SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Raspoređivanje na nesrodnim strojevima

Marko Đurasević

Voditelj: *Prof. dr. sc. Domagoj Jakobović*

Zagreb, travanj, 2013

Sadržaj

1. Uvod

Raspoređivanje je proces dodjele ograničenih sredstava određenom skupu aktivnosti s ciljem optimiranja mjerila vrednovanja te raspodjele. Sredstva mogu predstavljati različite stvari, primjerice strojeve u nekom proizvodnom procesu (na primjer tiskarski strojevi u štampariji), procesor na računalu i slično. S druge strane, aktivnosti predstavljaju razne operacije, kao što su primjerice operacije u proizvodnji ili izvođenje računalnih programa. Sredstva i aktivnosti su opisani različitim karakteristikama koje su bitne za raspoređivanje. Tako su primjerice aktivnosti opisane karakteristikama kao što su trajanje, vrijeme raspoloživosti aktivnosti, rok za dovršavanje aktivnosti i slično. U literaturi se za aktivnosti veoma često koristi naziv zadatak ili posao, dok se za sredstvo koristi naziv stroj.

Jedan od glavnih ciljeva proučavanja problema raspoređivanja je smanjenje troškova uporabe sredstava. Nažalost, pokazano je kako veliki broj ovakvih problema predstavlja problem koji je veoma težak za rješavanje. Iako su ljudima danas na raspolaganju veoma brza i moćna računala, problemi koji se nastoje riješiti su isto tako postali mnogo složeniji i teži. Za mnoge takve probleme nije moguće definirati egzaktni algoritam kojim bi se mogla dobiti rješenja, već je za dobivanje rješenja potrebno upotrijebiti različite heurističke metode.

Iako postoje različite varijante raspoređivanja s obzirom na vrstu strojeva, kao što su paralelni identični strojevi, jednoliki strojevi, nesrodni strojevi i slično ovaj rad će se fokusirati samo na jednu vrstu, i to na raspoređivanje zadataka na nesrodnim strojevima.

Cilj ovog rada jest dati kratak uvod u raspoređivanje na nesrodnim strojevima te kratki pregled metoda koje se koriste za raspoređivanje. U sklopu rada kratko će se opisati raspoređivanje pomoću metaheuristika, dok će se veći naglasak staviti na pregled različitih heuristika koje su posebno izrađene upravo za problem raspoređivanja.

U drugom poglavlju dat će se veoma kratak uvod u problematiku raspoređivanja na nesrodnim strojevima, kao i opis bitnih karakteristika samih poslova i načina na koji se rasporedi mogu ocijeniti, koji je dovoljan za osnovno razumijevanje tog područja. U trećem poglavlju bit će opisane heurističke metode pomoću kojih se mogu generirati rasporedi, pri čemu će se samo kratko opisati metaheuristički postupci, dok će se ponešto detaljnije opisati heuristike koje su posebno razvijene za rješavanje ove problematike. Na kraju će se u petom poglavlju dati kraći zaključak o ovome području.

2. Okruženje nesrodnih strojeva

Okruženje nesrodnih strojeva jest takvo okruženje gdje svaki stroj obrađuje pojedini posao proizvoljno definiranom brzinom. Bitno je za naglasiti da se zbog toga ne mogu raditi usporedbe strojeva, tako da se primjerice zaključi da je neki stroj nekoliko puta brži, odnosno sporiji od nekog drugog stroja. Strojevi su u potpunosti međusobno nezavisni. Intuitivno je jasno da je zbog toga broj različitih mogućnosti na koje je moguće te poslove rasporediti veoma velik. Naime, pokazano je da traženje optimalnog rasporeda za ovakav problem spada u klasu NP teških problema, što znači da će se ovakav problem morati rješavati nekom od raznih heurističkih metoda.

Prije nego što se krene na pregled samih heurističkih metoda koje se koriste za raspoređivanje na nesrodnim strojevima, potrebno je kratko dati opis samih poslova, kao i njihovih bitnih karakteristika, kao i način na koji se može ocijeniti kvaliteta dobivenog rasporeda.

