Univerzitet u Beogradu

Matematički fakultet

Rešavanje problema raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama primenom hibridnih metaheurističkih metoda

Master rad

Student:
Dušan Džamić

Mentor:
dr Miroslav Marić

septembar 2014.

Mentor:	dr Miroslav Marić, Matematički fakultet, Univerzitet u Beogradu
Članovi komisije:	dr Gordana Pavlović Lažetić, *Matematički fakultet,* *Univerzitet u Beogradu*
	dr Zorica Stanimirović, *Matematički fakultet,* *Univerzitet u Beogradu*
Datum odbrane:	_____

Rešavanje problema raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama primenom hibridnih metaheurističkih metoda

Apstrakt - *Problem raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama (Hybrid flow shop (HFS) scheduling problem) često se javlja u automatizovanim industrijskim postrojenjima, kao što su postrojenja za proizvodnju stakla, građevinskih materijala, čelika, papira i tekstila. U ovom radu su predstavljeni metaheuristički algoritmi optimizacije rojevima čestica (PSO), simuliranog kaljenja (SA), tabu pretraživanja (TS) i metoda promenljivih okolina (VNS) za rešavanje HFS problema u cilju minimizacije vremena potrebnog za realizaciju procesa proizvodnje. Pored toga, u pokušaju da se iskoriste dobre osobine svake metaheuristike, predložena su dva hibridna metaheuristička algoritma PSO-TS i PSO-VNS-SA. Svi predloženi algoritmi su testirani na skupu instanci iz literature za HFS problem. Eksperimentalni rezultati pokazuju da hibridni metaheuristički algoritam PSO-VNS-SA dostiže kvalitetna rešenja kao i drugi predloženi algoritmi u literaturi.*

Ključne reči: *višefazna proizvodnja; optimizacija rojevima čestica; tabu pretraga; metoda promenljivih okolina; simulirano kaljenje; hibridni metaheuristički algoritmi.*

Abstract - *Hybrid flow shops (HFS) are common manufacturing environments in many industries, such as the glass, building materials, steel, paper and textile industries. In this paper, we propose metaheuristic methods for solving the HFS scheduling problem with minimum makespan objective: a particle swarm optimization (PSO), tabu search (TS), simulated annealing (SA) and variable neighbourhood search (VNS) algorithm. In addition, two hybrid metaheuristic algorithms PSO-TS and PSO-VNS-SA have been proposed in an attempt to take advantage of the good features from more than one metaheuristic. The proposed algorithms are tested on the well-known benchmark problems for HFS problem. Experimental results show that the proposed hybrid algorithm PSO-VNS-SA is very competitive for the hybrid flow shop scheduling problem.*

Keywords: *hybrid flow shop; particle swarm optimization; tabu search; variable neighbourhood search; simulated annealing; hybrid metaheuristic algorithm.*

Sadržaj

Uvod

Glavna snaga računara se odlikuje u brzom sprovođenju složenih računskih operacija koje su sastavni deo mnogih problema. Ipak postoje problemi koji uz pomoć računara i egzaktnih metoda nisu rešivi u realnom vremenu. Takvi su i problemi kombinatorne optimizacije, kod kojih je broj (dopustivih) rešenja konačan, ali izuzetno velik. Kroz razvoj teorije algoritama došlo se do saznanja da mnogi problemi kombinatorne optimizacije pripadaju klasi NP-teških problema za čije rešavanje su poznati algoritmi u najboljem slučaju eksponencijalne složenosti. Eksponencijalni rast se najbolje može razumeti na poznatoj priči o otkriću šaha, od strane "skromnog"matematičara koji je na pitanje cara kako da mu se oduži zatražio da mu za prvo polje šahovske table preda jedno zrno pšenice, za drugo polje dva zrna, za treći četiri, za četvrto osam i tako za svako sledeće polje dva puta više zrna nego za prethodno. Car se iznenadio skromnošću matematičara i naredio svojim slugama da pripreme vreću pšenice. Kada su počeli da računaju shvatili su da careve zalihe nisu dovoljne čak i da ih je sto puta više jer broj zrna pšenice koji se dobija računanjem je 18.446.744.073.709.551.615. Kako u jednom kubnom metru pšenice staje približno 15.000.000 zrna, to znači kada se prevede, da bi matematičar Seta trebao dobiti 12.000 km^3 žita. Poređenja radi, ako bi se pšenica čuvala u skladištu osnove 4 x 10 metara, visina bi mu bila 300.000.000 km, što je jednako putu od Zemlje do Sunca i nazad.

Postavlja se pitanje kako pristupiti problemima kod kojih je prostor dopustivih rešenja izuzetno velik. Prvo zapažanje je da u praksi često nije potrebno egzaktno rešenje nekog problema već je dovoljno i približno rešenje. U tu svrhu došlo je do razvoja heurističkih metoda koje za relativno kratko vreme mogu da daju veoma kvalitetna rešenja koja često i jesu optimalna, ali ne postoji garancija i dokaz optimalnosti. Ovakav pristup privukao je veliku pažnju akademske zajednice jer je jedini praktični način za pronalaženje kvalitetnih rešenja optimizacionih problema, pa se sve više pažnje posvećuje razvoju i usavršavanju heurističkih metoda. To je dovelo i do razvoja univerzalnih heuristika tzv. metaheuristika. Metaheuristike predstavljaju algoritme koje se sastoje od uopštenih skupova pravila koje se mogu primeniti na raznovrsne probleme optimizacije dok su klasične heuristike namenjene pojedinačnim problemima i koriste konkretne osobine datog problema pri njegovom rešavanju.

Cilj ovog rada je primena i implementacija metaheurističkih algoritama za rešavanje problema raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama (Hybrid flow shop - HFS) koji se često susreće u mnogim industrijskim postrojenjima, na primer, u proizvodnji stakla, čelika, papira i tekstila. HFS problem predstavlja kombinaciju klasičnog problema višefazne proizvodnje (Flow shop problem) gde svaki posao prolazi redom kroz nivoe u kojima je dostupna jedna mašina koja obrađuje posao i problema proizvodnje sa paralelnim mašinama (Job shop scheduling) gde postoji jedna faza proizvodnje u kojoj je dostupan određen broj identičnih mašina koje rade paralelno. Oba pojedinačna problema su veoma dobro istražena u teoriji raspoređivanja i opisan u velikom broju radova, međutim ne i njihova kombinacija. HFS problem se smatra osnovnim problemom za modeliranje realnih automatizovanih industrijskih postrojenja. U osnovni model mogu se uključiti dodatna ograničenja,

na primer vreme neophodno za pauzu između obrade dva posla.

U prvom poglavlju predstavljene su metaheurističke metode koje će biti korišćene za rešavanje HFS problema. Detaljno su opisane ideje i osnovni koncepti metode simuliranog kaljenja, tabu pretraživanja, metode promenljivih okolina i optimizacije rojevima čestica. U drugom poglavlju detaljno je opisan problem višefazne proizvodnje sa paralelnim mašinama, njegov matematički model kao i primeri primene samog problema. Kroz treće poglavlje opisuju se detalji implementacije izabranih metaheurističkih algoritama, dok se u četvrtom poglavlju daje predlog za hibridizaciju spomenutih algoritama u cilju da se dobre osobine dve ili više metaheurističke metode kombinuju u novu složeniju metodu za rešavanje velikih instanci problema. U petom poglavlju predstavljeni su rezultati dobijeni predloženim algoritimima i upoređeni sa rezultatima iz relevantnih radova. U završnom delu rada i šestom poglavlju dat je zaključak na osnovu eksperimentalnih rezultata i mogućnost daljih istraživanja.

1

Metaheurističke metode

Metaheuristike se sastoje od uopštenih skupova pravila koja se mogu primeniti za rešavanje raznovrsnih problema optimizacije. Predstavljaju značajno, a najčešće i jedino, sredstvo u rešavanju realnih problema baziranih na optimizaciji, a osim toga, mogu se primeniti za ubrzavanje egzaktnih metoda tako što se iskoriste za brzo dobijanje dobrog početnog rešenja.

Osobine koje jedna efikasna metaheuristika treba da poseduju kako bi obezbedila značaj i na praktičnom i na teorijskom planu [16] su:

- **jednostavnost**: treba da budu zasnovane na jednostavnim i lako razumljivim pravilima;
- **preciznost:** koraci kojima se opisuje metaheuristička metoda treba da su formulisani precizno, po mogućnosti matematičkim terminima;
- **doslednost:** svi koraci metode treba da budu u skladu sa pravilima kojima je metaheuristika definisana;
- **efikasnost:** primena metaheuristike na neki konkretan problem treba da obezbedi dobijanje rešenja bliskih optimalnom za većinu realnih primera, naročito za zvanične test primere raspoložive u toj klasi;
- **efektivnost:** za svaki konkretan problem, metoda mora da obezbedi optimalno ili rešenje blisko optimalnom u razumnom vremenu izvršavanja;
- **robusnost:** metoda treba da daje podjednako dobre rezultate za širok spektar primera iz iste klase, a ne samo za neke odabrane test primere;
- **jasnoća:** treba da bude jasno opisana kako bi se lako razumela i, što je još važnije, lako implementirala i koristila;
- **univerzalnost:** principi kojima je metoda definisana treba da budu opšteg karaktera kako bi se sa lakoćom mogla primeniti na nove probleme.

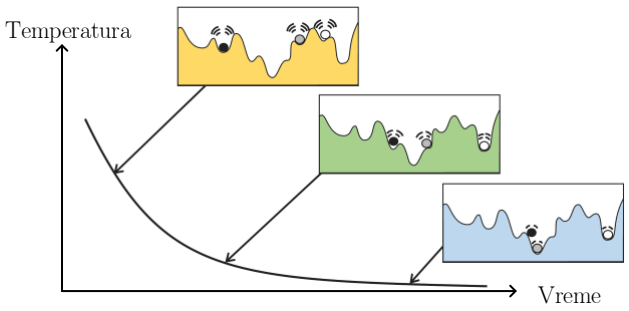
Do danas razvijene su mnoge metaheurističke metode optimizacije, najčešće po uzoru na neke poznate procese, prvenstveno u biologiji (genetski algoritmi i razne specijalne varijante evolutivnih algoritama, neuralne mreže, mravlje kolonije, kolonije pčela), u fizici (simulirano kaljenje), ali i metode inspirisane lokalnim pretraživanjem sa raznim idejama da se izbegne zamka lokalnog minimuma (iterativno lokalno pretraživanje, metoda promenljivih okolina, tabu pretraživanje) i druge [30].

1.1 Metoda simuliranog kaljenja

Simulirano kaljenje (Simulated Annealing, SA) je metoda zasnovana na lokalnom pretraživanju, uz mehanizam inspirisan procesom kaljenja čelika koji omogućava izlazak iz lokalnog optimuma. Algoritam je preldožen 1983. godine od strane Kirkpatrika i drugih [24].

Pri procesu metalurškog kaljenja čelika cilj je oplemenjivanje metala tako da on postane čvršći. Da bi se postigla čvrstoća metala potrebno je njegovu kristalnu rešetku pomeriti tako da ima minimalnu potencijalnu energiju. Prvi korak u kaljenju čelika je zagrevanje do određene visoke temperature, a zatim nakon kratkog zadržavanja na toj temperaturu počinje postepeno hlađenje. Pri postepenom hlađenju atomi metala nakon procesa kaljenja formiraju pravilnu kristalnu rešetku i time se postiže energetski minimum kristalne rešetke. Važno je napomenuti da brzo hlađenje može da uzrokuje pucanje metala.

Funkcija cilja za koju se traži globalni optimum se može posmatrati kao energija kristalne rešetke ako je potrebno minimizovati funkciju cilja ili negativna energija kristalne rešetke ako je potrebno maksimizovati funkciju cilja. Metoda počinje izborom početnog rešenja i postavljanjem početne temperature na relativno visoku vrednost. Zatim se na slučajan način bira jedno rešenje iz okoline tekućeg rešenja. Ako je to rešenje bolje, onda ono postaje novo tekuće rešenje. Ako je novo rešenje lošije od tekućeg, ono ipak može da postane tekuće rešenje ali sa određenom verovatnoćom. Verovatnoća prihvatanja lošijeg rešenja obično zavisi od parametra koji predstavlja temperaturu i vremenom opada kako se algoritam izvršava. Ovakav pristup obezbeđuje izlazak iz lokalnog optimuma. U početku je ta verovatnoća velika, pa će u cilju prevazilaženja lokalnog optimuma lošije rešenje biti prihvaćeno. Pred kraj izvršavanja algoritma verovatnoća prihvatanja lošijeg rešenja je jako mala i to se verovatno neće ni desiti, jer se smatra da je optimum dostignut ili se nalazi blizu najboljeg posećenog rešenja, pa se izbegava pogoršanje tekućeg rešenja.



Slika 1.1: Ilustracija metoda SA tokom izvršavanja [25]

Prirodno, algoritam SA sadrži parametar temperature i šemu hlađenja. Najčešće temperatura opada eksponencijalno s parametrom v iz intervala $(0, 1)$. Ako parametar v ima veoma malu vrednost hlađenje je prebrzo što može dovesti do upadanja u lokalni optimum (pucanje metala). U suprotnom ako je parametar v jako blizak jedinici hlađenje je jako sporo, pa će se pretraga pretvoriti u nasumičnu pretragu velikog prostora rešenja zbog velike verovatnoće da se u početku bolja rešenja zamenjuju lošijim. Iz tog razloga potrebno je odabrati parametar v tako da ima dovoljno veliku vrednost i da temperatura pri kraju procesa bude niska što dovodi do stabilizacije rešenja u jednom lokalnom području.

Pri svakoj temperaturi slučajno biranje rešenja iz okoline i njegova provera se izvršava više puta, a ne samo jednom, kako bi se rešenje stabilizovalo na tekućoj temperaturi. Opisani koraci u metodi simuliranog kaljenja mogu se predstaviti algoritmom 1.1.

```
x = x_0;
T = T_max;
ponavljaj{
  ponavljaj odredeni broj puta{
    generisanje slucajnog resenja x' u okolini resenja x;
    ΔE = f(x') − f(x);
    ako (ΔE ≤ 0){
      x = x';
    inace{
      prihvatanje resenja x' sa verovatnocom e^(−ΔE/T);
    }
  }
  T = T · v;
}(dok nije zadovoljen uslov zaustavljana);
```

Algoritam 1.1: Metoda simuliranog kaljenja

Metoda simuliranog kaljenja tokom svog izvršavanja može više puta da poseti jedno isto rešenje. Ova situacija se može izbegnuti hibridizacijom sa algoritmima koji koriste memorijske strukture za čuvanje posećenih rešenja o kojima će kasnije biti reči.

Pojava metode simuliranog kaljenja (1983) prouzrokovala je novi pristup pri rešavanju problema kombinatorne optimizacije i podstakla razvoj metaheurističkih algoritama. U kasnijim godinama predstavljeno je mnogo novih algoritama, najviše baziranih na prirodnim fenomenima, kao što su metoda zasnovana na ponašanju mrava (Ant Colony Optimization – ACO) [10], metoda zasnvana na ponašanju roja čestica (Particle Swarm Optimization – PSO) [11], metoda veštačkog imuniteta (Artificial Immune System – AIS) [20].

1.2 Tabu pretraga

Tabu pretraživanje (Tabu search - TS) predstavlja metaheuristiku zasnovanu na lokalnom pretraživanju koja koristi memoriju kako bi prevazišla zaglavljivanje u lokalnom optimumu. Metoda je predložena 1986. godine od strane Glovera [13] i detaljno opisana u [14] [15] [16], a zanimljivo je da je nezavisno iste godine Hansen predložio sličan pristup nazvan najbrži spust / najsporiji uspon [19].

Metoda tabu pretraživanja u svom nazivu sadrži reč "tabu"koja može da ukaže na karakteristike same metode. Reč "tabu" se koristi da označi zabranu i potiče iz Tonganskog jezika, gde su je Aboridžini na ostrvu Tonga koristili da označe predmete koji se ne smeju dodirivati zato što su sveti. Danas se reč "tabu" koristi da označi nešto o čemu nije društveno prihvatljivo otvoreno govoriti, odnosno nešto zabranjeno jer sadrži rizik. Kako metoda lokalnog pretraživanja sadrži rizik za dobijanje lokalnog optimuma, zabrana takvih rešenja se može koristiti kao mera zaštite. To je glavna ideja tabu pretrage koja rešenja označena kao "tabu" zanemaruju sa ciljem da se dostigne što bolje rešenje i izađe iz lokalnog optimuma. U realnom svetu tabu se menja kroz vreme za šta je odgovorna neka vrsta "društvene memorije". Kod tabu pretraživanja upravo memorija ima glavnu ulogu za dodeljivanje i menjanje tabu statusa kroz vreme pri tabu pretraživanju.

