SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1991

RASPOREĐIVANJE U OKRUŽJU NESRODNIH STROJEVA UPORABOM EVOLUCIJSKOG RAČUNANJA

Karlo Knežević

Zagreb, srpanj 2011.

Posvećujem ovaj rad svojim roditeljima,
sestri,
bakama i djedu.

Zahvaljujem mentoru, doc.dr.sc. Domagoju Jakoboviću,
na pruženim savjetima tijekom izrade ovog rada.

Sadržaj

Uvod

Raspoređivanje je proces koji se bavi dodjelom ograničenih sredstava skupu aktivnosti u cilju optimiranja jednog ili više mjerila vrednovanja. Ovisno o danim uvjetima, sredstva i aktivnosti mogu poprimiti mnoge oblike. Sredstva mogu biti strojevi u proizvodnom pogonu, procesor računala, spremnički i komunikacijski uređaji u računalnom sustavu itd. Aktivnosti mogu biti razne operacije u proizvodnji, izvođenja računalnih programa, popravci u radionici i druge. Rješenje problema raspoređivanja je raspored koji definira kada će se i na kojem sredstvu odvijati pojedina aktivnost. Vrednovanje rasporeda moguće je definirati po puno različitih kriterija, kao što je ukupno vrijeme izvođenja, broj zakašnjelih aktivnosti itd [4].

Raspoređivanje u posljednje vrijeme, razvojem tehnologije i nastankom sve složenijih zahtjeva, znatno dobiva na važnosti. Računarstvo u oblaku (engl. *cloud computing*) primjer je jedne od aktualnih tehnologija, čija učinkovitost i brzina izvođenja zadataka uvelike ovisi o rasporedu zahtjeva na poslužiteljima. Razvoj tehnologije doprinio je razvoju novih tehnika raspoređivanja od kojih je jedna predstavljena u ovom radu. Riječ je o algoritmu kolonije mrava, algoritmu zasnovanom na inteligenciji roja i jednom od algoritama strojnog učenja. Bez razvoja tehnologije taj algoritam ne bi bio primjenjiv u ovom stupnju u kojem jest, jer uvelike ovisi o brzini rada računala.

U ovom radu naglasak je na raspoređivanju u okružju nesrodnih strojeva. Za navedeno okružje postoji niz heuristika raspoređivanja, a jedna od najpoznatijih metoda je min-min heuristika. Ovaj heuristički algoritam pruža zadovoljavajuća rješenja, ali moguće je njegovo poboljšanje, tako da su u nekim slučajevima postignuti bolji rezultati uporabom algoritma kolonije mrava.

Ispitivanja su provedena za dva načina raspoređivanja, predodređeno raspoređivanje i raspoređivanje na zahtjev, i četiri različite heuristike koje koristi algoritam kolonije mrava. Ispitana je dobrota rasporeda u odnosu na veličinu kolonije mrava i broj generacija mrava.

Cilj ovog rada je pokazati dobrotu heurističkog algoritma min-min u odnosu na algoritam kolonije mrava.

Rad čini pet poglavlja u kojima se navode problemi koji se rješavaju, kao i način njihovog rješavanja. U prvom poglavlju opisuje se algoritam kolonije mrava, svojstva mrava u prirodi i umjetnih mrava, implementacija funkcija koje algoritam koristi i vrste mravljih algoritama. U drugom poglavlju pojašnjava se pojam raspoređivanja u okružju nesrodnih strojeva, zapis okruženja raspoređivanja, način vrednovanja rasporeda i okružje nesrodnih strojeva. Također će detaljnije biti pokazana primjena raspoređivanja na nesrodnim strojevima. U trećem poglavlju predstavljeno je rješenje problema raspoređivanja na nesrodnim strojevima pomoću algoritma kolonije mrava. U poglavlju se opisuje funkcija dobrote i korištene heuristike, kao i grafička interpretacija implementacije. U četvrtom poglavlju pokazan je izgled aplikacije sa svim mogućnostima odabira raspoređivanja i optimizacija. U zadnjem, petom, poglavlju, prikazani su rezultati mjerenja.

1. Algoritam kolonije mrava

Ovo poglavlje obrađuje tematiku evolucijskog računanja, a detaljnije opisuje algoritam kolonije mrava. Navedeni su motivi za imitiranje ponašanja mrava i algoritamska implementacija, kao i vrste mravljih algoritama.

1.1. Evolucijsko računanje

U računarskoj znanosti, evolucijsko računanje dio je umjetne inteligencije, polje računarska inteligencija, koje se bavi kombinatornim optimizacijama. Evolucijsko računanje primjer je metaheuristike i danas se dijeli na dva velika područja [5]:

1. evolucijski algoritmi (engl. *evolutionary algorithms*),
2. algoritmi zasnovani na inteligenciji roja (engl. *swarm intelligence*).

Evolucijski algoritmi definiraju se kao postupci optimiranja, učenja i modeliranja, koji se temelje na mehanizmu evolucije u prirodi. Njihov nastanak uzrokovali su prvenstveno pokušaji primjene načela evolucije pri rješavanju nekih problema, ali i nastojanja za boljim razumijevanjem samo evolucije u prirodi. Ideja evolucijskih algoritama je da rade s populacijom rješenja nad kojima se primjenjuju evolucijski operatori (selekcija, križanje, mutacija, supstitucija) čime populacija iz generacije u generaciju posrtaje sve bolja i bolja [1]. Evolucijski algoritmi su:

- genetski algoritmi,
- evolucijsko programiranje,
- evolucijske strategije,
- genetsko programiranje.

Algoritmi zasnovani na inteligenciji roja definiraju se na sličan način kao evolucijski algoritmi, ali temelje se na sociološko-psihološkim principima i pružaju uvid u sociološka ponašanja. Baziraju se na populaciji rješenja. U odnosu na evolucijske algoritme, ne postoji mehanizam stvaranja nove generacije iz postojeće, tj. ne postoje evolucijski operatori, već se modeliraju individualni i socijalni faktori, čime se stvara novo znanje i nova informacija [1]. Najpoznatija dva algoritma zasnovana na inteligenciji roja su:

- algoritam (optimizacija) kolonije mrava,
- algoritam (optimizacija) roja čestica.

Osnovna prednost evolucijskog računanja jest što preslikava rješenja problema faktorijelne ili eksponencijalne složenosti u probleme polinomne složenosti. Međutim, pronađena rješenja u većini slučajeva su optimalna ili vrlo blizu optimalnih.

1.2. Mravi u prirodi

Mravi su izuzetno jednostavna bića – usporedimo li ih, primjerice, s čovjekom. No i tako jednostavna bića, zahvaljujući socijalnim interakcijama, postižu zadivljujuće rezultate. Uočeno je da mravi uvijek pronalaze najkraći put između izvora hrane i njihove kolonije, što im omogućuje da hranu dopremaju maksimalno brzo.

Uzrok tog svojstva jest specifična mravlja strategija u kojoj glavnu ulogu igra kemijska supstanca feromon kojeg mravi ostavljaju za sobom nakon što pronađu hranu. Na taj način indirektno signaliziraju drugim mravima da prate taj trag kako bi i oni pronašli isti izvor hrane. Takav oblik komunikacije poznat je pod nazivom stigmergije (engl. *stigmergy*) gdje agenti komuniciraju preko okoline koja je modificirana drugim agentima [3].

Inspiraciju za optimizaciju kolonijom mrava dao je eksperiment koji je proveo Goss sa suradnicima, 1989. godine, s argentinskim mravima (lat. *Iridomyrmex humilis*). Kako bi istražili ponašanje mrava, Gross i suradnici napravili su niz eksperimenata koristeći dvokraki most postavljen između mravinjaka i hrane. Tijekom eksperimenta varirane su duljine krakova mosta. U prvom eksperimentu oba kraka bila su jednako duga i mravi su u početku, u podjednakom broju, krenuli preko oba kraka. Nakon nekog vremena, dominantni dio mrava kretao se samo jednim krakom – slučajno odabranim. Objašnjenje eksperimenta jest da krak kojeg je nasumično odabrala većina mrava sadrži veću koncentraciju feromona koja ne hlapi jer se isti mravi u koloniju vraćaju istim krakom. Zbog jednako intenzivnog feromonskog mirisa, i drugi mravi počinju koristiti taj krak. Time je postignuta konvergencija [2].

Sljedeći eksperiment napravljen je s dvokrakim mostom kod kojeg je jedan krak dvostruko dulji od drugog. U svim pokušajima eksperimenta pokazalo se da najveći broj mrava, nakon nekog vremena, bira kraći krak. Ovakvo ponašanje na makroskopskoj razini, rezultat je interakcije na mikroskopskoj razini, individualnih i

socijalnih faktora. Takvo ponašanje temelj je onog što se danas naziva izranjajuća inteligencija (engl. *emerging intelligence*) [2].

Kako bi se provjerila dinamika, odnosno sposobnost prilagodbe mrava na promjene, napravljen je treći eksperiment. Kod ovog eksperimenta mravinjak i izvor hrane najprije su spojeni jednokrakim mostom. Mravi se kreću tim krakom od hrane i natrag. Nakon pola sata, kada se situacija stabilizirala, mostu je dodan drugi, dvostruko kraći krak. Međutim, eksperiment je pokazao da se najveći broj mrava i dalje nastavio kretati duljim krakom, zahvaljujući stvorenom jakom feromonskom tragu [2].

1.3. Pojednostavljeni matematički model

Temelj za matematički model jest teorija grafova jer se algoritam kolonije mrava generalizira na traženje Hamiltonovog ciklusa u grafu. Neka je $G = (V, E)$ povezani graf, gdje je V skup svih čvorova, a E skup svih lukova. Broj čvorova je $V = n$. Rješenje problema predstavlja ciklus na grafu koji povezuje izvorišni i odredišni čvor S, a čija duljina je jednaka broju čvorova.

Sa svakim lukom (i, j) grafa G povezana je varijabla τ_{ij}koja predstavlja umjetni trag feromona. U fazi inicijalizacije, na sve se bridove postavi ista količina feromona. Tragove feromona čitaju i pišu mravi. U svakom čvoru grafa dolazi do stohastičke odluke koji je čvor sljedeći. $k - ti$ mrav lociran u čvoru i koristi trag feromona τ_{ij} za izračun vjerojatnosti u koji čvor $j \in N_i$ treba otići. N_i je skup svih indeksa svih čvorova u koje je u koraku k moguće prijeći iz čvora i. Simbol α je konstanta.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}}, ako\ j \in N_i^k \\ 0, ako\ j \notin N_i^k \end{cases} \tag{1.1}$$

Nakon što odabere sljedeći čvor, mrav ponavlja proceduru donošenja odluke o odlasku u sljedeći čvor opisan jednadžbom (1.1) sve dok ne stigne do ciljnog čvora. Mravlji algoritmi pripadaju obitelji *konstrukcijskih algoritama* jer do konačnog rješenja dolaze akumulirajući dio po dio rješenja [3].

Kako bi se izbjeglo obilaženje istih čvorova više puta, svaki mrav ima vlastitu *tabu listu* u kojoj se pamti koji čvorovi su posjećeni. Dolaskom u ciljni čvor,

na temelju *funkcije dobrote*, mrav zna kolika je vrijednost puta kojeg je prošao. Mravi feromone osvježavaju nakon dolaska u ciljno stanje, a količina feromona proporcionalna je dobroti rješenja. Ažuriranje se radi za sve bridove kojima je mrav prošao prema izrazu (1.2).

$$\tau_{ij} = \tau_{ij} + \Delta\tau^{k} \tag{1.2}$$

Isparavanje feromona određeno je izrazom (1.3).

$$\tau_{ij} = \tau_{ij} \cdot (1 - \rho) \tag{1.3}$$

Konstanta ρ označava brzinu isparavanja ili postotak feromona koji ne isparava, a vrijednost je iz intervala $\langle 0,1 \rangle$. Temeljeći se opisanim principima, napisan je optimizacijski algoritam (1.1).

```
ponavljaj dok nije kraj
        ponovi za svakog mrava
                stvori rješenje
                vrednuj rješenje
        kraj ponovi
        odaberi podskup mrava
        ponovi za odabrane mrave
                ažuriraj feromonske tragove
        kraj ponovi
        ispari feromonske tragove
kraj ponavljanja
```

Slika 1.1 Jednostavan mravlji algoritam

Algoritam mrava radi s populacijom od m mrava, a ciklus se iterira c puta. Svih m mrava pusti se da stvore rješenja i ta se rješenja vrednuju. Kada svih m mrava stvori prijedloge rješenja, odabire se n mrava koji će obaviti ažuriranje tragova, a pri tome vrijednost n može biti jednaka m, manja od m ili jednaka jedan.

Pojednostavljenim matematičkim modelom opisana su svojstva mrava i kolonije. Svojstva kolonije kao cjeline su [3]:

- dobra rješenja mogu proizaći kao rezultat kolektivne interakcije,
- svaki mrav koristi samo privatne informacije i lokalne informacije čvora kojeg posjećuje,
- mravi komuniciraju samo indirektno,
- mravi se sami po sebi ne adaptiraju, već mijenjaju način prezentacije, problema i percepcije drugih mrava.

Svojstva mrava kao jedinke su [3]:

- mrav traži najisplativije rješenje,
- svaki mrav ima memoriju M_k za pohranu informacija o putu i može je koristiti:
  - za generiranje smislenog rješenja,
  - za evaluaciju pronađenog rješenja,
  - za povratak po istom putu.
- mrav k može se pomaknuti u bilo koji čvor j u susjedstvu N_i^k,
- mravu k može se dodjeliti početno stanje i jedan ili više terminirajućih uvjeta,
- mrav stvara rješenje inkrementalno i konstrukcija završava kada je zadovoljen jedan od uvjeta zaustavljanja,
- odabir sljedećeg čvora temelji se na vjerojatnosti,
- vjerojatnost pojedine odluke mrava temelji se na:
  - vrijednostima pohranjenim u strukturi čvora $A_{ij} = [a_{ij}]$,
  - privatnoj memoriji mrava,
  - ograničenjima problema.
- pomicanjem iz čvora i u čvor j mrav ažurira trag feromona,
- kada je izgradio rješenje i vratio se na izvorište, mrav umire i oslobađa resurse.

1.4. Algoritam Ant Colony System

Algoritam *Ant Colony System* podskup je algoritma *Ant System* kojeg su predložili Dorigo i Gambardella. U odnosu na algoritam *Ant System*, algoritam se razlikuje u tri glavne točke [3]:

1. bolje se iskorištava stečeno iskustvo pretraživanja ostalih mrava preko snažnije funkcije izbora sljedećeg čvora,
2. globalno feromonski trag isparava i ažurira samo mrav s najboljim rješenjem,
3. svaki put kad mrav prolazi bridom (i, j), lokalno isparava feromonski trag u namjeri da poveća istraživanje bridova ostalih mrava.

U algoritmu Ant System definirano je slučajno proporcionalno pravilo (engl. *random proportional rule*) koje određuje vjerojatnost prelaska iz čvora i u čvor j.

