od njih mogu biti vremena postavljanja, privremeni prekid rada stroja, redoslijed izvođenja poslova te ograničenje u kojem se svi poslovi ne mogu izvršiti na bilo kojem stroju. Ograničenja uvode dodatnu kompleksnost pri rješavanju takvog problema stoga je potrebno dodatno prilagoditi strategiju rješavanja.

2.3. Načini rješavanja problema raspoređivanja

Klasa problema uvelike utječe pri odabiru algoritma za rješavanje problema. Npr. u dinamičkim sustavima obično se koriste algoritmi koji iterativno grade rješenje. Dok se kod statičkih najčešće koriste metode heurističkog pretraživanja.

Najčešće se u stvarnom svijetu pojavljuju problemi dinamičkog sustava gdje unaprijed nisu poznate informacije o sustavu. Kod takvih problema potrebno je donositi odluke za vrijeme izvođenja sustava na temelju samo trenutno dostupnih informacija. Algoritmi koji rješavaju takve probleme moraju imati kratko vrijeme izvođenja i najčešće spadaju u skupinu heuristika. Jedan od takvih algoritama je i ATC (*Apparent Tardiness Cost*).

2.4. ATC

ATC je heuristika koja koja gradi raspored za vrijeme izvođenja sustava. Razvili su ju *Vepsalainen* i *Morton* (1987. godine) [8] te *Ow* i *Morton* (1989.) [8]. ATC heuristika koristi $ATCpravilo$ (2.1) za računanje prioriteta posla j na stroju i u trenutku t [8].

$$I_{ij}(t) = \frac{w_j}{p_{ij}} \exp\left[- \frac{max(d_j - p_{ij} - t, 0)}{k\overline{p}} \right] \tag{2.1}$$

OZNAKA	OPIS
t	trenutno vrijeme
j	posao
$I_j(t)$	prioritet posla j u trenutku t
w_j	težina posla (signifikantnost)
p_{ij}	vrijeme izvođenja posla j na stroju i
d_j	vremensko ograničenje posla
$\overline{p}$	prosječno trajanje izvođenja neraspoređenih poslova
k	parametar

Tablica 2.4: Značenje oznaka izraza 2.1

Ako vrijednost parametra k teži u beskonačnost tada ATC pravilo postaje $WSPT$ (*Weighted Shortest Processing Time*). S druge strane, ako vrijednost parametra k teži prema nuli i postoji najviše jedan posao koji kasni, ATC pravilo postaje LSR (*Least Slack Remaining*). Ako postoji više poslova koji kasne ATC pravilo postaje $WSPT$

poslova koji kasne. Odgovarajuća vrijednost parametra k je kompromis između navedene dvije krajnosti, a najčešće se određuje eksperimentalno [8].

ATC algoritam za svaki neraspoređeni posao j računa prioritet u trenutku t na stroju i. Pronalazi se posao s najvećim prioritetom te se određuje stroj na kojem će se taj posao najranije izvršiti. Ako je taj stroj trenutno dostupan posao j mu se dodjeljuje. Taj proces se ponavlja dokle god ima neraspoređenih poslova u sustavu. Nakon što se izgradi takav raspored, određuje se njegova dobrota nekim od kriterija testiranja (**tablica 2.3**) [8].

Algoritam 1 ATC

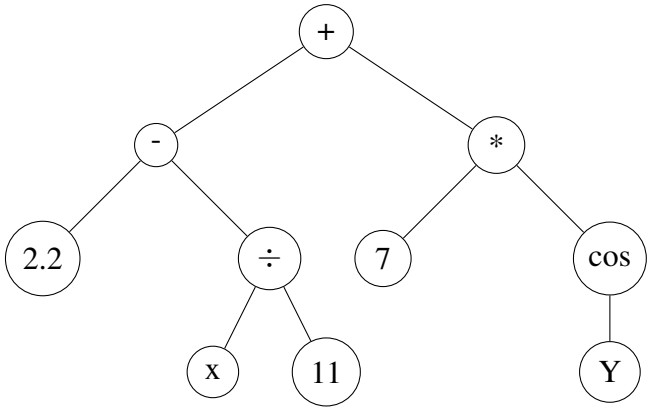
1: $n \leftarrow$ broj poslova
2: $m \leftarrow$ broj strojeva
3: $raspored[m][n]$ = []
4: **while** broj neraspoređenih poslova $\neq 0$ **do**
5: $t \leftarrow$ trenutno vrijeme
6: $nerasporedeniPoslovi \leftarrow$ dostupni neraspoređeni poslovi u sustavu
7: **while** $nerasporedeniPoslovi$ nije prazan **do**
8: // pronađi posao s najboljim prioritetom
9: **for all** $j \in nerasporedeniPoslovi, i \in m$ **do**
10: $prioritet \leftarrow I_{ij}(t)$ ▷ 2.1
11: **end for**
12: $posao \leftarrow$ posao s najvećim prioritetom
13: $stroj \leftarrow$ stroj na kojem će se $posao$ najranije izvršiti
14: $raspored[stroj] \xleftarrow{\text{ako je stroj dostupan}} posao$
15: **end while**
16: **end while**
17: $fitnes \leftarrow$ evaluiraj $raspored$

3. Genetsko programiranje

Genetsko programiranje (skraćeno GP) je automatiziran optimizacijski postupak razvoja računalnih programa ili funkcija, čija je namjena rješavanje većinom složenih problema iz područja računarstva, ali i problema na koje svakodnevno nailazimo [6]. Temelji se na teoriji genetskih algoritama. Pomoću evolucijskih algoritama pretražuje se prostor rješenja tražeći programe, tj. funkcije koje najbolje rješavaju zadani problem. Jedan od najpoznatijih takvih problema je simbolička regresija gdje se pomoću genetskog programiranja traži funkcija koja najbolje opisuje ulazne empirijske podatke.

3.1. Prikaz jedinke

Svaki se računalni program i funkcija može prikazati pomoću stabla gdje unutarnji čvorovi stabla imaju ulogu operatora, a listovi ulogu operanada. Pritom su skupovi operatora i operanada (terminala) unaprijed definirani.



$$(2.2 - \tfrac{x}{11}) + (7 + \cos y)$$

Slika 3.1: Primjer jedinke genetskog programiranja

3.2. GP kao poseban slučaj evolucijskog algoritma

Genetsko programiranje koristi iterativni postupak koji jasno sadrži elemente evolucijskog programiranja kao što su:

1. generacija inicijalne populacije
2. primjena genetskih operatora nad jedinkama populacije
3. analiza, tj. evaluacija populacije.

Algoritam 2 Evolucijski algoritam

```
1: populacija ← generiraj početnu populaciju
2: evaluiraj početnu populaciju
3: while nije zadovoljen kriterij zaustavljanja do
4:     odaberi jedinke koje će sudjelovati u genetskim operacijama
5:     primjeni genetske operatore
6:     evaluiraj nova rješenja
7: end while
```

3.3. Populacija

Populacija je skup jedinki iste vrste koje postoje u istom prostoru. Svaka jedinka predstavlja potencijalno rješenje problema. Veličina populacije je konstanta kroz iteracije algoritma, a jedinke se mijenjaju korištenjem genetskih operatora.

3.4. Genetski operatori

Svaku iteraciju algoritma nastoji se stvoriti bolje jedinke koristeći operatore križanja i mutacije. U procesu križanja sudjeluju dvije jedinke, odabrane selekcijskim operatorom, na temelju kojih se stvara njihov potomak nekim od operatora križanja. Najčešće je riječ o zamjeni dva podstabla jedinki. U procesu mutacije sudjeluje jedna jedinka koja se mutira nekim od operatora mutacije. Najčešća je zamjena podstabla unutar jedinke.