Pored informacija o trenutno najboljem rešenju i vrednosti funkcije cilja, TS metoda pamti kretanje preko rešenja iz nekoliko prethodnih iteracija, a kasnije se te informacije koriste za izbor narednog rešenja kao i za modifikaciju okoline koja neće sadržati tabu rešenja. Tako se oklonina rešenja menja kroz iteracije i zavisi od sadržaja memorije. Memorija u kojoj se čuvaju informacije o zabranjenim potezima naziva se tabu lista.

Metoda kreće od jednog rešenja, a zatim koristi metodu lokalne pretrage sve dok ne dostigne lokalni optimum. Kada se dostigne lokalni optimum rešenje koje ga predstavlja se pamti u tabu listi i isključuje iz okoline za pretraživanje u nekoliko narednih iteracija. Zatim se ponovo počinje sa lokalnim pretraživanjem, ali u tako modifikovanoj okolini. Nakon isteka tabu statusa nekog rešenja, ono se vraća u okolinu, a neka druga postaju zabranjena.

Osnovni koraci tabu pretraživanja mogu se predstaviti algoritmom 1.2.

```
x = x_0;
x_opt = x
TabuLista=∅
ponavljaj{
  pronaci najbolje resenje x' u okolini resenja x\TL;
  if(f(x') < f(x_opt){
    x_opt = x';
  }
  TL = TL ∪ x';
  if(|TL| > d){
    TL = TL\x^t;
  }
  s = x';
}(dok nije zadovoljen uslov zaustavljana);
```

Algoritam 1.2: Tabu pretraživanje

Kako je tabu lista glavni mehanizam za sprečavanje zaglavljivanja u lokalnom optimumu i kruženja u okolini nekog rešenja potrebno je posvetiti pažnju njenoj implementaciji i prilagoditi je konkretnom problemu. Kako bi se smanjila memorija potrebna za smeštanje rešenja u tabu listu često se pamte samo neka svojsta rešenja, tako da se u tabu listi mogu čuvati samo zabranjeni potezi koji dovode do zabranjenih rešenja. Moguće je koristiti i nekoliko tabu lista pa se u tom slučaju potez smatra zabranjenim ako se nalazi u svim listama, a inače je dozvoljen. Pri čuvanju poteza u tabu listi radi uštede prostora može doći do znatne restrikcije samog pretraživačkog prostora odnosno do zabrane svih rešenja koja imaju isto svojstvo kao i ono rešenje koje se zaista zabranjuje nekim potezom. Na taj način metoda je sprečena da poseti veliki skup rešenja, a ne samo ona koja su dobijena u nekoliko poslednjih iteracija. Pored toga, kruženje u okolini nekog rešenja nije u potpunosti sprečeno i u direktnoj je vezi sa parametrom koji određuje dužinu tabu liste. Ako je dužina tabu liste prevelika, izražena je diversifikacija pretraživanja što može dovesti do slučajnog kretanja u prostoru rešenja. Nasuprot tome, ako je dužina tabu liste suviše mala povećava se mogućnost kruženja u okolini nekog rešenja.

Da bi se prevazišli navedeni nedostaci ove metode, moguće je uvesti ukidanje tabu statusa nekom potezu. Jedan od najjednostavnijih načina je korišćenje tabu liste promenljive dužine. Drugi često korišćeni pristup je uvođenje praga značajnosti kojim se opisuju dobra rešenja, pa se tako na primer ukida tabu status potezu koji vodi u novo najbolje rešenje. Funkcija koja proverava prag značajnosti nekog rešenja naziva se aspiracijska funkcija.

Metoda tabu pretrage može se unaprediti uvođenjem različitih vrsta memorija. Tako se u procesu pretraživanja mogu koristiti kratkoročna, srednjoročna i dugoročna memorija pri čemu svaka ima svoje svojstvo. Kratkoročna memorija se koristi upravo za formiranje tabu liste i ima ulogu u sprečavanju kruženja u okolini nekog rešenja. Srednjoročna memorija ima

zadatak da obezbedi intenzifikaciju pretraživanja u nekom regionu u kome se nalazi trenutno najbolje rešenje. Svrha dugoročne memorije je diversifikacija pretrage odnosno prenošenje pretrage u regione pretraživačkog prostora koji nisu istraženi. I u srednjoročnoj i u dugoročnoj memoriji pamte se svojstva rešenja, a zatim se u procesu intezifikacije biraju rešenja koja imaju zajednička svojstva sa trenutno najboljim rešenjem dok se u procesu diversifikacije biraju rešenja kod kojih su ta svojstva različita od trenutno najboljeg rešenja. Sama ideja tabu pretraživanja da se za upravljanje pretragom koriste informacije dobijene iz prethodne pretrage može se povezati sa metodama informisanog pretraživanja u polju veštačke inteligencije.

Još jedno poboljšanje predložili su Battiti i drugi 1994. [4] tj. predložili su reaktivno tabu pretraživanje kako bi se izbegla velika zavisnost metode tabu pretraživanja od parametara koji određuju dužinu tabu liste, aspiracijsku funkciju, tabu vreme itd. Glavna ideja je da se automatski obavlja modifikacija parametara u potrazi za što boljim rešenjem.

Između metode tabu pretraživanja i simuliranog kaljenja postoji razlika u potrazi za najboljim rešenjem. Prva razlika je eksploatacija memorijskih struktura u algoritmu tabu pretraživanja, što je odsutno kod simuliranog kaljenja. Uvođenje memorijskih struktura podrazumeva i različitosti u samom mehanizmu pretrage koja se kod simuliranog kaljenja zasniva na slučajnosti, nasuprot pretrage TS algoritma. Tabu pretraživanje pretražuje celu tekuću okolinu (ili njen veći deo) da bi identifikovao poteze visokog kvaliteta. Ovo se razlikuje od simuliranog kaljenja gde se na slučajan način bira potez i na njega primenjuje unapred definisani kriterujum prihvatanja koji odlučuje da li će se taj potez izvršiti. Kriterijum prihvatanja poteza zanemaruje kvalitet ostalih mogućih poteza. Pomenuta razlika ukazuje na prednost tabu pretraživanja na polju pretrage okoline. Često se koristi hibridizacija u kojoj metoda simuliranog kaljenja generiše početno rešenje za metodu tabu pretrage. SA metoda zbog svoje prirode jednostavno može naći kvalitetne delove pretraživačkog prostora koje će TS metoda efikasno istražiti. I pored pozitivnog uticaja tabu liste u sprečavanju ciklusa, može se desiti da se jave ciklusi što predstavlja nedostatak tabu pretraživanja. To je slučaj kada je dužina ciklusa veća od dužine tabu liste (dužina tabu liste je osetljiv parametar TS algoritma o čemu je bilo reči). U takvim situacijama deterministička priroda tabu pretraživanja je nedostatak jer upravo ona otežava prevazilaženju ciklusa. U cilju prevazilaženja nedostataka TS algoritma, može se razmatrati njegova kombinacija sa SA. U takvoj kombinaciji stohastička karakteristika simuliranog kaljenja da slučajno bira rešenje sprečava pojavu ciklusa u pretrazi.

1.3 Metoda promenljivih okolina

Metoda promenljivih okolina (Variable Neighborhood Search – VNS) je metaheuristika zasnovana na lokalnom pretraživanju koja pokušava da prevaziđe lokalni optimum menjanjem strukture okoline. Predložena je od strane Mladenovića i Hansena [32], a prvi put ideja metode izložena je 1995. [31]. Osnovna ideja metode se bazira na sistematskoj promeni okolina unutar lokalnog pretraživanja. Zato je neophodno uvesti više okolina, promenom metrike u odnosu na koju se definiše ili povećavanjem rastojanja u odnosu na istu metriku.

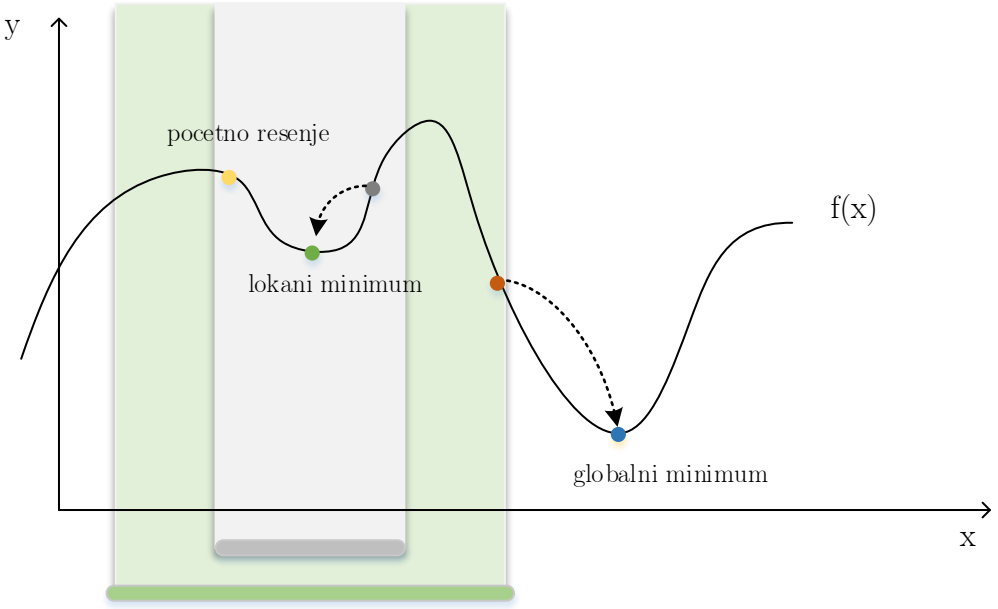
Metoda promenljivih okolina se oslanja na tri uočene činjenice [31] :

(1) Lokani optimum u jednoj okolini ne mora biti i lokalni optimum u nekoj drugoj okolini.

(2) Globalni optimum je lokalni optimum u odnosu na bilo koju okolinu.

(3) Lokalni optimumi u različitim okolinama su međusobno bliski za većinu problema.

Činjenice (1) i (2) se mogu uočiti na primeru funckije jedne promenljive, slika 1.2.



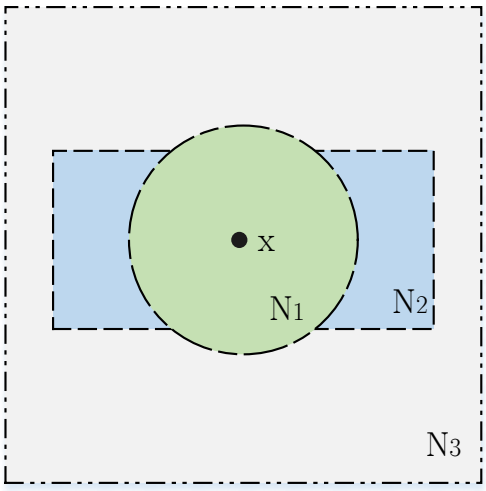
Slika 1.2: Promena okolina na primeru funkcije jedne promenljive

Uvođenje više okolina ima smisla imajući u vidu prve dve navedene činjenice. Treba obratiti pažnju da druga činjenica ne garantuje da rešenje koje je lokalni optimum u svim izabranim okolinama ujedno i globalni optimum, već ukazuje na to da ako neko rešenje nije lokalni optimum u nekoj okolini ono ne može biti ni globalni optimum. Treća činjenica je empirijska i podrazumeva da lokalni optimum često pruža neke informacije o globalnom optimumu pa je potrebno detaljnije istraživanje okoline lokalnog optimuma jer se tu očekuje popravljanje trenutno najboljeg rešenja. Ove tri navedene činjenice mogu se iskoristiti na tri različita načina deterministički, stohastički ili kombinovano.

1.3.1 Metoda promenljivog spusta

Korišćenjem determinističkog pristupa dobija se metoda promenljivog spusta (Variable Neighborhood Descent, VND). Prvi korak za realizaciju metode se sastoji u odabiru okolina N_k, $k = 1, 2, \ldots, k_{max}$. Metoda počinje sa radom od početnog rešenja x, a zatim se realizuje lokalno pretraživanje od rešenja x u odnosu na svaku od odabranih okolina i to po redosledu indeksa k. Specijalno za $k = 1$ dobija se obično lokalno pretraživanje.

Na slici 1.3 je prikazana upotreba više okolina, i to sa napomenom da okoline ne moraju biti ugnježdene, odnosno ne moraju biti izvedene iz iste metrike. Metoda VND pretražuje okolinu N_1 dok u njoj postoji mogućnost poboljšanja trenutno najboljeg rešenja. Kada se odredi lokalni optimum u odnosu na okolinu N_1, ažurira se najbolje tekuće rešenje i započinje lokalna pretraga u odnosu na okolinu N_2. Ako se popravi trenutno najbolje rešenje, lokalna pretraga ponovo počinje od okoline N_1 a inače se pretraga usmerava na narednu neispitanu okolinu. Metoda promenljivog spusta se završava ako trenutno najbolje rešenje ne može da se popravi ni u jednoj okolini, tj. ako je trenutno najbolje rešenje lokalni optimum u odnosu na sve okoline N_k, $k = 1, 2, \cdot, k_{max}$. Opisani koraci mogu se prikazati algoritmom 1.3.



Slika 1.3: Niz okolina za rešenje x

$x = x_0$;
$x_{opt} = x$;
ponavljaj{
  $k = 1$;
  ponavljaj{
    x'=LokalnaPretraga(x, N_k);
    ako($f(x') < f(x_{opt})$){
      $x_{opt} = x'$;
      $k = 1$;
    }
    onda{
      $k = k + 1$;
    }
  }(dok $k \neq k_{max}$)
}(dok nije zadovoljen kriterijum zaustavljanja)

Algoritam 1.3: Metoda promenljivog spusta

1.3.2 Redukovana metoda promenljivih okolina

U slučaju stohastičkog pristupa metoda se naziva redukovana metoda promenljivih okolina (Reduced Variable Neighborhood Search, RVNS). Kod ovog pristupa izostavlja se lokalno pretraživanje, pa se metoda sastoji samo od sistematske promene okoline i izbor jednog slučajnog rešenja u svakoj okolini. Metoda promenljivog spusta je, korisna kod rešavanja optimizacionih problema velike dimenzije, kao i za dobijanje dobrog početnog rešenja za kratko vreme jer ne sadrži složenu i dugotrajnu lokalnu pretragu.

Prvi korak za realizaciju se opet sastoji u odabiru okolina N_k, $k = 1, 2, \ldots, k_{max}$ i početnog rešenja x. Zatim se bira jedno slučajno rešenje u okolini $N_1(x)$ i ako se na taj način popravi trenutno najbolje rešenje ono postaje tekuće, u suprotnom se prelazi na izbor slučajnog rešenja u okolini N_2 itd. Kod prvog prihvatanja slučajnog rešenja, indeks okoline se postavlja na 1 za sledeću iteraciju. Niz okolina je potrebno pažljivo birati, tako da svaka sledeća okolina bude veće kardinalnosti od prethodne. Ako su sve okoline definisane u odnosu na istu metriku rastojanje između rešenja treba da raste u svakoj sledećoj okolini. Koraci RVNS metode mogu se predstaviti pseudokodom 1.4.

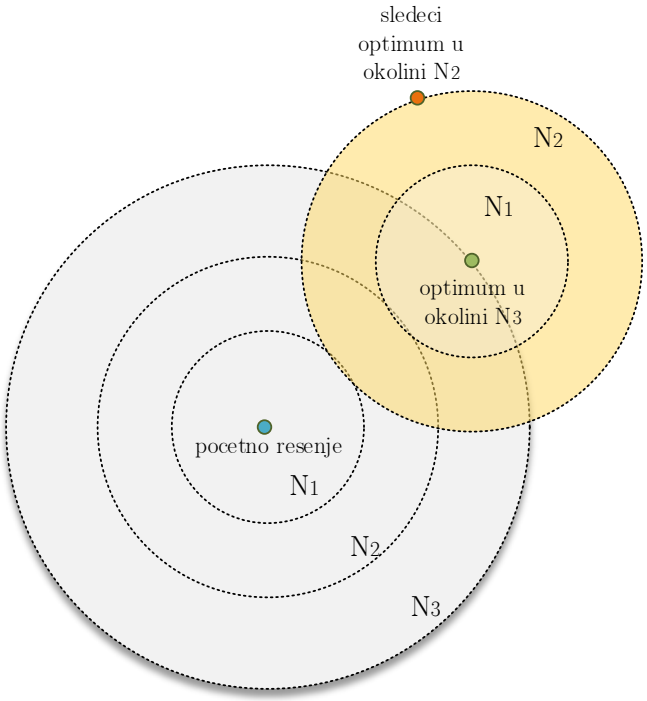
```
x = x_0;
x_opt = x;
ponavljaj{
  k = 1;
  ponavljaj{
    x'=SlucajnoResenje(N_k);
    akp(f(x') < f(x_opt)){
      x_opt = x';
      k = 1;
    }
    onda{
      k = k + 1;
    }
  }(dok k ≠ k_max)
}(dok nije zadovoljen kriterijum zaustavljanja)
```

Algoritam 1.4: Redukovana metoda promenljivih okolina

1.3.3 Osnovna metoda promenljivih okolina

Kombinacijom prethodna dva principa, tj. sistematskom (deterministčkom) promenom okolina, slučajnim izborom početnog rešenja u tekućoj okolini i primenom lokalne pretrage počev od tog, slučajnog rešenja, dobija se osnovna metoda promenljivih okolina (Basic Variable Neighborhood Search, BVNS). Ovo je najrasprostranjenija varijanta metode promenljivih okolina jer obezbeđuje više preduslova za dobijanje kvalitetnijih konačnih rešenja.