Jednadžba (1.4) koristi dvije komponente: jakost prethodno deponiranog feromonskog traga te vrijednost heurističke funkcije.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \left(\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta} \right)}, ako\ j \in N_i^k \\ 0, ako\ j \notin N_i^k \end{cases} \tag{1.4}$$

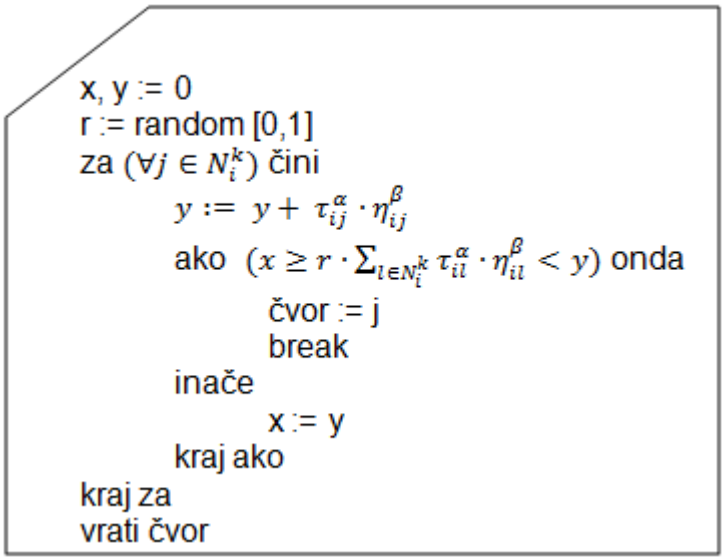
η_{ij} je heuristička informacija koja govori koliko je dobro iz čvora i prijeći u čvor j. To je informacija koja se može odrediti statički, unaprijed definirana, ili dinamički, mijenja se tijekom izvođenja algoritma. Pritom treba paziti da funkcija koja opisuje heurističku informaciju bude optimistična.

Parametri α i β određuju ponašanje algoritma. Ako je $\alpha = 0$, utjecaj feromonskog traga se poništavai pretraživanje se vodi samo heurističkom informacijom. Ako je $\beta = 0$, utjecaj heurističke informacije se poništava i ostaje utjecaj isključivo feromonskog traga, što često dovodi do prebrze konvergencije suboptimalnom rješenju.

Algoritam Ant Colony System ima snažniju funkciju izbora sljedećeg čvora, odnosno slučajno proporcionalno pravilo definirano je izrazom (1.5).

$$j = \begin{cases} argmax_{l \in N_i^k} \left\{ \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta} \right\}, ako\ q \leq q_0 \\ J, ako\ q > q_0 \end{cases} \tag{1.5}$$

Varijabla q ima slučajnu vrijednost uniformno distribuiranu na intervalu $[0,1]$, q_0 je parametar intervala $[0,1]$, a J je slučajna varijabla određena vjerojatnosnom funkcijom (1.4). S vjerojatnošću q_0 mrav radi najbolji mogući odabir čvora na temelju feromonskih tragova i heurističke informacije. Kaže se da mrav iskorištava stečeno znanje [3]. S vjerojatnošću $(1 - q_0)$ mrav istražuje bridove susjednih neposjećenih čvorova. Pseudokod na slici (1.2) prikazuje moguću implementaciju istraživanja bridova.



Slika 1.2 Pseudokod istraživanja bridova

Mijenjanje parametra q_0 dopušta promjenu stupnja istraživanja susjednih bridova u odnosu na odabir čvora na temelju stečenog znanja. Stečeno znanje konvergira lokalnoj okolini najboljeg rješenja.

U algoritmu Ant Colony System globalno ažuriranje feromonskih tragova, nakon svakog ciklusa, izvodi mrav s najboljim rješenjem prema izrazu (1.6).

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{bs}, \qquad \forall (i,j) \in T^{bs} \tag{1.6}$$

Izraz $\Delta\tau_{ij}^{bs}$ jednak je, ukoliko optimum predstavlja minimum, recipročnoj vrijednosti rješenja najboljeg mrava, $\frac{1}{C^{bs}}$. Važno je uočiti da se feromonski tragovi ažuriraju duž bridova kojim je prošao najbolji mrav, a ne duž svih bridova kao u algoritmu Ant System. Ta činjenica važna je iz razloga što se složenost izračunavanja reducirala s $O(n^2)$ na $O(n)$, gdje je n broj čvorova [3].

Lokalno ažuriranje feromonskih tragova u algoritmu Ant Colony System obavljaju mravi odmah nakon što su prošli bridom (i,j), a ažuriranje je određeno izrazom (1.7).

$$\tau_{ij} = (1-\vartheta) \cdot \tau_{ij} + \vartheta \cdot \tau_0 \tag{1.7}$$

Vrijednost ϑ parametra, koji označava lokalnu brzinu isparavanja, je iz intervala $\langle 0,1 \rangle$, a τ_0 je inicijalna vrijednost feromonskog traga na grafu G. Pokazuje se da je poželjna vrijednost parametra ϑ iz intervala $[0.1,0.5]$, a inicijalna vrijednost od τ_0 postavlja se iskustveno ili se koristi izraz (1.8).

$$\tau_0 = \frac{m}{C^{nn}} \tag{1.8}$$

Vrijednost C^{nn} najkraća je duljina puta pronađena nekim jednostavnim algoritmom, npr. algoritam najbližeg susjeda ili min-min algoritam, a vrijednost m može biti jednaka jedan ili broju mrava.

Ideja je dobiti određenu procjenu duljine puta od koje će mravi dalje tražiti bolja rješenja te ponuditi optimalnu početnu točku za rad algoritma. Uz premalu vrijednost τ_0, pretraga će brzo biti usmjerena prema području koje su mravi slučajno odabrali u prvoj iteraciji, a uz preveliki τ_0, količine feromona koje mravi ostavljaju u svakoj iteraciji bit će premale da bi mogle usmjeravati pretragu, pa će se morati potrošiti puno ciklusa kako bi mehanizam isparavanja odnio višak feromona.

Slikom (1.3) prikazan je pseudokod algoritma Ant Colony System.

```
ponavljaj dok nije kraj
        ponovi za svakog mrava
                stvori rješenje
                vrednuj rješenje
        kraj ponovi
        ispari feromonske tragove
        pronađi mrava s najboljim rješenjem
        najbolji mrav vrši ažuriranje feromonskog traga na ruti kojom je prošao
kraj ponavljanja
```

Slika 1.3 Pseudokod algoritma Ant Colony System

1.5. Vrste mravljih algoritama

Algoritam kolonije mrava općenit je naziv za skup algoritama koji se temelje na jednostavnom mravljem algoritmu (§ 1.3). Motiv za daljnju nadogradnju jednostavnog algoritma jest bolja pretvorba $NP - teških$ i $NP - potpunih$ problema u P probleme. Prvi algoritam kolonije mrava je *Ant System*. Predložene su tri inačice algoritma [3]:

1. algoritam temeljen na gustoći mrava (engl. *ant-density*),
2. algoritam temeljen na brojnosti mrava (engl. *ant-quantity*),
3. algoritam temeljen na ciklusima mrava (engl. *ant-cycle*).

Prve dvije inačice odbačene su zbog loših svojstava, a treća inačica evoluirala je u niz drugih algoritama.

Neposredni slijednici algoritma Ant System su:

- elitistički algoritam (engl. *Elitist Ant System*),
- algoritam rangirajućeg sustava mrava (engl. *Rank-based Ant System*),
- *Max-Min* algoritam (engl. *Max-Min Ant System*).

Glavne modifikacije u odnosu na Ant System su pravila ažuriranja feromonskih tragova i dodatna ograničenja, koja uvodi *Max-Min* algoritam.

Uključivanje novih temeljnih mehanizama, algoritam *Ant System* proširuje se u sljedeće algoritme [3]:

- *Ant Colony System*,
- ANTS (engl. *Approximate Nondeterministic Tree Search*),
- radna okolina hiperkocke za algoritam kolonije mrava,
- paralelna implementacija algoritma kolonije mrava.

2. Problem raspoređivanja u okružju nesrodnih strojeva

U ovom poglavlju prikazani su elementi koji čine problem raspoređivanja i neki pristupi rješavanju toga problema. Opisan je način zapisa okruženja raspoređivanja, podjela uvjeta raspoređivanja i vrednovanje rasporeda. Definirano je okružje nesrodnih strojeva i heuristika koja rješava zadani problem. Na kraju poglavlja opisana je moguća primjena izrade rasporeda u okružju nesrodnih strojeva.

2.1. Postupak raspoređivanja

Raspoređivanje je u širem smislu postupak izrade bilo kakvog rasporeda. U različitim okruženjima postupak izrade rasporeda može se podijeliti na više stupnjeva, ovisno o algoritmu i problemu. U problemima koji uključuju samo jedno sredstvo, raspoređivanje se sastoji od određivanja redoslijeda aktivnosti na tom sredstvu. Ako se koristi više odvojenih sredstava, prije izrade redoslijeda potrebno je odrediti koje aktivnosti će se odvijati na kojem sredstvu.

U oba slučaja, postupak izrade cjelokupnog rasporeda naziva se raspoređivanjem. U problemima sa više sredstava, postupak dodjeljivanja aktivnosti sredstvima naziva se pridruživanjem (engl. *matching*), a postupak izrade redoslijeda na pojedinom sredstvu uređivanjem (engl. *sequencing*) [4].

Raspoređivanje je proces u kojem se dodjeljuju ograničena sredstva (engl. *resources*) skupu aktivnosti koje se trebaju izvršiti, ovisno o danom okruženju raspoređivanja i ograničenjima. Aktivnost je bilo kakva operacija koja se treba izvršiti u određenom vremenu, na određenom sredstvu.

Složenost postupka raspoređivanja mijenja se ovisno o okolini u kojoj se vrši raspoređivanje. Povećanjem složenosti okoline, raste i složenost raspoređivanja poslova u toj okolini. Shodno povećanju složenosti problema raspoređivanja, raste potreba za algoritmima koji će u što kraćem roku rezultirati prihvatljivim rješenjima problema. Za većinu problema koji se pojavljuju, budući da je nemoguće razviti algoritam koji bi bio egzaktan, razvijaju se heuristički algoritmi [4].

Takvi algoritmi koriste metode pretraživanja prostora stanja, a metode je moguće primijeniti u statičkim i dinamičkim okružjima raspoređivanja. Evolucijsko

računanje, na kojem je težište ovog rada, a koje je opisano u (§ 1.), jedno je od mogućih postupaka rješavanja problema raspoređivanja.

Važan element raspoređivanja je posao (engl. *job*), koji se može sastojati od više aktivnosti, ali najčešće je riječ o jednoj nedjeljivoj aktivnosti. Cilj postupka raspoređivanja je omogućiti izvođenje posla, te su svojstva poslova varijable samog algoritma raspoređivanja. Skup svih poslova označava se s J, dok se pojedini posao označava s J_j. Skup zadataka (engl. *task*), ili aktivnosti, označava se s T, dok se pojedini zadatak označava s T_j. U nastavku su navedena relevantna svojstva poslova za sam postupak raspoređivanja [4].

Trajanje izvođenja (p_j)

Svaki posao izvršava se u određenom vremenu, te se ta količina vremena naziva trajanje izvođenja (engl. *processing time*). Označava se oznakom p_j, pri čemu je j indeks posla. U okruženju više strojeva, trajanje izvođenja posla J_j na stroju i označava se oznakom p_{ij}.

Vrijeme pripravnosti (r_j)

Trenutak u kojem posao postaje raspoloživ za izvođenje naziva se vrijeme pripravnsti (engl. *ready time, release time*) i označava se oznakom r_j. Prije vremena pripravnosti nije moguće izvođenje posla u sustavu.

Vrijeme željenog završetka (d_j)

Trenutak do kojeg bi posao trebao završiti s izvođenjem naziva se vrijeme željenog završetka (engl. *due date*) i označava se s d_j. U slučaju da se izvođenje posla završi nakon tog trenutka, dolazi do stvaranja određenog troška i ovo ograničenje vrlo je važno u stvaranju kvalitetnog rasporeda.

Vrijeme nužnog završetka ($\tilde{d}_j$)

Vrijeme nužnog završetka (engl. *deadline, drop dead time*), koje se označava oznakom $\tilde{d}_j$, je vremenski trenutak do kojeg posao mora završiti kako bi raspored bio valjan.

Težina (w_j)

Vrijednost koja određuje prioritet posla u sustavu naziva se težina (engl. *weight*), te se označava oznakom w_j. Ta se vrijednost najčešće koristi kod vrednovanja

rasporeda, gdje predstavlja mjeru kvalitete rasporeda. U slučaju da se vrednovanje rasporeda obavlja na temelju više kriterija, moguće je svakom poslu pridružiti više težinskih vrijednosti.

2.2. Zapis okruženja raspoređivanja

Okolina raspoređivanja zapisuje se koristeći standardni zapis $\alpha|\beta|\gamma$, koji su 1979. Godine uveli R. Graham, E. Lawer, J.K. Lenstra i A. Rinooy Kan. Svako polje u zapisu odgovara elementu koji mora biti definiran kako bi okolina bila odrediva. Ti elementi svojstva su strojeva, svojstva zadataka i kriterij vrednovanja rasporeda. Obzirom na pojavu sve složenijih okružja, kontinuirano se uvode nove oznake i time proširuje navedeni zapis.

2.2.1. Okolina strojeva

Prvo polje u zapisu, α, određuje okolinu strojeva na kojima se poslovi izvode. Ovdje će biti navedene najčešće okoline strojeva [4].

Jedan stroj (1)

U sustavu postoji samo jedan stroj (engl. *single machine*) i na njemu se izvode svi poslovi. Ovakvo okruženje može se promatrati kao specijalni slučaj svih ostalih složenijih okolina.

Paralelni identični strojevi (Pm)

U sustavu postoji m identičnih strojeva koji rade paralelno (engl. *parallel identical machines*). Svaki se posao sastoji od jedne operacije i može se izvesti na bilo kojem stroju. Brzina obrade, budući da su strojevi identični, jednaka je za svaki stroj, pa za vrijeme izvođenja nije potrebno navoditi njegovu oznaku. Ako broj strojeva m nije naveden, smatra se da je proizvoljan, što vrijedi i za sljedeća navedena okruženja.

Jednoliki strojevi (Qm)

U ovakvom okruženju svaki stroj ima različitu brzinu, no ona je jednaka za svaki zadatak koji se na njemu obavlja. Brzina jednolikog stroja i (engl. *uniform machines*) označava se oznakom s_i. Time trajanje obavljanja posla j na stroju i, koje se označava p_{ij}, dobiva vrijednost $\frac{p_j}{s_i}$.

Nesrodni strojevi (Rm)

U okruženju nesrodnih strojeva (engl. *unrelated machines*) svaki stroj ima različitu brzinu za svaki pojedini posao u sustavu. Trajanje izvođenja posla definirano je za svaku kombinaciju posla i stroja, te se označava p_{ij}, kao i u prethodnom slučaju.

Otvorena obrada (Om)

U okruženju otvorene obrade (engl. *open shop*) svaki posao J_j sastoji se od više nedjeljivih operacija, pri čemu se broj operacija posla označava s n_j, a pojedina operacija s T_{nj}. Svaka opercija može se obaviti jedino na za nju predviđenom stroju, te je broj operacija jednak broju strojeva.

