# Konstruiranje blokovnih dizajna i jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem genetskih algoritama

**Zrinski, Tin**

Sveučilište u Zagrebu

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Tin Zrinski

# Konstruiranje blokovnih dizajna i jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem genetskih algoritama

DOKTORSKI RAD

Zagreb, 2022.

Sveučilište u Zagrebu

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Tin Zrinski

# Konstruiranje blokovnih dizajna i jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem genetskih algoritama

DOKTORSKI RAD

Mentor:

prof. dr. sc. Dean Crnković

Zagreb, 2022.

University of Zagreb

FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS

Tin Zrinski

# Constructing block designs and strongly regular graphs with prescribed automorphism group using genetic algorithms

DOCTORAL DISSERTATION

Supervisor:

prof. dr. sc. Dean Crnković

Zagreb, 2022.

# IZJAVA O IZVORNOSTI RADA

Ja, Tin Zrinski, student Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu, s prebivalištem na adresi [redacted] ovim putem izjavljujem pod materijalnom i kaznenom odgovornošću da je moj doktorski rad pod naslovom: Konstruiranje blokovnih dizajna i jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem genetskih algoritama, isključivo moje autorsko djelo, koje je u potpunosti samostalno napisano uz naznaku izvora drugih autora i dokumenata korištenih u radu.

U Zagrebu, listopad 2022.

Tin Zrinski

# ZAHVALA

U prvom redu zahvaljujem svom mentoru, prof. dr. sc. Deanu Crnkoviću, na zanimljivoj temi koju mi je predložio kao osnovu moje doktorske disertacije. Iskreno mu hvala na vođenju, pomoći, sugestijama, uputama i vrijednim idejama koje su učinile da se moje znanstveno istraživanje i ova doktorska disertacija ostvare. Hvala mu na predanosti, smirenosti, razumijevanju, podršci i poticanju.

Veliko hvala i članovima Povjerenstva za ocjenu doktorske disertacije, izv. prof. dr. sc. Vedranu Krčadincu, izv. prof. dr. sc. Vedrani Mikulić Crnković i izv. prof. dr. sc. Andrei Švob koji su svojim vrijednim primjedbama i kritikama doprinijeli kvaliteti i razumljivosti ovog rada.

Nadalje, zahvaljujem i mentorici svog diplomskog rada, prof. dr. sc. Sanji Rukavina koja je uvijek našla vremena i volje da mi pomogne i u nastavku mog školovanja.

Hvala svim članovima Seminara za konačnu matematiku na praćenju mojih izlaganja koja su vodila k izradi ove disertacije. Posebno hvala kolegi dr. sc. Matteu Mraviću za svu podršku i pomoć još od prve godine preddiplomskog studija i također hvala kolegici Matei Zubović uz koje dvoje je bilo lakše prebroditi slušanje i polaganje kolegija na doktorskom studiju. Zahvaljujem i kolegicama doc. dr. sc. Mariji Maksimović i dr. sc. Sari Ban na njihovoj potpori i materijalima koji su mi pomogli tijekom doktorskog studija. Hvala također izv. prof. dr. sc. Bojanu Crnkoviću i doc. dr. sc. Doris Dumičić Danilović na vrijednim savjetima koje su mi dali tijekom istraživanja.

Naposljetku, ogromno hvala mojoj obitelji, sestri, mami, tati i noni te prijateljima koji su mi pružali podršku tijekom cijelog obrazovanja. Posebno hvala Borni, Moreni i Valentini za sve odslušane jadikovke i podijeljene sretne trenutke.

# SAŽETAK

Konstrukcija blokovnih dizajna s određenim dopustivim parametrima se često pokušava izvesti za određeni skup parametara uz pretpostavljanje nekih dodatnih ograničenja na strukturu dizajna kako bi pretraživanje bilo računalno izvedivo. Prirodno ograničenje je pretpostavka da određena grupa automorfizama djeluje na dizajn. Jedna od metoda konstruiranja blokovnih dizajna s pretpostavljenom grupom automorfizama je metoda koja koristi orbitne matrice, a sastoji se od dvaju koraka: konstrukcije orbitnih matrica za pretpostavljenu grupu automorfizma i konstrukcije blokovnih dizajna za orbitne matrice dobivene na ovaj način (ovaj korak se naziva indeksiranje orbitnih matrica). Indeksiranje orbitnih matrica obično se provodi metodom iscrpnog pretraživanja. Međutim, ponekad iscrpno pretraživanje nije izvedivo jer postoji previše slučajeva koje bi trebalo provjeriti. Slično, jako regularni grafovi s određenim dopustivim parametrima se mogu konstruirati korištenjem orbitnih matrica za pretpostavljenu grupu automorfizma.

Genetski algoritam je metoda pretraživanja koja se koristi u računarstvu za pronalazak točnih ili približno točnih rješenja za probleme optimizacije i pretraživanja. Genetski algoritam oponaša prirodnu evoluciju, odnosno temelji se na optimizaciji populacije, podskupa cijelog prostora pretraživanja. Kao i u prirodi, populacija se sastoji od jedinki koje se mogu razmnožavati i nad kojima mogu djelovati određene mutacije te se na taj način stvaraju nove jedinke s boljim ili lošijim svojstvima od prethodnih. Cilj algoritma je usmjeriti populaciju ka stvaranju boljih jedinki što može rezultirati pronalaskom optimalnih rješenja zadanog problema.

U ovoj doktorskoj disertaciji opisujemo korištenje genetskog algoritma u koraku indeksiranja orbitnih matrica za konstrukciju blokovnih dizajna i jako regularnih grafova s pretpostavljenom grupom automorfizama. Korištenjem ovog pristupa konstruirali smo nove blokovne dizajne, točnije nove Steinerove sustave s parametrima $S(2,5,45)$ i nove simetrične dizajne s parametrima $(71,15,3)$ te nove jako regularne grafove s parametrima $(96,19,2,4)$ i $(96,20,4,4)$.