Osnovna verzija metode promenljivih okolina polazi od jednog početnog rešenja, u svakoj iteraciji u okolini tekućeg rešenja na slučajan način se bira susedno rešenje koje postaje polazno rešenje za lokalno pretraživanje. Lokalno pretraživanje se izvršava do pronalaska lokalnog optimuma. Ukoliko je lokalni optimum bolji od tekućeg rešenja, onda on postaje novo tekuće rešenje. Ako to nije slučaj, proširuje se okolina tekućeg rešenja i u njoj se ponovo slučajno bira susedno rešenje. Kada se popravi trenutno najbolje rešenje ceo postupak se ponavlja od prve okoline u nizu. Primer rada metode ilustrovan je na slici 1.4.



Slika 1.4: Osnovna metoda promenljivih okolina

Opisani koraci se mogu predstaviti pseudokodom 1.5.

```
x = x_0;
x_opt = x;
stop = 0;
ponavljaj{
  k = 1;
  ponavljaj{
    x'=SlucajnoResenje(N_k);
    x''=LokalnaPretraga(x');
    ako(f(x'') < f(x_opt)){
      x_opt = x'';
      k = 1;
    }
    inace{
      k = k + 1;
    }
    ako(zadovoljen kriterijum zaustavljana){
      stop = 1;
    }
  }(dok k ≠ k_max || stop ≠ 1)
}(dok nije zadovoljen kriterijum zaustavljanja)
```

Algoritam 1.5: Osnovna metoda promenljivih okolina

Lokalnog pretraživanja u VNS metodi u nekim slučajevima može biti neefikasno. Tada je moguće upotrebiti tabu pretraživanja umesto lokalnog pretraživanja jer tabu pretraživanje pomoću svojih memorijskih struktura može prevazići prvi lokalni optimum koji zaustavlja obično lokalno pretraživanje. Na taj način se smanjuje verovatnoća da novo rešenje bude lošije od tekućeg rešenja. To dalje smanjuje potrebu za čestom promenom okoline, što može povoljno uticati na vreme izvršavanja programa u nekim slučajevima. Izbor lošeg susednog rešenja u tekućoj okolini utiče na efikasnost izvršavanja faze pretrage. U [33] predloženo je da se elementi tabu pretraživanja uključe u proces pretrage i u proces biranja susednog rešenja u tekućoj okolini sa ciljem popravke efikasnosti VNS metode. Predloženi algoritam nazvan je Variable Neighborhood Tabu Search – VNTS.

1.4 Optimizacija rojevima čestica

Optimizacija rojevima čestica (Particle swarm optimization - PSO) predstavlja metodu inspirisanu inteligencijom grupe. Posmatrajući prirodni proces udruživanja jedinki u grupe, može se primetiti da bespomoćne jedinke postaju ozbiljan protivnik kada se udruže. Na taj način akumuliraju inteligenciju stvarajući kolektivnu inteligenciju, pa interakcijom i razmenom znanja između jedinki cela grupa veoma brzo napreduje. U to posmatranjem jata ptica u potrazi za hranom kao skupa čestica uočeno je da svaka čestica tj. ptica koristi nekoliko pravila za prilagođavanje svog leta kao što su:

- izbegavanje sudara,
- prilagođavanje brzine leta,
- pokušaj ostanka u blizini ostalih ptica.

Inspirisani ovim uočenim činjenicama, Eberhart i Kennedy [11] dolaze na ideju kreiranja algoritma za rešavanje optimizacionih problema. U sam algoritam uključuju i sociološku interekciju između čestica u roju, tako da svaka čestica pamti svoje do tada pronađenu najbolju poziciju, ima uvid u najbolje pozicije svojih suseda i kretanje usmerava uzimajući u obzir obe komponente.

U PSO metodi svaka čestica odgovara jednom potencijalnom dopustivom rešenju i kreće se u d-dimenzionom dopustivom prostoru rešenja. Čestice pokušavaju da poprave svoju poziciju koristeći svoje iskustvo iz prethodnih pozicija, ali i iskustva drugih čestica (celog roja ili podskupa-okoline čestice). Poziciju čestice i, $i = 1, 2 \ldots, n$ u roju od n čestica karakteriše d-dimenzioni vektor položaja $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ i vektor brzine, odnosno gradijenta (pravca) u kome bi se čestica kretala bez drugih uticaja $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$. Najbolju poziciju čestice i do tekućeg trenutka predstavlja vektor $p_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ a najbolju poziciju celog roja vektor $p_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$.

Osnovni PSO algoritam se sastoji od dela inicijalizacije gde se generiše roj čestica u pretraživačkom prostoru na slučajnim pozicijama i sa slučajnim pravcima kretanja čestica. U tako generisanom roju potrebno je pronaći i sačuvati najbolju česticu. Zatim sve dok nije ispunjen kriterijum zaustavljanja izvršava se ažuriranje vektora brzine i pomeranje čestica na nove pozicije. U tako dobijenom roju čestica ažuriraju se najbolje pozicije čestica ako je rešenje popravljeno kao i najbolja čestica celog roja ako je došlo do popravljanja trenutno najboljeg rešenja.

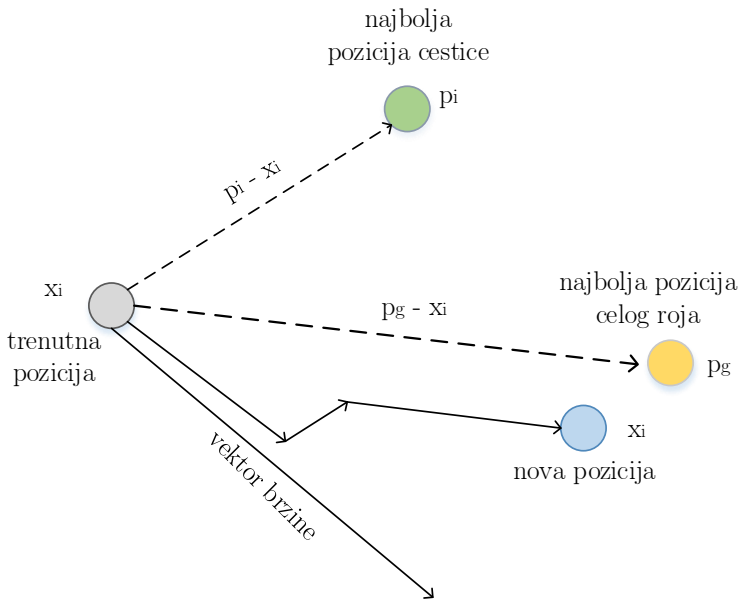
U svakoj iteraciji j ažuriranje vektor brzine čestice i se obavlja pomoću sledeće formule:

$$v_{id}^{j} = w \cdot v_{id}^{j-1} + c_1 \cdot r_1 \cdot (p_{id}^{j} - x_{id}^{j}) + c_2 \cdot r_2 \cdot (p_{gd}^{j} - x_{id}^{j}),$$

gde je w parametar inercije koji kontroliše uticaj prethodnih brzina čestice na trenutnu, c_1 faktor kognitivnog učenja (uticaj iskustva čestice), c_2 faktor socijalnog učenja (uticaj iskustva celog roja) i r_1, r_2 slučajne konstante iz izabrane uniformne raspodele $U[0, 1]$. Pomeranje čestice i na novu poziciju X_i u iteraciji j se vrši po formuli:

$$x_{id}^{j+1} = x_{id}^{j} + v_{id}^{j}$$

Vrednosti v_{id} se ograničavaju na vrednosti iz unapred zadatog segmenta $[V_{min}, V_{max}]$, kako čestica ne bi izašla iz dopustivog prostora pretrage. Odnos između lokalnog i globalnog pretraživanja prostora dopustivih rešenja se može kontrolisati parametar w. Za veće vrednosti w pojačava se globalna pretraga, dok se za manje vrednosti w pojačava lokalna pretraga.



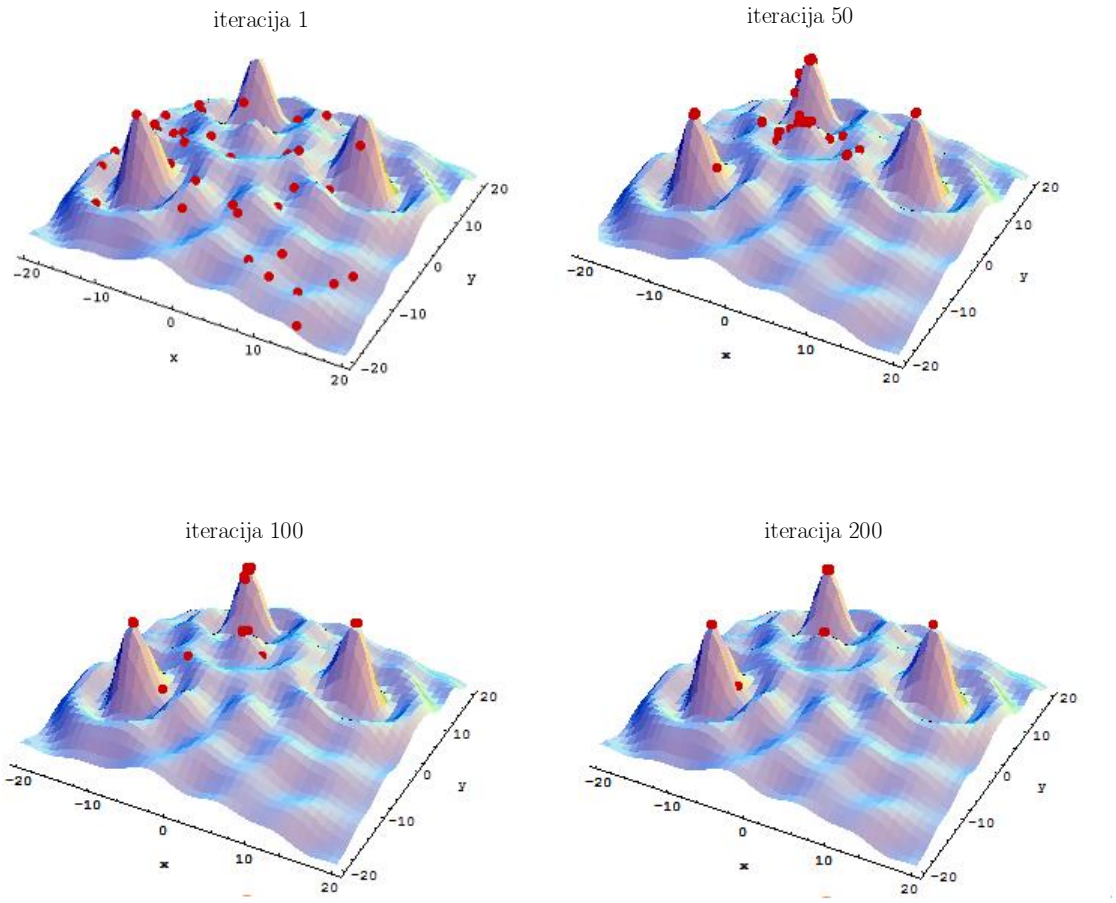
Slika 1.5: Pomeranje čestice na novu poziciju [40]

Opisani koraci se mogu predstaviti algoritmom 1.6.

```
slucajno generisi roj cestica;
ponavljaj{
  f_i = f(X_i);
  za sve cestice i{
    za svaku koridnatu d{
      v_id = w · v_id + c_1 · r_1 · (p_id − x_id) + c_2 · r_2 · (p_gd − x_id),
      x_id = x_id + v_id
    }
    ako (f(X_i) < f(p_i)) onda{
      p_i = X_i;
    }
    ako (f(X_i) < f(p_g)){
      p_g = X_i;
    }
  }
}(dok nije zadovoljen kriterijum zaustavljanja)
```

Algoritam 1.6: Optimizacija rojevima čestica

Na slici 1.6 prikazane su pozicije čestica pri traženju maksimalne vrednosti funkcije dve promenljive.



Slika 1.6: Primena PSO algoritma za pronalaženje maksimuma funkcije dve promenljive

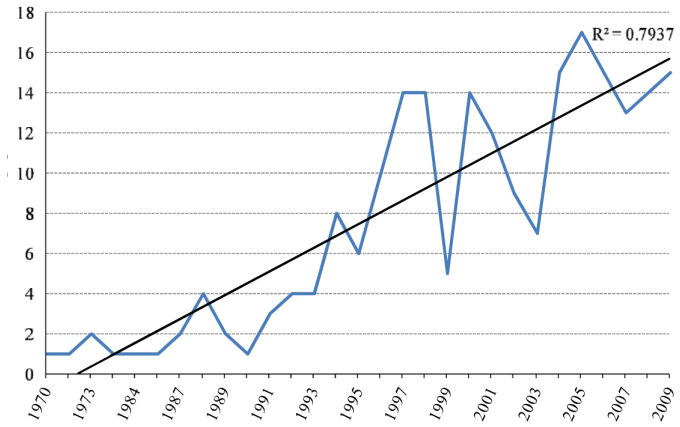
2

Problem višefazne proizvodnje sa paralelnim mašinama

Problem raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama (Hybrid flow shop - HFS) je nastao kombinacijom klasičnog problema višefazne proizvodnje (Flow shop problem) gde svaki posao prolazi redom kroz nivoe u kojima je dostupna jedna mašina koja obrađuje posao i problema proizvodnje sa paralelnim mašinama (Job shop scheduling) gde postoji jedna faza proizvodnje u kojoj je dostupan određen broj identičnih mašina koje rade paralelno. Oba pojedinačna problema su veoma dobro istražena u teoriji raspoređivanja i opisan u velikom broju radova, međitim ne i njihova kombinacija. Postoji nekoliko kriterijuma za ocenu kvaliteta rasporeda poslova, a najčešći je ukupno vreme izvršavanje svih poslova (makespan) zbog praktičnog značaja. Pokazano je da HFS problem pripada klasi NP teških problema kada je cilj minimizovati ukupno vreme izvršavanja svih poslova [18]. Problem trgovačnog putnika može se posmatrati kao specijalni slučaj problema proizvodnje sa paralelnim mašinama gde trgovački putnik odgovara mašini, a gradovi odgovaraju poslovima [26].