3.5. Funkcija dobrote

Kako bi se mogla usporediti kvaliteta dobivenih rješenja potrebno je evaluirati jedinke funkcijom dobrote. Definiranje funkcije dobrote jedan je od ključnih problema genetskog programiranja. Ona mora biti što jednostavnija, a pritom dobro opisati kvalitetu rješenja danog problema. Razlog tome je što ona ima utjecaj na čitav algoritam jer se nastoji osigurati da bolje jedinke prežive i da se kombiniraju, dok bi loše trebale postupno izumirati.

3.6. Kriterij zaustavljanja

Algoritam obustavlja izvršavanje kada zadovolji neki od kriterija zaustavljanja. Neki od kriterija zaustavljanja su broj iteracija (generacija), vrijednost funkcije dobrote, vremensko ograničenje i slično.

3.7. Primjena GP-a na problem raspoređivanja

Genetsko programiranje moguće je primijeniti i na problem raspoređivanja [3]. Prvi korak rješavanja tog problema je definirati skup operatora (**tablica 3.1**) i terminala (**tablica 3.2**) pomoću kojih ćemo izgraditi funkciju prioriteta poslova. Pritom je potrebno odabrati sve relevantne terminale koji pružaju bitne informacije o problemu.

Definirani terminali su većinom za jednostavne parametre poslova (npr. pt, $pmin$...) dok su neki i složeniji te ovise o trenutnom stanju sustava (SL, MR). Ideja iza složenih terminalnih čvorova je olakšati evoluciju genetskom programiranju tako da ne mora sam razvijati te složene terminale koristeći one jednostavnije.

OZNAKA	OPIS
+	operator binarnog zbrajanja
-	operator binarnog oduzimanja
*	operator binarnog množenja
÷	operator binarnog dijeljenja
pos	unarni operator, $pos(x) = max(x, 0)$

Tablica 3.1: Skup operatora

Sljedeći korak je definirati pravilo raspoređivanja. Ono se sastoji od dva dijela, prioritetne funkcije te sheme za izradu rasporeda. Prioritetna funkcija će se izgraditi

OZNAKA	OPIS
pt	vrijeme izvođenja posla j na stroju i (p_{ij})
$pmin$	minimalno vrijeme izvođenja na svim strojevim
$pavg$	srednje vrijeme izvođenja na svim strojevima
PAT	vrijeme dok stroj s minimalnim vremenom izvođenja ne postane slobodan
MR	vrijeme dok trenutni stroj ne postane slobodan
age	vrijeme koje je posao proveo u sustavu
dd	vremenski trenutak kad posao mora završiti (d_j)
w	težina posla
SL	dopuštena odgoda uz brzinu dotičnog stroja, $-max(d_j - p_{ij} - t, 0)$

Tablica 3.2: Skup terminala

pomoću genetskog programiranja koristeći odabrane terminale i operatore. Shemu za izradu rasporeda definiramo sami. Svaki put kada je potrebno donijeti odluku o raspoređivanju shema za izradu rasporeda će odrediti za koje poslove i strojeve će se izračunati prioritet, te će na temelju izračunatih prioriteta donijeti odluku koji posao će se idući rasporedi i na koji stroj. Primjer sheme za izradu rasporeda dan je algoritmom 3.

Algoritam 3 Shema za izradu rasporeda

```
1:  while postoje neraspoređeni poslovi do
2:      čekaj dok posao ne postane dostupan ili se ne završi
3:      for all dostupne poslove i strojeve do
4:          izračunaj prioritet π_ij posla j na stroju i
5:      end for
6:      for all dostupne poslove do
7:          odredi najbolji stroj (onaj za koji se postiže najbolji iznos prioriteta π_ij)
8:      end for
9:      while postoje poslovi čiji je najbolji stroj dostupan do
10:         odredi najbolji prioritet takvih poslova
11:         rasporedi posao s najboljim prioritetom
12:     end while
13: end while
```

4. Rješavanje problema raspoređivanja s dodatnim ograničenjima

Kako bi se analizirala prikladnost predloženog rješenja genetskim programiranjem, ono će biti isprobano nad različitim problemima raspoređivanja koji sadrže dodatno definirana ograničenja. Dobiveni rezultati bit će uspoređeni s rezultatima ATC heuristike prilagođene definiranim ograničenjima.

4.1. Korištena ograničenja

U ovom radu korištena ograničenja biti će vrijeme postavljanja, privremeni prekid u radu strojeva, redoslijed izvođenja poslova te ograničenje u kojem se poslovi ne mogu izvoditi na bilo kojem stroju.

4.2. Postavke eksperimenata

Kako bi dobili uvid u koliko je dobro predloženo rješenje koristeći genetsko programiranje generirana su dva skupa, skup za testiranje i skup za učenje, svaki s po šezdeset različitih instanci problema. Skup za učenje korišten je za izgradnju prioritetne funkcije, dok su na skupu za testiranje korištene najbolje prioritetne funkcije dobivene na skupu za učenje, kako bi se ocijenila njihova kvaliteta. Za svaku instancu problema izračunat je kriterij testiranja Twt (**tablica 2.3**), a ukupna suma svih nam daje konačanu vrijednost kriterija testiranja Twt_{uk} (4.1).

$$Twt_{uk} = \sum_{i=0}^{i<60} Twt_i \tag{4.1}$$

U svakom eksperimentu algoritam je pokrenut trideset puta s parametrima definiranima u tablici 4.1.

PARAMETAR	VRIJEDNOST
selekcija	turnirska eliminacijska
veličina turnira	3
veličina populacije	1000
vjerojatnost mutacije	0.3
kriterij zaustavljanja (broj generacija)	80
maksimalna dubina stabla	5
minimalna dubina stabla	1

Tablica 4.1: Parametri algoritma

Konačan zaključak o uspješnosti eksperimenta biti će usporedba najbolje vrijednosti Twt_{uk} dobivena prilagođenom ATC heuristikom i medijana Twt_{uk} dobivenih u trideset pokretanja algoritma genetskog programiranja. Dodatno, bit će prikazani i rezultati dobiveni genetskim algoritmom za statičku inačicu problema koji pokazuju koliko dobra rješenja je upće moguće dobiti za razmatrane probleme.

4.3. Vrijeme postavljanja

Vrijeme postavljanja (OVP) označava vrijeme koje je potrebno stroju za prilagodbu za izvođenje idućeg posla. U ovom radu vrijeme se generira za svaki par poslova uniformnom razdiobom u intervalu od nula do 5 ([0,5]), uz iznimku da prvi posao koji se izvodi na stroju nema vrijeme postavljanja [8].

