Optimizacija problema raspoređivanja primjenom genetičkog algoritma s nišama.

Abrashi, Arijan

Sveučilište u Zagrebu
FAKULTET STROJARSTVA I BRODOGRADNJE

OPTIMIZACIJA PROBLEMA RASPOREĐIVANJA PRIMJENOM GENETIČKOG ALGORITMA S NIŠAMA

DOKTORSKI RAD

Mentor:
prof. dr. sc.
NEDELJKO ŠTEFANIĆ

ARIJAN ABRASHI,
dipl. ing. stroj.

Zagreb, 2011.

PODACI ZA BIBLIOGRAFSKU KARTICU

UDK:	658.512
Ključne riječi:	problem raspoređivanja, genetički algoritam, niše, Hamiltonova sličnost
Znanstveno područje:	TEHNIČKE ZNANOSTI
Znanstveno polje:	Strojarstvo
Institucija u kojoj je rad izrađen:	Fakultet strojarstva i brodogradnje, Sveučilište u Zagrebu
Mentor rada:	Prof. dr. sc. Nedeljko Štefanić
Broj stranica:	219
Broj slika:	147
Broj tablica:	18
Broj korištenih bibliografskih jedinica:	59
Datum obrane:	2.5.2011.
Povjerenstvo:	Prof. dr. sc. Nikola Šakić, predsjednik Prof. dr. sc. Nedeljko Štefanić, mentor Prof. dr. sc. Branko Novaković Prof. dr. sc. Ivo Čala Doc. dr. sc. Marin Golub, Fakultet elektrotehnike i računarstva, Zagreb
Institucija u kojoj je rad pohranjen:	Fakultet strojarstva i brodogradnje

PREDGOVOR

Sve tvrtke koje se bave proizvodnjom moraju ispuniti tri bitne funkcije. To su proizvodnja, prodaja i distribucija. Razumijevanje planiranja, raspoređivanja resursa i kontrole procesa proizvodnje presudno je za uspješnost poslovanja. Na žalost, u vrijeme kada je seljenje proizvodnje u zemlje s jeftinijom radnom snagom postalo globalni trend, mogućnost preciznijeg i pravilnijeg planiranja i raspoređivanja resursa postaje uvjet opstanka na globalnom tržištu. Novi proizvodni pogoni i uvođenje nove radne snage zahtijevaju planiranje nekoliko mjeseci unaprijed. Planovi se ne mogu uvijek ispuniti zbog nepredviđenih događaja izvan našeg utjecaja. Zastoji na strojevima, kašnjenje s isporukom materijala te iznenadna promjena zahtjeva kupca ili korisnika usluge mogu bitno utjecati na izrađene planove i rasporede proizvodnje. Brza reakcija na promjene u proizvodnom procesu te izrada novih planova i rasporeda postaju nužnost.

Potkraj 19. stoljeća problem nije bio toliko jasno izražen. Industrijska je proizvodnja bila tek u povojima. Tvornice su bile relativno malene i jednostavne, jer su strojevi morali biti blizu centralnom energetskom izvoru. U takvim uvjetima proizvodnju je s dosta uspjeha mogao kontrolirati poslovođa, koji je bio zadužen za sve-od zapošljavanja radne snage i nabave materijala do organizacije proizvodnje i distribucije gotovih proizvoda. Situacija se naglo počela mijenjati oko 1890. godine kada je zbog uvođenja električnih motora nestala potreba za lociranjem opreme za proizvodnju blizu centralnoga energetskog izvora. Proizvodnja je postala sve raznolikija i složenija te su tvornice počele proizvoditi sve više različitih proizvoda. U takvim uvjetima poslovođa više nije mogao osobno kontrolirati sve aspekte proizvodnje. Razdvajanje planiranja proizvodnje i njene provedbe bio je logičan korak. Dvije su osobe u tome procesu imale iznimno bitnu ulogu. Frederick Winslow Taylor predložio je osnivanje ureda za planiranje proizvodnje, a Henry Laurence Gantt osmislio je inovativni dijagram za kontrolu proizvodnje. Njegov je dijagram jednostavno omogućivao poslovođi praćenje proizvodnje tako da je u svakom trenutku znao odvija li se ona u roku, odnosno kasnili ili ide ispred plana.

Ganttov dijagram do danas je ostao najvažniji alat za kontrolu procesa proizvodnje ali sam za sebe nije dovoljan za optimalno iskorištenje raspoloživih resursa. Naime, proizvodnja se može organizirati na manje ili više optimalan način. Da bi se odredio optimalan ili gotovo optimalan raspored proizvodnje

razvijeno je mnogo algoritama od kojih su najpoznatiji SPT[1], FCFS[2] i EDD[3] za rasporede s jednim strojem te Johnsonovo pravilo za raspored s dva stroja. Za sve veće probleme ne postoji egzaktan algoritam za kreiranje optimalnog rasporeda. Kako je iz teorije kompleksnosti poznato da problem raspoređivanja pripada grupi NP-teških problema, potreban je posve drukčiji pristup optimiranju rasporeda. Sama pripadnost NP-teškoj grupi problema upozorava na to da nije realno očekivati kako će se pronaći algoritam koji može uspješno riješiti problem raspoređivanja u razumnom vremenu. Razvojem računalne tehnologije otvorila se mogućnost primjene heurističkih algoritama poput simuliranoga kaljenja, tabu-pretraživanja, mravljih kolonija i genetičkih algoritama. U ovom doktorskom radu dan je originalni znanstveni doprinos u razvoju genetičkih algoritama pri rješavanju problema raspoređivanja. Uporabom inovativnog načina kreiranja niša, mogućnost održavanja genetske raznolikosti populacije znatno je uvećana, a to bitno pridonosi kvaliteti rješenja.

[1] SPT – najkraće vrijeme obrade (*engl. Shortest Processing Time*)
[2] FCFS – prvi dolazak, prva obrada (*engl. First Come First Served*)
[3] EDD – najkraće predviđeno vrijeme završetka (*engl. Earliest Due Date*)

Zahvaljujem svom mentoru prof. dr. sc. Nedeljku Štefaniću i prof. dr. sc. Nikoli Šakiću na svesrdnoj pomoći tijekom izrade doktorskog rada te na usmjeravanju tijekom poslijediplomskog studija.

Zahvaljujem također mr. sc. Zdravku Mužeku, direktoru Ekonerga, na velikoj pomoći i razumijevanju.

SADRŽAJ

Predgovor 3
Sadržaj 7
Sažetak / Ključne riječi 10
Abstract / Keywords 11
Popis slika 13
Popis tablica 19
Popis oznaka 21

1. Uvod 27
   1.1. Hipoteza rada 29
   1.2. Opis istraživanja 30
2. Problem raspoređivanja 33
   2.1. Notacija 33
      2.1.1. Okoliš procesa 34
      2.1.2. Svojstva i ograničenja procesa 36
      2.1.3. Funkcija cilja 36
         2.1.3.1. Funkcije cilja koje ovise o vremenu završetka 38
         2.1.3.2. Funkcije cilja koje ovise o predviđenom vremenu završetka 38
   2.2. Redukcija problema raspoređivanja 39
   2.3. Težina problema 42
   2.4. NP-teški problemi raspoređivanja (primjer) 44
   2.5. Pravila raspoređivanja na problemu s jednim strojem 47
      2.5.1. SPT – najkraće vrijeme obrade 48
      2.5.2. FCFS – Prvi dolazak – prva obrada 51
      2.5.3. EDD – najkraće predviđeno vrijeme završetka 52
      2.5.4. CR – kritični odnos 54
   2.6. Definicija problema (JSSP) 56
   2.7. Klase rasporeda 58
      2.7.1. Pravila za stvaranje rasporeda 59
3. Uvod u genetičke algoritme 63
   3.1. Notacija genetičkog algoritma 66
   3.2. Shema-teorem i hipoteza građevnih blokova 66
      3.2.1. Operator selekcije 67
      3.2.2. Operator križanja 69
      3.2.3. Operator mutacije 70
      3.2.4. Hipoteza građevnih blokova 71

3.3. Prikaz rješenja (kromosoma) 71
3.3.1. Binarni prikaz rješenja 72
3.3.2. Prikaz rješenja za permutacijski problem 73
3.3.3. Kodiranje prema referentnoj listi 74
3.4. Metoda selekcije 76
3.4.1. Stohastički odabir sa zamjenom (SSR) 78
3.4.1.1. Model Goldberga i Segresta 79
3.4.2. Stohastički odabir bez zamjene (SSWR) 83
3.4.3. Usporedba SSR i SSWR selekcije 83
3.4.4. Deterministički odabir (DS) 84
3.4.5. Stohastička turnirska selekcija (STS) 86
3.4.6. Stohastički odabir ostatka 88
3.4.7. Stohastički univerzalni odabir (SUS) 90
3.5. Genetički operatori 92
3.5.1. Operatori inverzije 93
3.5.2. Operatori križanja 94
3.5.2.1. Operator križanja s jednom točkom prekida 95
3.5.2.2. Operator križanja s dvije točke prekida 96
3.5.2.3. Uniformni operator križanja 97
3.5.2.4. PMX operator križanja 98
3.5.2.5. OX operator križanja 99
3.5.2.6. UCX operator križanja 100
3.5.2.7. CX operator križanja 101
3.5.2.8. LOX operator križanja 103
3.5.2.9. SXX operator križanja 104
3.5.3. Operatori mutacije 105
3.5.4. Uvjeti prekida genetičkoga algoritma 108
3.5.4.1. Broj generacija 109
3.5.4.2. Gubitak genetske raznolikosti 109
3.5.4.3. Nemogućnost pronalaska boljeg rješenja 109
4. Genetički algoritmi i problem raspoređivanja 111
4.1. Kodiranje zasnovano na disjunkcijskom grafu 112
4.2. Kodiranje zasnovano na povlaštenim listama 115
4.3. Kodiranje zasnovano na operacijama 117
4.4. Kodiranje prema referentnoj listi 119
5. Genetički algoritmi s nišama 123
5.1. Dosadašnji radovi 123
5.1.1. Algoritam s predodabirom (Preselection algoritam) 124
5.1.2. Algoritam grupiranja (Crowding algoritam) 124
5.1.3. Deterministički algoritam grupiranja (Deterministic Crowding algoritam) 125

5.1.4. Ograničeno natjecanje ... 126
5.1.5. Algoritam s raspodjelom funkcije cilja (Fitness Sharing algoritam) ... 127
5.2. Uloga niše u predloženom algoritmu ... 129
5.3. Sličnost između jedinki u populaciji ... 129
5.4. Hamiltonova sličnost (novi pristup) ... 132
5.5. Algoritam s raspodjelom funkcije cilja i Hamiltonova sličnost ... 135
5.6. Uloga skaliranja vrijednosti funkcije cilja ... 138
6. Diploidni kromosom ... 143
6.1. Usporedba haploidnog i diploidnog kromosoma ... 145
6.2. Operator križanja i diploidni kromosom (mejoza) ... 152
6.2.1. Prirodni proces ... 153
6.2.2. Operator križanja u predloženom algoritmu ... 154
6.3. Dominacija ... 155
6.3.1. Hollstienova shema dominacije u dvije pozicije ... 156
6.3.2. Hollstienova shema dominacije s tri vrijednosti na jednoj poziciji ... 156
6.3.3. Ryanova aditivna shema dominacije ... 157
6.4. Predloženi model ... 160
7. Eksperiment ... 161
7.1. Definicija benchmark problema ... 161
7.2. Svojstva predloženoga genetičkog algoritma ... 165
7.3. Parametri algoritma ... 169
7.4. Ispitivanje algoritma na mt benchmark problemima ... 170
7.5. Ispitivanje algoritma na la benchmark problemima ... 174
7.5.1. Utjecaj evolucijskih parametara ... 177
7.5.2. La 10x5 problemi ... 178
7.5.3. La 15x5 problemi ... 181
7.5.4. La 20x5 problemi ... 184
7.5.5. La 10x10 problemi ... 187
7.5.6. La 15x10 problemi ... 191
7.5.7. La 20x10 problemi ... 195
7.5.8. La 30x10 problemi ... 199
7.5.9. La 15x15 problemi ... 204
8. Zaključak ... 211
8.1. Znanstveni doprinos ... 213
8.2. Smjernice daljnjeg rada ... 213
9. Bibliografija ... 215
Životopis ... 219
Biography ... 219

SAŽETAK / KLJUČNE RIJEČI

U doktorskom su radu dana osnovna svojstva genetičkoga algoritma te njegove prednosti i nedostaci. Posebno su razmatrane metode selekcije, operatori križanja i mutacije te direktan i indirektan način prezentacije kromosoma. Također je analizirana razlika između haploidnoga i diploidnoga kromosoma te je dana analiza utjecaja diploidnoga kromosoma na očuvanje genetske raznolikosti populacije.

Predložen je i ispitan genetički algoritam s nišama (GA) koji se za usporedbu jedinki u populaciji koristi takozvanom Hamiltonovom sličnošću. Prednost Hamiltonove sličnosti je u tome da ne postoji potreba za poznavanjem problema koji se rješava da bi se uspješno usporedila dva člana populacije (u konkretnom slučaju dva rasporeda). Algoritam je ispitan na dvije grupe poznatih *benchmark* problema, *mt*[4] i *la*[5]. Dobiveni rezultati pokazuju manju standardnu devijaciju predloženog algoritma u odnosu prema jednostavnom genetičkom algoritmu (SGA), što jasno upozorava na njegovu superiornost.

Osim Hamiltonove sličnosti, predloženo je i vremenski zavisno skaliranje funkcije cilja koje u suradnji s nišama znatno umanjuje mogućnost zapinjanja genetičkog algoritma u jednom od manje poželjnih lokalnih optimuma. Na kraju su date smjernice za daljnja istraživanja na području genetičkih algoritama s nišama.

Ključne riječi: genetički algoritam, niše, Hamiltonova sličnost, vremenski zavisno skaliranje funkcije cilja, raspoređivanje operacija po strojevima

[4] Muth i Thompson
[5] Lawrence

ABSTRACT / KEYWORDS

In this PhD thesis basic characteristics of genetic algorithm have been presented along with its advantages and disadvantages. Selection methods, crossover and mutation operators, and direct and indirect chromosome presentation had especially been considered. The difference between haploid and diploid chromosomes was also analyzed and the influence of a diploid chromosome on the genetic diversity of a population presented.

A niching genetic algorithm (GA), which uses so-called Hamilton similarity for comparison of individuals in a population, was proposed and tested. The advantage of the Hamilton similarity lies in the fact that there is no need for context sensitive information in order to successfully compare two population members. Furthermore, the algorithm was tested on two famous benchmark problems (JSSP) – *mt* and *la*. The statistical results of the test were given. The significantly smaller standard deviation of the proposed GA compared to Simple GA clearly demonstrates its superiority.

In addition to the Hamilton similarity, time dependent fitness scaling was also proposed, which in conjunction with niching significantly reduces the probability of the genetic algorithm getting trapped in one of the less desirable local optima. Finally, suggestions for future research are given.

Keywords: genetic algorithm, niching, Hamilton similarity, time dependent fitness scaling, Job shop scheduling problem (JSSP)

POPIS SLIKA

Slika 2.1.1. Funkcije cilja 37
Slika 2.2.1. Hijerarhija funkcija cilja 40
Slika 2.3.1. Vannov dijagram za P, NP, NP-potpunu i NP-tešku klasu problema (P≠NP) 43
Slika 2.4.1. Ganttov dijagram za NP-potpun problem raspoređivanja 45
Slika 2.4.2. Primjer klike 45
Slika 2.4.3. Raspored zasnovan na problemu klike 47
Slika 2.5.1. Redosljed poslova 49
Slika 2.5.2. SPT 49
Slika 2.5.3. FCFS 51
Slika 2.5.4. EDD 53
Slika 2.5.5. CR 56
Slika 2.6.1. Ganttov dijagram i dopušteni lijevi pomak a)poluaktivni raspored b)aktivni raspored 58
Slika 2.7.1. Klase rasporeda 59
Slika 3.4.1. Broj potencijalnih potomaka (SSR) 77
Slika 3.4.2. Stohastički odabir sa zamjenom (r=1) 81
Slika 3.4.3. Stohastički odabir sa zamjenom (r=3) 82
Slika 3.4.4. Stohastički odabir sa zamjenom (r=10) 82
Slika 3.4.5. Usporedba SSR i SSWR 83
Slika 3.4.6. Deterministički odabir 85
Slika 3.4.7. Deterministički odabir (r=3) 85
Slika 3.4.8. Turnirska selekcija (s=2) 87
Slika 3.4.9. Turnirska selekcija (s=3) 88
Slika 3.4.10. Stohastički odabir ostatka bez zamjene (r=1) 89
Slika 3.4.11. Stohastički odabir ostatka bez zamjene (r=6) 89
Slika 3.4.12. Stohastički univerzalni odabir (SUS) 90
Slika 3.4.13. Stohastički univerzalni odabir za r=3 91
Slika 3.4.14. Stohastički univerzalni odabir za r=10 92
Slika 3.5.1. Kromosom prije primjene operatora inverzije 93
Slika 3.5.2. Kromosom nakon primjene operatora inverzije 93
Slika 3.5.3. Direktan prikaz 95
Slika 3.5.4. Indirektan prikaz 95

Slika 3.5.5. Roditelji s prikazanom točkom prekida između 2. i 3. gena 95
Slika 3.5.6. Potomci nastali primjenom operatora križanja s jednom točkom prekida 96
Slika 3.5.7. Roditelji s prikazanim točkama prekida..... 96
Slika 3.5.8. Potomci nastali primjenom operatora križanja s dvije točke prekida... 96
Slika 3.5.9. Roditelji s pripadajućim predloškom 97
Slika 3.5.10. Potomci nastali primjenom uniformnog operatora križanja..... 97
Slika 3.5.11. Roditelji s prikazanom točkom prekida..... 98
Slika 3.5.12. Potomci s uklonjenim konfliktnim genima..... 98
Slika 3.5.13. Potomci nastali primjenom PMX operatora križanja 99
Slika 3.5.14. Roditelji s prikazanom točkom prekida..... 99
Slika 3.5.15. Potomci s uklonjenim genima nakon točke prekida 99
Slika 3.5.16. Potomci nastali primjenom OX operatora križanja..... 100
Slika 3.5.17. Potomci s parcijalnim unosom gena..... 100
Slika 3.5.18. Potomci nastali primjenom UCX operatora križanja..... 101
Slika 3.5.19. Prvi korak stvaranja potomaka primjenom CX operatora križanja.... 102
Slika 3.5.20. Drugi korak stvaranja potomaka primjenom CX operatora križanja .. 102
Slika 3.5.21. Završetak prve faze kreiranja potomaka primjenom CX operatora križanja..... 102
Slika 3.5.22. Početak druge faze kreiranja potomaka primjenom CX operatora križanja..... 103
Slika 3.5.23. Potomci nastali primjenom CX operatora križanja 103
Slika 3.5.24. Roditelji s prikazanim točkama prekida 103
Slika 3.5.25. Potomci s uklonjenim genima iz intervala drugog roditelja..... 104
Slika 3.5.26. Pomicanje poznatih gena izvan intervala 104
Slika 3.5.27. Potomci nakon primjene LOX operatora križanja 104
Slika 3.5.28. Roditelji s označenom sekvencom identičnih gena..... 105
Slika 3.5.29. Potomci nakon primjene SXX operatora križanja 105
Slika 3.5.30. Vjerojatnost poboljšanja primjenom operatora mutacije..... 108
Slika 4.1.1. Disjunkcijski graf (neusmjereni)..... 113
Slika 4.1.2. Disjunkcijski graf (usmjereni) 113
Slika 4.1.3. Prikaz kromosoma zasnovanog na disjunkcijskom grafu 114
Slika 4.1.4. Ganttov dijagram zasnovan na disjunkcijskom grafu (polu aktivan raspored) 114
Slika 4.2.1. Kromosom zasnovan na povlaštenim listama 115
Slika 4.2.2. Unos prve operacije s povlaštene liste 116
Slika 4.2.3. Povlaštena lista nakon unosa prve operacije 116

Slika 4.2.4. Raspored nakon unosa novih triju operacija s povlaštene liste. 116
Slika 4.2.5. Povlaštena lista 117
Slika 4.2.6. Konačni raspored operacija prema povlaštenoj listi 117
Slika 4.3.1. Kromosom zasnovan na operacijama 118
Slika 4.3.2. Poluaktivni raspored (kodiranje zasnovano na operacijama) 118
Slika 4.3.3. Aktivni raspored (kodiranje zasnovano na operacijama) 118
Slika 4.4.1. Zapis kromosoma zasnovan na operacijama 119
Slika 4.4.2. Unos prvog člana u kromosom 120
Slika 4.4.3. Unos drugog člana u kromosom 120
Slika 4.4.4. Kromosom zasnovan na referentnoj listi 120
Slika 4.4.5. Roditelji (kodiranje zasnovano na operacijama). 121
Slika 4.4.6. Roditelji (kodiranje zasnovano na referentnoj listi) 121
Slika 4.4.7. Potomci (kodiranje zasnovano na referentnoj listi). 122
Slika 4.4.8. Potomci (kodiranje zasnovano na operacijama) 122
Slika 5.1.1. Primjer funkcije cilja 126
Slika 5.1.2. Potomci i roditelji 127
Slika 5.1.3. Funkcija dijeljenja ovisno o konstanti. 128
Slika 5.3.1. Sinusoidna funkcija 130
Slika 5.5.1. Postotak zajedničkih gena 136
Slika 5.5.2. Funkcija dijeljenja. 137
Slika 5.6.1. Funkcija cilja (primjer) 138
Slika 5.6.2. Korigirana funkcija cilja (translacija) 139
Slika 5.6.3. Korigirana funkcija cilja (potenciranje) 140
Slika 5.6.4. Modifikacija funkcije cilja 142
Slika 5.6.1. Brezin moljac (dvije varijante). 144
Slika 6.1.1. Brzina gubitka nula (slabiji gen) za koeficijent mutacije 0,0033 – haploidni kromosom. 148
Slika 6.1.2. Brzina gubitka nula (slabiji gen) za koeficijent mutacije 0,0033 – diploidni kromosom 151
Slika 6.1.3. Razlika u brzini gubitka nula (slabiji gen) za koeficijent mutacije 0,0033 152
Slika 6.2.1. Proces mejoze 154
Slika 6.3.1. Primjer prelaska s genotipa na fenotip. 155
Slika 6.3.2. Shema dominacije u dvije pozicije 156
Slika 6.3.3. Shema dominacije s tri vrijednosti na jednoj poziciji 157
Slika 6.3.4. Zijevalica (Antirrhinum majus) u tri varijante. 158
Slika 6.3.5. Aditivna shema dominacije. 159

Slika 6.4.1. Shema dominacije za predloženi algoritam 160
Slika 7.2.1. Oscilacija vrijednosti najbolje i najlošije jedinke u generaciji 165
Slika 7.2.2. Promjena varijance pojedinih gena kroz generacije 167
Slika 7.2.3. Frekvencija pojavljivanja pojedinih rješenja kroz generacije..... 168
Slika 7.4.1. mt6 (55 vremenskih jedinica) 170
Slika 7.4.2. mt10 (937 vremenskih jedinica)..... 171
Slika 7.4.3. Frekvencija pojavljivanja pojedinih rješenja u 600 evolucija (SGA, Hamilton)..... 172
Slika 7.4.4. Frekvencija pojavljivanja pojedinih rješenja u 600 evolucija (GT, Hamilton) 173
Slika 7.4.5. mt20 (1175 vremenskih jedinica) 174
Slika 7.5.1. la01 (666 vremenskih jedinica) 179
Slika 7.5.2. la02 (657 vremenskih jedinica) 179
Slika 7.5.3. la03 (593 vremenske jedinice) 180
Slika 7.5.4. la04 (609 vremenskih jedinica) 180
Slika 7.5.5. la05 (593 vremenske jedinice) 181
Slika 7.5.6. la06 (926 vremenskih jedinica) 182
Slika 7.5.7. la07 (890 vremenskih jedinica) 182
Slika 7.5.8. la08 (863 vremenske jedinice) 183
Slika 7.5.9. la09 (951 vremenska jedinica) 183
Slika 7.5.10. la10 (958 vremenskih jedinica) 184
Slika 7.5.11. la11 (1222 vremenske jedinice)..... 185
Slika 7.5.12. la12 (1039 vremenskih jedinica)..... 185
Slika 7.5.13. la13 (1150 vremenskih jedinica)..... 186
Slika 7.5.14. la14 (1292 vremenske jedinice)..... 186
Slika 7.5.15. la15 (1207 vremenskih jedinica)..... 187
Slika 7.5.16. la16 (956 vremenskih jedinica) 188
Slika 7.5.17. la17 (785 vremenskih jedinica) 188
Slika 7.5.18. la18 (855 vremenskih jedinica) 189
Slika 7.5.19. la19 (863 vremenske jedinice) 190
Slika 7.5.20. la20 (913 vremenskih jedinica) 191
Slika 7.5.21. la21 (1068 vremenskih jedinica)..... 192
Slika 7.5.22. la22 (956 vremenskih jedinica) 193
Slika 7.5.23. la23 (1032 vremenske jedinice)..... 193
Slika 7.5.24. la24 (977 vremenskih jedinica) 194
Slika 7.5.25. la25 (1008 vremenskih jedinica)..... 195
Slika 7.5.26. la26 (1237 vremenskih jedinica)..... 196

Slika 7.5.27. la27 (1287 vremenskih jedinica) 197
Slika 7.5.28. la28 (1252 vremenske jedinice) 197
Slika 7.5.29. la29 (1212 vremenskih jedinica) 198
Slika 7.5.30. la30 (1367 vremenskih jedinica) 199
Slika 7.5.31. la31 (1784 vremenske jedinice) 200
Slika 7.5.32. la32 (1850 vremenskih jedinica) 201
Slika 7.5.33. la33 (1719 vremenskih jedinica) 202
Slika 7.5.34. la34 (1721 vremenska jedinica) 203
Slika 7.5.35. la35 (1888 vremenskih jedinica) 204
Slika 7.5.36. la36 (1296 vremenskih jedinica) 205
Slika 7.5.37. la37 (1449 vremanskih jedinica) 206
Slika 7.5.38. la38 (1252 vremenske jedinice) 207
Slika 7.5.39. la39 (1268 vremenskih jedinica) 208
Slika 7.5.40. la40 (1270 vremenskih jedinica) 209

POPIS TABLICA

Tablica 2.5.1. Tehnološki raspored operacija na jednom stroju. 48
Tablica 2.5.2. Rezultati primjene pravila SPT 50
Tablica 2.5.3. Rezultati primjene pravila FCFS 52
Tablica 2.5.4. Rezultati primjene pravila EDD 54
Tablica 2.5.5. Prvi korak primjene pravila CR. 55
Tablica 2.5.6. Drugi korak primjene pravila CR 55
Tablica 2.5.7. Treći korak primjene pravila CR. 55
Tablica 2.5.8. Posljednji korak primjene pravila CR 56
Tablica 2.6.1. Tehnološki raspored operacija s pripadajućim vremenima trajanja . . 57
Tablica 6.3.1. Veza fenotipa i genotipa. 159
Tablica 7.1.1. Tehnološki raspored za mt6 benchmark problem. 161
Tablica 7.1.2. Tehnološki raspored za mt10 benchmark problem 162
Tablica 7.1.3. Trajanje operacija za mt10 benchmark problem 162
Tablica 7.1.4. Tehnološki raspored za mt20 benchmark problem 163
Tablica 7.1.5. la01 do la40 problemi (najbolja rješenja) 164
Tablica 7.4.1. Statistička usporedba algoritama 172
Tablica 7.5.1. Evolucijski parametri 175
Tablica 7.5.2. Rezultati za pojedine la probleme 176

POPIS OZNAKA

Oznaka	Mjerna jedinica	Opis
1	–	jedan stroj
a_n	–	element seta od n brojeva
A	–	veličina seta kod problema DIJELJENJA
b	–	broj veza
B	–	binarni broj
C	–	konstanta
c	–	veličina klike
C_j	–	vrijeme završetka posljednje operacije
C_{max}	–	vrijeme izlaska posljednjeg posla iz sustava
C_{sr}	–	prosječno vrijeme završetka
C_{sum}	–	suma izlaznih vremena svih poslova
$C_{te.sum}$	–	ponderirana suma izlaznih vremena svih poslova
d	–	udaljenost (sličnost) dviju jedinki
d_j	–	predviđeno vrijeme završetka posla (j)
E	–	rubovi
E_j	–	preuranjenost
E_{max}	–	maksimalna preuranjenost
E_{sr}	–	prosječna preuranjenost
E_{sum}	–	ukupna preuranjenost svih poslova
$E_{te.sum}$	–	srednja ponderirana preuranjenost svih poslova
f'	–	korigirana funkcija cilja
f_0	–	vrijednost nula u ukupnoj vrijednosti kromosoma
f_1	–	vrijednost jedinica u ukupnoj vrijednosti kromosoma
f_a	–	broj jedinica (funkcija cilja)
f_A	–	vrijednost funkcije cilja jedinke tipa A
f_B	–	vrijednost funkcije cilja jedinke tipa B
F_j	–	vrijeme boravka u radionici
F^i	–	funkcija za pretvaranje binarnih brojeva u decimalne
F_m	–	linijska obrada

Oznaka	Mjerna jedinica	Opis
f_{min}	–	minimalna vrijednost funkcije cilja u generaciji
f_{max}	–	maksimalna vrijednost funkcije cilja u generaciji
FF_c	–	prilagodljiva linijska obrada
FJ_c	–	prilagodljiva opća obrada
G	–	graf
G_{gr}	–	granična (tranzicijska) generacija
G_{max}	–	broj generacija (trajanje algoritma)
g	–	razina srodnosti dviju jedinki
H	–	shema
I	–	individualni član populacije
i	–	broj jedinki tipa A
j	–	broj jedinki tipa B
J_m	–	opća obrada
k	–	broj čvorova
l	–	broj veza u kliki
l_x	–	preciznost genetičkog algoritma
L_j	–	vrijeme kašnjenja
L_{max}	–	maksimalno kašnjenje
L_{sr}	–	prosječno vrijeme kašnjenja
L_{sum}	–	ukupno vrijeme kašnjenja svih poslova
$L_{te.sum}$	–	ukupno ponderirano kašnjenje svih poslova
n	–	broj poslova
N	–	broj shema
N_p	–	broj jedinki u populaciji
n_A	–	broj jedinki tipa A
n_B	–	broj jedinki tipa B
n_a	–	broj zajedničkih predaka
M	–	ukupan broj gena (nula i jedinica) u populaciji
m	–	broj strojeva
m	–	operator mutacije
m_0	–	broj nula u populaciji
m_1	–	broj jedinica u populaciji
o	–	red sheme

Oznaka	Mjerna jedinica	Opis
O_{ij}	–	operacija posla (j) na stroju (i)
O_m	–	otvorena obrada
P	–	udio nula u populaciji
p	–	definicija rješenja problema
$\tilde{p}$	–	vrijednost na određenoj poziciji u kromosomu
$\bar{p}$	–	kromosom kodiran prema referentnoj listi
p^+	–	vjerojatnost poboljšanja nakon primjene operatora mutacije
p_A	–	vjerojatnost odabira jedinke A
p_B	–	vjerojatnost odabira jedinke B
p_c	–	vjerojatnost križanja
p_m	–	vjerojatnost mutacije
p_d	–	vjerojatnost presijecanja sheme
P_m	–	paralelni identični strojevi
p_{ij}	–	vrijeme trajanja operacije posla (j) na stroju (i)
p_{jk}	–	matrica trajanja pojedinih operacija
p_m	–	vjerojatnost mutacija
p_p	–	vjerojatnost opstanke sheme nakon primjene operatora križanja
p_s	–	vjerojatnost pojavljivanja pojedinih jedinka u grupi roditelja
p_{trans}	–	vjerojatnost transformacije
Q	–	udio jedinica u populaciji
Q_m	–	jednoliki strojevi
R_m	–	nesrodni strojevi
r_Φ	–	odnos funkcija cilja
r	–	operator križanja
r_j	–	vrijeme početka posla (j)
s	–	operator selekcije
S	–	set brojeva problema DIJELJENJA
S_1	–	prvo rješenje problema DIJELJENJA
S_2	–	drugo rješenja problema DIJELJENJA
sh	–	funkcija dijeljenja
s_j	–	trenutak raspoloživosti

Oznaka	Mjerna jedinica	Opis
t	–	generacija (vrijeme)
t_s	–	veličina turnira
T	–	kriterij prekida algoritma
T_j	–	vrijeme zaostajanja
T_{jk}	–	matrica tehnološkog procesa
T_{max}	–	maksimalno zaostajanje
T_{sr}	–	prosječno vrijeme zaostajanja
T_{sum}	–	ukupno zaostajanje svih poslova
$T_{te.sum}$	–	ukupno ponderirano zaostajanje svih poslova
U_j	–	jedinična kazna
u	–	prvi posao u paru
V	–	vektori
v	–	drugi posao u paru
v_j	–	brzina rada stroja
w_j	–	prioritet posla (j)
x	–	nezavisna varijabla
z	–	parametar dodatnog kašnjenja
Z	–	očekivani broj pojavljivanja
α	–	okoliš unutar kojeg se odvija proces za koji se pravi raspored
φ	–	koeficijent eksponencijalnog skaliranja
β	–	ograničenje procesa
γ	–	funkcija cilja
δ	–	koeficijent linearnog skaliranja
η	–	broj jedinki u populaciji
λ	–	broj potomaka
μ	–	broj roditelja
ω	–	koeficijent kapaciteta niše
σ_{share}	–	prag tolerancije sličnosti
Δ	–	definirana dužina sheme
Φ	–	funkcija cilja
Ω	–	genetički operatori
Ψ	–	tranzicijska funkcija
CR		kritični odnos

Oznaka	Mjerna jedinica	Opis
EDD		pravilo prema kojem posao s najskorijim predviđenim vremenom završetka biti će prvi procesuiran
FC		funkcija cilja
FCFS		pravilo prema kojem će posao koji prvi stigne u radionicu biti prvi procesuiran
FISFS		pravilo prema kojem se odabire posao koji se najduže nalazi u radionici
GA		genetički algoritam
LOR		pravilo prema kojem se odabire posao koji ima najmanje neobavljenih operacija
LR		referentna lista
LSK		pravilo prema kojem se odabire posao koji ima najmanju mogućnost klizanja
LWK		pravilo prema kojem se odabire posao na kojem je preostalo najmanje rada
MOR		pravilo prema kojem se odabire posao koji ima najviše neobavljenih operacija
NP		nedeterminističko polinomno vrijeme
P		polinomno vrijeme
RND		pravilo prema kojem se posao odabire nasumično
SPT		pravilo prema kojem posao koji najkraće traje obavlja se prvi
prmp		prisvajanje
rcrc		recirkulacija

1. UVOD

Genetički algoritmi, kao dio šireg područja umjetne inteligencije, sve se češće koriste u rješavanju problema iz stvarnog života. Još prije petnaestak godina genetički algoritmi bili su poznati tek nekolicini znanstvenika, profesora i njihovih studenata te je njihovo izučavanje bilo ograničeno na akademsku zajednicu. Danas je situacija znatno drukčija. Za genetičke algoritme zainteresirane su različite grane znanosti: od ekonomije, političkih znanosti, psihologije i lingvistike pa sve do imunologije, biologije i računarstva. Sve šire zanimanje za genetičke algoritme posljedica je činjenice da genetički algoritmi zaista uspješno rješavaju probleme koji spadaju u kategoriju tzv. teško rješivih (NP-teških).

Genetičke algoritme razvio je John Holland [1] [2] [3] sa suradnicima na sveučilištu Michigan 1975. godine. Baziraju se na principima prirodne selekcije i genetike i spadaju u grupu tehnika pretraživanja prostora koje su usmjerene jer ne pretražuju nasumice cijeli prostor rješenja (*engl. guided random search technique*). Mogućnost pronalaženja globalnog ekstrema u prostoru pretraživanja u kojem postoji veći broj lokalnih ekstrema jedna je od najvećih prednosti genetičkih algoritama u odnosu prema klasičnim determinističkim metodama. Nadalje, problem koji se rješava, odnosno funkcija cilja ne mora biti derivabilna, što je preduvjet za neke determinističke metode. Genetički algoritmi stohastička su metoda pronalaženja globalnog ekstrema te samim time imaju i neke nedostatke. Naime, iako mogu uspješno izbjeći lokalne ekstreme, ne može se pouzdano ustvrditi je li rješenje dobiveno genetičkim algoritmom zaista globalni ekstrem ili je samo riječ o rješenju koje se nalazi u njegovoj blizini. Doduše, za velik broj problema iz stvarnog života gdje globalni ekstrem nije poznat, to i nije pretjerano bitno jer je dobiveno rješenje »dovoljno« dobro. Nadalje, metoda je iznimno »stabilna«, što joj omogućuje primjenu na brojnim različitim problemima uz minimalne modifikacije.

Genetički su algoritmi zapravo iznimno jednostavni. Naime, svako potencijalno rješenje zadanog problema zapisuje se u obliku kromosoma. Svaki je kromosom sastavljen od gena koji korespondiraju sa svojstvima zadanog problema. Što je problem složeniji, i broj je gena veći, a samim time i pripadajući kromosom duži. Ukupan skup svih kromosoma koji predstavljaju potencijalna rješenja zadanog problema naziva se populacija. Početna se populacija kreira nasumce, tako da se s velikom vjerojatnošću može tvrditi da

su sva rješenja u početnoj (nultoj) generaciji relativno loša, ili da se poslužimo terminologijom iz biologije, ni jedna jedinka iz početne populacije nije pretjerano dobro prilagođena okolišu. Bez obzira na to što ni jedno rješenje (osim pukom slučajnošću) nije ni blizu optimalnom, nisu sva rješenja jednaka. Uporabom funkcije cilja može se odrediti koja su rješenja bolja (bolje prilagođena), a koja lošija. Priroda se pobrinula da bolji pojedinci zbog svoje prilagođenosti prežive i prenesu svoj genetski materijal na sljedeću generaciju koja bi trebala biti bolja (prilagođenija okolišu) od prijašnje. Na taj su se način pojedine vrste prilagodile okolišu i preživjele. Istom analogijom koristi se i genetički algoritam kad traži optimalno rješenje (najprilagođeniju jedinku). Naime, na osnovi funkcije cilja određuje se koji je član populacije bliži optimalnom rješenju. Boljim je članovima dana veća mogućnost da postanu roditelji i prenesu svoje gene na sljedeću generaciju. Daleko od toga da je lošijim članovima populacije onemogućeno da postanu roditelji; zapravo im je samo vjerojatnost znatno umanjena. Kako je riječ o stohastičkom procesu, i lošiji članovi postaju roditelji, ali u znatno manjem broju slučajeva u odnosu prema boljim članovima. Odabrani roditelji stvaraju članove nove generacije miješanjem vlastitih gena. Tako se iz generacije u generaciju stvaraju sve bolja i bolja rješenja sve dok se ne zadovolje unaprijed zadani uvjeti za prekid evolucije.

Danas je poznat velik broj varijanti genetičkih algoritama koji su se razvili iz Hollandova osnovnog modela, a radi uklanjanja nekog nedostatka poput malene genetske raznolikosti populacije ili nemogućnosti istodobnog održavanja većeg broja rješenja. Istodobno s razvitkom teorijskih osnova, efikasnost genetičkih algoritama ispitana je u praksi. U ovom radu usredotočit ćemo se na permutacijsku klasu problema, a posebno na problem raspoređivanja operacija po strojevima.

Nadalje, u ovom radu ispitat će se primjenjivost metode genetičkih algoritama s nišama u sprječavanju preuranjene konvergencije, odnosno sprječavanju kretanja rješenja prema lokalnom ekstremu. Tako će se omogućiti pronalaženje većeg broja različitih lokalnih ekstrema, a samim time i vjerodostojnije pronalaženje globalnog ekstrema.

Prvi pokušaji rješavanja permutacijskih problema pojavili su se prije više od 20 godina, a među njima su istaknuta četiri:

- problem trgovačkog putnika (Goldberg & Lingle, 1985) [4]
- raspoređivanje (Davis, 1985) [5]
- projektiranje integriranih krugova (Louis & Rawling, 1991) [6]
- planiranje ruta (Blanton Jr. & Weinwright, 1993) [7].

U odnosu prema navedenim primjerima, današnje znanje na području genetičkih algoritama omogućuje velik napredak u rješavanju permutacijskih problema [8]. K tome, bolje poznavanje mehanizma selekcije te razvoj novih varijanti genetičkog operatora križanja za potrebe permutacijskih problema dovelo je genetičke algoritme na jedno od važnijih mjesta među optimizacijskim tehnikama. Genetički algoritmi zbog svoje jednostavnosti i pouzdanosti imaju velik potencijal za unapređivanje te zaslužuju veću pozornost i ulaganja i akademske zajednice i privrede.

1.1. HIPOTEZA RADA

Dosadašnja iskustva vezana uz genetičke algoritme upozoravaju na niz nedostataka. Jedan je od važnijih nedostataka moguća prerana konvergencija prema lokalnom ekstremu. Problem se pokušao riješiti na niz načina, od različitih načina selekcije do povećanja mogućnosti zadržavanja genetske raznolikosti populacije.

U ovom radu ispitano je daju li genetički algoritmi koji populaciju dijele na više segmenata (niša) bolje rezultate prilikom rješavanja problema raspoređivanja, a preliminarna ispitivanja upozoravaju na realnost te hipoteze. Prije svega tu se misli na stvaranje nekoliko skupina rješenja od kojih se svaka kreće prema jednom od lokalnih ekstrema. Tako će se povećati vjerojatnost da je među njima i globalni ekstrem.

Ako bismo ponovili provedbu jednostavnoga genetičkog algoritma (*engl. simple genetic algorithm*) dovoljno veliki broj puta, dobili bismo skupinu rješenja koja se ponaša prema Gaussovoj razdiobi. Relativno veliko rasipanje rješenja pokazalo bi da je potrebno algoritam ponoviti relativno velik broj puta (nekoliko stotina) da bi se dobilo »optimalno« rješenje, ili rješenje blizu optimalnog. U ovom radu ispitat ćemo imaju li rješenja genetičkog algoritma s nišama manje rasipanje te je li potrebno provesti manji broj ponavljanja algoritma za jednako kvalitetno rješenje. U uvjetima brzih poslovnih promjena, kada nam na raspolaganju stoji sve manje vremena za optimalnu organizaciju proizvodnje, presudno je imati metodu (algoritam) koja može vjerodostojno pronaći zadovoljavajuće rješenje s malim brojem pokušaja (evolucija).

1.2. OPIS ISTRAŽIVANJA

Hipoteza rada, odnosno istraživanje o tome imaju li genetički algoritmi s nišama manje rasipanje rješenja nego jednostavni genetički algoritam, provest će se istraživanjem podijeljenim u sedam koraka:

1. Kreiranje jednostavnoga genetičkog algoritma
   - Radi usporedbe dvaju algoritama prvo će se kreirati jednostavni genetički algoritam (SGA) te odrediti optimalni parametri algoritma (veličina populacije, broj evolucija, vrsta operatora križanja i drugo)

2. Provedba jednostavnoga genetičkog algoritma na standardnom *benchmark* problemu[6]
   - Genetički će se algoritam ponoviti onoliko puta koliko je potrebno da najbolja rješenja iz svake provedbe algoritma čine Gaussovu razdiobu. Tako će se saznati koliko je rasipanje dobivenih rješenja za jednostavni genetički algoritam

3. Odabir metode za usporedbu sličnosti rasporeda i analiza valjanosti odabira
   - Prije kreiranja algoritma s nišama potrebno je pronaći način usporedbe dvaju rasporeda kako bi se odredila njihova pripadnost određenoj niši. Naglasak treba staviti na tzv. Hamiltonovu sličnost

4. Kreiranje algoritma s nišama
   - Potrebno je kreirati više različitih varijanti genetičkih algoritama s nišama kako bi se moglo, nakon provedbe istraživanja, ustvrditi je li pojedina varijanta s nišama bolja od SGA algoritma

5. Provedba istraživanja s jednakim brojem pokušaja (evolucija) algoritma s nišama za više varijanti
   - Radi usporedbe sa SGA izvršit će se jednak broj provedbi svake odabrane varijante algoritma s nišama kako bi se moglo analizirati rasipanje rezultata

[6] Standardni problem za koji je poznato rješenje i služi za mjerenje efikasnosti rješenja (Informatički rječnik)

6. Usporedba dobivenih rezultata između genetičkog algoritma s nišama i jednostavnoga genetičkog algoritma
7. Donošenje zaključka

2. PROBLEM RASPOREĐIVANJA

Problem raspoređivanja iznimno se često pojavljuje u praksi u brojnim oblicima. Kad govorimo o problemu raspoređivanja, općenito mislimo na alokaciju resursa tako da se optimira jedna ili više funkcija cilja. Za inženjere strojarstva raspoređivanje operacija po strojevima vjerojatno je najbitnija forma problema raspoređivanja, ali svakako nije jedina. Primjerice, rezervacija poletno-sletne piste na aerodromu također spada u probleme raspoređivanja. Za informatičare, s druge strane, alokacija procesorske snage predstavlja problem raspoređivanja.

Kako problem raspoređivanja može imati mnogo oblika, tijekom godina razvitka i istraživanja nastalo je mnogo modela koji uz manja ili veća pojednostavljenja opisuju većinu klasa problema raspoređivanja. Osnovno je svojstvo klase problema raspoređivanja stupanj slobode koji osoba ili program koji izrađuje raspored ima na raspolaganju. Da bi se pojedine klase problema raspoređivanja mogle egzaktno opisati potrebno je definirati notaciju koja će se pri tome koristiti. Budući da je u literaturi poznato više varijanti notacija, u ovom je radu korištena ponešto prilagođena notacija opisana u knjizi »Scheduling: Theory, Algorithms, and Systems – Michael Pinado, Springer 2008« [9].

2.1. NOTACIJA

Problem raspoređivanja može se opisati kao set od n poslova koje treba procesuirati na m strojeva. Ako se posao može podijeliti na više operacija tada se svaka operacija može označiti s O_{ij} gdje indeks (i) predstavlja stroj, a indeks (j) predstavlja posao. Nadalje, za svaku operaciju mogu se definirati četiri svojstva:

- p_{ij} – vrijeme trajanja operacije posla (j) na stroju (i). Indeks (i) može se izostaviti ako je trajanje operacije jednako na svim strojevima, odnosno kada posao ima samo jednu operaciju,
- r_j – vrijeme početka posla (j) predstavlja najranije vrijeme kada posao (j) može započeti,
- d_j – predviđeno vrijeme završetka posla (j). Završetak posla nakon predviđena vremena dopušteno je, ali se za prekoračenje plaćaju penali.

Minimiziranje ovog svojstva često se koristi kao funkcija cilja prilikom stvaranja rasporeda,

- w_j – prioritet važnosti pojedinog posla (j) pred ostalim poslovima.

Prema predloženoj notaciji, sve klase problema raspoređivanja mogu se opisati trima parametrima $\alpha|\beta|\gamma$. Navedeni parametri imaju sljedeće značenje:

- α – predstavlja okoliš unutar kojeg se proces za koji se kreira raspored odvija. Za definiranje rasporeda nužno je navesti jednu i samo jednu vrijednost za ovaj parametar,
- β – predstavlja svojstva i ograničenja procesa. Za definiranje problema može se navesti više svojstava. Postoje i klase problema kod kojih ni jedno svojstvo ili ograničenje procesa nije navedeno,
- γ – predstavlja funkciju cilja koju treba minimizirati.

2.1.1. Okoliš procesa

Prema predloženoj notaciji definirano je devet vrsta okoliša unutar kojih se proces raspoređivanja može odvijati. Varijante okoliša pobrojene su i opisane redom od najjednostavnijih do najsloženijih:

- 1 – jedan stroj. Slučaj s jednim strojem najjednostavniji je okoliš i predstavlja specijalni slučaj složenijih okoliša
- P_m - paralelni identični strojevi. U sustavu postoji (m) identičnih strojeva postavljenih paralelno jedan u odnosu prema drugom. Poslovi koji se obavljaju u ovakvom okolišu imaju samo jednu operaciju koja može biti izvedena na bilo kojem od identičnih strojeva
- Q_m – jednoliki strojevi. U sustavu postoji (m) strojeva koji rade pri različitim brzinama, a postavljeni su paralelno jedan u odnosu prema drugom. Brzina rada pojedinog stroja označena je s v_j a vrijeme koje se za pojedini posao (posao ima samo jednu operaciju) provede na stroju (i) označava se s p_j/v_i. Ako se u promatranom okolišu nalaze identični strojevi (strojevi jednake brzine $v_i = 1$), tada je $p_{ij} = p_j$ što predstavlja P_m varijantu okoliša
- R_m – nesrodni strojevi. Ovaj okoliš predstavlja proširenje prethodnog okoliša tako da su svi strojevi u okolišu različiti. Pojednostavljeno to znači da brzina kojom se obavlja pojedina operacija ovisi o stroju i

poslu. Vrijeme u kojem se posao (j) obavlja na stroju (i) označava se s p_j/v_{ij}. Varijante okoliša R_m i Q_m identične su ako brzina rada stroja ne ovisi o poslu koji se obavlja $v_{ij} = v_j$

- F_m – linijska obrada. Okoliš u kojem postoji (m) strojeva postavljenih u seriju. Poslovi koji se obavljaju u ovom okolišu imaju jednak broj operacija koliko je i strojeva te svi poslovi moraju biti procesuirani na svim strojevima istim redoslijedom. Posao koji prvi uđe u proizvodnju prvi će iz nje i izići (nema promjene redoslijeda poslova između pojedinih operacija). Ako je dopušteno mijenjati redoslijed poslova na pojedinim strojevima između operacija, takav okoliš ima u svojstvu β oznaku permutacije (*engl. perm*)
- FF_c – prilagodljiva linijska obrada. Predstavlja poopćenu varijantu F_m okoliša. Umjesto (m) strojeva u seriji kod FF_c varijante postoji (c) serijski postavljenih grupa identičnih strojeva koji unutar grupe stoje paralelno. Svaki posao mora proći kroz sve grupe strojeva, s time da u svakoj grupi može biti procesuiran na bilo kojem od identičnih strojeva. Važno je napomenuti da za razliku od F_m okoliša ne postoji ograničenje vezano uz redoslijed pristupa pojedinim grupama strojeva
- J_m – opća obrada. Okoliš u kojem svaki posao mora proći sve strojeve prema unaprijed određenom rasporedu. Postoji dodatna varijanta u kojoj pojedini poslovi mogu biti procesuirani na pojedinim strojevima više puta. Takva varijanta rasporeda ima u svojstvu β oznaku recirkulacije (*engl. rcrc*)
- FJ_c – prilagodljiva opća obrada. Predstavlja poopćenu varijantu J_m okoliša. Poopćenje se sastoji u tome da je svih (m) strojeva zamijenjeno s (c) grupa identičnih strojeva postavljenih paralelno.
- O_m – otvorena obrada. Vrsta okoliša u kojem je potrebno sve poslove procesuirati na svim strojevima. Za razliku od J_m okoliša, neka su ograničenja uklonjena. Prije svega ne postoji unaprijed definiran redoslijed kojim poslovi moraju biti procesuirani na pojedinim strojevima. Također, vrijeme provedeno na određenom stroju može biti nula, što u principu znači da svaki posao ne mora biti procesuiran na svim strojevima

2.1.2. Svojstva i ograničenja procesa

Navedenim okolišima mogu biti nametnuta različita ograničenja. Najčešće korištena pobliže se opisuju:

- s_j – trenutak raspoloživosti. Ako se pojavljuju u β parametru klase problema raspoređivanja, tada pojedini poslovi nisu raspoloživi za procesuiranje prije s_j. Ako parametar nije naveden, tada su svi poslovi odmah spremni za procesuiranje
- prmp – prisvajanje. Svojstvo upozorava na mogućnost prekida procesuiranja posla na određenom stroju prije završetka. Naime, raspored dopušta da se procesuirani posao prekine te započne neki drugi posao. Već obavljeni dio posla ne gubi se te kad prekinuti posao ponovo dođe na isti stroj, potrebno je odraditi samo preostali dio posla
- rcrc – recirkulacija. Svojstvo koje se primjenjuje samo kod J_m i FJ_c kada pojedini posao može biti obrađen više puta na pojedinom stroju ili grupi strojeva

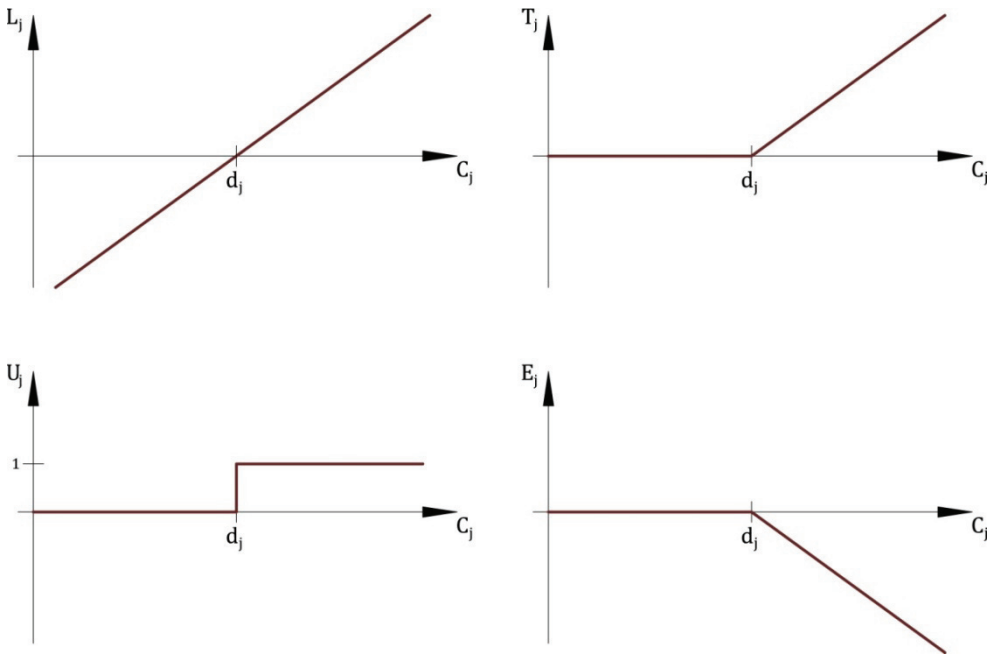
2.1.3. Funkcija cilja

Funkcija cilja predstavlja minimizaciju jednog od parametara vremena. Na području raspoređivanja definirano je pet bitnih parametara (C_j, L_j, T_j, E_j, U_j).

Vrijeme završetka operacije posla (j) na stroju (i) označava se C_{ij}. Vrijeme završetka posla jednako je vremenu završetka posljednje operacije te se označava s C_j.

Funkcija cilja može biti i minimizacija vremena kašnjenja (*engl. lateness*) L_j.

$$L_j = C_j - d_j \qquad (1)$$



Slika 2.1.1. Funkcije cilja

Vrijednost vremena kašnjenja L_j pozitivna je ako je došlo do kašnjenja, odnosno negativna ako je posao završen prije vremena.

Ako se ne želi da funkcija cilja ima negativne vrijednosti, govorimo o vremenu zaostajanja (*engl. tardiness*) T_j.

$$T_j = max(C_j - d_j, 0) = max(L_j, 0) \qquad (2)$$

Isto je tako moguće definirati funkciju cilja koja pokušava minimizirati vrijeme preranog završetka posla – preuranjenost (*engl. earliness*) E_j. Kada naručilac ne želi preuzeti robu ili uslugu prije ugovorena roka, proizvođaču je ta roba dok je na skladištu dodatan trošak.

$$E_j = max(d_j - C_j, 0) \qquad (3)$$

Ako vrijednost vremena kašnjenja nije bitna već je samo važno postoji li kašnjenje, tada je moguće definirati jediničnu kaznu (*engl. unit penalty*) U_j.

$$U_j = \begin{cases} 1 \; ako \; je \; C_j > d_j \\ u \; suprotnom \; 0 \end{cases} \qquad (4)$$

Navedene funkcije cilja ne uzimaju u obzir važnost pojedinih operacija. Naime, ako je neka operacija važnija, to bitno utječe na optimalan raspored, jer bi takva operacija trebala biti izvršena ranije.

Funkcije cilja mogu se podijeliti u dvije kategorije:

- one koje ovise o vremenu završetka C_j,
- one koje ovise o predviđenom vremenu završetka d_j.

2.1.3.1. Funkcije cilja koje ovise o vremenu završetka

U ovu kategoriju spadaju sljedeće funkcije cilja:

- $C_{max} = max_{j=1}^{N} C_j$ – plan rada. Vrijednost predstavlja vrijeme izlaska posljednjeg posla iz sustava. U praksi to znači dobro iskorištenje resursa (strojeva)
- $C_{sr} = \sum_{j=1}^{N} C_j / n$ – prosječno vrijeme završetka. Vrijednost predstavlja prosječno vrijeme izlaska svih poslova iz sustava
- $C_{sum} = \sum_{j=1}^{N} C_j$ – vrijednost predstavlja sumu izlaznih vremena svih poslova. Optimiranje ove funkcije cilja preferira postavljanje kraćih operacija na početak rasporeda.
- $C_{te.sum} = \sum_{j=1}^{N} w_j C_j$ – optimiranje funkcije cilja preferira postavljanje na početak rasporeda kraće i važnije operacije

U praksi je $max_{j=1}^{N} C_j$ najzastupljenija funkcija cilja kada se razmatra J_m varijanta okoliša. Rezultati prikazani u ovom radu dobiveni su upravo primjenom $max_{j=1}^{N} C_j$ funkcije cilja.