# SUMMARY

Construction of block designs with certain admissible parameters is often attempted for a particular set of parameters with the assumption of some additional constraints on the design structure in order to make the search computationally feasible. A natural constraint is the assumption that a given group of automorphisms acts on the design. One of the methods for constructing block designs with a prescribed automorphism group is the method that uses so called orbit matrices. It consists of two steps: construction of orbit matrices for the given automorphism group and construction of block designs for the orbit matrices obtained in this way (this step is called "indexing of orbit matrices"). Indexing is usually performed by exhaustive search. However, sometimes exhaustive search is not feasible because there are too many cases to check. Similarly, strongly regular graphs with certain admissible parameters can be constructed using orbit matrices for the prescribed automorphism group.

Genetic algorithms are search methods used in computing whose objective is to find exact or approximate solutions to optimization and search problems. A genetic algorithm mimics natural evolution, that is, it is based on optimizing a population (a subset of the entire search space). As in nature, the population consists of individuals that can reproduce and that can be affected by certain mutations, thus creating new individuals with better or worse properties than the previous ones. The goal of the algorithm is to direct the population towards creating better individuals, which can result in finding optimal solutions to a given problem.

In this doctoral dissertation, we describe the use of a genetic algorithm in the step of indexing of orbit matrices for the construction of block designs and strongly regular graphs with a prescribed automorphism group. Using this approach, we have constructed new block designs, namely new Steiner systems with parameters $S(2,5,45)$, new symmetric designs with parameters $(71,15,3)$ and new strongly regular graphs with parameters $(96,19,2,4)$ and $(96,20,4,4)$.

# KLJUČNE RIJEČI

blokovni dizajn, jako regularan graf, grupa automorfizama, orbitna matrica, genetski algoritam

# KEYWORDS

block design, strongly regular graph, automorphism group, orbit matrix, genetic algorithm

# SADRŽAJ

# UVOD

Teorija dizajna je grana kombinatorne i diskretne matematike koja se bavi egzistencijom, konstrukcijom i klasifikacijom konačnih incidencijskih struktura. Teorija dizajna doživljava nagli procvat od 50-tih godina 20. stoljeća zbog svoje široke primjene te razvoja računala i programske podrške. Primjena teorije dizajna nalazi se u raznim prirodnim i tehničkim znanostima. Neki od primjera su dizajniranje eksperimenata u biologiji, tehnologija umrežavanja u informatici te teorija kodiranja i kriptografija u matematici.

Teorija grafova je grana kombinatorne i diskretne matematike koja se bavi proučavanjem grafova, matematičkih struktura korištenih da bi se predstavile relacije koje uključuju dva elementa određene kolekcije. Temelje teorije grafova dao je još u prvoj polovici 18. stoljeća Leonhard Euler rješavanjem problema Königsberških mostova. Grafovi se koriste kao modeli za razne probleme iz svakodnevnog života, u računalnoj znanosti, kemiji, biologiji, fizici, društvenim znanostima, lingvistici i dr.

Genetski algoritmi (GA) su metode pretraživanja i optimizacije temeljene na populaciji koje su inspirirane prirodnim procesom evolucije. U svakom koraku algoritma, obrađuje se podskup cijelog prostora mogućih rješenja, zvan populacija. Populacija se sastoji od jedinki, a svaka jedinka sastavljena je od gena koji se mogu mutirati. Umjesto pronalaženja optimalnog rješenja unutar cijelog prostora rješenja, algoritam se koncentrira na optimizaciju odabrane populacije. Svaka jedinka predstavlja moguće rješenje (optimum), koje se procjenjuje pomoću funkcije dobrote. U svakoj iteraciji algoritma, određeni broj najbolje rangiranih jedinki - roditelja bira se za stvaranje novih, često boljih jedinki - djece. Djeca nastaju određenom vrstom rekombinacije gena - križanja i zamjenjuju najgore rangirane pojedince u populaciji, kako bi se povećala vjerojatnost za konvergenciju ka optimumu. Nakon što nastanu djeca, dopušta se njihova mutacija i stvara se sljedeća generacija populacije. Postupak se ponavlja sve dok se ne postigne uvjet prekida. Uobičajeni uvjeti prekida su: pronađena jedinka koja zadovoljava optimalne kriterije, nastupanje stagnacije u smislu da naredne iteracije algoritma više ne daju bolje re-

zultate, dostizanje unaprijed definiranog maksimalnog broja generacija. Ova metoda pokazuje se vrlo učinkovitom za rješavanje raznih optimizacijskih problema, uključujući neke NP-teške probleme (vidi [13]), kao i problema gdje je ostvarivo rješenje jedino optimalno rješenje - kao što je to u slučaju konstrukcije blokovnih dizajna ili jako regularnih grafova. Genetski algoritam korišten je u [45] za konstrukciju blokovnih dizajna i u [15] za pronalaženje unitala kao podstruktura simetričnih dizajna.

Konstrukcija blokovnih dizajna s određenim dopustivim parametrima se često pokušava izvesti za jedan određeni skup parametara uz pretpostavljanje nekih dodatnih ograničenja na strukturu dizajna kako bi pretraživanje bilo računalno izvedivo. Prirodno ograničenje je pretpostavka da određena grupa automorfizama djeluje na dizajn. Jedna od metoda konstruiranja blokovnih dizajna s pretpostavljenom grupom automorfizama je metoda koja koristi orbitne matrice. Ova metoda je opisana u [35] i uspješno su je koristili Z. Janko i drugi za konstrukciju blokovnih dizajna, na primjer u [18], [22], [36], [37], [38].

U cilju da se omogući konstrukcija blokovnih dizajna koja ne bi bila izvediva izvođenjem iscrpnog pretraživanja (engl. *exhaustive search*), predlažemo kombinaciju metode koja koristi orbitne matrice i genetskog algoritma. Koristeći ovaj pristup konstruirali smo 35 novih Steinerovih sustava s parametrima $S(2,5,45)$ i 22 nova simetrična dizajna s parametrima (71,15,3). Prema našim saznanjima, do sada je bilo poznato 30 Steinerovih sustava $S(2,5,45)$ (vidi [40]). To su Steinerovi sustavi konstruirani u [12], [16], [41], [47] i oni koji su konstruirani metodom opisanom u [48]. Do na izomorfizam, do sada je bilo poznato 148 simetričnih (71,15,3) dizajna, od kojih je 146 koji dopuštaju automorfizam reda šest opisano u [20], a dva međusobno dualna dizajna koji imaju $E_8$ kao punu grupu automorfizma dobiveni su iz kodova kao što je opisano u [19]. Heurističke metode korištene su i ranije za konstruiranje blokovnih dizajna i drugih kombinatornih struktura, na primjer u [31], [32], [39], [45], [46]. Međutim, koliko nam je poznato, ovo je prva primjena genetskog algoritma kao heurističke metode za konstrukciju blokovnih dizajna s pretpostavljenom grupom automorfizma, korištenjem orbitnih matrica.