4.3.1. Prilagodba ATC heuristike - ATCS

ATC pravilo potrebno je prilagoditi navedenom ograničenju kako bi kvaliteta rasporeda bila bolja, tj. kako bi ukupno kašnjenje u sustavu bilo manje. Novo dobiveno pravilo definirano izrazom 4.2 naziva se ATCS [8].

$$I_{ij}(t,l) = \frac{w_j}{p_{ij}} \exp\left[-\frac{max(d_j - p_{ij} - t, 0)}{k_1\overline{p}}\right] \exp\left[\frac{s_{lj}}{k_2\overline{s}}\right] \tag{4.2}$$

ATCS uvodi u klasično ATC pravilo novi član u kojem se uzima u obzir vrijeme postavljanja danog posla na odabrani stroj kao i prosječna vrijednost svih vremena

OZNAKA	OPIS
l	indeks posljednje obavljenog posla na stroju
s_{lj}	vrijeme postavljanja posla j nakon posla l
$\overline{s}$	srednja vrijednost vremena postavljanja
k_2	parametar

Tablica 4.2: Značenje novih oznaka izraza 4.2

postavljanja. Na taj način nastoji se dodatno staviti naglasak na one parove poslova i strojeva koji imaju manja vremena postavljanja. ATCS pravilo uvodi i novi parametar k_2 koji se, isto kao parametar k_1 klasičnog ATC pravila, određuje eksperimentalno te njegova vrijednost ovisi o problemu koji rješavamo. Ono je identično ATC algoritmu (**algoritam 1**). Jedina razlika je u što se $prioritet$ ne računa prema izrazu 2.1 već prema 4.2.

4.3.2. Prilagodba genetskog programiranja

Kako bi se izgradnja rasporeda genetskim programiranjem bolje prilagodila ograničenju vremenom postavljanja uvest ćemo dodatne terminale **tablica 4.3**. Nije potrebno promijeniti shemu za izradu rasporeda te se koristi ona ranije navedena (algoritam 3).

OZNAKA	OPIS
$smin$	minimalna vrijednost vremena postavljanja
$sAvg$	srednja vrijednost vremena postavljanja
$setMac$	vrijeme postavljanja posla na stroju

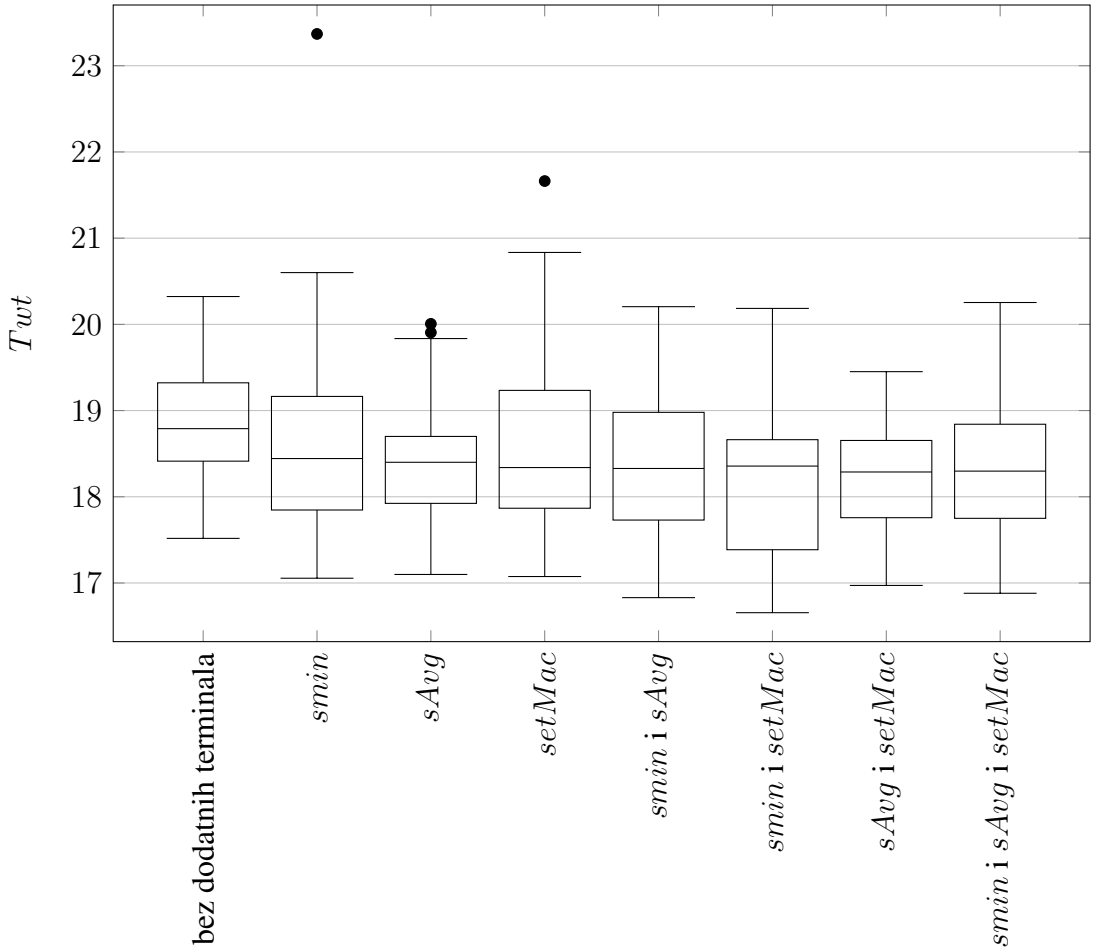
Tablica 4.3: Dodatni terminali za ograničenje vremenom postavljanja

4.3.3. Rezultati

Slika 4.1 prikazuje rješenja dobivena genetskim programiranjem s dodatnim terminalima, bez dodatnih terminala te s kombinacijama dodatnih terminala. Dodatni terminali modelirani su s ciljem da GP može izgraditi pravila raspoređivanja koja imaju različite informacije vezane uz vremena postavljanja. Pravila raspoređivanja na temelju vrijednosti danih informacija mogu donijeti odluku o tome da li je isplativo rasporediti posao na dani stroj ili ne. Da je to stvarno tako može se vidjeti na slici 4.1. Na grafu se može primijetiti kako su rezultati bez dodatnih terminala lošiji nego oni s dodatnim ter-

minalima, štoviše, bilo koja kombinacija dodatnih terminala dovela je do poboljšanja rezultata.

Rezultati s dodatnim terminalima imaju otprilike podjednak medijan. Od pojedinačno dodanih terminala rezultati koji se ističu su za terminal $sAvg$. Rezultati dobiveni tim terminalom su međusobno blizu te smo generalno pronašli bolja rješenja. Kombinacijom terminala $smin$ i $setMac$ uspjeli smo pronaći najbolje rješenje za dani problem.



Slika 4.1: Rezultati GP-a za ograničenje vremena postavljanja

Koristeći ATCS pravilo dobiveni rezultati znatno su lošiji nego oni dobiveni genetskim programiranjem. Najbolji rezultat postignut je s parametrima $k_1 = 0.95$ i $k_2 = 0.1$, a iznosi 21.8318. Genetsko programiranje se pokazalo boljim rješenjem kada u sustavu imamo prisutno ograničenje vremenom izvođenja. Najbolji rezultati ostvareni su kada su korištena sva tri terminalna čvora što pokazuje da pravila raspoređivanja rade bolje što imaju više dostupnih informaciju o ograničenju.

	min	$median$	max
GA [7]	10.72999	11.03636	11.36008
GP			
bez dodatnih terminala	17.5176	18.84	20.3229
$smin$	17.0553	18.47425	23.3688
$sAvg$	17.099	18.419	20.0061
$setMac$	17.075	18.39615	21.6623
$smin$ i $sAvg$	16.8301	18.3608	20.2052
$smin$ i $setMac$	16.6555	18.35715	20.1851
$sAvg$ i $setMac$	16.9717	18.39735	19.4514
$smin$ i $sAvg$ i $setMac$	16.8808	18.31885	20.2535

Tablica 4.4: Usporedba rezultata GP-a za ograničenje vremena postavljanja

4.4. Ograničenje prikladnih strojeva

Kod ograničenja prihladnih stojeva (OPS) svaki stroj definira poslove koji se na njemu mogu izvoditi. Na taj način smanjuje se prostor rješenja zato što postoje specifični poslovi koji se mogu izvršiti samo na podskupu danih resursa.