2.1.3.2. Funkcije cilja koje ovise o predviđenom vremenu završetka

U ovu kategoriju spadaju funkcije cilja koje pokušavaju minimizirati sljedeća svojstva:

- $L_{max} = max_{j=0}^{N} L_j$ – maksimalno kašnjenje. Vrijednost predstavlja najgore kršenje predviđenog vremena završetka posla
- $L_{sr} = \sum_{j=1}^{N} L_j / n$ – srednje vrijeme kašnjenja. Vrijednost predstavlja srednju vrijednost razlike između C_j i d_j. Bitno je istaknuti da poslovi koji su završeni prije predviđena roka, zbog svog predznaka, negativno utječu na prosječno vrijeme kašnjenja

- $L_{sum} = \sum_{j=1}^{N} L_j$ - ukupna vremena kašnjenja svih poslova
- $L_{te.sum} = \sum_{j=1}^{N} w_j L_j$ – ukupno ponderirano kašnjenje svih poslova
- $T_{max} = max_{j=1}^{N} T_j$ - maksimalno zaostajanje. U praksi to znači da je funkcija cilja direktno proporcionalna vremenu kašnjenja onog posla koji najviše odstupa od predviđenog vremena
- $T_{sr} = \sum_{j=1}^{N} L_j /n$ - prosječno vrijeme zaostajanja. Za razliku od L_{sr}, poslovi koji su završeni prije predviđena roka ne utječu na vrijednost funkcije cilja
- $T_{sum} = \sum_{j=1}^{N} T_j$ – ukupno zaostajanje
- $T_{te.sum} = \sum_{j=1}^{N} w_j T_j$ – ukupno ponderirano zaostajanje
- $E_{max} = max_{j=1}^{N} E_j$ – maksimalna preuranjenost. Funkcija cilja direktno je proporcionalna najvećem vremenu preranog završetka
- $E_{sr} = \sum_{j=1}^{N} E_j/n$ – prosječna preuranjenost
- $E_{sum} = \sum_{j=1}^{N} E_j$ – ukupna preuranjenost
- $E_{te.sum} = \sum_{j=1}^{N} E_j$ – srednja ponderirana preuranjenost
- $\sum_{j=1}^{N} U_j$ – broj poslova u kašnjenju. Potrebno je minimizirati broj poslova koji završavaju nakon predviđena roka

2.2. REDUKCIJA PROBLEMA RASPOREĐIVANJA

Prema prikazanoj notaciji problemi raspoređivanja kreću se od vrlo jednostavnih poput $1||\sum C_j$ pa sve do vrlo složenih kao što je $Q_m|prec|w_j C_j$. Mogućnost reduciranja jednog problema na drugi vrlo je važno svojstvo u teoriji raspoređivanja. Naime, ako postoje dva problema $\alpha|\beta|\gamma$ i $\alpha'|\beta'|\gamma'$ takva da se algoritam drugoga problema može primijeniti za rješavanja prvoga, tada kažemo da se problem $\alpha|\beta|\gamma$ može reducirati na $\alpha'|\beta'|\gamma'$.

$$\alpha|\beta|\gamma \propto \alpha'|\beta'|\gamma' \tag{5}$$

Neke su redukcije jednostavne poput:

$$\alpha|\beta|\sum C_j \propto \alpha|\beta|\sum w_j C_j \tag{6}$$

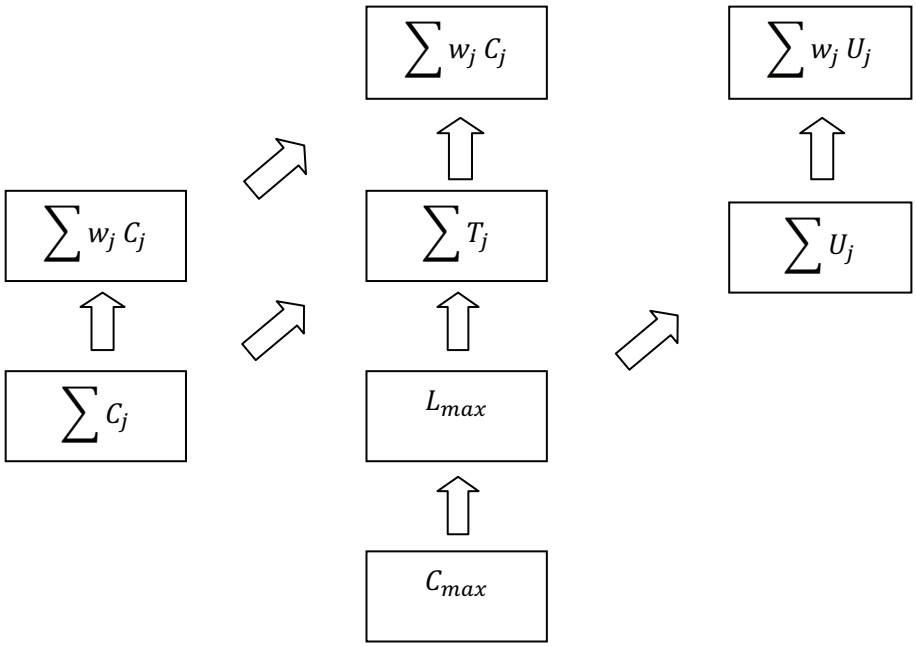
gdje algoritam za rješavanje problema s ponderiranom funkcijom cilja služi za rješavanje neponderiranog problema. Također, redukcija problema tako da se koristi algoritam sa složenijom okolinom spada u jednostavnije redukcije. Tako primjerice:

$$P_m|\beta|\gamma \propto Q_m|\beta|\gamma \propto R_m|\beta|\gamma \qquad (7)$$

također spada u jednostavnije redukcije. Korištenjem samo jednostavnih redukcija moguće je postaviti odnose između velikog broja klasa rasporeda poput:

$$1||\sum C_j \propto 1||\sum w_j C_j \propto P_m||\sum w_j C_j \propto Q_m|prec|\sum w_j C_j \qquad (8)$$

Postoje i klase rasporeda čije redukcije nisu moguće. Primjerice, klasu $P_m||\sum w_j\, C_j$ nije moguće usporediti s klasom $J_m||C_{max}$. S druge strane, moguće su redukcije između klasa koje se razlikuju samo u funkciji cilja (γ). Slika 2.2.1. prikazuje moguće redukcije.



Slika 2.2.1. Hijerarhija funkcija cilja

Sve redukcije na slici 2.2.1. logične su osim dvije:

$$\alpha|\beta|L_{max} \propto \alpha|\beta|\sum U_j \qquad (9)$$

i

$$\alpha|\beta|L_{max} \propto \alpha|\beta|\sum T_j \qquad (10)$$

Može se dokazati da se algoritmi za rješavanje problema $\alpha|\beta|\sum U_j$ i $\alpha|\beta|\sum T_j$ mogu uz manje modifikacije upotrijebiti za rješavanje problema $\alpha|\beta|L_{max}$.

Pretpostavimo da postoji algoritam koji rješava $\alpha|\beta|\sum U_j$ klasu problema. Istodobno je potrebno odrediti je li isti algoritam primjenjiv na problem $\alpha|\beta|L_{max}$ koji se sastoji od poslova kojima je predviđeno vrijeme završetka $(d_1, \dots, d_n)$. Poanta je dokaza u pronalaženju minimalne vrijednosti parametra z takvog da za problem $\alpha|\beta|\sum U_j$ s poslovima koji imaju predviđeno vrijeme završetka $(d_1 + z, \dots, d_n + z)$ postoji raspored S kojem je optimalna vrijednost funkcije cilja $\sum U_j = 0$. Pošto raspored S nema poslova u kašnjenju, može se tvrditi da je vrijeme završetka svakog posla

$$C_j \leq d_j + z \qquad (11)$$

Kako je z minimalan, zasigurno postoji posao kojem je vrijeme završetka $C_j = d_j + z$. Iz toga proizlazi da za poslove koji imaju predviđeno vrijeme završetka $(d_1, \dots, d_n)$ maksimalno kašnjenje za raspored S iznosi $L_{max} = z$. Ovim jednostavnim dokazom pokazali smo da je redukcija

$$\alpha|\beta|L_{max} \propto \alpha|\beta|\sum U_j \qquad (12)$$

moguća. Poznavanje mogućnosti redukcije jedne klase problema na drugu pridonosi jednostavnom određivanju težine problema. Primjerice, ako je problem A definiran kao jednostavan, a problem B može se reducirati na problem A, tada se može zaključiti da je i problem B jednostavan.

2.3. TEŽINA PROBLEMA

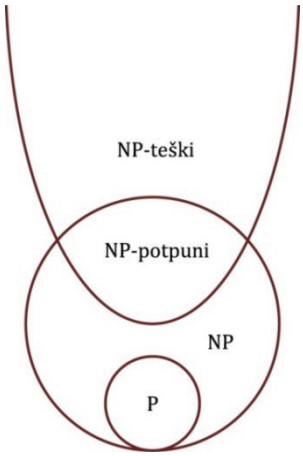
Problem se može okarakterizirati kao jednostavan ili lako rješiv ako postoji algoritam koji u polinomnom vremenu može odrediti optimalno rješenje. Kreirati algoritam koji rješava neki problem samo po sebi nije dovoljno. Naime, katkad je vrijeme potrebno za pronalaženje rješenja predugačko. Vrijeme koje je potrebno za izvršavanje algoritma izravno ovisi o veličini problema. Tako, primjerice, ako se rješava problem trgovačkog putnika, vrijeme izvršenja algoritma direktno je proporcionalno broju gradova te se algoritam smatra efikasnim ako je vrijeme koje troši pri izvršavanju polinomno s obzirom na veličinu problema, tj. ako mu je potrebno najviše Cn^k koraka, gdje je C konstanta, a k prirodan broj. Takve probleme odluke koji se mogu riješiti deterministčkim Turingovim strojem u polinomnom vremenu nazivamo P[7] klasom problema. S druge strane postoji i NP klasa problema odluke čiji se problemi mogu riješiti nedeterminističkim Turingovim strojem u polinomnom vremenu. Takvi problemi imaju svojstvo da im se rješenja mogu učinkovito provjeriti. Vrijeme izvršavanja algoritma i dalje direktno ovisi o veličini problema, s tom razlikom da se veličina problema nalazi u eksponentu. Pojednostavljeno govoreći, problemi koji pripadaju NP[8] klasi izrazito su teško rješivi, odnosno vrijeme potrebno za njihovo rješavanje izvan dosega je današnjih računala.

Najvažnije pitanje računalne znanosti trenutačno glasi: je li NP klasa jednaka P klasi. Postavljeno pitanje implicira da prividno vrlo teški problemi imaju vrlo jednostavno rješenje. Odgovor na ovo pitanje još je nepoznat, dapače institut Clay Mathematics ponudio je nagradu od milijun dolara onome tko pruži dokaz za prihvaćanje, odnosno odbacivanje te tvrdnje. Danas prevladava mišljenje da tvrdnja P=NP nije točna; dapače većina znanstvenika smatra da problem uopće nije rješiv. Da bi se donekle približilo odgovoru na postavljeno pitanje identificirani su najteži problemi u NP klasi. Navedena skupina problema naziva se NP-potpunim problemima. Važna je jer ako se pronađe algoritam za rješenje te klase problema u polinomnom vremenu, tada je moguće riješiti sve probleme iz NP klase u polinomnom vremenu. Naravno, da bi bilo moguće pronaći algoritam, potrebno je poznavati barem jedan NP-potpuni problem. Prvi NP-potpuni problem pronašao je Stephen Cook 1971.

[7] P – Polynomial time
[8] NP – Nondeterministic Polynomial time

[10] godine. Riječ je o problemu CIRCUIT SAT u kojem je za određeni elektronički krug s *n* binarnih ulaza {0, 1} i jednim binarnim izlazom potrebno odrediti postoji li kombinacija ulaza koja daje 1 na izlazu.

Vrlo brzo nakon pronalaska prvoga problema iz klase NP-potpunih problema Richard Karp [11] objavio je ideju o uporabi redukcije pojedinih problema na jedan od poznatih NP-potpunih problema (npr. CIRCUIT SAT) te dokazao postojanje dodatnih 21 problema koji spadaju u NP-potpunu klasu. Danas je poznato više od 3000 NP-potpunih problema, među kojima je i velik broj problema raspoređivanja. Općenito gledano, većina problema raspoređivanja spada u NP-teško klasu problema, što znači da su barem jednako teški kao i NP problemi ali nužno ne spadaju u NP klasu problema. Na slici 2.3.1. prikazan je odnos P, NP, NP-potpunih i NP-teških problema. Riječ je samo o pretpostavci te ne postoji dokaz da je njihov odnos upravo onakav kako je to prikazano na slici 2.3.1.



Slika 2.3.1. Vannov dijagram za P, NP, NP-potpunu i NP-teško klasu problema (P≠NP)

Osim prikazane redukcije između problema raspoređivanja može se provesti i redukciju problema raspoređivanja na neki drugi poznati NP-potpuni problem. Takva redukcija omogućuje dokazivanje da su neki naizgled jednostavni problemi raspoređivanja zapravo iznimno teški, odnosno da za njih nije moguće osmisliti algoritam čiji je broj potrebnih koraka u polinomnom odnosu s veličinom problema.

2.4. NP-TEŠKI PROBLEMI RASPOREĐIVANJA (PRIMJER)

Kao primjer će nam poslužiti dva problema $P2||C_{max}$ i $1|prec|\sum U_j$ koji se mogu redom svesti na problem DIJELJENJA (*engl. partition*) i na problem KLIKE (*engl. clique*).

DIJELJENJE je problem kod kojeg je potrebno podijeliti set od n brojeva $\{a_1, \dots, a_n\}$ na dva jednaka dijela odnosno mora vrijediti

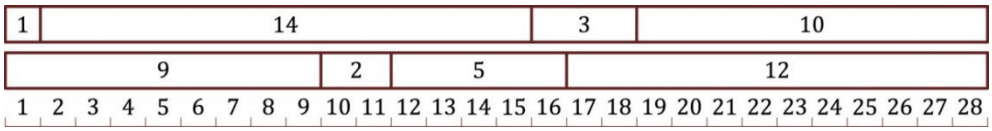
$$\sum a_j = 2A \tag{13}$$

i

$$\sum_{x \in S} x = A \tag{14}$$

Za bilo koji primjer klase $P2||C_{max}$ (okoliš s dva identična stroja) koji se sastoji od n poslova čije je trajanje p_j jednako a_j može se tvrditi da je C_{max} jednak A ako i samo ako takva podjela postoji. Ako postoji set brojeva S čija je suma jednaka A, tada je zasigurno suma preostalih brojeva u setu također jednaka A. Iz toga proizlazi da se poslovi koji korespondiraju s brojevima u setu S izvršavaju na prvom stroju dok se ostali poslovi izvršavaju na drugom stroju. Budući da je trajanje rasporeda na oba stroja jednako dugačko i iznosi A, ne postoji raspored koji bi imao manji C_{max}. Opisanom redukcijom pokazali smo da je $P2||C_{max}$ NP-težak problem jer smo ga uvođenjem određenih pojednostavljenja sveli na problem DIJELJENJA za koji znamo da je NP-potpun problem. Na ovaj način upozorili smo na to da za njegovo rješavanje ne postoji algoritam koji bi ga riješio u polinomnom vremenu.

Pokazat ćemo to jednim primjerom. Potrebno je podijeliti set {1,2,3,5,9,10,12,14} na dva jednaka dijela (A=28). Budući da je riječ o NP-potpunom problemu, nije ga moguće izračunati u polinomnom vremenu. S druge strane, ako znamo rješenje, moguće je njegovu ispravnost provjeriti u polinomnom vremenu. Potrebno je zbrojiti brojeve i vidjeti je li zbroj iznosi 28. Za navedeni primjer znamo da postoji rješenje $S_1 = \{1,14,3,10\}$ i $S_2 = \{9,2,5,12\}$. Pretočeno u raspored koji odgovara $P2||C_{max}$ može se napraviti Ganttov dijagram prikazan na slici 2.4.1..



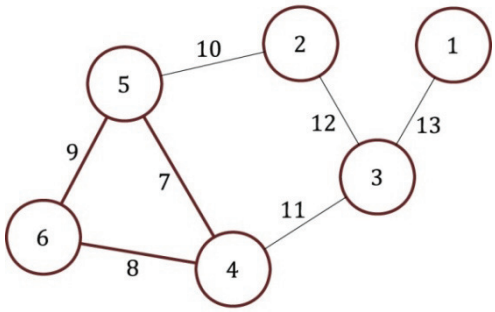
Slika 2.4.1. Ganttov dijagram za NP-potpun problem raspoređivanja

Iz slike se jasno vidi da oba stroja rade ukupno 28 vremenskih jedinica, što je ujedno i optimalno rješenje.

Sljedeći problem na prvi pogled izgleda vrlo jednostavan, jer se okoliš sastoji od samo jednog stroja. Ograničenje koje pojedinim poslovima daje prednost bitno povećava složenost problema te ga svrstava u skupinu NP-teških problema. Da je tome tako pokazat ćemo svođenjem na problem KLIKE. Navedeni problem na popisu je 21 problema za koje je Richard Karp dokazao pripadnost NP-potpunoj grupi redukcijom iz poznatog CIRCUIT SAT problema[9].

Problem KLIKE sastoji se od odgovora na pitanje postoji li najmanje (c) čvorova u grafu (G) koji su međusobno povezani. Na slici 2.4.2. prikazan je graf $G = (V, E)$[10] sa $k = 6$ čvorova i $b = 7$ veza. U konkretnom primjeru postoji klika veličine $c = 3$. Za svaku kliku moguće je egzaktno odrediti broj veza prema formuli:

$$l = \frac{1}{2}c(c-1) \qquad (15)$$



Slika 2.4.2. Primjer klike

[9] *Reducibility Among Combinatorial Problems*
[10] G – Graph; V – Vertices; E - Edges

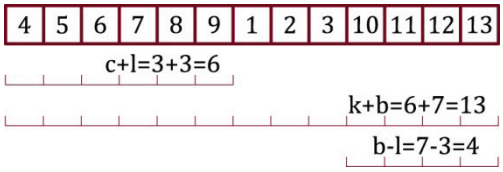
te za primjer na slici 2.4.2. klika ima 3 veze.

$$l = \frac{1}{2}c(c-1) = \frac{1}{2}3(3-1) = 3 \tag{16}$$

U primjeru rasporeda klase $1|prec|\sum U_j$ veličine $n = k + b$ postoje dvije vrste poslova: poslovi koji korespondiraju čvorovima i poslovi koji korespondiraju vezama. Vrijeme procesuiranja svih poslova neovisno o vrsti iznosi $p_j = 1$. Kako bi se odredio broj zakašnjelih poslova U_j, potrebno je odrediti predviđeno vrijeme završetka poslova d_j. Poslovi koji korespondiraju s čvorovima imaju predviđeno vrijeme završetka $d_j = k + b$, a poslovi koji korespondiraju s vezama imaju $d_j = c + l$. Također je za problem potrebno definirati ograničenje prednosti izvršenja pojedinih poslova. Konkretno, prije izvršenja pojedinog posla koji korespondira s vezama nužno je da budu provedena oba posla koja korespondiraju s čvorovima. Tako primjerice za svaku trojku koja se sastoji od poslova u, v i (u, v) postoje sljedeća ograničenja:

$$\begin{matrix} u \rightarrow (u, v) \\ v \rightarrow (u, v) \end{matrix} \tag{17}$$

Za ovako postavljen problem može se tvrditi da postoji raspored s točno $b - l$ zakašnjelih poslova ako postoji graf s klikom veličine c. Za stvaranje rasporeda potrebno je prvo rasporediti poslove (c) koji odgovaraju čvorovima u grafu pa odmah iza njih zbog ograničenja prednosti poslove koji odgovaraju vezama unutar klike (l). Vrijeme potrebno da se obavi posljednji posao vezan uz veze u kliki iznosi $c + l$, što znači da ni jedan od tih poslova ne kasni. Budući da su za raspoređivanje preostali samo poslovi koji ne pripadaju zadanoj kliki, lako se može uočiti da $m - l$ poslova koji korespondiraju s preostalim vezama neće biti obavljen na vrijeme. Naime, predviđeno je vrijeme završetka za poslove koji korespondiraju s vezama $d_j = c + l$, a to je vrijeme već postignuto. S druge strane, poslovi koji korespondiraju s čvorovima koji ne pripadaju kliki bit će obavljani na vrijeme jer je njihovo predviđeno vrijeme završetka jednako ukupnom trajanju rasporeda $d_j = k + b$. Raspored koji korespondira problemu prikazanom na slici 2.4.2. prikazan je na slici 2.4.3.



Slika 2.4.3. Raspored zasnovan na problemu klike

Iz slike se vidi da je broj zakašnjelih poslova 4, što u potpunosti odgovara broju veza između čvorova koji ne pripadaju kliki.

Dva prikazana primjera redukcije pojedinih klasa raspoređivanja na jedan od NP-potpunih problema neupitno dokazuju pripadnost pojedinih klasa raspoređivanja grupi najtežih kombinatornih problema. Posebno je bitno uočiti da među najteže kombinatorne probleme spadaju i problemi raspoređivanja s jednim strojem.

2.5. PRAVILA RASPOREĐIVANJA NA PROBLEMU S JEDNIM STROJEM

Analiza problema s jednim strojem iznimno je bitna jer se navedeni okoliš može smatrati podvrstom nekih drugih složenijih okoliša. Naime, okoliš u kojem se nalazi samo jedan stroj može se smatrati okolišem s uskim grlom (*engl. capacity constrained resources*). Kompliciraniji problemi raspoređivanja koji se odvijaju u okolišu s više strojeva, ako imaju samo jedno usko grlo, mogu se svesti na problem s jednim strojem. Kako bi se odredio optimalan raspored poslova moguće je primijeniti tzv. prioritetna pravila koja pružaju relativno dobre smjernice za kreiranje rasporeda. Ovdje je bitno istaknuti da odabir pravila odnosno smjernica ovisi o funkciji cilja, tako da se s promjenom funkcije cilja mijenjaju smjernice, odnosno sve smjernice nisu dobro prilagođene svakom problemu. U praksi se najčešće koriste sljedeća pravila:

- SPT – (*engl. Shortest Processing Time*). Posao koji najkraće traje obavlja se prvi
- FCFS – (*engl. First Come First Served*). Posao koji prvi stigne u radionicu biti će prvi procesuiran

- EDD – (*engl. Earliest Due Date*). Posao s najskorijim predviđenim vremenom završetka biti će prvi procesuiran
- CR – (*engl. Critical Ration*). Kritični je odnos indeks koji se dobiva dijeljenjem vremena preostalog do predviđena završetka posla i preostalog dijela posla. Posao s najmanjim kritičnim odnosom obavlja se prvi

Primjenom pojedinih pravila moguće je dobiti posve različite rasporede od kojih svi svakako nisu optimalni. Uzet ćemo primjer u kojem je potrebno na jednom stroju rasporediti pet poslova sa sljedećim svojstvima.

Tablica 2.5.1. Tehnološki raspored operacija na jednom stroju

Posao	p_j	d_j	r_j
A	6	8	0
B	3	18	3
C	9	23	5
D	2	6	0
E	8	13	1

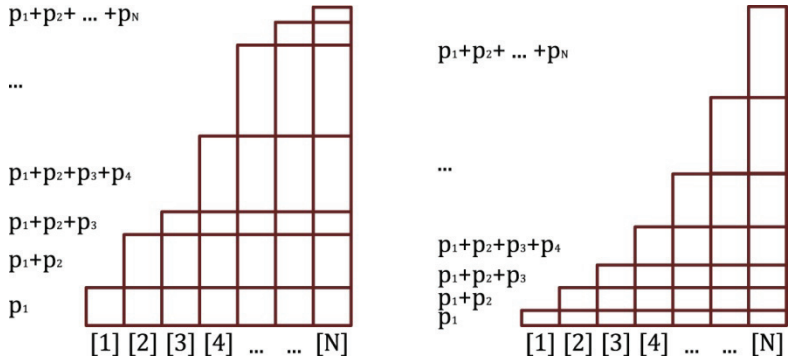
2.5.1. SPT – najkraće vrijeme obrade

Pravilo SPT izvorno dolazi iz računarstva, gdje je primarno primijenjeno u dizajnu operativnih sustava [12]. Naime, projektantima operativnih sustava najvažniji je cilj bio držati što manji broj poslova na čekanja.

Pravilo prema kojem se favoriziraju najkraći poslovi vrlo je bitno jer daje optimalno rješenje za probleme klase $1||\sum C_j$ i $1||\sum F_j$

$$\sum_{j=1}^{N} F_j = \sum_{j=1}^{N} (C_j - r_j) = \sum_{j=1}^{N} C_j - \sum_{j=1}^{N} r_j \qquad (18)$$

Kako je $\sum r_j$ konstanta i ne ovisi o rasporedu poslova, pravilo je primjenjivo na obje klase problema. Naime, potrebno je svesti na minimum vrijeme koje je za pojedini posao potrebno u radionici. Budući da okoliš u kojem se odvija proces sadrži samo jedan stroj, posao koji se obavlja na početku izaziva kašnjenje svih preostalih poslova. Stoga je oportuno na početak rasporeda staviti kraće poslove. Slika 2.5.1. zorno pokazuje ispravnost navedene tvrdnje.

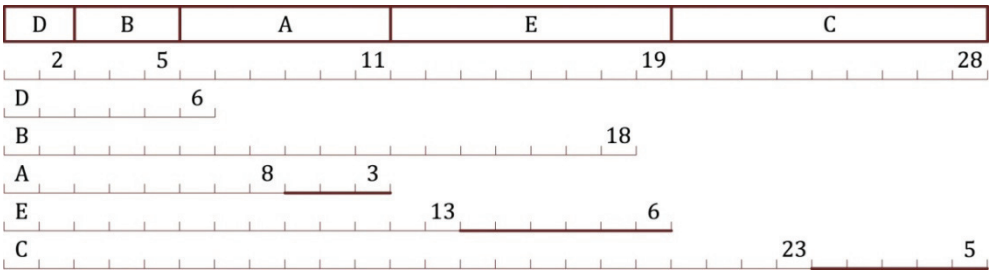


Slika 2.5.1. Redosljed poslova

Naime, svaki posao u rasporedu utječe na kašnjenje samo onih poslova koji su prema rasporedu poslije njega. Budući da su svi poslovi došli u radionicu istodobno ($r_j = 0$), svakom se pojedinom poslu u vrijeme provedeno u radionici uračunavaju vremena svih poslova koji su obavljeni prije njega. Da bi se minimiziralo zbroj vremena provedeno u radionici za sve poslove potrebno je smanjiti površinu grafa prikazanog na slici 2.5.1. Površina grafa najmanja je ako su poslovi poredani prema sljedećem pravilu:

$$p_1 \leq p_2 \leq p_3 \leq p_4 \leq \cdots \leq p_N \qquad (19)$$

Potkrijepimo to primjenom SPT pravila na tehnološki raspored prikazan u tablici 2.5.1.



Slika 2.5.2. SPT

Za prikazani raspored bitno je da su poslovi poredani prema dužini trajanja (D, B, A, E, C). Osim dvaju najkraćih poslova (B, D), prema predviđenu vremenu završetka svi ostali poslovi kasne. Kako su za kraj ostavljeni najduži poslovi, rezultat je više nego logičan.

Kako bi se mogla usporediti kvaliteta raznih pravila i smjernica potrebno je definirati određena mjerila efikasnosti. Za usporedbu koristit će se $\sum F_j$, $maxF_j$, $\sum C_j$, $maxC_j$, $\sum L_j$, $maxL_j$, $\sum T_j$, $maxT_j$.

Tablica 2.5.2. Rezultati primjene pravila SPT

Posao	p_j	d_j	r_j	C_j	$F_j = C_j - r_j$	$L_j = C_j - d_j$	T_j
A	6	8	0	11	11	3	3
B	3	18	0	5	5	-13	0
C	9	23	0	28	28	6	6
D	2	6	0	2	2	-4	0
E	8	13	0	19	19	6	6
				$\sum C_j$=65	$\sum F_j$=65	$\sum L_j$=-2	$\sum T_j$=15
				$maxC_j$=28	$maxF_j$ =28	$maxL_j$=6	$maxT_j$=6

Dobivene vrijednosti $\sum C_j$ i $\sum F_j$ minimalne su za probleme klase $1||\sum C_j$ $1||\sum F_j$ odnosno ne postoji pravilo kojim bi se postigle manje vrijednosti. Važno je da je u navedenom problemu $r_j = 0$, a ne kako je opisano u tablici 2.5.1. Ako bi poslovi dolazili u radionicu u različita vremena, tada bi bila riječ o dinamičkom problemu klase $1|r_j|\sum C_j$ koji spada u kategoriju NP-teških problema te nije poznat algoritam kojim ga je moguće riješiti u polinomnom vremenu. Uvođenjem dodatnog ograničenja, odnosno dopuštanjem prekidanja poslova ako u radionicu dođe posao čije je vrijeme trajanja kraće od preostalog vremena posla koji se trenutačno obavlja, tada problem ponovno postaje rješiv u polinomnom vremenu uporabom pravila SRPT[11].

[11] SRPT - Shortest Remaining Processing Time

2.5.2. FCFS – Prvi dolazak – prva obrada

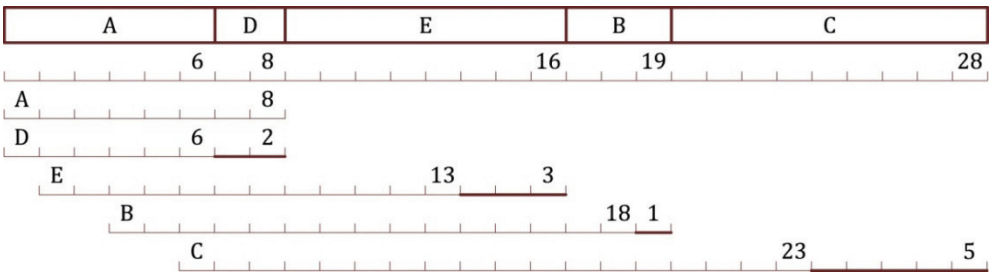
Primjenom pravila SPT i SRPT, poslovi koji imaju dugačko vrijeme izvršenja neprirodno se dugo zadržavaju u radionici. Da bi se to izbjeglo potrebno je promijeniti kriterij koji se minimizira, čime se mijenja i klasa problema.

$$1||maxF_j \qquad (20)$$

Optimalno rješenje dobiva se primjenom pravila FCFS. Pojednostavljeno to znači da se u raspoređivanju operacija po strojevima pokušava napraviti raspored u kojem niti jedan posao nije predugo u radionici.

FCFS je pravilo raspoređivanja koje se primjenjuje u veoma velikom broju djelatnosti te je stoga iznimno popularno. Primjerice, dolazak na blagajnu u prodavaonici odvija se prema pravilu FCFS. Odabir sjedećeg mjesta na stadionu također se provodi prema pravilu FCFS. Čak se i neke vrlo složene radnje poput ukrcavanja u zrakoplove provode prema pravilu FCFS. Naime, neki niskotarifni prijevoznici dopuštaju svojim putnicima da sjednu na bilo koje mjesto, ovisno o vremenu čekiranja.

Prema primjeru iz tablice 2.5.1. poslovi dolaze u radionicu redom A, D, E, B i na kraju C. Kako je na raspolaganju samo jedan stroj, posao D može doći na obradu tek nakon što je posao A posve obavljen i tako redom. Uvrštavanjem poslova prema redoslijedu dolaska u radionicu dobiva se raspored kao što je prikazano na slici 2.5.3.



Slika 2.5.3. FCFS

Kao što se iz slike može vidjeti, poslovi B, C, D i E kasne u odnosu prema predviđenu vremenu završetka, a to svakako nije povoljno.

Za primjer definiran tablicom 2.6.1. prilikom primjene metode FCFS dobivaju se sljedeće vrijednosti mjerila efikasnosti.

Tablica 2.5.3. Rezultati primjene pravila FCFS

Posao	p_j	d_j	r_j	C_j	$F_j = C_j - r_j$	$L_j = C_j - d_j$	T_j
A	6	8	0	6	6	–2	0
B	3	18	3	19	16	1	1
C	9	23	5	28	23	5	5
D	2	6	0	8	8	2	2
E	8	13	1	16	15	3	3
				$\sum C_j$=77	$\sum F_j$=68	$\sum L_j$=9	$\sum T_j$=11
				$maxC_j$=28	$maxF_j$ =23	$maxL_j$=5	$maxT_j$=5

Usporedbom pravila SPT i FCFS za problem $1||\sum C_j$ očigledno je da je pravilo SPT bolje. Rezultat je logičan jer za navedenu klasu problema pravilo SPT daje optimalno rješenje. S druge strane za problem klase $1||maxF_j$ pravilo FCFS daje bolje rješenje jer je ono optimalno.

Na kraju je potrebno obratiti pozornost na jednu činjenicu. Naime, ako svi poslovi dolaze u radionicu istodobno ($r_j = 0$), tada pravilo FCFS neće koristiti jer se problem svodi na problem $1||maxC_j$. Za razmatrani primjer $maxC_j$ uvijek je 28, bez obzira na odabrani raspored, odnosno svi rasporedi su optimalni.

2.5.3. EDD – najkraće predviđeno vrijeme završetka

Do sada nisu razmatrani rasporedi koji ovise o predviđenom vremenu završetka poslova (d_j). Prva klasa problema koja će se razmatrati je $1||\sum L_j$. Problem se može riješiti uporabom već opisanog pravila SPT.

$$\sum_{j=1}^{N} L_j = \sum_{j=1}^{N} (C_j - d_j) = \sum_{j=1}^{N} C_j - \sum_{j=1}^{N} d_j \qquad (21)$$

Naime, navedenom analizom se može uočiti da je minimiziranje $\sum L_j$ ekvivalentno minimiziranju $\sum C_j$ što znači da su $1||\sum L_j$ i $1||\sum C_j$ klase problema koje se mogu riješiti istim pravilima.

Situacija je značajno složenija kod $1||\sum T_j$ problema jer ne postoji jednostavna supstitucija »max« operatora te tako ne postoji niti pravilo koje daje optimalno rješenje.

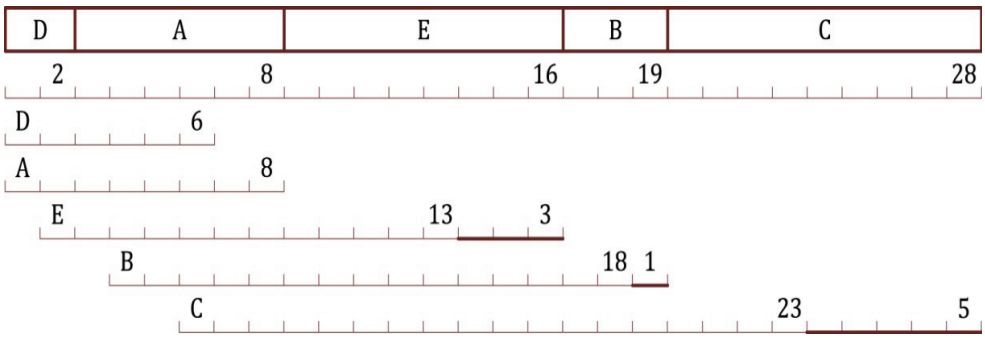
$$T_j = max(C_j - d_j, 0) = max(L_j, 0) \tag{22}$$

Dapače, Du i Leung [13] su dokazali da se radi o NP-teškom problemu. Jednako tako nestatička klasa problema $1|r_j|\sum L_j$ dokazano spada u NP-teške probleme.

Za preostale dvije nestatičke klase $1||maxL_j$ i $1||maxT_j$ postoji pravilo koje daje optimalno rješenje. Prema tom pravilu potrebno je poslove poredati prema predviđenim vremenima završetka poslova.

$$d_1 \leq d_2 \leq d_3 \leq d_4 \leq \cdots \leq d_N \tag{23}$$

Primjenom tog pravila dobiva se sljedeće rješenje.



Slika 2.5.4. EDD

Tablica 2.5.4. Rezultati primjene pravila EDD

Posao	p_j	d_j	r_j	C_j	$F_j = C_j - r_j$	$L_j = C_j - d_j$	T_j
A	6	8	0	8	8	0	0
B	3	18	3	19	16	1	1
C	9	23	5	29	23	5	5
D	2	6	0	2	2	–4	0
E	8	13	1	16	15	3	3
				$\sum C_j$=74	$\sum F_j$=64	$\sum L_j$=5	$\sum T_j$=9
				$maxC_j$=28	$maxF_j$ =23	$maxL_j$=5	$maxT_j$=5

2.5.4. CR – kritični odnos

Prema ovom pravilu posao koji ima najmanji kritični odnos ima prioritet u odnosu na ostale poslove. Pojednostavljeno to znači da se taj posao nalazi najbliže zoni kašnjenja ili već kasni. Kritični odnos se određuje prema formuli.

$$CR = \frac{d_j - \ trenutno\ vrijeme}{\sum_{i=1}^{M} t_{ij}} \qquad (24)$$

Primjenom ovog pravila vrlo je jednostavno odrediti koji posao treba sljedeći izvršiti te je pravilo balans između pravila SPT koje uzima u obzir samo trajanje poslova i pravila EDD koje u obzir uzima samo predviđeno vrijeme završetka. Naime, CR se smanjuje kako se približava predviđeno vrijeme završetka, te istodobno favorizira dugačke poslove. S druge strane, bitan mu je nedostatak što se postupak određivanja kritičnog odnosa mora ponoviti poslije raspoređivanja svakog posla.

Primjenom pravila CR na primjer definiran u tablici 2.5.1. dobivaju se sljedeći rezultati u prvom koraku.

Tablica 2.5.5. Prvi korak primjene pravila CR

Posao	p_j	d_j	r_j	C_j	CR
Trenutno vrijeme = 0					
A	6	8	0	6	(8-0)/6=1,333
B	3	18	0	–	(18-0)/3=6,000
C	9	23	0	–	(23-0)/9=2,556
D	2	6	0	–	(6-0)/2=3,000
E	8	13	0	–	(13-0)/8=2,625

Posao A ima najmanji CR, pa se obavlja prvi. Kad se obavio posao A, ponovo se izračunava CR, ali ovaj put bez operacije A i s trenutačnim vremenom = 6.

Tablica 2.5.6. Drugi korak primjene pravila CR

Posao	p_j	d_j	r_j	C_j	CR
Trenutno vrijeme = 6					
B	3	18	0	–	(18–6)/3=4,000
C	9	23	0	–	(23–6)/9=1,888
D	2	6	0	8	(6–6)/2=0,000
E	8	13	0	–	(13–6)/8=0.875

Posao D ima CR jednak nuli, što znači da kasni. Osim njega kasni i posao E jer mu je CR manji od jedan.

Tablica 2.5.7. Treći korak primjene pravila CR

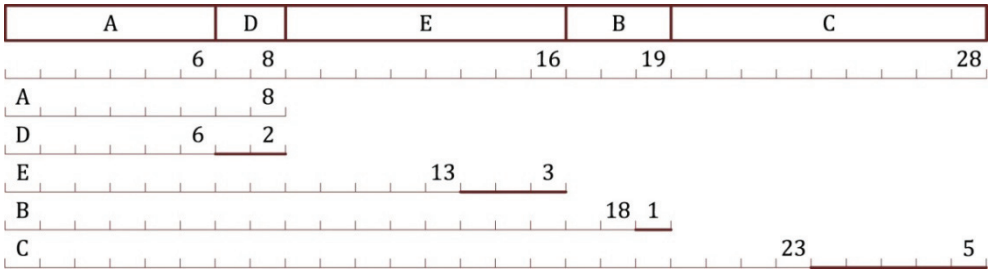
Posao	p_j	d_j	r_j	C_j	CR
Trenutno vrijeme = 8					
B	3	18	0	–	(18–8)/3=3,333
C	9	23	0	–	(23–8)/9=1,666
E	8	13	0	16	(13–8)/8=0.625

U trećem koraku za obavljanje odabran je posao E. Preostaje posljednje određivanje kritičnog odnosa za zadnja dva posla.

Tablica 2.5.8. Posljednji korak primjene pravila CR

Posao	p_j	d_j	r_j	C_j	CR
Trenutno vrijeme = 16					
B	3	18	0	–	(18–16)/3=0,666
C	9	23	0	–	(23–16)/9=0,777

Kao što se iz posljednjeg izračuna kritičnog odnosa može vidjeti, oba posljednja posla kasne, s time da se na obradu prvo šalje posao B. Na slici 2.5.5. prikazan je raspored dobiven primjenom pravila CR.



Slika 2.5.5. CR

2.6. DEFINICIJA PROBLEMA (JSSP)

Problem raspoređivanja koji se razmatra u ovom radu spada u klasu $J_m||C_{max}$ te se općenito može formulirati na sljedeći način [14]:

- Svaki se posao mora provesti na svakom stroju, i to redoslijedom koji je unaprijed zadan tehnološkim procesom.
- Svaki stroj u jednom trenutku može procesuirati samo jedan posao.
- Provođenje posla (j) na stroju (i) naziva se operacija O_{ij}.

- Operacija O_{ij} zahtijeva ekskluzivno korištenje stroja (i) u trajanju od p_{ij}.
- Početno i završno vrijeme operacije O_{ij} označava se sa r_{ij} i d_{ij}. Raspored predstavlja skup završnih vremena svih operacija koje zadovoljavaju već navedena ograničenja.
- Vrijeme potrebno za dovršetak svih poslova naziva se plan obrade i označava se sa C_{max}. Po definiciji $C_{max} = max_{j=1}^{N} C_j$.

Za pobliže objašnjenje problema raspoređivanja poslužit ćemo se jednostavnim primjerom u kojem treba rasporediti tri posla na tri stroja, s time da se svaki posao sastoji od tri operacije. U tablici 2.6.1. dan je tehnološki redoslijed operacija za svaki posao s pripadajućim vremenima trajanja. Tako se primjerice posao 1 prvo obavlja na stroju 3 u trajanju od 3 vremenske jedinice, potom na stroju 1 u trajanju od 2 vremenske jedinice te na kraju na stroju 2 u trajanju od 3 vremenske jedinice.

Tablica 2.6.1. Tehnološki raspored operacija s pripadajućim vremenima trajanja

Posao	**Stroj (trajanje operacije)**		
1	3(3)	1(2)	2(3)
2	1(1)	2(4)	3(3)
3	1(2)	3(2)	2(1)

Tehnološki proces opisan tablicom 2.6.1. u potpunosti definira matrica T_{jk}:

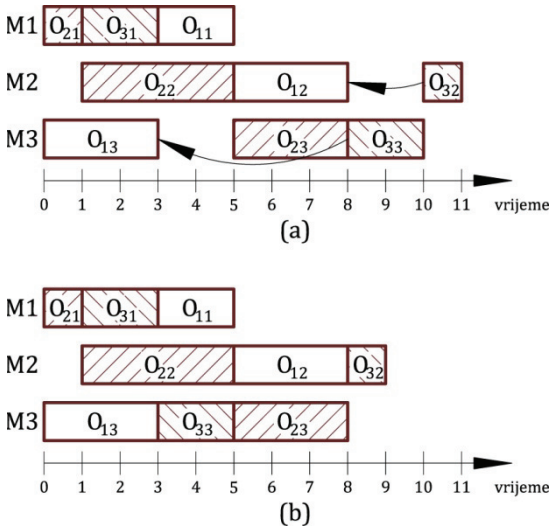
$$T_{jk} = \begin{vmatrix} 3 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{vmatrix} \qquad (25)$$

Istodobno matrica trajanja pojedinih operacija označava se sa p_{jk}te glasi:

$$p_{jk} = \begin{vmatrix} 3 & 2 & 2 \\ 1 & 4 & 3 \\ 2 & 2 & 1 \end{vmatrix} \qquad (26)$$

Najjednostavniji prikaz rasporeda može se dati uporabom Ganttova dijagrama. Slika 2.6.1. prikazuje svaku operaciju iz 3x3 problema u obliku pravokutnika gdje koordinata x predstavlja trajanje pojedine operacije. Činjenica da

su na slici 2.6.1. prikazana dva Ganttova dijagrama koja istodobno prikazuju identična rješenja, upozorava na to da za potpunu definiciju rješenja treba razmotriti pitanje prezentacije rješenja te pitanje klase rasporeda.



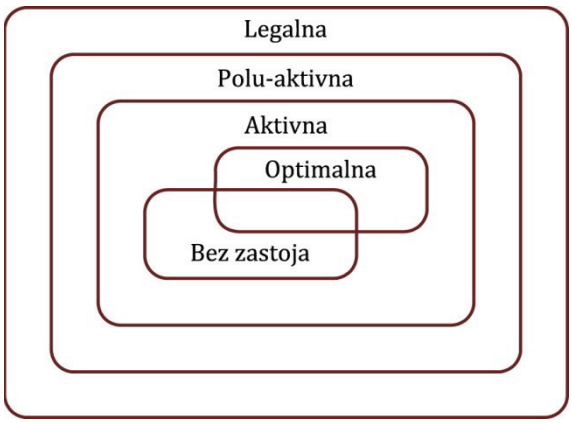
Slika 2.6.1. Ganttov dijagram i dopušteni lijevi pomak a)poluaktivni raspored b)aktivni raspored

2.7. KLASE RASPOREDA

Ganttov dijagram na slici 2.6.1. prikazuje aktivni (*engl. active schedule*) i poluaktivni raspored (*engl. semi-active schedule*). Osim njih postoji i takozvani raspored bez strojnog zastoja (*engl. non-delay schedule*), ali krenimo redom.

Prema definiciji, kada se ni jedna operacija u rasporedu ne može pomaknuti u lijevo a da ne naruši vremena drugih operacija, takav raspored nazivamo aktivnim. Ako se pobliže promotri Ganttov dijagram na slici 2.6.1., lako se može uočiti da se operacija O_{33} može pomaknuti ispred operacije O_{23} bez utjecaja na operaciju O_{23}. Takav potez naziva se dopušteni lijevi pomak (*engl. permissible left shift*), a novonastali raspored smatra se aktivnim. Sve dok postoji mogućnost provedbe lijevog dopuštenog pomaka raspored se smatra poluaktivnim.

Aktivni je raspored svakako bolji od poluaktivnog jer predstavlja podskup rasporeda bliži optimalnom (za C_{max} funkciju cilja). Osim aktivnog i poluaktivnog rasporeda postoji i raspored bez strojnog zastoja koji je još uži podskup rasporeda u odnosu na aktivni raspored. Po definiciji, raspored bez strojnog zastoja legalan je raspored kod kojeg ni jedan stroj nije u fazi čekanja dok god ima operacija za procesuiranje te je podvrsta aktivnoga rasporeda [15]. Do sada je predloženo nekoliko algoritama koji su primjenjivali raspored bez zastoja, ali općenito gledano, njegovo korištenje nije preporučljivo. Kao što to slika 2.7.1. prikazuje, iako raspored bez strojnog zastoja dodatno sužava prostor pretraživanja, on se ne podudara s optimalnim rješenjem te može usmjeriti algoritam u krivom smjeru. Jedina bitna prednost takvog rasporeda jednostavnija je konstrukcija u odnosu na aktivni raspored, što nije zanemarivo ako je brzina izračuna bitnija od globalnog optimuma.



Slika 2.7.1. Klase rasporeda

2.7.1. Pravila za stvaranje rasporeda

Problem JSSP iznimno je težak optimizacijski problem te ne postoji generički algoritam za njegovo rješavanje koji daje optimalno rješenje u razumnu vremenu. U prošlosti razni su istraživači predlagali razna pravila i smjernice koji su s manjim ili većim uspjehom rješavali problem JSSP. S vremenom se broj predloženih pravila povećavao tako da su već 1977. Panwalker i Iskander [16] pobrojili više od stotinu takvih pravila. Većina se pravila temelji na pravilima koja su primijenjena na problemu s jednim strojem i svode se na sljedeće: ako

je operacija na stroju m^* upravo završila, potrebno je odrediti sljedeću operaciju koja će se odvijati na tom stroju. Skup poslova J^* predstavlja poslove koji trebaju biti izvršeni na stroju m^*. Koji će se od navedenih poslova izvršiti određuje odabrano pravilo. Algoritam ima tri osnovna koraka:

- određivanje skupa poslova J^*
- određivanje prioritetnog indeksa za svaki posao $j \in J^*$
- odabir posla koji ima najbolji prioritetni indeks.

Iz navedenog algoritma može se odmah uočiti da odabir sljedeće operacije izravno ovisi o načinu određivanja prioritetnog indeksa. Kao što su Panwalker i Iskander pokazali, postoji više od stotinu načina kako odrediti prioritetni indeks. Pravila najsličnija pravilima opisanim u poglavlju 2.5. jesu:

- SPT – (*engl. shortest processing time*). Odabire se posao s najkraćim vremenom procesuiranja na stroju m^*
- LWR – (*engl. least work remaining*). Odabire se posao na kojem je preostalo najmanje rada da bi ga se dovršilo, računajući sve operacije
- FCFS – (*engl. first come first served*). Odabire se posao koji obradu na odabranom stroju čeka najduže
- FISFS – (*engl. first in system first served*). Odabire se posao koji se najduže nalazi u radionici
- MOR – (*engl. most number of operations remaining*). Odabire se posao koji ima najviše neobavljenih operacija
- LOR – (*engl. least number of operation remaining*). Odabire se posao koji ima najmanje neobavljenih operacija
- LSK - (*engl. least slack remaining*). Odabire se posao koji ima najmanju mogućnost klizanja
- EDD – (*engl. earliest due date*)- Odabire se posao koji ima najranije predviđeno vrijeme završetka
- RND – (*engl. random*). Posao se odabire nasumično.

Kao i u raspoređivanju na jednom stroju, različita pravila daju bolje odnosno lošije rezultate za različite funkcije cilja. Tako primjerice LWR ima tendenciju stvaranja rasporeda u kojima dugački poslovi kasne, što daje loše rezultate ako je funkcija cilja zasnovana na kašnjenju ili zaostajanju. Analizu pojedinih pravila proveli su već spomenuti Panwalker i Iskander te Pidado [9] 1995. i Sule [17] 1997. godine. Njihova analiza pokazala je da ne postoji mogućnost

generalizacije odnosno da nije moguće, osim za jednostavne klase problema, odrediti idealno pravilo za pojedine funkcije cilja. Imajući to na umu, u ovom doktorskom radu analizirana je mogućnost korištenja genetičkog algoritma s nišama kao metoda rješavanja problema JSSP za koji ne postoji algoritam koji daje optimalno rješenje u razumnu vremenu.

3. UVOD U GENETIČKE ALGORITME

Genetički je algoritam heuristička metoda optimiranja inspirirana evolucijom te funkcionira tako da zapisuje potencijalna rješenja u obliku strukture vrlo slične kromosomu. Iako je proces pretraživanja prostora rješenja stohastičan, istodobno je i usmjeren jer se iz generacije u generaciju prenose informacije o prostoru koji se pretražuje. Algoritam se svodi na stvaranje inicijalne generacije koja se obično kreira nasumičnim odabirom brojeva te se jedinkama koje su bolje prilagođene rješenju koje se istražuje dodijeli veća mogućnost reprodukcije. Danas postoje dvije definicije genetičkih algoritama. Uža definicija, koja se i češće rabi, zasniva se na razmatranjima Johna Hollanda i njegovih učenika. S druge strane, postoji i šira definicija koja obuhvaća sve algoritme koji uključuju populaciju, selekciju i rekombinaciju kao metodu kreiranja novih rješenja [18]. Iako se genetički algoritmi vezuju gotovo isključivo uz problem optimiranja funkcije, mogućnost njihove primjene iznimno je široka. Razlog je njihove široke primjene u njihovoj robusnosti te mogućnosti direktne primjene na velikom setu problema uz minimalne prilagodbe algoritma. Postoji velik broj različitih genetičkih algoritama, ali svi oni korijene vuku iz jednostavnoga genetičkog algoritma (*engl. simple genetic algorithm*).

Proceduru rada jednostavnog genetičkog algoritma najjednostavnije je definirati putem pseudokoda:

```
Generiranje inicijalne populacije;
Određivanje vrijednosti jedinki nove generacije (funkcija cilja);
Ponavljanje dok nije zadovoljen uvjet prekida algoritma;
    Odabir jedinki za reprodukciju;
    Kreiranje potomaka uporabom operatora križanja;
    Eventualno mutiranje pojedinih potomaka;
    Određivanje vrijednosti jedinki nove generacije (funkcija cilja);
Kraj ponavljanja;
```

Prije iscrpna objašnjenja rada jednostavnoga genetičkog algoritma potrebno je definirati neke pojmove. Prije svega to su populacija, generacija, kromosom, gen i funkcija cilja. Genetički algoritmi rade s većim brojem rješenja (jedinki) istodobno, koje se zajedno nazivaju populacija. Svaka jedinka u populaciji ima dva prikaza, genotipski i fenotipski. Fenotipski je prikaz uobičajeniji u stvarnom životu i to je ono što prvo vidimo kad sretnemo npr. nekog čovjeka. Već na

prvi pogled može se uočiti koju boju očiju promatrani čovjek ima. Promatrana boja očiju posljedica je gena koje čovjek nosi. Ukupni skup gena koje promatrani čovjek nosi naziva se genotip ili genotipski prikaz i direktno je odgovoran za sve osobine promatranog čovjeka-od boje očiju i kose pa sve od visine i očekivana trajanja života. Iz navedene rasprave jednostavno je zaključiti da je boja očiju fenotipski prikaz genotipa koji se ne vidi.

Podjednaki princip vrijedi i za bilo koji problem koji se pokušava riješiti uporabom genetičkoga algoritma. Naime, svaki problem ima određen broj svojstava koji ga jedinstveno definiraju. Upravo ta svojstva treba zapisati u obliku gena koji zajedno čine kromosom. Ako je problem jednostavan, odnosno ima mali broj svojstava, broj gena također će biti malen, a time i kromosom kratak. Svi kromosomi odnosno jedinke zajedno čine populaciju nad kojom se provodi genetički algoritam.

Genetički algoritmi započinju fazom generiranja inicijalne populacije. Inicijalna populacija (nulta generacija) obično se sastoji od nasumce generiranih jedinki te je iznimno malena vjerojatnost da je u njoj optimalno rješenje. Da bi se odredilo koje su jedinke bolje a koje lošije potrebno je svaku jedinku podvrgnuti provjeri prema funkciji cilja. Ovo je jedino mjesto u genetičkom algoritmu na kojem je potrebno jedinke koje su zapisane u obliku kromosoma (genotip) pretvoriti u fenotipski oblik kako bi im se odredila stvarna vrijednost. Određivanje vrijednosti pojedinih jedinki (rješenja) iznimno je bitno jer će se na osnovi te vrijednosti provoditi selekcija roditelja koji će stvoriti novu generaciju. Poznato je iz Darwinove teorije da u prirodi preživljavaju samo jedinke koje su bolje prilagođene okolišu te tako prenose svoje gene na novu generaciju koja bi trebala biti još bolje prilagođena okolišu. Istovjetan princip rabe genetički algoritmi kada traže optimalno rješenje ili najprilagođeniju jedinku. Na osnovi vrijednosti jedinke (prikaz prilagođenosti okolišu, odnosno rješenju problema) provodi se selekcija budućih roditelja. Jedinke s većom vrijednošću funkcije cilja imaju proporcionalno veće šanse postati roditelji. Razni istraživači predložili su različite metode selekcije koje su se pokazale više ili manje uspješne. Kako pojedine metode funkcioniraju odnosno koliko su uspješne bit će pobliže opisano u poglavlju 3.4.

Sljedeći je korak u jednostavnom genetičkom algoritmu stvaranje potomaka uporabom operatora križanja. Operator križanja služi za miješanje genetskoga materijala dvaju roditelja te kreiranje nove jedinke. Neki operatori križanja odmah stvaraju dva potomka, a drugi stvaraju samo jednog potomka pri svakom križanju. Detaljna objašnjenja rada pojedinih operatora križanja dana su u poglavlju 3.5.2. Kako je genetski materijal upotrijebljen za kreiranje nove

jedinke došao od najboljih roditelja, pretpostavlja se da će i potomci nastali ovim putem imati jednake, ako ne i bolje osobine od svojih roditelja. Ovdje treba istaknuti da katkad nastanu i jedinke koje su lošije od svojih roditelja (lošije su prilagođene okolišu), ali to nije problem jer će prema principima genetičkoga algoritma imati relativno male šanse da postanu roditelji i prenesu svoj relativno loš genetski materijal na novu generaciju.

Posljednji korak u kreiranju nove generacije uporaba je operatora mutiranja. Naime, u prirodi obično postoji iznimno velik broj jedinki određene vrste (osim ako nije riječ o ugroženoj vrsti) koje mogu miješati svoj genetski materijal. S genetičkim algoritmom to nije tako. Zbog velikog računalnog opterećenja genetički algoritmi obično rade s relativno malim brojem jedinki unutar populacije. Posljedica je relativno malena genetska raznolikost, što u konačnici može dovesti do nemogućnosti pronalaženja globalnog ekstrema. Da bi se taj problem riješio, uveden je operator mutacije koji povremeno izmijeni pojedine gene tako da novonastali potomak minimalno odstupa od svojih roditelja, što omogućuje pretraživanje većeg prostora rješenja. Istovjetan je princip prisutan u prirodi. Pri kopiranju gena s roditelja na potomke povremeno nastaju pogreške te se pojavljuje osobina koja ne postoji ni u jednog od roditelja. Ako se pokaže da je nova osobina dobra, odnosno da jedinki koja ju posjeduje daje određenu prednost u prilagodbi okolišu, velika je vjerojatnost da će se ta osobina prenijeti na novu generaciju. Tako se pojedine vrste prilagođavaju okolišu te nastaju nove vrste. Broj pogrešaka pri kopiranju iznimno je malen te su zbog toga evolucijske promjene iznimno spore. Tako malena vjerojatnost mutacije nije poželjna kod genetičkih algoritama jer se u njih očekuje znatno brže pronalaženje rješenja, odnosno prilagođavanje okolišu. Obično se za operator mutacije uzima vjerojatnost do oko 0,01. Veće vrijednosti mutacije pretvorit će genetički algoritam u nasumično pretraživanje prostora rješenja, što svakako nije poželjno.

Kad je stvorena nova generacija, postupak se ponavlja dok se ne zadovolji uvjet prekida algoritma. Uvjeti prekida mogu biti različiti-od zadanog broja generacija pa do smanjenja genetske raznolikosti ispod zadane granice ili nemogućnosti pronalaženja boljeg rješenja (jedinke) kroz zadani broj generacija.

