Pregled okruženja i postojećih postupaka raspoređivanja

1 Raspoređivanje na jednom stroju

Iako najjednostavniji po svojoj strukturi, problemi raspoređivanja na jednom stroju i dalje su NP teški pod većinom uvjeta, ovisno o kriteriju ocjenjivanja [Mar 04]. Iako su u praksi manje zastupljeni od drugih okruženja, postupci rješavanja problema na jednom stroju često su osnova složenijih postupaka za raspoređivanje u drugim okolinama. Na primjer, u postupcima promjenjivog uskog grla sustava (engl. *shifting bottleneck*) problem raspoređivanja na više strojeva iterativno se svodi na problem raspoređivanja na jednom stroju, pri čemu je potreban učinkovit i brz postupak rješavanja ugrađenog (engl. *embedded*) problema.

Za sve promatrane inačice raspoređivanja na jednom stroju vrijede sljedeće pretpostavke:

- svi podaci o poslovima unaprijed su poznati (predodređeno raspoređivanje),
- raspoređivanje se obavlja dinamički (tijekom rada sustava),
- poslovi su neprekidivi,
- stroj (sredstvo) je neprekidno raspoloživ.

Navedene pretpostavke su u skladu sa uvjetima koji vrijede u većini prikaza ovog problema u literaturi, pa su zadržane i ovdje poradi lakše usporedbe učinkovitosti. S druge strane, to ne znači da se opisani postupci ne mogu upotrijebiti u sustavima u kojima neke od ovih pretpostavki nisu ispunjene (npr. podaci o poslovima se uvijek mogu mijenjati tijekom rada sustava).

1.1 Statička okolina raspoređivanja

Statička okolina raspoređivanja podrazumijeva raspoloživost svih poslova od početka rada sustava, odnosno vrijedi da su vremena pripravnosti svakog posla jednaka nuli. Pri tome je pretpostavljeno da ne postoje niti ograničenja u redoslijedu poslova niti slijedno ovisna trajanja postavljanja među poslovima.

Pravila raspoređivanja u statičkoj okolini

Ocjena kvalitete rezultata dobivenih genetskim programiranjem provedena je koristeći nekoliko postojećih pravila raspoređivanja. Za svako od pravila navedena je funkcija kojom se određuje prioritet poslova, a najbolji prioritet je onaj sa najvećom vrijednošću. Prioritet pojedinog posla sa rednim brojem j označen je sa π_j.

- WSPT (engl. *weighted shortest processing time*), funkcija prioriteta definirana kao:

$$\pi_j = w_j / p_j \,. \tag{1.1}$$

- EDD (engl. *earliest due date*), definirano sa:

$$\pi_j = 1/d_j \,. \tag{1.2}$$

- Montagne pravilo [Dim 01, Mor 93] (kratica: MON), definirano kao:

$$\pi_j = \left(w_j / p_j\right) \cdot \left(1 - \frac{d_j}{\sum_{i=1}^{n} p_i}\right), \tag{1.3}$$

gdje je n broj poslova koje treba rasporediti.

- Rachamadugu & Morton pravilo (kratica: RM) [Moh 83], definirano sa:

$$\pi_j = \left(w_j / p_j\right) \left[\exp\left(\frac{-\left(d_j - p_j - time\right)^+}{k \cdot p_{AV}}\right)\right], \tag{1.4}$$

gdje je p_{AV} prosječno trajanje svih neraspoređenih poslova (tj. svih onih za koje se računaju prioriteti), *time* je trenutno vrijeme, a operator '+' definiran je kao:

$$x^+ = \max\{x, 0\}. \tag{1.5}$$

Parametar k može ovisiti o dotičnoj inačici problema, a autori predlažu vrijednost 2 za probleme na jednom stroju. Izraz $\left(d_j - p_j - time\right)$ je dopuštena odgoda (engl. *slack*), tj. količina vremena koju još možemo potrošiti prije početka obrade nekog posla a da on ne zakasni.

Navedena pravila raspoređivanja mogu se koristiti u različitim okruženjima, pa je za svako okruženje potrebno definirati točan način primjene pravila. U statičkom okruženju sa jednim strojem uporaba pravila raspoređivanja opisana je algoritmom 1.

```
dok (postoje neraspoređeni poslovi)
{     čekaj dok stroj ne postane raspoloživ;
      odredi prioritete π_j svih neraspoređenih poslova;
      odaberi posao sa najboljim prioritetom;
      rasporedi posao s najboljim prioritetom na stroj;
}
```

Algoritam 1. Postupak raspoređivanja po pravilima za statičke uvjete

U algoritmu 1 'najbolji' prioritet može značiti onaj sa najvećom ili najmanjom vrijednošću, ovisno o tome kako je definirano pojedino pravilo. Pored navedenih pravila, za kriterij težinskog zbroja zakašnjelih poslova implementirana je heuristika iz [Mor 93] zasnovana na Hodgsonovom algoritmu (koji daje optimalno rješenje za netežinsku inačicu problema). Razlika u odnosu na originalni Hodgsonov algoritam je u tome da se na kraj rasporeda premještaju poslovi sa najmanjim omjerom težine i trajanja, dok se u originalnoj netežinskoj inačici promatra samo najdulje trajanje. Postupak je prikazan kao algoritam 2, a u prikazu rezultata označen je sa 'UWT1'. Treba napomenuti da su poslovi koji su u postupku 'izbačeni' iz liste zapravo prebačeni na kraj rasporeda, a njihov međusobni poredak je nebitan (za dani kriterij) jer su svi zakašnjeli.

```
poredaj poslove u listu po EDD algoritmu;
izračunaj vremena završetka svih poslova po trenutnom redoslijedu;
dok (postoje zakašnjeli poslovi u listi)
{     k = prvi zakašnjeli posao;
      pronađi posao r za koji vrijedi w_r / p_r = min{w_i / p_i}, i = 1..k
      obriši posao r iz liste;
```

```
      izračunaj nova vremena završetka svih poslova;
}
rasporedi poslove u listi po dobivenom redoslijedu;
rasporedi obrisane poslove po proizvoljnom redoslijedu;
```

Algoritam 2. Heuristika za težinski zbroj zakašnjelih poslova (UWT1)

1.2 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.

