Pregled rezultata za ostvarena okruženja raspoređivanja

***Napomena**: najnoviji rezultati u okviru projekta se mogu vidjeti u radu koji je u postupku objavljivanja, "**Evolving Priority Scheduling Heuristics with Genetic Programming**" (dostupno na web stranici projekta)*

1 Vrednovanje rasporeda

Vrednovanje rasporeda moguće je obaviti po više kriterija, koji su često vrlo složeni i međusobno proturječni [Jon 98]. Kriteriji vrednovanja rasporeda u velikoj mjeri utječu na izbor odgovarajućeg algoritma raspoređivanja. Ocjena rasporeda moguća je nakon izvođenja rasporeda i nakon prikupljanja izlaznih veličina u sustavu. Slijedeći konvenciju iz [Leu 04], izlazne veličine sustava opisane su velikim slovima.

- **Vrijeme završetka** C_j (engl. *completion time*) - trenutak u kojemu aktivnost *j* završava izvođenje
- **Protjecanje** F_j (engl. *flowtime*) – količina vremena koju je neka aktivnost provela u sustavu:

$$F_j = C_j - r_j \tag{1.1}$$

- **Kašnjenje** L_j (engl. *lateness*) – razlika (pozitivna ili negativna) između vremena završetka i vremena željenog završetka:

$$L_j = C_j - d_j \tag{1.2}$$

- **Zaostajanje** T_j (engl. *tardiness*) – pozitivni iznos kašnjenja neke aktivnosti; ako je kašnjenje negativno, zaostajanje je jednako nuli:

$$T_j = \max\{0, L_j\} \tag{1.3}$$

- **Preuranjenost** E_j (engl. *earliness*) – negativni iznos kašnjenja neke aktivnosti; ako je kašnjenje pozitivno, preuranjenost je jednaka nuli:

$$E_j = \max\{0, -L_j\} \tag{1.4}$$

- **Zakašnjelost** U_j – označava je li neka aktivnost prekoračila željeno vrijeme završetka:

$$U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases} \tag{1.5}$$

Mjerila vrednovanja rasporeda

Na osnovu navedenih veličina definira se većina mjerila vrednovanja. Neki od najčešćih kriterija opisani su u nastavku.

- Ukupna duljina rasporeda $C_{\max}$ (engl. *makespan*) – ukupna duljina rasporeda je posljednje vrijeme završetka svih poslova u sustavu:

$$C_{\max} = \max\{C_j\} \tag{1.6}$$

- Najveće kašnjenje $L_{\max}$ (engl. *maximum lateness*) – najveće kašnjenje definirano je kao

$$L_{\max} = \max\{L_j\} \tag{1.7}$$

- Težinsko protjecanje F_w (engl. *weighted flowtime*) – definira se kao suma težinskog protjecanja svih poslova:

$$F_w = \sum_j w_j F_j \tag{1.8}$$

- Težinsko zaostajanje T_w (engl. *weighted tardiness*) – jednako je težinskoj sumi zaostajanja svih poslova:

$$T_w = \sum_j w_j T_j \tag{1.9}$$

- Težinski zbroj zaostalih poslova ili težinska zakašnjelost U_w (engl. *weighted number of tardy jobs*) – definira se kao težinska suma svih zaostalih poslova:

$$U_w = \sum_j w_j U_j \tag{1.10}$$

- Težinska preuranjenost i težinsko zaostajanje ET_w (engl. *weighted earliness and weighted tardiness*) – definira se kao zbroj težinske preuranjenosti i težinskog zaostajanja, uz posebne težinske faktore za obje vrijednosti:

$$ET_w = \sum_j \left(w_{Ej} E_j + w_{Tj} T_j \right) \tag{1.11}$$

Navedena su mjerila vrednovanja najviše upotrebljavana u literaturi i primjeni, iako ni približno ne pokrivaju sve korištene kriterije. Osim opisanih izraza moguće je koristiti i netežinske inačice, gdje se pretpostavlja da je težina svakog zadatka jednaka [Rus 97]. Ovaj uvjet u svim slučajevima predstavlja olakšanje postupka raspoređivanja no najčešće nije prikladan za opis stvarnih uvjeta raspoređivanja gdje postoji različit jedinični trošak za različite poslove.

Posljednji od navedenih kriterija primjer je *nepravilnog* (engl. *nonregular*) mjerila jer uključuje trošak u slučaju ranijeg završetka posla. *Pravilni* kriteriji su oni kod kojih raniji završetak aktivnosti nikada ne povećava funkciju troška. Opravdanost opisanog nepravilnog kriterija valjana je u situaciji kada naručitelj ne želi primiti dovršeni proizvod prije dogovorenog roka (željenog vremena završetka), a cijena skladištenja može biti veća od cijene održavanja posla u sustavu (engl. *just-in-time environment*). U slučajevima kriterija koji uključuju više vrijednosti, kao u potonjem, opravdano je pretpostaviti da je jedinična cijena jedne vrijednosti različita od druge, kao što je u ovom primjeru jedinični trošak zaostajanja (w_{Tj}) u općenitom slučaju različit od jediničnog troška preuranjenosti (w_{Ej}). Isto tako, jedinični trošak protjecanja može biti različit od navedenih, u kojem slučaju se dodatno označava sa w_{Fj} .

2 Definiranje funkcije cilja

Ocjena dobrote jedinki, a ujedno i učinkovitosti promatranih pravila raspoređivanja, obavlja se ovisno o okruženju i definiranom mjerilu vrednovanja rasporeda na većem broju ispitnih primjera. Prilikom odabira kriterija, za pojedina okruženja birani su samo netrivijalni kriteriji, odnosno oni za koje ne postoji optimalno rješenje. U situaciji kada je neko rješenje potrebno ocijeniti s obzirom na cijeli skup ispitnih primjera sa različitim svojstvima, funkciju cilja treba definirati tako da ocjena za sve primjere utječe podjednakom težinom na konačni rezultat. Stoga se po uzoru na [Moh 83, Dha 78] funkcija cilja za različite kriterije za pojedini primjer sa indeksom *i* iz ispitnog skupa definira na sljedeći način:

- za težinsko zaostajanje:

$$f_i = \frac{\sum_{j=1}^{n} w_j T_j}{n \cdot \bar{w} \cdot \bar{p}}, \tag{1.12}$$

- za težinski zbroj zakašnjelih poslova (težinska zakašnjelost):

$$f_i = \frac{\sum_{j=1}^{n} w_j U_j}{n \cdot \bar{w}}, \tag{1.13}$$

- za težinsko protjecanje:

$$f_i = \frac{\sum_{j=1}^{n} w_j F_j}{n \cdot \bar{w} \cdot \bar{p}}, \tag{1.14}$$

- za ukupnu duljinu rasporeda:

$$f_i = \frac{\max\{C_j\}}{n \cdot \bar{p}}, \tag{1.15}$$

gdje je n broj poslova u ispitnom primjeru, $\bar{w}$ je srednja vrijednost težina poslova, a $\bar{p}$ je srednje trajanje poslova. Vrijednost se za pojedini ispitni primjer dijeli sa brojem poslova u dotičnom primjeru kako bi se postigla podjednaka težina za primjere sa različitim brojem poslova. U slučajevima gdje pojedini kriterij uključuje težine, dodatno se obavlja dijeljenje sa srednjom težinskom vrijednošću, a ako kriterij uključuje i neku količinu vremena ovisnu o trajanju poslova, u nazivnik se stavlja i srednje trajanje obrade. Ukupna dobrota neke jedinke dobiva se zbrajanjem svih funkcija cilja za pojedine ispitne primjere:

$$F = \sum_i f_i \,. \tag{1.16}$$

3 Raspoređivanje na jednom stroju

3.1 Statička okolina raspoređivanja

Zadatak je genetskog programiranja definirati takvu funkciju prioriteta koja će postići što bolje rezultate (ovisno o zadanom kriteriju). U pozadini ovog nastojanja je ideja da je, osim prikazanih pravila raspoređivanja, moguće pronaći još neke izraze koji bi na bolji način od postojećih pravila mogli iskoristiti dostupne podatke o problemu.

Prikaz rješenja u ovom okruženju je stablo koje predstavlja funkciju prioriteta (jedinka je prikazana stablom). Skup mogućih čvorova stabla mora omogućiti sustavu da na učinkovit način predstavi rješenje problema. Odabir elemenata u skupu čvorova načinjen je ručno za svako pojedino okruženje. U odabiru čvorova nastoji se koristiti one elemente koji su relevantni za specifično okruženje i uvjete raspoređivanja. Iako je u nekim slučajevima odluka o uključivanju ili izostavljanju nekog elementa trivijalna (npr. u statičkom problemu nema smisla uvoditi vrijeme pripravnosti poslova), općenito je to zadatak koji zahtijeva znanje o problemu koji se rješava. Budući da kvaliteta mogućeg rješenja najviše ovisi o svojstvima skupa čvorova, odabir elemenata nije trivijalan i u biti predstavlja najtežu zadaću u rješavanju postavljenog problema.

Elemente skupa čvorova dijelimo u dvije skupine: skup funkcijskih čvorova i podatkovnih čvorova (listova stabla). U slučaju statičkih uvjeta raspoređivanja na jednom stroju, relevantni podaci koji bi u svakom slučaju trebali biti uključeni su trajanje poslova, njihove težine i željena vremena završetka. Korisnu informaciju sustavu mogu predstavljati i zbroj trajanja svih poslova te zbroj željenih vremena završetka. Budući je primjena pravila raspoređivanja takva da se u svakom koraku ponovno računaju prioriteti svih neraspoređenih poslova, povijest rada sustava ne utječe na odluku o sljedećem poslu. Iz toga su razloga u skup listova dodani broj preostalih poslova te zbroj njihovih trajanja. Kako bi postupak imao dobru informaciju o žurnosti raspoređivanja nekog posla, dodana je i informacija o vremenu za koje posao još može pričekati prije početka rada, a da ne prekorači željeno vrijeme završetka (dopuštena odgoda).

Tablica 1. Popis čvorova za statički problem na jednom stroju

Oznaka funkcijskog čvora	Definicija
ADD	binarni operator zbrajanja
SUB	binarni operator oduzimanja
MUL	binarni operator množenja
DIV	zaštićeno dijeljenje: $DIV(a,b) = \begin{cases} 1, \text{ ako } \lvert b \rvert < 0.000001 \\ a/b, \text{ inače} \end{cases}$
POS	unarni operator '+': $POS(a) = \max\{a, 0\}$
Oznaka podatkovnog čvora	**Definicija vrijednosti podatkovnog čvora**
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
N	ukupni broj poslova
Nr	preostali broj poslova (koji još nisu raspoređeni)

SP	zbroj trajanja svih poslova
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$

U skup funkcijskih čvorova uvršteni su čvorovi koji podržavaju četiri osnovne računske operacije, s tom razlikom da je operatoru dijeljenja dodana 'zaštita' od dijeljenja sa premalom vrijednošću. Također je definiran unarni operator koji vraća pozitivnu vrijednost argumenta odnosno nulu ako je argument negativan. Pregled konačnog skupa čvorova prikazan je u tablici 1.

Problem težinskog zaostajanja

Prva skupina pokusa u statičkom okruženju jednoga stroja provedena je za optimiranje težinskog zaostajanja poslova. Težinsko zaostajanje je najčešće rabljeni kriterij u ovom okruženju, a oznaka ovoga problema u $\alpha|\beta|\gamma$ zapisu je $1|\ |\sum w_j T_j$. Ocjena svakog od pravila dobivena je dijeljenjem sa prosječnom težinom, trajanjem ili brojem poslova u dotičnom primjeru.

Najbolje dobiveno rješenje odabrano je iz 30 provedenih pokusa po uspješnosti rješenja na skupu ispitnih primjera za ocjenu, koji se ne koriste u učenju. Rješenje se može prikazati u obliku stabla u kojemu je i predstavljeno unutar genetskog programa. Zbog svoje veličine, rješenja se obično prikazuju u nekom drugom zapisu, od kojih je najuobičajeniji infiksni zapis, koji se koristi i u ovom radu. Jedno dobiveno rješenje predstavljeno je izrazom na slici 1.

$\pi =$ `pt/w-pos((pt+Nr)/pt)+((SL+pt)*(2*SL^2/w-2*w)/Nr^2*w+pos(pos((SL+pt+(2*SL+pt)*(SL+w)/w)/SD*pt*(2*SL^2/w-w-(SL+pt/w+SL/(w+SD)*(Nr+SL))*(Nr+SPr)-pos((Nr+SD)/(Nr+SL)+pt+SL+(2*pt+2*Nr+SD)/(Nr+SL)+pt/w-pos(Nr/pt))))))/SPr+SL`

Slika 1. Zapis rješenja – izvedena funkcija prioriteta

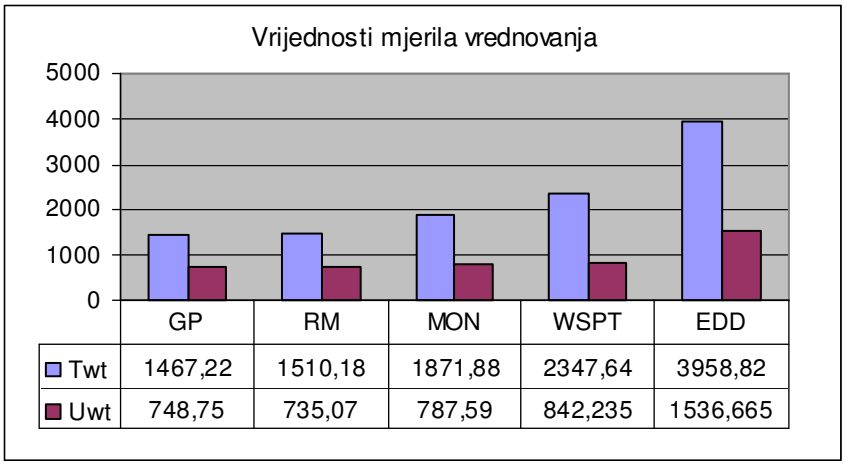
Potrebno je napomenuti da je priroda dobivenog rješenja gotovo uvijek takva da ono nije analitički 'ispravno', budući su u rješenje u većini slučajeva uključeni i elementi koji imaju vrlo mali utjecaj na konačni rezultat, odnosno predstavljaju 'šum'. Ova je osobina svojstvena svim rješenjima u većini primjena genetskog programiranja, no to ne umanjuje njihovu korisnost u uporabi.

Rezultati za sva promatrana pravila dani su u dva dijela; u prvom dijelu prikazan je ukupni iznos funkcije dobrote za sve ispitne primjere. Budući da iznos dobrote predstavlja troškove rasporeda (u evolucijskom procesu provodi se postupak minimiziranja), za prvi dio prikaza rezultata vrijedi pravilo 'manje je bolje'. U drugom dijelu rezultata, za svaki ispitni primjer pronađeno je najbolje rješenje (dobiveno od bilo kojeg pravila), a potom je za svako pravilo utvrđen postotak ispitnih primjera u kojima pravilo daje rješenje jednako najboljemu. Ovaj dio rezultata pokušava pokazati u kolikom dijelu slučajeva bi određeno pravilo pronašlo rješenje koje nije lošije od svih drugih pravila, odnosno u kolikom postotku određeno pravilo dominira nad drugim promatranim pravilima.

U prikazu kvalitete rješenja, poradi potpunijeg uvida u učinkovitost pojedine metode, dani su rezultati i za težinsko zaostajanje (oznaka 'Twt') i za težinsku zakašnjelost (oznaka 'Uwt'), iako se kao kriterij prilikom postupka učenja koristi samo težinsko zaostajanje. Na ovaj način može se doći i do zaključaka o 'korisnosti' određenog kriterija, tj. po kojem je

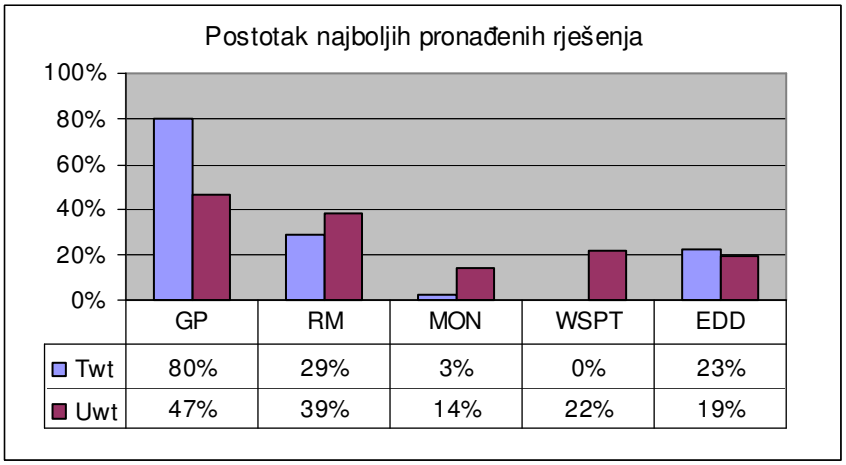
kriteriju najisplativije optimirati a da i ostala mjerila postignu relativno dobru vrijednost. Vrijednosti težinske zakašnjelosti pomnožene su sa odgovarajućim faktorom za pojedini skup primjera kako bi iznosi u grafičkom prikazu bili sumjerljivi sa vrijednostima težinskog zaostajanja.

Na slikama 2 i 3 prikazani su rezultati istoga pravila na skupu ispitnih primjera za ocjenu. Iz prikaza rezultata može se vidjeti da je pravilo dobiveno genetskim programiranjem uspješnije od ostalih promatranih pravila. Sam iznos ukupnog težinskog zaostajanja nije puno bolji od sljedeće najbolje metode, no u broju primjera u kojima se postiže najbolje nađeno rješenje, izvedeno pravilo uvjerljivo prednjači.



	GP	RM	MON	WSPT	EDD
Twt	1467,22	1510,18	1871,88	2347,64	3958,82
Uwt	748,75	735,07	787,59	842,235	1536,665

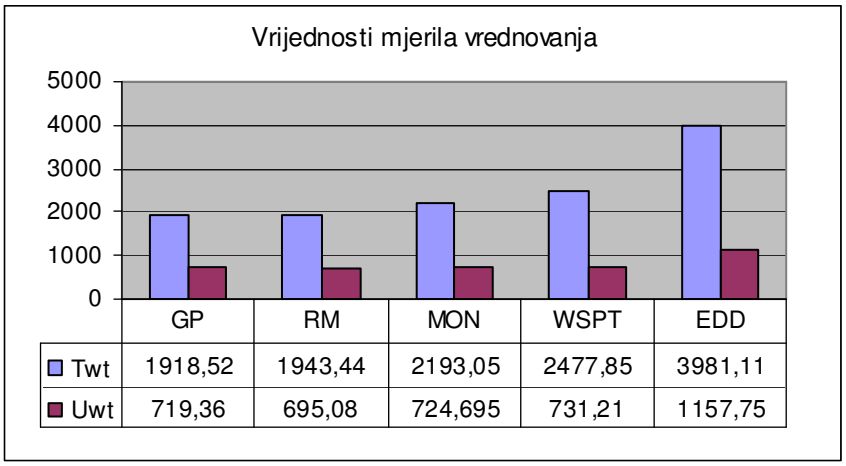
Slika 2 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	RM	MON	WSPT	EDD
Twt	80%	29%	3%	0%	23%
Uwt	47%	39%	14%	22%	19%

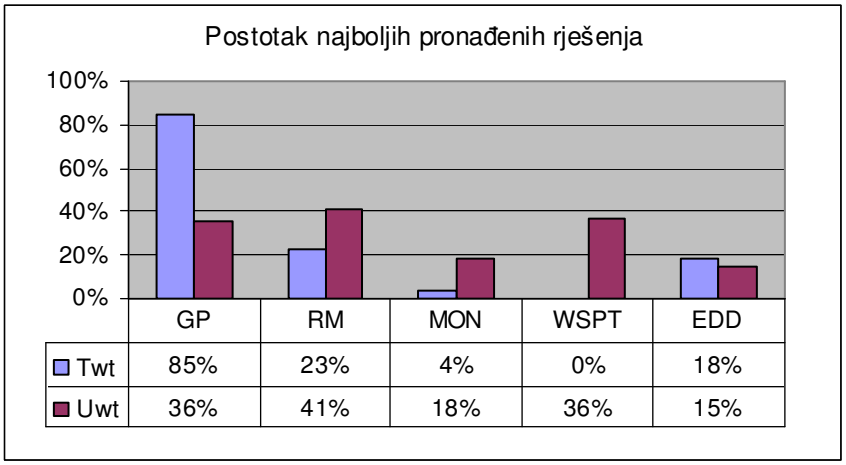
Slika 3 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Konačno, uspješnost istoga pravila uspoređena je na skupu primjera iz literature koji se sastoji od 375 primjera (po 125 primjera sa 40, 50 i 100 poslova). Rezultati na ovom skupu prikazani su koristeći jednaka mjerila na slikama 4 i 5.