Obrada tijeka (Fm)

Za razliku od prethodno navedene okoline, o okolini obrade tijeka (engl. *flow shop*) postoji još jedno ograničenje. Naime, za sve poslove utvrđen je jednak redoslijed obavljanja operacija. Tako se operacija T_{1j} obavlja na stroju $P1$, zatim operacija T_{2j} na stroju $P2$ itd.

Proizvoljna obrada (Jm)

U okruženju proizvoljne obrade (engl. *job shop*) moguće je da se operacije izvode više puta na jednom stroju, a niti jednom na nekom drugom stroju. Također se moraju izvoditi određenim redoslijedom i na određenim strojevima, kao i u prethodno navedenom okružju.

2.2.2. Uvjeti raspoređivanja

Uvjeti raspoređivanja odnose se na različite mogućnosti izrade rasporeda te na vremenski odnos između same izrade rasporeda i njegovog izvođenja u nekom stvarnom sustavu. Uvjeti raspoređivanja mogu se podijeliti po nekoliko neovisnih osnova koje su opisane u nastavku [4].

Raspoloživost parametara

Budući da svi algoritmi raspoređivanja koriste neke od parametara sustava (npr. broj poslova, vrijeme dolaska itd.) kao ulazne veličine, najvažniji čimbenik u odabiru algoritma raspoređivanja je raspoloživost parametara sustava. Po pitanju raspoloživosti parametara, sustavi raspoređivanja dijele se na dvije skupine:

- *Predodređeno raspoređivanje* (engl. *offline scheduling*) – u ovom obliku raspoređivaja pretpostavlja se da su sve potrebne vrijednosti poznate prije izrade rasporeda i prije samog rada sustava. Isto tako, u svakom trenutku rada sustava poznate su sve ulazne veličine koje opisuju budućnost sustava (kao što su dolasci poslova),
- *Raspoređivanje na zahtjev* (engl. *online scheduling*) – za razliku od prethodnog modela, u raspoređivanju na zahtjev odluke se donose na temelju trenutno dostupnih podataka, bez znanja o eventualnoj budućnosti sustava. Primjerice, značajke poslova poznate su tek kad neki posao dođe u sustav, odnosno, kada postane raspoloživ, a ne prije početka izrade rasporeda.

Pouzdanost parametara

Neovisno o tome jesu li vrijednosti parametara poznate ili ne prije izrade rasporeda, okolina raspoređivanja može se razlikovati i po tome koliko su vrijednosti tih parametara pouzdane, odnosno kojom preciznošću su procijenjene. Po tom kriteriju raspoređivanje se može grubo podijeliti u dvije skupine:

- *Deterministčko raspoređivanje* – u deterministčkom raspoređivanju smatra se da su vrijednosti parametara sustava određene s dovoljno velikom preciznošću, bez obzira u kojem trenutku postaju poznate,
- *Stohastičko raspoređivanje* – unutar stohastičkog raspoređivanja ne postoje pouzdane procjene vrijednosti parametara sustava, već se stvarne vrijednosti mogu očitati tek nakon završetka rada određenog dijela sustava.

2.3. Vrednovanje rasporeda

Vrednovanje rasporeda odnosi se na njegovo ocjenjivanje prema nekom od kriterija. Odluka o kojem se kriteriju radi ima važan utjecaj i na izbor algoritma raspoređivanja koji će se koristiti. Mjerila vrednovanja, tj. kriteriji definiraju se u ovisnosti o izlaznim veličinama u sustavu, koje će sad biti navedene [4].

- *Vrijeme završetka* C_j (engl. *completion time*) je trenutak u kojem se završava izvođenje aktivnosti j,
- *Protjecanje* F_j (engl. *flowtime*) je količina vremena koju je u sustavu provela aktivnost j:

$$F_j = C_j - r_j \tag{2.1}$$

- *Kašnjenje* L_j (engl. *lateness*) je razlika između vremena završetka i vremena željenog završetka aktivnosti j:

$$L_j = C_j - d_j \tag{2.2}$$

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

$$T_j = \max\{0, L_j\} \tag{2.3}$$

- *Preuranjenost* E_j (engl. *earliness*) je negativan iznos kašnjenja aktivnosti j, odnosno jednako je nuli ako je kašnjenje pozitivno:

$$E_j = \max\{0, -L_j\} \tag{2.4}$$

- *Zakašnjelost* U_j govori je li aktivnost j prekoračila željeno vrijeme završetka ili nije:

$$U_j = \begin{cases} 1: T_j > 0 \\ 0: T_j \leq 0 \end{cases} \tag{2.5}$$

Na osnovu ovih veličina navedeni su najčešće korištenih kriterija vrednovanja rasporeda. U slučaju da je težina svakog zadatka jednaka, vrijednost težine ne utječe na ocjenu rasporeda te je samim time postupak raspoređivanja olakšan, no to najčešće nije slučaj kod stvarnih uvjeta raspoređivanja.

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

$$C_{max} = \max\{C_j\} \tag{2.6}$$

- *Najveće kašnjenje* L_{max} (engl. *maximum lateness*) definirano je kao:

$$L_{max} = \max\{L_j\} \tag{2.7}$$

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

$$F_w = \sum_j w_j F_j \tag{2.8}$$

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

$$T_w = \sum_j w_j T_j \tag{2.9}$$

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

$$U_w = \sum_j w_j U_j \tag{2.10}$$

- *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:

$$ET_w = \sum_j (w_{Ej}E_j + w_{Tj}T_j) \tag{2.11}$$

U posljednjem navedenom kriteriju uključen je trošak preuranjeno završenog posla, što nije slučaj kod tzv. pravilnih kriterija, te je kriterij primjer tzv. nepravilnog mjerila.

2.4. Okružje nesrodnih strojeva

Okružje nesrodnih strojeva definirano je u (§ 2.2.1.), a u ovom potpoglavlju bit će definirane pretpostavke koje vrijede za okružje nesrodnih strojeva i postupci raspoređivanja na nesrodnim strojevima.

2.4.1. Pretpostavke i postupci raspoređivanja za okružje nesrodnih strojeva

U okružju nesrodnih strojeva raspoređivanje može biti predodređeno ili na zahtjev. U većini stvarnih sustava, izvodi se raspoređivanje na zahtjev (§ 2.2.2.). Pretpostavke koje u ovom radu vrijede za okružje nesrodnih strojeva [4]:

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

Prilikom raspoređivanja na nesrodnim strojevima, dolaze do izražaja dvije faze raspoređivanja poslova: pridruživanje i uređivanje (§ 2.1.). Neki postupci raspoređivanja ove dvije faze obavljaju odvojeno, a kod nekih je jedna sadržana u drugoj. Također, dvije je 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 njegovu dolasku, čime se zapravo obavlja samo pridruživanje poslova. Uređivanje poslova na stroju podrazumijeva redoslijed izvođenja jednak redu prispjeća, tj. uređivanje po *FCFS* (engl. *first come first served*). Druga skupina algoritama ne raspoređuje svaki posao po njegovu

dolasku, već se u predodređenim vremenskim intervalima zajednički raspoređuju svi do tada pristigli, a još nepokrenuti poslovi (engl. *batch mode*) [4].

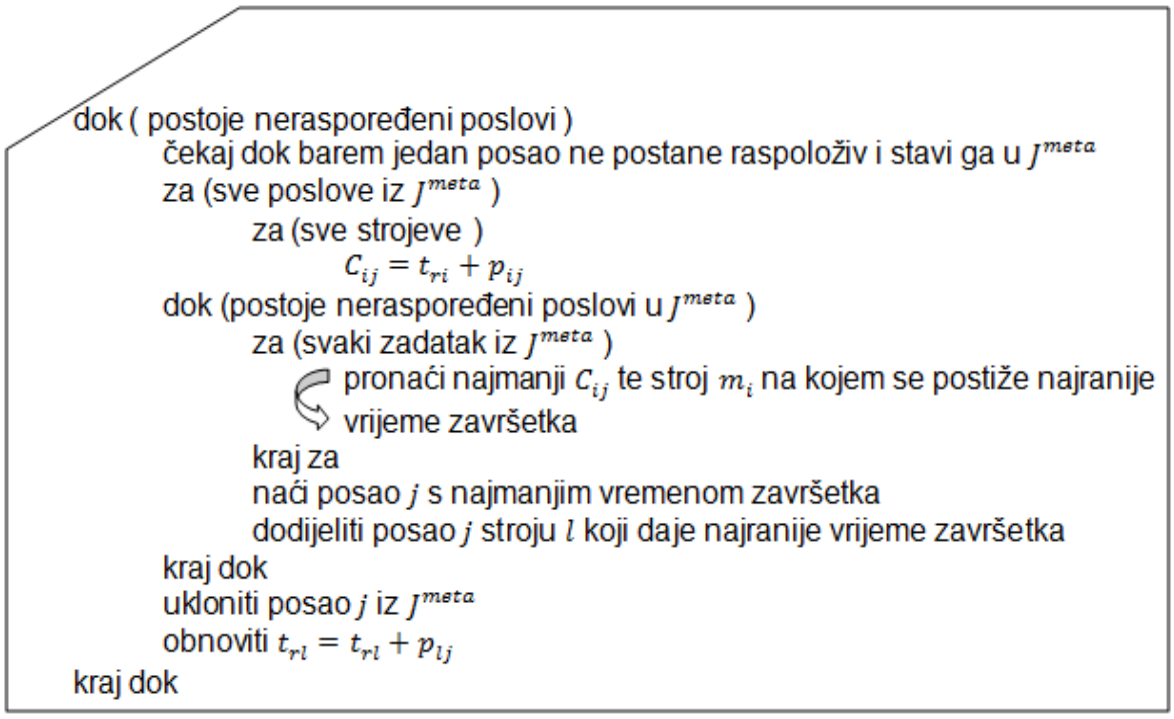
U većini primjena algoritmi iz druge skupine postižu bolje rezultate. Za okruženje nesrodnih strojeva postoji niz heuristika raspoređivanja, no jedna od najpoznatijih metoda je min-min heuristika.

2.4.2. Min-min algoritam

Prije opisa algoritma, potrebno je uvesti sljedeće oznake [4]:

- p_{ij} je trajanje izvođenja posla j na stroju m_i; sva trajanja izvođenja unaprijed su poznata s dovoljnom preciznošću,
- t_{ri} je vrijeme pripravnosti stroja m_i (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 za 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 jošp nije započelo pa su podložni novom pridruživanju.

U nastavku je prikazan i opisan algoritam min-min.

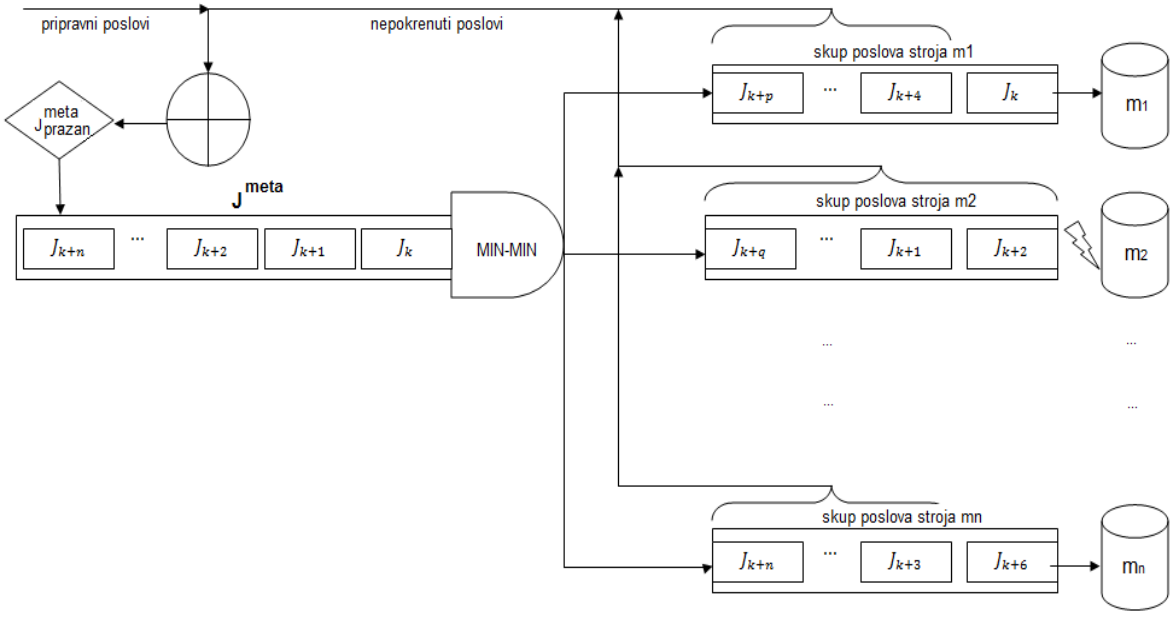


Slika 2.1 Min-min postupak raspoređivanja [4]

Min-min postupak prikazan je kao algoritam slikom 2.1. 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.

Važan element uporabe opisanog postupka raspoređivanja je i odabir trenutka raspoređivanja nagomilanih poslova. U stvarnim se uvjetima obično definira neki vremenski interval nakon kojeg se algoritam ponovno primjenjuje. Određivanje veličine intervala u velikoj je mjeri ovisno o vrsti sustava u kojemu se raspoređivanje obavlja. U ovom radu, u postupku simulacije, primjenjuje se najbolji slučaj: algoritam se izvodi svaki puta kada novi posao dođe u sustav.

Na slici 2.2 grafički je prikazano raspoređivanje i uređivanje pomoću min-min algoritma. Pošto je stroj m_2 zauzet, a novi posao dolazi u sustav, niti jedan posao iz skupa poslova stroja m_2 neće biti izvršen, već se ponovno vraćaju u skup J^{meta}.



Slika 2.2 Grafički prikaz algoritma min-min

3. Rješavanje problema raspoređivanja u okružju nesrodnih strojeva uporabom algoritma kolonije mrava

U ovom poglavlju prikazana je grafička interpretacija i prikaz rješenja. Predstavljen je pseudokod funkcije dobrote (engl. *fitness function*) i korištena heuristička pravila u svrhu korištenja parametra η u izrazu (1.5.). Rješenje problema raspoređivanja u okružju nesrodnih strojeva ostvareno je uporabom algoritma Ant Colony System.