Pravila raspoređivanja u dinamičkoj okolini

U dinamičkoj okolini moguće je primjenjivati i pravila opisana u prethodnom odjeljku, no za drugačiju okolinu postavlja se pitanje interpretacije pravila. Budući svi poslovi nisu raspoloživi u svakom trenutku, prilikom odabira mogućeg sljedećeg posla mora se definirati koji podskup poslova se uzima u obzir. Isto tako, potrebno je definirati hoće li se prioritet poslova koji još nisu pripravni računati na način različit od računanja prioriteta za raspoložive poslove.

Prvo pitanje na koje treba odgovoriti je dopustiti li uopće čekanje na neki posao kojemu je vrijeme pripravnosti u budućnosti, te koliko daleko u budućnost je potrebno promatrati. U određenom sustavu čekanje može biti i nedozvoljeno, no ovdje će se razmatrati samo slučaj dozvoljenog čekanja (uz minimalne izmjene, pravila se mogu uporabiti i u uvjetima nedozvoljenog čekanja). Moguće je dokazati da, uz regularni kriterij raspoređivanja na jednom stroju, eventualno čekanje na neki budući posao ne treba biti dulje od trajanja najkraćeg raspoloživog neraspoređenog posla (jednaki pristup uporabljen je i u nekim drugim alatima raspoređivanja [Lek 03]). Po tom pravilu, u postupak ocjenjivanja ulaze samo oni poslovi čije je vrijeme pripravnosti prije završetka najkraćeg raspoloživog posla, odnosno samo oni poslovi j za koje vrijedi:

$$\left|r_j - time\right| \leq \min_i \{p_i\}, \forall i : r_i \leq time \tag{1.6}$$

Drugi problem javlja se prilikom definicije funkcije prioriteta pojedinog pravila: budući će pravilo izvedeno genetskim programiranjem imati na raspolaganju informaciju o vremenu pripravnosti posla, potrebno je tu informaciju uključiti i u primjenu ostalih pravila koja se koriste za usporedbu. Stoga se u primjeni svih pravila koja sama po sebi podrazumijevaju raspoloživost posla, trajanju obrade posla dodaje količina vremena za koju posao postaje pripravan, a ta se veličina označava sa ar (engl. *arrival*), odnosno:

$$ar = \max\{r_j - time, 0\}. \tag{1.7}$$

Naravno, ukoliko je neko pravilo po svojoj definiciji namijenjeno dinamičkoj okolini (odnosno uključuje informaciju o vremenu pripravnosti posla), tada se vrijednosti u dotičnom pravilu ne mijenjaju. Opisani način primjene pravila raspoređivanja u dinamičkoj okolini može se opisati algoritmom 3.

```
dok (postoje neraspoređeni poslovi)
{     čekaj dok stroj ne postane raspoloživ;
      p_jMIN = trajanje najkraćeg raspoloživog posla;
```

```
      odredi prioritete π_j svih poslova za koje je |r_j − time| < p_jMIN
      odaberi posao sa najboljim prioritetom;
      rasporedi posao s najboljim prioritetom na stroj;
}
```

Algoritam 3. Postupak raspoređivanja po pravilima za dinamičke uvjete

Algoritam 3 može se smatrati ovojnicom oko pojedinog pravila raspoređivanja, odnosno *meta-algoritmom* koji definira primjenu različitih pravila. Uz definirani način primjene, promatrana pravila raspoređivanja u dinamičkoj okolini navedena su u nastavku.

- WSPT pravilo, funkcija prioriteta zadana sa:

$$\pi_j = w_j / \left(p_j + ar_j \right). \tag{1.8}$$

- EDD pravilo, funkcija prioriteta jednaka izrazu (1.2).
- Montagne pravilo (MON), uz funkciju prioriteta:

$$\pi_j = \left(w_j / \left(p_j + ar_j \right) \right) \cdot \left(1 - \frac{d_j}{\sum_{i=1}^{n} p_i} \right). \tag{1.9}$$

- Rachamadugu & Morton (RM) pravilo, definirano kao:

$$\pi_j = \left(w_j / \left(p_j + ar_j \right) \right) \left[\exp\left(\frac{-\left(d_j - \left(p_j + ar_j \right) - time \right)^+}{k \cdot p_{AV}} \right) \right]. \tag{1.10}$$

- Pravilo prilagođeno dinamičkoj okolini, *X-dispatch dottleneck dynamics heuristic* [Mor 93] (kratica: XD); funkcija prioriteta je:

$$\pi_j = \left(w_j / p_j \right) \left[\exp\left(\frac{-\left(d_j - p_j - time \right)^+}{k \cdot p_{AV}} \right) \right] \cdot \left(1 - B \frac{\left(r_j - time \right)^+}{p_{AV}} \right), \tag{1.11}$$

  gdje je p_{AV} prosječno trajanje svih neraspoređenih poslova, a parametar B je opisan izrazom $B = 1.3 + \rho$. Parametar ρ opisuje prosječno opterećenje stroja, a za potrebe ispitnih primjera u ovom radu njegova je vrijednost postavljena na 1.

U prethodno opisanoj statičkoj okolini nema potrebe za dopuštanjem prekidivosti poslova, budući su svi poslovi raspoloživi u isto vrijeme. U dinamičkoj okolini je moguće uključiti i prekidanje, u kojem slučaju prestaje potreba za umetnutim čekanjem pa se odluka o pokretanju sljedećeg posla svodi samo na trenutno pripravne poslove.