M. Behbahani u svom doktoratu [3] izrađenom pod mentorstvom C. Lama i u [4] uvodi orbitne matrice jako regularnih grafova i razvija algoritam za konstrukciju orbitnih matrica jako regularnih grafova za djelovanje automorfizma prostog reda. Iako se orbitne matrice blokovnih dizajna uspješno primjenjuju od 1980-ih godina za konstrukciju blokovnih dizajna, u radu Behbahanija i Lama [4] prvi su put definirane i korištene orbitne matrice jako regularnih grafova. Direktna konstrukcija matrice susjedstva jako regularnog grafa često predugo traje čak i za gra-

fove s malo vrhova. Da bi se konstrukciju grafova učinilo mogućom, može se koristiti metoda konstrukcije uz pomoć orbitnih matrica koju su uveli Behbahani i Lam. Najprije se pretpostavi djelovanje određene grupe automorfizma i odrede sve moguće distribucije duljina orbita. Zatim se, pomoću u tu svrhu razvijenih algoritama i na njima temeljenih računalnih programa, konstruiraju odgovarajuće orbitne matrice jako regularnih grafova. Nakon toga se iz dobivenih orbitnih matrica konstruiraju matrice susjedstva jako regularnog grafa. M. Maksimović i D. Crnković su u [43] i u [17] napravili generalizaciju navedenog rezultata, odnosno izradili algoritme i računalne programe za konstrukciju orbitnih matrica jako regularnih grafova za djelovanje grupe automorfizama čiji red nije nužno prost broj i konstruirali nove jako regularne grafove iz dobivenih orbitnih matrica. Konstrukcija jako regularnih grafova iz orbitnih matrica uspješno je korištena i u radu D. Crnkovića, A. Švob i V. D. Toncheva [23].

U cilju da se omogući konstrukcija jako regularnih grafova koja ne bi bila izvediva izvođenjem iscrpnog pretraživanja, predlažemo kombinaciju metode koja koristi orbitne matrice i genetskog algoritma. Koliko nam je poznato, ovo je prva primjena genetskog algoritma za konstrukciju jako regularnih grafova s pretpostavljenom grupom automorfizma, korištenjem orbitnih matrica.

Poznati jako regularni grafovi $\mathrm{SRG}(96,19,2,4)$ i $\mathrm{SRG}(96,20,4,4)$ konstruirani su u [7], [9] i [50]. F. Ihringer je u [34] nedavno objavio mnoge nove $\mathrm{SRG}(96,19,2,4)$ i $\mathrm{SRG}(96,20,4,4)$ koji imaju grupe automorfizama malih redova (uključujući i trivijalne). Koristeći genetski algoritam na opisani način, konstruirali smo ukupno 21, od kojih je 15 novih, jako regularnih grafova $\mathrm{SRG}(96,19,2,4)$ i ukupno 22 nova jako regularna grafa $\mathrm{SRG}(96,20,4,4)$.

Ova doktorska disertacija podijeljena je u pet poglavlja. U poglavlju 1 navest ćemo osnovne definicije i poznate rezultate koji se tiču teorije grupa, teorije dizajna i teorije grafova koje ćemo koristiti u nastavku rada. Definirat ćemo i jako regularne grafove i navesti osnovne rezultate vezane uz njih. Također, definirat ćemo i pojmove orbitnih matrica blokovnih dizajna te orbitnih matrica jako regularnih grafova i navesti neka njihova osnovna svojstva. U poglavlju 2 prikazat ćemo kratki povijesni pregled evolucijskog računanja i genetskih algoritama. Opisat ćemo kako su genetski algoritmi modelirani kao imitacija procesa prirodne evolucije. Definirat ćemo osnovne pojmove vezane uz genetske algoritme, kao što su prostor pretraživanja, krajolik dobrote, funkcija dobrote, križanje, mutacija i selekcija. Opisat ćemo na koji način radi jednostavni genetski algoritam i prikazati pseudokod takvog algoritma. U poglavlju 3 ponovit ćemo istraživanje i rezultate I. Martinjaka i M.-O. Pavčevića iz članka [45] koji se bavi korište-

njem genetskog algoritma za konstruiranje blokovnih dizajna bez pretpostavljenog djelovanja grupe automorfizama. U svome članku, predstavili su genetski algoritam za konstrukciju 2-$(v,k,\lambda)$ dizajna pretraživanjem prostora rješenja samo na temelju parametara $v$, $k$ i $\lambda$ tih dizajna bez ikakvih dodatnih uvjeta. Korištenjem ovog algoritma uspjeli su pronaći nove jednostavne dizajne s parametrima 2-$(14,4,6)$. U poglavlju 4 uvest ćemo metodu konstrukcije blokovnih dizajna koja kombinira genetski algoritam i metodu za konstruiranje dizajna s pretpostavljenom grupom automorfizama korištenjem orbitnih matrica. Ovu metodu primijenit ćemo za konstrukciju novih Steinerovih sustava s parametrima $S(2,5,45)$ i novih simetričnih dizajna s parametrima (71,15,3). U poglavlju 5 uvest ćemo metodu konstrukcije jako regularnih grafova koja kombinira genetski algoritam i metodu za konstruiranje jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem orbitnih matrica. Ovu metodu primijenit ćemo za konstrukciju novih jako regularnih grafova SRG$(96,19,2,4)$ i SRG$(96,20,4,4)$.