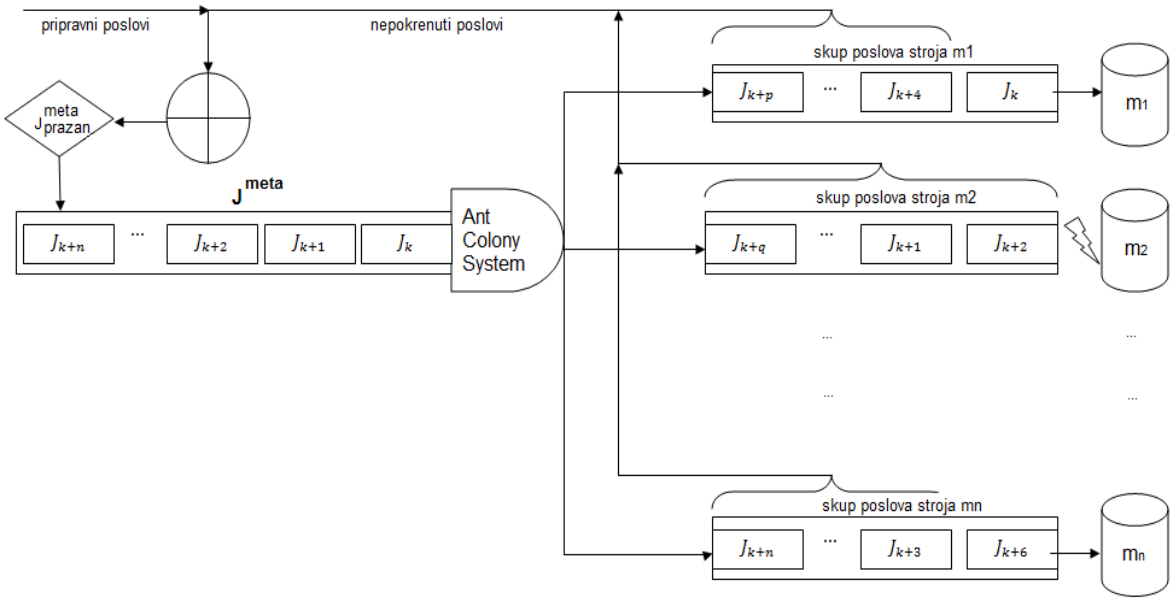
3.1. Uvjeti raspoređivanja i vrednovanje rasporeda

Algoritam Ant Colony System implementiran je za predodređeno raspoređivanje i raspoređivanje na zahtjev (§ 2.2.2.). Svrha implementiranja dva tipa raspoređivanja, po kriteriju uvjeta raspoređivanja, je utvrditi kada algoritam min-min postiže bolje rezultate, a kada algoritam Ant Colony System.

Raspoređivanje na zahtjev implementirano je na isti način kao i min-min postupak raspoređivanja, uz iznimku da pridruživanje i uređivanje (§ 2.1.) radi algoritam Ant Colony System (slika 3.1.).Radi usporedivosti algoritama, i algoritam Ant Colony System implementiran je tako da se izvodi svaki puta kada novi posao dođe u sustav.

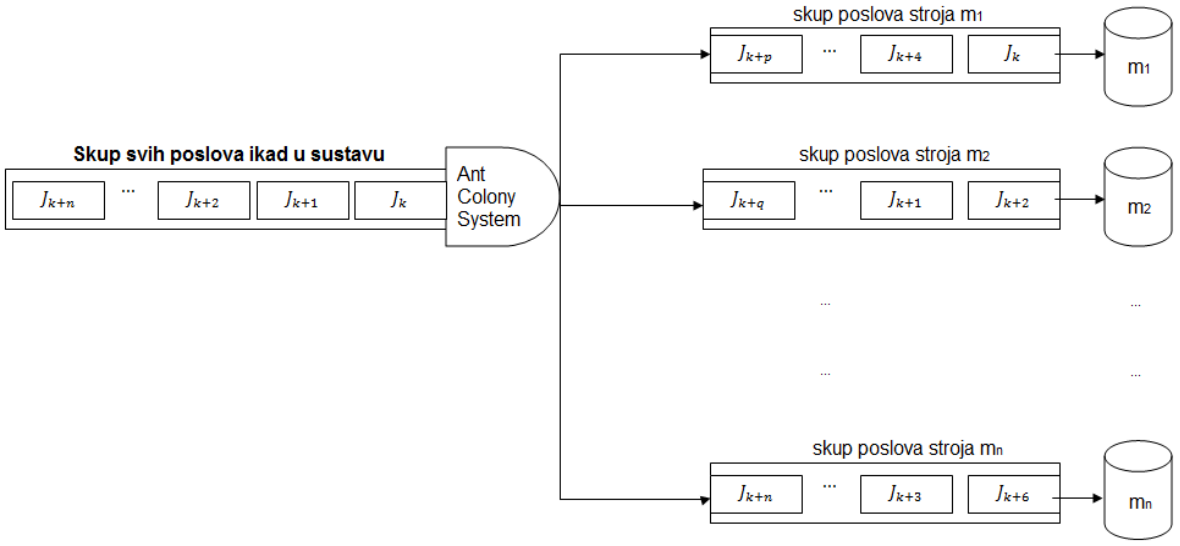
Raspoređivanje na zahtjev ima veću složenost od predodređenog raspoređivanja jer je prije svakog pokretanja algoritma Ant Colony System potrebno ponovno inicijalizirati feromonske tragove na vrijednost τ_0, rasporediti mrave u čvorove poslova koji se nalaze u skupu J^{meta}, a za neraspoložive poslove svakome mravu upisati u njegovu $tabu\ listu$ da ne smije posjetiti čvorove tih poslova.

Prilikom raspoređivanja na zahtjev dovoljan je mali broj ciklusa jer je prosječna maksimalna veličina skupa J^{meta} jednaka $\frac{N_{job}}{2}$, a u određenim trenucima, na početku rada sustava i na kraju rada, nalazi se vrlo malen broj poslova (između 10% i 15%) i velik broj ciklusa vrlo bi usporio izvođenje programa. Unatoč velikom broju ciklusa, osjetno poboljšanje rezultata nije vidljivo jer velik broj mrava i malen broj poslova u sustavu dovodi do brze konvergencije rješenja. U programu je najveći dozvoljen broj ciklusa jednak pet.



Slika 3.1 Raspoređivanje na zahtjev algoritmom Ant Colony System

Predodređeno raspoređivanje algoritam Ant Colony System izvodi po definiciji (§ 2.2.2.). Slikom 3.2. prikazan je način izvođenja predodređenog raspoređivanja.



Slika 3.2 Predodređeno raspoređivanje algoritmom Ant Colony System

Stvoren raspored optimiran je s obzirom na jedan od sljedeća četiri uvjeta raspoređivanja (§ 2.3., izrazi (2.6), (2.8.), (2.9) i (2.10)):

- ukupna duljina rasporeda,
- težinsko protjecanje,
- težinsko zaostajanje,

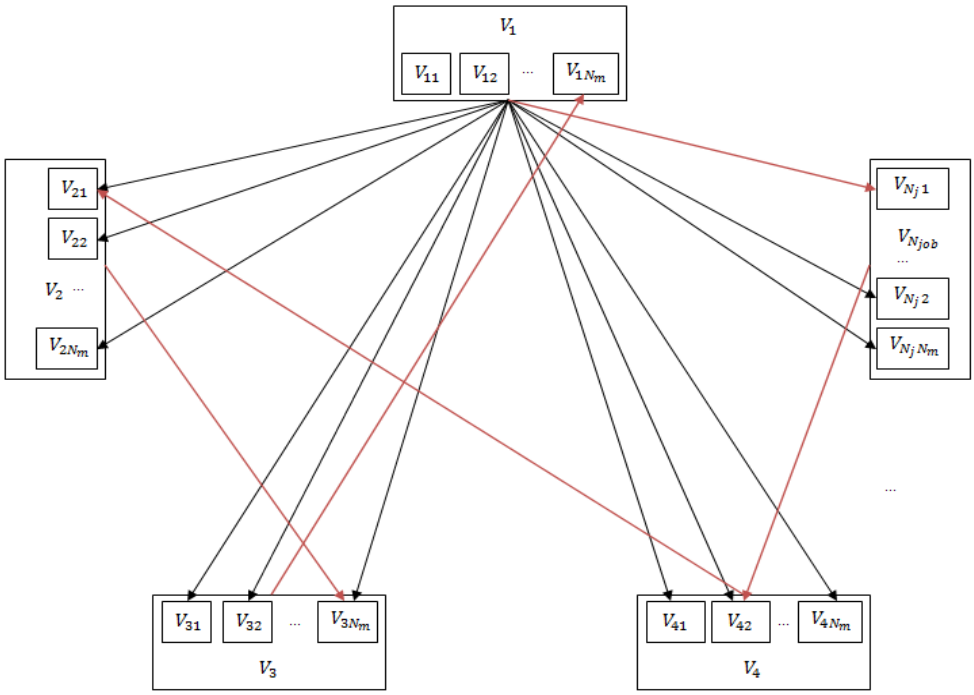
- težinska zakašnjelost.

3.2. Grafička interpretacija

Kao što je opisano u §1.3., rješenje problema može se interpretirati grafom G, gdje je $G = (V, E)$. U rješenju problema raspoređivanja u okružju nesrodnih strojeva, veličina skupa V jednaka je broju poslova, $|V| = N_{job}$. Svaki posao predstavlja jedan čvor, V. Svaki čvor V_i sastoji se od $N_{machines}$ podčvorova, gdje $N_{machines}$ predstavlja broj nesrodnih strojeva u okružju, dakle $\forall V_i, |V_i| = N_{machines}$. Svaki čvor V_i spojen je s $(N_{job} - 1)N_{machines}$ bridova.

Veličine struktura podataka za feromonske tragove i heurističke informacije iznose $N_{job}^2 \cdot N_{machines}$. Upravo iz tog razloga, složenost ažuriranja feromonskih tragova iznosi $O(n^3)$.

Slikom 3.3. prikazana je grafička interpretacija rješenja. Crnim strelicama prikazani su bridovi koji spajaju čvor V_1 s preostalim čvorovima. Crvenim strelicama prikazano je jedno od mogućih rješenja raspoređivanja. U poglavlju 3.3. objašnjeno je predstavljanje rješenja.



Slika 3.3 Grafička interpretacija rješenja algoritma Ant Colony System

Prilikom inicijalizacije, svaki mrav postavi se u jedan podčvor čvora V_i. Potreban broj mrava da bi se zauzeli svi podčvorovi, svih čvorova, jest N_{ant}, gdje je N_{ant} definiran kao:

$$N_{ant} = N_{job} N_{machines} \tag{3.1}$$

U implementaciji nije potrebno koristiti N_{ant} broj mrava jer se povećava vrijeme izvođenja programa. Stoga je korišten izraz (3.2.) za ukupan broj mrava u sustavu.

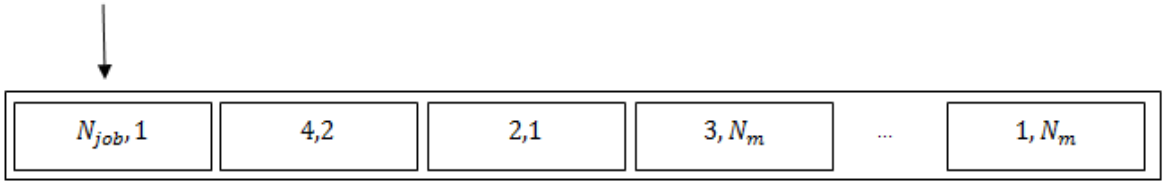
$$N_{ant} = k N_{job} N_{machines} \tag{3.2}$$

Faktor k je iz intervala $\langle 0,1]$, a program je izveden za vrijednosti $\left\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}$.

3.3. Predstavljanje rješenja

Pridruživanje i uređivanje (§ 2.1.) implementirani algoritam Ant Colony System radi istovremeno. Rješenje ili raspored koji algoritam stvara prikazan je u obliku prioritetnog vektora. Prioritetni vektor sastoji se od N_{job} elemenata, a svaki element sastoji se od dvije informacije: rednog broja posla i rednog broja stroja na kojem će se posao izvoditi. Svaki mrav računa vlastiti raspored, koji se nakon obiđenih N_{job} čvorova evaluira pomoću funkcije dobrote.

Nakon što se prioritetni vektor preda funkciji za određivanje dobrote rasporeda, iz prioritetnog vektora se jedan po jedan posao, s početka prema kraju vektora, ekstrahira i dodaje u skup poslova stroja m_k (slike 3.1., 3.2.). Crvenim strelicama na slici 3.3. prikazano je jedno moguće rješenje problema. Slikom 3.4. prikazan je prioritetni vektor koji prikazuje rješenje sa slike 3.3.



Slika 3.4 Prikaz rasporeda u obliku prioritetnog vektora

Strelica pokazuje početak vektora i početak dekodiranja vektora. Nakon dekodiranja poslovi moraju zadržati svojstvo prioriteta, odnosno, posao j koji je ispred posla k, a oboje se izvode na stroju m, nakon pridruživanja moraju ostati u istom poretku.

3.4. Funkcija dobrote

Implementirana funkcija dobrote (engl. *fitness function*) simulira izvođenje poslova u okružju nesrodnih strojeva prema stvorenom rasporedu. Prilikom simulacije, funkcija bilježi završetak izvođenja posla j na stroju m i tu vrijednost pohranjuje u strukturu podataka. Završetak izvođenja posla, C_j, najvažniji je parametar jer ostale mjerene vrijednosti (§ 3.1.) direktno ovise o njoj. Slika 3.5. prikazuje algoritam implementirane funkcije dobrote.

```
funkcija fitness( prioritetni vektor )
        za (svaki posao i)
                prioritet π_i=ekstrahiraj iz prioritetnog vektora
        kraj za
        dok (postoje neporcesuirani poslovi)
                aktivniPoslovi = poslovi čiji prethodnik je završio izvođenje
                sortiraj aktivniPoslovi po π_i
                dok (postoje poslovi u AktivniPoslovi i postoje slobodni strojevi za izvođenje)
                        dodijeli posao stroju na koji je raspoređen
                kraj dok
        upiši završetak obrade posla
        kraj dok
kraj funkcija
```

Slika 3.5 Pseudokod algoritma za evaluaciju dobrote rasporeda

Povratna vrijednost funkcije dobrote ovisi o kriteriju optimizacije rasporeda. Implementirane su četiri moguće povratne vrijednosti opisane u § 3.1.

3.5. Heurističke funkcije

U izrazu (1.5.) koristi se heuristika η. U implementaciji algoritma Ant Colony System implementirane su četiri statičke heurističke funkcije. Pojam *statička funkcija* označava da vrijednosti funkcije izračunaju na početku izrade rasporeda i da se tijekom izrade više ne mijenjaju.

Implementirane heurističke funkcije su [4]:

- *EDD* (engl. *earliest due date*) definirano je kao:

$$\eta_j = \frac{1}{d_j} \tag{3.3}$$

- *SPT* (engl. *shortest processing time*) definirano je kao:

$$\eta_{ij} = \frac{1}{p_{ij}} \tag{3.4}$$

- *WSPT* (engl. *weighted shortest processing time*) definirano je kao:

$$\eta_{ij} = \frac{w_j}{p_{ij}} \tag{3.5}$$

- *MON* (*Montagne*) definirano je kao:

$$\eta_{ij} = \frac{w_j}{p_{ij}}\left(1 - \frac{d_j}{\sum_{i=1}^{n} p_{ij}}\right) \tag{3.6}$$

pri čemu je n ukupan broj poslova.

4. Izgled i rad aplikacije

Program za rješavanje raspoređivanja u okružju nesrodnih strojeva implementiran je u programskom jeziku $C++$, u razvojnom okruženju $Microsoft\ Visual\ Studio\ 2010$. Korištena je proceduralna paradigma prilikom programiranja.