1.3 Ograničenja u redoslijedu zadataka

Postoje li ograničenja u redoslijedu poslova, ona su opisana relacijom $\prec$. Izrazom $J_i \prec J_k$ definira se da posao J_i mora završiti prije pokretanja posla J_k. Ograničenja u redoslijedu poslova najčešće se prikazuju usmjerenim grafom u kojemu čvorovi odgovaraju poslovima, a lukovi definiraju relaciju ovisnosti između dva čvora. Dok je u dosadašnjim uvjetima pripravnost posla označavala i njegovu raspoloživost, pojam raspoloživosti posla u

sustavu sa ograničenjima nešto se usložnjava, budući da za pokretanje nekoga posla moraju biti ispunjena dva uvjeta:

- posao J_i mora biti pripravan, odnosno $r_i \leq time$ te
- svi poslovi koji neposredno prethode poslu J_i moraju biti završeni.

U okruženju jednoga stroja, prilikom raspoređivanja dovoljno je provjeriti jesu li svi prethodnici nekoga posla pokrenuti, budući se raspoređivanje uvijek obavlja kada stroj postane slobodan. U okruženjima sa više strojeva potrebno je provjeriti trenutnu obradu na svim strojevima u sustavu. U nekim okruženjima kao što je proizvoljna obrada, za niz operacija nekoga posla podrazumijeva se da su međusobno ovisne i podložne prethodno zadanom redoslijedu, no osim toga može postojati i ovisnost poslova koje te operacije čine.

U $\alpha|\beta|\gamma$ zapisu ova okolina ima oznaku $1|prec|*$. Najčešća mjerila vrednovanja rasporeda koja se uzimaju u obzir u ovoj okolini su također težinsko zaostajanje i težinska zakašnjelost, dok se za probleme najvećeg zaostajanja i težinskog protjecanja mogu razviti optimalni algoritmi.

Pravila raspoređivanja uz ograničenja u redoslijedu

Uz ograničenja u redoslijedu poslova mogu se koristiti i već opisana pravila, uz dodatnu provjeru koja će spriječiti raspoređivanje posla kojemu nisu već raspoređeni svi neposredni prethodnici. Naravno, budući takva pravila ne koriste informaciju o zadanim ograničenjima, takav način raspoređivanja se obično ne koristi. U sustavu sa ograničenjima, međusobne ovisnosti poslova obično su predstavljene usmjerenim grafom. Pravila raspoređivanja u ovoj okolini kao osnovnu informaciju najčešće upotrebljavaju razinu čvora u stablu (engl. *node level*). Razina čvora definirana je uz pomoć duljine puta u stablu na sljedeći način: duljina puta u stablu (engl. *lentgh of a path*) je zbroj trajanja svih čvorova na zadanom putu, a razina čvora je duljina najduljeg puta od zadanog čvora do nekog izlaznog čvora (do čvora bez sljedbenika). Razina izlaznog čvora jednaka je trajanju posla kojega čvor predstavlja.

Uz prethodne definicije, moguće je opisati sljedeća pravila raspoređivanja:

- HL (engl. *highest level*) pravilo prvo raspoređuje raspoloživi posao sa najvećom razinom (funkcija prioriteta pravila jednaka je razini čvora).
- LNS (engl. *largest number of successors*) pravilo prvo raspoređuje posao koji ima najviše neposrednih sljedbenika (funkcija prioriteta jednaka je broju podčvorova zadanog čvora).

Podrazumijeva se da u obzir za odabir dolaze samo raspoloživi poslovi, odnosno oni za koje su zadovoljena ograničenja redoslijeda. Osim navedenih, u uporabi su još neke inačice pravila, no najčešće se sreću upravo opisana pravila [Auy 03].

Za potrebe ispitivanja u ovome radu implementiran je i heuristički algoritam načinjen po uzoru na Sidneyev postupak, a kojega autori [Dha 78] nazivaju 'Sidney algoristikom'. Izraz 'algoristika' je kovanica od riječi algoritam i heuristika, a označava postupak koji pod nekim uvjetima daje optimalno rješenje (egzaktni algoritam), a pod općenitim uvjetima ponaša se kao heuristika. Navedeni postupak će se u daljem tekstu nazivati Sidney heuristika (kratica: SIDNEY). Ovaj postupak koristi pojmove *početnog skupa* (engl. *initial set*), *jednostavnog početnog skupa* (engl. *simple initial set*) i udruženih vrijednosti skupa koji su definirani na sljedeći način:

- početni skup je skup poslova za koje vrijedi da su svi njihovi prethodnici također članovi toga skupa;
- jednostavni početni skup nekoga posla sadrži zadani posao i sve njegove prethodnike.

Za bilo koji skup poslova možemo definirati udružene vrijednosti skupa (engl. *aggregate values*) kao zbroj odgovarajućih vrijednosti svih elemenata toga skupa. Npr. udruženo trajanje skupa poslova p_S jednako je zbroju trajanja svih poslova u skupu:

$$p_S = \sum_{j \in S} p_j \,. \tag{1.12}$$

Na sličan način možemo definirati udruženu težinu skupa itd. Uz prethodne definicije, Sidney heuristika prikazana je kao algoritam 4.

```
S = skup svih neraspoređenih poslova;
dok (S nije prazan)
{     definiraj jednostavne početne skupove za sve poslove iz S;
      A = jedn. poč. skup sa najvećim udruženim w_A/p_A ;
      ako(A ima samo jedan element)
      {   rasporedi posao iz A;
          S = skup svih neraspoređenih poslova;
      }
      inače
          S = A bez posljednjeg elementa;
}
```

Algoritam 4. Sidney heuristika za probleme s ograničenjima

U opisanom algoritmu, 'posljednji element' jednostavnog početnog skupa je uvijek onaj posao od kojega počinjemo graditi skup, tj. onaj koji u tome skupu nema sljedbenika. Može se dokazati [Dha 78] da ovaj algoritam daje optimalno rješenje kriterija težinskog zaostajanja za sustave u kojima su ograničenja predstavljena stablom u kojemu svaki čvor ima najviše jednog neposrednog sljedbenika (engl. *assembly tree*, *intree*). Za probleme sa općenitim oblikom stabla, algoritam na promatranom skupu primjera pronalazi rješenja čija su odstupanja unutar 0.2% od optimuma.