# 1. OSNOVNI POJMOVI

U ovom poglavlju navest ćemo osnovne definicije i rezultate koji se tiču teorije grupa, teorije dizajna i teorije grafova koje ćemo koristiti u nastavku rada. Definirat ćemo i jako regularne grafove i navesti osnovne rezultate vezane uz njih. Također, definirat ćemo i pojmove orbitnih matrica blokovnih dizajna te orbitnih matrica jako regularnih grafova i navesti neka njihova osnovna svojstva.

## 1.1. GRUPE

Dokazi tvrdnji iz teorije grupa koje ovdje navodimo mogu se pronaći u [11] i [42].

**Definicija 1.1.1.** Skup $G$ uz binarnu operaciju $\cdot : G \times G \to G$ čini grupu ako vrijedi:

1. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, za sve $a, b, c \in G$ (asocijativnost),
2. postoji element $e \in G$ takav da je $a \cdot e = e \cdot a = a$, za sve $a \in G$ (postojanje neutralnog elementa),
3. za sve $a \in G$ postoji element $a^{-1} \in G$ takav da je $a \cdot a^{-1} = a^{-1} \cdot a = e$ (postojanje inverza).

**Definicija 1.1.2.** Grupa $G$ je komutativna (Abelova) ako zadovoljava svojstvo $a \cdot b = b \cdot a$, za sve $a, b \in G$.

**Napomena 1.1.3.** Trivijalna grupa je grupa $I = \{e\}$.

**Definicija 1.1.4.** Ako je $G$ konačan skup, $G$ je konačna grupa i $|G|$ je red grupe. Ako je $G$ beskonačan skup, $G$ je beskonačna grupa.

**Definicija 1.1.5.** Element $a$ grupe $G$ je reda $n$, u oznaci $|a| = n$, ako je $n$ najmanji prirodan broj takav da je $a^n = e$, gdje je $e$ neutralni element grupe $G$. Ako takav $n$ ne postoji, $a$ je beskonačnog reda.

**Definicija 1.1.6.** Neka je $G$ grupa. Podskup $H$ od $G$ koji je i sam grupa uz istu binarnu operaciju kao i $G$ je podgrupa grupe $G$. Označavamo $H \leq G$.

**Definicija 1.1.7.** Neka su $G$ i $H$ grupe. Preslikavanje $f : G \to H$ za koje vrijedi

$$f(g \cdot h) = f(g) \cdot f(h), \text{ za sve } g,h \in G$$

je homomorfizam iz grupe $G$ u grupu $H$.

**Definicija 1.1.8.** Neka je $f : G \to H$ homomorfizam grupa i neka je $e_H$ neutralni element u grupi $H$. Skup

$$\text{Ker}(f) = \{g \in G \mid f(g) = e_H\}$$

je jezgra homomorfizma $f$, a skup

$$\text{Im}(f) = \{f(g) \mid g \in G\}$$

je slika homomorfizma $f$.

**Propozicija 1.1.9.** Neka je $f : G \to H$ homomorfizam grupa i neka je $e_G$ neutralni element u grupi $G$. Tada vrijedi:

$$f \text{ je injektivan} \Leftrightarrow \text{Ker}(f) = \{e_G\},$$
$$f \text{ je surjektivan} \Leftrightarrow \text{Im}(f) = H.$$

**Definicija 1.1.10.** Injektivni homomorfizam je monomorfizam. Surjektivni homomorfizam je epimorfizam. Bijektivni homomorfizam je izomorfizam.

**Definicija 1.1.11.** Grupe $G$ i $H$ su izomorfne ako postoji izomorfizam grupa iz $G$ u $H$. Označavamo: $G \cong H$.

**Definicija 1.1.12.** Homomorfizam grupe $G$ na samu sebe ($f : G \to G$) je endomorfizam, a izomorfizam grupe $G$ na samu sebe je automorfizam. Skup svih homomorfizama sa $G$ u $H$ označavamo $\text{Hom}(G,H)$, a skup svih automorfizama grupe $G$ označavamo $\text{Aut}(G)$.

**Napomena 1.1.13.** Neka je $G$ grupa i $\text{Aut}(G)$ skup svih automorfizama grupe $G$. $\text{Aut}(G)$ uz operaciju kompozicije funkcija čini grupu.

**Definicija 1.1.14.** $\text{Aut}(G)$ je puna grupa automorfizama grupe $G$.

**Definicija 1.1.15.** Neka je $G$ grupa i $X \subseteq G$. Presjek svih podgrupa od $G$ koje sadrže X je podgrupa generirana sa $X$ i označava se sa $\langle X \rangle$. Ako je $X$ konačan skup, $X = \{x_1, x_2, \ldots, x_n\}$, pišemo $\langle X \rangle = \langle x_1, x_2, \ldots, x_n \rangle$.

**Propozicija 1.1.16.** Neka je $G$ grupa i $X$ neprazan podskup od $G$. Tada $\langle X \rangle$ sadrži sve konačne umnoške $x_1^{k_1} \cdot x_2^{k_2} \cdots x_m^{k_m}$, gdje je $x_i \in X$, $k_i \in \mathbb{Z}$ za svaki $i \in \{1, 2, \ldots, m\}$, $m \in \mathbb{N}$.

**Definicija 1.1.17.** Ako je $X \subseteq G$ i $\langle X \rangle = G$, $X$ je skup generatora grupe $G$.

**Definicija 1.1.18.** Grupa $G = \langle x \rangle$ generirana jednim elementom $x$ je ciklička grupa.

**Napomena 1.1.19.** Red cikličke grupe $\langle x \rangle$ je red elementa $x$, u oznaci $|x|$.

Neka je $\Omega$ neprazan skup. Skup $S(\Omega)$ svih bijekcija (permutacija) sa skupa $\Omega$ na skup $\Omega$ uz operaciju kompozicije funkcija čini grupu.

**Definicija 1.1.20.** $S(\Omega)$ je simetrična grupa na skupu $\Omega$. Ako je $G \leq S(\Omega)$, $G$ je permutacijska grupa na $\Omega$.

**Napomena 1.1.21.** Ukoliko je $|\Omega| = n$, $n \in \mathbb{N}$, onda ćemo $S(\Omega)$ označavati sa $S_n$. $S_n$ je simetrična grupa stupnja $n$. Red te grupe je $|S_n| = n!$. Ako je $G \leq S_n$, $G$ je permutacijska grupa stupnja $n$.