2.1. Svojstva poslova

Kao što je već navedeno, posao, odnosno zadatak, jest određena aktivnost koja se postupkom raspoređivanja želi dodijeliti nekom stroju da se na njemu izvodi, odnosno izvršava. Ovisno o okruženjima u kojima se događa raspoređivanje, poslovi se mogu sastojati od više aktivnosti ili se pak posao sastoji od samo jedne jedinstvene i nedjeljive aktivnosti. U nastavku teksta će se navesti najvažnija svojstva kojima su opisani poslovi i koji predstavljaju temelj za razumijevanje postupaka raspoređivanja. Sva ova svojstva, njihovi nazivi, kao i njihove oznake su preuzete iz [1].

Skup svih poslova označava će se s J, dok će se pojedini posao označavati s J_i, gdje indeks i predstavlja redni broj posla. Najvažnija svojstva koja su, ovisno o okolini, pridružena svakom poslu su:

- **Trajanje izvođenja posla** p_{ij} (engl. *processing time*) – označava trajanje koje je potrebno da se posao J_j obavi na stroju i.
- **Vrijeme pripravnosti posla** r_j (engl. *ready time, release time*)– označava vrijeme, odnosno trenutak u kojemu postaje raspoloživ za raspoređivanje. Prije svog vremena pripravnosti se niti jedan posao ne može izvoditi na nekom stroju.
- **Vrijeme željenog završetka posla** d_j (engl. *due date*) – označava vrijeme do kojeg je poželjno, odnosno do kojeg se očekuje da će pojedini posao završiti. Naravno, vrijeme završetka niti u kojem slučaju ne osigurava da će pojedini posao stvarno završiti prije tog vremena, ali osigurava da se u slučaju da posao završi nakon tog vremena stvara određeni trošak.
- **Vrijeme nužnog završetka posla** $\bar{d}_j$ (engl. *deadline,drop dead time*) – označava strogo vremensko ograničenje do kojega posao nužno mora završiti.
- **Težina posla** w_j(engl. *weight*) – označava prioritet nekog zadatka u sustavu. Najčešće se koristi pri određivanju troškova kod ocjenjivanja rasporeda, gdje ona predstavlja neku stvarnu mjeru kvalitete samog rasporeda.

U prethodno navedenim karakteristikama su korišteni pojmovi „vrijeme“ i „trajanje“. Pojam „vrijeme“ označava neki jedinstveni trenutak na vremenskoj liniji, dok pojam „trajanje“ označava određenu količinu vremena, odnosno interval između dva jedinstvena trenutka.

2.2. Ocjena kvalitete rasporeda

Svaki raspored koji je generiran pojedinom metodom mora se moći ocijeniti na neki način. Naravno, treba uzeti u obzir da neće u svim primjenama raspoređivanja svi kriteriji biti jednako važni. Stoga je potrebno razviti nekoliko različitih načina ocjenjivanja rasporeda. No, prije nego što se kratko opišu ti načini ocjenjivanja, prvo je potrebno definirati neke izlazne veličine sustava, koje će nam dalje poslužiti za ocjenjivanje samog rasporeda. Sve oznake i nazivi izlaznih veličina su preuzeti iz [1].

- **Vrijeme završetka** C_j(engl. *completion time*) – predstavlja trenutak u kojemu je posao s indeksom j završio s izvođenjem
- **Protjecanje** F_j (engl. *flowtime*) – predstavlja količinu vremena koju je neki posao proveo u sustavu, odnosno količinu vremena od kada je posao postao pripravan za izvođenje, do trenutka kada je on završio s izvođenjem: $F_j = C_j - r_j$
- **Kašnjenje** L_j (engl. *lateness*) – predstavlja razliku između vremena završetka posla i vremena željenog završetka posla (potrebno je uočiti da ona može biti pozitivna i negativna): $L_j = C_j - d_j$
- **Zaostajanje** T_j (engl. *tardiness*) – predstavlja pozitivan iznos kašnjenja nekog posla (ukoliko je kašnjenje negativno, ono se postavlja na nulu): $T_j = \max\{0, L_j\}$
- **Preuranjenost** E_j (engl. *earliness*) – predstavlja negativan iznos kašnjenja nekog posla, odnosno koliko je ona ranije završila od željenog vremena završetka: $E_j = \max\{0, -L_j\}$
- **Zakašnjelost** U_j – predstavlja oznaku koja pokazuje je li neka aktivnost prekoračila željeno vrijeme završetka: $U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}$

Nakon što su sada opisane sve potrebne izlazne veličine sustava, mogu se opisati i načini vrednovanja dobivenih rasporeda.

