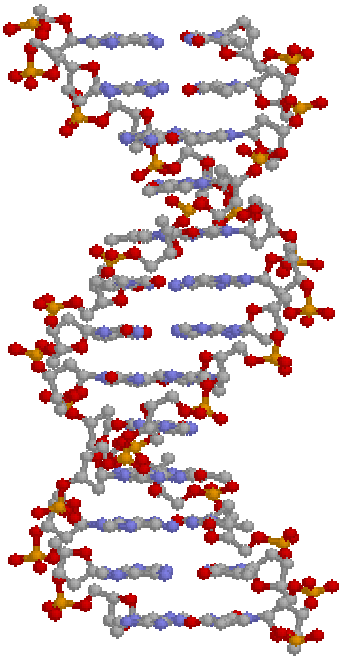


Marin Golub

Genetski algoritam

Drugi dio

Postupci selekcije
Paralelni genetski algoritmi
Novi model GPGA

Verzija 2.2 od 4. listopada 2004. godine

SADRŽAJ

POPIS OZNAKA

D	broj dretvi,
I	broj iteracija sekvencijskog genetskog algoritma,
I_U	ukupan broj iteracija,
I_E	broj evaluacija,
M	mortalitet, broj jedinki za eliminaciju ili generacijski jaz,
M_i	migracijski interval,
M_s	migracijska stopa,
N	veličina populacije,
N_P	broj procesora,
N_R	broj jedinki koje sudjeluju u selekciji i reprodukciji,
Ω	prostor rješenja,
$P_{k,j}$	vjerojatnost da će u nekom trenutku neka od $j=D$ dretvi, koje obavljaju k-turnirsku eliminacijsku selekciju i reprodukciju, obavljati posao uzalud
$P'_{k,j}$	vjerojatnost da će u nekom trenutku dretva s oznakom $j \leq D$ obavljati uzalud k-turnirsku eliminacijsku selekciju i reprodukciju
S_N	broj susjeda,
S	broj slugu,
T	trajanje preuzimanja,
T_1	trajanje izvođenja jedne iteracije,
T_c	trajanje komunikacije,
T_f	trajanje evaluacije,
T_{uk}	ukupno trajanje izvođenja programa,
a	ubrzanje,
b	broj binarnih znamenaka (broj bitova kod binarnog prikaza) ili duljina kromosoma,
d	dimenzija problema ili broj nepoznanica,
$d(x)$	funkcija dobrote,
d_s	kumulativna vrijednost dobrote,
$f(x)$	funkcija cilja,
f_i	frekvencija migracije,
k	veličina turnira ili broj jedinki koje sudjeluju u turnirskoj selekciji,
λ	broj djece,
μ	broj roditelja,
p_c	vjerojatnost križanja,
$p_{kE}(i)$	vjerojatnost eliminacije i-te jedinke uporabom k-turnirske selekcije,
$p_{kG}(i)$	vjerojatnost selekcije i-te jedinke za sljedeću generaciju uporabom k-turnirske selekcije,
p_m	vjerojatnost mutacije gena,
$p_m(i)$	vjerojatnost mutacije i gena odjednom,
p_M	vjerojatnost mutacije kromosoma,
s	selekcijska razlika,
s_I	selekcijski intenzitet,
r	reprodukcijska stopa,
t	vrijeme, redni broj iteracije ili generacije.

1. UVOD

Genetski algoritam je recept koji kazuje što treba raditi s genetskim materijalom kako bi se s određenom vjerojatnošću nakon određenog vremena postiglo zadovoljavajuće rješenje zadanog optimizacijskog problema. Genetski materijal je skup svojstava koji opisuju neku jedinku. Genetski algoritam ne određuje na koji način je genetski materijal pohranjen u radni spremnik, niti kako treba manipulirati genetskim materijalom, već samo kaže da se genetski materijal treba razmjenjivati, slučajno mijenjati, a bolje jedinke trebaju s većom vjerojatnošću preživljavati selekciju. Kako će se bolje jedinke selektirati za reprodukciju, te kako će se i s kolikim vjerojatnostima obaviti križanje i mutacija, određuje se na temelju iskustva i eksperimentalno, tj. heuristički. Stoga postoji velika sloboda u izradi genetskih algoritama. Cijena te slobode su loši ili nikakvi rezultati koji se najčešće dobivaju s genetskim algoritmom koji nema podešene parametre ili nema genetske operatore prilagođene problemu. U ovom, drugom dijelu, bavit ćemo se upravo optimiranjem samog genetskog algoritma.

Podešavanjem (optimiranjem) parametara genetskog algoritma mogu se postići zadovoljavajući rezultati. Međutim, sâm postupak podešavanja je dugotrajan proces. Primjerice, postupak podešavanja samo tri parametra genetskog algoritma opisanog u poglavlju 6 trajao je oko 100 sati, unatoč tome što je jedan evolucijski proces trajao svega nekoliko minuta. Nadalje, uporabe li se genetski algoritmi za rješavanje teških optimizacijskih problema s velikim brojem nepoznanica, postupak optimiranja može potrajati danima, a postupak podešavanja parametara mjesecima i na danas najbržim računalima. Jedan od načina kako skratiti trajanje postupka optimiranja jest paralelizacija genetskih algoritama.

Poboljšavanjem mogućnosti računala nastoje se sve teži problemi riješiti u što je moguće kraćem vremenu. Proširivanjem radnog spremnika, dodavanjem procesora te umrežavanjem računala stvara se radno okruženje u kojem se posao može brže obaviti. Evolucija u prirodi se odvija potpuno paralelno. Stoga je za očekivati da je i genetski algoritam, koji je preslika prirodnog evolucijskog procesa, moguće lako paralelizirati. Osnovna ideja paralelizacije nekog algoritma je podijeliti posao na podzadatke, a podzadatke podijeliti računalima, odnosno procesorima. Dakle, za ostvarenje paralelnog genetskog algoritma (PGA) treba odrediti što će se odvijati paralelno, na koji način i da li je populacija zajednička ili podijeljena na manje dijelove - subpopulacije. Već površnim pregledom postojećih modela PGA može se uočiti cijeli spektar mogućnosti paralelizacije genetskih algoritama. Paralelno se mogu obavljati ili samo neki genetski operatori ili svi. Na koji način će se ostvariti PGA uvelike ovisi o: arhitekturi računala, načinu na koji su računala (ako ih ima više) ili više procesora (ako se radi o višeprocesorskom računalnom sustavu) međusobno povezani i o operacijskom sustavu. Bez obzira na arhitekturu računala i topologiju mreže, operacijski sustav mora na neki način omogućiti procesorima da paralelno obavljaju posao i da nekako međusobno komuniciraju, odnosno razmjenjuju podatke – u ovom slučaju jedinke. Mnogi autori, vjerojatno potaknutim nizom problema pri izgradnji paralelnog genetskog algoritma pogodnog za izvođenje na određenoj paralelnoj arhitekturi računala, rješavaju problem na drugi način: izgradnjom specifičnih “GA računala” koja su prilagođena genetskom algoritmu. U ovom dijelu opisan je postupak oblikovanja specifičnog modela PGA prilagođenog višeprocesorskim računalima. U kratkom poglavlju 6.6 su prikazane neke male izmjene u spomenutoj arhitekturi kako bi se dodatno ubrzao proces optimiranja.

Genetski algoritam je vremenski zahtjevan i troši najviše procesorskog vremena od bilo koje druge metode optimiranja. Cilj paralelizacije genetskog algoritma je skraćenje vremena izvođenja. Idealno, genetski algoritam bi trebao trajati onoliko puta kraće, koliko računalo ima procesora. Naravno, u stvarnosti se postižu ubrzanja koja su manja od broja procesora. U kolikoj mjeri će se postići propusnost računala[1] za genetski algoritam, ovisi s jedne strane o nizu čimbenika vezanih uz arhitekturu računala (primjerice, iskoristivost sabirnice, kako je ostvaren protokol za održavanje jednoznačnosti preslika podatkovnih objekata i sl.), a s druge strane uvelike ovisi o samom algoritmu, odnosno o tome koliki se dio posla obavlja paralelno te da li se koriste neki od mehanizama međusobnog isključivanja.

[1] Propusnost računala definira se kao recipročna vrijednost vremena potrebnog za obavljanje zadanog algoritma i mjeri se brojem operacija u jedinici vremena [RIB86].

2. GENETSKI ALGORITMI I NAČINI POBOLJŠANJA NJIHOVE DJELOTVORNOSTI

2.1. Struktura genetskih algoritama

Genetski algoritam je heuristička metoda slučajnog i usmjerenog pretraživanja prostora rješenja koja imitira prirodni evolucijski proces [BEA91, BUI94, COR92, MIC94, SRI94, WHI96]. Genetski algoritam služi za rješavanje težih optimizacijskih problema, za koje ne postoji egzaktna matematička metoda rješavanja ili su *NP*-teški pa se za veći broj nepoznanica ne mogu riješiti u zadanom vremenu.

Posao koji obavlja genetski algoritam može se opisati jednom rečenicom: nakon što se generira početna populacija, genetski algoritam ciklički obavlja selekciju boljih jedinki koje potom sudjeluju u reprodukciji, sve dok nije zadovoljen uvjet završetka evolucijskog procesa (slika 2.1). Reprodukcija stvara nove jedinke uz pomoć genetskih operatora križanja i mutacije. Križanje prenosi svojstva roditelja na djecu, a mutacija slučajno mijenja svojstva jedinke. Genetski algoritam ne specificira kako se križanjem prenose svojstva roditelja na djecu, kako se slučajno mijenjaju svojstva jedinki, kako se selektiraju bolje jedinke za reprodukciju, niti kako se generira početna populacija. Upravo je ta sloboda u odabiru vrste križanja, mutacije, selekcije i inicijalizacije otežavajuća okolnost u procesu izgradnje genetskog algoritma za rješavanje specifičnog optimizacijskog problema. Naime, pokazalo se da ne postoji takav skup genetskih operatora za koji bi GA, ako se primijeni za rješavanje proizvoljnog skupa optimizacijskih problema, davao superiorne rezultate u odnosu na GA s nekim drugim operatorima [FOG99, MAC96, WOL96a, WOL96b].

```
Genetski_algoritam(){
   generiraj_početnu_populaciju();
   dok (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      selektiraj_bolje_jedinke_za_reprodukciju();
      reprodukcijom_generiraj_novu_populaciju();
   }
}
```

Slika 2.1. **Genetski algoritam**

Genetski algoritam obavlja genetske operatore nad populacijom jedinki $J_i \in \mathbf{J}$, gdje je **J** skup svih mogućih rješenja. Primjerice, za binarni prikaz, gdje se jedinke sastoje od *b* binarnih znamenaka, kardinalni broj skupa **J** je broj svih mogućih rješenja i iznosi $\#\mathbf{J}=2^b$. *Jedinke* se nazivaju još i *potencijalna rješenja*, jer genetski algoritam manipulirajući genetskim materijalom jedinki, postiže rješenje [BLI95]. U računalnom žargonu, jedinka je nekakva struktura podataka koja se sastoji od kromosoma i vrijednosti funkcije cilja. Jedinka ili kromosom se sastoji od gena koji opisuju svojstva jedinke. Primjerice, za spomenuti binarni prikaz i za problem optimiranja funkcije cilja realne varijable $f(\vec{x})$, geni su binarne znamenke s pomoću kojih se određuju vrijednosti x_i vektora $\vec{x}$.

Populacija $P=\{J_1,J_2,...,J_i,...,J_N\} \in \mathbf{J}^N$ se sastoji od *N* jedinki. Početna populacija se najčešće generira potpuno slučajno, ali može biti i uniformna (sve jedinke su jednake) ili se može sastojati od *usađenih* rješenja[2] dobivenih nekim drugim optimizacijskim postupkom [COR92]. Početna populacija $P(0)$ se s vremenom (iz generaciju u generaciju) mijenja i u trenutku (generaciji) *t* ima oznaku $P(t)$. Obično je uvjet završetka evolucijskog procesa unaprijed zadan broj iteracija *I*.

Selekcijom se odabiru bolje jedinke za reprodukciju. Kvaliteta jedinki mjeri se s pomoću funkcije cilja[3] $f:\mathbf{J}\rightarrow\mathbf{R}$. Neki postupci selekcije zahtijevaju da funkcija cilja ne može biti negativna, a niti je poželjno da funkcija cilja poprima samo velike, približno jednake vrijednosti. Stoga se obično u svakom koraku obavlja translacija funkcije cilja, tj. u svakoj iteraciji oduzima se najmanja vrijednost funkcije cilja u cijeloj populaciji. Rezultat je funkcija *dobrote* $d:\mathbf{J}\rightarrow\mathbf{R}^+$ koja se računa prema izrazu:

$$d=f-f_{min}(t), \qquad (2.1)$$

gdje je $f_{min}(t)$ najmanja vrijednost funkcije cilja u generaciji *t* [BAE94, GOL96, MIC94]. Jedinka J_i je bolja od jedinke J_k, ako je $d_i>d_k$. Postupci skaliranja i translacije funkcije cilja navedeni su u [BAE94].

Genetski operatori: selekcija, križanje i mutacija u svakoj iteraciji modificiraju populaciju *P*. Stoga se može reći da GA pretražuje prostor rješenja mijenjajući populaciju u svakoj iteraciji uz pomoć nekakve složene funkcije $g:\mathbf{J}^N\rightarrow\mathbf{J}^N$. Mutacija i križanje pretražuju prostor rješenja, a selekcija koristi samo informaciju unutar populacije, odnosno ne traži nova rješenja, već favorizira bolje jedinke [BAE94]. Prostor rješenja $\mathbf{J}^N$ naziva se još i prostorom pretraživanja.

Danas još nije poznat postupak uz pomoć kojeg bi se za zadani problem odredili genetski operatori i prikaz rješenja. Postupak izgradnje genetskog algoritma temelji se na vlastitom i tuđem iskustvu (obično iz literature) te na temelju

[2] engleski: *seeded solutions*
[3] engleski: *fitness function*

eksperimentiranja i podešavanja raznih kombinacija genetskih operatora (većinom se to obavlja metodom pokušaja i pogrešaka).

Postupak izgradnje genetskog algoritma može se u grubo razložiti u sljedećih nekoliko koraka:

- stvarni problem postaviti kao optimizacijski problem (primjerice, problem rasporeda se svodi na minimizaciju broja pogrešaka u rasporedu);
- odrediti prikaz i funkciju dobrote;
- odrediti pojedine genetske operatore;
- eksperimentalno podesiti parametre na jednostavnijem problemu za koji je poznato rješenje;
- eksperimentalno obaviti fino podešavanje parametara na stvarnom problemu;
- ukoliko postupak optimiranja stvarnog problema traje predugo, odabrati najpogodniji paralelni model GA i eksperimentalno podesiti dodatne parametre.

Detaljniji pregled svih genetskih operatora nalazi se u prvom dijelu ovog rada.

2.2. Načini poboljšanja djelotvornosti genetskih algoritama

Djelotvornost genetskog algoritma se može poboljšati na tri načina: povećanjem vjerojatnosti postizanja dobrih rješenja, povećanjem kvalitete dobivenih rješenja i skraćenjem trajanja izvođenja. Trajanje izvođenja se može skratiti također na tri načina: povećanjem brzine konvergencije čime se smanjuje broj iteracija, smanjenjem trajanja izvođenja jedne iteracije i paralelnim izvođenjem cijelog genetskog algoritma ili samo pojedinih genetskih operatora. Navedeni ciljevi mogu se ostvariti podešavanjem parametara, optimiranjem izvornog teksta programa i paralelizacijom genetskog algoritma (tablica 2.1).

Tablica 2.1. **Ciljevi i načini poboljšanja djelotvornosti genetskih algoritama**

POBOLJŠANJE DJELOTVORNOSTI GENETSKIH ALGORITAMA	
CILJEVI	**NAČINI**
• povećanje vjerojatnosti postizanja dobrih rješenja	➢ podešavanje parametara
• povećanje kvalitete dobivenih rješenja	
• skraćenje trajanja izvođenja programa	
o povećanje brzinu konvergencije i smanjenje broj iteracija	
o skraćenje trajanje izvođenja jedne iteracije	➢ optimiranje izvornog teksta programa
o paralelizacija	➢ paralelno izvođenje genetskih operacija

Povećanje vjerojatnosti postizanja dobrih rješenja, povećanje kvalitete rješenja i smanjenje broja iteracija može se postići podešavanjem parametara genetskog algoritma. Podešavanje parametara je dugotrajan posao, jer se obavlja isključivo eksperimentalno. Mnogi autori se bave tom problematikom, pa se u literaturi mogu pronaći neki skupovi parametara, koji su podešeni za određeni genetski algoritam primijenjen za rješavanje određenog optimizacijskog problema. Ta tuđa iskustva mogu poslužiti kao dobre smjernice ili početne vrijednosti prilikom eksperimentalnog podešavanja parametara za specifičan genetski algoritam i specifičan optimizacijski problem. Detalji o načinima podešavanja parametara (na temelju tuđih iskustava i eksperimentalno) nalaze se u poglavlju 7.3. S obzirom da je proces podešavanja parametara dugotrajan, prilikom izgradnje genetskog algoritma treba voditi računa da broj parametara bude što je moguće manji. Jednako tako poželjno je da se paraleliziranjem stvara što je moguće manji broj novih parametara. Genetski algoritam opisan u ovom dijelu, osim što je pogodan za paralelno izvođenje, ima svega tri parametra: veličinu populacije, vjerojatnost mutacije i broj iteracija. Njegovim paraleliziranjem dobiva se još samo jedan novi parametar - broj dretvi. Taj parametar se ni ne treba podešavati, jer je, u idealnom slučaju, broj dretvi jednak broju procesora.

Trajanje izvođenja jedne iteracije može se skratiti optimiranjem izvornog koda. Time su iscrpljene sve mogućnosti skraćenja trajanja izvođenja sekvencijskog GA, odnosno genetskog algoritma koji se izvodi na jednoprocesorskom računalu. Paraleliziranjem algoritma dodatno se skraćuje trajanje izvođenja. Paralelizirati se mogu svi ili samo neki genetski operatori. Ako se već ne obavljaju svi genetski operatori paralelno, poželjno je da se paralelno obavljaju oni operatori, koji troše najviše procesorskog vremena.

Dijeljenjem posla na N_P podzadataka, gdje je N_P broj procesora, očekuje se u idealnom slučaju skraćenje vremena računanja N_P puta. Ubrzanje a je broj koji se dobije dijeljenjem potrebnog vremena za rješavanje zadanog problema određenim algoritmom na jednoprocesorskom računalu s potrebnim vremenom za rješavanje istog problema s istim algoritmom na N_P procesora. Idealno, ujedno i maksimalno ubrzanje jednako je broju procesora: $a_{max}=N_P$.

3. MODELI PARALELNIH GENETSKIH ALGORITAMA

3.1. Podjela paralelnih genetskih algoritama

Evolucija u prirodi je paralelni proces. Genetski algoritam je svojevrsna apstrakcija prirodne evolucije i također ga je lako paralelizirati. U zadnjih desetak godina intenzivno se razmatraju mogućnosti paraleliziranja genetskih algoritama. Ideja o izgradnji specifičnih paralelnih računala je još starija: još pedesetih godina John Holland predlaže specifičnu arhitekturu paralelnih računala koja bi bila pogodna za simulaciju evolucije prirodnih vrsta [CAN98a].

Paralelni genetski algoritmi koriste se za rješavanje težih optimizacijskih problema. Teži problemi zahtijevaju velike populacije i velike duljine kromosoma, zbog čega postupak optimiranja duže traje. Osnovna motivacija paralelizacije genetskih algoritama je ubrzati njihovo izvođenje na višeprocesorskim računalima ili na više umreženih računala. U početku istraživanja paralelnih genetskih algoritama nastojala su se paralelizacijom poboljšati i ostala svojstva genetskog algoritma kao što je, primjerice, brzina konvergencije. Naime, osim ubrzanja, neki modeli paralelnih genetskih algoritama pokazali su bolja svojstva u odnosu na isti takav sekvencijski genetski algoritam: pronalaze bolja rješenja s manjim brojem iteracija nego odgovarajući sekvencijski GA [CAN95]. Međutim, kao što će se to kasnije pokazati, radi se o specijalnom slučaju kada sekvencijski GA nema dobro podešene parametre. U stvarnosti se teži da PGA ima ista svojstva kao i odgovarajući sekvencijski GA s podešenim parametrima.

Cilj paralelizacije je skratiti trajanje izvođenja složenih primjenskih programa, a da se pritom ne naruše njihova svojstva. Osnovna ideja paraleliziranja programa jest raščlanjivanje sekvencijskog (serijskog) programa na nezavisne podzadatke koji se mogu izvoditi paralelno. Kod genetskog algoritma treba odrediti što će se obavljati paralelno. Genetski algoritam ciklički ponavlja jedan te isti posao. Iz iteracije u iteraciju izračunavaju se vrijednosti funkcije cilja i genetski operatori djeluju nad populacijom jedinki. Nameću se dva pristupa paraleliziranja genetskih algoritama:

- *standardni pristup* – paralelizirati genetske operatore i izračunavati vrijednosti funkcije cilja paralelno ili
- *dekompozicijski pristup* – podijeliti populaciju na manje dijelove - subpopulacije i obavljati cijeli genetski algoritam nad subpopulacijama.

U prvom slučaju se paralelizira samo posao evaluacije. *Evaluacija* ili *vrednovanje* je postupak izračunavanja vrijednosti funkcije cilja. Jednako kao i kod sekvencijalnog genetskog algoritma, genetski operatori djeluju samo nad jednom, zajedničkom populacijom pa se takav model naziva jednopopulacijski model. U drugom slučaju se populacija dijeli na nekoliko subpopulacija pa se takav model naziva višepopulacijski model, odnosno genetski algoritam se naziva multipopulacijskim paralelnim genetskim algoritmom [CAN98a, CAN99a, TAL91]. Očekuje se skraćenje trajanja izvođenja multipopulacijskog PGA jer subpopulacije broje manje jedinki od inicijalne populacije pri jednopopulacijskom modelu.

Napretkom tehnologije izrade sklopovlja pojavila su se i sklopovska rješenja tzv. masovnih paralelnih računala koja su se primijenila i za rješavanje optimizacijskih problema koristeći paralelne genetske algoritme. Stoga se u novije vrijeme u literaturi [MIC94, CAN98a, LIN97a] predlaže novi model paralelnih genetskih algoritama koji koristi mogućnosti masovno paralelnih računala kao što je, primjerice, računalo MasPar MP1 [LOG92].

Postoje nekoliko mogućih razina paraleliziranja genetskih algoritama: paraleliziranje na razini populacije, na razini jedinki te na razini evaluacije [TOM99]. Prema razini paralelizacije, postoje tri osnovna načina podjele sekvencijskog genetskog algoritma na podzadatke:

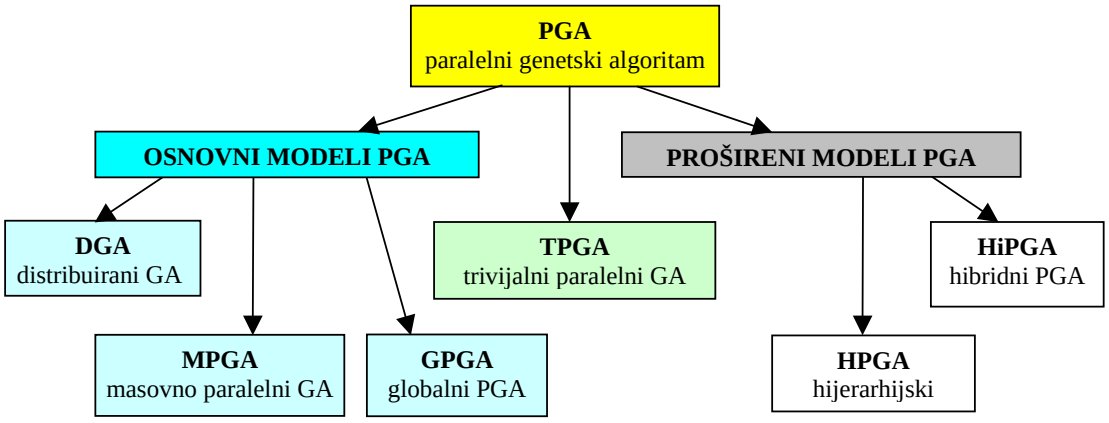
- *Krupnozrnata podjela* je podjela velike populacije na manje dijelove – subpopulacije. U ovom se slučaju radi o dekompozicijskom pristupu ili o već spomenutom višepopulacijskom paralelnom genetskom algoritmu koji je raspodijeljen tako da se paralelno izvodi nekoliko genetskih algoritama nad manjim populacijama.

- *Sitnozrnata podjela* je ekstremni oblik podjele velike populacije na subpopulacije veličine jedne jedinke. Svaki procesor obavlja genetske operatore nad njemu dodijeljenom jedinkom i nad susjednim jedinkama. Ovakva podjela je također predstavnik višepopulacijskog modela.

- Moguće je paralelno obavljati evaluaciju, tj. izračunavanje vrijednosti funkcije cilja, dok se genetski operatori izvode sekvencijski. Takva podjela se naziva podjelom na "*gospodara*" i "*sluge*". *Gospodar* obavlja genetski algoritam nad zajedničkom populacijom, stoga je to jednopopulacijski model. U svakoj iteraciji *sluge* paralelno izračunavaju vrijednosti funkcije cilja nakon što *gospodar* obavi svoj sekvencijski dio posla.

U ovisnosti o načinu podjele genetskog algoritma na podzadatke, tri su osnovna modela paralelnih genetskih algoritama:

- *Distribuirani genetski algoritam*[4] (kratica: DGA) ili raspodijeljeni genetski algoritam sastoji se prema krupnozrnatoj[5] podjeli od nekoliko subpopulacija[6] pa se naziva još i višepopulacijski genetski algoritam[7]. To je najpopularniji model paralelnih genetskih algoritama.
- *Masovno paralelni genetski algoritam*[8] (kratica: MPGA) se prema sitnozrnatoj[9] podjeli sastoji od N_p procesora koji predstavljaju N_p jedinki. Dakle, veličina populacije je jednaka broju procesora, odnosno $N=N_p$. Svaki procesor obavlja genetske operatore nad svojom jedinkom i nad susjednim jedinkama.
- *Globalni paralelni genetski algoritam*[10] (kratica: GPGA) je predstavnik podjele na *gospodara* i *sluge*. Paralelni dio posla obavljaju *sluge*, dok sekvencijski *gospodar*. *Sluge* su zadužene samo za evaluaciju jedinki, a *gospodar* obavlja sve ostale genetske operatore.



Slika 3.1. **Podjela paralelnih genetskih algoritama**

Važno je napomenuti da samo globalni paralelni genetski algoritam ima ista svojstva kao i sekvencijski genetski algoritam pa su sva teorijska razmatranja vezana uz sekvencijski genetski algoritam primjenjiva i na globalni paralelni genetski algoritam. Ostali modeli značajno mijenjaju način izvođenja genetskog algoritma pa je teorijska analiza rada takvih algoritama još u začecima [CAN95].

Spomenuta tri osnovna modela se mogu međusobno kombinirati ili nadograditi s nekom drugom metodom optimiranja. Prema tome, postoje još dva proširena modela paralelnih genetskih algoritama:

- *Hijerarhijski paralelni genetski algoritam*[11] (kratica: HPGA) je kombinacija prethodna tri modela. Primjerice, to može biti distribuirani genetski algoritam na nekoliko međusobno povezanih računala, a na svakom od računala obavlja se globalni paralelni genetski algoritam nad subpopulacijama.
- *Hibridni paralelni genetski algoritam*[12] (kratica: HyPGA) je kombinacija jednog od prethodna četiri modela s nekim drugim algoritmom za lokalno pretraživanje. Najčešće se radi o nekoj od gradijentnih metoda koje se primjenjuju nakon određenog broja iteracija i to samo nad nekim jedinkama.

Posljednji, ali i najjednostavniji model paralelnih genetskih algoritama je:

- *Trivijalni paralelni genetski algoritam*[13] (kratica: TPGA). Radi se o više genetskih algoritama koji se paralelno obavljaju na nekoliko potpuno nezavisnih računala (računala ne moraju biti povezana) kako bi se, primjerice, statistički obradili eksperimentalno dobiveni rezultati ili kako bi se odredio optimalan skup parametara [TOM99]. Ova ekstremno jednostavna paralelna metoda je, u stvari, izuzetno korisna upravo za statističku analizu algoritma. Primjerice, isti algoritam pokrene se na nekoliko odvojenih računala s različitim početnim uvjetima i promatra se kvaliteta dobivenog rješenja. Potom se rezultati statistički obrađuju. S obzirom da je genetski algoritam stohastički proces, prikupljanje statističkih podataka je od izuzetne važnosti [TOM99].

---

[4] engleski: *distributed genetic algorithm*
[5] naziva se i krupnozrnatim genetskim algoritmom, engleski: *coarse-grained GA*, kratica: ***cgGA***
[6] engleski: *subpopulation*, ali se još češće koristi izraz *deme*
[7] engleski: *multiple-population* ili *multiple-deme parallel genetic algorithm*
[8] engleski: *massively parallel genetic algorithm*
[9] engleski: *fine-grained GA*, kratica: ***fgGA***
[10] engleski: *master-slave genetic algorithm* ili *micro-grained GA*, kratica: ***mgGA***
[11] engleski: *hierarchical parallel genetic algorithm*
[12] engleski: *hybrid parallel genetic algorithm*
[13] izvorni naziv na engleskom jeziku je *embarassingly parallel genetic algorithm*

3.2. Distribuirani genetski algoritam

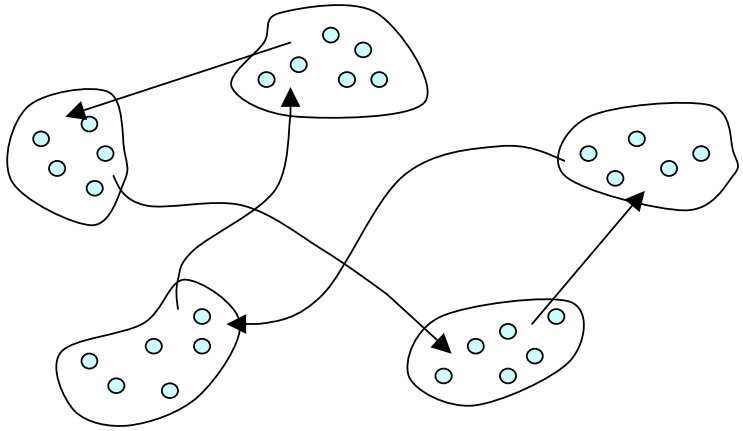
Osnovno obilježje distribuiranog genetskog algoritma je raspodijeljena populacija. Umjesto da djeluje kao sekvencijski genetski algoritam nad jednom velikom populacijom, distribuirani genetski algoritam je raspodijeljen i djeluje nad više manjih populacija ili otoka. Model DGA je nastao uslijed potrebe da se umrežena računala iskoriste za paralelno obavljanje genetskih algoritama. Genetski algoritmi se obavljaju u čvorovima. Čvorovi su najčešće umrežena računala, ali mogu biti i procesori u višeprocesorskom sustavu koji međusobno komuniciraju preko zajedničkog radnog spremnika. Genetski algoritmi u čvorovima mogu se međusobno razlikovati. Zajednički im je optimizacijski problem koji svi paralelno rješavaju. Čvorovi međusobno razmjenjuju jedinke u nadi da će novopristigla jedinka u novoj okolini potaknuti pretraživanje još neistraženog područja prostora rješenja i na taj način postići još bolje rješenje.

Više genetskih algoritama paralelno djeluju nad subpopulacijama i svakih M_i iteracija izmjenjuju dobivena rješenja (slika 3.2). Izmjenjivanje jedinki naziva se *migracijom*. Subpopulacije komuniciraju (izmjenjuju podatke, tj. jedinke ili *migrante*) preko komunikacijskog kanala. Subpopulacije su relativno izolirane kako bi genetski algoritam pretraživao različite dijelove prostora rješenja. Takav model se može implementirati na nekoliko umreženih računala ili na jednom višeprocesorskom računalu. Komunikacijski kanal može biti lokalna mreža, globalna mreža ili Internet, zajednički radni spremnik, cjevovod, red poruka, varijable okoline, itd. Ime *distribuirani* ili *raspodijeljeni* genetski algoritam dobio je po tome što se može implementirati i na računala MIMD arhitekture s distribuiranim radnim spremnikom [CAN95].

```
Distribuirani_genetski_algoritam(){
   inicijaliziraj_P_populacija();
   dok (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      za svaku subpopulaciju obavljaj paralelno{
         evaluiraj();
         ako( (broj_iteracija % period_izmjene) == 0 ){
            migracija();   // izmjeni jedinke
         }
         selektiraj();
         križaj();
         mutiraj();
      }
   }
}
```

Slika 3.2. **Distribuirani genetski algoritam**

Multipopulacijski model paralelnih genetskih algoritama je najjednostavniji za implementaciju na umreženim računalima pa se najčešće koristi u praksi i predmet je mnogih znanstvenih istraživanja [CAN98a, CAN98b]. Stoga se pod pojmom paralelnih genetskih algoritama često podrazumijeva distribuirani genetski algoritam [MUN93, MUT92]. U literaturi se još naziva i krupnozrnatim genetskim algoritmom [LIN97a, GOO97, EBY97, CAN98a] ili otočnim paralelnim genetskim algoritmom[14] [WHI99, CAN98a, TOM99]. Naziv krupnozrnati genetski algoritam je dobio zbog toga što je podijeljen na veće skupine jedinki – subpopulacije (slika 3.3), za razliku od sitnozrnatog (masovno paralelni genetski algoritam) koji je podijeljen na najsitnije moguće subpopulacije veličine jedne jedinke.



Slika 3.3. **Otočni model djelomično izoliranih subpopulacija**

[14] engleski: *island parallel genetic algorithm* ili *island distributed genetic algorithm*

Ima mnoštvo primjera u prirodi gdje jedna te ista vrsta živi u potpuno razdvojenim subpopulacijama, primjerice, na više kontinenata. Te razdvojene vrste evoluiraju paralelno. Analogija s prirodom i dostupnost umreženih računala su razlozi velikoj popularnosti DGA.

Veličina subpopulacije je obično manja od populacije koju koristi sekvencijski genetski algoritam pa se očekuje skraćenje trajanja izvođenja. Međutim, vrijeme trajanja izvođenja programa nije kraće onoliko puta koliko je manja subpopulacija od populacije. Naime, treba uzeti u obzir trajanje razmjene jedinki (vrijeme trajanja potrebno za komunikaciju među subpopulacijama) i vrijeme trajanja potrebno za sinkronizaciju (ako se razmjena jedinki odvija sinkrono). Stoga, trajanje izvođenja zbog spore komunikacije i učestale sinkronizacije može biti još i duže od ekvivalentnog sekvencijskog genetskog algoritma [BUD98].

Ukupna veličina populacije je suma svih subpopulacija. Broj subpopulacija i veličina pojedine subpopulacije su parametri koji značajno utječu na ponašanje algoritma [CAN95]. Broj subpopulacija jednak je broju procesora (N_p) ili broju računala na kojima se distribuirani genetski algoritam izvodi. Početna populacija *P(0)* veličine *N* distribuira se na N_p procesora. Obično se raspodjeljuje jednoliko [WHI99] tako da je veličina otočne populacije (N_{otok}) jednaka na svakom od otoka.

Svaki procesor obavlja sekvencijski genetski algoritam nad svojim otokom i izmjenjuje M_s jedinki svakih M_i iteracija. M_i se naziva migracijskim intervalom[15], a M_s migracijskom stopom[16] [MUT92]. Treba naglasiti da migriranje ovdje ne znači seljenje jedinki iz jedne subpopulacije u drugu, već odabir *boljih* jedinki iz jedne subpopulacije čije će kopije nadomjestiti *lošije* jedinke u drugoj subpopulaciji.

Postupak migracije obavlja novi genetski operator: operator migracije. Način na koji će se obaviti migracija i kada se ona zbiva određuje pet dodatnih parametara:

- M_i - *migracijski interval* ili period izmjene jedinki između procesora,
- M_s - *migracijska stopa* ili broj jedinki koja se izmjenjuje,
- *strategija odbira boljih jedinki*,
- *strategija odabira jedinki za eliminaciju* i
- *topologija razmjene jedinki*.

3.2.1. Migracijski interval

Migracijski interval M_i je broj iteracija između dvije migracije. Operator migracije ne djeluje u svakoj iteraciji, kao što je to slučaj s ostalim genetskim operatorima, već svakih M_i iteracija. Često se u literaturi umjesto migracijskog intervala koristi pojam frekvencija migracije f_i. Frekvencija migracije je učestalost izmjene jedinki i jednaka je recipročnoj vrijednosti migracijskog intervala:

$$f_i = M_i^{-1}. \qquad (3.1)$$

Distribuirani genetski algoritam pretražuje prostor rješenja s pomoću nekoliko relativno izoliranih subpopulacija čime se nastoji izbjeći prerana konvergencija k lokalnom optimumu. Migracijski interval je parametar koji značajno utječe na brzinu konvergencije genetskog algoritma [CAN99b]. Prevelika učestalost migracije dovodi do podjednakih subpopulacija i vjerojatno do suboptimalnog rješenja [LOG92]. Povećava li se broj iteracija između migracija omogućuje se subpopulacijama nezavisniji razvoj, a time se izbjegava prerana konvergencija.

Migracijski interval može biti:

- konstantan,
- promjenjivi:
  - potpuno slučajan ili
  - uvjetovan.

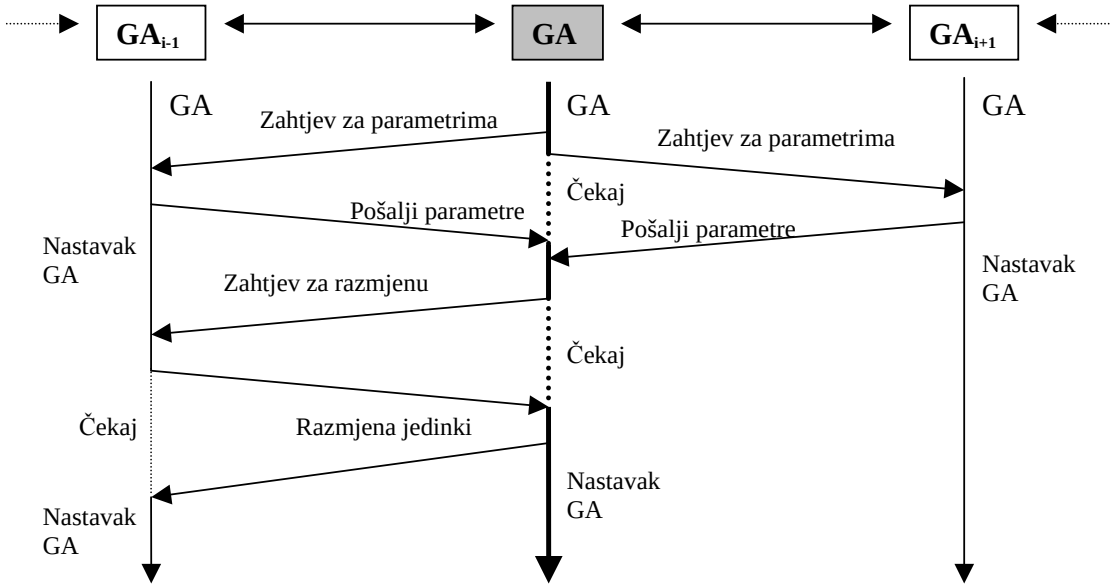
U slučaju da je migracijski interval konstantan, subpopulacije razmjenjuju rješenja svakih unaprijed određenih M_i iteracija. Promjenjivi migracijski interval može se odabrati potpuno slučajno ili ovisi o nekim drugim parametrima kao što je, primjerice, raspršenost populacije. Za potpuno slučajan migracijski interval važan je parametar $\overline{M_i}$ – prosječna vrijednost migracijskog intervala. Uvjetovani migracijski interval se, primjerice, određuje uporabom tzv. sigma-izmjenjivačkog algoritma [MUN93]. U tom slučaju, migracija među subpopulacijama se odvija onda kada standardna devijacija rješenja subpopulacije postane manja od unaprijed određene (zadane) vrijednosti.