1.4 Slijedno ovisna trajanja postavljanja

Prilikom raspoređivanja niza aktivnosti na nekom sredstvu, podrazumijeva se da je u trajanje aktivnosti uključeno i vrijeme potrebno za eventualnu prilagodbu sredstva ili aktivnosti prije početka obrade. Ukoliko je za svaki posao trajanje prilagodbe neovisno o tome koji je posao prethodno završen na stroju, to se trajanje ne odvaja od ukupnog trajanja obrade nekoga posla. Primjer ovakve situacije može biti zamjena konteksta u memoriji računala prije početka rada slijedećeg procesa, koja je obično neovisna o vrsti prethodnog procesa. Moguće je, međutim, da trajanje prilagodbe sredstva poslu (ili obratno) ovisi o tome koji je posao prethodno obavljen, u kojem se slučaju govori o slijedno ovisnim trajanjima postavljanja. Tada se za svaki mogući uređeni par poslova, od kojih jedan prethodi drugome na nekom stroju, definira količina vremena potrebna za prilagodbu. Trajanje postavljanja obično ovisi u većoj ili manjoj mjeri o trajanjima obrade poslova na koje se odnosi, no može biti i potpuno neovisno. Ova se okolina raspoređivanja prikazuje kao $1|s_{lj}|*$.

Po pitanju kriterija raspoređivanja, osim težinskog zaostajanja i protjecanja kao kriterij se može uporabiti i ukupna duljina rasporeda, budući da izmjenom redoslijeda poslova utječemo na ukupno utrošeno vrijeme (što u problemu bez trajanja postavljanja na jednom stroju nije slučaj).

Pravila raspoređivanja uz trajanja postavljanja

Problemu slijedno ovisnih trajanja postavljanja poklanja se dosta pažnje, pa se za ovu okolinu može naći nekoliko kvalitetnih pravila [Lee 04, Lee 97, Cic 01a, Mor 93]. Mogući

nedostatak većine raspoloživih pravila može predstavljati to što su ona tek prilagođena okolini uz trajanja postavljanja, a ne napravljena specifično za tu namjenu. Pored samih pravila, dobiveno rješenje se u većini slučajeva može poboljšati uporabom metoda pretraživanja [Cic 03, Cha 04], no kvaliteta krajnjeg rješenja u velikoj mjeri ovisi o početnom rješenju dobivenim nekim od pravila raspoređivanja.

Gotovo sva do sada opisana pravila mogu se prilagoditi ovim uvjetima raspoređivanja na način opisan u [Mor 93]; vrijednosti prioriteta posla dobivenom primjenom originalnog pravila oduzima se određena vrijednost koja procjenjuje trošak čekanja zbog dodatnog trajanja postavljanja. Ako je izvorni prioritet posla po nekom pravilu označen sa π_j, tada se prioritet za slučaj trajanja postavljanja dobiva kao

$$\pi_{lj} = \pi_j - \frac{s_{lj}}{p_{AV} \cdot p_j} \tag{1.13}$$

gdje je *l* indeks prethodno obavljenog posla, s_{lj} trajanje postavljanja između posla *l* i posla *j*, a p_{AV} prosječno trajanje svih neraspoređenih poslova. Primjenom opisane prilagodbe, za potrebe ocjene učinkovitosti promatrana su sljedeća pravila:

- WSPT pravilo uz trajanja postavljanja, definirano sa

$$\pi_j = \frac{w_j}{p_j} - \frac{s_{lj}}{p_{AV} \cdot p_j}, \tag{1.14}$$

- MON pravilo uz trajanja postavljanja, definirano kao

$$\pi_j = \left(w_j / p_j\right) \cdot \left(1 - \frac{d_j}{\sum_{i=1}^{n} p_i}\right) - \frac{s_{lj}}{p_{AV} \cdot p_j}, \tag{1.15}$$

- RM pravilo uz trajanja postavljanja, definirano sa

$$\pi_j = \left(w_j / p_j\right) \left[\exp\left(\frac{-\left(d_j - p_j - time\right)^+}{k \cdot p_{AV}}\right)\right] - \frac{s_{lj}}{p_{AV} \cdot p_j}, \tag{1.16}$$

gdje je *l* indeks prethodno obavljenog posla za sva pravila.

Osim navedenih, uporabljeno je i ATCS pravilo (engl. *Apparent Tardiness Cost with Setups*, [Lee 97]) namijenjeno upravo okolini sa trajanjima postavljanja. ATCS pravilo definirano je sljedećim izrazom:

$$\pi_j = \frac{w_j}{p_j} \exp\left(-\frac{\left(d_j - p_j - time\right)^+}{k_1 \bar{p}} - \frac{s_{lj}}{k_2 \bar{s}}\right), \tag{1.17}$$

gdje je $\bar{p}$ prosječno trajanje svih poslova, $\bar{s}$ je prosječno trajanje postavljanja za sve parove poslova, a k_1 i k_2 su parametri heuristike. Parametri k_1 i k_2 računaju se prije početka raspoređivanja i služe za procjenu stupnja težine problema po pitanju željenih vremena završetaka i trajanja postavljanja. Autori predlažu sljedeći način računanja ovih parametara; prvo se procjenjuju vrijednosti parametra *T* (postotka zaostajanja) i *R* (područja zaostajanja):

$$R = \frac{d_{jMAX} - d_{jMIN}}{\hat{C}_{MAX}}, \quad T = 1 - \frac{d_{AVG}}{\hat{C}_{MAX}}, \tag{1.18}$$

gdje su d_{jMAX}, d_{jMIN} i d_{AVG} najveće, najmanje i srednje željeno vrijeme završetka, a $\hat{C}_{MAX}$ je procjena ukupnog trajanja koja se računa kao:

$$\hat{C}_{MAX} = \sum_{j=1}^{n} p_j + n \cdot \bar{s} \,. \tag{1.19}$$

Osim navedenoga, procjenjuje se i omjer trajanja postavljanja i trajanja obrade poslova kao:

$$\eta = \frac{\bar{s}}{\bar{p}}, \tag{1.20}$$

nakon čega se parametri k_1 i k_2 dobivaju sljedećim izrazima:

$$\begin{aligned} k_1 &= \begin{cases} 4.5 + R, & \text{za } R \le 0.5 \\ 6 - 2R, & \text{za } R > 0.5 \end{cases}, \\ k_2 &= \frac{T}{2\sqrt{\eta}}. \end{aligned} \tag{1.21}$$