- **Ukupna duljina rasporeda** C_{max} (engl. *makespan*) – predstavlja posljednje vrijeme završetka svih poslova u sustavu: $C_{max} = \max\{C_j\}$
- **Najveće kašnjenje** L_{max} (engl. *maximum lateness*) – predstavlja najveće vrijeme kašnjenja nekog posla u sustavu: $L_{max} = \max\{L_j\}$
- **Težinsko protjecanje** F_w (engl. *weighted flowtime*) – definira se kao suma težinskog protjecanja svih poslova u sustavu $F_w = \sum_j w_j U_j$
- **Težinsko zaostajanje** T_w (engl. *weighted tardiness*) – definira se kao težinska suma zaostajanja svih poslova: $F_w = \sum_j w_j T_j$
- **Težinski broj zaostalih poslova ili težinska zakašnjelost** U_k (engl. *weighted number of tardy jobs*) – definirana je kao težinska suma svih zaostalih poslova: $F_w = \sum_j w_j U_j$

- **Težinska preuranjenost i težinsko zaostajanje** ET_w(engl. *wighted earliness and weighted tardiness*) – definirano je kao zbroj težinske preuranjenosti i težinskog zaostajanja, s time da su težinski faktori posebno definirani za obje te vrijednosti: $ET_w = \sum_j (w_{Ej}E_j + w_{Tj}T_j)$

Naravno da ovdje navedeni načini pomoću kojih se mogu ocijeniti rasporedi nisu jedini koji se koriste, a moguće je izraditi i posebne ocjene za neke specifične rasporede, no ovdje navedeni predstavljaju skup najčešće korištenih načina ocjenjivanja rasporeda.

Koji će se od navedenih kriterija za ocjenjivanje rasporeda koristiti ovisi o tome koji su prioriteti postavljeni na sam raspored. Za pojedine poslove je bitnije da se oni svi što prije završe, dok je za druge bitnije da ima što manje kašnjenja poslova, ali nije toliko bitno hoće li ukupno trajanje poslova biti što je manje moguće. Moguće je, naravno, i pojedinim poslovima dati veći prioritet, tako da će ukupna ocjena više ovisiti o tome kako su raspoređeni ti poslovi. Raspored je moguće ocijeniti i s nekoliko kriterija, pa onda odabrati onaj daje najbolje rezultate za te kriterije.

Vidljivo je kako su mogućnosti ocjenjivanja rasporeda mnogobrojne i kako odabir načina ocjenjivanja rasporeda ovisi o zahtjevima i prioritetima koji su postavljeni na raspored.

3. Tehnike raspoređivanja na nesrodnim strojevima

Kao što je već ranije napomenuto, a što je vjerojatno i intuitivno jasno iz prethodnih poglavlja, raspoređivanje na nesrodnim strojevima predstavlja veoma težak i zahtjevan problem za koji ne postoji jedinstven način rješavanja, odnosno algoritam pomoću kojeg bi se moglo dobiti optimalno rješenje. Upravo zbog tih razloga, za rješavanje ovog problema raspoređivanja koriste se heurističke metode, koje nam ne moraju nužno dati optimalno rješenje problema, ali mogu dati neko dovoljno dobro rješenje koje zadovoljava naše potrebe. Heurističke metode koje se koriste za rješavanje ovog problema moguće je podijeliti u algoritme koji pretražuju prostor rješenja i algoritme koji rješenje grade izravno. Prvo će se veoma kratko opisati prva vrsta algoritama, a onda će se nešto detaljnije opisati druga vrsta algoritama, i izdvojiti pojedini algoritmi koji se ističu unutar te skupine.