Distribuirani genetski algoritam može biti sinkroni ili asinkroni. Sinkroni distribuirani genetski algoritam sinkrono razmjenjuje jedinke, tj. sve subpopulacije odjednom obavljaju razmjenu. U načelu, DGA je sinkroni jer se procesi sinkroniziraju za vrijeme faze razmjene podataka (komunikacije) [TOM99]. S druge strane, spomenuti DGA s uvjetovanim migracijskim intervalom je primjer asinkronog algoritma, jer svaka populacija zasebno odlučuje kada će obaviti razmjenu jedinki. Na slici 3.4 prikazan je komunikacijski protokol između subpopulacija za asinkroni distribuirani genetski algoritam [MUN93].

Najveći mogući migracijski interval (ili najmanja moguća frekvencija migracije) jednak je ukupnom broju iteracija. To znači da tijekom optimiranja nema migracije i samo se na kraju evolucijskog procesa izmjenjuju dobivena rješenja. Pokazalo se da je rješenje dobiveno u tom slučaju lošije od rješenja koje se dobiva sa sekvencijskim

[15] engleski: *migration interval*
[16] engleski: *migration rate* ili *migration size*

genetskim algoritmom s jednom populacijom čija je veličina jednaka zbiru svih subpopulacija [CAN98a]. Naime, pri malim frekvencijama migracije, subpopulacije ostaju nezavisne i genetski algoritam istražuje različita područja prostora rješenja. Međutim, u tom slučaju se pretraživanje prostora rješenja odvija s manjom efikasnošću, jer genetski algoritam djeluje nad manje jedinki. Za premale vrijednosti frekvencije migracije svojstva DGA se degradiraju na razinu svojstava sekvencijskog genetskog algoritma s veličinom populacije N_{otok}. Drugim riječima, takav DGA ima znatno lošija svojstva od odgovarajućeg sekvencijskog GA s veličinom populacije N.



Slika 3.4. **Asinkroni distribuirani genetski algoritam**

Druga je krajnost najmanji migracijski interval (najveća frekvencija migracije) jedinične duljine. Što znači da jedinke migriraju u svakoj iteraciji. Ta mogućnost ispitana je u [CAN99b]. I u tom slučaju, dobivena rješenja su lošija od rješenja koja se dobivaju sa sekvencijskim genetskim algoritmom, osim ako se u svakoj iteraciji ne razmjenjuju najbolje jedinke. Dva su nedostatka DGA s prevelikom frekvencijom migracije: suboptimalna rješenja se u početku evolucijskog procesa prebrzo prošire po svim subpopulacijama pa svi čvorovi pretražuju ista područja prostora rješenja i zbog učestale migracije komunikacijski kanal može postati usko grlo postupka optimiranja.

Vrijednosti frekvencije migracije za koje subpopulacije postaju izolirane ili kada se suboptimalno rješenje počinje prebrzo širiti po subpopulacijama nazivaju se kritičnim vrijednostima frekvencije migracije [CAN98b]. Negdje u intervalu između gornje i donje kritične vrijednosti frekvencije migracije nalazi se optimalna vrijednost tog parametra za koji DGA postiže približno jednake rezultate kao i sekvencijski genetski algoritam.

Osjetljivost distribuiranog genetskog algoritma na migracijski interval može se opravdati *teorijom ravnoteže* [CAN98a]. Ta teorija tvrdi da populacija većinu vremena provodi u ravnoteži (kada nema značajnijih genetskih promjena) sve dok neka veća promjena u okolini ne izazove evolucijske promjene u populaciji. Veća promjena u okolini je i dolazak neke nove jedinke iz druge subpopulacije. Dolaskom nove jedinke s novim svojstvima narušava se postignuta ravnoteža i potiču se evolucijske promjene. Pokazalo se da se nova rješenja pronalaze nedugo nakon izmjene jedinki [CAN98a].

3.2.2. Migracijska stopa

Migracijska stopa (M_s) je broj jedinki koje se razmjenjuju svakih M_i iteracija. Slično kao i migracijski interval, ovaj parametar utječe na raznolikost populacije. Naime, ako je mala frekvencija migracije, subpopulacije će biti međusobno različitije i obrnuto. Jednako tako će se subpopulacije međusobno više razlikovati, ako se razmjenjuje mali broj jedinki. Nadalje, ako migracijska stopa teži k veličini populacije, tada će subpopulacije biti međusobno sličnije. To se i poklapa s principom populacijske genetike, koji kaže da se bolje osobine jedinki brže šire u malim populacijama nego u velikim [CAN98a]. Najčešće je migracijska stopa jednaka jedinici (razmjenjuje se samo jedna jedinka) ili vrijednosti blizu jedan, tj. razmjenjuje se relativno mali postotak subpopulacije.

3.2.3. Strategije odabira jedinki

Strategija odabira jedinki[17] značajno utječe na selekcijski pritisak, a time i na brzinu konvergencije algoritma. Selekcijski pritisak je veći što bolje jedinke imaju veću vjerojatnost preživljavanja u odnosu na loše jedinke i obrnuto (vidi poglavlje 4.2). Konvergencija k rješenju (bilo ono i lokalno) je spora, ako je selekcijski pritisak previše slab. S druge strane, preveliki selekcijski pritisak uzrokuje prebrzu konvergenciju, najčešće k lokalnom optimumu

[17] engleski: *migration policy*

[CAN99b]. Pravilnim odabirom broja jedinki za migraciju i strategije odabira jedinki može se postići tzv. superlinearno ubrzanje.

Superlinearno ubrzanje[18] je ubrzanje koje je veće od broja procesora: $a>N_P$ [TOM99]. Skraćenje trajanja izvođenja PGA uzrokuje dodatni selekcijski pritisak. Premali selekcijski pritisak uzrokuje sporu konvergenciju, pa sekvencijski algoritam sporo konvergira. Raspodijeli li se populacija na N_P jednakih dijelova i primijeni li se distribuirani genetski algoritam, migracija dodatno povećava selekcijski pritisak koji uzrokuje bržu konvergenciju, a time u kraćem vremenu dolazi do rješenja [CAN99b, CAN99c]. Tako je, zapravo, superlinearno ubrzanje posljedica skraćenja broja iteracija. U stvarnosti, za jednak broj iteracija, ubrzanje je uvijek manje od broja procesora ($a<N_P$), jer se stanovito vrijeme troši za međusobnu komunikaciju i sinkronizaciju.

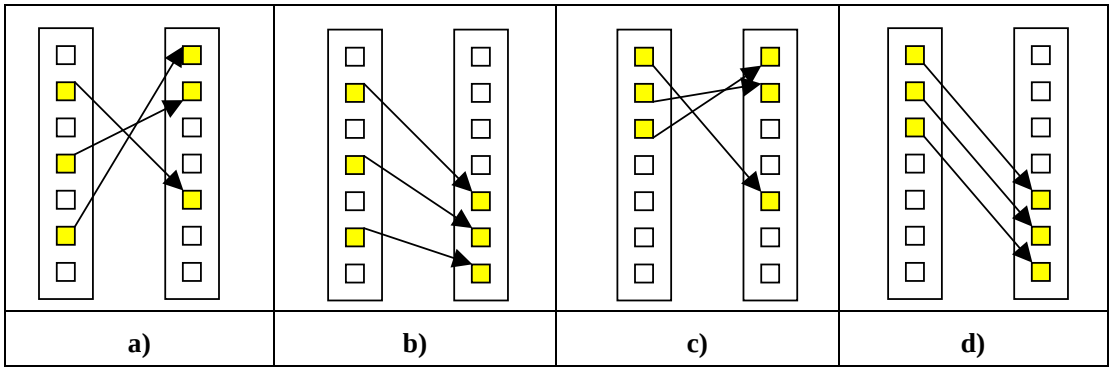
Prilikom razmjene treba nekako odabrati jedinke za slanje susjedima kao i jedinke za eliminaciju. Za oba slučaja moguće su dvije strategije:

- *strategija slučajnog odabira* i
- *strategija odabira najboljih*, odnosno *najlošijih jedinki*.

Strategije odabira jedinki se razlikuju prema načinu odabira jedinki za slanje i jedinki za eliminaciju. Budući se jedinke mogu birati nasumice ili po kriteriju dobrote, postoje sljedeće četiri kombinacije:

- *strategija slučajnog odabira* - slučajno se odabiru jedinke za slanje i za eliminaciju (slika 3.5a);
- *miješane strategije*:
  - slučajno se odabiru jedinke za slanje, a nadomještaju se najgore jedinke (slika 3.5b);
  - najbolje jedinke se odabiru za migraciju, a nadomještaju se slučajno odabrane jedinke (slika 3.5c) i
- *strategija odabira najboljih i najgorih jedinki* - najbolje jedinke migriraju i nadomještaju najgore jedinke (slika 3.5d).

Na slici 3.5 su jedinke razvrstane od najbolje na vrhu do najgore na dnu.



Slika 3.5. **Strategije odabira jedinki**

a) Slučajan odabir jedinki za slanje i eliminaciju
b) Slučajno odabrane jedinke nadomještaju najlošije
c) Najbolje jedinke nadomještaju slučajno odabrane
d) Najbolje jedinke nadomještaju najgore

U primjeni se najčešće koriste treći ili četvrti način: najbolje jedinke se odabiru za migraciju, a nadomještaju se slučajno odabrane ili najlošije jedinke. Strategija odabira najboljih jedinki za migraciju i najlošijih jedinki za eliminaciju u kombinaciji s prevelikom frekvencijom migracije dovodi do prerane konvergencije.

Neki autori predlažu strategije slučajnog odabira, ali između jedinki natprosječne dobrote za migraciju, a između jedinki ispodprosječne dobrote jedinke za eliminaciju [LOG92].

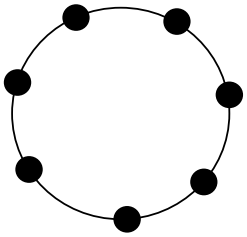
3.2.4. Topologija razmjene jedinki

Pokazalo se da na svojstva distribuiranog genetskog algoritma utječe i topologija razmjene jedinki [GOO97, CAN99d] iako se taj parametar distribuiranog genetskog algoritma dugo vremena zapostavljao [CAN98a]. Topologija razmjene jedinki je plan po kojem čvorovi razmijenjuju jedinke.

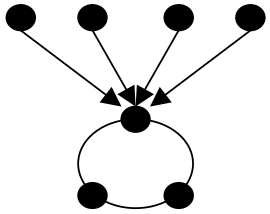
Slika 3.6 prikazuje najjednostavniju prstenastu topologiju koja se najviše koristi u primjeni. Primjer dvoslojne otočne[19] topologije je prikazan na slici 3.7, a višeslojne topologije na slici 3.8. Pokazalo se da je topologija s više slojeva robusnija, jer je opasnost od zaglavljivanja u lokalnim optimumima manja. Dobivena rješenja nižeg sloja se prikupljaju u jednoj ili više subpopulacija višeg sloja. Ukoliko neki od genetskih algoritama nižeg sloja zaglavi u

[18] engleski: *superlinear speedup*
[19] GA s otočnom topologijom naziva se *injection island GA* (kratica *iiGA*)

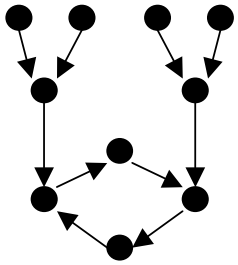
lokalnom optimumu, njegovo rješenje (lokalni optimum) će eliminirati genetski algoritam višeg sloja uporabom selekcije. Primjer primjene višeslojne topologije dan je u [EBY97, GOO97].



Slika 3.6. **Prstenasta topologija**



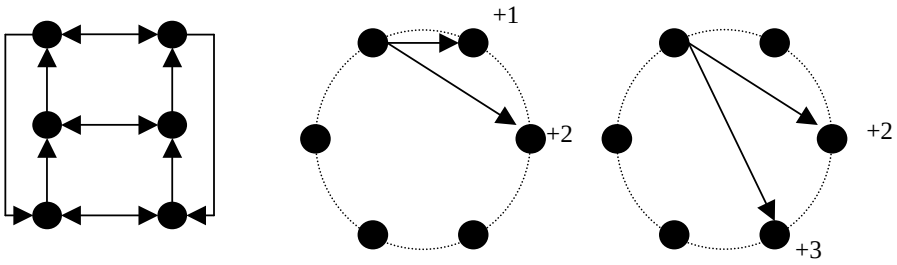
Slika 3.7. **Dvoslojna otočna topologija**



Slika 3.8. **Tipična višeslojna otočna topologija s jednim susjedom**

Pored navedenih topologija, može se koristiti i slučajna topologija: genetski algoritam svaki puta prije migracije slučajno odabire kojoj će subpopulaciji poslati svoje jedinke [TOM99].

Broj susjeda ili stupanj povezanosti[20] je također važno obilježje distribuiranih genetskih algoritama [CAN98a, CAN98b, CAN99d]. Slika 3.9 prikazuje nekoliko primjera topologija s dva susjeda. Svaki čvor šalje svoje najbolje jedinke dvama čvorovima. Jednako tako, svaki čvor dobiva najbolje jedinke od dva čvora. Ti čvorovi se nazivaju susjednim čvorovima. Susjedni čvorovi od kojih se dobivaju i kojima se šalju jedinke mogu biti različiti (slika 3.9b i slika 3.9c), tj. od jednih čvorova se primaju jedinke, a drugim čvorovima se šalju jedinke.



Slika 3.9. **Primjeri topologija s dva susjeda**

a) Ljestvičasta topologija
b) Kružna +1+2 topologija
c) Kružna +2+3 topologija

Brzina širenja dobrih rješenja među subpopulacijama ovisi o broju susjeda kojima se šalju jedinke. Ako je topologija "gusta", tj. jedinke jedne subpopulacije migriraju k više subpopulacija, tada će se brzo proširiti dobro rješenje po svim subpopulacijama [CAN98a]. Međutim, svaka komunikacija, odnosno razmjena jedinki traje određeno vrijeme i s povećanjem gustoće topologije, povećava se i cijena komunikacije.

Topologija može biti statička ili dinamička, ovisno o tome da li se s vremenom mijenja ili ne. Obično se koristi statička topologija[21], koja se zadaje unaprijed i ostaje nepromijenjena do kraja izvođenja algoritma [CAN95]. Čvorovi u dinamičkom modelu topologije nasumice odabiru čvorove kojima će slati jedinke (radi se o već spomenutoj slučajnoj topologiji) ili je shema po kojoj će se obavljati razmjena jedinki tijekom evolucije unaprijed zadana.

[20] engleski: *degree of connectivity*
[21] engleski: *stepping stone model*

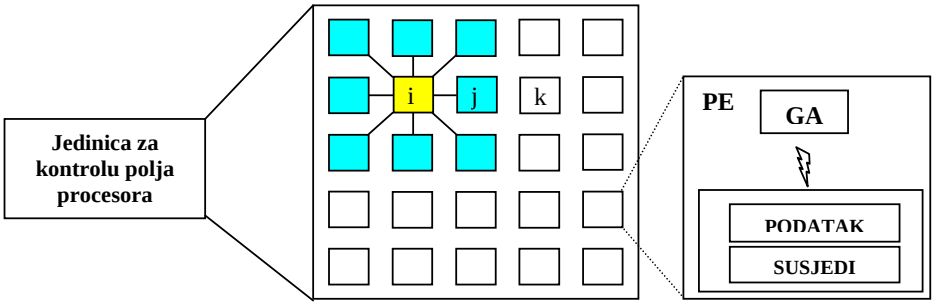
Tablica 3.1. **Prednosti i nedostaci distribuiranog genetskog algoritma**

Prednosti	Nedostaci
• jednostavan za implementaciju na višeprocesorskom sustavu s raspodijeljenim kao i sa zajedničkim radnim spremnikom • raspodijeljena populacija sprječava preranu konvergenciju k lokalnom optimumu, iako DGA brže konvergira od sekvencijskog genetskog algoritma • mogućnost superlinearnog ubrzanja	• mnogo novih parametara (čak sedam) • primjenjuje se još jedan, dodatni genetski operator: *migracija* • teško je odrediti optimalan skup dodatnih parametara, a naročito migracijski interval • često je cijena komunikacije prevelika (previše procesorskog vremena se troši na komunikaciju, a premalo na optimiranje)

Jednostavnost implementacije na umreženim računalima i manja vjerojatnost zaglavljivanja u lokalnom optimumu su najznačajnije prednosti, a ujedno i razlozi zbog čega se distribuirani genetski algoritmi najviše istražuju. Kao što je već spomenuto, superlinearno ubrzanje nije moguće postići ako sekvencijski GA ima pravilno podešene parametre. Međutim, za neke vrste selekcija ne postoji mogućnost podešavanja brzine konvergencije, kao što je to primjerice slučaj s proporcionalnom selekcijom (poglavlje 4.4). U slučaju da je pri toj selekciji konvergencija prespora, jedino se migracijom ona može ubrzati. Na taj način se postiže superlinearno ubrzanje. Među nedostacima ističe se prevelik broj novih (dodatnih) parametara zbog čega je postupak podešavanja algoritma za rješavanje specifičnog optimizacijskog problema otežan.

3.3. Masovno paralelni genetski algoritam

Masovno paralelni genetski algoritam[22] je vrlo sličan distribuiranom genetskom algoritmu. Od DGA se razlikuje po tome što se migracija obavlja samo između susjednih procesora. Za razliku od ostalih modela paralelnih genetskih algoritama, model masovno paralelnih genetskih algoritama (slika 3.10) zahtijeva višeprocesorsko računalo koje se sastoji od mnogo (nekoliko stotina ili čak tisuća) procesora. Primjer takvog računala je MasPar MP1 računalo s 2048 procesora [LOG92]. Takav model paralelnih genetskih algoritama naziva se još *mrežnim modelom*[23] polja procesorskih elemenata [TAL92, TOM99].



Slika 3.10. **Model masovno paralelnog genetskog algoritma s 25 procesorskih elemenata**

Svaki procesor (procesorski element, kratica PE) ima pohranjenu jednu jedinku u svojem internom spremniku, koja se naziva jedinkom procesora. Stoga je veličina populacije ograničena brojem procesora. Naravno, ukoliko veličina populacije mora biti veća od broja procesora (jer u suprotnom genetski algoritam ne daje zadovoljavajuće rezultate), postoji mogućnost da neki ili svi procesorski elementi imaju pohranjeno više od jedne jedinke u svojem internom spremniku.

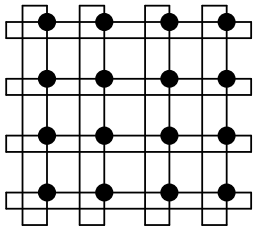
```
Masovno_paralelni_genetski_algoritam(){
   generiraj_paralelno_populaciju_slučajnih_jedinki();
   dok (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      evaluiraj();    // evaluiraj paralelno svaku jedinku
      selektiraj();   // selektiraj paralelno jedinku za reprodukciju
                         među susjedima
      reproduciraj(); // obavi paralelno reprodukciju među odabranom
                         jedinkom iz prethodnog koraka i vlastitom
                         jedinkom
      nadomjesti();   // nadomjesti paralelno vlastitu jedinku s
                         novodobivenom jedinkom
   }
}
```

Slika 3.11. **Masovno paralelni genetski algoritam**

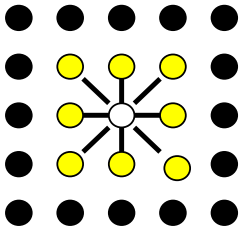
[22] engleski: *massively parallel genetic algorithm*
[23] engleski *grid* ili *fine-grained model of parallel genetic algorithm*

Svi procesori paralelno obavljaju sve genetske operatore i evaluaciju (slika 3.11). Unarne operatore: evaluaciju i mutaciju procesorski elementi obavljaju nad pripadajućom jedinkom, tj. onom jedinkom koja se nalazi u internoj memoriji tog procesora. Binarni operator križanje procesorski elementi obavljaju nad svojim podatkom i nekim od susjeda. Selekcijom se odabire susjedna jedinka s kojim će se obaviti križanje. Dijete dobiveno križanjem sprema se u internu memoriju, tj. nadomješćuje jednog od roditelja. Radi se o generacijskoj selekciji jer se u svakoj iteraciji izmjenjuje cijela populacija, odnosno cijela jedna generacija.

Svaki procesorski element je povezan sa svojim susjedima. Slika 3.12 prikazuje torus od 16 procesorskih elemenata. Svaki procesor ima četiri susjeda. Na susjednoj slici 3.13 svaki procesorski element je povezan s osam susjeda. Broj susjeda je važan upravo zbog toga jer se selekcija i križanje ne obavlja nad cijelom populacijom, već samo među susjedima.



Slika 3.12. **Primjer polja procesorskih elemenata**

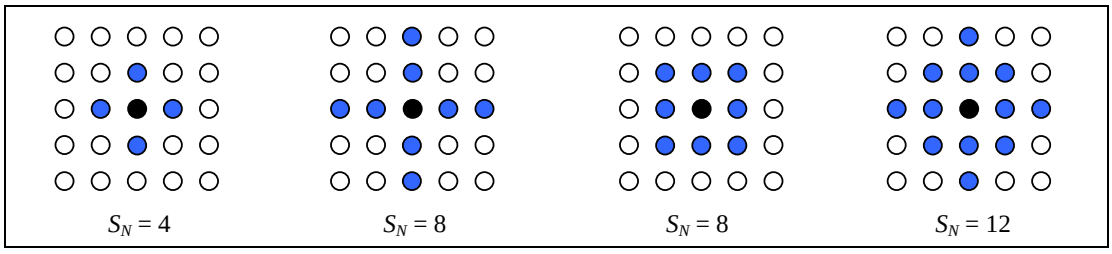


Slika 3.13. **Svaki procesorski element je povezan sa svojim susjedima**

Slično kao i kod DGA, populacija je na neki način podijeljena na subpopulacije. Veličina subpopulacije jednaka je broju susjeda plus jedan, jer u selekciji sudjeluju susjedi i pripadajuća jedinka. Subpopulacije se međusobno preklapaju. Što je broj susjeda manji, subpopulacije su međusobno izoliranije. S jedne strane, podjela na subpopulacije omogućava nezavisnije pretraživanje prostora rješenja. S druge strane, preklapanje omogućava širenje (migraciju) dobrih rješenja po cijeloj populaciji [CAN98a]. Dobra rješenja se mogu širiti po cijeloj populaciji, jer se susjedi preklapaju (na slici 3.10 jedinka *j* je susjed jedinkama *i* i *k*). S porastom broja susjeda algoritam poprima sve lošija svojstva, jer se suboptimalna rješenja u početku evolucijskog procesa prebrzo prošire po cijeloj populaciji [CAN98a].

S druge strane, pretpostavimo slučaj kada je broj susjeda mali i neka je u početku evolucijskog procesa genetski algoritam pronašao lokalni optimum u jednom od procesorskih elemenata. Takvo krivo suboptimalno rješenje se neće brzo proširiti cijelom populacijom, jer su subpopulacije udaljenih procesorskih elemenata međusobno izolirane. Za vrijeme dok se lokalni optimum sporo širi (zbog malog broja susjeda), genetski algoritmi u pojedinim čvorovima imaju vremena pretraživati druga područja prostora rješenja i pronaći neko bolje rješenje od onog suboptimalnog. Stoga je broj susjeda obično puno manji od ukupne veličine populacije.

Novi parametri su: topologija i broj susjeda (slika 3.14). Za razliku od sekvencijskog algoritma koji kontrolira raznolikost populacije samo s dva parametra (veličinom populacije i vjerojatnosti mutacije), masovno paralelni genetski algoritam dodatno kontrolira raznolikost populacije i selekcijski pritisak s tim novim parametrima [SAR96]. Broj susjeda S_N određuje *stupanj izoliranosti* i značajno utječe na raznolikost, odnosno raspršenost populacije. Što je broj susjeda manji, stupanj izoliranosti je veći, ali je zato i raznolikost populacije veća [LIN97a]. Utjecaj topologije i broja susjeda na svojstva genetskog algoritma dan je u [SAR96].



Slika 3.14. **Primjeri topologija susjeda**

Broj mogućih susjeda ovisi o arhitekturi masovnog paralelnog računala. S porastom broja susjeda raste i opterećenje na komunikacijskim kanalima. Broj poruka razaslanih po komunikacijskim kanalima značajno utječe na svojstva paralelnog programa. U slučaju kada bi svaki procesorski element bio susjed svim ostalim procesorskim elementima, broj poruka koji bi razmjenjivao takav genetski algoritam rastao bi s kvadratom veličine populacije [TAL91]. Broj susjeda je obično mali i iznosi četiri ili osam kao što je to slučaj u [LOG92, TAL92].

Ako procesorski elementi imaju dovoljno veliki vlastiti spremnik, svaki procesor može sadržavati više jedinki, pa čak i cijelu subpopulaciju [LOG92]. U tom slučaju radi se o hijerarhijskom genetskom algoritmu, gdje se na višoj razini obavlja masovno paralelni genetski algoritam, a na nižoj razini distribuirani genetski algoritam. Prema načinu

obilježavanja hijerarhijskih genetskih algoritama takav algoritam se naziva MPGA/DGA hijerarhijskim genetskim algoritmom (poglavlje 3.5, slika 3.21).

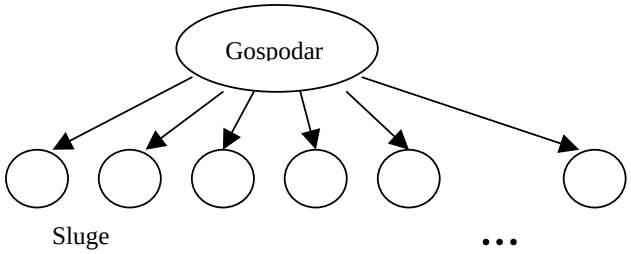
Tablica 3.2. **Prednosti i nedostaci masovno paralelnog genetskog algoritma**

Prednosti	Nedostaci
• gotovo linearno ubrzanje s porastom broja procesorskih elemenata	• za preveliki broj susjeda usko grlo postaje komunikacijski kanal • treba odrediti dva nova parametra: broj i topologiju susjeda • zahtjeva se računalo s velikim brojem procesora

Vrijeme izvođenja programa je gotovo onoliko puta kraće, koliko računalo ima procesora [TAL91]. S druge strane, upravo je zahtjev za posebnim paralelnim računalom s velikim brojem procesora najveći nedostatak MPGA.

3.4. Globalni paralelni genetski algoritam

Globalni paralelni genetski algoritam se sastoji od dretve gospodara i više dretvi slugu. Gospodar raspodjeljuje posao slugama i po potrebi sinkronizira proces razmjene jedinki (slika 3.15).



Slika 3.15. **Globalni paralelni genetski algoritam**

Tradicionalni globalni paralelni genetski algoritam sastoji se od jednog gospodara i više slugu koji obavljaju samo evaluaciju, tj. izračunavaju vrijednosti funkcije cilja (slika 3.16). Gospodar obavlja sve genetske operatore osim evaluacije i raspodjeljuje jedinke slugama. Radi se o jednopopulacijskom modelu, jer dretva gospodar ima pohranjenu cijelu populaciju u svojem korisničkom segmentu podataka. Sâm gospodar obavlja sve genetske operatore nad cijelom populacijom. Komunikacija između gospodara i slugu odvija se kada gospodar razašilje jedinke slugama i kada sluge vraćaju gospodaru izračunate vrijednosti funkcije cilja. Međutim, iako sluge najčešće obavljaju samo evaluaciju, oni mogu obavljati i ostale genetske operatore: selekciju križanje i mutaciju, ali se tada u nazivu izostavlja atribut tradicionalni.

```
Tradicionalni_globalni_paralelni_genetski_algoritam(){
   generiraj_početnu_populaciju_jedinki();
   dok (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      // SLUGE:
      evaluiraj_paralelno(); // evaluiraj paralelno jedinke
      // GOSPODAR:
      selektiraj();          // selektiraj jedinku za reprodukciju
      krizaj();              // obavi reprodukciju nad odabranim jedinkama
      mutiraj();
   }
}
```

Slika 3.16. **Tradicionalni globalni paralelni genetski algoritam**

Globalni paralelni genetski algoritam se relativno lako može implementirati na računalima sa zajedničkim (dijeljenim) radnim spremnikom[24] kao i na računalima s distribuiranim (raspodijeljenim) radnim spremnikom [CAN95, CAN98a, TOM99]. Najrasprostranjeniji paralelni računalni sustavi s raspodijeljenim radnim spremnikom su lokalne mreže. Stoga je isprva tradicionalni GPGA zamišljen za izvođenje na višeprocesorskom sustavu s raspodijeljenim radnim spremnikom. U tom modelu samo gospodar ima pristup jedinkama, jer je cijela populacija smještena u radnom spremniku gospodara. Tijekom evolucije gospodar mijenja jedinke mutacijom i stvara nove jedinke križanjem. Za svaku izmijenjenu ili novostvorenu jedinku treba se izračunati vrijednost funkcije cilja. Gospodar šalje te jedinke, odnosno cijeli genetski materijal slugama. Kao što je već rečeno, komunikacija između gospodara i slugu odvija se u dva navrata: kada gospodar slugama šalje genetski materijal za evaluaciju i kada sluge

[24] engleski: *share memory*

vraćaju gospodaru izračunate vrijednosti funkcije cilja. Na višeprocesorskom sustavu sa zajedničkim radnim spremnikom populacija je zajednička svim dretvama. Svaki od slugu može pristupati i obavljati evaluaciju nad njemu dodijeljenim jedinkama. Jedinke slugama dodjeljuje gospodar dinamički tijekom izvođenja programa, ali ne treba slati slugama cijeli genetski materijal, već samo redni broj jedinke ili kazaljku na jedinku za koju treba izračunati vrijednost funkcije cilja. Jednako tako, sluge ne trebaju vratiti gospodaru izračunate vrijednosti, već direktno upisuju te vrijednosti u spremničke lokacije na koje pokazuju poslane im kazaljke. Stoga je komunikacija preko zajedničkog radnog spremnika značajno brža od bilo kakve druge komunikacije nekom računalnom mrežom.

Komunikacija se obavlja na početku i na kraju faze evaluacije. Za vrijeme obavljanja svog posla sluge ne komuniciraju niti međusobno niti s gospodarom. Pretpostavlja se da je vrijeme potrebno za izračunavanje funkcije cilja puno veće od vremena koje se troši za komunikaciju između sluge i gospodara, jer bi, u suprotnom slučaju, GPGA obavljao posao sporije od sekvencijskog genetskog algoritma.

Globalni paralelni genetski algoritam može biti sinkroni ili asinkroni. Ako gospodar čeka dok sluge obave posao pa tek onda prelazi u sljedeću iteraciju, radi se o sinkronom algoritmu. Za razliku od ostalih paralelnih modela, sinkroni GPGA ima ista svojstva kao i sekvencijski GA, osim što je brži [CAN98a, CAN98b]. Primjerice, križanje pri DGA se obavlja nad dijelom populacije, dok GPGA kao i sekvencijski GA križanje obavlja nad cijelom populacijom. Povrh toga, GPGA nema nikakvih dodatnih parametara, dok DGA ima čak sedam novih parametara.

U svakoj iteraciji obavlja se N evaluacija, gdje je N veličina populacije. Evaluaciju obavljaju sluge nad onim jedinkama koje im pošalje gospodar. Čim obave svoj posao, sluge vraćaju izračunate vrijednosti gospodaru. Gospodar raspodjeljuje jedinke slugama i čeka dok sluge ne vrate izračunate vrijednosti ili eventualno obavlja evaluaciju nad ostatkom jedinki koje nisu razaslane slugama. U tom slučaju gospodar, nakon što zadnjem sluzi pošalje jedinke za evaluaciju, paralelno s njim obavlja evaluaciju nad ostatkom jedinki. Zadnji sluga i gospodar istodobno završavaju s evaluacijom, samo što gospodar treba još pričekati dok i taj zadnji sluga ne pošalje izračunate vrijednosti funkcije cilja. Pretpostavlja se da su ostale sluge već obavile svoj posao i poslale vrijednosti funkcije cilja gospodaru [CAN98c]. Ako gospodar obavlja neki drugi posao u trenutku kada mu stigne poruka s vrijednostima funkcije cilja od nekog od slugu, poruka se sprema u red poruka i tamo čeka sve dok je ne pročita gospodar.

U idealnom slučaju, kada bi vrijeme utrošeno za komunikaciju između gospodara i slugu bilo jednako nuli, ubrzanje bi bilo proporcionalno broju procesora. Međutim, vrijeme utrošeno za komunikaciju između procesora najčešće nije zanemarivo i neka iznosi T_c. Vrijeme trajanja koje potroši gospodar na slanje skupa jedinki svakom sluzi iznosi $S \cdot T_c$. Na kraju svake iteracije, gospodar čeka dok mu zadnji sluga ne pošalje vrijednosti funkcije cilja na što se dodatno potroši T_c vremena. Stoga ukupno vrijeme za komunikaciju u svakoj iteraciji iznosi $(S+1) \cdot T_c$. Nadalje, vrijeme trajanja evaluacije jedne jedinke neka iznosi T_f. Kod sekvencijskog genetskog algoritma ukupno trajanje evaluacije iznosi $N \cdot T_f$. Kod globalnog paralelnog genetskog algoritma evaluaciju obavljaju paralelno gospodar i S slugu. Neka sve sluge imaju dodijeljen jednak broj jedinki za evaluaciju i neka je vrijeme trajanja evaluacije konstantno. Tada je ukupno trajanje evaluacije skraćeno $S+1$ puta i iznosi $N \cdot T_f/(S+1)$. Ukupno vrijeme trajanja jedne iteracije, uz pretpostavku da je vrijeme potrebno da se obave genetski operatori zanemarivo u odnosu na vrijeme potrebno za evaluaciju i komunikaciju između gospodara i slugu, iznosi:

$$T_{uk} = (S+1) \cdot T_c + \frac{N \cdot T_f}{S+1}. \qquad (3.2)$$

Optimalni broj slugu S^* se dobiva ako deriviramo izraz (3.2) i izjednačimo ga s nulom [CAN98c]:

$$\frac{\partial T_{uk}}{\partial S} = T_c - \frac{N \cdot T_f}{(S^*+1)^2} = 0 \Rightarrow S^* = \sqrt{\frac{N \cdot T_f}{T_c}} - 1 \qquad (3.3)$$

Do sada je opisana sinkrona verzija tradicionalnog GPGA, jer gospodar čeka dok i zadnja dretva ne izračuna i pošalje vrijednosti funkcije cilja. Gospodar tek nakon što prikupi sve vrijednosti funkcije cilja započinje s izvođenjem sljedeće iteracije, jer u postupku selekcije moraju biti poznate vrijednosti funkcije cilja. U asinkronoj verziji tradicionalnog GPGA gospodar ne čeka da dretve-sluge obave svoj posao, već čim razašalje jedinke slugama i evaluira ostatak jedinki počinje s izvođenjem sljedeće iteracije. Tada se može dogoditi da prilikom postupka selekcije neke jedinke imaju krive vrijednosti funkcije cilja (stare vrijednosti iz prethodne iteracije). Stoga asinkroni algoritam ima lošija svojstva od sinkronog, odnosno sekvencijskog algoritma [CAN98a].

Tablica 3.3. **Prednosti i nedostaci tradicionalnog globalnog paralelnog genetskog algoritma**

Prednosti	Nedostaci
• sinkroni GPGA se ponaša jednako kao i sekvencijski GA osim što se izvodi brže na paralelnom računalu • jednostavan za implementaciju • nema dodatnih parametara	• nema značajnijeg skraćenja vremena izvođenja ukoliko je vrijeme trajanja izračunavanja funkcije cilja zanemarivo u odnosu na trajanje izvođenja genetskih operatora

Najznačajnija prednost tradicionalnog globalnog paralelnog genetskog algoritma nad ostalim paralelnim modelima je što se cijela teorija vezana uz sekvencijski GA može primijeniti i na sinkroni GPGA. S druge strane, jedini nedostatak je što se paralelizira samo evaluacija.

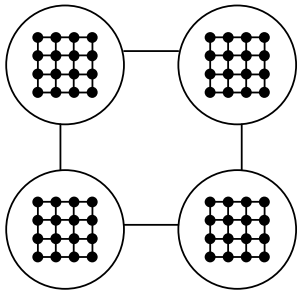
3.5. Hijerarhijski paralelni genetski algoritam

Hijerarhijski paralelni genetski algoritam[25] dobiva se kombinacijom dvaju od tri osnovna modela paralelnih genetskih algoritama. Tablica 3.4 prikazuje nekoliko modela hijerarhijskih paralelnih genetskih algoritama.

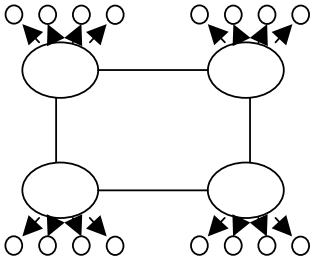
Tablica 3.4. **Primjeri hijerarhijskih modela paralelnih genetskih algoritama**

Viši nivo	Niži nivo	Slika
DGA	MPGA	Slika 3.17
DGA	GPGA	Slika 3.18
DGA	DGA	Slika 3.19
DGA	MPGA, DGA i GPGA	Slika 3.20
MPGA	DGA	Slika 3.21
MPGA	GPGA	Slika 3.22

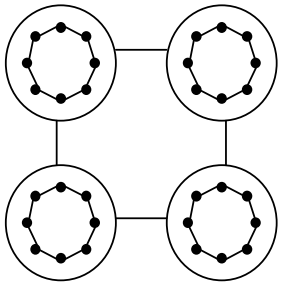
Primjerice, neka su na raspolaganju nekoliko masovno paralelnih računala spojenih u lokalnu mrežu. Model hijerarhijskog paralelnog genetskog algoritma bi izgledao ovako: na višem nivou je distribuirani genetski algoritam, a na nižem (na svakom računalu) je masovno paralelni (**Error! Reference source not found.**). S obzirom da je hijerarhijski paralelni model mješavina dvaju od tri ili čak svih triju osnovnih modela genetskih algoritama, naziva se još i miješanim paralelnim genetskim algoritmom[26] [CAN98a].



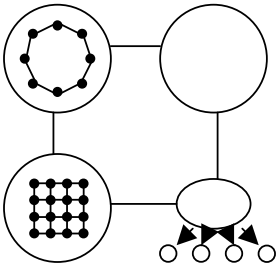
Slika 3.17. **DGA/MPGA hijerarhijski GA**



Slika 3.18. **DGA/GPGA hijerarhijski GA**



Slika 3.19. **DGA/DGA hijerarhijski GA**



Slika 3.20. **Primjer miješanog hijerarhijskog GA**

Na umreženim računalima s više procesora je moguće izvesti hijerarhijski paralelni genetski algoritam koji je na višem nivou distribuirani genetski algoritam, a na nižem globalno paralelni genetski algoritam (slika 3.18). Druga mogućnost je hijerarhijski paralelni genetski algoritam koji je i na višem i na nižem nivou distribuirani genetski algoritam (slika 3.19).

Na nižem nivou ne moraju biti svi isti paralelni modeli i mogu se, osim međusobno, kombinirati i s običnim sekvencijskim genetskim algoritmom kao što je to slučaj na slici 3.20.

Hijerarhijski modeli značajno ubrzavaju izvođenje algoritma. Neka je globalno paralelni genetski algoritam a_g puta brži, a distribuirani genetski algoritam a_d puta brži od sekvencijskog genetskog algoritma. Tada je hijerarhijski model koji kombinira ta dva modela točno $a_g \cdot a_d$ puta brži od serijske verzije programa [CAN98a]. To je ujedno i

[25] Neki autori pod pojmom *hibridni* PGA podrazumijevaju hijerarhijski PGA (kao primjerice u [TOM99]). U ovom radu se pridjev *hibridni* rabi za modele optimizacijskih algoritama koji kombiniraju genetski algoritam s nekim drugim optimizacijskim postupcima kako je to opisano u poglavlju 3.6.