Problem je prvi put formulisan 1971. godine [3], a postao je jako aktuelan od 1997. godine kada je broj radova naglo porastao. Ukupno postoji preko 200 radova u kojima se istražuje HFS problem ili neke njegove varijante. Uz to u 54% radova izučava se problem proizvodnje u m nivoa, dok olakšani HFS problem sa 2 nivoa je izučavan u 31.63% radova. [39]. Detaljan prikaz broja objavljenih radova o problemu višefazne proizvodnje sa paralelnim mašinama tokom prethodnih godina prikazan je na slici 2.1

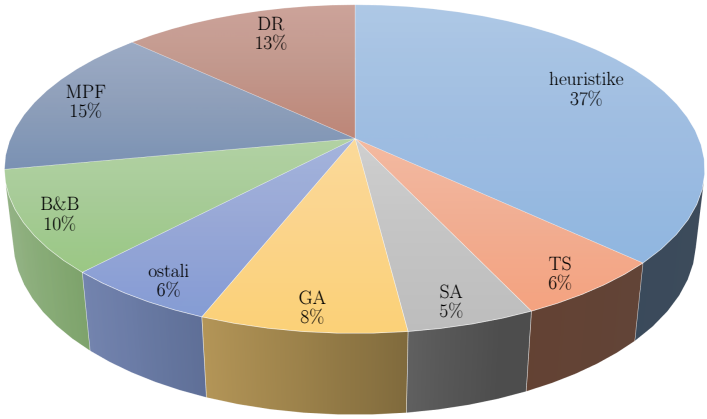
U literaturi se mogu pronaći različiti pristupi za rešavanje HFS problema, počev od egzaktnih preko heurističkih i metaheurističkih algoritama. Najčešće korišćena egzaktna metoda je algoritam grananja i odsecanja. Brah i Hunsucker u svom radu [6] opisali su algoritam grananja i odsecanja dok su Portman i Vignier [37] predstavili poboljšanje bazirano na uvođenju genetskog algoritma u egzaktni rešavač. Neron i drugi autori [34] dali su predlog korišćenja algoritma energetskog rezonovanja i globalnih operatora za poboljšanje efikasnosti algoritma grananja i odsecanja. Pored egzaktnih metoda razvijeno je i nekoliko heurističkih algoritama. Gupta [18] je razvio heuristiku za pronalaženje minimalnog vremena proizvodnje kod dvofaznog HFS problema sa jednom mašinom u drugom nivou. Kahraman i drugi autori [22] su predstavili efikasan paralelni pohlepni algoritam za modifikovani HFS sa višeprocesorskim poslovima. Poslednjih godina metaheuristike predstavljaju najpopularniji pristup za rešavanje problema višefazne proizvodnje sa paralelnim mašinama. Janiak i Kozan [21] primenili su algoritam simuliranog kaljenja i tabu pretragu za rešavanje HFS problema sa ciljem minimi-



Slika 2.1: Broj radova o HFS problemu u periodu od 1971. godine do 2009. godine

zovanja troškova. Genetski algoritam je primenjen na HFS problem sa ciljem minimizacije vremena proizvodnje od strane više autora [2, 5, 36]. Zastupljenost metoda koje su korišćene za rešavanje HFS problema prikazana je na slici 2.2 (MPF - matematičko programiranje i formulacija, DR - pravila slanja).

Poznat skup instanci za HFS problem generisan je od strane Carliera i Nerona [7], i korišćen je za testiranje nekih metaheuristika kao što su veštački imuni sistem (AIS) [12, 35] genteski algoritmi (GA) [5], optimizacija mravljim kolonijama(ACO) [1, 23].



Slika 2.2: Zastupljenost metoda za rešavanje HFS problema

U preko 60% radova vezanih za HFS problem funkcija cilja predstavlja minimizaciju ukupnog vremena proizvodnje upravo zbog njenog praktičnog značaja, dok su ostale ocene kvaliteta rasporeda poslova zastupljene u veoma malim procentima.

2.1 Opis problema

Problem raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama (Hybrid flow shop (HFS) scheduling problem) se odnosi na optimizaciju procesa prioizvodnje, odnosno raspoređivanje n poslova na određenom skupu mašina grupisanih u k nivoa, gde svaki nivo odgovara jednoj fazi proizvodnog procesa. Na svakom nivou s, $s = 1, 2, \ldots, k$, raspoloživo je m_s identičnih paralelnih mašina, pri čemu je $m_s \geq 2$ na bar jednom nivou s. Svaki posao $j = 1, 2, \ldots n$ treba da bude obrađen na jednoj od mašina na svakom nivou, pri čemu ni jedna mašina ne može da obrađuje više od jednog posla istovremeno, i u jednom trenutku posao može da se obrađuje na tačno jednoj mašini. Kada se počne sa obradom jednog posla, posao se kompletira bez prekida. Pripremno vreme za eventualno podešavanje mašine je uključena u vreme obrade posla i nema ograničenja skladištenja između nivoa. Cilj HFS problema je rasporediti poslove po mašinama na svakom nivou i odrediti redosled poslova koje treba obaviti na svakoj mašini, tako da ukupno vreme proizvodnje, odnosno vreme završetka poslednjeg posla bude minimalno.

2.2 Matematička formulacija

Matematička formulacija problema [34] višefazne proizvodnje sa paralelnim mašinama zahteva uvođenje sledeće notacije:

j - indeks poslova
s - indeks nivoa
i - indeks mašina
S_{js} - vreme početka posla j u nivou s
F_{js} - vreme završetka posla j u nivou s
P_{js} - vreme trajanja posla j u nivou s
m_s - broj raspoloživih mašina u nivou s

$$X_{jis} = \begin{cases} 1, & \text{posao } j \text{ dodeljen mašini } i \text{ u nivou } s \\ 0, & \text{inače} \end{cases}$$

$$Y_{jgs} = \begin{cases} 1, & \text{posao } j \text{ prethodi poslu } g \text{ u nivou } s \\ 0, & \text{inače} \end{cases}$$

Korišćenjem navedene notacije, opisani problem se može formulisati kao problem celobrojnog linearnog programiranja:

$min \quad C_{max}$ pri uslovima:

$$C_{max} \geq F_{js},\ s = 1, \ldots, k,\ j = 1, 2, \ldots, n \tag{2.1}$$

$$F_{js} = Sjs + Pjs,\ s = 1, \ldots, k,\ j = 1, 2, \ldots, n \tag{2.2}$$

$$\sum_{i=1}^{m_s} X_{ijs} = 1,\ s = 1, 2, \ldots, k,\ j = 1, 2, \ldots, n \tag{2.3}$$

$$F_{js} \leq S_{j(s+1)},\ s = 1, 2, \ldots, k-1 \tag{2.4}$$

$$S_{xs} \geq F_{ys} - LY_{xys},\ \forall(x, y),\ s = 1, 2, \ldots, k \tag{2.5}$$

$$X_{jis} \in \{0, 1\}, Y_{jis} \in \{0, 1\},\ s = 1, 2, \ldots, k,\ i = 1, 2 \ldots, m_s,\ j = 1, 2, \ldots, n \tag{2.6}$$

Uslovi (2.1) i (2.2) definišu maksimalno vreme trajanja svih poslova. Uslov (2.3) obezbeđuje da se svaki posao u svakom nivou izvršava na samo jednoj mašini. Uslov (2.4) garantuje da se posao može započeti u tekućem nivou samo ako je njegova obrada u prethodnom nivou završena. Uslovi (2.4) i (2.5) obezbeđuju da svaka mašina može da izvršava samo jedan posao u jednom trenutku i da se tekući posao može započeti samo ako je posao koji mu prethodi završen. Uslovom (2.6) definišemo binarne promenljive problema.

Problem višefazne proizvodnje sa paralelnim mašinama se smatra osnovnim problemom za modeliranje realnih automatizovanih industrijskih postrojenja. U osnovni model mogu se uključiti dodatna ograničenja, na primer kapacitet skladišta između dve faze proizvodnje razmatranje skladištenje ili vreme neophodno za pauzu između obrade dva posla. Tekstilne kompanije dosta pažnje posvećuju proučavanju problema višefazne proizvodnje upravo zbog postupka obrade svojih proizvoda. Odeća se obrađuje u uzastopnim proizvodnim postrojenjima sa većim brojem identičnih mašina. Operacije (poslovi) koji se često javljaju su pletenje, tkanje, sečenje i bojenje tkanina.

Primene se mogu naći u građevinskoj industriji pri proizvodnji betosnkih blokova [17]. Betonski blokovi se proizvode po nalogu klijenta u nekoliko veličina i od nekoliko vrsta betona. Vrsta betona utiče na jačinu bloka, težinu, mogućnost toplotne izolacije kao i njegovu cenu, dok veličina bloka zavisi od njegove buduće upotrebe. Blokovi se dobijaju korišćenjem kalupa različitih veličina ali ni broj ni vrsta kalupa nisu strogo određeni već zavise od naloga klijenta. Svaki kalup odgovara jednom poslu, koje prolazi kroz uzastopnih osam faza u sledećem redosledu:

(1) priprema komponenti,

(2) sjedinjavanje komponenti i punjenje kalupa,

(3) učvršćivanje betona u kalupu,

(4) odsecanje blokova od formiranog betona,

(5) kontrola kvaliteta,

(6) finalno učvršćivanje blokova na visokoj temperaturi i pod visokim pritiskom,

(7) dostavljanje blokova u magacin,

(8) transprtovanje blokova klijentu.

Faze 2, 3, 4 su osetljive i završeni posao ne sme čekati na prelazak u sledeću fazu, dok između faza 6 i 7 ne postoji privremeno skladište. Cilj je minimizovati potrebno vreme proizvodnje. Matematički model ovako opisanog problema se lako može dobiti iz prethodno navedenog matematičkog modela HFS problema dodavanjem jednog ograničenja:

$$S_{js} - F_{j(s-1)} = 0, \quad s = 2, 3, 4$$

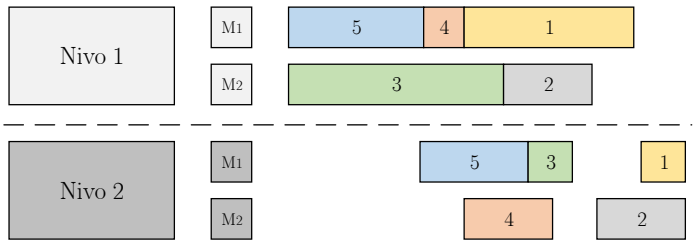
koje upravo opisuje da završeni posao ne sme čekati na prelazak u sledeću fazu u nivoima 2,3,4. Slične primene se mogu naći u elektronskoj i papirnoj industriji kao i u neproizvodnim oblastima kao što su internet servisi i upravljački sistemi.

3

Rešavanje HFS problema metaheurističkim algoritmima

U prvom poglavlju predstavljeni su osnovni koncepti metaheurističkih metoda koje će biti korišćene za rešavanje problema višefazne proizvodnje sa paralelnim mašinama. U ovom poglavlju biće izloženi detalji implementacije svake metode. Za potrebe svih metaheurističkih algoritama potrebno je definisati pretraživački prostor izborom kodiranja rešenja. Način kodiranja dopustivih rešenja, kao i azbuka simbola nad kojom se ono vrši predstavljaju važan korak u implementaciji metaheurističkih metoda. Univerzalni način kodiranja ne postoji tako da tehnike kodiranja variraju od problema do problema. Kodiranje treba da omogući da se sa što manjim brojem simbola i što prirodnije izraze glavne karakteristike rešenja.

Kod HFS problema traži se raspored poslova kojim ih treba obaviti na svakoj mašini u svakom nivou, tako da ukupno vreme proizvodnje bude minimalno. Raspored poslova koji treba obaviti u nekom nivou se može predstaviti kao niz celih brojeva, pa se za kodiranje rešenja može odabrati permutacija skupa indeksa n poslova koja označava redosled dolaska poslova na prvu slobodnu mašinu u prvom nivou.



Slika 3.1: Dekodiranje rešenja

Dekodiranje rešenja se može predstaviti na jednostavnom primeru HFS problema sa 5 poslova i 2 nivoa. Oba nivoa sadrže po dve identične paralelne mašine. Na slici 3.1 je prikazano rešenje (5, 3, 4, 1, 2) i raspored izvršavanja poslova pri dolasku poslova u navedenom redosledu na prvu slobodnu mašinu. Sekvenca (3, 5, 1, 3, 4) ne predstavlja rešenje HFS problema u datom kontekstu jer sadrži posao 3 na dve pozicije tj. ne predstavlja ispravnu permutaciju skupa poslova $\{1, 2, 3, 4, 5\}$ gde se pod permutacijom podrazumeva niz koji sadrži svaki element datog konačnog skupa jednom i samo jednom.

Kod svih metaheuristika mogu se koristiti različiti kriterijumi zaustavljanja u zavisnosti od potrebe i načina testiranja. Često se kombinuju dva ili više kriterijuma čime se smanjuje mogućnost da se loše proceni kada treba prekinuti algoritam. Kriterijumi zaustavljanja koji su korišćeni u metaheurističkim algoritmima pri rešavanju HFS problema su ponavljanje najboljeg rešenja maksimalni unapred zadati broj puta, dostizanje optimalnog rešenja (ako je ono unapred poznato) i ograničeno vreme izvršavanja algoritma.

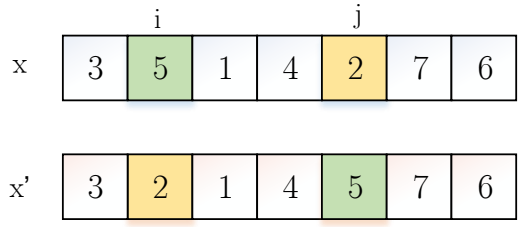
3.1 Metoda simuliranog kaljenja

Početno rešenje

Generisanje početnog rešenja se vrši kreiranjem sekvence indeksa poslova $x = (1, 2, 3, \ldots, n)$, gde je n broj poslova i primene funkcije koja vrši zamene na slučajno odabranim pozicijama u nizu x. Tako dobijeno početno rešenje i dalje predstavlja permutaciju indeksa poslova i predstavlja dopustivo rešenje za HFS problem.

Okolina rešenja

Okolina sekvence x se dobija zamenom mesta poslova na poziciji i i j. Ovakva transformacija rešenja je složenosti $O(1)$ i izvršava se veoma brzo. Za fiksiranu temperaturu T generisanje slučajnog rešenja x' u okolini rešenja x se izvršava k puta kako bi se obezbedila stabilizacija rešenja na tekućoj temperaturi. Postupak dobijanja u okolini rešenja x prikazano je na slici 3.2.



Slika 3.2: Zamena poslova na pozicijama i i j

Verovatnoća prihvatanja lošijeg rešenja

Ukoliko je došlo do poboljšanja trenutno najboljeg rešenja, rešenje x' se prihvata i koristi za sledeću iteraciju. Ako nije došlo do poboljšanja najboljeg trenutnog rešenja potrebno je sa određenom verovatnoćom prihvatiti rešenje x'. Verovatnoća treba da zavisi od temperature, tako da pri visokoj temperaturi verovatnoća prihvatanja lošijeg rešenja bude veća, dok pri niskim temperaturama ta verovatnoća bude bliska 0. Osim toga, za jako loša rešenja verovatnoća prihvatanja treba da bude mala za razliku od rešenja koja ne odstupaju puno od tekućeg rešenja. Iz navedenih razloga, verovatnoća prihvatanja se računa pomoću odstupanja od tekućeg rešenja $\Delta = f(x) - f(x')$ i tekuće temperature T uz primenu eksponencijalne funkcije $(\alpha)^{-\frac{\Delta}{T}}$, gde je $0 < \alpha < 1$. Broj dobijen na ovaj način pripada segmentu $[0, 1]$ jer su odstupanje Δ i temperatura T pozitivni brojevi pa predstavlja verovatnoću koja ima gore navedena svojstva. Koraci metode simuliranog kaljenja prikazani su pseudokodom 1.1.

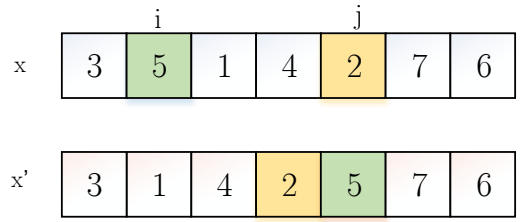
3.2 Tabu pretraga

Početno rešenje

Početno rešenje se dobija na identičan način kao kod metode simuliranog kaljenja tj. ukoliko ono nije prosleđeno vrši se kreiranje sekvence indeksa poslova $x = (1, 2, 3, \ldots, n)$, gde je n broj poslova uz primenu funkcije koja vrši zamene na slučajno odabranim pozicijama u nizu x.

Lokalna pretraga

Lokalna pretraga se izvršava u odnosu na okolinu rešenja x, koja se dobija uklanjanjem posla sa jedne pozicije i i njegovo umetanje na neku drugu poziciju j u sekvenci x. Lokalna pretraga se ne završava pri prvom poboljšanju rešenja već se vrši potpuna pretraga okoline. (Best improvement). Postupak dobijanja rešenja x' iz okoline rešenja x prikazano je na slici 3.3.



Slika 3.3: Umetanje posla i na poziciju j

Tabu lista

Važan deo implementacije tabu pretraživanja je upravo tabu lista čiji mehanizam omogućava algoritmu prevazilaženje lokalnog optimuma. Kako se u lokalnoj pretrazi vrši pomeranje posla p sa pozicije i na poziciju j, tabu potezom možemo proglasiti vraćanje posla p sa pozicije j na prethodnu poziciju i što predstavlja uređenu trojku (p, i, j). Može se primetiti da ovaj tip tabu poteza ne ograničava pretragu u velikoj meri, ali i da se ciklusi mogu pojaviti ako se posao p premesti na poziciju k a zatim sa pozicije k na poziciju i. Stroži tabu bi podrazumevao vraćanje posla p na poziciju i (bez razmatranja njegove trenutne pozicije) a to se može predstaviti kao uređeni par (p, i), dok se tabu koji u potpunosti onemogućava pomeranje posla p sa svoje nove pozicije, može jednostavno označiti sa (p). Tabu lista je realizovana heš mapom koja potez (p, i) slika u broj koji predstavlja na koliko iteracija je dati potez zabranjen. Pri dodavanju tabu poteza u listu broj zabranjenih iteracija se određuje na slučajan način iz skupa $[D_{min}, D_{max}] \bigcap \mathbb{N}$.

