Razvoj pravila raspoređivanja korištenjem genetskog programiranja

Marko Đurasević
Fakultet elektrotehnike i računarstva
Unska 3, Zagreb
Email: marko.durasevic@fer.hr

***Sažetak*— Raspoređivanje je postupak koji se primjenjuje u mnogim područjima ljudske djelatnosti, a posebnu važnost ima u proizvodnim procesima u kojima je iznimno bitno postići što veću učinkovitost. Iz tog razloga potrebni su postupci kojima će se moći izgraditi efikasni rasporedi i koji će moći biti primijenjeni u praksi.**

U ovom radu dana je formalna definicija raspoređivanja te su opisana neka od često korištenih okruženja raspoređivanja iz literature. Također su opisani različiti postupci koji se mogu iskoristiti za stvaranje rasporeda. Poseban naglasak stavljen na razvoj novih pravila raspoređivanja korištenjem genetskog programiranja te je dan detaljan pregled literature koja se bavi upravo ovim područjem pri čemu su kroz rad istaknuti najnoviji smjerovi istraživanja u tom području.

***Ključne riječi*— genetsko programiranje, raspoređivanje, pravila raspoređivanja.**

I. Uvod

Raspoređivanje je proces u kojem se određeni skup sredstava dodjeljuje određenom skupu aktivnosti [1]. Rješenje problema raspoređivanja jest raspored koji određuje koje će aktivnosti biti raspoređene na koja sredstva i kojim redoslijedom. Cilj raspoređivanja jest izgraditi raspored koji zadovoljava i optimira određene kriterije (primjerice minimizacija ukupne duljine rasporeda ili minimizacija maksimalnog kašnjenja nekog posla). Veoma često se pojavljuje i potreba za optimizacijom nekoliko kriterija u isto vrijeme, koji mogu biti međusobno oprečni, što problem čini još težim. Nažalost, velika većina problema raspoređivanja spada u klasu NP teških problema što znači da nisu rješivi metodama iscrpne pretrage u nekom "razumnom" vremenskom periodu, već je potrebno koristiti različite heurističke, odnosno metaheurističke postupke, kako bi se dobila rješenja za zadane probleme.

Raspoređivanje ima veliku primjenu u mnogim granama ljudske djelatnosti. Tako se na primjer proces raspoređivanja primjenjuje u zrakoplovstvu [2][3], bolnicama [4], proizvodnji poluvodiča [5][6], heterogenim računalnim sustavima [7][8][9][10], *cloud* okruženju [11] i slično. Ono što raspoređivanje čini još težim jest to što nije moguće razviti jedan jedinstveni postupak koji bi se mogao primijeniti u svim domenama, jer su u svakoj domeni postavljena različita ograničenja i različiti zahtjevi nad sam raspored. Osim toga, od postupaka se često očekuje da se oni mogu iznimno brzo prilagoditi promjenama koje se mogu dogoditi u sustavu i brzo izgraditi novi raspored (primjerice pojava novog pacijenta ili hitnog zahtjeva za nekim proizvodom, kvar određenog stroja i slično). Jedna olakotna okolnost u raspoređivanju jest ta da najčešće nije potrebno pronaći optimalan raspored, već neki raspored koji je dovoljno "dobar" za zadane kriterije.

U ovom radu dan je pregled postupaka koji se mogu koristiti za izradu rasporeda, pri čemu je posebno stavljen naglasak na primjenu genetskog programiranja s ciljem automatske izrade pravila raspoređivanja. U II. poglavlju dana je formalna definicija raspoređivanja te su opisani modeli koji se najčešće pojavljuju u literaturi. U III. poglavlju kategorizirani su postupci koji se mogu primijeniti na izradu rasporeda te su neki od njih ukratko opisani. Kratki opis genetskog programiranja i njegova primjena u izradi pravila raspoređivanja s detaljnim pregledom literature dani su u IV. poglavlju. Konačno, u V. poglavlju dan je kratak zaključak i navedene su neke moguće smjernice za daljnje istraživanje.

II. Problem raspoređivanja

Kao što je već ranije spomenuto, raspoređivanje je postupak kojim se ograničeni skup sredstava dodjeljuje nekom skupu aktivnosti. U literaturi se često koriste ponešto drugačiji nazivi, pa se tako sredstva nazivaju strojevima, dok se aktivnosti nazivaju poslovima. Ukupni broj dostupnih strojeva se najčešće označava s m, dok se ukupni broj poslova označava s n. Konkretan posao se obično označava indeksom j, dok se konkretan stroj označava indeksom i. Svaki posao koji je potrebno rasporediti ima definirane određene atribute koji određuju svojstva tog posla. Atributi poslova koji se najčešće koriste u raspoređivanju su [1][12]:

- **Trajanje izvođenja** p_j (engl. *processing time*) - označava trajanje izvođenja posla j. Ako trajanje posla ovisi o stroju na kojem se izvodi, onda se koristi oznaka p_{ij}, gdje i predstavlja indeks stroja na kojem se posao izvodi.
- **Vrijeme pripravnosti posla** r_j (engl. *release time*) - označava vrijeme u kojem posao postaje dostupan u sustavu. Prije tog vremena posao se ne može izvršavati na nekom od strojeva.
- **Vrijeme željenog završetka** d_j (engl. *due date*) - označava vrijeme do kojeg je poželjno da posao završi s izvođenjem (odnosno ako posao ne završi do tog vremena, stvara se određeni gubitak).
- **Vrijeme nužnog završetka** $\bar{d}_j$ (engl. *deadline*) - označava vrijeme do kojeg posao nužno mora završiti.
- **Težina posla** w_j (engl. *weight*) - označava važnost pojedinog posla.

Kako postoji velik broj problema raspoređivanja, javila se je potreba za uvođenjem jedinstvene nomenklature pomoću koje se navedeni problemi mogu opisati. Tako se problem raspoređivanja može opisati uređenom trojkom $\alpha|\beta|\gamma$, gdje α predstavlja okolinu strojeva, β predstavlja određene karakteristike problema, a γ kriterij koji se optimira. U nastavku je dan pregled nekih mogućih vrijednosti za svaki element trojke.