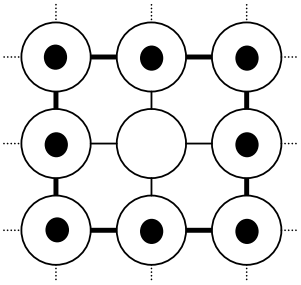
[26] engleski: *mixed parallel genetic algorithm*

najveća prednost tog modela. S druge strane, ističe se nedostatak isti kao i kod MPGA: postavljaju se veliki zahtjevi nad sklopovljem. Primjerice, minimalna konfiguracija za izvođenje najjednostavnijeg DGA/GPGA hijerarhijskog modela se sastoji od barem tri umrežena računala s dva procesora. Naime, minimalnu prstenastu topologiju DGA čine tri otoka, a minimalni broj procesora, da bi se uopće moglo ostvariti paralelno izvođenje GPGA, jest dva.

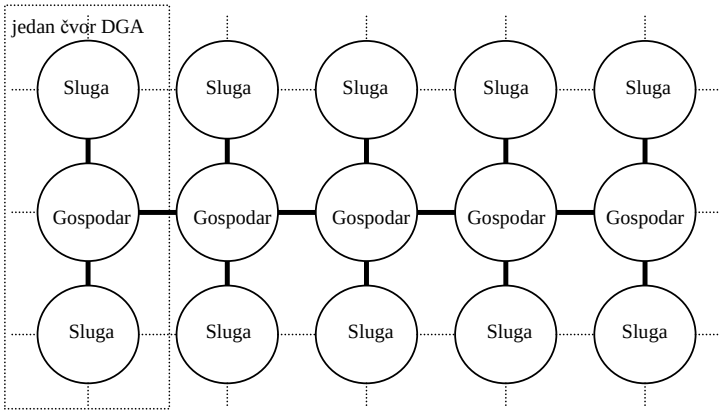
Tablica 3.5. **Prednosti i nedostaci hijerarhijskog genetskog algoritma**

Prednosti	Nedostaci
• ima najbolja svojstva • ukupno ubrzanje je produkt ubrzanja pojedinih modela	• veliki zahtjevi nad sklopovljem: nekoliko umreženih paralelnih računala ili masovno paralelno računalo s dovoljno velikim spremnikom na svakom procesorskom elementu da u njega stane cijela subpopulacija • ima mnogo parametara (ukupan broj parametara jednak je sumi svih parametara pojedinih modela PGA)

Hijerarhijski model gotovo u pravilu na višem nivou ima distribuirani genetski algoritam kako to pokazuju svi dosad navedeni primjeri. Na višem nivou mogao bi biti i masovno paralelni model PGA kada bi svaki procesorski element masovno paralelnog računala imao dovoljno veliki interni spremnik. Tada bi na nižem nivou mogao biti sam distribuirani genetski algoritam (slika 3.21) ili u kombinaciji s globalno paralelnim genetskim algoritmom (slika 3.22).



Slika 3.21. **MPGA/DGA hijerarhijski genetski algoritam**



Slika 3.22. **MPGA/DGA/GPGA hijerarhijski genetski algoritam**

3.6. Hibridni paralelni genetski algoritam

U kombinaciji s bilo kojim od četiri prethodno navedenih modela paralelnih genetskih algoritama (DGA, GPGA, MPGA ili HPGA) hibridni paralelni genetski algoritam (HiPGA) obavlja u svakoj iteraciji osim genetskih operatora i lokalno pretraživanje[27]. Za lokalno pretraživanje se najčešće koriste gradijentne metode[28] [JON93, MIT94, MUE91a, MUE91b, MUE92a]. Primjeri gradijentnih metoda su: metoda najbržeg spusta i Newton-Raphsonova metoda. Primjeri ostalih optimizacijskih metoda su: traženje po koordinatnim osima, postupak po Hookeu i Jeevesu

[27] engleski: *local search*
[28] engleski: *hill-climbing*

i simpleks postupak po Nelderu i Meadu [TUR89]. Nakon evaluacije svih novostvorenih ili izmijenjenih jedinki iteracija nije gotova, već se obavlja dodatno optimiranje uporabom neke druge optimizacijske metode. Početna točka je obično najbolja jedinka ili skup najboljih jedinki dobivenih u toj iteraciji.

Slika 3.23 prikazuje jedan oblik hibridnog paralelnog genetskog algoritma koji u kombinaciji s masovno paralelnim genetskim algoritmom obavlja lokalno pretraživanje [MUE91a]. Eksperimentalno je pokazano da je takav hibridni paralelni genetski algoritam efikasniji od bilo koje gradijentne metode s više početnih slučajno odabranih točaka [MUE92a].

```
Hibridni_paralelni_genetski_algoritam(){
   generiraj_paralelno_populaciju_slučajnih_jedinki();
   dok (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      evaluiraj();          // evaluiraj paralelno svaku jedinku
      optimiraj_lokalno(); // za svaku jedinku paralelno obavi lokalno
                              pretraživanje uporabom neke gradijentne
                              metode
      selektiraj();         // selektiraj paralelno jedinku za
                              reprodukciju među susjedima
      reproduciraj();       // obavi paralelno reprodukciju među
                              odabranom jedinkom iz prethodnog koraka
                              i vlastitom jedinkom
      optimiraj_lokalno(); // dijete paralelno optimiraj lokalno

      ako (je dijete bolje od roditelja){
         nadomjesti();      // nadomjesti paralelno vlastitu jedinku s boljom
      }                        novodobivenom jedinkom
   }
}
```

Slika 3.23. **Primjer hibridnog paralelnog genetskog algoritma**

Poznato je da genetski algoritam daje loše rezultate u finom podešavanju rješenja [MIC94]. Jedan od načina rješavanja tog problema je uporaba varijabilne duljine kromosoma. Pri završetku evolucijskog procesa kromosom postaje sve duži, čime se povećava preciznost [GOL96]. Povećavanjem preciznosti omogućuje se genetskom algoritmu fino podešavanje rješenja, ali tek na kraju evolucijskog procesa. Velika duljina kromosoma na početku evolucijskog procesa samo otežava grubo pretraživanje prostora rješenja. Drugi način na koji se može riješiti problem finog podešavanja rješenja jest upravo hibridni paralelni genetski algoritam. Hibridni genetski algoritam ima sva dobra svojstva sekvencijskog genetskog algoritma, s tim da je još poboljšano fino podešavanje rješenja.

Tablica 3.6. **Prednosti i nedostaci hibridnog genetskog algoritma**

Prednosti	Nedostaci
• bolje i brže fino podešavanje rješenja • brža konvergencija	• veća vjerojatnost zaostajanja u lokalnom optimumu zbog manje raspršenosti rješenja

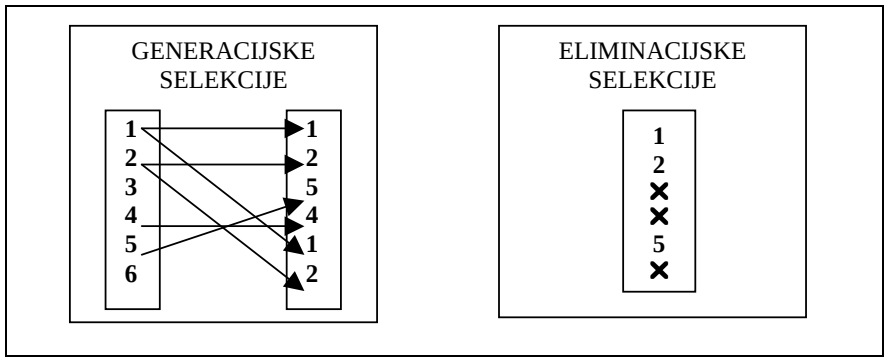
4. POSTUPCI SELEKCIJE

4.1. Podjela

Genetski algoritmi koriste selekcijski mehanizam za odabir jedinki koje će sudjelovati u reprodukciji. Selekcijom se omogućava prenošenje boljeg genetskog materijala iz generacije u generaciju. Zajedničko svojstvo svih vrsta selekcija je veća vjerojatnost odabira boljih jedinki za reprodukciju. Postupci selekcije se međusobno razlikuju po načinu odabira boljih jedinki. Prema načinu prenošenja genetskog materijala boljih jedinki u sljedeću iteraciju postupci selekcije se dijele na (slika 4.1):

- *generacijske selekcije* - selekcijom se direktno biraju bolje jedinke čiji će se genetski materijal prenijeti u sljedeću iteraciju i
- *eliminacijske selekcije* - biraju se loše jedinke za eliminaciju, a bolje jedinke prežive postupak selekcije.

Generacijskom selekcijom se odabiru bolje jedinke koje će sudjelovati u reprodukciji. Između jedinki iz populacije iz prethodnog koraka biraju se bolje jedinke i kopiraju u novu populaciju. Ta nova populacija, koja sudjeluje u reprodukciji, naziva se *međupopulacijom* [BLI95]. Na taj način se priprema nova generacija jedinki za postupak reprodukcije. To je ujedno prvi nedostatak generacijskih selekcija, jer se u radnom spremniku odjednom nalaze dvije populacije jedinki. Broj jedinki koje prežive selekciju je manji od veličine populacije. Najčešće se ta razlika u broju preživjelih jedinki i veličine populacije popunjava duplikatima preživjelih jedinki [MIC94]. Pojava duplikata u sljedećoj iteraciji je drugi nedostatak generacijskih selekcija, jer duplikati nikako ne doprinose kvaliteti dobivenog rješenja, već samo usporavaju evolucijski proces [DAV91, GOL96]. Generacijska selekcija postavlja granice među generacijama. Svaka jedinka egzistira točno samo jednu generaciju. Izuzetak su jedino najbolje jedinke koje žive duže od jedne generacije i to samo ako se primijeni elitizam (poglavlje 5.3).



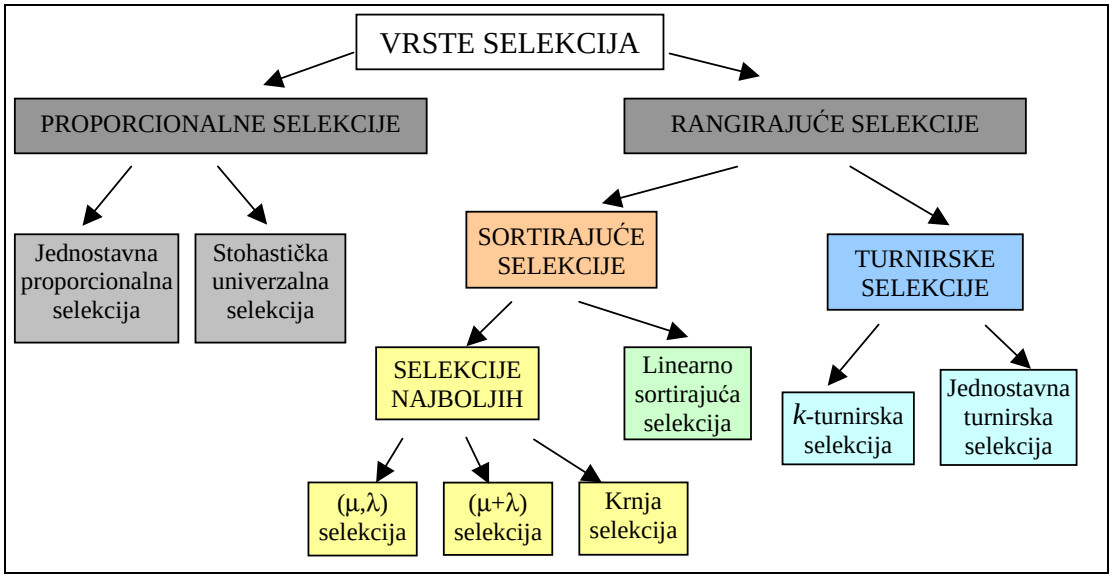
Slika 4.1. **Podjela postupaka selekcije prema načinu prenošenja genetskog materijala boljih jedinki u novu iteraciju**

Kod eliminacijskih selekcija bolje jedinke preživljavaju mnoge iteracije pa stroge granice između generacija nema, kao što je to slučaj kod generacijskih selekcija. Eliminacijska selekcija briše *M* jedinki. Parametar *M* naziva se *mortalitetom* ili *generacijskim jazom*. Izbrisane jedinke bivaju nadomještene jedinkama koje se dobivaju reprodukcijom. Eliminacijskom selekcijom se otklanjaju oba nedostatka generacijske selekcije: nema dvije populacije u istoj iteraciji i u postupku selekcije se ne stvaraju duplikati. Na prvi pogled je uvođenje novog parametra *M* nedostatak eliminacijske selekcije, jer se svakim dodatnim parametrom značajno produžava postupak podešavanja algoritma. Međutim, uvođenjem novog parametra, nestaje parametar vjerojatnost križanja, jer križanje treba obaviti točno onoliko puta koliko je potrebno da se nadopuni populacija do početne veličine *N*. Što je veća vjerojatnost križanja, treba biti veći mortalitet i obrnuto. Primjerice, ako se koristi uniformno križanje, koje producira jednu novu jedinku, tada se točno *M* puta treba ponoviti križanje kako bi se nadopunila populacija, odnosno zamijenile eliminirane jedinke.

Selekcijom se osigurava prenošenje *boljeg* genetskog materijala s *većom* vjerojatnošću u sljedeću iteraciju. Ovisno o metodi odabira boljih jedinki kod generacijskih selekcija, odnosno loših jedinki kod eliminacijskih selekcija, postupci selekcije se dijele na *proporcionalne* i *rangirajuće* [CAN99b]. Važno je napomenuti da je zajedničko obilježje svim selekcijama veća vjerojatnost preživljavanja bolje jedinke od bilo koje druge lošije jedinke. Selekcije se razlikuju prema načinu određivanja vrijednosti vjerojatnosti selekcije određene jedinke. Proporcionalne selekcije odabiru jedinke s vjerojatnošću koja je proporcionalna dobroti jedinke, odnosno vjerojatnost selekcije ovisi o kvocijentu dobrote jedinke i prosječne dobrote populacije. Broj jedinki s određenim svojstvom u sljedećoj iteraciji je proporcionalan kvocijentu prosječne dobrote tih jedinki i prosječne dobrote populacije. Tako glasi dio teorema sheme koji se odnosi na selekciju[29] [MIC94].

[29] Teorem sheme odnosi se na genetski algoritam s jednostavnom mutacijom, križanjem s jednom točkom prekida i jednostavnom proporcionalnom selekcijom. Teorem sheme glasi ovako: broj jedinki koje sadrže shemu niskog reda (red sheme je broj bitova koji čine shemu i što je on manji, to će mutacija teže uništiti shemu), kratke definirane duljine (što su sheme kraće, teže ih je križanjem uništiti) i iznadprosječne dobrote (ovaj dio teorema sheme odnosi se na selekciju) raste eksponencijalno. Stoga bi uporabom samo proporcionalne selekcije, bez križanja i mutacije, teorem sheme glasio ovako: broj jedinki u sljedećoj iteraciji koje sadrže shemu *S* je proporcionalan kvocijentu prosječne dobrote tih jedinki i prosječne dobrote populacije [BAU95, GOL89, MIC94].

Rangirajuće selekcije odabiru jedinke s vjerojatnošću koja ovisi o položaju jedinke u poretku jedinki sortiranih po dobroti. Rangirajuće selekcije se dijele na *sortirajuće* i *turnirske* selekcije. Sortirajuće selekcije su: linearno sortirajuća selekcija, μ-λ selekcije i krnja selekcija. Turnirske selekcije se dijele prema broju jedinki koje sudjeluju u *turniru*. Na slici 4.2 prikazana je podjela selekcija na proporcionalne i rangirajuće, te na njihove podvrste.



Slika 4.2. **Vrste selekcija**

Generacijske proporcionalne selekcije su predmet mnogih teorijskih razmatranja [ALT94, BAU95, FOS95, GOL89, GRE93, MIC94, MUE91a, SPE98a, SPE98b] što olakšava analizu rada genetskih algoritama s takvom selekcijom. Međutim, proporcionalna selekcija obavlja u svakoj iteraciji izračunavanje srednje vrijednosti dobrote populacije i translaciju prema izrazu (2.1). Osim toga, obično se ugrađuje i elitizam, pa treba u svakoj iteraciji pronaći unaprijed zadan broj najboljih jedinki. Dakle, mora se obaviti barem djelomično sortiranje jedinki prema dobroti. Postupak sortiranja, kao i postupak izračunavanja nekih statističkih vrijednosti troši stanovito procesorsko vrijeme. Štoviše, kod paraleliziranja ti postupci se označavaju kao kritični odsječci pa ih je potrebno sinkronizirati (jer dretve mogu nastaviti s radom tek kad se oni obave). Sinkronizacijski mehanizmi dodatno usporavaju algoritam. U novije vrijeme intenzivno se razmatraju mogućnosti rangirajućih selekcija baš zbog navedenih nedostataka proporcionalnih selekcija [CAN99a, CAN99b, MIL95, WHI96, YOS99].

Svojstva selekcije koja utječu na efikasnost rada genetskog algoritma su: trajanje postupka selekcije i selekcijski pritisak. Trajanje postupka selekcije je moguće skratiti paralelizacijom, naravno, ukoliko je postupak moguće paralelizirati i to po mogućnosti bez sinkronizacije (da se ne troši vrijeme za sinkronizaciju).

Genetski algoritam eksplicitno ne definira genetske operatore (slika 2.1), tj. genetski algoritam ne kaže na koji način, odnosno *kako* genetski operatori obavljaju svoj posao, već *što* trebaju učiniti. Selekcija odabire bolje jedinke za reprodukciju. Križanjem se svojstva roditelja prenose na djecu, a mutacijom se slučajno mijenja genetski materijal. Kako će križanje osigurati prenošenje genetskog materijala na djecu i kako će mutacija izmijeniti genetski materijal, ovisi o samom problemu koji se rješava i o vrsti prikaza rješenja. Tako, primjerice, križanje s jednom točkom prekida nije pogodno za rješavanje problema trgovačkog putnika, ali se zato može primijeniti za optimiranje funkcija realne varijable i binarni prikaz. Jednako tako nije važno kako će selekcija odabrati bolje jedinke za reprodukciju, već je važno da bolje jedinke imaju veću vjerojatnost preživljavanja od lošijih. Osim toga, važno je i *koliko* veću vjerojatnost odabira imaju bolje jedinke od lošijih.

4.2. Osnovni pojmovi

Selekcijski pritisak je odnos vjerojatnosti preživljavanja dobrih i loših jedinki. Drugim riječima, selekcijski pritisak je sposobnost preživljavanja jedinki. Neka vjerojatnosti selekcije dvije jedinke s oznakama i i j iznose $p(i)$ i $p(j)$ respektivno. Neka je jedinka s oznakom i bolja od one s oznakom j, tj. neka je $d(i)>d(j)$, odnosno $p(i)>p(j)$. Selekcijski pritisak je veći što je kvocijent $p(i)/p(j)$ veći ili što je veća razlika $p(i)-p(j)$. Selekcija ima veliki selekcijski pritisak ukoliko s velikom vjerojatnošću prenosi bolje jedinke u iduću iteraciju, odnosno s velikom vjerojatnošću eliminira jedinke ispodprosječne dobrote [MIL95]. Selekcijski pritisak utječe na kvalitetu rješenja. Genetski algoritmi s velikim selekcijskim pritiskom pronalaze prije rješenje, tj. brže konvergiraju žrtvujući kvalitetu dobivenog rješenja, jer brža konvergencija uzrokuje veću vjerojatnost zaglavljivanja u lokalnom optimumu [CAN99b]. S druge strane, ako je selekcijski pritisak premali, nepotrebno se troši vrijeme na beskorisne iteracije jer je konvergencija u tom slučaju prespora [MIL95].

Parametri selekcije utječu na selekcijski pritisak, odnosno na brzinu konvergencije algoritma. Na konvergenciju algoritma ne utječe samo selekcija, nego i ostali operatori, pa brzina konvergencije ne može biti mjera za selekcijski

pritisak. Selekcijski pritisak se posredno određuje (kvantificira) uz pomoć *trajanja preuzimanja*, *selekcijske razlike*, *selekcijskog intenziteta*, *gubitka raznovrsnosti* ili *reprodukcijske stope*.

Trajanje preuzimanja[30] je prosječan broj generacija nakon kojih se populacija sastoji od N duplikata najbolje jedinke, ukoliko se u svakoj iteraciji uporabi samo operator selekcije bez križanja i mutacije. Drugim riječima, mjeri se broj iteracija nakon kojih se eliminiraju sve jedinke osim najbolje. Što je trajanje preuzimanja manje, selekcijski pritisak je veći i obrnuto.

Definicija 1 (trajanje preuzimanja)

Neka se populacija $P(t)$ sastoji od skupa jedinki $\{J_1(t), J_2(t), ..., J_N(t)\}$, tj.

$$P(t)=\{J_1(t), J_2(t), ..., J_N(t)\},$$

gdje je $t\in\mathbf{N}$ redni broj generacije (generacijski GA) ili redni broj iteracije (eliminacijski GA)[31].

Neka je, nadalje, $P(0)$ početna (inicijalna) populacija veličine N i $J_{max}\in P(0)$ najbolja jedinka početne populacije, pri čemu je $max\in[1,N]$. Neka se nova populacija $P(t+1)$ dobiva samo uporabom operatora selekcije $g : \mathbf{J}^N\rightarrow \mathbf{J}^N$, tj. neka je:

$$P(t+1)=g[P(t)].$$

Neka $\iota[P(t)]$ označava predikat koji je istinit kada je $P(t)=[J_1(t)=J_2(t)=...=J_N(t)=J_{max}]$. Tada se

$$T=\min\{t \mid \iota[g^t(P(0))]=\text{ISTINITO}\}$$

naziva *trajanje preuzimanja* operatora selekcije g.

Trajanje preuzimanja poprima najmanju vrijednost kada je vjerojatnost selekcije svih jedinki jednaka i iznosi $p=1/N$ (slučajno pretraživanje). Osim o vrsti selekcije i njezinim kontrolnim parametrima, trajanje preuzimanja za proporcionalne selekcije ovisi i o funkciji cilja.

Selekcijom preživljavaju bolje jedinke i očekuje se da prosječna vrijednost preživjelih jedinki bude veća od prosječne vrijednosti cijele populacije. Posljedica selekcijskog pritiska je *selekcijska razlika*[32] između preživjelih jedinki i cijele populacije [CAN99b]. Selekcijska razlika s je razlika prosječne vrijednosti dobrote preživjelih jedinki $\overline{d_p}$ i prosječne vrijednosti dobrote svih jedinki $\overline{d}$ u svakoj iteraciji t:

$$s(t) = \overline{d_p}(t) - \overline{d}(t) . \qquad (4.1)$$

Prosječna vrijednost dobrote svih jedinki koje čine populaciju računa se prema izrazu:

$$\overline{d}(t) = \frac{1}{N}\sum_{i=1}^{N} d_i(t) . \qquad (4.2)$$

Nadalje, neka je $\sigma(t)$ standardna devijacija svih dobrota u populaciji u generaciji t:

$$\sigma(t) = \sqrt{\frac{\sum_{i=1}^{N}\left[d_i(t) - \overline{d}(t)\right]^2}{N}} . \qquad (4.3)$$

Uz pretpostavku da dobrota populacije d_i ima normalnu razdiobu $N[\overline{d}(t),\sigma(t)]$ u generaciji t, selekcijska razlika je proporcionalna standardnoj devijaciji populacije:

$$s(t) = s_I \cdot \sigma(t) , \qquad (4.4)$$

pri čemu se faktor $s_I=s(t)/\sigma(t)$ naziva *selekcijskim intenzitetom* [BAE95a].

Naravno, dobrota nema normalnu razdiobu za sve funkcije cilja i u svakom trenutku t. Međutim, za većinu funkcija cilja pred kraj evolucijskog procesa ($t>>$) razdioba vrijednosti funkcije dobrote populacije može se dobro aproksimirati normalnom razdiobom [BAE95a].

Selekcijski pritisak je veći što je veća selekcijska razlika, odnosno selekcijski intenzitet. Selekcijska razlika ovisi o standardnoj devijaciji populacije koja se mijenja iz generacije u generaciju, a selekcijski intenzitet je konstantna vrijednost koja ovisi samo o vrsti selekcije i njezinim parametrima. Stoga se selekcijski intenzitet može koristiti kao mjera za selekcijski pritisak uz ograničenje da vrijednosti funkcije dobrote populacije imaju normalnu razdiobu.

Gubitak raznovrsnosti je postotak jedinki koje nisu selektirane za reprodukciju.

Reprodukcijska stopa je omjer broja jedinki s određenom vrijednošću dobrote poslije i prije selekcije [BLI95]. Neka je $Q(d)$ broj jedinki čija je vrijednost funkcije dobrote jednaka d i to prije, a $Q^*(d)$ poslije selekcije. Reprodukcijska stopa r za zadanu vrijednost dobrote d računa se prema izrazu:

---

[30] engleski: *takeover time*
[31] Zbog sličnosti s prirodnom evolucijom, u literaturi se t naziva još i vrijeme.
[32] engleski: *selection differential*

$$r(d) = \begin{cases} \dfrac{Q^*(d)}{Q(d)} & : \quad Q(d) > 0 \\ 0 & : \quad Q(d) = 0 \end{cases}$$

Svi postupci selekcije favoriziraju bolje jedinke pa je njihova reprodukcijska stopa veća od jedan. Odnosno, sve selekcije kažnjavaju loše jedinke pa je reprodukcijska stopa za loše jedinke manja od jedan. Selekcijski pritisak je veći što je reprodukcijska stopa za bolje jedinke veća, odnosno što je za loše jedinke manja.

Najslabiji selekcijski pritisak imaju proporcionalne selekcije. Veći selekcijski pritisak imaju linearno sortirajuća i turnirske selekcije, a najveći (μ,λ) selekcije [BAE94].

Poželjno je da selekcijski mehanizam bude takav da se na jednostavan način može kontrolirati selekcijski pritisak. Dodatni zahtjevi nad postupkom selekcije prema Bäcku [BAE94] su:

- promjena kontrolnih parametara selekcije mora biti jednostavna, a posljedice te promjene moraju biti predvidive;
- poželjno je da selekcija ima samo jedan kontrolni parametar i
- raspon selekcijskog pritiska treba biti što veći.

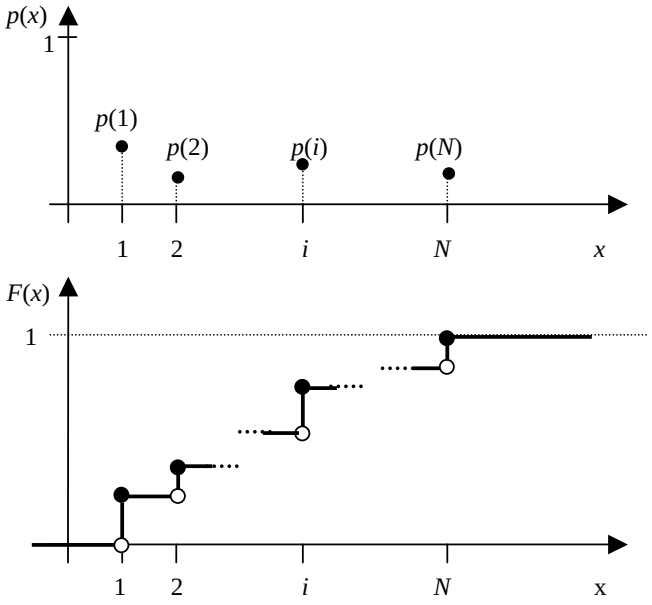
Nažalost većina postupaka selekcije ne zadovoljava niti prvi zahtjev [BAE94].

Kumulativna dobrota $d_s(i)$ je suma vrijednosti dobrote svih jedinki koje nose oznaku manju ili jednaku i:

$$d_s(i) = \sum_{n=1}^{i} d(n) \qquad (4.5)$$

4.3. Vjerojatnost selekcije

Neka je Ω prostor elementarnih događaja: $\Omega=\{E_1,E_2,...,E_i,....,E_N\}$, gdje je E_i elementarni događaj selekcija i-te jedinke. Neka je postupkom selekcije odabrana jedinka s oznakom x. Neka je X diskretna slučajna varijabla koja poprima vrijednosti $x \in \mathrm{R}$, gdje je $\mathrm{R}=\{1,2,3,...,i,...,N\}$ diskretan skup od N elemenata. Oznaka $p(i)=P\{x=i\}$ označava vjerojatnost selekcije i-te jedinke, $J_i \in P$ (slika 4.3).



Slika 4.3. **Vjerojatnost selekcije *p* i pripadajuća funkcija razdiobe *F***

Suma svih vjerojatnosti mora biti jednaka jedan:

$$\sum_{i=1}^{N} p(i) = 1 . \qquad (4.6)$$

Kako bi razlikovali vjerojatnost selekcije jedinke za sljedeću iteraciju od vjerojatnosti selekcije jedinke za eliminaciju, dodaju se oznake G, odnosno E, te je $p_G(i)$ vjerojatnost selekcije i-te jedinke za sljedeću iteraciju, a $p_E(i)$ je vjerojatnost eliminacije i-te jedinke.

Funkcija F neka je funkcija razdiobe slučajne varijable X: $F(x)=P\{X \le x\}$, gdje je

$$F(x) = \sum_{i=1}^{x} p(i) . \qquad (4.7)$$

Na slici 4.3b je prikazan primjer funkcije razdiobe *F*.

Kod rangirajućih selekcija spomenuta oznaka jedinke ovisi o njenom rangu. Najbolja jedinka ima oznaku 1, druga jedinka po dobroti ima oznaku 2 i tako sve do najslabije jedinke koja ima oznaku *N*. Jednako tako, jedinke mogu nositi oznake obrnutim redoslijedom: tako da najbolja ima oznaku *N*, a najgora oznaku 1. Kod proporcionalnih selekcija oznaka ne odgovara rangu jedinke, već predstavlja samo redni broj jedinke u populaciji (prva jedinka ne mora biti najbolja, kao što niti zadnja ne mora biti najgora).

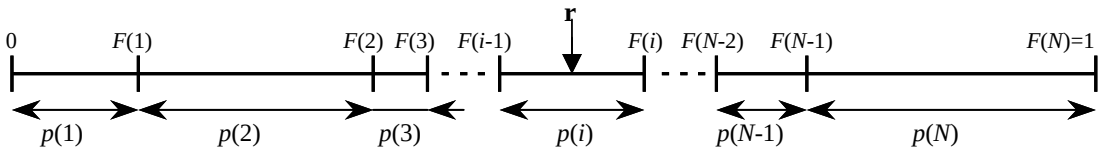
4.4. Proporcionalne selekcije

4.4.1. Jednostavna proporcionalna selekcija

Holland je 1975. godine predstavio jednostavni genetski algoritam s jednostavnom proporcionalnom selekcijom[33], jednostavnom mutacijom i križanjem s jednom točkom prekida. Osim toga, Holland je još tada postavio temelje teorijske analize rada genetskih algoritama opisavši jednostavni genetski algoritam hipotezom gradivnih blokova i donekle predvidjevši ponašanje algoritma uz pomoć teorema sheme (vidi napomenu 29 na stranici 22).

Vjerojatnost preživljavanja jedinke pri proporcionalnoj selekciji proporcionalna je njezinoj dobroti. Često se pri opisu proporcionalnih selekcija koristi analogija s kotačem ruleta. Analogija nije sasvim potpuna, jer svi brojevi na obodu kotača ruleta zauzimaju jednake kružne isječke, dok je zamišljena veličina kružnog isječka jedinke proporcionalna njezinoj dobroti, tj. vjerojatnosti njezine selekcije.

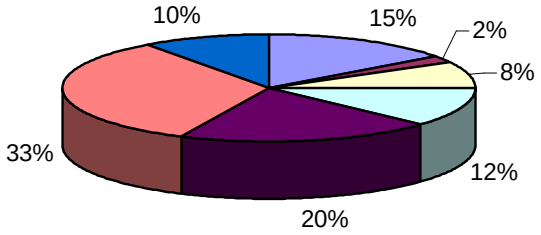
Generacijska proporcionalna selekcija u svakom koraku generira slučajni broj *r* u intervalu [0,1]. Ako je *r* u intervalu [*F*(*i*-1), *F*(*i*)], gdje je *F*(*x*) već opisana funkcija razdiobe, tada je selektirana *i*-ta jedinka (slika 4.4). Taj se postupak ponavlja sve dok se ne odabere *N* jedinki, s tim da se jedna te ista jedinka može odabrati proizvoljan broj puta.



Slika 4.4. **Selekcija *i*-te jedinke uporabom jednostavne proporcionalne selekcije**

Vjerojatnost selekcije jedinke, tj. veličina njenog kružnog isječka, određuje se na temelju dobrote jedinke d_i i ukupne dobrote populacije Σd_i. Vjerojatnost selekcije *i*-te jedinke $p_G(i)$ koja će sudjelovati u reprodukciji jednaka je kvocijentu dobrote jedinke i ukupne dobrote populacije [BAE94]:

$$p_G(i) = \frac{d_i}{\sum_{n=1}^{N} d_n}. \qquad (4.8)$$



Slika 4.5. **Primjer dobrota raspoređenih po *kotaču ruleta***

Primjerice, neka se populacija sastoji od 7 jedinki čije su dobrote: 10, 15, 2, 8, 12, 20 i 33. Dobrota populacije je suma dobrota svih jedinki i iznosi 100. Vjerojatnosti selekcije su 10/100=0.1 (10%), 15/100=0.15 (15%), itd. (slika 4.5).

Vjerojatnost eliminacije *i*-te jedinke za eliminacijsku proporcionalnu selekciju iznosi:

$$p_E(i) = \frac{d_{\max} - d_i}{N \cdot d_{\max} - \sum_{i=1}^{N} d_i}. \qquad (4.9)$$

[33] engleski: *roulette wheel selection*

Eliminacijska proporcionalna selekcija ne koristi dvije populacije niti ne stvara duplikate. Povrh toga, eliminacijskom selekcijom je inherentno ugrađen elitizam. Najbolja jedinka se ne može eliminirati, jer je vjerojatnost selekcije najbolje jedinke, prema izrazu (4.9) jednaka nuli $p_E(max)=0$.

Postupak proporcionalne selekcije nije moguće paralelizirati bez sinkronizacije izračunavanja ukupne dobrote. Prije no što više dretvi počinje obavljati selekciju, jedna od njih treba izračunati ukupnu dobrotu kako bi se mogla izračunati vjerojatnost selekcije pojedine jedinke. Međutim, dretve troše procesorsko vrijeme prilikom ulaska i izlaska iz kritičnih odsječaka koristeći neki od mehanizama međusobnog isključivanja (semafori, uvjetne varijable i sl.). Povrh toga, značajan udio procesorskog vremena dretve troše i na čekanje (npr. u redu semafora) dok se ne zadovolji neki uvjet, odnosno dok neka druga dretva ne završi određeni dio posla. U nekim slučajevima, kako je to eksperimentalno pokazano, algoritam će više vremena potrošiti na sinkronizaciju, nego na postupak selekcije [BUD98]. To je razlog zašto se sâm postupak proporcionalne selekcije ne paralelizira. Ipak, ako GA s proporcionalnom selekcijom djeluje samo nad dijelom populacije, tada se više takvih algoritama mogu izvoditi istodobno. Stoga se proporcionalne selekcije mogu primijeniti kod distribuiranih genetskih algoritama, jer se selekcija obavlja nezavisno i paralelno nad nekoliko odvojenih subpopulacija (računa se ukupna dobrota subpopulacije, a ne cijele populacije). Ona se može primijeniti i kod masovno paralelnih genetskih algoritama, jer, slično kao i kod DGA, selekcija se obavlja samo nad dijelom populacije, točnije samo nad susjedima. Tradicionalni globalni genetski algoritam također može koristiti bilo koji postupak selekcije, pa i proporcionalnu selekciju, jer je paraleliziran samo postupak evaluacije (selekcija, križanje i mutacija se odvijaju sekvencijski). Općenito, za globalni paralelni genetski algoritam, proporcionalna selekcija nije pogodna upravo zbog nemogućnosti efikasne paralelizacije.

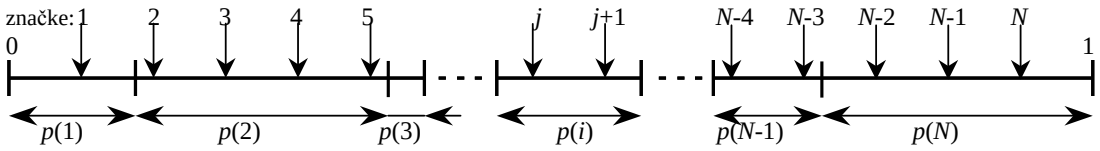
Izrazi (4.8) i (4.9) ne vrijede ako funkcija dobrote poprima negativne vrijednosti. U tom bi slučaju, prema navedenim izrazima, vjerojatnost selekcije mogla postati negativna, što nema smisla. Stoga, u općem slučaju, funkcija dobrote ne može biti jednaka funkciji cilja. Da bi se otklonio ovaj nedostatak, prije izračunavanja vjerojatnosti selekcije mora se obaviti sortiranje (poglavlje 4.5.2) ili translacija [GOL96]. Postupak translacije opisan izrazom (2.1) zahtijeva dodatni postupak pronalaženja najslabije jedinke u svakoj iteraciji.

Loše odabrana funkcija dobrote može postupak selekcije pretvoriti u slučajno odabiranje. Primjerice, neka je $f(x)=1000+\sin(x)$, gdje je $x\in[-2\pi, 2\pi]$. Ako se populacija sastoji od $N=100$ jedinki, vjerojatnost selekcije bilo koje jedinke bit će približno 0.01. Točnije, prema izrazu (4.8) vjerojatnost selekcije najbolje moguće jedinke za navedeni primjer iznosi 0.01001, a najlošije 0.00999. Posljedica toga je da će najbolja jedinka u populaciji preživjeti selekciju sa samo 0.002% većom vjerojatnošću od najlošije jedinke. To znači da gotovo nema selekcijskog pritiska, odnosno, genetski algoritma će se ponašati kao da ni nema selekcije. Uslijed nedostatka selekcijskog pritiska, genetski algoritam se pretvara u slučajno pretraživanje.

Jednostavna proporcionalna selekcija je vremenski najzahtjevnija, nije pogodna za paralelizaciju, postavlja ograničenja nad funkcijom dobrote ili se moraju obaviti neke vremenski zahtjevne predradnje i u svakom koraku se mora računati kumulativna dobrota. Unatoč navedenim brojnim nedostacima, jednostavna proporcionalna selekcija se i danas najviše primjenjuje upravo zahvaljujući poznatim teorijskim osnovama vezanih uz GA s takvom selekcijom [BAU95, CAN99b, GRE93, MIC94, MUE99, SPE98].

4.4.2. Stohastička univerzalna selekcija

Stohastička univerzalna selekcija se razlikuje od jednostavne proporcionalne selekcije prema načinu odabira jedinki. Vjerojatnost selekcije je ista i izračunava se također prema izrazu (4.8) [RUD94, CAN99b]. Jednostavna proporcionalna selekcija se obavlja u N koraka: N puta se generira slučajan broj prema kojem se određuje koja jedinka je selektirana. Stohastička univerzalna selekcija obavlja odabir svih jedinki u jednom koraku: generira se N ekvidistantnih znački tako da se interval [0,1] podijeli na $N+1$ jednakih odsječaka (slika 4.6).



Slika 4.6. **Selekcija svih *N* jedinki u jednom koraku uporabom stohastičke univerzalne selekcije**

Broj znački koje pokazuju na pojedinu jedinku određuje koliko će se njezinih kopija prenijeti u sljedeću populaciju. Na slici 4.6 je prikazan primjer u kojem se prenosi: jedna kopije jedinke J_1, četiri kopije jedinke J_2, ni jedan kopija jedinke J_3, dvije kopije jedinke J_i, itd.

4.4.3. Prednosti i nedostaci proporcionalnih selekcija

Nedostaci proporcionalnih selekcija su:

- Nad funkcijom cilja, ako se ona izravno koristi za izračunavanje vjerojatnosti selekcije, postavljena su sljedeća ograničenja: ne smije poprimati negativne vrijednosti i ne smije poprimati samo približno jednake velike vrijednosti. U prvom slučaju bi prema izrazu (4.8) vjerojatnost selekcije bila negativna, a u drugom slučaju se

genetski algoritam pretvara u postupak slučajnog pretraživanja, jer sve jedinke imaju približno istu vjerojatnost selekcije. Rješenje tih problema je funkcija dobrote koja se dobiva translacijom funkcije cilja. Translacija funkcije cilja se mora obaviti u svakom koraku i to usporava rad genetskog algoritma;

- Nemoguće je paralelizirati postupak selekcije bez sinkronizacije dretvi, jer se u svakoj iteraciji mora izračunati ukupna dobrota populacije koja mora biti dostupna svim dretvama;
- Eliminacijska proporcionalna selekcija se mora obavljati sekvencijski, jer svaka eliminacija jedinke ima za posljedicu promjenu vjerojatnosti eliminacije svih ostalih (preživjelih) jedinki;
- Troše znatno više procesorskog vremena od rangirajućih selekcija;
- Nemaju mogućnosti podešavanja selekcijskog pritiska (nema parametra s kojim bi se mogao podešavati selekcijski pritisak);
- Selekcijski pritisak ovisi o funkciji dobrote. To znači da je za svaku funkciju cilja drugačiji selekcijski pritisak, a time i brzina konvergencije koja direktno utječe na kvalitetu dobivenih rješenja.