3.3 Metoda promenljivih okolina

Početno rešenje

Za dobijanje početnog rešenje VNS metode koristi se redukovana metoda promenljivih okolina na, koja se sastoji u slučajnom izboru rešenja u nizu okolina N_k. Okolina N_k sekvence x se dobija postupkom umetanja posla sa pozicije i na poziciju j i to k puta (slika 3.3) Sve okoline su definisane u odnosu na istu metriku, ali sa različitim rastojanjem. Ovakav tip okoline omogućava konstrukciju dobre sekvence u početnim koracima.

Okolina rešenja

Za realizaciju VNS algoritama i definisanje okoline korišćeno je pet tipova okolina tj. pet vrsti transformacije koje se sistematski menjaju tokom rada algoritma. Osnovna okolina N_k sekvence x dužine n dobija se na jedan od sledećih načina:

(1) invertovanjem indeksa poslova između dve pozicija i i j, pri čemu je $|j-i| > 2$ (slika 3.4).

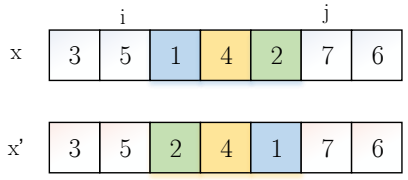
(2) premeštanjem podsekvence od pozicije 0 do pozicije i na kraj sekvence x i premeštanjem podsekvence od pozicije j do pozicije n na početak sekvence x (slika 3.5).

(3) umetanjem indeksa posla sa pozicije i na poziciju j (slika 3.3).

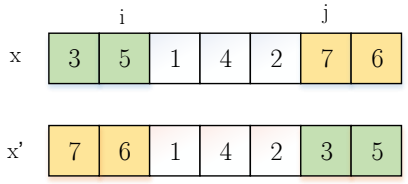
(4) zamenom indeksa poslova na pozicijama i i j (slika 3.2)

(5) zamenom indeksa poslova na pozicijama i_1 i j_1 i umetanjem posla sa pozicije i_2 na poziciju j_2 (slika 3.6).

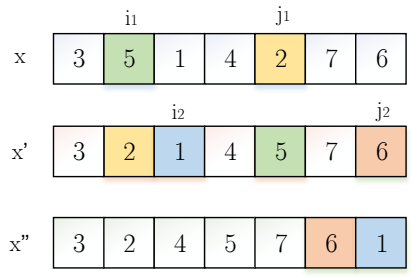
Svaka osnovna okolina N_k se sastoji od niza okolina koje se dobijaju izvršavanjem l transformacija istog tipa, i takvu okolinu označavamo $N_{k,l}$. Na primer, okolina $N_{3,4}$ se dobija četiri uzastopnim transformacijama zamene poslova i i j. Primenom navedenih transformacija na rešenje x, dobijeno rešenje je uvek dopustivo.



Slika 3.4: Invertovanje sekvence poslova



Slika 3.5: Zamena podsekvenci poslova



Slika 3.6: Kompozicija zamene i umetanja

Lokalna pretraga

U literaturi se često navodi važnost brzine lokalne pretrage kod VNS algoritma. Iz tog razloga lokalna pretraga koristi tip okoline (4) i izvršava se do prvog poboljšanja (First improvement) za razliku od lokalne pretrage tabu pretraživanja koja je implementirana kao potpuna pretraga.

3.4 Optimizacija rojevima čestica

Početna populacija

Čestice u početnoj populaciji se kreiraju od inicijalne sekvence $x = (1, 2, 3, \ldots, n)$ na koju se primenjuje funkcija koja vrši zamene na slučajno odabranim pozicijama. Na slučajan način za svaku česticu generiše se mutacioni faktor koji simulira vektor brzine čestice. Nakon generisanja čestica, pronalazi se najbolja čestica p_b.

Pomeranje čestica

Ažuriranje pozicije čestice se vrši korišćenjem tri operatora. Prvi operator O_1 usmerava česticu u pravcu vektora brzine, i realizovan je kao operator mutacije. Operator O_2 koristi iskustvo čestice za njeno usmeravanje i realizovan je kao operator ukrštanja trenutne pozicije X sa najboljom pozicijom čestice p_i, dok operator O_3 koristi iskustvo roja čestica i realizovan je kao operator ukrštanja trenutne pozicije X sa najboljom pozicijom celog roja p_b.

$$X_i = \underbrace{c_2 O_3(\underbrace{c_1 O_2(\overbrace{wO_1(X_i)}^{\text{kretanje u pravcu}}, p_i)}_{\text{kognitivno učenje}}, p_b)}_{\text{socijalno učenje}}$$

Operator mutacije

Na svaku česticu roja se prvo primenjuje operator mutacije koji predstavlja kretanje čestice u pretraživačkom prostoru bez drugih uticaja. Faktor izvršavanja mutacije se reguliše parametrom w. U algoritam su ugrađena dva tipa mutacije, svaki tip se realizuje sa verovatnoćom $\frac{1}{2}$. U oba slučaja data je sekvenca poslova x i dve pozicije i i j u sekvenci x. Okolina sekvence x je definisana u zavisnosti od tipa mutacije:

- **Zamena.** Okolina sekvence x se dobija zamenom mesta poslova na poziciji i i j. Ovakav tip mutacije je potreban kada je poznato dobro rešenje i zahteva male promene za poboljšanje (slika 3.2).
- **Insercija.** Okolina sekvence x se dobija umetanjem posla sa pozicije i na poziciju j. Ovakav tip mutacije diverzifikuje pretragu i omogućava konstrukciju dobre sekvence u početnim koracima algoritma (slika 3.3).

Operator ukrštanja

Na sekvencu dobijenu primenom operatora mutacije dva puta se primenjuje operator ukrštanja [28], jednom u svrhu kognitivnog, a drugi put u cilju socijalnog razvoja čestice. Kognitivni razvoj se može dobiti uzimanjem podsekvence poslova iz najbolje poznatog rešenja u kome se čestica nalazila, dok se socijalni razvoj može dobiti uzimanjem podsekvence iz najboljeg rešenja celog roja. Upravo iz tih razloga socijalno i kognitivno učenje se realizuju operatorom ukrštanja dve sekvence X_1 i X_2 koji se izvršava u četiri koraka:

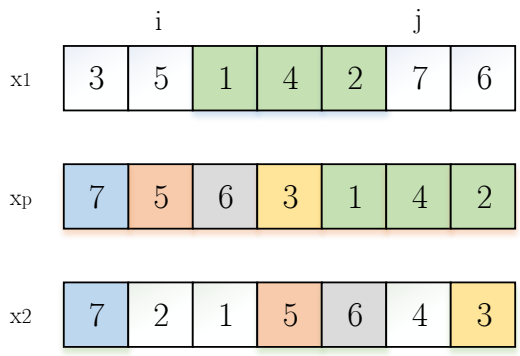
(1) Kreiranje prazne sekvence X_p,

(2) Biranje dve slučajne pozicije i i j u prvoj sekvenci poslova X_1,

(3) Kopiranje podsekvence poslova između dve izabrane pozicije iz koraka 1 na levi ili desni kraj sekvence X_p,

(4) Prazne pozicije u sekvenci poslova X_p se popunjavaju redom poslovima iz druge roditeljske sekvence X_2.

Rezultat sprovedenog ukrštanja sekvenci X_1 i X_2 se nalazi u sekvenci X_p. Primer ukrštanja dve sekvence dat je na slici 3.7



Slika 3.7: Operator ukrštanja

4

Hibridizacija metaheurističkih algoritama za rešavanje HFS problema

Bez obzira na efikasnost implementacije, nijedna metaheuristicka metoda ne garantuje optimalnost dobijenog rešenja, šta više izuzetno je teško predvideti kvalitet dobijenog rešenja tj. njegovo odstupanje od optimalnog rešenja. Da bi se povećala efikasnost metaheurističkih metoda, u poslednje vreme u literaturi se pojavljuju kombinacije raznih metoda, tzv. hibridini algoritmi. Često se vrše hibridizacije populacijskih metaheuristika (population based) tako da se na određen broj jedinki na osnovu nekog kriterijuma primenjuje druga metaheuristika koja radi nad jednim rešenjem (single solution based). U literaturi se mogu naći hibridni algoritmi nastali od GA i TS, PSO i SA, a kombinacija GA i LS evoluirala je u novu metaheuristiku poznatu pod imenom memetski algoritam (Memetic Algorithm) i uspešno je primenjen na različite probleme. [30]).

U ovom poglavlju biće predložene hibridizacije spomenutih algoritama iz prvog poglavlja u cilju da se dobre osobine dve ili više metaheurističke metode sjedine u novu složeniju metodu za rešavanje velikih instanci problema raspoređivanja poslova u višefaznoj proizvodnji sa paralelnim mašinama. Hibridizacije metaheurističkih metoda se mogu izvršiti na više načina počev od jednostavnog nadovezivanja jedne na drugu pa do njihovog preplitanja u toku rada [29].

4.1 Predloženi hibridni algoritam PSO-TS

Ideja hibridizacije PSO i TS metaheuristika je da PSO algoritam ima ulogu diversifikacije, dok TS treba da vrši usmeravanje pretrage u prostoru dopustivih rešenja. PSO-TS algoritam je implementiran kao osnovni PSO algoritam uz to da se u početnoj populaciji na najbolju česticu primenjuje TS algoritam sa ograničenjem od β iteracija bez popravke rešenja. U svakoj iteraciji PSO algoritma na najbolju česticu i jednu česticu sa najmanjom vrednošću funkcije cilja primenjuje se TS algoritam. Ukoliko u PSO algoritmu nakon θ iteracija nije došlo do poboljšanja funkcije cilja izvršava se zamena 40% od ukupnog broja čestica. Sve čestice se sortraju na osnovu vrednosti funkcije cilja u rastućem poretku. Prvih 40 % čestica se zamenjuje novim česticama koje se dobijaju algoritmom tabu pretrage sa ograničenjem od β iteracija bez popravke rešenja.

4.2 Predloženi hibridni algoritam PSO-VNS-SA

Kod PSO, VNS i SA hibridizacije ideja je iskoristiti SA metodu za brzo pronalaženje dobrih pozicija čestica uz održavanje njihove raznovrsnosti. VNS algoritam treba da doprinese brzom lokalnom pretragom koja relativno dobro pozicioniranim česticama SA algoritmom pronalazi još bolju poziciju. Iz tog razloga, nakon generisanja početne populacije i određivanja najbolje čestice u roju, pokušava se sa popravkom pozicije najbolje čestice pomoću VNS metode sa ograničenjem od γ sekundi. U svakoj generaciji na slučajan način se bira: 20% čestica na koje se primenjuje metoda simuliranog kaljenja, 10% čestica i najbolja čestica na koje se primenjuje VNS algoritam sa ograničenjem od γ sekundi.

5

Eksperimentalni rezultati

5.1 Instance

Za testiranje predloženih metaheurističkih metoda korišćene su instance kreirane od strane J.Carliera i E.Nerona [7]. Tri karakteristična parametra instanci su broj poslova, broj nivoa i broj raspoloživih identičnih mašina u svakom nivou. Na primer, oznaka instance $j15c10b1$ označava problem sa 15 poslova i 10 mašina. Karakteri j i c se odnose na poslove i nivoe, respektivno, dok karakter b označava strukturu raspoređenih mašina. Poslednji broj 1 u nazivu instance, označava indeks instance datog tipa. Značenje karaktera koji označavaju strukturu raspoređenih mašina je:

(a) U srednjem nivou je raspoloživa samo jedna mašina, u ostalim po tri,

(b) U prvom nivou je raspoloživa samo jedna mašina, u ostalima po tri,

(c) U srednjem nivou su raspoložive dve mašine, u ostalima po tri,

(d) U svakom nivou su raspoložive po tri mašine.

Primer jedne instance prikazan je pseudokodom 5.1. Prva linija sadrži broj poslova, druga linija broj nivoa kroz koje poslovi prolaze, a treća linija sadrži broj raspoloživih mašina u svakom nivou. U narednim linijama su data vremena izvršavanja posla za svaki nivo i svaki posao pojedinačno. Recimo, u navedenom primeru instance, na nivou 1, za posao 1 je potrebno 10 jedinica vremena, a za posao 2 je potrebno 3 jedinice vremena. Slično se dobijaju vremena izvršavanja i za drugi i treći nivo.

```
Job:2
Stage:3
Machine:2,1,2
#1
1=10
2=3
#2
1=11
2=4
#3
1=15
2=9
```

Algoritam 5.1: Primer instance

Skup od 77 instanci sadrži 53 laka i 24 teška problema. Raspored mašina ima veliki uticaj na težinu problema pa tako problemi sa (a) i (b) rasporedom su lakši za rešavanje, dok su rasporedi (c) i (d) kompleksniji za rešavanje. Autori u [34] su sve probleme podelili u dve grupe. U grupu lakših problema svrstali su probleme do kojih se optimalno rešenje dostiže u kratkom vremenu. Takvi su svi problemi sa rasporedima mašina (a) i (b) tipa, i 10 problema sa 10 poslova i rasporedom (c) tipa (j10c10x). Ostali problemi (2x(c) raspored i 2x(d) raspored) spadaju u grupu teških problema. Za sve instance problema u [34] su izračunate donje granice (LB) koje garantuju vrednost manju ili jednaku od optimalne. Dostizanje (LB) vrednosti garantuje optimalnost rešenja, a u suprotnom se može koristi kao mera za računanje odstupanja dobijenog rešenja.

5.2 Rezultati testiranja metaheurističkih algoritama

Svi predloženi algoritmi su implementirani u programskom jeziku C++ i testirani na računaru sa Intel Core i5, 3.30GHz procesorom i 4GB radne memorije. Metaheurističke metode se zaustavljaju pri dostizanju LB vrednosti ili nakon isteka 60s. Ukoliko nije pronađeno rešenje čija vrednost funkcije cilja iznosi LB i isteklo je vreme izvršavanja prihvata se najbolje do tada pronađeno rešenje. Parametri koji su korišćeni za testiranje predloženih metaheurističkih algoritama prikazani su u tabeli 5.1.

Tabela 5.1: Parametri korišćeni za testiranje predloženih algoritama

	Metaheuristika			
Parametar	SA	TS	VNS	PSO
Početna temperatura (T)	1000	-	-	-
Eksponent (α)	0.97	-	-	-
Parametar hlađenja (v)	0.99	-	-	-
Završna temperatura	0.1	-	-	-
Minimalna dužina tabu liste D_{min}	-	5	-	-
Maksimalna dužina tabu liste D_{max}	-	20	-	-
Broj iteracija RVND	-	-	100	-
Broj čestica	-	-	-	15
Mutacioni faktor w	-	-	-	0.1
Kognitivni faktor C_1	-	-	-	0.8
Socijalni faktor C_2	-	-	-	0.8

Svi problemi sa (a) i (b) rasporedom mašina rešeni su uspešno SA, VNS i PSO metaheuristikama. TS metoda nije dostigla optimalno rešenje jednog problema sa (b) rasporedom mašina. Posmatrajući skup lakih problema SA, VNS i PSO algoritmi su pronašli optimalno rešenje u 88,7 % slučajeva dok je TS algoritam uspešno rešio 86,8 % problema iz lakše kategorije. Na skupu težih problema izdvaja se VNS metaheuristika sa 62,5% uspešnosti, zatim sledi TS sa 58%, PSO sa 50 % i na kraju SA sa 45% rešenih problema. Treba naglasiti da se SA algoritam u proseku izvršavao samo 0.09s zbog prirode primenjenog kriterijuma zaustavljanja, tj. dostizanje temeprature 0.1. Prosečno vreme izvršavanja ostalih algoritama je od 14s do 18s i prikazano je na slici 5.2. Najmanje prosečno odstupanje u procentima imaju rešenja dobijena TS a zatim VNS algoritmom. Detaljan prikaz prosečnog odstupanja dobijenih rešenja u zavisnosti od primenjene metaheuristike prikazan je grafički na slici 5.2.

Sumirani rezultati dobijeni predloženim algoritmima prikazani su u tabeli 5.2 sa datim procentom i brojem uspešno rešenih problema iz svake grupe.