	GP	RM	MON	WSPT	EDD
Twt	1918,52	1943,44	2193,05	2477,85	3981,11
Uwt	719,36	695,08	724,695	731,21	1157,75

Slika 4 Optimiranje težinskog zaostajanja – skup primjera iz literature

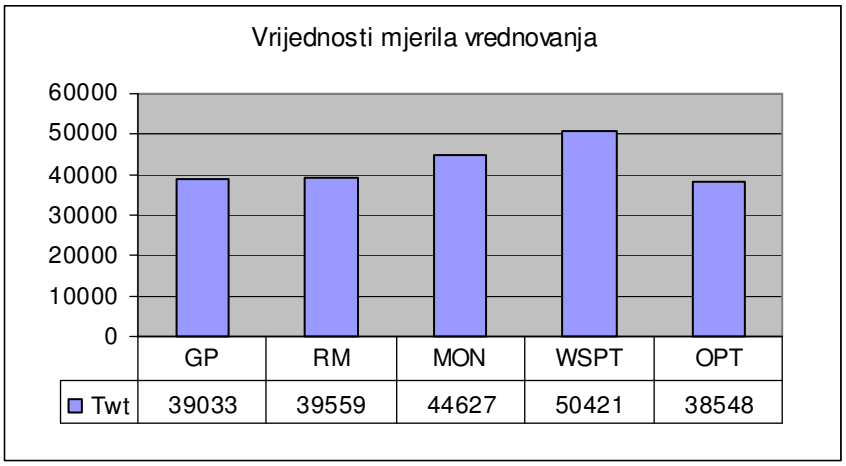


	GP	RM	MON	WSPT	EDD
Twt	85%	23%	4%	0%	18%
Uwt	36%	41%	18%	36%	15%

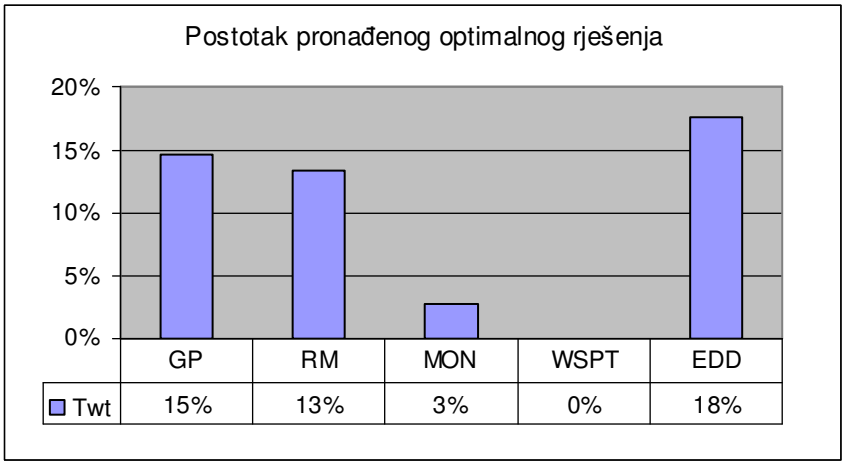
Slika 5 Postoci dominacije za težinsko zaostajanje – skup primjera iz literature

Budući su uz primjere iz literature navedene optimalne vrijednosti rješenja za problem težinskog zaostajanja, zanimljivo je usporediti promatrana pravila uz vrijednost koja se postiže optimalnim rasporedom (oznaka 'OPT'). Potrebno je napomenuti da su optimalne vrijednosti za te ispitne primjere na raspolaganju samo u nenormiranom obliku – iznos kriterija dobiven je po definiciji težinskog zaostajanja bez dijeljenja sa brojevima poslova, srednjom težinskom vrijednošću itd. Bez obzira na tu razliku, pravilo izvedeno za normirane kriterije pokazuje gotovo jednaku učinkovitost i uz ovakav način ocjenjivanja. Na slici 6 prikazane su vrijednosti za ukupno težinsko zaostajanje svih postupaka (osim za EDD čija je postignuta vrijednost duplo veća – lošija od najbolja tri rješenja) i optimalnog rješenja.

Također je zanimljivo pogledati u kojem broju slučajeva određeno pravilo daje optimalno rješenje. Ovi su postoci prikazani na slici 7, a budući da optimalno rješenje ima vrijednost 100%, nije uključeno u prikaz.



Slika 6 Vrijednosti težinskog zaostajanja uz optimalno rješenje



Slika 7 Postoci pronalaženja optimalnog rješenja – skup primjera iz literature

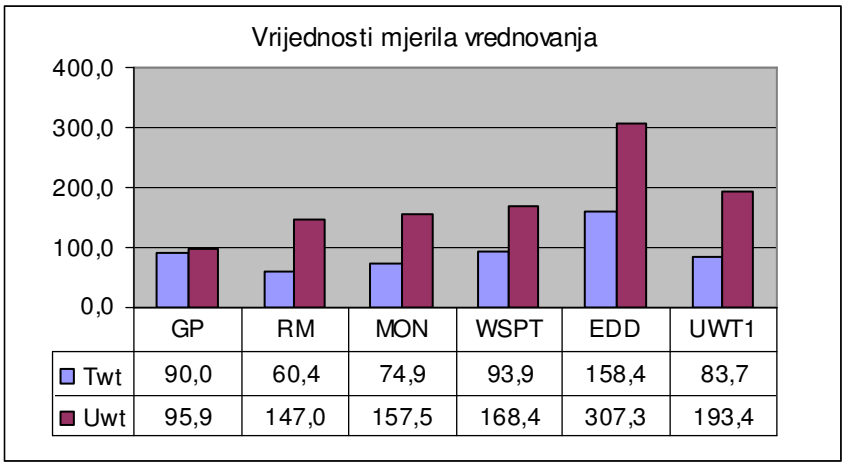
Rezultat koji pokazuje da najveći broj pronalaženja optimalnog rješenja postiže EDD pravilo, iako ima uvjerljivo najlošiju učinkovitost po ukupnom kriteriju, nije iznenađujuć. Naime, svi ispitni primjeri u kojima pravila daju optimalno rješenje gotovo su isključivo primjeri sa 0% očekivanog zaostajanja (tj. parametar T je nula, a parametar R je relativno mali), a vrijednost ukupnog zaostajanja uz optimalni raspored jednaka je nuli. Za slučajeve u kojima niti jedan posao ne zaostaje, može se dokazati [Mor 93] da EDD pravilo daje optimalno rješenje, pa broj takvih slučajeva u ispitnim primjerima izravno utječe na postotak pronalaženja optimuma po EDD pravilu.

Problem težinske zakašnjelosti

Problem težinske zakašnjelosti poslova može se opisati situacijom u kojoj postoje narudžbe i rokovi dostave za neke proizvode, a u slučaju prekoračenja roka za određeni posao plaća se konstantni iznos troška. Iznos troška u slučaju zakašnjelosti ovisi o težini pojedinoga posla ali ne ovisi o količini vremena za koju je posao prekoračio zadani rok. Drugim riječima, nastoji se minimizirati težinski zbroj troškova zakašnjelosti. U $\alpha|\beta|\gamma$ zapisu oznaka problema je $1|\ |\sum w_j U_j$.

Genetski program ovdje ima jednak zadatak kao u slučaju težinskog zaostajanja – cilj je pronaći takvu prioritetnu funkciju koja će, koristeći je u obliku pravila, minimizirati opisane troškove. Algoritam primjene pravila raspoređivanja isti je kao i za prethodni kriterij.

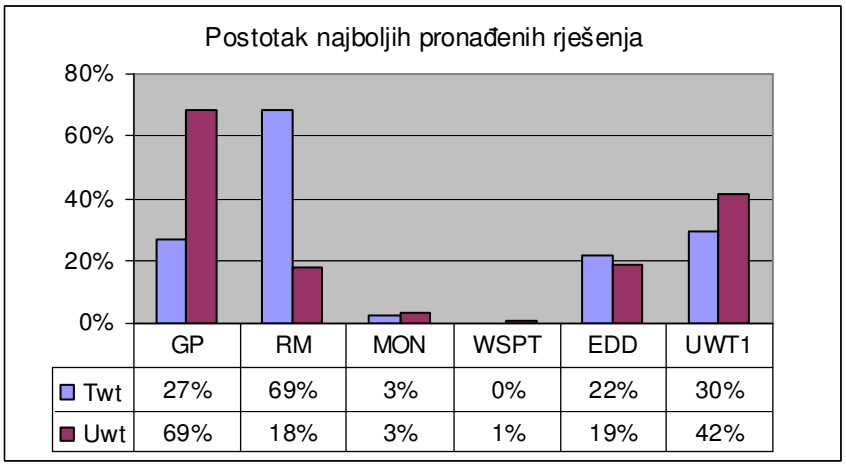
Budući je težinska zakašnjelost mjerilo koje ovisi o sličnim veličinama kao i težinsko zaostajanje, genetskom programu su na raspolaganju jednaki funkcijski i podatkovni elementi kao i u prethodnom primjeru (tablica 1). Prilikom uspoređivanja rješenja, u skup promatranih pravila dodana je i heuristika za problem težinske zakašnjelosti koja je u prikazu rezultata označena sa 'UWT1'. Za potrebe ovoga kriterija provedeno je 20 pokusa sa jednakim vrijednostima parametara kao i u prethodnom primjeru. Najbolje rješenje također je odabrano promatrajući učinkovitost pravila na skupu za ocjenu. Na slikama 8 i 9 prikazani su rezultati svih promatranih pravila na skupu od 600 ispitnih primjera za ocjenu.



	GP	RM	MON	WSPT	EDD	UWT1
Twt	90,0	60,4	74,9	93,9	158,4	83,7
Uwt	95,9	147,0	157,5	168,4	307,3	193,4

Slika 8 Optimiranje težinske zakašnjelosti – skup primjera za ocjenu

Iz rezultata se vidi da rješenje stvoreno genetskim algoritmom uvjerljivo dominira nad ostalim pravilima, no očit je 'pad učinkovitosti' po pitanju drugog kriterija, težinskog zaostajanja. Ukoliko nam je kao mjerilo važna jedino zakašnjelost, ovakav je rezultat i više nego dobar. Ukoliko, međutim, tražimo ujednačenu kvalitetu pravila, tada je bolji izbor optimiranje po kriteriju zaostajanja.



	GP	RM	MON	WSPT	EDD	UWT1
Twt	27%	69%	3%	0%	22%	30%
Uwt	69%	18%	3%	1%	19%	42%

Slika 9 Postoci dominacije za težinsku zakašnjelost – skup primjera za ocjenu

3.2 Dinamička okolina raspoređivanja

U dinamičkoj okolini raspoređivanja uvode se vremena pripravnosti poslova koja označuju trenutak kada može započeti obrada nekog posla. U $\alpha|\beta|\gamma$ zapisu ova okolina ima

oznaku $1|r_j|*$. Najčešća mjerila vrednovanja rasporeda koja se uzimaju u obzir u ovoj okolini su također težinsko zaostajanje i težinska zakašnjelost.

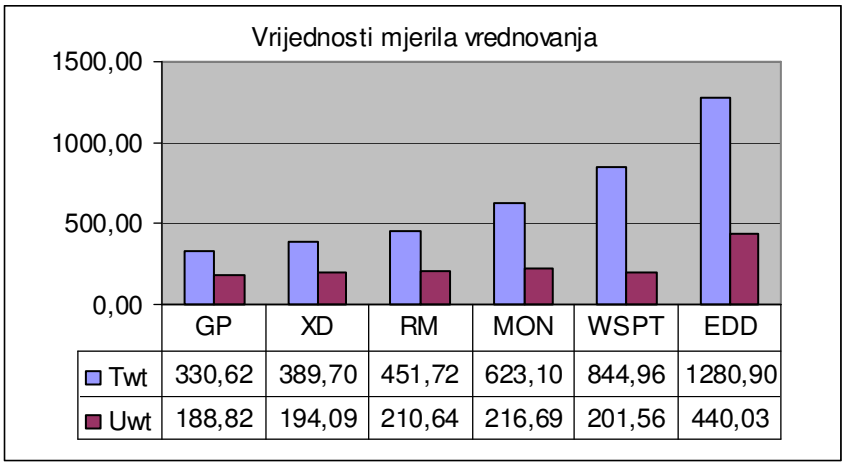
U ovoj je okolini prikaz rješenja genetskog programiranja identičan prikazu u prethodnom poglavlju, tj. jedinka je predstavljena stablom koje zamjenjuje funkciju prioriteta pravila raspoređivanja. I dalje vrijedi pravilo da se najboljim poslom smatra onaj čija je vrijednost funkcije prioriteta najmanja. Procesu učenja potrebno je omogućiti uporabu informacije o vremenu pripravnosti pojedinoga posla, te također definirati koji se poslovi uzimaju u obzir prilikom ocjenjivanja. Stoga je u skup podatkovnih čvorova, u odnosu na skup iz prethodne okoline, dodana količina vremena za koju posao postaje pripravan. Konačni skup čvorova prikazan je u tablici 2.

Tablica 2. Popis čvorova za dinamički problem na jednom stroju

Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
N	ukupni broj poslova
Nr	preostali broj poslova (koji još nisu raspoređeni)
SP	zbroj trajanja svih poslova
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
AR	vrijeme do pripravnosti posla, $\max\{r_j - time, 0\}$

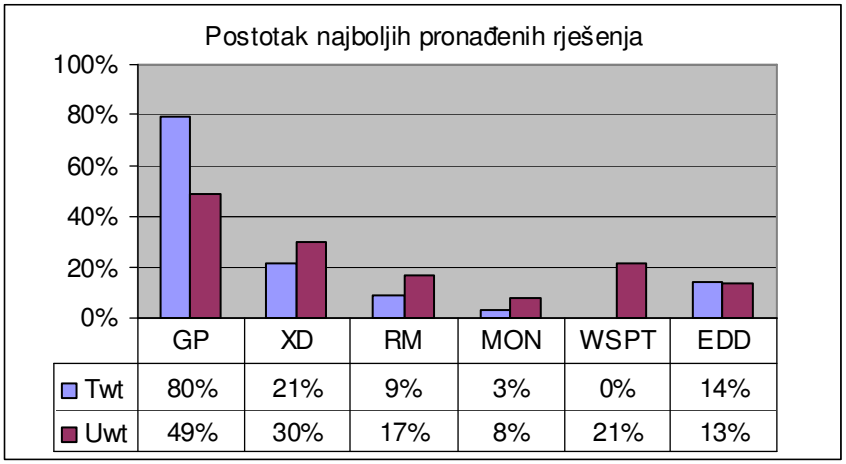
Problem težinskog zaostajanja

Za rješavanje problema težinskog zaostajanja u dinamičkoj okolini provedeno je 20 pokusa Rezultati najboljeg dobivenog rješenja za problem težinskog zaostajanja i skup ispitnih primjera za ocjenu prikazani su na slikama 10 i 11.



Slika 10 Optimiranje težinskog zaostajanja – skup primjera za ocjenu

Iz prikaza rezultata može se vidjeti sljedeće: pravila koja nisu posebno osmišljena za dinamičku okolinu postižu lošija rješenja, a pravilo koje eksplicitno uzima u obzir informaciju o vremenu pripravnosti posla postiže bolje rezultate. Najbolji postignuti rezultat uvjerljivo je pokazalo pravilo izvedeno genetskim programiranjem, uz 15% bolju vrijednost mjerila vrednovanja nego sljedeće najbolje pravilo. Isto tako, vidljivo je da izvedeno pravilo postiže najbolju učinkovitost i na drugom kriteriju (težinsko kašnjenje), što govori u prilog tvrdnji da je težinska zakašnjelost 'isplativiji' kriterij optimiranja.



Slika 11 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

3.3 Ograničenja u redoslijedu zadataka

Jedinka populacije genetskog programiranja predstavljena je stablom koje zamjenjuje funkciju prioriteta pojedinoga posla. Interpretacija prioriteta uzima u obzir samo one poslove koje je moguće rasporediti s obzirom na ograničenja, kao i kod ostalih pravila. Genetskom programu su osim već opisanih podataka na raspolaganju i podaci o razini čvora u stablu ovisnosti te broj neposrednih sljedbenika nekog posla. U tablici 3 prikazan je skup čvorova koji su na raspolaganju genetskom programu u gradnji jedinke.

Tablica 3. Popis čvorova za problem s ograničenjima na jednom stroju

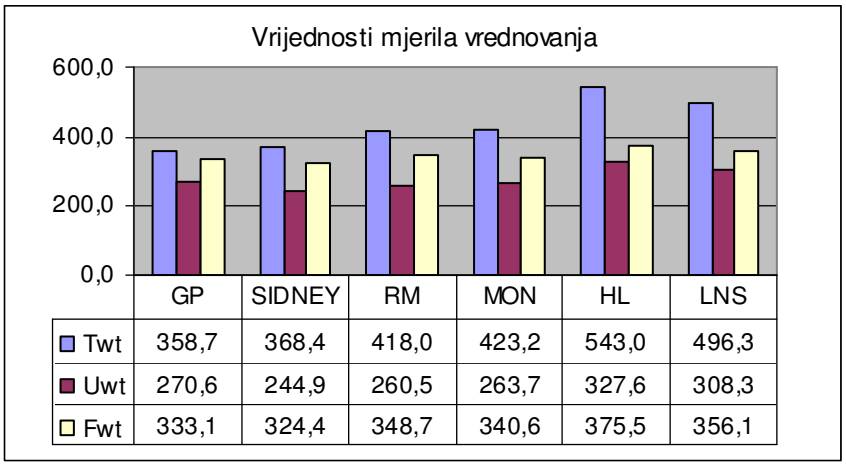
Oznaka funkcijskog čvora	Definicija

ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
N	ukupni broj poslova
Nr	preostali broj poslova (koji još nisu raspoređeni)
SP	zbroj trajanja svih poslova
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
SC	broj neposrednih sljedbenika posla
LVL	razina čvora koji predstavlja posao u grafu ovisnosti

Problem težinskog zaostajanja

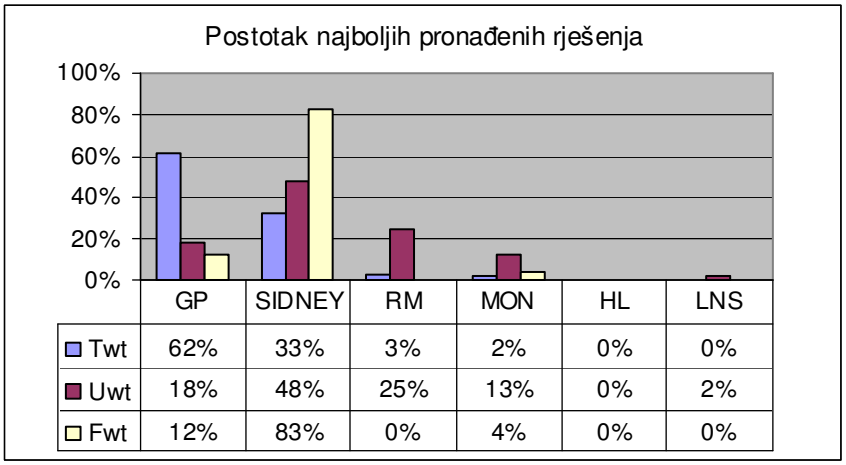
Iako se u ispitnoj okolini sa ograničenjima osim težinskog zaostajanja javljaju još neki netrivijalni kriteriji, najveća se pažnja i dalje posvećuje navedenom mjerilu. Za optimiranje kriterija težinskog zaostajanja provedeno je 20 pokusa, a rezultati su pregledno prikazani na slikama 12 i 13. U postupku usporedbe uključena su i RM te MON pravila zbog relativno dobrih postignutih rezultata. U prikaz je također dodan i kriterij težinskog protjecanja (oznaka: Fwt) poradi naglaska na učinkovitost Sidney heuristike.

Iz prikaza se može primijetiti da su jednostavna HL i LNS pravila postigla vrlo loše vrijednosti na ispitnim primjerima. Nešto bolje su prošla pravila općenite namjene RM i MON koja ne uzimaju u obzir informaciju o ovisnosti zadataka, no svejedno postižu prihvatljive rezultate. Najbolja rješenja dobivaju se uporabom Sidney algoritma i pravila izvedenog genetskim programiranjem. Potrebno je napomenuti da Sidney algoritam dominira nad ostalim pravilima po pitanju uspješnosti na više kriterija istovremeno, što je i očekivano budući da ovaj algoritam predstavlja složeniju proceduru od pravila raspoređivanja koje gradi rješenje u jednom prolazu. Poredbe radi, rješavanje skupa ispitnih primjera za ocjenu Sidney algoritmom traje nekoliko minuta na jednom računalu, dok se za pravila raspoređivanja rješenja dobivaju u manje od 2 sekunde.