Za ispravan rad aplikacija očekuje pet datoteka, od kojih je datoteka *fitness.txt* glavna i u njoj se nalaze podaci o broju skupa poslova za raspoređivanje, broj nesrodnih strojeva za pojedini skup poslova i imena preostalih četiri datoteka. U preostale četiri datoteke pojedinačno moraju za svaki skup poslova biti definirani parametri koje koristi program za izradu rasporeda (§ 2.1.):

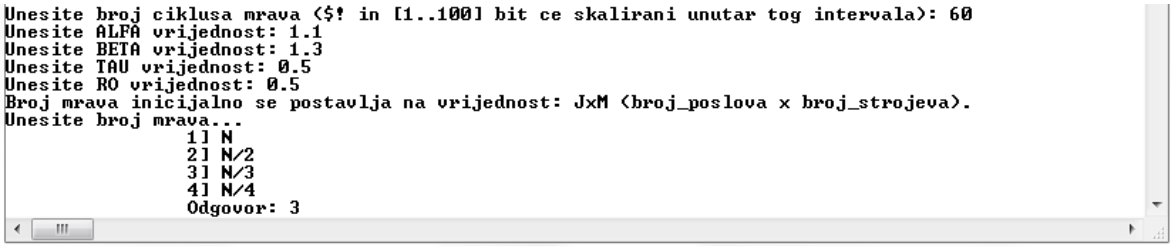
- vremena pripravnosti,
- trajanje izvođenja posla na strojevima,
- vrijeme željenog/nužnog završetka,
- težina.

Pri pokretanju aplikacije korisnik odabire uvjet raspoređivanja, kriterij optimizacije i heurističku informaciju za algoritam Ant Colony System. Zatim korisnik odabire interval skupova poslova za koje želi izraditi raspored. Korisniku je ponuđena opcija da se, nakon upisa gornje i donje granice intervala skupa poslova, upišu predodređene vrijednosti za algoritam Ant Colony System ili da ih sam korisnik upisuje. Slikom 4.1. prikazan je ulazni dio aplikacije.

```
FAKULTET ELEKTROTEHNIKE I RACUNARSTVA   !!      SVEUCILISTE U ZAGREBU   !! ZEMRIS
Student:        Karlo Knezevic, 0036443568, Racunarska znanost
Mentor: doc.dr.sc. Domagoj Jakobovic, ZEMRIS
_____________________________________________________________________________

Odaberite jednu od ponudjenih mogucnosti...

                1] Predodredeno rasporedivanje
                2] Rasporedivanje na zahtjev
                Odgovor: 1

Odaberite kriterij optimizacije...

                1] Cmax, duljina rasporeda
                2] Fw, tezinsko protjecanje
                3] Tw, tezinsko zaostajanje
                4] Uw, tezinska zakasnjelost
                Odgovor: 4

Odaberite tip heuristike...

                1] SPT, 1/p
                2] WSPT, w/p
                3] EDD, 1/d
                4] MON, w/p*(1-d/sum(p))
                Odgovor: 4


Unesite pocetni indeks skupa poslova za analizu (prvi je 0): 0
Unesite zavrsni indeks skupa poslova za analizu (posljednji je 120): 120
Zelite li pretpostavljene vrijednosti u programu ili zelite upisivati vrijednosti [1-da/0-ne]?
```

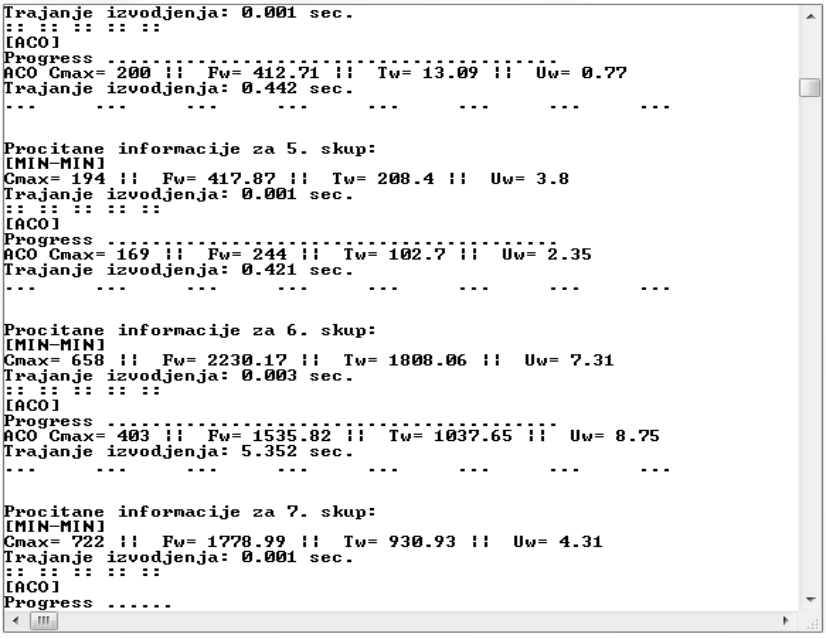
Slika 4.1 Ulazni dio aplikacije za raspoređivanje poslova na nesrodnim strojevima

Ukoliko korisnik odabere vlastito upisivanje vrijednosti, od korisnika se očekuje upis parametara koji zahtjeva formula (1.5.) i broj ciklusa mrava (slika 4.2.).



Slika 4.2 Korisnikov upis vrijednosti parametara u slučaju nepretpostavljenog odabira

Nakon što se upišu svi ulazni podaci, program započinje s radom. Kao što je napomenuto u (§ 1), cilj programa je usporediti dobrotu algoritma Ant Colony System u odnosu na heuristički algoritam min-min. Nakon obrade svakog rasporeda, za svaki skup, u aplikacijski prozor upisuju se mjerene vrijednosti u programu (§ 3.1.). Osim ispisa u aplikacijski prozor, podaci se ispisuju u tri izlazne datoteke. Osim podataka o parametrima rasporeda i vremenu izrade rasporeda, u treću datoteku upisuje se dominacija algoritma Ant Colony System nad algoritmom min-min. Postotak dominacije je mjera koja daje udio ispitnih primjera u kojima određeni algoritam postiže rezultat bolji od rezultata svih testiranih algoritama ili nije lošiji od rezultata bilo kojeg drugog algoritma. Slika 4.3. prikazuje izgled aplikacijskog prozora tijekom izvođenja programa.



Slika 4.3 Aplikacijski prozor tijekom izvođenja programa

5. Rezultati mjerenja

Pretpostavljene vrijednosti parametara iz formule (1.5.), vrijednosti broja ciklusa i broja mrava prikazane su tablicom 5.1.

Tablica 5.1 Pretpostavljene vrijednosti parametara u programu

α	β	ρ	τ_0	$broj\ ciklusa$	N_{ant}
1.1	1.3	0.5	0.5	40	$\frac{1}{2}N_{job}N_{machines}$

Mjerenja su provedena za predodređeno raspoređivanje i raspoređivanje na zahtjev. Pritom su testirane četiri karakteristike za algoritam Ant Colony System:

- veličina kolonije (broj mrava),
- broj ciklusa,
- tip heuristike (§ 3.5.),
- vrijeme izračunavanja rasporeda.

Ispitivanja su provedena na procesorima $QuadCore\ AMD\ Athlon\ II, 2800\ MHz$ i $AMD\ Thurion\ 64, 2100\ MHz$. Prosječno vrijeme izvođenja rasporeda za 120 skupova poslova od $\{12{,}25{,}50{,}100\}$ poslova raspoređivanih na $\{3{,}6{,}10\}$ nesrodna stroja iznosi 2.5 sati.

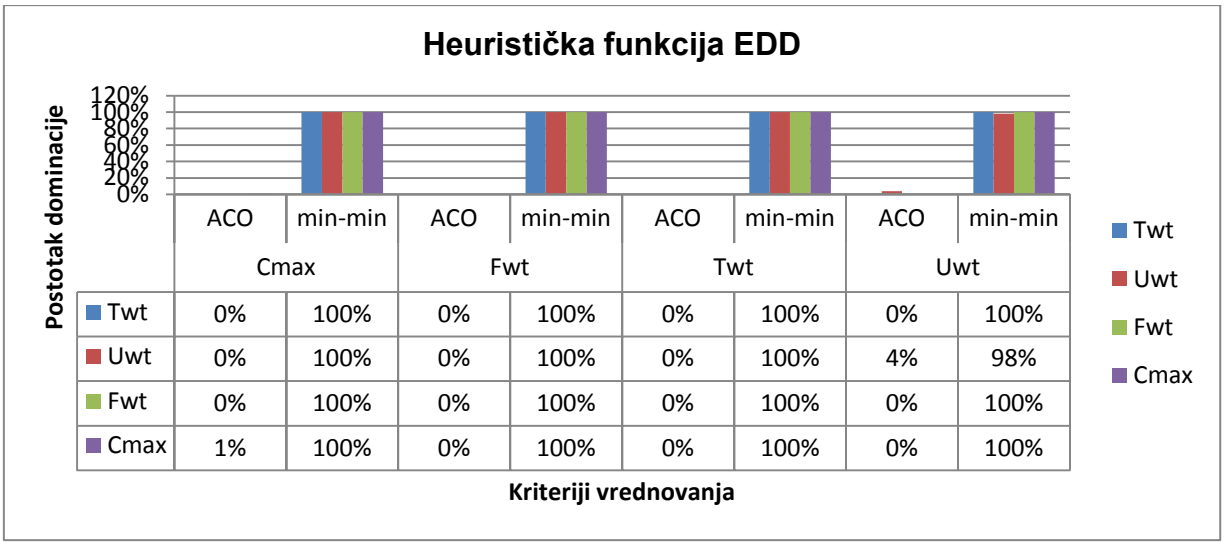
Za svaku od četiri karakteristike provedeno je optimiranje po svim kriterijima navedenim u §3.1. Testiranja su provedena na 120 skupova poslova od po $12, 25, 50$ i 100 poslova.

5.1. Predodređeno raspoređivanje

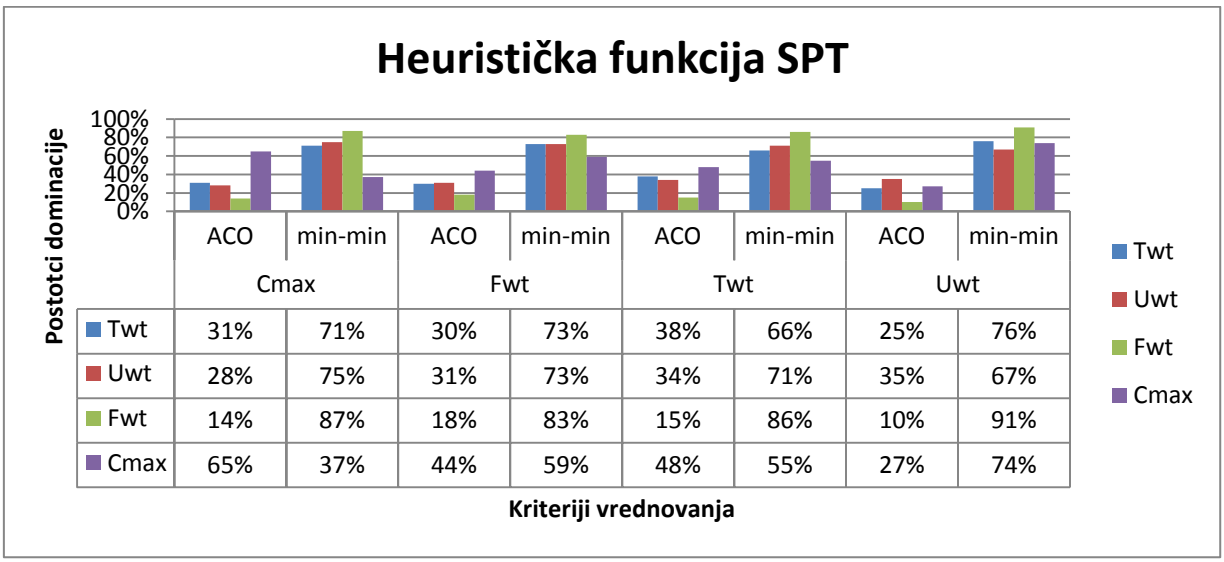
Prilikom testiranja u uvjetima predodređenog raspoređivanja, htjela se, između ostalog, otkriti jedna od četiri optimalne heuristike (§ 3.5.). Ta heuristika koriti se u svim daljnjim testiranjima, kao i u raspoređivanju na zahtjev.

5.1.1. Utjecaj tipa heurističke funkcije

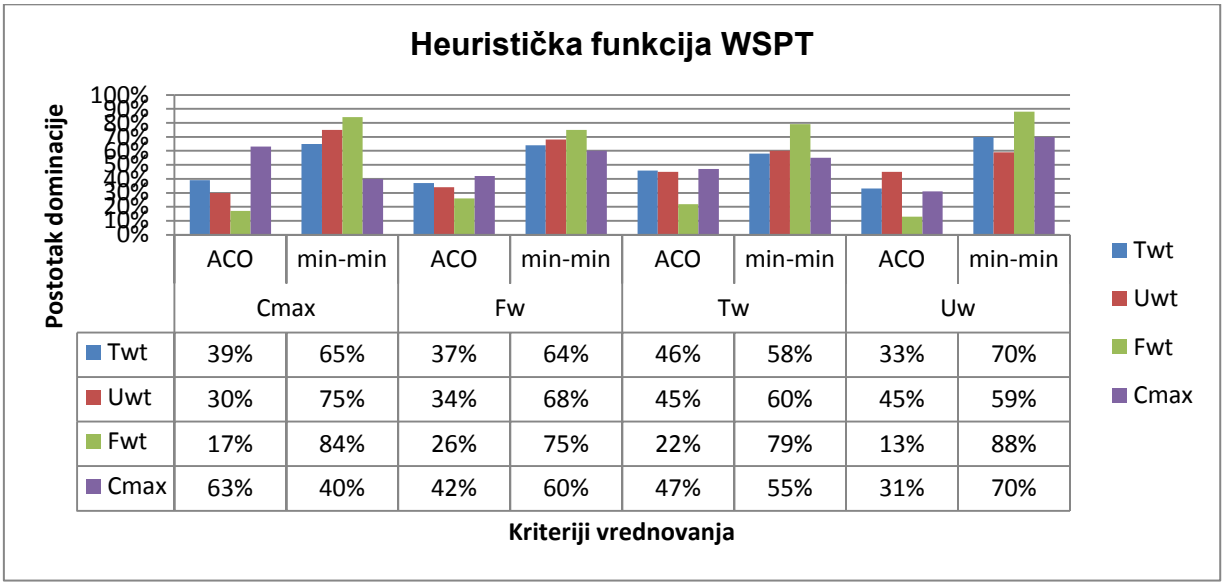
Slike 5.1, 5.2, 5.3 i 5.4 pokazuju postotak dominacije za sve navedene kriterije iz §3.1.