3.1 Metode pretraživanja prostora stanja

Kao što se mnogi drugi NP-teški problemi rješavaju uz pomoć raznih evolucijskih i populacijskih algoritama, tako se i problem raspoređivanja može napasti ovim algoritmima. Ovakvim algoritmima je naravno teško dobiti optimalna rješenja, ali se nastoji dobiti rješenja koja su dovoljno dobra i koja zadovoljavaju postavljene uvjete. Algoritmi koji se često primjenjuju u ove svrhe su: genetski algoritmi, simulirano kaljenje, optimizacija rojem čestica, genetsko simulirano kaljenje, tabu pretraživanje i mnogi drugi.

Generiranje rasporeda pomoću ovih algoritama se pokazala kao dobra praksa. Ukoliko je algoritam dobro dizajniran i implementiran, on može dati veoma dobra rješenja za zadani problem. Kako se ipak radi o nedeterminističkim algoritmima čija krajnja rješenja uvelike ovise o početnim rješenjima koja se obično generiraju potpuno nasumično, veoma je često potrebno po nekoliko puta izvoditi takve algoritme da se dobiju bolja rješenja. No taj problem može se riješiti tako da se početna populacija ne generira baš sasvim slučajno, već da se prilikom generiranja početne populacije koriste pojedine heuristike za izradu početnih rješenje (takve heuristike će biti opisane u sljedećem poglavlju).

Naravno da ovi algoritmi imaju i poneke nedostatke, koji imaju za posljedicu da ih nije uvijek moguće primijeniti za rješavanje ovog problema. Svakako jedan od najčešće navođenih nedostataka ovih postupaka je njihova sporost. Izvođenje ovakvih algoritama je veoma dugotrajno i ukoliko je potrebno izgenerirati raspored u veoma kratkom vremenu, njihovi rezultati će najčešće biti veoma loši. Ovakvi algoritmi su također dosta složeni za implementaciju, i potrebno je utrošiti mnogo vremena za traženje idealnih parametara za njihovo izvođenje, kako bi se dobili što bolji rezultati. Ponekad je također moguće da se uvjeti raspoređivanja mijenjaju tijekom vremena (dolazak novih poslova). Ovakvi algoritmi nisu baš prikladni za takvo raspoređivanje, nego se oni uglavnom koriste za raspoređivanje u statičkom i predodređenom okruženju.

Kao što je vidljivo, ovakvi načini za izradu rasporeda predstavljaju dobro rješenje ukoliko je dostupno dovoljno vremena za izradu samog rasporeda i ako se radi o predodređenom statičkom raspoređivanju. U suprotnome, ovakve metode nije

moguće primijeniti te je potrebno potražiti neki drugi način za izradu rasporeda, a to su upravo metode koje rješenja grade izravno.

3.2 Metode za izravnu izgradnju rješenja

U slučajevima kada nam je bitna brzina i u uvjetima kada je potrebno brzo reagirati na promjene u stanju sustava, najčešće se koriste upravo postupci koji rješenje grade izravno. To su posebno razvijene heuristike koje se koriste upravo za rješavanje ovog problema. Ove heuristike, najčešće nisu ništa drugo, nego zapravo funkcije koje na temelju stanja sustava određuju kako rasporediti pojedini posao.

Ovakvi postupci raspoređivanja nude nekoliko prednosti. Prije svega brzina izvođenja ovih postupaka je gotovo zanemariva, posebno uspoređujući ih s pojedinim evolucijskim postupcima. Ovi postupci su mnogo jednostavniji za implementaciju, jer se obično radi o algoritmima od nekoliko jednostavnih koraka. Ovakvi algoritmi veoma brzo reagiraju na promjene, što znači da se sve eventualne promjene u sustavu mogu odmah uzeti u obzir.