	GP	SIDNEY	RM	MON	HL	LNS
Twt	358,7	368,4	418,0	423,2	543,0	496,3
Uwt	270,6	244,9	260,5	263,7	327,6	308,3
Fwt	333,1	324,4	348,7	340,6	375,5	356,1

Slika 12 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	SIDNEY	RM	MON	HL	LNS
Twt	62%	33%	3%	2%	0%	0%
Uwt	18%	48%	25%	13%	0%	2%
Fwt	12%	83%	0%	4%	0%	0%

Slika 13 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

3.4 Slijedno ovisna trajanja postavljanja

Za potrebe izvođenja funkcije prioriteta koja u obzir uzima i trajanje postavljanja, genetskom je programu za svaki ispitivani posao poznato trajanje postavljanja sa prethodnog posla. Osim toga, algoritmu je na raspolaganju i prosječno trajanje postavljanja, no ne za sve poslove već samo srednje trajanje postavljanja sa prethodnog na sve ostale poslove. U tablici 4 prikazan je skup čvorova koji su na raspolaganju genetskom programu u gradnji pojedinog rješenja.

Tablica 4. Popis čvorova uz trajanja postavljanja na jednom stroju

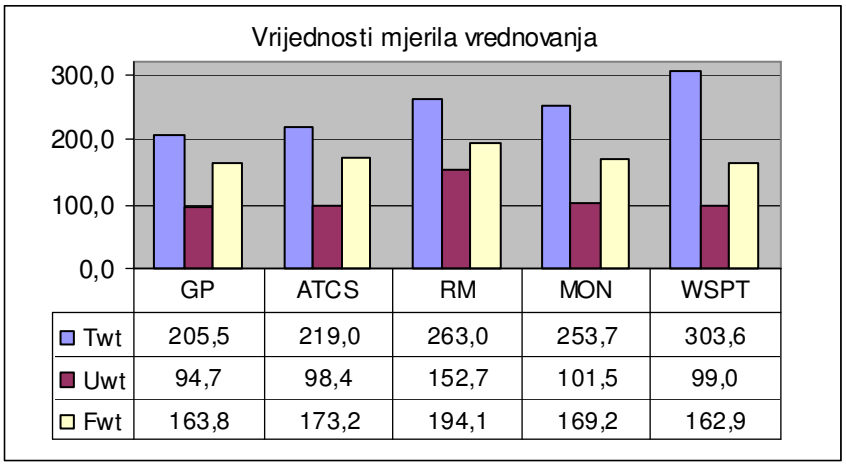
Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)

dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
Nr	preostali broj poslova (koji još nisu raspoređeni)
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
STP	trajanje postavljanja sa prethodnog na promatrani posao, s_{lj}
Sav	prosječno trajanje postavljanja sa prethodnog (l) na sve ostale poslove, $\frac{1}{n-1}\sum_{j=1}^{n} s_{lj}$

Problem težinskog zaostajanja

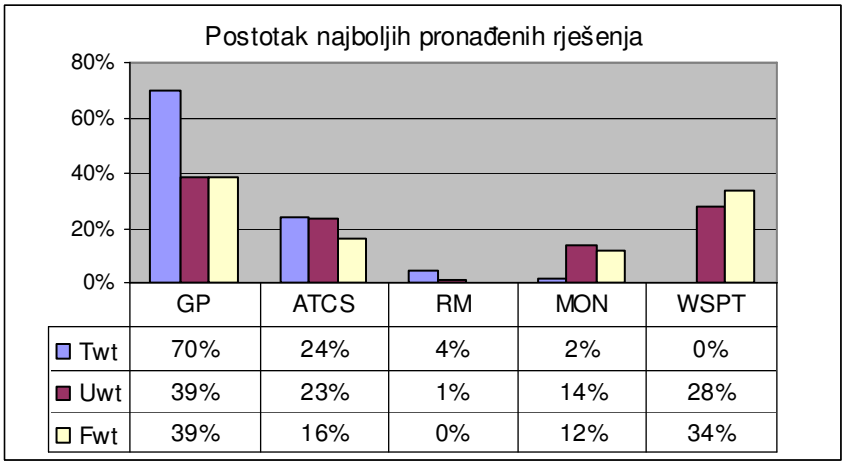
U postupku izvođenja pravila raspoređivanja za problem težinskog zaostajanja provedeno je 20 pokusa uz pretpostavljene vrijednosti parametara. Za potrebe ocjene učinkovitosti prikazani su rezultati pravila ATCS, RM, MON i WSPT. Osim težinskog zaostajanja i težinske zakašnjelosti, uvršten je i kriterij težinskog protjecanja budući je uz ove uvjete raspoređivanja također netrivijalan (moguće je prikazati i ukupno trajanje rasporeda, no nijedna od promatranih metoda, s izuzetkom WSPT pravila, nije namijenjena optimiranju toga kriterija).

U postupku ocjenjivanja pravila dana su dva prikaza rezultata; za jedan od prikaza vrijednost parametra η postavljena je na 0.5, kao i u primjerima za učenje, a za drugi je vrijednost postavljena na 1 (odnosno, promijenjen je omjer prosječnog trajanja postavljanja i trajanja obrade). Na taj se način može ispitati koliko je izvedeno pravilo osjetljivo s obzirom na promjene prosječnog trajanja postavljanja. Na slikama 14 i 15 prikazani su rezultati usporedbe uz $\eta = 0.5$.



	GP	ATCS	RM	MON	WSPT
Twt	205,5	219,0	263,0	253,7	303,6
Uwt	94,7	98,4	152,7	101,5	99,0
Fwt	163,8	173,2	194,1	169,2	162,9

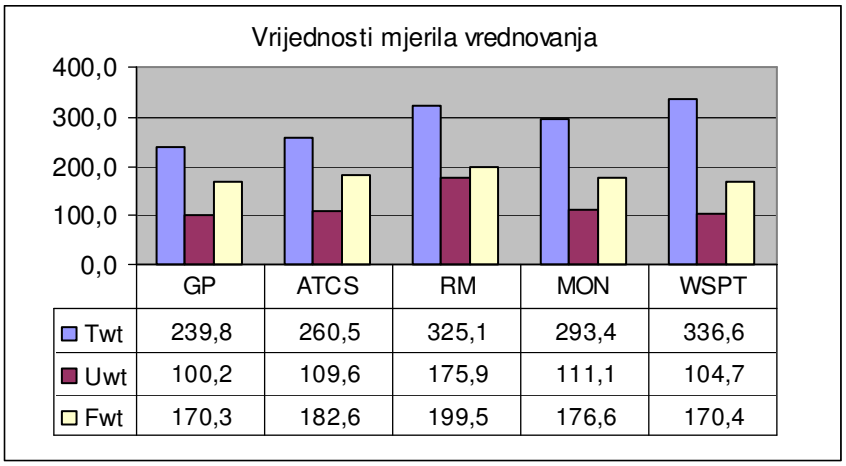
Slika 14 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	ATCS	RM	MON	WSPT
Twt	70%	24%	4%	2%	0%
Uwt	39%	23%	1%	14%	28%
Fwt	39%	16%	0%	12%	34%

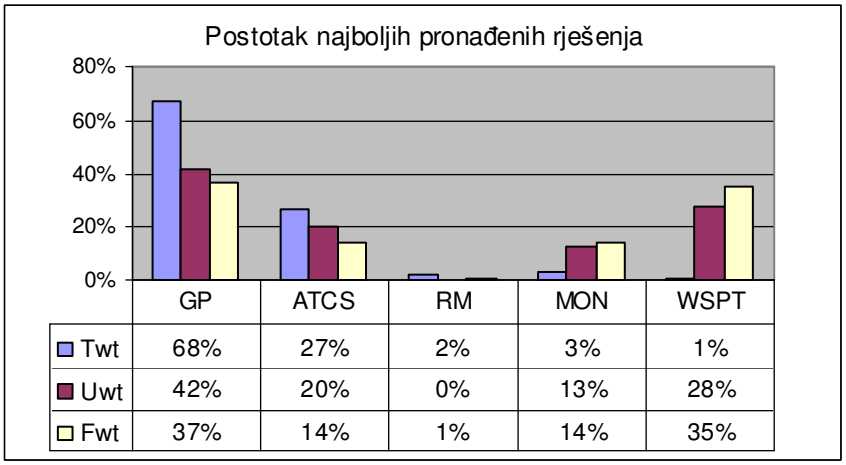
Slika 15 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Iz prikaza je vidljivo da pravilo izvedeno genetskim programiranjem pronalazi najbolje rješenje u većini slučajeva, dok ATCS pravilo prednjači nad svim ostalim promatranim pravilima. Na slikama 16 i 17 prikazani su rezultati uz povećano prosječno trajanje postavljanja, odnosno $\eta = 1$.



	GP	ATCS	RM	MON	WSPT
Twt	239,8	260,5	325,1	293,4	336,6
Uwt	100,2	109,6	175,9	111,1	104,7
Fwt	170,3	182,6	199,5	176,6	170,4

Slika 16 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	ATCS	RM	MON	WSPT
Twt	68%	27%	2%	3%	1%
Uwt	42%	20%	0%	13%	28%
Fwt	37%	14%	1%	14%	35%

Slika 17 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Iz rezultata se može zaključiti da relativna učinkovitost pravila ne ovisi u velikoj mjeri o prosječnom trajanju postavljanja, iako se u primjeni mogu pojaviti i neki primjeri s ekstremnijim vrijednostima.

3.5 Pravila raspoređivanja u složenijim okolinama

U prethodnim odjeljcima, učinkovitost pravila raspoređivanja izvedenog genetskim programiranjem prikazana je u okolinama za koje već postoji relativno velik broj raspoloživih pravila. Opisani ispitni uvjeti predstavljaju najčešće promatrane probleme raspoređivanja na jednom stroju, a rezultati postignuti na tom okruženju mogu se pokazati korisnima u postupcima raspoređivanja složenijih okruženja sa više strojeva. Jedan od ciljeva projekta je pokazati da se i u takvim okolinama mogu izvesti pravila čija je učinkovitost barem toliko dobra kao i učinkovitost postojećih metoda. No još važnije područje primjene izvođenja pravila raspoređivanja su okoline za koje postoji malo prikladnih pravila, a pitanje odabira 'najboljeg' pravila predstavlja dodatni problem. Upravo je u tim uvjetima isplativa uporaba automatiziranog stvaranja pravila prilagođenog promatranom sustavu i danim okolnostima.

U ovom odjeljku prikazat će se izvođenje pravila raspoređivanja za neke manje zastupljene okoline raspoređivanja na jednom stroju. U literaturi se može pronaći velik broj različitih problema raspoređivanja koji se, algoritamski gledano, svode na neka od prethodno

opisanih okruženja ili njihove kombinacije. Budući postoji velik broj mogućih varijacija jednog oblika problema, ovdje će se obraditi samo manji broj primjera.

Raspoređivanje uz trajanja postavljanja i ograničenja u redoslijedu

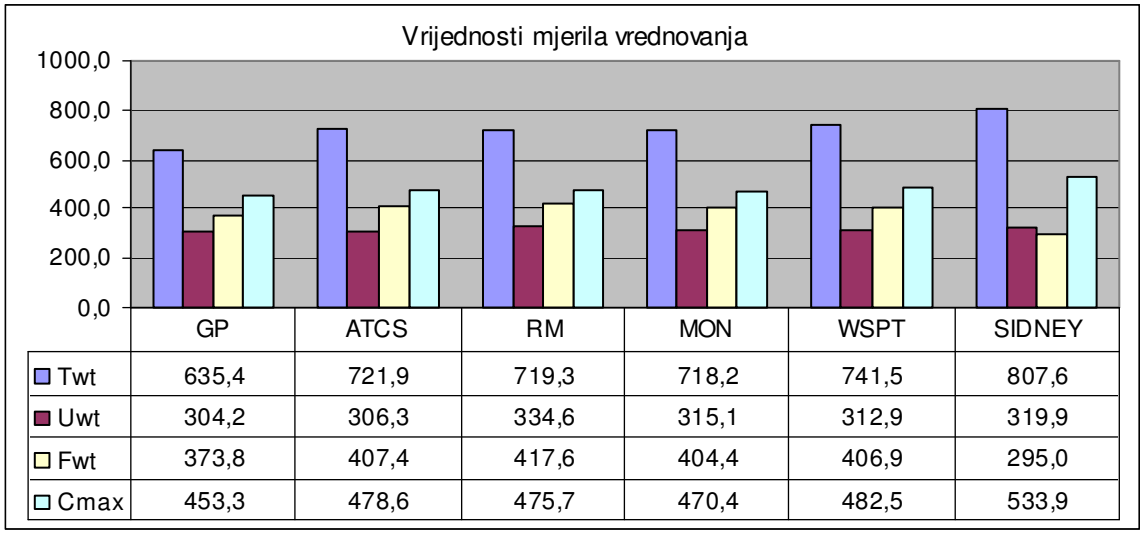
Postojanje slijedno ovisnih trajanja postavljanja može uključivati i istovremeno postojanje ograničenja u redoslijedu izvođenja poslova. Ova okolina raspoređivanja predstavlja dodatni problem jer se u obzir moraju uzimati ograničenja i istovremeno minimizirati troškovi uslijed izmjene poslova. Dok se za slučaj trajanja postavljanja gotovo svako pravilo može prilagoditi, informaciju o ograničenjima je teže uklopiti u neko pravilo. Naravno, uvijek se može postići očuvanje zadanog rasporeda, no za učinkovito rješenje potrebno je na neki način iskoristiti strukturu danih ograničenja. Za potrebe ispitivanja, na ovoj okolini su primijenjena pravila ATCS, RM, MON, WSPT (posljednja tri prilagođena za trajanja postavljanja) te Sidney heuristika. Pravila HL, LNS i EDD su postigla samo značajno lošije rezultate pa nisu niti prikazana u poredbi. Za potrebe definiranja ispitnih primjera rabljena je metodologija opisana u odjeljcima uz ograničenja i trajanja postavljanja (prilikom određivanja trajanja postavljanja ignorira se činjenica da neke kombinacije parova prethodnog i trenutnog posla nisu moguće zbog dodanih ograničenja). Izvođenje pravila genetskim programiranjem odvija se na jednak način kao i u spomenutim odjeljcima, s tim da se skup čvorova u ovom primjeru dobiva unijom skupova za pojedine okoline. Slijedeći ovu strategiju, ukupan skup mogućih čvorova u stablu rješenja prikazan je u tablici 5.

Tablica 5. Popis čvorova uz trajanja postavljanja i ograničenja u redoslijedu

Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
Nr	preostali broj poslova (koji još nisu raspoređeni)
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
STP	trajanje postavljanja s prethodnog na promatrani posao, s_{lj}
Sav	prosječno trajanje postavljanja sa prethodnog (l) na sve ostale poslove, $\frac{1}{n-1}\sum_{j=1}^{n} s_{lj}$

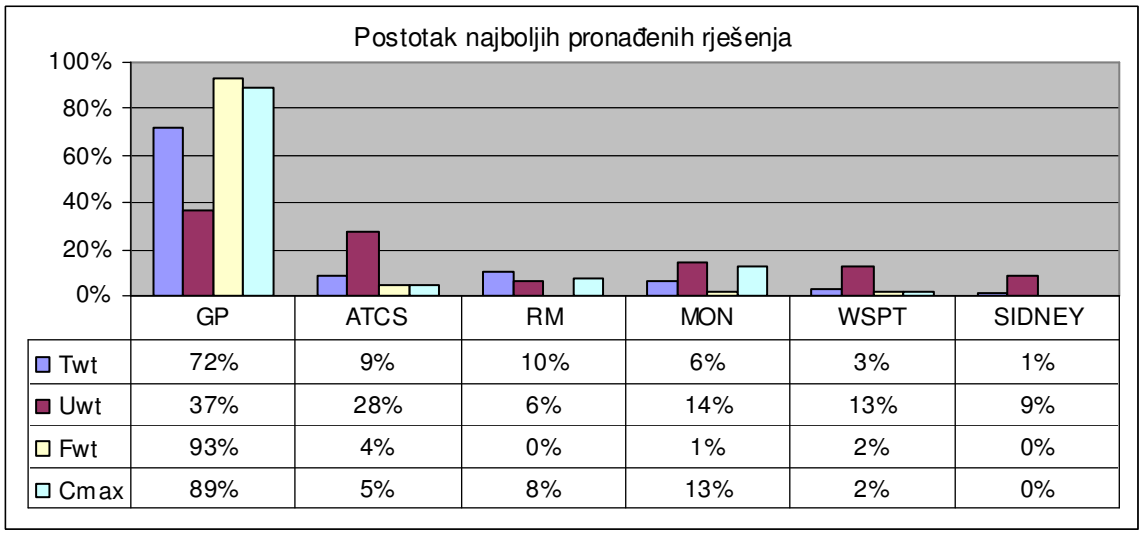
SC	broj neposrednih sljedbenika posla
LVL	razina čvora koji predstavlja posao u grafu ovisnosti

U postupku izvođenja pravila provedeno je 20 pokusa uz pretpostavljene parametre evolucijskog procesa. Proces genetskog programiranja i dalje se temelji na optimiranju težinskog zaostajanja, budući se to mjerilo pokazalo isplativim u smislu učinkovitosti sa gledišta više kriterija. Na slikama 18 i 19 prikazani su usporedni rezultati za 600 ispitnih primjera za ocjenu.



	GP	ATCS	RM	MON	WSPT	SIDNEY
Twt	635,4	721,9	719,3	718,2	741,5	807,6
Uwt	304,2	306,3	334,6	315,1	312,9	319,9
Fwt	373,8	407,4	417,6	404,4	406,9	295,0
Cmax	453,3	478,6	475,7	470,4	482,5	533,9

Slika 18 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	ATCS	RM	MON	WSPT	SIDNEY
Twt	72%	9%	10%	6%	3%	1%
Uwt	37%	28%	6%	14%	13%	9%
Fwt	93%	4%	0%	1%	2%	0%
Cmax	89%	5%	8%	13%	2%	0%

Slika 19 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Pravilo dobiveno genetskim programiranjem pokazuje bolje rezultate za sve kriterije osim za težinsko protjecanje, gdje najbolji rezultat daje Sidney heuristika. Iako je postotak dominacije genetskog programa veći od dominacije za Sidney heuristiku, očito postoji određen broj primjera gdje je izvedeno pravilo postiglo značajno lošiji rezultat po pitanju

težinskog protjecanja. S druge strane, gledamo li težinsko zaostajanje ili ukupnu duljinu rasporeda, vidljiva je dominacija izvedenog pravila.

Raspoređivanje u dinamičkoj okolini uz trajanja postavljanja

Većina opisanih ispitnih okolina, kao i većina ispitnih primjera iz literature, pretpostavlja statičku okolinu raspoređivanja, gdje su svi poslovi pripravni od početka rada sustava. Za razliku od toga, dinamička okolina uključuje i dodatni uvjet u obliku vremena pripravnosti poslova, što u velikom broju slučajeva vjernije prikazuje stvarne uvjete raspoređivanja. Ukoliko u sustavu dopuštamo dinamičke dolaske poslova, postavlja se pitanje kada je moguće započeti proces postavljanja nekoga posla na stroju. U ovakvim slučajevima teorija raspoređivanja ne daje konkretne odgovore, pa će se u ovom radu pretpostaviti da postavljanje stroja za neki posao ne može započeti prije vremena pripravnosti toga posla (ova odluka je u skladu sa većinom sustava raspoređivanja [Lek 03]).