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] [7].

4.4.1. Prilagodba ATC heuristike

Kod ograničenja prikladnih strojeva neće se mijenjati ATC pravilo već će se napraviti modifikacija u samoj logici ATC heuristike. Svi poslovi j koji se ne mogu izvršiti na stroju i biti će preskočeni i neće se računati njihov prioritet. Pri dodjeli posla stroju preskočit će se svi strojevi koji ga ne mogu izvoditi (**algoritam 6**). Na taj način se osigurava ispravnost rješenja bez velikih modifikacija originalnog ATC pravila.

4.4.2. Prilagodba GP-a

Kako bi se genetskim programiranjem pronašao raspored sa što manjim kašnjenjem napravljene su dvije promjene, dodani su novi terminali (**tablica 4.5**) te je napravljena modifikacija u shemi za izradu rasporeda (**algoritam 5**). Pomoću modifikacija u shemi

Algoritam 4 ATC prilagođen za ograničenje prihladnih strojeva

1: $n \leftarrow$ broj poslova
2: $m \leftarrow$ broj strojeva
3: $raspored[m][n]$ = []
4: **while** broj neraspoređenih poslova $\neq 0$ **do**
5: $t \leftarrow$ trenutno vrijeme
6: $nerasporedeniPoslovi \leftarrow$ dostupni neraspoređeni poslovi u sustavu
7: **while** $nerasporedeniPoslovi$ nije prazan **do**
8: // pronađi posao s najboljim prioritetom
9: **for all** $j \in nerasporedeniPoslovi, i \in m$ **do**
10: **if** posao j se može izvršiti na stroju i **then**
11: $prioritet \leftarrow I_j(t)$ ▷ 2.1
12: **end if**
13: **end for**
14: $posao \leftarrow$ posao s najvećim prioritetom
15: $stroj \leftarrow$ stroj na kojem će se $posao$ najranije izvršiti
16: $raspored[stroj] \xleftarrow{\text{ako je stroj dostupan i može izvršiti } posao} posao$
17: **end while**
18: **end while**
19: $fitnes \leftarrow$ evaluiraj $raspored$

osigurano je da posao neće biti raspoređen na stroj na kojem se ne može izvoditi, a novim terminalima uvodimo dodatne informacije o ograničenju.

OZNAKA	OPIS
$emfj$	broj strojeva na kojima se posao može izvršiti
$rjfm$	broj dostupnih poslova koji se mogu izvršiti na stroju
$amfj$	broj dostupnih strojeva koji mogu izvršiti posao

Tablica 4.5: Dodatni terminali za ograničenje prikladnih strojeva

4.4.3. Rezultati

Slika 4.2 prikazuje dobivene rezultate problema s ograničenjem prikladnih strojeva korištenjem genetskog programiranja. Dodatni terminali su modelirani s ciljem kako bi prioritetna funkcija mogla kod izračuna raspolagati sa informacijama vezanim uz

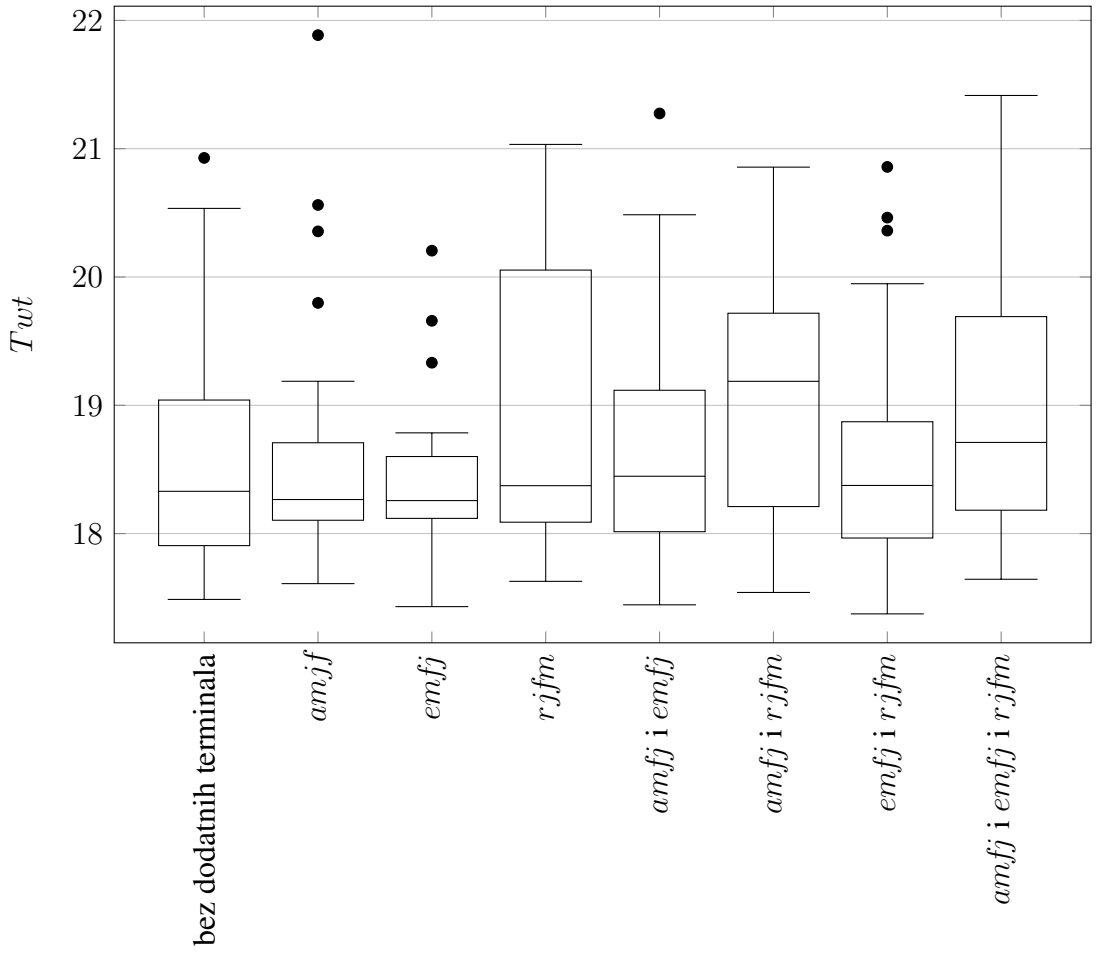
Algoritam 5 Prilagođena shema za izradu rasporeda ograničenju prikladnih strojeva

```
1: while postoje neraspoređeni poslovi do
2:     čekaj dok posao ne postane dostupan ili se ne završi
3:     for all dostupne poslove i strojeve do
4:         if posao j se može izvršiti na stroju i then
5:             izračunaj prioritet π_ij posla j na stroju i
6:         end if
7:     end for
8:     for all dostupne poslove do
9:         odredi najbolji stroj (onaj za koji se postiže najbolji iznos prioriteta π_ij i na
       kojem se može izvesti posao)
10:    end for
11:    while postoje poslovi čiji je najbolji stroj dostupan do
12:        odredi najbolji prioritet takvih poslova
13:        rasporedi posao s najboljim prioritetom
14:    end while
15: end while
```

ograničenje prikladnih strojeva te tako izgraditi raspored s manjim kašnjenjem. Eksperiment je pokazao kako je informacija koji nosi terminal $emfj$ (broj dostupnih poslova koji se mogu izvršiti na stroju) najkorisnija. Koristeći samo taj terminal generalno su dobivena bolja rješenja koja su međusobno relativno blizu, dok su kod npr. terminala $rjfm$ dobivena lošija rješenja s velikom razlikom između količine kašnjenja. Zanimljivo je napomenuti kako su kombinacijom najboljeg i najgoreg pojedinačnog terminala pronađena najbolja rješenja za dani problem. To zapravo pokazuje kako nije prikladno terminalne čvorove ocijeniti individualno, jer u sinergiji s drugim čvorovima mogu dovesti do puno boljih rezultata.