Naravno da ovi postupci donose sa sobom i određene nedostatke. Sigurno najveći nedostatak ovih postupaka jest da daju rješenja koja su lošija od rješenja dobivena nekim složenijim postupkom (kao primjerice nekim složenim genetskim algoritmima). To je svakako posljedica jednostavnosti ovakvih algoritama. Osim toga, nije uvijek potpuno jasno koju je heuristiku najbolje odabrati za pojedini kriterij ocjenjivanja rasporeda. Uspješnost izvođenja heuristike uvelike ovisi i o svojstvima poslova koje je potrebno rasporediti, tako da u nekim specifičnim slučajevima heuristike koje općenito daju lošije rezultate, mogu generirati bolji raspored za taj specifični problem. No kako je izvođenje ovih heurističkih metoda jako kratko, može se za neki problem izvesti više njih te odabrati raspored koji najviše udovoljava postavljenim kriterijima.

U nastavku poglavlja bit će opisani najbitniji predstavnici ove skupine algoritama.

3.2.1 OLB

OLB (engl. *Opportunistic Load Balancing*) je jednostavna heuristika koja svaki posao nasumično dodjeljuje na idući slobodni stroj, ne uzimajući u obzir trajanje izvođenja tog posla na pojedinim strojevima.

3.2.2 MET

MET (engl. *Minimum Execution Time*) je heuristika koja svaki posao pridjeljuje stroju koji ima najmanje očekivano trajanje izvođenja. Ne uzimaju se u obzir dostupnost stroja, kao niti njegova opterećenost.

3.2.3 MCT

MCT (engl. *Minimum Completition time*) je heuristika koja za svaki posao računa najmanje očekivano trajanje izvođenja ukoliko bi se taj posao dodao na taj stroj. Odabire se onaj stroj za koji se dobila najmanja vrijednost očekivanog trajanja

izvođenja te se na taj stroj raspoređuje odabrani posao. Ova heuristika uzima u obzir samo jedan posao u svakom trenutku.

3.2.4 Min-min algoritam

Svakako jedna od najpopularnijih i najčešće korištenih heuristika koja daje veoma dobre rezultate je Min-min algoritam. Iako se odlikuje iznimnom jednostavnošću, prosječni rezultati dobiveni ovim algoritmom su jedni od najboljih od svih heuristika opisanih u ovom radu.

Postupak počinje sa skupom neraspoređenih poslova. U prvoj fazi postupka, za svaki posao iz skupa se traži stroj koji će dati najmanje očekivano vrijeme završetka za taj posao. Nakon toga, u drugoj fazi, postupak raspoređuje posao koji postiže najbolje očekivano vrijeme završetka na stroj na kojem se to vrijeme završetka može ostvariti. Nakon što je posao raspoređen na odabrani stroj potrebno je ponovo izračunati očekivano vrijeme završetka svih raspoređenih poslova na svakom od strojeva. Nakon što je to obavljeno, postupak se ponavlja, sve dok nisu raspoređeni svi poslovi.

Iz opisa algoritma vidljivo je da se ovaj algoritam kao funkciju odlučivanja koristi upravo očekivano vrijeme završetka, te da ovaj postupak radi upravo tako da se odabire onaj posao, za koji će nakon raspoređivanja očekivano vrijeme završetka biti najmanje moguće. Složenost ovog algoritma je $O(ms^2)$ gdje m predstavlja broj strojeva, a s predstavlja broj poslova.

U opisima algoritama koji slijede vidjet će se kako mnogi od njih koriste sličnu logiku kao i Min-min algoritam, štoviše, neke će heuristike koristiti Min-min algoritam u jednoj od svojih faza izvođenja.

3.2.5 Max-min algoritam

Druga jako popularna i često korištena heuristika je svakako Max-min algoritam. Sam algoritam je veoma sličan Min-min algoritmu.

Postupak također počinje sa skupom još neraspoređenih poslova. Prva faza je u potpunosti jednaka prvoj fazi Min-min algoritma, za svaki posao se traži stroj koji će za taj posao dati najmanje očekivano vrijeme završetka. U drugoj fazi se, za razliku od Min-min postupka, ne raspoređuje onaj posao koji daje minimalno očekivano vrijeme završetka, već onaj koji daje maksimalno očekivano vrijeme završetka među svim poslovima. Ostatak postupka je isti kao kod Min-min algoritma, izračuna se očekivano vrijeme završetka svih strojeva i nakon toga se ponavlja cijeli postupak dok svi ostali poslovi nisu raspoređeni.