Nasuprot navedenim nedostacima, postoji samo jedna prednost proporcionalnih selekcija:

- Analiza rada genetskog algoritma je olakšana, jer gotovo sva teorijska istraživanja se temelje na proporcionalnim selekcijama. Stoga se, unatoč svim navedenim nedostacima, najčešće i koristi u praksi. Tek u novije vrijeme se teorijski razmatraju ostali postupci selekcija.

4.5. Rangirajuće selekcije

4.5.1. Podjela i svojstva rangirajućih selekcija

Kod rangirajućih selekcija vjerojatnost selekcije ne ovisi izravno o dobroti, već o položaju jedinke u poretku jedinki sortiranih po dobroti.

Jedinke se mogu sortirati ili s rastućom ili padajućom dobrotom, odnosno po rastućim ili padajućim vrijednostima funkcije cilja. U prvom slučaju je: $f_1 \le f_2 \le f_3 \le ... \le f_i \le ... \le f_N$, a u drugom: $f_1 \ge f_2 \ge f_3 \ge ... \ge f_i \ge ... \ge f_N$.

Pri sortiranju s padajućom dobrotom, najbolja jedinka ima indeks i=1, a najlošija i=N, te je i-1 broj jedinki koje su bolje od i-te jedinke.

Za rangirajuću selekciju nije važan iznos dobrote, već njen odnos prema dobroti ostalih jedinki. Isto tako nije važno da li je funkcija dobrote negativna ili poprima samo približno jednake vrijednost (vidi primjer funkcije $f(x)$=1000+sin(x) u poglavlju 4.4.1). Kod rangirajućih selekcija funkcija dobrote nema nikakvih ograničenja i jednaka je funkciji cilja:

$$d(x)=f(x). \qquad (4.10)$$

Rješenje x_1 je bolje od rješenja x_2, ako je $f(x_1)>f(x_2)$ i ukoliko se traži maksimum funkcije f. Nadalje, ako je $f(x_1)=f(x_2)$, bilo koje od dva rješenja se može proglasiti *boljim* i prenijeti ga u novu generaciju. Nad funkcijom dobrote se ne postavljaju nikakva ograničenja (ne treba biti derivabilna, neprekidna i sl.). Dovoljno je da se za svaki par *potencijalnih rješenja* x_1 i x_2, koji su elementi prostora rješenja (skupa Ω), moguće odrediti da li je vrijednosti funkcija cilja $f(x_1)$ veća, jednaka ili manja od $f(x_2)$.

U skupinu rangirajućih selekcija spadaju sortirajuće i turnirske selekcije. Sortirajuće selekcije obavljaju sortiranje jedinki prema njihovoj vrijednosti funkcije cilja. Postupak troši značajni dio od ukupnog procesorskog vremena, jer se ponavlja u svakoj iteraciji. S obzirom da je sortiranje vremenski zahtijevan posao u odnosu na trajanje same selekcije, sortirajuće selekcije se rijetko koriste. Postupak sortiranja se ne isplati paralelizirati za veličine populacija kakve se najčešće koriste u primjeni (od 30 do 100 jedinki). Stoga rangirajuće selekcije nisu pogodne za paralelno izvođenje. Postupak sortiranja značajno usporava proces selekcije i ako je ikako moguće, treba ga izbjeći kao i postupak sinkronizacije.

S druge strane, za turnirske selekcije važan je samo odnos između nekoliko slučajno odabranih jedinki i nije potrebno obavljati sortiranje cijele populacije. Prema broju slučajno odabranih jedinki iz skupa svih jedinki (cijele populacije), turnirske selekcije se dijele na 2-turnirske, 3-turnirske, 4-turnirske itd.

4.5.2. Linearno sortirajuća selekcija

Kod linearno sortirajuće selekcije vjerojatnost selekcije je proporcionalna rangu, odnosno poziciji jedinke u poretku jedinki sortiranih po dobroti. Vjerojatnost selekcije za generacijsku linearno sortirajuću selekciju izračunava se prema izrazu:

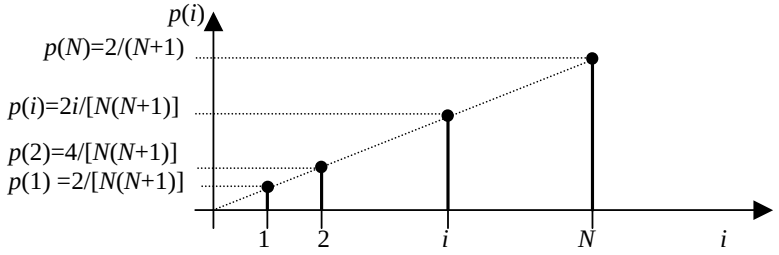
$$p(i) = \frac{i}{\sum_{i=1}^{N} i} = \frac{2i}{N(N+1)}, \qquad (4.11)$$

s tim da najbolja jedinka ima indeks *N*, a najgora 1. Vjerojatnost eliminacije za eliminacijsku linearno sortirajuću selekciju izračunava se prema istom izrazu (4.11), samo što su jedinke sortirane od najgore do najbolje, tj. najbolja jedinka ima indeks 1, a najgora *N*.

Suma vjerojatnosti selekcija svih jedinki mora biti jednaka 1:

$$\sum_{i=1}^{N} p(i) = \sum_{i=1}^{N} \frac{2i}{N(N+1)} = \frac{2}{N(N+1)} \sum_{i=1}^{N} i = \frac{2}{N(N+1)} \cdot \frac{N(N+1)}{2} = 1 \qquad \blacksquare$$

Slika 4.11 prikazuje razdiobu vjerojatnosti generacijske linearno sortirajuće selekcije. Točke koje predstavljaju vjerojatnost selekcije nalaze na pravcu (otuda i naziv linearna selekcija). Pravac ima nagib 2/[*N*(*N*+1)]. Svojstvo generacijske selekcije je da *i*-ta jedinka ima *i* puta veću vjerojatnost preživljavanja od najgore jedinke s oznakom 1. S druge strane, svojstvo eliminacijske selekcije je da *i*-ta jedinka ima *i* puta veću vjerojatnost eliminacije od najbolje jedinke s oznakom 1.



Slika 4.7. **Razdioba vjerojatnosti selekcije za generacijsku linearno sortirajuću selekciju**

Vjerojatnost selekcije najbolje ili najlošije jedinke može biti unaprijed zadana. Primjerice, vjerojatnost eliminacije najbolje jedinke treba biti jednaka nuli, ako se elitizmom štiti najbolja jedinka od eliminacije. Izraz (4.11) ne uključuje mogućnost slobodnog izbora vjerojatnosti selekcije neke jedinke, jer prema tom izrazu vjerojatnost selekcije ovisi samo o rangu jedinke i veličini populacije. Nagib pravca na kojem leže vjerojatnosti selekcija ovisi i o vjerojatnosti selekcije jedinke s oznakom 1 (ta jedinka je ili najbolja ili najlošija). Razdiobu vjerojatnosti linearno sortirajuće selekcije ispunjava sljedeća dva uvjeta:

- vjerojatnosti selekcije moraju ležati na pravcu, tj. $p(i)=a+b\cdot i$ i
- suma vjerojatnosti selekcije svih jedinki mora biti jednaka 1: $\Sigma p(i)=1$.

Uzevši u obzir ta dva uvjeta mogu se izračunati vrijednosti *a* (sjecište pravca s osi *y*) i *b* (nagib pravca). I konačno, općeniti izraz za izračunavanje vjerojatnosti selekcije, koja ovisi o rangu jedinke *i*, veličini populacije *N* i vjerojatnosti selekcije jedinke s oznakom 1, glasi:

$$p(i) = \frac{N(N+1)\cdot p(1) - 2}{N(N-1)} + \frac{2 - 2N \cdot p(1)}{N(N-1)} \cdot i \,. \qquad (4.12)$$

Uvrsti li se za vrijednost vjerojatnosti selekcije jedinke s oznakom 1 (to je najbolja jedinka kod eliminacijske selekcije, a najgora kod generacijske selekcije) $p(1) = 2/[N(N+1)]$ (slika 4.11) u izraz (4.12) dobiva se izraz (4.11). Zatim, dovoljno je u izraz (4.12) uvrstiti $p(1)=0$ kako bi se ugradio elitizam kod genetskog algoritma s eliminacijskom linearno sortirajućom selekcijom bez izmjene izvornog teksta programa. U tom slučaju izraz za vjerojatnost selekcije glasi:

$$p(i) = \frac{2}{N(N-1)} \cdot (i-1). \qquad (4.13)$$

Isti izraz vrijedi i za generacijsku linearnu sortirajuću selekciju gdje je vjerojatnost selekcije najgore jedinke jednaka nuli.

Na sličan način, kao što se nagib pravca može podešavati s vjerojatnošću selekcije $p(1)$ u izrazu (4.12), u izrazu za izračunavanje vjerojatnosti selekcije po Bäcku nagib pravca se podešava uz pomoć parametra η:

$$p(i) = \frac{\eta}{N} - \frac{2(\eta-1)}{N(N-1)} \cdot (i-1), \qquad (4.14)$$

gdje najbolja jedinka nosi oznaku 1.

4.5.3. Selekcije najboljih

Selekcije najboljih odabiru unaprijed zadani broj najboljih jedinki. Moguće su tri vrste selekcije: ($\mu+\lambda$) selekcija, (μ,λ) selekcija i krnja selekcija. Navedene podvrste selekcije najboljih se međusobno razlikuju prema skupu jedinki iz kojeg se odabiru najbolje jedinke. Najbolje jedinke se mogu birati iz skupa samo roditelja, roditelja i djece te samo djece. Prvo se odabiru slučajnim postupkom roditelji, a zatim se njihova djeca (zajedno ili bez roditelja)

sortiraju i preživljavaju samo određeni broj najboljih jedinki. μ je veličina populacije roditelja, a λ je veličina populacije djece[34] [BAE93, BAE95c].

Kod (μ+λ) selekcije slučajno se odabire μ roditelja iz populacije. Njihovim se križanjem stvara λ djece. Iz skupa roditelja i djece se zatim najboljih μ jedinki selektira za sljedeću generaciju. Postupak se ponavlja sve dok se ne popuni nova generacija s N novih jedinki, odnosno $\frac{N}{\mu}$ puta ili se μ jedinki višekratno kopira kako bi se popunila populacija.

Pri (μ,λ) selekciji također se slučajno odabire iz populacije μ roditelja i njihovim križanjem se stvara λ djece. Međutim, djece je više od roditelja ($\lambda \geq \mu$) i μ najbolje djece se selektira za sljedeću generaciju [BAE94, BAE95a]. Vjerojatnost selekcije i-te jedinke za križanje iznosi:

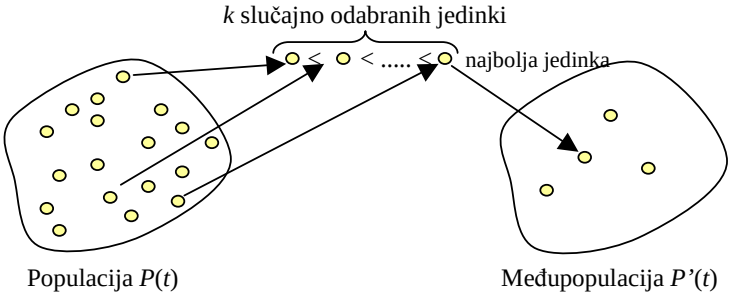
$$p(i) = \begin{cases} 1/\mu & : \quad 1 \leq i \leq \mu \\ 0 & : \quad \mu < i \leq N \end{cases} \qquad (4.15)$$

Obje opisane selekcije kombiniraju rekombinaciju sa slučajnim odabirom jedinki, tj. u istom koraku se obavljaju i selekcija i križanje [CAN99b]. Postupak križanja je pogodan za paraleliziranje, jer N/μ dretvi može paralelno birati roditelje i križati ih. Roditelji se ne mijenjaju kod postupka križanja, nego se samo čitaju njihove vrijednosti. Međutim, nakon što sve dretve obave svoj posao, treba obaviti sortiranje i roditelja i djece kod (μ+λ) selekcije ili samo djece kod (μ,λ) selekcije. To znači da se dretve trebaju nekako sinkronizirati prije no što se obavi sortiranje.

Krnja selekcija[35] odabire n najboljih jedinki i kopira ih N/n puta [CAN99b, MUE94]. Vjerojatnost selekcije i-te jedinke u bazen za parenje iznosi: $p(i)=1$, za $i=1,2,...,n$, dok je za ostale jedinke ta vjerojatnost jednaka nuli.

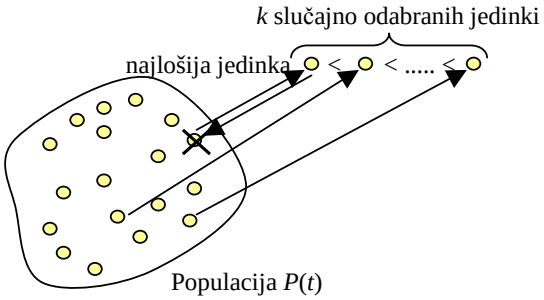
4.5.4. *k*-turnirske selekcije

k-turnirska selekcija ($k=2,3,4,...,N$) s jednakom vjerojatnošću odabire k jedinki između kojih selektira najbolju ili najlošiju jedinku [BAE95a, BLI95, CAN99a, CAN99b, MIL95, WHI96]. Generacijska turnirska selekcija N puta slučajno odabire k jedinki, između njih selektira najbolju jedinku i kopira je u međupopulaciju (slika 4.8).



Slika 4.8. **Generacijska *k*-turnirska selekcija**

Eliminacijska turnirska selekcija M puta odabire k jedinki i eliminira najlošiju među njima (slika 4.9). Parametar k naziva se *veličinom turnira*[36]. Prema veličini turnira, turnirske selekcije se dijele na binarnu turnirsku selekciju (k=2), 3-turnirsku selekciju, 4-turnirsku selekciju, itd.



Slika 4.9. **Eliminacijska *k*-turnirska selekcija**

Za turnirsku selekciju nije potrebno sortirati jedinke, iako vjerojatnost selekcije ovisi o rangu jedinke, kao i kod sortirajućih selekcija. U postupku turnirske selekcije važan je samo međusobni odnos između k slučajno odabranih jedinki. Prema vrijednosti funkcije cilja određuje se za svaki par jedinki koja je od njih bolja, odnosno lošija.

[34] engleski: *parent and offspring population size*
[35] engleski: *truncation selection*
[36] engleski: *tournament size*

Neka najbolja jedinka ima oznaku 1, a najgora N. Najbolja jedinka se ne može nikako eliminirati s bilo kojom eliminacijskom turnirskom selekcijom, jer će k-1 ostalih slučajno odabranih jedinki biti sigurno lošije od najbolje jedinke. Čuvanje najbolje jedinke ili više najboljih jedinki naziva se elitizmom. Elitizam je kod eliminacijskih turnirskih selekcija inherentno ugrađen, tj. vjerojatnost eliminacije najbolje jedinke jednaka je nuli: $p(1)=0$.

Turnirske selekcije su posebno pogodne za paraleliziranje [TOM99], jer se selekcija odvija nad samo manjim dijelom populacije, točnije, nad k jedinki, gdje je $k \leq N$. Štoviše, broj k je obično znatno manji od veličine populacije ($k << N$).

Generacijska k-turnirska selekcija se može izvoditi paralelno na sljedeći način: više dretvi paralelno slučajno odabire k jedinki iz populacije i najbolju jedinku od k odabranih kopira u novu populaciju. Ta će jedinka sudjelovati u reprodukciji. Dretve ponavljaju taj postupak sve dok se ne popuni nova populacija. Eliminacijska k-turnirska selekcija može se paralelizirati na sličan način: više dretvi paralelno slučajno odabire k jedinki iz populacije, ali odabiru najlošiju jedinku i izbrišu je iz populacije. Postupak se ponavlja sve dok se ne izbriše M jedinki. Preživjele jedinke će sudjelovati u reprodukciji, a njihova djeca će nadopuniti populaciju. Reprodukcija i selekcija se mogu odvijati u istom koraku kod eliminacijske k-turnirske selekcije: nakon što se izbriše najlošija jedinka od k slučajno odabranih, nju nadomjesti dijete dobiveno križanjem dviju jedinki od k-1 preživjelih.

Nezavisno jedni od drugih, Baeck [BAE95a] te Miller i Goldberg [MIL95] su odredili selekcijski intenzitet za k-turnirsku selekciju. Vrijednosti selekcijskog intenziteta za k-turnirsku generacijsku selekciju, gdje je k=1,2,3,4,5, prikazani su u tablici [BAE95a, MIL95]:

Tablica 4.1. **Selekcijski intenzitet za *k*-turnirsku generacijsku selekciju**

k	s_I
1	0
2	0.56419
3	0.84628
4	1.02938
5	1.16296

Neka se, primjerice, koristi samo 3-turnirska generacijska selekcija (bez ostalih genetskih operatora). Uz pretpostavku da u generaciji t vrijednosti funkcija dobrote poprimaju normalnu razdiobu $N[\overline{f}(t), \sigma(t)]$, može se predvidjeti prosječna dobrota populacije u slijedećoj generaciji (to je prosječna dobrota preživjelih, odnosno selektiranih jedinki). $\overline{f_p}(t+1)$ prema izrazima (4.1), (4.4) i (4.10) te tablici 4.1 je:

$$\overline{f_p}(t+1) = \overline{f}(t) + S_I \cdot \sigma(t) = \overline{f}(t) + 0.84628 \cdot \sigma(t) .$$

Način na koji će se genetski algoritam podijeliti na podzadatke i kvaliteta dobivenih rješenja uvelike ovisi o vrsti selekcije. U ovom radu predloženi model paralelnih genetskih algoritama temelji se na 3-turnirskoj selekciji. Stoga su k-turnirske selekcije izdvojene, detaljnije opisane i razrađene u poglavlju 5.

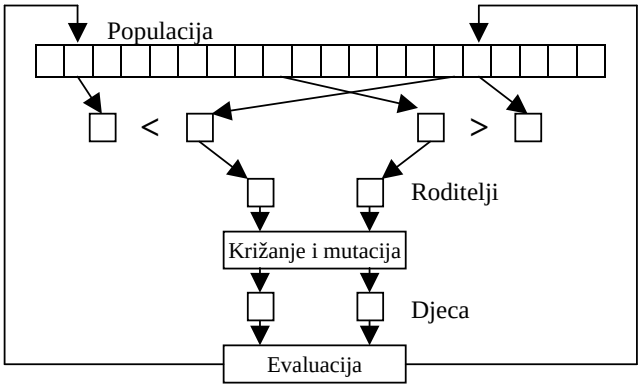
4.5.5. Jednostavna turnirska selekcija

Jednostavna turnirska selekcija[37] je specifičan oblik eliminacijske binarne turnirske selekcije prilagođene paralelnom izvođenju [YOS99]. U jednom koraku slučajnim postupkom se odabiru dva para jedinki. U svakom paru se lošija jedinka selektira za eliminaciju. Križanjem preživjelih jedinki generira se dvoje djece, koja se potom mutiraju i evaluiraju. Taj par novostvorenih jedinki nadomještće eliminirane jedinke. Na taj način genetski algoritam s jednostavnom turnirskom selekcijom u istom koraku obavlja i selekciju i reprodukciju. Slika 4.10 prikazuje pseudokod genetskog algoritma s opisanom turnirskom selekcijom. Na slici 4.11 shematski je prikazan postupak selekcije i reprodukcije u istom koraku.

```
Genetski_algoritam_s_jednostavnom_turnirskom_selekcijom(){
   dok(nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
      selektiraj_slučajno_dvije_jedinke();
      Roditelj_A = bolja_jedinka_od_dvije_selektirane();
      Eliminirana_A = lošija_jedinka_od_dvije_selektirane();
      selektiraj_slučajno_dvije_jedinke_između_preostalih_jedinki();
      Roditelj_B = bolja_jedinka_od_dvije_selektirane();
      Eliminirana_B = lošija_jedinka_od_dvije_selektirane();
      križanjem_roditelja_nadomjesti_eliminirane_jedinke();
      mutiraj_djecu(pm);
      evaluiraj_djecu();
   }
}
```

Slika 4.10. **Genetski algoritam s jednostavnom turnirskom selekcijom**

[37] engleski: *simplified tournament selection*



Slika 4.11. **Slikovni prikaz jednostavne turnirske selekcije**

4.6. Izbor operatora selekcije

Izbor operatora selekcije svodi se na izbor selekcije koja je najpogodnija za paralelno izvođenje. Poželjno je da selekcija, osim što se na jednostavan način može paralelizirati, ima mogućnost podešavanja selekcijskog pritiska uz pomoć nekog parametra. Kako je to već opisano, postupci selekcije se dijele na generacijske i eliminacijske. Generacijske selekcije nisu pogodne za paraleliziranje, jer se reprodukcija mora obavljati odvojeno od selekcije. Reprodukcija može početi tek kada generacijska selekcija formira novu populaciju. Generacijska selekcija troši više spremničkog prostora jer algoritam obavlja posao nad dvije populacije. Povrh toga, generacijskom selekcijom se generiraju duplikati. Duplikate je poželjno nakon reprodukcije eliminirati, što dodatno usporava rad genetskog algoritma. Stoga je eliminacijska selekcija pogodnija za paralelno izvođenje.

Postupci selekcije se dijele i prema tome kako funkcija cilja utječe na vjerojatnost selekcije. Ako funkcija cilja *izravno* utječe na vjerojatnost selekcije, tako da je vjerojatnost selekcije proporcionalna vrijednosti funkcije cilja, radi se o proporcionalnim selekcijama. Između brojnih nedostataka proporcionalnih selekcija navedenih u poglavlju 4.4.3 ističu se: nemogućnost paralelizacije eliminacijske proporcionalne selekcije bez sinkronizacije, nemogućnost jednostavne kontrole selekcijskog pritiska i prevelika ovisnost ponašanja genetskog algoritma o funkciji cilja. Dakle, selekciju pogodnu za paralelno izvođenje treba tražiti među rangirajućim selekcijama. Sortirajuće selekcije su nepogodne za paralelizaciju upravo zbog postupka sortiranja jedinki koji se mora obaviti u svakoj iteraciji. Osim toga, jednostavan eksperiment u kojem se mjerilo vrijeme trajanja eliminacijske proporcionalne i eliminacijske 3-turnirske selekcije dao je sljedeći rezultat: eliminacijska proporcionalna selekcija troši 3.4 puta više procesorskog vremena od eliminacijske 3-turnirske selekcije. Prema tome, eliminacijske turnirske selekcije su najpogodnije za paralelno izvođenje. S tim se slažu i mnogi drugi autori [CAN99a, CAN99b, TOM99, WHI96, YOS99].

Turnirske selekcije obavljaju selekciju samo nad dijelom populacije slično kao i distribuirani genetski algoritam. DGA koristi izolirane subpopulacije nad kojima se paralelno obavljaju svi genetski operatori. Nad pojedinom subpopulacijom obavljaju se genetski operatori sekvencijski. Razlika između GA s turnirskom selekcijom i DGA je samo u stupnju izoliranosti subpopulacija i načinu razmjene podataka. S obzirom da DGA koristi izolirane subpopulacije, upotrebljava se dodatni operator migracije (koji sa sobom nosi cijeli niz novih parametara) kako bi se razmijenio genetski materijal između subpopulacija. Turnirska selekcija u jednoj iteraciji djeluje nad k slučajno odabranih jedinki. U sljedećoj iteraciji, selekcija djeluje nad drugih k slučajno odabranih jedinki. Ta dva skupa jedinki nisu disjunktna, odnosno takve *subpopulacije* se mogu preklapati. Preklapanje jedinki omogućuje razmjenu podataka između subpopulacija. Stoga nema potrebe za dodatnim mehanizmom razmjene genetskog materijala, niti za novim parametrima. Na taj način je otklonjen najveći nedostatak DGA, tj. genetski algoritam s turnirskom selekcijom nema niti jedan novi parametar.

Turnirsku selekciju, gotovo bez ikakvog ograničenja, mogu obavljati više dretvi paralelno. Problem nastaje samo kada istu jedinku selektira više dretvi odjednom. Tada će generacijska turnirska selekcija generirati duplikat, što nije greška, već opisani nedostatak svih generacijskih turnirska selekcija. S druge strane, konzistentnost podataka se može narušiti ako se radi o eliminacijskoj turnirskoj selekciji koja u istom koraku obavlja i selekciju i reprodukciju. Naime, kada dvije dretve selektiraju jednu te istu jedinku za eliminaciju (brisanje), prva, *brža* dretva će selektiranu jedinku nadomjestiti djetetom, a druga, *sporija* dretva će prepisati preko nje svoj rezultat. To znači da je prva dretva obavljala posao te iteracije uzalud. Općenito, ako više dretvi selektira u nekoj iteraciji istu jedinku za eliminaciju, samo će ona posljednja - najsporija dretva korisno obavljati posao te iteracije. Opisana pojava, kada više dretvi asinkrono mijenjaju zajednički podatak, naziva se *utrkom*[38].

Ostaje još pitanje, koja je podvrsta eliminacijske turnirske selekcije najbolja za paralelno izvođenje?

[38] engleski: *data race*

5. ELIMINACIJSKE TURNIRSKE SELEKCIJE

5.1. Vjerojatnost selekcije za turnirske selekcije bez duplikata

5.1.1. Binarna turnirska eliminacijska selekcija bez duplikata

Binarna turnirska selekcija ili 2-*turnirska selekcija* bez duplikata slučajno odabire dvije različite jedinke iz populacije [RUD94]. Pri generacijskoj selekciji bolja jedinka kopirati će se u sljedeću populaciju, a pri eliminacijskoj selekciji lošija jedinka će se iz populacije izbrisati.

Vjerojatnost selekcije ovisi o položaju jedinke u poretku jedinki sortiranih po dobroti. U daljnjem tekstu će se pokazati da vjerojatnost selekcije jedinke s oznakom (indeksom) *i* uporabom 2-turnirske selekcije iznosi:

$$p_2(i) = \frac{\binom{i}{2} - \binom{i-1}{2}}{\binom{N}{2}}, \qquad (5.1)$$

gdje je $\binom{n}{k}$ broj načina na koji možemo iz skupa {1,2,...,n} izabrati *k* različitih elemenata:

$$\binom{n}{k} = \frac{n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{k!} = \frac{\prod_{i=0}^{k-1}(n-i)}{k!}, \qquad (5.2)$$

gdje su *n* i *k* prirodni brojevi, tj. $n,k\in\mathbf{N}$.

Vjerojatnost eliminacije *i*-te jedinke uporabom eliminacijske binarne turnirske selekcije iznosi: $p_{2E}(i)=p_2(i)$ prema izrazu (5.1), gdje je i=1,2,3, ...,N s tim da je $f_1 \geq f_2 \geq f_3 \geq \ldots \geq f_i \geq \ldots \geq f_N$. Oznaka *2E* označava da se radi o 2-turnirskoj eliminacijskoj selekciji. Vjerojatnost selekcije *i*-te jedinke uporabom generacijske binarne turnirske selekcije se računa prema istom izrazu: $p_{2G}(i)=p_2(i)$, ali su jedinke drugačije poredane tako da je $f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_i \leq \ldots \leq f_N$. Dakle, izraz (5.1) vrijedi i za eliminacijsku i za generacijsku selekciju, razlika je samo u načinu označavanja.

Dvije jedinke se iz skupa od *N* jedinki mogu odabrati na $\binom{N}{2}$ načina. Lošija jedinka od dvije nasumice odabrane, imat će veću oznaku. Ako je *i*-ta jedinka odabrana za eliminaciju, znači da su obje jedinke odabrane iz skupa jedinki koje imaju indeks manji ili jednak *i*. Broj kombinacija parova nasumice odabranih jedinki iz skupa {1,2,3,...,*i*} (prvih *i* jedinki) iznosi $\binom{i}{2}$. Vjerojatnost da par slučajno odabranih jedinki bude iz skupa od prvih *i* jedinki iznosi: $\frac{\binom{i}{2}}{\binom{N}{2}}$. Od te vjerojatnosti treba oduzeti vjerojatnost da obje jedinke imaju oznaku manju od *i*: $\frac{\binom{i-1}{2}}{\binom{N}{2}}$, (jer *i*-ta jedinka mora biti selektirana, a ona druga mora biti s manjom oznakom od *i*) i dobiva se izraz (5.1).

Uvrštavanjem izraza (5.2) u izraz (5.1) dobiva se:

$$p_{2E}(i) = \frac{2}{N(N-1)}\cdot(i-1). \qquad (5.3)$$

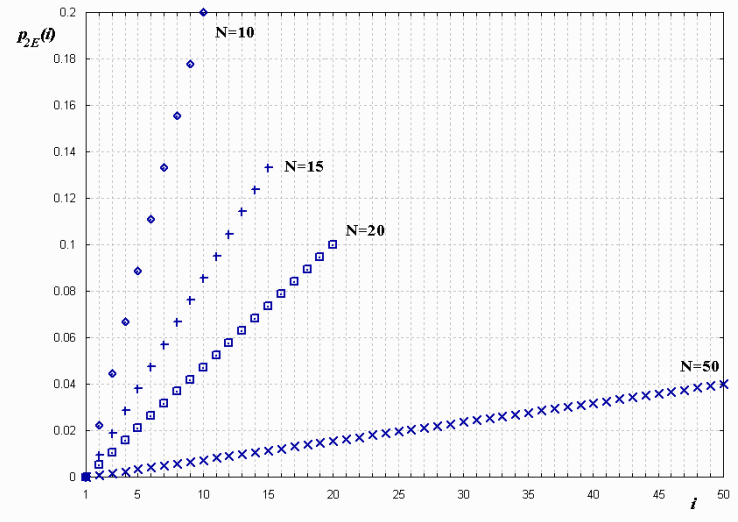
Može se provjeriti da je suma vjerojatnosti eliminacije svih jedinki jednaka 1:

$$\sum_{i=1}^{N} p_{2E}(i) = \sum_{i=1}^{N}\frac{2(i-1)}{N(N-1)} = \frac{2}{N(N-1)}\sum_{i=1}^{N}(i-1) = \frac{2}{N(N-1)}\left(\sum_{i=1}^{N} i - N\right)$$

$$= \frac{2}{N(N-1)}\left(\frac{N(N+1)}{2} - N\right) = \frac{2}{N(N-1)}\cdot\frac{N(N+1)-2N}{2} = 1 \qquad \blacksquare$$

Vjerojatnosti selekcije jedinki nalaze se na pravcu (slika 5.1), što je svojstvo linearno sortirajuće selekcije. Binarna turnirska selekcija je specijalni oblik linearno sortirajuće selekcije za koju je vjerojatnost selekcije jedinke s oznakom 1 jednaka nuli. Stoga su izrazi za izračunavanje vjerojatnosti selekcije za te dvije selekcije jednaki (usporediti izraze (5.3) i (4.13)). To znači da genetski algoritam s binarnom turnirskom selekcijom ima potpuno ista svojstva kao i genetski algoritam s linearno sortirajućom selekcijom s parametrom $p(1)$=0. Te dvije selekcije razlikuju se samo u trajanju: binarna turnirska selekcija kraće traje, jer se ne obavlja sortiranje jedinki. Binarna

turnirska selekcija ima još jednu prednost nad linearno sortirajućom selekcijom: jednostavnija je za ugradnju, tj. izvorni tekst programa je značajno jednostavniji i kraći.



Slika 5.1. **Vjerojatnosti eliminacije za binarnu turnirsku eliminacijsku selekciju za veličine populacije od 10, 15, 20 i 50 jedinki**

Binarna turnirska selekcija je najjednostavnija od svih selekcija. Međutim, u ovom radu predloženi model GPGA koristi 3-turnirsku selekciju, jer se, kako će se to u daljnjem tekstu pokazati, na jednostavan način može obavljati zajedno s reprodukcijom u istom koraku. Stoga, slijedi razmatranje 3-turnirske selekcije na sličan način kao što je to učinjeno s binarnom turnirskom selekcijom.

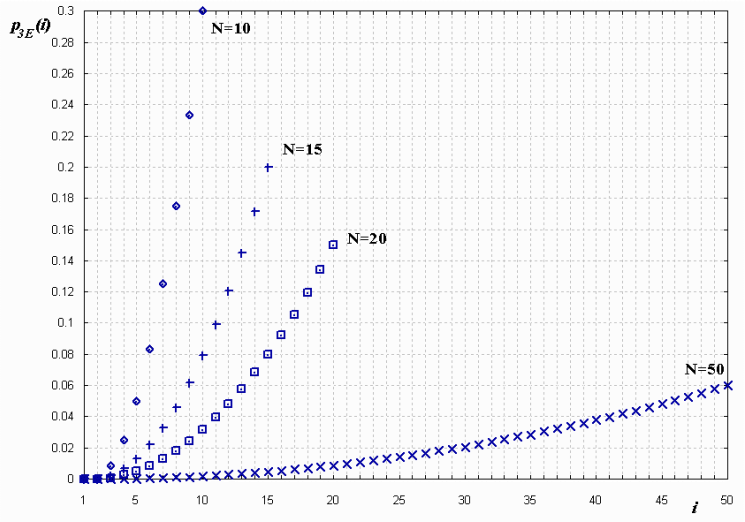
5.1.2. 3-turnirska eliminacijska selekcija bez duplikata

3-turnirska selekcija bez duplikata slučajno odabire tri različite jedinke iz populacije i kopira najbolju jedinku u novu populaciju (generacijska 3-turnirska selekcija) ili eliminira najlošiju jedinku (eliminacijska 3-turnirska selekcija).

Vjerojatnost eliminacije *i*-te jedinke uporabom 3-turnirske selekcije iznosi:

$$p_{3E}(i) = \frac{\binom{i}{3} - \binom{i-1}{3}}{\binom{N}{3}} = \frac{3(i-1)(i-2)}{N(N-1)(N-2)}, \qquad (5.4)$$

gdje je i=1,2,3, ...,N i $f_1 \geq f_2 \geq f_3 \geq \ldots \geq f_i \geq \ldots \geq f_N$.



Slika 5.2. **Vjerojatnosti eliminacije za 3-turnirsku eliminacijsku selekciju za veličine populacije populacije od 10, 15, 20 i 50 jedinki**

Tri jedinke se iz skupa od N jedinki mogu odabrati na $\binom{N}{3}$ načina. Najlošija jedinka, od tri nasumice odabrane, imat će najveću oznaku. Ako je i-ta jedinka odabrana za eliminaciju, znači da su sve tri jedinke odabrane iz skupa jedinki koje imaju indeks manji ili jednak i. Broj kombinacija parova nasumice odabranih jedinki iz skupa od prvih i jedinki iznosi $\binom{i}{3}$. Vjerojatnost da slučajno odabrane jedinke budu iz skupa od prvih i jedinki iznosi: $\frac{\binom{i}{3}}{\binom{N}{3}}$. Od te vjerojatnosti treba oduzeti vjerojatnost da sve tri jedinke imaju oznaku manju od i: $\frac{\binom{i-1}{3}}{\binom{N}{3}}$, (jer i-ta jedinka mora biti selektirana, a one druge dvije moraju imati oznake manje od i) i dobiva se izraz (5.4).

Moguće je provjeriti da je suma vjerojatnosti eliminacije svih jedinki jednaka 1:

$$\sum_{i=1}^{N} p_{3E}(i) = \sum_{i=1}^{N} \frac{3(i-1)(i-2)}{N(N-1)(N-2)} = \frac{3}{N(N-1)(N-2)} \sum_{i=1}^{N} (i^2 - 3i + 2) = \frac{3}{N(N-1)(N-2)} \left(\sum_{i=1}^{N} i^2 - 3\sum_{i=1}^{N} i + 2N \right)$$

Uvrštavanjem izraza za sume konačnih redova potencija u brojnik dobiva se:

$$\sum_{i=1}^{N} p_{3E}(i) = \frac{3}{N(N-1)(N-2)} \left(\frac{N(N+1)(2N+1)}{6} - 3\frac{N(N+1)}{2} + 2N \right) = \frac{3}{N(N-1)(N-2)} \cdot \frac{N(N-1)(N-2)}{3} = 1 \blacksquare$$

5.1.3. Općeniti slučaj *k*-turnirske selekcije bez duplikata

k-turnirska selekcija bez duplikata slučajno odabire k različitih jedinki iz populacije i kopira najbolju jedinku (od k slučajno odabranih) u novu populaciju ili eliminira najlošiju jedinku, ovisno o tome da li se radi o generacijskoj ili eliminacijskoj selekciji.

Na sličan način, kao što se dosad pokazalo za binarnu i 3-turnirsku selekciju, u daljnjem tekstu će se pokazati da vjerojatnost eliminacije i-te jedinke uporabom k-turnirske eliminacijske selekcije iznosi:

$$p_{kE}(i) = \frac{\binom{i}{k} - \binom{i-1}{k}}{\binom{N}{k}}, \qquad (5.5)$$

gdje je i=1,2,3, ...,N i vrijedi nejednakost $f_1 \geq f_2 \geq f_3 \geq \ldots \geq f_i \geq \ldots \geq f_N$.

k jedinki se iz skupa od N jedinki mogu odabrati na $\binom{N}{k}$ načina. Najlošija jedinka, od k nasumice odabranih, imat će najveću oznaku. Ako je i-ta jedinka odabrana za eliminaciju, znači da je svih k jedinki odabrano iz skupa jedinki koje imaju indeks manji ili jednak i. Broj kombinacija parova nasumice odabranih jedinki iz skupa od prvih i jedinki iznosi $\binom{i}{k}$. Vjerojatnost da slučajno odabrane jedinke budu iz skupa od prvih i jedinki iznosi: $\frac{\binom{i}{k}}{\binom{N}{k}}$. Od te vjerojatnosti treba oduzeti vjerojatnost da svih k jedinki imaju oznaku manju od i: $\frac{\binom{i-1}{k}}{\binom{N}{k}}$, (jer i-ta jedinka mora biti selektirana, a one ostale moraju imati oznake manje od i) i dobiva se izraz (5.5).