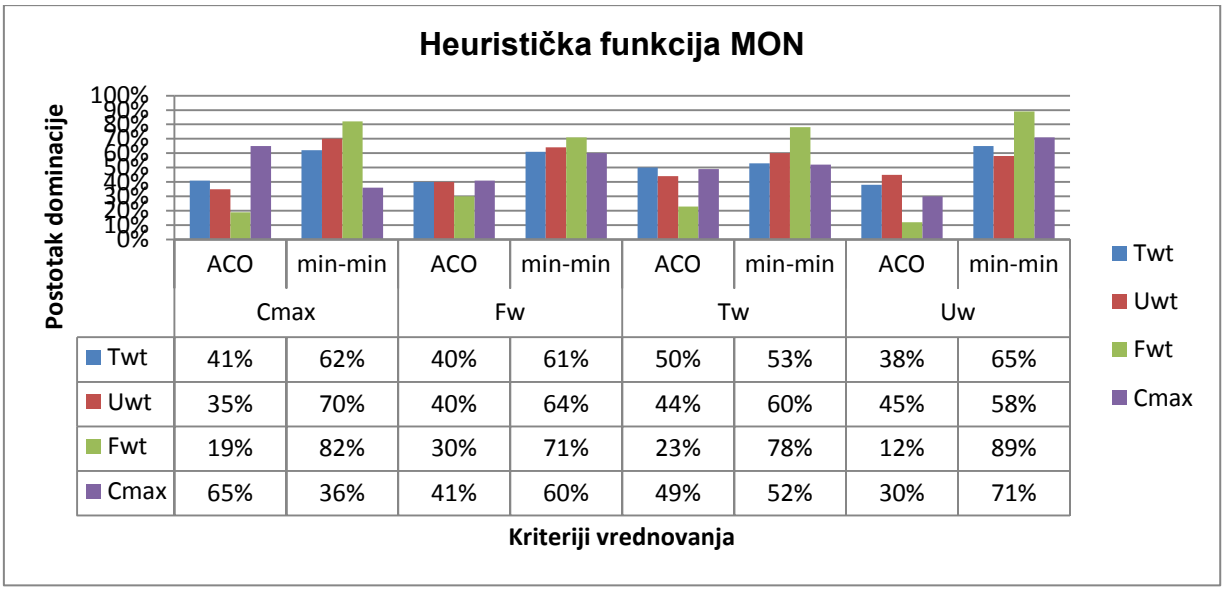
Slika 5.1 Postotak dominacije za heurističku funkciju EDD



Slika 5.2 Postotak dominacije za heurističku funkciju SPT



Slika 5.3 Postotak dominacije za heurističku funkciju WSPT

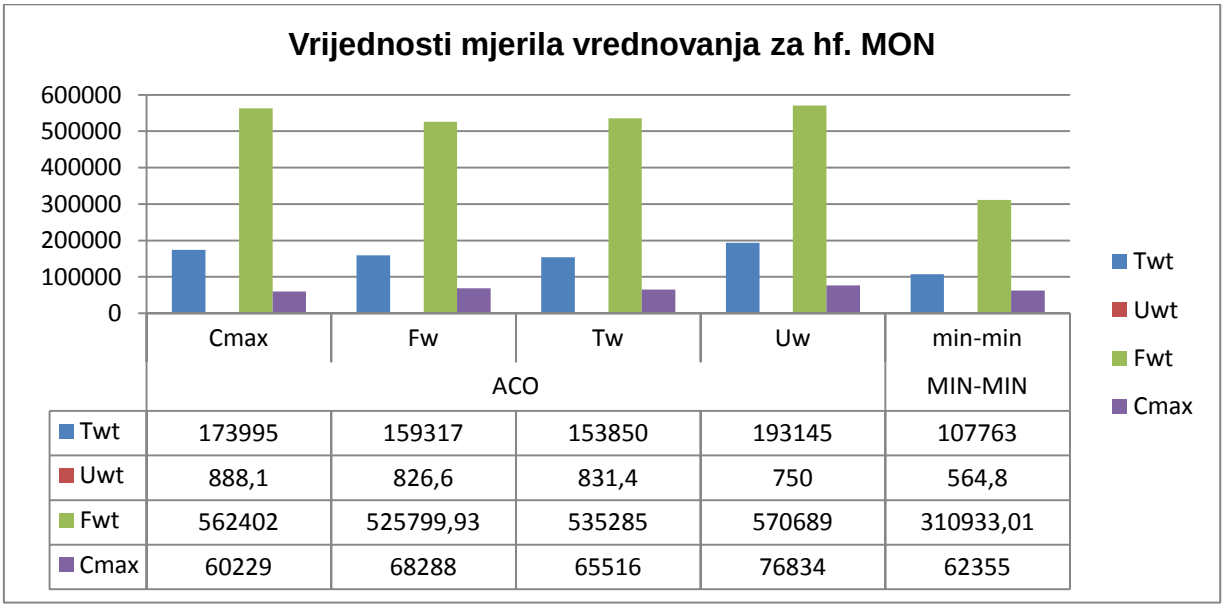


	Cmax ACO	Cmax min-min	Fw ACO	Fw min-min	Tw ACO	Tw min-min	Uw ACO	Uw min-min
Twt	41%	62%	40%	61%	50%	53%	38%	65%
Uwt	35%	70%	40%	64%	44%	60%	45%	58%
Fwt	19%	82%	30%	71%	23%	78%	12%	89%
Cmax	65%	36%	41%	60%	49%	52%	30%	71%

Slika 5.4 Postotak dominacije za heurističku funkciju MON

Iz rezultata se može uočiti da je učinkovitost MON heurističke funkcije najveća i uglavnom podjednaka učinkovitosti min-min algoritma, a EDD heurističke funkcije najlošija. Funkcija $WSPT$ neznatno postiže bolje rezultate od funkcije SPT, a razlog je što prilikom izrade rasporeda u obzir uzima prioritet posla. Razlog što funkcija EDD postiže tako loše rezultate jest što prilikom izrade rasporeda u obzir uzima samo vrijeme željenog završetka posla, a na temelju toga ne može odrediti na kojem stroju posao postiže optimalno izvršavanje, pa na slučajan način odabire stroj. Iz razloga što funkcija MON postiže najbolje rezultate, koristi se u svim daljnjim ispitivanjima.

Slika 5.5 prikazuje vrijednosti mjerila vrednovanja za funkciju MON.

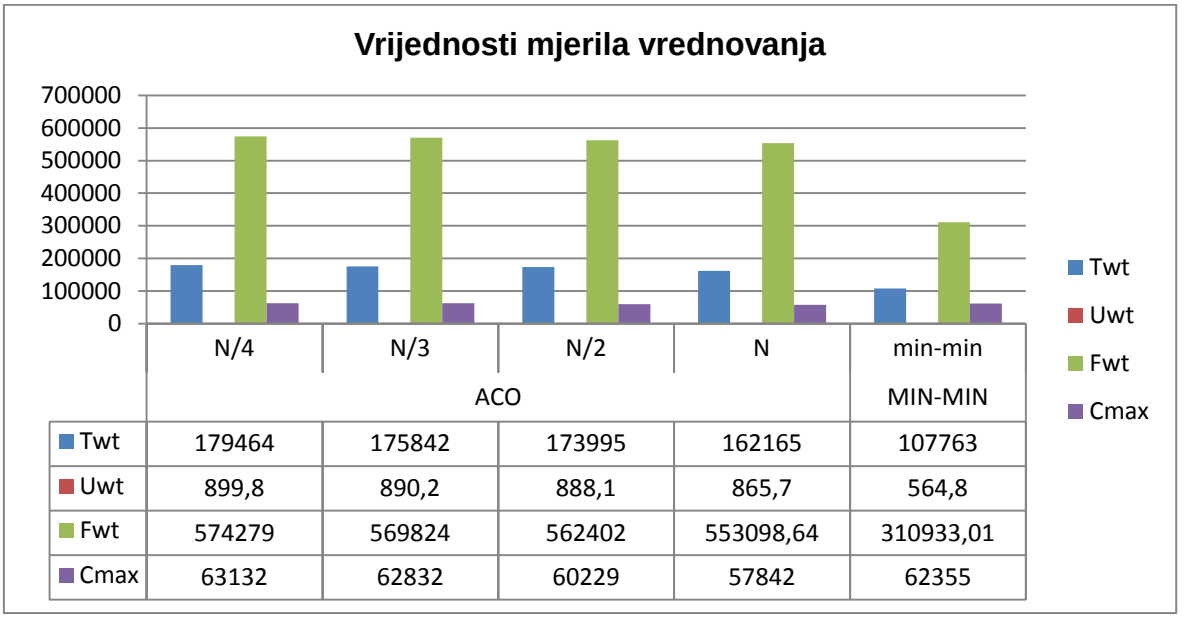


	Cmax ACO	Fw ACO	Tw ACO	Uw ACO	min-min MIN-MIN
Twt	173995	159317	153850	193145	107763
Uwt	888,1	826,6	831,4	750	564,8
Fwt	562402	525799,93	535285	570689	310933,01
Cmax	60229	68288	65516	76834	62355

Slika 5.5 Vrijednosti mjerila vrednovanja za heurističku funkciju *MON*

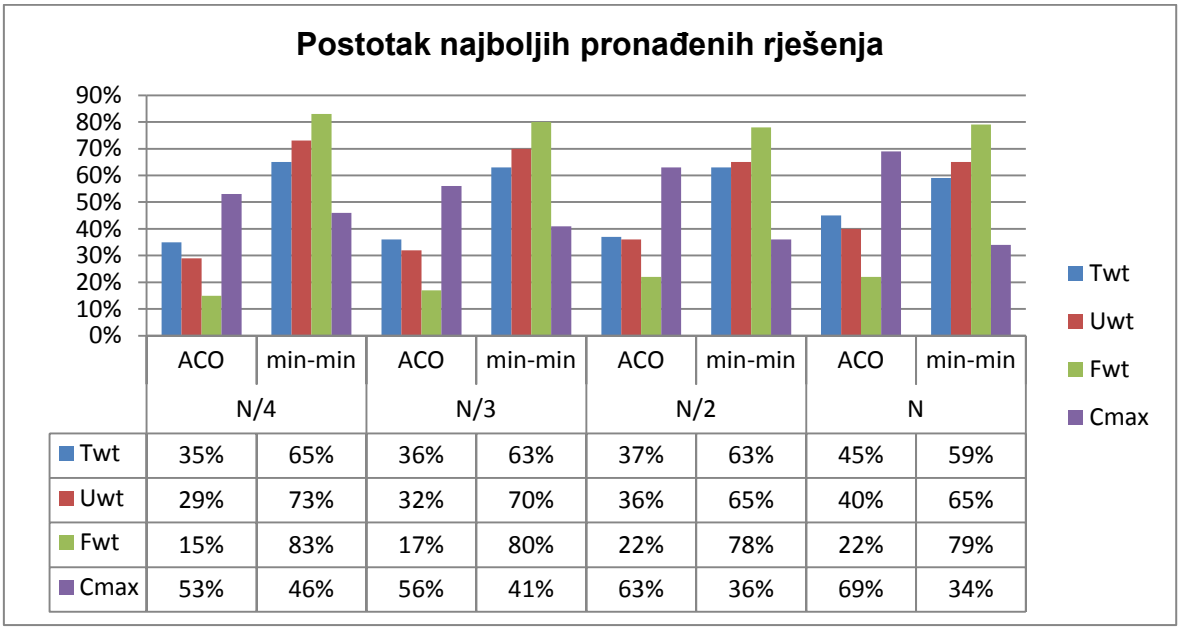
5.1.2. Utjecaj veličine kolonije

Slike 5.6 i 5.7 prikazuju rezultate optimizacije ukupne duljine rasporeda obzirom na broj mrava u koloniji. Faktor k je iz skupa $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$ (§3.2).



	ACO				MIN-MIN
	N/4	N/3	N/2	N	min-min
Twt	179464	175842	173995	162165	107763
Uwt	899,8	890,2	888,1	865,7	564,8
Fwt	574279	569824	562402	553098,64	310933,01
Cmax	63132	62832	60229	57842	62355

Slika 5.6 Optimiranje ukupne duljine rasporeda obzirom na broj mrava



	N/4		N/3		N/2		N	
	ACO	min-min	ACO	min-min	ACO	min-min	ACO	min-min
Twt	35%	65%	36%	63%	37%	63%	45%	59%
Uwt	29%	73%	32%	70%	36%	65%	40%	65%
Fwt	15%	83%	17%	80%	22%	78%	22%	79%
Cmax	53%	46%	56%	41%	63%	36%	69%	34%

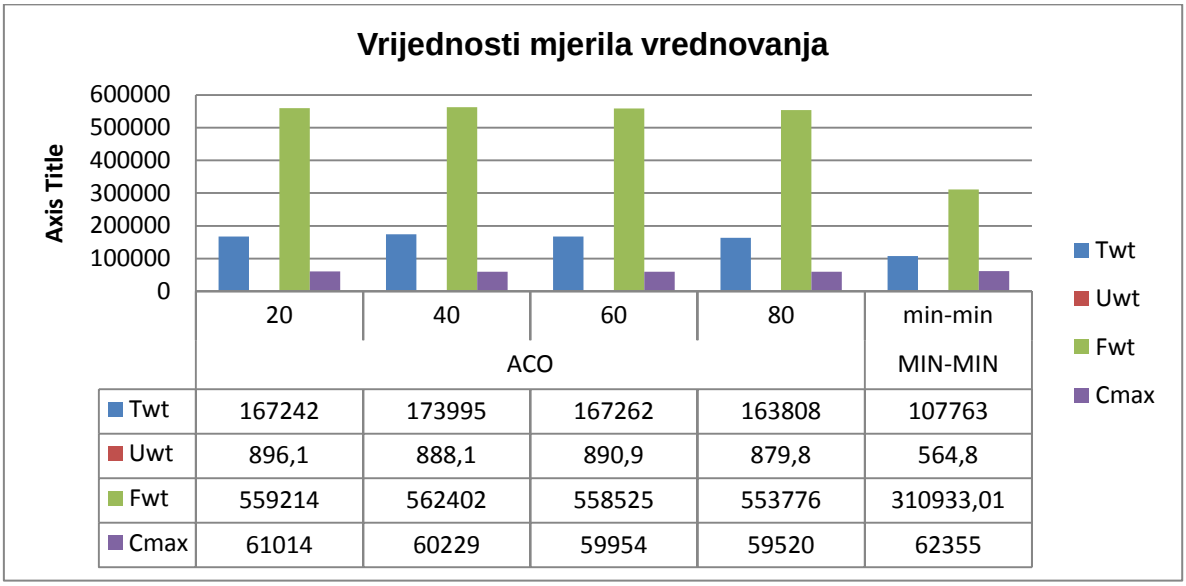
Slika 5.7 Postotci dominacije za ukupnu duljinu rasporeda obzirom na broj mrava

Iz rezultata se može uočiti da veći broj mrava pozitivno utječe na rezultat. Što je kolonija veća, u ciklusu prolaska kroz čvorove stvori se više rješenja, od kojih proizađe samo najbolje rješenje. Također, što više mrava ima, to prilikom

početnog određivanja položaja mrava pokrije se veći broj čvorova i podčvorova u grafu G.