3.1. NOTACIJA GENETIČKOG ALGORITMA

Svaki genetički algoritam sastoji se od osam bitnih dijelova [19]

$$GA = (I, \Phi, \Omega, \Psi, s, T, \mu, \lambda, I) \tag{27}$$

Navedena će notacija poslužiti tijekom cijelog rada, a značenja su pojedinih elemenata ovakva:

- I – individualni članovi populacije
- $\Phi: I \rightarrow IR$ – funkcija cilja kojom se pojedinim članovima populacije pridružuje realna vrijednost
- $\Omega = \left\{m_{\{p_m\}}: I^\mu \rightarrow I^\mu,\ r_{\{p_c,z\}}: I^\mu \rightarrow I^\mu, r_{\{p_c\}}: I^\mu \rightarrow I^\mu\right\}$ – genetički operatori
- $\Psi(P) = s(m_{\{p_m\}}(\ r_{\{p_c,z\}}(P))$ - tranzicijska funkcija koja opisuje proces prelaska iz jedne generacije u drugu. Iz funkcije je vidljivo da se prvo primjenjuje operator križanja te poslije njega operator mutacije
- $s_{\Theta_s}: I^\lambda \rightarrow I^\mu$ - operator selekcije koji broj članova populacije mijenja iz λ u μ. U najčešćoj varijanti λ i μ su jednaki. Θ_s predstavlja skup parametara koji definiraju proces selekcije
- $T: I^\mu \rightarrow \{true, false\}$ - kriterij prekida algoritma
- μ - broj roditelja
- λ – broj potomaka.

3.2. SHEMA-TEOREM I HIPOTEZA GRAĐEVNIH BLOKOVA

Genetički algoritmi opisuju stohastički proces i nije ih jednostavno matematički formulirati. Prikladnu formulaciju rada genetičkih algoritama dao je Holland kada je uveo tezu da genetički algoritmi procesuiraju sheme (podskupove sličnih jedinki čija se sličnost ogleda u jednakim vrijednostima gena na pojedinim pozicijama unutar kromosoma).

Prilikom definicije shema-teorema poslužit će genetički algoritam s binarnim prikazom kako ga je inicijalno definirao Holland. Za razliku od pojedinih jedinki $\vec{a} \in \{0,1\}$ unutar populacije, shema je definirana kao zapis dužine l, čiji se alfabet sastoji od tri znamenke $\{0,1,*\}$, gdje znamenka * predstavlja bilo koji znak (u slučaju binarnog zapisa 0 ili 1). Za jedinku $\vec{a} \in IB^l$ kažemo da je instanca sheme $H \in \{0,1,*\}$ ako i samo ako su fiksne pozicije (0,1) u shemi H identične odgovarajućim pozicijama u jedinki $\vec{a}$.

Holland je definirao i dva kriterija koji omogućuju ispitivanje utjecaja genetičkih operatora selekcije, križanja i mutacije na shemu. Ako je primjerice definirana shema H=(10**1), tada se jednostavno može odrediti set od četiri instance navedene sheme.

$$I(H) = \{(10001), (10011), (10101), (10111)\} \tag{28}$$

Prvo je svojstvo red sheme $o(H)$ koji predstavlja broj fiksnih pozicija. Shema iz primjera ima tri fiksne pozicije te joj je $o(H) = 3$

$$o(H) = \left|\{i|h_i \in \{0,1\}\}\right| \tag{29}$$

Osim broja fiksnih pozicija od osobitog značaja je i njihova međusobna maksimalna udaljenost. Svojstvo koje definira maksimalnu udaljenost između fiksnih pozicija naziva se definirana dužina $\Delta(H)$.

$$\Delta(H) = max\{i|h_i \in \{0,1\}\} - min\{i|h_i \in \{0,1\}\} \tag{30}$$

Za navedeni primjer ona iznos $\Delta(10 ** 1) = 5 - 1 = 4$

Navedena svojstva su iznimno bitna za određivanje vjerojatnosti preživljavanja pojedine sheme nakon primjene genetičkih operatora selekcije, križanja i mutacije.

3.2.1. Operator selekcije

U originalnom radu Holland [1] je koristio proporcionalnu selekciju, te će njezin utjecaj na preživljavanje sheme biti prvi istražen.

$$p_s\left(\overrightarrow{a_i}(t)\right) = \frac{\Phi(\overrightarrow{a_i}(t))}{\sum_{j=1}^{\mu} \Phi(\overrightarrow{a_i}(t))} \tag{31}$$

Nakon primjene proporcionalne selekcije u sljedećoj generaciji se može očekivati $N(H(t+1))$ shema

$$N\big(H(t+1)\big) = N\big(H(t)\big)\frac{\Phi(H(t))}{\frac{1}{\mu}\sum_{i=1}^{\mu} \Phi(\overrightarrow{a_i}(t))} \tag{32}$$

s time da je

$$\Phi\big(H(t)\big) = \frac{1}{N\big(H(t)\big)} \sum_{\vec{a}(t)\in I\left(H(t)\right)\cap P(t)} \Phi(\vec{a}\,(t)) \tag{33}$$

prosječna vrijednost sheme svih članova populacije u trenutku t koji su ujedno instanca sheme $H(t)$. Ukoliko pretpostavimo da je $H(t)$ shema s natprosječnom vrijednošću funkcije cilja, tada se njezina vrijednost može zapisati kao

$$\Phi\big(H(t)\big) = \overline{\Phi}(t) + \ c \cdot \overline{\Phi}(t) \tag{34}$$

Uz uvjet da je $c > 0$ tada je očekivani broj shema

$$N\big(H(t+1)\big) = N\big(H(t)\big)\frac{\overline{\Phi}(t) + c \cdot \overline{\Phi}(\mathrm{t})}{\overline{\Phi}(t)} = (1+c) \cdot N(H(t)) \tag{35}$$

Iz navedenog proizlazi da je broj shema u prvoj generaciji

$$N\big(H(1)\big) = (1+c) \cdot N\big(H(0)\big) \tag{36}$$

U drugoj generaciji

$$\begin{aligned} N\big(H(2)\big) = (1+c) \cdot N\big(H(1)\big) = (1+c) \cdot (1+c) \cdot N\big(H(0)\big) = \\ = (1+c)^2 \cdot N(H(0)) \end{aligned} \tag{37}$$

Generalizirano

$$N\big(H(t)\big) = N\big(H(0)\big) \cdot (1+c)^t \tag{38}$$

3.2.2. Operator križanja

Vjerojatnost preživljavanja pojedine sheme nakon primjene operatora križanja, pri čemu se kromosom presijeca na samo jednom mjestu, uvelike ovisi o definiranoj dužini $\Delta(H)$ kromosoma. Kromosomi koji imaju veću definiranu dužinu imaju manju mogućnost preživljavanja jer je vjerojatnost da se mjesto presijecanja kromosoma nalazi unutar sheme iznimno velika. Ilustrirat ćemo to jednim primjerom.

Član populacije $\overrightarrow{a_1} = \{1,0,0,1,0,1,1,1\}$ u sebi sadrži dvije sheme jednakog ranga, ali različite definirane dužine H_1={*,0,*,*,*,*,*,1} i H_2={*,*,*,*,0,1,*,*}. Prva shema ima rang $o(H_1) = 2$ i definiranu dužinu $\Delta(H_1) = 8 - 2 = 6$, a rang je druge sheme također $o(H_2) = 2$, ali je definirana dužina značajno manja $\Delta(H_1) = 6 - 5 = 1$.

Primjenom operatora križanja član populacije može se presjeći na l – 1 = 8 –1 = 7 mjesta. Vjerojatnost presijecanja prve sheme iznosi:

$$p_d = \frac{\Delta(H_1)}{l-1} \cdot \left(1 - \frac{N(H_1(t))}{\mu}\right) \tag{39}$$

gdje $N\big(H_1(t)\big) = |I(H_1(t)) \cap P(t)|$ predstavlja broj instanci sheme H_1u populaciji. Ako je shema H_1 u oba roditelja, tada neće biti njezine destrukcije. Stoga se može zaključiti da će prekid nastati samo u

$$1 - \frac{N\big(H_1(t)\big)}{\mu} \tag{40}$$

članova populacije. Prema Goldbergu, taj se dio može zanemariti te se vjerojatnost destrukcije sheme može prikazati kao:

$$p_{d1} = \frac{\Delta(H_1)}{l-1} = \frac{7}{9-1} = \frac{7}{8} \tag{41}$$

Na isti se način može odrediti vjerojatnost destrukcije druge sheme

$$p_{d2} = \frac{\Delta(H_2)}{l-1} = \frac{1}{9-1} = \frac{1}{8} \tag{42}$$

Vjerojatnost je opstanka sheme nakon primjene operatora križanja

$$p_p \geq 1 - p_c \cdot \frac{\Delta(H_1)}{l-1} \tag{43}$$

s time da je p_c vjerojatnost uporabe operatora križanja.

Nakon definiranja utjecaja operatora križanja na vjerojatnost opstanka sheme jednostavno se može odrediti kombinirani učinak operatora selekcije i križanja na očekivani broj shema u sljedećoj generaciji.

$$N\big(H(t+1)\big) \geq N(H(t)) \cdot \frac{\Phi(H(t))}{\overline{\Phi}(\mathrm{t})} \cdot \left(1 - p_c \cdot \frac{\Delta(H(t))}{l-1}\right) \tag{44}$$

3.2.3. Operator mutacije

Operator mutacije najjednostavniji je operator jer nasumično mijenja pojedine gene u kromosomu s vjerojatnošću p_m. Da bi shema opstala uz primjenu operatora mutacije nužno je da sve fiksne pozicije prežive. Svaka fiksna pozicija preživljava s vjerojatnošću $1 - p_m$. Kako su svi geni nezavisni, vjerojatnost je preživljavanja cijele sheme

$$p_p = (1 - p_m)^{o(H(t))} \tag{45}$$

Uz veoma male vrijednosti $p_m \ll 1$ jednadžba poprima jednostavniji oblik

$$p_p = 1 - o(H(t)) \cdot p_m \tag{46}$$

Kombinirani učinak sva tri operatora (selekcija, križanje, mutacija) može se opisati jednadžbom koju je predložio Goldberg

$$N\big(H(t+1)\big) \geq N(H(t)) \cdot \frac{\Phi(H(t))}{\overline{\Phi}(\mathrm{t})} \cdot \cdot \left(1 - p_c \cdot \frac{\Delta(H(t))}{l-1} - o(H(t)) \cdot p_m\right) \tag{47}$$

Goldbergova jednadžba je nešto jednostavnija u odnosu na originalnu Hollandovu jednadžbu

$$N\big(H(t+1)\big) \geq N(H(t)) \cdot \frac{\Phi(H(t))}{\overline{\Phi}(\mathrm{t})} \cdot \cdot \left(1 - p_c \cdot \frac{\Delta(H(t))}{l-1} \cdot \left(1 - \frac{N(H_1(t))}{\mu}\right)\right) \cdot (1 - p_m)^{o(H(t))} \tag{48}$$

3.2.4. Hipoteza građevnih blokova

Građevni su blokovi sheme s natprosječnom vrijednošću funkcije cilja i kratkom definiranom dužinom (implicitno to znači i niskim rangom). Prema shema-teoremu takve sheme dobivaju sve veći broj sudjelovanja u stvaranju sljedeće generacije. Broj pokušaja raste s eksponencijalnom progresijom kako genetički algoritam napreduje. Na taj način građevni blokovi korak po korak dolaze do optimalnog rješenja ili rješenja blizu optimalnog. Na žalost, ne postoje čvrsti dokazi da je tome tako pa se ta teorija naziva hipoteza građevnih blokova. Dapače, postoje primjeri koji zorno pokazuju da hipoteza građevnih blokova može navesti genetički algoritam na krivi trag.

3.3. PRIKAZ RJEŠENJA (KROMOSOMA)

Poznato je više načina prikaza rješenja genetičkoga algoritma, poput prikaza s binarnim brojevima, cijelim brojevima, brojevima s pomičnim zarezom itd. U gotovo svim teorijskim razmatranjima upotrebljava se binarni prikaz kromosoma (rješenja) jer je on primijenjen na samom početku razvoja genetičkih algoritama te se pokazao prikladnim za najveći broj problema. Kasnije su se pojavili prikazi rješenja koji su bili specijalizirani za određenu vrstu problema

(npr. za permutacijski problem) jer su omogućivali kreiranje ispravnih rješenja. Takav način prikaza povlači i razvoj specijalnih operatora križanja i mutacije koji neće ugroziti ispravnost rješenja.

3.3.1. Binarni prikaz rješenja

Genetički algoritmi rade s binarnim brojevima dužine l

$$I \in IB^l \tag{49}$$

te služe u rješavanju kontinuiranih problema oblika:

$$f: \prod_{i=1}^{n} [u_i, v_i] \rightarrow IR \tag{50}$$

uz uvjet da je $u_i < v_i$

Genetički algoritmi povezuju realne vektore problema:

$$\vec{x} \in \prod_{i=1}^{n} [u_i, v_i] \tag{51}$$

s njihovom binarnom reprezentacijom u genetičkom algoritmu:

$$\vec{a} = (a_1, \dots, a_l) \in IB^l \tag{52}$$

Da bi se veza uspostavila potrebno je binarni zapis podijeliti na n segmenata. U pravilu su ti segmenti jednaki $l = n \cdot l_x$ ali to nije uvjet. Dekodiranje binarnog rješenja u realno obavlja se tako da se pojedini dijelovi binarnog rješenja pretvaraju u realne brojeve u rasponu $[u_i, v_i]$ uporabom odgovarajuće funkcije.

$$\begin{gathered} F^i: IB^{l_x} \rightarrow [u_i, v_i] \\ F^i(a_{i1}, \dots, a_{il_x}) \end{gathered} \tag{53}$$

Primjerice, funkcija za pretvaranje binarnih brojeva u decimalne brojeve glasi:

$$F^i(a_{i1}, \dots, a_{il_x}) = u_i + \frac{v_i - u_i}{2^{l_x} - 1} \cdot \left(\sum_{j=0}^{l_x - 1} a_{i(l_x - j)} \cdot 2^j \right) \tag{54}$$

Iz navedene funkcije može se vidjeti da kvaliteta mapiranja ovisi o veličini l_x. Što je on veći, rješenje genetičkog algoritma može biti preciznije jer se prostor pretraživanja dijeli na manje dijelove. Naime, prostor pretraživanja podijeljen je u obliku mreže čiji su razmaci jednaki

$$\Delta x_i = \frac{v_i - u_i}{2^{l_x} - 1} \tag{55}$$

Razmak mreže pokazuje kolika je maksimalna preciznost genetičkoga algoritma. Naime, genetički algoritam kao rješenje daje jedan od čvorova mreže što nipošto ne znači da je riječ o globalnom optimumu. Što je razmak unutar mreže manji, moguće je bliže prići globalnom optimumu. Ako se želi smanjiti veličina mreže, nužno je povećati l_x.

Za unaprijed poznatu preciznost odnosno razmak mreže koji se želi postići, l_x može se direktno odrediti iz sljedeće jednadžbe:

$$l_x = \frac{\log\left(\frac{v_i - u_i}{\Delta x_i} + 1\right)}{log2} \tag{56}$$

Odabir l_x direktno utječe na dužinu kromosoma, što pojednostavljeno znači da podjela prostora pretraživanja na mrežu s malim razmacima zahtijeva uporabu dugačkog kromosoma.

3.3.2. Prikaz rješenja za permutacijski problem

Permutacijski problem postavlja pred genetički algoritam poseban izazov jer zbog njegove prirode postoji velika mogućnost neispravnih rješenja koja nepovoljno utječu na rad algoritma. Prikaz rješenja putem decimalnih cjelobrojnih vrijednosti najčešći je prikaz rješenja (kromosoma) u genetičkim algoritmima koja se primjenjuju za rješavanje permutacijske klase problema. Bitno je istaknuti da postoje dvije varijante rješenja navedenog problema, direktna i indirektna.

Direktan prikaz rješenja veoma je povoljan jer je transformacija iz genotipa u fenotip i obrnuto relativno jednostavna. Kako ništa samo po sebi nije idealno, i direktan prikaz rješenja ima neke iznimno negativne posljedice. Naime, uporaba klasičnih operatora mutacije i križanja gotovo u 100% slučajeva daje neispravno rješenje, a to svakako treba izbjeći. Da bi se doskočilo tom problemu mogu se primijeniti dva pristupa:

1. Prevođenje rješenja na ispravan oblik (*engl. harmonization*) – ovakav pristup omogućuje uporabu klasičnih operatora križanja i mutacije koji u pravilu daju neispravno rješenje, a koje se onda korigira odnosno prevodi na najbliže ispravno. Ovako dobiveno rješenje u pravilu ne ulazi u daljnji postupak selekcije već se selekcija provodi s originalnim rješenjem. Neki istraživači predložili su uporabu ispravnog (prevedenog) rješenja u postupku selekcije, ali istraživanja su pokazala da postupak (*engl. forcing*) bitno pridonosi smanjenju genetske raznolikosti populacije, što svakako nije poželjno
2. Korištenje specijaliziranih operatora križanja i mutacije – kako bi se izbjeglo prevođenje neispravnih rješenja na ispravan oblik razni istraživači na području genetičkih algoritama predložili su određeni broj operatora koji uvijek daju ispravna rješenja. Neki od njih biti će prikazani u poglavlju 3.5.2.

S druge strane, indirektan način prikaza ima, kao što mu i ime kaže, indirektnu vezu između fenotipa i genotipa. Prelazak iz genotipa u fenotip malo je složeniji, ali s druge stane omogućuje uporabu klasičnih operatora mutacije i križanja. Na kraju ove kratke rasprave postavlja se pitanje koja je varijanta bolja. Odgovor je relativno jednostavan. Da je jedna varijanta bolja, druga ne bi postojala. Stoga je na svakom pojedinom istraživaču da odluči koja je varijanta pogodnija za problem koji rješava. U vezi s problemom koji se rješava u ovom radu odlučili smo se za indirektan pristup, i to zbog tendencije da predloženi algoritam bude što sličniji procesima prijenosa genetskog materijala u prirodi. Kako specijalne varijante križanja i mutacija opisane u poglavlju 3.5.2. ne postoje u prirodi odluka je jasna sama po sebi.

3.3.3. Kodiranje prema referentnoj listi

Najpoznatiji je primjer indirektnog prikaza »kodiranje prema referentnoj listi« [20] te je odabran za vrstu prikaza predloženog genetičkog algoritma iz ovog rada.

Za početak potrebno je kreirati referentnu listu s brojem članova jednakim dužini kromosoma N (broj gena u kromosomu).

$$RL = \{1, \cdots, N\} \tag{57}$$

Uobičajeno je da članovi referentne liste budu poredani po veličini, ali to nije uvjet. Kodiranje započinje određivanjem indeksa prvoga elementa rješenja u referentnoj listi. Primjerice, ako je rješenje definirano kao

$$p = (5, 6, 8, 2, 4, 1, 9, 3, 7) \tag{58}$$

a referenta lista je

$$RL_0 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \tag{59}$$

lako je uočiti da se prvi element nalazi na poziciji 5 u referentnoj listi te se broj 5 upisuje u kromosom, a referentna lista smanjuje se za jedan te sada glasi

$$RL_1 = \{1, 2, 3, 4, 6, 7, 8, 9\} \tag{60}$$

Sljedeći je element broj 6, a on je u novoj referentnoj listi također na poziciji 5. Stoga se broj 5 upisuje u kromosom, a referentna lista ponovo se smanjuje za jedan element. Postupak se ponavlja dok se referentna lista u potpunosti ne isprazni. Konačan izgled kromosoma kodiranog prema referentnoj listi glasi:

$$\bar{p} = (5, 5, 6, 2, 3, 1{,}3, 1, 1) \tag{61}$$

Bitno je naglasiti da vrijednosti koje su na pojedinim pozicijama moraju zadovoljiti sljedeći kriterij

$$1 \leq \tilde{p} \leq N - i + 1 \tag{62}$$

gdje je

$$1 \leq i \leq N \tag{63}$$

Pojednostavnjeno to znači da se na prvoj poziciji mogu nalaziti sve vrijednosti u rasponu od 1 do N, a na drugoj se poziciji u kromosomu mogu nalaziti sve

vrijednosti od 1 do *N*-1 i tako redom. Iz toga slijedi da se na posljednjoj poziciji u kromosomu može nalaziti samo vrijednost 1. Kako su vrijednosti na pojedinim pozicijama u kromosomu nezavisne u odnosu na vrijednosti u drugim pozicijama, nije moguće neispravno rješenje uporabom klasičnih operatora križanja i mutacije. Upravo ova činjenica glavni je argument zbog kojeg je kodiranje prema referentnoj listi odabrano za prikaz predložen u ovom radu.

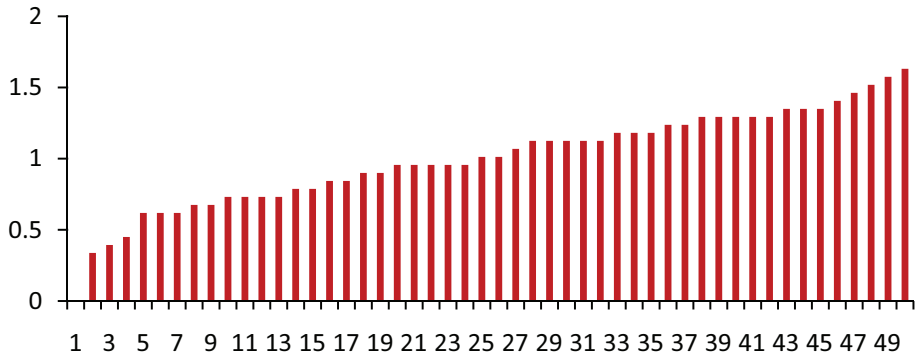
3.4. METODA SELEKCIJE

Selekcija roditelja za stvaranje nove generacije jedan je od temelja jednostavnog genetičkog algoritma. Pošto je De Jong postavio temelje genetičkih algoritama, mnogi su istraživači ispitivali mogućnosti poboljšanja metode selekcije. Da bi se moglo realno odrediti koja je metoda selekcije bolja, potrebno je postaviti niz mjerljivih parametara te precizno naznačiti što je metoda selekcije. Selekcija je proces kojim se određuje koliko će puta pojedina jedinka postati roditelj, što izravno utječe na broj potomaka na koje će prenijeti svoje gene. Proces se može podijeliti na dva dijela:

- određivanje broja pokušaja postajanja roditeljem koje određena jedinka može očekivati
- pretvaranje potencijalnog broja pokušaja u stvaran broj potomaka.

Određivanje broja pokušaja postajanja roditeljem pretvaranje je vrijednosti jedinke, dobivene primjenom funkcije cilja, u očekivani broj sudjelovanja u stvaranju sljedeće generacije. Drugi se dio odnosi na pretvaranje decimalnog broja dobivenog u prvom koraku u cjelobrojni broj, koji se odnosi na stvaran broj potomaka.

Broj potencijalnih potomaka rijetko je cjelobrojna vrijednost. Da je tome tako, najbolje pokazuje slika 3.4.1. na kojoj se zorno vide vrijednosti broja potencijalnih potomaka za svakog pojedinog člana populacije. Također je iz slike vidljivo da se za većinu članova populacije broj potencijalnih potomaka kreće između 0,5 i 1,5 (grafički se prikaz odnosi na stohastički odabir sa zamjenom (*engl. Stochastic Sampling with Replacement*) primijenjen na benchmark problemu TM6 s 50 članova u populaciji).



Slika 3.4.1. Broj potencijalnih potomaka (SSR)

Mjerila koja služe za definiranje vrijednosti metode selekcije definirao je Baker [21] 1987. godine. Baker je definirao tri parametra:

- Odstupanje (*engl. bias*) – apsolutna razlika između stvarne i očekivane vjerojatnosti odabira pojedine jedinke. Optimalna je vrijednost odstupanja nula i to znači da nema razlike između očekivanog broja sudjelovanja u reprodukciji (prvi korak selekcije) i broja potomaka (drugi korak selekcije).
- Raspon (*engl. spread*) – raspon mogućih pokušaja koje jedinka može ostvariti. Postoji gornja i donja granica. Donja granica predstavlja teorijski najmanji broj pokušaja, a gornja granica predstavlja teorijski najveći broj pokušaja sudjelovanja u stvaranju nove generacije. Stvarni broj pokušaja mora se nalaziti unutar definiranog raspona. Poželjno je postići što manji raspon.
- Složenost (*engl. complexity*) – pojedine metode selekcije iznimno su složene te zahtijevaju veliko procesorsko vrijeme. Bolja je ona metoda koja ne pridonosi složenosti genetičkog algoritma.

Danas je poznat veći broj metoda selekcije. U ovom radu ćemo prikazati sedam najzastupljenijih te navesti njihove prednosti i nedostatke.

- Stohastički odabir sa zamjenom (*engl. Stochastic sampling with replacement*)
- Stohastički odabir bez zamjene (*engl. Stochastic sampling without replacement*)

- Deterministički odabir (*engl. Deterministic sampling*)
- Stohastička turnirska selekcija (*engl. Stochastic tournament selection*)
- Stohastički odabir ostatka sa zamjenom (*engl. Remainder stochastic sampling with replacement*)
- Stohastički odabir ostatka bez zamjene (*engl. Remainder stochastic sampling without replacement*)
- Stohastički univerzalni odabir (*engl. Stochastic universal sampling*)

3.4.1. Stohastički odabir sa zamjenom (SSR)

Stohastički odabir sa zamjenom (SSR) uz turnirsku selekciju najrasprostranjenija je metoda selekcije primijenjena u genetičkim algoritmima. Glavni su razlozi njene rasprostranjenost jednostavnost algoritma i to što je riječ o metodi koju je na samim počecima razvoja genetičkih algoritama predložio De Jong. Metoda je još poznata pod imenom »kotač ruleta« zbog svoje velike sličnosti s okretanjem ruletnoga kotača. Naime, svakom članu populacije dodijeljeno je područje na ruletnom kotaču proporcionalno relativnoj vrijednosti člana populacije, što znači da će bolji članovi imati veći prostor na ruletnom kotaču, a time i veće šanse da budu izabrani za roditelja. Okretanjem ruletnoga kotača biraju se jedan po jedan roditelji za stvaranje nove generacije. Očekivan broj pojavljivanja pojedinog člana među odabranim roditeljima određuje se dijeljenjem vrijednosti kromosoma sa sumom vrijednosti svih kromosoma.

$$p_s(\vec{a}) = P\{\vec{a} \in (P(t)) | \vec{a} \in P(t)\} \tag{64}$$

$$p_s(\overrightarrow{a_i}(t)) = \frac{\Phi(\overrightarrow{a_i}(t))}{\sum_{j=1}^{\mu} \Phi(\overrightarrow{a_i}(t))} \tag{65}$$

Da bi se odredio očekivan broj pojavljivanja $\eta(\overrightarrow{a_i}(t))$ pojedinog člana $\overrightarrow{a_i}(t) \in P(t)$ među odabranim roditeljima, potrebno je očekivanu vjerojatnost pomnožiti s brojem jedinki u populaciji.

$$\eta(\overrightarrow{a_i}(t)) = \mu \cdot p_s(\overrightarrow{a_i}(t)) \tag{66}$$

Primjer određivanja broja potencijalnih potomaka prikazan je na slici 3.4.1. iz prethodnog poglavlja. Iz slike se zorno vidi da je broj potencijalnih potomaka direktno proporcionalan relativnoj vrijednosti svake pojedine jedinke, što znači da je odstupanje jednako nuli. S druge strane kod SSR selekcije ne postoji garancija da će i stvarni broj jedinki biti makar približno jednak potencijalnom broju potomaka. Teorijski je moguća situacija u kojoj je jedna te ista jedinka odabrana u svakom pokušaju, a to nije poželjno. Iz navedene analize može se vidjeti da iako je odstupanje jednako nuli, raspon je iznimno velik, što upozorava na vrlo veliku stohastičku pogrešku.

3.4.1.1. Model Goldberga i Segresta

Uporaba Markovljevih lanaca u modeliranju genetičkoga algoritma pokazalo se veoma uspješnom, pogotovo u produbljivanju razumijevanja procesa selekcije, mutacije, križanja i utjecaja funkcije cilja. Model koji su Goldberg i Segrest postavili 1987. [22] godine bavio se populacijom jedinki čiji je kromosom imao samo jedan gen. Takav pristup uvelike pojednostavnjuje problem, ali istodobno daje ključne poglede na proces selekcije.

Ponešto drukčiji pristup može se pronaći u literaturi [23], gdje se umjesto jedinki čiji se kromosom sastoji od jednog gena (jedinke 0 i 1) rabe dvije grupe identičnih jedinki (jedinke A i B) čija veličina kromosoma nije bitna dok god su sve jedinke u grupi identične.

Uporabom Markovljevih lanaca može se odrediti vjerojatnost tranzicije populacije s (i) jedinki tipa A i $(N_p - i)$ jedinki tipa B u populaciju s (j) jedinki tipa A i $(N_p - j)$ jedinki tipa B.

$$i \equiv n_{A,t} \tag{67}$$

$$j \equiv n_{A,t+1} \tag{68}$$

Kod stohastičkog odabira sa zamjenom vjerojatnost odabira točno određene jedinke tipa A iz populacije od N_p jedinki iznosi

$$p_A = \frac{\Phi_A}{\sum \Phi} \tag{69}$$

gdje je Φ_A vrijednost funkcije cilja bilo koje jedinke tipa A.

Vjerojatnost da će bilo koja jedinka tipa A biti odabrana iznosi

$$p_A = \frac{n_A \Phi_A}{\sum \Phi} \tag{70}$$

gdje je n_A broj jedinki tipa A u populaciji.

Budući da u populaciji postoje samo dvije vrste jedinki, tada vrijedi:

$$N_p = n_A + n_B \tag{71}$$

iz čega proizlazi da je vjerojatnost odabira bilo koje jedinke tipa A:

$$p_A = \frac{n_A \Phi_A}{n_A \Phi_A + n_B \Phi_B} \tag{72}$$

Na identičan način određuje se vjerojatnost odabira bilo koje jedinke tipa B:

$$p_B = \frac{n_B \Phi_B}{n_A \Phi_A + n_B \Phi_B} \tag{73}$$

Ako se uvede parametar r_Φ koji predstavlja odnos funkcije cilja jedinke tipa A u odnosu na jedinku tipa B:

$$r_\Phi = \frac{\Phi_A}{\Phi_B} \tag{74}$$

vjerojatnosti p_A i p_B glase:

$$p_A = \frac{r_\Phi n_A}{r_\Phi n_A + (N_p - n_A)} \tag{75}$$

$$p_B = \frac{N_p - n_A}{r_\Phi n_A + (N - n_A)} \tag{76}$$

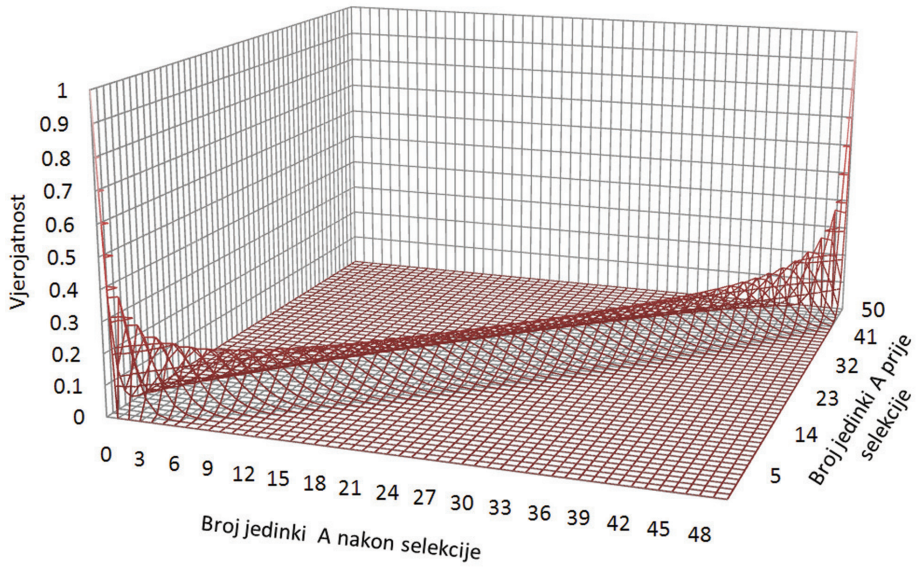
Ako se, radi jednostavnosti zapisa, zamijeni n_A sa *i,* vjerojatnost transformacije iz stanja s (i) jedinki tipa A u stanje s (j) jedinki tipa A glasi:

$$p_{trans}[i,j] = \binom{N_p}{j} (p_A)^j (p_B)^{N_p - j} \qquad (77)$$

Uvođenjem p_A i p_B dolazi se do jednadžbe za vjerojatnost tranzicije za SSR selekciju:

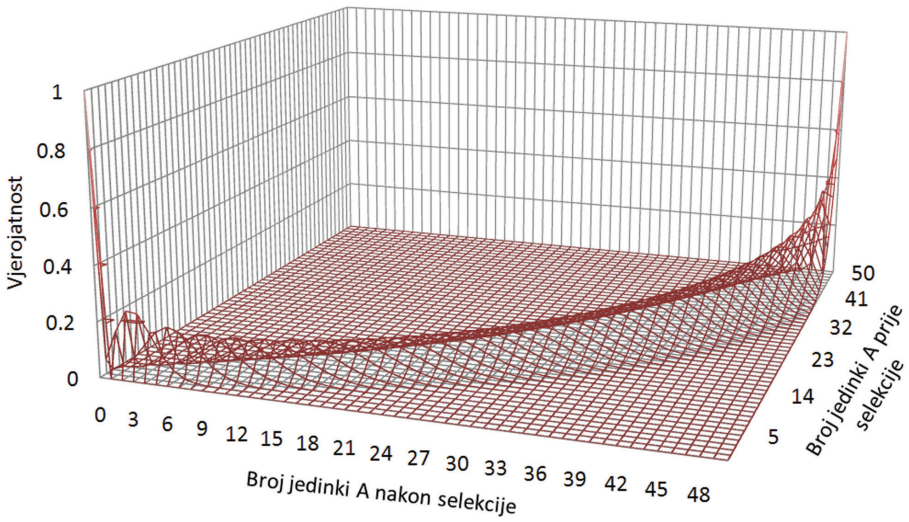
$$p_{trans} = [i,j] \binom{N_p}{j} \left(\frac{r_\Phi i}{r_\Phi i + (N_p - i)} \right)^j \left(\frac{N_p - i}{r_\Phi i + (N_p - i)} \right) \qquad (78)$$

Na slici 3.4.2. prikazan je graf tranzicije za populaciju od 50 članova i $r_\Phi = 1$.



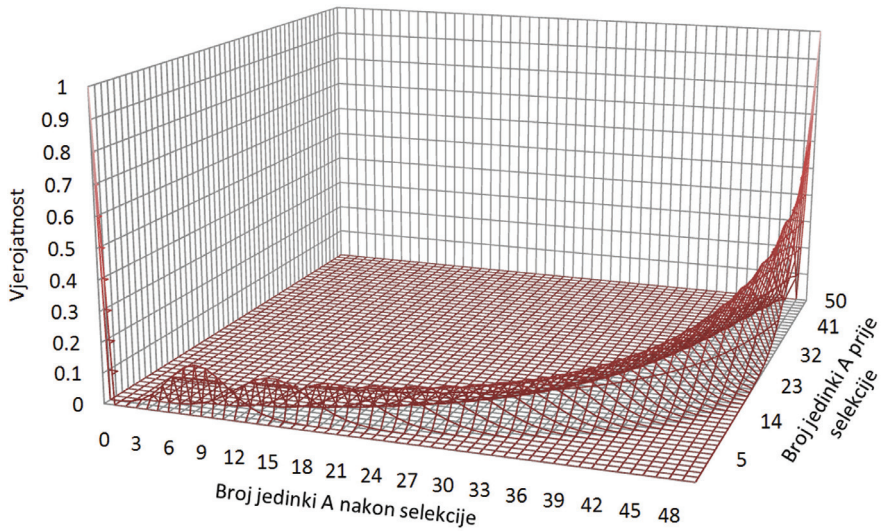
Slika 3.4.2. Stohastički odabir sa zamjenom (r=1)

S druge strane, ako je $r_\Phi = 3$, na slici 3.4.3. vrlo se jednostavno može uočiti da se graf tranzicije vidno pomiče u smjeru grupe jedinki tipa A koje imaju tri puta veću vrijednost funkcije cilja.



Slika 3.4.3. Stohastički odabir sa zamjenom (r=3)

Još ekstremnija situacija prikazana je na slici 3.4.4. gdje je odnos funkcija cilja povećan na 10 u korist grupe jedinki tipa A. Na slici 3.4.4. vrlo se dobro uočava veličina raspona, što bitno utječe na stohastičnost procesa selekcije.



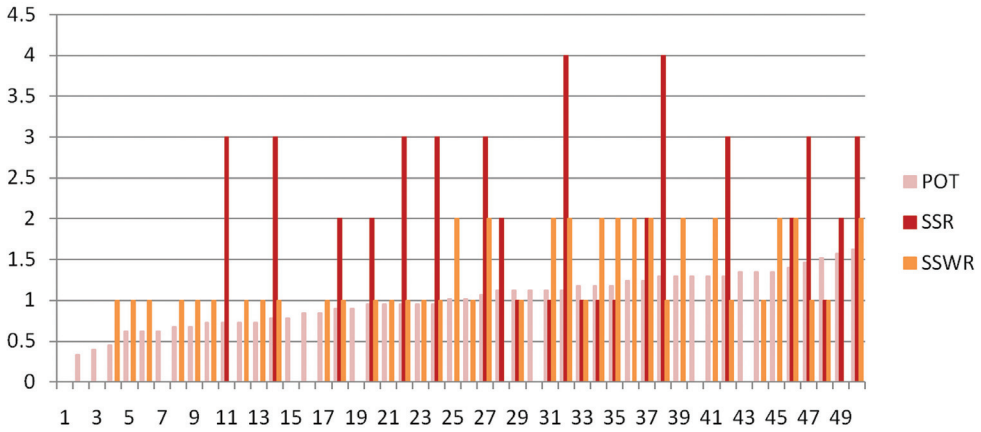
Slika 3.4.4. Stohastički odabir sa zamjenom (r=10)

3.4.2. Stohastički odabir bez zamjene (SSWR)

Problem velikog raspona pokušao se riješiti metodom stohastičkog odabira bez zamjene (SSWR), koju je razvio De Jong. Metoda SSR nema spoznaje o prethodnim odabirima, što znači da kad se jednom odabere jedinka da postane roditelj, ne znači da ona više ne može biti odabrana. Naprotiv, može biti odabrana baš svaki put. Da bi se spriječilo pretjerano često odabiranje pojedine jedinke razvijena je nova metoda: stohastički odabir bez zamjene. Za svaku jedinku izračunat je broj potencijalnih potomaka. Kad se pojedina jedinka odabere u grupu roditelja, njen broj potencijalnih potomaka smanjuje se za jedan. Kada broj potencijalnih potomaka padne ispod nule, ta jedinka više ne može biti odabrana. Prednost je ove metode bitno smanjen raspon, ali odstupanje na žalost više nije nula jer u svim odabirima osim prvog postoji razlika između relativne vrijednosti jedinke i očekivanog broja odabira.

3.4.3. Usporedba SSR i SSWR selekcije

Usporedbu dviju selekcija provest ćemo na *benchmark* problemu raspoređivanja operacija po strojevima. Primjer se sastoji od šest strojeva i šest operacija. Inicijalna se populacija sastoji od 50 nasumce odabranih jedinki. Svrha je ispitivanja provjeriti pojavljuje li se u pojedinih selekcija dominacija pojedinih članova populacije, odnosno imaju li pojedini članovi više potomaka nego što zaslužuju prema vrijednosti funkcije cilja.



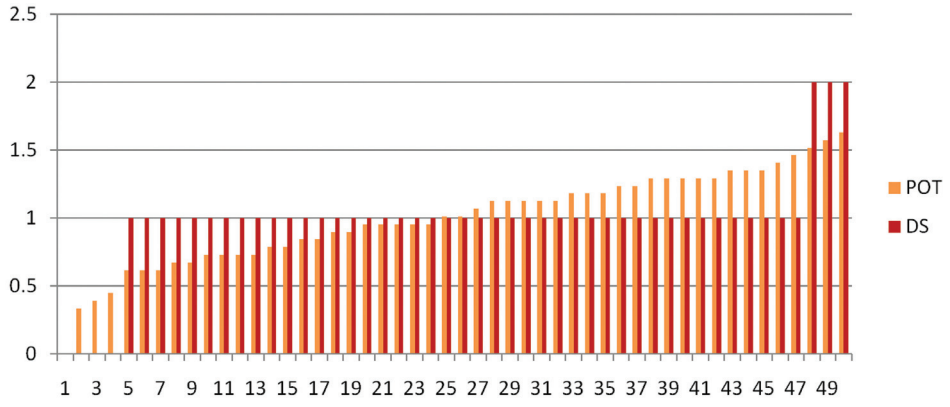
Slika 3.4.5. Usporedba SSR i SSWR

Na slici 3.4.5. članovi populacije poredani su prema vrijednosti funkcije cilja od najlošijeg prema najboljem. Oznaka POT pokazuje broj potencijalnih potomaka za svakoga člana populacije. Važno je uočiti da je riječ o vrijednostima s decimalnim zarezom koje treba pretvoriti u stvaran broj potomaka.

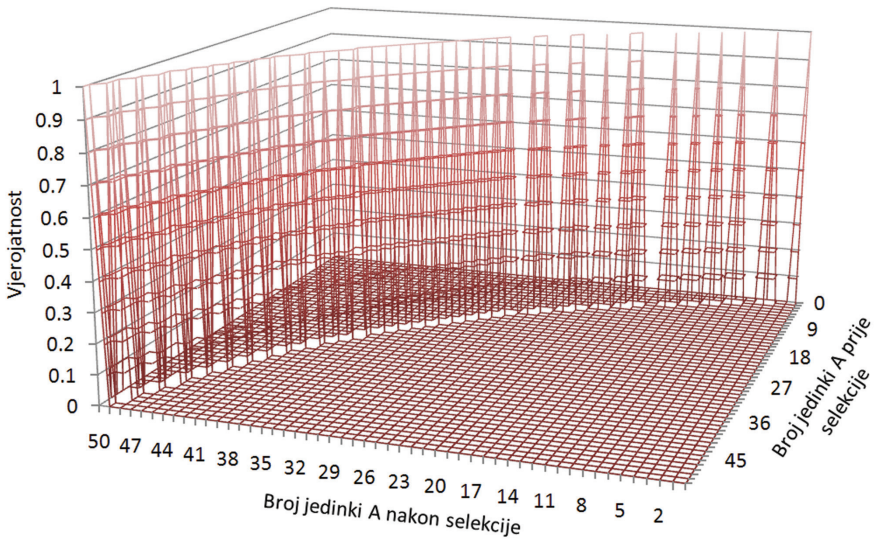
Slika na zoran način pokazuje da, primjenjuje li se metoda selekcije SSR, članovi 32 i 39 imaju nerazmjerno velik broj potomaka, što prema vrijednosti funkcije cilja nije bilo očekivano. Za njih možemo reći da dominiraju procesom selekcije. S druge strane, članovi 43, 44 i 45 nemaju ni jednog potomka iako imaju veću vrijednost funkcije cilja. Navedeni primjer na zoran način pokazuje da metoda selekcije SSR ima veliku stohastičku grešku. S druge strane, metoda selekcije SSWR omogućuje većem broju kromosoma da postanu roditelji te se tako smanjuje stohastička greška. Nadalje, iz slike 3.4.5. vidi se da ni za jednog člana populacije broj potencijalnih potomaka bitno ne odstupa od stvarnog broja potomaka, što nije tako kod metode SSR.

3.4.4. Deterministički odabir (DS)

Deterministički odabir (DS) metoda je koja je razvijena radi uklanjanja stohastičnosti iz selekcije. Naime, kad se odredi broj potencijalnih potomaka, za svaku jedinku automatski se u grupu roditelja prebacuje onoliko jedinki koliko iznosi njihova cjelobrojna vrijednost broja potencijalnih potomaka. Na primjeru navedenom u metodi SSR to znači da se među roditelje automatski svrstavaju sve jedinke od 25 do 50, jer imaju broj potencijalnih potomaka veći od jedan. U grupi roditelja ostalo je još 24 mjesta koja će biti popunjena jedinkama čiji je ostatak najveći. U konkretnom primjeru to su jedinke od 4 do 24 te jedinke 48, 49 i 50. Time je raspon bitno smanjen, ali se pojavila i određena razina odstupanja jer se u drugom krugu izbora ne odlučuje na osnovi prave vrijednosti funkcije cilja. To se najbolje vidi na primjeru najlošijih članova, koji u potpunosti gube mogućnost dolaska u grupu roditelja, što jasno upozorava na postojanje odstupanja. Iako prema Bakerovim parametrima spada u relativno dobre metode selekcije, DS se ne koristi pretjerano često. Razlog je u tome što je genetički algoritam po definiciji stohastički usmjeren proces te tu nema mjesta za determinističku varijantu selekcije.



Slika 3.4.6. Deterministički odabir



Slika 3.4.7. Deterministički odabir (r=3)

Za razliku od ostalih metoda selekcije, kod determinističkog se odabira sa stopostotnom vjerojatnošću može odrediti broj pojedinih jedinki nakon selekcije, što se zorno vidi na slici 3.4.7.

3.4.5. Stohastička turnirska selekcija (STS)

Turnirska je selekcija metoda u kojoj se odabire određen broj jedinki koje će sudjelovati u turniru. Pobjednik je turnira jedinka koja ima najveću vrijednost te se kao pobjednik automatski prebacuje u grupu roditelja. Broj jedinki odabranih za sudjelovanje u turniru kreće se od 2 pa sve do cijele populacije. Ako se veličina turnira povećava, automatski se povećava pritisak selekcije[12], što znači da se favoriziraju najbolji članovi populacije. Ako, primjerice, u turnir ulaze svi članovi populacije, pobjednik je turnira zasigurno najbolja jedinka populacije te je pritisak selekcije maksimalan. S druge strane, ako je veličina turnira malena, npr. 2 jedinke, tada se može lako dogoditi da u grupu roditelja dođu manje vrijedne jedinke jer je pritisak selekcije malen.

Za turnirsku selekciju u kojoj sudjeluju samo dva člana $t_s = 2$ i uz uvjet

$$\Phi_A < \Phi_B \tag{79}$$

vjerojatnost da jedinka tipa A postane pobjednik turnira glasi

$$p_A = \left(\frac{n_A}{N_p}\right)^2 \tag{80}$$

što znači da pobjeda u turniru može pripasti jedinki iz grupe A samo ako su obje jedinke iz te grupe. S druge strane, vjerojatnost da jedinka iz grupe B pobijedi u turniru glasi

$$p_B = 1 - \left(\frac{n_A}{N_p}\right)^2 \tag{81}$$

Ako se, radi jednostavnosti zapisa, zamijeni n_A sa i, vjerojatnost transformacije iz stanja s (i) jedinki tipa A u stanje s (j) jedinki tipa A glasi

$$p_{trans}[i,j] = \binom{N_p}{j} (p_A)^j (p_B)^{N_p - j} \tag{82}$$

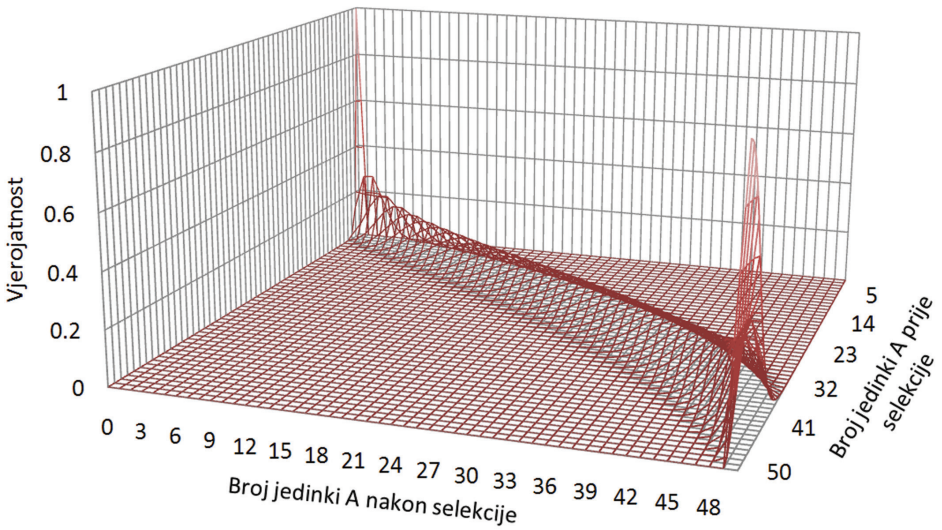
[12] Pritisak selekcije – *engl. selection pressure*

Uvrštavanjem vrijednosti za p_Ai p_B dobiva se konačna funkcija za vjerojatnost tranzicije

$$p_{trans} = [i,j] = \binom{N_p}{j}\left(\left(\frac{i}{N_p}\right)^2\right)^j\left(1-\left(\frac{i}{N_p}\right)^2\right)^{N_p-j} \tag{83}$$

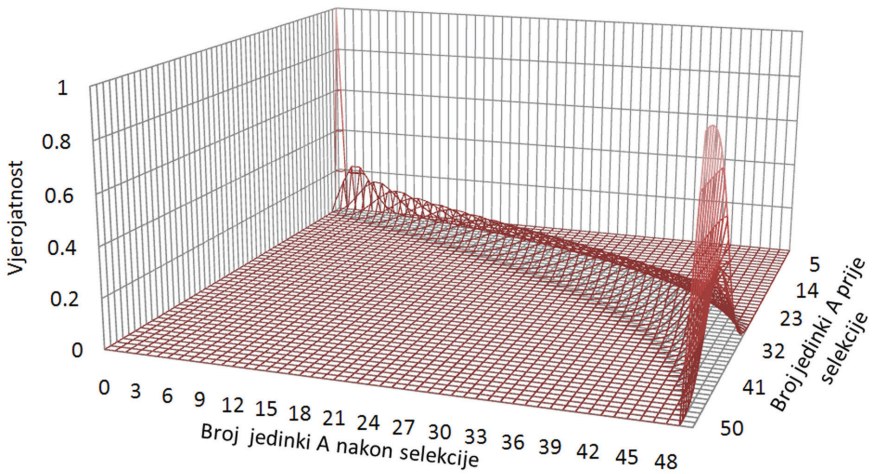
Poopćena varijanta vjerojatnosti tranzicije za svaku veličinu turnira (s)

$$p_{trans} = [i,j] = \binom{N_p}{j}\left(\left(\frac{i}{N_p}\right)^s\right)^j\left(1-\left(\frac{i}{N_p}\right)^s\right)^{N_p-j} \tag{84}$$



Slika 3.4.8. Turnirska selekcija (s=2)

Iz navedene analize vidi se da turnirska selekcija omogućuje iznimno jednostavnu prilagodbu pritiska selekcije, što je svakako vrijedna osobina, ali ima i velik raspon i odstupanje. Da bi se raspon donekle smanjio, odabir članova populacije koji će sudjelovati u turniru može se provesti prema metodi SSR, a ne nasumce. Tada se turnirska selekcija naziva stohastička turnirska selekcija.



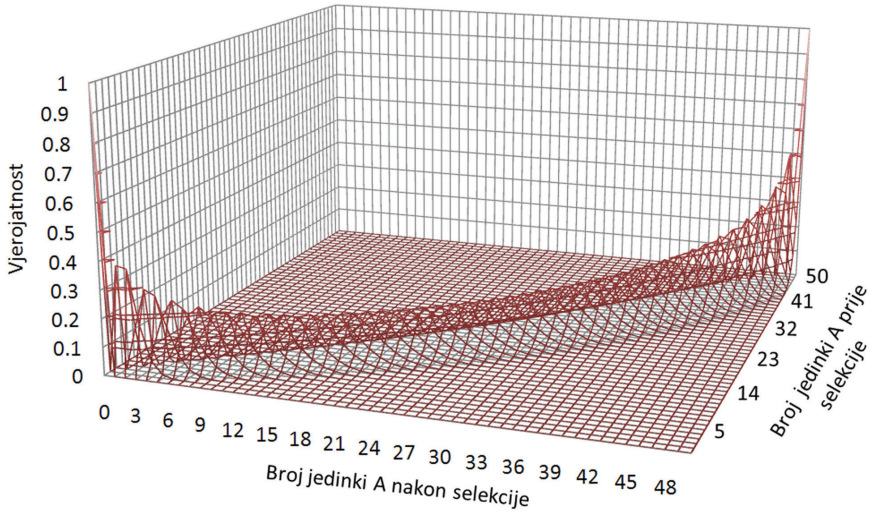
Slika 3.4.9. Turnirska selekcija (s=3)

3.4.6. Stohastički odabir ostatka

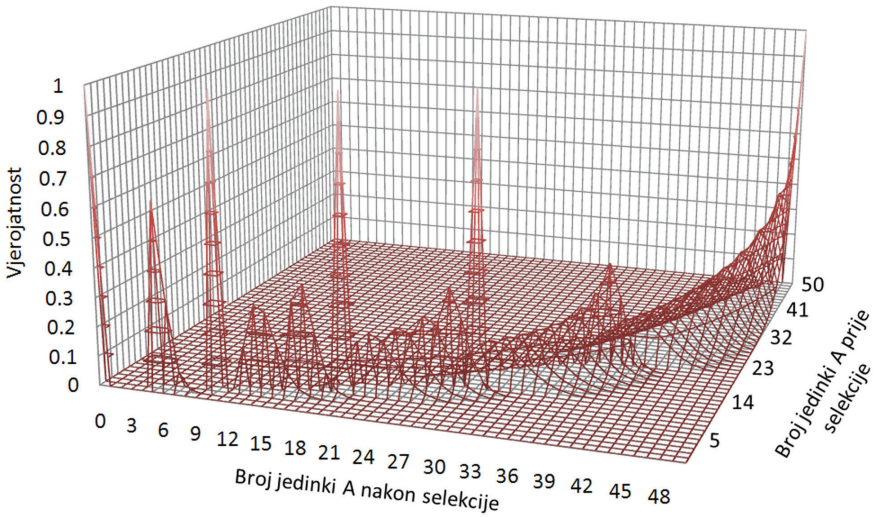
Metodu stohastičkoga odabira ostatka predložila je Brindle [24], a odvija se u dva koraka. Prvi je korak identičan prvom koraku metode determinističkog odabira (DS). Naime, prvo se u grupu roditelja prebacuje onoliko kopija pojedine jedinke koliko iznosi njihova cjelobrojna vrijednost očekivanoga broja potomaka. U drugom koraku za razliku od determinističkog odabira postoje dvije mogućnosti za popunjavanje preostalih mjesta. Prva je varijanta da se odabir preostalih jedinki obavlja prema stohastičkom odabiru sa zamjenom (SSR), s time da se očekivan broj pojavljivanja pojedine jedinke među roditeljima određuje prema ostatku, a ne prema punoj vrijednosti jedinke. Ta se metoda naziva stohastički odabir ostatka sa zamjenom (RSSR) te ima odstupanje jednako nuli, ali je raspon nešto veći, pa se može dogoditi da sva preostala mjesta zauzme jedna te ista jedinka, što svakako nije poželjno. Druga varijanta funkcionira istovjetno kao stohastički odabir bez zamjene (SSWR) jer kad pojedina jedinka bude izabrana u drugom krugu više ne može biti ponovno odabrana. Navedena varijanta naziva se stohastički odabir ostatka bez zamjene (RSSWR). Zbog nemogućnosti ponovnog izbora raspon je bitno smanjen, ali se malo povećalo odstupanje u korist jedinki s manjim ostatkom.

Na slici 3.4.10. mogu se uočiti dva apsorpcijska stanja. Populacija se nalazi u apsorpcijskom stanju ako je vjerojatnost prelaska u određeno novo stanje jednaka jedinici (100%). Pojednostavnjeno, to znači da je ishod selekcije una-

prijed poznat. Sve do sada opisane metode selekcije imale su po dva apsorpcijska stanja, i to kada se populacija sastoji samo od jedne klase jedinki. Bez obzira na odabranu selekciju svi će roditelji biti iz te klase jedinki.



Slika 3.4.10. Stohastički odabir ostatka bez zamjene (r=1)



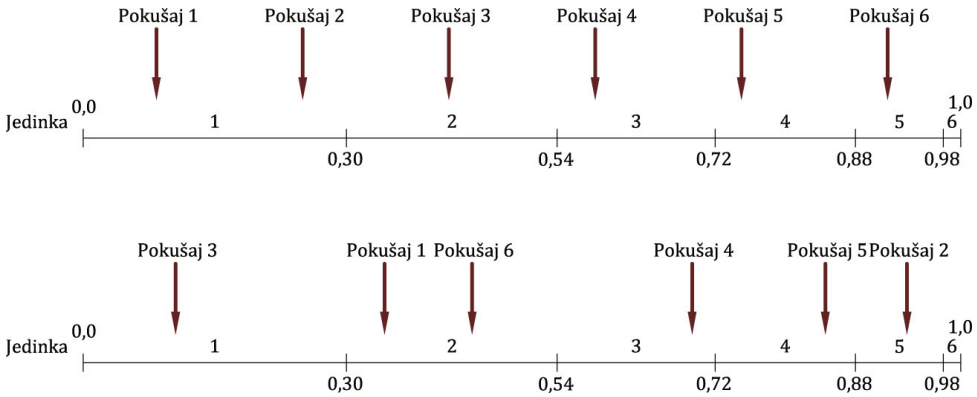
Slika 3.4.11. Stohastički odabir ostatka bez zamjene (r=6)

Stohastički odabir ostatka bez zamjene vrsta je selekcije kod koje je povećanjem odnosa funkcija cilja dviju grupa u korist grupe A moguće pojavljivanje dodatnih apsorpcijskih stanja. Konkretno, ako se r_Φ poveća na 6, pojavit će se dodatna tri apsorpcijska stanja.

Tako se iz slike 3.4.11. može vidjeti da kada se u populaciji prije selekcije nalaze 2 jedinke klase A, sa stopostotnom sigurnošću može se ustvrditi da će čak 10 jedinki klase A biti odabrano za buduće roditelje. Upravo ta stopostotna vjerojatnost upozorava na postojanje apsorpcijskog stanja.