Na sličan način kao u prethodna dva primjera, može se provjeriti da je suma vjerojatnosti eliminacije svih jedinki uporabom k-turnirske eliminacijske selekcije jednaka 1:

$$\sum_{i=1}^{N} p_{kE}(i) = \sum_{i=1}^{N} \frac{\binom{i}{k} - \binom{i-1}{k}}{\binom{N}{k}} = \frac{1}{\binom{N}{k}} \left[\binom{1}{k} - \binom{0}{k} + \binom{2}{k} - \binom{1}{k} + \binom{3}{k} - \binom{2}{k} + \ldots + \binom{N}{k} - \binom{N-1}{k} \right] = \frac{\binom{N}{k}}{\binom{N}{k}} = 1. \quad \blacksquare$$

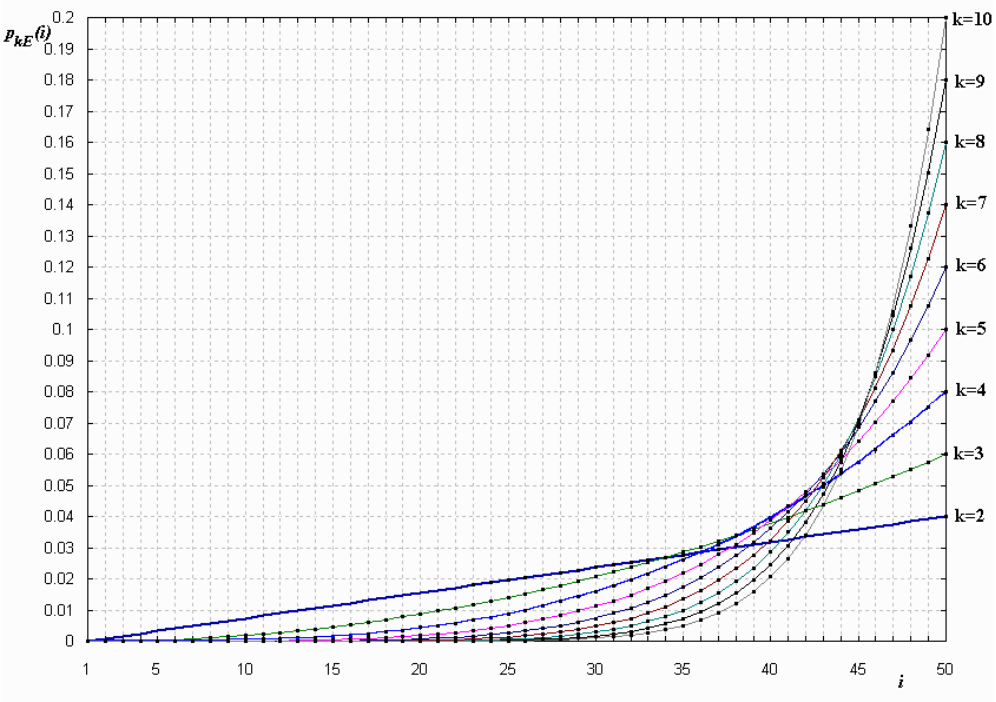
Uvrštavanjem izraza (5.2) u izraz (5.5) dobiva se:

$$p_{kE}(i) = \frac{k(i-1)(i-2)\ldots(i-k+1)}{N(N-1)(N-2)\ldots.(N-k+1)} = \frac{k\prod_{j=1}^{k-1}(i-j)}{\prod_{j=0}^{k-1}(N-j)}$$

I konačno, izraz s pomoću kojeg se na jednostavniji način računa vjerojatnost eliminacije *i*-te jedinke uporabom *k*-turnirske eliminacijske selekcije glasi:

$$p_{kE}(i) = \frac{k}{N} \prod_{j=1}^{k-1} \frac{(i-j)}{(N-j)}. \qquad (5.6)$$

Slika 5.3 prikazuje ovisnost vjerojatnosti selekcije *i*-te jedinke o parametru *k*. Prikazane su vjerojatnosti selekcije za veličinu populacije 50, a parametar *k* poprima vrijednosti od 2 do 10. Iz slike je vidljivo da selekcijski pritisak raste kako se povećava parametar *k*. Međutim, selekcijski pritisak nije jednoliko raspoređen na *bolje* i *slabije* jedinke. Primjerice, neka su *bolje* jedinke one koje nose oznaku manju od 25, a ostale su *slabije*. Za binarnu turnirsku selekciju, vjerojatnost selekcije je točno proporcionalna rangu jedinke pa tako, primjerice jedinka s oznakom 50 ima točno dvostruko veću vjerojatnost selekcije od jedinke s rednim brojem 25. Kod 3-turnirske selekcije to nije slučaj.



Slika 5.3. **Ovisnost vjerojatnosti eliminacije o parametru *k* za *k*-turnirsku selekciju bez duplikata**

Selekcijski pritisak se može na vrlo jednostavan način podešavati kod genetskih algoritama s *k*-turnirskom selekcijom i to uz pomoć parametra *k*. Što je veći *k*, veći je i selekcijski pritisak nad slabijim jedinkama, odnosno, bolje jedinke se više favoriziraju [MIL95]. Iz slike 5.3 se može zaključiti da s vrijednošću *k* ne treba pretjerivati, jer već za 10-turnirsku selekciju 60% populacije (prvih 30 od 50 jedinki) gotovo nikad ne mogu biti selektirane. S porastom vrijednosti parametra *k*, raste i broj jedinki čija je vjerojatnost selekcije jednaka ili približno jednaka nuli. Najboljih *k*-1 jedinki ima vjerojatnost selekcije jednaku nuli: $p_{kE} = 0$ za $i<k$, odnosno $p_{kE} \neq 0$ za $i \geq k$.

Primjerice, pri 10-turnirskoj selekciji 9 najboljih jedinki ima vjerojatnost selekcije jednaku nuli, a slijedećih 20 jedinki približno jednaku nuli. 10-turnirska selekcija će selektirati jedinke s oznakom većom od 30 s vjerojatnošću od 99.7%. Odnosno, vjerojatnost da će biti selektirana jedinka između preostalih 30 iznosi 0.3%. Dakle, selekcijski pritisak između te dvije skupine jedinki je ogroman, dok selekcijskog pritiska između jedinki s oznakom manjom od 31 gotovo da i nema. Ekstremni slučaj je *N*-turnirska selekcija, gdje se selektira samo jedinka s oznakom *N*, dok je vjerojatnost selekcije za ostalih *N*-1 jedinki jednaka nuli. Radi se o selekciji isključivo najbolje jedinke, ako se radi o generacijskoj *N*-turnirskoj selekciji ili najlošije jedinke, ako se radi o eliminacijskoj *N*-turnirskoj selekciji.

5.2. Vjerojatnost selekcije za turnirsku selekciju s duplikatima

5.2.1. Vjerojatnost selekcije prema Bäcku

Odabir *k* jedinki iz populacije se može obaviti na dva načina: sa i bez duplikata. Dosad je opisana turnirska selekcija bez duplikata koja slučajno odabire *k* različitih jedinki iz populacije (slika 5.4). U ovom poglavlju je opisana turnirska selekcija s duplikatima koja slučajno odabire *k* jedinki, koje ne moraju biti nužno različite (slika 5.5). Izvorni tekst programa turnirske selekcije bez duplikata je proširen postupkom detekcije i eliminacije duplikata (masno otisnuti dio izvornog teksta na slici 5.4).

```
int Turnirska_selekcija(int *P){
  int i,j,z,sel[k];
  for(i=0;i<k; ){
    z=1;
    sel[i]=slucajni_broj_između(1,N);
    for(j=0;j<i;j++){
      if(sel[j]==sel[i]) z=0;
    }
    if(z) k++;
  }
  return( najbolji(sel) );
}
```

Slika 5.4. **Turnirska selekcija bez duplikata**

```
int Turnirska_selekcija(int *P){
  int i,sel[k];
  for(i=0;i<k; ){
    sel[i]=slucajni_broj_između(1,N);
  }
  return( najbolji(sel) );
}
```

Slika 5.5. **Turnirska selekcija s duplikatima**

Bäck, Miller, Goldberg, Blickle i Thiele su izračunali vjerojatnost selekcije i selekcijski intenzitet za turnirsku selekciju koja selektira *k* ne nužno različitih jedinki [BAE94, MIL95, BLI95]. Drugim riječima, slučajno se izabire *k* jedinki, s tim da se jedna te ista jedinka može odabrati više puta. Vjerojatnost selekcije, prema Bäcku, za *k*-turnirsku selekciju s duplikatima glasi:

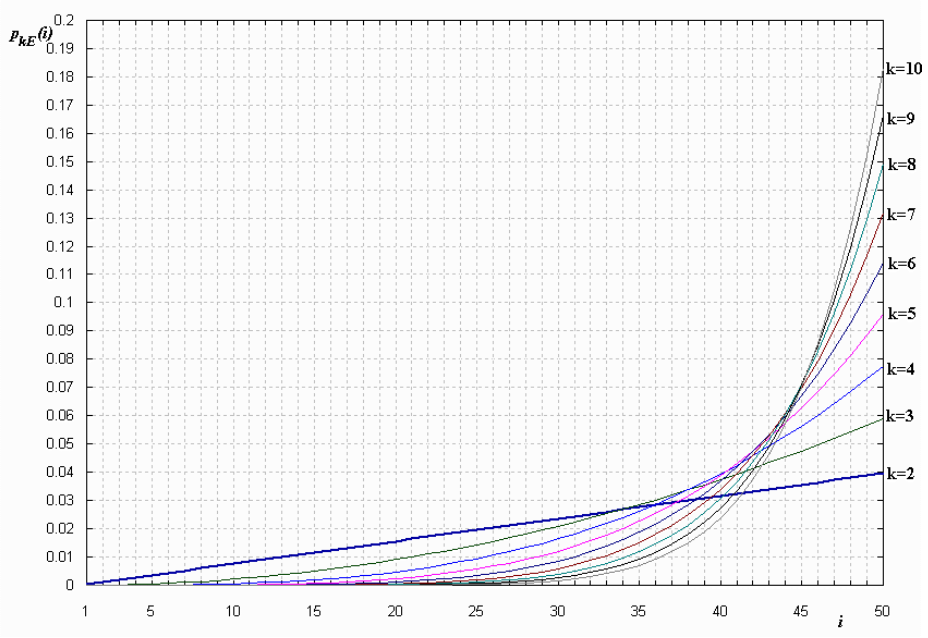
$$p_k(i) = \left(\frac{i}{N}\right)^k - \left(\frac{i-1}{N}\right)^k, \qquad (5.7)$$

gdje su jedinke poredane po dobroti: $f_1 \le f_2 \le f_3 \le \ldots \le f_i \le \ldots \le f_N$.

Da bi *i*-ta jedinka bila selektirana uporabom turnirske selekcije s duplikatima, svih *k* slučajno odabranih jedinki mora imati oznaku manji ili jednaku *i*, te barem jedna od *k* odabranih jedinki mora imati oznaku *i*. Vjerojatnost da nasumice odabrana jedinka ima oznaku manju ili jednaku *i* iznosi i/N. Vjerojatnost da svih *k* jedinki ima oznaku manju ili jednaku *i* iznosi $(i/N)^k$. Ako se od tog broja oduzme vjerojatnost da *sve* jedinke nose oznaku manju ili jednaku od *i*-1 (što znači da je barem jedna od *k* odabranih jedinki ima oznaku *i*), a koja iznosi $[(i\text{-}1)/N]^k$, dobiva se izraz (5.7).

Ukoliko najbolja jedinka ima oznaku *N* (a ne 1), dobiva se izvorni oblik izraza za izračunavanje vjerojatnosti selekcije koju je Bäck koristio u svojim radovima:

$$p_k(i) = \left(\frac{N+1-i}{N}\right)^k - \left(\frac{N-i}{N}\right)^k. \qquad (5.8)$$

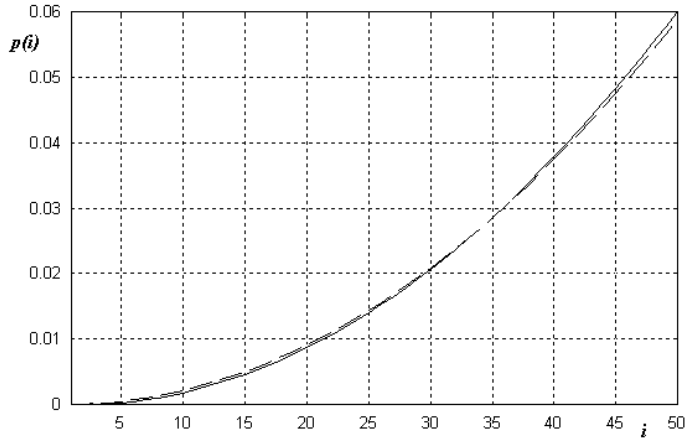


Slika 5.6. **Ovisnost vjerojatnosti eliminacije o parametru *k* za turnirsku selekciju s duplikatima**

U ovom radu je korišten i razrađen model turnirske selekcije bez duplikata. Na prvi pogled, turnirska selekcija bez duplikata je složenija i duže traje (usporediti izvorni tekst programa na slikama 5.4 i 5.5). Osim toga, karakteristične vrijednosti selekcije, kao što su selekcijski intenzitet, reprodukcijska stopa i gubitak raznovrsnosti, dosad su izračunate za turnirsku selekciju s duplikatima [BLI95]. Poznavajući karakteristične vrijednosti za određenu selekciju, može se donekle predvidjeti ponašanje genetskog algoritma (vidi primjer koji se odnosi na 3-turnirsku selekciju na kraju poglavlja 4.5.4). Zatim, ako se koristi eliminacijska turnirska selekcija iza koje slijedi reprodukcija nad preostalim jedinkama, reprodukcija može generirati duplikate. Duplikati samo koče evolucijski proces. Stoga se u literaturi predlaže mehanizam za eliminaciju duplikata [MIC94], koji je znatno složeniji i troši znatno više vremena od dijela programa za provjeru duplikata u turnirskoj selekciji (masno označeni dio koda na slici 5.4). Nadalje, elitizam nije inherentno ugrađen u turnirskoj selekciji s duplikatima, kao što je to slučaj s eliminacijskom turnirskom selekcijom bez duplikata. Vjerojatnost eliminacije najbolje jedinke (to je, u ovom slučaju, jedinka s oznakom 1) je doduše mala i iznosi $p_k(1)=1/N^k$ (prema izrazu (5.7)), ali nije jednaka nuli.

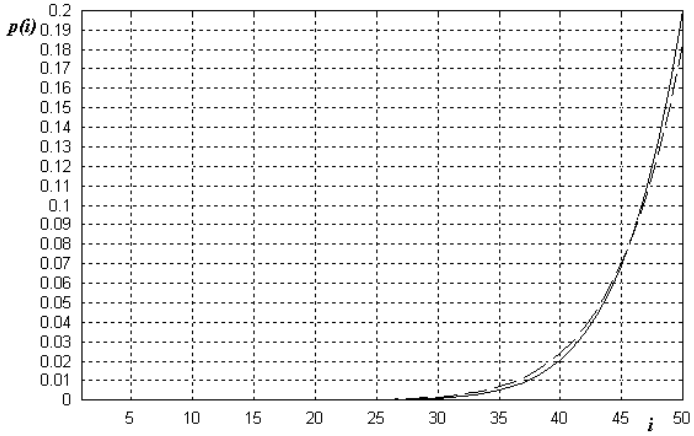
Turnirska selekcija bez duplikata je složenija, ali ne zahtijeva nikakav dodatni mehanizam za eliminaciju duplikata. Veličina skupa zaštićenih jedinki se (bez pisanja dodatnog koda) podešava s pomoću parametra k. Broj jedinki zaštićenih od eliminacije iznosi k-1. Štoviše, s ovakvim inherentno ugrađenim elitizmom čuva se od svake izmjene k-1 jedinki, jer reprodukcija mijenja podatke samo u onim spremničkim lokacijama gdje su se nalazile eliminirane jedinke.

Razdiobe vjerojatnosti za turnirsku selekciju sa i bez duplikata su vrlo slične (usporediti slike 5.3 i 5.6). Na slijedećim slikama se još bolje vide razlike, odnosno sličnosti. Za veličinu populacije N=50 i za $k\leq 10$, vjerojatnost selekcije se razlikuje maksimalno 1.71% i to za jedinku s najvećom oznakom (i=50) i za k=10. Odstupanja su manja, što je parametar k manji.



Slika 5.7. **Razdioba vjerojatnosti za 3-turnirsku selekciju s duplikatima (crtkano) i bez duplikata (puna crta), N=50**

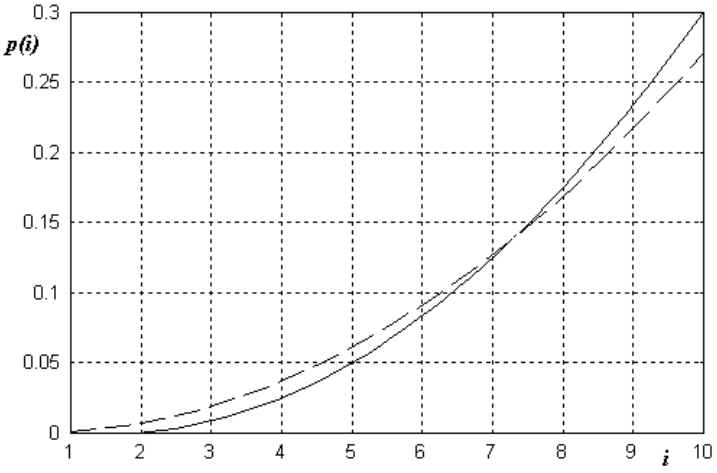
Najveće odstupanje iznosi 0.11% za i=50.



Slika 5.8. **Razdioba vjerojatnosti za 10-turnirsku selekciju s duplikatima (crtkano) i bez duplikata (puna crta), N=50**

Najveće odstupanje iznosi 1.71% za i=50.

Za manje populacije odstupanja su samo malo veća. Tako, primjerice, za 3-turnirsku selekciju, koja djeluje nad populacijom veličine N=10, najveće odstupanje je 2.9% (slika 5.9). Za veće vrijednosti parametra k odstupanje bi bilo još veće, ali ionako za tako male veličine populacije veći k ima za posljedicu djelovanje selekcije samo nad manjim dijelom populacije. To se dobro vidi na slici 5.8, gdje selekcija praktično djeluje samo na jedinke s oznakom većom od 30, tj. djeluje samo nad 40% populacije, što svakako nije dobro svojstvo selekcije. Selekcija treba djelovati nad svim jedinkama osim nad najboljom ili nad malom skupinom najboljih jedinki koje su zaštićene elitizmom.



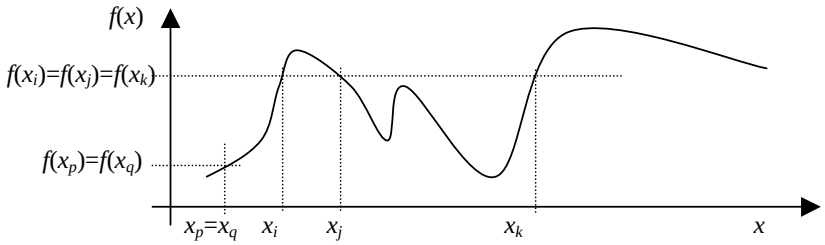
Slika 5.9. **Razdioba vjerojatnosti za 3-turnirsku selekciju s duplikatima (crtkano) i bez duplikata (puna crta), N=10**

Najveće odstupanje iznosi 2.9% za i=10.

Ostaje još problem što dosad nisu izračunate karakteristične vrijednosti (trajanje preuzimanja, selekcijski intenzitet, itd.) za turnirsku selekciju bez duplikata. Međutim, razdiobe vjerojatnosti za turnirske selekcije sa i bez duplikata su vrlo slične, pa je za očekivati da se karakteristične vrijednosti neće mnogo razlikovati.

5.2.2. Vjerojatnost selekcije prema Blickleu

Više jedinki mogu imati istu vrijednost funkcije dobrote. Isti kromosomi (duplikati) imaju iste vrijednosti funkcije cilja, pa i dobrote. Jednako tako i različiti kromosomi mogu imati iste vrijednosti funkcija cilja (slika 5.10). Dosad su jedinke s istom vrijednošću funkcije dobrote imale različite vrijednosti vjerojatnosti selekcije, jer je jedinka J_1 bila bolja od jedinke J_2, ako je bilo zadovoljena nejednakost: $f(x_1) \geq f(x_2)$. Blickle inzistira na tome da jedinke s istom vrijednošću funkcije dobrote imaju i jednaku vjerojatnost selekcije.



Slika 5.10. **Primjer funkcije cilja s više lokalnih optimuma**

Definicija 2 (razdioba dobrote[39])

Funkcija q:$\mathbf{R}\rightarrow\mathbf{Z_0}^+$ svakoj vrijednosti funkcije dobrote $d\in\mathbf{R}$ pridružuje broj jedinki iz populacije $P\in\mathbf{J}^\mathbf{N}$ koje imaju tu vrijednost funkcije dobrote. q se naziva razdiobom dobrote populacije P.

Definicija 3 (razdioba kumulativne dobrote[40])

Neka je n broj jedinstvenih vrijednosti dobrote i neka je $d_1<d_2<\ ...\ <d_n$ ($n\leq N$) poredak vrijednosti dobrote. $Q(d_i)$ označava broj jedinki koje imaju vrijednost dobrote jednaku ili manju od d_i, i naziva se razdiobom kumulativne dobrote:

$$Q(d_i)=\begin{cases} 0 & : \quad i<1 \\ \sum_{j=1}^{i} s(d_j) & : \quad 1\leq i\leq n \\ N & : \quad i>n \end{cases} \qquad (5.9)$$

Vjerojatnost selekcije, prema Blickleu, za k-turnirsku selekciju iznosi:

$$p_k(i)=\left(\frac{Q(d_i)}{N}\right)^k-\left(\frac{Q(d_{i-1})}{N}\right)^k, \qquad (5.10)$$

gdje je i=1,2,...,n ($n\leq N$). Dokaz je dan u [BLI95]. $p_k(i)$ ne označava vjerojatnost selekcije jedinke koja nosi oznaku i, već vjerojatnost selekcije jedinke čija vrijednost dobrote je d_i. Ako su sve vrijednosti dobrota populacija različite, tada je $q(d_i)$=1 za svaki $i\in[1,N]$, a $Q(d_i)=i$. U tom slučaju izraz (5.10) poprima isti oblik kao i izraz (5.7).

[39] engleski: *fitness distribution*
[40] engleski: *cumulative fitnes distribution*

Najčešće jedinke imaju istu vrijednost dobrote jedino ako imaju potpuno iste gene (to su jedinke *duplikati*). Stoga je za izračunavanje vjerojatnosti selekcije dovoljan izraz (5.7), ako genetski algoritam ima ugrađeni neki od mehanizama za eliminaciju duplikata.

5.3. Elitizam i *k*-turnirske eliminacijske selekcije

Elitizam je čuvanje najbolje jedinke ili nekoliko najboljih jedinki od bilo kakve izmjene ili eliminacije tijekom evolucijskog procesa. Elitizam je inherentno ugrađen kod genetskog algoritma s nekom od eliminacijskih turnirskih selekcija bez duplikata, jer je vjerojatnost eliminacije najbolje jedinke jednaka nuli. Binarna eliminacijska turnirska selekcija čuva samo najbolju jedinku: prema izrazu (5.3) $p_{2E}(1)=0$. S obzirom da se najbolja jedinka ne može nikako izbrisati djelovanjem eliminacijske binarne turnirske selekcije, nije potrebno dodavati nikakav dodatni programski kod za čuvanje najbolje jedinke od eliminacije. 3-turnirska eliminacijska selekcija čuva dvije najbolje jedinke, jer je $p_{3E}(1)= p_{3E}(2)=0$ prema izrazu (5.4). Općenito, *k*-turnirska eliminacijska selekcija bez duplikata štiti *k*-1 najboljih jedinki od eliminacije, jer upravo toliko nul točaka ima polinom u izrazu (5.6).

6. NOVI MODEL GLOBALNOG PARALELNOG GENETSKOG ALGORITAMA

6.1. Zašto GPGA?

Zahtjevi koje bi trebao zadovoljavati *idealni* paralelni genetski algoritam su:

- Paralelni genetski algoritam treba biti približno onoliko puta brži od sekvencijskog genetskog algoritma koliko računalo ima procesora. U idealnom slučaju, paralelno se obavljaju svi genetski operatori i PGA se izvodi onoliko puta brže koliko računalo ima procesora, odnosno kolika je propusnost računala.
- Broj parametara treba biti što manji kako bi se što lakše obavilo podešavanje algoritma. Idealno, PGA nema dodatnih parametara, tj. ima samo minimalan broj onih parametara koje mora imati odgovarajući sekvencijski genetski algoritam.
- Rješenje koje se dobije optimiranjem uporabom PGA treba biti iste kvalitete kao i rješenje dobiveno sekvencijskim algoritmom. Idealno, PGA ima potpuno ista svojstva kao odgovarajući sekvencijski genetski algoritam.

Paralelni genetski algoritam se izvodi na dostupnim paralelnim računalima, pa u ovom slučaju dodatni zahtjev glasi:

- Model paralelnog genetskog algoritma treba biti pogodan za ugradnju na višeprocesorsko računalo (s proizvoljnim brojem procesora) sa zajedničkim radnim spremnikom.

Izbor modela PGA ovisi o arhitekturi raspoloživih paralelnih računala te o topologiji mreže računala. Osnovni modeli paralelnih genetskih algoritama su: distribuirani, masovno paralelni i globalni. Distribuirani genetski algoritam je prilagođen za izvođenje na umreženim računalima. DGA je pogodan za izvođenje na svim paralelnim arhitekturama, što je što je ujedno i glavna prednost tog modela. Nasuprot fleksibilnosti algoritma, stoji problem podešavanja novih parametara (kojih ima čak sedam). Dakle, DGA ne zadovoljava drugi i treći zahtjev koje bi trebao zadovoljavati idealni paralelni genetski algoritam. Masovno paralelni genetski algoritam se može implementirati isključivo na računala s velikim brojem procesora (50 i više), pa ne zadovoljava četvrti zahtjev. MPGA ne zadovoljava ni drugi, a ni treći zahtjev, jer, kao i DGA, ima dodatne parametre: veličinu i topologiju susjeda. Tradicionalni GPGA je također pogodan za sve paralelne arhitekture i ima ista svojstva kao i sekvencijski GA (osim što je brži), ali paralelno se odvija samo evaluacija. Paralelizirana je samo evaluacija, jer se pretpostavlja da ona zahtijeva značajno više procesorskog vremena od ostalih genetskih operatora. Tradicionalni GPGA ne zadovoljava prvi zahtjev, tj. nije niti približno N_p puta brži, ako evaluacija ne traje značajno duže od svih ostalih genetskih operacija koje se ne obavljaju paralelno.

Prošireni modeli su: hijerarhijski i hibridni PGA. Hijerarhijski PGA je kombinacija dva ili sva tri osnovna modela. Daje najbolje rezultate, ali je sklopovski najzahtjevniji: zahtijeva više umreženih paralelnih računala. Hibridni PGA kombinira GA s nekim drugim algoritmom optimizacije. Posljednji model PGA je tzv. trivijalni PGA, koji služi isključivo za statističku analizu genetskih algoritama kako bi se, primjerice, podesili parametri genetskog algoritma. Prošireni modeli paralelnih genetskih algoritama (hijerarhijski i hibridni) koriste neki od osnovnih modela, pa će se ti modeli moći izvoditi na određenim paralelnim računalima, ukoliko je neki od osnovnih modela pogodan za izvođenje na njima. Stoga je dovoljno razmotriti mogućnosti paraleliziranja osnovnih modela.

Za svaki od navedenih modela paralelnih genetskih algoritama je poznato kakva su računala potrebna za njegovo izvođenje. S druge strane, ako su na raspolaganju neka specifična računala na kojima se treba riješiti zadani optimizacijski problem uporabom genetskih algoritama, nije jednostavno odrediti najpogodniji model PGA. Štoviše, najčešće treba napraviti neke kompromise i prilagoditi neki od paralelnih modela toj specifičnoj arhitekturi računala i specifičnom problemu. Primjer prilagodbe genetskog algoritma određenoj arhitekturi računala je masovno paralelni genetski algoritam koji je prilagođen izvođenju na računalima s mnogo procesora i s raspodijeljenim radnim spremnikom.

Koji je od paralelnih modela GA najpogodniji za izvođenje na računalima s više procesora (točnije, ne mnogo, već nekoliko procesora: 2, 4 ili 8) i sa zajedničkim radnim spremnikom? Na raspolaganju je operacijski sustav koji podržava višedretvenost, jer uz pomoć višedretvenosti se na vrlo jednostavan način mogu izvoditi paralelni programi koji koriste zajednički radni spremnik [PHA98].

Masovno paralelni genetski algoritam nije pogodan za izvođenje na navedenom paralelnom računalu, jer zahtijeva znatno više procesora od, primjerice, dva. Distribuirani genetski algoritam je pogodan za sve paralelne arhitekture pa i ovu s nekoliko procesora i zajedničkim radnim spremnikom. No, na dvoprocesorskom računalu je moguće ostvariti DGA sa samo dvije subpopulacije, što je obično premalo. U većini primjena DGA koristi se više od četiri subpopulacije [CAN99d, EBY97, GOO97]. Zatim, zajednički radni spremnik se može iskoristiti samo kao komunikacijski kanal za razmjenu jedinki. Zadnji, ali najvažniji nedostatak je dugotrajno podešavanje DGA prije same primjene. Naime, da bi se poboljšala svojstva distribuiranog genetskog algoritma trebaju se podesiti novi parametri: broj subpopulacija, veličina subpopulacija, migracijski interval, migracijska stopa, strategija odabira boljih jedinki, strategija odabira jedinki za eliminaciju i topologija razmjene jedinki. Naravno, zajedno s podešavanjem novih parametara treba podesiti i standardne parametre genetskog algoritma: vjerojatnost mutacije, vjerojatnost križanja i broj iteracija. Podešavanje parametara je vremenski vrlo zahtjevan posao (pogotovo ako ih ima toliko mnogo), jer se obavlja eksperimentalno.

Najbliži zadovoljenju svim postavljenim zahtjevima je globalni paralelni genetski algoritam. GPGA zadovoljava tri od četiri postavljena zahtjeva. Jedino prvi zahtjev nije zadovoljen, jer se paralelizira samo evaluacija, a ne svi genetski operatori. Novi model GPGA bi trebao zadržati svojstva tradicionalnog GPGA, s tim da ima i nova svojstva: da se svi genetski operatori izvode paralelno i da je *ubrzanje* otprilike jednako broju procesora. Naravno, pretpostavlja se da je propusnost računala idealna, tj. da je jednaka broju procesora.

6.2. Opis novog modela globalnog paralelnog genetskog algoritma

Predloženi novi model PGA spada u skupinu modela GPGA, jer koristi zajedničku populaciju, mada je ona u pojedinim trenucima raspodijeljena na subpopulacije. Subpopulacije se dinamički mijenjaju. U svakoj iteraciji dretva slučajnim postupkom odabire *k* jedinki koje čine subpopulaciju. Selekcija i reprodukcija se moraju obavljati sekvencijski nad jednom subpopulacijom jedinki, a paralelizacija se ostvaruje slično kao i kod DGA – tako da više dretvi obavlja cijeli GA. Dobro svojstvo GPGA je zajednička populacija. Populacija bi u novom modelu trebala biti istodobno i zajednička za sve dretve, ali i podijeljena na manje dijelove. Dakle, osnovna razlika između DGA i predloženog modela GPGA je u tome što DGA ima podijeljenu populaciju stalno, tijekom cijelog evolucijskog procesa, a novi model dinamički mijenja jedinke koje čine subpopulacije i to u svakoj iteraciji.

```
GA_s_3-turnirskom_eliminacijskom_selekcijom_bez_duplikata(){
  inicijaliziraj_populaciju(P);
  sve dok(nije_zadovoljen_uvjet_završetka_evolucijskog procesa){
    // odaberi slučajno tri različite jedinke iz populacije
    roditelj1 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    radi{
      roditelj2 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(roditelj1==roditelj2);
    radi{
      dijete = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(dijete==roditelj1 || dijete==roditelj2);
    // dijete treba biti najlošija jedinka od tri slučajno odabrane
    ako(f(dijete)>f(roditelj1)) zamijeni(dijete,roditelj1);
    ako(f(dijete)>f(roditelj2)) zamijeni(dijete,roditelj2);
    // reprodukcija
    dijete = križaj(roditelj1,roditelj2);
    mutiraj(dijete);
    evaluiraj(dijete);
  }
}
```

Slika 6.1. **Sekvencijski GA s 3-turnirskom eliminacijskom selekcijom bez duplikata**

Eliminacijska turnirska selekcija omogućuje obavljanje i selekcije i reprodukcije nad jednom jedinkom u istom koraku. Genetski algoritam pogodan za paralelizaciju (za izvođenje na višeprocesorskom računalu sa zajedničkom memorjiom) glasi ovako: u svakoj iteraciji genetski algoritam slučajno odabire *k* jedinki između kojih bira najlošiju. Ta najlošija jedinka biva zamijenjena s djecom preživjelih *k*-1 roditelja. Križanje je binarni operator (dva roditelja sudjeluju u križanju) i u idealnom slučaju *k*-1 mora biti jednako 2. Roditelji trebaju biti različiti jer će u suprotnom križanje generirati duplikat. Iz toga proizlazi da je najpogodnija 3-turnirska eliminacijska selekcija bez duplikata za novi model PGA.

```
void POPULACIJA::TurnirGA(void){
  int mama,tata,dijete,pom;

  // ODABERI TROJICU, ali pazi da ne odabereš iste
  mama = sluc(VEL_POP);
  do tata = sluc(VEL_POP); while(tata==mama);
  do dijete = sluc(VEL_POP); while( (dijete == mama) || (dijete == tata) );

  // KOJI JE OD ODABRANIH NAJSLABIJI?
  if( a[dijete]->fx > a[tata]->fx){ pom=dijete; dijete=tata; tata=pom; }
  if( a[dijete]->fx > a[mama]->fx){ pom=dijete; dijete=mama; mama=pom; }

  // REPRODUKCIJA = KRIŽANJE + MUTACIJA
  *a[dijete] = *a[mama] + *a[tata];  // KRIŽANJE
  *a[dijete] <= PM;                  // MUTACIJA

  // EVALUACIJA
  a[dijete]->evaluiraj();
}
```

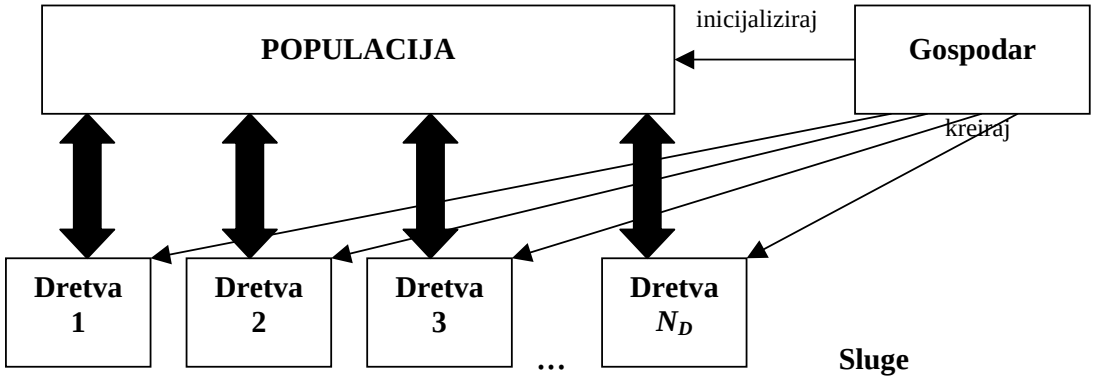
Slika 6.2. **Dio izvornog teksta programa u kojem se obavlja genetski algoritam s 3-turnirskom eliminacijskom selekcijom bez duplikata (posao slugu)**

Slika 6.1 prikazuje jezgru genetskog algoritma s 3-turnirskom eliminacijskom selekcijom bez duplikata. Taj dio posla obavljaju dretve sluge. Na slici 6.2 je prikazana implementacija jezgre genetskog algoritma u programskom jeziku C++. Kao što je vidljivo iz slike, genetski algoritam s 3-turnirskom eliminacijskom selekcijom je vrlo jednostavan za implementaciju: izvorni tekst programa je vrlo kratak i pregledan.

Opisani genetski algoritam s 3-turnirskom eliminacijskom selekcijom je pogodan za paralelizaciju i to svih genetskih operacija: selekcije, križanja, mutacije i evaluacije. Opisani novi model GPGA objedinjuje dobra svojstva DGA i tradicionalnog GPGA. Paraleliziranje genetskih operacija ostvaruje se na sličan način kao i kod DGA: podjelom populacije na subpopulacije. Međutim, razlika je u tome što je suma jedinki svih subpopulacija različita od N (a ne jednaka N kao što je to slučaj kod DGA). Subpopulacije se mogu preklapati. Veličina subpopulacije jednaka je k, a broj dretvi koje će obavljati genetski algoritam s 3-turnirskom selekcijom bez duplikata je D. Broj jedinki N_R koje sudjeluje u selekciji i reprodukciji u trenutku t je manji ili jednak veličini populacije i manji ili jednak produktu broja dretvi i veličine turnira, jer se subpopulacije mogu preklapati:

$$N_R \leq \min\{N, k \cdot D\}. \qquad (6.1)$$

Predloženi model paralelnog genetskog algoritma je najsličniji tradicionalnom globalnom genetskom algoritmu. Novi model GPGA se razlikuje od tradicionalnog GPGA u poslu koji gospodar i sluge obavljaju. U tradicionalnom GPGA sluge obavljaju samo evaluaciju, a gospodar sve ostale genetske operatore. Onaj dio posla koji obavljaju sluge se obavlja paralelno. S obzirom da novi model GPGA treba obavljati *sve* genetske operatore paralelno, gotovo sav posao obavljaju sluge. Gospodar samo inicijalizira populaciju i paralelni genetski algoritam. Preciznije rečeno, gospodar generira početnu populaciju i pokreće dretve sluge. Na kraju evolucijskog procesa, kada sve dretve obave svoj posao, gospodar ispisuje rješenje.



Slika 6.3. **Novi model globalnog paralelnog genetskog algoritma**

```
void *Dretva(void *x){
  for(rbr_gen=0;rbr_gen<MAX_GEN;rbr_gen++) p->TurnirGA();
}

void GPGA(void){
  p = new POPULACIJA(N);
  for(int i=0;i<BR_DRETVI;i++)
    if(thr_create(NULL,0,Dretva,NULL,THR_BOUND,NULL)){
      cout << "Ne mogu kreirati dretvu" << "\n";
      exit(1);
    }
  while( thr_join(0,NULL,NULL) == 0 );
  a[najbolji()]->ispisi();
}
```

Slika 6.4. **Dio izvornog teksta programa u kojem se obavlja inicijalizacija PGA (posao gospodara)**

Dretve su sve međusobno jednake, tj. obavljaju isti posao. Jezgra genetskog algoritma se paralelizira pokretanjem više dretvi koje obavljaju 3-turnirsku eliminacijsku selekciju bez duplikata i reprodukciju.

```
Dretva_s_3-turnirskom_eliminacijskom_selekcijom_bez_duplikata(){
  // posao koji obavljaju SLUGE
  sve dok(nije_zadovoljen_uvjet_završetka_evolucijskog procesa){
    // odaberi slučajno tri različite jedinke iz populacije
    roditelj1 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    radi{
      roditelj2 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(roditelj1==roditelj2);
    radi{
      dijete = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(dijete==roditelj1 || dijete==roditelj2);
    // dijete treba biti najlošija jedinka od tri slučajno odabrane
    ako(f(dijete)>f(roditelj1)) zamijeni(dijete,roditelj1);
    ako(f(dijete)>f(roditelj2)) zamijeni(dijete,roditelj2);
    // reprodukcija
    dijete = križaj(roditelj1,roditelj2);
    mutiraj(dijete);
    evaluiraj(dijete);
  }
}

GPGA(){
  // posao koji obavlja GOSPODAR
  inicijaliziraj_populaciju(P);
  za i=1 do broj_procesora radi{
    stvori_novu_dretvu();
  }
  ispisi_rjesenje();
}
```

Slika 6.5. **GPGA s 3-turnirskom eliminacijskom selekcijom bez duplikata**

6.3. Asinkroni model globalnog paralelnog genetskog algoritma

6.3.1. Problem nejednoznačnosti podataka

Opisani model globalnog paralelnog genetskog algoritma omogućuje obavljanje cijelog genetskog algoritma paralelno. Više dretvi odjednom pristupa zajedničkim podacima. Veličina turnira k određuje nad koliko jedinki svaka od D dretvi obavlja genetske operatore, odnosno nad kojim dijelom zajedničkih podataka pojedina dretva obavlja genetske operatore. Opisani genetski algoritam obavlja 3-turnirsku eliminacijsku selekciju bez duplikata i reprodukciju nad tri jedinke. Ukoliko se pri odabiru jedinki za selekciju i reprodukciju ne vodi računa da li neka druga dretva istodobno pristupa istim podacima može se čitanjem i pisanjem po zajedničkim spremničkim lokacijama, gdje se nalaze odabrane jedinke, narušiti jednoznačnost podataka. Od tri slučajno izabrane jedinke samo se jedna mijenja. Ta jedinka je najslabija od tri odabrane. Samo jedna dretva, od više njih koje su selektirale istu jedinku za eliminaciju, će korisno obavljati posao (vidjet će se efekti njezinog djelovanja). Ostale jedinke beskorisno obavljaju tu iteraciju. Ona dretva koja zadnja upiše vrijednost u spremničku lokaciju najslabije jedinke, prepisat će tu vrijednost preko one vrijednosti koju je upisala dretva prije nje. Posljedica višestruke selekcije za eliminaciju jedne te iste jedinke je smanjenje broja iteracija. Ako je poznata vjerojatnost da dretva obavlja posao uzalud, može se izračunati koliko dodatnih iteracija mora obaviti paralelni genetski algoritam, a da broj *korisnih* iteracija bude jednak broju iteracija koje obavlja odgovarajući sekvencijski algoritam[41]. Tada bi asinkroni GPGA imao ista svojstva kao i odgovarajući sekvencijski genetski algoritam.