Kao što se iz opisa vidi, radi se o potpuno istom algoritmu, samo se posao koji će se rasporediti odabire po drugačijem kriteriju. Ovaj postupak nastoji izvršiti poslove koji se dulje izvode što prije. U slučajevima kada postoje poslovi koji se izvode puno dulje od ostalih, ovaj postupak može dati bolja rješenja od Min-min algoritma. Složenost postupka je jednaka kao i složenost Min-min algoritma, dakle ona iznosi $O(ms^2)$.

3.2.6 Duplex algoritam

Ovaj algoritam je doslovno kombinacija prethodno dva navedena algoritma, odnosno Min-min i Max-min algoritma. Izvode se oba navedena algoritma i na kraju odabire bolje od dva dobivena rješenja. U ovakvoj izvedbi, ovaj algoritam je primjenjiv na samo Na taj način se uvijek dobiva bolje rješenje. Kako su troškovi ovakvih algoritama veoma maleni, gotovo zanemarivi, čak ni izvođenje dva ovakva algoritma neće predstavljati neko opterećenje (posebice ako se mogu izvoditi paralelno u isto vrijeme, a ne sekvencijalno).

Postoji heuristika koja je također kombinira Min-min i Max-min heuristiku, no na ponešto drugačiji način. Ova heuristika je detaljnije opisana u [3]. Ukratko, ova heuristika u svakom koraku odabire hoće li se za raspoređivanje upotrijebiti Min-min ili Max-min. Na temelju određenih uvjeta, odnosno funkcije koja na temelju stanja susta odlučuje koja će se od ovih heuristika upotrijebiti. Pokazalo se da ova heuristika postiže rezultate koji su jednaki ili čak i bolji od načina da se individualno primjene Min-min i Max-min heuristike (kriterij mjerenja je bio ukupna duljina rasporeda).

3.2.7 Sufferage algoritam

Ideja ove heuristike jest da se na određeni stroj rasporedi točno onaj posao koji bi najviše „propatio” (engl. *to suffer*) ukoliko se ne bi rasporedio točno na taj određeni stroj.

Za svaki posao se, osim minimalnog očekivanog vremena završetka posla, traži i drugo najmanje vrijeme završetka tog posla na nekom stroju. Razlika između ta dva vremena se definira kao „patnja” (engl. *sufferege*). Nakon toga, odabere se posao koji ima najveću vrijednost „patnje” i rasporedi taj posao na stroj na kojem on ima najmanje očekivano vrijeme završetka. Ova heuristika obično daje rezultate koji su nešto bolji od Max-min heuristike, no obično još uvijek lošiji od Min-min heuristike. Postoji i poboljšanje ove heuristike o kojem se može više pročitati u [6].

3.2.8 LJRF-SJRF algoritam

Puni naziv ove heuristike je *Longest Job to Fastest Resource - Shortest Job to Fastest Resource.* Ova heuristika započinje sa skupom neraspoređenih poslova. Isto kao i Min-min algoritam, prvo se za svaki posao odredi minimalno očekivano vrijeme završetka. Nakon toga se posao s najmanjom vrijednošću očekivanog vremena završetka postavi kao najkraći posao (SJFR), a posao s najvećom vrijednošću očekivanog vremena završetka kao najdulji posao (LJFR). Na početku algoritam dodijeli m najduljih poslova na m dostupnih strojeva, a nakon toga se alternira između odabira najkraćeg i najduljeg posla kojeg će rasporediti na određeni stroj. Rezultati dobiveni ovom heuristikom obično su dosta lošiji od rezultata dobivenih s heuristikama Min-min i Max-min i nije baš često upotrebljavana. Spomenuta je samo radi opisa principa koji je primijenjen ovom heuristikom.