3.4.7. Stohastički univerzalni odabir (SUS)

Stohastički univerzalni odabir (SUS) metoda je selekcije s minimalnim rasponom i odstupanjem svedenim na nulu. Metoda selekcije prvi se put spominje u djelu J. Bakera 1987. godine [21]. Proces selekcije se za razliku od stohastičkog odabira sa zamjenom odvija u jednom koraku. Jednostavno se umjesto jednog pokazivača koristi N_p pokazivača raspoređenih na jednake udaljenosti. Kao što se vidi na slici 3.4.12. potrebno je odrediti poziciju prvog pokazivača.

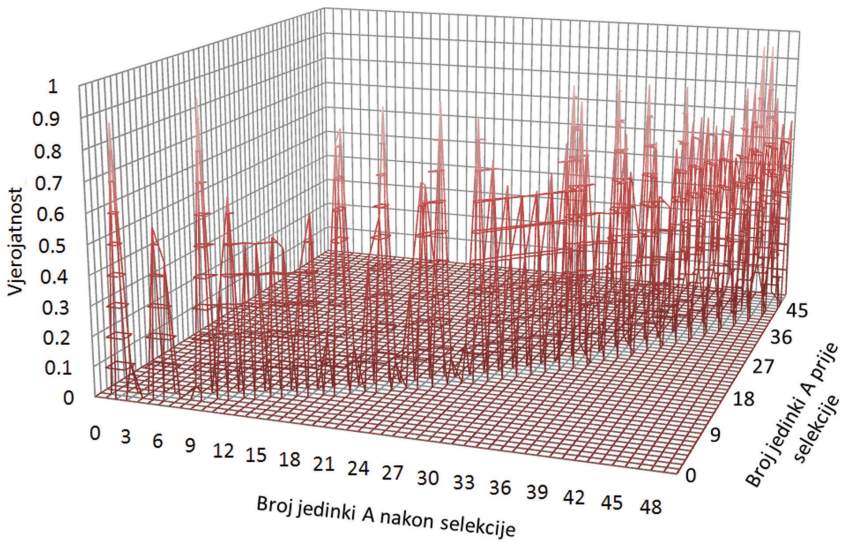


Slika 3.4.12. Stohastički univerzalni odabir (SUS)

Pozicija se određuje nasumičnim odabirom u rasponu od 0 do Sum/N_p gdje je Sum kumulativna vrijednost svih jedinki u populaciji (u konkretnom slučaju

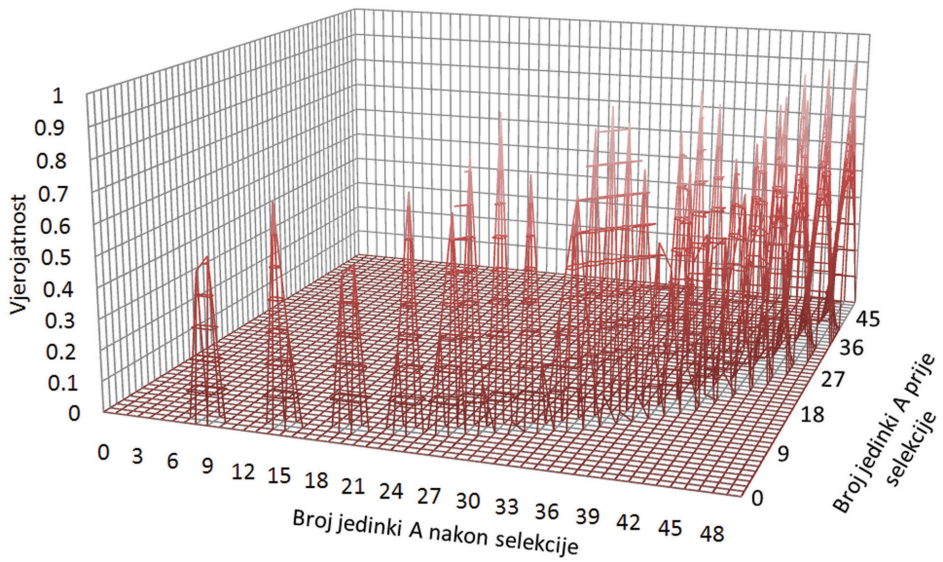
iznosi 1), a N_p broj je jedinki. Pošto je odabrana pozicija prvog pokazivača, ostali se postavljaju na udaljenosti od $1 \cdot Sum/N_p$, $2 \cdot Sum/N_p$ i tako redom počevši od prvog pokazivača. U grupu roditelja odabrane su jedinke koje pojedini pokazivač pokazuje. Tako se na slici 3.4.12. može uočiti da je jedinka (1) odabrana u grupu roditelja 2 puta, a jedinka (6) posve je izostavljena. U donjem dijelu slike prikazan je identičan primjer, ali uz uporabu stohastičkog odabira bez zamjene. Budući da se pokazivači odabiru nezavisno jedan od drugog, može se dogoditi da jedinka (1) koja ima najveću vrijednost funkcije cilja bude odabrana samo jednom, što ne odgovara njenoj vrijednosti. Slika 3.4.12. prikazuje upravo jedan takav slučaj.

Kao i za sve ostale vrste selekcije moguće je odrediti vjerojatnost tranzicije populacije od 50 članova nakon uporabe selekcije SUS. Za $r_\Phi = 3$, odnosno kada jedna grupa članova populacije ima vrijednost tri puta veću od druge, graf tranzicije prikazan je na slici 3.4.13.



Slika 3.4.13. Stohastički univerzalni odabir za r=3

Iz slike se može zorno vidjeti da je raspon znatno manji u odnosu na ostale vrste selekcija. Isto vrijedi i kad je odnos grupa jedinki mnogo nepovoljniji. Naime, na slici 3.4.14. prikazan je graf tranzicije kada je $r_\Phi = 10$.



Slika 3.4.14. Stohastički univerzalni odabir za r=10

Ovdje se vidi da bez obzira na to kolika je vrijednost neke grupe jedinki u odnosu na drugu grupu postoje samo dva tranzicijska stanja u koja populacija može prijeći nakon primjene metode selekcije SUS. Razlika je samo u vremenu koje je potrebno da vrednija grupa zauzme cijelu populaciju.

3.5. GENETIČKI OPERATORI

Genetički operatori predstavljaju bit svakog genetičkog algoritma te njihov pravilan odabir bitno utječe na sposobnost genetičkog algoritma da se približi i u konačnici pronađe globalni optimum.

Genetički operatori mogu se podijeliti u tri grupe:

- operatori inverzije
- operatori križanja
- operatori mutacije.

3.5.1. Operatori inverzije

Operator inverzije smatra se jednim od klasičnih genetičkih operatora i opisan je već u prvim radovima. S vremenom je polagano nestao i iz literature i iz istraživanja. Naime, operator funkcionira tako da nasumično odabere segment kromosoma te invertira poredak gena na odabranom segmentu. Cilj je takvog postupka dovesti gene s većom nelinearnom interakcijom bliže jedan drugome. Problem s operatorom inverzije u tome je da ako ga se primjenjuje na kromosomima koji opisuju problem na direktan način (permutirajući problem), nije ništa drugo nego masovna mutacija. S druge strane, primjena ovog operatora na kromosomima s indirektnim prikazom (kodiranje prema referentnoj listi) može uzrokovati stvaranje neispravnih rješenja jer nije zadovoljen uvjet iz jednadžbe (62). Tako primjerice, ako se odabere segment između trećeg i petog gena za provedbu operatora inverzije,

5	5	6	2	3	1	3	1	1

Slika 3.5.1. Kromosom prije primjene operatora inverzije

nastaje neispravan kromosom

5	5	3	2	6	1	3	1	1

Slika 3.5.2. Kromosom nakon primjene operatora inverzije

jer gen na poziciji 5 ne zadovoljava jednadžbu (62)

$$1 \leq \tilde{p} \leq N - i + 1 \qquad (62)$$
$$1 \leq \tilde{p} \leq 9 - 5 + 1$$
$$1 \leq \tilde{p} \leq 5$$

Naime, novonastali kromosom na poziciji 5 ima vrijednost 6, a dopuštena je vrijednost gena za tu poziciju između 1 i 5.

Stoga operator inverzije nije primijenjen u genetičkom algoritmu predloženom u ovom radu.

3.5.2. Operatori križanja

Operator križanja najvažniji je genetički operator [25], a funkcija mu je razmjena genetskog materijala između dva kromosoma. Operator kako ga ova definicija prikazuje implicira postojanje dvaju kromosoma. U ovom radu to neće biti tako, jer će se u predloženom algoritmu koristiti diploidni kromosom u odnosu na uobičajeni haploidni, što omogućuje uporabu operatora križanja u mejotičkom procesu za koji je potreban samo jedan kromosom (jedan roditelj). Radi jednostavnijeg izlaganja, operator križanja prvo će biti opisan na haploidnoj varijanti kromosoma te će se potom opis proširiti na diploidnu varijantu (poliploidnu varijantu) kromosoma.

U literaturi je opisan veliki broj operatora križanja od kojih četiri možemo smatrati tzv. klasičnim varijantama:

- operator križanja s jednom točkom prekida
- operator križanja s dvije točke prekida
- operator križanja s N točaka prekida
- uniformni operator križanja.

Navedeni operatori križanja primjenjivi su samo na kromosomima koji su prikazani na indirektan način. Osim navedenih operatora križanja razni autori [20] predložili su operatore križanja primjenjive na kromosomima prikazanim na direktan način, od kojih će šest najvažnijih biti pobliže opisano. To su redom:

- PMX [4] – *engl. Partially Mapped Crossover*
- OX [26] – *engl. Order Crossover*
- UCX [27] – *engl. Uniform Order Crossover*
- CX [28] [26] – *engl. Cycle Crossover*
- LOX [29] – *engl. Location-based Crossover*
- SXX [30] – *engl. Subsequence Exchange Crossover*

Opisi svih varijanti operatora križanja bit će prikazani na dva primjera. Prvi će se odnositi na direktan prikaz kromosoma (geni direktno predstavljaju operacije), a drugi na indirektan način prikaza (geni direktno ne predstavljaju operacije), uz napomenu da je riječ o identičnim kromosomima prikazanim na dva različita načina.

Roditelj 1	5	6	8	2	4	1	9	3	7
Roditelj 2	3	5	1	9	6	4	2	7	8

Slika 3.5.3. Direktan prikaz

Isti kromosomi (roditelji) prikazani na indirektan način uporabom kodiranja prema referentnoj listi glase:

Roditelj 1	5	5	6	2	3	2	3	1	1
Roditelj 2	3	4	1	6	3	2	1	1	1

Slika 3.5.4. Indirektan prikaz

3.5.2.1. Operator križanja s jednom točkom prekida

Operator s jednom točkom prekida najjednostavniji je operator križanja. Primjenjuje se isključivo na kromosomima prikazanim na indirektan način jer postoji velika vjerojatnost stvaranja neispravnog potomka (potomak čiji se genotipski prikaz ne može prevesti u ispravan fenotipski prikaz) ako se ovaj operator primijeni na kromosomima zapisanim na direktan način. Pri provedbi operatora križanja potrebno je odabrati mjesto prekida. Ono se odabire nasumično i može biti između bilo koja dva gena u kromosomu. U konkretnom primjeru za mjesto prekida odabrana je pozicija između drugoga i trećega gena, kao što to prikazuje slika 3.5.5.

Roditelj 1	5	5	6	2	3	2	3	1	1
Roditelj 2	3	4	1	6	3	2	1	1	1

Slika 3.5.5. Roditelji s prikazanom točkom prekida između 2. i 3. gena

Primjenom operatora križanja novonastali potomci preuzimaju gene obaju roditelja tako da od jednog roditelja preuzimaju gene prije točke prekida a od drugog nakon točke prekida. Na ovakav način dobiju se dva potomka prikazana na slici 3.5.6.

Potomak 1	5	5	1	6	3	2	1	1	1
Potomak 2	3	4	6	2	3	2	3	1	1

Slika 3.5.6. Potomci nastali primjenom operatora križanja s jednom točkom prekida

Kako je riječ o indirektnom prikazu kromosoma, oba su potomka zajamčeno ispravna.

3.5.2.2. Operator križanja s dvije točke prekida

Operator križanja s dvije točke prekida ne razlikuje se bitno od prije opisanog operatora. Razlika je samo u tome da se odabiru dvije točke prekida te potomci od jednog roditelja nasljeđuju sve gene osim onih koji se nalaze između točaka prekida koje nasljeđuju od drugog roditelja. Kao što se može vidjeti iz primjera na slici 3.5.7., odabrane su dvije točke prekida između drugog i trećeg gena i šestog i sedmog.

Roditelj 1	5	5	6	2	3	2	3	1	1
Roditelj 2	3	4	1	6	3	2	1	1	1

Slika 3.5.7. Roditelji s prikazanim točkama prekida

Primjenom opisanog operatora križanja stvorena su dva potomka prikazana na slici 3.5.8.

Potomak 1	5	5	1	6	3	2	3	1	1
Potomak 2	3	4	6	2	3	2	1	1	1

Slika 3.5.8. Potomci nastali primjenom operatora križanja s dvije točke prekida

Kao i u prethodnom primjeru, ovako dobiveni potomci posve su ispravni. Iz svega navedenog može se zaključiti da ispravnost novonastalih potomaka ne

ovisi o broju točaka prekida te se on može kretati u rasponu od 1 do *N*-1, gdje *N* predstavlja broj gena u kromosomu. Općenito, svi ti operatori križanja mogu se nazvati zajedničkim imenom operatori križanja s N točaka prekida.

3.5.2.3. Uniformni operator križanja

Uniformni operator križanja rabi posebno kreiran predložak koji za svaki gen u kromosomu definira od kojeg će roditelja pojedini gen prijeći na potomka. Slika 3.5.9. prikazuje jedan takav predložak.

Roditelj 1	5	5	6	2	3	2	3	1	1
Roditelj 2	3	4	1	6	3	2	1	1	1
Predložak	0	0	1	0	1	1	0	1	1

Slika 3.5.9. Roditelji s pripadajućim predloškom

Prema navedenom predlošku, prvi i drugi gen u kromosomu bit će naslijeđeni od drugog roditelja, a treći će gen potomak naslijediti od prvog roditelja. Primjenom navedenog operatora križanja dobivaju se potomci sljedećih svojstava:

Potomak 1	3	4	6	6	3	2	1	1	1
Potomak 2	5	5	1	2	3	2	3	1	1

Slika 3.5.10. Potomci nastali primjenom uniformnog operatora križanja

Nule i jedinice u predlošku predstavljaju vjerojatnost nasljeđivanja gena od prvog roditelja. Ako u predlošku stoji nula, tada to znači da je vjerojatnost nasljeđivanja gena od prvog roditelja jednaka nuli. S druge strane, ako u predlošku stoji jedinica, tada je vjerojatnost nasljeđivanja gena od prvog roditelja stopostotna.

Vjerojatnost nasljeđivanja gena od pojedinog roditelja ne treba nužno pratiti binarnu logiku. Vjerojatnost se može kretati u cijelom rasponu od nule do jedinice.

Tako, primjerice, ako u predlošku stoji 0,6, to znači da će potomak s vjerojatnošću od 60% gen naslijediti od prvog roditelja, a vjerojatnost nasljeđivanja tog istog gena od drugog roditelja iznosi 40%. Operator križanja opisan na ovaj način naziva se p-uniformni operator križanja.

3.5.2.4. PMX operator križanja

PMX (*engl. partially mapped crossover*) operator križanja spada u grupu operatora koji se primjenjuju na kromosomima zapisanim na direktan način. Osnovno mu je svojstvo pokušaj zadržavanja što većeg broja gena na mjestima na kojima su se izvorno nalazili u roditelja. Naime, svi operatori križanja koji se primjenjuju na direktno prikazanim kromosomima imaju tendenciju u određenoj mjeri izmiješati gene, pa se kod potomaka pojedini geni pojavljuju na pozicijama na kojima se ne nalaze ni u jednog roditelja. Kao i kod klasičnog operatora križanja s jednom točkom prekida, i kod PMX operatora križanja potrebno je odabrati točku prekida. Kao što to prikazuje slika 3.5.11. točka prekida odabrana je između petog i šestog gena.

Roditelj 1	5	6	8	2	4	1	9	3	7
Roditelj 2	3	5	1	9	6	4	2	7	8

Slika 3.5.11. Roditelji s prikazanom točkom prekida

Ako bi se primijenio klasični operator križanja, nastao bi konflikt između nekoliko gena koji bi se u kromosomu potomka nalazili više puta. Na slici 3.5.12. ti su geni zamijenjeni zvjezdicom.

Potomak 1	5	6	8	2	4	*	*	7	*
Potomak 2	3	5	1	9	6	*	*	*	7

Slika 3.5.12. Potomci s uklonjenim konfliktnim genima

Geni koji nedostaju trebaju se zamijeniti genima drugog roditelja u točnom redoslijedu kako se pojavljuju u kromosomu roditelja. Prvom potomku nedostaju vrijednosti 1, 3 i 9 te ih je potrebno upisati u kromosom potomka redo-

slijedom 3, 1, 9. Isto vrijedi i za drugog potomka, kojem nedostaju geni 2, 4 i 8. Oni se upisuju redom 8, 2, 4 kako se pojavljuju u kromosomu prvoga roditelja. Tako nastaju potomci prikazani na slici 3.5.13.

Potomak 1	5	6	8	2	4	3	1	7	9
Potomak 2	3	5	1	9	6	8	2	4	7

Slika 3.5.13. Potomci nastali primjenom PMX operatora križanja

3.5.2.5. OX operator križanja

OX (*engl. order crossover*) operator križanja neznatno se razlikuje od PMX operatora. Dok PMX operator pokušava zadržati pozicije pojedinih gena u odnosu na roditelje, OX operator želi zadržati apsolutan poredak gena kakav se može pronaći kod roditelja. Stoga OX operator neće iza točke prekida unijeti ni jedan gen, za razliku od PMX operatora koji je automatski unio gene koji nisu bili u konfliktu. Demonstracija OX operatora križanja bit će prikazana na istom primjeru kao i PMX operator.

Roditelj 1	5	6	8	2	4	1	9	3	7
Roditelj 2	3	5	1	9	6	4	2	7	8

Slika 3.5.14. Roditelji s prikazanom točkom prekida

Za početak potrebno je odabrati točku prekida. Ona će u prikazanom primjeru biti na istom mjestu kao i kod PMX operatora, između petog i šestog gena. Kao što se to zorno vidi na slici 3.5.15., nakon primjene klasičnog operatora križanja s prekidom u jednoj točki niti jedan gen nakon točke prekida nije unesen već je zamijenjen zvjezdicom.

Potomak 1	5	6	8	2	4	*	*	*	*
Potomak 2	3	5	1	9	6	*	*	*	*

Slika 3.5.15. Potomci s uklonjenim genima nakon točke prekida

Gene koje u potomcima predstavljaju zvjezdice potrebno je zamijeniti genima istim redoslijedom kako se oni pojavljuju u roditelja. Budući da prvom potomku nedostaju geni 1,3,7 i 9, upisuju se u kromosom potomka redoslijedom 3,1,9,7 jer su u tom poretku u kromosomu drugog roditelja. Na isti se način popunjavaju geni koji nedostaju u kromosomu drugog potomka, ali ovaj put prema redoslijedu kako se pojavljuju u kromosomu prvog roditelja. Konačan izgled potomaka prikazan je na slici 3.5.16.

Potomak 1	5	6	8	2	4	3	1	9	7
Potomak 2	3	5	1	9	6	8	2	4	7

Slika 3.5.16. Potomci nastali primjenom OX operatora križanja

Na osnovi OX operatora križanja Ono [31] je razvio JOX – Job-based crossover.

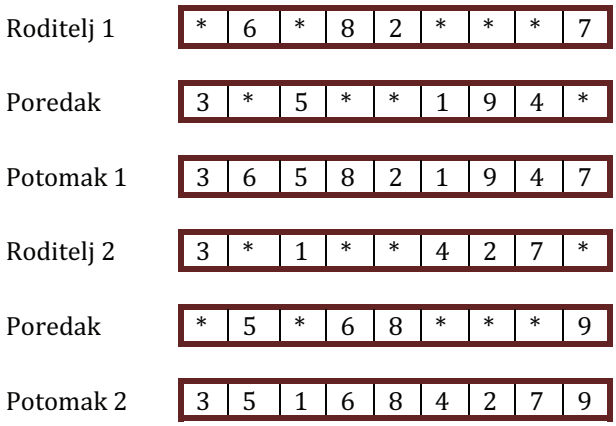
3.5.2.6. UCX operator križanja

UCX (*engl. uniform order crossover*) operator i OX operator veoma su slični. Principijelna je razlika u tome da je OX operator zasnovan na klasičnom operatoru s jednom točkom prekida, a UCX operator zasniva se na uniformnom operatoru križanja. Za početak potrebno je unijeti u kromosome potomaka sve gene koji su predloškom dopušteni. Ukratko to znači da se u kromosom prvog potomka unose svi geni koji su u prvom roditelju, a u predlošku su označeni s 1. Isto vrijedi i za drugog potomka, s tom iznimkom da se unose geni drugog roditelja koji su u predlošku označeni s 0, kako je to prikazano na slici 3.5.17.

Roditelj 1	5	6	8	2	4	1	9	3	7
Roditelj 2	3	5	1	9	6	4	2	7	8
Predložak	0	1	0	1	1	0	0	0	1
Potomak 1	*	6	*	8	2	*	*	*	7
Potomak 2	3	*	1	*	*	4	2	7	*

Slika 3.5.17. Potomci s parcijalnim unosom gena

Geni koji nedostaju unose se redoslijedom kojim se pojavljuju u drugog roditelja. Slika 3.5.18. prikazuje gene koji nedostaju te konačan izgled potomaka.

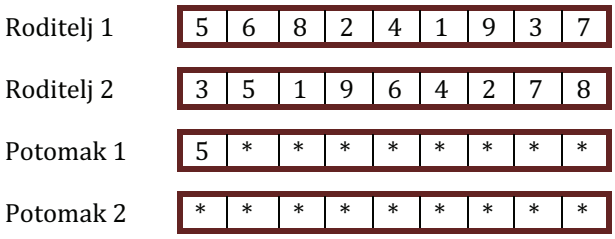


Slika 3.5.18. Potomci nastali primjenom UCX operatora križanja

3.5.2.7. CX operator križanja

PMX i OX operator veoma su slični i imaju jednu zajedničku osobinu. Naime, sva tri operatora imaju tendenciju stavljanja na pojedina mjesta u kromosomu gene koji se na tom mjestu ne nalaze ni u jednog od roditelja. Primjerice na poziciji 6 (prva pozicija nakon točke prekida) roditelji imaju vrijednosti 1 i 4, a u potomaka na tom mjestu nalaze se vrijednosti 3 i 8 neovisno o tome rabi li se operator križanja PMX ili OX. Navedeni se problem u manjoj mjeri javlja kod UCX operatora jer kod njega barem jedan od potomaka na svakom mjestu ima gen koji se na toj poziciji može pronaći kod roditelja. CX (*engl. cycle crossover*) operator uspješno rješava taj problem, ali na žalost rezultati dobiveni njegovom uporabom ne obečavaju. Prema empirijskim podacima koje je predstavio Urlich Bodenhofer iz Fuzzy Logic Loboratoriuma Linz-Hegenberg [20], OX operator je 11% bolji od PMX operatora i 15% bolji od CX operatora.

CX operator križanja funkcionira prema sljedećem principu. Prvo se prvi gen iz prvog roditelja kopira na prvu poziciju u prvom potomku kako je to prikazano na slici 3.5.19.

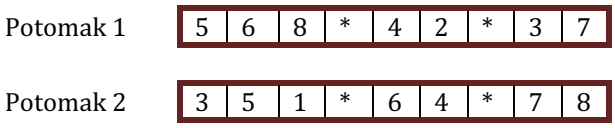


Slika 3.5.19. Prvi korak stvaranja potomaka primjenom CX operatora križanja

Kako ne želimo da se na prvoj poziciji u kromosomima potomaka pojavi gen koji na toj poziciji nije postojao u roditelja, ne preostaje nam ništa drugo nego da prvu poziciju u kromosomu drugog potomka popunimo vrijednošću 3. To znači da vrijednost 3 mora biti upisana u kromosom prvog potomka na poziciju osam ako želimo zadržati gene na istim pozicijama na kojima su bili kod roditelja.

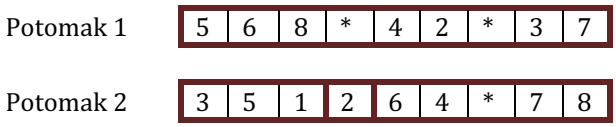
Slika 3.5.20. Drugi korak stvaranja potomaka primjenom CX operatora križanja

Postupak se nastavlja dok se ne naiđe na gen koji je već upisan.



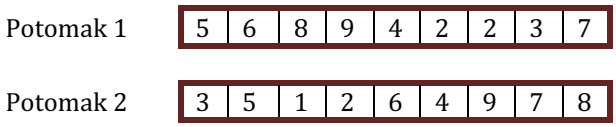
Slika 3.5.21. Završetak prve faze kreiranja potomaka primjenom CX operatora križanja

Tada se cijeli postupak ponavlja, ali tako da se vrijednost iz prvog kromosoma prvo upiše u drugog potomka. Na slici 3.5.22. taj je gen posebno uokviren.



Slika 3.5.22. Početak druge faze kreiranja potomaka primjenom CX operatora križanja

Na ovaj način dobije se konačan izgled obaju potomaka.



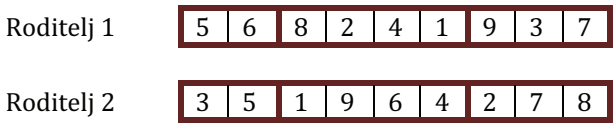
Slika 3.5.23. Potomci nastali primjenom CX operatora križanja

Dogodi li se da sve pozicije u kromosomima potomaka popune a da se ne naiđe na gen koji je već bio popunjen, tada će novi potomci biti identični svojim roditeljima, što znači da nije bilo križanja.

3.5.2.8. LOX operator križanja

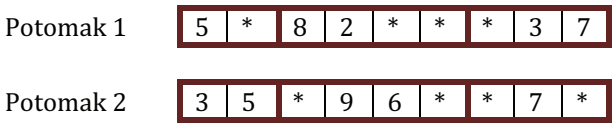
LOX (*engl. location-based crossover*) operator razvili su Falkenauer i Bouffouix 1991. godine. Prema nekim istraživanjima [32], LOX operator daje bolje rezultate od dosad opisanih operatora.

LOX operator zasniva se na operatoru križanja s dvije točke prekida, pa se prvo odabiru te dvije točke, i to nasumce. Dio kromosoma između točaka prekida naziva se interval. U konkretnom primjeru prikazanom na slici 3.5.24. točke prekida nalaze se između drugog i trećeg te šestog i sedmog gena u kromosomu.



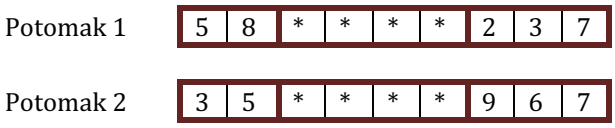
Slika 3.5.24. Roditelji s prikazanim točkama prekida

Geni koji su unutar intervala jednog roditelja, u drugog roditelja zamjenjuju se zvjezdicom.



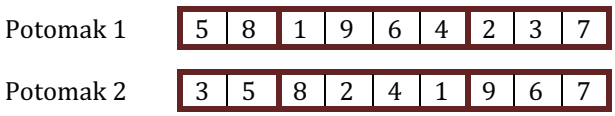
Slika 3.5.25. Potomci s uklonjenim genima iz intervala drugog roditelja

Potom se svi geni označeni zvjezdicom pomiču unutar intervala. Rezultat bi takvog postupka bio da bi svi geni unutar intervala trebali biti označeni zvjezdicom, kako je to prikazano na slici 3.5.26.



Slika 3.5.26. Pomicanje poznatih gena izvan intervala

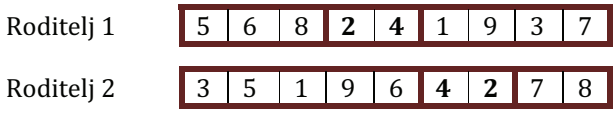
U završnom koraku potrebno je zamijeniti originalne intervale. Potomci dobiveni na ovaj način prikazani su na slici 3.5.27.



Slika 3.5.27. Potomci nakon primjene LOX operatora križanja

3.5.2.9. SXX operator križanja

SXX (*engl. subsequence exchange crossover*) operator 1995. razvio je Komayashi, a glavna mu je svrha pronalaženje sekvenci u kromosomima roditelja koji sadrže identičan set gena, ali u drukčijem poretku. Dužina sekvence može se kretati od dva gena pa do *N*-1, gdje *N* predstavlja broj gena u kromosomu. Na primjeru koji služi za opis pojedinih operatora križanja lako je uočiti sekvencu dužine 2. Ona je na slici 3.5.28. otisnuta masnim slovima.



Slika 3.5.28. Roditelji s označenom sekvencom identičnih gena

Prema definiciji SXX operatora, da bi se dobila dva nova potomka potrebno je zamijeniti navedene sekvence u kromosomima roditelja. Dobiveni potomci prikazani su na slici 3.5.29.

Slika 3.5.29. Potomci nakon primjene SXX operatora križanja

Ako se pažljivije analiziraju oba roditelja, može se identificirati još jedna sekvenca dužine dva gena. Na žalost, sekvenca 1 9 identična je u oba roditelja te je uporaba SXX operatora križanja nad tom sekvencom beskorisna.

3.5.3. Operatori mutacije

Operator mutacije iznimno je bitan za uspješnost genetičkog algoritma. Neki istraživači idu čak toliko daleko da tvrde kako je mutacija najbitniji operator. U stvarnosti genetički algoritam relativno uspješno radi i bez operatora mutacije, što upozorava na to da su te tvrdnje donekle preuveličane. Glavna je uloga operatora mutacije pokušaj zadržavanja veće genetske raznolikosti populacije. Naime, ako je populacija malena, postoji velika vjerojatnost da će vrlo brzo nastupiti dominacija najprilagođenije jedinke te da će osiromašiti genetski materijal populacije. Problem se može riješiti na dva načina: povećanjem učestalosti mutacije, što posredno dovodi do stohastičnosti algoritma, ili povećanjem veličine populacije. Dosadašnja istraživanja pokazala su da mutacija treba ostati relativno mala ili vremenski regulirana. Još je Holland predložio algoritam u kojem se vjerojatnost mutacije s vremenom smanjuje, da bi na kraju evolucije ona bila jednaka nuli. Hollandov se prijedlog pokazao vrlo smislenim usporedi li se utjecaj mutacije na početku evolucije i na njenom

kraju. Naime, dok se još algoritam nije usmjerio prema jednom od optimuma, poželjno je koristiti veću vjerojatnost mutacije jer se time omogućuje pretraživanje većeg prostora rješenja. Kako evolucija odmiče, tako algoritam konvergira prema jednom od lokalnih optimuma te bi velika vjerojatnost mutacije negativno utjecala na konvergenciju. Da je tome tako pokazao je D. H. Ackley 1987. godine na problemu prebrojavanja jedinica[13]. Naime, u opisanom problemu funkcija cilja direktno je proporcionalna broju jedinica u kromosomu s binarnom prezentacijom te je Ackley pokazao da pozitivan utjecaj veličine vjerojatnosti mutacije opada s brojem jedinica u kromosomu, odnosno približavanjem globalnom optimumu. Ako vjerojatnost poboljšanja nakon primjene operatora mutacije definiramo kao:

$$p_{\vec{a}}^{+} = P\left\{f\left(m_{\{p_m\}}(\vec{a})\right) > f(\vec{a})\right\} \tag{85}$$

tada je vjerojatnost mutacije $p \equiv p_m$ i $q = 1 - p$.

U ovako jednostavnom problemu postoje dva scenarija poboljšanja uporabom operatora mutacije. U prvom scenariju, kada ni jedna jedinica ne mutira, potrebno je da barem jedna nula mutira kako bi se broj jedinica povećao. Vjerojatnost da ni jedna jedinica ne mutira iznosi:

$$q^{f_a} \tag{86}$$

gdje je f_a funkcija cilja odnosno broj jedinica. S druge strane mora mutirati barem jedna od $l - f_a$ nula. Ovisno o broju (j) nula koje mutiraju postoji

$$\binom{l - f_a}{j} \tag{87}$$

kombinacija mutacije j nula uz vjerojatnost p^j. Također ostaje $l - f_a - j$ nula koje ne mutiraju uz vjerojatnost

$$q^{l - f_a - j} \tag{88}$$

Ukupna vjerojatnost poboljšanja prema prvom scenariju iznosi:

[13] Problem prebrojavanja jedinica - *engl. counting ones problem*

$$p_1^+ = q^{f_a} \sum_{j=1}^{l-f_a} \binom{l-f_a}{j} p^j q^{l-f_a-j} \tag{89}$$

Drugi je scenarij malo složeniji. Prema tom scenariju, između $i = 1$ i $i = f_a$ jedinica mutira. Tada mora mutirati $j = i + 1$ nula kako bi nakon mutacije bilo više jedinica. Vjerojatnost mutacije nula slična je kao u prvom scenariju uz napomenu da ovisi o broju mutiranih jedinica

$$\sum_{j=i+1}^{l-f_a} \binom{l-f_a}{j} p^j \, q^{l-f_a-j} \tag{90}$$

Navedenu vjerojatnost treba pomnožiti s vjerojatnošću mutacije (i) jedinica te sumirati za sve vrijednosti (i)

$$p_2^+ = \sum_{i=1}^{f_a} \left[\binom{f_a}{i} p^i q^{f_a-i} \left(\sum_{j=i+1}^{l-f_a} \binom{l-f_a}{j} p^j q^{l-f_a-j} \right) \right] \tag{91}$$

Za $j = 1$ izraz u uglatim zagradama reducira se na p_1^+ iz čega proizlazi da je

$$p_{\vec{a}}^+ = p_1^+ + p_2^+ = \sum_{i=0}^{f_a} \left[\binom{f_a}{i} p^i q^{f_a-i} \left(\sum_{j=i+1}^{l-f_a} \binom{l-f_a}{j} p^j q^{l-f_a-j} \right) \right] \tag{92}$$

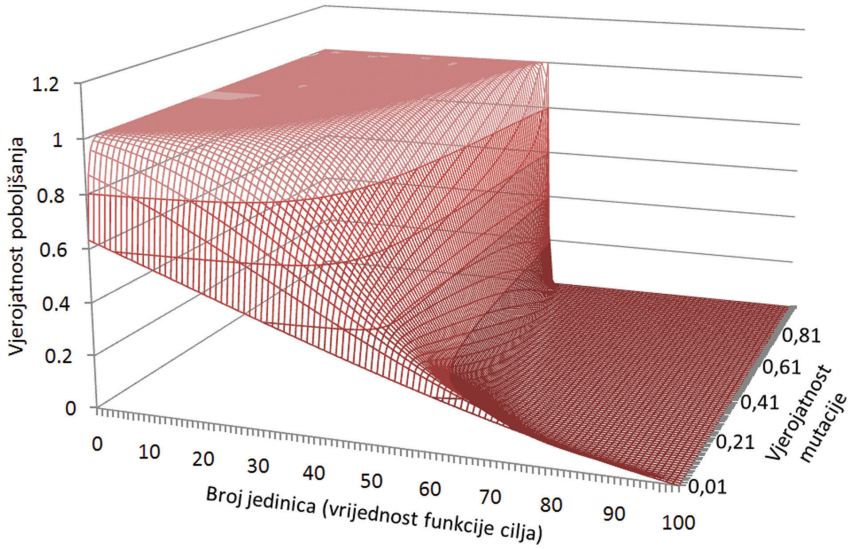
Jednostavno je primijetiti da vjerojatnost poboljšanja zbog uporabe operatora mutacije ovisi o vjerojatnosti mutacije i veličini funkcije cilja, odnosno broju jedinica.

Na slici se može uočiti da vjerojatnost poboljšanja stalno opada s porastom vjerojatnosti mutacije dok god je funkcija cilja malena. Kako se funkcija cilja povećava, odnosno kako algoritam konvergira prema lokalnom optimumu, tako i vjerojatnost poboljšanja opada. U jednom trenutku, kada broj jedinica nadmaši broj nula, vjerojatnost poboljšanja opada s porastom vjerojatnosti

mutacije. Neznatnu iznimku čine iznimno male vjerojatnosti mutacije čije povećanje uzrokuje povećanje vjerojatnosti poboljšanja. Potrebno je istaknuti da je riječ o iznimno maloj vjerojatnosti mutacije

$$0 < p_m < 0{,}1 \tag{93}$$

Navedeni rezultati zorno pokazuju da je Hollandov prijedlog o vremenski reguliranoj mutaciji ispravan te da veće vrijednosti vjerojatnosti mutacije, osim na samom početku evolucije, negativno utječu na konvergenciju algoritma.



Slika 3.5.30. Vjerojatnost poboljšanja primjenom operatora mutacije

3.5.4. Uvjeti prekida genetičkoga algoritma

Genetički algoritam, kao i svi iterativni algoritmi, mora imati uvjet prekida. U literaturi se mogu pronaći različiti prijedlozi, od kojih će tri biti pobliže definirana:

- prekid nakon određenog broja generacija
- prekid zbog gubitka genetske raznolikosti
- prekid algoritma nakon što u unaprijed zadanom broju generacija nije pronađeno bolje rješenje.

3.5.4.1. Broj generacija

Prekidanje algoritma nakon unaprijed zadana broja generacija najjednostavniji je uvjet prekida algoritma. Ovakav se način prekida algoritma najčešće rabi jer omogućuje statističku usporedbu predloženog algoritma s algoritmima drugih istraživača. Naime, moguće je usporediti vrijednost dobivenog rješenja nakon jednakog broja generacija ili jednakog broja ispitanih članova generacije.

3.5.4.2. Gubitak genetske raznolikosti

Gubitak genetske raznolikosti cjelokupne generacije služi kao uvjet prekida genetičkog algoritma kada definiranje fiksnog broja generacija nije moguće ili nije poželjno. Kako članovi generacije konvergiraju prema optimalnom rješenju, tako se smanjuje genetska raznolikost članova populacije (članovi populacije postaju genetski sve sličniji). Činjenica da je genetska raznolikost populacije smanjena uzrokuje smanjenu istraživačku mogućnost genetičkog algoritma. Ako je genetički algoritam pronašao rješenje koje pripada lokalnom optimumu, zbog male genetske raznolikosti populacije, algoritam se ne može odvojiti od pronađenog optimuma. U takvoj situaciji daljnji rad algoritma besmislen je, jer je vjerojatno već pronađeno rješenje koje pripada lokalnom optimumu te daljnja poboljšanja nisu vjerojatna. Jedina mogućnost za odvajanje od lokalnog optimuma u genetičkom je operatoru mutacije. Budući da je vjerojatnost mutacije kod genetičkih algoritama relativno mala, maksimalno 3%, nije realno očekivati da će operator mutacije riješiti problem.

3.5.4.3. Nemogućnost pronalaska boljeg rješenja

Nemogućnost pronalaženja boljeg rješenja kroz određen broj generacija također može biti uvjet za prekid genetičkog algoritma. Naime, katkad nije jednostavno odrediti kolika je sličnost između pojedinih jedinki u populaciji, odnosno kod velikih populacija izračun može trajati poprilično dugo. Tada se definira broj generacija koje algoritam može proći a da ne pronađe bolje rješenje. Nakon što taj broj generacija istekne (npr. 50), algoritam se prekida.

4. GENETIČKI ALGORITMI I PROBLEM RASPOREĐIVANJA

Uporaba genetičkih algoritama u rješavanju problema raspoređivanja pojavila se vrlo rano u razvoju genetičkih algoritama. Samo deset godina pošto je John Holland sa suradnicima postavio osnove genetičkih algoritama L. Davis prvi je pokušao riješiti problem raspoređivanja [5] pomoću genetičkog algoritma. Nakon tog pokušaja zaredali su radovi drugih istraživača. Razlog tako veliku interesu lako je pronaći u činjenici da je problem raspoređivanja sveprisutan u mnogim područjima ljudskog života. K tome, problem raspoređivanja spada u kategoriju permutacijskih problema te se može povezati s još nekim problemima poput problema trgovačkog putnika, što dodatno povećava njegov značaj. Tijekom razvoja genetičkih algoritama vezanih uz problem raspoređivanja razni istraživači predložili su različite metode kodiranja kromosoma: [33]

- kodiranje zasnovano na operacijama (*engl. operation-based representation*) [34]
- kodiranje zasnovano na poslovima (*engl. job-based representation*) [35]
- kodiranje zasnovano na povlaštenim listama (*engl. preference list-based representation*) [5]
- kodiranje zasnovano na odnosu parova poslova (*engl. job pair relation-based representation*) [36]
- kodiranje zasnovano na prioritetima (*engl. priority role-based representation*) [37]
- kodiranje zasnovano na disjunkcijskom grafu (*engl. disjunctive graph-based representation*)
- kodiranje zasnovano na vremenu završetka (*engl. completion time-based representation*) [38]
- kodiranje zasnovano na strojevima (*engl. machine-based representation*)
- kodiranje zasnovano na slučajnim brojevima (*engl. random-based representation*).

Sve navedene vrste kodiranja mogu se podijeliti u dvije grupe:

- direktan pristup – raspored je kodiran u kromosom (kodiranje zasnovano na operacijama, poslovima, parovima poslova, vremenu završetka i slučajnim brojevima)
- indirektan pristup – raspored nije kodiran u kromosom (kodiranje zasnovano na povlaštenim listama, prioritetu, disjunkcijskom grafu i strojevima).

Od navedenih načina kodiranja pobliže će biti opisani kodiranje zasnovano na disjunkcijskom grafu jer omogućuje korištenje binarnog prikaza, kodiranje zasnovano na povlaštenim listama jer je korišteno u prvom pokušaju rješavanja problema raspoređivanja uporabom genetičkog algoritma te svakako u praksi najzastupljenije kodiranje zasnovano na operacijama.

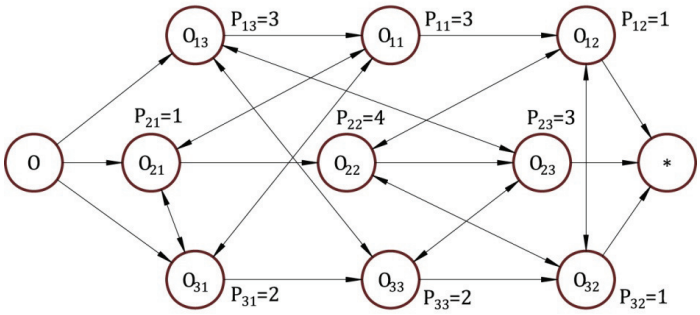
4.1. KODIRANJE ZASNOVANO NA DISJUNKCIJSKOM GRAFU

Genetički algoritmi kako ih je zamislio John Holland zasnovani su na binarnoj prezentaciji kromosoma. Na žalost, takav način prikaza nije najbolje prilagođen rješavanju permutacijskih vrsta problema te je njegova vrijednost donekle precijenjena [39]. Stoga prve varijante genetičkih algoritama koji su se bavili problemom raspoređivanja nisu koristile binarnu prezentaciju kromosoma. Opis rasporeda uporabom disjunkcijskog grafa poznat je još od 1964. godine kada su ga u radu »Les problemes d'ordonnancement avec contraintes disjunctives« predložili B. Roy i B. Sussmann. Genetički algoritam s binarnom prezentacijom koji se zasnivao na svojstvima disjunkcijskoga grafa predložili su Tamaki i Nishikawa tek 1992. godine.

Raspored se može opisati kao disjunkcijski graf $G = (V, C \cup D)$, gdje je V set čvorova koji predstavljaju operacije, C je set konjuktivnih lukova koji predstavljaju tehnološki proces, a D je set disjunkcijskih lukova koji čine parove operacija koji moraju biti provedeni na istom stroju.

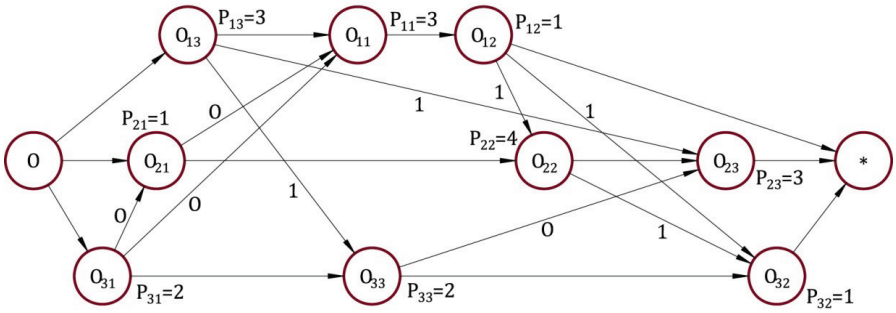
Na slici 4.1.1. može se uočiti da neusmjereni disjunkcijski lukovi spajaju čvorove (operacije) kojima je drugi indeks jednak (isti stroj). Da bi disjunkcijski graf predstavljao neki određen raspored potrebno je sve neusmjerene disjunkcijske lukove pretvoriti u usmjerene. Na ovaj način dobiven raspored spada u kategoriju poluaktivnih rasporeda ako zadovoljava još jedan bitan

uvjet. Naime, da bi raspored bio legalan novonastali usmjereni disjunkcijski graf mora biti acikličan. Kao što se vidi iz slike 4.1.1., kod cikličkog disjunkcijskog grafa nije moguće odrediti prvu operaciju na pojedinom stroju pa takav graf ne predstavlja ispravan raspored. Kao što će kasnije biti pokazano, postoji rješenje za navedeni problem.



Slika 4.1.1. Disjunkcijski graf (neusmjereni)

Kao što je već spomenuto, opisivanje usmjerenoga grafa u obliku binarnoga kromosoma riješili su Tamaki i Nishikawa tako što su su sve usmjerene disjunkcijske lukove ovisno o njihovu smjeru označili sa 0 ili 1. Ako se dvije operacije O_{ij} i O_{kj} provode na istom stroju (drugi im je indeks jednak) te ako je $i < k$ i ako je disjunkcijski luk usmjeren od operacije O_{ij} prema operaciji O_{kj} tada se navedeni luk označava s 1. Ako je luk usmjeren u suprotnom smjeru, bio bi označen s 0.



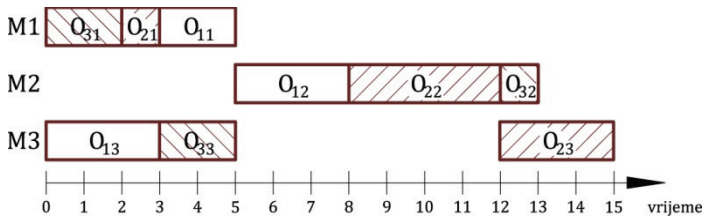
Slika 4.1.2. Disjunkcijski graf (usmjereni)

Ovako se mogu označiti svi usmjereni disjunkcijski lukovi koji predstavljaju određen raspored. Zapisivanjem nula i jedinica u obliku vektora kreiran je binarni kromosom nad kojim je moguće provesti operacije genetičkog algoritma. Raspored prikazan na slici 4.1.2. s pravilno označenim usmjerenim disjunkcijskim lukovima u obliku kromosoma glasio bi:

0	0	0	1	1	1	1	1	0

Slika 4.1.3. Prikaz kromosoma zasnovanog na disjunkcijskom grafu

Prednost binarnog zapisa kromosoma više je nego jasna. Svi klasični operatori križanja i mutacije mogu se izvršiti nad kromosomom te ne postoji potreba za uporabom specijalističkih varijanti opisanih u poglavlju 3.5. Vrlo je jednostavno napraviti Ganttov dijagram na osnovu usmjerenog disjunkcijskog grafa, pogotovu ako se sve operacije razmaknu tako da se svi usmjereni lukovi kreću s lijeva na desno kao što je prikazano na slici 4.1.2.



Slika 4.1.4. Ganttov dijagram zasnovan na disjunkcijskom grafu (polu aktivan raspored)

Međutim, ova prezentacija nije bez nedostataka jer uporabom klasičnih operatora križanja i mutacije mogu nastati neispravna rješenja (rasporedi). Neispravnost rasporeda očituje se u već spomenutom cikličkom obliku disjunkcijskoga grafa, što na neki način treba riješiti. Problemu se može pristupiti na dva načina. Za neispravno rješenje mogu se nametnuti velike sankcije (penali) ili se nekim usmjerenim disjunkcijskim lukovima može promijeniti smjer tako da graf više ne bude cikličan. Postupak prevođenja cikličnoga grafa u aciklični naziva se harmonizacija [36]. Kod harmonizacije je bitno da novonastali raspored mora biti što sličniji originalnom rasporedu, što znači da treba promi-

jeniti smjer što manjem broju usmjerenih disjunkcijskih lukava, odnosno Hemmingova udaljenost između dva binarna zapisa mora biti što manja. Proces harmonizacije dijeli se na lokalnu i globalnu harmonizaciju. Lokalna harmonizacija treba riješiti problem cikličnosti među operacijama na istom stroju. Nakon što lokalna harmonizacija riješi sve strojeve još može postojati cikličnost među njima. Rješavanje tog problema zadatak je globalne harmonizacije. Bitna je značajka harmonizacije da novo dobiveno rješenje služi samo za izračun funkcije cilja dok se originalni kromosom ne mijenja. Na taj je način zadržana genetska raznolikost unutar populacije. S druge strane moguće je evoluciju nastaviti i s novonastalom varijantom kromosoma. Takav se postupak naziva *forcing* i ima neke dobre i neke loše osobine. Naime, novonastali potomak ima neke osobine koje nije mogao naslijediti od roditelja nego ih je zapravo stekao tijekom života. U prirodi nije poznata situacija u kojoj se na potomke prenose stečene osobine (lamarckizam[14]) te se *forcing* ne može smatrati prirodnim. Posljedica je *forcinga* i smanjenje genetskog materijala populacije, što direktno dovodi do preuranjene konvergencije. S druge strane, ovakvim postupkom bitno se poboljšava kvaliteta dobivenih rješenja.

4.2. KODIRANJE ZASNOVANO NA POVLAŠTENIM LISTAMA

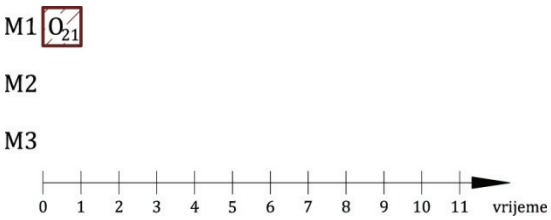
Kodiranje zasnovano na povlaštenim listama predložio je L. Davis 1985. godine. Po načinu kodiranja spada u indirektnu grupu jer kromosom ne predstavlja raspored nego povlaštenu listu operacija za svaki pojedini stroj. Kromosom zapisan u ovom obliku za primjer opisan u poglavlju 2.6. glasi:

2	3	1	1	2	3	2	1	3

Slika 4.2.1. Kromosom zasnovan na povlaštenim listama

[14] Ideja da organizmi mogu prenijeti karakteristike stečene za života na svoje potomke. Ime je dobilo po francuskom biologu Jean-Baptistu Lamarcku

Važno je primijetiti da je kromosom podijeljen u tri grupe (broj strojeva) te da su u svakoj grupi sve operacije. Navedene grupe mogu se nazivati i genima jer su nedjeljive, odnosno operator križanja može prekinuti kromosom samo između grupa, a nikako posred grupe. Kako se povlaštena lista pretvara u raspored? U prvom koraku potrebno je odrediti operacije koje se nalaze na najvišim pozicijama na povlaštenoj listi te za njih provjeriti zadovoljavaju li tehnološki raspored. U konkretnom primjeru, prema povlaštenoj listi odabiru se operacije O_{21}, O_{12} i O_{23}. Od navedenih operacija tehnološki raspored zadovoljava samo operacija O_{21}. Sukladno tome to je prva operacija koja se unosi u raspored.

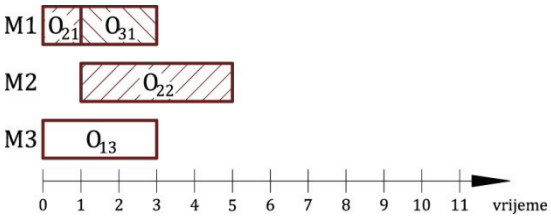


Slika 4.2.2. Unos prve operacije s povlaštene liste

Pošto je odabrana operacija unesena u raspored potrebno je odabrati sljedeće tri operacije prema novoj povlaštenoj listi, čime se popis mogućih operacija povećava na pet.

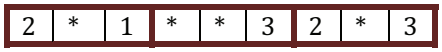
2	3	1	*	2	3	2	1	3

Slika 4.2.3. Povlaštena lista nakon unosa prve operacije



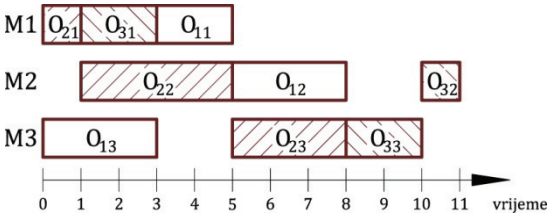
Slika 4.2.4. Raspored nakon unosa novih triju operacija s povlaštene liste

Lista odabranih operacija glasi O_{12}, O_{23}, O_{31}, O_{22} i O_{13}. Od navedenih operacija prema tehnološkom procesu mogu se rasporediti O_{31}, O_{22} i O_{13}, čime se lista ponovo smanjuje na dvije operacije koje se ne mogu rasporediti.



Slika 4.2.5. Povlaštena lista

Dodavanjem posljednjih triju operacija na povlaštenu listu dobiva se konačni popis operacija koje se definitivno mogu rasporediti. Lista se sastoji od operacije O_{12}, O_{23}, O_{11}, O_{32}, O_{33}, a konačni raspored prikazan je na slici 4.2.6.



Slika 4.2.6. Konačni raspored operacija prema povlaštenoj listi

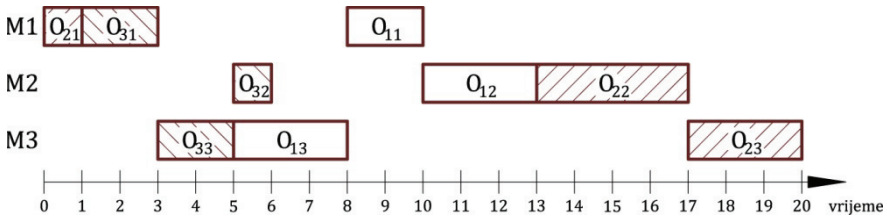
4.3. KODIRANJE ZASNOVANO NA OPERACIJAMA

Kodiranje zasnovano na operacijama pripada kategoriji direktnog pristupa gdje je raspored direktno kodiran u kromosom. Postoje dva pristupa. Prvi se zasniva na problemu trgovačkog putnika te je u njemu svakoj operaciji dodijeljeno ime baš kao da je riječ o gradovima. Međutim, za razliku od problema trgovačkog putnika, kod problema raspoređivanja postoje još i tehnološka ograničenja (poredak operacija za svaki posao), a zbog toga ne predstavljaju svi kromosomi ispravan raspored. Kako bi riješili taj problem Gen, Tsujimura i Kobota [34] predložili su varijantu u kojoj se sve operacije istog posla označavaju istim imenom. Pažljivijim razmatranjem kromosoma na slici 4.3.1. može se uočiti da se na ovaj način ne može napraviti neispravan raspored kao u prethodnom slučaju.

2	3	3	3	1	1	1	2	2

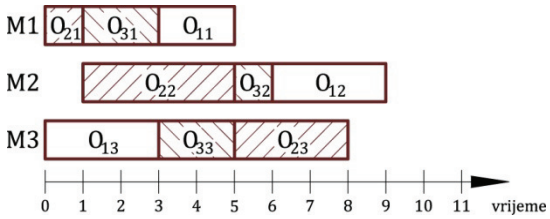
Slika 4.3.1. Kromosom zasnovan na operacijama

Kromosom prikazan na slici 4.3.1. može se predstaviti u obliku Ganttova dijagrama tako da se redom očitavaju operacije. Prva operacija u kromosomu pripada poslu 2. Poznavanjem tehnološkog procesa može se zaključiti da ta dvojka pripada operaciji O_{21}. Posljednja dva gena u kromosomu također predstavljaju operacije na poslu 2, i to redom O_{22} i O_{23}. Tako se može odrediti redoslijed svih operacija definiranih prikazanim kromosomom. Konačni redoslijed operacija glasi O_{21}, O_{31}, O_{33}, O_{32}, O_{13}, O_{11}, O_{12}, O_{22} i O_{23}. Navedenim se redom operacije upisuju u Ganttov dijagram koji je za konkretan primjer prikazan na slici 4.3.2.



Slika 4.3.2. Poluaktivni raspored (kodiranje zasnovano na operacijama)

Prikaz na slici 4.3.2. dobiven je uporabom poluaktivnog rasporeda. Znatno se bolji raspored može dobiti uporabom aktivnog rasporeda, kao što je prikazano na slici 4.3.3.



Slika 4.3.3. Aktivni raspored (kodiranje zasnovano na operacijama)

Kodiranje zasnovano na operacijama relativno je jednostavno, ali ima bitan nedostatak. Naime u poglavlju 2.2. navedeno je da klasični operatori mutacije i križanja nisu primjenjivi na kromosome koji se koriste direktnim načinom kodiranja jer mogu uzrokovati stvaranje neispravnih rasporeda. Upravo kodiranje zasnovano na operacijama spada u tu kategoriju.

4.4. KODIRANJE PREMA REFERENTNOJ LISTI

Sva tri prikazana načina kodiranja uz prednosti donose i neka više ili manje ozbiljna ograničenja. Problem uporabe harmonizacije u kodiranju zasnovanom na disjunkcijskom grafu ili nemogućnost korištenja klasičnih operatora križanja i mutacije nije zanemariv.