6.3.2. Vjerojatnost da dretva posao obavlja uzalud

Više dretvi zbog nesinkroniziranog pristupa zajedničkim podacima može selektirati i mijenjati istu jedinku. Zbog toga će dretve, na već opisani način, neke iteracije obavljati uzalud. Model GPGA predložen u ovom radu koristi 3-turnirsku selekciju. Stoga će se prvo za tu selekciju pokazati kolika je vjerojatnost da jedna od dvije dretve obavlja posao uzalud. Nakon toga slijedi poopćenje tog izraza za proizvoljnu veličinu turnira k i za proizvoljan broj dretvi.

Vjerojatnost selekcije i-te jedinke uporabom 3-turnirske eliminacijske selekcije $p_{3E}(i)$ izračunava se prema izrazu (5.4). Neka se GPGA obavlja na dvoprocesorskom računalu (N_P=2) i neka je broj dretvi jednak broju procesora D=N_P=2. Vjerojatnost da će i-tu jedinku selektirati za eliminaciju i jedna i druga dretva iznosi $p_{3E}(i)^2$. U tom će slučaju jedna od te dvije dretve posao obavljati uzalud. Dakle, $p_{3E}(i)^2$ je vjerojatnost da će jedna od dvije dretve obavljati posao uzalud, ako obje selektiraju baš i-tu jedinku.

[41] *Odgovarajući* sekvencijski GA je u ovom slučaju opisani model GPGA sa samo jednom dretvom.

Općenito, vjerojatnost da će jedna od dvije dretve obavljati posao uzalud, tj. vjerojatnost da će obje dretve selektirati isti jedinku, jednaka je sumi vjerojatnosti dvostruke selekcije svake pojedine jedinke:

$$P'_{3,2}(N) = \sum_{i=1}^{N} p_{3E}(i)^2 \,. \qquad (6.2)$$

Prema tome, vjerojatnost da dretva obavlja posao uzalud je funkcija veličine turnira k, broja dretvi D i veličine populacije N: $P=f(k,D,N)$.

Općenito, oznaka $P'_{k,j}$ predstavlja vjerojatnost da će dretva koja obavlja posao k-turnirske eliminacijske selekcije i nosi oznaku $j \le D$ obavljati posao uzalud. Oznaka $P_{k,D}$ označava vjerojatnost da će *neka* od D dretvi u nekom trenutku obavljati posao uzalud. Jedna od D dretvi neće sigurno obavljati posao uzalud, odnosno vjerojatnost beskorisne selekcije s jednom dretvom je 0, tj. $P_{k,1}=0$, ili još preciznije $P_{k,1}=P'_{k,1}=0$. Vjerojatnost da *jedna* od *dvije* dretve posao obavlja uzalud računa se prema izrazu (6.2). Obje dretve će s jednakom vjerojatnošću obavljati posao uzalud, pa je:

$$P_{3,2} = \frac{P'_{3,2}}{2} \,. \qquad (6.3)$$

Vjerojatnost da jedna od dvije dretve (j=2) uporabom 3-turnirske eliminacijske selekcije (k=3) na dvoprocesorskom računalu (D=2) obavlja posao uzalud dobiva se uvrštavanjem izraza (5.4) u izraz (6.2):

$$P'_{3,2}(N) = \sum_{i=1}^{N} p_{3E}(i)^2 = \sum_{i=1}^{N} \left[\frac{\binom{i}{3} - \binom{i-1}{3}}{\binom{N}{3}} \right]^2 = \sum_{i=1}^{N} \left[\frac{3(i-1)(i-2)}{N(N-1)(N-2)} \right]^2$$

$$= 9 \frac{\sum_{i=1}^{N} [(i-1)(i-2)]^2}{[N(N-1)(N-2)]^2} = 9 \frac{\sum_{i=1}^{N} \left(i^4 - 6i^3 + 13i^2 - 12i + 4\right)}{[N(N-1)(N-2)]^2} \,.$$

Uvrštavanjem izraza za sume konačnih redova potencija u brojnik dobiva se:

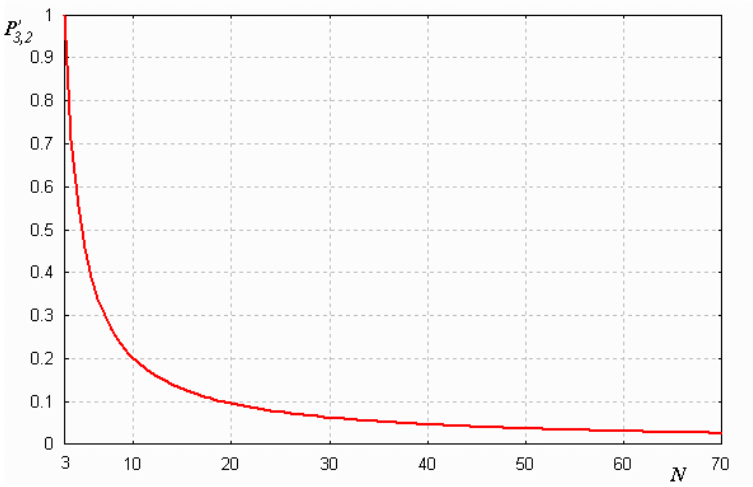
$$\sum_{i=1}^{N} \left(i^4 - 6i^3 + 13i^2 - 12i + 4\right) = \frac{N(N+1)(2N+1)(3N^2+3N-1)}{30} - \frac{3N^2(N+1)^2}{2} +$$

$$+ \frac{13N(N+1)(2N+1)}{6} - 6N(N+1) + 4N = \frac{N^5}{5} - N^4 + \frac{5}{3}N^3 - N^2 + \frac{2N}{15} \,.$$

Konačno, uvrsti li se dobiveno u izraz za $P'_{3,2}$ dobiva se izraz za vjerojatnost da će obje dretve, koje obavljaju 3-turnirsku eliminacijsku selekciju, selektirati jednu te istu jedinku:

$$P'_{3,2}(N) = \frac{9N^4 - 45N^3 + 75N^2 - 45N + 6}{5N(N-1)^2(N-2)^2} \,. \qquad (6.4)$$

Za veličinu populacije 3, vjerojatnost dvostruke selekcije uporabom 3-turnirske eliminacijske selekcije jednaka je 1, jer se dvije najbolje jedinke ne mogu selektirati za eliminaciju i ostaje samo jedna (ona najlošija jedinka) koju će uvijek obje dretve selektirati za eliminaciju. Što je veća populacija, vjerojatnost da će neka dretva obavljati posao uzalud je manja (slika 6.6).



Slika 6.6. **Vjerojatnost dvostruke selekcije s dvije dretve i 3-turnirskom selekcijom je funkcija veličine populacije**

Za binarnu turnirsku selekciju se istim postupkom dobiva izraz za $P'_{2,2}$:

$$P'_{2,2}(N)=\sum_{i=1}^{N}\left[\frac{\binom{i}{2}-\binom{i-1}{2}}{\binom{N}{2}}\right]^2=\sum_{i=1}^{N}\left[\frac{2(i-1)}{N(N-1)}\right]^2=\frac{4}{N^2(N-1)^2}\left(\sum_{i=1}^{N}i^2-2\sum_{i=1}^{N}i+N\right).$$

Uvrštavanjem izraza za konačnu sumu redova prvih i drugih potencija dobiva se:

$$P'_{2,2}(N)=\frac{2(2N-1)}{3N(N-1)}. \qquad (6.5)$$

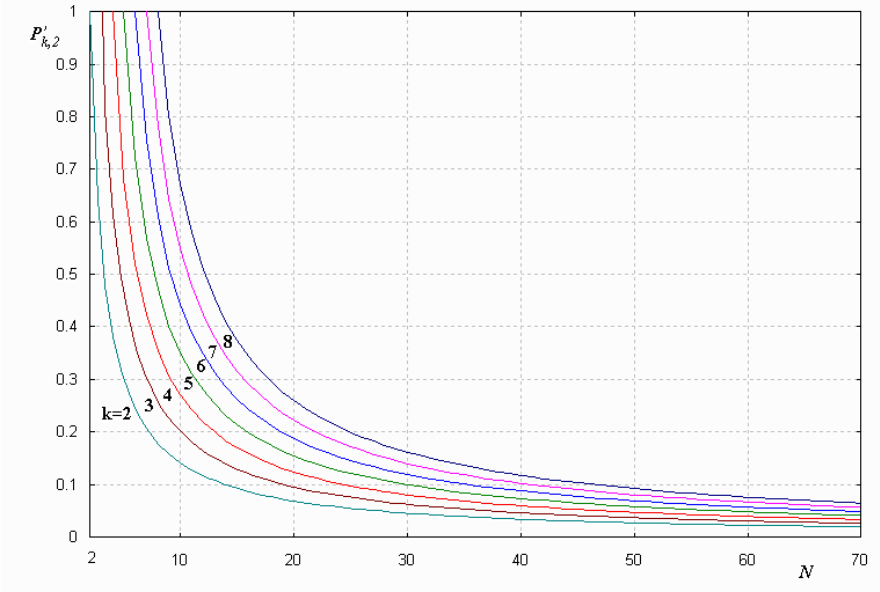
Općenito, za *k*-turnirsku selekciju, vjerojatnost dvostruke selekcije, odnosno vjerojatnost da jedna od dvije dretve beskorisno selektira jedinku za eliminaciju i reprodukciju iznosi:

$$P'_{k,2}(N)=\sum_{i=1}^{N}p_{kE}(i)^2. \qquad (6.6)$$

Uvrštavanjem izraza za vjerojatnost selekcije (5.5), odnosno izraza (5.6) u izraz (6.6) dobiva se:

$$P'_{k,2}(N)=\sum_{i=k}^{N}\left[\frac{\binom{i}{k}-\binom{i-1}{k}}{\binom{N}{k}}\right]^2=\sum_{i=k}^{N}\left[\frac{k\prod_{j=1}^{k-1}(i-j)}{\prod_{j=0}^{k-1}(N-j)}\right]^2=\sum_{i=k}^{N}\left[\frac{k}{N}\prod_{j=1}^{k-1}\frac{(i-j)}{(N-j)}\right]^2. \qquad (6.7)$$

Vjerojatnost eliminacije *k*-1 najboljih jedinki jednaka je nuli. Stoga je svejedno da li suma ide od 1 do *N* kako je to slučaj u izrazu (6.6) ili od *k* do *N* kako je to slučaj, primjerice, u izrazu (6.7).



Slika 6.7. **Vjerojatnost dvostruke selekcije s dvije dretve u ovisnosti o veličini populacije *N* i parametru *k***

Povećavanjem veličine turnira povećava se i vjerojatnost dvostruke selekcije (slika 6.7). Za veličinu turnira $k=N$, vjerojatnost dvostruke selekcije je 1: $\lim_{N\to k}P_{k,D}(N)=1$. Za beskonačno veliku populaciju, vjerojatnost dvostruke selekcije je 0: $\lim_{N\to\infty}P_{k,D}(N)=0$.

Selekcijski pritisak je manji što je veličina turnira manja. Podešavanjem selekcijskog pritiska podešava se brzina konvergencije. Manja brzina konvergencije znači da genetski algoritam treba obaviti više iteracija. Međutim, time se postiže kvalitetnije rješenje jer je vjerojatnost zaglavljivanja u lokalnom optimumu manja. I obrnuto, veća brzina konvergencije znači da će genetski algoritam u kraćem vremenu pronaći možda lošije rješenje. U načelu, bolje je duže računati pa pronaći bolje rješenje, nego kraće računati pa pronaći vjerojatno pogrešno (suboptimalno) rješenje. Stoga je manji *k* bolji izbor. To se poklapa i sa zahtjevom da vjerojatnost višestruke selekcije bude što je moguće manja, jer će tada dretve manje iteracija obavljati uzalud. Naime, vjerojatnost da dretva obavlja posao uzalud pada što je manja veličina turnira (slika 6.7). Dakle, dobar izbor bi bio turnirska selekcija s malom veličinom turnira: 2 ili 3. Predloženi model GPGA obavlja 3-turnirsku selekciju, jer je za selekciju i reprodukciju, koje se obavljaju u istom koraku, potrebno selektirati barem tri jedinke (jedna jedinka se eliminira, a dvije sudjeluju u reprodukciji). Moguće

je ostvariti selekciju i reprodukciju u jednom koraku s većom veličinom turnira, ali zbog opisanih razloga (vjerojatnost da dretva obavlja posao uzalud treba biti što manja, a i selekcijski pritisak treba biti što manji) odabrana je minimalna vrijednost veličine turnira.

Dosad je pokazana ovisnost vjerojatnosti da dretva obavlja posao uzalud o veličini populacije i veličini turnira, ali za konstantnu vrijednost broja dretvi, točnije, za dvije dretve: D=2. Slijede razmatranja ovisnosti vjerojatnosti višestruke selekcije o broju dretvi D.

Neka se asinkroni globalni paralelni genetski algoritam sastoji od D dretvi: $j_1, j_2, ...j_D$. Neka dretva u svakoj iteraciji (svaki puta kada pristupa zajedničkim podacima) nosi jednu od D oznaka: 1,2,3,...,D. Neka dretva koja zadnja zapiše vrijednost u zajedničku spremničku lokaciju ima najmanju oznaku od svih ostalih dretvi koje su pristupile istom podatku. Jedino će ta dretva obavljati korisno tu iteraciju, dok će ostale dretve koje su pristupile istom podatku obavljati posao uzalud. Budući je algoritam asinkroni, dretve proizvoljnim redoslijedom pristupaju podacima. Stoga, u svakoj iteraciji neka druga dretva nosi oznaku jedan. Dretva koja nosi oznaku različitu od jedan može i ne mora obavljati posao uzalud.

Primjerice, neka tri dretve (D=3) obavljaju 3-turnirsku eliminacijsku selekciju (k=3) i neka se populacije sastoji od pet jedinki (N=5). Jedinke s oznakama 1 i 2 ne mogu biti selektirane za eliminaciju, zbog toga što je inherentno ugrađen elitizam s 3-turnirskom eliminacijskom selekcijom. Dretve j_1, j_2 i j_3 mogu pristupati zajedničkim podacima, tj. zajedničkim jedinkama s oznakama 3, 4 i 5 na 27 načina kako je to prikazano u tablici :

Tablica 6.1. **Načini selekcije 3 od 5 jedinki uporabom 3 dretve (j_1, j_2 i j_3) s 3-turnirskom selekcijom i odgovarajućim vjerojatnostima selekcije p u promilima**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1																											
2																											
3	**123**	**12**	**12**	**13**	**1**	**1**	**13**	**1**	**1**	**23**	**2**	**2**	**3**			**3**			**23**	**2**	**2**	**3**			**3**		
4		**3**		**2**	**23**	**2**		**3**		**1**	**13**	**1**	**12**	**123**	**12**	**1**	**13**	**1**		**3**		**2**	**23**	**2**		**3**	
5			**3**			**3**	**2**	**2**	**23**			**3**			**3**	**2**	**2**	**23**	**1**	**1**	**13**	**1**	**1**	**13**	**12**	**12**	**123**
p	*1*	*3*	*6*	*3*	*9*	*18*	*6*	*18*	*36*	*3*	*9*	*18*	*9*	*27*	*54*	*18*	*54*	*108*	*6*	*18*	*36*	*18*	*54*	*108*	*36*	*108*	*216*

Vjerojatnosti selekcije prema izrazu (5.6) za pojedine jedinke iznose: $p_{3E}(1)=p_{3E}(2)=0$, $p_{3E}(3)=1/10$, $p_{3E}(4)=3/10$ i $p_{3E}(5)=6/10$. Primjerice, prema tablici 6.1, sve tri dretve na prvi način (od 27 mogućih načina) selektiraju jedinku s oznakom 3. Vjerojatnost tog događaja je $p_{3E}(3)^3=1/1000$ (1‰). Kada se to dogodi, dretva koja posljednja upiše novu vrijednost u zajedničku spremničku lokaciju će korisno obavljati posao. Ta dretva koja korisno obavlja posao nosi oznaku 1. Ostale dvije dretve s oznakama 2 i 3 će obavljati posao uzalud. Zatim, neka dretve s oznakama 1 i 2 selektiraju jedinku s oznakom 3, a dretva s oznakom 3 selektira za eliminaciju jedinku s oznakom 4 (drugi stupac u tablici 6.1). Vjerojatnost tog događaja je $p_{3E}(3)^2 \cdot p_{3E}(4)=3/1000$ (3‰). U tom slučaju, dretve s oznakama 1 i 3 će korisno obavljati posao, a dretva s oznakom 2 će s vjerojatnošću 3/1000 (3‰) posao obavljati uzalud.

Dretva koja nosi oznaku 1 nikad ne obavlja posao uzalud. Dretva s oznakom 2 obavlja posao uzalud samo onda kada dretva s oznakom 1 pristupa istom podatku u zajedničkom radnom spremniku. To su načini selekcije 1,2,3,13,14,15,25,26 i 27 u tablici 6.1. Vjerojatnost da dretva s oznakom 2 obavlja posao uzalud iznosi: 1+3+6+9+27+54+36+108+216=460‰. Zapravo, to je vjerojatnost višestruke selekcije za dvije dretve (prema izrazu (6.2)): $P'_{3,2}(5) = p_{3E}(1)^2 + p_{3E}(2)^2 + p_{3E}(3)^2 + p_{3E}(4)^2 + p_{3E}(5)^2 = 0.46$.

Dretva s oznakom 3 obavlja posao uzalud onda kada bilo koja od preostale dvije dretve pristupa istom podatku u zajedničkom radnom spremniku. To su načini selekcije 1,4,5,7,9,10,11,14, 17,18,19,21,23,24 i 27 u tablici 6.1. Vjerojatnost da dretva s oznakom 3 obavlja posao uzalud iznosi:

1+3+9+6+36+3+9+27+54+108+6+36+54+108+216=676‰.

Općenito, za D dretvi i k-turnirsku selekciju vrijedi sljedeće:

- Dretva s oznakom 1 nikad ne obavlja posao uzalud:

$$P'_{k,1} = 0; \qquad (6.8)$$

- Dretva s oznakom 2 će posao jedne iteracije obavljati uzalud prema izrazu (6.6);
- Za svaku sljedeću dretvu vjerojatnost da posao jedne iteracije dretva obavlja uzalud jednaka je jedan minus vjerojatnost da dretva korisno obavlja posao (p=1-q). U daljnjem tekstu će se pokazati da vjerojatnost da dretva s oznakom j obavlja posao uzalud iznosi:

$$P'_{k,j} = 1 - \sum_{i=1}^{N} p_{kE}(i) \cdot q_{kE}(i)^{j-1}, \qquad (6.9)$$

gdje je $j \le D$.

Vjerojatnost eliminacije *i*-te jedinke uporabom *k*-turnirske selekcije $p_{kE}(i)$ računa se prema izrazima (5.5) ili (5.6). Dretva s oznakom *j* će korisno obavljati posao neke iteracije, ako dretve s oznakama manjim od *j* selektiraju neke druge jedinke, samo ne onu koju je selektirala dretva s oznakom *j*. Vjerojatnost da će *i*-tu jedinku selektirati samo jedna od *D* dretvi i to ona s oznakom *j* iznosi $p_{kE}(i)\cdot q_{kE}(i)^{j-1}$, jer *j*-1 dretvi s oznakama manjim od *j* trebaju selektirati neku drugu jedinku, samo ne *i*-tu. Dretva može korisno obavljati posao na *N* načina: može selektirati bilo koju od *N* jedinki, a da ostale dretve selektiraju neke druge jedinke. Prema tome, vjerojatnost da će dretva s oznakom *j* korisno obavljati posao jedne iteracije jednaka je sumi vjerojatnosti da sama selektira svaku od *N* jedinki:

$$Q'_{k,j} = \sum_{i=1}^{N} p_{kE}(i)\cdot q_{kE}(i)^{j-1} . \qquad (6.10)$$

Vjerojatnost da dretva obavlja posao uzalud jednaka je jedan minus vjerojatnost da dretva korisno obavlja posao ($P'_{k,j}$=1-$Q'_{k,j}$) i na taj se način dobije izraz (6.9).

Izraz (6.9) vrijedi općenito za dretve sa svim oznakama, pa i za one s oznakama 1 i 2. Moguće je provjeriti da za *j*=1 izraz (6.9) poprima oblik izraza (6.8):

$$P'_{k,1} = 1-\sum_{i=1}^{N} p_{kE}(i)\cdot q_{kE}(i)^{0} = 1-\sum_{i=1}^{N} p_{kE}(i) = 1-1 = 0 . \blacksquare$$

Nadalje, uzme li se u obzir da je *p*=1-*q* i $\sum_i p(i) = 1$ za *d*=2, moguće je provjeriti da tada izraz (6.9) poprima oblik izraza (6.8):

$$P'_{k,2} = 1-\sum_{i=1}^{N} p_{kE}(i)\cdot q_{kE}(i) = 1-\sum_{i=1}^{N}\left[p_{kE}(i)\cdot\left(1-p_{kE}(i)\right)\right] = 1-\sum_{i=1}^{N} p_{kE}(i)+\sum_{i=1}^{N} p_{kE}(i)^2 = 1-1+\sum_{i=1}^{N} p_{kE}(i)^2 = \sum_{i=1}^{N} p_{kE}(i)^2 . \blacksquare$$

I konačno, s obzirom da svaka dretva u sustavu bez sinkronizacije može poprimiti bilo koju oznaku s jednakom vjerojatnošću, vjerojatnost da neka dretva u sustavu s *D* dretvi obavlja posao uzalud iznosi:

$$P_{k,D} = \frac{1}{D}\sum_{d=1}^{D} P'_{k,d} . \qquad (6.11)$$

Uvrštavanjem izraza (6.9) u izraz (6.11) dobiva se izraz za vjerojatnost da dretva obavlja posao uzalud u nekoj iteraciji:

$$P_{k,D} = 1-\frac{1}{D}\sum_{d=1}^{D}\sum_{i=1}^{N} p_{kE}(i)\cdot q_{kE}(i)^{d-1} , \qquad (6.12)$$

odnosno, vjerojatnost da dretva korisno obavlja posao iznosi:

$$Q_{k,D} = \frac{1}{D}\sum_{d=1}^{D}\sum_{i=1}^{N} p_{kE}(i)\cdot q_{kE}(i)^{d-1} , \qquad (6.13)$$

jer je:

$$P_{k,D} + Q_{k,D} = 1 . \qquad (6.14)$$

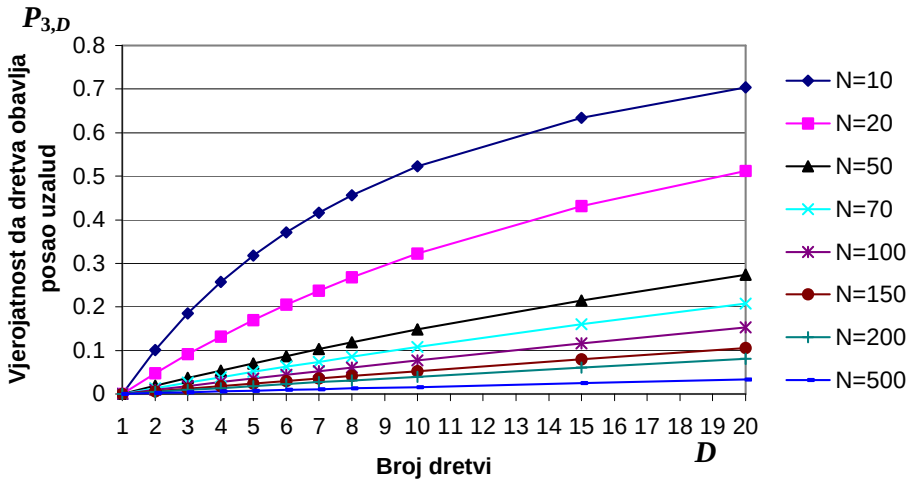
Vjerojatnost da neka dretva u sustavu s *D* dretvi obavlja posao uzalud raste kako se smanjuje veličina populacije *N*, povećava broj dretvi *D* ili se povećava veličina turnira *k* (tablica 6.2 i slika 6.8).

Tablica 6.2. **Vjerojatnost da će dretva beskorisno obavljati posao u ovisnosti o veličini populacije *N* i broja dretvi *D* za GPGA s 3-turnirskom selekcijom ($P_{3,D}$)**

	Broj dretvi *D*									
N	1	2	3	4	5	6	7	8	10	20
10	0	0.1004	0.1849	0.2566	0.3178	0.3705	0.4162	0.4561	0.5221	0.7039
20	0	0.0474	0.0913	0.1319	0.1695	0.2045	0.2371	0.2674	0.3221	0.5121
50	0	0.0184	0.0362	0.0535	0.0703	0.0867	0.1026	0.118	0.1476	0.2734
70	0	0.013	0.0258	0.0383	0.0506	0.0626	0.0734	0.0859	0.1082	0.2076
100	0	0.0091	0.0181	0.0269	0.0356	0.0442	0.0526	0.061	0.0773	0.1524
150	0	0.006	0.012	0.0179	0.0238	0.0296	0.0359	0.0411	0.0523	0.1055
200	0	0.0045	0.009	0.0135	0.0179	0.0223	0.0267	0.031	0.0396	0.0806
500	0	0.0018	0.0036	0.0054	0.0072	0.009	0.0107	0.0125	0.016	0.0334

Vrijednosti veličine populacije za sekvencijski genetski algoritam najčešće poprimaju vrijednosti u intervalu od 70 do 200 jedinki. Predloženi asinkroni model paralelnog genetskog algoritma ima vrlo slična svojstva kao i sekvencijski genetski algoritam pa se za veličine populacije uzimaju iste vrijednosti. Predloženi model GPGA je prilagođen izvođenju na računalima sa svega nekoliko procesora (od 2 do 8 procesora). Stoga su očekivane vjerojatnosti da GPGA obavlja posao uzalud s dvije do osam dretvi označene sivo u tablici 6.2 i zaokružene

crtkanom linijom na slici 6.8. Posebno su naznačena četiri područja. Za više dretvi treba veličina populacije biti veća, ako se, primjerice, želi držati vjerojatnost beskorisnog djelovanja dretve do 5%.



Slika 6.8. **Ovisnost vjerojatnosti da dretva obavlja posao 3-turnirske selekcije i reprodukcije uzalud o veličini populacije i broju dretvi**

6.3.3. Broj iteracija

U opisanom modelu asinkronog globalnog paralelnog genetskog algoritma dretve obavljaju s određenom vjerojatnošću posao uzalud. Te vjerojatnosti su izračunate u prethodnom poglavlju. Da bi asinkroni GPGA imao ista svojstva kao i odgovarajući sekvencijski genetski algoritam, zbroj *korisnih* iteracija svih dretvi mora biti jednak broju iteracija koji obavi sekvencijski genetski algoritam.

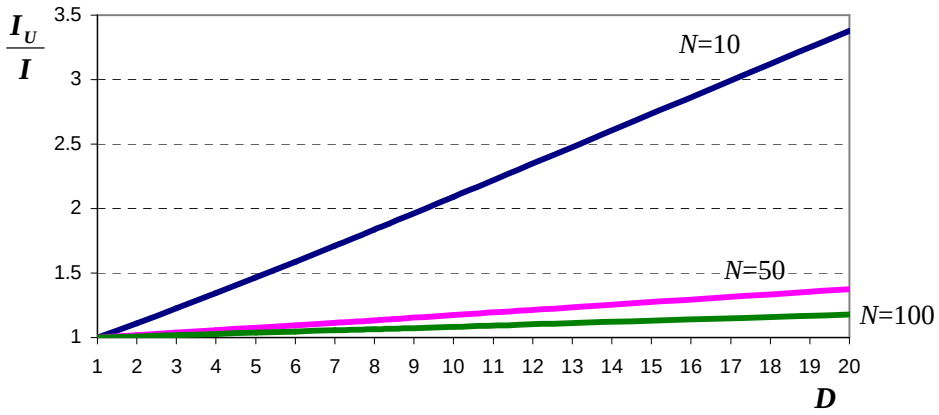
Neka je broj iteracija koje obavlja sekvencijski genetski algoritam jednak I, a ukupan broj iteracija koji obavi paralelni genetski algoritam neka je I_U. Ukupan broj iteracija koje obavi GPGA jednak je zbroju iteracija svih D dretvi. Pretpostavlja se da sve dretve obave otprilike jednak broj iteracija, jer su sve dretve jednakog prioriteta. Dakle, svaka dretva obavi I_U/D iteracija. Kada bi broj iteracija GPGA bio jednak broju iteracija koje obavi sekvencijski GA, efikasnost GPGA bi bila lošija, jer asinkroni GPGA obavi određeni broj iteracija uzalud. Za poznatu vjerojatnost da dretva obavlja posao jedne iteracije uzalud (izraz (6.12)), odnosno vjerojatnost da dretva obavlja korisno posao jedne iteracije (izraz (6.13)), moguće je izračunati potreban ukupan broj iteracija. Razlika ukupnog broja iteracija i očekivanog broja beskorisnih iteracija treba biti jednaka broju iteracija sekvencijskog GA:

$$I_U - I_U \cdot P_{k,D} = I. \qquad (6.15)$$

Uz pomoć izraza (6.15) dobiva se izraz za ukupan ili *potreban* broj iteracija GPGA, a da pritom GPGA postiže rješenje iste kvalitete kao i sekvencijski GA:

$$I_U = \frac{I}{Q_{k,D}}. \qquad (6.16)$$

Drugim riječima, GPGA i sekvencijski GA za jednak broj *korisnih* iteracija (I) postižu jednako kvalitetno rješenje (slika 6.9).



Slika 6.9. **Potreban broj iteracija da bi asinkroni GPGA s 3-turnirskom selekcijom postizao ista rješenja kao i odgovarajući sekvencijski GA**

Primjerice, neka sekvencijski GA s 3-turnirskom eliminacijskom selekcijom obavi 100.000 iteracija nad populacijom koja se sastoji od 50 jedinki. Neka je na raspolaganju dvoprocesorsko računalo (N_P =2). Odgovarajući asinkroni GPGA s dvije dretve (D=N_P=2) treba obaviti ukupno I_U=100.000/$Q_{3,2}$=101.871 iteracija. Dakle, GPGA treba obaviti 1.871 iteracija više od odgovarajućeg sekvencijskog GA.

6.4. Sinkroni model globalnog paralelnog genetskog algoritma

U prethodnom poglavlju opisan je asinkroni GPGA i problemi vezani uz nejednoznačnost podataka kada više dretvi mijenja podatke u istim spremničkim lokacijama. Predloženo je rješenje povećanje broja iteracija, koje, naravno, ima za posljedicu usporenje rada algoritma. Drugo rješenje istog problema je sinkronizacija dretvi prilikom pristupa zajedničkim podacima. Zajednički podaci su jedinke. Svako čitanje i izmjena zajedničkih podataka treba se označiti kao kritični odsječak i zaštititi ga s nekim od mehanizama za međusobno isključivanje.

Mehanizmi za sinkronizaciju rada dretvi su: binarni semafor, opći semafor, uvjetne varijable, varijable međusobnog isključivanja, itd. Za sinkronizaciju rada dretvi moguće je koristiti i redove poruka ili cjevovode. Od svih navedenih mehanizama pokazalo se da je vremenski najmanje zahtjevna sinkronizacija s pomoću varijabli za međusobno isključivanje. Neka varijabla za međusobno isključivanje je ili zaključana ili otključana. Ako je varijabla zaključana, dretva koja želi uću u kritični odsječak čekat će tako dugo dok neka druga dretva ne otključa tu varijablu.

Najjednostavniji način sinkronizacije rada dretvi je s pomoću N varijabli međusobnog isključivanja. Svaka jedinka ima svoju varijablu za međusobno isključivanje. U određenom trenutku samo jedna dretva može čitati ili pisati po spremničkim lokacijama neke jedinke. Taj najjednostavniji način sinkronizacije daje i najlošije rezultate. Osim što dretve troše vrijeme na zaključavanje i otključavanje varijabli, dretve se međusobno čekaju kada žele pristupiti istim podacima. Dakle, kada dretve žele pristupiti istim podacima, one se neće obavljati paralelno, već sekvencijski. Prvo će jedna dretva obaviti posao nad trima jedinkama, pa tek onda ona druga koja je selektirala jednu, dvije ili sve tri iste jedinke kao i ona prva. Vrijeme potrošeno za sinkronizacijski mehanizam, zajedno s vremenom koje dretve potroše zbog toga što se međusobno čekaju, čini znatan udio u ukupnom vremenu koje potroši GPGA [BUD98].

Poboljšanje se može postići tako da se ne zaključavaju sve tri jedinke, nego samo najlošija - ona koja će biti zamijenjena djetetom preostale (preživjele) dvije jedinke. Vjerojatnost da će se dvije dretve međusobno čekati (zato što žele eliminirati i nodomjestiti istu jedinku) je ista kao i vjerojatnost da neka dretva obavlja posao uzalud pri asinkronom modelu. Asinkroni model mora obaviti dodatnu iteraciju, a sinkroni troši vrijeme na čekanje. Štoviše, potroši i nešto više vremena koliko potroše mehanizmi za međusobno isključivanje. Dakle, za postizanje rješenja određene kvalitete takav sinkroni model potroši ukupno više vremena od asinkronog modela koji treba odraditi dodatne iteracije.

```
Dretva_s_međusobno_isključivim_odabirom_tri_jedinke(){
  sve dok(nije_zadovoljen_uvjet_završetka_evolucijskog procesa){
    mutex_lock(zajednička_varijabla_zaključavanja);
    // kritični odsječak: odaberi slučajno tri različite jedinke iz
    // populacije, ali pazi da jedinke nisu zauzete
    radi{
      roditelj1 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(zastavica[roditelj1]==1);
    zastavica[roditelj1] = 1;
    radi{
      roditelj2 = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(zastavica[roditelj2]==1);
    zastavica[roditelj2] = 1;
    radi{
      dijete = odaberi_slučajno_jednu_jedinku_iz_populacije();
    }sve dok(zastavica[dijete]==1);
    zastavica[dijete] = 1;
    // kraj kritičnog odsječka
    mutex_unclock(zajednička_varijabla_zaključavanja);
    // dijete treba biti najlošija jedinka od tri slučajno odabrane
    ako(f(dijete)>f(roditelj1)) zamijeni(dijete,roditelj1);
    ako(f(dijete)>f(roditelj2)) zamijeni(dijete,roditelj2);
    // reprodukcija
    dijete = križaj(roditelj1,roditelj2);
    zastavica[roditelj1] = zastavica[roditelj2] = 0;
    mutiraj(dijete);
    evaluiraj(dijete);
    zastavice[dijete] = 0;
  }
}
```

Slika 6.10. **Dretva s 3-turnirskom eliminacijskom selekcijom bez duplikata i sa sinkronizacijskim mehanizmom koji štiti odabir tri jedinke za selekciju i reprodukciju**

Rješenje problema kako izgraditi bolji sinkroni model od dva prethodno opisana, je u samoj prirodi genetskih algoritama. Genetski algoritam s 3-turnirskom selekcijom *nasumice* odabire jedinke koje će sudjelovati u reprodukciji (sve jedinke imaju jednaku vjerojatnost sudjelovanja u reprodukciji). Ako je već za genetski algoritam svejedno koje su to tri jedinke, tada, ako je neka nasumice odabrana jedinka zauzeta, ne treba baš ona sudjelovati u reprodukciji. Jednostavno treba odabrati neku drugu jedinku. Stoga je vjerojatnost selekcije već zauzete jedinke jednaka nuli. Posljedica toga je još jedno pojednostavljenje: ne treba *N* varijabli za međusobno isključivanje, već samo jedna. Kritični odsječak nije izmjena neke jedinke, već je to postupak *slučajnog* odabira jedinki.

Neka umjesto *N* varijabli međusobnog isključivanja postoji polje zastavica od *N* elemenata. Svaka jedinka ima svoju zastavicu, koja pokazuje da li nad tom jedinkom neka dretva obavlja genetske operatore. Pristup zastavicama je zaštićen sa samo jednom varijablom međusobnog isključivanja. Svaka dretva prije no što uđe u kritični odsječak u kojem će odabrati tri jedinke za reprodukciju, treba zaključati varijablu međusobnog isključivanja. Potom slučajno odabire tri jedinke čije zastavice ne pokazuju zauzeće. Postupak odabira jedinki traje kratko, jer dretve ne trebaju čekati na oslobođenje slučajno odabrane jedinke, ako je ona zauzeta, već jednostavno odabiru neku drugu jedinku. Primjer dretve s takvim sinkronizacijskim mehanizmom prikazan je na slici 6.10. Odabir tri jedinke za reprodukciju označen je kao kritični odsječak koji može istodobno obavljati u nekom trenutku samo jedna dretva.

Roditelji su u postupku reprodukcije dvije od tri slučajno odabrane jedinke. Te jedinke se tijekom postupka reprodukcije ne mijenjaju, već se njihovi kromosomi samo čitaju. Ako se podaci samo čitaju, nije ih potrebno isključivo čitati. U najgorem slučaju se može dogoditi da neka druga dretva izmijeni jednog roditelja (ili čak više dretvi mogu izmijeniti oba roditelja) i da neki od roditelja postane lošiji od jedinke koja je određena za eliminaciju. Taj (malo vjerojatni) događaj se može tretirati kao dodatna mutacija.

6.5. Prednosti i nedostaci

Predloženi model globalnog paralelnog algoritma ima svoje prednosti, a i nedostatke u odnosu na ostale modele paralelnih genetskih algoritama. Namijenjen je izvođenju na višeprocesorskom sustavu sa svega nekoliko procesora (2, 4 ili 8) i sa zajedničkim radnim spremnikom. Broj procesora je ulazni parametar algoritma. Novi model GPGA se može bez izmjene izvornog teksta programa izvoditi na računalu s proizvoljnim brojem procesora, pa tako i na jednoprocesorskom računalo (tada je to sekvencijski, a ne paralelni algoritam).

Za očuvanje dvije najbolje jedinke od bilo kakve izmjene nije potrebno pisati niti jedne linije dodatnog teksta programa, jer je elitizam inherentno ugrađen u 3-turnirskoj selekciji. Izvršni program je kratak (≈48kB) i najvjerojatnije može cijeli stati u priručni spremnik procesora zajedno s tri jedinke nad kojima djeluju genetski operatori. Za binarni prikaz (koji se ujedno i najčešće koristi) genetski operatori se mogu ostvariti uporabom samo logičkih funkcija NE, I, ILI, XILI i POMAKNI koje su ujedno sastavni dio svakog strojnog jezika računala. Algoritam je stoga vrlo jednostavan za implementaciju i na najnižem nivou, tj. strojnom jeziku računala.

Tablica 6.3. **Prednosti i nedostaci novog modela GPGA nad ostalim modelima PGA**

Prednosti	Nedostaci
• najpogodniji za izvođenje na višeprocesorskom računalu sa zajedničkim radnim spremnikom • jednostavan za implementaciju • izvorni tekst programa je kratak i može stati sav u priručni spremnik procesora • algoritam se može izvoditi na računalnom sustavu s proizvoljnim brojem procesora bez ijedne izmjene u izvornom kodu programa • elitizam je inherentno ugrađen • svi genetski operatori se obavljaju paralelno • nema potreba za mehanizmom međusobne komunikacije između dretvi • može raditi bez sinkronizacije • trajanje izvođenja za zadani broj iteracija ne ovisi o veličini populacije • ima ista svojstva kao i sekvencijski algoritam • postiže se ubrzanje koje je približno jednako broju procesora	• kod asinkronog modela broj iteracija se mora povećati, jer dretva može obavljati neke iteracije uzalud ili se mora ugraditi sinkronizacijski mehanizam • sinkroni model troši dodatno vrijeme za učestalu sinkronizaciju i na međusobno čekanje dretvi kada istodobno žele ući u kritični odsječak • ne može se koristiti neki drugi postupak selekcije osim turnirske selekcije

Za razliku od tradicionalnog modela GPGA u kojem se samo evaluacija izvodi paralelno, a ostali genetski operatori sekvencijski, u predloženom modelu GPGA svi se operatori izvode paralelno. Nema potrebe za ugradnjom nikakvog mehanizma za međusobnu komunikaciju dretvi, jer se podaci nalaze u zajedničkom radnom spremniku. Unatoč tome što su podaci zajednički, moguće je izvoditi dretve paralelno bez ikakve sinkronizacije kako je to opisano u poglavlju 6.3.