A. Okolina strojeva

Element α predstavlja okolinu strojeva koja se koristi u raspoređivanju te sadrži samo jedan element. Najčešće korištena (naravno ne i jedina) okruženja u literaturi su [1][12]:

- **Jedan stroj** 1 (engl. *single machine*) - postoji samo jedan stroj na kojem se poslovi izvode.
- **Paralelni identični strojevi** Pm (engl. *parallel identical machines*) - postoji m jednakih strojeva i obrada svakog posla je jednaka na svakom stroju.
- **Jednoliki strojevi** Qm (engl. *uniform machines*) - postoji m strojeva i svaki stroj ima zasebnu brzinu s_i. Izvođenje posla na nekom stroju se onda računa kao $p_{ij} = \frac{p_j}{s_i}$.
- **Nesrodni strojevi** Rm (engl. *nesrodni strojevi*) - postoji m strojeva i stroj obrađuje svaki posao proizvoljno definiranom brzinom.
- **Obrada tijeka** Fm (engl. *flow shop*) - svaki posao se sastoji od broja operacija koji je jednak broju strojeva, pri čemu se svaka operacija mora obaviti na zasebnom stroju i to redoslijedom koji je isti za sve poslove.
- **Proizvoljna obrada** Jm (engl. *job shop*) - svaki posao se sastoji od broja operacija koji je jednak broju strojeva, no redoslijed izvođenja tih operacija po strojevima se može razlikovati za svaki posao.
- **Otvorena obrada** Om (engl. *open shop*) - svaki posao se sastoji od određenog broja operacija koji ne mora biti jednak broju strojeva, i redoslijed izvođenja tih operacija po strojevima se može razlikovati za svaki posao.

B. Karakteristike problema raspoređivanja

Element β predstavlja određene karakteristike problema raspoređivanja te može sadržavati proizvoljan broj elemenata (može biti i prazan). Najčešće korištene karakteristike su [1][12]:

- **Prekidivost** $prmp$ (engl. *preemption*) - posao koji se izvodi može prekinuti s izvođenjem i nastaviti s izvođenjem kasnije bez gubitaka.
- **Ograničenje redoslijeda** $prec$ (engl. *precedence constraints*) - određeni poslovi ne mogu početi s izvođenjem ako prethodno nisu izvršeni neki drugi poslovi.
- **Trajanje postavljanja** s_{jk} (engl. *setup time*) - trajanje potrebno da se stroj pripremi za izvođenje posla j ako se je na njemu prije izvodio posao k.
- **Vremena pripravnosti** r_j (engl. *release dates*) - koriste se vremena pripravnosti poslova. Ako ovaj element nije zadan, podrazumijeva se da su svi poslovi dostupni od početka rada sustava.
- **Kvarovi** *brkdwn* (engl. *breakdown*) - strojevi ne moraju biti raspoloživi cijelo vrijeme, odnosno mogu se "pokvariti".

C. Kriteriji optimizacije

Element γ predstavlja kriterij koji je potrebno optimirati te se najčešće sastoji od samo jednog elementa. Najčešći kriteriji koji se optimiraju su [1][12]:

- **Ukupna duljina rasporeda** C_{max} (engl. *makespan*) - $C_{max} = \max(C_j)$, gdje je C_j vrijeme završetka posla j.
- **Težinsko kašnjenje** TWT (engl. *total weighted tardines*) - $TWT = \sum_j w_j T_j$, gdje je $T_j = \max(0, C_j - d_j)$.
- **Težinsko protjecanje** FWT (engl. *total weighted flowtime*) - $FWT = \sum_j w_j F_j$, gdje je $F_j = C_j - r_j$.
- **Težinski zbroj zakašnjeih poslova** U_w (engl. *wighted number of tardy jobs*) - $U_w = \sum_j w_j U_j$, gdje je

$$U_j = \begin{cases} 1 & T_j > 0 \\ 0 & T_j = 0 \end{cases}.$$

D. Uvjeti i načini izrade rasporeda

Za kraj je još potrebno navesti i različite uvjete i načine izrade rasporeda o kojima će uvelike ovisiti postupci koji će biti primjenjivi za izradu rasporeda. Ovisno o raspoloživosti parametara razlikujemo [12]:

- **Predodređeno raspoređivanje** (engl. *offline scheduling*) - vrijednosti svih potrebnih parametara su dostupne prije početka izrade rasporeda i izvođenja sustava (poznati su svi poslovi koji će se pojaviti, kao i svi atributi tih poslova).
- **Raspoređivanje na zahtjev** (engl. *online scheduling*) - nisu poznate vrijednosti svih parametara unaprijed. Primjerice, podaci o poslu postanu dostupni tek onda kada sam posao postane dostupan.

Neovisno o prethodnoj podjeli, u ovisnosti o pouzdanosti parametara raspoređivanje se može podijeliti na:

- **Determinističko raspoređivanje** (engl. *deterministic scheduling*) - vrijednosti parametara su pouzdane.
- **Stohastičko raspoređivanje** (engl. *stochastic scheduling*) - vrijednosti parametara ne mogu se pouzdano procijeniti, već se one modeliraju određenim vjerojatnosnim raspodjelama.

Ovisno o tome kako se raspored izrađuje razlikujemo:

- **Statičko raspoređivanje** (engl. *static scheduling*) - cjelokupni raspored se izradi prije početka rada sustava.
- **Dinamičko raspoređivanje** (engl. *dynamic scheduling*) - raspored se izrađuje paralelno s radom sustava.

III. Postupci izrade rasporeda

Problem raspoređivanja može se riješiti korištenjem raznih postupaka, no o karakteristikama samog problema ovisit će koji će postupci biti najpogodniji. Postupke koji se primjenjuju za rješavanje problema raspoređivanja općenito možemo podijeliti na egzaktne postupke, aproksimativne postupke i heurističke postupke. Egzaktni postupci, kao što im i samo ime kaže, mogu dati točno rješenje problema raspoređivanja. U ove postupke ubrajamo različite oblike matematičkog programiranja (linearno i cjelobrojno programiranje) [1], postupak *branch and bound* [1][13] i dinamičko programiranje [1]. Nedostatak ovih postupaka je taj što su oni računalno iznimno zahtjevni i neprimjenjivi za složenije probleme raspoređivanja. Iz tog razloga razvijeni su aproksimativni algoritmi, koji rade kompromis između trajanja izvođenja i optimalnosti rješenja. Ovi algoritmi pronalaze rješenje u polinomijalnom vremenu, no ne garantiraju pronalazak optimalnog rješenja, već rješenja koja su za određeni postotak lošija od optimalnog rješenja [14]. Nedostatak ovih postupaka je taj što su primjenjivi isključivo za slučaj statičkog raspoređivanja.

Heuristički postupci također ne garantiraju pronalazak optimalnog rješenja, već dovoljno "dobrog" rješenja, no bez ikakve informacije koliko će dobiveno rješenje biti udaljeno od optimalnog rješenja. Heurističke postupke općenito dijelimo na konstruktivne i unapređivačke.