3.2.9 Min-max algoritam

Min-max algoritam je nešto novija heuristička metoda koja je predložena u [2]. Ona se sastoji od dvije faze za dodjeljivanje svih poslova na strojeve. U prvoj fazi algoritma se kao metrika koristi očekivano vrijeme završetka, dok se u drugoj fazi koristi najmanje trajanje izvršavanja posla. U prvom koraku se dakle započinje sa skupom svih neraspoređenih poslova. Ova je faza u potpunosti ista kao prva faza Min-min algoritma, naime za svaki posao se odredi stroj na za koji taj posao ima najmanje očekivano vrijeme završetka. U drugoj fazi će se rasporediti onaj posao koji ima najveći omjer između najmanjeg vremena izvršavanja (vrijeme potrebno da se posao izvrši na najbržem stroju) i vremena izvođenja na stroju koji je odabran za ovaj posao u prvoj fazi algoritma.

Ideja iza ovog algoritma je da se odabere onaj par poslova i strojeva koji će dati bolje (kraće) vrijeme izvršavanja tog posla od stroja na kojem bi se inače taj posao izvodio. Pokazano je kako je kako ovaj algoritam daje rezultate koji su u rangu, pa čak i bolji od Min-min algoritma po pitanju ukupne duljine rasporeda.

3.2.10 Min-mean algoritam

Ova heuristika je također nešto novija i mlađa od ostalih navedenih heuristika. Min-mean heuristika radi tako da u prvom koraku generira gotov raspored uz pomoć Min-min metode. U drugom koraku se računa srednja vrijednost trenutka dovršetka rada (trenutak kada su se svi poslovi koji su bili raspoređeni na taj stroj izvršili) svih strojeva. U idućoj fazi se odabiru samo oni strojevi koji imaju očekivano vrijeme dovršetka poslova veće od srednje vrijednosti. Za svaki takav stroj provodi se ponovno raspoređivanje svih poslova, na način da se pojedini posao dodijeli onom stroju za kojeg će očekivano vrijeme izvođenja biti najmanje moguće. Logika ove heuristike je ta da se pomoću metode koja daje inače dosta dobre rezultate generira početni raspored, koji se onda kroz drugu fazu pokušava još više rafinirati te se njegovo ukupno vrijem izvršavanja nastoji smanjiti.

Eksperimentalni rezultati koji su provedeni su pokazali kako ovaj algoritam postiže rezultate koji su bolji od algoritama kao što su Min-min i Max-min. Ovaj algoritam je pokazao da je možda prilikom dizajniranja heuristika potrebno krenuti novim putem, odnosno da nije potrebno pronaći novi način koji će direktno raspoređivati poslove na strojeve i time postići bolje očekivano vrijeme izvođenja, već možda iskoristiti neku postojeću metodu kako bi se takav raspored generirao, a onda ga nekim drugim algoritmom pokušati poboljšati. Više o rezultatima i detaljima ove heuristike može se naći u [4].

Postoji i varijanta ove metode koja uz jednu jako malu promjenu daje ponešto bolje rezultate. Promjena koju je potrebno napraviti jest da se srednjoj vrijednosti trenutka završetaka rada svih strojeva doda određena konstantna vrijednost. Eksperimentalnim rezultatima je pokazano da je ta vrijednost konstante idealna ako se uzme 7% dobivene srednje vrijednosti. Rezultati dobiveni na taj način su dosta bolji od metode bez dodavanja konstante srednjoj vrijednosti. Ostatak algoritma je potpuno analogan osnovnoj inačici Min-mean algoritma. Više o poboljšanoj inačici Min-mean algoritma kao i eksperimentalnim rezultatima koji su dobiveni može se naći u [5].

4. Zaključak

Napretkom industrije, znanosti i tehnologije dolazi do sve izraženijih zahtjeva za izradom što boljih rasporeda obavljanja određenih poslova s ciljem smanjenja trajanja izvođenja i povećanja produktivnosti. Nažalost, takvi rasporedi su veoma složeni te najčešće dobivanje optimalnog rješenja za taj problem nije moguće obaviti u nekom razumnom vremenu pomoću nekog egzaktnog algoritma ili postupka. Zbog tog razloga, pribjegava se korištenju posebno izrađenih postupaka, odnosno heuristika kojima se nastoji ostvariti rješenje koje je dovoljno dobro, ali ne nužno i optimalno.