Kod ovog ograničenja vidljivo je da dodatni terminalni čvorovi nisu doveli do velikog poboljšanja rezultata, što može biti iz razloga što ovo ograničenje neće imati prevelikog utjecaja na redoslijed izvođenja poslova i stoga sama informacija o tom ograničenju nije značajna pri odabiru posla koji će se rasporediti

Prilagođenim ATC algoritmom dobivena su znatno lošija rješenja. Najbolje rješenje ATC algoritma je za vrijednost parametra $k = 0.45$, a iznosi 20.3027. Genetsko programiranje se pokazalo boljim izborom kada je u sustavu prisutno ograničenje prikladnih strojeva.



Slika 4.2: Rezultati GP-a s ograničenjem prikladnih strojeva

	min	$median$	max
GA [7]	14.57869383	15.14603253	15.93751493
GP			
bez dodatnih terminala	17.487	18.33235	20.9281
$amfj$	17.6101	18.267	21.8853
$emfj$	17.4308	18.2598	20.2053
$rjfm$	17.6279	18.46785	21.0335
$amfj$ i $emfj$	17.4449	18.53885	21.2748
$amfj$ i $rjfm$	17.5413	19.21165	20.8564
$emfj$ i $rjfm$	17.3743	18.3826	20.8582
$amfj$ i $emfj$ i $rjfm$	17.6441	18.77075	21.4149

Tablica 4.6: Usporedba rezultata GP-a s ograničenjem prikladnih strojeva

4.5. Privremeni prekid u radu strojeva

Privremeni prekid u radu (OPP) definira vremenska razdoblja u kojima je stroj u nemogućnosti izvoditi poslove. Mogući uzroci su kvarovi strojeva, održavanje strojeva i slično. Prekidi koji se događaju poznati su unaprijed prije izrade rasporeda (npr. periodi za održavanje strojeva).

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

$$SP_i = \frac{\sum_i^m \sum_j^n p_{ij}}{m^2} \tag{4.3}$$

gdje p_{ij} predstavlja vrijeme izvođenja posla j na stroju i, n broj poslova, a m broj strojeva. Trajanja kvarova za svaki od problema generiraju se iz intervala $[0, SP_i]$, a njihov broj i duljina 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 duljina trajanja raznolikija [7].

4.5.1. Prilagodba ATC heuristike

Kako bi se ATC prilagodio ograničenju neće se mijenjati prioritetna funcija već način na koji ATC raspoređuje poslove. Ideja je preskočiti one strojeve na kojim trenutno traje prekid ili će se dogoditi prekid za vrijeme izvođenja posla. Na taj način će se osigurati da se na stroj neće rasporediti posao tijekom prekida rada stroja (**algoritam 6**).

4.5.2. Prilagodba GP-a

Strategija prilagodbe GP-a biti će slična kao i kod prethodnog ograničenja. Definirani su novi terminali (**tablica 4.8**) te je prilagođena shemu za izradu rasporeda (**algoritam 7**) kako bi se osiguralo da posao neće biti raspoređen na stroj tijekom perioda prekida rada stroja.

Algoritam 6 ATC prilagođen za ograničenje prekidom rada strojeva

1: $n \leftarrow$ broj poslova
2: $m \leftarrow$ broj strojeva
3: $raspored[m][n]$ = []
4: **while** broj neraspoređenih poslova $\neq 0$ **do**
5: $t \leftarrow$ trenutno vrijeme
6: $nerasporedeniPoslovi \leftarrow$ dostupni neraspoređeni poslovi u sustavu
7: **while** $nerasporedeniPoslovi$ nije prazan **do**
8: // pronađi posao s najboljim prioritetom
9: **for all** $j \in nerasporedeniPoslovi, i \in m$ **do**
10: **if** na stroju i se neće dogoditi prekid za vrijeme izvođenja posla j **then**
11: $prioritet \leftarrow I_j(t)$ ▷ 2.1
12: **end if**
13: **end for**
14: $posao \leftarrow$ posao s najvećim prioritetom
15: $stroj \leftarrow$ stroj na kojem će se $posao$ najranije izvršiti bez prekida
16: $raspored[stroj] \xleftarrow{\text{ako je stroj dostupan i neće se dogoditi prekid}} posao$
17: **end while**
18: **end while**
19: $fitnes \leftarrow$ evaluiraj $raspored$

OZNAKA	OPIS
$bwhde$	hoće li se dogoditi prekid tijekom izvođenja posla, $bwhde = \begin{cases} 1, & \text{prekid će se dogoditi} \\ 0, & \text{prekid se neće dogoditi} \end{cases}$
$norb$	broj preostalih prekida
$tube$	vrijeme do završetka prekida
$tunb$	vrijeme do sljedećeg prekida

Tablica 4.7: Dodani terminali za ograničenje prekidom rada strojeva

Algoritam 7 Prilagođena shema za izradu rasporeda ograničenju prekidom rada strojeva

1: **while** postoje neraspoređeni poslovi **do**
2: čekaj dok posao ne postane dostupan ili se ne završi
3: **for all** dostupne poslove i strojeve **do**
4: **if** neće se dogoditi prekid na stroju i za vrijeme izvršavanja posla j **then**
5: izračunaj prioritet π_{ij} posla j na stroju i
6: **end if**
7: **end for**
8: **for all** dostupne poslove **do**
9: odredi najbolji stroj (onaj za koji se postiže najbolji iznos prioriteta π_{ij} i na kojem se neće dogoditi prekid za vrijeme izvršavanja)
10: **end for**
11: **while** postoje poslovi čiji je najbolji stroj dostupan **do**
12: odredi najbolji prioritet takvih poslova
13: rasporedi posao s najboljim prioritetom
14: **end while**
15: **end while**