**Teorem 1.1.22.** *(Cayley)* Svaka grupa $G$ je izomorfna podgrupi grupe $S(G)$. Posebno, ako je $G$ konačna grupa reda $n$, tada je ona izomorfna podgrupi simetrične grupe $S_n$. Drugim riječima, svaka grupa je izomorfna nekoj permutacijskoj grupi.

**Definicija 1.1.23.** Neka su $i_1, i_2, \ldots, i_r$ međusobno različiti elementi skupa $I_n = \{1, 2, \ldots, n\}$. Sa $(i_1, i_2, \ldots, i_r)$ označit ćemo permutaciju koja preslikava

$$i_1 \mapsto i_2, i_2 \mapsto i_3, \ldots, i_{r-1} \mapsto i_r, i_r \mapsto i_1$$

dok ostale elemente skupa $I_n$ fiksira. Permutacija $(i_1, i_2, \ldots, i_r)$ je ciklus duljine $r$. Ciklus duljine 2 je transpozicija.

Svaki ciklus duljine $n \geq 2$ $(a_1, a_2, \ldots, a_n)$ se može prikazati kao kompozicija transpozicija, odnosno $(a_1, a_2, \ldots, a_n) = (a_1, a_2)(a_1, a_3) \ldots (a_1, a_n)$. Ciklusi $(a_1, a_2, \ldots, a_k)$ i $(b_1, b_2, \ldots, b_l)$ su disjunktni ako je

$$\{a_1, a_2, \ldots, a_k\} \cap \{b_1, b_2, \ldots, b_l\} = \emptyset.$$

Svaki element simetrične grupe $S_n$ može se jednoznačno do poretka prikazati kao kompozicija disjunktnih ciklusa. Parna (neparna) permutacija je ona permutacija koja se može prikazati kao kompozicija parnog (neparnog) broja transpozicija.

**Definicija 1.1.24.** Grupa parnih permutacija na konačnom skupu je alternirajuća grupa. Alternirajuća grupa na skupu $\{1,2,\ldots,n\}$ je alternirajuća grupa stupnja $n$, u oznaci $A_n$.

**Definicija 1.1.25.** Neka je $G$ grupa s neutralnim elementom $e$ i $\Omega$ neprazan skup. Grupa $G$ djeluje na skup $\Omega$ ako je zadano preslikavanje $. : G \times \Omega \to \Omega$ za koje vrijedi:

1. $e.x = x,\ x \in \Omega$,
2. $g.(h.x) = (g \cdot h).x,\ g,h \in G,\ x \in \Omega$.

Skup $\Omega$ na kojem je definirano djelovanje grupe $G$ je $G$-skup. Slika djelovanja elementa $g \in G$ na element $x \in \Omega$ označava se sa $g.x$.

**Definicija 1.1.26.** Na skupu $\Omega$ na koji djeluje grupa $G$ definiramo relaciju $\sim$ na sljedeći način:

$$x \sim y \ \Leftrightarrow\ (\exists g \in G)\ g.x = y.$$

Ta relacija je $G$-povezanost.

**Propozicija 1.1.27.** Neka je $\Omega$ neprazan skup na koji djeluje grupa $G$. Relacija $G$-povezanosti je relacija ekvivalencije na skupu $\Omega$.

**Definicija 1.1.28.** Neka je $\Omega$ neprazan skup na koji djeluje grupa $G$. Klase ekvivalencije s obzirom na relaciju $G$-povezanosti su $G$-orbite (ili samo orbite). Broj elemenata neke orbite je duljina te orbite. Orbita elementa $x \in \Omega$ je skup

$$G.x = \{g.x \mid g \in G\}.$$

**Definicija 1.1.29.** Neka je $\Omega$ neprazan skup na koji djeluje grupa $G$. Neka je $x \in \Omega$. Ako je $g.x = x$ za svaki $g \in G$, onda je $x$ fiksna točka za to djelovanje.

**Definicija 1.1.30.** Neka je $\Omega$ neprazan skup na koji djeluje grupa $G$. Neka je $x \in \Omega$. Skup

$$G_x = \{g \in G \mid g.x = x\}$$

je stabilizator elementa $x$ za djelovanje grupe $G$ na $\Omega$.

**Napomena 1.1.31.** Vrijedi da je $G_x \leq G$ za svaki $x \in \Omega$.

**Definicija 1.1.32.** Neka je $G$ permutacijska grupa na skupu $\Omega$ i neka je $X \subseteq \Omega$. Onda je skup $G_X = \{g \in G \mid g.x = x,\ x \in X\}$ točkovni stabilizator skupa $X$ u grupi $G$. Označimo: $g.X = \{g.x \mid x \in X\}$. Skup $G(X) = \{g \in G \mid g.X = X\}$ je skupovni stabilizator skupa $X$ u grupi $G$.

**Napomena 1.1.33.** Vrijedi da je $G_X = \bigcap_{x \in X} G_x$ te $G_X \leq G$ i $G(X) \leq G$ za svaki $X \subseteq \Omega$.

**Korolar 1.1.34.** Neka je $\Omega$ konačan skup i $G$ konačna grupa, $G \leq S(\Omega)$. Tada vrijedi

$$|G.x| = \frac{|G|}{|G_x|},\ \forall x \in \Omega.$$

**Definicija 1.1.35.** Neka je $G$ grupa, $H \leq G$ i $g \in G$. Skup $gH = \{g \cdot h \mid h \in H\}$ je lijeva susjedna klasa podgrupe $H$ određena elementom $g$. Skup svih lijevih susjednih klasa podgrupe $H$ označavamo sa $G/H = \{gH \mid g \in G\}$.

Skup $Hg = \{h \cdot g \mid h \in H\}$ je desna susjedna klasa podgrupe $H$ određena elementom $g$. Skup svih desnih susjednih klasa podgrupe $H$ označavamo sa $H\backslash G = \{Hg \mid g \in G\}$.