Unapređivački postupci započinju s nekim kompletnim rasporedom i nastoje ga unaprijediti različitim manipulacijama. U ove postupke ubrajamo simulirano kaljenje [15], tabu pretragu [16], evolucijske algoritme [17][18], umjetnu koloniju mrava [19] i slične metaheurističke postupke. Prednost ovih postupaka je što mogu naći jako dobra rješenja za različita okruženja raspoređivanja i različite kriterije. S druge strane, ti algoritmi su računalno dosta zahtjevni te ih je zbog njihove stohastičke prirode potrebno pokrenuti i po nekoliko puta kako bismo bili sigurni da su dobivena dobra rješenja. Ovi postupci su također primjenjivi uglavnom u predodređenom raspoređivanju, jer su im potrebne sve informacije o sustavu prije nego što izrade raspored i ne mogu brzo reagirati na promjene u sustavu.

Istraživanje na području unapređivačkih postupaka je iznimno aktivno. U [20] i [21] dan je iznimno detaljan pregled primjene evolucijskog računarstva za izradu rasporeda. Zadnjih nekoliko godina naglasak je stavljen na poboljšanju ovih postupaka kako bi se postigli što bolji rezultati [22][23][24][25][26][27]. U [28] i [29] dan je detaljan pregled raspoređivanja s vremenima postavljanja, dok je u [30] dana primjena genetskog algoritma na problem raspoređivanja s ograničenjima redoslijeda među poslovima. Od novijih inačica evolucijskih postupaka može se istaknuti primjena memetičkih algoritama [31], imperijalističkih evolucijskih algoritama [32], biogeografijskih optimizacijskih algoritama [33] i imunoloških algoritama [34]. Također je zadnjih godina stavljen sve veći naglasak i na višekriterijsku optimizaciju [25][35][36].

Konstruktivni heuristički postupci započinju s praznim rasporedom te iterativno izgrađuju raspored dodajući posao po posao u raspored. Prednost ovih postupaka je što su iznimno brzi i jednostavni. Zbog svoje brzine, kao i zbog toga što veoma brzo i lako mogu reagirati na promjene, oni su često primjenjivi u dinamičkom raspoređivanju. Doduše, njihov najveći nedostatak također proizlazi iz njihove jednostavnosti. Naime, ovi postupci najčešće nisu u stanju izgraditi rasporede jednake kvaliteta, kao što su to rasporedi dobiveni korištenjem unapređivačkih postupaka. U ove postupke se ubrajaju pravila raspoređivanja te neki postupci pohlepnog pretraživanja [37][38][39].

Pravila raspoređivanja su jednostavni postupci koji određuju koji će se od trenutno dostupnih poslova rasporediti na koji od trenutno dostupnih strojeva. To rade na način da svakom paru posao-stroj odrede prioritet te rasporede onaj par koji ima najveći (ili najmanji) prioritet. Iz opisa se dakle može zaključiti da se pravila raspoređivanja uglavnom razlikuju po tome kako se izračunavaju prioriteti. Ovdje odmah dolazi do izražaja i njihov nedostatak, a taj je da kvalitetna pravila nije lako razviti i da se nova pravila moraju razviti za svaki novi kriterij i za svaku kombinaciju kriterija raspoređivanja. Zbog svojih karakteristika pravila raspoređivanja najčešće se koriste u stvarnim sustavima za izradu rasporeda. Iz tog razloga puno istraživanja bavilo se je upravo razvojem pravila raspoređivanja za različita okruženja. U [1] [40] dan je pregled raznih pravila raspoređivanja. U radu [41] opisano je pravilo raspoređivanja za jedan stroj koje uzima u obzir vremena postavljanja. U [8] i [42] dan je opširan pregled pravila raspoređivanja za nesrodne strojeve. Zadnjih nekoliko godina razvijeno je mnogo novih pravila raspoređivanja za nesrodne strojeve [7][9][43][44][45][46], što pokazuje kako je razvoj pravila raspoređivanja još uvijek dosta aktivno područje. Važnost pravila raspoređivanja svakako pokazuju i radovi koji se bave pregledom pravila raspoređivanja primijenjenih u praksi [5][6].

Genetski algoritmi ne moraju se uvijek nužno iskoristiti za rješavanje problema raspoređivanja što je pokazano u [47]. U tom radu genetski algoritmi su iskorišteni kako bi se pronašle instance problema raspoređivanja za koje ljudski razvijena pravila raspoređivanja donose suboptimalne odluke i time izrađuju loše rasporede. Na taj način genetski algoritmi mogu se iskoristiti kako bi se pronašle slabosti u pravilima raspoređivanja i kako bi se one ispravile ili kako bi se izradila nova pravila raspoređivanja prilagođena za takve situacije.

Kako postoji veliki broj okruženja strojeva s različitim karakteristikama i kriterijima te jednostavno nije moguće za svaku kombinaciju ručno izraditi kvalitetna pravila raspoređivanja, javila se je potreba da se na neki način automatizira izrada takvih postupaka, ali i za određivanje koja su pravila raspoređivanja prikladna za koje probleme. Genetski algoritmi često se koriste kako bi se odredio redoslijed primjene pravila raspoređivanja u sustavu [48]. U [49] koristi se metoda Gaussovih procesa kako bi se s obzirom na određena svojstva sustava odredilo koje je pravilo raspoređivanja najprikladnije. U svrhu izrade novih pravila često se koriste metode strojnog učenja kako bi se izradili modeli koji na temelju trenutnog stanja sustava mogu odrediti koji bi posao trebalo rasporediti na koji stroj. U tu svrhu koriste se različiti postupci strojnog učenja kao neuronske mreže [50], stabla odlučivanja [51][52], linearna klasifikacija [53] i mnogi drugi postupci otkrivanja znanja iz podataka [54][55][56][57]. Osim postupaka strojnog učenja za izradu novih pravila nerijetko se još koriste i neizrazita logika [58][59][60] kao i genetsko programiranje koje će u idućem poglavlju biti detaljno obrađeno.