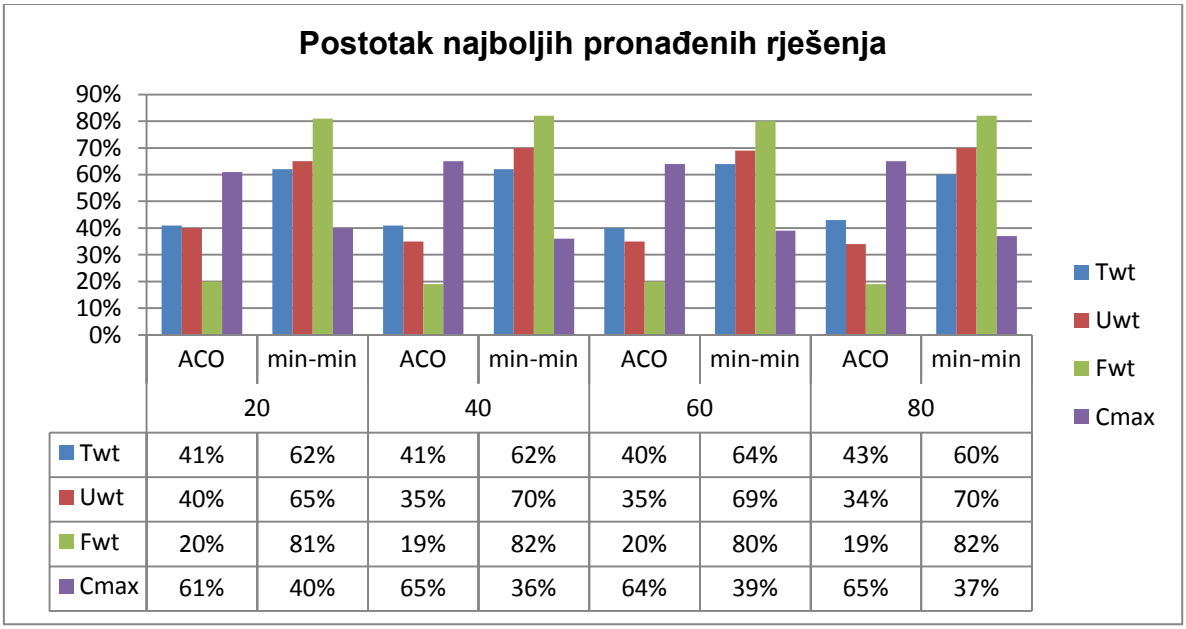
5.1.3. Utjecaj broja ciklusa

Slike 5.8 i 5.9 prikazuju rezultate optimizacije ukupne duljine rasporeda obzirom na broj ciklusa. Broj ciklusa, c, je is skupa $\{20,40,60,80\}$.



	20	40	60	80	min-min MIN-MIN
Twt	167242	173995	167262	163808	107763
Uwt	896,1	888,1	890,9	879,8	564,8
Fwt	559214	562402	558525	553776	310933,01
Cmax	61014	60229	59954	59520	62355

Slika 5.8 Optimiranje ukupne duljine rasporeda obzirom na broj ciklusa



	ACO	min-min	ACO	min-min	ACO	min-min	ACO	min-min
	20		40		60		80	
Twt	41%	62%	41%	62%	40%	64%	43%	60%
Uwt	40%	65%	35%	70%	35%	69%	34%	70%
Fwt	20%	81%	19%	82%	20%	80%	19%	82%
Cmax	61%	40%	65%	36%	64%	39%	65%	37%

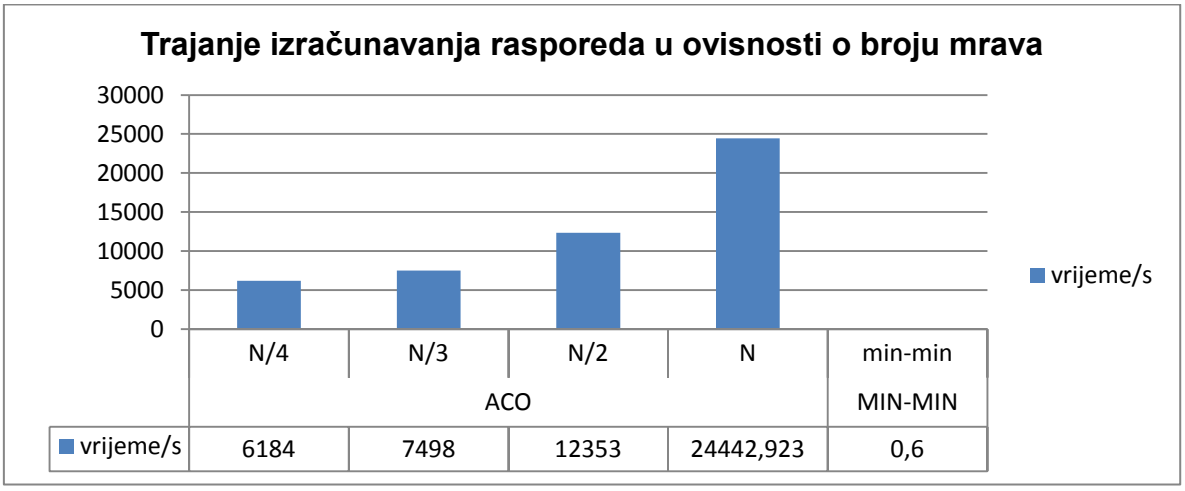
Slika 5.9 Postotci dominacije za ukupnu duljinu rasporeda obzirom na broj ciklusa

Iz rezultata se može uočiti da veći broj ciklusa pozitivno utječe na rezultat. Što je broj ciklusa veći, ukupna duljina rasporeda se smanjuje i povećava se dominacija nad algoritmom min-min. Objašnjenje takvog rezultata je što veći broj

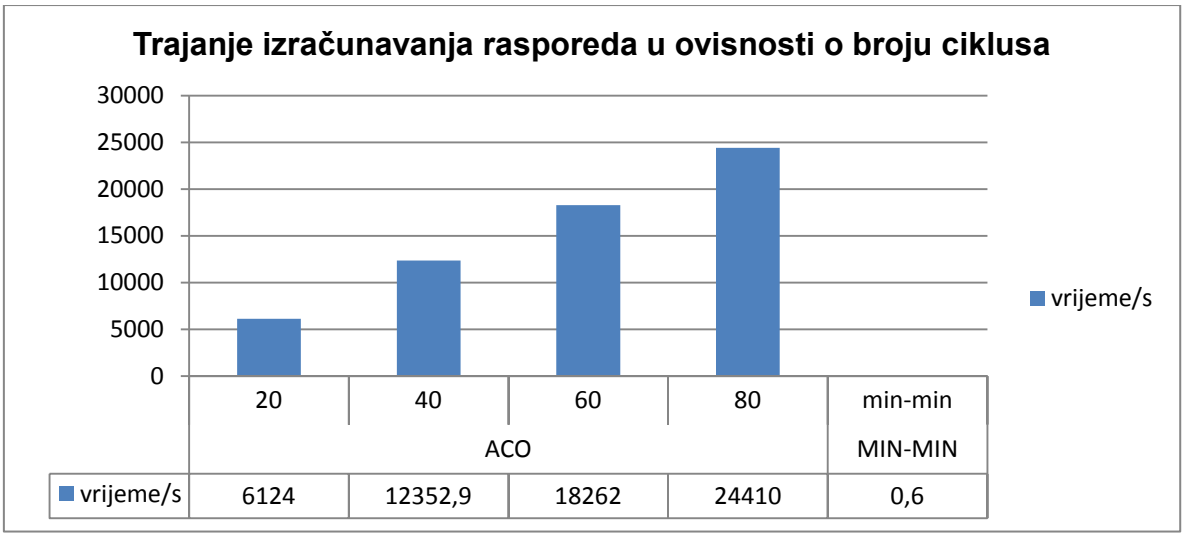
ciklusa, kao i mrava, dovodi do brže konvergencije optimalnom rješenju, a to se događa jer mravi više puta mogu ažurirati feromonske tragove.

5.1.4. Vrijeme izračunavanja rasporeda

Prosječno izračunavanje rasporeda za pretpostavljene vrijednosti (tablica 5.1) iznosi $2.5\,h$. Slike 5.10 i 5.11 prikazuju vrijeme izračunavanja rasporeda obzirom na broj mrava i na broj ciklusa.



Slika 5.10 Trajanje izračunavanja rasporeda u ovisnosti o broju mrava



Slika 5.11 Trajanje izračunavanja rasporeda u ovisnosti o broju ciklusa

Iz rezultata se uočava da je trajanje izračunavanja rasporeda u ovisnosti o broju mrava $eksponencijalne$ složenosti. Razlog takvoj složenosti su popratni efekti povaćenja broja mrava. Osim većeg broja ažuriranja feromonskih tragova, treba se osvježiti i tabu lista i ponovno rasporediti sve mrave u početne čvorove.

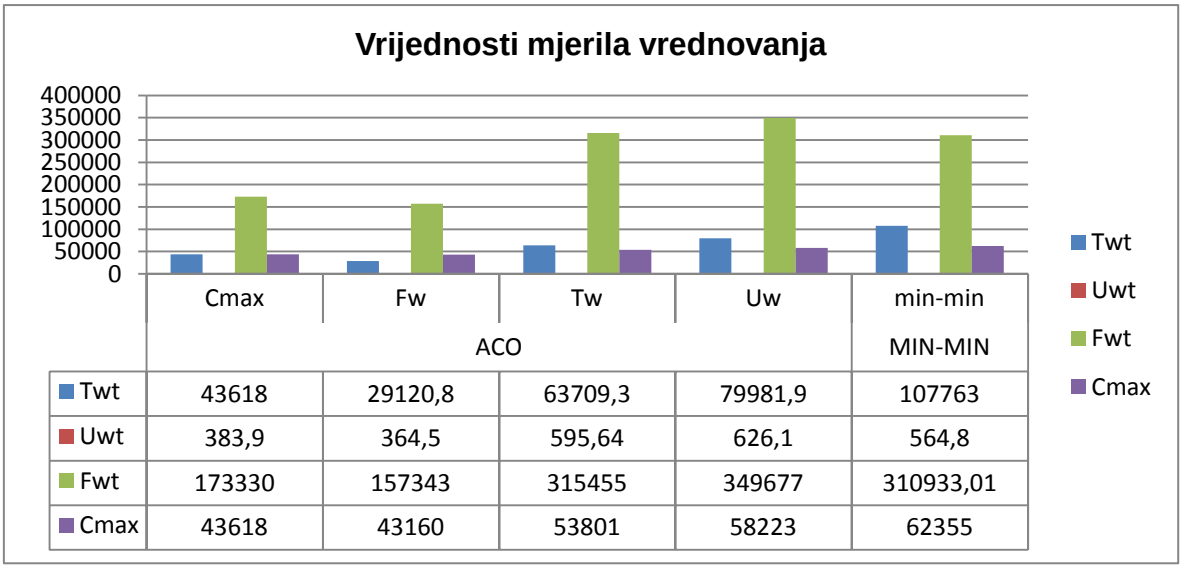
Za razliku od ovisnosti o broju mrava, trajanje izračunavanja rasporeda u ovisnosti o broju ciklusa je $linearne$ složenosti jer ne postoje popratni efekti.

5.2. Raspoređivanje na zahtjev

Prilikom raspoređivanja na zahtjev korištena je heuristička funkcija MON. Testirana su dva uvjeta: broj mrava i trajanje izvođenja. U svakom testiranju, osim testirane varijable, vrijednosti ostalih varijabli jednake su vrijednostima iz tablice 5.1.

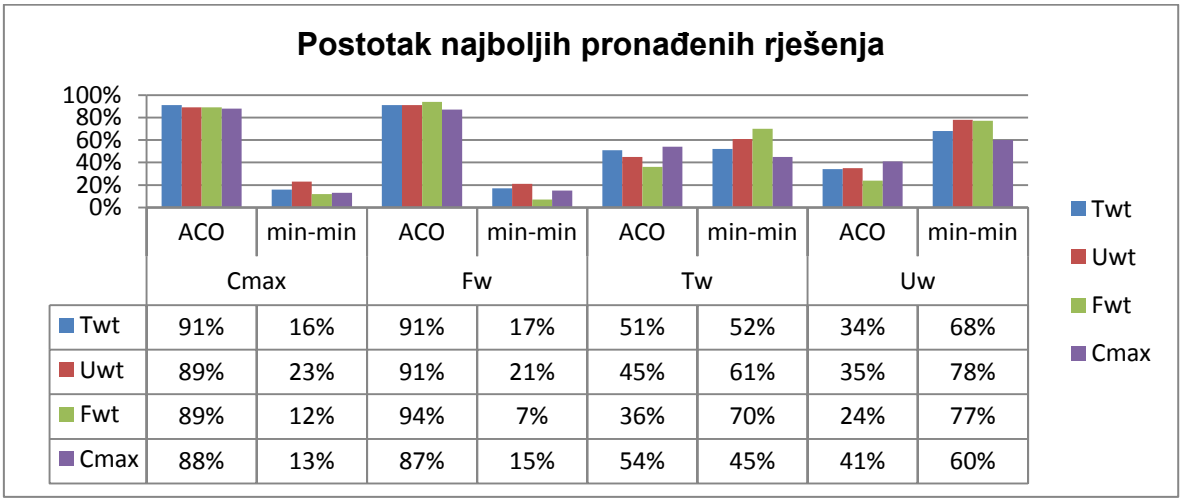
5.2.1. Rezultati za heurističku funkciju MON

Slikama 5.12 i 5.13 prikazani su rezultati optimizacije svih kriterija za heurističku funkciju MON.



	Cmax	Fw	Tw	Uw	min-min
	ACO				MIN-MIN
Twt	43618	29120,8	63709,3	79981,9	107763
Uwt	383,9	364,5	595,64	626,1	564,8
Fwt	173330	157343	315455	349677	310933,01
Cmax	43618	43160	53801	58223	62355

Slika 5.12 Vrijednosti mjerila vrednovanja za heurističku funkciju *MON*



	ACO	min-min	ACO	min-min	ACO	min-min	ACO	min-min
	Cmax		Fw		Tw		Uw	
Twt	91%	16%	91%	17%	51%	52%	34%	68%
Uwt	89%	23%	91%	21%	45%	61%	35%	78%
Fwt	89%	12%	94%	7%	36%	70%	24%	77%
Cmax	88%	13%	87%	15%	54%	45%	41%	60%

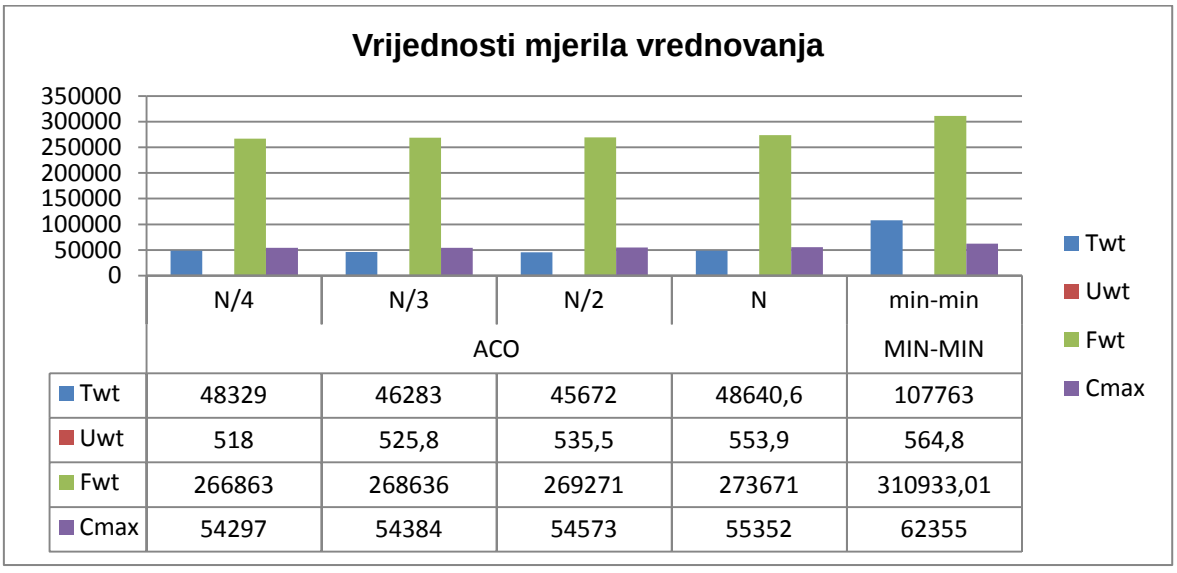
Slika 5.13 Postotci dominacije za heurističku funkciju *MON*

Rezultati testiranja pokazuju da prilikom raspoređivanja na zahtjev, za kriterije optimiranja ukupne duljine rasporeda i težinskog protjecanja, algoritam Ant

Colony System u potpunosti dominira algoritmom min-min. Razlog takvih rezultata je da velik broj mrava na relativno malenom broju poslova u sustavu vrlo brzo konvergira optimalnom rješenju. Malen broj poslova (§ 3.1) dovodi višestruke raspodjele mrava po čvorovima i podčvorovima grafa G te uzrokuje višestruka rješenja za svaki podčvor, od kojih se izabire najbolje rješenje i to je razlog brze konvergencije. Optimiranje po kriteriju težinskog zaostajanja ima vrlo slične rezultate kao i algoritam min-min, a optimiranje po težinskoj zakašnjelosti ima lošije rezultate od algoritma min-min. Razlog lošim rezultatima je brza konvergencija suboptimalnom rješenju, čija posljedica može biti forsirano optimiranje parametra koji je funkcija drugih parametara. Formule 2.10 i 2.5 pobliže opisuju ovisnost težinske zakašnjelosti o drugim parametrima. Zakašnjelost (formula 2.5) je funkcija od zaostajanja (formula 2.3), koja je funkcija od kašnjenja (formula 2.2). Optimirajući raspored po parametru težinske zakašnjelosti, optimiraju se zapravo svi parametri o kojima funkcija ovisi, a ako parametara ima puno, optimizacija je spora i loša.