Trajanje izvođenja jedne iteracije ne ovisi o veličini populacije, jer u jednoj iteraciji algoritam djeluje samo nad tri slučajno odabrane jedinke, a ne nad svih *N* jedinki kao što je to slučaj sa svim ostalim postupcima selekcije. Stoga i ukupno trajanje izvođenja ne ovisi o veličini populacije, već samo o broju iteracija. Međutim, kvaliteta dobivenog rješenja značajno ovisi o oba parametra: i o broju iteracija i o veličini populacije. Stoga ukupno trajanje izvođenja

ipak posredno ovisi i o veličini populacije, ako je unaprijed zadana kvaliteta rješenja koja se optimiranjem mora postići.

Sinkroni GPGA ima potpuno ista svojstva kao i sekvencijski algoritam. Asinkroni model GPGA nema baš ista svojstva kao i sekvencijski GA, jer se moraju odraditi dodatne iteracije. Asinkroni GPGA koji odradi zadani broj *korisnih* iteracija (ma koliko ukupan broj iteracija bio), ima također potpuno ista svojstva kao i sekvencijski GA. To je važno za teorijsku analizu rada algoritma, jer se sva teorijska razmatranja sekvencijskog algoritma mogu primijeniti i na opisani model GPGA.

Posljednja, ali i najvažnija prednost je da predloženi paralelni genetski algoritam na N_P-procesorskom računalu postiže rješenje iste kvalitete kao i odgovarajući sekvencijski algoritam na jednoprocesorskom računalu, ali gotovo N_P puta brže. Točnije, onoliko puta brže kolika je propusnost paralelnog računala.

Nasuprot brojnim navedenim prednostima, novi model GPGA ima i nekoliko nedostataka. Asinkroni model obavlja određeni broj iteracija uzalud zbog toga što dretve istodobno mogu mijenjati iste jedinke. Naime, broj neregularnih iteracija se može izračunati, dakle i predvidjeti, pa se za toliko može povećati ukupan broj iteracija. Posljedica je da asinkroni GPGA ipak nema ubrzanje jednako broju procesora, kao što bi to bio slučaj da nema tih dodatnih iteracija. Drugo rješenje problema neregularnih iteracija je sinkroni model GPGA. Sinkroni model GPGA ne dozvoljava pristup više od jedne dretve istom podatku, ali taj postupak zahtijeva dodatno vrijeme koje troše mehanizmi za međusobno isključivanje i dretve troše dodatno vrijeme na čekanje za ulazak u kritični odsječak. Opisani model GPGA koristi 3-turnirsku selekciju ili, uz manje izmjene u izvornom tekstu programa, bilo koju k-turnirsku selekciju, gdje je k>2. Proporcionalne i sortirajuće selekcije nisu pogodne za taj model.

Novi model globalnog paralelnog genetskog algoritma može biti sinkroni i asinkroni. Teško je odrediti koji je od ta dva modela bolji, jer svaki ima svoje prednosti i nedostatke:

Tablica 6.4. **Prednosti i nedostaci sinkronog i asinkronog modela GPGA**

Model GPGA	Prednosti	Nedostaci
ASINKRONI	• izvorni tekst programa je kraći • jednostavniji za implementaciju • izvođenje iteracije kraće traje, jer nema sinkronizacijskog mehanizma • broj beskorisnih iteracija je predvidiv (može se izračunati)	• dretva može obavljati iteraciju uzalud pa se takva *beskorisna* iteracija mora ponoviti
SINKRONI	• nije narušena jednoznačnost podataka • ima potpuno ista svojstva kao i odgovarajući sekvencijski GA	• izvorni tekst programa je složeniji • treba voditi računa da ne dođe do potpunog zastoja • sinkronizacijski mehanizmi troše stanovito procesorsko vrijeme • dretve se međusobno čekaju kada više njih želi ući u kritični odsječak

Asinkroni model je jednostavniji za implementaciju, jer nema sinkronizacijskih mehanizama pa je i izvorni tekst programa kraći. Trajanje jedne iteracije je također kraće zbog toga što nema sinkronizacije. Međutim, upravo zbog toga što nema sinkronizacije dretve mogu istovremeno mijenjati iste podatke. Ako više dretvi istodobno upisuje vrijednost u istu spremničku lokaciju, ostat će zapisana vrijednost samo jedne dretve (one koja zadnja upiše podatak), dok će ostale dretve tu iteraciju obavljati uzalud. Takva beskorisna iteracija se mora ponoviti. Broj beskorisnih iteracija, odnosno broj ponovljenih iteracija, je predvidiv i može se izračunati (vidi poglavlje 6.3.3).

Sinkroni model je složeniji i troši se procesorsko vrijeme na sinkronizaciju dretvi. Ukoliko se više dijelova programa označi kao kritični odsječak, mora se paziti da ne dođe do potpunog zastoja. S druge strane, nije narušena konzistentnost podataka, kao što je to slučaj pri asinkronom modelu. Prednost sinkronog modela nad asinkronim je i u tome što ima potpuno ista svojstva (bez izvođenja dodatnih iteracija) kao i odgovarajući sekvencijski genetski algoritam.

6.6. Sklopovska potpora novom modelu GPGA

6.6.1. Načini prilagodbe arhitekture računala genetskim algoritmima

U ovom poglavlju razmotrene su mogućnosti prilagodbe procesora novom modelu GPGA. Svojstva GPGA bi se mogla dodatno poboljšati povećanjem brzine izvođenja, ako se arhitektura računala prilagodi osebujnim zahtjevima genetskih algoritama. Ustanovljeno je da se na tri načina može skratiti trajanje optimiranja:

- prepuštanjem *kontrole jednoznačnosti podataka* samom algoritmu, umjesto protokolu za održavanje jednoznačnosti preslika podatkovnih objekata,

- sklopovskim ostvarenjem *generatora slučajnih brojeva* i
- proširenjem instrukcijskog skupa procesora specifičnim instrukcijama *križaj* i *mutiraj*.

6.6.2. Priručni spremnik

Protokol za održavanje jednoznačnosti preslika podatkovnih objekata[42] usporava asinkroni GPGA, a na sinkroni GPGA nema utjecaja. Svaki put kada se dogodi promjena podatka, koji se nalaze u priručnom spremniku dva ili više procesora, protokol za održavanje jednoznačnosti preslika podatkovnih objekata proglašava jednu od trenutno postojećih vrijednosti za isti podatak valjanom. Zatim, tu valjanu vrijednost kopira na sva ostala mjesta gdje se nalazi preslika tog podatka: u priručni spremnik procesora i u zajednički radni spremnik. Taj posao bi se mogao izbjeći kada bi se podaci mogli dobavljati direktno iz zajedničkog radnog spremnika u registre procesora i obrnuto. Time bi se omogućila programska kontrolira jednoznačnost podataka u višeprocesorskom sustavu. Pristup zajedničkom radnom spremniku je doduše vremenski zahtjevniji od pristupa priručnom spremniku, ali bi se na taj način izbjeglo učestalo nepotrebno pozivanje protokola za održavanje jednoznačnosti preslika podataka.

Isključivanjem protokola za održavanje jednoznačnosti (što je na nekim računalima moguće učiniti programski) ne bi se postiglo nikakvo poboljšanje svojstava asinkronog GPGA. Štoviše, ako je priručni spremnik dovoljno velik da u njega stane veliki dio ili čak cijela populacija, podaci više ne bi bili zajednički, već bi populacija bila razdijeljena na subpopulacije. Subpopulacije bi postale izolirane i GPGA bi se pretvorio u neki degenerativni oblik DGA s umanjenim brojem iteracija i bez operatora migracije. Kvaliteta postignutog rješenja bila bi značajno lošija od odgovarajućeg sekvencijskog genetskog algoritma.

Za asinkroni GPGA bi bilo idealno da je izvršni program u priručnom spremniku, a populacija u zajedničkom radnom spremniku. Na početku svake iteracije dobavile bi se tri jedinke, po mogućnosti, direktno u registre procesora, a na kraju iteracije, rezultat bi se spremio u zajednički radni spremnik. To bi bilo moguće ostvariti samo ako je na neki način omogućena programska kontrola sadržaja priručnog spremnika.

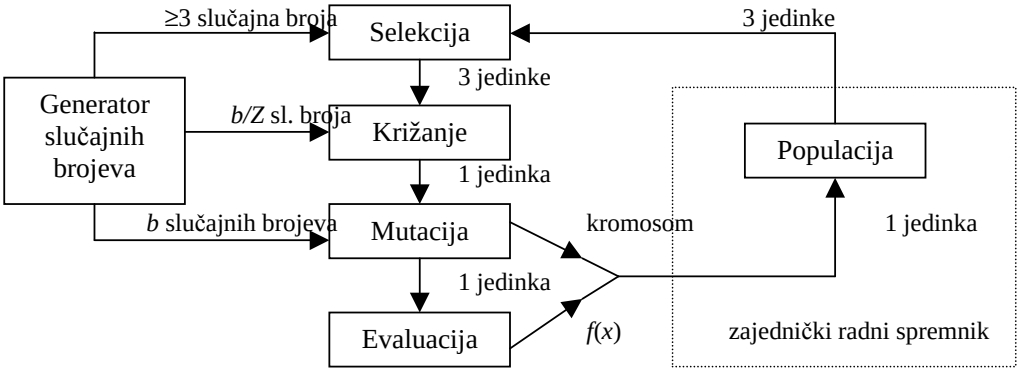
6.6.3. Generator slučajnih brojeva

Genetski algoritam s 3-turnirskom selekcijom u svakoj iteraciji *slučajno* odabire jedinke koje će sudjelovati u reprodukciji, obavlja križanje koje *slučajno* odabire točke prekida i mutacijom *slučajno* mijenja informaciju. Prema tome, ulazni podaci genetskim operatorima su jedinke i slučajno generirani brojevi.

Kvaliteta niza slučajnih brojeva ne utječe značajno na svojstva genetskog algoritma. Meysenburg i Foster su ispitivali efikasnost genetskog algoritma na nizu različitih generatora pseudoslučajnih brojeva. Zaključili su da generator pseudoslučajnih brojeva ima vrlo mali ili nikakav utjecaj na efikasnost genetskog algoritma [MEY99].

Neka je slučajno generirani broj duljine Z bitova. Genetski algoritam s k-turnirskom selekcijom bez duplikata, uniformnim križanjem i mutacijom u svakoj iteraciji koristi barem $k+b/Z+2$ slučajno generirana broja. Selekcija koristi barem k slučajnih brojeva za slučajan odabir jedinki: ukoliko se odabere jedna te ista jedinka više od jedanput, treba se ponovno slučajnim postupkom odabrati nova jedinka sve dok se ne odabere k različitih jedinki. Uniformno križanje koristi slučajno generiran niz bitova koji je jednake duljine kao i kromosom: b bitova. Stoga je potrebno generirati b/Z slučajnih brojeva. Operator mutacije koristi barem dva slučajno generirana broja (izmijenjena mutacije) ili b slučajnih brojeva (jednostavna mutacija).

Tok podataka za genetski algoritam s 3-turnirskom eliminacijskom selekcijom bez duplikata, uniformnim križanjem i jednostavnom mutacijom prikazan je na slici 6.11.



Slika 6.11. **Slikovni prikaz toka podataka**

[42] engleski: *cache coherence protocol*

Generator slučajnih brojeva bi se mogao sklopovski ostvariti kao dodatni sklop unutar procesora. Slučajan broj bi se mogao pohraniti u neki poseban registar *R*. Taj registar bi, za razliku od ostalih registara, imao dodatno svojstvo: svakim čitanjem njegovog sadržaja pokretao bi se generator slučajnih brojeva. Ulaz u sklop za generiranje slučajnih brojeva bilo bi sjeme slučajnih brojeva. Sjeme može biti vrijeme kada je procesor uključen u rad.

6.6.4. Križanje i mutacija kao dodatne instrukcije procesora

Uniformno križanje, nasuprot križanja s jednom, dvije ili nekoliko točaka prekida, daje najbolje rezultate na širokom spektru problema [MIC94, MUE94, WIL97]. Uniformno križanje se može ostvariti i uz pomoć logičkih operacija. Prvo treba izdvojiti one bitove koji su jednaki i prepisati ih. To se postiže logičkom funkcijom *AB*, gdje su *A* i *B* roditelji. Na mjestima gdje su bitovi različiti, slučajno se odabiru geni roditelja *A* ili *B*. Maska slučajnih bitova po kojima se roditelji razlikuju dobiva se logičkom operacijom EXILI: $A \oplus B$. Dakle, križanje se može ostvariti logičkim funkcijama *I*, *ILI* i *EXILI*. Instrukcija *križaj*(*A*,*B*,*R*) bi obavljala sljedeću logičku funkciju:

$$D=AB+R(A \oplus B), \qquad (6.17)$$

gdje je *R* slučajno generiran kromosom, a *D* je mjesto gdje se sprema rezultat. Minimizacijom dobivenog logičkog izraza dobiva se:

$$D=AB+RA+RB. \qquad (6.18)$$

Logičke funkcije *I*, *ILI* i *EXILI* su ostvarene na nivou instrukcija svakog procesora. Da bi se ubrzalo izvođenje genetskog algoritma, najbolje je koristiti logičke funkcije na nivou strojnog jezika računala. Proširenjem skupa instrukcija s instrukcijom *križaj* mogli bi se postići još bolji rezultati. Argumenti nove instrukcije bila bi dva registra. Prije poziva instrukcije *križaj* registar_X, registar_Y, treba pohraniti roditelje *A* i *B* u registre procesora registar_X i registar_Y.

Nadalje, prema eksperimentalnim rezultatima iz [GOL01] vidljivo je da mutacija troši najviše procesorskog vremena, unatoč tome što je to najjednostavniji genetski operator. Razlog tome je učestalo pozivanje funkcije koja generira slučajan broj. Programski je moguće trajanje izvođenja mutacije skratiti, ali i dalje mutacija troši znatan udio procesorskog vremena. Ukoliko bi generator slučajnih brojeva bio riješen sklopovski, problem bi bio riješen, tj. značajno bi se skratilo trajanje izvođenja mutacije.

Međutim, bez obzira da li je generator slučajnih brojeva ostvaren sklopovski ili ne, trajanje izvođenja mutacije može se dodatno skratiti, ako se mutacija ostvari kao instrukcija procesora. Također, ma kako se ostvarila mutacija, da li na nivou bitova (jednostavna mutacija), ili na nivou kromosoma (izmijenjena mutacija), postupak bi se mogao dodatno ubrzati instrukcijom procesora koja bi slučajno mijenjala jedan bit informacije dijela kromosoma čija je veličina (broj bitova) jednaka veličini podatkovnog registra procesora. Taj postupak se može izraziti kao logička funkcija, slično kao i kod križanja. Nova procesorska instrukcija *mutiraj*(A) bi obavljala sljedeću logičku funkciju nad registrom A:

$$A=A \oplus (1<<R\%b), \qquad (6.19)$$

gdje je *R* slučajan broj. Navedenim izrazom se slučajnim postupkom mijenja jedan bit registra A.

7. KARAKTERISTIČNE KRIVULJE

7.1. Što je cilj eksperimentiranja?

Ciljevi koji se eksperimentiranjem trebaju postići su:

- odrediti optimalan skup parametara;
- utvrditi koliko pojedini genetski operator troši procesorskog vremena i skratiti trajanje izvođenja jedne iteracije koliko je god to moguće;
- odrediti ovisnost trajanja optimiranja i kvalitete dobivenog rješenja sekvencijskog GA o veličini populacije, broju iteracija i broju dretvi;
- usporediti svojstva sinkrone i asinkrone verzije GPGA;
- odrediti ovisnost trajanja optimiranja i kvalitete dobivenog rješenja GPGA o broju dretvi.

Prije mjerenja utrošenog procesorskog vremena treba odrediti optimalni skup parametara, jer parametri genetskog algoritma utječu na trajanje optimiranja. Trajanje optimiranja je proporcionalno broju iteracija. Trajanje mutacije je proporcionalno vjerojatnosti mutacije. Nešto je složenija ovisnost potrošnje procesorskog vremena o veličini populacije, jer ovisi o vrsti selekcije. Primjerice, vrijeme trajanja genetskog algoritma s turnirskim selekcijama ne ovisi o veličini populacije.

Postupak eksperimentalnog podešavanja parametra vjerojatnosti mutacije može se odraditi u sljedećim koracima:

1. Za konstantne vrijednosti parametara veličine populacije i broja iteracija treba grubo odrediti područje u kojem genetski algoritam daje najbolje rezultate. Dovoljna preciznost vjerojatnosti mutacije u ovom koraku je u postocima.
2. Za iste konstantne vrijednosti parametara veličine populacije i broja iteracija u intervalu vjerojatnosti mutacije određenom u prethodnom koraku eksperimentalno treba fino podesiti vjerojatnost mutacije. Preciznost parametra vjerojatnosti mutacije je povećana i u ovom koraku se zadaje u promilima.
3. Ukoliko postignuta kvaliteta rješenja nije zadovoljavajuća, povećati veličinu populacije i broj iteracija. Očekuje se manja vrijednost parametra vjerojatnosti mutacije, pa treba smanjiti donju i gornju granicu intervala u kojem se u 2. koraku fino podešavao taj parametar.
4. Ponavljati drugi i treći korak dok kvaliteta rješenja ne postane zadovoljavajuća.

7.2. Podešavanje parametara

7.2.1. Određivanje skupa parametara za podešavanje

Osnovni parametri genetskog algoritma su: veličina populacije (N), duljina kromosoma (b), vjerojatnost mutacije (p_m), vjerojatnost križanja (p_c), selekcijski pritisak i broj iteracija (I). U ovisnosti o modelu genetskog algoritma i vrsti selekcije koja se koristi, ovisi i broj parametara genetskog algoritma. Primjerice, genetski algoritam s eliminacijskom selekcijom umjesto parametra vjerojatnost križanja ima parametar generacijski jaz ili mortalitet (M). Drugi primjer je DGA koji ima, osim osnovnih, čak sedam dodatnih parametara koji utječu na efikasnost paralelnog genetskog algoritma.

Veličina populacije utječe na kvalitetu rješenja i na vrijeme trajanja genetskog algoritma. Ako se koristi k-turnirska selekcija, veličina populacije izravno ne utječe na vrijeme trajanja optimizacije, već utječe na kvalitetu rješenja. Želi li se postići rješenje iste kvalitete s većom populacijom, genetski algoritam treba odraditi više iteracija. Za više iteracija treba, naravno, više vremena. Bez obzira na vrstu selekcije, veličina populacije znatno utječe na kvalitetu rješenja, a time izravno ili neizravno (ovisno o vrsti selekcije) na trajanje optimizacijskog postupka.

Duljina kromosoma jednaka je umnošku dimenzije problema (broja nepoznanica) i broju bitova kod binarnog prikaza. Broj bitova utječe na preciznost rješenja, koje je unaprijed zadano. Taj je parametar unaprijed zadan i ne treba ga podešavati. Ono što treba učiniti jest podesiti ostale parametre genetskog algoritma tako da genetski algoritam postiže u određenom vremenu dovoljno kvalitetno rješenje za unaprijed zadanu veličinu kromosoma.

Genetski algoritam je naročito osjetljiv na vjerojatnost mutacije pa treba posvetiti posebnu pažnju prilikom podešavanja tog parametra. Vjerojatnost križanja ne utječe značajnije na svojstva genetskog algoritma. Štoviše, genetski algoritam koji u istom koraku obavlja i selekciju i reprodukciju nema tog parametra, jer se križanje obavlja u svakoj iteraciji. Smanjenje broja osnovnih parametara je bio jedan od razloga zašto se u predloženom modelu GPGA primjenjuje 3-turnirska eliminacijska selekcija zajedno s reprodukcijom u istoj iteraciji.

Selekcijski pritisak se mjeri uz pomoć selekcijske razlike s, selekcijskog intenziteta s_I, reprodukcijske stope r ili uz pomoć trajanja preuzimanja T. Selekcijski pritisak se kod GA s turnirskom selekcijom podešava uz pomoć parametra k, gdje je k=2,3,...,N. Što je veći k, veći je i selekcijski pritisak. Veći selekcijski pritisak ima za posljedicu i bržu konvergenciju zbog čega algoritam kraće traje, ali ima i veću vjerojatnost zaglavljivanja u lokalnom optimumu. Turnirska selekcija, i s najmanje mogućom vrijednošću parametra k (k_{min}=2), ima veći selekcijski pritisak od proporcionalnih selekcija i rangirajućih selekcija [BAE94]. Stoga, zapravo i nema izbora prilikom odabira

parametra *k*: parametar *k* mora biti što je moguće manji. U slučaju da se u istom koraku obavlja i selekcija i reprodukcija, tri ili više jedinki trebaju sudjelovati u turnirskoj selekciji (jedna jedinka se eliminira i nadomjesti djetetom dvaju slučajno odabranih roditelja od *k*-1 preživjelih jedinki). Dakle, želimo li u istom koraku obavljati i turnirsku selekciju i reprodukciju, minimalna vrijednost parametra *k* je tri.

Posljednji parametar iz osnovnog skupa parametara genetskog algoritma je broj iteracija. Što je veći broj iteracija, očekuje se bolje rješenje. Trajanje izvođenja algoritma proporcionalno je broju iteracija. Za kontradiktorne zahtjeve: postići što je moguće kvalitetnije rješenje u što je moguće kraćem vremenu, treba učiniti kompromis: genetski algoritam treba obaviti *dovoljan* broj iteracija da postigne *zadovoljavajuće* rješenje. Nadalje, često je unaprijed zadano maksimalno vrijeme trajanja algoritma, kao što je to slučaj u primjeni genetskih algoritama u sustavima za rad u stvarnom vremenu [BUD99]. Ako je zadano vrijeme trajanja genetskog algoritma T_{uk}, zapravo je zadan i broj iteracija I, ukoliko je poznato vrijeme trajanja jedne iteracije T_1:

$$I = \frac{T_{uk}}{T_1}. \qquad (7.1)$$

Novi model GPGA ima ista svojstva i isti skup parametara kao i odgovarajući sekvencijski genetski algoritam. Naravno, novi model GPGA i sekvencijski GA razlikuju se u brzini izvođenja na višeprocesorskom računalu. Za rješavanje aproksimacijskog problema korišten je genetski algoritam s binarnim prikazom, uniformnim križanjem s detekcijom duplikata, jednostavnom mutacijom i 3-turnirskom eliminacijskom selekcijom, koji u istom koraku obavlja i selekciju i reprodukciju. Broj parametara sveden je na već opisani način s početnih šest na svega tri: veličinu populacije, vjerojatnost mutacije i broj iteracija. Preostala tri parametra su unaprijed određena: duljina kromosoma je određena sa zadanom preciznošću, vjerojatnost križanja jednaka je jedan za dvije od tri selektirane jedinke i selekcijski pritisak je određen parametrom *k*=3. Skup parametara koji nisu unaprijed određeni problemom ili tipom genetskog algoritma čine: veličina populacije N, broj iteracija I i vjerojatnost mutacije p_m.

7.2.2. Eksperimentalni rezultati podešavanja parametara iz literature

Mnogi autori su na nizu primjera eksperimentalno nastojali odrediti skup parametara genetskog algoritma koji bi se dao primijeniti na širokom spektru problema [DEJ90, GRE86, SRI94]. Pokazalo se da je to nemoguće učiniti, jer se skup optimalnih parametara razlikuje od problema do problema. Naime, ponašanje genetskog algoritma uvelike ovisi o funkciji cilja. Ponašanje GA se ocjenjuje kvalitetom dobivenog rješenja i vremenom izvođenja, odnosno brzinom konvergencije. Ipak, neki autori preporučaju određene skupove parametara kao dobar početak ili pokazatelj gdje treba tražiti optimalni skup parametara.

Tablica 7.1. **Veličina populacije i vjerojatnost mutacije koji se predlažu u literaturi**

Autori i reference	Veličina populacije	Vjerojatnost mutacije
DeJong i Spears [DEJ90, HAR99]	50-100	1‰
Grefenstette [GRE86]	30	1%
Srinivas i Patnaik [SRI94]	100	1‰

Neki su autori išli i korak dalje, pa su na temelju eksperimentalnih rezultata nastojali pronaći zavisnosti među parametrima [GIG98, GOL99]. Rezultat takvih istraživanja su izrazi uz pomoć kojih se mogu odrediti neki parametri tako da algoritam daje zadovoljavajuće rezultate. Međutim, treba naglasiti da izrazi vrijede samo ako se genetski algoritam primijeni za rješavanje nekog problema iz skupa problema na kojima je dotični autor obavljao eksperimente. Tako su, primjerice, Williams i Crossley empirički nastojali u [WIL97] utvrditi pravilnosti za određivanje optimalne veličine populacije i vjerojatnosti mutacije. Obavili su niz eksperimenata na problemu optimiranja dvije dvodimenzijske funkcije realne varijable, šesterodimenzijskoj Rosenbrock-ovoj funkciji, četverodimenzijskoj Griewank-ovoj funkciji i na problemu dizajna konzola tako da mogu podnijeti odgovarajući teret. Rezultat eksperimentiranja je savjet da veličina populacije mora biti barem četiri puta veća od broja bitova kod binarnog prikaza:

$$N \geq 4b. \qquad (7.2)$$

Isti autori predlažu da se vjerojatnost mutacije za poznate parametre N i b računa prema izrazu:

$$p_m = \frac{b+1}{2N \cdot b}. \qquad (7.3)$$

Drugi primjer se navodi u [MUE92b], gdje se citiraju Schafel, Caruana, Eshelman i Das, te se koriste njihove empiričke formule. Spomenuti autori su objedinili tri parametra u jedan izraz koji glasi:

$$\ln N + 0.93 \ln p_m + 0.45 \ln b = 0.56. \qquad (7.4)$$

Jednakost (7.4) se dobro može aproksimirati izrazom:

$$N \cdot p_m \cdot \sqrt{b} = 1.7. \qquad (7.5)$$

Već prema tablici 7.1 se nazire da za veće populacije vjerojatnost mutacije treba biti manja. Isti zaključak se može izvući i na temelju izraza (7.3), (7.4) i (7.5). Uzimajući u obzir samo činjenicu da za veće populacije treba vjerojatnost mutacije biti manja, treba obaviti manje eksperimenata prilikom podešavanja parametara.

U ovom radu prilikom podešavanja parametara nisu se uzimali u obzir spomenuti savjeti i izrazi. Stoga je obavljen veliki broj eksperimenata kako bi se eksperimentalno provjerile navedene pravilnosti.

7.2.3. Određivanje vremena trajanja preuzimanja

Trajanje preuzimanja je prosječan broj iteracija nakon kojih se, primjenom samo operatora selekcije, populacija sastoji samo od najbolje jedinke ili određenog broja najboljih jedinki iz početne populacije. Selekcijski pritisak se može mjeriti i s pomoću selekcijske razlike, intenziteta ili s pomoću reprodukcijske stope, ali najjednostavnije je odrediti trajanje preuzimanja. Naime, želi li se eksperimentalno odrediti trajanje preuzimanja, izvorni tekst programa genetskog algoritma treba neznatno izmijeniti. Treba izbaciti (staviti u komentar) sve genetske operatore osim selekcije, te detektirati generaciju kada je populacija uniformna (kada se sastoji od samo jedne – najbolje jedinke) i ispisati redni broj te generacije. Prosječan broj generacija, kada je populacija postala uniformna, je trajanje preuzimanja.

Trajanje preuzimanja za *k*-turnirsku selekciju izračunali su Goldberg i Deb [BAE94, CAN99c]:

$$T = \frac{\ln N + \ln(\ln N)}{\ln k}. \qquad (7.6)$$

Neka je u generaciji t udio dobrih jedinki u populaciji jednak P_t, a Q_t=1-P_t neka je udio loših jedinki. Drugim riječima, P_t je kvocijent broja dobrih jedinki i veličine populaciju u trenutku t. Primjenom k-turnirske selekcije loša jedinka će preživjeti samo onda ako su sve slučajno odabrane jedinke iz skupa loših jedinki $Q_{t+1} = Q_t^k$, odnosno $Q_t = Q_0^{k^t}$.

Prema tome, udio dobrih jedinki u trenutku t iznosi:

$$P_t = 1 - (1 - P_0)^{k^t}. \qquad (7.7)$$

Za izvođenje izraza (7.6) iz izraza (7.7), Goldberg i Deb su pretpostavili sljedeće:

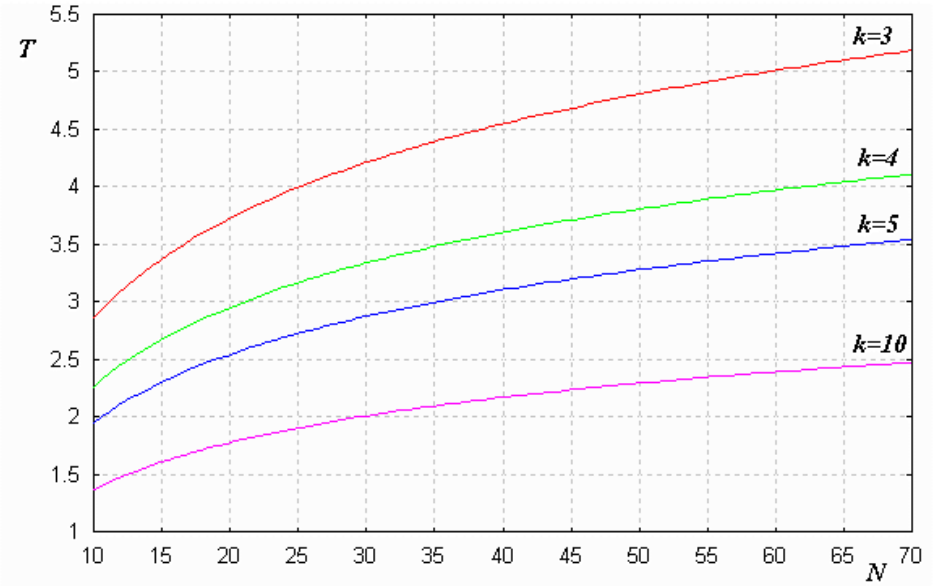
- početna (inicijalna) populacija se sastoji od samo jedne dobre jedinke: P_0=1/N,
- populacija u generaciji T se sastoji od N-1 dobrih jedinki: P_T=1-1/N i
- populacije se sastoji od velikog broja jedinki: N>> tako da je ln(1-1/N)≈-1/N.

Uzevši u obzir navedene pretpostavke, izraz (7.7) poprima oblik:

$$\frac{N-1}{N} = 1 - \left(1 - \frac{1}{N}\right)^{k^T} \Rightarrow k^T = \log_{\left(1-\frac{1}{N}\right)} \frac{1}{N} = \frac{\ln \frac{1}{N}}{\ln\left(1 - \frac{1}{N}\right)} = \left|\ln\left(1 - \frac{1}{N}\right) \approx \frac{1}{N}\right| = N \ln N$$

iz čega se dobiva izraz (7.6) za trajanje preuzimanja:

$$T = \log_k (N \ln N) = \frac{\ln N + \ln \ln N}{\ln k}. \blacksquare$$



Slika 7.1. **Trajanje preuzimanja kao funkcija veličine populacije *N* i veličine turnira *k***

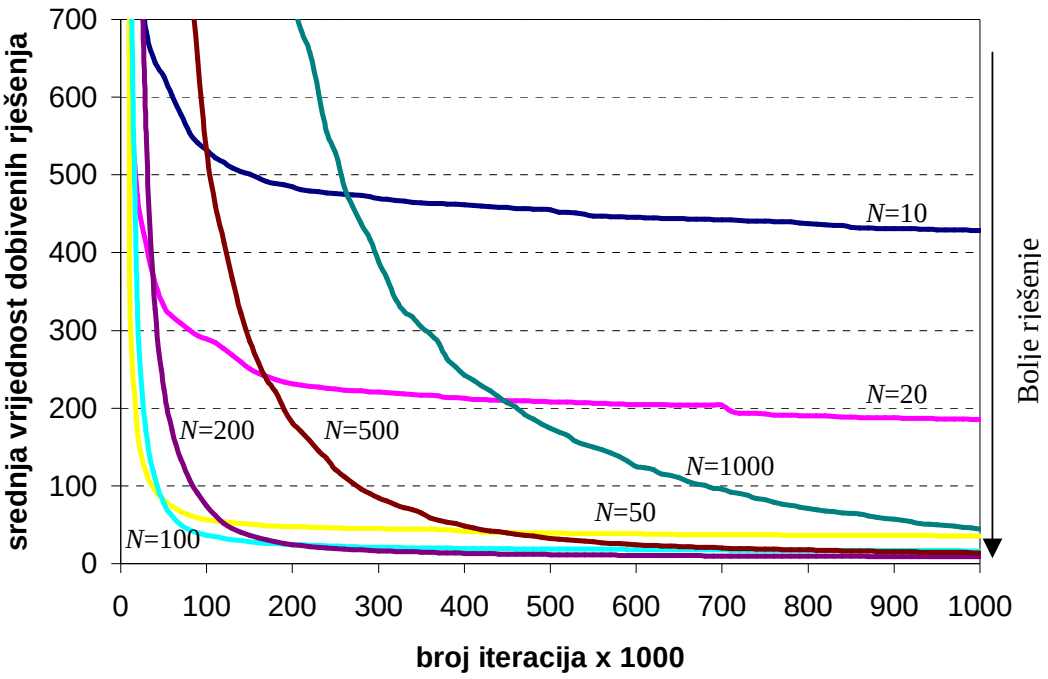
Na slici 7.1 prikazano je trajanje preuzimanja za turnirsku selekciju prema izrazu (7.6) i to za veličine turnira k=3,4,5,10 i za veličine populacije od 10 do 70. Selekcijski pritisak je veći što je trajanje preuzimanja manje. Prema tome, selekcijski pritisak raste kako se smanjuje veličina populacije i kako raste parametar k.

7.2.4. Primjer eksperimentalnog podešavanja veličine populacije i broja iteracija

Tablica 7.2. **Srednja vrijednost dobivenih rješenja za određeni broj iteracija i veličinu populacije**

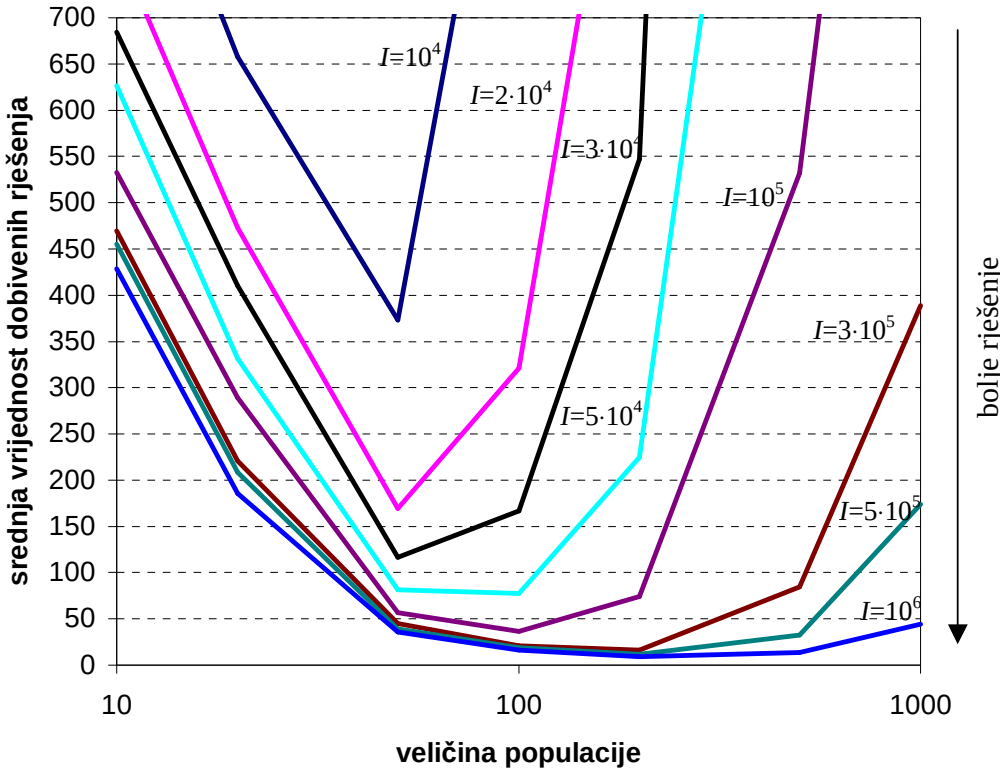
broj iteracija	veličina populacije						
	10	20	50	100	200	500	1000
20000	777.9	472.9	169.4	321.1	1089.2	9923.4	17687.8
50000	626.5	331.4	81.2	77.4	224.7	1458.1	9130.8
70000	578.7	309.3	66.5	50.0	134.6	944.8	3254.9
100000	532.4	288.8	56.7	36.3	74.2	532.1	1580.2
200000	485.2	231.3	47.6	24.5	24.5	181.4	727.3
300000	469.6	220.5	44.8	20.9	16.4	84.7	388.5
500000	455.4	208.1	39.2	18.6	11.6	32.4	174.0
700000	442.1	203.4	37.0	17.3	9.9	19.8	95.5
1000000	428.2	185.5	35.7	16.5	8.9	13.8	44.5

Rezultati iz tablice 7.2 prikazani su na dva načina na slikama 7.2 i 7.3. Na slici 7.2 prikazana je ovisnost dobivene kvalitete rješenja o broju iteracija. Na toj slici prikazana je cijela evolucija rješenja za 10^6 iteracija. Kvaliteta rješenja je veća što je srednja vrijednost dobivenih rješenja manja. Slika 7.3 prikazuje ovisnost kvalitete rješenja o veličini populacije.



Slika 7.2. **Evolucija rješenja u ovisnosti o veličini populacije**

Za premale veličine populacije (10 i 20 jedinki), genetski algoritam konvergira k lokalnom optimumu i tu ostaje, tj. povećavanje broja iteracija ne pomaže. Za veličine populacije od 50 do 100 jedinki genetski algoritam znatno brže konvergira, a i dobivena rješenja su zadovoljavajuća već za 110000 iteracija i veličinu populacije N=50, odnosno za 70000 iteracija i veličinu populacije N=100. Naravno, što je veća populacija, pronalaze se bolja rješenja. Povećava li se još populacija, konvergencija je sporija, treba više iteracija, ali je vjerojatnost zaglavljivanja u lokalnom optimumu manja i za *dovoljan* broj iteracija postiže se bolje rješenje. Broj iteracija 10^6 je već za veličinu populacije od 500 jedinki premalo, jer genetski algoritam s 200 jedinki postiže za 10^6 iteracija bolje rješenje. Prema tome, za veće populacije se postiže bolje rješenje, ali za znatno veći broj iteracija (slika 7.3). Primjerice, za konstantan broj iteracija I=30000, optimalna veličina populacije je oko 50 jedinki. Za veće populacije, primjerice, od 100 jedinki, treba odraditi barem 50000 iteracija ili još bolje 100000 iteracija. S druge strane, ne isplati se povećavati broj iteracija, a veličinu populacije ostaviti na istoj vrijednosti. Primjerice, ako se već odradi 300000 iteracija, onda je veličina populacije od 100 jedinke premala, jer daje lošije rješenje od genetskog algoritma s 200 jedinki (slika 7.3).



Slika 7.3. **Kvaliteta rješenja u ovisnosti o veličini populacije i broju iteracija**

Vjerojatnost mutacije se za vrijeme eksperimentiranja držala konstantnom. Za male populacije iznosila je 1.2%, a za veće 1%. Za veće populacije se mogu postići još nešto bolja rješenja, ali finim podešavanjem vjerojatnosti mutacije, koje se postiže opisan postupkom u prethodnom poglavlju.