IV. Izrada pravila raspoređivanja genetskim programiranjem

Genetsko programiranje je vrsta evolucijskog računanja iznimno slična genetskim algoritmima, no od kojih se razlikuje po tome što jedinke u genetskom programiranju predstavljaju određene matematičke izraze ili računalne programe [61][62][63]. Ti izrazi predstavljeni su u obliku stabla čiji unutarnji čvorovi predstavljaju matematičke operatore te se nazivaju funkcijskim čvorovima, dok listovi stabla predstavljaju određene konstante odnosno varijable i nazivaju se terminalnim čvorovima. U prvom koraku GP generira početni skup potencijalnih tješenja, pri čemu pojedino rješenje nazivamo jedinkom, dok skup trenutnih rješenja nazivamo populacijom. Početni skup rješenja se najčešće generira nasumično, no nerijetko se koriste i neki napredniji postupci kako bi se poboljšala kvaliteta početnog skupa rješenja. Cilj GP-a jest korištenjem inicijalnog skupa rješenja pronaći neko dovoljno "dobro" rješenje. To se postiže na način da se nad populacijom primjenjuju različiti genetski operatori kao što su križanje, mutacija i selekcija. Križanje je genetski operator koji kombinira dvije jedinke u jednu novu jedinku s ciljem da njihova kombinacija bude bolja. Nad novodobivenom jedinkom se tada često primjenjuje operator mutacije, koji uvodi slučajne promjene u jedinku. Tim slučajnim promjenama želi se spriječiti brza konvergencija u neki od lokalnih minimuma i natjerati algoritam na istraživanje šireg područja. Jedinka koja je dobivena križanjem i mutacijom se tada ubacuje nazad u populaciju na način da zamijeni neku od postojećih jedinki. Operator selekcije određuje kako će se odabrati jedinke koje će se križati, kao i jedinka koja će biti zamijenjena. Jedinke koje su "bolje" imat će veću vjerojatnost da budu odabrane za križanje od "lošijih". Kvalitetu jedinki određujemo na način da ispitamo kvalitetu rješenja koje one predstavljaju nad nekim skupom problema te na neki način mjerimo njihovu uspješnost.

Genetsko programiranje naišlo je na široku primjenu i korištenjem genetskog programiranja ostvareni su mnogi rezultati bolji ili jednako dobri kao rezultati ostvareni od strane ljudi [64]. Također, genetsko programiranje je naišlo i na široku primjenu u klasifikaciji [65][66]. Upravo iz razloga što genetsko programiranje jedinke predstavlja u obliku matematičkih izraza ono se može upotrijebiti kao hiper-heuristika, odnosno kao postupak koji može izgraditi nove heurističke postupke [67][68]. U slučaju raspoređivanja to će biti upravo pravila raspoređivanja opisana u prošlom poglavlju.

Početak korištenja genetskog programiranja u svrhu izrade pravila raspoređivanja započelo je prije petnaestak godina s radovima [69] i [70]. U tim radovima veći naglasak je bio na tome da se genetsko programiranje koristi kako bi se zapravo odredio redoslijed primjene jednostavnih pravila raspoređivanja (slično kao korištenjem genetskog algoritma opisanog u prošlom poglavlju), dok je veoma mali naglasak stavljen na samu izradu novih pravila raspoređivanja. Prvi rad koji se je isključivo bavio izradom pravila raspoređivanja korištenjem GP-a bio je [71]. Jedinke u GP-u predstavljaju matematičku formulu koja se koristi za određivanje prioriteta pojedinih poslova, pri čemu funkcijski čvorovi predstavljaju matematičke ili logičke izraze (primjerice zbrajanje, množenje, minimum, naredbe grananja), dok terminalni čvorovi predstavljaju varijable koje označavaju svojstva poslova ili trenutno stanje sustava (primjerice trajanje izvođenja poslova, trenutno vrijeme sustava i slično). U tom radu izrađeno je nekoliko pravila raspoređivanja za okruženje s proizvoljnom obradom. Također, u tom radu isprobana su tri modela raspoređivanja. U prvom modelu svaki stroj ima jednako pravilo raspoređivanja, u drugom modelu svaki stroj ima zasebno pravilo raspoređivanja i u trećem modelu svi strojevi imaju jednako pravilo raspoređivanja osim jednog stroja koji predstavlja usko grlo (engl. *bottleneck*) sustava i imao vlastito pravilo raspoređivanja. Iako se je treći model pokazao kao najuspješniji, njegov nedostatak je taj što zahtijeva određeno predznanje o sustavu jer je potrebno unaprijed odrediti usko grlo sustava.

Od tada pa nadalje veliki fokus istraživanja je upravo stavljen na primjenu genetskog programiranja za izradu novih pravila raspoređivanja. U [72] evoluirana su pravila raspoređivanja koja uzimaju u obzir kvarove strojeva i u usporedbi s klasičnim heuristikama, evoluirane heuristike su postigle bolje rezultate. U radu [73] GP je primijenjen za izradu pravila raspoređivanja u okruženju jednog stroja te su dobiveni rezultati uspoređeni s nekim klasičnim heuristikama. Pokazalo se je kako su heuristike evoluirane od strane GP-a kompetitivne s ljudski razvijenim heuristikama. Da evoluirane heuristike mogu biti i značajno bolje od klasičnih heuristika pokazano je u [74]. U tom radu je također osmišljen i postupak za primjenu u okruženju proizvoljne obrade koji rješava problem s detekcijom stroja koji predstavlja usko grlo iz [71]. Naime, umjesto jednog matematičkog izraza, evoluiraju se tri izraza, gdje prvi izraz služi kao diskriminativni izraz, koji bi trebao odrediti da li trenutni stroj predstavlja usko grlo ili ne, te ovisno o rezultatu koristi se drugi, odnosno treći izraz.

Izrada pravila raspoređivanja korištenjem GP-a u okruženju paralelnih strojeva je nažalost jako slabo obrađeno te je jedan od rijetkih radova koji se time bavi [75]. U navedenom radu naglasak je bio isključivo na proporcionalne strojeve i pokazano je kako se i za ovo okruženje mogu postići rezultati koji su bolji ili kompetitivni s klasičnim heuristikama. Također su u ovom radu evoluirana i pravila raspoređivanja za slučaj kada su bila prisutna vremena postavljanja između poslova. I u ovom slučaju su se pravila raspoređivanja razvijena GP-om pokazala boljima. Primjena GP za izradu pravila raspoređivanja u okruženju nesrodnih strojeva kratko je dotaknuta u radovima [12] i [76]. U tim radovima pokazano je kako dobiveni postupci raspoređivanja postižu rezultate koji su uglavnom bolji od rezultata dobivenih od strane klasičnih pravila raspoređivanja.

Prva primjena GP-a za evoluiranje pravila raspoređivanja u svrhu minimizaciju više kriterija opisana je u [77]. Iako je višekriterijska optimizacija bila ostvarena na veoma jednostavan način, kao srednja vrijednost triju različitih kriterija, rezultatima je pokazano kako su evoluirana pravila raspoređivanja postigla bolje rezultate od klasičnih postupaka raspoređivanja. Doduše u [78] je pokazano kako rezultati iz prethodnog rada ipak nisu toliko dobri kada se primjene u dinamičkom raspoređivanju iz razloga što su navedena pravila bila prilagođena za statičko raspoređivanje. U tom radu veliki naglasak je stavljen upravo na dobru izradu primjera za učenje te je donesen zaključak kako su najbolji rezultati dobiveni za slučaj kada se u svakoj generaciji GP-a koriste novogenerirani primjeri za učenje. U radu su postignuti bolji rezultati od klasičnih heuristika, te je također isprobana varijanta GP-a koja koristi takozvani "pogled unaprijed" (engl. *lookahead*) te

uzima u obzir i neke poslove koji dolaze u bliskoj budućnosti, no ta varijanta nije postigla značajno bolje rezultate.