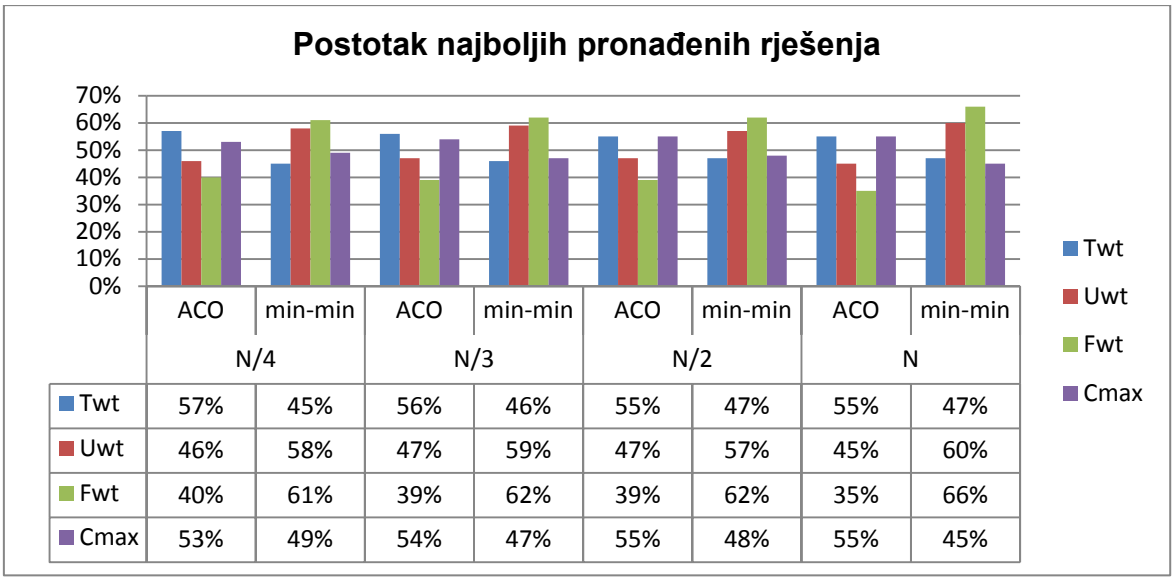
5.2.2. Utjecaj veličine kolonije

Slike 5.14 i 5.15 prikazuju rezultate optimizacije ukupne duljine rasporeda obzirom na broj mrava u koloniji. Faktor k je iz skupa $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$ (§3.2).



	ACO				MIN-MIN
	N/4	N/3	N/2	N	min-min
Twt	48329	46283	45672	48640,6	107763
Uwt	518	525,8	535,5	553,9	564,8
Fwt	266863	268636	269271	273671	310933,01
Cmax	54297	54384	54573	55352	62355

Slika 5.14 Optimiranje težinskog zaostajanja obzirom na broj mrava



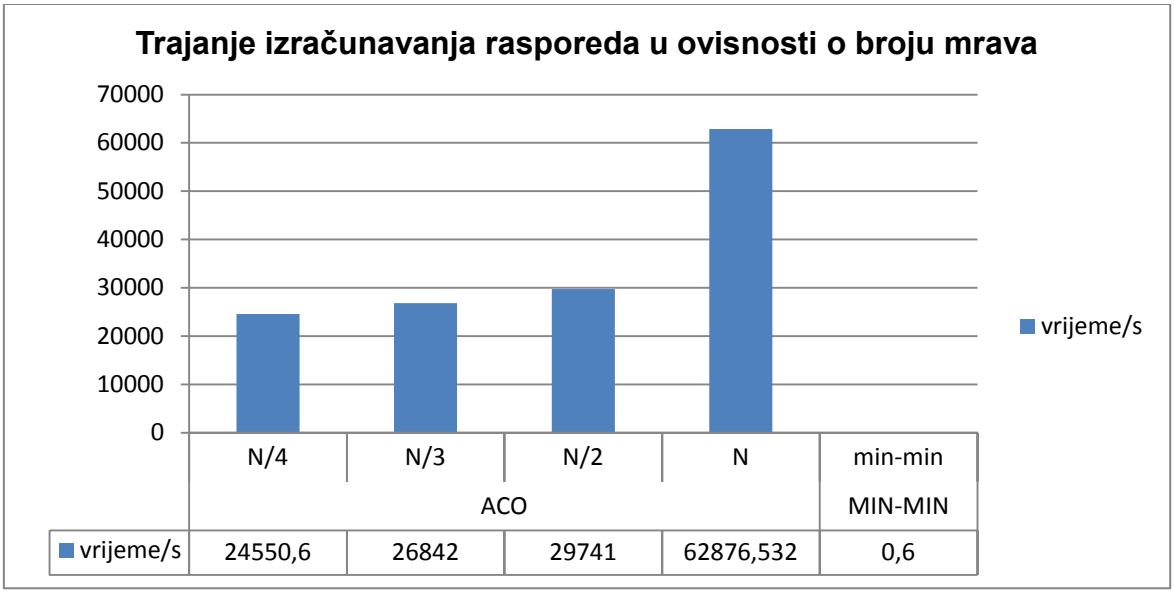
	N/4 ACO	N/4 min-min	N/3 ACO	N/3 min-min	N/2 ACO	N/2 min-min	N ACO	N min-min
Twt	57%	45%	56%	46%	55%	47%	55%	47%
Uwt	46%	58%	47%	59%	47%	57%	45%	60%
Fwt	40%	61%	39%	62%	39%	62%	35%	66%
Cmax	53%	49%	54%	47%	55%	48%	55%	45%

Slika 5.15 Postotci dominacije za težinsko zaostajanje obzirom na broj mrava

Iz rezultata se uočava da broj velik broj mrava negativno utječe na težinsko zaostajanje poslova.

5.2.3. Vrijeme izračunavanja rasporeda

Prosječno izračunavanje rasporeda za pretpostavljene vrijednosti (tablica 5.1) iznosi $6.5\ h$. Slika 5.16 prikazuje vrijeme izračunavanja rasporeda obzirom na broj mrava.



	N/4 ACO	N/3 ACO	N/2 ACO	N ACO	min-min MIN-MIN
vrijeme/s	24550,6	26842	29741	62876,532	0,6

Slika 5.16 Trajanje izračunavanja rasporeda u ovisnosti o broju mrava

Kao i kod predodređenog raspoređivanja, trajanje izračunavanja rasporeda u ovisnosti o broju mrava kod raspoređivanja na zahtjev ima $eksponencijalnu$ složenost zbog popratnih efekata. Ovdje je vremensko izvršavanje dulje jer se

popratni efekti događaju ne samo između skupova poslova, nego i svaki put kad novi posao uđe u skup J^{meta} (§ 3.1).

U tablici 5.2. prikazan je faktor usporenja raspoređivanja na zahtjev pomoću algoritma kolonije mrava u odnosu na algoritam min-min. Izmjereno je prosječno vrijeme, u milisekundama, jednog raspoređivanja na zahtjev. Za svaku kombinaciju broja poslova i broja strojeva, nasumično je izabran skup poslova nad kojim se provodilo ispitivanje. Za algoritam kolonije mrava korišteni su parametri iz tablice 5.1.

Tablica 5.2 Faktor usporenja jednog raspoređivanja na zahtjev algoritmom kolonije mrava u odnosu na algoritam min-min

Broj strojeva	**Broj poslova**	**milisekunde**		***Faktor usporenja***
		t_r(min-min)	**t_r(ACO)**	
3	12	0.167	17.31	104
	25	0.268	72.974	273
	50	0.153	446.549	2919
	100	0.204	3676.59	18023
6	12	0.05	15.471	310
	25	0.243	105.0	432
	50	0.108	650.327	6022
	100	0.301	4149.16	13785
10	12	0.112	137.4	1227
	25	0.104	142.24	1368
	50	0.072	882.737	12261
	100	0.204	9846.43	48267

Razlog sporosti raspoređivanja na zahtjev algoritmom kolonije mrava leži u činjenici da prilikom svakog raspoređivanja moraju se instancirati sve strukture podataka koje algoritam kolonije mrava koristi (matrice feromonskih tragova i matrica heuristike, tabu lista, položaj mrava). Faktor usporenja to je veći što je umnožak broja strojeva i broja poslova veći, a o tom umnošku direktno ovisi broj mrava i veličine svih navedenih struktura podataka (§ 3.2 i § 3.3).

Zaključak

Usporedba algoritama Ant Colony System i min-min pokazuje da oba algoritma imaju prednosti i mane. Najveća prednost algoritma min-min je što u izrazito kratkom vremenu (≈ 0.5 sekundi) raspoređuje poslove i pritom ostvaruje relativno dobre rezultate, u nekim slučajevima bolje od algoritma Ant Colony System. Mana algoritma je što su ti rezultati daleko od optimalnih vrijednosti.

Algoritam Ant Colony System najbolje rezultate postiže za heurističku funkciju MON, od preostalih ispitanih. Prilikom predodređenog raspoređivanja algoritam postiže bolje rezultate od min-min kod optimiranja ukupne duljine rasporeda; težinsko zaostajanje i težinska zakašnjelost su približno isti (min-min je u većini bolji); težinsko protjecanje postiže lošije rezultate od min-min. Prilikom raspoređivanja na zahtjev, algoritam Ant Colony System postiže izrazito dobre rezultate za optimizaciju ukupne duljine rasporeda i težinskog protjecanja; težinsko zaostajanje je približno isto (min-min je ponekad bolji); za težinsku zakašnjelost postiže lošije rezultate od min-min.

Povećanje broja mrava uzrokuje linearno poboljšanje prethodno dobivenih rezultata. Povećanje broja ciklusa, također, uzrokuje linearno poboljšanje prethodno dobivenih rezultata, no u puno manjoj mjeri. Broj mrava u algoritmu vezan je uz eksponencijalnu složenost programa, a broj ciklusa vezan je uz linearnu složenost. Najveća mana algoritma je izrazito dugo izvođenje u odnosu na algoritam min-min.

Pokazano je da raspoređivanje u okružju nesrodnih strojeva uporabom evolucijskog računanja postiže zadovoljavajuće rezultate. Evolucijski algoritam Ant Colony System davao bi još bolja rješenja da su pretpostavljeni parametri u programu bili bolji, ali traženje njihovog optimalnog iznosa zahtjeva veće računalne resurse. Kao najveća mana algoritma pokazuje se vrijeme izvođenja rasporeda pa se algoritam Ant Colony System ne preporuča za dinamičke sustave u kojima je brzina raspoređivanja bitna, već za statičke sustave u kojima je bitna optimalnost.

Literatura

[1] Čeri, M. et al., *Evolucijski algoritmi*, siječanj 2008., *Projekt: Evolucijski algoritmi*, http://www.zemris.fer.hr/~golub/ga/studenti/projekt2007/index.html, 5. svibnja 2011.

[2] Čupić, M., *Prirodom inspirirani optimizacijski algoritmi*, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Fakultet elektrotehnike i računarstva, 2009.

[3] Dorigo, M., Stützle, T., *Ant Colony Optimization*, MIT Press, Cambridge, MA, 2004.

[4] Jakobović, D., *Raspoređivanje zasnovano na prilagodljivim pravilima*, doktorska disertacija, Fakultet elektrotehnike i računarstva, 2005.

[5] *Ant colony optimization algorithms*, 20. travnja 2011., *Ant colony optimization algorithms*, http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms, 5. svibnja 2011.

Sažetak

Raspoređivanje u okružju nesrodnih strojeva uporabom evolucijskog računanja

U radu je opisan postupak raspoređivanja i okružje raspoređivanja na nesrodnim strojevima. Za raspoređivanje koristi se algoritam evolucijskog računanja Ant Colony System. Algoritam je uspoređen s postojećim heurističkim algoritmom min-min. Ostvareno je okružje u kojem je moguće provesti usporedbu učinkovitosti oba algoritma. Usporedba algoritama napravljena je za dva tipa raspoređivanja: predodređeno raspoređivanje i raspoređivanje na zahtjev. Algoritam evolucijskog računanja Ant Colony System pokazuje sličnu ili bolju učinkovitost u usporedbi s heurističkim algoritmom min-min. Navedene su prednosti i mane korištenja oba algoritma prilikom raspoređivanja u okružju nesrodnih strojeva.

Ključne riječi: raspoređivanje u okružju nesrodnih strojeva, evolucijsko računanje, Ant Colony System, heuristički algoritam min-min, predodređeno raspoređivanje, raspoređivanje na zahtjev

Scheduling for parallel unrelated machines using evolutionary algorithms

This BSc thesis gives a description of the scheduling process and parallel unrelated machines environment. Ant Colony System evolutionary algorithm is used for scheduling. The algorithm is compared to the existing heuristic algorithm min-min. A framework which enables the comparison of efficiency of both algorithms is implemented. Two types of scheduling comparison are implemented: offline scheduling and online scheduling. Ant Colony System evolutionary algorithm exhibits similar or, in some cases, better efficiency compared to the heuristic algorithm min-min. Advantages and disadvantages of using both algorithms in scheduling for parallel unrelated machines are given.

Keywords: parallel unrelated machines scheduling, evolutionary algorithms, Ant Colony System, heuristic algorithm min-min, offline scheduling, online scheduling