SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Usporedba prikaza rješenja za okruženje nesrodnih strojeva

Ivan Vlašić
Voditelj: *Izv. prof. dr. sc. Domagoj Jakobović*

Zagreb, svibanj 2017.

SADRŽAJ

1. Uvod

Okruženje nesrodnih strojeva sastoji se od $N = 1, ..., n$ poslova od kojih se svaki mora izvršiti na samo jednom od $M = 1, ..., m$ strojeva. Strojevi se smatraju nesrodnima ako vrijeme obavljanja posla ovisi o stroju na kojem se taj posao izvršava.

Svaki posao je opisan s nekoliko svojstava koja uključuju vrijeme obrade p_{ij} koje označava vrijeme izvršenja posla j na stroju i; vrijeme objavljivanja r_{ij} koje označava vrijeme dolaska posla; vrijeme kraja d_j koje označava vrijeme do kojeg posao treba završiti i težina w_j koja označava važnost posla.

Za rješavanje problema raspoređivanja u okruženju nesrodnih strojeva u ovom radu će se koristiti genetski algoritam koji simulira proces prirodne evolucije kako bi pronašao dovoljno dobra rješenja.

Cilj je implementirati nekoliko različitih prikaza rješenja za okruženje nesrodnih strojeva te usporediti njihove rezultate. Kako se prikazi razlikuju, razlikovat će se i operatori koji se koriste.

2. Opis problema

Cilj problema raspoređivanja je doći do optimalnog rasporeda poslova na strojevima kako bi se optimizirao određeni kriterij ocjenjivanja.

Iako se u literaturi koristi optimizacija nekoliko različitih parametara, u ovom radu će se koristiti ukupno težinsko kašnjenje Twt. Ukupno težinsko kašnjenje se definira kao suma produkata težina posla i njegovog kašnjenja.

$Twt = \sum_j w_j T_j$

w_j je težina posla s indeksom j, a T_j kašnjenje stroja s indeksom j koje je definirano kao razlika između stvarnog i predviđenog vremena završetka posla.

Problem raspoređivanja se može klasificirati u nekoliko grupa ovisno o pojedinim svojstvima. Ovisno o dostupnosti parametara poput informacija o svim poslovima koji će doći u sustav problemi raspoređivanja se mogu podijeliti u grupu u kojoj su te informacije poznate od početka te grupu u kojoj informacije o poslu postanu dostupne tek kad taj posao dođe u sustav.

Problemi raspoređivanja također mogu biti statični kod kojih je cijeli raspored raspoređivanja konstruiran prije izvršavanja sustava te dinamički kod kojih se raspored gradi tijekom izvršavanja sustava.

Naglasak će biti na statičkom raspoređivanju gdje su informacije o poslovima dostupne od početka sustava.

Za kreiranje rasporeda se mogu koristiti različiti algoritmi optimizacije no u ovom radu će naglasak biti na genetskom algoritmu.

Genetski algoritam je tehnika optimizacije koja može biti korištena za pronalazak rješenja visoke kvalitete. Glavni princip je simulacija procesa prirodne evolucije u kojoj bolje jedinke preživljavaju te se njihov genetski materijal prenosi u sljedeće generacije.

Kako bi se jedinke mogle uspoređivati potrebno je definirati funkciju dobrote koja određuje koliko je pojedino rješenje dobro u odnosu na naše kriterije optimizacije. Pri tome se koriste operatori selekcije koji biraju jedinke iz trenutne populacije na način da one s većom dobrotom imaju i veću šansu biti izabrane za daljnji proces križanja i

mutacije gdje se kombiniraju svojstva dvije jedinke te uvode slučajne promjene kako bi se proširio prostor pretraživanja.

Algoritam iteracijski primjenjuje navedene operatore dok se ne postigne kriterij zaustavljanja koji može biti zadana dobrota, broj iteracija i slično.

Budući da u kvaliteti dobivenog rješenja veliku ulogu igra prikaz jedinke, cilj ovog rada je implementirati nekoliko različitih prikaza s odgovarajućim operatorima te usporediti dobivena rješenja.

Svi prikazi će za operator selekcije koristi turnirsku selekciju veličine $k = 3$. Turnirska selekcija izabire k jedinki iz trenutne populacije te dvije s najvećom dobrotom križa i s određenom vjerojatnosti primjenjuje operator mutacije prije ubacivanja dobivenog djeteta u populaciju.

3. Prikazi rješenja

Implementirati će se tri različita prikaza rješenja koji uključuju rješenje u obliku vektora realnih brojeva, rješenje koje poslove raspoređuje u grupe za izvršenje na pojedinom stroju i rješenje u obliku matrice.