Inačica GP-a pod nazivom GEP (engl. *gene expression programming*) [79][80] prvi put je upotrijebljena za izradu pravila raspoređivanja u [81]. U navedenom radu GEP je primijenjen za izradu pravila raspoređivanja u okruženju jednog stroja. Pokazano je kako GEP može pronaći pravila raspoređivanja koja postižu ponešto bolje rezultate od pravila raspoređivanja evoluiranih GP-om. Ostale prednosti GEP-a su brža konvergencija te da su pronađena pravila raspoređivanja ponešto jednostavnija od onih dobivenih GP-om. Isti autori su u [82] GEP iskoristili za izradu pravila raspoređivanja u okruženju proizvoljne obrade i također pokazali kako su evoluiranim pravilima dobiveni rezultati koji su bolji od rezultata dobivenih klasičnim heuristikama. Bez obzira na prikazane prednosti, GEP još uvijek nije toliko prihvaćen kao GP za izradu pravila raspoređivanja.

Značajna analiza kvalitete rasporeda dobivenih korištenjem evoluiranih pravila raspoređivanja u ovisnosti o složenosti izraza pravila raspoređivanja napravljena je u [83]. Ustanovljeno je kako sve složenija pravila uistinu postižu bolje rezultate, no da kod složenijih pravila postoji velika mogućnost da je došlo do prenaučenosti i specijaliziranosti za posebne slučajeve. U radu [84] dana je detaljna analiza parametara genetskog programiranja kako bi se ustanovili idealni parametri (oni koji neće rezultirati prenaučenosti). U ovom radu je također proučeno i okruženje jednog stroja u kojem su definirana ograničenja u redoslijedu izvođenja poslova kao i vremena postavljanja između poslova. Pravila raspoređivanja su postigla uglavnom bolje rezultate od svih klasičnih heuristika.

U [85] napravljena je analiza prikaza u GP-u. Analizirana su tri različita prikaza. Prvi prikaz umjesto prioriteta poslova određuje koju je jednostavnu heuristiku potrebno upotrijebiti s obzirom na trenutno stanje sustava. Drugi prikaz koristi aritmetički prikaz koji određuje prioritete poslova, dok je treći prikaz kombinacija prva dva. Osim samih pravila raspoređivanja evoluirana je vrijednost pod nazivom *nondelay* koja određuje koliki će pogled u budućnost sustavu biti na raspolaganju. Kroz detaljnu analizu rezultata pokazano je kako su najbolji rezultati postignuti za treću vrstu prikaza. Također je pokazano kako rezultati dobiveni pravilima raspoređivanja izrađenim pomoću GP-a još uvijek zaostaju za rezultatima koji su postignuti primjerice genetskim algoritmima, što je i očekivano.

Zanimljiva kombinacija genetskog algoritma i GP-a prikazana je u [86]. U ovom radu GP se koristi da se pronađu dobra pravila raspoređivanja, dok se genetski algoritam koristi da se pronađe najbolja podjela pravila raspoređivanja po strojevima (jer je svaki stroj koristio vlastito pravilo raspoređivanja). Rezultati su pokazali kako je ovaj pristup dao bolje rezultate nego kada je primijenjeno samo jedno pravilo raspoređivanja za sve strojeve.

Iznimno inovativan postupak prikazan je u radu [87]. U tom radu GP je primijenjen na evoluiranje iterativnih pravila raspoređivanja. Iterativna pravila raspoređivanja, osim što koriste informacije o trenutnom stanju sustava, koriste i informacije o rasporedima koja su ta pravila prethodno izradila (primjerice broj zakašnjelih poslova). Iz tog razloga raspored je potrebno izraditi nekoliko puta iterativnom primjenom takvog pravila raspoređivanja. Iz toga odmah proizlazi da takva pravila mogu biti primijenjena samo u statičkom raspoređivanju. U radu je također napravljena i detaljna analiza o tome na koje je početne vrijednosti potrebno postaviti informacije prošlih rasporeda u prvoj iteraciji kada ne postoji prethodno izgrađeni raspored. Osim toga isprobana je i inačica s "pogledom unaprijed" i jedna druga inačica koja koristi postupak lokalne pretrage pod nazivom varijabilna pretraga susjedstva (engl. *variable neighbourhood search*) [88].

Problem heuristika izrađenih pomoću GP-a jest njihova "kratkovidnost" odnosno slab globalni pogled na sam sustav. Iz tog razloga jedan dio istraživanja je orijentiran i na poboljšanje pravila raspoređivanja u tom pogledu. U radu [89] to se je nastojalo postići proširenjem svojstava sustava koja su na raspolaganju GP-u, u obliku terminalnih čvorova, dok izgrađuje pravila raspoređivanja. Već i tim jednostavnim poboljšanjem pokazano je kako su dobiveni bolji rezultati od osnovnog GP-a.

Korištenjem GP-a se mogu poboljšati i neki postojeći postupci. Primjerice, NEH heuristika koja se primjenjuje za okruženje permutacijske obrade tijeka. Navedeni postupak poslove interno rangira na određeni način (mogli bismo reći po određenom prioritetu). Korištenjem genetskog programiranja moguće je pronaći alternativni izraz za računanje prioriteta za koji će navedena heuristika postići bolje rezultate [48]

U radu [90] pokazano je kako GP može uspješno pronaći optimalno pravilo raspoređivanja za poseban slučaj okruženja proizvoljne obrade u statičkom raspoređivanju. Time je pokazano kako GP stvarno može pronaći iznimno dobra pravila raspoređivanja. Također je u ovom radu proučeno nekoliko varijanti GP-a. Isprobana je varijanta GP-a koja za svaki stroj izrađuje posebno pravilo raspoređivanja i varijanta koja evoluira jedno pravilo za sve strojeve. Također je isprobana varijanta GP-a koja svaku generaciju stvara nove primjere za učenje i varijanta koja svakih deset generacija stvara nove primjere za učenje. Nažalost iz rezultata se nije moglo zaključiti koje su od ovih metoda bolje, jer ovisno o ispitnim primjerima pojedine metode su se pokazale boljima, no niti jedna se nije pokazala dominantnom kroz sve ispitne primjere.