Način kodiranja koji nema ni jedan od navedenih problema kodiranje je prema referentnoj listi [20]. Naprosto, referentna lista omogućuje prevođenje direktnog kodiranja u indirektno i obrnuto, ali tako da se mogu koristiti klasični operatori mutacije i križanja. Primjer kodiranja zasnovanog na referentnoj listi prevest će kodiranje zasnovano na operacijama u oblik pogodan za klasične operatore.

Za početak je potrebno napraviti referentnu listu. Budući da je riječ o prikazu modificiranog kodiranja zasnovanog na operacijama, referentna lista će se sastojati od popisa mogućih operacija. U konkretnom primjeru rasporeda 3x3 iz poglavlja 2.2., referentna lista sastoji se od 9 elemenata i glasi

$$RL_0 = \{1,1,1,2,2,2,3,3,3\} \tag{94}$$

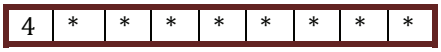
Na osnovi navedene referentne liste kodirat će se kromosom iz prethodnog poglavlja.

2	3	3	3	1	1	1	2	2

Slika 4.4.1. Zapis kromosoma zasnovan na operacijama

Postupak se sastoji od zamjene vrijednosti gena s odgovarajućim brojem pozicije na kojoj se ta vrijednost nalazi u referentnoj listi. Budući da je vrijed-

nost prvog gena 2, potrebno je potražiti prvo pojavljivanje vrijednosti 2 u referentnoj listi. U konkretnom primjeru vrijednost 2 nalazi se na četvrtom mjestu u referentnoj listi te se u kromosomu zasnovanom na referentnoj listi na prvo mjesto upisuje 4, kao što se može vidjeti na slici 4.4.2.



Slika 4.4.2. Unos prvog člana u kromosom

Istovremeno se referentna lista smanjuje za jednog člana

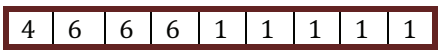
$$RL_1 = \{1,1,1,2,2,3,3,3\} \tag{95}$$

U sljedećem se koraku u referentnoj listi treba potražiti pozicija prvog pojavljivanja vrijednosti sljedećega gena. U konkretnom primjeru riječ je o vrijednosti 3. Vrijednost 3 prvi se put u skraćenoj referentnoj listi nalazi na poziciji 6 te će se ta vrijednost upisati na drugo mjesto u kromosom pa ponovo skratiti referentnu listu za jedno mjesto:

Slika 4.4.3. Unos drugog člana u kromosom

$$RL_2 = \{1,1,1,2,2,3,3\} \tag{96}$$

Postupak se ponavlja dok se ne popune sve vrijednosti kromosoma, odnosno dok se dužina referentne liste ne smanji na nulu. Kromosom kodiran prema referentnoj listi za navedeni primjer nakon provedbe svih devet koraka glasi:



Slika 4.4.4. Kromosom zasnovan na referentnoj listi

U opisanom načinu kodiranja bitno je da se vrijednosti pojedinih gena ponašaju prema određenoj zakonitosti. Naime, ako se pozicija pojedinog gena u

kromosomu (*engl. locus*) označi s (i), tada se (i) kreće u rasponu od 1 do N. N predstavlja dužinu kromosoma. Vrijednost p_i koja se može upisati u svaku poziciju mora zadovoljavati sljedeći uvjet:

$$1 \leq p_i \leq N - i + 1 \qquad (97)$$

Pojednostavnjeno to znači da se u prvu poziciju kromosoma može upisati bilo koja vrijednost od 1 do N, a na posljednjoj poziciji može stajati samo vrijednost 1. Navedeno je pravilo posve nebitno u primjeni klasičnih operatora križanja, ali je iznimno bitno kod operatora mutacije, jer operator mora na neki način znati unutar kojeg raspona može mutirati vrijednosti gena na određenoj poziciji.

Postupak dekodiranja kromosoma u originalnu varijantu jednako je jednostavan kao i prvobitno kodiranje. Cijeli postupak kodiranja i dekodiranja bit će prikazan na primjeru križanja dvaju kromosoma uporabom operatora križanja s jednom točkom prekida. Slika 4.4.5. prikazuje roditelje.

Roditelj 1	3	1	3	2	2	1	2	1	2
Roditelj 2	1	3	3	2	1	2	1	2	3

Slika 4.4.5. Roditelji (kodiranje zasnovano na operacijama)

Izgled kromosoma roditelja nakon provedbe kodiranja zasnovanog na referentnoj listi prikazan je na slici 4.4.6.

Roditelj 1	7	1	6	3	3	1	2	1	1
Roditelj 2	1	6	6	3	1	2	1	1	1

Slika 4.4.6. Roditelji (kodiranje zasnovano na referentnoj listi)

Nakon primjene operatora križanja s jednom točkom prekida (pozicija prekida odabrana je između četvrtog i petog kromosoma) kreirana su dva potomka koji su također kodirani prema referentnoj listi.

Potomak 1	7	1	6	3	1	2	1	1	1
Potomak 2	1	6	6	3	3	1	2	1	1

Slika 4.4.7. Potomci (kodiranje zasnovano na referentnoj listi)

Dobivene je potomke potrebno pretvoriti u varijantu kodiranja zasnovanog na operacijama tako da se jednostavno može napraviti Ganttov dijagram. Varijanta potomaka zasnovanih na operacijama prikazana je na slici 4.4.8.

Potomak 1	3	1	3	2	1	2	1	2	3
Potomak 2	1	3	3	2	2	1	2	1	3

Slika 4.4.8. Potomci (kodiranje zasnovano na operacijama)

Kao što se iz slike 4.4.8. može vidjeti, novokreirani potomci predstavljaju potpuno ispravne rasporede iako je upotrijebljen operator križanja s jednom točkom prekida, koji ima izrazito nepovoljan utjecaj na kromosome koji su kodirani na direktan način, a koriste se u rješavanju permutacijskog tipa problema. Na ovaj se način ne mogu nad svim varijantama direktnog kodiranja primijeniti klasični operatori križanja i mutacija. Naravno da takav postupak nije bez nedostataka. Naime, iako se zbog vrlo kratkog kromosoma na navedenom primjeru to ne može uočiti, geni iza točke prekida mogu poprimiti izrazito nasumičan raspored, a to svakako nije povoljno svojstvo. Posljedica je navedenog svojstva nemogućnost preciznijeg pretraživanja prostora problema. To je donekle manje izraženo kod problema raspoređivanja po strojevima jer je moguće primijeniti aktivni raspored koji donekle dokida nasumičnost operacija u zadnjem dijelu kromosoma. S druge strane, kod permutacijskih problema poput problema trgovačkog putnika, gdje ne postoji ništa slično procesu kreiranja aktivnog rasporeda, kodiranje zasnovano na referentnoj listi pokazuje određen stupanj stohastičnosti, pogotovo kod velikih problema.

5. GENETIČKI ALGORITMI S NIŠAMA

Genetičkim algoritmima s nišama cilj je podjela populacije na veći broj stabilnih grupa (niša) unutar kojih je potrebno pronaći optimalno rješenje. U prostoru pretraživanja u kojem postoji veći broj lokalnih optimuma presudna je mogućnost istodobnog kreiranja i zadržavanja većeg broja rješenja kroz cijeli tijek evolucije kako bi se izbjeglo zapinjanje algoritma u jednom od lokalnih optimuma. Pronalaženje globalnog optimuma u odnosu na lokalne nije samo svojstvo algoritma s nišama. Naime, svaki genetički algoritam koji ima manji pritisak selekcije te se koristi manje destruktivnim operatorima križanja i mutacije, ima veće šanse pronaći globalni optimum. Problem s tako postavljenim algoritmom u tome je da algoritam pretražuje bitno širi prostor definicije problema, ali ne može se fokusirati na dobra rješenja. S druge strane, povećanje pritiska selekcije dovodi do preuranjene konvergencije, što također nije poželjno. Rješenje je ovog naoko nerješivog problema u formiranju niša koje imaju tzv. restauracijski pritisak. Naime, unutar niša moguće je povećati pritisak selekcije a da to ne dovede do preuranjene konvergencije na globalnoj razini.

Bitno je istaknuti da algoritmi s nišama imaju određena povoljna svojstva, poput restauracijskog pritiska, koja općenito mogu biti uspješno primjenjiva [23] neovisno o problemu koji se rješava. Upravo ta činjenica bila je glavni motiv prilikom odabira algoritma s nišama za rješavanje problema raspoređivanja.

5.1. DOSADAŠNJI RADOVI

Veći se broj istraživača s manjim ili većim uspjehom okušao u kreiranju i održavanju stabilnih niša unutar populacije. Među prvim genetičkim algoritmima koji su u nekom obliku stvarali niše bili su Cavicchijev algoritam s predodabirom (*engl. Preselection*) i DeJongov algoritam grupiranja (*engl. Crowding*) [40] [41] [42] [43] [44]. Oba algoritma spadaju u kategoriju algoritama s ograničenom zamjenom (*engl. restricted replacement*) te se u svom originalnom obliku nisu bavila kreiranjem niša već očuvanjem genetske raznolikosti. Razloge njihove neuspješnosti analizirao je i opisao Mahfoud 1992.

godine te na osnovi te analize predložio algoritam pod nazivom deterministički algoritam grupiranja (*engl. Deterministic Crowding*). Deterministički algoritam grupiranja [42] za razliku od svojih prethodnika pokazao se iznimno uspješnim u kreiranju i održavanju niša. Osim njih, poznati su i vrlo slični algoritmi koji ugrađuju ograničeno natjecanje (*engl. restricted competition*) umjesto ograničene zamjene, ali najuspješnija metoda kreiranja i zadržavanja niša svakako je algoritam s raspodjelom funkcije cilja.

5.1.1. Algoritam s predodabirom (Preselection algoritam)

Cavicchijev algoritam s predodabirom spada u kategoriju *stady-state* genetičkih algoritama, što znači da se u svakoj generaciji ne izmjenjuju svi članovi populacije već samo ograničen broj jedinki. Algoritam se zasniva na tome da su potomci u pravilu slični svojim roditeljima te želi li se zadržati genetska raznolikost, što je primarni cilj algoritma s predodabirom, pojedini potomak treba zamijeniti jednog od roditelja, u pravilu lošijeg. Dogodi li se da je potomak lošiji od lošijeg roditelja, zamjene neće biti. Razlog što su roditelji odabrani za jedinke koje će biti zamijenjene vrlo je jednostavan. Naime, Cavicchio je smatrao da je usporedba svih jedinki u populaciji u računalnom smislu preskupa te da nije poželjno u cjelokupnoj populaciji tražiti najsličniju jedinku. Uvažavajući takvo mišljenje, algoritam uvijek lošiju i sličnu jedinku traži među roditeljima, što bitno ubrzava proces evolucije.

5.1.2. Algoritam grupiranja (Crowding algoritam)

Pet godina poslije Cavicchija, DeJong je predložio algoritam grupiranja u kojem je uklonjeno ograničenje prema kojem potomci mogu zamijeniti samo svoje roditelje. Algoritam grupiranja također je algoritam *stady-state* te se i kod njega ne mijenjaju svi članovi populacije u svakoj generaciji. Nakon što su uporabom proporcionalne selekcije i genetičkog operatora križanja stvoreni novi potomci koji trebaju biti ubačeni u populaciju, potrebno je odabrati jednak broj jedinki koje će napustiti populaciju. Za razliku od algoritma s predodabirom, to nužno ne moraju biti roditelji. Naime, za svaku novu jedinku potrebno je nasumično odabrati CF jedinki (CF – faktor grupiranja[15]). U

[15] CF - crowding factor

odabranoj grupi potrebno je pronaći najsličniju jedinku te nju zamijeniti novonastalim potomkom. Sličnost između jedinki definira se genotipski. [42]

Algoritam je donekle mogao zadržati genetsku raznolikost, što se i očekivalo, ali je također imao veliku stohastičku grešku, pogotovo kod malih CF. Naime, ako bi CF iznosio na primjer dva, jednako kao kod algoritma s predodabirom, tada bi zamjena najsličnije jedinke bila istinski ugrožena, jednostavno zato što su dvije navedene jedinke odabrane nasumice, a kod algoritma s predodabirom riječ je o roditeljima koji apriori imaju određeni stupanj sličnosti s potomcima. Stoga CF mora biti značajno veći. U ekstremnom slučaju CF može biti jednak broju jedinki u populaciji, što dovodi do usporedbe svake jedinki sa svakom jedinkom u populaciji. Navedenu je situaciju Cavicchio pokušao izbjeći pod svaku cijenu. Bitno je istaknuti da je procesorska snaga računala u sedamdesetim godinama prošlog stoljeća bila znatno manja nego danas, ali isto tako valja imati na umu da se broj kombinacija povećava eksponencijalnom progresijom s povećanjem veličine populacije, čime se po složenosti algoritam približava algoritmu s raspodjelom funkcije cilja[16]. Istraživanja koje je proveo DeJong pokazala su najbolje rezultate ako je CF postavljen na 2 ili 3.

5.1.3. Deterministički algoritam grupiranja (Deterministic Crowding algoritam)

Analizirajući nedostatke prethodnih metoda Mahfoud, je predložio deterministički algoritam grupiranja. Prema njegovu modelu potrebno je kreirati $N/2$ parova roditelja gdje je N broj jedinki u populaciji. Kreirani parovi uporabom operatora križanja i eventualno operatora mutacije stvaraju dva nova potomka. Novonastali potomci natječu se sa svojim roditeljima za mjesto u sljedećoj generaciji. Natjecanje je zapravo turnir u kojem pobjednik odlazi u sljedeću generaciju uz uvjet da se parovi u turniru odabiru prema sličnosti. Postoje dvije kombinacije turnirskih parova: prvi roditelj – prvi potomak i drugi roditelj – drugi potomak, odnosno prvi roditelj – drugi potomak i drugi roditelj – prvi potomak. Za turnir se odabire kombinacija parova koja pokazuje najveću sličnost, s time da se sličnost može odrediti i genotipski i fenotipski.

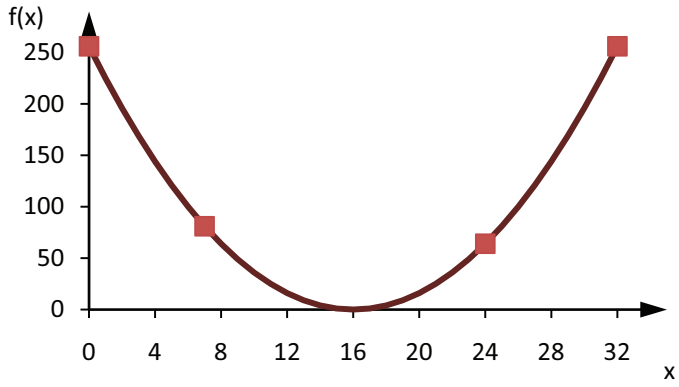
Deterministički algoritam grupiranja ima dvije bitne prednosti u odnosu na algoritam s raspodjelom funkcije cilja (poglavlje 5.1.5.). Prvo, nije potrebno odrediti radijus sličnosti za uspješan rad algoritma, što posve uklanja potrebu

[16] Fitness Sharing

za poznavanjem domene problema koji se rješava (radijus sličnosti nužan je preduvjet za uspješan rad algoritma s raspodjelom funkcije cilja što u algoritam implicitno unosi znanje o problemu koji se rješava). Deterministički algoritam grupiranja ima i implicitno ugrađen elitizam, što znači da jednom pronađen globalni optimum ne može nestati iz populacije. Ništa slično ne postoji kod algoritma s raspodjelom funkcije cilja; dapače, elitizam je u suprotnosti s logikom algoritma s raspodjelom funkcije cilja.

5.1.4. Ograničeno natjecanje

Do sada je uvijek bilo riječi o ograničavanju grupe jedinki protiv kojih se potomci bore za mjesto u sljedećoj generaciji. Donekle drukčiji pristup je ograničavanje borbe pojedinih genotipski ili fenotipski bitno različitih jedinki za ulogu roditelja tijekom procesa selekcije. Postoji dobro opravdanje za dopuštanje sparivanja samo sličnih jedinki, pri čemu se implicitno stvaraju niše. Ako se promotri funkcija cilja prikazana na slici 5.1.1. [2], lako se može uočiti da postoje dva optimuma.

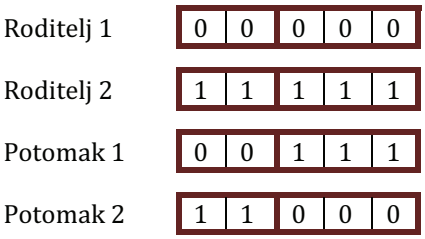


Slika 5.1.1. Primjer funkcije cilja

Funkcija cilja definirana je formulom

$$f(x) = (x - 16)^2, x \in \{0 \dots 32\} \qquad (98)$$

Funkcija doseže maksimum za vrijednosti 0 (00000) i 32 (11111). Ako se upravo te dvije optimalne, ali bitno različite jedinke odaberu za roditelje te novi potomci nastanu uporabom operatora križanja s jednom točkom prekida (za točku prekida odabrana je pozicija između drugog i trećeg gena kao što je prikazano na slici 5.1.2.),



Slika 5.1.2. Potomci i roditelji

novonastali potomci bit će mnogo lošiji od svojih roditelja, što upozorava na potrebu ograničavanja mogućnosti sparivanja genotipski bitno različitih jedinki. Ako se ovakva situacija dogodi primjerice kod algoritma grupiranja, kreirani potomci nikako ne mogu svladati svoje roditelje i napredovati u sljedeću generaciju, čime nije pretrpljena bitna šteta. Stvarna šteta izražava se u usporenoj konvergenciji zbog učestalog stvaranja loših potomaka.

5.1.5. Algoritam s raspodjelom funkcije cilja (Fitness Sharing algoritam)

Algoritam s raspodjelom funkcije cilja najčešće je korišten i najuspješniji algoritam za stvaranje niša [45] [46]. Princip rada zasniva se na Hollandovu konceptu prema kojem pojedine jedinke zauzimaju određenu nišu do njenog maksimalnog kapaciteta. U trenutku kada niša dosegne svoj kapacitet, za pojedine jedinke postaje poželjno potražiti neku manje napučenu nišu. Naime, sve jedinke koje se nalaze unutar jedne niše prisiljene su dijeliti resurse koje ta niša pruža. Algoritam s raspodjelom funkcije cilja kao resurs koji se dijeli koristi funkciju cilja. Naprosto se svim jedinkama njihova originalna vrijednost umanji proporcionalno broju sličnih jedinki u populaciji, odnosno broju jedinki u određenoj niši. U praksi to izgleda ovako. Za početak je potrebno odrediti funkciju dijeljenja (*engl. sharing function*) između promatrane jedinke

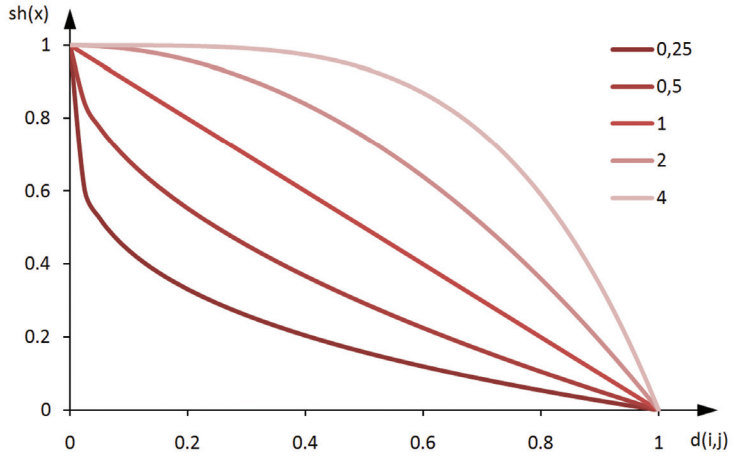
i svih ostalih jedinki u populaciji. Funkcija dijeljenja proporcionalna je genotipskoj ili fenotipskoj udaljenosti dviju jedinki $d(i,j)$. Udaljenost jedinki kreće se u rasponu od 0 do 1. Ako su dvije jedinke identične, tada im je $d(i,j) = 0$. Ako jedinke nisu identične, odnosno njihova različitost prelazi određen prag tolerancije, tada je vrijednost $d(i,j) = 1$, što znači da ne utječu jedna na drugu odnosno da ne pripadaju istoj niši. Sve ostale jedinke po sličnosti se nalaze unutar tih dvaju ekstrema.

$$f' = \frac{f(i)}{\sum_{j=1}^{n} sh\big(d(i,j)\big)} \tag{99}$$

Poznavanjem sličnosti dviju jedinki moguće je odrediti i utjecaj jedne jedinke na drugu. Najčešće je korištena funkcija dijeljenja

$$sh = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}}\right)^{\omega} & d < \sigma_{share} \\ 0 & u\ suprotnom \end{cases} \tag{100}$$

gdje σ_{share} predstavlja prag tolerancije sličnosti, a ω je konstanta za reguliranje oblika funkcije dijeljenja. Najčešće korištena vrijednost koeficijenta ω je 1 kada funkcija dijeljenja poprima oblik pravca. Na slici 5.1.3. prikazana je funkcija dijeljenja za nekoliko vrijednosti konstante ω.



Slika 5.1.3. Funkcija dijeljenja ovisno o konstanti ω

Odabir pojedinih koeficijenata ovisi o problemu koji se rješava kao i o metodi kojom se određuje sličnost pojedinih jedinki. Sličnost pojedinih jedinki najčešće se određuje genotipski iako postoje i fenotipska rješenja. Određivanje stupnja sličnosti jedinki kod složenijih problema katkad je teško; jer nije moguće jednostavno odrediti koliko su dva rješenja slična.

5.2. ULOGA NIŠE U PREDLOŽENOM ALGORITMU

Tradicionalan pristup kod algoritama s nišama svakako je stvaranje većeg broja više ili manje neovisnih grupa jedinki unutar populacije, ali to nije i jedini pristup. Kao što je navedeno u uvodu, cilj predloženog algoritma nije stvoriti i održavati veći broj niša u populaciji nego onemogućiti dominaciju pojedinih jedinki te spriječiti preuranjenu konvergenciju.

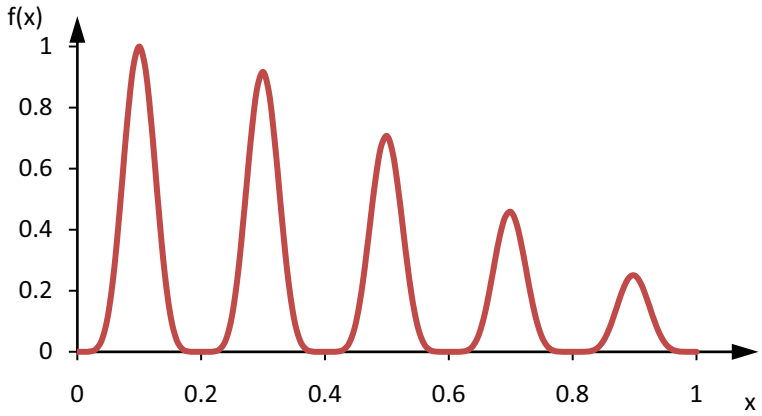
Naime, kod složenijih problema, a problem raspoređivanja to svakako jest, koji ujedno imaju i velik prostor pretraživanja, potrebno je kreirati algoritam koji istodobno ima istraživačke i eksploatacijske sposobnosti. U praksi to znači da algoritam mora moći pretražiti velik prostor definicije problema te na područjima na kojima detektira bolja rješenja provesti duže vrijeme te ga u većoj mjeri istražiti. Upravo su genetički algoritmi s nišama iznimno pogodni za navedeni način rada te za razliku od genetičkih algoritama bez niša održavaju veći broj grupa unutar populacije te time detaljnije istražuju pojedine dijelove prostora definicije problema. Bez obzira na kreiranje i održavanje grupa cilj algoritma u konačnici je pronalaženje samo jednog jedinog najboljeg rješenja.

5.3. SLIČNOST IZMEĐU JEDINKI U POPULACIJI

Pravilno određivanje sličnosti između jedinki najvažnije je za sve algoritme koji stvaraju niše. Kao što je već spomenuto, jedinke se mogu uspoređivati genotipski i fenotipski. Genotipska je usporedba jednostavnija i mnogo se češće rabi, ali nije idealna. Na primjeru vrlo jednostavne sinusoidne funkcije pokazat ćemo kakve probleme može izazvati »naivna« uporaba genotipske usporedbe jedinki.

Problem će biti prikazan na primjeru sinusoidne funkcije preuzete iz rada Goldberga i Richardsona iz 1987. godine [45] i glasi:

$$f(x) = e^{-2(ln2)\frac{x-0,1^2}{0,8}} sin^6(5\pi x) \tag{101}$$



Slika 5.3.1. Sinusoidna funkcija

Navedena je sinusoidna funkcija idealna za korištenje algoritma s nišama jer je iz slike 5.3.1. vidljivo da postoji pet maksimuma funkcije čije vrijednosti treba detektirati. Ako bi se primijenio algoritam bez niša, postojala bi realna opasnost da algoritam konvergira prema jednom od lokalnih optimuma funkcije, što svakako nije poželjno. Da bi se mogao primijeniti algoritam s nišama potrebno je odrediti sličnost između pojedinih jedinki. Za potrebe genetičkog algoritma kromosomi su kodirani u binarnom obliku dužine 30. Binarni kromosom dužine 30 može kodirati ukupno 2^{30} različitih jedinki, što je za potrebe ovog primjera više nego dovoljno. Kromosomi su kodirani na sljedeći način:

0,00	***0000***000000000000000000000000000	*0,5,*	***1000***000000000000000000000000000
0,05	***0000***110011001100110011001100110011	*0,55*	***1000***110011001100110011001100110011
0,10	***0001***100110011001100110011001100110	*0,60*	***1001***100110011001100110011001100110
0,15	***0010***011001100110011001100110011010	*0,65*	***1010***011001100110011001100110011010

0,20	***0011***00110011001100110011001101	*0,70*	***1011***00110011001100110011001101
0,25	***0100***00000000000000000000000000	*0,75*	***1100***00000000000000000000000000
0,30	***0100***11001100110011001100110011	*0,80*	***1100***11001100110011001100110011
0,35	***0101***10011001100110011001100110	*0,85*	***1101***10011001100110011001100110
0,40	***0110***01100110011001100110011010	*0,90*	***1110***01100110011001100110011010
0,45	***0111***00110011001100110011001101	*0,95*	***1111***00110011001100110011001101
		1,00	***1111***11111111111111111111111111

Iz tablice se lako može zaključiti da je za određivanje pripadnosti niši dovoljno poznavati samo prve četiri najznačajnije znamenke, a preostalih je 26 manje-više nebitno. Da je tome tako vidi se iz slike 5.3.1., gdje svi brojevi od 0 do 0,2 pripadaju prvoj niši. U binarnom zapisu svi kromosomi koji započinju s 0000, 0001 i 0010 pripadaju prvoj niši te jedinke koje u prva četiri gena imaju navedene vrijednosti utječu jedna na drugu, a sve ostale jedinke u populaciji nemaju utjecaja. Navedeni primjer vrlo je pojednostavljen, ali upozorava na bitno: od 30 gena samo prva četiri utječu na pripadnost niši. Tako primjerice kromosomi mogu pripadati dvjema potpuno različitim nišama iako imaju čak 29 identičnih gena.

*0**0**0110011001100110011001100110*
*0**1**0110011001100110011001100110*

Kod složenijih problema odabir četiriju ključnih gena nije jednostavan zadatak, pogotovo ako priroda problema nije poznata.

Rješenje problema pokušalo se pronaći u fenotipskoj usporedbi. Naime, kod fenotipske usporedbe genotip svake jedinke svodi se na nekoliko osnovnih osobina ili parametara koji služe za određivanje sličnosti pojedinih jedinki. S fenotipskom usporedbom čest je problem da dvije jedinke s bitno različitim genotipom mogu imati slične, pa i identične fenotipe. Također je moguće iz dva gotovo identična genotipa dobiti dva bitno različita fenotipa. Posebno je to izraženo kod permutacijskih problema, gdje je fenotipsko iskazivanje pojedinog gena pod velikim utjecajem ostalih gena u kromosomu. Preliminarna istraživanja vezana uz usporedbu jedinki za problem raspoređivanja koji se razmatra u ovom radu pokazala su iznimno loše rezultate kod fenotipske usporedbe. Detaljnijom analizom ustanovljeno je da je primjena aktivnog rasporeda najvažnije za označavanje jedinki kao izrazito sličnih (pripadnost istoj niši) iako one genotipski nisu slične.

U ovom radu predloženo je kompromisno rješenje u obliku Hamiltonove sličnosti, koje rabi genotipsku sličnost, ali bez određivanja ključnih gena.

5.4. HAMILTONOVA SLIČNOST (NOVI PRISTUP)

Za uspješnost svakoga genetičkog algoritma, najvažnije je očuvanje genetske raznolikosti unutar populacije. Uporaba niša svakako je jedna od najboljih metoda, ali kao što je već spomenuto, postoji problem određivanja sličnosti jedinki. U algoritmu predloženom u ovom radu koristi se tzv. Hamiltonova sličnost. Metoda je kao potpuno nov pristup uvedena u ovom radu, a ime je dobila po W. D. Hamiltonu, koji je u dva rada postavio matematičke osnove za određivanje vjerojatnosti pojavljivanja određenoga gena u populaciji. Prema njegovoj teoriji, svi geni koje jedna jedinka posjeduje u jednakom su omjeru dobiveni od obaju roditelja. To na prvi pogled izgleda neutemeljeno i neuvjerljivo, uzme li se u obzir da je broj gena koje pojedina jedinka nasljeđuje od svakog od roditelja direktno ovisan o mjestu prekida prilikom primjene operatora križanja. I zaista, teorijski je moguće da novonastala jedinka naslijedi sve gene od samo jednog roditelja, ali samo kada se koristi haploidni kromosom. Hamiltonova teorija nastala je promatranjem organizama na višem stupnju razvoja koji svi, bez iznimke, imaju diploidne kromosome. Kod organizama s diploidnim kromosomom na svakoj poziciji unutar kromosoma postoje dva komplementarna gena od kojih se samo jedan iskazuje, odnosno utječe na fenotip organizma. Kako funkcionira diploidni kromosom u odnosu na haploidni te koje su mu prednosti i nedostaci detaljnije će biti objašnjeno u 6. poglavlju.

Budući da diploidni kromosom ima na svakoj poziciji po dva gena, prilikom stvaranja nove jedinke svaki od roditelja daje samo po jedan gen iz svakoga para u kromosomu. Kod diploidnog organizma samo su spolne stanice haploidne jer će njihovim spajanjem nastati diploidan organizam. Primjerice, čovjek ima 46 kromosoma, a njegove spolne stanice imaju samo 23 kromosoma (polovica ukupnog broja kromosoma). Spajanjem tih istih spolnih stanica nastaje novi organizam s potpunim diploidnim kromosomom. Iz svega navedenog lako se može zaključiti da diploidni organizmi točno pola svojih gena nasljeđuju od svakog od roditelja. Upravo ta pravilnost okosnica je teorije sličnosti predložene u ovom doktorskom radu.

Zna li se koliko je gena svaka jedinka naslijedila od svakog roditelja, a to je uvijek 50%, tada se može i bez razmatranja fenotipa jedinke odrediti sličnost između pojedinih jedinki. Usporedimo za početak oca i sina. Budući da sin ima dva roditelja, jednostavno je zaključiti da sin i otac imaju 50% identičnih gena, naravno ako zanemarimo neke rijetke varijante poput činjenice da pojedini geni mogu nastati ili nestati mutacijom ili da je sin od majke naslijedio neki gen koji otac ima, a nije ga prenio na sina. Uz navedena pojednostavnjenja bez gubitka općenitosti možemo smatrati da je broj zajedničkih gena uvijek 50%. Ako je poznat odnos između roditelja i djece, moguće je prema Hamiltonovoj teoriji odrediti i odnose između svih srodnika te postotak zajedničkih gena odnosno postotak sličnosti. Prvo je pitanje koje se nameće samo od sebe kakav je odnos djeda i unuka. Odgovor na ovo pitanje vrlo je jednostavan. Ako je djed na svog sina prenio 50% svojih gena, a on na unuka daljnjih 50%, tada je očito da unuk posjeduje 25% djedovih gena. Poopćivanjem ovog primjera moguće je odrediti zajednički broj gena za svaku kombinaciju srodnika koji su u izravnom srodstvu.

$$zajednički\ geni, \% = \left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\ldots\left(\frac{1}{2}\right) = \left(\frac{1}{2}\right)^n \qquad (102)$$

gdje *n* predstavlja generacijsku razliku između dva srodnika. U primjeru djeda i unuka, generacijska razlika je 2, stoga vrijedi

$$zajednički\ geni, \% = \left(\frac{1}{2}\right)^2 = 0{,}25\ (25\%) \qquad (103)$$

Drugo bitno pitanje vezano je uz neizravne srodnike, kao što su brat i sestra ili ujak i nećak. Budući da ne postoji izravna rodbinska veza, postavlja se pitanje može li se odrediti broj, odnosno postotak zajedničkih gena između brata i sestre, ili općenito između bilo koja dva pripadnika jedne obitelji. Odgovor na ovo pitanje također je potvrdan. Za početak potrebno je razmotriti najjednostavniji slučaj, npr. odnos dviju sestara. Da bi se njih dvije moglo smatrati sestrama, moraju imati barem jednoga zajedničkog roditelja. Kao što je već napomenuto, generacijska udaljenost između pojedinih jedinki izravno određuje postotak zajedničkih gena. Kako između dvije sestre ne postoji generacijska udaljenost, nego obje pripadaju istoj generaciji, stječe se dojam da bi prema Hamiltonovoj teoriji postotak zajedničkih gena trebao biti 100%. To nije točno. Naime, sestre nisu izravni srodnici te se na njih mora primijeniti

Hamiltonova teorija u općenitijem obliku. Naime, generacijska udaljenost između dviju sestara određuje se tako da se zbroje generacijske udaljenosti svake sestre od njihova najbližeg zajedničkog pretka. U konkretnom slučaju sestre imaju barem jednog zajedničkog roditelja. Bitno je da se cijelo vrijeme spominje »barem« jedan roditelj jer postojanje dvaju zajedničkih roditelja, kako ćemo pokazati, bitno utječe na broj zajedničkih gena. U konkretnom je slučaju generacijska udaljenost prve sestre od zajedničkog roditelja 1. Isto vrijedi i za drugu sestru, što znači da se, zbroje li se njihove generacijske udaljenosti od roditelja, dobiva njihova međusobna udaljenost koja iznosi 2.

$$zajednički\ geni, \% = \left(\frac{1}{2}\right)^2 = 0{,}25\ (25\%) \qquad (104)$$

Iz svega navedenog može se zaključiti da dvije sestre koje imaju samo jednoga zajedničkog roditelja imaju 25% zajedničkih gena, što se u potpunosti podudara s brojem zajedničkih gena koje imaju djed i unuk. Hamiltonova teorija zorno pokazuje da dvije sestre koje imaju samo jednoga zajedničkog roditelja nisu međusobno genetički sličnije od unuka i djeda. Naravno, nameće se pitanje što se događa, kada sestre imaju dva zajednička roditelja. Dva zajednička roditelja omogućuju sestrama nasljeđivanje istovjetnih gena preko dvaju predaka iz iste generacije, u konkretnom slučaju dvaju roditelja. U tom slučaju formula za broj zajedničkih gena glasi

$$zajednički\ geni, \% = n_a \cdot \left(\frac{1}{2}\right)^n \qquad (105)$$

gdje je n_a broj zajedničkih predaka u istoj generaciji te može poprimiti vrijednosti 1 ili 2. Ako dvije jedinke imaju samo jednoga zajedničkog pretka, tada se nova formula ne razlikuje od prethodne. S druge strane, ako postoje dva zajednička pretka u istoj generaciji, kao u primjeru sestara s dva zajednička roditelja, n_a poprima vrijednost 2.

$$zajednički\ geni, \% = 2 \cdot \left(\frac{1}{2}\right)^2 = 0{,}5\ (50\%) \qquad (106)$$

Pojednostavljeno to znači da dvije sestre s dva zajednička roditelja imaju dvostruko više zajedničkih gena nego kada imaju samo jednoga zajedničkog roditelja. Nadalje, dvije sestre jednako su genetički slične kao i otac i kći ili

majka i kći, što dovodi do vrlo zanimljivog zaključka. S evolucijskoga gledišta roditelju je njegovo dijete jednako slično, a time i bitno kao i njegov brat ili sestra ako imaju dva zajednička roditelja.

Idući primjer odnosi se na sličnost ujaka i nećaka. Princip je isti. Potrebno je odrediti prvoga zajedničkog pretka te generacijsku udaljenost između promatranih jedinki i zajedničkoga pretka. Zajednički je predak unukov djed ili ujakov otac. Generacijska udaljenost između unuka i djeda jasna je i iznosi 2, a također je poznata generacijska udaljenost između oca (djed) i sina (ujak) te iznosi 1. Uz pretpostavku da je djed jedini predak, postotak zajedničkih gena iznosi

$$zajednički\ geni, \% = 1 \cdot \left(\frac{1}{2}\right)^3 = 0{,}125\ (12{,}5\%) \qquad (107)$$

Ako je pak i baka zajednička, tada je i broj zajedničkih gena dva puta veći i iznosi

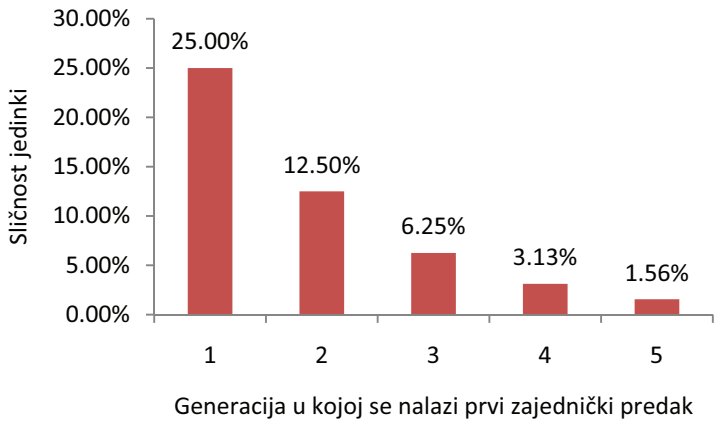
$$zajednički\ geni, \% = 2 \cdot \left(\frac{1}{2}\right)^3 = 0{,}25\ (25\%) \qquad (108)$$

Na ovaj način može se odrediti postotak zajedničkih gena, a time i genotipska sličnost svih članova populacije. U genetičkim algoritmima nije uobičajeno da veći broj jedinki ima dva zajednička roditelja. Dapače, takva je situacija više iznimka nego pravilo.

5.5. ALGORITAM S RASPODJELOM FUNKCIJE CILJA I HAMILTONOVA SLIČNOST

Hamiltonova je sličnost uvedena u genetički algoritam radi definiranja količine pojedinih gena u populaciji. Naime, gene koji se prečesto pojavljuju u populaciji treba na neki način ograničiti. Kako je algoritam s raspodjelom funkcije cilja vrlo dobra metoda za ograničavanje dominacije pojedinih jedinki, a time i njihovih gena, u ovom je radu algoritam s raspodjelom funkcije cilja korišten kao osnova predložena modela, što automatski znači da je potrebno odrediti radijus utjecaja pojedinih jedinki, odnosno definirati prag sličnosti nakon

kojeg jedinke više nemaju utjecaj jedna na drugu. Uporabom Hamiltonove sličnosti problem je definiran na nešto drukčiji način. Naime, ako su poznati svi članovi populacije u posljednjih pet generacija, tada se može odrediti za svakog člana trenutačne populacije srodstvo s bilo kojim drugim članom do petoga koljena. Algoritam ne postavlja ograničenje u broju generacija koje se ispituju. Provjera srodstva do pete generacije odabrana je iz praktičnih razloga jer se s povećanjem broja generacija eksponencijalno povećava broj predaka koje treba ispitati kako bi se ustanovilo postojanje srodstva između pojedinih jedinki. K tome, broj zajedničkih gena koje dvije jedinke posjeduju smanjuje se na samo 1,56% ako imaju prvog zajedničkog pretka tek u petom koljenu. Iz navedenog se može zaključiti da algoritam smatra jedinke koje imaju prvoga zajedničkog pretka iza pete generacije potpuno genetički različitim, odnosno da je utjecaj jedne jedinke na drugu jednak nuli. Kao što se vidi iz slike 5.5.1., s povećanjem broja generacija do prvoga zajedničkog srodnika smanjuje se broj zajedničkih gena.



Slika 5.5.1. Postotak zajedničkih gena

Dijagram također prikazuje način stvaranja niša s relativno velikim radijusom utjecaja. Naime, na svaku pojedinu jedinku utječu gotovo sve jedinke u populaciji, ili preciznije, one koje imaju više od 1,5 posto zajedničkih gena. S druge strane, želi li se kreirati veći broj manjih niša, potrebno je smanjiti radijus utjecaja pojedinih jedinki. To se postiže tako da se ne uzimaju u obzir

jedinke koje imaju zajedničkoga pretka do petog koljena već se provjera srodnosti prekida npr. u drugoj generaciji. Tako se bitno smanjuje broj jedinki koje pripadaju istoj niši. Ovdje je bitno istaknuti da pojedine jedinke vrlo često pripadaju većem broju niša istodobno, odnosno niše se preklapaju.

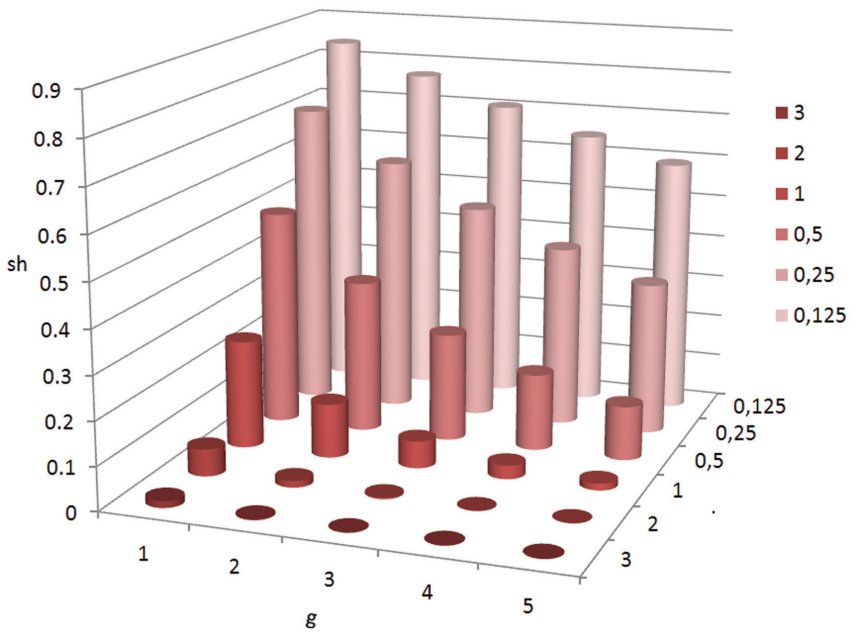
Funkcija dijeljenja smanjuje se povećanjem broja generacija do prvoga zajedničkog pretka.

$$sh = \begin{cases} \left(\left(\frac{1}{2}\right)^{g+1}\right)^{\omega} & g < \sigma(razina\ srodnosti) \\ 0 & u\ suprotnom \end{cases} \qquad (109)$$

Razina srodnosti predstavlja broj prethodnih generacija u kojima se provjerava srodnost dviju jedinki. Ako se u zadanom broju generacija ne pronađe zajednički predak, smatra se da jedinke nisu u srodstvu te da je

$$sh\big(g(i,j)\big) = 0 \qquad (110)$$

*



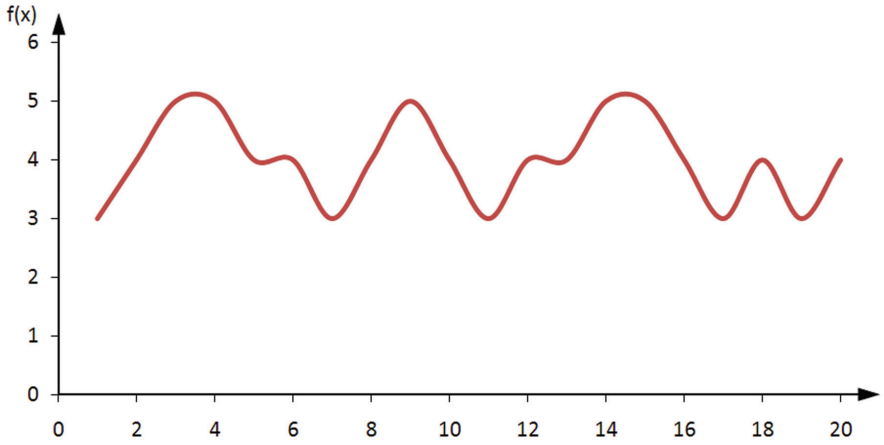
Slika 5.5.2. Funkcija dijeljenja

Parametar ω predstavlja korekciju utjecaja pojedinih jedinki. Naime, ako se za ω odabere 1, što je uobičajena varijanta, tada je prema slici 5.5.1. posve jasno koliki je utjecaj pojedinih jedinki ovisno o generaciji u kojoj se nalazi njihov prvi zajednički predak. Parametar ω služi za modifikaciju utjecaja. Naime, ako se za ω odabere broj manji od 1, tada jedinke unutar niše više utječu jedna na drugu. Obrnuta je situacija ako se za ω odabere broj veći od 1. Slika 5.5.2. prikazuje funkciju dijeljenja ako se za ω odaberu 3, 2, 1, 0.5, 0.25 i 0.125. Zbog utjecaja jedinki na pripadnike iste niše vrijednost se svake pojedine jedinke smanjuje

$$f' = \frac{f(i)}{\sum_{j=1}^{n} sh\big(g(i,j)\big)} \tag{111}$$

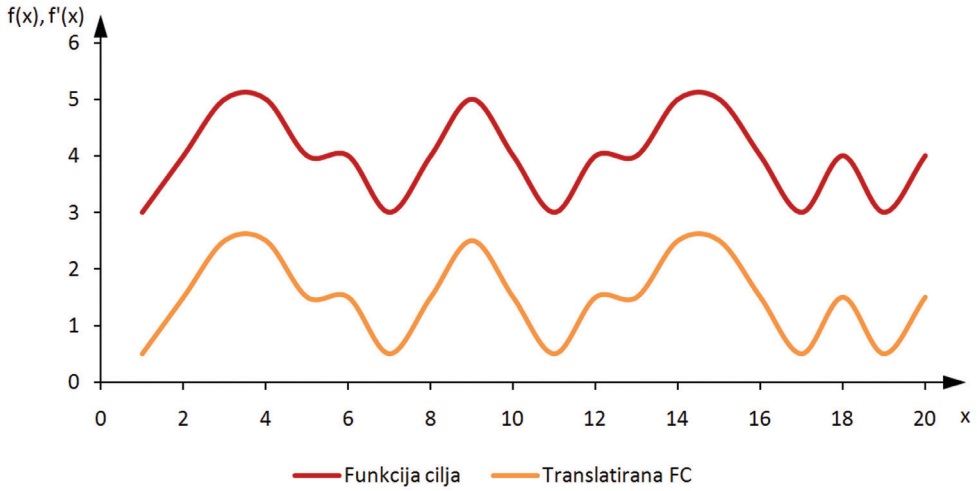
5.6. ULOGA SKALIRANJA VRIJEDNOSTI FUNKCIJE CILJA

Skaliranje vrijednosti funkcije cilja iznimno je bitno prilikom stvaranja niša. Može se također reći da je za većinu funkcija cilja skaliranje nužno za uspješno stvaranje niša uporabom algoritma s raspodjelom funkcije cilja. Radi ilustracije problema koristit će se pojednostavljeni primjer čija je funkcija cilja prikazana na slici 5.6.1.



Slika 5.6.1. Funkcija cilja (primjer)

Funkcija cilja može poprimiti vrijednosti između 3 i 5,1. Zbog uporabe algoritma s raspodjelom funkcije cilja, funkcija cilja u ovom obliku nije primjenjiva. Naime, funkcija cilja u točki 9 poprima vrijednost 5. Ako se na istoj poziciji pojave dvije identične jedinke, d(i, j) = 0, tada je funkcija dijeljenja bez obzira na odabrani ω najmanje 2 (udaljenost od identične jedinke pribrojena udaljenosti od same sebe). Tada je korigirana funkcija cilja za odabranu jedinku manja od 5/2, što je manje od najmanje nekorigirane vrijednosti funkcije cilja. Takva situacija bitno smanjuje mogućnost odabira promatrane jedinke u grupu roditelja za sljedeću generaciju, što svakako nije poželjno. Da bi se to izbjeglo potrebno je provesti skaliranje vrijednosti funkcije cilja tako da se cijela funkcija cilja u nepromijenjenu obliku spusti niže (translatira) prema osi apscise. U konkretnom slučaju funkcija cilja smanjena je za 2. Na slici 5.6.2. korigirana funkcija cilja prikazana je žutom bojom. Analizu utjecaja veličine linearnog skaliranje vrijednosti funkcije cilja na sposobnost kreiranja niša pokazao je Rasmus K. Ursem u radu *When Sharing Fails*. [47].



Slika 5.6.2. Korigirana funkcija cilja (translacija)

Funkcija cilja korigirana na ovaj način može uspješno poslužiti za stvaranje niša jer kod pojave identičnih ili vrlo sličnih jedinki, koje evidentno pripadaju istoj niši, nema korekcije funkcije cilja ispod vrijednosti najlošije nekorigirane vrijednosti funkcije cilja. U konkretnom slučaju, ako bi se na poziciji 9 nalazile

tri identične jedinke, tada bi njihova korigirana funkcija cilja iznosila oko 0,8, što je još iznad 0,5 koliko iznosi najmanja vrijednost funkcije cilja. Tek pri pojavi pet identičnih ili vrlo sličnih jedinki niša bi se našla na rubu kapaciteta.

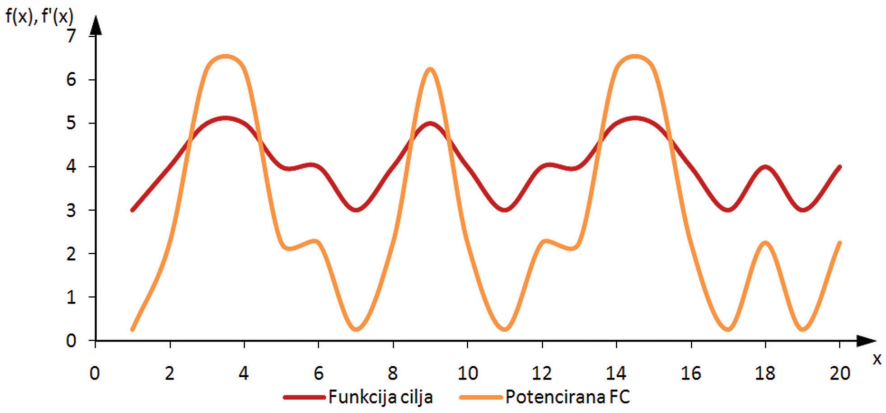
Kapacitet niše dodatno se može povećati ako se korigirana funkcija cilja dodatno spusti tako da se minimum funkcije postavi na nulu. Takvim se postupkom jedinkama s najmanjom vrijednosti funkcije cilja posve onemogućuje odabir u grupu roditelja, što znači bitan gubitak genetske raznolikosti. Alternativno je rješenje uporaba eksponencijalne funkcije tako da se postigne veća razlika između najlošijih i najboljih članova populacije.

$$f' = (f - \delta)^{\varphi} \tag{112}$$

Uz uvjet da je funkcija cilja na cijelom svojem prostoru pozitivna, koeficijent δ može poprimiti vrijednosti između 0 i minimuma funkcije cilja. S druge strane, koeficijent φ nema bitna ograničenja osim da je veći od 1. Na slici 5.6.3., prikazana je korekcija funkcije cilja prema formuli

$$f' = (f - 2{,}5)^2 \tag{113}$$

na osnovi koje nastaje korigirana funkcija cilja s vrlo izraženim nišama.



Slika 5.6.3. Korigirana funkcija cilja (potenciranje)

Jasno da postupak nije bez nedostataka. Naime, u nekih funkcija cilja čije je rješenje u veoma uskom prostoru, odnosno pripadajuća je niša vrlo malena, može doći do preuranjene konvergencije prema jednom od lokalnih optimuma. Rješenje je tog problema u progresivnom povećanju koeficijenta φ, što znači da se funkcija cilja postupno mijenja kako evolucija odmiče, odnosno niše postaju sve izraženije. Nova generacijski izmjenjiva funkcija cilja glasi

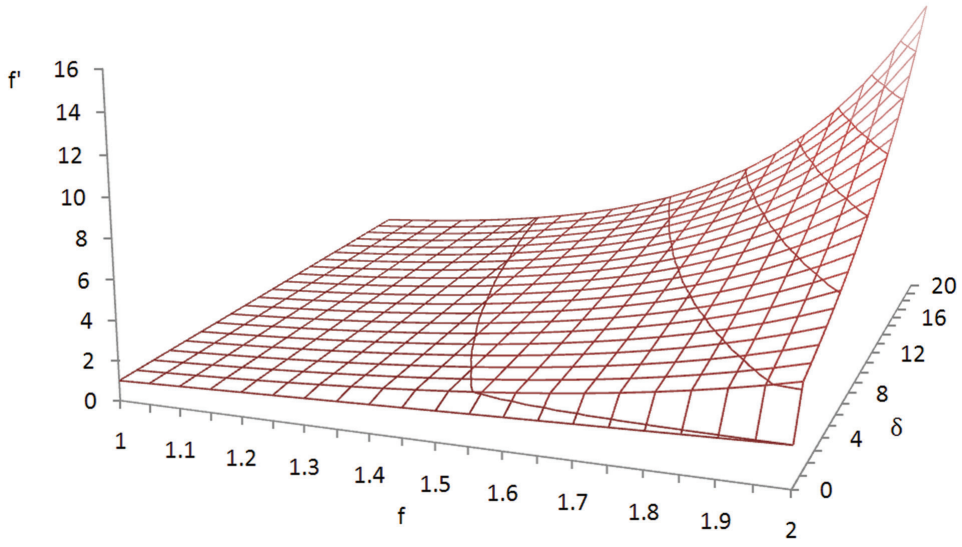
$$f' = (f - \delta)^{\frac{G_{max}}{G_{gr}} + \varphi} \tag{114}$$

gdje je G_{max} trenutačna generacija, a G_{gr} granična (tranzicijska) generacija evolucije nakon koje niše postaju izražene. Prema navedenoj formuli, ako je koeficijent φ jednak 0 u nultoj (početnoj) generaciji, funkcija je cilja na cjelokupnom prostoru pretraživanja jednaka 1, što znači da nema ni jedne niše (postupak selekcije posve je stohastičan). Kako evolucija odmiče, potenciranje se povećava dok se ne dosegne granična generacija kada je korigirana funkcija cilja jednaka stvarnoj funkciji cilja.

$$f' = (f - \delta)^{\frac{G_{max}}{G_{gr}}} = (f - \delta)^1 = f - \delta \tag{115}$$

U svim generacijama poslije granične generacije G_{gr} niše postaju sve izraženije. Pravilnim odabirom G_{gr} moguće je fino vremenski prilagoditi kapacitet niše.

Radi boljeg rada algoritma potrebno je napraviti i veću razliku između globalnog optimuma i bliskih lokalnih optimuma kako bi se dodatno izbjeglo zadržavanje algoritma u području lokalnih optimuma. Potenciranje funkcije cilja povećava razliku između lokalnih optimuma, ali katkad to nije dovoljno. Ako su lokalni optimumi po svojoj vrijednosti zaista blizu globalnom optimumu, tada je potrebno uvesti osim vremenski (generacijski) promjenjivog potenciranja i potenciranje ovisno o vrijednosti funkcije cilja. Prema ovom konceptu vrijednost eksponenta potenciranja to je veća što je funkcija cilja veća te je i ovaj put od 1 do 2. Tako je primjerice funkcija cilja najboljeg člana generacije potencirana s 2, a najlošijeg s 1. Na slici 5.6.4. prikazana je promjena funkcije cilja ovisno o vrijednosti nemodificirane funkcije cilja i vremenu (generacije).



Slika 5.6.4. Modifikacija funkcije cilja

Kao što se iz slike zorno može vidjeti, nemodificirana funkcija cilja iz nulte generacije vrlo je jednostavna. Riječ je o običnoj linearnoj funkciji koja odmah prelazi u eksponencijalnu funkciju. Nova modificirana funkcija cilja glasi

$$f' = (f - \delta)^{\left(\frac{G_{max}}{G_{gr}}\right)\left(\frac{f - f_{min}}{f_{max} - f_{min}}\right) + \varphi} \qquad (116)$$

Potenciranje funkcije cilja ovisno o njezinoj vrijednosti ima još jednu bitnu funkciju. Pred kraj algoritma, kada u populaciji ostaju samo relativno dobro prilagođene jedinke (jedinke s relativno visokim vrijednostima nemodificirane funkcije cilja), postoji opasnost da se dobre jedinke ne mogu istaknuti te može doći do nasumična odabira roditelja. Ako se koristi linearni odabir vrijednosti potencije funkcije cilja, tada će se jedinka i s malo većom vrijednošću funkcije cilja izboriti za nadprosječan broj ulazaka u skupinu roditelja. Takva situacija pred kraj evolucije dovodi do efikasna uništavanja svih niša osim jedne te njene potpune eksploatacije.

6. DIPLOIDNI KROMOSOM

Gotovo sve životinjske vrste, uz nekoliko iznimki, imaju diploidne kromosome, a haploidni se kromosom u pravilu pronalazi kod vrsta na nižem stupnju razvoja. Neke vrste biljaka imaju i poliploidne kromosome, što je korak dalje u odnosu na jednostavni haploidni kromosom.

Haploidni kromosom ima neke bitne nedostatke te se u evolucijskom smislu pokazao nedovoljno dobrim. Naime, haploidni kromosom koji za svaku osobinu ima samo po jedan gen ne može na duži rok obraniti ili sačuvati slabije gene ili slabije kombinacije gena, već ih postupno gubi te izravno osiromašuje genetsku raznolikost populacije. U takvim odnosima razvoj pojedinih vrsta bitno je usporen jer se uvelike oslanja na mutaciju za koju je poznato da je vrlo rijetka. Vrste koje posjeduju haploidni kromosom ne mogu se dovoljno brzo prilagoditi promjenama u okolišu.