3.1. Prikaz vektorom realnih brojeva

Prvi prikaz koristi vektor veličine n gdje n predstavlja broj poslova koje je potrebno obaviti u sustavu. Vrijednosti u vektoru se nalaze u intervalu $[0, m)$ gdje m predstavlja broj strojeva na kojima je potrebno obaviti zadane poslove.

Dekodiranje se obavlja na način da cijeli dio svakog broja predstavlja stroj na kojem će se zadani posao izvršiti, a ostatak prioritet koji taj posao ima na zadanom stroju.

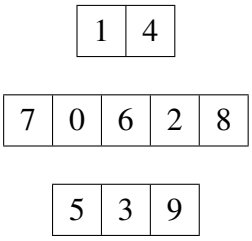
Na slici 3.1 je ilustriran prikaz jedinke u sustavu u kojem se nalazi 7 poslova i 3 stroja.

Poslovi na pozicijama 0, 4 i 6 će se obaviti na stroju 0, posao na poziciji 3 na poslu 1 te poslovi na pozicijama 1, 2 i 5 na stroju 2. Ovisno o zadanom načinu određivanja prioriteta, na stroju 0 će najveću prednost imati posao na poziciji 4, zatim 6 i na kraju 0. Isti princip određivanja se primjenjuje i za sve ostale strojeve.

Za ovaj prikaz rješenja je korišteno 15 različitih operatora križanja, a za operator mutacije je korišten jednostavni operator koji samo mijenja vrijednost nasumičnog određenog polja vektora.

0.23	2.52	2.13	1.14	0.86	2.34	0.54

Slika 3.1: Prikaz vektorom realnih brojeva



Slika 3.2: Prikaz grupama poslova

3.2. Prikaz grupama poslova

Prikaz grupama poslova se sastoji od jednog vektora za svaki pojedini stroj koji sadrže indekse poslova koje je potrebno izvršiti na tom zadanom stroju.

Na slici 3.2 je ilustriran prikaz jedinke u sustavu s 10 poslova i 3 stroja.

Poslovi na pozicijama 1 i 4 će se izvršiti na stroju 0, poslovi na pozicijama 7, 0, 6, 2 i 8 na stroju 1 te poslovi na pozicijama 5, 3 i 9 na stroju 2.

Za operator križanja se u prvom koraku iz jednog roditelja za svaki stroj nasumično biraju točke prekide te se potom svi poslovi do te pozicije prebacuju u istoimene strojeve u djetetu. Potom se za svaki stroj u drugom roditelju poslovi koji još nisu raspoređeni u djetetu u njega i ubacuju u odgovarajuće strojeve.

Na taj se način dobije nova jedinka koja u svakom stroju ima određen, nasumičan broj poslova iz strojeva s odgovarajućim indeksom iz oba roditelja.

Za operator mutacije se nasumično biraju stroj iz kojeg uzimamo posao te pozicija tog posla i stroj u koji ubacujemo posao te pozicija na koju se posao ubacuje. U slučaju da se posao ne može ubaciti na određenu poziciju u novom stroju, posao se dodaje na kraj.

3.3. Prikaz matricom

Prikaz matricom koristi matricu veličine $m \times n$ gdje m predstavlja broj strojeva u sustavu, a n broj poslova. Svakom stroju je pridružen jedan redak matrice koji na križanjima sa stupcima za one poslove koji se izvršavaju na tom stroju ima vrijednost različitu od 0. Polja matrice poprimaju vrijednosti u intervalu $(0, 1]$ te određuju prioritet izvršavanja posla u stroju u kojem se nalaze.

Na slici 3.3 je ilustriran prikaz jedinke u sustavu s 5 poslova i 3 stroja. Poslovi na pozicijama 0 i 3 će se izvršiti na stroju 0, posao na poziciji 1 na stroju 1 te poslovi na pozicijama 2 i 4 na stroju 3.

$$\begin{bmatrix} 0.21 & 0 & 0 & 0.72 & 0 \\ 0 & 0.48 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0.24 \end{bmatrix}$$

Slika 3.3: Prikaz matricom

Uzimajući u obzir prioritete obavljanja, na stroju 0 će prednost imati posao na poziciji 3, a na stroju 2 posao na poziciji 2.

Operator križanja stvara novu jedinku tako da sa vjerojatnosti od 50% uzima stupac iz jednog ili iz drugog roditelja.

Za mutaciju se koriste dva različita operatora. Prvi nasumično bira jedan od poslova i dodjeljuje mu novi prioritet dok drugi premješta posao u novi, nasumično odabrani stroj.