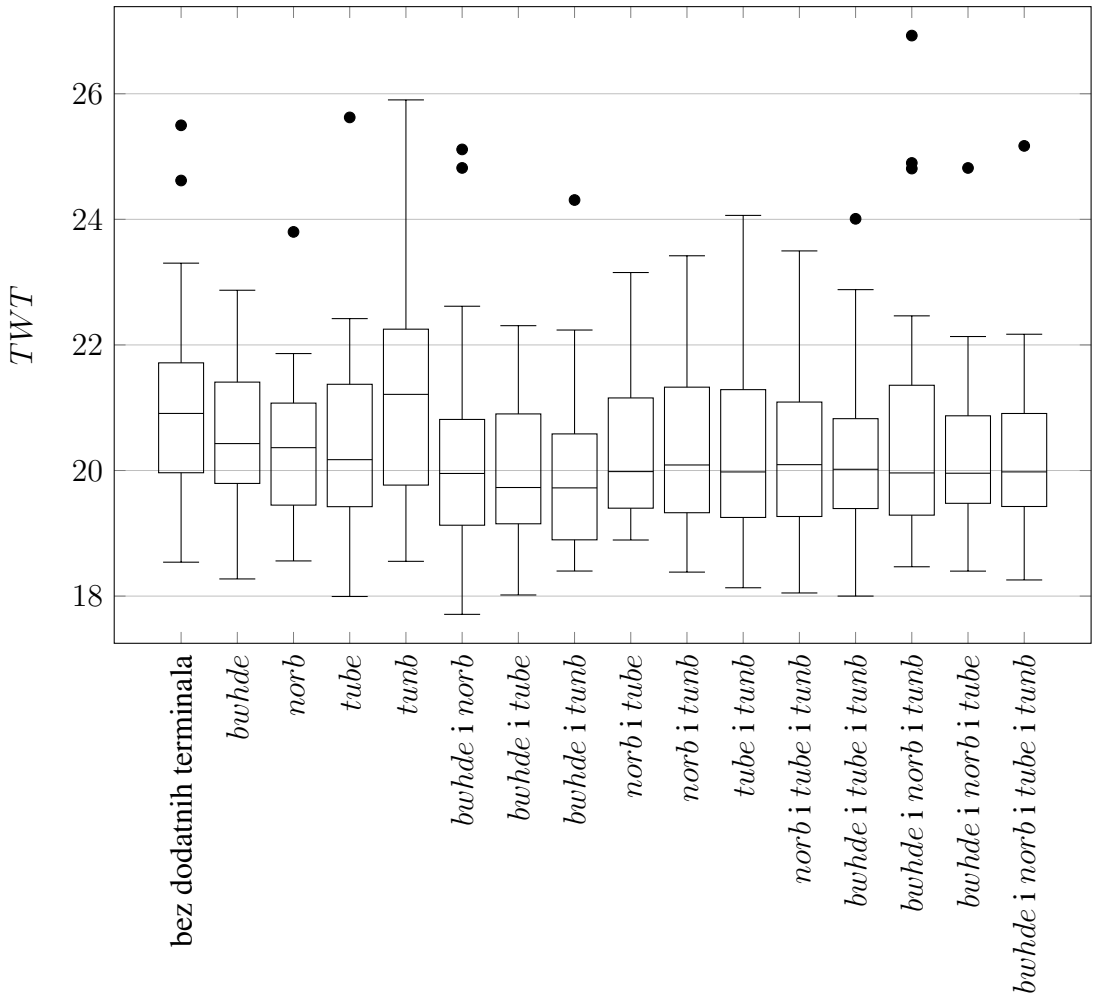
4.5.3. Rezultati

Slika 4.3 i **tablica 4.8** prikazuje rezultate dobivene genetskim programiranjem kada je u sustavu prisutno ograničenje privremenim prekidom rada stroja. Iz slike se odmah primjećuje smanjenje kašnjenja kada prioritetna funkcija raspolaže dodatnim informacijama o ograničenju. Od pojedinačnih terminala najviše se istaknuo $tube$ s najmanjim iznosom medijana te najbolje pronađenim rješenjem naspram drugih pojedinačnih terminala. Znatno lošijim od svih ostalih terminala pokazao se $tunb$ što upućuje na to da informacija o preostalom vremenu do sljedećeg prekida sama za sebe ne doprinosi ikakvom poboljšanju. Zanimljivo je primijetiti kako su kombinacijom najgoreg pojedinačnog terminala $tunb$ i terminala $bwhde$ dobivena najbolja rješenja. Informacije koje pojedinačno ne doprinose značajno poboljšanju, u kombinaciji tvore smisleniju prioritetnu funkciju.

Genetsko programiranje se pokazalo boljom opcijom kada u sustavu imamo ograničenje prekidom rada strojeva naspram prilagođene ATC heuristike. Najbolje rješenje pronađeno ATC heuristikom je uz vrijednost parametra $k_1 = 1.5$, a iznosi 22.5257.

	min	$median$	max
GA [7]	11.35476603	11.68943258	11.88922483
GP			
bez dodatnih terminala	18.5401	20.95185	25.4976
$bwhde$	18.2733	20.5281	22.8709
$norb$	18.5601	20.37445	23.7999
$tube$	17.9937	20.1921	25.6233
$tunb$	18.5537	21.3422	25.9022
$bwhde$ i $norb$	17.7093	19.97095	25.1135
$bwhde$ i $tube$	18.0179	19.7833	22.3066
$bwhde$ i $tunb$	18.3991	19.7803	24.3077
$norb$ i $tube$	18.8933	20.0648	23.153
$norb$ i $tunb$	18.382	20.09715	23.4198
$tube$ i $tunb$	18.1327	20.28005	24.0628
$norb$ i $tube$ i $tunb$	18.0499	20.0949	23.4972
$bwhde$ i $tube$ i $tunb$	18.0001	20.0364	24.0084
$bwhde$ i $norb$ i $tunb$	18.4671	20.2396	26.9264
$bwhde$ i $norb$ i $tube$	18.3977	19.9608	24.8181
$bwhde$ i $norb$ i $tube$ i $tunb$	18.2572	20.05885	25.1696

Tablica 4.8: Usporedba rezultata GP-a s ograničenjem privremenim prekidom u radu stroja



Slika 4.3: Rezultati GP-a s ograničenjem prekidom rada strojeva

4.6. Ograničenje redoslijedom izvođenja

Ograničenje redoslijedom izvođenja (ORI) za svaki posao definira poslove koji se moraju izvršiti prije njega, odnosno definira njegove prethodnike. Ono onemogućava da se neki od poslova rasporedi na bilo koji stroj dok god se nisu izvršili svi njegovi prethodnici.

U svrhu korištenja ovog ograničenja, potrebno je pobrinuti se da poslovi sljedbenici nisu raspoređeni prije poslova prethodnika. Osim toga, potrebno je prilagoditi način evaluacije jedinke da u ukupno vrijeme izvršavanja pojedinog posla uračuna i vrijeme koje je potrebno za čekanje izvršavanja poslova prethodnika na ostalim strojevima.

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 [7].

4.6.1. Prilagodba ATC heuristike

Kako bi ATC heuristika izgradila ispravan raspored potrebno je napraviti modifikacije algoritma prikazanog **algoritmom 8**. Stroju se dodjeljuje onaj posao s najvećim prioritetom koji nema neraspoređenih prethodnika. Na taj način se osiguralo da prethodnici posla neće biti raspoređeni nakon posla te je smanjeno čekanje posla na izvršavanje prethodnika.

4.6.2. Prilagodba GP-a

Kako bi se GP prilagodio ograničenju napravljene su modifikacije sheme izrade rasporeda kako se posao ne bi rasporedio prije svojih prethodnika (**algoritam 9**) te su dodani novi terminali (**tablica 4.9**) kako bi uveli dodatne informacije o ograničenju.