Mogućnost primjene GP-a i za ponešto drugačije probleme raspoređivanja pokazana je u [91] i [92]. Ovi radovi se bave malo drugačijim problemom raspoređivanja pod nazivom OAS (engl. *order acceptance and scheduling*). Kod ovog problema raspoređivanja nije potrebno samo rasporediti poslove po strojevima, već i odabrati koji će se poslovi prihvatiti kako bi se maksimizirao dobitak. U [91] prikazana je osnovna ideja tog postupka i predložena su dva modela. Prvi model se sastoji samo od jednog izraza koji određuje prioritete poslova, ali se koristi i za odluku o tome hoće li se neki posao prihvatiti ili ne i to na način da iznos prioriteta mora biti veći od neke unaprijed definirane konstante da bi se posao prihvatio. Drugi model se sastoji od dva izraza, od kojih se jedan koristi za određivanje treba li se posao prihvatiti, a drugi za određivanje prioriteta posla. Suprotno očekivanjima pokazalo se kako je prvi model polučio bolje rezultate. U radu [92] nastavljen je rad na tom problemu te je predloženo korištenje sekvencijskog genetskog programiranja *engl. sequential genetic programming* (SGP). Predloženi postupak razlikuje se po načinu kako se evoluirani izrazi ocjenjuju. Naime u klasičnom GP-u uzima se u obzir

samo konačna vrijednost kriterija dobivena na temelju konačno izgrađenog rasporeda, dok se kod SGP-a ocjenjuje svaka pojedina odluka postupka, odnosno svaki pojedini odabir posla koji će se rasporediti. Pokazalo se kako je takva evaluacija poboljšala postignute rezultate.

Osim za izradu pravila raspoređivanja, GP može se koristiti i za evoluiranje izraza koju se mogu koristiti za jedan malo drugačiji problem raspoređivanja poznat pod nazivom DDAR (engl. *due date assignment rules*). Cilj ovog problema jest za poslove koji dolaze u sustav odrediti njihovo očekivano vrijeme završetka. U [93] GP je iskorišten kako bi u isto vrijeme evoluiralo dva izraza, jedan za određivanje očekivanog vremena završetka i drugi koji se koristi za određivanje prioriteta strojeva. Isprobana su dva GP-a za višekriterijsku optimizaciju te je dodatno predložen jedan novi algoritam za višekriterijsku optimizaciju koji je polučio najbolje rezultate. U radu [94] je nastavljen rad na ranije navedenom postupku te je dana opširnija analiza rezultata dobivenih višekriterijskom optimizacijom.

Pregled primjene GP-a za izradu pravila raspoređivanja dan je u radu [95]. U navedenom radu dan je detaljan pregled radova po raznim kategorijama. Može se zaključiti kako je istraživanje na ovom području veoma aktivno i kako se ono zadnjih nekoliko godina konstantno povećava. U navedenom radu je također istaknuto i nekoliko mogućih smjerova za daljnja istraživanja u ovom području.

V. Zaključak

U ovom radu dan je kratki uvod u teoriju raspoređivanja te pregled postupaka koji se mogu koristiti za izgradnju rasporeda, pri čemu je naglasak posebno stavljen na korištenje GP-a u svrhu izrade pravila raspoređivanja.

Pokazano je kako je GP u stanju evoluirati pravila raspoređivanja koja mogu ostvariti veoma dobre rezultate. To je pokazano i usporedbom s klasičnim pravilima raspoređivanja koja su razvijena od strane ljudskih eksperata. Pravilima koja su bila razvijena GP-om postignuti su rezultati koji su najčešće bili usporedivi ili bolji od rezultata postignutih od strane klasičnih pravila raspoređivanja. Ta činjenica potvrđuje kako je razvoj pravila raspoređivanja genetskim programiranjem više nego isplativo.

Zadnjih nekoliko godina istraživanje je usmjereno na poboljšavanje GP-a za izradu pravila raspoređivanja kako bi se dobila što bolja pravila raspoređivanja. No bez obzira na velik broj radova koji je objavljen u ovom području, postoji još mnogo prostora za poboljšanja. To se posebice može uočiti po tome što je najveći dio istraživanja orijentiran prema okruženju proizvoljne obrade, dok su neka druga okruženja uvelike zanemarena (primjerice nesrodni i jednoliki strojevi). Također postoje još i neki postupci koji se još nisu primijenili na GP u svrhu izrade pravila raspoređivanja, kao što je to primjerice semantičko genetsko programiranje [96], intervalna aritmetika [97] te rollout heuristika [98][99][100]. Osim toga u [95] također je predloženo nekoliko otvorenih područja za mogući budući rad.

Izrada pravila raspoređivanja putem GP-a tek je u začecima i radi se o veoma novom pristupu, no kroz dosadašnji rad pokazano je kako ovaj pristup ima veliki potencijal i kako je samo pitanje vremena kada će ga biti moguće primijeniti u praksi.

References

[1] Michael Pinedo. *Scheduling Therory, Algorithms and Systems*.

[2] V H L Cheng, L S Crawford, and P K Menon. Air traffic control using genetic search techniques. *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)*, 1:249–254, 1999.

[3] James V. Hansen. Genetic search methods in air traffic control. *Computers and Operations Research*, 31(3):445–459, 2004.

[4] Sanja Petrovic and Elkin Castro. A Genetic Algorithm for Radiotherapy Pre-treatment Scheduling. *Applications of Evolutionary Computation*, 6025(August 2015):462–471, 2010.

[5] T. C. Chiang, Y. S. Shen, and L. C. Fu. *A new paradigm for rule-based scheduling in the wafer probe centre*, volume 46. 2008.

[6] Amrusha Varadarajan and Subhash C. Sarin. A survey of dispatching rules for operational control in wafer fabrication. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 12(PART 1), 2006.

[7] Hesam Izakian, Ajith Abraham, and Václav Snášel. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, CSO 2009*, 1:8–12, 2009.

[8] Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen, and Richard F. Freund. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. *Journal of Parallel and Distributed Computing*, 59(2):107–131, 1999.

[9] XiaoShan He, Xianhe Sun, and Gregor Laszewski. QoS guided Min-Min heuristic for grid task scheduling. *Journal of Computer Science and Technology*, 18(4):442–451, 2003.

[10] Lee Wang, Howard Jay Siegel, Vwani P. Roychowdhury, and Anthony a. Maciejewski. Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach. *Journal of Parallel and Distributed Computing*, 47(1):8–22, 1997.

[11] Tarun Goyal and Aakanksha Agrawal. Host Scheduling Algorithm Using Genetic Algorithm in Cloud Computing Environment. 1(1):7–12, 2013.