7.2.5. Optimalan skup parametara

Veličina populacije *N* i broj iteracija *I* su međusobno zavisni. Za veću populaciju treba odraditi više iteracija da bi se postiglo rješenje iste kvalitete. Osim što se za veće populacije treba obaviti više iteracija i vjerojatnost mutacije mora biti manja. Prema tome, svi su parametri genetskog algoritma međusobno zavisni. Drugim riječima, promjenom jednog parametra moraju se podesiti svi ostali parametri, želi li se zadržati ista kvaliteta dobivenog rješenja.

Iz prethodnog poglavlja bi se moglo zaključiti da je optimalan skup parametara beskonačno velika populacija i beskonačno velik broj iteracija bez mutacije, tj. za vjerojatnost mutacije jednako nuli. Od takvog genetskog algoritma nema koristi, jer je za njegovo izvođenje potrebno beskonačno vremena. Budući je vrijeme ograničeno (najčešće je unaprijed zadano), zadan je i broj iteracija, jer je utrošeno vrijeme za optimiranje proporcionalno broju iteracija (izraz (7.1)).

Tablica 7.3. **Predloženi skupovi parametara**

Broj iteracija ***I***	Veličina populacije ***N***	Vjerojatnost mutacije p_m
10 000 – 50 000	50	0.012
50 000 – 70 000	70	0.010
70 000 – 150 000	100	0.009
200 000 – 500 000	200	0.008

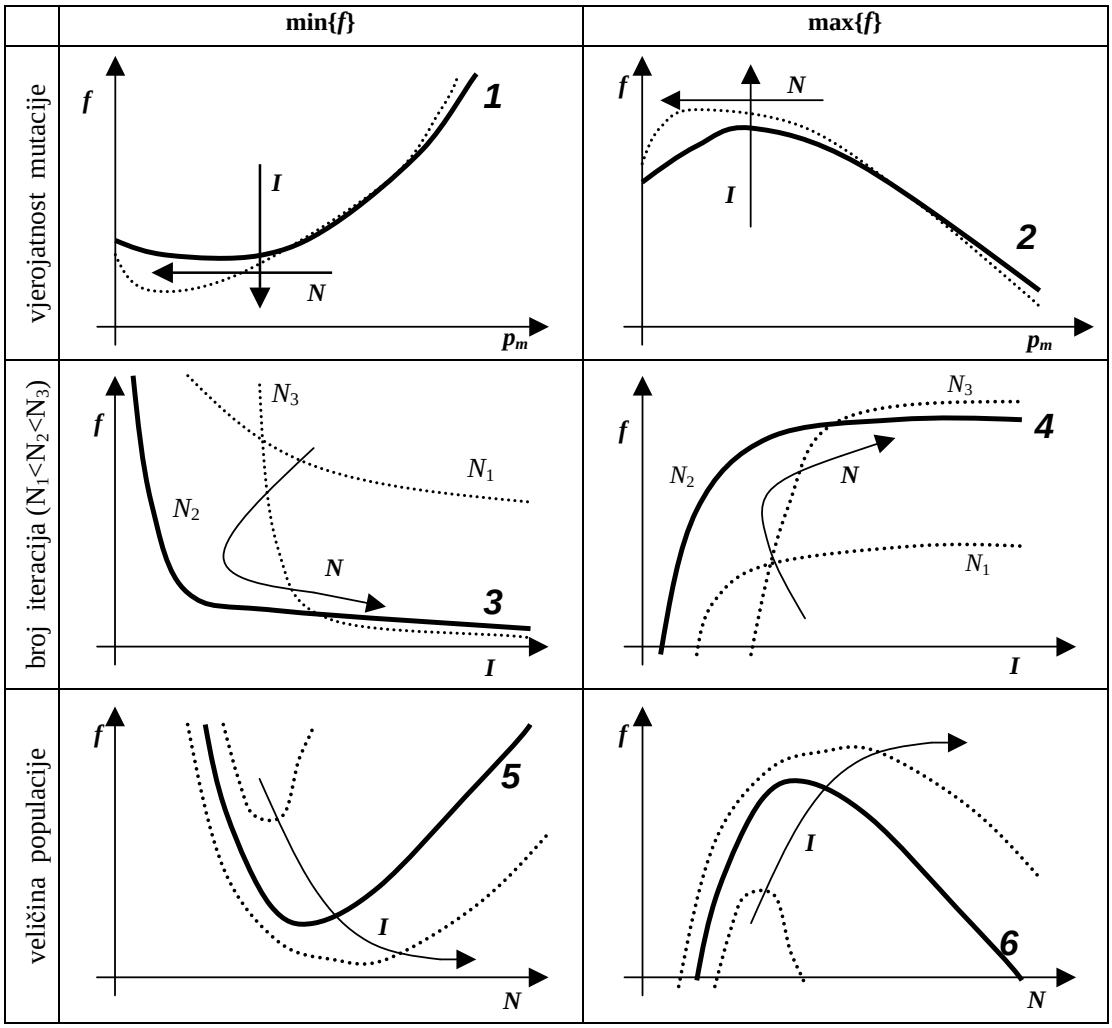
7.2.6. Karakteristične krivulje kvalitete rješenja

Kvaliteta dobivenog rješenja za dva slična skupa parametara može biti bitno različita, jer je genetski algoritam osjetljiv na svaku promjenu parametara. Rezultati optimiranja se mogu znatno poboljšati finim podešavanjem parametara. Dakle, optimalni skupovi parametara iz literature mogu poslužiti samo kao smjernice gdje treba tražiti optimalno skup parametara za određen genetski algoritam primijenjen na određenom optimizacijskom problemu.

Jednako tako, da bi se izbjeglo dugotrajno eksperimentalno određivanje parametara, treba iskoristiti sljedeće pravilnosti u ponašanju genetskog algoritma:

- povećavanjem broja iteracija se u pravilu postiže bolje rješenje;
- za veće populacije, treba obaviti više iteracija, a vjerojatnost mutacije treba smanjiti;
- jednako kvalitetna rješenja se mogu postići za više skupova parametara.

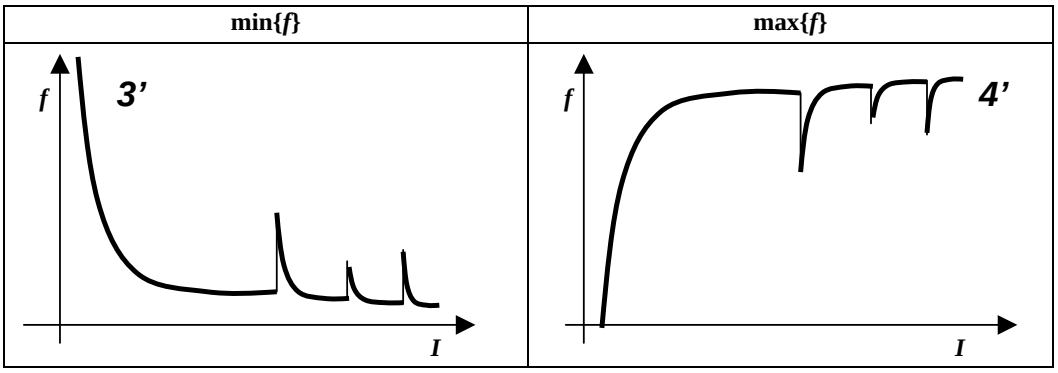
Poželjno je unaprijed znati broj iteracija i za poznati broj iteracija eksperimentalno odrediti veličinu populacije i vjerojatnost mutacije. Obično je unaprijed poznato raspoloživo vrijeme za optimiranje, a time i broj iteracija. Ukoliko se s tako određenim skupom parametara ne postiže zadovoljavajuće rješenje, broj iteracija se mora povećati. S druge strane, ako se s dobivenim skupom parametara postiže zadovoljavajuće rješenje, treba pokušati i s manjim brojem iteracija. Tako sve dok se ne dobije rješenje koje ne zadovoljava. Na taj način je moguće odrediti gornju i donju granicu vremena izvođenja, odnosno broja iteracija.



Slika 7.4. **Karakteristične krivulje kvalitete rješenja u ovisnosti o parametrima GA**

Na slici 7.4 prikazano je šest karakterističnih krivulja ovisnosti rješenja optimiranja o tri osnovna parametra: veličini populacije, broju iteracija i vjerojatnosti mutacije. Crtkane linije predstavljaju novu krivulju ako se promijeni neki od parametara. Strelica na slici označava smjer promjene krivulje ukoliko se poveća naznačeni parametar. Primjerice, na krivulji ovisnosti rješenja o veličini populacije, u slučaju da se traži minimum funkcije *f* (krivulja 5 na slici 7.4), povećanjem broja iteracija krivulja se širi, a dno krivulje se pomiče dolje desno.

Krivulje su skicirane na temelju niza eksperimentalnih rezultata iz [GOL01]. U literaturi su spomenute krivulje često koriste kao sredstvo s pomoću kojeg se uspoređuju razni tipovi genetskih algoritama. Najčešće se prikazuje ovisnost rješenja o broju iteracija (krivulje 3 ili 4) [ARA99, BAE93, BAE95b, CAN99b, CAN99c, COR92, CRA92, ELB96, HAR94, HAR99, KAR93, KEN98, MAI94, MIL95, MUE98b, MUN93, SAR96, SCH92, TAL91, TAL92, TYN99]. Na slici 7.4 krivulje se odnose na genetski algoritam s ugrađenim elitizmom. Ukoliko elitizam nije ugrađen, krivulje ovisnosti rješenja o broju iteracija imaju specifičan zupčasti oblik pri kraju evolucijskog procesa koji je prikazan na slici 7.5 [CAR93, FOR93, GOH99, GOO97, PAN95, YOS99]. Zupčasti oblik je posljedica gubljenja pronađenog rješenja, ukoliko se ono ne zaštiti elitizmom.



Slika 7.5. **Karakteristična krivulja rješenja u ovisnosti o broju iteracija za genetski algoritam bez elitizma**

Nadalje, ovisnost kvalitete rješenja o veličini populacije prikazan je u [LIN97]. Prema navedenim referencama i na temelju cijelog niza eksperimenata provedenih na različitim optimizacijskim problemima, čiji su rezultati objavljeni u [BUD98, GOL96, GOL97, GOL98a, JAK98], zamijećena je sljedeća pravilnost: krivulje zadržavaju svoj oblik bez obzira na optimizacijski problem. Međutim, iste krivulje skicirane na temelju eksperimentalnih rezultata dva različita optimizacijska problema jesu slične po obliku, ali se razlikuju po vrijednostima optimalnih skupova parametara.

8. ZAKLJUČAK

Višeprocesorska računala s nekoliko (najčešće dva ili četiri) procesora i sa zajedničkim radnim spremnikom su danas najrasprostranjenija paralelna računala. Analizom postojećih paralelnih modela genetskih algoritama pokazalo se da je globalni paralelni genetski algoritam (GPGA) najpogodniji za izvođenje na takvim računalima. Međutim, tradicionalni model GPGA obavlja paralelno samo jedan dio genetskog algoritma. Ostatak genetskog algoritma obavlja se sekvencijski. Povrh toga, paralelni dio posla se obavlja naizmjenično sa sekvencijskim dijelom, pa te poslove treba nekako sinkronizirati što dodatno usporava paralelni genetski algoritam. Stoga je u ovom dijelu opisan specifičan oblik GPGA prilagođen višeprocesorskim računalima sa zajedničkom memorijom. Zasebno se razmatra sikrona i asinkrona verzija GPGA. Opisani model GPGA koristi 3-turnirsku eliminacijsku selekciju bez duplikata s inherentno ugrađenim elitizmom, uniformno križanje s detekcijom i otklanjanjem duplikata te jednostavnu mutaciju.

Razlika između tradicionalnog GPGA i predloženog GPGA je u poslu kojeg gospodar i sluge obavljaju. Kod tradicionalnog GPGA sluge obavljaju samo evaluaciju i to samo onih jedinki koje im pošalje gospodar. Gospodar obavlja sve ostale genetske operatore i raspoređuje posao slugama. Nadalje, selekcija se ne može obavljati sve dok se nije obavila evaluacija barem dijela jedinki ili čak cijele populacije. Stoga se posao gospodara i slugu treba nekako sinkronizirati. S obzirom da za mnoge optimizacijske probleme sav taj posao koji obavlja gospodar nije zanemariv u odnosu na posao evaluacije, jasna je potreba za nekim novim modelom PGA koji bi i taj dio posla gospodara obavljao paralelno. Kod novog modela GPGA gospodar obavlja samo inicijalizaciju populacije i stvara potreban broj dretvi-slugu, koje obavljaju *sve* genetske operatore paralelno. Nema potrebe za nekim dodatnim mehanizmom međusobne komunikacije između gospodara i slugu te slugu međusobno, jer je populacija zajednička i nalazi se u zajedničkom radnom spremniku.

Predloženi model GPGA objedinjuje dobra svojstva tradicionalnog GPGA i distribuiranog genetskog algoritma. Zajedničko dobro svojstvo novog i tradicionalnog modela GPGA je to što su oba modela pogodna za izvođenje na višeprocesorskom računalu sa zajedničkim radnim spremnikom. Dobra svojstva DGA, a koja su na neki način preslikana na novi model GPGA, jesu: svi genetski operatori se obavljaju paralelno i populacija je podijeljena. Samo se inicijalizacija ne obavlja paralelno, ali ona ionako zanemarivo kratko traje i izvodi se samo prije početka evolucijskog procesa. Zatim, populacija nije podijeljena na razdvojene subpopulacije, već na skupine od tri jedinke i to u svakoj iteraciji druge tri jedinke. Subpopulacije nisu statične kao kod DGA, već se jedinke, koje čine subpopulaciju, dinamički izmjenjuju. Na taj način je izbjegnut operator migracije, a s njim i svi dodatni parametri. Novi model GPGA nema niti jednog dodatnog parametra. Ima svega tri osnovna parametra: veličinu populacije, vjerojatnost mutacije i broj iteracija.

Analizom raznih postupaka selekcije pokazalo se da je 3-turnirska eliminacijska selekcija najpogodnija, ne samo za paralelno izvođenje, već i za objedinjavanje selekcije i reprodukcije u jednu cjelinu. 3-turnirska eliminacijska selekcija odabire *lošu* jedinku za eliminaciju i dvije *bolje* jedinke za križanje. Doduše, dretva obavlja sekvencijski selekciju i reprodukciju u svakom koraku, ali taj isti posao može obavljati paralelno i neka druga dretva nad neke druge tri jedinke. Na taj način se ipak svi genetski operatori obavljaju paralelno, slično kao i kod DGA.

Prilikom donošenja odluke da li koristiti predloženi model GPGA za rješavanje nekog optimizacijskog problema trebaju se uzeti u obzir sljedeće činjenice:

- paralelizacija ima smisla kada se s pomoću genetskog algoritma nastoje riješiti složeniji problemi (jer je obično za rješavanje jednostavnih optimizacijskih problema sekvencijski algoritam dovoljno brz);
- složeniji problemi zahtijevaju veće populacije;
- novi model GPGA je prilagođen za višeprocesorski sustav sa svega nekoliko procesora, tj. broj procesora je puno manji od veličine populacije ($N_P << N$), jer je u suprotnom masovni paralelni genetski algoritam najpogodniji za implementaciju;
- sinkroni GPGA poziva jezgrine funkcije operacijskog sustava za međusobno isključivanje dretvi pa je instrukcijska dretva nešto duža od instrukcijske dretve asinkronog GPGA i može se dogoditi da ne stane cijela u priručni spremnik procesora.

Prema tome, za rješavanje složenijih optimizacijskih problema (koji zahtijevaju veće populacije) na raspoloživim paralelnim računalima (s dva ili najviše četiri procesora) pogodniji je asinkroni GPGA. Ako je na raspolaganju računalo s više procesora od četiri ili su populacije manje veličine (manje od 100) pogodniji je sinkroni GPGA.

Prednosti novog modela GPGA nad ostalim paralelnim modelima su: jednostavan je za implementaciju; elitizam je inherentno ugrađen; svi genetski operatori se obavljaju paralelno; nema potrebe za komunikacijskim mehanizmom jer dretve razmjenjuju podatke preko zajedničkog radnog spremnika; radi sa i bez sinkronizacije i program se može izvoditi na računalnom sustavu s proizvoljnim brojem procesora bez izmjene izvornog teksta programa. Nedostaci su: novi model GPGA se ne može ostvariti s proporcionalnim i sortirajućim selekcijama, sinkroni GPGA dodatno troši vrijeme za sinkronizaciju, a asinkroni na odrađivanje dodatnih iteracija. Međutim, broj dodatnih iteracija za asinkroni GPGA je moguće predvidjeti na način kako je to opisano u poglavlju 6.3.

POPIS LITERATURE

[ABR92] Abramson, D., Abela, J., *A Parallel Genetic Algorithm for Solving the School Timetabling Problem*, 15 Australian Computer Science Conference, Hobart, Feb. 1992. Dostupno na Internet adresi: http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Abramson.Abela.A_Parallel_Genetic_Algorithm_for_Solving_the_School_Timetabling_Problem.ps.gz.

[ALT94] Altenberg, L., *The Schema Theorem and Prices Theorem*, 1994. Dostupno na Internet adresi: http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Altenburg.The_Schema_Theorem_and_Prices_Theorem.ps.gz.

[ARA99] Arabas, J., Miazga, P., *Computer Aided Design of a Layout of Planar Circuits by Means Evolutionary Algorithms*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 1-18, March 1999.

[BAE93] Bäck, T., Schwefel, H.-P., *An Overview of Evolutionary Algorithms for Parameter Optimization*, Evolutionary Computation 1(1), pp. 1-23, 1993. Dostupno na Internet adresi: http://ls11-www.informatik.uni-dortmund.de/people/baeck/papers/ec93.ps.gz.

[BAE94] Bäck, T., *Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms*, Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, Piscataway NJ, pp. 57-62, 1994. Dostupno na Internet adresi: http://ls11-www.informatik.uni-dortmund.de/people/baeck/papers/wcci94-sel.ps.gz.

[BAE95a] Bäck, T., *Generalized Convergence Models for Tournament- and (μ,λ)-Selection*, Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, pp. 2-8, 1995.

[BAE95b] Bäck, T., Beielstein, T., Naujoks, B., Heistermann, J., *Evolutionary Algorithms for the Optimization of Simulation Models Using PVM*, EuroPVM '95: Second European PVM User's Group Meeting, pp. 277-282, Hermes, Paris, 1995.

[BAE95c] Bäck, T., *Order Statistics for Convergence Velocity Analysis in Simplified Evolutionary Algorithms*, Foundations of Genetic Algorithms, Vol. 3, Morgan Kaufmann, pp. 91-102, San Mateo CA, 1995.

[BAU95] Baum, E.B., Boneh, D., Garrett, C., *On genetic algorithms*, Computational learning theory, pp. 230-239, 1995.

[BEA91] Beaty, S.J., *Genetic Algorithms and Instruction Scheduling*, Microarchitecture, pp. 206-211, 1991.

[BLI95] Blickle, T., Thiele, L., *A Mathematical Analisys of Tournament Selection*, Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, pp. 2-8, 1995.

[BUD96] L. Budin, M. Golub, A. Budin, *Traditional Techniques of Genetic Algorithms Applied to Floating-Point Chromosome Representations*, Proceedings of the 41st Annual Conference KoREMA, Opatija, 1996, pp.93-96.

[BUD98] Budin, L., Golub, M., Jakobović, D., *Parallel Adaptive Genetic Algorithm*, International ICSC/IFAC Symposium on Neural Computation NC'98, Vienna, pp. 157-163, 1998.

[BUD99] Budin, L., Jakobović, D., Golub, M., *Genetic Algorithms in Real-Time Imprecise Computing*, IEEE International Symposium on Industrial Electronics ISIE'99, Bled, Vol. 1, pp. 84-89, 1999.

[BUI94] Bui, T.N., Moon, B.R., *A Fast and Stable Hybrid Genetic Algorithm for the Ratio-Cut Partitioning Problem on Hypergraphs*, Design automation, pp. 664-669, 1994.

[BUR97] Burke, E.K., Varley, D.B., *A Genetic Algorithms Tutorial Tool for Numerical Function Optimisation*, Integrating technology into computer science education, pp. 27–30, 1997.

[CAN95] Cantú-Paz E., *A Summary of Research on Parallel Genetic Algorithms*, 1995. Dostupno na Internet adresi: http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Cantu-Paz.A_Summary_of_Research_on_Parallel_Genetic_Algorithms.ps.gz.

[CAN98a] Cantú-Paz, E., *A Survey of Parallel Genetic Algorithms*, Calculateurs Paralleles, Vol. 10, No. 2. Paris: Hermes, 1998. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/Publications/cantupaz/survey.ps.Z.

[CAN98b] Cantú-Paz, E., *Designing Scalable Multi-population Parallel Genetic Algorithms*, 1998. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/98009.ps.Z.

[CAN98c] Cantú-Paz, E., *Designing Efficient Master-slave Parallel Genetic Algorithms*, Genetic Programming: Proceedings of the Third Annual Conference, San Francisco, CA, pp. 455-462, 1998.

[CAN99a] Cantú-Paz, E., Goldberg, D.E., *Parallel Genetic Algorithms with Distributed Panmictic Populations*, 1999. Dostupno na Internet adresi: http://www-illigal.ge.uiuc.edu/cgi-bin/orderform/orderform.cgi.

[CAN99b] Cantú-Paz, E., *Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms*, 1999. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99015.ps.Z.

[CAN99c] Cantú-Paz, E., *Migration Policies and Takeover Times in Parallel Genetic Algorithm*, 1999. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99008.ps.Z.

[CAN99d] Cantú-Paz, E., *Topologies, Migration Rates and Multi-population Parallel Genetic Algorithms*, 1999. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99007.ps.Z.

[CAR93] Carte, B., Park, K., *How good are GAs at finding large cliques: an experimental study*, 1993. Dostupno na Internet adresi: ftp://cs-ftp.bu.edu/techreports/93-015-ga-clique.ps.Z.

[CHI99] Chimiak, J., Kolodziejczyk, J., Opaka, M., *Synthesis of Combinational Binary Circuits Based on GA with Entropy Evaluation*, Proceedings of EUROGEN99 – Short Course on Evolutionary Algorithms in Engineering and Computer Science, Jyvaskyla, Finska, 1999.

[COR92] Corcoran, A.L., Wainwright, R.L., *A Genetic Algorithm for Packing in Three Dimensions*, Applied computing, Vol. II, Technological challenges of the 1990's, pp. 1021-1030, 1992.

[CRA92] Crawford, K.D., *Solving the n-Queens Problem Using Genetic Algorithms*, Applied computing (vol. II) technological challenges of the 1990's, pp. 1039-1047, 1992.

[DAV91] Davis, L., "*Handbook of Genetic Algorithms*", Van Nostrand Reinhold, New York, 1991.

[DAS97] Das, S.K., Sen, S.K., *A New Location Update Strategy for Cellular Networks and its Implementation using a Genetic Algorithm*, Mobile computing and networking, pp. 185–194, 1997.

[DEJ90] DeJong, K.A., Spears, Q.M., *An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms*, Proceedings First Workshop Parallel Problem Solving from nature, Springer-Verlag, Berlin, pp. 38-47, 1990.

[DIC97] Dick, R.P., Jha, N.K., *MOGAC: A Multiobjective Genetic Algorithm for the Co-Synthesis of Hardware-Software Embedded Systems*, Computer-aided design, pp. 522-529, 1997.

[EBY97] Eby, D., Averill, R. C., Gelfand, B., Punch, W.F., Mathews, O., Goodman, E.D., *An Injection Island GA for Flywheel Design Optimization*, Invited Paper, Proceedings EUFIT '97, – 5th European Congress on Intelligent Techniques and Soft Computing, forthcoming, Sept. 1997. Dostupno na Internet adresi: http://garage.cps.msu.edu/papers/GARAGe97-05-04.ps.

[EIC93] Eick, C.F., Jong, D., *Learning Bayesian Classification Rules through Genetic Algorithms*, Information and knowledge management, pp. 305-313, 1993.

[ELB96] Elbaum, R., Sidi, M., *Topological design of local-area networks using genetic algorithms*, IEEE/ACM Trans. Networking 4, pp. 766–778, 1996.

[ESB92] Esbensen, H., *A Genetic Algorithm for Macro Cell Placement*, European Design Automation, pp. 52-57, 1992.

[FOG99] Fogel, D.B., *Some Recent Important Foundational Results in Evolutionary Computation*, članak u knjizi "Evolutionary Algorithms in Engineering and Computer Science", Miettinen, K., Neittaanmaki, P., Makela, M.M., Periaux, J., John Wiley & sons, LTD, 1999.

[FOR93] Forrest, S., Javornik, B., Smith, R.E., Perelson, A.S., *Using genetic algorithms to explore pattern recognition in the immune system*, Evolutionary Computation, Vol. 1, No. 3, pp. 191-211, 1993. Dostupno na Internet adresi: ftp://ftp.cs.unm.edu/pub/forrest/immune-92.ps.gz.

[FOR96] Forrest, S., *Genetic algorithms*, ACM Computer Surveys, 28, 1, p.p. 77 – 80, March, 1996.

[FOS95] Foster, J.A., *Genetic Algorithm Hardness and Approximation Complexity: A Research Agenda*, Technical Report No. LAL 95-04, 1995. Dostupno na Internet adresi: ftp://ftp.cs.uidaho.edu/pub/foster/papers/tr-ga-complexity.ps.gz.

[GIG98] Giuere, P., Goldberg, D.E., *Population Sizing for Optimum Sampling with Genetic Algorithms: A Case Study of the Onemax Problem*, Proceedings of the Third Annual Programming Conference, Madison, WI, July 22-25, 1998.

[GOH99] Goh, G.K.M., Foster, J.A., *Evolving Molecules for Drug Design Using Genetic Algorithms*, 1999. Dostupno na Internet adresi: ftp://ftp.cs.uidaho.edu/pub/foster/papers/molecules-cec99.pdf.

[GOL89] Goldberg, D.E., "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley, 1989.

[GOL94] Goldberg, D.E., *Genetic and Evolutionary Algorithms Come of Age*, Commun. ACM 37, pp. 113–119, 1994. Dostupno na Internet adresi: www.acm.org/dl.

[GOL96] Golub, M., *Vrednovanje uporabe genetskih algoritama za aproksimaciju vremenskih nizova*, magistarski rad, Zagreb, 1996. Dostupno na Internet adresi: http://www.zemris.fer.hr/~golub/magrad.html.

[GOL97] Golub, M., Budin Posavec, A., *Using Genetic Algorithms for Adapting Approximation Functions*, Proceedings of the 19th International Conference ITI'97, Pula, 1997, pp. 451-456.

[GOL98a] Golub, M., Jakobović, D., *A Few Implementations of Parallel Genetic Algorithm*, Proceedings of the 20th International Conference ITI'98, Pula, pp. 507-512, 1998.

[GOL98b] Goldberg, D.E., *A meditation on the application of genetic algorithms*, 1998. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/98003.ps.Z.

[GOL99] Goldberg, D.E., *Using time efficiently: Genetic-evolutionary algorithms and the continuation problem*, 1998. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99002.ps.Z.

[GOL01] Golub, M., *Poboljšavanje djelotvornosti paralelnih genetskih algoritama*, doktorska disertacija, Zagreb, 2001.

[GOO97] Goodman, E.D., Averill, R.C., Punch, W.F., Eby, D.J., *Parallel Genetic Algorithms in the Optimization of Composite Structures*, Second World Conference on Soft Computing (WSC2), June, 1997. Dostupno na Internet adresi: http://garage.cps.msu.edu/papers/GARAGe97-05-02.ps.

[GRE86] Grefenstette, J.J., *Optimization of Control Parameters for Genetic Algorithms*, IEEE Trans. Systems, Man and Cybernetics, Vol. SMC-16, No.1, Jan./Feb., pp. 122-128, 1986.

[GRE93] Grefenstette, J.J., *Genetic algorithms and machine learning*, Computational learning theory , pp. 3-4, 1993. Dostupno na Internet adresi: www.acm.org/dl.

[HAA99] Haataja, J., *Using Genetic Algorithms for Optimization: Technology Transfer in Action*, članak u knjizi "Evolutionary Algorithms in Engineering and Computer Science", Miettinen, K., Neittaanmaki, P., Makela, M.M., Periaux, J., John Wiley & sons, LTD, 1999.

[HAR91] Hart, W.E., Belew, R. K., *Optimizing an arbitrary function is hard for Genetic Algorithms*, Proceedings Fourth Intl. Conf. on Genetic Algorithms, p.p. 190-195, Los Altos, CA, Morgan-Kaufman, 1991.

[HAR94] Hart, W.E., Kammeyer, T.E., Belew, R.K., *The role of development in genetic algorithms*, Foundations of Genetic Algorithms III, Morgan Kauffman, 1994.

[HAR99] Harik, G., Lobo, F., *A parameter-less genetic algorithm*,1999. Dostupno na Internet adresi: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99008.ps.Z.

[JAC97] Jackson, D., Fovargue, A., *The use of animation to explain genetic algorithms*, Computer science education, p.p. 243 – 247, 1997.

[JAK98] Jakobović, D., Golub, M., *Adaptive Genetic Algorithm*, Proceedings of the 20th International Conference ITI'98, Pula, pp. 519-524, 1998.

[JON93] Jones, T., Rawlins, G.J.E., *Reverse Hillclimbing, Genetic Algorithms and the Busy Beaver Problem*, Proceedings of the Fifth International Conference on Genetic Algorithms, 17-21 July, p.p. 70-83, 1993.

[KAR93] Karr, C.L., *Genetic Algorithms for Modelling, Design, and Process Control*, Information and knowledge management, p.p. 233-238, 1993.

[KAV97] Kavian, M., McLenaghan, R.G., Geddes, K.O., *Application of Genetic Algorithms to the Algebraic Simplification of Tensor Polynomials*, Symbolic and algebraic computation , pp. 93–100, 1997.

[KEN98] Kennedy, J., Spears, W.M., *Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator*, IEEE International Conference on Evolutionary Computation, pp. 78-83, 1998.

[KRU95] Kruiskamp, W., Leenaerts, D., *DARWIN: CMOS opamp Synthesis by means of a Genetic Algorithm*, Proceedings of the 32nd ACM/IEEE conference on Design automation, pp. 433-438, 1995.

[LIN97a] Lin, S.C., Goodman, E.D., Punch, W.F., *Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems*, Evolutionar Programming VI, Proceedings Sixth Internat. Conf., EP97, Springer Verlag, NY, P. J. Angeline, et al., eds.,Indianapolis, pp. 383-394, June, 1997.

[LIN97b] Lin, S.C., Goodman, E.D., Punch, W.F., *A Genetic Algorithm Approach to Dynamic Job-Shop Scheduling Problems*, Proceedings Seventh Internat. Conf. on Genetic Algorithms, Morgan Kaufmann Publishers, San Francisco, pp. 481-488, July, 1997.

[LOG92] Logar, A.M., Corwin, E.M., English, T.M., *Implementation of massively parallel genetic algorithms on the MasPar MP-1*, Applied computing (vol. II) technological challenges of the 1990's, pp. 1015-1020, 1992.

[MAC96] Macready, W.G., Wolpert, D.H., *What Makes an Optimization Problem Hard?*, 1996. Dostupno na Internet adresi: ftp://ftp.santafe.edu/pub/wgm/hard.ps.

[MAI94] Maini, H., Mehrotra, K., Mohan, C., Ranka, S., *Genetic Algorithms for Graph Partitioning and Incremental Graph Partitioning*, Supercomputing '94, pp. 449-457, 1994.

[MEY99] Meysenburg, M.M., Foster, J.A., *Randomness and GA performance, Revisited*, 1999. Dostupno na Internet adresi: ftp://ftp.cs.uidaho.edu/pub/foster/papers/prng-icga99.pdf.

[MIC94] Michalewicz, Z., "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer-Verlag, Berlin, 1994.

[MIC99] Michalewicz, Z., Esquivel, S., Gallard, R., Michalewicz, M., Tao, G., Trojanowski, K., *The Spirit of Evolutionary Algorithms*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 1-18, March 1999.

[MIL95] Miller, B.L., Goldberg, D.E., *GAs Tournament Selection and the Effects of Noise*, 1995. Dostupno na Internet adresi:http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Miller.Goldberg.GAs_Tournament_Selection_and_the_Effects_of_Noise.ps.gz.

[MIT94] Mitchell, M., Holland, J.H., Forrest, S., *When will a genetic algorithm outperform hill climbing?*, Advances in Neural Information Processing Systems, Vol. 6, San Mateo, CA: Morgan Kaufmann, 1994.

[MUE91a] Mühlenbein, H., *Evolution in Time and Space – The Parallel Genetic Algorithm*, Foundations of Genetic Algorithms, G. Rawlins (ed.), pp. 316-337, Morgan-Kaufman, 1991. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-91_01.ps.

[MUE91b] Mühlenbein, H., *Asynchronous parallel search by the parallel genetic algorithm*, Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processsing", V. Ramachandran ed. IEEE, pp. 526-533, 1991. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-96_08.ps.

[MUE92a] Mühlenbein, H., *Parallel Genetic Algorithms in Combinatorial Optimization*, Computer Science and Operations Research, Pergamon Press, New York, pp. 441-456, 1992. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-92_01.ps.

[MUE92b] Mühlenbein, H., *How Genetic Algorithms Really Work: Mutation and Hill-climbing*, Parallel Problem Solving from Nature (PPSN II), pp. 15-26, North-Holland, 1992., Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-92_02.ps.

[MUE94] Mühlenbein, H., *The optimal population size for uniform crossover and truncation selection*, 1994. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-94_11.ps.

[MUE98a] Mühlenbein, H., *The equation for the response to selection and its use for prediction*, Appeared in: Evolutionary Computation 5(3), pp. 303-346, 1998. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-98_01.ps.

[MUE98b] Mühlenbein, H., Mahnig, T., *Schemata, Distributions and Graphical Models in Evolutionary Optimization*, Alberto Ochoa Rodriguez Centre of Artificial Intelligence(ICIMAF), Cuba, 1998. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-98_02.ps.

[MUE99] Mühlenbein, H., Mahning, T., *Convergence Theory and Applications of the Factorized Distribution Algorithm*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 19-32, March 1999. Dostupno na Internet adresi: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-98_03.ps.

[MUN93] Munetomo, M., Takai, Y., Sato, Y., *An Efficient Migration Scheme for Subpopulation-Based Asynchronously PGA*, Hokkaido University Information Engineering Technical Report HIER-IS-9301, Jul., 1993.

[MUT92] Mutalik, P.P., Knight, L.R., Blanton, J.L., Wainwright, R.L., *Solving combinatorial optimization problems using parallel simulated annealing and parallel genetic algorithms*, Applied computing (vol. II) technological challenges of the 1990's, pp. 1031-1038, 1992.

[PAN95] Pannu, A.S., *Using Genetic Algorithms to Inductively Reason with Cases in the Legal Domain*, Artificial intelligence and law, pp. 175-184, 1995.

[PHA98] Pham, T.Q., Garg, P.K., *Multithreaded Programming*, Prentice Hall, 1998.

[POD99] Podgorelec, V., *Nursing Personnel Allocation Optimization with Evolutionary Algorithms*, Proceedings of the Third International ICSC Symposia on Intelligent Industrial Autimation, IIA'99 and Soft Computnig, SOCO'99, Genova, Italy, pp. 662-667, June 1-4,1999.

[REI93] Reif, J.,H., "*Synthesis of Parallel Algorithms*", Morgan Kaufmann Publishers, San Mateo, California, 1993.

[RIB86] Ribarić, S., "*Arhitektura računala pete generacije*", Tehnička knjiga Zagreb, Zagreb, 1986.

[RUD94] Rudnick, E.M., Patel, J.H, Greenstein, G.S., Niermann, T.M., *Sequential circuit test generation in a genetic algorithm framework*, Design automation, pp. 698-704, 1994.

[SAR96] Sarma, J., De Jong, K., *An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms*, Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN96), Berlin, Germany, pp. 22-26, Sept., 1996.

[SCH92] Schraudolph, N.N., Belew, R.K., *Dynamic parameter encoding for Genetic Algorithms*, Machine Learning, 9:9-21, 1991.

[SCH95] Schoeneburg E., Heinzmann F., Feddersen S., "*Genetische Algorithmen und Evolutions-strategien*", Addison-Wesley, 1995.

[SMI99] Smith, J.E., Vavak, F., *Replacement Strategies in Steady State Genetic Algorithms: Dynamic Environments*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 49-60, March 1999.

[SPE98a] Spears, W.M., De Jong., K.A., *Dining with GAs: Operator Lunch Theorem*, In Proceedings of Foundations of Genetic Algorithms, Springer-Verlag, 1998. Dostupno na Internet adresi: www.aic.nrl.navy.mil/~spears/papers.

[SPE98b] Spears, W.M., *The Role of Mutation and Recombination in Evolutionary Algorithms*, Ph.D. Dissertation, George Mason University, Fairfax, Virginia, 1998. Dostupno na Internet adresi: www.aic.nrl.navy.mil/~spears/papers.

[SRI94] Srinivas, M., Patnaik, L.M. (1994), *Genetic Algorithms: A Survey*, Computer, Vol. 27-6, pp.17-26, June 1994.

[TAL91] Talbi, E.-G., Bessière, P., *A parallel genetic algorithm for the graph partitioning problem*, Supercomputing, pp. 312 – 320, 1991.

[TAL92] Talbi, E.-G., Muntean, T., *A New Approach for the Mapping Problem: A Parallel Genetic Algorithm*, 1992. Dostupno na Internet adresi: www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Talbi.Muntean.A_New_Approach_for_the_Mapping_Problem.A_Parallel_Genetic_Algorithm.ps.gz.

[TES99] Testa, L., Esterline, A.C., Dozier, G.V., *Evolving Efficient Theme Park Tours*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 1-18, March 1999.

[TYN99] Tyni, T., Ylinen, J., *Improving the Performance of Genetic Algorithms with a Gene Bank*, Proceedings of EUROGEN99 – Short Course on Evolutionary Algorithms in Engineering and Computer Science, Jyvaskyla, Finska, 1999.

[TOM95] Tompkins, G., Azadivar, F., *Genetic Algorithms in Optimizing Simulated Systems*, Winter simulation conference, pp. 757-761, 1995.

[TOM99] Tomassini, M., *Parallel and Distributed Evolutionary Algorithms: A Review*, članak u knjizi "Evolutionary Algorithms in Engineering and Computer Science", Miettinen, K., Neittaanmaki, P., Makela, M.M., Periaux, J., John Wiley & sons, LTD, 1999.

[TUR89] Turk, S., Budin, L., "*Analiza i projektiranje računalom*", Školska Knjiga Zagreb, 1989.

[TUS98] Tuson, A.L., Ross, P., *Adapting Operator Settings In Genetic Algorithms*, Accepted for publication in Evolutionary Computation, (ISSN 10636560, 20 pages, estimated) – the original submitted version is also available as DAI Research Report 821, 1998.

[WIL97] Williams, E.A., Crossley, W.A., *Empirically-Derived Population Size and Mutation Rate Guidelines for a Genetic Algorithm with Uniform Crossover*, 1997., Dostupno na Internet adresi: http://roger.ecn.purdue.edu/~crossley/wsc2/paper/wsc2-97.html.

[WHI96] Whitley, D., *A Genetic Algorithm Tutorial*, 1996. Dostupno na Internet adresi: http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Whitley.A_Genetic_Algorithm_Tutorial.ps.gz.

[WHI99] Whitley, D., Rana, S., Heckendorn, R.B., *The Island Model Genetic Algorithm: On Separability, Population Size and Convergence*, CIT – Journal of Computing and Information Technology, Vol.7, No.1, pp. 33-48, March, 1999.

[WOL96a] Wolpert, D.H., Macready, W.G., *No Free Lunch Theorems for Search*, 1996. Dostupno na Internet adresi: ftp://ftp.santafe.edu/pub/wgm/nfl.ps.

[WOL96b] Wolpert, D.H., *No Free Lunch Theorems for Optimization*, 1996. Dostupno na Internet adresi: ftp://ftp.santafe.edu/pub/dhw_ftp/nfl.search.published.ps.Z.

[YOS99] Yoshida, N., Yasuoka, T., Moriki, T., *Parallel and Distributed Processing in VLSI Implementation of Genetic Algorithms*, Proceedings of the Third International ICSC Symposia on Intelligent Industrial Autimation, IIA'99 and Soft Computnig, SOCO'99, June 1-4, Genova, Italy, pp. 450-454, 1999.