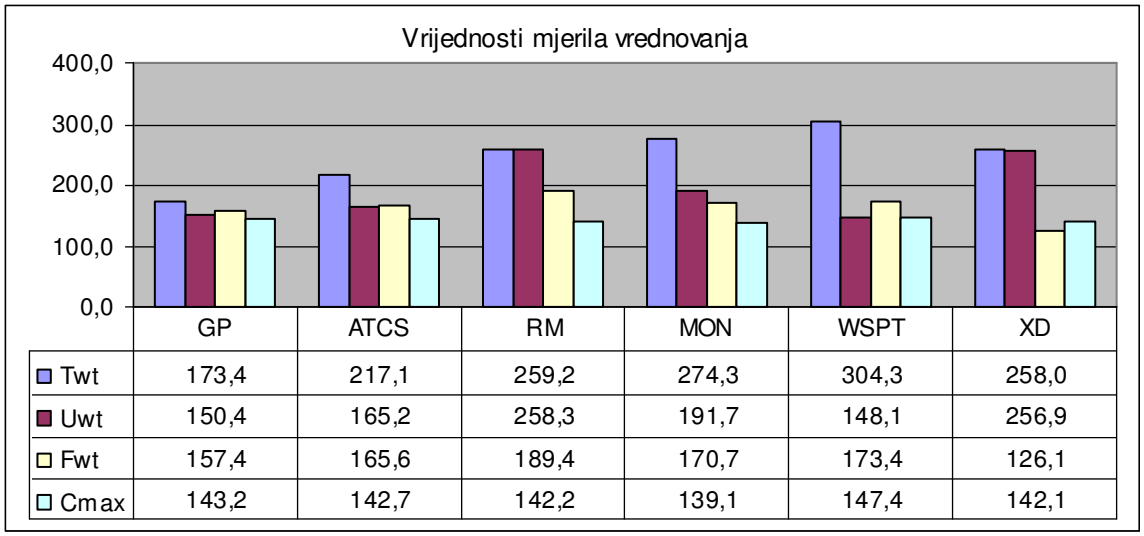
Promatrani postupci raspoređivanja u ovoj okolini su ATCS, MON, RM, XD i WSPT pravilo, od kojih su posljednja četiri prilagođena za trajanja postavljanja. Slično prethodnom primjeru, skup mogućih čvorova koji čine stablo jedinke dobiven je unijom elemenata skupova za dinamički problem i za problem slijedno ovisnih trajanja postavljanja, a definiran je tablicom 6.

Tablica 6. Popis čvorova za dinamički problem uz trajanja postavljanja

Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
Nr	preostali broj poslova (koji još nisu raspoređeni)
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
STP	trajanje postavljanja sa prethodnog na promatrani posao, s_{lj}
Sav	prosječno trajanje postavljanja sa prethodnog (l) na sve ostale poslove, $\frac{1}{n-1}\sum_{j=1}^{n} s_{lj}$

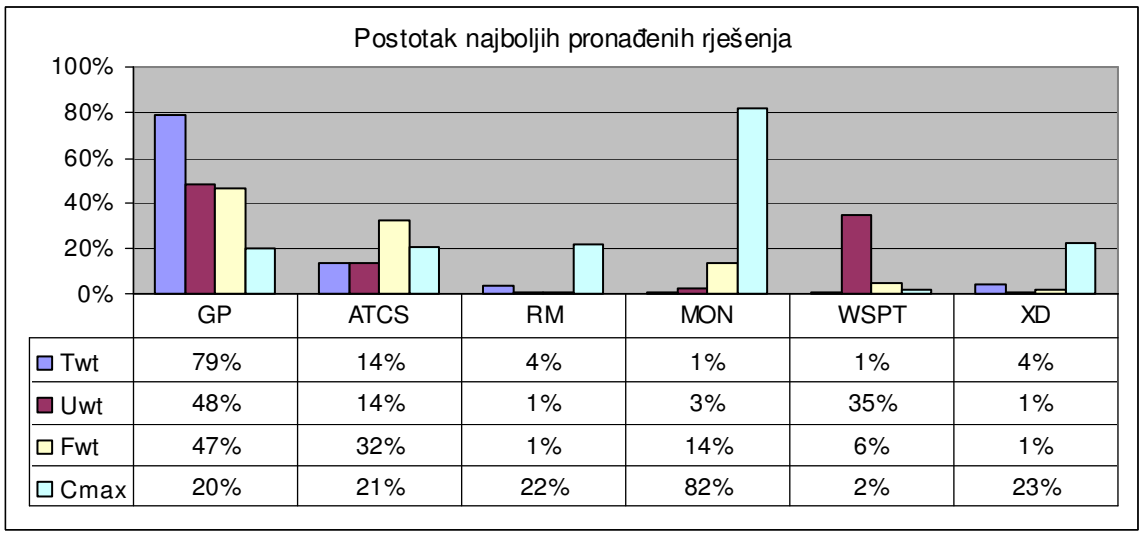
AR	vrijeme do pripravnosti posla, $\max\{r_j - time, 0\}$

Kao kriterij optimiranja uporabljeno je težinsko zaostajanje, te je uz pretpostavljene parametre provedeno 20 pokusa. Na slikama 20 i 21 mogu se vidjeti rezultati usporedbe na ispitnim primjerima za ocjenu.



	GP	ATCS	RM	MON	WSPT	XD
Twt	173,4	217,1	259,2	274,3	304,3	258,0
Uwt	150,4	165,2	258,3	191,7	148,1	256,9
Fwt	157,4	165,6	189,4	170,7	173,4	126,1
Cmax	143,2	142,7	142,2	139,1	147,4	142,1

Slika 20 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	ATCS	RM	MON	WSPT	XD
Twt	79%	14%	4%	1%	1%	4%
Uwt	48%	14%	1%	3%	35%	1%
Fwt	47%	32%	1%	14%	6%	1%
Cmax	20%	21%	22%	82%	2%	23%

Slika 21 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Iz rezultata se može primijetiti da su različita pravila postigla najbolji rezultat za pojedine kriterije: WSPT pravilo daje najbolju vrijednost za težinsku zakašnjelost ('Uwt'), XD pravilo za težinsko protjecanje ('Fwt'), MON pravilo za ukupnu duljinu rasporeda ('Cmax'), a pravilo dobiveno genetskim programiranjem za osnovni kriterij težinskog zaostajanja. Promatramo li stupanj dominacije pravila, izvedeno pravilo prednjači nad ostalima u broju najboljih rješenja, osim za kriterij ukupne duljine rasporeda, gdje uvjerljivo najbolji postotak postiže MON pravilo.

Raspoređivanje u dinamičkoj okolini uz ograničenja u redoslijedu

Kao posljednji primjer u ovom odjeljku promatra se problem raspoređivanja s ograničenjima u redoslijedu uz dinamičke dolaske poslova. Iako se na ovakav primjer rijetko nailazi u stvarnim uvjetima, sasvim je moguće zamisliti i takvu okolinu raspoređivanja; npr. rad sustava može uključivati obradu podataka sa vanjskih mjernih jedinica, s time da se pojedini dijelovi izračuna moraju obaviti prije nekih drugih, a podaci u sustav dolaze u različitim vremenskim trenucima.

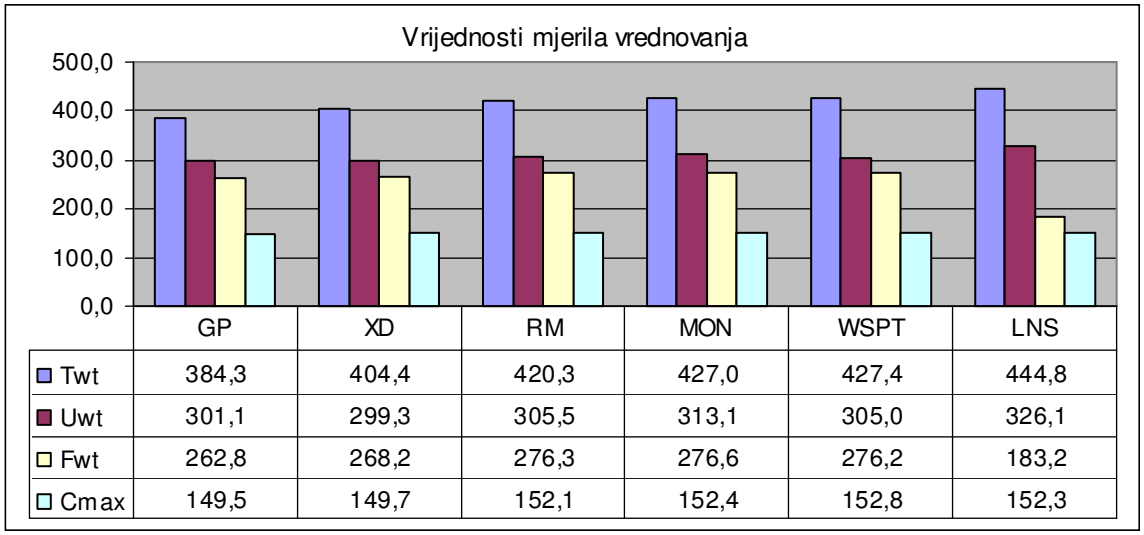
Ispitni primjeri su za ovu okolinu generirani uz proizvoljnu ovisnost poslova te uz proizvoljna vremena pripravnosti. Uz ovakav način definiranja ispitnih primjera, moguće je neslaganje vremena pripravnosti i zadanog redoslijeda poslova (tj. posao prethodnik može imati kasnije vrijeme pripravnosti od posla sljedbenika). Ovakav pristup u svakom slučaju usložnjava postupak raspoređivanja, jer algoritam mora uzeti u obzir i informaciju o dolasku i o poretku poslova kako bi donio što bolju odluku. Takvi algoritmi su vrlo rijetki, pa je smisleno s pomoću postupka učenja oblikovati pravilo prilagođeno zadanim uvjetima. Promatrana pravila raspoređivanja u ovom primjeru su XD, RM, MON, WSPT i LNS pravilo. Ostala pravila nisu uvrštena u rezultate jer daju znatno lošija rješenja – Sidney heuristika, na primjer, budući podrazumijeva statičku raspoloživost poslova, postiže i dvostruko lošije vrijednosti od navedenih pravila.

Izvođenje pravila genetskim programiranjem odvija se na način jednak već opisanome, uz skup mogućih čvorova stabla prikazan u tablici 7.

Tablica 7. Popis čvorova za dinamički problem uz ograničenja u redoslijedu

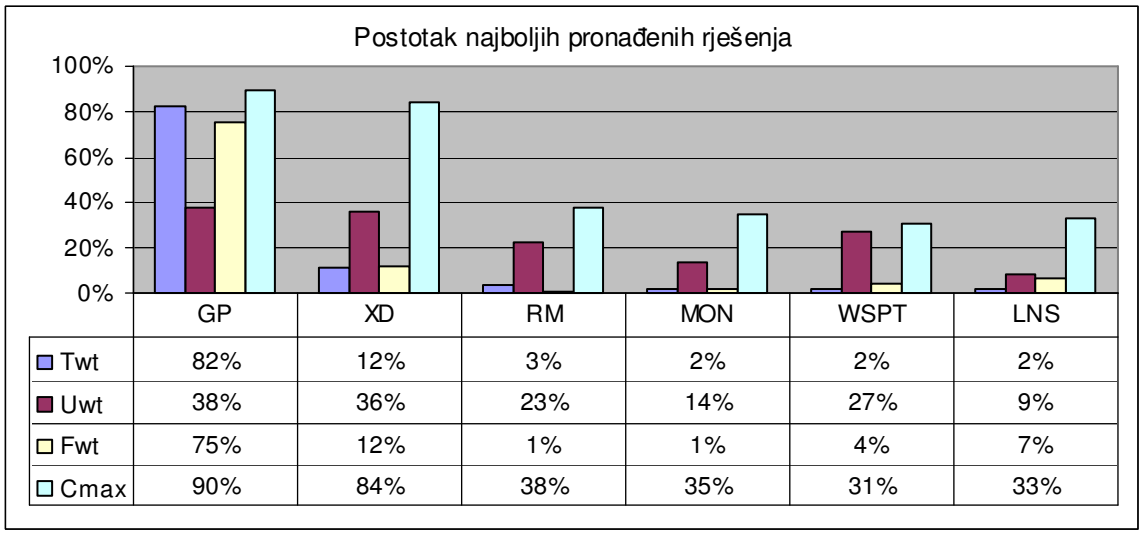
Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje obrade (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
Nr	preostali broj poslova (koji još nisu raspoređeni)
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
AR	vrijeme do pripravnosti posla, $\max\{r_j - time, 0\}$
SC	broj neposrednih sljedbenika posla
LVL	razina čvora koji predstavlja posao u grafu ovisnosti

Za potrebe optimiranja opisane okoline provedeno je 20 pokusa. Usporedni rezultati za ispitne primjere za ocjenu prikazani su na slikama 22 i 23.



	GP	XD	RM	MON	WSPT	LNS
Twt	384,3	404,4	420,3	427,0	427,4	444,8
Uwt	301,1	299,3	305,5	313,1	305,0	326,1
Fwt	262,8	268,2	276,3	276,6	276,2	183,2
Cmax	149,5	149,7	152,1	152,4	152,8	152,3

Slika 22 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	XD	RM	MON	WSPT	LNS
Twt	82%	12%	3%	2%	2%	2%
Uwt	38%	36%	23%	14%	27%	9%
Fwt	75%	12%	1%	1%	4%	7%
Cmax	90%	84%	38%	35%	31%	33%

Slika 23 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

U ovom slučaju rješenje dobiveno genetskim programiranjem dominira nad ostalim pravilima za sve promatrane kriterije, mada je postupak učenja uvažavao samo težinsko zaostajanje. Približno jednak rezultat za težinsku zakašnjelost i ukupnu duljinu rasporeda postiže XD pravilo, a iz postotaka dominacije može se vidjeti da oba postupka u velikom broju slučajeva postižu jednaku vrijednost po pitanju ukupne duljine rasporeda.

4 Raspoređivanje na paralelnim jednolikim strojevima

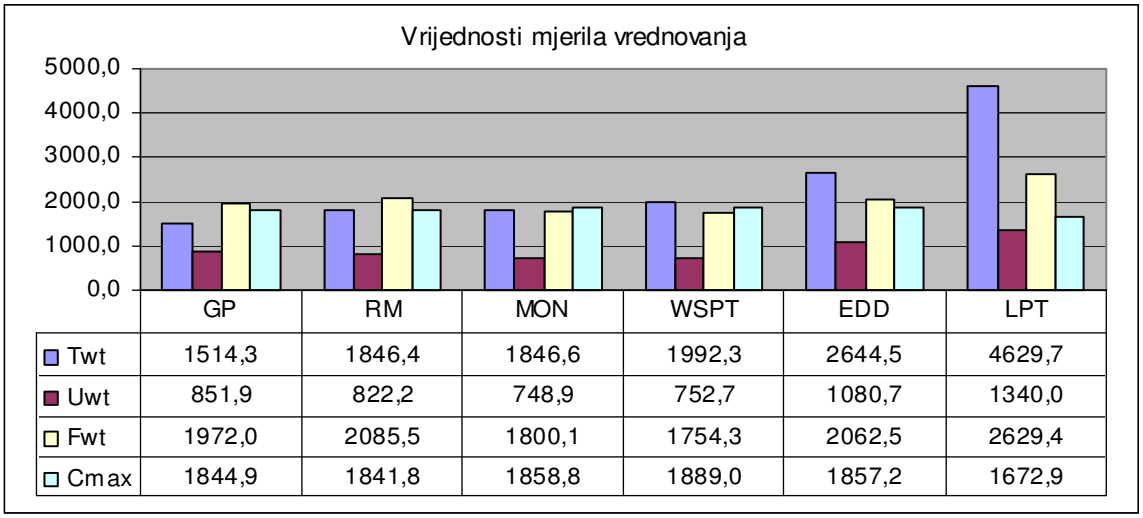
4.1 Statička okolina raspoređivanja

U statičkoj okolini svi poslovi su raspoloživi od početka rada sustava, a jednaka pretpostavka vrijedi i za strojeve. Zadatak genetskog programiranja je naći funkciju prioriteta koja će, dati što bolje rješenje po pitanju zadanog mjerila vrednovanja rasporeda. Jedinka genetskog programa je i u ovom okruženju predstavljena stablom koje predstavlja funkciju prioriteta. Većina naziva čvorova zadržana je iz prethodnog okruženja, iako je definicija vrijednosti za neke čvorove promijenjena. Genetskom je programu na raspolaganju skup čvorova prikazan u tablici 8.

Tablica 8. Popis čvorova za raspoređivanje na jednolikim strojevima

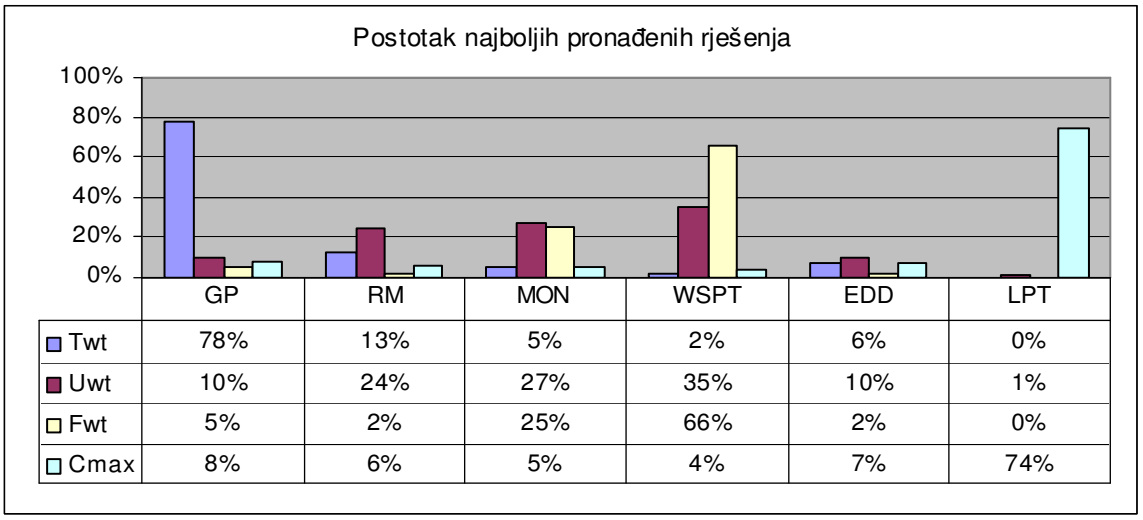
Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	**Definicija vrijednosti podatkovnog čvora**
pt	nominalno trajanje obrade posla (p_j)
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
Nr	preostali broj poslova (koji još nisu raspoređeni)
SPr	zbroj trajanja preostalih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda, $\max\{d_j - p_j - time, 0\}$
SLs	dopuštena odgoda uz brzinu stroja, $\max\{d_j - p_j / s_k - time, 0\}$
SPD	brzina stroja na kojemu se trenutno računaju prioriteti (s_k)
Msm	zbroj brzina svih strojeva, tj. efektivni broj strojeva ($\hat{m}$)
STP	trajanje postavljanja sa prethodnog na promatrani posao, s_{lj}
Sav	prosječno trajanje postavljanja sa prethodnog (l) na sve ostale poslove, $\frac{1}{n-1}\sum_{j=1}^{n} s_{lj}$

U statičkoj okolini provedene su dvije vrste pokusa: u jednoj skupini rješavan je osnovni problem težinskog zaostajanja, a u drugoj skupini isti problem uz trajanja postavljanja. Za osnovni problem bez trajanja postavljanja provedeno je 20 pokusa. Za skup ispitnih primjera za ocjenu, usporedni rezultati prikazani su na slikama 24 i 25.