Iako se ovakvi problemi mogu rješavati raznim metaheurističkim postupcima, kao što su primjerice genetski algoritmi, simulirano kaljenje, tabu pretraživanje i slično, takav način ima određene nedostatke (od kojih su trajanje izvođenja i njihova velika složenost sigurno jedni od najznačajnijih) koji su izvor motivacije za traženje alternativnih metoda za provedbu raspoređivanja. Rezultat te motivacije jesu razne heuristike koje su specijalno razvijene i posebno prilagođene za rješavanje ovakvih problema. Ovakvi heuristički postupci se odlikuju iznimnom jednostavnošću i brzinom izvođenja. Bez obzira na njihovu iznimnu jednostavnost, rezultati koji su dobiveni njihovom primjenom su veoma dobri i primjenjivi u praksi. Problem koda takvih metoda je svakako to što se njima dobivaju rasporedi koji su relativno lošiji u odnosu na rezultate dobivene dobro izrađenim evolucijskim algoritmima. Osim toga, ovisno o konfiguraciji problema raspoređivanja pojedine metode će davati bolje rezultate te stoga nije moguće uvijek primijeniti jednu te istu metodu za raspoređivanje. No kako je brzina izvođenja ovakvih metoda veoma kratka, moguće je generirati rasporede pomoću više različitih metoda te odabrati onaj koji je najbolji među njima.

Trenutno postoji veoma mnogo različitih heurističkih metoda za rješavanje problema raspoređivanja. Dvije najpopularnije metode su sigurno Min-min i Max-min. No bez obzira na to još uvijek se istražuju novi postupci kojima se nastoje postići još bolji rasporedi. Iz tog razloga javlja se sve više novih postupaka kojima se nastoje postići bolja rješenja rasporeda i u tome se nerijetko i uspijeva (Min-mean, Min-max).

Kao što se može zaključiti u ovom području postoji još mnogo prostora za napredak i istraživanje. Jedan smjer istraživanja svakako predstavlja razvoj jednostavnih heuristika radi postizanja boljih rasporeda, dok drugi pokušava kombinirati principe iz metaheurističkih postupaka s ovim prilagođenim heuristikama i na taj način ostvariti bolje rasporede pomoću hibridnih postupaka. U svakom slučaju radi se o veoma dinamičnom području koje nudi još mnogo prostora za daljnje istraživanje i proučavanje.

5. Literatura

[1] Jakobović, D. „Raspoređivanje zasnovano na prilagodjivim pravilima“, http://www.zemris.fer.hr/~yeti/doktorat/doktorat.pdf

[2] Hesam Izakian, Ajith Abraham, „Comparison of Heuristics for Scheduling Independendent Tasks on Heterogenous Distributed Environments“, 2009., Computational Sciences and Optimization

[3] Kobra Etminani, M. Naghibzadeh, „A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling“, 2007. Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on

[4] Kamalam.G.K and Murali Bhaskaran.Vm „A New Heuristic Approach: Min-Mean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems“, 2012. International Journal of Machine Learning and Computing

[5] G. K. Kamalam and V. Murali Bhaskaran, „An Improved Min-Mean Heuristic Scheduling Algorithm for Mapping Independent Tasks on Heterogenous Computing Environment“, 2010., International Journal of Computational Cognition

[6] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, “Heuristics for Scheduling Parameter Sweep Applications in Grid Environments“ 2000., Heterogeneous Computing Workshop

6. Sažetak

U ovom radu napravljen je kratki uvod u problematiku raspoređivanja na nesrodnim strojevima. Opisana su osnovna svojstva strojeva i zadataka koji se raspoređuju na njih koji su bitni za razumijevanje ove problematike. Osim toga opisani su i kriteriji kojima se napravljeni rasporedi mogu generirati.

U ostatku rada daje se omanji pregled metoda koje se mogu koristiti za izradu rasporeda za okolinu nesrodnih strojeva. Kratko su opisani metaheuristički postupci, njihove prednosti i nedostaci. Nadalje opisana je i druga mogućnost generiranja rasporeda korištenjem posebno razvijenih heuristika za rješavanje ovih problema te su opisane prednosti i nedostaci u usporedbi s metaheurističkim postupcima. Napravljen je pregled relevantnih postupaka koji se koriste u ovom području, te je dan njihov kratki opis.