**Definicija 1.1.36.** Neka je $H \leq G$. Kardinalni broj skupa $G/H$ je indeks podgrupe $H$ u $G$ i označava s $[G : H]$.

**Propozicija 1.1.37.** Ako je $G \leq S(\Omega)$, tada za svaki $x \in \Omega$ vrijedi

$$|G.x| = [G : G_x].$$

**Teorem 1.1.38.** *(Lagrangeov teorem)* Neka je $G$ konačna grupa i $H \leq G$. Tada je

$$|G| = [G : H] \cdot |H|.$$

**Definicija 1.1.39.** Neka je $H$ podgrupa grupe $G$ i $g \in G$. Grupu $gHg^{-1}$ označavamo sa $H^g$. Grupe $H$ i $H^g$ su međusobno konjugirane.

**Definicija 1.1.40.** Podgrupa $H$ grupe $G$ za koju vrijedi

$$H = gHg^{-1} = H^g, \forall g \in G,$$

je normalna podgrupa od $G$ i pišemo $H \trianglelefteq G$.

**Napomena 1.1.41.** Grupa $A_n$ je normalna podgrupa grupe $S_n$ za svaki $n \in \mathbb{N}$.

**Propozicija 1.1.42.** Neka je $f : G \to H$ homomorfizam grupa. Tada je

$$\mathrm{Ker}(f) \trianglelefteq G,\ \mathrm{Im}(f) \leq H.$$

Ako je $H \trianglelefteq G$ tada su skupovi lijevih i desnih susjednih klasa podgrupe $H$ jednaki pa ih možemo jednostavno označiti sa $G/H$. U tom slučaju se na skupu $G/H$ može definirati struktura grupe, o čemu govori sljedeći teorem.

**Teorem 1.1.43.** Neka je $H \trianglelefteq G$. Tada je $G/H$ grupa u odnosu na množenje definirano sa $(xH)(yH) = (xy)H$. Preslikavanje $\pi : G \to G/H$, $\pi(g) = gH$, je epimorfizam i $\mathrm{Ker}(\pi) = H$.

**Definicija 1.1.44.** Neka je $H \trianglelefteq G$. Grupa $G/H$ uz operaciju iz prethodnog teorema je kvocijentna grupa, a homomorfizam $\pi : G \to G/H,\ \pi(g) = gH$, je kanonski homomorfizam ili projekcija sa $G$ na $G/H$.

**Definicija 1.1.45.** Neka su $G_i$ grupe za $i \in \{1,2,\ldots n\}$. Direktni produkt grupa $G_i$ je skup $G = \{(g_1, g_2, \ldots, g_n) \mid g_i \in G_i,\ i = 1,2,\ldots,n\}$, u oznaci $G = G_1 \times G_2 \times \ldots \times G_n$.

**Propozicija 1.1.46.** Direktni produkt $G$ je grupa s obzirom na operaciju

$$(g_1, g_2, \ldots, g_n)(g_1', g_2', \ldots g_n') = (g_1 \cdot g_1', g_2 \cdot g_2', \ldots, g_n \cdot g_n').$$

Neka su $N$ i $H$ grupe i neka je zadan homomorfizam

$$\varphi : H \to \mathrm{Aut}(N).$$

Neka je $G = \{(n,h) \mid n \in N, h \in H\}$ skup na kojemu je definirana sljedeća operacija:

$$(n_1, h_1) * (n_2, h_2) = (n_1 \cdot \varphi_{h_1}(n_2), h_1 \cdot h_2),$$

gdje je $\varphi_h = \varphi(h)$. Vrijedi da je $(G, *)$ grupa.

**Definicija 1.1.47.** Prethodno definirana grupa $G$ je semidirektni produkt grupa $N$ i $H$ određen sa $\varphi$, u oznaci $N \rtimes_\varphi H$ ili $N : H$.

**Definicija 1.1.48.** Neka je $G$ grupa i $p$ prost broj. Grupa $G$ je $p$-grupa ako za svaki $g \in G$ vrijedi da je $|g| = p^k$ za neki $k \in \mathbb{N}_0$.

**Definicija 1.1.49.** $p$-grupa je elementarno Abelova ako je Abelova grupa i za svaki $g \in G$, $g \neq e$, vrijedi $|g| = p$. Za $n \in \mathbb{N}, n \geq 2$, oznaka za elementarno Abelovu grupu reda $n$ je $E_n$.

**Definicija 1.1.50.** Grupa simetrija pravilnog mnogokuta je diedralna grupa. Grupu koju čini $2n$ simetrija pravilnog $n$-terokuta, $n \geq 3$, označavat ćemo s $D_{2n}$.

**Teorem 1.1.51.** *(Frobenius)* Neka je $G$ konačna grupa s neutralnim elementom $e$ i $H \leq G$, $H \neq \{e\}$, $H \neq G$, takva da je $H \cap H^g = \{e\}$ za svaki $g \in G \backslash H$. Tada je

$$N = \left( G - \bigcup_{g \in G} H^g \right) \cup \{e\}$$

normalna podgrupa grupe $G$ takva da je $G = HN$ i $H \cap N = \{e\}$.

**Definicija 1.1.52.** Grupa $G$ koja zadovoljava pretpostavku iz prethodnog teorema je Frobeniusova grupa. Podgrupa $N$ je Frobeniusova jezgra, a podgrupa $H$ Frobeniusov komplement.

**Napomena 1.1.53.** Frobeniusova grupa $G$ je semidirektni produkt grupa $N$ i $H$. Oznaka za Frobeniusovu grupu reda $n$ je $F_n$.

**Definicija 1.1.54.** Grupa kvaterniona je grupa

$$Q_8 = \{\pm 1, \pm i, \pm j, \pm k\},$$

s neutralnim elementom 1, gdje je $x^2 = -1, x \in \{\pm i, \pm j, \pm k\}$ te $i \cdot j = k$, $j \cdot k = i$, $k \cdot i = j$, $j \cdot i = -k$, $k \cdot j = -i$, $i \cdot k = -j$.

## 1.2. INCIDENCIJSKE STRUKTURE

U ovom poglavlju definirat ćemo osnovne pojmove vezane uz incidencijske stukture. Uz definicije, dat ćemo i neke od osnovnih svojstava navedenih struktura.