	GP	RM	MON	WSPT	EDD	LPT
Twt	1514,3	1846,4	1846,6	1992,3	2644,5	4629,7
Uwt	851,9	822,2	748,9	752,7	1080,7	1340,0
Fwt	1972,0	2085,5	1800,1	1754,3	2062,5	2629,4
Cmax	1844,9	1841,8	1858,8	1889,0	1857,2	1672,9

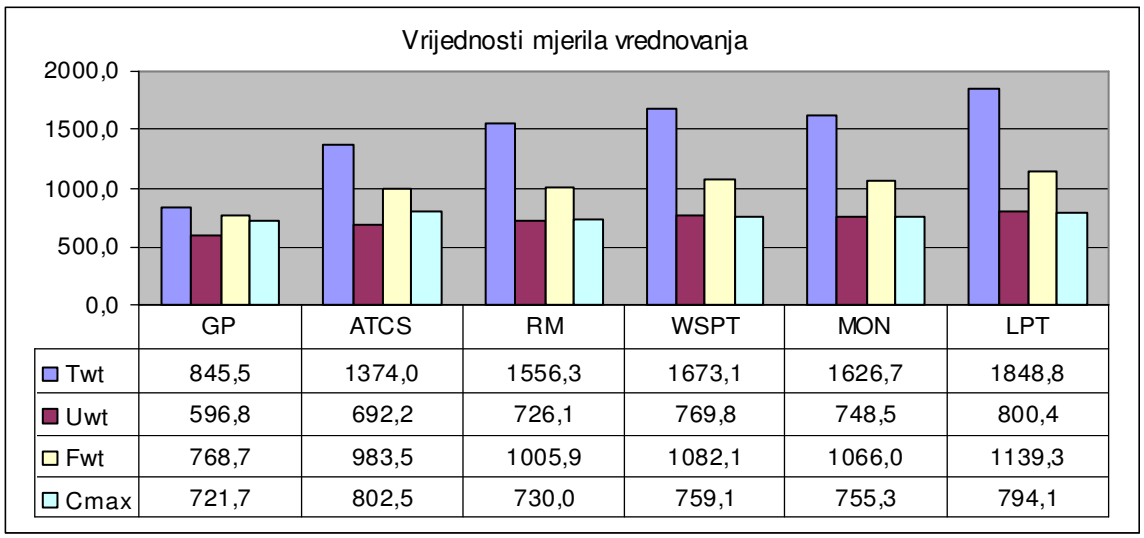
Slika 24 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	RM	MON	WSPT	EDD	LPT
Twt	78%	13%	5%	2%	6%	0%
Uwt	10%	24%	27%	35%	10%	1%
Fwt	5%	2%	25%	66%	2%	0%
Cmax	8%	6%	5%	4%	7%	74%

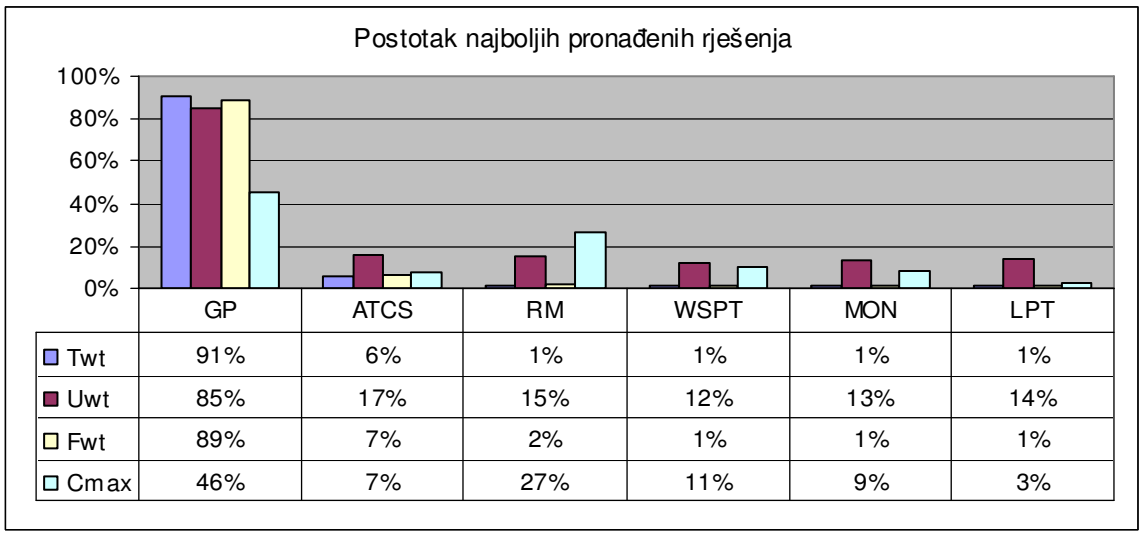
Slika 25 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Vidljivo je da su kriteriji optimiranja uglavnom podijeljeni među heuristikama: pravilo dobiveno genetskim programiranjem uvjerljivo je najbolje po pitanju težinskog zaostajanja, WSPT za težinsko protjecanje a LPT pravilo postiže najbolje rezultate za ukupnu duljinu rasporeda. Za problem sa trajanjima postavljanja također je provedeno 20 pokusa, a u promatrana pravila uvršteno je i ATCS pravilo namijenjeno ovoj vrsti problema. Na slikama 26 i 27 prikazani su usporedni rezultati po pravilima.



	GP	ATCS	RM	WSPT	MON	LPT
Twt	845,5	1374,0	1556,3	1673,1	1626,7	1848,8
Uwt	596,8	692,2	726,1	769,8	748,5	800,4
Fwt	768,7	983,5	1005,9	1082,1	1066,0	1139,3
Cmax	721,7	802,5	730,0	759,1	755,3	794,1

Slika 26 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



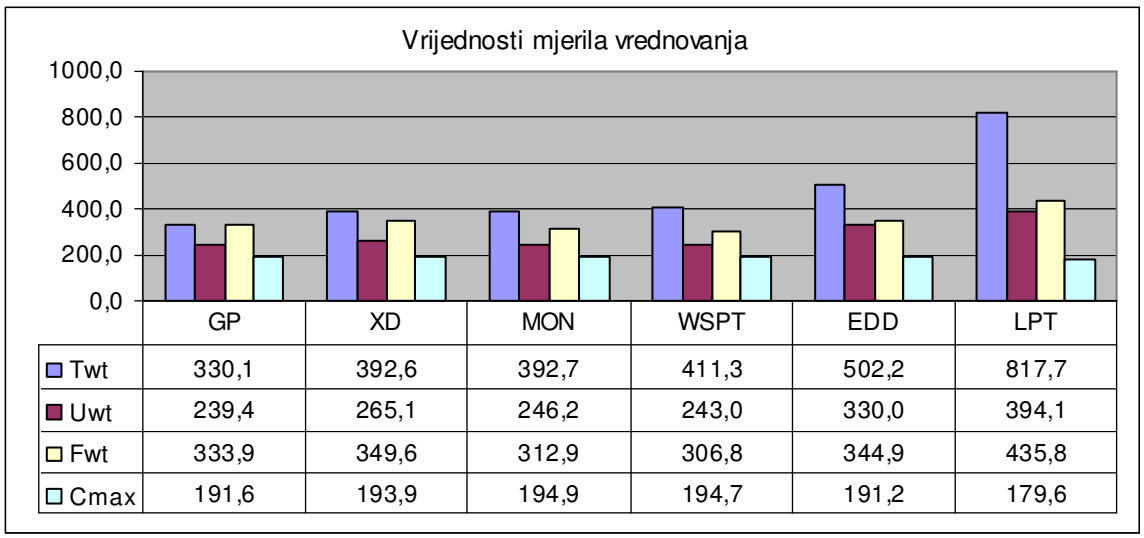
	GP	ATCS	RM	WSPT	MON	LPT
Twt	91%	6%	1%	1%	1%	1%
Uwt	85%	17%	15%	12%	13%	14%
Fwt	89%	7%	2%	1%	1%	1%
Cmax	46%	7%	27%	11%	9%	3%

Slika 27 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Iz rezultata se može uočiti da genetsko programiranje lakše nalazi rješenje koje je relativno bolje od ostalih pravila u slučaju 'manje uobičajene' okoline raspoređivanja, kao što je postojanje trajanja postavljanja za jednolike paralelne strojeve.

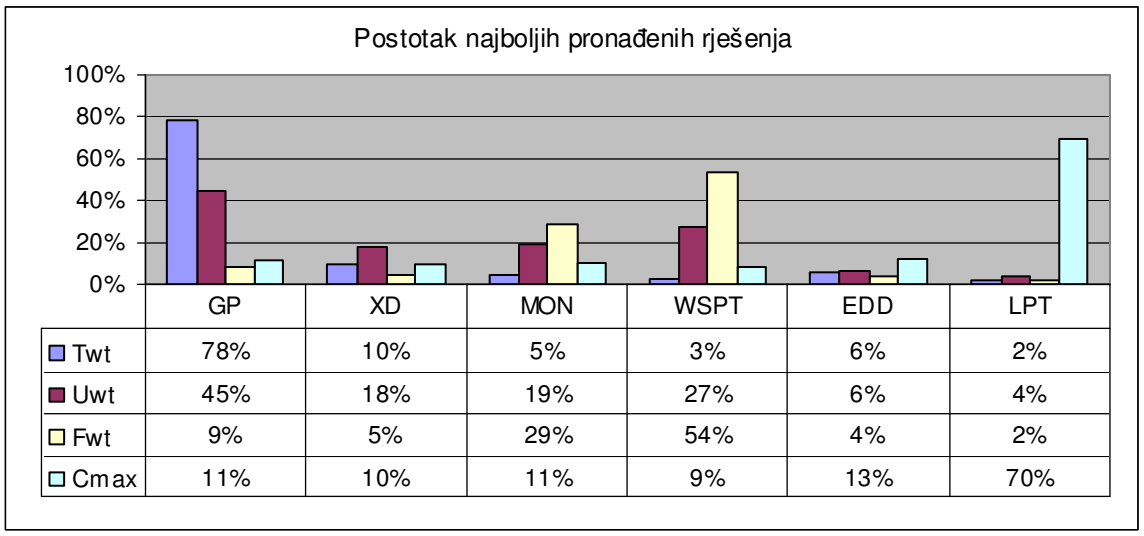
4.2 Dinamička okolina raspoređivanja

U dinamičkoj okolini uvedena su i vremena pripravnosti poslova, dok su strojevi i dalje raspoloživi od početka rada sustava. U postupku izvođenja pravila raspoređivanja genetskim programiranjem korišten je isti skup čvorova kao u tablici 8. Provedeni pokusi su podijeljeni u dvije skupine, ovisno o postojanju trajanja postavljanja među poslovima. Za obje okoline provedeno je optimiranje težinskog zaostajanja i ukupne duljine rasporeda. Za sve inačice problema načinjeno je po 20 pokusa. Za problem bez trajanja postavljanja uz optimiranje težinskog zaostajanja rezultati su prikazani na slikama 28 i 29.



	GP	XD	MON	WSPT	EDD	LPT
Twt	330,1	392,6	392,7	411,3	502,2	817,7
Uwt	239,4	265,1	246,2	243,0	330,0	394,1
Fwt	333,9	349,6	312,9	306,8	344,9	435,8
Cmax	191,6	193,9	194,9	194,7	191,2	179,6

Slika 28 Optimiranje težinskog zaostajanja – skup primjera za ocjenu

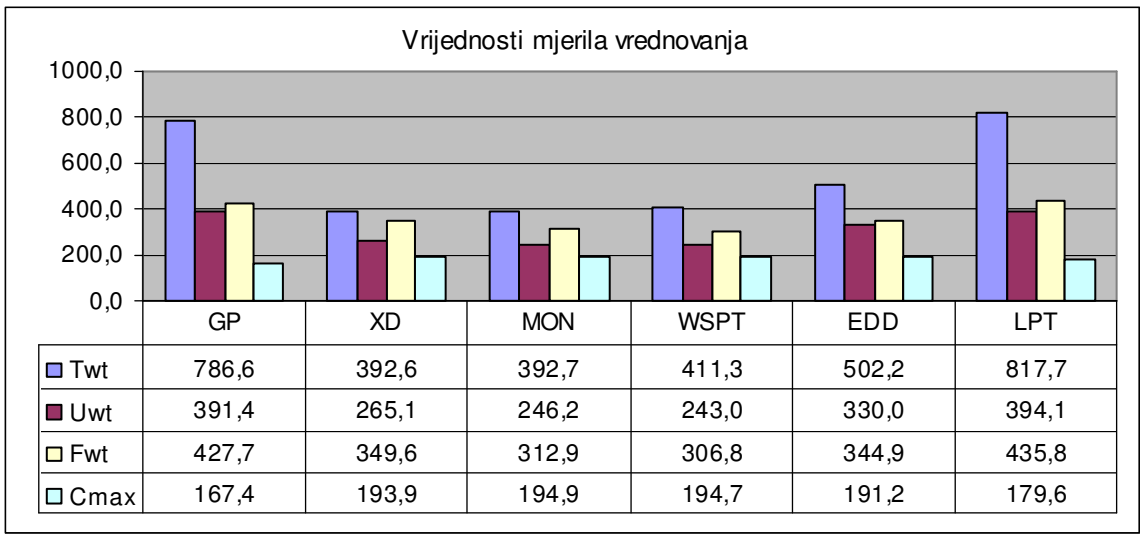


	GP	XD	MON	WSPT	EDD	LPT
Twt	78%	10%	5%	3%	6%	2%
Uwt	45%	18%	19%	27%	6%	4%
Fwt	9%	5%	29%	54%	4%	2%
Cmax	11%	10%	11%	9%	13%	70%

Slika 29 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

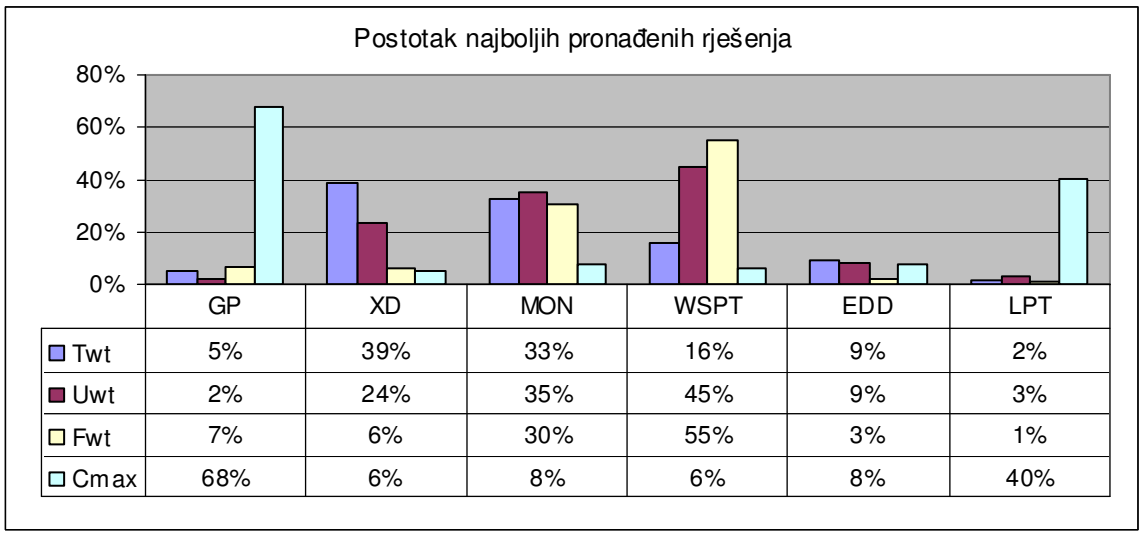
Iz prikaza rezultata je vidljivo da pravilo dobiveno genetskim programiranjem nalazi najbolje rješenje po pitanju težinskog zaostajanja. Za težinsko protjecanje najbolje rješenje postiže se WSPT pravilom, a najbolju ukupnu duljinu rasporeda daje (očekivano) LPT pravilo. Na jednakom skupu promatranih pravila uspoređeni su i rezultati optimiranja ukupne duljine rasporeda, koji su prikazani na slikama 30 i 31.

Može se uočiti da je pravilo izvedeno genetskim programiranjem uspješno po kriteriju ukupne duljine rasporeda, dok je učinkovitost slaba za ostale kriterije. Iz rezultata je također vidljivo da je teško postići visoku kvalitetu rješenja za više kriterija istovremeno. Tražimo li otprilike jednaku učinkovitost po više kriterija, najbolje je upotrijebiti neko od postojećih pravila 'opće namjene', no želimo li optimirati pojedino mjerilo, tada stvaranje prikladnog pravila postupkom učenja predstavlja dobar izbor.



	GP	XD	MON	WSPT	EDD	LPT
Twt	786,6	392,6	392,7	411,3	502,2	817,7
Uwt	391,4	265,1	246,2	243,0	330,0	394,1
Fwt	427,7	349,6	312,9	306,8	344,9	435,8
Cmax	167,4	193,9	194,9	194,7	191,2	179,6

Slika 30 Optimiranje ukupne duljine rasporeda – skup primjera za ocjenu

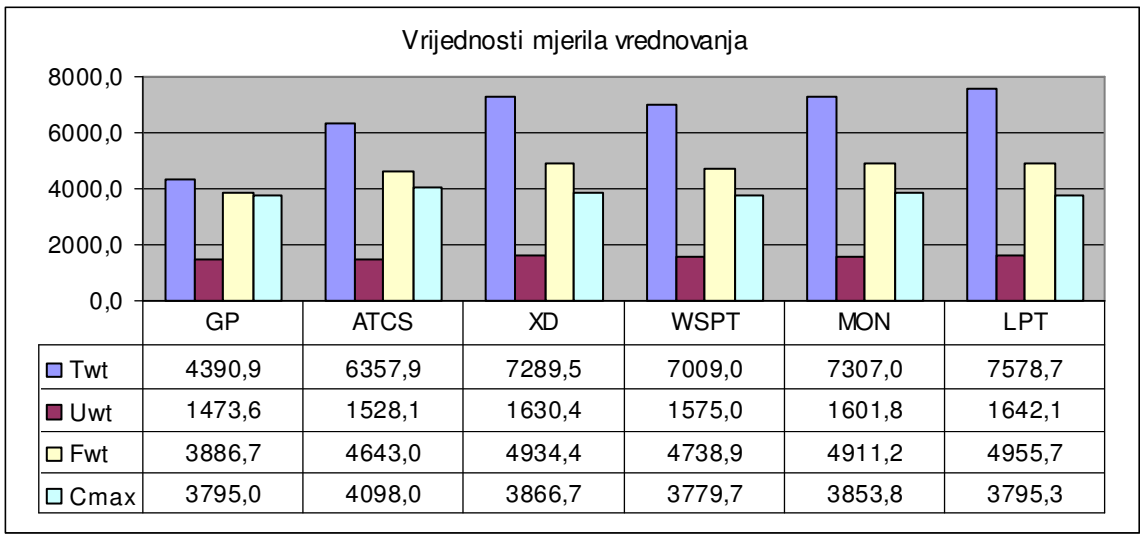


	GP	XD	MON	WSPT	EDD	LPT
Twt	5%	39%	33%	16%	9%	2%
Uwt	2%	24%	35%	45%	9%	3%
Fwt	7%	6%	30%	55%	3%	1%
Cmax	68%	6%	8%	6%	8%	40%

Slika 31 Postoci dominacije za ukupnu duljinu rasporeda – skup primjera za ocjenu

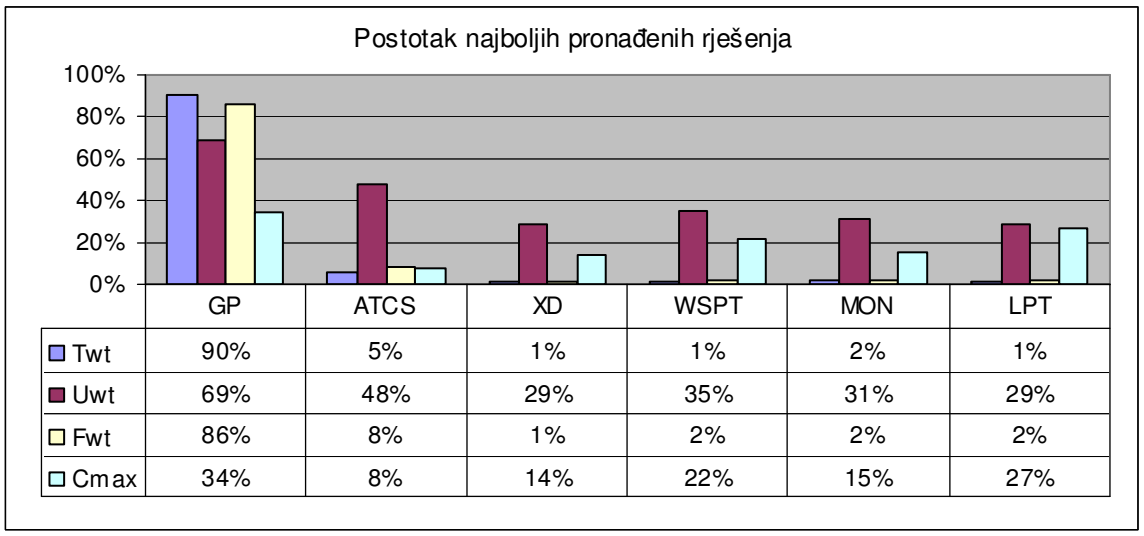
Rezultati problema raspoređivanja uz trajanja postavljanja i optimiranje težinskog zaostajanja prikazani su na slikama 32 i 33.

Kao što je vidljivo iz prikaza, u okoj okolini optimiranje po jednom kriteriju daje bolju uspješnost na više kriterija, djelomično zbog nedostatka postojećih algoritama u prilagodbi ovim uvjetima raspoređivanja. Za jednake uvjete provedeni su i pokusi uz optimiranje ukupne duljine rasporeda, a rezultati najboljeg pronađenog rješenja prikazani su na slikama 34 i 35.