S druge strane, diploidni kromosom za razliku od haploidnog ima na svakoj poziciji dva gena. To u praksi znači da za svaku osobinu postoje dvije varijante od kojih se samo jedna iskazuje u obliku fenotipa. Još je Gregor Mendel u svom epohalnom radu [48] pokazao na primjeru graška koliki utjecaj na evoluciju i odabir vrste ima diploidni kromosom, odnosno da primjerci koji fenotipski nemaju izraženu neku osobinu mogu stvoriti potomke koji tu osobinu imaju vidljivo izraženu. Takvo nešto posve je isključeno kod vrsta s haploidim kromosomom.

Školski primjer brze prilagodbe vrste na promjenu okoliša predstavlja brezin moljac (*Biston betularia*) u Engleskoj u razdoblju od stotinu godina industrijske revolucije. Naime, brezin je moljac u velikoj većini slučajeva svijetlo obojen (*typica*), pa se izvrsno prikriva na svijetloj kori drveta i lišajevima koji na njemu rastu. S druge strane poznata je i varijanta koja je posve crne boje (*carbonaria*).

Crna varijanta dugo nije bila poznata jer se pojavljivala u veoma malom broju, ispod 0,01%. Broj je bio malen jednostavno zato što je moljac bio iznimno uočljiv na bijeloj brezinoj kori te su ga njegovi prirodni neprijatelji znatno lakše uočavali. Situacija se iz temelja promijenila na početku industrijske revolucije u Engleskoj. Zbog velike uporabe ugljena za pokretanje sve većeg broja strojeva atmosfera se u okolici velikih industrijskih gradova poput Manchestera osjetno pogoršala. Zbog utjecaja sumporovog dioksida lišajevi su odumirali, a brezina prirodno bijela boja potamnjela je zbog visoke koncen-

tracije čađe u zraku. U takvim okolnostima pretežno svijetao moljac (*typica*) postao je lak piljen svojim prirodnim neprijateljima. U isto vrijeme naglo se povećao postotak crne varijante moljca (*carbonaria*) zbog bolje prilagođenosti okolišu te je crna vrsta do 1895. godine zauzela 98% populacije brezinih moljaca.

Slika 5.6.1. Brezin moljac (dvije varijante)

Engleska danas zbog bolje zakonske regulative nije zagađena kao na počecima industrijske revolucije te pojava crnih moljaca u populaciji više nije toliko učestala. Ponovni pomak populacije brezina moljca prema svjetlijoj varijanti zorno upozorava na veliku prilagodljivost vrste na promjenu okoliša. Takva brza prilagodba ne bi bila moguća da je brezin moljac imao haploidni kromosom.

Genetički algoritmi u pravilu, se služe haploidnim kromosomom, zbog njegove jednostavnosti. Razlog je u tome da je potrebno riješiti iznimno složen problem dominacije. Naime, prevođenje genotipa jedinke s haploidnim kromosomom u odgovarajući fenotip znatno je jednostavnije u odnosu na jedinke koje imaju diploidni kromosom. Problem je u tome da se u kromosomu za svaku osobinu nalaze po dva gena od kojih će se samo jedan iskazati. Iskazivanje pojedinoga gena definirano je odnosom gena na istoj poziciji u kromosomu. Navedeni odnos naziva se dominacija te ima izniman utjecaj na proces evolucije.

U već spomenutom radu Gregora Mendela dominacija je uočena i opisana na primjeru graška, odnosno na obliku zrna kao na jednoj od njegovih osobina. Zrno može poprimiti dva oblika, što upozorava na to da na poziciji u kro-

mosomu koja je odgovorna za oblik zrna mogu stajati samo dvije vrijednost. Takav binarni sustav dominacije veoma je važan i za genetičke algoritme koji se zasnivaju na binarnoj prezentaciji kromosoma, jer su prve varijante dominacije razvijene upravo za tu varijantu. U prirodi postoje osobine koje ne poprimaju samo dva stanja. Tako primjerice gen odgovoran za tip krvne grupe može imati jednu od tri vrijednosti A, B i 0. A i B varijante su dominantne u odnosu na varijantu 0, što znači da će diploidni organizam s genima A0 i AA biti fenotipski iskazan kao krvna grupa tipa A. Identična je situacija i s varijantama B0 i BB koje se iskazuju kao krvna grupa B. U oba slučaja nije moguće fenotipski razlikovati jedinke. Budući da je varijanta gena 0 recesivna, može se iskazati samo ako su oba gena 0. Na kraju preostaje definirati varijantu kada je jedan gen A, a drugi B. Budući da su obje varijante dominantne, fenotipski će se obje i skazati te će krvna grupa biti AB. Opisana situacija naziva se ko-dominacija i u prirodi je nije jednostavno detektirati. Kad je posrijedi genetički algoritam predložen u ovom radu, dominacija je definirana za gene koji mogu imati više od dvije vrijednosti na istoj poziciji. Definicija i detaljno objašnjenje dominacije dani su u poglavlju 6.3. Bitno je naglasiti da u predloženom modelu dominacije nema ko-dominacije kao kod primjera krvne grupe.

6.1. USPOREDBA HAPLOIDNOG I DIPLOIDNOG KROMOSOMA

Primarni je cilj diploidnog kromosoma zadržavanje u populaciji slabijih gena ili slabije skupine gena koji u određenom trenutku evolucije nisu od velike koristi, ali bi u doglednoj budućnosti mogli postati. Takav princip omogućuje zadržavanje veće genetske raznolikosti populacije, a samim time povećava vjerojatnost pronalaska globalnog optimuma. Tvrdnju da diploidni kromosom zadržava veću genetsku raznolikost ispitali su i dokazali Goldberg i Smith [49] na problemu prebrojavanja jedinica 1987. godine. Na istovjetnom je problemu u poglavlju 3.5.3. prikazan utjecaj mutacije na pronalaženje globalnog optimuma.

U navedenom problemu kromosomi se sastoje samo od nula i jedinica, s time da nule doprinose vrijednosti kromosoma sa f_0, a jedinice pridonose sa f_1. Ukupan broj gena (nula i jedinica) u populaciji iznosi M. Broj nula u populaciji iznosi m_0, a broj je jedinica m_1.

Udio nula u populaciji može se u svakom trenutku jednostavno odrediti:

$$P(t) = \frac{m_0(t)}{m_0(t) + m_1(t)} = \frac{m_0(t)}{M} \tag{117}$$

Isto vrijedi i za udio jedinica u populaciji

$$Q(t) = \frac{m_1(t)}{m_0(t) + m_1(t)} = \frac{m_1(t)}{M} = 1 - P(t) \tag{118}$$

Budući da je vrijednost nula manja od vrijednosti jedinica, faktor r je uvijek veći od 1:

$$r = \frac{f_1}{f_0} > 1 \tag{119}$$

Očekuje se da će se tijekom evolucije broj nula smanjivati u korist broja jedinica. Ako diploidni kromosom ima tendenciju obraniti slabije gene, u konkretnom slučaju nule, tada bi s vremenom kod diploidnog kromosoma trebalo ostati više nula nego kod haploidnog kromosoma.

Tijekom analize koristit će se pojednostavljeni genetički algoritam koji se koristi samo operatorima selekcije i mutacije. Kod takvog genetičkog algoritma broj nula odnosno jedinica mijenja se tijekom vremena prema sljedećem pravilu:

$$m_0(t+1) = \frac{f_0 m_0(t)}{\bar{f}(t)}(1 - p_m) + \frac{f_1 m_1(t)}{\bar{f}(t)} p_m \tag{120}$$

$$m_1(t+1) = \frac{f_1 m_1(t)}{\bar{f}(t)}(1 - p_m) + \frac{f_0 m_0(t)}{\bar{f}(t)} p_m \tag{121}$$

Prema navedenim jednadžbama broj jedinica u sljedećoj generaciji proporcionalan je broju jedinica koje su prošle postupak selekcije, umanjen za broj jedinica koje su mutirale i time postale nule. Tom broju potrebno je pridodati broj nula koje su mutirale i postale jedinice. Isto se objašnjenje može primijeniti i na broj nula u sljedećoj generaciji.

Iz navedenih formula jednostavno se može odrediti udio nula u sljedećoj generaciji:

$$P(t+1) = \frac{m_0(t+1)}{m_0(t+1) + m_1(t+1)} \tag{122}$$

iz čega proizlazi:

$$P(t+1) = \\ = \frac{\frac{f_0 m_0(t)}{\bar{f}(t)}(1-p_m) + \frac{f_1 m_1(0)}{\bar{f}(t)} p_m}{\frac{f_0 m_0(t)}{\bar{f}(t)}(1-p_m) + \frac{f_1 m_1}{\bar{f}(t)} p_m + \frac{f_1 m_1}{\bar{f}(t)}(1-p_m) + \frac{f_0 m_0(t)}{\bar{f}(t)} p_m} \tag{123}$$

Uvođenjem r dobiva se:

$$P(t+1) = \\ = \frac{m_0(t)(1-p_m) + r m_1(t) p_m}{m_0(t)(1-p_m) + r m_1(t) p_m + r m_1(t)(1-p_m) + m_0(t) p_m} \tag{124}$$

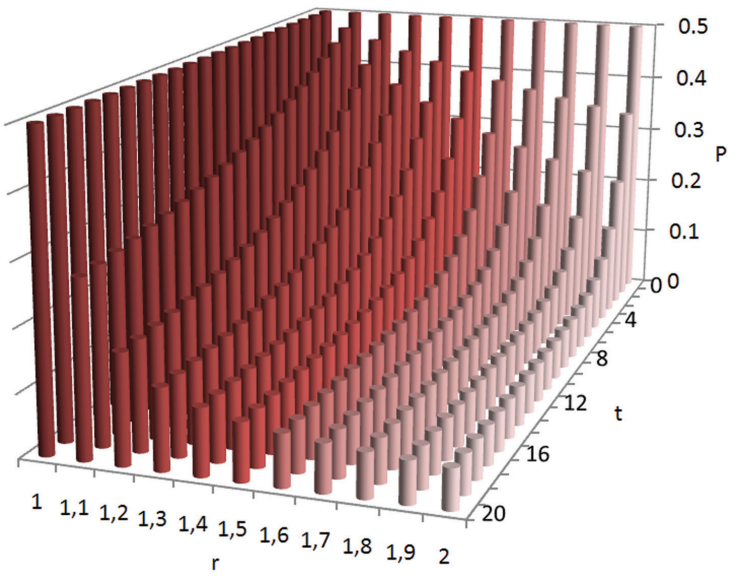
Daljnjim uvođenjem omjera nula i jedinica u trenutku (t) dobiva se:

$$P(t+1) = \frac{P(t)(1-p_m) + rQ(t)p_m}{P(t)(1-p_m) + rQ(t)p_m + rQ(t)(1-p_m) + P(t)p_m} \tag{125}$$

iz čega slijedi:

$$P(t+1) = \frac{P(t)(1-p_m) + r\big(1-P(t)\big)p_m}{P(t) + r\big(1-P(t)\big)} \tag{126}$$

Dobivena funkcija pokazuje promjenu broja nula u populaciji iz generacije u generaciju ovisno o r i koeficijentu mutacije.



Slika 6.1.1. Brzina gubitka nula (slabiji gen) za koeficijent mutacije 0,0033 – haploidni kromosom

Kao što se može vidjeti iz slike 6.1.1., slabiji gen, u konkretnom slučaju nula, s vremenom nestaje iz populacije dok ne dosegne određenu konstantnu vrijednost koja izravno ovisi o koeficijentu mutacije p_m. Ako bi koeficijent mutacije bio postavljen na nulu, tada bi jedinice u potpunosti preuzele populaciju, odnosno u populaciji ne bi ostala niti jedna nula. S druge strane, ako bi koeficijent mutacije bio 1 (100%), udio nula u populaciji bi se uz manje oscilacije kretao oko 0,5 (50%). Iz navedene analize može se zaključiti da se haploidni kromosom vrlo brzo rješava slabijih gena, pogotovo kada je r velik, što dovodi do brzog gubitka genetske raznolikosti populacije. Ponašanje diploidnog kromosoma trebalo bi barem intuitivno biti bolje u pogledu zadržavanja slabijih gena odnosno nula. Razlog je u tome što se kombinacija gena 01 iskazuje kao jedinica te tako uspijeva obraniti nulu koja opstaje u populaciji.

Problem je postavljen na identičan način kao i kod haploidnog kromosoma. Kromosom se sastoji od nula i jedinica s doprinosima vrijednosti kromosoma od f_0 i f_1. Očekivani broj pojavljivanja dviju nula u kromosomu iznosi

$$Z(00) = \langle broj\ parova \rangle \times P(00) \tag{127}$$

Broj parova gena u diploidnom kromosomu uvijek je jednak polovici ukupnog broja gena $M/2$ tako da je očekivani broj parova nula:

$$Z(00) = \frac{M}{2}\frac{m_0(m_0-1)}{M(M-1)} \tag{128}$$

što približno iznosi

$$Z(00) \approx \frac{m_0^2}{2M} \tag{129}$$

Na identičan način očekivani broj parova jedinica iznosi

$$Z(11) \approx \frac{m_1^2}{2M} \tag{130}$$

Kad su poznate vrijednosti očekivanih pojavljivanja parova nula i jedinica potrebno je odrediti broj očekivanih pojavljivanja parova 01 i 10. Kako su to jedine preostale kombinacije parova, njihov očekivani broj pojavljivanja iznosi

$$Z(01\ ili\ 10) \approx \frac{M}{2} - \frac{m_0^2}{2M} - \frac{m_1^2}{2M} = \frac{M^2 - m_0^2 - m_1^2}{2M} \tag{131}$$

Znajući da se kombinacija 01 i 10 uvijek iskazuju kao 1 jer je jedinica dominantna vrijednost, broj nula i jedinica u sljedećoj generaciji može se definirati kao

$$\begin{aligned} m_0(t+1) = {} & 2\frac{f_0}{\bar{f}(t)}\frac{m_0^2}{2M}(1-p_m) + 2\frac{f_1}{\bar{f}(t)}\frac{m_1^2}{2M}p_m + \\ & + \frac{f_1}{\bar{f}(t)}\frac{M^2 - m_0^2(t) - m_1^2(t)}{2M} \end{aligned} \tag{132}$$

$$m_1(t+1) = 2\frac{f_0}{\bar{f}(t)}\frac{m_0^2}{2M}p_m + 2\frac{f_1}{\bar{f}(t)}\frac{m_1^2}{2M}(1-p_m) + \\ + \frac{f_1}{\bar{f}(t)}\frac{M^2 - m_0^2(t) - m_1^2(t)}{2M} \tag{133}$$

U navedenim formulama bitno je uočiti da je broj nula i jedinica nastao selekcijom i mutacijom parova istovjetnih vrijednosti uvećanim dva puta jer se u tim parovima zaista nalaze dvije nule ili dvije jedinice.

Udio nula u sljedećoj generaciji iznosi

$$P(t+1) = \frac{m_0(t+1)}{m_0(t+1) + m_1(t+1)} \tag{134}$$

Uvrštavanjem vrijednosti $m_0(t+1)$ i $m_1(t+1)$ te dijeljenjem nazivnika i brojnika s M i f_0 dobiva se

$$P(t+1) = \\ = \frac{2\frac{f_0}{\bar{f}(t)f_0}\frac{m_0^2}{2M^2}(1-p_m) + 2\frac{f_1}{\bar{f}(t)f_0}\frac{m_1^2}{2M^2}p_m + \frac{f_1}{\bar{f}(t)f_0}\frac{M^2 - m_0^2(t) - m_1^2(t)}{2M^2}}{2\frac{f_0}{\bar{f}(t)f_0}\frac{m_0^2}{2M^2} + 2\frac{f_1}{\bar{f}(t)f_0}\frac{m_1^2}{2M^2} + 2\frac{f_1}{\bar{f}(t)f_0}\frac{M^2 - m_0^2(t) - m_1^2(t)}{2M^2}} \tag{135}$$

U jednadžbu se nadalje može uvrstiti

$$r = \frac{f_1}{f_0} \tag{136}$$

te udio nula i jedinica u populaciji u vremenu t

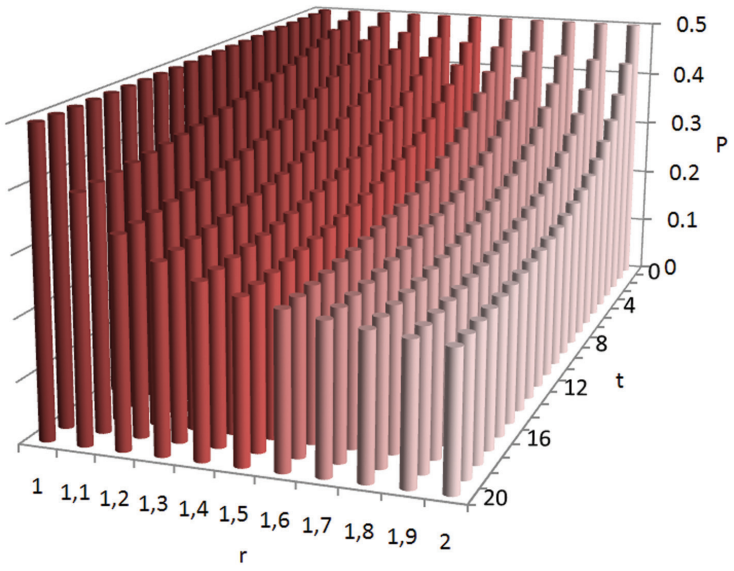
$$P(t) = \frac{m_0(t)}{m_0(t) + m_1(t)} = \frac{m_0(t)}{M} \tag{137}$$

$$Q(t) = \frac{m_1(t)}{m_0(t) + m_1(t)} = \frac{m_1(t)}{M} = 1 - P(t) \tag{138}$$

čime se dobiva

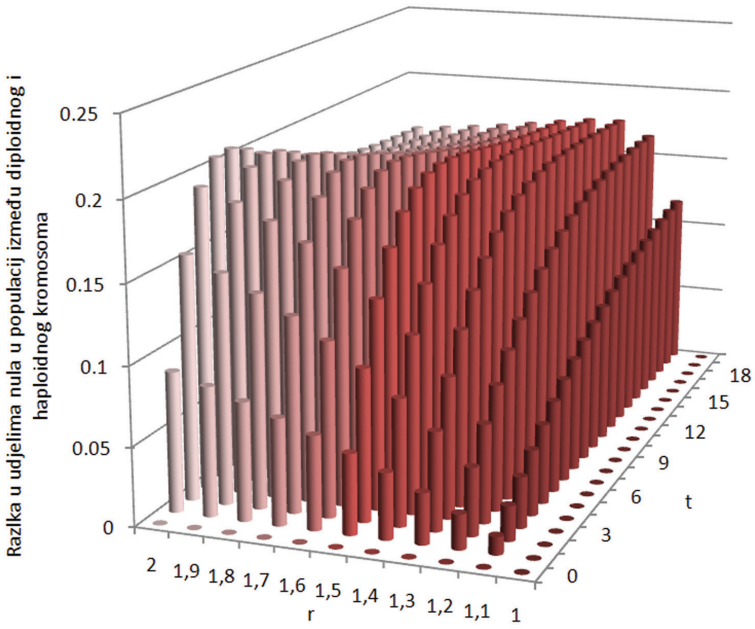
$$P(t+1) = \frac{2P^2(t)(1-p_m) + 2rQ^2(t)p_m + r\left(1 - P^2(t) - Q^2(t)\right)}{2P^2(t) + 2rQ^2(t) + 2r\left(1 - P^2(t) - Q^2(t)\right)} \quad (139)$$

Na slici 6.1.2. prikazana je promjena broja nula iz generacije u generaciju ovisno o *r* i koeficijentu mutacije:



Slika 6.1.2. Brzina gubitka nula (slabiji gen) za koeficijent mutacije 0,0033 – diploidni kromosom

Usporede li se slike 6.1.1. i 6.1.2. može se uočiti da je udio nula u populaciji za iste uvjete bitno veći u diploidnog u haploidnog kromosoma. Razlika u udjelima prikazana je na slici 6.1.3.



Slika 6.1.3. Razlika u brzini gubitka nula (slabiji gen) za koeficijent mutacije 0,0033

Slika zorno pokazuje da je točna intuitivna procjena po kojoj diploidni kromosom više čuva slabije gene. Upravo je ova analiza uz Hamiltonovu usporedbu glavni razlog zašto je diploidni kromosom korišten u algoritmu predloženom u ovom radu.

6.2. OPERATOR KRIŽANJA I DIPLOIDNI KROMOSOM (MEJOZA)

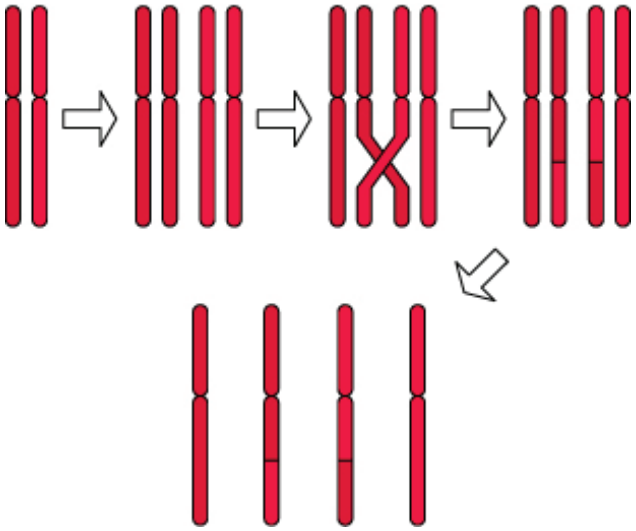
Operator križanja, kako je opisan u poglavlju 3.5.2. vrijedi samo za haploidne kromosome. Situacija je nešto drugačija s diploidnim kromosomom. Naime, diploidni se kromosom u prirodi nikada ne dijeli po dužini na dva dijela već se podjela događa uzdužno, pri čemu nastaju dva haploidna kromosoma koji sudjeluju u stvaranju novog potomka. Novonastali je potomak ponovo diplo-

idan. Na ovaj se način tijekom evolucije izmjenjuju haploidna i diploidna varijanta kromosoma. Radi boljeg i jednostavnijeg razumijevanja procesa križanja kod diploidnog kromosoma te svih pojednostavnjenja koja genetički algoritam donosi potrebno je na početku detaljnije opisati prirodni proces.

6.2.1. Prirodni proces

U svih diploidnih organizama postoje dvije vrste stanica: stanice koje nastaju mitozom i stanice koje nastaju mejozom. Većina stanica u diploidnom organizmu spada u varijantu koja nastaje dijeljenjem odnosno mitozom. Stoga nam nisu pretjerano bitne jer ne sudjeluju u razmjeni genetskih informacija roditelja prilikom stvaranja potomaka. S druge strane, tijekom embrionalnog razvoja nastaju spolne stanice koje imaju posebnu ulogu tijekom reprodukcije (u biljaka spolne stanice iz kojih se dobiva pelud nastaju tijekom cvjetanja, a ne tijekom embrionalnog razvoja). Spolne stanice prolaze kroz proces mejoze (proces diobe spolnih stanica u dva koraka). Rezultat su mejoze haploidne stanice koje sudjeluju u reprodukciji. To što je riječ o haploidnim, a ne o diploidnim stanicama osobito je važno jer spajanjem dviju haploidnih stanica, po jedne od svakog roditelja, nastaje novi diploidni organizam. Ako bi se u proces stvaranja novog organizma ušlo s diploidnim stanicama, rezultat bi bio udvostručivanje broja kromosoma, što bi imalo nesagledive posljedice. Primjerice, čovjek koji ima 46 kromosoma imao bi potomke sa 92 kromosoma. U sljedećoj generaciji broj kromosoma povećao bi se na 184 i tako dalje. Upravo mejoza sprečava takvo eksponencijalno nakupljanje kromosoma.

Za razliku od mitoze, mejoza se sastoji od dvije sukcesivne podjele stanica logično nazvanih mejoza I i mejoza II. Kao što je prikazano na slici 6.2.1., proces započinje repliciranjem kromosoma. Važno je uočiti da je tijekom mejoze I veza između kromatida postojana, za razliku od procesa mitoze. Tijekom procesa podjele homologni kromosomi položeni su duž staničnog ekvatora radi osiguranja pravilne kromosomske podjele na dvije stanice. Dok su kromosomi položeni duž staničnog ekvatora, dolazi do razmjene genetskog materijala (križanje), pri čemu nastaju nove kombinacije gena na pojedinim pozicijama u kromosomu. Kad je razmjena genetskog materijala provedena, dolazi do razilaženja kromosoma, pri čemu nastaju dvije nove stanice. Time je proces mejoze I završio. Drugi korak mejoze (mejoza II) vrlo je sličan procesu mitoze, ali kako je broj kromosoma u svakoj stanici već smanjen na pola, novonastale su stanice haploidne. Kako su tijekom mejoze I nastale dvije stanice, poslije završetka cjelokupnog procesa mejoze nastaju četiri stanice.



Slika 6.2.1. Proces mejoze

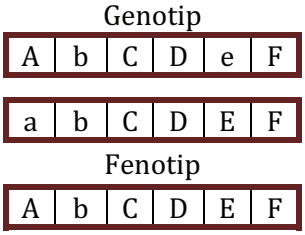
6.2.2. Operator križanja u predloženom algoritmu

Uobičajena varijanta uporabe operatora križanja u genetičkim algoritmima križanje je kromosoma dvaju roditelja. Takav način uporabe operatora križanja uobičajen je za haploidne kromosome. Kako je u ovom radu predložen algoritam koji upotrebljava diploidnu varijantu kromosoma, pruža se mogućnost da se genetički algoritam barem na području križanja približi prirodnom procesu. Tako više nema križanja između kromosoma dvaju roditelja, već se križanje odvija unutar jednog kromosoma (roditelja). Diploidni kromosom sastoji se od dva niza gena (dva haploidna kromosoma) koji posve opisuju jedinku. Upravo ta dva niza gena sudjeluju u procesu križanja, pri čemu nastaju dva haploidna kromosoma od kojih će samo jedan sudjelovati u stvaranju nove jedinke. Za razliku od prirodnog procesa, predloženi algoritam rabi pojednostavljenu varijantu procesa mejoze koji se odnosi samo na mejozu II. Time na kraju procesa nastaju samo dvije haploidne stanice, a ne četiri kao kod prirodnog procesa. Po jedna od novonastalih haploidnih stanica svakog od roditelja sudjeluje u stvaranju nove diploidne jedinke. Budući da novonastala jedinka nosi genetske osobine obaju roditelja, potrebno je radi određivanja

njezine vrijednosti (funkcija cilja) definirati koji će se od dva komplementarna gena iskazati odnosno biti dominantan. Upravo je problem dominacije razlog rjeđe uporabe diploidnih (poliploidnih) kromosoma u genetičkim algoritmima.

6.3. DOMINACIJA

Diploidni kromosom bitno se ne razlikuje od haploidnog kromosoma, barem sa stanovišta genetičkih algoritama. Razlika je samo u tome da je kod diploidnog kromosoma nužno postojanje mehanizma za odabir jednog od dvaju komplementarnih gena na istoj poziciji u kromosomu koji će se fenotipski iskazati. Taj mehanizam zove se dominacija. Da bi dominacija bila moguća, nužno je postojanje dominantnih i recesivnih gena. Primjerice, ako velikim slovima označimo dominantne gene, a malim slovima njihove komplementarne recesivne gene kao što je prikazano na slici 6.3.1., dominantni gen (velika slova) uvijek je iskazan, a recesivni je gen iskazan samo kada su oba gena recesivna. U navedenom primjeru samo se na drugoj poziciji u kromosomu nalaze oba recesivna gena (mala slova) te je to jedino mjesto na kojem se recesivni gen iskazao u fenotipu. Genetičari će reći da se dominantni gen iskazuje u heterozigotnoj varijanti (npr. aA) te u homozigotnoj varijanti (npr, BB), a recesivni gen iskazuje se samo u homozigotnoj varijanti (npr. dd).



Slika 6.3.1. Primjer prelaska s genotipa na fenotip

Dominacija je intenzivno proučavana od početaka razvoja genetičkih algoritama, ali je prvi pravi iskorak učinio Hollstien 1971. godine opisavši dva jednostavna mehanizma dominacije.

6.3.1. Hollstienova shema dominacije u dvije pozicije[17]

Prva shema dominacije koju je predložio Hollstien prikazivala je svaki binarni gen s dva gena (funkcionalni gen i gen za modifikaciju). Funkcionalni je gen poprimao normalne binarne vrijednosti 0 i 1, a s druge strane modificirajući gen mogao je poprimiti vrijednosti *m* i *M*.

	0M	0m	1M	1m
0M	0	0	0	0
0m	0	0	0	1
1M	0	0	1	1
1m	0	1	1	1

Slika 6.3.2. Shema dominacije u dvije pozicije

Prema Hollstienovu prijedlogu 0 je uvijek dominantan gen ako se na određenoj poziciji u kromosomu nalazi barem jedan *M* modificirajući gen. Mapa dominacije prikazana je na slici 6.3.2. Hollstien je ubrzo uvidio da se može kreirati još jednostavnija shema uvođenjem treće vrijednosti na svakoj poziciji u kromosomu. Novonastala shema naziva se shema dominacije s tri vrijednosti na jednoj poziciji.

6.3.2. Hollstienova shema dominacije s tri vrijednosti na jednoj poziciji[18]

Iako je riječ o binarnom zapisu kromosoma, alfabet koristi tri vrijednosti. Prema Hollstienovu prijedlogu to su {0, 1, 2}. Uvođenjem treće vrijednosti problem dominacije direktno je zapisan u pojedinom genu te nije potreban

[17] Two-locus
[18] Single-locus triallelic

dodatni zapis koji definira dominaciju. U navedenom zapisu 1 predstavlja recesivnu jedinicu, a 2 predstavlja dominantnu jedinicu. Tako se dobiva mapa dominacije prikazana na slici 6.3.3.

	0	1	2
0	0	0	1
1	0	1	1
2	1	1	1

Slika 6.3.3. Shema dominacije s tri vrijednosti na jednoj poziciji

Predložena shema dominacije još se i danas smatra najjednostavnijom i najčišćom shemom predloženom za genetičke algoritme jer predstavlja kombinaciju sheme dominacije i vrijednosti gena u jednoj jedinoj vrijednosti. Rezultati koje su Hollstien i poslije Holland postigli uporabom navedene sheme dominacije dvojaki su. Naime, iako je njezinom uporabom postignuta veća genetska raznolikost u populaciji, kao što je i očekivano, cjelokupni algoritam nije pokazao bitno bolje rezultate. Razlog je u tome da je ispitivanje algoritma provedeno na stacionarnom problemu. Uporabom navedene sheme dominacije na nestacionarnom problemu koji su proveli Goldberg i Smith 1987. [49] pokazala se puna vrijednost diploidnog kromosoma. Bez obzira na svoju očitu vrijednost predloženi model suočava se s dva bitna problema. Naime, postoji određeno odstupanje (*engl. bias*) jer se 1 iskazuje dva puta češće od 0 te model nije jednostavno proširiti na varijantu kada na jednoj poziciji mogu biti više od dvije vrijednosti (nebinarna varijanta kromosoma).

6.3.3. Ryanova aditivna shema dominacije

Prvo u radu [50] predstavljenom na *ECAI Workshop on Genetic Algorithms* 1994. te detaljnije u svojem doktorskom radu 1996. [51] C. Ryan predložio je shemu dominacije koja nema preferenciju ni prema kojoj od vrijednosti gena. Njegovo razmatranje motivirano je prirodnim procesom nepotpune dominacije. Naime, kao što je već naznačeno na primjeru krvne grupe, u prirodi se pojavljuju varijante gena kod kojih potomci nemaju osobine samo jednog od

roditelja već potomci iskazuju istodobno osobine obaju roditelja. Najpoznatiji su primjer crveni i bijeli cvjetovi zijevalice (Antirrhinum majus) čijim križanjem nastaju ružičasti cvjetovi

Slika 6.3.4. Zijevalica (Antirrhinum majus) u tri varijante

U konkretnom slučaju ni crvena ni bijela varijanta nije dominantna te u prvoj generaciji nastaju ružičasti cvjetovi. Križanjem ružičastih cvjetova nastaje 25% bijelih, 25% crvenih i 50% ružičastih cvjetova, što na zoran način upozorava na postojanje nepotpune dominacije.

Opisivanje navedene situacije u genetičkom algoritmu uporabom binarnog zapisa {0, 1} nije dovoljno. Naime, binarni zapis može opisati bijele i crvene cvjetove, ali ne i ružičaste. Uvođenjem treće vrijednosti uvest će se odstupanje (*engl. bias*) u korist jedne od boja. Stoga je uvođenje dviju novih vrijednosti optimalno rješenje. Ryan je genima dodijelio vrijednosti {2, 3, 7, 9} te ih je nazvao A, B, C, D. Zbrajanjem gena na istoj poziciji dobivaju se vrijednosti koje određuju koji će se gen iskazati. Tako se sve varijante u kojima je zbroj manji od 11 iskazuju kao 0, a varijante veće ili jednake 11 iskazuju se kao 1

Tablica 6.3.1. Veza fenotipa i genotipa

Fenotip 0	Fenotip 1
AA (2+2=4)	AD (2+9=11)
AB (2+3=5)	BD (3+9=12)
BB (3+3=6)	CC (7+7=14)
AC (2+7=9)	CD (7+9=16)
BC (3+7=10)	DD (9+9=18)

Prema tablici veze između fenotipa i genotipa definira se aditivna shema dominacije prema Ryanu.

	A	B	C	D
A	0	0	0	1
B	0	0	0	1
C	0	0	1	1
D	1	1	1	1

Slika 6.3.5. Aditivna shema dominacije

Opisana shema dominacije ne preferira ni jednu vrijednost gena kao Hollstienova shema dominacije, što znači da odstupanja (*engl. bias*) nema.

6.4. PREDLOŽENI MODEL

Do sada opisane sheme dominacije isključivo su orijentirane na binarnu prezentaciju kromosoma. Kako predloženi algoritam rabi dekadski način prezentacije, potrebno je definirati potpuno novu shemu dominacije.

Predložena shema dominacije zasnovana je na parnosti odnosno neparnosti zbroja vrijednosti gena na istoj poziciji. Ako je zbroj neparan, tada je dominantan gen s manjom vrijednosti i obrnuto; ako je zbroj paran, tada je dominantan gen s većom vrijednosti. Tako, primjerice, ako geni imaju vrijednosti 5 i 8 te je njihov zbroj 13 (neparan broj), dominantan je gen s vrijednošću 5. Primjer potpune sheme dominacije za gene od 1 do 5 prikazan je na slici 6.4.1.

	1	2	3	4	5
1	1	1	3	1	5
2	1	2	2	4	2
3	3	2	3	3	5
4	1	4	3	4	4
5	5	2	5	4	5

Slika 6.4.1. Shema dominacije za predloženi algoritam

Prednost je predložene sheme u tome da su sve vrijednosti gena jednako često dominantne i recesivne te ju je moguće proširiti na bilo koji broj različitih vrijednosti gena. K tome, shema dominacije inkorporirana je u kromosom te ne postoji razlika između funkcionalnih gena i gena za modifikaciju.

7. EKSPERIMENT

Teorijska razmatranja opisana u prethodnim poglavljima rezultirala su stvaranjem genetičkog algoritma s nišama čiji je rad eksperimentalno provjeren. Eksperimentalna provjera provedena je na poznatim benchmark problemima. Prije svega riječ je o *mt6*, *mt10* i *mt20* problemima koje su definirali Muth i Thompson 1963. [52] te o 40 problema (*la01-la40*) koje je definirao S. Lawrence. *Mt* problemi poslužili su za statističku usporedbu jednostavnoga genetičkog algoritma s predloženim algoritmom, a *la* problemi poslužili su za provjeru ponašanja algoritma kod povećanja veličine problema uz istodobno zadržavanje osnovnih parametara konstantnim.

7.1. DEFINICIJA BENCHMARK PROBLEMA

Muth i Thompson definirali su samo tri problema od kojih je *mt6* (6 poslova i 6 strojeva) iznimno jednostavan i ne predstavlja ni u jednom pogledu velik problem. Definicija problema dana je u tablici 7.1.1.

Tablica 7.1.1. Tehnološki raspored za *mt6* benchmark problem

Posao	Strojevi					
1	3	1	2	4	6	5
2	2	3	5	6	1	4
3	3	4	6	1	2	5
4	2	1	3	4	5	6
5	3	2	5	6	1	4
6	2	4	6	1	5	3

Posao	Trajanje operacija					
1	1	3	6	7	3	6
2	8	5	10	10	10	4
3	5	4	8	9	1	7
4	5	5	5	3	8	9
5	9	3	5	4	3	1
6	3	3	9	10	4	1

S druge strane, *mt10* i *mt20* problemi su gotovo identični i znatno teže rješivi. Bitna je razlika u tome da se *mt20* izvodi na upola manje strojeva, a ima dva puta više poslova. Ako se pažljivije promotri tehnološki raspored za oba problema, lako se može uočiti da prvih deset poslova u oba problema imaju

identičan tehnološki raspored. Dapače, druga operacija na svih deset poslova *mt10* problema odgovara prvoj operaciji poslova 11-20 na *mt20* problemu i tako redom. Problem *mt10* ostao je nerješiv više od 20 godina te s pravom spada u klasične benchmark probleme. Tehnološki raspored za *mt10* problem prikazan je u tablicama 7.1.2. i 7.1.3.

Tablica 7.1.2. Tehnološki raspored za *mt10* benchmark problem

Posao	**Strojevi**									
1	1	2	3	4	5	6	7	8	9	10
2	1	3	5	10	4	2	7	6	8	9
3	2	1	4	3	9	6	8	7	10	5
4	2	3	1	5	7	9	8	4	10	6
5	3	1	2	6	4	5	9	8	10	7
6	3	2	6	4	9	10	1	7	5	8
7	2	1	4	3	7	6	10	9	8	5
8	3	1	2	6	5	7	9	10	8	4
9	1	2	4	6	3	10	7	8	5	9
10	2	1	3	7	9	10	6	4	5	8

Tablica 7.1.3. Trajanje operacija za *mt10* benchmark problem

Posao	**Trajanje operacija**									
1	29	78	9	36	49	11	62	56	44	21
2	43	90	75	11	69	28	46	46	72	30
3	91	85	39	74	90	10	12	89	45	33
4	81	95	71	99	9	52	85	98	22	43
5	14	6	22	61	26	69	21	49	72	53
6	84	2	52	95	48	72	47	65	6	25
7	46	37	61	13	32	21	32	89	30	55
8	31	86	46	74	32	88	19	48	36	79
9	76	69	76	51	85	11	40	89	26	74
10	85	13	61	7	64	76	47	52	90	45

Tehnološki raspored problema *mt20* prikazan je u tablici 7.1.4.

Tablica 7.1.4. Tehnološki raspored za *mt20* benchmark problem

Posao	Strojevi				
1	1	2	3	4	5
2	1	2	4	3	5
3	2	1	3	5	4
4	2	1	5	3	4
5	3	2	1	4	5
6	3	2	5	1	4
7	2	1	3	4	5
8	3	2	1	4	5
9	1	4	3	2	5
10	2	3	1	4	5
11	2	4	1	5	3
12	3	1	2	4	5
13	1	3	2	4	5
14	3	1	2	4	5
15	1	2	5	3	4
16	2	1	4	5	3
17	1	3	2	4	5
18	1	2	5	3	4
19	2	3	1	4	5
20	1	2	3	4	5

Posao	Trajanje operacija				
1	29	9	49	62	44
2	43	75	69	46	72
3	91	39	90	12	45
4	81	71	9	85	22
5	14	22	26	21	72
6	84	52	48	47	6
7	46	61	32	32	30
8	31	46	32	19	36
9	76	76	85	40	26
10	85	61	64	47	90
11	78	36	11	56	21
12	90	11	28	46	30
13	85	74	10	89	33
14	95	99	52	98	43
15	6	61	69	49	53
16	2	95	72	65	25
17	37	13	21	89	55
18	86	74	88	48	79
19	69	51	11	89	74
20	13	7	76	52	45

Pored *mt* problema razmatrat će se rad algoritma *na la01-la40* problemima. Od navedenih 40 problema sedam se smatra iznimno teškim. To su redom *la21* (15x10), *la24* (15x10), *la25* (20x10), *la27* (20x10), *la29* (20x10), *la38* (15x15), i *la40* (15x15). U tablici 7.1.5. prikazan je popis *la* problema s optimalnim rješenjima ili teoretski najkraćim rasporedom (*eng. lower bound*).

Tablica 7.1.5. *la01* do *la40* problemi (najbolja rješenja)

Problem	PxS[19]	Opt[20]	LB[21]	Problem	PxS	Opt	LB
la01	10x5	666	-	la21	15x10	-	1046
la02	10x5	655	-	la22	15x10	927	-
la03	10x5	597	-	la23	15x10	1032	-
la04	10x5	590	-	la24	15x10	-	935
la05	10x5	593	-	la25	15x10	-	977
la06	15x5	926	-	la26	20x10	1218	-
la07	15x5	890	-	la27	20x10	-	1235
la08	15x5	863	-	la28	20x10	1216	-
la09	15x5	951	-	la29	20x10	-	1130
la10	15x5	958	-	la30	20x10	1355	-
la11	20x5	1222	-	la31	30x10	1784	-
la12	20x5	1039	-	la32	30x10	1850	-
la13	20x5	1150	-	la33	30x10	1719	-
la14	20x5	1292	-	la34	30x10	1721	-
la15	20x5	1207	-	la35	30x10	1888	-
la16	10x10	945	-	la36	15x15	1268	-
la17	10x10	784	-	la37	15x15	1397	-
la18	10x10	848	-	la38	15x15	-	1196
la19	10x10	842	-	la39	15x15	1233	-
la20	10x10	907	-	la40	15x15	1222	-

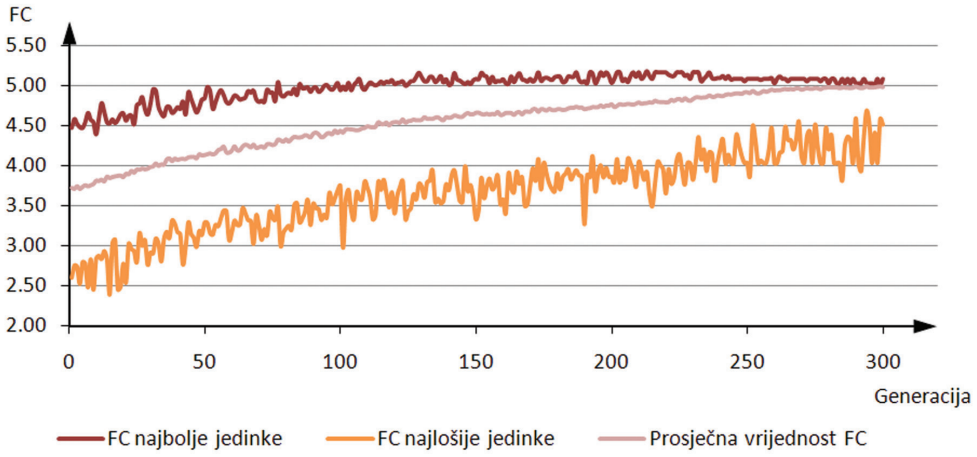
Definicija (tehnološki raspored i trajanje operacija) svakog pojedinačnog *la* problema može se preuzeti s internetske adrese

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt

[19] P – Posao; S - Stroj
[20] Optimalno rješenje
[21] LB (engl. Lower Bound) – za probleme kod kojih nije poznato optimalno rješenje navedeno je trajanje teorijski najkraćeg rasporeda

7.2. SVOJSTVA PREDLOŽENOGA GENETIČKOG ALGORITMA

Pojedini dijelovi predloženoga genetičkog algoritma poput selekcije, operatora križanja i mutacije već su zasebno opisani u prethodnim poglavljima. Za potrebe eksperimenta sve je cjeline trebalo objediniti u jedan jedinstven algoritam.



Slika 7.2.1. Oscilacija vrijednosti najbolje i najlošije jedinke u generaciji

Nit vodilja u stvaranju genetičkoga algoritma bila je njegova što je moguće veća sličnost s prirodnim procesima. Stoga je odabran indirektan način zapisivanja operacija kako bi se mogli koristiti klasični operatori križanja i mutacije. K tome, ni na jednom mjestu u algoritmu se na koriste globalne informacije, odnosno ni jednoj jedinki nisu poznate informacije o nekoj drugoj jedinki. Tako se, primjerice, u algoritmu ne koristi općeprihvaćen i često korišten elitizam. Elitizmu je u prvom redu cilj sačuvati najbolju jedinku (ili više njih) u generaciji. Time se postiže stabilniji rad algoritma te bolji ukupni rezultati. Na žalost, takva praksa u prirodnom svijetu nije uobičajena, barem ne u organizama na višem stupnju razvoja. U razvijenijih organizama nije moguće kloniranje kako bi najbolja jedinka neizmijenjena prešla u sljedeću generaciju. Posljedica je toga mogućnost da najbolja jedinka iduće generacija bude lošija od najbolje jedinke iz prethodne generacije. Situacija nije ni najmanje neuobičajena. Slika 7.2.1. to na najbolji način pokazuje. Naime, na slici 7.2.1. prikazana je oscilacija vrijednosti najbolje i najlošije jedinke u generaciji. Algoritam ni u jednom trenutku ne koristi informacije vezane uz

problem koji se rješava. Time je posve izbjegnuta ovisnost algoritma o problemu, odnosno nemogućnost primjene algoritma na drugim klasama problema. Algoritam je povrh problema raspoređivanja uspješno ispitan na problemu trgovačkog putnika i OSSP-u (*engl. Open Shop Scheduling Problem*).

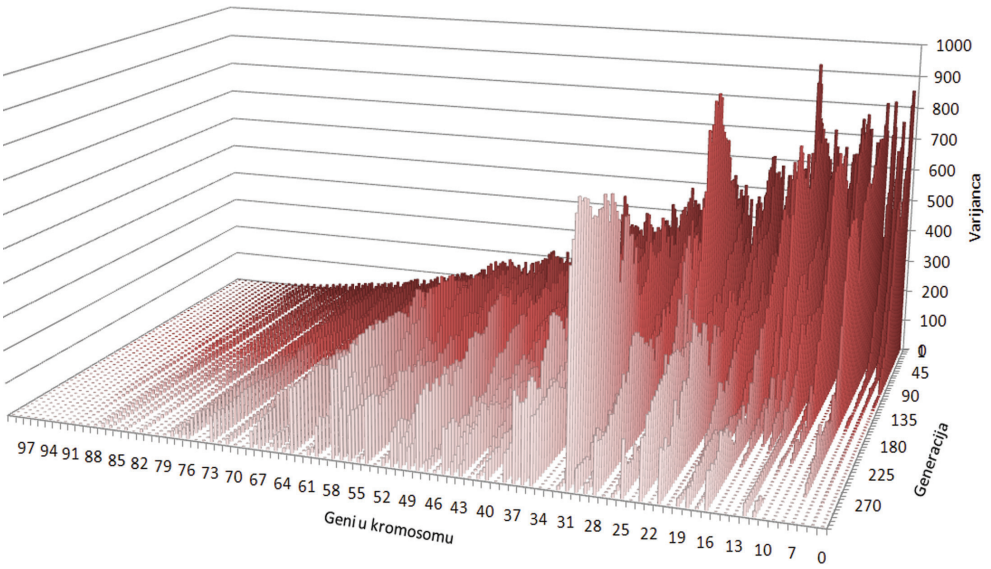
Sljedeće svojstvo koje bitno utječe na rad algoritma je shema dominacije. U poglavlju 6.3. opisane su neke od najuspješnijih shema dominacije te je predložena shema dominacije za dekadski zapis kromosoma. Kako je već navedeno, pri stvaranju algoritma pokušalo se u najvećoj mogućoj mjeri poštovati prirodne zakone. Stoga su odbačene sve sheme dominacije koje je 1981. predložila i analizirala Anne Brindle [2] [24]. Ona je u svojem doktorskom radu predložila sljedeće sheme dominacije:

- nasumična, fiksna, globalna shema dominacije (*engl. Random, fixed, global dominance*)
- varijabilna, globalna shema dominacije (*engl. Variable, global dominance*)
- deterministička, varijabilna, globalna shema dominacije (*eng. Deterministic, variable, global dominance*)
- shema dominacije zasnovana na nasumičnom odabiru kromosoma (*engl. Choose a random chromosome*)
- dominacija boljeg kromosoma (*engl. Dominance of the better chromosome*)
- adaptivna shema dominacije u kojoj haploidni kromosom kontrolira diploidni (*engl. Haploid controls diploid adaptive dominance*).

Predložene sheme dominacije imale su velik odjek u znanstvenoj zajednici koja im je zamjerila nepostojanje teorijskih bioloških osnova. Primjerice, neke od predloženih shema dominacije rabe globalne informacije (podaci o drugim članovima populacije) kako bi se odredila dominacija na lokalnoj razini. Takve sheme dominacije zasigurno nemaju biološku osnovu. K tome, adaptivna shema dominacije kod koje haploidni kromosom kontrolira dominaciju diploidnog kromosoma podrazumijeva razdvajanje funkcionalnih gena i gena za modifikaciju, što također ne postoji u prirodi. U trenutku kada je Anne Brindle predložila navedenu shemu dominacije, Hollstein je već bio predložio svoju poznatu shemu dominacije s tri vrijednosti na jednoj poziciji koja je te dvije funkcije objedinila. Zbog svega navedenog, pri stvaranju predloženoga genetičkog algoritma vodilo se posebno računa da se izbjegne uporaba globalnih informacija te se što je moguće više pokušalo približiti biološkim osnovama.

Sljedeće je bitno svojstvo algoritma s nišama mogućnost očuvanja genetske raznolikosti populacije. Na slici 7.2.2. prikazana je promjena varijance gena u kromosomu kroz generacije. Primjer prikazan na slici odnosi se na *mt10* benchmark problem s 300 članova u populaciji. Iz slike se jasno može očitati nekoliko svojstava predloženog algoritma. Prije svega, kako je riječ o kodiranju prema referentnoj listi, tako se i broj različitih vrijednosti koje pojedini gen može poprimiti smanjuje, počevši od prvoga gena pa sve do posljednjeg gdje je varijanca jednaka nuli jer posljednji gen može poprimiti samo jednu vrijednost.

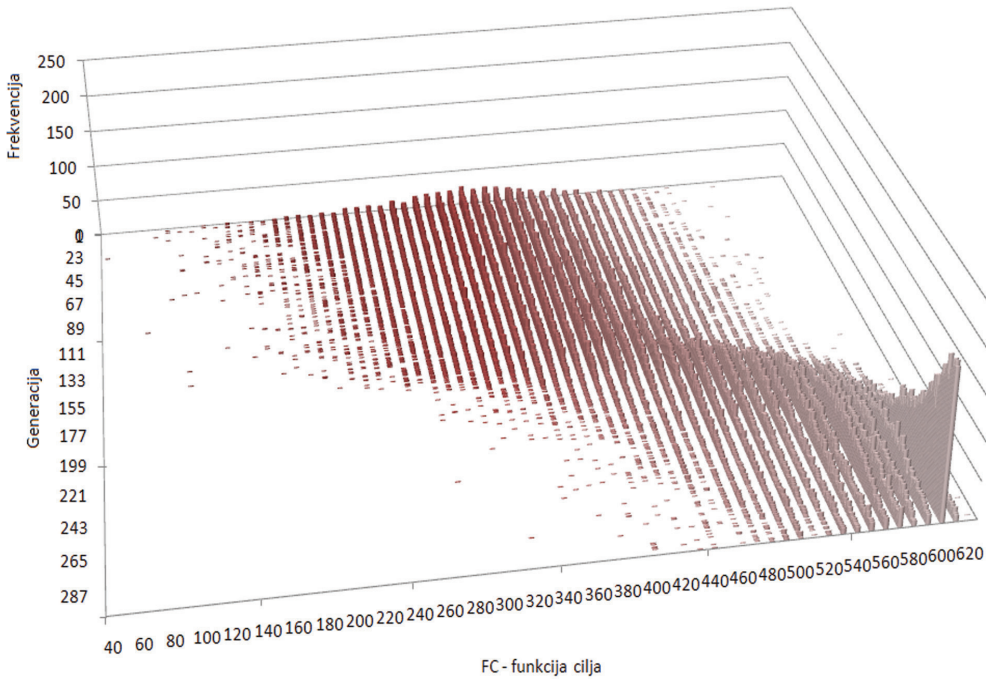
Drugo je bitno svojstvo to da se prvo fiksiraju vrijednosti prvih gena u kromosomu, što se iskazuje naglim padom varijance. Bitno je istaknuti da do pada varijance dolazi nakon što algoritam prođe graničnu generaciju $G_{gr} = 150$, poslije čega se namjerno urušavaju niše kako bi se provela eksploatacija samo jedne potencijalno najbolje niše. Za algoritme s nišama (jasno je vidljivo iz slike) posebno je to da je genetska raznolikost (varijanca) na gotovo svim genima ostala u okvirima inicijalne genetske raznolikosti. Dapače, na nekim pozicijama u kromosomu je vidljivo i povećanje varijance. Takvo povećanje



Slika 7.2.2. Promjena varijance pojedinih gena kroz generacije

varijance na prvi pogled nije logično jer se genetska raznolikost populacije može povećati samo uporabom operatora mutacije, koji je u konkretnom slučaju bio iznimno malen, samo 0,0033 te sigurno nije bitno pridonio povećanju varijance. Stvaran uzročnik rasta varijance na pojedinim genima uporaba je diploidnoga kromosoma koji je uspješno obranio slabije gene koji su se naknadno ponovo razmnožili kao posljedica kreiranja niša.

K tome, uvedena je generacijski promjenjiva funkcija cilja posljedica koje je odgođeno povećanje pritiska selekcije, što se zorno vidi na slici 7.2.3. Nakon 150. generacije (pola ukupnog trajanja evolucije), kada su niše potpuno formirane, dolazi do povećanja pritiska selekcije te pomaka populacije prema boljim rješenjima. Pred kraj evolucije algoritam ima tendenciju posve demontirati niše kako bi istražio samo jednu nišu radi pronalaženja najprilagođenije jedinke u odabranoj niši.



Slika 7.2.3. Frekvencija pojavljivanja pojedinih rješenja kroz generacije

7.3. PARAMETRI ALGORITMA

Istraživanja odnosno provjera kvalitete rada predloženoga genetičkog algoritma provedena je tako da je algoritam s odgovarajućim konfiguracijskim parametrima pokretan između 50 i 600 puta ovisno o benchmark problemu koji se optimizirao. Na taj je način dobivena statistički prihvatljiva razdioba na osnovi koje se može odgovoriti na pitanje iz teze ovoga doktorskoga rada, to jest jesu li rješenja dobivena predloženim algoritmom bliža optimumu te je li rasipanje rješenja manje u odnosu na jednostavni genetički algoritam. Na samom početku načelno se očekivalo da će algoritam raditi bolje od jednostavnoga genetičkog algoritma te je stoga potražena usporedba s nekim svjetski poznatim algoritmom za koji postoje javno dostupni statistički podaci. Odabir je pao na GT-genetički algoritam koji su predložili Yamada i Nakano, a čiji su statistički podaci objavljeni u doktorskom radu *Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems* na sveučilištu Kyoto 2003. godine [53].

Prilikom pokretanja genetičkoga algoritma predloženog u ovom doktorskom radu korišteni su sljedeći parametri:

- veličina populacije – veličina populacije držana je konstantnom tijekom trajanja algoritma
- dužina kromosoma – dužina kromosoma direktno je proporcionalna problemu koji se rješava
- vrsta i vjerojatnost mutacije
- vrsta i vjerojatnost križanja
- broj generacija predaka koji definiraju nišu – parametar koji direktno utječe na radijus utjecaja pojedine jedinke na ostale članove populacije. Ako je parametar veći, kreirana je niša veća
- broj generacija – parametar koji određuje trajanje i uvjet je prekida algoritma
- granična generacija – generacija nakon koje se bitno povećava pritisak selekcije
- eksponent – parametar koji utječe na skaliranje funkcije cilja radi jednostavnijeg kreiranja niša
- vrsta selekcije.

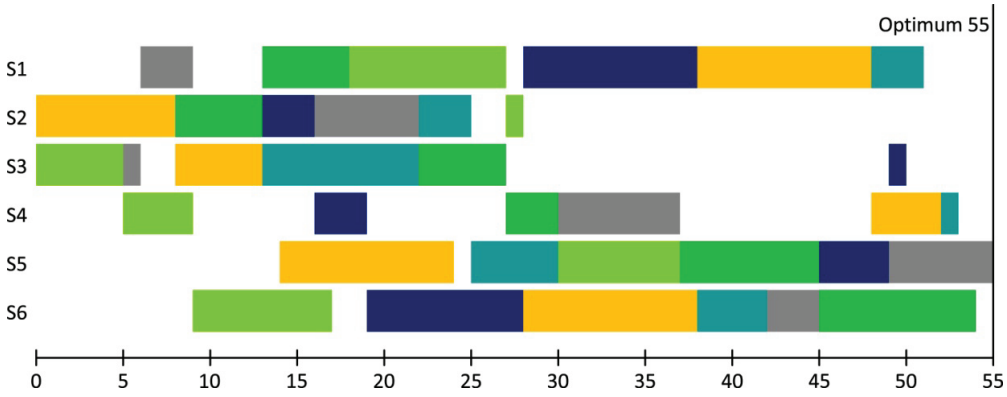
7.4. ISPITIVANJE ALGORITMA NA *MT* BENCHMARK PROBLEMIMA

Predloženi genetički algoritam s nišama prvo je ispitan na Muth i Thompson *benchmark* problemima. Prvo ispitivanje provedeno je na *mt6* problemu koji ne predstavlja računalni problem za algoritam. Gotovo pri svakom pokretanju algoritma, čak i pri relativno malenim populacijama i kratkim evolucijama, algoritam je pronašao optimalno rješenje.