Algoritam 8 ATC prilagođen za ograničenje redoslijedom izvođenja

1: $n \leftarrow$ broj poslova
2: $m \leftarrow$ broj strojeva
3: $raspored[m][n]$ = []
4: **while** broj neraspoređenih poslova $\neq 0$ **do**
5: $t \leftarrow$ trenutno vrijeme
6: $nerasporedeniPoslovi \leftarrow$ dostupni neraspoređeni poslovi u sustavu
7: **while** $nerasporedeniPoslovi$ nije prazan **do**
8: // pronađi posao s najboljim prioritetom
9: **for all** $j \in nerasporedeniPoslovi, i \in m$ **do**
10: **if** posao j nema neraspoređenih prethodnika **then**
11: $prioritet \leftarrow I_j(t)$ ▷ 2.1
12: **end if**
13: **end for**
14: $posao \leftarrow$ posao s najvećim prioritetom
15: $stroj \leftarrow$ stroj na kojem će se $posao$ najranije izvršiti
16: $raspored[stroj] \xleftarrow{\text{ako je stroj dostupan i } posao \text{ nema neraspoređenih prethodnika}} posao$
17: **end while**
18: **end while**
19: $fitnes \leftarrow$ evaluiraj $raspored$

OZNAKA	OPIS
$nous$	broj neraspoređenih sljedbenika trenutno razmatranog posla
$avss$	prosječni $slack$ svih sljedbenika trenutno razmatranog posla, $slack = d_j - (t + p_{ij})$
mss	minimalan $slack$ svih sljedbenika trenutno razmatranog posla
$mxss$	maximalan $slack$ svih sljedbenika trenutno razmatranog posla
wsl	težinsko kašnjenje svih sljedbenika trenutno razmatranog posla
scl	broj svih čvorova stabla sljedbenika za dani posao
$mscl$	najveća dubina stabla sljedbenika za dani posao

Tablica 4.9: Dodani terminali za ograničenje redoslijedom izvođenja

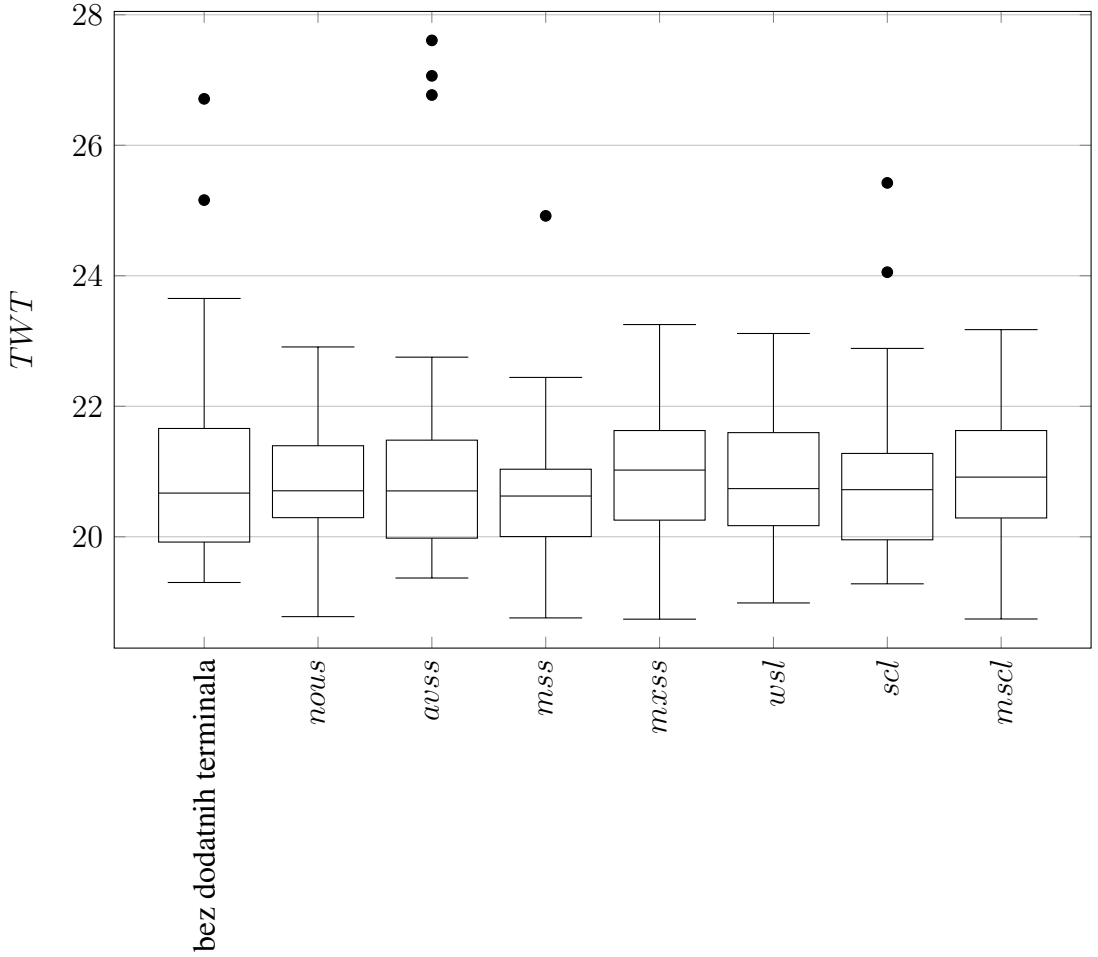
Algoritam 9 Prilagođena shema za izradu rasporeda ograničenju redoslijedom izvođenja

```
1: while postoje neraspoređeni poslovi do
2:     čekaj dok posao ne postane dostupan ili se ne završi
3:     for all dostupne poslove i strojeve do
4:         if raspoređeni su svi prethodnici posla then
5:             izračunaj prioritet π_ij posla j na stroju i
6:         end if
7:     end for
8:     for all dostupne poslove do
9:         odredi najbolji stroj (onaj za koji se postiže najbolji iznos prioriteta π_ij)
10:    end for
11:    while postoje poslovi čiji je najbolji stroj dostupan i čiji prethodnici su već raspoređeni do
12:        odredi najbolji prioritet takvih poslova
13:        rasporedi posao s najboljim prioritetom
14:    end while
15: end while
```

4.6.3. Rezultati

Slika 4.4 i **tablica 4.10** prikazuju rezultate dobivene genetskim programiranjem kada je u sustavu prisutno ograničenje redoslijedom izvođenja poslova. Ako usporedimo vrijednosti medijana svih rezultata primjećuje se da dodatni terminali nisu odviše pridonjeli poboljšanju. Razlog tome može biti činjenica da što je manji dio poslova ovisan, broj sljedbenika koje posao ima nema nekog prevelikog utjecaja na odluku koji će posao biti raspoređen. Osim toga, zbog međuovisnosti poslova neće svi poslovi ni biti uzeti u obzir što umanjuje broj poslova od kojih moramo odabrati onog koji će se dodijeliti stroju, tj. rasporediti.

Prilagođenom ATC heuristikom najbolje rješenje pronađeno je za vrijednosti parametra $k1 = 1.3$, a iznosi 20.1955. Za razliku od prethodnih ograničenja, ATC se kod ograničenja redoslijedom izvođenja poslova pokazala dobrim rješenjem. Uspoređujući dobiveno rješenje s medijanima dobivenih rezultata s GP-om može se reći da se ATC heuristika pokazala boljom.