Ovo se pravilo smatra najuspješnijim za probleme sa trajanjima postavljanja i gotovo se uvijek koristi u slučajevima kada neki složeniji postupak pretraživanja traži kvalitetno početno rješenje. Podrazumijeva se da se sva pravila primjenjuju kao i u ostalim statičkim uvjetima, odnosno po algoritmu 1.

2 Raspoređivanje na paralelnim jednolikim strojevima

U klasičnoj teoriji raspoređivanja postoje tri glavne vrste okruženja raspoređivanja na paralelnim strojevima: identični, jednoliki i nesrodni strojevi. Problem raspoređivanja na identičnim strojevima može se smatrati specijalnim slučajem raspoređivanja na jednolikim strojevima u kojemu su brzine svih strojeva jednake. Često se i postupci raspoređivanja u literaturi navode zajednički za ta dva okruženja. Iz toga se razloga ovdje razmatraju samo primjeri za okruženje jednolikih strojeva, dok su identični strojevi izostavljeni.

Za sve promatrane inačice raspoređivanja na paralelnim jednolikim strojevima vrijede sljedeće pretpostavke:

- svi podaci o poslovima unaprijed su poznati (predodređeno raspoređivanje),
- raspoređivanje se obavlja dinamički (tijekom rada sustava),
- poslovi su neprekidivi,
- strojevi su neprekidno raspoloživi od početka rada sustava.

Pretpostavka o mogućnosti korištenja strojeva od početka rada označava statičku raspoloživost strojeva. Iako se ovdje ne promatra dinamička raspoloživost strojeva, odnosno situacija kada su vremena pripravnosti strojeva različita od nule, sva se ispitivana pravila mogu vrlo jednostavno prilagoditi ovoj promjeni, budući se u postupku raspoređivanja ionako čeka da neki stroj postane slobodan prije donošenja odluke.

Za postupke raspoređivanja na paralelnim identičnim ili jednolikim strojevima sa dinamičkom raspoloživošću i uz bilo koji pravilni kriterij problem se uvijek može svesti na problem sa samo jednim strojem. Može se dokazati [Mor 93] da je optimalno rješenje moguće

dobiti na sljedeći način: poredati sve poslove u 'odgovarajući' niz te rasporediti redom svaki posao iz niza na stroj koji najprije može završiti promatrani posao. Ovaj postupak je moguć budući se svaki posao može obaviti na bilo kojem stroju. Naravno, najveći problem je upravo pronalaženje 'odgovarajućeg' redoslijeda poslova koji bi na ovaj način raspoređivanja dao i optimalno konačno rješenje.

Kako se ovaj oblik okruženja može svesti na problem uređivanja poslova u niz, moguće je bez puno promjena za raspoređivanje iskoristiti pravila namijenjena raspoređivanju na jednom stroju. Uputno je, međutim, pravilu raspoređivanja dati dodatne informacije o svojstvima okoline. U okruženju jednolikih strojeva postoji nekoliko netrivijalnih kriterija ocjene rasporeda, no zbog opsega istraživanja pokusi se i dalje temelje na kriteriju težinskog zaostajanja.

2.1 Raspoređivanje uz pomoć pravila na jednolikim strojevima

Budući je pravilo raspoređivanja u užem smislu definirano samo funkcijom prioriteta poslova, za svako okruženje potrebno je definirati odgovarajući način primjene pravila. Za okruženje jednolikih strojeva, sva promatrana pravila, uključujući i pravilo izvedeno genetskim programiranjem, primjenjuju se po postupku prikazanom kao algoritam 5.

```
dok (postoje neraspoređeni poslovi)
{     čekaj dok neki stroj (k) i barem jedan posao ne postanu raspoloživi;
      odredi prioritete svih raspoloživih poslova na stroju k;
      rasporedi posao s najboljim prioritetom na stroj k;
}
```

Algoritam 5. Postupak raspoređivanja po pravilima na jednolikim strojevima

Ukoliko su u istom trenutku raspoloživa dva ili više strojeva, raspoređivanje se obavlja slijedno na jednom po jednom stroju (drugim riječima, ne obavlja se usporedba prioriteta nekoga posla na više od jednoga stroja istovremeno). Ovakav je pristup opravdan budući se svi poslovi mogu izvoditi na svim strojevima, a brzine strojeva su proporcionalne pa nema potrebe za određeni posao tražiti 'najbolji' stroj. U okruženju nesrodnih strojeva ova tvrdnja ne vrijedi, pa se za to okruženje primjenjuje drugačiji postupak.

Pravila raspoređivanja definirana u poglavlju o raspoređivanju na jednom stroju ovdje se mogu iskoristiti u neizmijenjenom obliku. Ukoliko je dopuštena dinamička raspoloživost poslova, trajanje obrade posla se također uvećava za količinu vremena do trenutka dolaska posla kako je opisano u odjeljku 1.2. Isto tako, u slučaju postojanja trajanja postavljanja određeno pravilo moguće je prilagoditi na način opisan u odjeljku 1.4 po izrazu (1.13). Na taj se način u ovom poglavlju upotrebljavaju pravila WSPT, MON, ATCS i EDD. Osim navedenih, uvršteno je i LPT (engl. *longest processing time*) pravilo čija je funkcija prioriteta definirana kao

$$\pi_j = p_j . \tag{1.22}$$

LPT pravilo uvijek odabire najdulji raspoloživi posao, što u ovom okruženju vodi do dobrih rezultata po pitanju ukupne duljine rasporeda.