Algoritam je ispitan uz sljedeće parametre:

- veličina populacije – 50
- dužina kromosoma – 36
- vrsta i vjerojatnost mutacije – mutacija jednoga gena uz vjerojatnost (0,02)
- vrsta i vjerojatnost križanja – križanje s prekidom u jednoj točki (0,95)
- broj generacija predaka koje definiraju nišu – 5
- broj generacija – 20
- granična generacija – 10
- eksponent – 1
- vrsta selekcije – univerzalni stohastički odabir.

Optimalan je raspored dugačak 55 vremenskih jedinica. Najbolji pronađeni raspored također je dugačak 55 vremenskih jedinica i prikazan je na slici 7.4.1.

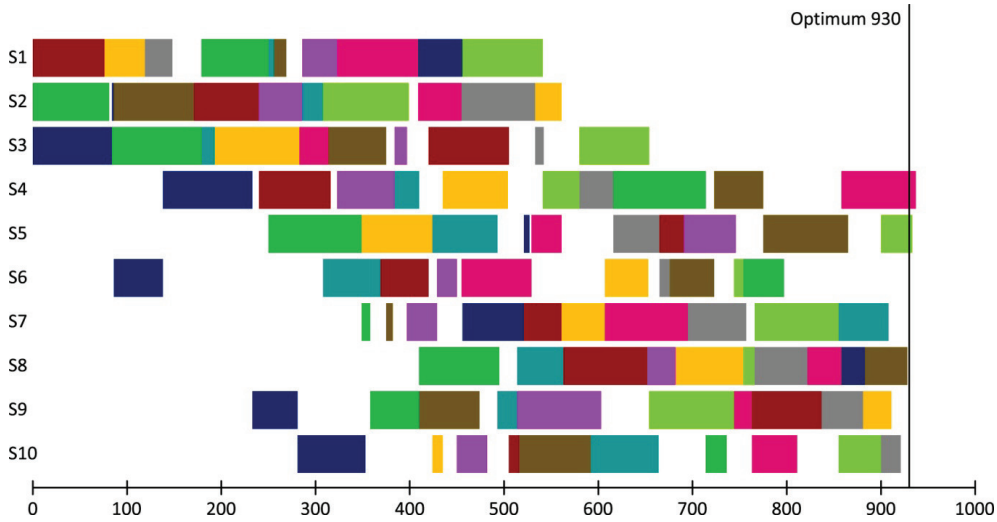


Slika 7.4.1. *mt6* (55 vremenskih jedinica)

Za razliku od *mt6*, problem *mt10* znatno je teži. Njegovo optimalno rješenje nije bilo pronađeno više od 20 godina. Najkraći raspored koji je kreirao predloženi algoritam dugačak je 937 vremenskih jedinica, a optimalno je rješenje dugačko 930.

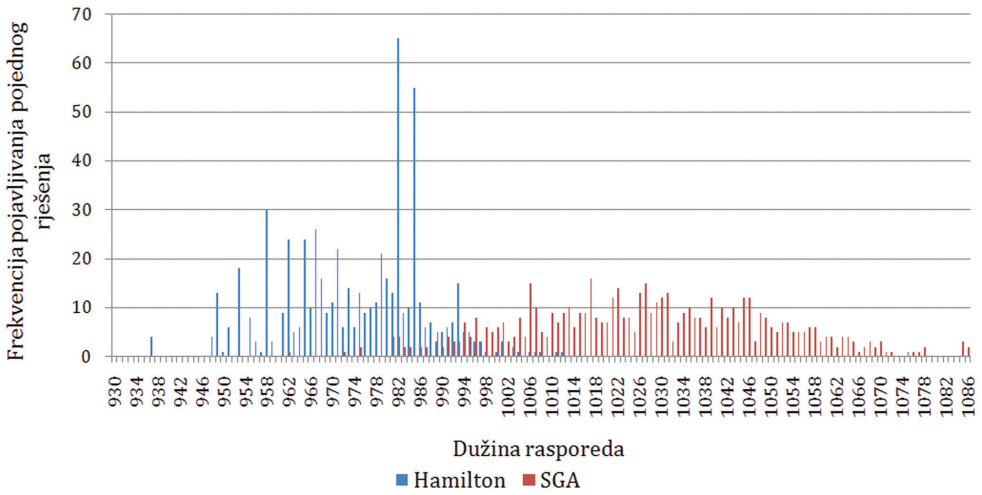
Algoritam je ispitan uz sljedeće parametre:

- veličina populacije – 300
- dužina kromosoma – 100
- vrsta i vjerojatnost mutacije – mutacija jednog gena uz vjerojatnost (0,0033)
- vrsta i vjerojatnost križanja – križanje s prekidom u jednoj točki (0,95)
- broj generacija predaka koje definiraju nišu – 5
- broj generacija – 250
- granična generacija – 150
- eksponent – 4
- vrsta selekcije – univerzalni stohastički odabir



Slika 7.4.2. mt10 (937 vremenskih jedinica)

Radi statističke provjere i usporedbe s jednostavnim genetičkim algoritmom evolucija je pokrenuta 600 puta te je frekvencija pojavljivanja pojedinih rješenja prikazana na slici 7.4.3.



Slika 7.4.3. Frekvencija pojavljivanja pojedinih rješenja u 600 evolucija (SGA, Hamilton)

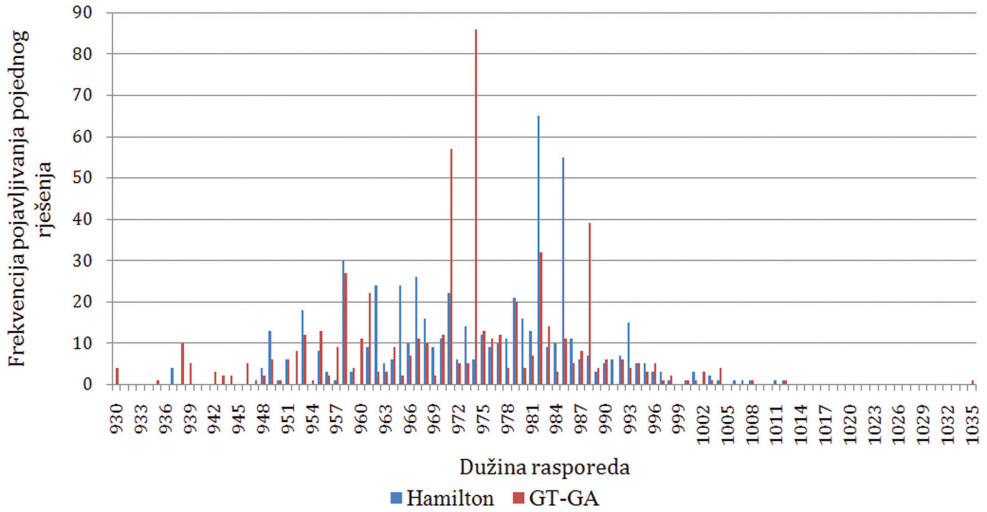
Iz slike se jasno vidi da predloženi algoritam ima manje rasipanje i da je pomaknut u lijevo prema optimalnom rješenju. Statistička usporedba obaju algoritama dana je u tablici 7.4.1.

Tablica 7.4.1. Statistička usporedba algoritama

	Hamilton	SGA	GT-GA
Broj evolucija	600	600	600
Standardna devijacija	12,9812	22,26397	14,40027
Aritmetička srednja vrijednost	974,4033	1027,142	973,4183
Medijan	976	1027	
Maksimalna vrijednost	1012	1086	1035
Minimalna vrijednost	937	962	930
Broj generacija	250	250	200
Veličina populacije	300	300	1000

Osim jednostavnog genetičkog algoritma identična usporedba provedena je i sa GT genetičkim algoritmom. Rezultati GT algoritma također su prikazani u tablici 7.4.1.

Usporedba dobivenih rezultata upozorava na dva bitna svojstva predloženoga algoritma. Prije svega rasipanje dobivenih rješenja nešto je manje kod predloženog algoritma u odnosu na poznati GT genetički algoritam. Ako se još uzme u obzir i to da je predloženi algoritam ispitan sa znatno manjom populacijom, odnosno manjom genetskom raznolikošću, rezultat je to vrjedniji. S druge strane GT-GA 4 puta je od 600 pokušaja pronašao optimalno rješenje, a predloženi algoritam to nije uspio ni jedanput. Razlog je u uporabi iznimno malene populacije kako bi se dobilo na brzini rada algoritma. Naime, uporaba niša te potreba uspoređivanja svih jedinki kako bi se odredila pripadnost pojedine jedinke određenoj niši bitno usporava algoritam. Stoga su jedinke u populaciji svedene na najmanji mogući broj.

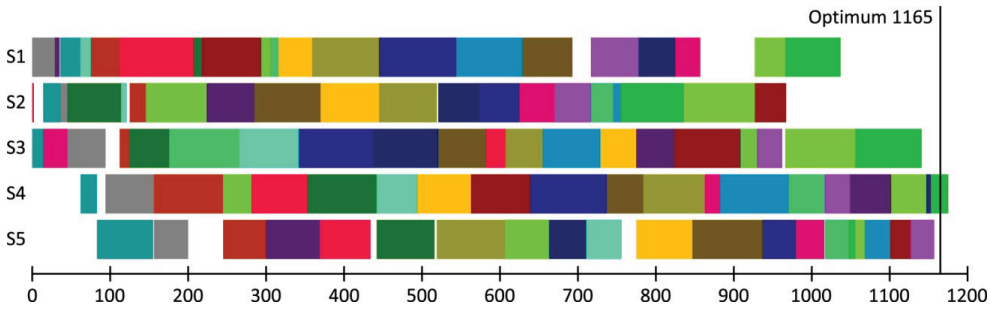


Slika 7.4.4. Frekvencija pojavljivanja pojedinih rješenja u 600 evolucija (GT, Hamilton)

Posljednje ispitivanje iz *mt* grupe benchmark problema provedeno je na *mt20* problemu. Njegov optimalan raspored dugačak je 1165 vremenskih jedinica, a predloženi je algoritam došao vrlo blizu na 1175 vremenskih jedinica, pa je odstupanje manje od 1% od optimalnog rješenja, a Ganttov dijagram za dobiveno rješenje prikazan je na slici 7.4.5.

Algoritam je ispitan uz sljedeće parametre:

- veličina populacije – 300
- dužina kromosoma – 100
- vrsta i vjerojatnost mutacije – mutacija jednoga gena uz vjerojatnost (0,0033)
- vrsta i vjerojatnost križanja – križanje s prekidom u jednoj točki (0,95)
- broj generacija predaka koje definiraju nišu – 5
- broj generacija – 250
- granična generacija – 150
- eksponent – 4
- vrsta selekcije – univerzalni stohastički odabir.



Slika 7.4.5. *mt20* (1175 vremenskih jedinica)

7.5. ISPITIVANJE ALGORITMA NA *LA* BENCHMARK PROBLEMIMA

La benchmark problemi mogu se podijeliti u osam grupa sa po pet problema u svakoj. Problemi u svakoj grupi jednake su veličine po broju strojeva i poslova koji se obavljaju. *La* grupa problema poslužit će za ispitivanje ponašanja algoritma na različitim veličinama problema uz istodobno držanje većine parametara konstantnim. Algoritam je za svaki problem pokrenut 50 puta kako bi se

dobio statistički relevantan rezultat. S povećanjem veličine problema povećavao se i prostor pretraživanja pa je bilo nužno produžiti trajanje algoritma, odnosno smanjiti kapacitet niša. Zbog nepovoljna utjecaja veličine populacije na performanse algoritma populacija je na svih 40 problema zadržana iznimno nisko, na samo 300 jedinki. Svi ostali parametri poput vrste selekcije, vrste i vjerojatnosti operatora križanja itd. ostali su isti za sva ispitivanja.

Algoritam je ispitan uza sljedeće nepromjenjive parametre:

- veličina populacije – 300
- vrsta i vjerojatnost mutacije – mutacija jednoga gena uz vjerojatnost (0,0033)
- vrsta i vjerojatnost križanja – križanje s prekidom u jednoj točki (0,95)
- broj generacija predaka koje definiraju nišu – 5
- eksponent – 4
- vrsta selekcije – univerzalni stohastički odabir.

Parametri koji su se mijenjali ovisno o grupi *la* problema prikazani su u tablici 7.5.1. S veličinom problema povećavan je broj generacija, odnosno trajanje algoritma te je smanjivan kapacitet niša kako bi algoritam mogao istražiti veći prostor rješenja.

Tablica 7.5.1. Evolucijski parametri

Klasa problema	G	G_{gr}	ω
la01-la05 (10x5)	300	150	1
la06-la10 (15x5)	400	200	1
la11-la15 (20x5)	400	200	1
la16-la20 (10x10)	400	200	1
la21-la25 (15x10)	500	250	1
la26-la30 (20x10)	750	375	0,2
la31-la35 (30x10)	1000	600	0,2
la36-la40 (15x15)	800	480	0,2

Tablica 7.5.2. Rezultati za pojedine *la* probleme

	la01	**la02**	**la03**	**la04**	**la05**	**la06**	**la07**	**la08**	**la09**	**la10**
Arit. sr. v.	666	669,4	617,7	606,7	593	926	890	863	951	958
Median	666	669,5	617	607	593	926	890	863	951	958
Najbolji	666	657	609	593	593	926	890	863	951	958
Najlošiji	666	682	622	612	593	926	890	863	951	958
Optimum	666	655	597	590	593	926	890	863	951	958
σ	0	6,217	2,184	4,834	0	0	0	0	0	0
	la11	**la12**	**la13**	**la14**	**la15**	**la16**	**la17**	**la18**	**la19**	**la20**
Arit. sr. v.	1222	1039	1150	1292	1207	977,9	787,5	875,7	875	922,2
Median	1222	1039	1150	1292	1207	979	787	875	873	918
Najbolji	1222	1039	1150	1292	1207	956	785	855	863	913
Najlošiji	1222	1039	1150	1292	1207	982	804	884	886	944
Optimum	1222	1039	1150	1292	1207	945	784	848	842	902
σ	0	0	0	0	0	6,129	3,435	6,506	4,793	9,324
	la21	**la22**	**la23**	**la24**	**la25**	**la26**	**la27**	**la28**	**la29**	**la30**
Arit. sr. v.	1093,8	980,8	1041,3	998,7	1026,2	1257,2	1300,1	1279,2	1248,8	1396,4
Median	1096	982,5	1039	1002	1024	1265,5	1299,5	1278	1251	1397.5
Najbolji	1068	956	1032	977	1008	1237	1287	1252	1212	1367
Najlošiji	1125	994	1061	1020	1064	1278	1315	1307	1276	1418
Optimum	-	927	1032	935	977	1218	-	1216	-	1355
σ	14,42	7,913	8,484	8,875	11,230	12,389	8,343	14,365	13,757	13,272
	la31	**la32**	**la33**	**la34**	**la35**	**la36**	**la37**	**la38**	**la39**	**la40**
Arit. sr. v.	1785,5	1852,3	1720,3	1733,3	1898,1	1319,9	1462,1	1277,3	1309,9	1286,0
Median	1784	1850	1719	1730	1898	1322	1460	1278	1312	1284
Najbolji	1784	1850	1719	1721	1888	1296	1449	1252	1268	1270
Najlošiji	1791	1964	1736	1761	1928	1335	1490	1303	1335	1326
Optimum	1784	1850	1719	1721	1888	1268	1397	-	1233	1222
σ	2,158	3,815	4,433	10,632	9,649	10,099	10,873	12,967	14,753	11,357

Rezultati dobiveni provođenjem eksperimenta prikazani su u tablici 7.5.2. Od 40 problema za 36 je poznato optimalno rješenje. Od 36 poznatih optimuma predloženi algoritam pronašao je 18. Ovisno o klasi problema algoritam je imao više ili manje uspjeha. Tako je primjerice za klasu problema *la06-la10* (15x5) pronašao optimalno rješenje u svih 50 evolucija. Da je tome tako zorno se vidi iz rasipanja dobivenih rješenja. Naime, standardna je devijacija za svih pet problema jednaka 0. Nadalje, vrlo slični rezultati postignuti su za grupe problema *la11-la15* (20x5) i *la31-la35* (30x10). Navedeni rezultati upućuju na to da se algoritam jednako dobro ponaša za velike probleme kao i za malene. Naime, problemi klase *la31-la35*(30x10) imaju dva puta više operacija i strojeva u odnosu na probleme klase *la06-la10* (15x5) a rezultati su identični. Za sve je probleme standardna devijacija jednaka 0.

Zajedničko je svojstvo problema u kojih je pronađeno optimalno rješenje u odnosu na ostale veći »stupanj slobode«. Naime, ako je potrebno rasporediti veći broj poslova na manji broj strojeva, algoritam puno uspješnije pronalazi globalni optimum. S druge strane, ako je broj poslova jednak ili približno jednak broju strojeva, tada je pronalazak globalnoga optimuma zbog prirode problema bitno otežan. Sama činjenica da u grupama problema *la16-la20* (10x10) i *la36-la40* (15x15) nije pronađen ni jedan globalni optimum govori tome u prilog. Bez obzira na nepronalaženje globalnog optimuma, primjerice za klasu problema *la20* (10x10) u svih 50 pokretanja algoritma na svih pet problema najlošije rješenje uvijek je odstupalo od optimalnog rješenja između 2,55% do 4,65%. Istodobno su najbolja rješenja odstupala od optimalnog između 0,13% i 1,77%.

7.5.1. Utjecaj evolucijskih parametara

Za potrebe eksperimenta koeficijent mutacije i križanja držani su konstantnim jer su smatrani irelevantnim za konačan ishod. S druge stane, koeficijenti G_{gr} i φ pokazali su se iznimno važnim. Naime, inicijalno ispitivanje pokazalo je da ako se G_{gr} postavi suviše malen, pritisak selekcije prerano postaje prevelik te je proces kreiranja niša ozbiljno narušen. Potpuno suprotan, ali također negativan utjecaj ima veliki G_{gr}. Ako je G_{gr} suviše velik, algoritam nema vremena za eksploataciju najbolje niše u posljednjoj fazi evolucije. Za sve eksperimente G_{gr} postavljen je na približno polovicu evolucije.

$$G_{gr} \approx \frac{1}{2} \cdot G_{max} \tag{87}$$

S druge stane, uloga je koeficijenta φ da poveća pritisak selekcije, što implicitno pridonosi bržem kreiranju niša. Ne postoji eksplicitno pravilo za određivanje vrijednosti koeficijenta φ. Načelno se može smatrati da se kod korištenja većih populacija može također upotrijebiti i veći koeficijent φ. Naime, algoritmi koji rabe malene populacije i velike vrijednosti koeficijenta β imaju tendenciju zaglavljivanja u jednom od manje poželjnih lokalnih optimuma.

Osim koeficijenta φ razmatran je i koeficijent δ koji također služi za povećanje pritiska selekcije. Na žalost, za njegovo pravilno određivanje potrebno je poznavanje vrijednosti minimuma funkcije cilja kako se ne bi dobivale negativne vrijednosti. Kako u stvarnom životu nije moguće unaprijed znati ograničenja funkcije cilja čiji se optimum traži, ni na provedenom eksperimentu koeficijent δ nije korišten.

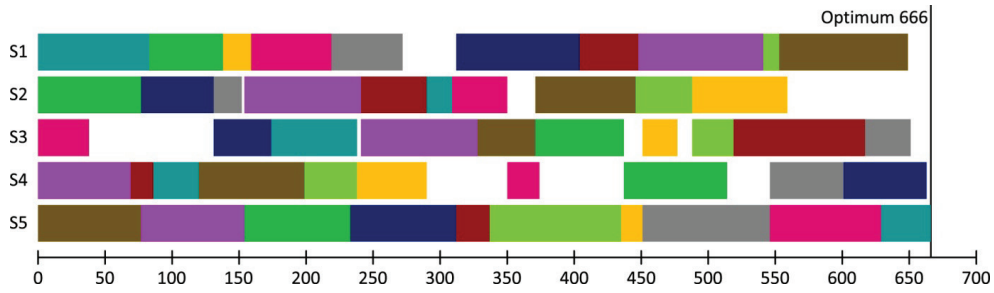
Posljednji bitan koeficijent je ω. Za malene vrijednosti koeficijenta ω kapacitet niše bitno je smanjen. Smanjenje kapaciteta niše posebno je važno kada se koristi malena populacija na relativno velikom problemu. Naime, ako je kapacitet niše velik, formiranje niša je otežano te se može očekivati dominacija trenutačno najbolje prilagođene jedinke. Kada je broj jedinki malen (u svim provedenim eksperimentima broj jedinki držan je relativno nisko; samo 300 jedinki), a prostor pretraživanja iznimno je velik (problemi 20x10, 30x10 i 15x15), koeficijent ω mora biti malen kao što je prikazano u tablici 7.5.1.

7.5.2. La 10x5 problemi

Lawrence 10x5 klasa problema rabi kromosome dužine 50 gena te je evolucija trajala 200 generacija. Klasa problema relativno je jednostavna i ne donosi bitnu teškoću za predloženi algoritam. Odstupanja od optimuma uvijek su manja od 5%. Dobiveni su sljedeći rezultati:

Problem – *la01* (setf1; F1)

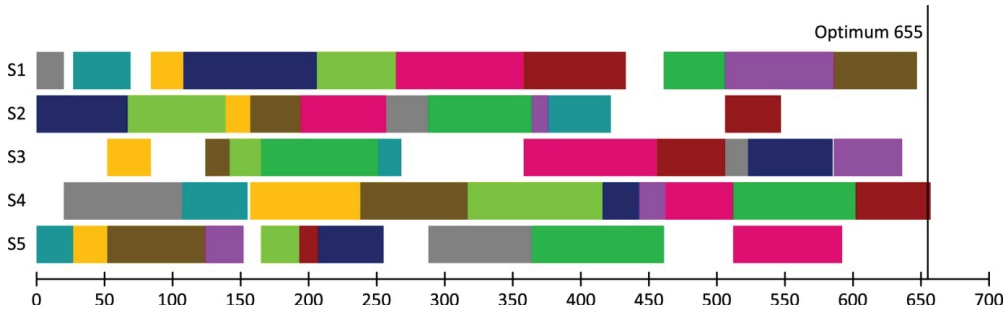
- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 666 (odstupanje od optimuma – 0%)
- Medijan – 666 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 666 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 666 (odstupanje od optimuma – 0%)
- Optimum – 666



Slika 7.5.1. *la01* (666 vremenskih jedinica)

Problem - *la02* (setf2; F2)

- Standardna devijacija – 6,217
- Aritmetička srednja vrijednost – 669,46 (odstupanje od optimuma 2,206%)
- Medijan – 669,5 (odstupanje od optimuma 2,213%)
- Maksimalna vrijednost – 682 (odstupanje od optimuma 4,122%)
- Minimalna vrijednost – 657 (odstupanje od optimuma 0,305%)
- Optimum – 655

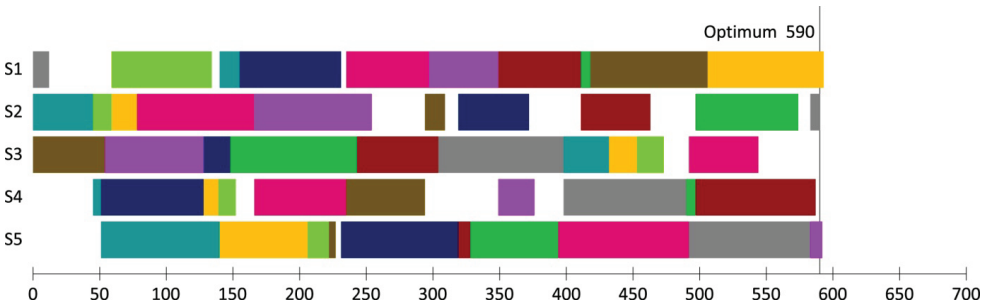


Slika 7.5.2. *la02* (657 vremenskih jedinica)

Problem *la03* (setf3; F3)

- Standardna devijacija – 4,837
- Aritmetička srednja vrijednost – 606,7 (odstupanje od optimuma 2.830%)

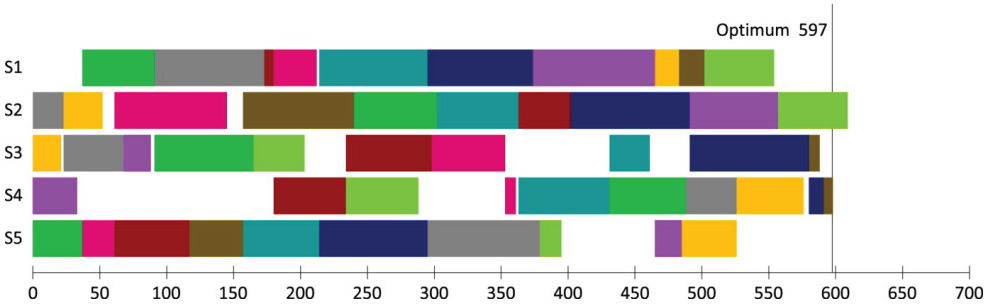
- Medijan – 607 (odstupanje od optimuma 2.881%)
- Maksimalna vrijednost – 612 (odstupanje od optimuma 3,728%)
- Minimalna vrijednost – 593 (odstupanje od optimuma 0,508%)
- Optimum - 590



Slika 7.5.3. *la03* (593 vremenske jedinice)

Problem *la04* (setf4; F4)

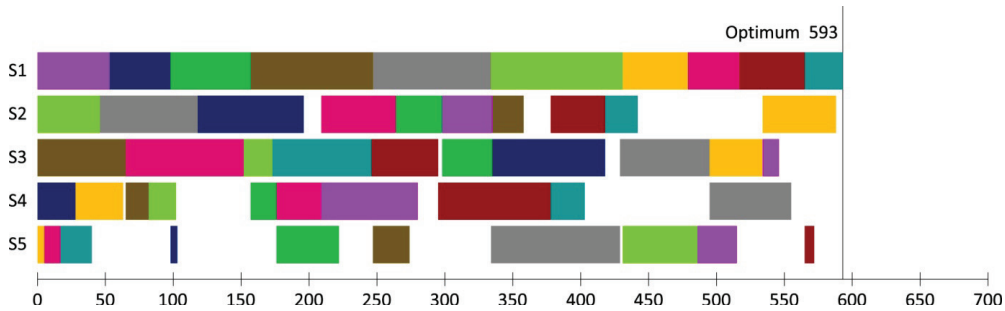
- Standardna devijacija – 2,184
- Aritmetička srednja vrijednost – 617,72 (odstupanje od optimuma 3,470%)
- Medijan – 617 (odstupanje od optimuma 3,350%)
- Maksimalna vrijednost – 622 (odstupanje od optimuma 4,187%)
- Minimalna vrijednost – 609 (odstupanje od optimuma 2.010%)
- Optimum - 597



Slika 7.5.4. *la04* (609 vremenskih jedinica)

Problem *la05* (setf5; F5)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 593 (odstupanje od optimuma – 0%)
- Medijan – 593 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 593 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 593 (odstupanje od optimuma – 0%)
- Optimum – 593



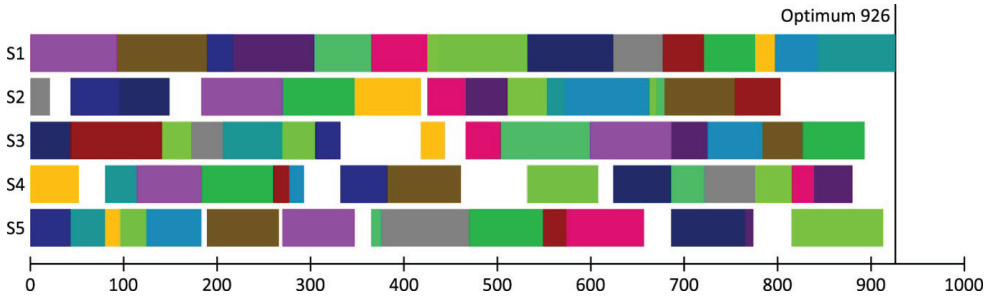
Slika 7.5.5. *la05* (593 vremenske jedinice)

7.5.3. La 15x5 problemi

Lawrence 15x5 klasa problema rabi kromosome dužine 75 te je evolucija trajala 250 generacija. Prema dobivenim rezultatima riječ je o iznimno jednostavnoj klasi problema. Algoritam uvijek pronalazi optimalno rješenje.

Problem *la06* (setg1; G1)

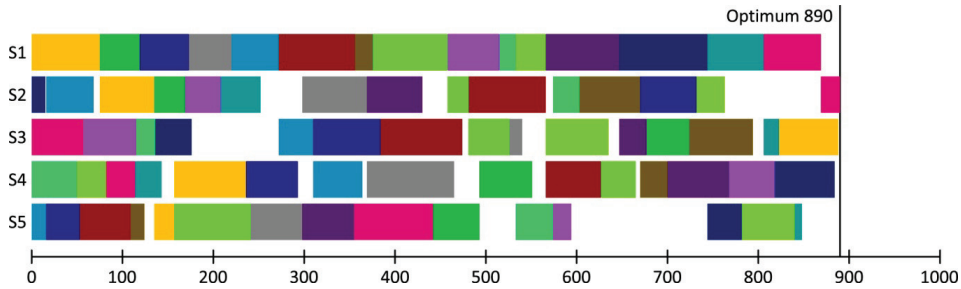
- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 926 (odstupanje od optimuma – 0%)
- Medijan – 926 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 926 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 926 (odstupanje od optimuma – 0%)
- Optimum – 926



Slika 7.5.6. *la06* (926 vremenskih jedinica)

Problem *la07* (setg2; G2)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 890 (odstupanje od optimuma – 0%)
- Medijan – 890 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 890 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 890 (odstupanje od optimuma – 0%)
- Optimum – 890

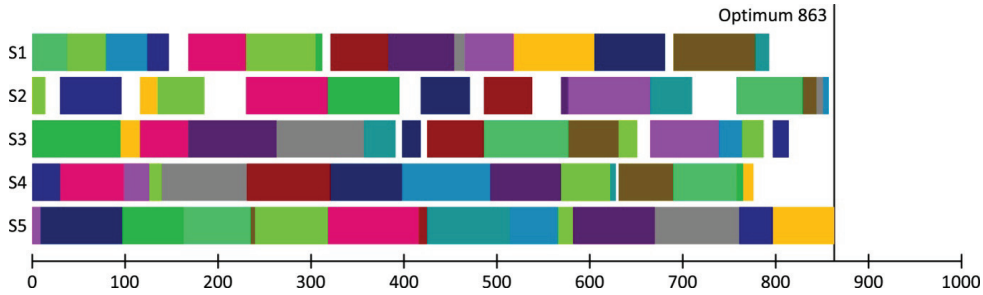


Slika 7.5.7. *la07* (890 vremenskih jedinica)

Problem *la08* (setg3; G3)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 863 (odstupanje od optimuma – 0%)
- Medijan – 863 (odstupanje od optimuma – 0%)

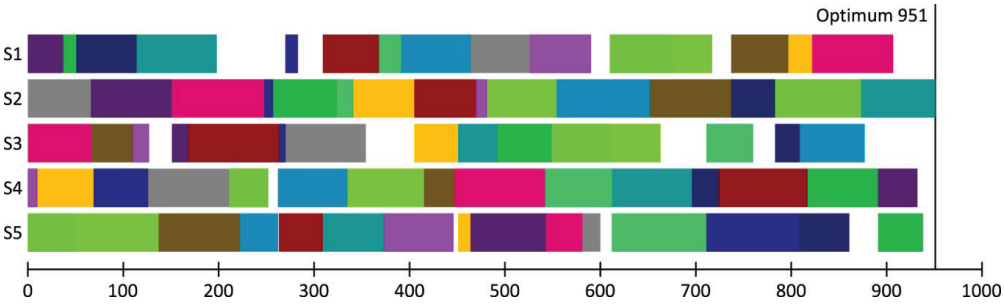
- Maksimalna vrijednost – 863 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 863 (odstupanje od optimuma – 0%)
- Optimum – 863



Slika 7.5.8. *la08* (863 vremenske jedinice)

Problem *la09* (setg4; G4)

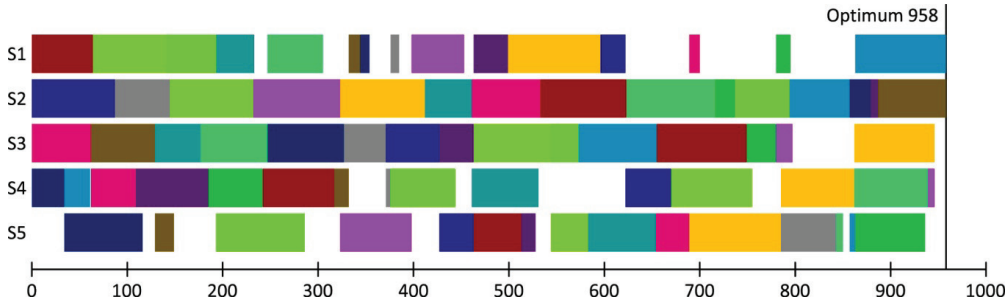
- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 951 (odstupanje od optimuma – 0%)
- Medijan – 951 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 951 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 951 (odstupanje od optimuma – 0%)
- Optimum – 951



Slika 7.5.9. *la09* (951 vremenska jedinica)

Problem *la10* (setg5; G5)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 958 (odstupanje od optimuma – 0%)
- Medijan – 958 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 958 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 958 (odstupanje od optimuma – 0%)
- Optimum – 958



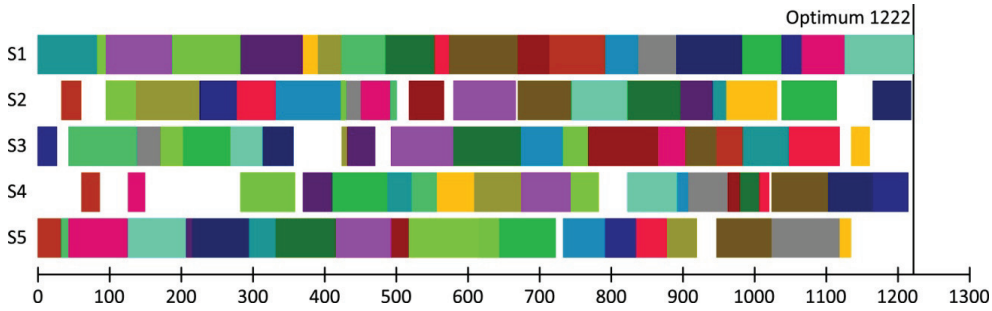
Slika 7.5.10. *la10* (958 vremenskih jedinica)

7.5.4. La 20x5 problemi

Lawrence 20x5 klasa problema rabi kromosome dužine 100 te je evolucija trajala 300 generacija. Također jednostavna klasa problema u kojoj algoritam uvijek pronalazi optimalna rješenja.

Problem *la11* (seth1; H1)

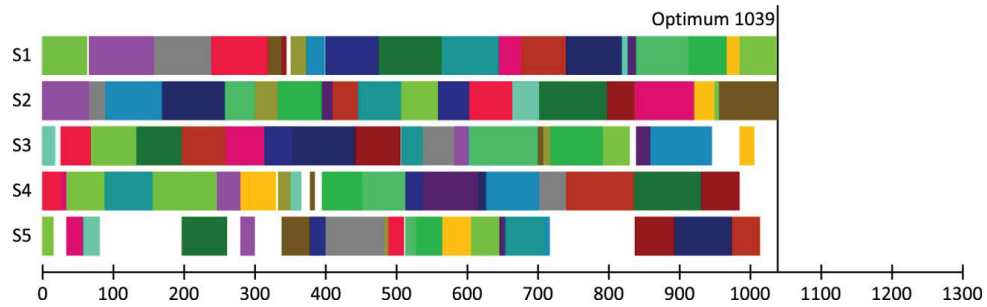
- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 1222 (odstupanje od optimuma – 0%)
- Medijan – 1222 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 1222 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 1222 (odstupanje od optimuma – 0%)
- Optimum – 1222



Slika 7.5.11. *la11* (1222 vremenske jedinice)

Problem *la12* (seth2; H2)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 1039 (odstupanje od optimuma – 0%)
- Medijan – 1039 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 1039 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 1039 (odstupanje od optimuma – 0%)
- Optimum – 1039

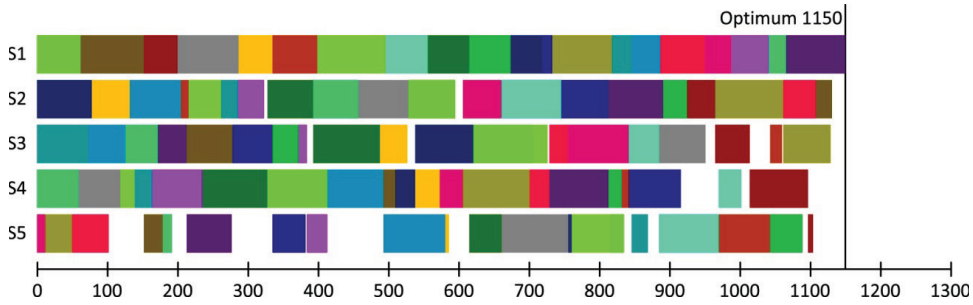


Slika 7.5.12. *la12* (1039 vremenskih jedinica)

Problem *la13* (seth3; H3)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 1150 (odstupanje od optimuma – 0%)
- Medijan – 1150 (odstupanje od optimuma – 0%)

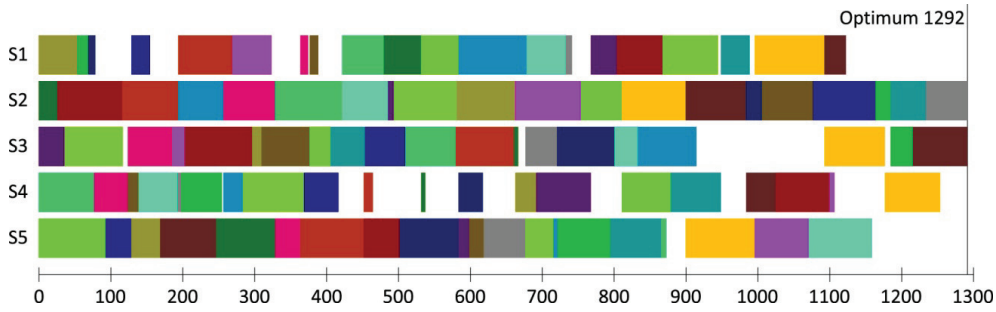
- Maksimalna vrijednost – 1150 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 1150 (odstupanje od optimuma – 0%)
- Optimum – 1150



Slika 7.5.13. *la13* (1150 vremenskih jedinica)

Problem *la14* (seth4; H4)

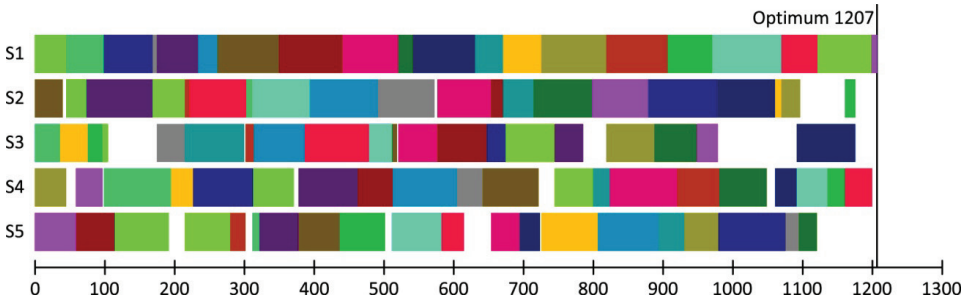
- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 1292 (odstupanje od optimuma – 0%)
- Medijan – 1292 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 1292 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 1292 (odstupanje od optimuma – 0%)
- Optimum – 1292



Slika 7.5.14. *la14* (1292 vremenske jedinice)

Problem *la15* (seth5; H5)

- Standardna devijacija – 0
- Aritmetička srednja vrijednost – 1207 (odstupanje od optimuma – 0%)
- Medijan – 1207 (odstupanje od optimuma – 0%)
- Maksimalna vrijednost – 1207 (odstupanje od optimuma – 0%)
- Minimalna vrijednost – 1207 (odstupanje od optimuma – 0%)
- Optimum – 1207



Slika 7.5.15. *la15* (1207 vremenskih jedinica)

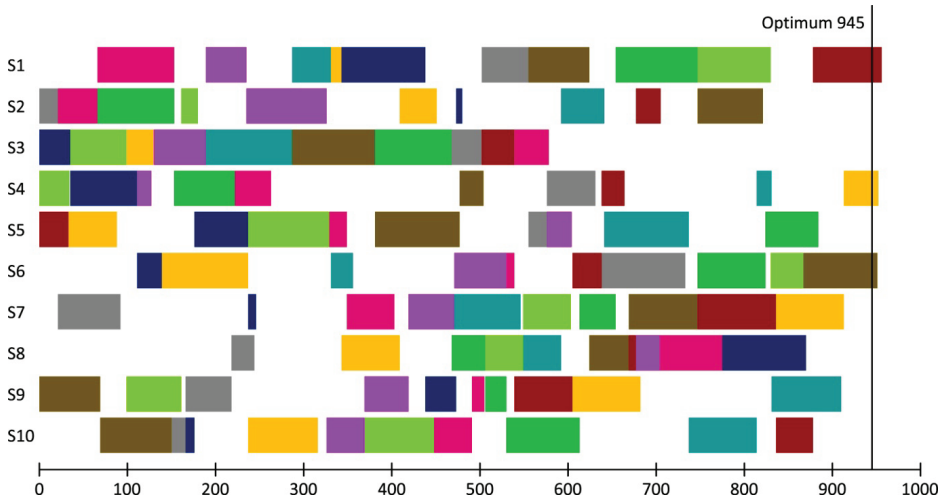
7.5.5. La 10x10 problemi

Lawrence 10x10 klasa problema rabi kromosome dužine 100 gena te je evolucija trajala 400 generacija. Zbog dvaput većeg broja operacija u svakom poslu klasa problema znatno je teža u odnosu na sve prethodne klase. Zbog povećanja prostora pretraživanja, a istodobna zadržavanja malene populacije (siromašan genetski materijal) algoritam nije pronašao optimalno rješenje ni za jedan problem. S druge strane, standardna devijacija ostala je relativno malena te su sva dobivena rješenja, uključujući najlošija, unutar 5% od globalnog optimuma.

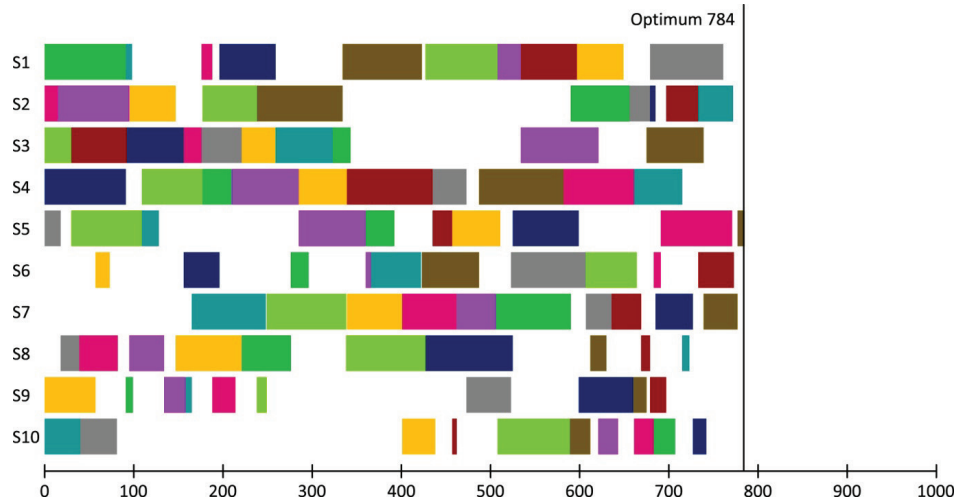
Problem *la16* (seta1; A1)

- Standardna devijacija – 6,129
- Aritmetička srednja vrijednost – 977,98 (odstupanje od optimuma – 3,489%)
- Medijan – 979 (odstupanje od optimuma – 3,597%)

- Maksimalna vrijednost – 982 (odstupanje od optimuma – 3,915%)
- Minimalna vrijednost – 956 (odstupanje od optimuma – 1,164%)
- Optimum – 945



Slika 7.5.16. *la16* (956 vremenskih jedinica)



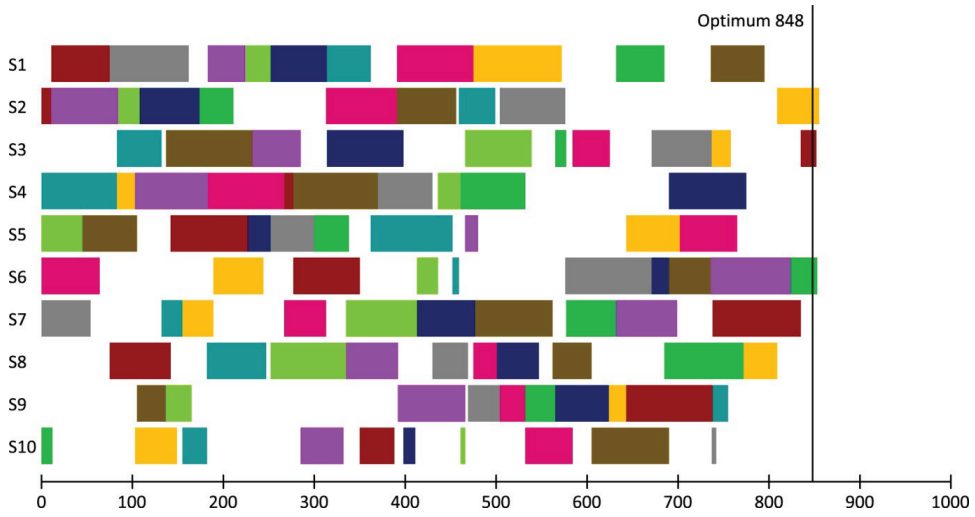
Slika 7.5.17. *la17* (785 vremenskih jedinica)

Problem *la17* (seta2; A2)

- Standardna devijacija – 3,435
- Aritmetička srednja vrijednost – 787,54 (odstupanje od optimuma – 0,451%)
- Medijan –787 (odstupanje od optimuma – 0,382%)
- Maksimalna vrijednost – 804 (odstupanje od optimuma – 2,551%)
- Minimalna vrijednost – 785 (odstupanje od optimuma – 0,127%)
- Optimum – 784

Problem *la18* (seta3; A3)

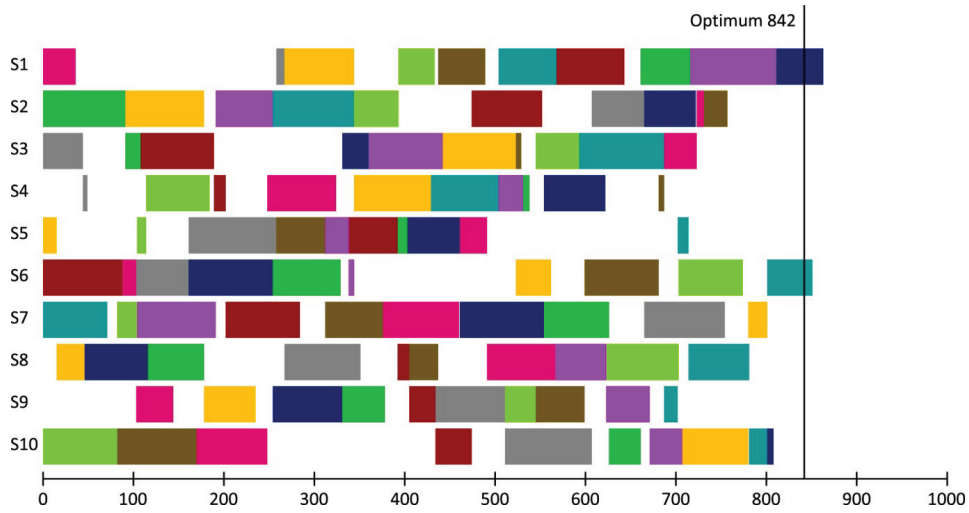
- Standardna devijacija – 6,506
- Aritmetička srednja vrijednost – 875,7 (odstupanje od optimuma – 3,266%)
- Medijan – 875 (odstupanje od optimuma – 3,301%)
- Maksimalna vrijednost – 884 (odstupanje od optimuma – 4,245%)
- Minimalna vrijednost – 855 (odstupanje od optimuma – 0,826%)
- Optimum – 848



Slika 7.5.18. *la18* (855 vremenskih jedinica)

Problem *la19* (seta4; A4)

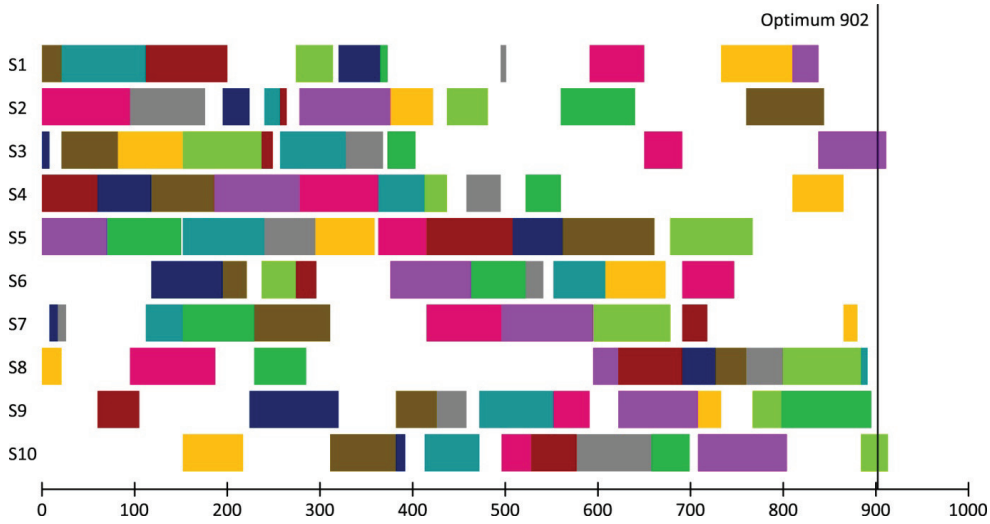
- Standardna devijacija – 4,793
- Aritmetička srednja vrijednost – 875,08 (odstupanje od optimuma – 3,193%)
- Medijan – 872,5 (odstupanje od optimuma – 2,889%)
- Maksimalna vrijednost – 886 (odstupanje od optimuma – 4,481%)
- Minimalna vrijednost – 863 (odstupanje od optimuma – 1,768%)
- Optimum – 842



Slika 7.5.19. *la19* (863 vremenske jedinice)

Problem *la20* (seta5; A5)

- Standardna devijacija – 9,324
- Aritmetička srednja vrijednost – 922,22 (odstupanje od optimuma – 2,241%)
- Medijan – 918 (odstupanje od optimuma – 1,773%)
- Maksimalna vrijednost – 944 (odstupanje od optimuma – 4,656%)
- Minimalna vrijednost – 913 (odstupanje od optimuma – 1,219%)
- Optimum – 902



Slika 7.5.20. *la20* (913 vremenskih jedinica)

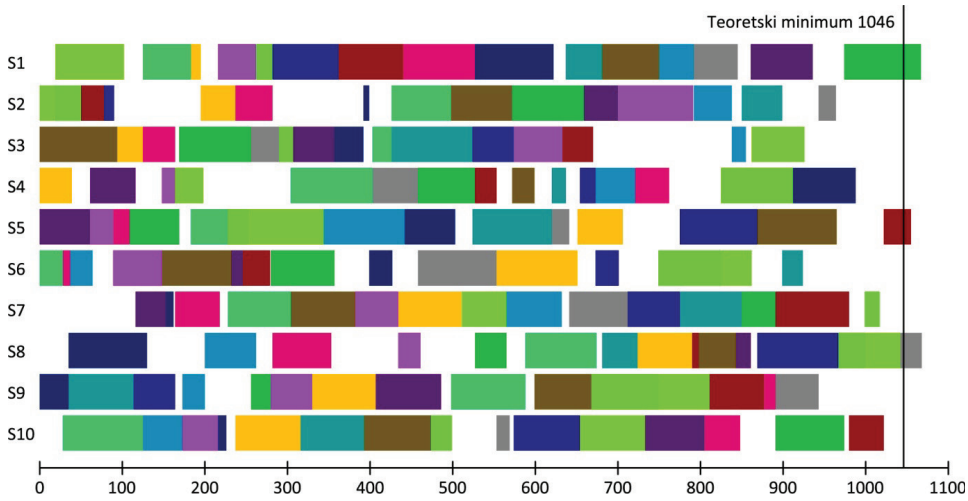
7.5.6. La 15x10 problemi

Lawrence 15x10 klasa problema rabi kromosome dužine 150 gena te je evolucija trajala 500 generacija. Problemi iz ove grupe posebno su važni jer su Applegate i Cook analizom problema koji se pojavljuju u literaturi odredili set problema poznatih pod imenom »10 teških problema« . Od pet problema iz grupe *la* 15x10 čak su se tri našla u popisu Applegetea i Cooka. To su redom *la21*, *la24* i *la25*. Da je riječ o iznimno teškim problemima najbolje se očitava u povećanoj standardnoj devijaciji na pojedinim problemima. Jedino je za problem *la23* pronađen globalni optimum.