	GP	ATCS	XD	WSPT	MON	LPT
Twt	4390,9	6357,9	7289,5	7009,0	7307,0	7578,7
Uwt	1473,6	1528,1	1630,4	1575,0	1601,8	1642,1
Fwt	3886,7	4643,0	4934,4	4738,9	4911,2	4955,7
Cmax	3795,0	4098,0	3866,7	3779,7	3853,8	3795,3

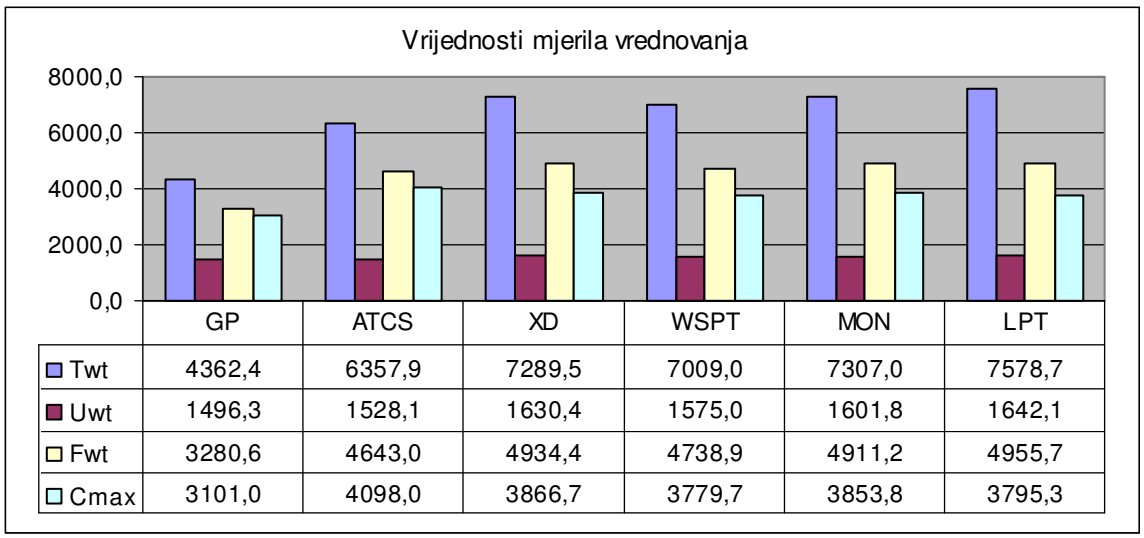
Slika 32 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	ATCS	XD	WSPT	MON	LPT
Twt	90%	5%	1%	1%	2%	1%
Uwt	69%	48%	29%	35%	31%	29%
Fwt	86%	8%	1%	2%	2%	2%
Cmax	34%	8%	14%	22%	15%	27%

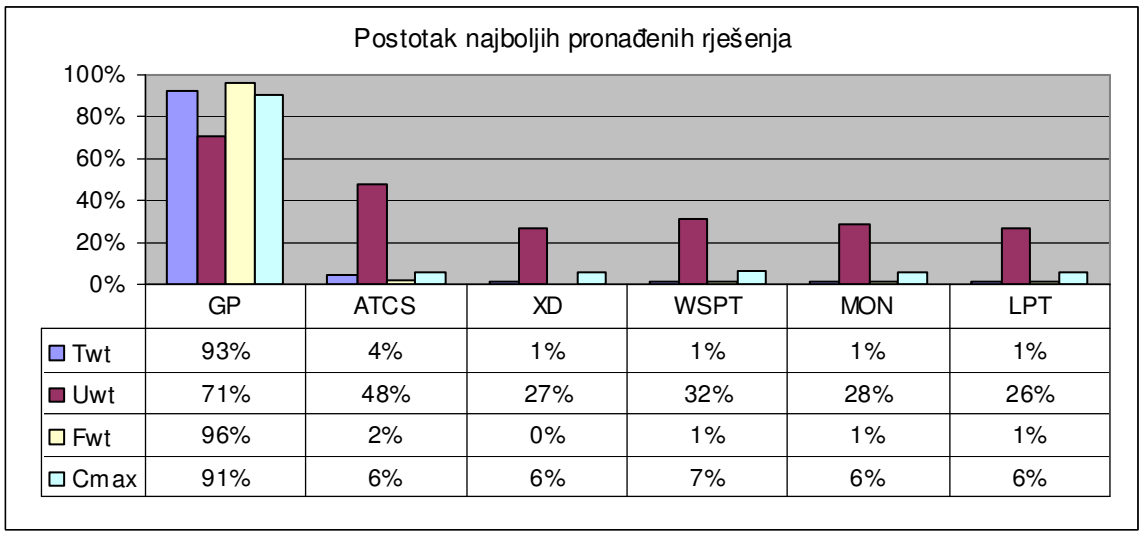
Slika 33 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Na ovom se primjeru može vidjeti da je kriterij ukupne duljine rasporeda isplativiji nego u primjeru bez trajanja postavljanja, budući je pravilo izvedeno genetskim algoritmom postiglo dobre rezultate po većini mjerila. Uzrok ovakvom učinku može se objasniti utjecajem prikladnog izbora redoslijeda poslova i odabirom manjeg ukupnog trajanja postavljanja, što znatno utječe na konačni rezultat. Ipak, ovakvu je dobru učinkovitost po više kriterija teško predvidjeti – na primjer, neko drugo rješenje genetskog programiranja moglo bi imati slične ili čak nešto bolje rezultate po pitanju težinskog zaostajanja, no lošije za ostale kriterije.



	GP	ATCS	XD	WSPT	MON	LPT
Twt	4362,4	6357,9	7289,5	7009,0	7307,0	7578,7
Uwt	1496,3	1528,1	1630,4	1575,0	1601,8	1642,1
Fwt	3280,6	4643,0	4934,4	4738,9	4911,2	4955,7
Cmax	3101,0	4098,0	3866,7	3779,7	3853,8	3795,3

Slika 34 Optimiranje ukupne duljine rasporeda – skup primjera za ocjenu



	GP	ATCS	XD	WSPT	MON	LPT
Twt	93%	4%	1%	1%	1%	1%
Uwt	71%	48%	27%	32%	28%	26%
Fwt	96%	2%	0%	1%	1%	1%
Cmax	91%	6%	6%	7%	6%	6%

Slika 35 Postoci dominacije za ukupnu duljinu rasporeda – skup primjera za ocjenu

5 Raspoređivanje na paralelnim nesrodnim strojevima

U postupku primjene pravila raspoređivanja na nesrodnim procesorima potrebno je u odluku uključiti i prikladnost stroja na kojemu se posao raspoređuje. U prethodnom je okruženju jedan od poslova bio raspoređen na prvi raspoloživi stroj, bez uvida u trajanja izvođenja na drugim strojevima. U ovom bi okruženju takav postupak dao lošije rezultate jer ne uzima u obzir sve postojeće strojeve. U ovom radu je stoga predložen algoritam koji u kombinaciji sa funkcijom prioriteta, izvedenom s pomoću genetskog programiranja, raspoređuje poslove na nesrodne strojeve. Ovaj postupak bi se mogao nazvati *meta-algoritmom* budući se kao dio algoritma javlja funkcija odabira koja tek treba biti definirana, a postupak opisuje način korištenja odgovarajuće funkcije. Opisani postupak prikazan je kao algoritam 1.

Algoritam se može opisati na sljedeći način: u prvom trenutku kada su barem jedan stroj i jedan posao raspoloživi (tj. barem neki posao se može pokrenuti), računaju se prioriteti svih raspoloživih poslova na svim strojevima. Potom se za svaki posao zabilježi stroj na kojemu posao postiže najbolju (po pretpostavci najmanju) vrijednost prioriteta i sama ta vrijednost. Postoji li barem jedan posao za kojega je najbolji odabrani stroj raspoloživ (odnosno barem jedan posao koji se, po najboljem prioritetu, može odmah pokrenuti), tada se odabire najbolja kombinacija posla i stroja i odgovarajući posao se pokreće. Ukoliko su za sve poslove najbolji odabrani strojevi raspoloživi tek u budućnosti, raspoređivanje se ne obavlja, već se čeka do prvog sljedećeg trenutka raspoloživosti nekog drugog *stroja* (budući da nijedan od raspoloživih strojeva nije 'dovoljno dobar' za bilo koji raspoloživi posao). Na taj se način prilikom sljedećeg pokušaja raspoređivanja u odabir uključuju i poslovi koji su u međuvremenu došli u sustav.

```
dok (postoje neraspoređeni poslovi)
{     čekaj dok barem jedan stroj i jedan posao nisu raspoloživi;
      za (sve raspoložive poslove)
          za (sve strojeve)
              π_ij = prioritet posla j za stroj i;
      za (sve raspoložive poslove)
          odredi najbolji stroj za svaki posao (onaj stroj koji za neki
                posao daje najmanju vrijednost π_ij);
      ako (postoji barem jedan posao čiji najbolji stroj je raspoloživ)
      {   odredi najbolji posao čiji najbolji stroj je raspoloživ;
          rasporedi odabrani posao na odabrani stroj;
      }
      inače
          čekaj do pojave sljedećeg raspoloživog stroja;
}
```

Algoritam 1. Meta-algoritam raspoređivanja na nesrodnim strojevima

Zadatak je genetskog programiranja pronaći takvu funkciju prioriteta koja će najbolje iskoristiti strukturu prikazanog meta-algoritma. Kako bi se u funkciji mogla maksimalno iskoristiti svojstva poslova i strojeva, potrebno je definirati odgovarajuće podatkovne strukture koje će u gradnji funkcije biti na raspolaganju genetskom programu. Skup mogućih funkcijskih i podatkovnih čvorova za ovo okruženje prikazan je u tablici 9. Među čvorovima se ne nalaze informacije o ukupnom broju poslova, njihovim dolascima itd., što odgovara raspoređivanju na zahtjev (nije dostupna nikakva informacija o budućnosti sustava).

Tablica 9. Popis čvorova za raspoređivanje na nesrodnim strojevima

Oznaka funkcijskog čvora	Definicija
ADD, SUB, MUL, DIV, POS	kao u tablici 1
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje izvođenja posla na promatranom stroju (p_{ij})
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)

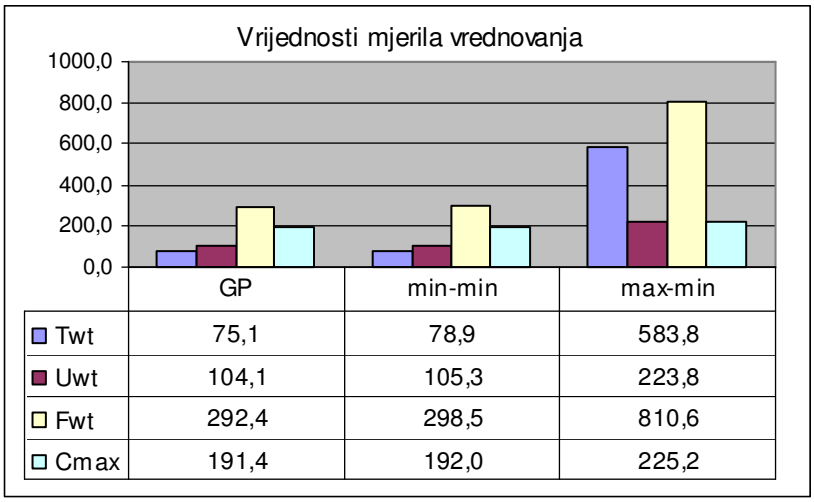
SL	dopuštena odgoda uz brzinu dotičnog stroja, $\max\{d_j - p_{ij} - time, 0\}$
pmin	najkraće trajanje izvođenja posla (za sve postojeće strojeve)
pavg	srednje trajanje izvođenja posla
PAT	strpljenje (od engl. *patience*) – količina vremena za koju će stroj koji za promatrani posao daje najkraće trajanje izvođenja biti raspoloživ; 0 ako dotični stroj jest raspoloživ
MR	količina vremena do raspoloživosti promatranog stroja (od engl. *machine ready*); 0 ako stroj jest raspoloživ
age	vrijeme koje je posao proveo u sustavu, $time - r_j$

Čvor 'pmin' označava najkraće moguće trajanje izvođenja koje posao može postići, što obično vrijedi samo za jedan od strojeva, dok čvor 'pavg' označava srednju vrijednost svih trajanja obrade nekoga posla. Čvor 'age' označava koliko vremena je proteklo od dolaska posla u sustav, 'MR' je količina vremena koja treba proći prije nego promatrani stroj postane raspoloživ, a 'PAT' je količina vremena koju bi posao trebao čekati da bude izveden na stroju koji mu daje najkraće vrijeme izvođenja. Potrebno je napomenuti da se, budući se ocjenjivanje obavlja za sve poslove i sve strojeve, vrijednosti ovih varijabli mogu definirati ovisno o promatranom poslu i stroju. To vrijedi za čvorove 'pt' i 'SL', dok čvorovi 'dd', 'w', 'pmin', 'pavg', 'PAT' i 'age' ovise samo o trenutnom poslu, a čvor 'MR' ovisi samo o promatranom stroju. Budući da algoritam računa sve kombinacije, uz oznake čvorova ne navode se indeksi koji bi označavali njihovu ovisnost.

Odabir dostupnih varijabli ovisit će i o funkciji cilja, odnosno kriteriju ocjenjivanja rasporeda, pa npr. u optimiranju ukupne duljine rasporeda neće biti potrebno uzimati u obzir težinu, željeno vrijeme završetka posla i dopuštenu odgodu.

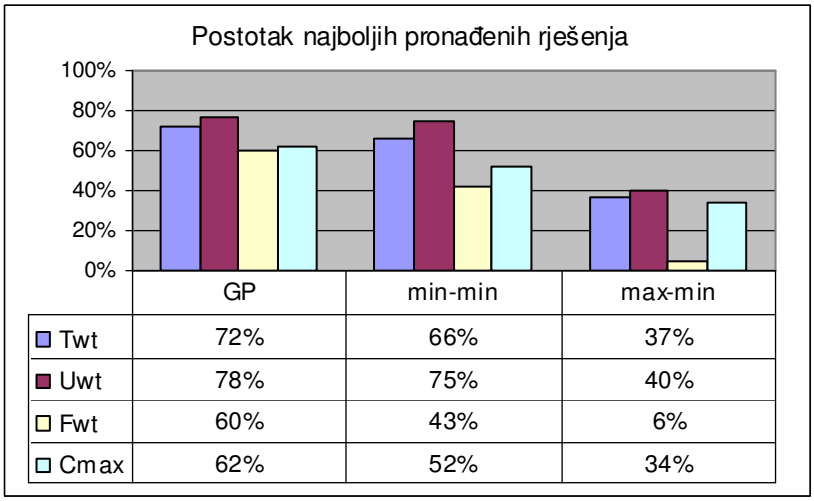
Ispitivanje učinkovitosti raspoređivanja na paralelnim nesrodnim strojevima provedeno je uz pretpostavke navedene za ovo okruženje: raspoređivanje se obavlja na zahtjev (podaci o budućnosti sustava nisu poznati), a poslovi i njihovi podaci postaju dostupni u trenutku njihovog vremena pripravnosti.

Jedan dio pokusa proveden je uz optimiranje netežinskog trajanja protjecanja (budući ostali promatrani algoritmi ne uzimaju težine u obzir); u tome su slučaju iz skupa čvorova izostavljeni čvorovi koji predstavljaju težine poslova, željeno vrijeme završetka i dopuštenu odgodu. Za potrebe optimiranja ovoga kriterija provedeno je 20 pokusa genetskog programiranja uz pretpostavljene vrijednosti parametara po algoritmu 1. Najbolja rješenja iz primjera za učenje primijenjena su na primjere za ocjenu kako bi se izabralo konačno rješenje u obliku funkcije prioriteta. Usporedni rezultati na skupu primjera za ocjenu prikazani su na slikama 36 i 37.



	GP	min-min	max-min
Twt	75,1	78,9	583,8
Uwt	104,1	105,3	223,8
Fwt	292,4	298,5	810,6
Cmax	191,4	192,0	225,2

Slika 36 Optimiranje trajanja protjecanja – skup primjera za ocjenu

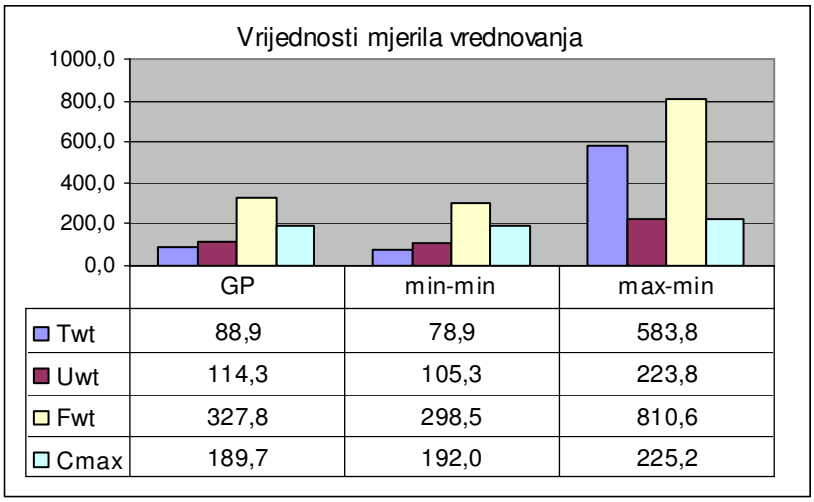


	GP	min-min	max-min
Twt	72%	66%	37%
Uwt	78%	75%	40%
Fwt	60%	43%	6%
Cmax	62%	52%	34%

Slika 37 Postoci dominacije za trajanje protjecanja – skup primjera za ocjenu

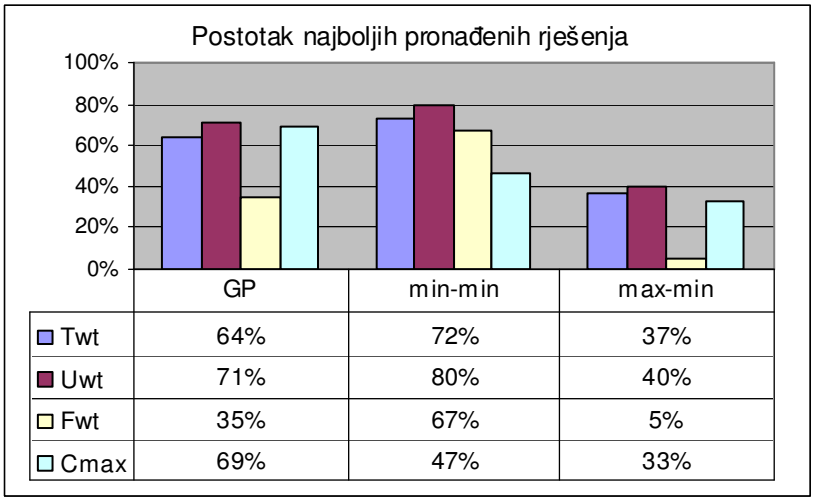
Iz rezultata se može uočiti da je učinkovitost min-min algoritma i algoritma izvedenog genetskim programiranjem uglavnom podjednaka, uz tek nešto izraženiju razliku kod postotaka dominacije za pojedine kriterije. Max-min algoritam je uvjerljivo najlošiji, no takav rezultat je vjerojatno djelomično uzrokovan i svojstvima ispitnih primjera.

Osim trajanja protjecanja, pokusi su provedeni i za kriterij ukupne duljine rasporeda kao dobrote rješenja evolucijskog procesa. Provedeno je 20 pokusa uz jednake podatkovne čvorove za jedinke genetskog programa. Rezultati za ovaj kriterij prikazani su na slikama 38 i 39.