[12] Domagoj Jakobović. Raspore đ ivanje zasnovano na prilagodljivim pravilima. 2005.

[13] Bya H. Land a. G. Doig A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[14] Jan Karel Lenstra. APPROXIMATION ALGORITHMS FOR SCHEDULING U N R E L A T E D PARALLEL M A C H I N E S Jan Karel LENSTRA. *Mathematical Programming*, 46(1):259–271, 1990.

[15] El-Ghazali Talbi. *METAHEURISTICS FROM DESIGN TO IMPLEMENTATION*. 2009.

[16] Jae Ho Lee, Jae Min Yu, and Dong Ho Lee. A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: Minimizing total tardiness. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2081–2089, 2013.

[17] year=2007 Engelbrecht, Andries P. *Computational Intelligence An Introduction*.

[18] Marko Čupić, Bojana Dalbelo Bašič, and Marin Golub. *Neizrazito, evolucijsko i neuroracunarstvo*.

[19] Chi-Wei Lin, Yang-Kuei Lin, and Han-Ting Hsieh. Ant colony optimization for unrelated parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, 67(1-4):35–45, 2013.

[20] Christos Dimopoulos and Ali M S Zalzala. Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation*, 4(2):93–113, 2000.

[21] Emma Hart, Peter Ross, and David Corne. Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines*, 6(2):191–220, 2005.

[22] Hong Zhou, Yuncheng Feng, and Limin Han. The hybrid heuristic genetic algorithm for job shop scheduling. *Computers and Industrial Engineering*, 40(3):191–200, 2001.

[23] Jie Gao, Mitsuo Gen, Linyan Sun, and Xiaohui Zhao. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, 53(1):149–162, 2007.

[24] Nhu Binh Ho, Joc Cing Tay, and Edmund M K Lai. An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, 179(2):316–333, 2007.

[25] Hong Zhou, Waiman Cheung, and Lawrence C. Leung. Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *European Journal of Operational Research*, 194(3):637–649, 2009.

[26] a. Costa, F. a. Cappadonna, and S. Fichera. A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2799–2817, 2013.

[27] Artur M. Kuczapski, Mihai V. Micea, Laurentiu a. Maniu, and Vladimir I. Cretu. Efficient generation of near optimal initial populations to enhance genetic algorithms for Job-Shop Scheduling. *Information Technology and Control*, 39(1):32–37, 2010.

[28] Ali Allahverdi, Jatinder N D Gupta, and Tariq Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, 1999.

[29] Ali Allahverdi, C. T. Ng, T. C E Cheng, and Mikhail Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

[30] Chunfeng Liu. A hybrid genetic algorithm to minimize total tardiness for unrelated parallel machine scheduling with precedence constraints. 2013:1–12, 2013.

[31] Hisao Ishibuchi, Tadashi Yoshida, and Tadahiko Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.

[32] M. Rabiee, Reza Sadeghi Rad, M. Mazinani, and R. Shafaei. An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines. *International Journal of Advanced Manufacturing Technology*, 71(5-8):1229–1245, 2014.

[33] S. F. Attar, M. Mohammadi, and R. Tavakkoli-Moghaddam. Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times. *International Journal of Advanced Manufacturing Technology*, 68(5-8):1583–1599, 2013.

[34] Ye Xu, Ling Wang, Sheng Yao Wang, and Min Liu. An effective teaching-learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. *Neurocomputing*, pages 121–135, 2013.

[35] Michele Pfund, John W. Fowler, and Jatinder N. D. Gupta. a Survey of Algorithms for Single and Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems. *Journal of the Chinese Institute of Industrial Engineers*, 21(3):230–241, 2004.

[36] Xiaoning Shen, Min Zhang, and Jingzht Fu. multi-objective dynamic job shop scheduling: a survey and prospects.

[37] Luis Fanjul-Peyro and Rubén Ruiz. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1):55–69, 2010.

[38] Luis Fanjul-Peyro and Rubén Ruiz. Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers and Operations Research*, 38(1):301–309, 2011.

[39] Luis Fanjul-Peyro and Rubén Ruiz. Scheduling unrelated parallel machines with optional machines and jobs selection. *Computers and Operations Research*, 39(7):1745–1753, 2012.

[40] Thomas E. Morton and David E. Pentico. *Heuristic scheduling systems*. 1993.

[41] Young Hoon Lee, Kumar Bhaskaran, and Michael Pinedo. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, 29(1):45–52, 1997.

[42] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, Richard F Freund, and Ladislau L Boloni. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.

[43] K. Etminani and M. Naghibzadeh. A Min-Min Max-Min selective algorihtm for grid task scheduling. *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, 2007.

[44] G K Kamalam and V Murali Bhaskaran. An Improved Min-Mean Heuristic Scheduling Algorithm for Mapping Independent Tasks on Heterogenous Computing Environment. *International Journal*, 8(4):85–91, 2010.

[45] G K Kamalam and Murali Bhaskaran V. A New Heuristic Approach : Min-mean Algorithm For Scheduling Meta-Tasks On Heterogeneous Computing Systems. 10(1):24–31, 2010.

[46] Lin Yang-Kuei and Lin Chi-Wei. Dispatching rules for unrelated parallel machine scheduling with release dates. *International Journal of Advanced Manufacturing Technology*, 67(1-4):269–279, 2013.

[47] Juergen Branke and Christoph W. Pickardt. Evolutionary search for difficult problem instances to support the design of job shop dispatching rules. *European Journal of Operational Research*, 212(1):22–32, 2011.

[48] José Antonio Vázquez-Rodríguez and Sanja Petrovic. A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics*, 16(6):771–793, 2010.

[49] Jens Heger, Torsten Hildebrandt, and Bernd Scholz-Reiter. Dispatching rule selection with Gaussian processes. *Central European Journal of Operations Research*, pages 1–15, 2013.

[50] Ahmed El-Bouri, Subramaniam Balakrishnan, and Neil Popplewell. Sequencing jobs on a single machine: A neural network approach. *European Journal of Operational Research*, 126(3):474–490, 2000.

[51] Xiaonan Li and Sigurdur Olafsson. Discovering dispatching rules using data mining. *Journal of Scheduling*, 8(6):515–527, 2005.

[52] Sigurdur Olafsson and Xiaonan Li. Learning effective new single machine dispatching rules from optimal scheduling data. *International Journal of Production Economics*, 128(1):118–126, 2010.

[53] Helga Ingimundardottir and Thomas Philip Runarsson. Supervised learning linear priority dispatch rules for job-shop scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6683 LNCS:263–277, 2011.