Problem *la21* (setb1;B1)

- Standardna devijacija – 14,420
- Aritmetička srednja vrijednost – 1093,86
- Medijan –1096

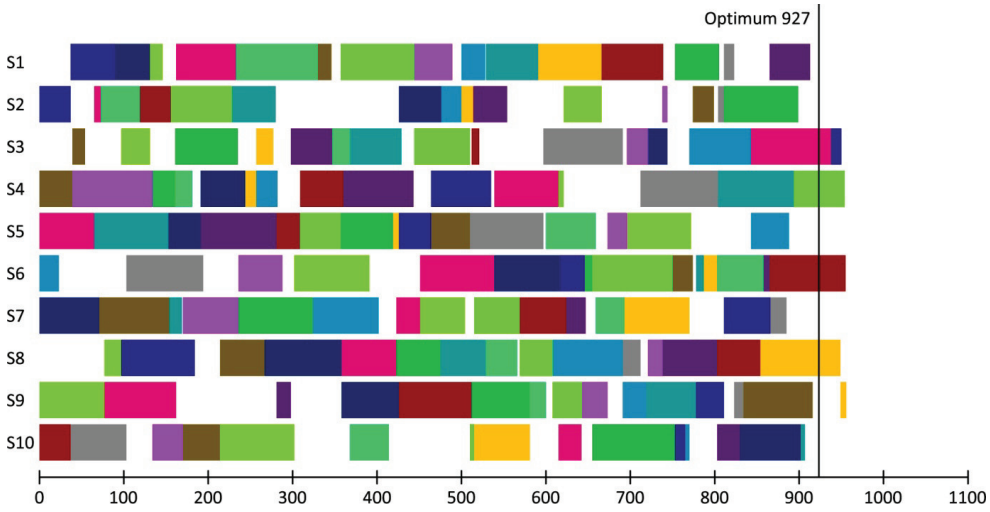
- Maksimalna vrijednost – 1125
- Minimalna vrijednost – 1068
- Optimum –



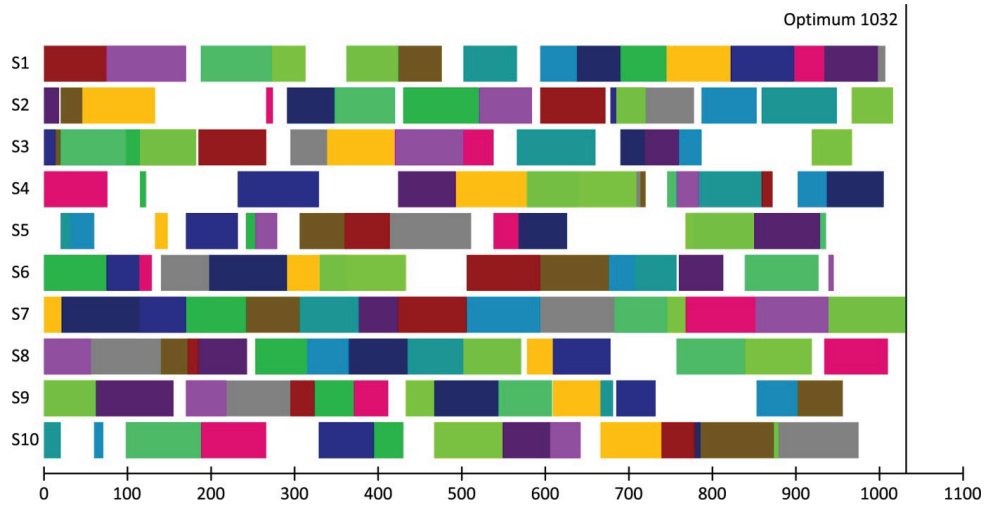
Slika 7.5.21. *la21* (1068 vremenskih jedinica)

Problem *la22* (setb2; B2)

- Standardna devijacija – 7,813
- Aritmetička srednja vrijednost – 980,82 (odstupanje od optimuma – 5,806%)
- Medijan – 982,5 (odstupanje od optimuma – 5,987%)
- Maksimalna vrijednost – 994 (odstupanje od optimuma – 7,228%)
- Minimalna vrijednost – 956 (odstupanje od optimuma – 3,128%)
- Optimum – 927



Slika 7.5.22. *la22* (956 vremenskih jedinica)



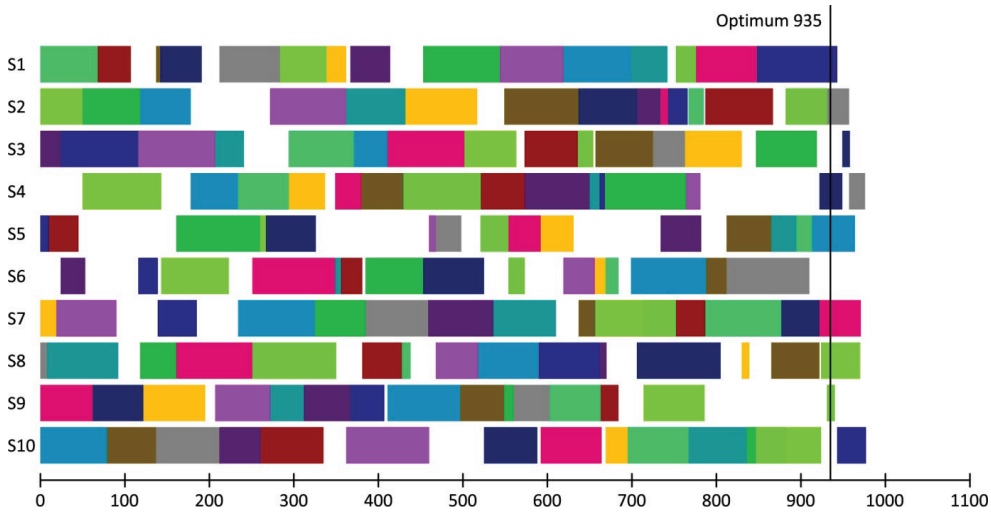
Slika 7.5.23. *la23* (1032 vremenske jedinice)

Problem *la23* (setb3; B3)

- Standardna devijacija – 8,484
- Aritmetička srednja vrijednost – 1041,34 (odstupanje od optimuma – 0,905%)
- Medijan –1039 (odstupanje od optimuma – 0,678%)
- Maksimalna vrijednost – 1061 (odstupanje od optimuma – 2,810%)
- Minimalna vrijednost – 1032 (odstupanje od optimuma – 0,000%)
- Optimum – 1032

Problem *la24* (setb4; B4)

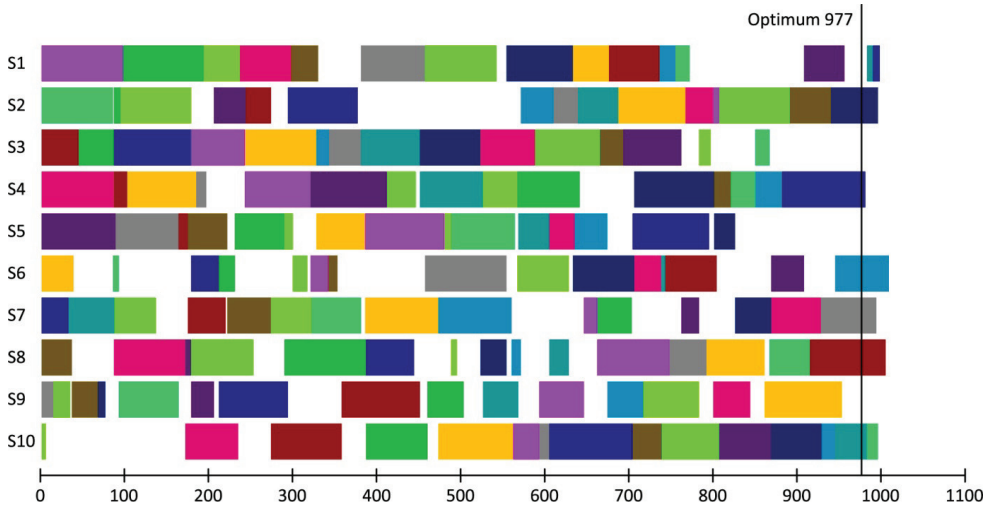
- Standardna devijacija – 8,875
- Aritmetička srednja vrijednost – 998.7 (odstupanje od optimuma – 6,813%)
- Medijan –1002 (odstupanje od optimuma – 7,166%)
- Maksimalna vrijednost – 1020 (odstupanje od optimuma – 9,091%)
- Minimalna vrijednost – 977 (odstupanje od optimuma – 4,492%)
- Optimum – 935



Slika 7.5.24. *la24* (977 vremenskih jedinica)

Problem *la25* (setb5; B5)

- Standardna devijacija – 11.230
- Aritmetička srednja vrijednost – 1026,2 (odstupanje od optimuma – 5,036%)
- Medijan – 1024 (odstupanje od optimuma – 4,811%)
- Maksimalna vrijednost – 1064 (odstupanje od optimuma – 8,905%)
- Minimalna vrijednost – 1008 (odstupanje od optimuma – 3,173%)
- Optimum – 977



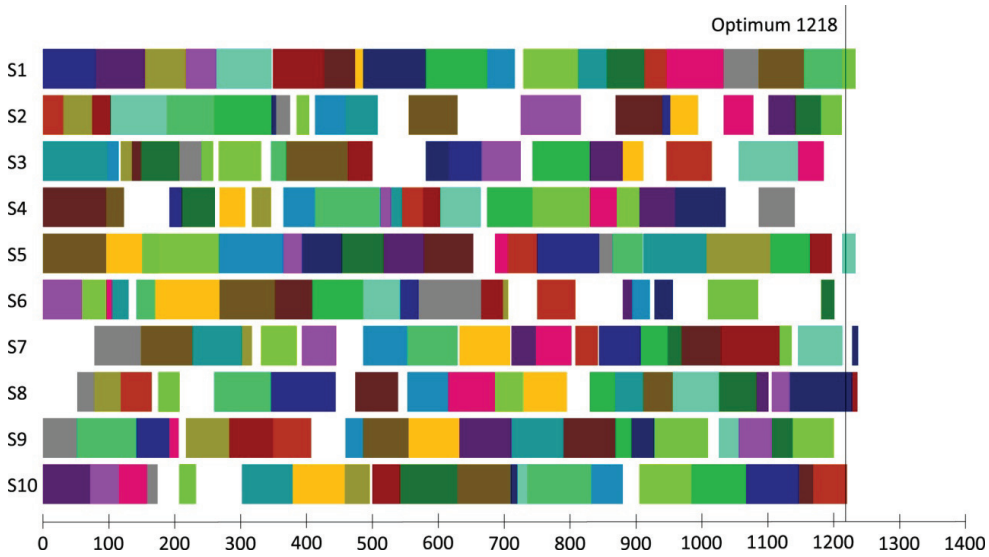
Slika 7.5.25. *la25* (1008 vremenskih jedinica)

7.5.7. La 20x10 problemi

Lawrence 20x10 klasa problema rabi kromosome dužine 200 gena te je evolucija trajala 750 generacija. Od pet problema u ovoj klasi čak za tri nije poznato optimalno rješenje tako da je procjena djelotvornosti algoritma na ovoj klasi otežana. Preostala dva problema iz ove klase za koja su optimalna rješenja poznata, algoritam je riješio zadovoljavajuće. Naime, za svaki problem algoritam je pokrenut 50 puta i iako optimalno rješenje nije pronađeno, sva ponuđena rješenja (rasporedi) nisu odstupala od optimalnog rješenja više od 5%, što se može smatrati dobrim rezultatom.

Problem *la26* (setc1; C1)

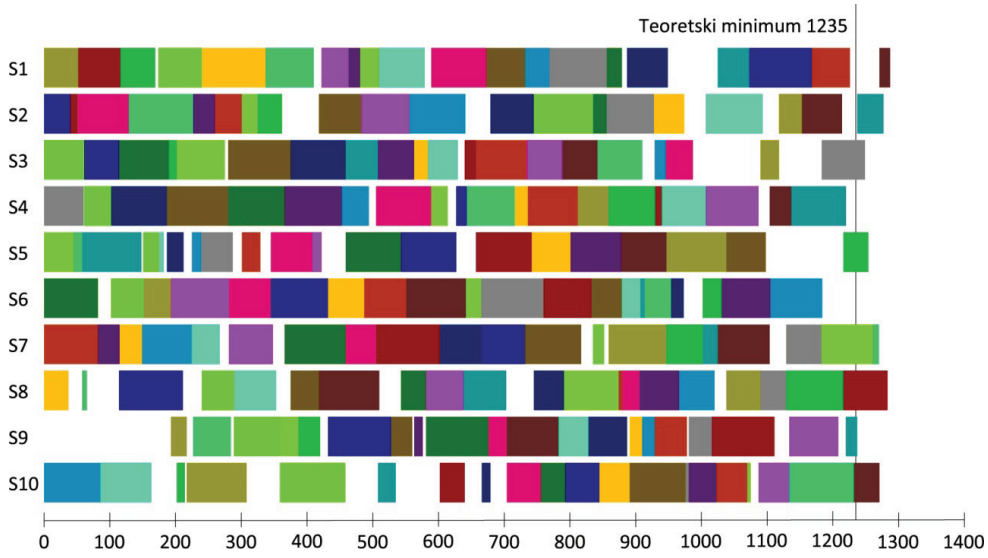
- Standardna devijacija – 12.389
- Aritmetička srednja vrijednost – 1257,2 (odstupanje od optimuma – 3,218%)
- Medijan – 1265,5 (odstupanje od optimuma – 3,900%)
- Maksimalna vrijednost – 1278 (odstupanje od optimuma – 4,926%)
- Minimalna vrijednost – 1237 (odstupanje od optimuma – 1,560%)
- Optimum – 1218



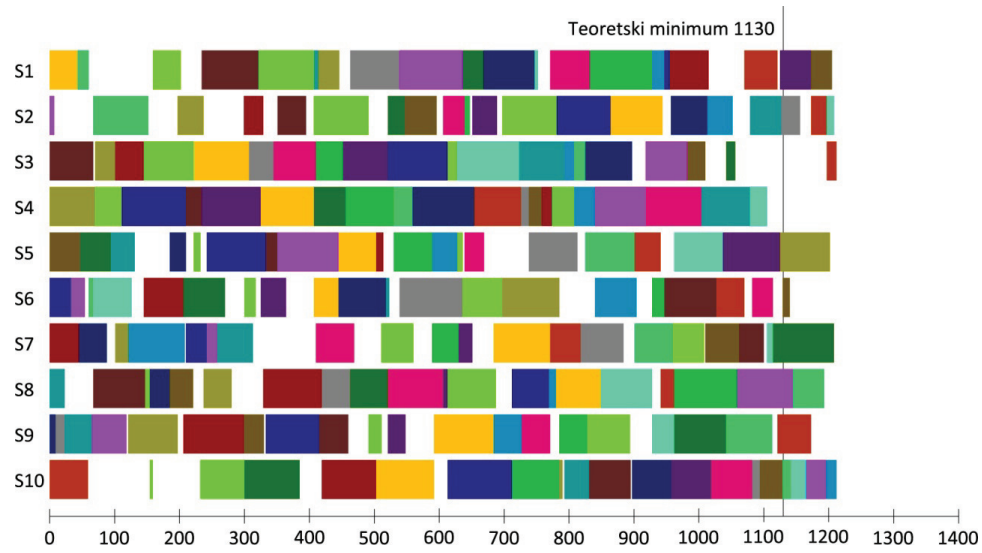
Slika 7.5.26. *la26* (1237 vremenskih jedinica)

Problem *la27* (setc2; C2)

- Standardna devijacija – 8,343
- Aritmetička srednja vrijednost – 1300,1
- Medijan –1299,5
- Maksimalna vrijednost – 1315
- Minimalna vrijednost – 1287
- Optimum –



Slika 7.5.27. *la27* (1287 vremenskih jedinica)



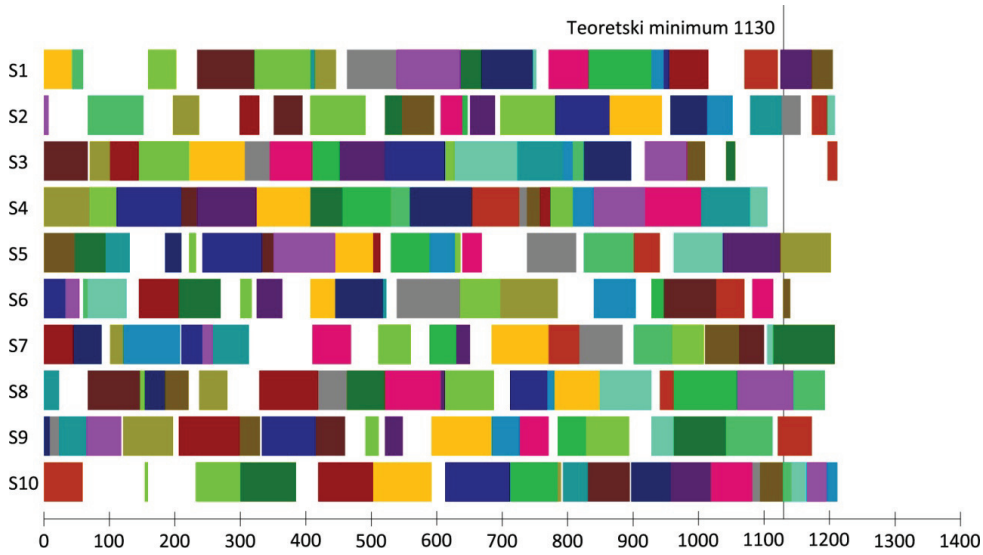
Slika 7.5.28. *la28* (1252 vremenske jedinice)

Problem *la28* (setc3; C3)

- Standardna devijacija – 14,365
- Aritmetička srednja vrijednost – 1278,2 (odstupanje od optimuma – 5,197%)
- Medijan –1278 (odstupanje od optimuma – 5,099%)
- Maksimalna vrijednost – 1307 (odstupanje od optimuma – 7,484%)
- Minimalna vrijednost – 1252 (odstupanje od optimuma – 2,961%)
- Optimum – 1216

Problem *la29* (setc4; C4)

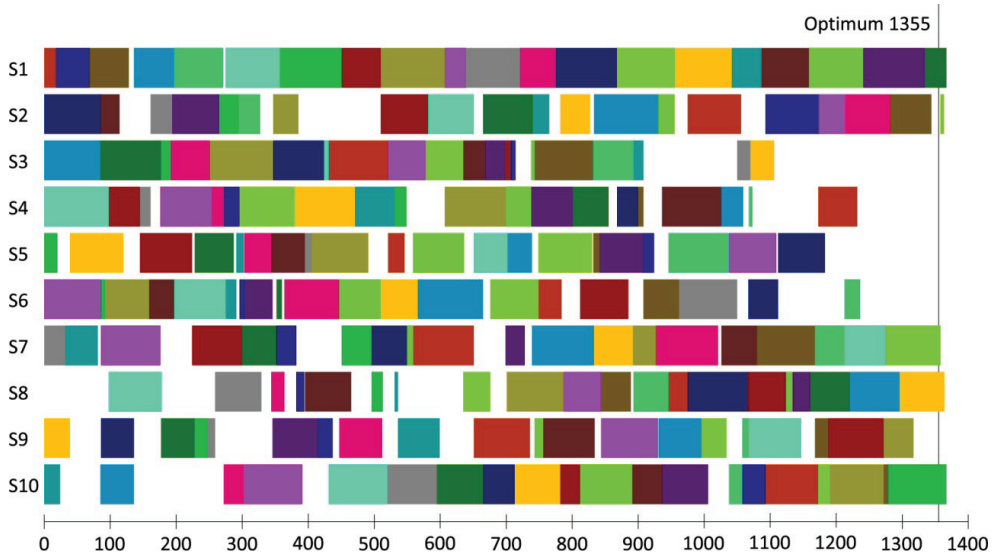
- Standardna devijacija – 13,757
- Aritmetička srednja vrijednost – 1248,8
- Medijan –1251
- Maksimalna vrijednost – 1276
- Minimalna vrijednost – 1212
- Optimum –



Slika 7.5.29. *la29* (1212 vremenskih jedinica)

Problem *la30* (setc5; C5)

- Standardna devijacija – 13,272
- Aritmetička srednja vrijednost – 1396,4 (odstupanje od optimuma – 3,055%)
- Medijan –1397,5 (odstupanje od optimuma – 3,137%)
- Maksimalna vrijednost – 1418 (odstupanje od optimuma – 4,649%)
- Minimalna vrijednost – 1367 (odstupanje od optimuma – 0,886%)
- Optimum – 1355



Slika 7.5.30. *la30* (1367 vremenskih jedinica)

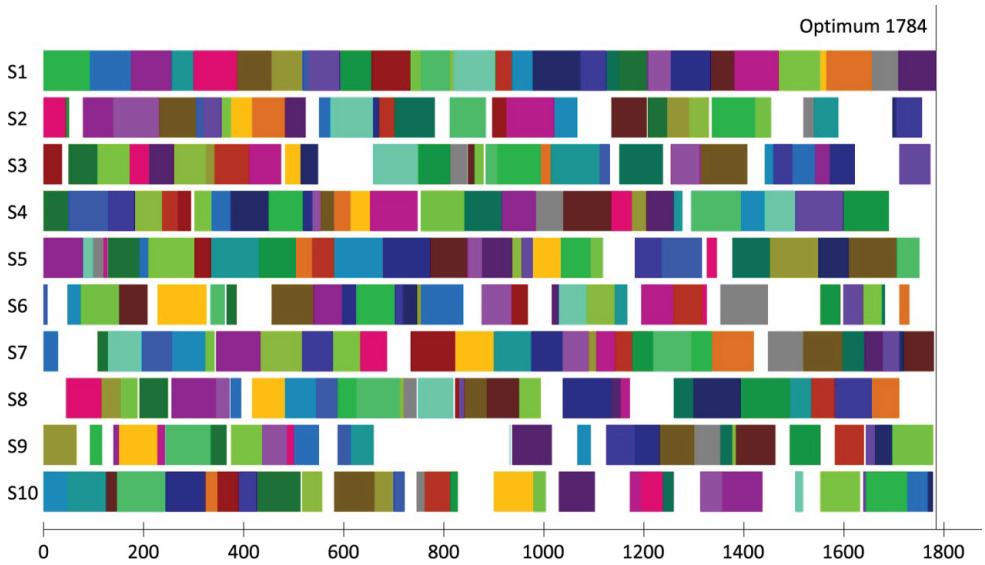
7.5.8. La 30x10 problemi

Lawrence 30x10 klasa problema rabi kromosome dužine 300 gena te je evolucija trajala 1000 generacija. Od svih ispitanih klasa problema ova rabi najduže kromosome, što znači je prostor pretraživanja najveći. Kako je i dalje veličina populacije držana na skromnih 300 jedinki, bilo je nužno produžiti trajanje evolucije na čak 1000 generacija. Rezultati ispitivanja opravdavaju odluku o

produženju evolucije jer su za svih pet problema pronađeni optimalni rasporedi. Za takav rezultat dijelom je zaslužan i veći »stupanj slobode« promatrane klase problema. Naime ova klasa problema ima povoljan odnos broja poslova i broja strojeva.

Problem *la31* (setd1; D1)

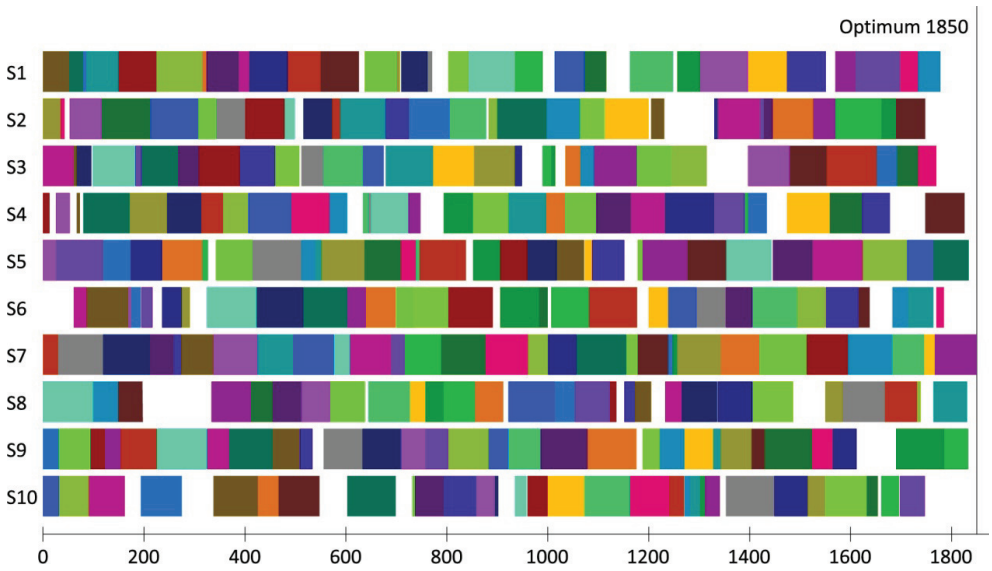
- Standardna devijacija – 2,158
- Aritmetička srednja vrijednost – 1785,5 (odstupanje od optimuma – 0,084%)
- Medijan –1784 (odstupanje od optimuma – 0,000%)
- Maksimalna vrijednost – 1791 (odstupanje od optimuma –0,392%)
- Minimalna vrijednost – 1784 (odstupanje od optimuma – 0,000%)
- Optimum – 1784



Slika 7.5.31. *la31* (1784 vremenske jedinice)

Problem *la32* (setd2; D2)

- Standardna devijacija – 3,815
- Aritmetička srednja vrijednost – 1852,3 (odstupanje od optimuma – 0,124%)
- Medijan – 1850 (odstupanje od optimuma – 0,000%)
- Maksimalna vrijednost – 1964 (odstupanje od optimuma – 6,162%)
- Minimalna vrijednost – 1850 (odstupanje od optimuma – 0,000%)
- Optimum – 1850

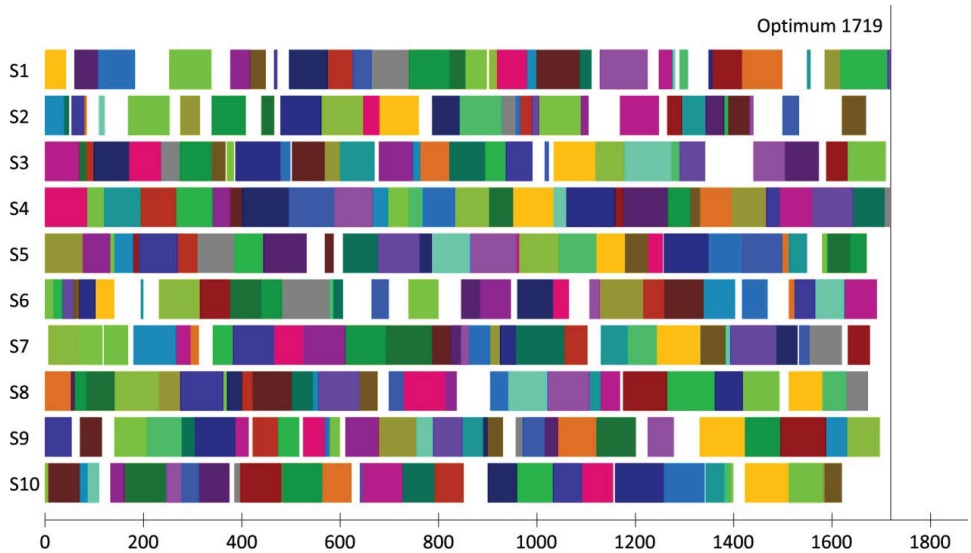


Slika 7.5.32. *la32* (1850 vremenskih jedinica)

Problem *la33* (setd3; D3)

- Standardna devijacija – 4,433
- Aritmetička srednja vrijednost – 1720,3 (odstupanje od optimuma – 0,076%)
- Medijan –1719 (odstupanje od optimuma – 0,000%)

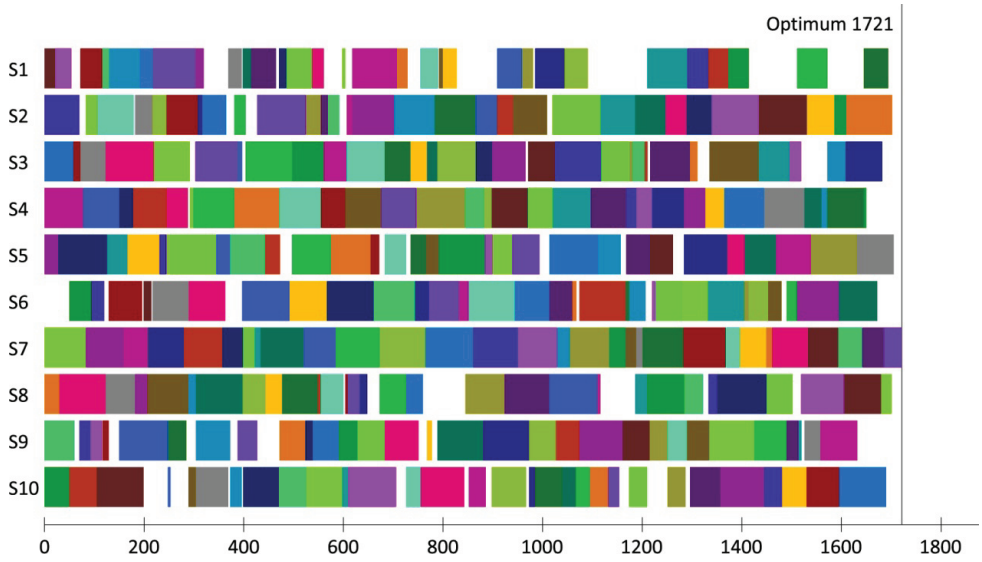
- Maksimalna vrijednost – 1736 (odstupanje od optimuma – 0,989%)
- Minimalna vrijednost – 1719 (odstupanje od optimuma – 0,000%)
- Optimum – 1719



Slika 7.5.33. *la33* (1719 vremenskih jedinica)

Problem *la34* (setd4; D4)

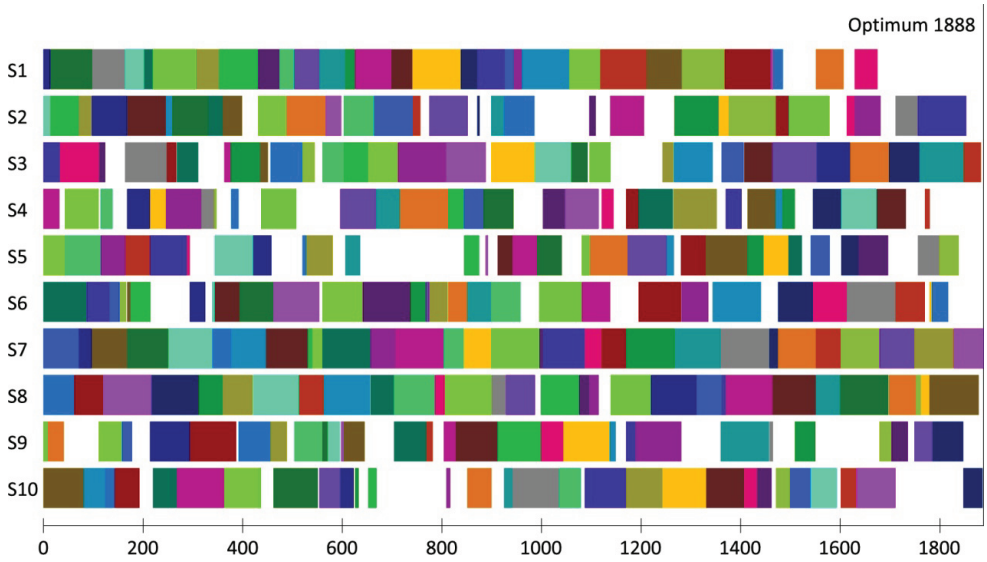
- Standardna devijacija – 10,632
- Aritmetička srednja vrijednost – 1733,3 (odstupanje od optimuma – 0,715%)
- Medijan –1730 (odstupanje od optimuma – 0,523%)
- Maksimalna vrijednost – 1761 (odstupanje od optimuma – 2,324%)
- Minimalna vrijednost – 1721 (odstupanje od optimuma – 0,000%)
- Optimum – 1721



Slika 7.5.34. *la34* (1721 vremenska jedinica)

Problem *la35* (setd5; D5)

- Standardna devijacija – 9,649
- Aritmetička srednja vrijednost – 1898,1 (odstupanje od optimuma – 0,535%)
- Medijan –1898 (odstupanje od optimuma – 0,530%)
- Maksimalna vrijednost – 1928 (odstupanje od optimuma – 2,119%)
- Minimalna vrijednost – 1888 (odstupanje od optimuma – 0,000%)
- Optimum – 1888



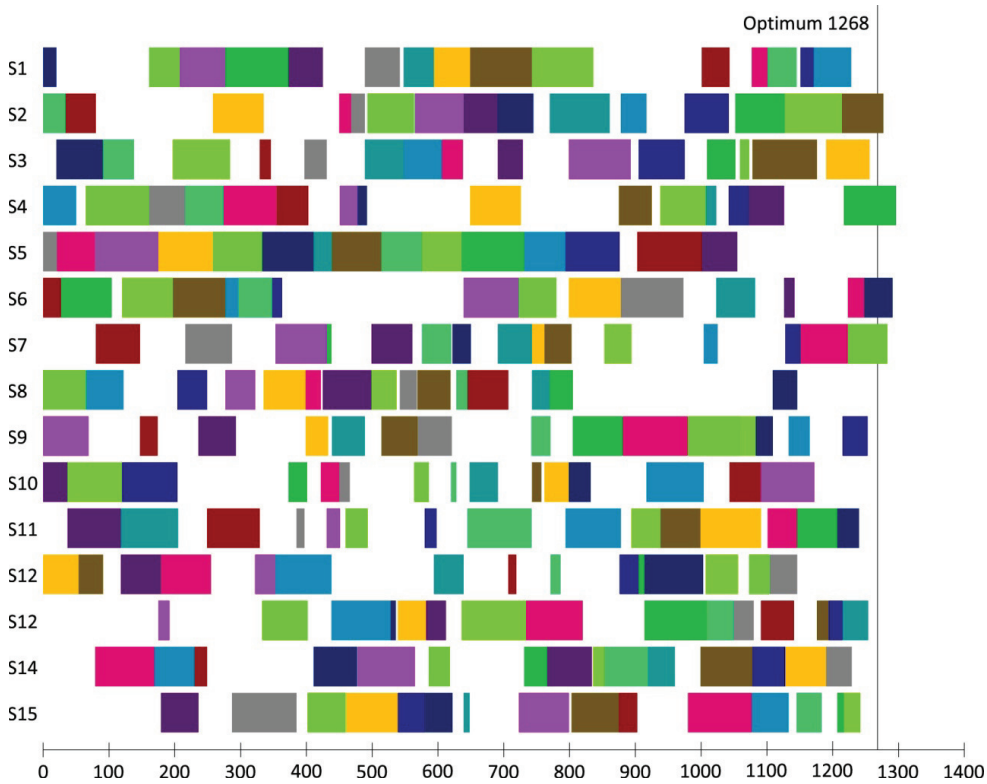
Slika 7.5.35. *la35* (1888 vremenskih jedinica)

7.5.9. La 15x15 problemi

Lawrence 15x15 klasa problema rabi kromosome dužine 225 gena te je evolucija trajala 800 generacija. *La 15x15* najteža je klasa problema. Zbog veličine problema i izrazito nepovoljna odnosa broja poslova i broja strojeva algoritam je pokazao malo lošije rezultate. Naime, maksimalno odstupanje od optimalnog rješenja kod *la39* i *la40* iznosilo je više od 8%, što nadilazi očekivane vrijednosti. Ovdje je važno napomenuti da najbolja rješenja ipak zadovoljavaju zadane kriterije. Problem se može riješiti povećanjem populacije, iznad 300 jedinki, ali bi to zbog načina stvaranja niša usporilo rad algoritma.

Problem *la36* (seti1; I1)

- Standardna devijacija – 10,099
- Aritmetička srednja vrijednost – 1319,9 (odstupanje od optimuma – 4,093%)
- Medijan –1322 (odstupanje od optimuma – 4,259%)
- Maksimalna vrijednost – 1335 (odstupanje od optimuma – 5,284%)
- Minimalna vrijednost – 1296 (odstupanje od optimuma – 2,208%)
- Optimum – 1268



Slika 7.5.36. *la36* (1296 vremenskih jedinica)

Problem *la37* (seti2; I2)

- Standardna devijacija – 10,873
- Aritmetička srednja vrijednost – 1462,1 (odstupanje od optimuma – 4,660%)
- Medijan –1460 (odstupanje od optimuma – 4,510%)
- Maksimalna vrijednost – 1490 (odstupanje od optimuma – 6,657%)
- Minimalna vrijednost – 1449 (odstupanje od optimuma – 3,722%)
- Optimum – 1397

Slika 7.5.37. *la37* (1449 vremanskih jedinica)

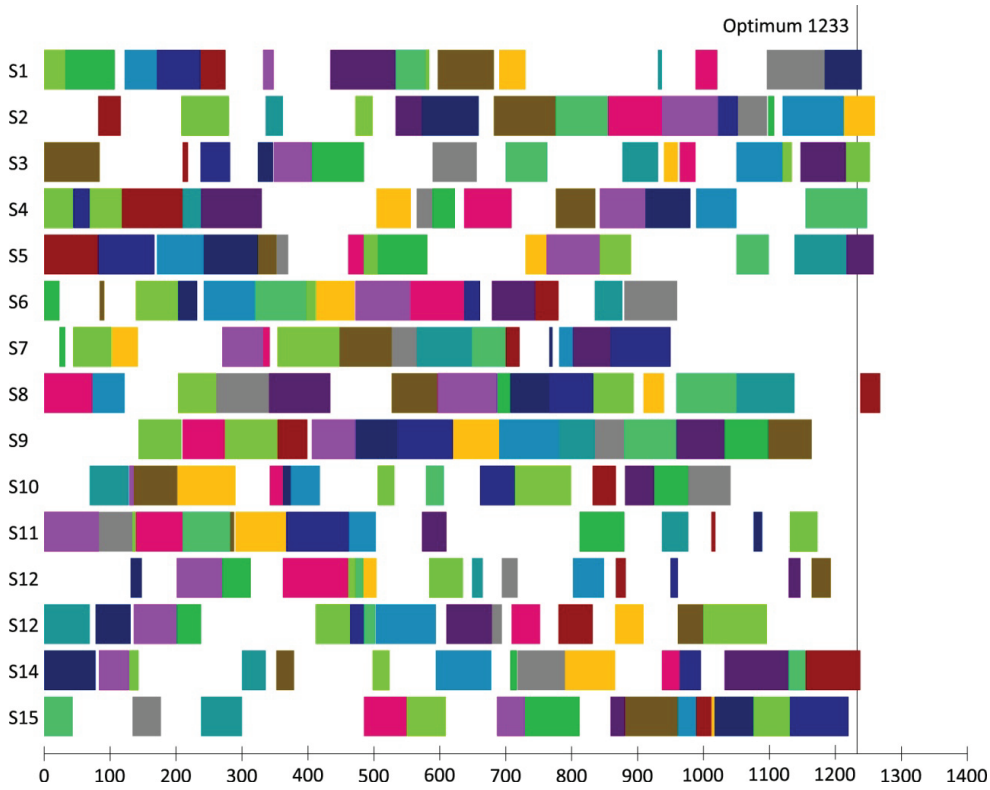
Problem *la38* (seti3; I3)

- Standardna devijacija – 12,967
- Aritmetička srednja vrijednost – 1277,3
- Medijan –1278
- Maksimalna vrijednost – 1303
- Minimalna vrijednost – 1252
- Optimum –

Slika 7.5.38. *la38* (1252 vremenske jedinice)

Problem *la39* (seti4; I4)

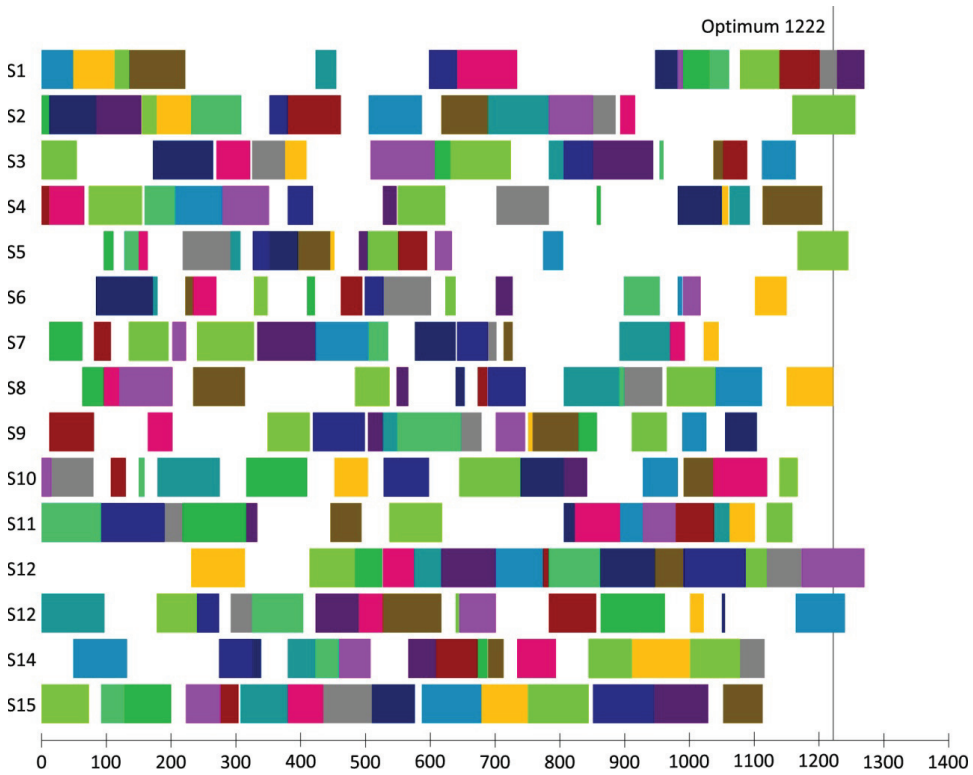
- Standardna devijacija – 14,753
- Aritmetička srednja vrijednost – 1309,9 (odstupanje od optimuma – 6,237%)
- Medijan – 1312 (odstupanje od optimuma – 6,407%)
- Maksimalna vrijednost – 1335 (odstupanje od optimuma – 8,273%)
- Minimalna vrijednost – 1268 (odstupanje od optimuma – 2,839%)
- Optimum – 1233



Slika 7.5.39. *la39* (1268 vremenskih jedinica)

Problem *la40* (seti5; I5)

- Standardna devijacija – 11,357
- Aritmetička srednja vrijednost – 1286 (odstupanje od optimuma – 5,237%)
- Medijan –1284 (odstupanje od optimuma – 5,074%)
- Maksimalna vrijednost – 1326 (odstupanje od optimuma – 8,511%)
- Minimalna vrijednost – 1270 (odstupanje od optimuma – 3,928%)
- Optimum – 1222



Slika 7.5.40. *la40* (1270 vremenskih jedinica)

8. ZAKLJUČAK

U vrijeme globalizacije, kada konkurencija postaje sve veća, potreba za optimalnom uporabom resursa postaje imperativ. Tema je ovoga doktorskog rada rješavanje problema raspoređivanja operacija po strojevima radi optimalna korištenja raspoloživih resursa. Problem nije jednostavno rješiv jer s računalnoga gledišta pripada klasi NP-teških problema, što znači da ga nije moguće egzaktno riješiti ni za relativno mali broj poslova i strojeva. Naime, vrijeme potrebno za izračun optimalnog rješenja eksponencijalno raste s veličinom problema, što na kraju donosi neprihvatljivo dugo trajanje izračuna.

Za uspješno rješavanje problema raspoređivanja u ovom doktorskom radu predložen je genetički algoritam s nišama. Genetički algoritmi s nišama razlikuju se od jednostavnog genetičkog algoritma utoliko što ne dopuštaju da najbolje prilagođena jedinka dominira procesom selekcije. To je svojstvo iznimno važno jer posredno smanjuje mogućnost da algoritam zapne u jednom od manje poželjnih lokalnih ekstrema. Razni istraživači predložili su različite metode kreiranja niša. Kao što je u radu opisano, algoritam s raspodjelom funkcije cilja najpoznatija je i najčešće primijenjena metoda. Na žalost, ta metoda ima nekoliko bitnih nedostataka. Prije svega, za određivanje pripadnosti jedinke pojedinoj niši, potrebno je svaku pojedinu jedinku (raspored operacija po strojevima) usporediti sa svim ostalim jedinkama u populaciji, što kod složenijih problema nije jednostavno. Za uspješnu usporedbu dviju jedinki, odnosno određivanje njihove sličnosti, nužno je barem djelomično poznavati prirodu problema koji se rješava, neovisno o tome rabi li se fenotipska ili genotipska usporedba. U konkretnom slučaju fenotipska usporedba članova populacije znači određivanje stupnja sličnosti dvaju Ganttovih dijagrama. S druge strane, genotipska usporedba odnosi se na usporedbu tih istih članova populacije, ali na razini kromosoma. Obje navedene usporedbe mogu u određenoj mjeri biti problematične. Naime, dva genotipski bitno različita rasporeda mogu rezultirati identičnim Ganttovim dijagramom. Pojednostavnjeno to znači da fenotipski zapis rješenja (Ganttov dijagram) pokazuje znatno veći stupanj sličnosti u odnosu na istovjetni genotipski zapis (kromosomi). Kako se ni jedna metoda nije pokazala zadovoljavajućom, u ovom

doktorskom radu predložena je nova metoda usporedbe zasnovana na generacijskoj udaljenosti između jedinki. Time je posve izbjegnuta potreba za poznavanjem problema koji se rješava. Algoritmu je potpuno svejedno rješava li problem raspoređivanja operacija po strojevima ili primjerice problem trgovačkog putnika. Na žalost, algoritam je zbog predložene metode uspoređivanja jedinki postao bitno složeniji. Naime, da bi se uspješno rabila generacijska udaljenost kao mjera sličnosti, jedinke u populaciji moraju imati diploidne kromosome, što bitno usložnjava rad algoritma.

Uvedeno je i vremenski regulirano skaliranje vrijednosti funkcije cilja, što omogućuje progresivno povećanje pritiska selekcije. Na početku evolucije pritisak selekcije manji je kako bi se omogućilo pravilno kreiranje niša, a pred kraj algoritma, kada se u populaciji u većoj mjeri očekuju samo dobro prilagođene jedinke (kraći Ganttovi dijagrami), pritisak selekcije povećava se kako bi se izbjegla stohastičnost procesa selekcije. Posljedica je takva progresivna povečanja pritiska selekcije demontiranje niša te povečana eksploatacija samo jedne od njih, radi pronalaženja globalnog opimuma.

Ispravnost predloženog modela uspješno je ispitana na dvjema skupinama *benchmark* problema: *mt* i *la*. Za navedene probleme postoje točno definirana optimalna rješenja (rasporedi) te se njihovom uporabom može usporediti predloženi algoritam s algoritmima ostalih istraživača. Rezultati dobiveni usporedbom predloženoga genetičkog algoritma i jednostavnoga genetičkog algoritma pokazuju da je hipoteza rada ispravna, odnosno da predloženi genetički algoritam ima manje rasipanje rješenja te je Gaussova krivulja normalne razdiobe pomaknuta u lijevo prema optimalnom rješenju. Činjenica da predloženi algoritam ima manje rasipanje upozorava na veću pouzdanost predloženog algoritma, što je u vremenu globalizacije vrlo bitno svojstvo. Algoritam je također uspoređen s poznatim GT genetičkim algoritmom te je pokazao približno jednake rezultate. Budući da je utjecaj veličine populacije na brzinu izvođenja predloženog algoritma izrazito negativan, veličina populacije u predloženom algoritmu držana je relativno nisko u odnosu na GT genetički algoritam, čime je dobiveni rezultat još važniji. Provjera ponašanja algoritma ovisno o veličini problema provedena je na *la benchmark* problemima. Dobiveni rezultati pokazuju da na kvalitetu rješenja bitno ne utječe veličina problema. Tako primjerice algoritam pronalazi optimalna rješenja za dvije bitno različite klase problema: *la06-la10* (15x5) i *la31-la35* (30x10), što navodi na zaključak da se predloženi genetički algoritam može primijeniti na problemima iz prakse.

8.1. ZNANSTVENI DOPRINOS

Osim što je algoritam uspješno primijenjen u rješavanju problema raspoređivanja operacija po strojevima moguće ga je bez dodatne prilagodbe iskoristiti za velik broj drugih permutacijskih problema poput planiranja ruta ili problema trgovačkog putnika. Za stvaranje takvog algoritma bilo je potrebno riješiti dva bitna problema. Prvi se problem svodio na određivanje sličnosti dviju jedinki u populaciji, što je uvjet za stvaranje niša, bez poznavanja problema koji se rješava. Problem je riješen uvođenjem tzv. Hamiltonove sličnosti, čime su posve odbačene relativno problematične fenotipske i genotipske usporedbe. Uvođenje Hamiltonove sličnosti stvorilo je dodatan problem. Naime, Hamiltonova se sličnost može primijeniti, samo ako su rješenja problema zapisana u obliku diploidnog kromosoma što dodatno usložnjava algoritam zbog uvođenja problema dominacije. Mapa dominacije te odabir odgovarajućeg modela prezentacije kromosoma predstavlja drugi bitan problem riješen u ovom doktorskom radu. Za model prezentacije odabrano je »kodiranje prema referentnoj listi«, što je omogućilo uporabu klasičnih operatora križanja i mutacije. Također je predložena proporcionalna mapa dominacije koja ne favorizira ni jednu vrijednost gena. Naime, sve vrijednosti gena jednako su često dominantne kao i recesivne.

Genetički algoritam predložen u ovom doktorskom radu zbog svoje univerzalnosti bitno pridonosi uspješnu rješavanju velikog broja strojarskih permutacijskih problema koje nije moguće egzaktno riješiti.

8.2. SMJERNICE DALJNJEG RADA

Iako je algoritam pokazao veoma dobre rezultate, još ima dosta prostora za unapređivanje. Prije svega shema dominacije definirana je poprilično arbitrarno te nije podvrgnuta procesu evolucije. Naime, iako je shema dominacije definirana tako da se ne favorizira ni jedna vrijednost gena (svi su geni jednaki broj puta dominantni kao i recesivni), evoluiranje sheme dominacije također je problem vrijedan daljnjeg istraživanja. Drugo, iz generacije u generaciju mijenjala se cijela populacija, što u prirodnom svijetu nije uobičajeno. Osim u nekih vrsta riba (npr. losos), takva se radikalna izmjena generacija ne pojavljuje. Naime, u prirodi je uobičajeno da roditelji brane svoje potomke ili im

ustupaju dio svojih resursa dok dovoljno ne ojačaju za samostalan život. Utjecaj živih roditelja na razvoj i uspješnost potomstva jedan je od aspekata algoritma s nišama koji bi također vrijedilo ispitati. Poznavanje cjelokupnog rodoslovnog stabla svake pojedine jedinke osnovni je uvjet za izmjenu resursa između roditelja i potomaka. Kako se u predloženom algoritmu sličnost jedinki definira njihovom generacijskom udaljenošću, svi uvjeti za uvođenje predloženog mehanizma već su ostvareni.

Za konačnu ocjenu vrijednosti predloženoga genetičkog algoritma nužno je ispitivanje na situacijama iz stvarnoga života koje imaju veći broj ograničenja, veći prostor pretraživanja rješenja i vremenski promjenjive funkcije cilja. Sama činjenica da predloženi genetički algoritam rabi diploidne kromosome ide u prilog tvrdnji da je u algoritam implicitno ugrađena veća mogućnost prilagodbe na dinamički definiran okoliš (vremenski promjenjiva funkcija cilja).

9. BIBLIOGRAFIJA

[1] Adaptation in Natural and Artificial Systems. Holland, J.H. Ann Arbor: The University of Michigan Press, 1975.

[2] Goldberg, David E. Genetic Algorithms in Search, Optimization, and Machine Learning. s.l.: Addison-Wesley, 1986.

[3] Mitchell, Melanie. An Introduction to Genetic Algorithms. Cambridge, Massachusetts: The MIT Press, 1998.

[4] Alleles, loci, and the traveling salesman problem. Goldberg, David E. and Lingle, R. 1985. Proceeding of an International Conference on Genetic Algorithms and their Applications. pp. 154–159.

[5] Job shop scheduling with genetic algorithms. Davis, L. 1985. Proceeding of an International Conference on Genetic Algorithms and their Applications. pp. 136–140.

[6] Designer Genetic Algorithms: Genetic Algorithms in Structure Design. Louis, S.J. and Rowlins, G.J.E. s.l.: Morgan Kauffman, 1991. Proceeding. of Fourth International Conference on Genetic Algorithms. pp. 53–60.

[7] Multiple vehicle routing with time and capacity constraints using genetic algorithms. Blanton, J.L. and Wainwright, R.L. San Francisco: Morgan Kaufmann, 1993. Proceedings of the 5th International Conference on Genetic Algorithms.

[8] Haupt, Randy L. and Haupt, Sue Ellen. Practical genetic algorithms. Hoboken, New Jersey: John Wiley & Sons, Inc, 2004.

[9] Pinado, Michael L. Theory, Algorithms, and Systems. s.l.: Springer, 1995. ISBN: 978-0-387-78934-7.

[10] Cook, Stephen. The Complexity of Theorem Proving Procedures. Seminar Toronto University. 1971.

[11] Karp, Richard. Reducibility Among Combinatorial Problem. Complexity of Computer Computations. 1972.

[12] Rose, Oliver. THE SHORTEST PROCESSING TIME FIRST (SPTF) DIS-PATCH RULE AND SOME VARIANTS IN SEMICONDUCTOR MANUFACTURING. Proceedings of the 2001 Winter Simulation Conference. 2001.

[13] Du, J. and Leung, Y.T. Minimizing total tardiness on one processor is NP-hard. Math. Operat. Res. 1990, pp. 483–495.

[14] Yamada, T. Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems. Kyoto: s.n., 2003.

[15] Mattfeld, D. and Bierwirth, C. An efficient genetic algorithm for job shop scheduling with tardiness objectives. European Journal of Operational Research. 2002, pp. 616–630.

[16] Panwalkar, S.S. and Iskander, W. A survey of scheduling rules. Operations Research. 1977, pp. 45–61.

[17] Sule, Dileep R. Production Planning And Industrial Scheduling: Examples, Case Studies And Applications. s.l.: CRC Press, 1997. ISBN: 1420044206.

[18] Whitley, Darrell. A Genetic Algorithm Tutorial. s.l.: Computer Science Department, Colorado State University, 1994.

[19] Bäck, T. Evolutionary Algorithms in Theory and Practice; Evolution Strategies, Evolutionary Programming, Genetic Algorithms. s.l.: Oxford University Press, 1996. ISBN 0-19-509971-0.

[20] Bodenhofer, U. Genetic Algorithms: Theory and Applications. Linz: Fuzzy Logic Laboratorium, 2003.

[21] Reducing bias and inefficiency in selection algorithms. Genetic algorithms and their applications. Baker, J.E. 1987. Proceedings of the Second International Conference on Genetic Algorithms.

[22] Finite Markov chain analysis of genetic algorithms. Goldberg, David E. and Segrest, P. Hillsdale: Lawrence Erlbaum Associates, 1987. Proceedings of the Second International Conference on Genetic Algorithms. pp. 1–8.

[23] Horn, J. Genetic Algorithms and the Evolution of Optimal Cooperative Populations. 1997.

[24] Brindle, A. Genetic Algorithms for Function Optimization. Edmonton: University of Alberta, Department of Computer Science, 1981.

[25] Fang, H.L. Genetic Algorithms in Timetabling and Scheduling. Edinburgh: University of Edinburgh, 1994.

[26] Applying Adaptive Algorithms to Epistatic Domains. Davis, L. 1985. Proceedings of the Ninth International Conference on Genetic Algorithms. pp. 162–164.

[27] Syswerda, G. Schedule Optimisation Using Genetic Algorithms. [book auth.] L. Davis. Handbook of Genetic Algorithms. s.l.: International Thomson Computer Press, 1996, pp. 335–349.

[28] A Study of Permutation Crossover Operators on the Traveling Salesman Problem. Oliver, I., Smith, D. and Holland, J. 1987. Proc. Second International Conference on Genetic Algorithms and their Applications.

[29] A genetic algorithm for job shop. Falkenauer, E. and Bouffoix, S. 1991. Proceedings of the 1991 IEEE International Conference on Robotics and Automation. pp. 824–829.

[30] An effcient genetic algorithm for job shop scheduling problems. Kobayashi, S., Ono, I. and Yamamura, M. Pittsburgh: s.n., 1995. Proceedings of 6th International Conference on Genetic Algorithms and their Applications. pp. 506–511.

[31] A genetic algorithm for job-shop scheduling problems using job-based order crossover. Ono, I., Yamamura, M. and Kobayashi, S. 1996. Proceedings of 1996 ICEC. pp. 547–552.

[32] Etiler, O., et al. A genetic algorithm for flow shop scheduling problems. Journal of the Operational Research Society. 2004, pp. 830–835.

[33] Gandibleux, X., et al. Metaheuristics for multiobjective optimisation. s.l.: Springer, 2004. p. 249.

[34] Solving job-shop scheduling problem using genetic algorithms. Gen, M., Tsujimura, Y. and Kubota, E. Ashikaga: s.n., 1994. Proc. of the 16th Int. Conf. on Computer and Industrial Engineering. pp. 576–579.

[35] A genetic based hybrid scheduler forgenerating static schedules in manufacturing contexts. Holsapple, C., et al. 1993. IEEE transactions on System, Man, Cybernetics. Vol. 23, pp. 953–971.

[36] Conventional genetic algorithm for job shop problems. Nakano, R. and Yamada, T. 1991. Proceedings of International Conference on Genetic Algorithms (ICGA '91). pp. 474–479.

[37] Evolution based learning in a job shop scheduling environment. Dondorf, U. and Pesch, E. 1995. Computer & Operations Research. pp. 25–40.

[38] Yamada, T. and Nakano, R. A genetic algorithm applicable to large-scale job-shop problems. [ed.] R. Männer and B. Manderick. Parallel Problem Solving from Nature 2. 1992, pp. 281–290.

[39] Falkenauer, E. Applying Genetic Algorithms to Realworld Problems. Brussels: Depertment of Applied Mechanics, Brussels University, 1996.

[40] Harik, B. Finding Multiple Solutions In Problems Of Bounded Difficulty. University of Illinois. 1994. IlliGAL Report No. 94002.

[41] De Jong, A. K. An analysis of the behavior of a class of genetic adoptive systems. Dissertation Abstracts International 36. 1975.

[42] Mahfoud, S.W. Crowding and preselection revisited. [ed.] R. Männer and B. Manderick. Parallel Problem Solving From Nature 2. 1992.

[43] Watson, R.A. Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis. 2002.

[44] Brownlee, J. Parallel niching genetic algorithms: a crowding perspective. s.l.: Swinburne University of Technology, 2004.

[45] Genetic algorithms with sharing for multimodal function optimization. Goldberg, D. and Richardson, J. Cambridge: L. Erlbaum Associates Inc., 1987. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. pp. 41–49.

[46] Mahfoud, S.W. Niching Methods for Genetic Algorithms. s.l.: Department of General Engineering, University of Illinois at Urbana Champaign, 1995.

[47] Ursem, R.K. When Sharing Fails. s.l.: EvALife project group, Department of Computer Science; University of Aarhus; Denmark.

[48] Mendel, Johann Gregor. Versuche über Pflanzen-Hybriden. Verhandlungen des naturforschenden Vereines in Brünn. 1865, Vol. IV.

[49] Nonstationary function optimization using genetic algorithms with dominance and diploidy. Goldberg, David E. and Smith, R. Cambridge: s.n., 1987. International Conference on Genetic Algorithms. pp. 59–68.

[50] The degree of oneness. Ryan, C. 1994. Proc. of the 1994 ECAI Workshop on Genetic Algorithms.

[51] Ryan, C. Reducing Premature Convergence in Evolutionary Algorithms. s.l.: National University of Ireland, 1996.

[52] Industrial Scheduling. Muth, J.F. and Thompson, G.L. Englewood Clffs: Prentice-Hall, 1963.

[53] Yamada, T. and Nakano, R. Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems. s.l.: Kyoto University, 2003.

[54] A genetic algorithm applicable to large-scale job-shop problems. Yamada, T. and Nakano, R. 1992. Proceedings of The Second International Conference on Parallel Problem Solving from. pp. 281–290.

[55] Cavicchio, D. J. Adaptive search using simulated evolution. Ann Arbor: University of Michigan, 1970.

[56] Learninig distribute reactive strategies by genetic programming for the general job shop problem. Atlan, L., Bonnet, J. and Naillon, M. 1993. Proceedings of the Seventh Annual Florida Artificial Intelligence Research Symposium.

[57] Exploring problem-specific recombination operators for job shop scheduling. Bagchi, S., et al. [ed.] R.K. Below and L.B. Booker. Proceedings of the Fourth International Conference on Genetic Algorithms.

[58] Genetic algorithms with sharing for multimodal function optimization. Goldberg, David E. and Richardson, J. Cambridge: Proceedings of the Second International Conference on Genetic Algorithms, 1987. pp. 41–49.

[59] Cantu-Paz, Erick and Goldberg, David E. Are Multiple Runs of Genetic Algorithms Better than One? Proceedings of the Genetic and Evolutionary Computation Conference. 2003.

10. ŽIVOTOPIS

Arijan Abrashi, rođen u Zagrebu 1. 12. 1974., maturirao je 1993. godine u X. gimnaziji. Iste godine upisuje Fakultet strojarstva i brodogradnje na kojem je diplomirao 1999. godine na Katedri za energetiku. Od 2. 11. 1999. radi u Institutu za energetiku i zaštitu okoliša kao tehnički konzultant u Odjelu za sustave održavanja. Posljediplomski studij upisao je u školskoj godini 2000./ 2001. Od 2007. predavač je na Zavodu za industrijsko inženjerstvo te aktivno sudjeluje u predavanjima i vježbana iz predmeta Informacijski menadžment.

Član je Hrvatskoga društva održavatelja te je objavio tri znanstvena rada iz područja genetičkih algoritama. Govori engleski jezik.

11. BIOGRAPHY

Arijan Abrashi, born in Zagreb, December 1, 1974., finished the X. grammar school in 1993. The same year he was enrolled in the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb where he received his B.Sc. degree in 1999. Since November 2, 1999, he has been working in the Energy and Environmental Protection Institute as technical consultant in the Maintenance Management Systems Department. Since 2007 he has been working as assistant at the Department of Industrial Engineering and Management and actively participates in lectures and exercises in the course of Information Management.

He is member of Croatian Maintenance Society and he has also published 3 papers in the field of genetic algorithms. He is fluent in English.