	GP	min-min	max-min
Twt	88,9	78,9	583,8
Uwt	114,3	105,3	223,8
Fwt	327,8	298,5	810,6
Cmax	189,7	192,0	225,2

Slika 38 Optimiranje ukupne duljine rasporeda – skup primjera za ocjenu

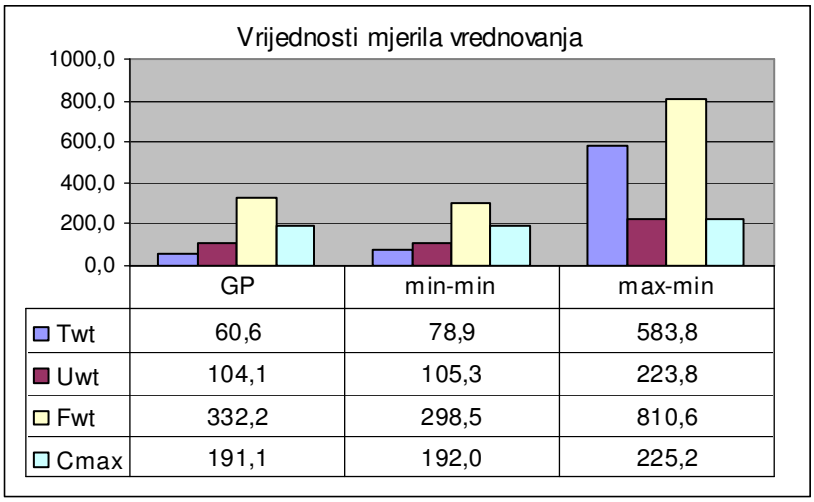


	GP	min-min	max-min
Twt	64%	72%	37%
Uwt	71%	80%	40%
Fwt	35%	67%	5%
Cmax	69%	47%	33%

Slika 39 Postoci dominacije za ukupnu duljine rasporeda – skup primjera za ocjenu

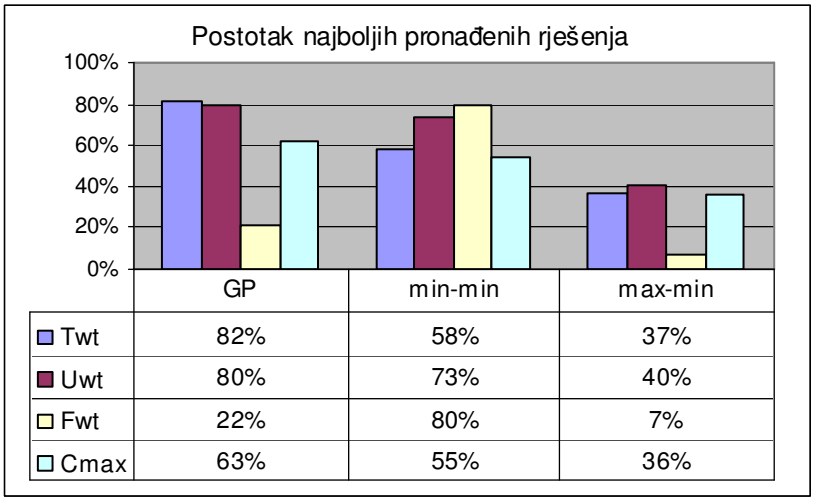
Lako je uočljivo da je rješenje dobiveno genetskim programiranjem uz kriterij ukupne duljine rasporeda kao dobrote jedinke dalo relativno lošije rješenje u usporedbi s ostalim algoritmima nego u prethodnom primjeru. Iako se prednost za promatrani kriterij (neznatno) povećala, učinkovitost algoritma je opala u ostalim prikazanim mjerilima. Iz toga se može zaključiti da je kriterij protjecanja 'isplativiji' u postupku učenja, budući daje rješenja čija je učinkovitost u usporedbi s ostalim algoritmima podjednaka po više mjerila.

Uvrštenje kriterija težinskog zaostajanja i težinske zakašnjelosti poslova u rezultatima u biti nije sasvim opravdano jer promatrani algoritmi, uključujući i onaj izveden genetskim programiranjem, ne koriste informaciju o težinama i željenim vremenima završetka poslova. Općenito je teže pronaći algoritam koji bi za ovo okruženje sastavio raspored po navedenim kriterijima. Stoga je, ilustracije radi, provedeno i izvođenje algoritma sa težinskim zaostajanjem kao funkcijom dobrote rješenja. U ovom su slučaju genetskom programu na raspolaganju i odgovarajući podatkovni čvorovi (težina posla, željeno vrijeme završetka, dopuštena odgoda za pojedini stroj). Za ovaj je kriterij provedeno 20 pokusa, a rezultati su na slikama 40 i 41.



	GP	min-min	max-min
Twt	60,6	78,9	583,8
Uwt	104,1	105,3	223,8
Fwt	332,2	298,5	810,6
Cmax	191,1	192,0	225,2

Slika 40 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	min-min	max-min
Twt	82%	58%	37%
Uwt	80%	73%	40%
Fwt	22%	80%	7%
Cmax	63%	55%	36%

Slika 41 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

6 Raspoređivanje za proizvoljnu obradu

U ovom se okruženju pravilo raspoređivanja može izvesti na jednak način kao i u prethodnim okolinama; drugim riječima, rješenje genetskog programiranja je funkcija prioriteta predstavljena stablom. Način primjene izvedenog pravila definiran je s time da je moguće da odabrana operacija ima vrijeme početka u budućnosti, odnosno dozvoljeno je umetnuto čekanje. Zadatak je na samom pravilu da na pravilan način iskoristi informaciju o vremenu dolaska i zaključi isplati li se čekati na određenu aktivnost.

Za potrebe izvođenja pravila, genetskom su programu na raspolaganju podaci o poslu i trenutnoj operaciji, kao i neke veličine koje opisuju trenutno stanje cijeloga sustava. Popis svih funkcijskih i podatkovnih čvorova koji se mogu javiti unutar jedinke dan je u tablici 10.

Tablica 10. Popis čvorova za raspoređivanje u proizvoljnoj obradi

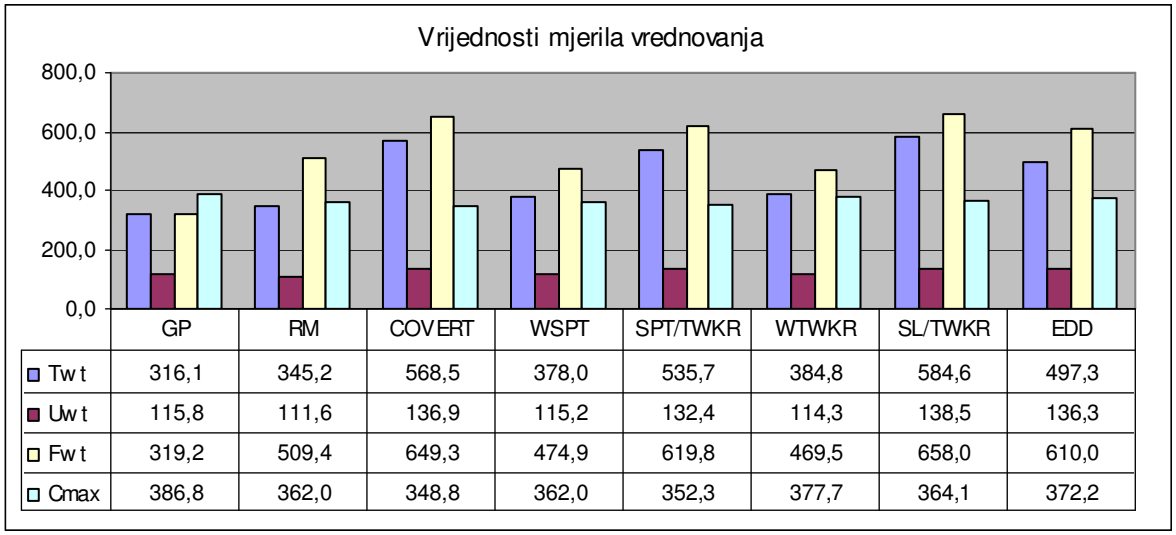
Oznaka funkcijskog čvora	Definicija

ADD, SUB, MUL, DIV, POS	kao u tablici 1
SQR	zaštićeni unarni operator kvadratnog korijena: $SQR(a) = \begin{cases} 1, \text{ako } a < 0 \\ \sqrt{a}, \text{inače} \end{cases}$
IFGT	logički operator odabira: $IFGT(a,b,c,d) = \begin{cases} c, \text{ ako } a > b \\ d, \text{ inače} \end{cases}$
Oznaka podatkovnog čvora	Definicija vrijednosti podatkovnog čvora
pt	trajanje izvođenja operacije određenog posla na promatranom stroju (p_{ij})
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
CLK	trenutno vrijeme
AR	količina vremena do dolaska operacije; $\max\{r_{ij} - time, 0\}$, gdje se r_{ij} definira kao završetak prethodne operacije promatranog posla (prije stroja i)
NOPr	broj neobavljenih operacija posla
TWK	ukupno trajanje svih operacija posla (twk_j)
TWKr	trajanje svih neobavljenih operacija posla ($twkr_j$)
PTav	prosječno trajanje svih operacija na promatranom stroju
HTR	omjer vremena koje je posao proveo u sustavu i ukupnog trajanja do sada obavljenih operacija (po engl. *head time ratio*)

Operator 'IFGT' promatra vrijednosti prva dva argumenta (prva dva podstabla) i na temelju njihovog odnosa kao rezultat vraća vrijednost trećeg odnosno četvrtog argumenta. Budući se može očekivati da je potreba za funkcijskim čvorovima 'IFGT' i 'SQR' manja od potrebe za ostalim čvorovima, oni su uvršteni u skup uz manju vjerojatnost pojavljivanja u stablu (prilikom mutacije). Podatkovni čvor 'HTR' trebao bi pravilu raspoređivanja omogućiti procjenu o količini vremena koju će posao još provesti u sustavu, pod pretpostavkom da je opterećenje sustava podjednako prije i poslije obavljanja trenutne operacije.

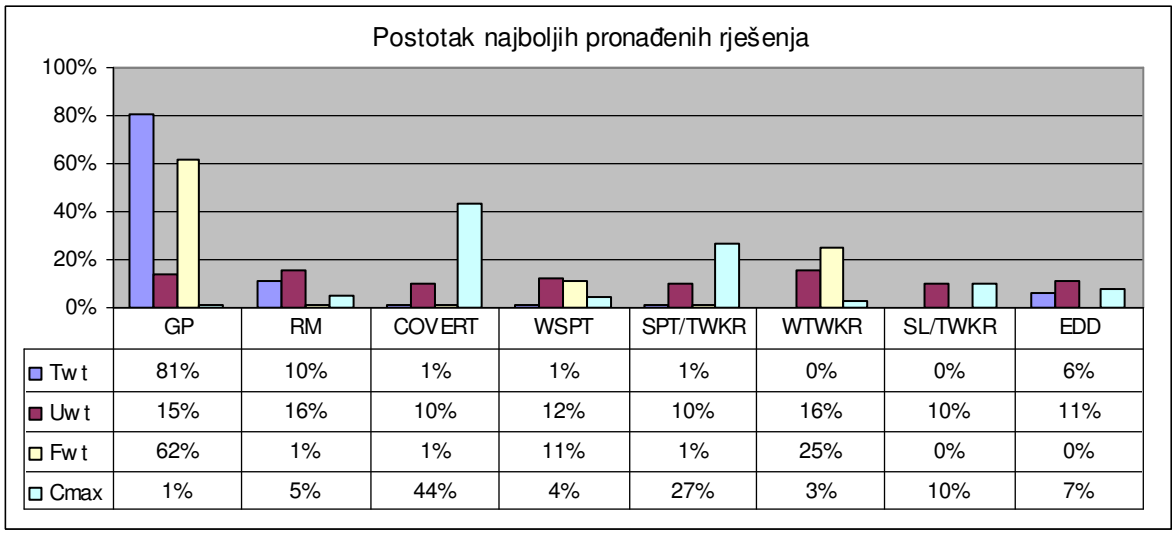
Prvi dio ispitivanja proveden je za statičku okolinu, u kojoj su svi poslovi raspoloživi od početka rada sustava, uz optimiranje kriterija težinskog zaostajanja. Provedeno je 20

pokusa iz kojih je odabrano najbolje rješenje prema skupu ispitnih primjera za ocjenu. Usporedni rezultati prikazani su na slikama 42 i 43.



	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
Twt	316,1	345,2	568,5	378,0	535,7	384,8	584,6	497,3
Uwt	115,8	111,6	136,9	115,2	132,4	114,3	138,5	136,3
Fwt	319,2	509,4	649,3	474,9	619,8	469,5	658,0	610,0
Cmax	386,8	362,0	348,8	362,0	352,3	377,7	364,1	372,2

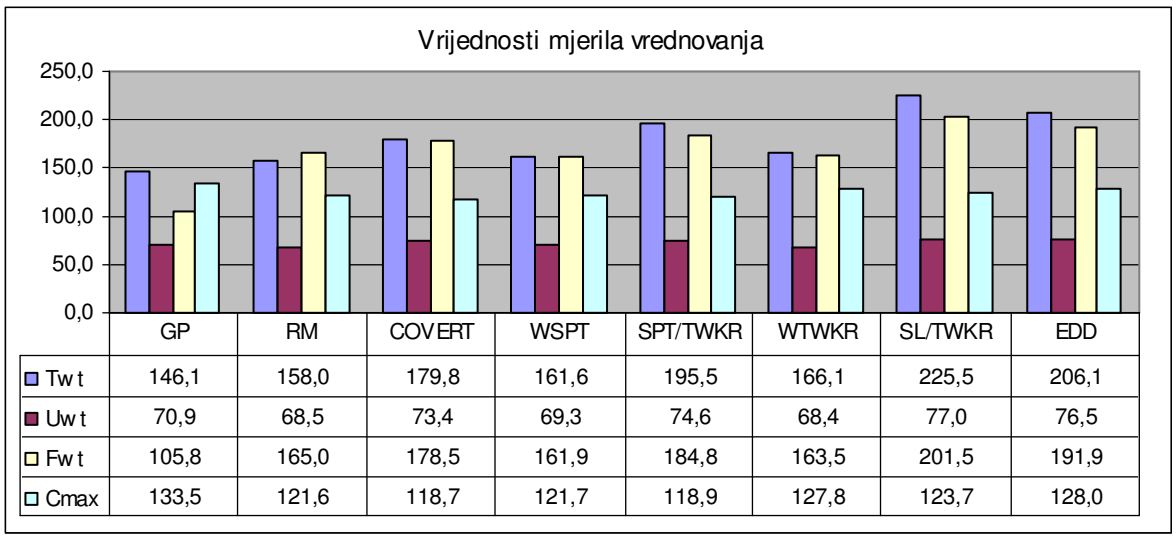
Slika 42 Optimiranje težinskog zaostajanja – skup primjera za ocjenu



	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
Twt	81%	10%	1%	1%	1%	0%	0%	6%
Uwt	15%	16%	10%	12%	10%	16%	10%	11%
Fwt	62%	1%	1%	11%	1%	25%	0%	0%
Cmax	1%	5%	44%	4%	27%	3%	10%	7%

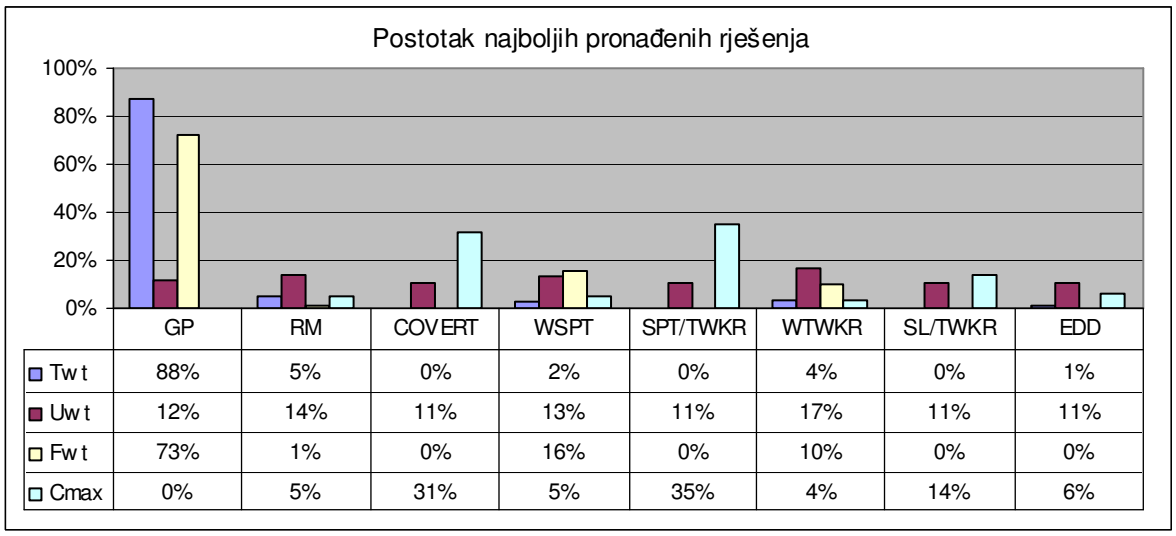
Slika 43 Postoci dominacije za težinsko zaostajanje – skup primjera za ocjenu

Iz rezultata je vidljivo da je pravilo izvedeno genetskim programiranjem najveću relativnu prednost postiglo za kriterije težinskog zaostajanja i težinskog protjecanja, dok istovremeno daje najlošije rezultate za ukupnu duljinu rasporeda. Po pitanju ukupne duljine rasporeda najveću učinkovitost pokazuje COVERT pravilo, što pomalo iznenađuje budući je to pravilo, po autorima, namijenjeno optimiranju težinskog zaostajanja. Isto rješenje dobiveno genetskim programiranjem uspoređeno je sa ostalim pravilima na skupu od 80 primjera iz literature, za koji su rezultati prikazani na slikama 44 i 45.



	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
Twt	146,1	158,0	179,8	161,6	195,5	166,1	225,5	206,1
Uwt	70,9	68,5	73,4	69,3	74,6	68,4	77,0	76,5
Fwt	105,8	165,0	178,5	161,9	184,8	163,5	201,5	191,9
Cmax	133,5	121,6	118,7	121,7	118,9	127,8	123,7	128,0

Slika 44 Optimiranje težinskog zaostajanja – skup primjera iz literature



	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
Twt	88%	5%	0%	2%	0%	4%	0%	1%
Uwt	12%	14%	11%	13%	11%	17%	11%	11%
Fwt	73%	1%	0%	16%	0%	10%	0%	0%
Cmax	0%	5%	31%	5%	35%	4%	14%	6%

Slika 45 Postoci dominacije za težinsko zaostajanje – skup primjera iz literature

Iz ovih se rezultata može uočiti da je relativni odnos pravila po pitanju učinkovitosti ostao gotovo nepromijenjen u odnosu na prethodni skup primjera. Na temelju toga mogli bi se izvesti zaključci o dobroj prilagodbi izvedenog pravila; no s obzirom na relativno mali broj ispitnih primjera u ovome skupu, za takve tvrdnje potrebna su dodatna ispitivanja uz promijenjene uvjete raspoređivanja.

Literatura

[Ada 02] T. P. Adams, *Creation of Simple, Deadline, and Priority Scheduling Algorithms using Genetic Programming,* Genetic Algorithms and Genetic Programming at Stanford 2002, http://www.genetic-programming.org/sp2002/Adams.pdf

[And 94] D. Andre, *Automatically Defined Features: The Simultaneous Evolution of 2-dimensional Featrue Detectors and an Algorithm for Using Them*, Advances in Genetic Programming, pp. 477-494, MIT Press, 1994.

[Atl 94] Bonnet L. Atlan, J.B. Polack, *Learning distributed reactive strategies by genetic programming for the general job shop problem,* Proceedings 7th annual Florida Artificial Intelligence Research Symposium, Pensacola, Florida, IEEE Press, 1994.

[Auy 03] Andy Auyeung, Iker Gondra, H. K. Dai, *Multi-heuristic list scheduling genetic algorithm for task scheduling,* Symposium on Applied Computing, Proceedings of the 2003 ACM symposium on Applied computing,Melbourne, Florida,Pages: 721 - 724, http://portal.acm.org/citation.cfm?doid=952532.952673

[Ban 02] W. Banzhaf, W. B. Langdon, *Some considerations on the reason for bloat,* Genetic Programming and Evolvable Machines, 3(1), pp. 81-91, March 2002.

[Ban 98] W. Banzhaf, P. Nordin, R. E. Kellet, F. D. Francone, *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann, 1998.

[Bea 05] Open BEAGLE: a versatile EC framework, http://beagle.gel.ulaval.ca/

[Bea 90] J. E. Beasley, *OR-Library: Weighted tardiness*, 1990, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html

[Bli 94] T. Blickle, L. Thiele, *Genetic Programming and Redundancy,* Proc. Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94), Saarbrücken, Germany, 1994. , http://www.handshake.de/user/blickle/publications/GPandRedundancy.ps

[Bli 96] Tobias Blickle, *Evolving Compact Solutions in Genetic Programming: A Case Study,* Parallel Problem Solving From Nature IV. Proceedings of the International Conference on Evolutionary Computation, LNCS, Vol. 1141, pp. 564-573, Springer-Verlag, 22-26 September 1996., http://www.blickle.handshake.de/publications/ppsn1.ps

[Bud 00] L. Budin, D. Jakobović, M. Golub, *Genetic Algorithms in Real-Time Imprecise Computing*, Journal of Computing and Information Technology CIT, Vol. 8, No. 3, September 2000, pp. 249-257.

[Bud 98] L. Budin, D. Jakobović, M. Golub, *Parallel Adaptive Genetic Algorithm*, Proc. Int. ICSC/IFAC Symposium on Neural Computation, NC'98, Vienna, September 23-25, 1998., pp. 157-163

[Bud 99] L. Budin, D. Jakobović, M. Golub, *Genetic Algorithms in Real-Time Imprecise Computing*, IEEE International Symposium on Industrial Electronics ISIE'99, Bled, 1999, Vol. 1, pp. 84-89.