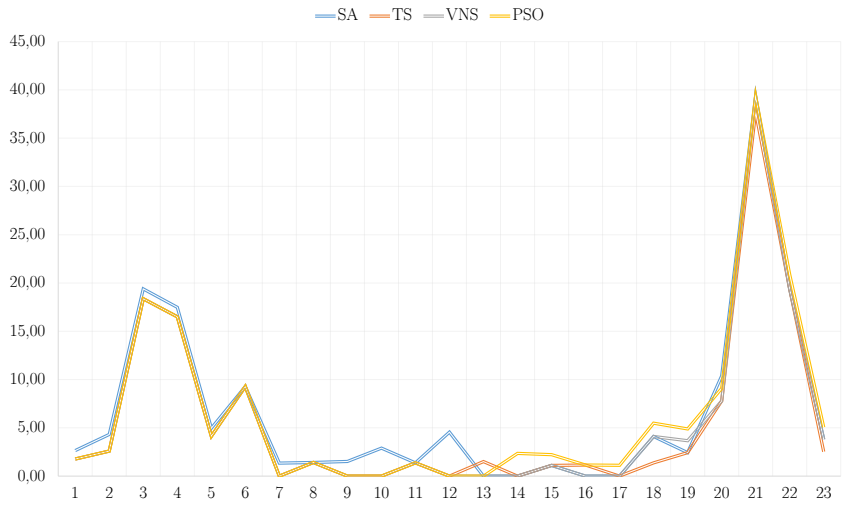
Tabela 5.2: Poređenje efikasnosti predloženih metaheurističkih metoda

	Prosečno		Uspešno rešeni problemi iz grupe			
	vreme	odstupanje	lakih # (53)	teških # (24)	lakih %	teških %
SA	**0.09**	2.56	47	11	88.7	45.8
TS	14.73	**2.36**	46	14	86.8	58.3
VNS	14.23	2.44	47	**15**	88.7	**62.5**
PSO	17.62	2.67	47	12	88.7	50.0

Detaljni rezultati testiranja pojedinačnih metaheuristika za sve instance problema dati su u tabeli 5.3. Kolona C_{max} predstavlja najbolje rešenje koje je algoritam dostigao nakon 10 pokretanja, dok kolona vreme predstavlja prosečno vreme rada algoritma. Vrednost u koloni GAP predstavlja srednje odstupanje rešenja od donje granice (LB) i računa se po formuli:

$$GAP = \frac{1}{10}\sum_{i=1}^{10} gap_i, \quad \text{gde je} \quad gap_i = 100 \cdot \frac{|rezultat_i - LB|}{LB}.$$

Kolona GAP se može koristiti za utvrđivanje efikasnosti testiranih algoritama za konkretnu instancu problema. Devijacija dobijenih rešenja dobijena po formuli $\sigma = \sqrt{\frac{1}{10}\sum_{i=1}^{10}(gap_i) - agap)^2}$ za sve testirane instance problema iznosi 0, pa sama kolona nije prikazana u tabeli. Poslednja kolona LB sadrži donju granicu instance problema.



Slika 5.1: Prosečno odstupanje dobijenih rešenja od LB vrednosti (instance teških problema)

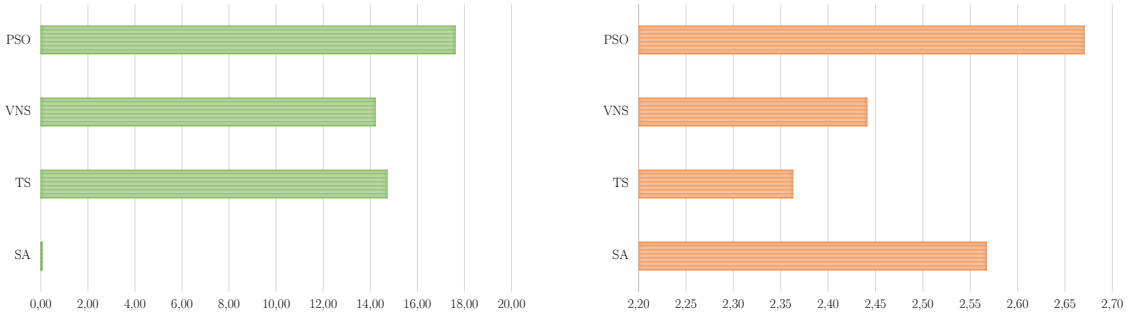
Tabela 5.3: Rezultati testiranja predloženih metaheurističkih algoritama

	$C_{\max}$				Vreme(s)				GAP				
	SA	TS	VNS	PSO	SA	TS	VNS	PSO	SA	TS	VNS	PSO	LB
j10c5a2	88	88	88	88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	88
j10c5a3	117	117	117	117	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	117
j10c5a4	121	121	121	121	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	121
j10c5a5	122	122	122	122	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	122
j10c5a6	110	110	110	110	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	110
j10c5b1	130	130	130	130	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	130
j10c5b2	107	107	107	107	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	107
j10c5b3	109	109	109	109	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	109
j10c5b4	122	122	122	122	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	122
j10c5b5	153	153	153	153	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	153
j10c5b6	115	115	115	115	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	115
j10c5c1	68	68	68	68	0.00	0.13	0.07	1.05	0.00	0.00	0.00	0.00	68
j10c5c2	75	74	74	74	0.21	4.04	0.04	0.35	1.35	0.00	0.00	0.00	74
j10c5c3	72	72	72	72	0.21	60.00	60.00	60.00	1.41	1.41	1.41	1.41	71
j10c5c4	67	66	66	66	0.21	0.05	0.01	0.30	1.52	0.00	0.00	0.00	66
j10c5c5	78	78	78	78	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	78
j10c5c6	71	69	69	69	0.06	0.00	0.00	0.14	2.90	0.00	0.00	0.00	69
j10c5d1	66	66	66	66	0.00	0.03	0.00	0.02	0.00	0.00	0.00	0.00	66
j10c5d2	74	74	74	74	0.22	60.00	60.00	60.00	1.37	1.37	1.37	1.37	73
j10c5d3	64	64	64	64	0.00	0.01	0.00	0.12	0.00	0.00	0.00	0.00	64
j10c5d4	70	70	70	70	0.00	0.04	0.00	0.10	0.00	0.00	0.00	0.00	70
j10c5d5	69	66	66	66	0.21	0.17	0.74	1.29	4.55	0.00	0.00	0.00	66
j10c5d6	62	62	62	62	0.00	0.04	0.00	0.06	0.00	0.00	0.00	0.00	62
j10c10a1	139	139	139	139	0.00	0.02	0.01	0.01	0.00	0.00	0.00	0.00	139
j10c10a2	158	158	158	158	0.00	0.10	0.05	0.03	0.00	0.00	0.00	0.00	158
j10c10a3	148	148	148	148	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	148
j10c10a4	149	149	149	149	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	149
j10c10a5	148	148	148	148	0.00	0.03	0.01	0.00	0.00	0.00	0.00	0.00	148
j10c10a6	146	146	146	146	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	146
j10c10b1	163	163	163	163	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	163
j10c10b2	157	157	157	157	0.00	0.06	0.00	0.04	0.00	0.00	0.00	0.00	157
j10c10b3	169	169	169	169	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	169
j10c10b4	159	159	159	159	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	159
j10c10b5	165	165	165	165	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	165
j10c10b6	165	165	165	165	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	165
j10c10c1	116	115	115	115	0.41	60.00	60.00	60.00	2.65	1.77	1.77	1.77	113
j10c10c2	121	119	119	119	0.40	60.00	60.00	60.00	4.31	2.59	2.59	2.59	116
j10c10c3	117	116	116	116	0.40	60.00	60.00	60.00	19.39	18.37	18.37	18.37	98
j10c10c4	121	120	120	120	0.39	60.00	60.00	60.00	17.48	16.50	16.50	16.50	103
j10c10c5	127	126	126	126	0.39	60.00	60.00	60.00	4.96	4.13	4.13	4.13	121
j10c10c6	106	106	106	106	0.43	60.00	60.00	60.00	9.28	9.28	9.28	9.28	97
j15c5a1	178	178	178	178	0.00	0.04	0.01	0.01	0.00	0.00	0.00	0.00	178
j15c5a2	165	165	165	165	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	165
j15c5a3	130	130	130	130	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	130
j15c5a4	156	156	156	156	0.00	0.07	0.01	0.00	0.00	0.00	0.00	0.00	156
j15c5a5	164	164	164	164	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	164
j15c5a6	178	178	178	178	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	178
j15c5b1	170	170	170	170	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	170
j15c5b2	152	152	152	152	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	152
j15c5b3	157	157	157	157	0.00	0.05	0.01	0.00	0.00	0.00	0.00	0.00	157
j15c5b4	147	147	147	147	0.00	0.03	0.01	0.01	0.00	0.00	0.00	0.00	147
j15c5b5	166	166	166	166	0.00	0.03	0.01	0.01	0.00	0.00	0.00	0.00	166
j15c5b6	175	175	175	175	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	175

*nastavak tabele 5.3

	$C_{\max}$				Vreme(s)				GAP				
	SA	TS	VNS	PSO	SA	TS	VNS	PSO	SA	TS	VNS	PSO	LB
j15c5c1	85	85	85	87	0.03	1.41	0.02	60.00	0.00	0.00	0.00	2.35	85
j15c5c2	91	91	91	92	0.35	55.47	48.17	60.00	1.11	1.11	1.11	2.22	90
j15c5c3	87	88	87	88	0.00	9.83	0.11	52.64	0.00	1.15	0.00	1.15	87
j15c5c4	89	89	89	90	0.02	1.74	0.01	57.33	0.00	0.00	0.00	1.12	89
j15c5c5	76	74	76	77	0.36	60.00	60.00	60.00	4.11	1.37	4.11	5.48	73
j15c5c6	91	91	91	91	0.00	0.17	0.01	1.33	0.00	0.00	0.00	0.00	91
j15c5d1	167	167	167	167	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	167
j15c5d2	84	84	85	86	0.36	60.00	60.00	60.00	2.44	2.44	3.66	4.88	82
j15c5d3	85	83	83	84	0.37	60.00	60.00	60.00	10.39	7.79	7.79	9.09	77
j15c5d4	85	84	85	85	0.37	60.00	60.00	60.00	39.34	37.70	39.34	39.34	61
j15c5d5	80	80	80	81	0.38	60.00	60.00	60.00	19.40	19.40	19.40	20.90	67
j15c5d6	82	81	82	83	0.36	60.00	60.00	60.00	3.80	2.53	3.80	5.06	79
j15c10a1	236	236	236	236	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	236
j15c10a2	200	200	200	200	0.00	0.15	0.01	0.23	0.00	0.00	0.00	0.00	200
j15c10a3	198	198	198	198	0.00	0.13	0.01	0.07	0.00	0.00	0.00	0.00	198
j15c10a4	225	225	225	225	0.00	0.06	0.01	0.01	0.00	0.00	0.00	0.00	225
j15c10a5	182	182	182	182	0.00	0.08	0.01	0.02	0.00	0.00	0.00	0.00	182
j15c10a6	200	200	200	200	0.00	0.01	0.01	0.03	0.00	0.00	0.00	0.00	200
j15c10b1	222	222	222	222	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	222
j15c10b2	187	187	187	187	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	187
j15c10b3	222	222	222	222	0.00	0.07	0.01	0.00	0.00	0.00	0.00	0.00	222
j15c10b4	221	221	221	221	0.00	0.03	0.01	0.00	0.00	0.00	0.00	0.00	221
j15c10b5	200	203	200	200	0.00	6.04	0.01	0.02	0.00	1.50	0.00	0.00	200
j15c10b6	219	219	219	219	0.00	0.09	0.01	0.00	0.00	0.00	0.00	0.00	219

- Instance problema koji spadaju u grupu teških problema označene su podebljanim slovima.



Slika 5.2: Grafički prikaz prosečnog vremena izvršavanja i prosečnog odstupanja

5.3 Rezultati testiranja hibridnih metaheurističkih algoritama

U nastavku su predstavljeni eksperimentalni rezultati dobijeni serijom testiranja predloženih hibridnih metaheurističkih algoritama **P**SO-**V**NS-**S**A i **P**SO-**T**S. Dobijeni rezultati su upoređeni sa rezultatima iz ostalih relevantnih radova. Algoritmi su implementirani u programskom jeziku C++ i izvršavani na računaru sa AMD Turion X2, 2.3GHz procesorom i 4GB radne memorije. Testiranje je izvršeno na skupu instanci problema kreiranim od strane Carliera i Nerona [7]. Kriterijum zaustavljanja je vremensko ograničenje od 1600s ili dostizanje LB vrednost. Najbolje dobijeno rešenje do ispunjenja kriterijuma zaustavljanja se prihvata kao rešenje problema.

U literaturi se mogu naći rezultati dobijeni pomoću metaheuristika kao što su veštački imuni sistem (AIS) [12], kvantum - inspirisan imuni sistem (QIA) [35], imuni sistem baziran na imunoglobulinu (IAIS) [8], genetski algoritmi (GA) [5] i optimizacija mravljim kolonijama(ACO) [23]. Sa vremenskim ograničenjem od 1600s testirani su AIS, GA, IAIS algoritmi, dok su QIA i ACO izvršavani za fiksiran broj iteracija. Najmanja vrednost C_{max} iz deset izvršavanja je prihvaćena kao najbolje rešenje.

AIS algoritam je implementiran korišćenjem Excel Macro-a (Visual Basic Applications for Excel) i izvršavan na računaru sa Intel P4, 1.7 GHz procesorom i 256 MB radne memorije. GA metaheuristika je implementirana u C++ jeziku i izvršavana na računaru sa Intel P4, 3 GHz procesorom i 512 MB radne memorije. QIA algoritam je implementiran u programskom paketu MATLAB i testiran je na računaru sa Pentium Dual, 1.6 GHz procesorom, dok za ACO algoritam slične informacije nisu navedene u [23].

Pored metaheurističkih algoritama predložen je i hibridni algoritam PSO-BH [27], koji kombinuje optimizaciju rojevima čestica i bottleneck heuristiku. Ideja bottleneck heuristike je u pronalaženju nivoa u kome je protok poslova smanjen i kreiranje sekvence posmatrajuci nivo koji prethode i slede u procesu proizvodnje. U 2014. godini predložena su još dva algoritma IDABC [9] koji predstavlja optimizaciju kolonijama pčela i hibridna metod promenljivih okolina HVNS [38]. Performanse svih algoritama sumirane su u tabeli 5.4.

Tabela 5.4: Poređenje predloženih algoritama sa ostalim algoritmima iz literature

	Laki problemi			Teški problemi		
	ukupan broj	% rešenih	GAP	ukupan broj	% rešenih	GAP
PSO-VNS-SA	53	**88.7**	0.99	24	66.7	2.97
PSO-TS	53	**88.7**	0.99	24	62.5	3.13
IAIS	53	**88.7**	0.99	24	66.7	2.97
QIA	29	100	0	12	0	5.04
AIS	53	**88.7**	0.99	24	66.7	3.13
GA	53	88.7	0.99	24	62.5	3.07
ACO	45	64.4	1.13	24	66.7	4.78
PSO-BH	53	**88.7**	**0.95**	24	**75**	**2.85**
HVNS	53	**88.7**	**0.95**	24	70.8	2.91
IDABC	53	**88.7**	**0.95**	24	70.8	2.91

Kako su svi algoritmi izvršavani u različitim uslovima, ne mogu se precizno uporediti na osnovu vremena izvršavanja. Poređenje navedenih algoritama je izvršeno na osnovu broja rešenih problema i odstupanja od donjih granica (LB). Na skupu lakih problema svi algoritmi osim ACO i QIA uspeli su da reše 47 od 53 problema (88.7%) sa približnim procentom odstupanja od 0.95% do 0.99 %. ACO je testiran na manjem broju instanci i uspešno je rešio 29 od 45 problema (64.4%) sa odstupanjem od 1.13 %. QIA algoritam ima procenat odstupanja nula ali je testiran samo na 29 od ukupno 53 instanci. Na skupu teških problema PSO-VNS-SA i IAIS uspeli su da reše 16 od 24 problema (66,7 %) sa procentom odstupanja (2.97 %), dok je druga predložena hibridna metaheuristika PSO-TS uspela da reši 15 od 24 problema kao i GA ali sa većim prosečnim odstupanjem od 3.13 %. PSO-BH algoritam je uspeo da reši 18 od 24 problema (75 %) sa najmanjim prosečnim odstupanjem od 2.85%, zatim slede HVNS i IDABC sa 70.8% uspešno rešenih problema i prosečnim odstupanjem od 2.91%. Detaljni prikaz uporednih rezultata nad instancama teških problema dat je tabelom 5.5.