**Definicija 1.2.1.** Incidencijska struktura $\mathscr{D}$ je uređena trojka $(\mathscr{P},\mathscr{B},\mathscr{I})$, pri čemu su $\mathscr{P}$ i $\mathscr{B}$ neprazni disjunktni skupovi, a $\mathscr{I} \subseteq \mathscr{P} \times \mathscr{B}$. Elementi skupa $\mathscr{P}$ su točke, a elementi skupa $\mathscr{B}$ blokovi (ili pravci). Relacija $\mathscr{I}$ je relacija incidencije. Incidencijska struktura je konačna ukoliko su skupovi njezinih točaka i blokova konačni skupovi.

Broj blokova iz $\mathscr{B}$ koji su incidentni s točkom $P \in \mathscr{P}$ je stupanj točke $P$. Broj točaka iz $\mathscr{P}$ koji su incidentni s blokom $x \in \mathscr{B}$ je stupanj bloka $x$.

**Propozicija 1.2.2.** Neka je $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$ incidencijska struktura koja ima $v$ točaka i $b$ blokova. Označimo stupnjeve točaka s $r_1, r_2, \ldots r_v$, a stupnjeve blokova s $k_1, k_2, \ldots k_b$. U tom slučaju vrijedi:

$$\sum_{i=1}^{v} r_i = \sum_{i=1}^{b} k_i.$$

**Korolar 1.2.3.** Neka je $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$ incidencijska struktura koja ima $v$ točaka i $b$ blokova. Neka je stupanj svake točke $r$, a stupanj svakog bloka $k$. Tada vrijedi:

$$vr = bk.$$

**Definicija 1.2.4.** Incidencijska struktura $\mathscr{D}' = (\mathscr{P}',\mathscr{B}',\mathscr{I}')$ je podstruktura incidencijske strukture $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$ ako je $\mathscr{P}' \subseteq \mathscr{P}$, $\mathscr{B}' \subseteq \mathscr{B}$ te ako je $\mathscr{I}' = (\mathscr{P}' \times \mathscr{B}') \cap \mathscr{I}$. $\mathscr{D}'$ je prava podstruktura incidencijske strukture $\mathscr{D}$ ako vrijedi $\mathscr{P}' \subset \mathscr{P}$.

**Definicija 1.2.5.** Incidencijska struktura $\mathscr{D}' = (\mathscr{P},\mathscr{B},\mathscr{I}')$ je komplementarna struktura incidencijske strukture $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$ ako je $\mathscr{I}' = (\mathscr{P} \times \mathscr{B}) \setminus \mathscr{I}$.

**Definicija 1.2.6.** Incidencijska struktura $\mathscr{D}^* = (\mathscr{P}^*,\mathscr{B}^*,\mathscr{I}^*)$, pri čemu je $\mathscr{P}^* = \mathscr{B}$, $\mathscr{B}^* = \mathscr{P}$ te $\mathscr{I}^* = \{(B,P) | (P,B) \in \mathscr{I}\}$ je dualna struktura incidencijske strukture $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$.

**Definicija 1.2.7.** Neka je $\mathscr{D} = (\mathscr{P},\mathscr{B},\mathscr{I})$ konačna incidencijska struktura koja ima $v$ točaka, označimo ih s $P_1, P_2, \ldots, P_v$ i $b$ blokova, označimo ih s $x_1, x_2, \ldots, x_b$. Takvoj incidencijskoj strukturi možemo pridružiti njezinu matricu incidencije, to jest, $v \times b$ matricu $M = [m_{i,j}]$, pri čemu je

$$m_{i,j} = \begin{cases} 1, (P_i, x_j) \in \mathscr{I}, \\ 0, (P_i, x_j) \notin \mathscr{I}. \end{cases}$$

**Napomena 1.2.8.** Neka je incidencijskoj strukturi $\mathscr{D}$ pridružena $v \times b$ matrica incidencije $M$. Neka je $J$ $v \times b$ matrica čiji su elementi na svim pozicijama jedinice. Tada za incidencijsku matricu $M'$ koja je pridružena komplementarnoj strukturi $\mathscr{D}'$ vrijedi $M' = J - M$, a za je incidencijsku matricu $M^*$ koja je pridružena dualnoj strukturi $\mathscr{D}^*$ vrijedi $M^* = M^T$.

**Definicija 1.2.9.** Neka su dane dvije incidencijske strukture: $\mathscr{D}_1 = (\mathscr{P}_1, \mathscr{B}_1, \mathscr{I}_1)$ i $\mathscr{D}_2 = (\mathscr{P}_2, \mathscr{B}_2, \mathscr{I}_2)$. Bijektivno preslikavanje $f : \mathscr{P}_1 \cup \mathscr{B}_1 \to \mathscr{P}_2 \cup \mathscr{B}_2$ je izomorfizam iz $\mathscr{D}_1$ u $\mathscr{D}_2$ ako vrijedi sljedeće:

1. $f$ preslikava $\mathscr{P}_1$ na $\mathscr{P}_2$ i $f$ preslikava $\mathscr{B}_1$ na $\mathscr{B}_2$,
2. $(P,x) \in \mathscr{I}_1 \Leftrightarrow (f(P), f(x)) \in \mathscr{I}_2$, $P \in \mathscr{P}_1$, $x \in \mathscr{B}_1$.

Ako je $\mathscr{D} = \mathscr{D}_1 = \mathscr{D}_2$, tada je preslikavanje $f$ automorfizam incidencijske strukture $\mathscr{D}$.

**Napomena 1.2.10.** Skup svih automorfizama incidencijske strukture $\mathscr{D}$ uz operaciju kompozicije funkcija čini punu grupu automorfizama incidencijske strukture $\mathscr{D}$ koju označavamo sa $\mathrm{Aut}(\mathscr{D})$. Bilo koja podgrupa $G$ pune grupe automorfizama $\mathrm{Aut}(\mathscr{D})$ je grupa automorfizama incidencijske strukture $\mathscr{D}$.