Promatrana su i pravila definirana posebno za ovo okruženje, iako su po osnovnom obliku najčešće rabljena u okruženju jednoga stroja. Toj skupini pripadaju sljedeća pravila:

- mRM (engl. *modified R&M*) pravilo, definirano sa:

$$\pi_j = \frac{w_j}{\left(p_j/\hat{m}\right)} \left[\exp\left(\frac{-\left(d_j - p_j/s_k - time\right)^+}{\left(p_{AV}/\hat{m}\right)} \right) \right]; \tag{1.23}$$

- mXD (engl. *modified X-dispatch*) pravilo za dinamičku okolinu, definirano sa:

$$\pi_j = \frac{w_j}{\left(p_j/\hat{m}\right)}\left[\exp\left(\frac{-\left(d_j-\left(p_j/s_k\right)-time\right)^+}{\left(p_{AV}/\hat{m}\right)}\right)\right]\cdot\left(1-B\frac{\left(r_j-time\right)^+}{\left(p_{AV}/\hat{m}\right)}\right), \qquad (1.24)$$

gdje je k indeks stroja za koji se računaju prioriteti poslova. Važno je naglasiti bitnu razliku u primjeni ova dva i ostalih pravila: u potonja dva pravila u izrazu se javlja brzina određenog stroja, te je prioritet koji pravilo dodjeljuje poslovima ovisan o tome na kojem se stroju raspoređivanje trenutno obavlja. Također se u slučaju postojanja trajanja postavljanja ova pravila ne prilagođuju na način jednak ostalima. Budući se u definiciji funkcije prioriteta ovih pravila javlja efektivni broj strojeva $\hat{m}$, prilagodba danih pravila za trajanja postavljanja definirana je izrazom

$$\pi_{lj} = \pi_j - \frac{s_{lj}}{\left(p_{AV}/\hat{m}\right)\cdot\left(p_j/s_k\right)}, \qquad (1.25)$$

gdje je l indeks prethodnog posla (na stroju k), s_k je brzina stroja a s_{lj} je trajanje postavljanja sa prethodnog na promatrani posao (neovisno o stroju).

3 Raspoređivanje na paralelnim nesrodnim strojevima

Okolina nesrodnih strojeva u velikoj se mjeri razlikuje od okruženja jednolikih strojeva razmatranog u prethodnom poglavlju. Iako se svaki posao i dalje može izvesti na bilo kojem stroju, ovaj put je pridruživanje posla određenom stroju od velike važnosti jer su trajanja obrade poslova na različitim strojevima potpuno neovisna. Potrebno je stoga algoritmu raspoređivanja omogućiti uvid u prikladnost pojedinoga posla pojedinome stroju i na učinkovit način iskoristiti tu informaciju.

U prethodnim poglavljima korišten je model predodređenog raspoređivanja, u kojemu su svi podaci o poslovima i njihovim dolascima bili dostupni. U ovom će se poglavlju, ponajprije zbog načina rada postupaka za usporedbu (sljedeći odjeljak), rabiti model raspoređivanja na zahtjev. U takvim uvjetima raspoređivanja podaci o poslovima dostupni su tek po dolasku posla u sustav, što se u većini primjera poklapa sa vremenom pripravnosti posla. Uz ovu razliku, vrijede sljedeće pretpostavke:

- podaci o poslovima nisu unaprijed poznati (raspoređivanje na zahtjev), što povlači dinamički način izrade rasporeda,
- poslovi su neprekidivi,
- strojevi su neprekidno raspoloživi od početka rada sustava.

3.1 Postupci raspoređivanja na nesrodnim strojevima

Prilikom raspoređivanja na nesrodnim strojevima, dolaze do izražaja dvije faze raspoređivanja poslova: pridruživanje, pri kojemu se poslovi dodjeljuju odgovarajućim strojevima, i uređivanje, pri čemu se određuje redoslijed eventualnog većeg broja poslova na jednom stroju. Neki postupci raspoređivanja ove dvije faze obavljaju odvojeno, a kod nekih je jedna sadržana u drugoj (npr. uređivanje se obavlja redom kojim su poslovi pridruženi stroju). Također je dvije različite metode pridruživanja i uređivanja moguće obuhvatiti u jedinstveni postupak raspoređivanja.

Algoritmi raspoređivanja u dinamičkim uvjetima raspoređivanja na zahtjev mogu se razlikovati po tome u kojem trenutku je neki posao raspoređen na odgovarajući stroj. Jedna skupina algoritama raspoređuje pristigli posao na neki od strojeva odmah po njegovom

dolasku, čime se zapravo obavlja samo pridruživanje poslova. Uređivanje poslova na stroju podrazumijeva redoslijed izvođenja jednak redu prispijeća, tj. uređivanje po FCFS (engl. *first come first served*) principu. Druga skupina algoritama ne raspoređuje svaki posao zasebno odmah po njegovu dolasku, već se u predodređenim vremenskim intervalima zajednički raspoređuju svi do tada pristigli a još nepokrenuti poslovi (engl. *batch mode*). Skup svih poslova (zadataka) koji su trenutno u sustavu, a izvođenje im nije pokrenuto, naziva se još i *meta-zadatkom*. Prilikom raspoređivanja, obavlja se i pridruživanje i uređivanje svih poslova u meta-zadatku. U većini primjena algoritmi iz druge skupine postižu bolje rezultate, pa će se u ovom radu promatrati postupci iz te skupine.