Tabela 5.5: Uporedni rezultati metaheurističkih algoritama

	$C_{\max}$								GAP								
	PSV	PPT	IAIS	AIS	GA	ACO	HVNS	IDABC	PSV	PT	IAIS	AIS	GA	ACO	HVNS	IDABC	LB
j10c5c1	68	68	68	68	68	68	68	68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	68
j10c5c2	74	74	74	74	74	75	74	74	0.00	0.00	0.00	0.00	0.00	1.35	0.00	0.00	74
j10c5c3	72	72	72	72	72	72	72	72	1.41	1.41	1.41	1.41	1.41	1.41	1.41	1.41	71
j10c5c4	66	66	66	66	66	66	66	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	66
j10c5c5	78	78	78	78	78	78	78	78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	78
j10c5c6	69	69	69	69	69	69	69	69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	69
j10c5d1	66	66	66	66	66	66	66	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	66
j10c5d2	74	74	74	73	74	75	73	73	1.37	1.37	1.37	0.00	1.37	2.74	0.00	0.00	73
j10c5d3	64	64	64	64	64	64	64	64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	64
j10c5d4	70	70	70	70	70	70	70	70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	70
j10c5d5	66	66	66	66	66	66	66	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	66
j10c5d6	62	62	62	62	62	62	62	62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	62
j15c5c1	85	85	85	85	85	85	85	85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	85
j15c5c2	90	91	90	91	91	90	90	90	0.00	1.11	0.00	1.11	1.11	0.00	0.00	0.00	90
j15c5c3	87	87	87	87	87	87	87	87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	87
j15c5c4	89	89	89	89	89	91	89	89	0.00	0.00	0.00	0.00	0.00	2.25	0.00	0.00	89
j15c5c5	74	75	74	74	74	77	74	74	1.37	2.74	1.37	1.37	1.37	5.48	1.37	1.37	73
j15c5c6	91	91	91	91	91	97	91	91	0.00	0.00	0.00	0.00	0.00	6.59	0.00	0.00	91
j15c5d1	167	167	167	167	167	167	167	167	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	167
j15c5d2	84	84	84	84	84	88	84	84	2.44	2.44	2.44	2.44	2.44	7.32	2.44	2.44	82
j15c5d3	82	83	82	83	83	86	82	82	6.49	7.79	6.49	7.79	7.79	11.69	6.49	6.49	77
j15c5d4	84	84	84	84	84	88	84	84	37.70	37.70	37.70	37.70	37.70	44.26	37.70	37.70	61
j15c5d5	79	79	79	80	79	84	79	79	17.91	17.91	17.91	19.40	17.91	25.37	17.91	17.91	67
j15c5d6	81	81	81	82	81	84	81	81	2.53	2.53	2.53	3.80	2.53	6.33	2.53	2.53	79

Za dalja poređenja efikasnosti PSO-VNS-SA, IAIS i AIS algoritama korišćene su instance težih problema većih dimenzija kreirane u [8]. Instance su generisane za $n = 25, 50, 100, 250, 500$, $s = 5, 10$, i tipovima a, b, c, d gde n i s predstavljaju broj poslova i broj nivoa, dok tip instance određuje ostale parametre prikazane u tabeli 5.6. Težina instanci problema se ogleda u većem broju mogućih rasporeda poslova i različitom broju paralelnih mašina u svakom nivou. Za razliku od prehodnog skupa instanci gde je naveći broj paralelnih mašina bio 3 (c i d tip), u novim skupu instanci pojavljuju se problemi sa 5 paralelnih mašina u jednoj fazi. Za poređenje sa rezultatima PSO-BH, HVNS i IDABC algoritama iskorišćene su instance kreirane u [27] za potrebe poređenja performansi AIS i PSO-BH algoritma. Za svaku instancu problema, broj

poslova je povećan na 30, mašine su raspoređene u 5 nivoa, u svakom niovu povećan je broj mašina sa 2 ili 3 na slučajan broj iz izabrane uniformne raspodele U[3,5] dok je vreme obrade posla vrednost iz izabrane uniformne raspodele U[1,100].

Tabela 5.6: Parametri novih instanci problema

Tip	Vreme obrade	Broj mašina
a	$U(1,50)$	$U(2,5)$
b	$U(1,50)$	$U(4,5)$
c	$U(1,100)$	$U(2,5)$
d	$U(1,100)$	$U(4,5)$

U [8] su predstavljeni rezultati dobijeni IAIS i AIS algoritmima u deset izvršavanja sa vremenskim ograničenjem od 100s. IAIS algoritam je na testovima pokazao znatno bolje performanse od AIS algoritma. Kako za nove instance ne postoje donja ograničenja LB, za svaku instancu problema je data najmanja, najveća, prosečna dobijena vrednost i standardno odstupanje na osnovu 10 izvršavanja. Testiranje PSO-VNS-SA algoritma je izvršeno deset puta za svaku instancu problema sa istim kriterijumom zaustavljanja kao kod IAIS i AIS, tj. sa vremenskim ograničenjem od 100s.
Dobijeni rezultati pokazuju da PSO-VNS-SA algoritam dobija kvalitetnija rešenja probleme sa manjim standardnim odstupanjem od IAIS i AIS algoritama. Za sve instance problema PSO-VNS-SA algoritam je dostigao bolje rešenje ili bolje prosečno rešenje u 10 izvršavanja. Dobijeni rezultati su upoređeni na osnovu odstupanja prosečnog dobijenog rešenja iz 10 izvršavanja od najboljeg dobijenog rešenja nekog od algoritama. Vrednost odstupanja u procentima je dato u koloni PD i izračunato je korišćenjem formule:

$$PD = 100 \cdot \frac{C_{\max}^{\text{avg}} - C_{\max}^{\text{best}}}{C_{\max}^{\text{best}}},$$

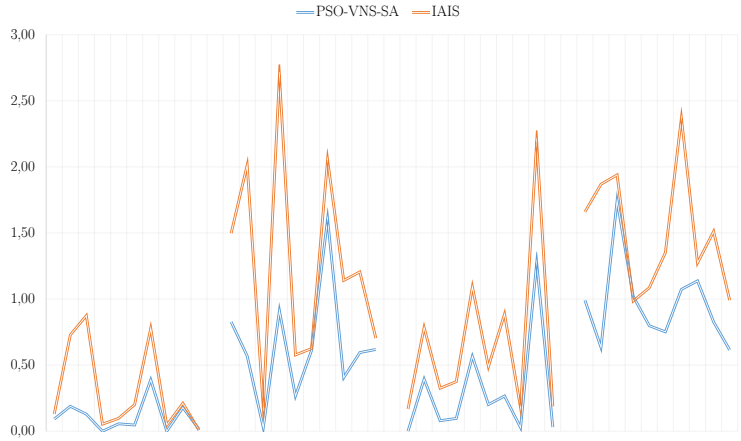
gde je $C_{\max}^{\text{avg}}$ prosečno dobijena vrednost rešenja u deset izvršavanja algoritma i $C_{\max}^{\text{best}}$ najmanja dobijena vrednost nekog od algoritama.

Dobijeni rezultati za sve instance problema dati su u dodatku (tabela 7.1, 7.2, 7.3) dok su za ilustraciju efikasnosti PSO-VNS-SA algoritma dobijeni rezultati predstavljeni u tabeli 5.7 na skup problema sa parametrima $n = 100$, $s = 5$.

Tabela 5.7: Poređenje rezultata metaheurističkih algoritama na instancama teških problema

	PSO-VNS-SA				IAIS				AIS				PD		
	min	avg	max	std	min	avg	max	std	min	avg	max	std	PVS	IAIS	AIS
j100c5a01	1313	1314.2	1317	0.98	1313	1314.7	1317	1.25	1333	1367.6	1400	18.47	0.09	0.13	4.16
j100c5a02	1277	1279.4	1283	2.94	1277	1286.3	1293	4.22	1295	1317.4	1337	15.31	0.19	0.73	3.16
j100c5a03	1315	1316.7	1330	4.45	1317	1326.5	1332	5.56	1377	1386.2	1401	8.11	0.13	0.87	5.41
j100c5a04	1298	1298	1298	0	1298	1298.7	1303	1.64	1300	1317.8	1353	18.43	0.00	0.05	1.53
j100c5a05	1433	1433.8	1434	0.4	1433	1434.4	1437	1.17	1439	1451.8	1473	9.64	0.06	0.10	1.31
j100c5a06	1292	1292.6	1293	0.49	1293	1294.6	1299	2.41	1311	1346.8	1375	18.94	0.05	0.20	4.24
j100c5a07	900	903.5	905	1.43	903	907	916	3.92	927	954.5	982	20.84	0.39	0.78	6.06
j100c5a08	1302	1302	1302	0	1302	1302.6	1305	1.26	1317	1335.9	1350	9.87	0.00	0.05	2.60
j100c5a09	949	949.7	952	1	948	950	952	1.33	965	986.5	1010	13.25	0.18	0.21	4.06
j100c5a10	1387	1387.2	1388	0.4	1387	1387.1	1388	0.32	1407	1427.6	1440	11.96	0.01	0.01	2.93
j100c5b01	701	706.8	715	4.49	708	711.5	716	2.46	974	1000.5	1050	21.86	0.83	1.50	42.72
j100c5b02	720	724.1	729	3.05	728	734.5	740	4.53	969	1005.8	1048	28.46	0.57	2.01	39.69
j100c5b03	726	726.2	727	0.4	726	727.3	730	1.34	969	996.2	1047	25.26	0.03	0.18	37.22
j100c5b04	669	675.1	678	2.47	678	687	697	6.73	904	953.6	990	26.21	0.91	2.69	42.54
j100c5b05	676	677.8	684	2.18	677	679.9	687	3.6	910	928.8	993	23.92	0.27	0.58	37.40
j100c5b06	735	739.5	744	3.2	735	739.6	743	2.72	1015	1034.6	1055	15.17	0.61	0.63	40.76
j100c5b07	672	682.8	691	6.32	679	685.8	693	4.1	955	982.6	1032	23.73	1.61	2.05	46.22
j100c5b08	666	668.7	671	1.68	669	673.6	680	3.53	935	953.9	979	15.73	0.41	1.14	43.23
j100c5b09	672	676	682	2.37	672	680.1	691	5.76	974	1004.5	1050	27.2	0.60	1.21	49.48
j100c5b10	712	715.4	720	2.37	711	716	719	2.87	942	963.1	1002	20.56	0.62	0.70	35.46
j100c5c01	2857	2857	2857	0	2857	2861.8	2876	7.91	2857	2890.4	2919	19.21	0.00	0.17	1.17
j100c5c02	2636	2646.4	2658	6.04	2652	2656.7	2664	3.97	2658	2709.1	2812	43.81	0.39	0.79	2.77
j100c5c03	2516	2518	2523	2.76	2519	2524.2	2536	5.25	2533	2612.9	2672	48.03	0.08	0.33	3.85
j100c5c04	2984	2986.9	2990	2.07	2988	2995.2	3014	7.7	3022	3084.6	3121	33.39	0.10	0.38	3.37
j100c5c05	1712	1721.7	1736	6.75	1719	1730.7	1753	9.72	1996	2031.9	2080	29.61	0.57	1.09	18.69
j100c5c06	2726	2731.5	2733	1.86	2732	2739.3	2757	8.23	2756	2816.3	2879	45.81	0.20	0.49	3.31
j100c5c07	2369	2375.3	2412	13.23	2369	2390	2412	16.96	2390	2457.2	2506	34.77	0.27	0.89	3.72
j100c5c08	2598	2598.7	2605	2.1	2598	2602.6	2605	3.31	2627	2657.9	2685	17.74	0.03	0.18	2.31
j100c5c09	1830	1853.3	1873	10.79	1837	1870.1	1911	23.33	1949	2006.1	2065	36.92	1.27	2.19	9.62
j100c5c10	2617	2617.8	2618	0.4	2619	2621.9	2625	1.52	2622	2636.2	2645	6.68	0.03	0.19	0.73
j100c5d01	1213	1223	1233	6.12	1211	1231.1	1241	7.95	1874	1920.1	1993	41.43	0.99	1.66	58.55
j100c5d02	1386	1394.8	1407	5.58	1392	1411.9	1430	11.1	1849	1913.9	2106	70.76	0.63	1.87	38.09
j100c5d03	1341	1364.4	1393	19.3	1348	1367	1398	13.9	1932	2005.9	2072	46.21	1.74	1.94	49.58
j100c5d04	1421	1435.5	1456	9.76	1425	1435	1447	8.15	1913	1959.4	2054	45.95	1.02	0.99	37.89
j100c5d05	1327	1337.6	1352	8.89	1328	1341.4	1369	11.72	1917	1944.8	2009	27.54	0.80	1.09	46.56
j100c5d06	1384	1394.4	1410	9.62	1384	1402.7	1425	14.18	1821	1875.7	1934	40.4	0.75	1.35	35.53
j100c5d07	1249	1262.4	1284	11.21	1269	1278.6	1295	8.9	1847	1911.9	1980	37.13	1.07	2.37	53.07
j100c5d08	1398	1413.9	1436	13.05	1404	1415.8	1429	8.73	1880	1939.5	1983	30.52	1.14	1.27	38.73
j100c5d09	1356	1367.2	1382	8.02	1365	1376.5	1396	8.15	1969	2034.1	2096	52.47	0.83	1.51	50.01
j100c5d10	1464	1473	1502	11.52	1464	1478.5	1511	14.55	1937	2010.6	2129	54.48	0.61	0.99	37.34
												Prosečna vrednost:	**0.96**	1.50	44.53

Na prikazanom skupu instanci rešenja dobijena PSO-VNS-SA algoritmom imaju najmanje prosečno odstupanja (0.96%). Za gotovo sve testirane instance problema maksimalne dobijene vrednosti PSO-VNS-SA algoritmom su manje od minimalnih dobijenih vrednosti AIS algoritmom, pa iz tog razloga je prosečno odstupanje rešenja AIS algoritma veliko (44.53 %). Grafički prikaz i poređenje odstupanja rešenja dobijenih PSO-VNS-SA i IAIS algoritmima dati su na slici 5.3.



Slika 5.3: Grafički prikaz odstupanja rešenja za instance teških problema

Testiranje PSO-VNS-SA algortima na instancama sa 30 poslova izvršeno je sa vremenskim ograničenjem od 200 sekundi. Kao i u [38], najbolje dobijeno rešenje u 20 izvršavanja se prihvata kao rešenje. Za razliku od IAIS i AIS algoritama nisu za sve algoritme dostupni podaci o najvećoj dobijenoj vrednost C_{max} kao i standardno odstupanje, tako da je grubo poređenje izvršeno na osnovu najbolje dobijenog rešenja. Dobijeni rezultati pokazuju da IDABC [9] može da dobije najkvalitetnija rešenja, dok PSO-BH [27] algoritam koji je na instancama sa 10 i 15 poslova pokazao najbolje rezultate sada na svim instancama dobija rešenja lošijeg kvaliteta u odnosu na ostale algoritme, upravo zbog bottleneck heuristike koja mu je davala prednost a sada zahteva više vremena kako broj poslova i paralelnih mašina raste. Rezultati su prikazani u tabeli 5.8.

Tabela 5.8: Poređenje rezultata PSO-VNS-SA algoritma sa ostalim rezultatima iz literature

	C_{max}			
	PSO-VNS-SA	PSO-BH	HVNS	IDABC
j30c5e1	466	471	464	463
j30c5e2	616	616	616	616
j30c5e3	597	602	595	593
j30c5e4	569	575	566	565
j30c5e5	603	605	601	600
j30c5e6	608	605	603	601
j30c5e7	626	629	626	626
j30c5e8	677	678	674	674
j30c5e9	645	651	643	642
j30c5e10	577	594	577	573
Prosečna vrednost:	598.4	602.6	596.5	**595.3**

Na datom skupu instanci problema predloženi PSO-VNS-SA algoritam dostiže rešenja prosečne vrednosti (598.4), što je manje od rešenja dobijenih PSO-BH algoritma (602.5) ali ne i od rešenja HVNS i IDABC algoritama.