[Bur 03] Edmund Burke, Steven Gustafson, Graham Kendall, *Ramped Half-n-Half Initialisation Bias in GP*, enetic and Evolutionary Computation -- GECCO-2003, LNCS, Vol. 2724, pp. 1800-1801, Springer-Verlag, 12-16 July 2003.

[Cha 04] Samarn Chantaravarapan, Jatinder N.D. Gupta, *Single Machine Group Scheduling with Setups to Minimize Total Tardiness,* 2004, http://www.pmcorp.com/PublishedPapers/Scheduling%20Publications/SingleMachineGroupSchedulingwithSetups.pdf

[Cha 96] Yih-Long Chang, Toshiyuki Sueyoshi, Robert Sullivan, *Ranking dispatching rules by data envelopment analysis in a job shop environment,* IIE Transactions, 28(8):631-642, 1996

[Che 99] V.H.L. Cheng, L.S. Crawford, P.K. Menon, *Air Traffic Control Using Genetic Search Techniques,* 1999 IEEE International Conference on Control Applications, August 22-27, Hawai'i, HA, http://www.optisyn.com/papers/1999/traffic_99.pdf

[Cic 01] Vincent A. Cicirello, Stephen F. Smith, *Ant Colony Control for Autonomous Decentralized Shop Floor Routing,* Fifth International Symposium on Autonomous Decentralized Systems March 26 - 28, 2001 Dallas, Texas p. 383, http://csdl.computer.org/comp/proceedings/isads/2001/1065/00/10650383abs.htm

[Cic 01a] Vincent Cicirello, Stephen Smith, *Randomizing Dispatch Scheduling Policies,* The 2001 AAAI Fall Symposium: Using Uncertainty Within Computation, November, 2001., http://www.ri.cmu.edu/pubs/pub_3789.html

[Cic 03] Vincent Cicirello, *Weighted Tardiness Scheduling with Sequence-Dependent Setups,* Technical report, The Robotics Institute, Carnegie Mellon Univeristy, 2003, http://www.cs.drexel.edu/~cicirello/benchmarks.html

[Cor 01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd ed., The MIT Press - McGraw-Hill, 2001.

[Cra 85] Nichael Lynn Cramer, *A Representation for the Adapative Generation of Simple Sequential Programs,* International Conference on Genetic Algorithms and their Applications [ICGA85], CMU, Pittsburgh, http://www.sover.net/~nichael/nlc-publications/icga85/index.html

[Dav 81] Ernest Davis, Jeffrey M. Jaffe, Algorithms for Scheduling Tasks on Unrelated Processors, Journal of the ACM, Volume 28 , Issue 4 (October 1981), pp. 721 – 736 http://portal.acm.org/citation.cfm?doid=322276.322284

[Dha 78] B. G. Dharan, T.E. Morton, Algoristics for Single Machine Sequencing with Precedence Constraints, Management Science 24, p. 1011-1020, 1978. http://www.ruf.rice.edu/~bala/files/dharan-morton-algoristics_for_sequencing-Mgt_Science_1978.pdf

[Dim 01] C. Dimopoulos, A. M. S. Zalzala, *Investigating the use of genetic programming for a classic one-machine scheduling problem,* Advances in Engineering Software, Volume 32, Issue 6 , June 2001, Pages 489-498, http://www.sciencedirect.com/

[Dim 99] Christos Dimopoulos, Ali M. S. Zalzala, *Evolving Scheduling Policies through a Genetic Programming Framework,* Proceedings of the Genetic and Evolutionary Computation Conference, Vol. 2, p. 1231, Morgan Kaufmann, 13-17 July 1999.

[Dim 99a] Dimopoulos C., Zalzala A.M.S., *A genetic programming heuristic for the one-machine total tardiness problem,* Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on , Volume: 3 , 6-9 July 1999

[Eib 00] Ágoston E. Eiben, Robert Hinterding, Zbigniew Michalewicz, *Parameter Control in Evolutionary Algorithms,* IEEE Trans. on Evolutionary Computation, 2000. http://citeseer.ist.psu.edu/eiben00parameter.html

[Gag 02] C. Gagné, M. Parizeau, *Open BEAGLE: A New Versatile C++ Framework for Evolutionary Computation*, Late Breaking papers at the Genetic and Evolutionary Computation Conference (GECCO-2002), New York, USA, 9-13 July 2002

[Gag 03] Christian Gagné, Marc Parizeau, Marc Dubreuil, *Distributed BEAGLE: An Environment for Parallel and Distributed Evolutionary Computations,* Proc. of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS) 2003, http://vision.gel.ulaval.ca/en/publications/Id_439/PublDetails.php

[Gat 97] C. Gathercole, P. Ross, *Tackling the Boolean Even N Parity Problem with Genetic Programming and Limited Error Fitness*, Genetic Programming 1997: Proceedings of the 2nd Annual Conference, pp. 119-127, San Francisco, 1997

[Gib 02] K. A. Gibbs, *Implementation and Evaluation of a Novel Branch Construct for Genetic Programming,* Genetic Algorithms and Genetic Programming at Stanford 2002, http://www.genetic-programming.org/sp2002/Gibbs.pdf

[Gol 00] M. Golub, D. Jakobović, *A New Model of Global Parallel Genetic Algorithm*, Proc. 22th Int. Conference ITI'00, Pula, June 13-16, 2000

[Gol 01] M. Golub, *Poboljšavanje djelotvornosti paralelnih genetskih algoritama*, doktorska disertacija, Fakulet elektrotehnike i računarstva, 2001.

[Gol 01a] M. Golub, D. Jakobović, L. Budin, *Parallelization of Elimination Tournament Selection without Synchronization*, Proc. 5th Int. Conf. on Intelligent Engineering Systems INES 2001, Helsinki, September 16-18., pp. 85-90., 2001.

[Gol 96] M. Golub, *Vrednovanje uporabe genetskih algoritama za aproksimaciju vremenskih nizova*, magistarski rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 1996.

[Gol 98] M. Golub, D. Jakobović, *A Few Implementations Of Parallel Genetic Algorithm*, Proc. 20th Int. Conference ITI'98, Pula, June 14-17 1998, pp.332-337

[Gre 01] William A. Greene, *Dynamic Load-Balancing via a Genetic Algorithm,* 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01) November 07 - 09, 2001 Dallas, Texas, http://csdl.computer.org/comp/proceedings/ictai/2001/1417/00/14170121abs.htm

[Han 04] James V. Hansen, *Genetic search methods in air traffic control,* Computers and Operations Research, v 31, n 3, March, 2004, p 445-459, http://www.sciencedirect.com/

[Hay 96] T. D. Haynes, D. A. Schoenefeld, R. L. Wainwright, *Type Inheritance in Strongly Typed Genetic Programming*, Advances in Genetic Programming II, P. J. Angeline, K. E. Kinnear, (eds), MIT Press, 1996.

[He 03] Xiaoshan He, Xian-He Sun, Gregor Von Laszewski, *A QoS Guided Scheduling Model in Grid Environment*, Journal of Computer Science and Technology, Volume 18 , Issue 4 (July 2003), pp. 442 – 451 http://www.cs.iit.edu/~scs/psfiles/jcst_XHe-5-28.pdf

[Hel 02] Terry M. Helm, Steve W. Painter, Robert Oakes, *A comparison of three optimization methods for scheduling maintenance of high cost, long-lived capital assets,* Winter Simulation Conference Proceedings, v 2, 2002, p 1880-1884

[Hin 97] Robert Hinterding, Zbigniew Michalewicz, Agoston E. Eiben, *Adaptation in Evolutionary Computation: A Survey,* IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 1997. http://citeseer.ist.psu.edu/hinterding97adaptation.html

[Iba 77] Oscar H. Ibarra, Chul E. Kim, *Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors*, Journal of the ACM, Volume 24 , Issue 2 (April 1977), pp. 280 – 289 http://portal.acm.org/citation.cfm?id=322011&jmp=abstract&dl=GUIDE&dl=ACM

[Jak 97] D. Jakobović, *Adaptive Genetic Operators in Elimination Genetic Algorithm*, Proc. 19th Int. Conference ITI'97, Pula, June 17-20 1997, pp.351-356

[Jak 98] D. Jakobović, M. Golub, *Adaptive Genetic Algorithm*, Proc. 20th Int. Conference ITI'98, Pula, June 14-17 1998, pp.351-356

[Jak 99] D. Jakobović, M. Golub, *Adaptive Genetic Algorithm*, Journal of Computing and Information Technology CIT, Vol. 7, No. 3, September 1999., pp. 229-236

[Jon 98] Albert Jones, Luis C. Rabelo, *Survey of Job Shop Scheduling Techniques,* NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, 1998., http://www.nist.gov/msidlibrary/summary/9820.html

[Kas 99] Joachim Käschel, Tobias Teich, Gunnar Köbernik, Bernd Meier, *Algorithms for the Job Shop Scheduling Problem - a comparison of different methods*, European Symposium on Intelligent Techniques ESIT ´99, June 3-4, 1999, Orthodox Academy of Crete, Greece http://www.erudit.de/erudit/events/esit99/12553_P.pdf

[Kin 93] Kenneth E. Kinnear, *Evolving a Sort: Lessons in Genetic Programming,* Proceedings of the 1993 International Conference on Neural Networks, Vol. 2, pp. 881-888, IEEE Press, 28 March -1 April 1993., ftp://cs.ucl.ac.uk/genetic/ftp.io.com/papers/kinnear.icnn93.ps.Z

[Koz 03] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, 2003.

[Koz 90] John R. Koza, *Genetically Breeding Populations of Computer Programs to Solve Problems in Artificial Intelligence ,* Proceedings of the Second International Conference on Tools for AI, Herndon,

Virginia, USA, 1990
http://citeseer.ist.psu.edu/koza90genetically.html

[Koz 90a] John R. Koza, *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems,* Stanford University Computer Science Department technical report STAN-CS-90-1314. June 1990.,
http://www.genetic-programming.com/jkpubs72to93.html

[Koz 92] J. R. Koza, *Genetic Programming – On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[Koz 94] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

[Koz 95] J. Koza, *Genetic Programming and Hill Climbing*, Machine Learning List, Vol. 7, No. 14, 18.9.1995.
http://www.ics.uci.edu/~mlearn/MLlist/v7/14.html

[Lan 00] W. B. Langdon, W. Banzhaf, *Genetic Programming Bloat without Semantics,* Parallel Problem Solving from Nature - PPSN VI 6th International Conference, LNCS, Vol. 1917, pp. 201-210, Springer Verlag, 16-20 September 2000.,
ftp://cs.ucl.ac.uk/genetic/papers/wbl_ppsn2000.ps.gz

[Lan 02] W. B. Langdon, R. Poli, *Foundations of Genetic Programming*, Springer-Verlag, 2002.

[Lan 05] William Langdon, Steven Gustafson, John Koza, *The Genetic Programming Bibliography*, 2005
http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html

[Lan 97] W. B. Langdon, R. Poli, *Genetic Programming Bloat with Dynamic Fitness,* Technical Report, University of Birmingham, School of Computer Science, Number CSRP-97-29, 3 December 1997.,
ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1997/CSRP-97-29.ps.gz

[Lan 98] W. B. Langdon, *Genetic Programming and Data Structures*, Kluwer Academic Publishers, 1998.

[Lee 04] S. M. Lee, A.A. Asllani, *Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming,* Omega, v 32, n 2, April 2004, p 145-53

[Lee 97] Young Hoon Lee, Kumar Bhaskaran, Michael Pinedo, *A heuristic to minimize the total weighted tardiness with sequence-dependent setups,* IIE Transactions, 29, 45-52, 1997.

[Lek 03] Lekin®, Flexible Job Shop Scheduling System
http://www.stern.nyu.edu/om/software/lekin/

[Leu 04] J. Y-T. Leung (ed.), *Handbook of scheduling,* Chapman & Hall/CRC, 2004.

[Leu 95] J. Y-T. Leung, *A survey of scheduling results for imprecise computation tasks*, Imprecise and approximate computation, Kluwer Academic Publishers, pp. 35-42, 1995

[Lop 01] A. Lopez-Ortiz, *Computational Theory FAQ*,
http://db.uwaterloo.ca/~alopez-o/comp-faq/faq.html

[Mar 04] Goran Martinović, *Postupci raspoređivanja u raznorodnim računalnim sustavima*, doktorska disertacija, Fakultet elektrotehnike i računarstva, Zagreb, 2004.

[Meg 05] Nicole Megow, Marc Uetz, Tjark Vredeveld, *Stochastic Online Scheduling on Parallel Machines,* G. Persiano and R. Solis-Oba (eds): Approximation and Online Algorithms, Lecture Notes in Computer Science 3351, pages 167-180, Springer, 2005.,
http://www.math.tu-berlin.de/~nmegow/muv05sos.pdf

[Mic 92] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, Berlin, 1992

[Miy 00] Kazuo Miyashita, *Job-Shop Scheduling with GP,* Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), pp. 505-512, Morgan Kaufmann, 10-12 July 2000.

[Moh 83] Ram Mohan, V. Rachamadugu, Thomas E. Morton, *Myopic Heuristics for the Weighted Tardiness Problem on Identical Parallel Machines,* Working Paper, The Robotics Institute, Carnegie-Mellon University, 1983.

[Mon 01] Patrick Monsieurs, Eddy Flerackers, *Detecting and Removing Inactive Code in Genetic Programs,* 2001,
http://alpha.luc.ac.be/~lucp1089/DetectingAndRemovingInactiveCode.pdf

[Mon 95] D. J. Montana, *Strongly Typed Genetic Programming*, Evolutionary Computation, 3(2):199-230, 1995.

[Mor 93] Thomas E. Morton, David W. Pentico, *Heuristic Scheduling Systems,* John Wiley & Sons, Inc., 1993.

[Nei 99] Michael O'Neill, Conor Ryan, *Automatic Generation of Caching Algorithms,* Evolutionary Algorithms in Engineering and Computer Science, pp. 127-134, John Wiley & Sons, 30 May - 3 June 1999., http://www.mit.jyu.fi/eurogen99/papers/oneill.ps

[Nei 99a] Michael O'Neill, Conor Ryan, *Automatic Generation of Programs with Grammatical Evolution*, 1999, http://citeseer.ist.psu.edu/276159.html

[Nor 94] P. Nordin, *A Compiling Genetic Programming System that Directly Manipulates the Machine Code*, Advances in Genetic Programming, pp. 311-331, MIT Press, 1994

[Nor 95] P. Nordin, W. Banzhaf, *Genetic Programming Controlling a Miniature Robot*, Working Notes for the AAAI Symposium on Genetic Programming, pp. 61-67, MIT, Cambridge, 1995.

[Nov 03] Sonja Novković, Davor Šverko, *A Genetic Algorithm With Self-Generated Random Parameters,* Journal of Computing and Information Technology - CIT, Vol. 11, No. 4, December 2003., pp. 271-284

[Ok 00] S. Ok, K. Miyashita, S. Nishihara, *Improving Performance of GP by Adaptive Terminal Selection,* Proc. of the Pacific Rim International Conference on Artificial Intelligence (PRICAI), pp.435-445, 2000, http://staff.aist.go.jp/k.miyashita/publications/PRICAI2000.ps

[Ok 01] S. Ok, K. Miyashita, K. Hase, *Evolving Bipedal Locomotion with Genetic Programming --- Preliminary Report,* Proc. of the Congress on Evolutionary Computation 2001, pp.1025-1032, 2001, http://staff.aist.go.jp/k.miyashita/publications/cec.ps

[Pat 97] Norman Paterson, Mike Livesey, *Evolving caching algorithms in C by genetic programming,* Genetic Programming 1997: Proceedings of the Second Annual Conference, pp. 262-267, Morgan Kaufmann, 13-16 July 1997., http://www.dcs.st-and.ac.uk/~norman/Pubs/cache.ps.gz

[Pen 00] Carlos Andrés Peña-Reyes, Moshe Sipper, *Evolutionary computation in medicine: an overview,* Artificial Intelligence in Medicine Volume 19, Issue 1, 1 May 2000, Pages 1-23, http://www.sciencedirect.com/

[Pin 04] M. Pinedo, *Offline Deterministic Scheduling, Stochastic Scheduling, and Online Deterministic Scheduling: A Comparative Overview*, Handbook of Scheduling, J. Y-T. Leung (ed.), Chapman & Hall/CRC, 2004.

[Pol 99] Riccardo Poli, *Parallel Distributed Genetic Programming,* New Ideas in Optimization, McGraw-Hill, 1999., http://citeseer.ist.psu.edu/328504.html

[Pru 04] K. Pruhs, J. Sgall, E. Torng, *Online scheduling*, Handbook of Scheduling, J. Y-T. Leung (ed.), Chapman & Hall/CRC, 2004.

[Rus 97] R. M. Russell, J. E. Holsenback, *Evaluation of greedy, myopic and less-greedy heuristics for the single machine, total tardiness problem,* Journal of the Operational Research Society (1997), 48, 640-646

[Sch 94] E. Schöneburg, F. Heinzmann, S. Feddersen, *Genetische Algorithmen und Evolutionsstrategien*, Addison-Wesley, 1994.

[Ser 01] F. Seredynski, J. Koronacki, C. Z. Janikow, *Distributed multiprocessor scheduling with decomposed optimization criterion,* Future Generation Computer Systems, Volume 17, Issue 4 , January 2001, Pages 387-396, http://www.sciencedirect.com/

[Ser 99] F. Seredynski, J. Koronacki, C. Z. Janikow, *Distributed Scheduling with Decomposed Optimization Criterion: Genetic Programming Approach,* Internationa Parallel and Distributed Processing Symposium, workshop: Bio-Inspired Solutions to Parallel Processing Problems, 1999., http://ipdps.eece.unm.edu/1999/biosp3/seredyns.pdf

[Sil 03] Sara Silva, Jonas Almeida, *Dynamic Maximum Tree Depth - A Simple Technique for Avoiding Bloat in Tree-Based GP ,* Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2003), pp. 1776-1787, Genetic and Evolutionary Computation Conference (GECCO-2003), Chicago, Illinois USA,

July-2003,
http://cisuc.dei.uc.pt/ecos/view_pub.php?id_p=109

[Sou 98] T. Soule, *Code Growth in Genetic Programming,* PhD Thesis, University of Idaho, 1998., http://www.cs.uidaho.edu/~tsoule/research/the3.ps

[Sri 94] M. Srinivas, L. M. Patnaik, *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*, IEEE Trans. Systems, Man and Cybernetics, April 1994.

[Sri 94a] M. Srinivas, L. M. Patnaik, *Genetic Algorithms: A Survey*, IEEE Computer, June 1994.

[Tac 94] W. A. Tackett, *Recombination, Selection and the Genetic Construction of Computer Programs*, PhD thesis, University of Southern California, Department of Electrical Engineering Systems, 1994.

[Tai 03] E. Taillard, *Scheduling Instances*, 2003. http://ina.eivd.ch/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html

[Tal 03] W. A. Talbott, *Automatic Creation of Team-Control Plans Using an Assignment Branch in Genetic Programming,* Genetic Algorithms and Genetic Programming at Stanford 2003, http://www.genetic-programming.org/sp2003/Talbott.pdf

[Tel 95] A. Teller, M. Veloso, *PADO: Learning Tree Structured Algorithms for Orchestration into an Object Recognition System*, Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, 1995.

[Vaz 00] Manuel Vazquez, L. Darrell Whitley, *A Comparison of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem,* Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000

[Wal 05] Scott S. Walker, Robert W. Brennan, Douglas H. Norrie, *Holonic Job Shop Scheduling Using a Multiagent System*, IEEE Intelligent Systems, 2/2005, pp. 50-57

[Wal 96] P. Walsh, C. Ryan, *Paragen: A Novel Technique for the Autoparallelisation of Sequential Programs Using Genetic Programming*, Genetic Programming 96: Proceedings of the 1st Annual Conference, pp. 406-409, MIT Press, 1996.

[Wan 03] J.-S. Wang, *Influences of Function Sets in Genetic Programming,* Genetic Algorithms and Genetic Programming at Stanford 2003, http://www.genetic-programming.org/sp2003/Wang.pdf

[Wol 97] D. H. Wolpert, W. G. Macready, *No Free Lunch Theorems for optimization*, IEEE Trans. on Evolutionary Computation, 1(1):67-82, 1997.

[Yin 03] Wen-Jun Yin, Min Liu, Cheng Wu, *Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming,* Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, pp. 1050-1055, IEEE Press, 8-12 December 2003.,

[Zha 96] B.-T. Zhang, H. Mühlenbein, *Adaptive Fitness Functions for Dynamic Growing/Pruning, of Program Trees*, Advances in Genetic Programming 2, pogl. 12, pp.241-256, MIT Press, 1996.