Za okruženje nesrodnih strojeva postoji niz heuristika raspoređivanja [Mar 04, Dav 81], no najpoznatije metode su min-min i max-min heuristike. Ova dva algoritma (pogotovo min-min) smatraju se ponajboljim heurističkim algoritmima za veliki broj problema raspoređivanja [He 03, Mar 04]. Algoritmi raspoređivanja koji će kasnije dobiti naziv min-min i max-min predloženi su još u radu [Iba 77] uz nekoliko drugih postupaka. Prije opisa algoritma, potrebno je uvesti sljedeće oznake:

- p_{ij} je trajanje izvođenja posla j na stroju m_i; sva trajanja izvođenja su unaprijed poznata sa dovoljnom preciznošću;
- t_{ri} je vrijeme pripravnosti stroja m_i (s obzirom na poslove koji se na njemu trenutno izvode);
- C_{ij} je vrijeme završetka izvođenja posla j na stroju m_i; budući su vremena završetka vezana samo uz poslove, za određivanje kriterija koristi se oznaka C_j kao vrijeme završetka posla j;
- J^{meta} je skup svih raspoloživih poslova koji još nisu pokrenuti; neki od tih poslova su možda dodijeljeni nekome stroju, ali njihovo izvođenje još nije započelo pa su podložni novom pridruživanju.

Min-min postupak je prikazan kao algoritam 6. U svakom koraku algoritam računa očekivano vrijeme završetka izvođenja poslova na svakom od strojeva, uzevši u obzir vrijeme pripravnosti stroja. Potom se za svaki posao pronalazi stroj koji daje najmanje očekivano vrijeme završetka. Postupak tada raspoređuje posao koji postiže najbliže vrijeme završetka (od svih poslova) na stroj koji tom poslu daje to vrijeme završetka. Max-min algoritam također za svaki posao pronalazi stroj koji daje najmanje vrijeme završetka, no raspoređuje se onaj posao koji ima *najveće* očekivano (minimalno) vrijeme završetka među svim poslovima. U primjeni se mogu naći i postupci zasnovani na min-min algoritmu, koji uz neke prilagodbe danom okruženju najčešće daju nešto bolje rezultate raspoređivanja [He 03].

```
za (sve poslove iz J^meta)
      za (sve strojeve)
          C_ij = t_ri + p_ij ;
dok (postoje neraspoređeni poslovi u J^meta)
{     za (svaki zadatak iz J^meta)
          pronaći najmanji C_ij te stroj m_i na kojemu se postiže najranije
          vrijeme završetka;
      naći posao j sa najmanjim vremenom završetka;
      dodijeliti posao j stroju l koji daje najranije vrijeme završetka;
      ukloniti posao j iz J^meta ;
      obnoviti t_rl = t_rl + p_lj ;
```

```
        obnoviti C_lj za sve preostale poslove iz J^meta ;
}
```

Algoritam 6. Min-min postupak raspoređivanja

Važan element uporabe opisanih postupaka raspoređivanja je i odabir trenutka raspoređivanja nagomilanih poslova. U stvarnim se uvjetima obično definira neki vremenski interval nakon kojega se algoritam ponovno primjenjuje. Određivanje veličine intervala u velikoj je mjeri ovisno o vrsti sustava u kojemu se raspoređivanje obavlja. U slučaju 'predugog' intervala može se dogoditi da neki pristigli posao, koji bi zbog 'dobrih' svojstava inače bio raspoređen na određeni stroj, ne bude raspoređen na taj stroj jer je na istom stroju već pokrenut posao koji je tom stroju dodijeljen u prethodnoj iteraciji raspoređivanja. Zbog toga se u projektu, u postupku simulacije, primjenjuje najbolji slučaj: algoritam se izvodi svaki puta kada novi posao dođe u sustav. Na taj način zajamčena je najbolja moguća učinkovitost algoritma.

4 Raspoređivanje za proizvoljnu obradu

Okruženje proizvoljne obrade predstavljeno je poslovima od kojih se svaki sastoji od zadanog broja operacija. Operacije se izvode predefiniranim redoslijedom na točno određenim strojevima. U najopćenitijem obliku proizvoljne obrade, svaki posao može imati proizvoljan broj operacija (različit od broja strojeva), a različite se operacije u slijedu mogu izvoditi i na istim strojevima (tzv. višestruka proizvoljna obrada, engl. *reentrant job shop*). U literaturi se ipak najčešće javlja inačica problema u kojoj je broj operacija svakog posla jednak broju strojeva, a svaki posao se izvodi na svakom stroju točno jednom. Iako se heuristike raspoređivanja bez dodatnih izmjena mogu uporabiti i u općenitijem obliku problema, poradi lakše usporedbe učinkovitosti s primjerima iz literature u ovom radu se rješava inačica u kojoj je broj operacija svakog posla jednak broju strojeva.

Raspoređivanje procesa proizvoljne obrade u pravilu se izvodi kao predodređeno raspoređivanje, pri čemu su poznati svi parametri poslova (broj operacija i njihova trajanja na pojedinim strojevima). Primjena pravila raspoređivanja, međutim, podrazumijeva dinamički način izrade rasporeda, gdje se odluka o pokretanju neke operacije na određenom stroju donosi tijekom rada sustava. Pored navedenoga, pretpostavlja se neprekidivost operacija i dostupnost svih strojeva u cjelokupnom trajanju rada sustava.

4.1 Postupci raspoređivanja za proizvoljnu obradu

Postupcima raspoređivanja u okruženju proizvoljne obrade poklanja se puno pažnje, pa za ovaj problem postoji velik broj načina rješavanja uz pomoć različitih tehnika (evolucijske metode, ekspertni sustavi, neizrazito odlučivanje i dr.) [Kas 99, Jon 98, Cic 01]. Primarni naglasak u ovom istraživanju je na postupcima koji se mogu primijeniti u obliku pravila raspoređivanja, odnosno u dinamičkim i promjenjivim uvjetima rada. Znatan broj primjera jednostavnijih pravila raspoređivanja za ovaj problem može se naći u [Cha 96].