4. Rezultati

Kako bi se implementirani prikazi usporedili koristit će se 60 problema raspoređivanja u kojima broj strojeva može biti 3, 6 ili 10, a broj poslova 12, 25, 50 ili 100 ovisno o promatranom problemu.

Svaki primjer će biti pokrenut 10 puta na populaciji veličine 30 jedinki, a postavljeno je ograničenje od milijun generacija kako bi algoritam uspio konvergirati.

Svi prikazi će biti promatrani i sa različitim vjerojatnostima mutacije koja može biti 0.3, 0.6 ili 0.9.

U tablici 4.1 ukupna dobrota je prikazana kao suma najboljih dobrota za svaki od primjera, dok je u tablici 4.2 prikazana suma prosječnih dobrota za svaki od promatranih primjera.

U tablici 4.1 ukupna dobrota je izračunata kao suma najboljih dobrota svakog od primjera, dok je u tablici 4.2 prvo izračunata prosječna dobrota za svaki od primjera tijekom 10 pokretanja te je rezultat suma tih prosjeka.

Može se primjetiti da prikazi dolaze do jako sličnih rješenja koja se u određenoj mjeri poboljšavaju rastom vjerojatnosti mutacije.

U analizi rezultata treba uzeti u obzir da su prikaz grupa poslova, a pogotovo prikaz matrice dosta složeniji od prikaza realnih brojeva te njihovo izvođenje, uz postizanje istih, ili u slučaju matrice malo lošijih rezultata, traje dosta duže.

U rezultatima mjerenja u tablici 4.2 se također primjećuje da porastom vjerojatnosti mutacije raste i prosječna kvaliteta rješenja, a prikaz realnih brojeva ostvaruje najbolje rezultate.

Iz navedenog se može zaključiti da prikaz realnih brojeva postiže najoptimalnije rezultate uzevši u obzir dobivena rješenja i potrebno vrijeme izvođenja za svaki od prikaza.

Tablica 4.1: Rezultati kao suma najboljih rješenja

Prikaz	Vjerojatnost mutacije		
	0.3	0.6	0.9
1.	9.86425904	9.57312844	9.52555654
2.	9.61311714	9.57599344	9.56612344
3.	9.94679144	9.73379734	9.55833044

Tablica 4.2: Rezultati kao suma prosječnih rješenja

Prikaz	Vjerojatnost mutacije		
	0.3	0.6	0.9
1.	10.74291	10.20773	9.90216
2.	10.79949	10.192083	9.8637
3.	11.23524	10.55469	9.97587

5. Zaključak

Iako kvaliteta rješenja u genetskom programiranju često u velikoj mjeri ovisi o načinu prikaza jedinke, u ovom slučaju se može primjetiti da je svaki od prikaza postigao iste ili slične rezultate.

Faktor koji može utjecati na odabir optimalnog rješenja su složenost i brzina izvođenja algoritma koja je za prikaz grupama poslova i prikaz matricom dosta veća od prikaza vektorom realnih brojeva.

Primjeti se i da porastom vjerojatnosti mutacije raste i kvaliteta rješenja jer se algoritmu dopušta da pretražuje širi prostor stanja, no često je potrebno pripaziti jer u određenim slučajevima prevelika vjerojatnost mutacije može poništiti utjecaj selekcije i križanja te dolaziti do nasumičnih rezultata koja bi u takvom slučaju bila puno lošija od usmjerenog pretraživanja koje se postiže funkcijom dobrote i optimalnim izborom parametara.

6. Literatura

1. Marko Đurasević, Domagoj Jakobović, Comparison of solution representations for scheduling in the unrelated machines environment

2. Eva Vallada, Rubén Ruiz, A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times

3. J. Behnamian, M. Zandieh b, S.M.T. Fatemi Ghomi, Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm

4. Chiuh-Cheng Chyu, Wei-Shung Chang, A Competitive Evolution Strategy Memetic Algorithm for Unrelated Parallel Machine Scheduling to Minimize Total Weighted Tardiness and Flow Time

7. Sažetak

Problemi raspoređivanja strojeva se često susreću u različitim industrijskim postrojenjima. U industriji, ovi strojevi često nemaju ista svojstva što može biti posljedica nadograđivanja ili drugačije funkcionalnosti. Zbog toga se problemi raspoređivanja nesrodnih strojeva susreću puno češće od srodnih strojeva kod kojih je vrijeme obavljanja posla isto neovisno o stroju.

Područje još nije potpuno istraženo te postoje različiti načini rješavanja problema kako bi se došlo do optimalnih rješenja i uštedilo na vremenu i resursima. U ovom radu je pokazano da se genetski algoritam može upotrijebiti u rješavanju navedenih optimizacija i postići dovoljno dobre rezultate sa upotrebom različitih prikaza jedinki.