[54] D a Koonce and Chang-Chun Tsa. Using data mining to find patterns in genetic algorithm solutions to a job shop schedule. *Computers and Industrial Engineering*, 38(3):361–374, 2000.

[55] a. L. Huyet. Optimization and analysis aid via data-mining for simulated production systems. *European Journal of Operational Research*, 173(3):827–838, 2006.

[56] a. K. Choudhary, J. a. Harding, and M. K. Tiwari. Data mining in manufacturing: A review based on the kind of knowledge. *Journal of Intelligent Manufacturing*, 20(5):501–521, 2009.

[57] S Kitamura. Rule Acquisition for Production Scheduling. *Science And Technology*, pages 2162–2167, 2003.

[58] Sanja Petrovic, Carole Fayad, Dobrila Petrovic, Edmund Burke, and Graham Kendall. Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research*, 159(1):275–292, 2008.

[59] Mohammad Shojafar, Saeed Javanmardi, Saeid Abolfazli, and Nicola Cordeschi. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Cluster Computing*, pages 829–844, 2015.

[60] Su Nguyen and Kay Chen Tan. A Dispatching rule based Genetic Algorithm for Order Acceptance and Scheduling. pages 433–440, 2015.

[61] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. 1992.

[62] William Langdon and Ricardo Poli. *Foundations of Genetic Programming*. 2002.

[63] Ricardo Poli, William Langdon, and Nicholas McPhee. *A Field Guide to Genetic Programming*. 2008.

[64] John R. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):251–284, 2010.

[65] P G Espejo, S Ventura, and F Herrera. A Survey on the Application of Genetic Programming to Classifcation. *Ieee Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, 40(2):121–144, 2010.

[66] Hajira Jabeen and Abdul Rauf Baig. Review of Classification Using Genetic Programming. 2(2):94–103, 2010.

[67] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. A Classification of Hyper-heuristics Approaches. *Handbook of Metaheuristics*, 57:449–468, 2010.

[68] Edmund K Burke, Mathew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. Exploring Hyper-heuristic Methodologies with Genetic Programming. *Computational Intelligence*, 1:177–201, 2009.

[69] Christos Dimopoulos and Ali M S Zalzala. A genetic programming heuristic for the one-machine total tardiness problem. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3(1):2207–2214, 1999.

[70] Christos Dimopoulos and Ali M S Zalzala. Investigating the use of genetic programming for a classic one-machine scheduling problem.

[71] K Miyashita. Job-shop scheduling with genetic programming. *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 505–512, 2000.

[72] Wen Jun Yin, Min Liu, and Cheng Wu. Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming. *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*, 2:1050–1055, 2003.

[73] Christopher D. Geiger, Reha Uzsoy, and Haldun Aytug. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling*, 9(1):7–34, 2006.

[74] D Jakobović and L Budin. Dynamic scheduling with genetic programming. *Genetic Programming*, 2006.

[75] Domagoj Jakobović, Leonardo Jelenković, and Leo Budin. Genetic Programming Heuristics for Multiple Machine Scheduling. *Proceedings of the 10th European Conference on Genetic Programming*, 4445:321–330, 2007.

[76] Domagoj Jakobovi, Igor Grudeni, and Leonardo Jelenkovi. Genetic Programming Heuristics for Scheduling of Dynamic Unrelated Machines.

[77] Joc Cing Tay and Nhu Binh Ho. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering*, 54(3):453–473, 2008.

[78] Torsten Hildebrandt, Jens Heger, and Bernd Scholz-Reiter. Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 257–264, 2010.

[79] Cândida Ferreira. Gene Expression Programming : A New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13(2):1–22, 2001.

[80] Xin Li, Chi Zhou, Weimin Xiao, and Peter C Nelson. Prefix Gene Expression Programming. *Gene Expression*, 2005.

[81] Li Nie, Xinyu Shao, Liang Gao, and Weidong Li. Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *International Journal of Advanced Manufacturing Technology*, 50(5-8):729–747, 2010.

[82] Li Nie, Liang Gao, Peigen Li, and Liping Zhang. Application of gene expression programming on dynamic job shop scheduling problem. *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 291–295, 2011.

[83] Erik Pitzer, Andreas Beham, Michael Affenzeller, Helga Heiss, and Markus Vorderwinkler. Production fine planning using a solution archive of priority rules. *LINDI 2011 - 3rd IEEE International Symposium on Logistics and Industrial Informatics, Proceedings*, pages 111–116, 2011.

[84] Domagoj Jakobović and Kristina Marasović. Evolving priority scheduling heuristics with genetic programming. *Applied Soft Computing Journal*, 12(9):2781–2789, 2012.

[85] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 17(5):621–639, 2013.

[86] Christoph W. Pickardt, Torsten Hildebrandt, Jürgen Branke, Jens Heger, and Bernd Scholz-Reiter. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*, 145(1):67–77, 2013.

[87] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Learning iterative dispatching rules for job shop scheduling with genetic programming. *International Journal of Advanced Manufacturing Technology*, 67(1-4):85–100, 2013.

[88] Pierre Hansen and Nenad Mladenovi. Variable neighborhood search: Principles and applications. *European Journal Of Operational Research*, 130, 2001.

[89] Rachel Hunt, Mark Johnston, Rachel Hunt, and Mark Johnston. Evolving " Less-myopic " Scheduling Rules for Dynamic Job Shop Scheduling with Genetic Programming. pages 927–934.

[90] Rachel Hunt, Mark Johnston, and Mengjie Zhang. Evolving Machine-Specific Dispatching Rules for a Two-Machine Job Shop using Genetic Programming. 2014.

[91] John Park, Su Nguyen, Mengjie Zhang, and Mark Johnston. Genetic programming for order acceptance and scheduling. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, (3):1005–1012, 2013.

[92] Su Nguyen, Mengjie Zhang, and Mark Johnston. A Sequential Genetic Programming Method to Learn Forward Construction Heuristics for Order Acceptance and Scheduling. pages 1824–1831, 2014.

[93] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, (i):10–15, 2012.

[94] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation*, 18(2):193–208, 2014.

[95] Juergen Branke, Su Nguyen, Christoph Pickardt, and Mengjie Zhang. Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation*, X(X):1–1, 2015.

[96] Maarten Keijzer and Vladan Babovic. Dimensionally Aware Genetic Programming. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:1069–1076, 1999.

[97] Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. *Improving Symbolic Regression with Interval Arithmetic and Linear Scaling*, 2003.

[98] Joseph Leung. *Handbook of SCHEDULING Algorithms, Models, and Performance Analysis*. 2004.

[99] Dimitri Bertsekas, John Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*.

[100] Dimitri Bertsekas. Rollout algorithms for discrete optimization: A survey.