6 Zaključak

U ovom radu razmatrana je primena metaheurističkih algoritama za rešavanje problema višefazne proizvodnje sa paralelnim mašinama u cilju minimizacije vremena proizvodnje. Performanse predloženih metaheuristika upoređene su na poznatom skupu instanci problema, koji je kreiran od strane Carliera i Nerona [7]. VNS algoritam je uspeo da reši najveći broj problema, rešenja dobijena TS algoritmom imaju najmanje prosečno odstupanje, dok SA algoritma ima najmanje prosečno vreme izvršavanja. U cilju da se dobre osobine svake metaheurističke metode sjedine u novu složeniju metodu predloženi su PSO-TS i PSO-VNS-SA algoritmi. Eksperimentalni rezultati u prvom delu pokazali su da predloženi PSO-VNS-SA algoritam nadmašuje drugi predloženi algoritam PSO-TS ali i nekoliko drugih poznatih algoritama iz literature QIA, ACO i GA. U drugom delu testiranja i za potrebe daljih poređenja IAIS, AIS i PSO-VNS-SA, IDABC, HVNS i PSO-BH algoritama korišćene su novi teži problemi veće dimenzije generisani u [8] i [27]. Dobijeni eksperimentalni rezultati na novim instancama problema pokazuju da PSO-VNS-SA algoritam može efikasnije da dostigne bolja rešenja i od AIS, IAIS i PSO-BH algoritma. Efikasnost PSO-VNS-SA algoritmu u odnosu na osnovni PSO algoritam daje SA metoda tako što brzo unapređuje pozicije čestica, a pri tome održava raznovrsnost među njima. VNS svoj doprinos daje kroz brzu lokalnu pretragu koja relativno dobro pozicioniranim česticama pronalazi još bolju poziciju.

Dalja istraživanja se mogu usmeriti u poboljšanje PSO-VNS-SA algoritma kroz dinamičko podešavanje parametra tokom izvršavanja, paralelizaciju PSO algoritma kao i modifikaciju algoritma za rešavanje srodnih problema proizvodnje sa dodatnim ograničenjima i drugačijim ocenama kvaliteta rasporeda poslova.

7

Dodatak

U nastavku su date tabela sa rezultatima testiranja PSO-VNS-SA algoritma na skupu od 400 instanci teških problema kreiranih u [8].

Tabela 7.1: Rezultati testiranja na teškim instancama problema sa 25 i 50 poslova

$n = 25$				$n = 50$			
	C_{max}		C_{max}		C_{max}		C_{max}
j25c5a01	303	j25c10a01	466	j50c5a01	763	j50c10a01	824
j25c5a02	265	j25c10a02	469	j50c5a02	468	j50c10a02	742
j25c5a03	374	j25c10a03	492	j50c5a03	667	j50c10a03	811
j25c5a04	357	j25c10a04	465	j50c5a04	763	j50c10a04	816
j25c5a05	323	j25c10a05	453	j50c5a05	726	j50c10a05	797
j25c5a06	392	j25c10a06	503	j50c5a06	574	j50c10a06	713
j25c5a07	262	j25c10a07	493	j50c5a07	674	j50c10a07	661
j25c5a08	249	j25c10a08	546	j50c5a08	731	j50c10a08	880
j25c5a09	430	j25c10a09	528	j50c5a09	688	j50c10a09	849
j25c5a10	438	j25c10a10	472	j50c5a10	690	j50c10a10	808
j25c5b01	240	j25c10b01	381	j50c5b01	358	j50c10b01	523
j25c5b02	250	j25c10b02	383	j50c5b02	344	j50c10b02	545
j25c5b03	237	j25c10b03	394	j50c5b03	400	j50c10b03	540
j25c5b04	236	j25c10b04	412	j50c5b04	414	j50c10b04	514
j25c5b05	222	j25c10b05	346	j50c5b05	395	j50c10b05	526
j25c5b06	280	j25c10b06	367	j50c5b06	406	j50c10b06	544
j25c5b07	226	j25c10b07	393	j50c5b07	408	j50c10b07	562
j25c5b08	229	j25c10b08	393	j50c5b08	437	j50c10b08	560
j25c5b09	243	j25c10b09	405	j50c5b09	447	j50c10b09	543
j25c5b10	229	j25c10b10	376	j50c5b10	403	j50c10b10	559
j25c5c01	713	j25c10c01	963	j50c5c01	1417	j50c10c01	1598
j25c5c02	753	j25c10c02	889	j50c5c02	968	j50c10c02	1555
j25c5c03	843	j25c10c03	902	j50c5c03	1357	j50c10c03	1774
j25c5c04	494	j25c10c04	905	j50c5c04	1422	j50c10c04	1664
j25c5c05	852	j25c10c05	940	j50c5c05	1351	j50c10c05	1645
j25c5c06	540	j25c10c06	930	j50c5c06	1286	j50c10c06	1567
j25c5c07	480	j25c10c07	956	j50c5c07	1318	j50c10c07	1495
j25c5c08	739	j25c10c08	1013	j50c5c08	1276	j50c10c08	1508
j25c5c09	764	j25c10c09	1016	j50c5c09	938	j50c10c09	1590
j25c5c10	787	j25c10c10	1040	j50c5c10	693	j50c10c10	1614
j25c5d01	417	j25c10d01	753	j50c5d01	784	j50c10d01	1083
j25c5d02	467	j25c10d02	782	j50c5d02	780	j50c10d02	1073
j25c5d03	534	j25c10d03	707	j50c5d03	757	j50c10d03	1005
j25c5d04	448	j25c10d04	740	j50c5d04	807	j50c10d04	1003
j25c5d05	458	j25c10d05	806	j50c5d05	879	j50c10d05	988
j25c5d06	518	j25c10d06	752	j50c5d06	810	j50c10d06	1097
j25c5d07	491	j25c10d07	810	j50c5d07	782	j50c10d07	1106
j25c5d08	456	j25c10d08	820	j50c5d08	779	j50c10d08	1072
j25c5d09	446	j25c10d09	772	j50c5d09	733	j50c10d09	1075
j25c5d10	516	j25c10d10	776	j50c5d10	774	j50c10d10	1100

Tabela 7.2: Rezultati testiranja na teškim instancama problema sa 100 i 250 poslova

$n = 100$				$n = 250$			
	C_{max}		C_{max}		C_{max}		C_{max}
j100c5a01	1313	j100c10a01	1398	j250c5a01	2162	j250c10a01	3344
j100c5a02	1277	j100c10a02	1530	j250c5a02	3315	j250c10a02	3391
j100c5a03	1315	j100c10a03	1468	j250c5a03	3357	j250c10a03	3443
j100c5a04	1298	j100c10a04	1476	j250c5a04	3123	j250c10a04	3513
j100c5a05	1433	j100c10a05	1424	j250c5a05	2203	j250c10a05	3358
j100c5a06	1292	j100c10a06	822	j250c5a06	3334	j250c10a06	3120
j100c5a07	900	j100c10a07	1348	j250c5a07	3397	j250c10a07	3382
j100c5a08	1302	j100c10a08	1331	j250c5a08	3174	j250c10a08	3314
j100c5a09	949	j100c10a09	997	j250c5a09	3442	j250c10a09	3457
j100c5a10	1387	j100c10a10	1348	j250c5a10	3167	j250c10a10	3305
j100c5b01	701	j100c10b01	887	j250c5b01	1716	j250c10b01	1787
j100c5b02	720	j100c10b02	820	j250c5b02	1668	j250c10b02	1927
j100c5b03	726	j100c10b03	833	j250c5b03	1691	j250c10b03	1876
j100c5b04	669	j100c10b04	814	j250c5b04	1653	j250c10b04	1861
j100c5b05	676	j100c10b05	861	j250c5b05	1660	j250c10b05	1858
j100c5b06	735	j100c10b06	897	j250c5b06	1731	j250c10b06	1829
j100c5b07	672	j100c10b07	873	j250c5b07	1641	j250c10b07	1766
j100c5b08	666	j100c10b08	808	j250c5b08	1682	j250c10b08	1865
j100c5b09	672	j100c10b09	788	j250c5b09	1724	j250c10b09	1944
j100c5b10	712	j100c10b10	827	j250c5b10	1716	j250c10b10	1818
j100c5c01	2857	j100c10c01	2886	j250c5c01	6162	j250c10c01	6587
j100c5c02	2636	j100c10c02	2669	j250c5c02	6467	j250c10c02	6632
j100c5c03	2516	j100c10c03	2880	j250c5c03	6627	j250c10c03	6969
j100c5c04	2984	j100c10c04	2959	j250c5c04	6578	j250c10c04	6897
j100c5c05	1712	j100c10c05	2108	j250c5c05	6432	j250c10c05	6841
j100c5c06	2726	j100c10c06	3303	j250c5c06	6501	j250c10c06	6489
j100c5c07	2369	j100c10c07	2876	j250c5c07	6573	j250c10c07	6735
j100c5c08	2598	j100c10c08	2849	j250c5c08	3251	j250c10c08	6718
j100c5c09	1830	j100c10c09	2787	j250c5c09	6227	j250c10c09	6654
j100c5c10	2617	j100c10c10	2694	j250c5c10	6010	j250c10c10	6480
j100c5d01	1213	j100c10d01	1769	j250c5d01	3340	j250c10d01	3688
j100c5d02	1386	j100c10d02	1826	j250c5d02	3295	j250c10d02	3675
j100c5d03	1341	j100c10d03	1695	j250c5d03	3242	j250c10d03	3704
j100c5d04	1421	j100c10d04	1652	j250c5d04	3469	j250c10d04	3611
j100c5d05	1327	j100c10d05	1670	j250c5d05	3231	j250c10d05	3723
j100c5d06	1384	j100c10d06	1778	j250c5d06	3348	j250c10d06	3710
j100c5d07	1249	j100c10d07	1700	j250c5d07	3325	j250c10d07	3470
j100c5d08	1398	j100c10d08	1708	j250c5d08	3370	j250c10d08	3817
j100c5d09	1356	j100c10d09	1759	j250c5d09	3374	j250c10d09	3678
j100c5d10	1464	j100c10d10	1779	j250c5d10	3453	j250c10d10	3738

Tabela 7.3: Rezultati testiranja na teškim instancama problema sa 500 poslova

$s = 5$		$s = 10$	
	C_{max}		C_{max}
j500c5a01	4393	j500c10a01	6723
j500c5a02	6466	j500c10a02	6922
j500c5a03	3378	j500c10a03	6668
j500c5a04	6516	j500c10a04	6541
j500c5a05	6497	j500c10a05	6779
j500c5a06	6599	j500c10a06	6485
j500c5a07	6691	j500c10a07	6920
j500c5a08	6185	j500c10a08	6784
j500c5a09	6174	j500c10a09	6477
j500c5a10	4416	j500c10a10	6451
j500c5b01	3316	j500c10b01	3399
j500c5b02	3106	j500c10b02	3429
j500c5b03	3311	j500c10b03	3483
j500c5b04	3174	j500c10b04	3545
j500c5b05	3231	j500c10b05	3484
j500c5b06	3278	j500c10b06	3457
j500c5b07	3352	j500c10b07	3483
j500c5b08	3271	j500c10b08	3492
j500c5b09	3251	j500c10b09	3530
j500c5b10	3286	j500c10b10	3493
j500c5c01	12858	j500c10c01	13269
j500c5c02	12794	j500c10c02	12602
j500c5c03	12911	j500c10c03	13424
j500c5c04	12993	j500c10c04	13082
j500c5c05	8780	j500c10c05	13250
j500c5c06	12572	j500c10c06	12547
j500c5c07	12309	j500c10c07	13195
j500c5c08	6408	j500c10c08	12780
j500c5c09	13049	j500c10c09	12951
j500c5c10	6390	j500c10c10	13262
j500c5d01	6598	j500c10d01	6643
j500c5d02	6476	j500c10d02	7012
j500c5d03	6619	j500c10d03	6615
j500c5d04	6520	j500c10d04	6934
j500c5d05	6477	j500c10d05	6982
j500c5d06	6467	j500c10d06	6850
j500c5d07	6542	j500c10d07	6976
j500c5d08	6417	j500c10d08	6907
j500c5d09	6559	j500c10d09	6881
j500c5d10	6735	j500c10d10	6873

8 Literatura

[1] Kemal Alaykỳran, Orhan Engin, and Alper Döyen. Using ant colony optimization to solve hybrid flow shop scheduling problems. *The international journal of advanced manufacturing technology*, 35(5-6):541–550, 2007.

[2] Mohammad Reza Amin-Naseri and Mohammad Ali Beheshti-Nia. Hybrid flow shop scheduling with parallel batching. *International Journal of Production Economics*, 117(1):185–196, 2009.

[3] TS Arthanari and KG Ramamurthy. An extension of two machines sequencing problem. *Opsearch*, 8(1):10–22, 1971.

[4] Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2):126–140, 1994.

[5] Walid Besbes, Taicir Loukil, and Jacques Teghem. Using genetic algorithm in the multiprocessor flow shop to minimize the makespan. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1228–1233. IEEE, 2006.

[6] Shaukat A Brah and John L Hunsucker. Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research*, 51(1):88–99, 1991.

[7] Jacques Carlier and Emmanuel Néron. An exact method for solving the multi-processor flow-shop. *RAIRO-Operations Research*, 34(01):1–25, 2000.

[8] Tsui-Ping Chung and Ching-Jong Liao. An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem. *Applied Soft Computing*, 13(8):3729 – 3736, 2013.

[9] Zhe Cui and Xingsheng Gu. An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems. *Neurocomputing*, 148(0):248 – 259, 2015.

[10] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006.

[11] Russ C Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[12] Orhan Engin and Alper Döyen. A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, 20(6):1083–1095, 2004.

[13] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[14] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[15] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[16] Fred Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Springer, 2003.

[17] Jozef Grabowski and Jaroslaw Pempera. Sequencing of jobs in some production system. *European Journal of Operational Research*, 125(3):535 – 550, 2000.

[18] Jatinder ND Gupta. Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, pages 359–364, 1988.

[19] Pierre Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on numerical methods in combinatorial optimization, Capri, Italy*, pages 70–145, 1986.

[20] Steven A Hofmeyr and Stephanie Forrest. Architecture for an artificial immune system. *Evolutionary computation*, 8(4):443–473, 2000.

[21] Adam Janiak, Erhan Kozan, Maciej Lichtenstein, and Ceyda Oğuz. Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *International journal of production economics*, 105(2):407–424, 2007.

[22] Cengiz Kahraman, Orhan Engin, İhsan Kaya, and R Elif Öztürk. Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach. *Applied Soft Computing*, 10(4):1293–1300, 2010.

[23] Safa Khalouli, Fatima Ghedjati, and Abdelaziz Hamzaoui. An integrated ant colony optimization algorithm for the hybrid flow shop scheduling problem. In *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on*, pages 554–559. IEEE, 2009.

[24] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

[25] Sergio Ledesma, Jose Ruiz, and Guadalupe Garcia. Simulated annealing evolution. *Simulated Annealing–Advances, Applications and Hybridizations, Intech Publishing, Croatia*, pages 210–218, 2012.

[26] Jan Karel Lenstra and Alexander Rinnooy Kan. Some simple applications of the travelling salesman problem. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, (BW 38/74):1–36, 1974.

[27] Ching-Jong Liao, Evi Tjandradjaja, and Tsui-Ping Chung. An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing*, 12(6):1755 – 1764, 2012.

[28] Miroslav Marić. An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem. *Computing and Informatics*, 29(2):183–201, 2012.

[29] Miroslav Marić, Zorica Stanimirović, and Srdjan Božović. Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, pages 1–21, 2013.

[30] Miroslav Marić, Zorica Stanimirović, and Predrag Stanojević. An efficient memetic algorithm for the uncapacitated single allocation hub location problem. *Soft Computing*, 17(3):445–466, 2013.

[31] Nenad Mladenovic. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. In *papers presented at Optimization Days*, page 112, 1995.

[32] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[33] Jose A Moreno Perez, J Marcos Moreno-Vega, and Inmaculada Rodriguez Martin. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151(2):365–378, 2003.

[34] Emmanuel Néron, Philippe Baptiste, and Jatinder ND Gupta. Solving hybrid flow shop problem using energetic reasoning and global operations. *Omega*, 29(6):501–511, 2001.

[35] Qun Niu, Taijin Zhou, and Shiwei Ma. A quantum-inspired immune algorithm for hybrid flow shop with makespan criterion. *J. UCS*, 15(4):765–785, 2009.

[36] Ceyda Oğuz and M Fikret Ercan. A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, 8(4):323–351, 2005.

[37] Marie-Claude Portmann, Antony Vignier, David Dardilhac, and David Dezalay. Branch and bound crossed with ga to solve hybrid flowshops. *European Journal of Operational Research*, 107(2):389–400, 1998.

[38] Jun qing Li, Quan ke Pan, and Fa tao Wang. A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem. *Applied Soft Computing*, 24(0):63 – 77, 2014.

[39] Rubén Ruiz and José Antonio Vázquez-Rodríguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18, 2010.

[40] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.