Slika 4.4: Rezultati GP-a za ograničenje redoslijedom izvođenja

	min	*median*	*max*
GA [7]	14.28788	15.7357	16.8074
GP			
bez dodatnih terminala	19.2993	20.6941	26.7114
nous	18.776	20.7365	22.9092
ass	19.3679	20.70685	27.6074
mss	18.7564	20.6362	24.9182
mxss	18.7373	21.02555	23.2523
wsl	18.9858	20.7829	23.1154
scl	19.2789	20.73535	25.4232
mscl	18.7399	20.95625	23.1746

Tablica 4.10: Usporedba rezultat GP-a s ograničenjem redoslijedom izvođenja poslova

4.7. Kombinacije ograničenja

Slika 4.11 prikazuje rezultate dobivene kada je u sustavu prisutno više različitih ograničenja. Korišteni terminali kod genetskog programiranja odabrani su na osnovu najboljeg medijana kod pojedinačnih ograničenja, a parametri ATC heuristike odabrani su na temelju najboljeg rezultata na skupu za učenje. Dodatno, izvršeni su eksperimenti za sve kombinacije ograničenja koristeći genetski algoritam (GA). Na temelju rezultata možemo zaključiti kako se genetsko programiranje pokazalo boljim rješenjem od ATC-a u većini slučajeva. ATC se pokazao boljim samo kod ograničenja redoslijedom izvođenja poslova te kombinacije ograničenja {OVP, OPS, OPP}, premda je upitno bi li GP bio lošiji kod te kombinacije ograničenja ako bi se optimirali odabrani terminali. Ako pogledamo najbolja rješenja dobivena genetskim programiranjem, primjećujemo da smo kod svih kombinacija ograničenja pronašli bolje rješenje od ATC-a.

GP s dodatnim terminalima pokazao se boljim nego bez dodatnih terminala. To je jasan indikator da s terminalima imamo mogućnost izgraditi bolja pravila raspoređivanja jer koriste dodatne informacije o sustavu te se mogu bolje prilagoditi.

	OVP OPS	OVP OPP	OPS OPP	OVP OPS OPP
GA				
min	16.76788	14.28788	19.70514	16.76788
$median$	17.45	15.73571	20.02518	17.45
max	18.3353	16.80741	20.54620	18.33529
ATC				
$k1$	0.6	1.9	0.2	0.6
$k2$	1.9	1.9	-	1.9
Twt	27.6076	30.0478	40.9037	37.8632
GP				
min	24.2597	23.2614	28.8009	35.9829
$median$	27.58485	25.26985	33.046	41.96735
max	46.5309	29.5905	39.0432	52.4947
GP - bez dodatnih terminala				
min	23.8205	23.761	30.5045	37.9792
$median$	26.7524	26.06835	34.09265	42.3346
max	30.9706	30.6188	85.9731	47.8872

Tablica 4.11: Usporedba rezultata za kombinacije različitih ograničenja

5. Zaključak

Cilj ovog rada bio je istražiti pristup rješavanju problema raspoređivanja u okruženju nesrodnih strojeva koristeći genetsko programiranje. Nakon što je definiran problem koji rješavamo opisana je ATC heuristika kao jedan od pristupa rješavanju problema raspoređivanja. U drugom dijelu rada opisana je ideja na kojoj se zasniva genetsko programiranje. Zatim je predložen skup operatora i terminala potrebnih da se genetsko programiranje prilagodi problemu raspoređivanja. Navedeni su osnovni dijelovi pravila raspoređivanja, prioritetna funkcija koju gradimo genetskim programiranjem te shema za izradu rasporeda koja donosi konačne odluke o rasporedu na osnovu vrijednosti prioriteta određenog prioritetnom funkcijom. U trećem dijelu definirana su ograničenja koja mogu postojati u sustavu te je za svaki tip ograničenja opisana prilagodba ATC heuristike i genetskog programiranja. Analizirano je kako se spomenuti algoritmi snalaze u takvom okruženju eksperimentima za svako pojedino ograničenje kao i za kombinaciju različitih ograničenja u sustavu.

U radu je dočarana kompleksnost rješavanja NP-teškog problema kao što je problem raspoređivanja. Premda nije moguće doći do optimalnog rješenja tog problema u razumnom vremenu, genetsko programiranje se pokazalo jako dobrom aproksimativnom metodom. Iz dobivenih rezultata eksperimenata možemo zaključiti kako je ono načelno bolje nego ATC heuristika. Pomoću GP-a dobili smo bolje rezultate za većinu pojedinačnih ograničenja kao i za kombinaciju različitih ograničenja. No kako bi GP potvrdio svoju nadmoć nad ATC heuristikom, potrebno je iskoristiti dani prostor za poboljšanje kao što je npr. optimiranje skupa korištenih terminala.

Kako bi dublje istražili potencijal koji se krije u genetskom programiranju, potrebno je osmisliti i isprobati nove terminalne čvorove kao i optimizirati njihovo korištenje za svako pojedinačno ograničenje te za kombinaciju različitih ograničenja.

LITERATURA

[1] Crawford L. Menon P. Cheng, V. *Air traffic control using genetic search techniques*, svezak 1. 1999. URL http://ieeexplore.ieee.org/document/806209/.

[2] J. V. Hansen. *Genetic search methods in air traffic control*, svezak 31. 2004. URL http://linkinghub.elsevier.com/retrieve/pii/S030505480200228.

[3] Karlo Knežević Marko Đurasević, Domagoj Jakobović. Adaptive scheduling on unrelated machines with genetic programming. 2015.

[4] Petrovic S. Ouelhadj, D. *A survey of dynamic scheduling in manufacturing systems*, svezak Journal of Scheduling, Vol. 12, No. 4. 2009. URL http://link.springer.com/10.1007/s10951-008-0090-8.

[5] Michael L. Pinedo. *Scheduling - Theory, Algorithms, and Systems*. 2012.

[6] Riccardo Poli, William B. Langdon, i Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. URL http://www.gp-field-guide.org.uk. (With contributions by J. R. Koza).

[7] Ivan Vlašić. Rješavanje problema raspoređivanja u okruženju nesrodnih strojeva korištenjem evolucijskih algoritama. 2018.

[8] Kumar Bhaskaran Yound Hoon Lee i Michael Pinedo. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. 1995.

[9] Marko Čupić. Scheduling school activities using evolutionary computation. 2011.

Automatski razvoj pravila raspoređivanja za probleme raspoređivanja s ograničenjima

Sažetak

Problem raspoređivanja je NP-težak problem, što znači da ga nije moguće optimalno riješiti u polinomijalnom vremenu te se za rješavanje takvih problema najčešće koriste aproksimativne metode. Jedna od tih metoda je i genetsko programiranje. U radu je opisana primjena genetskog programiranja na problem raspoređivanja u okruženju nesrodnih strojeva. Definirani su glavni dijelovi pravila raspoređivanja, prioritetna funkcija koju razvijamo pomoću genetskog programiranja te shema za izradu rasporeda koja donosi konačne odluke prilikom dodjeljivanja posla strojevima. Definirana su i neka ograničenja koja se mogu pojaviti u sustavu kao i prilagodba GP-a za svako navedeno ograničenje. Dodatno, napravljena je analiza kako se GP ponaša za svako pojedinačno ograničenje kao i za kombinaciju različitih ograničenja u sustavu.

Ključne riječi: raspoređivanje, okruženje nesrodnih strojeva, genetsko programiranje, pravila raspoređivanja, ograničenja u raspoređivanju

Automated design of dispatching rules for scheduling problems s with constraints

Abstract

Scheduling problems are NP-hard, which means that it is not possible to solve them optimally in polynomial time. The most commonly used methods for solving these types of problems are approximate methods. One of those methods is genetic programming. This thesis describes the application of genetic programming on scheduling problems in the unrelated machines environment. It defines main components of dispatching rules, a priority function which is developed by using genetic programming and schedule generation scheme which brings final decisions about job assignment to machine. It describes scheduling constraints which may appear in the system as well as the adjustment of genetic programming for those constraints. Finally, behaviour of genetic programming is analysed in an environment with one or multiple scheduling constraints.

Keywords: scheduling, unrelated machines environment, genetic programming, dispatching rules, scheduling constraints