Prije opisa samih pravila raspoređivanja, potrebno je definirati način korištenja pravila u okruženju proizvoljne obrade. Pravilo raspoređivanja primjenjuje se kada na nekom stroju treba donijeti odluku o pokretanju sljedeće operacije. Pravilo uzima u obzir informacije o samoj operaciji, poslu kojemu ta operacija pripada i stanju na stroju prije donošenja odluke. Općeniti postupak primjene pravila prikazan je kao algoritam 7. Prilikom rada sustava potrebno je voditi računa o tome koja je operacija nekoga posla sljedeća na redu za izvođenje, na kojem stroju se treba izvoditi i kada će biti raspoloživa. Na temelju te informacije pojedini stroj može zaključiti koliko operacija čeka na obradu.

Svim pravilima raspoređivanja na raspolaganju su informacije o poslovima, njihovim operacijama i stanju na strojevima, a prilikom definiranja funkcija prioriteta koriste se sljedeće oznake:

- p_{ij} – trajanje jedne operacije posla j na stroju i (budući svaki posao ima broj operacija jednak broju strojeva, a informacija o redoslijedu operacija ne koristi se u pravilu raspoređivanja, nije potrebno navoditi redni broj operacije);
- w_j – težinska vrijednost posla;
- d_j – željeno vrijeme završetka posla;
- twk_j – (engl. *total work*) ukupno trajanje svih operacija nekoga posla;
- $twkr_j$ – (engl. *total work remaining*) ukupno trajanje svih preostalih operacija nekoga posla (preostale operacije su sve one čije izvođenje još nije započelo).

```
dok (postoje neizvedene operacije)
{     čekaj dok neki stroj za koji postoje operacije ne postane raspoloživ;
      odredi prioritete svih operacija koje čekaju na stroj;
      rasporedi operaciju najvećeg prioriteta;
      odredi sljedeću operaciju posla;
      obnovi vrijeme raspoloživosti stroja i sljedeće operacije posla;
}
```

Algoritam 7. Postupak raspoređivanja po pravilima za proizvoljnu obradu

U nastavku su definirana pravila raspoređivanja korištena za ocjenu učinkovitosti. Uz svako pravilo navedena je funkcija prioriteta kojom se odabire sljedeća operacija, uz pretpostavku da je najbolja mjera jednaka najvećoj postignutoj vrijednosti funkcije prioriteta.

- WSPT pravilo, definirano sa:

$$\pi_j = \frac{w_j}{p_{ij}}. \tag{1.26}$$

- WSPT/TWKR pravilo definirano kao:

$$\pi_j = \frac{w_j \cdot twkr_j}{p_{ij}}. \tag{1.27}$$

- WTWKR (engl. *weighted total work remaining*) pravilo, definirano sa:

$$\pi_j = \frac{w_j}{twkr_j}. \tag{1.28}$$

- SLACK/TWKR pravilo (engl. *slack per remaining process time*), definirano sa:

$$\pi_j = \frac{w_j \cdot twkr_j}{\left(d_j - twkr_j\right)}. \tag{1.29}$$

- EDD pravilo, definirano sa:

$$\pi_j = \frac{1}{d_j}. \tag{1.30}$$

- COVERT pravilo (engl. *cost over time*) [Mor 93, str. 340], definirano kao:

$$\pi_j = \frac{w_j}{p_{ij}}\left[1 - \frac{\left(d_j - ELT_{ij} - time\right)^+}{h \cdot ELT_{ij}}\right]^+, \qquad (1.31)$$

gdje je h parametar heuristike, a ELT_{ij} (engl. *estimated lead time*) označava procjenu količine vremena koju će posao j provesti u sustavu nakon završetka operacije na trenutnom stroju (i). Ova se veličina može procijeniti na nekoliko načina, no najčešće se postavlja na umnožak zadane konstante sa preostalim trajanjem obrade, tj. $ELT_{ij} = k \cdot twkr_j$. Uobičajena vrijednost za k koja se i ovdje koristi je 3, a za parametar h autori sugeriraju vrijednost 0.5 [Mor 93].

- RM pravilo za proizvoljnu obradu [Mor 93, str. 340], definirano kao:

$$\pi_j = \frac{w_j}{p_{ij}} \cdot \exp\left[\frac{\left(d_j - ELT_{ij} - time\right)^+}{h \cdot \bar{p}_i}\right], \qquad (1.32)$$

gdje je $\bar{p}_i$ prosječno trajanje svih operacija na promatranom stroju, a h je parametar heuristike za koji je u ovom radu empirijski (promatrajući rezultate na ispitnim primjerima) postavljena vrijednost od 10. ELT_{ij} se računa na jednak način kao i za COVERT pravilo.

Iako se dolasci poslova u sustav u većini primjena smatraju statičkima, tj. svi su poslovi raspoloživi od početka rada, raspoređivanje na pojedinom stroju je inherentno dinamičko, budući će pojedina operacija nekoga posla biti raspoloživa za obradu tek nakon završetka svih operacija koje joj prethode. Iz tog razloga potrebno je definirati određivanje prioriteta onih operacija za koje se zna da trebaju određeni stroj, ali im je vrijeme dolaska u budućnosti. Taj se problem može riješiti na sljedeće načine:

- bez dozvoljenog čekanja – ocjenjuju se i u obzir za pokretanje dolaze samo one operacije koje se odmah mogu pokrenuti, odnosno čije vrijeme pripravnosti je manje ili jednako trenutnom vremenu;
- uz dozvoljeno čekanje – kao u poglavlju 1.2, trajanju izvođenja operacije dodaje se količina vremena koju je potrebno pričekati do početka izvođenja;
- uz dozvoljeno čekanje s računanjem prioriteta – trajanje izvođenja ostaje nepromijenjeno, ali se pravilu raspoređivanja na raspolaganje stavlja informacija o vremenu dolaska određene operacije.

Za pravila raspoređivanja iz literature koja se koriste za usporedbu primjenjuje se prvi pristup, budući su rezultati na skupu ispitnih primjera uz dozvoljeno čekanje u prosjeku lošiji od onih u kojima se čekanje ne dozvoljava. U postupku izvođenja pravila genetskim programiranjem primjenjuje se treći pristup, budući izvedeno pravilo ima priliku naučiti iskoristiti informaciju o vremenu dolaska operacije.

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.