**Definicija 1.2.11.** Incidencijska struktura $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ je samodualna ako je izomorfna svojoj dualnoj strukturi $\mathscr{D}^* = (\mathscr{P}^*, \mathscr{B}^*, \mathscr{I}^*)$.

# 1.3. DIZAJNI

Dokazi tvrdnji iz teorije dizajna koje ovdje navodimo mogu se pronaći u [59].

**Definicija 1.3.1.** Konačna incidencijska struktura $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ je $t$-$(v,k,\lambda)$ dizajn ako vrijedi:

1. $|\mathscr{P}| = v$,
2. svaki element skupa $\mathscr{B}$ incidentan je s točno $k$ elemenata skupa $\mathscr{P}$,
3. svakih $t$ elemenata skupa $\mathscr{P}$ incidentno je s točno $\lambda$ elemenata skupa $\mathscr{B}$.

Elementi skupa $\mathscr{P}$ su točke dizajna, a elementi skupa $\mathscr{B}$ blokovi dizajna. Nenegativni cijeli brojeve $t, v, k$ i $\lambda$ su parametri dizajna. Broj blokova dizajna $\mathscr{D}$ označavat ćemo sa $b$. Na parametre postavljamo uvjet $0 \leq t \leq k \leq v$ kako bismo izbjegli trivijalne slučajeve.

**Napomena 1.3.2.** Potpuni dizajn je dizajn čiji su blokovi svi $k$-člani podskupovi skupa točaka, a relacija incidencije inducirana je relacijom pripadnosti podskupu. Nepotpuni dizajn je dizajn koji ne sadrži sve $k$-člane podskupove kao blokove.

**Definicija 1.3.3.** Podstruktura dizajna $\mathscr{D}$ je poddizajn dizajna $\mathscr{D}$ ukoliko je i sama dizajn.

**Napomena 1.3.4.** Dizajni $\mathscr{D}_1$ i $\mathscr{D}_2$ su izomorfni ako su izomorfni kao incidencijske strukture. Automorfizam dizajna je izomorfizam tog dizajna na samog sebe.

**Definicija 1.3.5.** Dizajni s parametrima 2-$(v,k,\lambda)$ su blokovni dizajni ili 2-dizajni.

**Definicija 1.3.6.** Ako je u $t$-$(v,k,\lambda)$ dizajnu $t \geq 2$ i $\lambda = 1$, tada je $t$-$(v,k,\lambda)$ dizajn Steinerov sustav ili Steinerov $t$-dizajn i označavamo ga sa $S(t,k,v)$.

**Lema 1.3.7.** Neka je $\mathscr{D}$ 2-$(v,k,1)$ dizajn. Svaka dva bloka u dizajnu $\mathscr{D}$ sijeku se u jednoj točki ili se ne sijeku.

**Teorem 1.3.8.** Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ $t$-$(v,k,\lambda)$ dizajn. Tada je, za svaki cijeli broj $s$ za koji vrijedi $0 \leq s \leq t$, broj blokova incidentnih sa $s$ različitih točaka dizajna neovisan o izboru tih $s$ točaka i iznosi

$$\lambda_s = \lambda \frac{\binom{v-s}{t-s}}{\binom{k-s}{t-s}}.$$

$\mathscr{D}$ je $s-(v,k,\lambda_s)$ dizajn za svaki $s \in \{1,\ldots,t\}$.

**Korolar 1.3.9.** Iz prethodnog teorema slijedi rekurzija za parametre $\lambda_s$:

$$\lambda_s = \frac{v-s}{k-s}\,\lambda_{s+1}.$$

**Korolar 1.3.10.** U $t$-$(v,k,\lambda)$ dizajnu $\mathscr{D}$, svaka točka incidentna je s točno

$$r = \lambda_1 = \lambda\,\frac{\binom{v-1}{t-1}}{\binom{k-1}{t-1}}$$

blokova.

Dizajn $\mathscr{D}$ s parametrima $t$-$(v,k,\lambda)$ dizajn $\mathscr{D}$ ima točno

$$b = \lambda_0 = \lambda\,\frac{\binom{v}{t}}{\binom{k}{t}}$$

blokova.

**Definicija 1.3.11.** Uređena četvorka $(t,v,k,\lambda)$ je dopustiva ako je svaki

$$\lambda_s = \lambda\,\frac{\binom{v-s}{t-s}}{\binom{k-s}{t-s}}$$

prirodan broj, za sve $s = 0,1,\ldots,t$.

Uređena četvorka $(t,v,k,\lambda)$ je ostvariva ako postoji $t$-$(v,k,\lambda)$ dizajn.

**Napomena 1.3.12.** Dopustivost ne implicira ostvarivost.

**Korolar 1.3.13.** U 2-$(v,k,\lambda)$ dizajnu, svaka točka incidentna je s točno

$$r = \frac{\lambda(v-1)}{k-1}$$

blokova.

Dizajn s parametrima 2-$(v,k,\lambda)$ ima točno

$$b = \frac{vr}{k} = \frac{\lambda(v^2-v)}{k^2-k}$$

blokova.

**Definicija 1.3.14.** Red $t$-dizajna $\mathscr{D}$, gdje je $t \geq 2$, je broj $n = \lambda_1 - \lambda_2$. Specijalno, red 2-dizajna je $n = r - \lambda$.

**Teorem 1.3.15.** Neka je $M$ $(0,1)$-matrica dimenzija $v \times b$. Matrica $M$ je matrica incidencije 2-$(v,k,\lambda)$ dizajna ako i samo ako vrijede sljedeće jednakosti:

1. $MM^T = \lambda J_v + (r-\lambda)I_v$,

2. $u_v M = k u_b$,

pri čemu je $I_v$ jedinična matrica reda $v$, $J_v$ matrica reda $v$ čiji su svi elementi jedinice, a $u_v$, odnosno $u_b$, vektori duljina $v$, odnosno $b$, čiji su svi elementi jedinice.

**Definicija 1.3.16.** Neka su $\mathscr{D}_1$ i $\mathscr{D}_2$ dva $t$-$(v,k,\lambda)$ dizajna. Dizajni $\mathscr{D}_1$ i $\mathscr{D}_2$ su izomorfni ako su izomorfni kao incidencijske strukture.

**Napomena 1.3.17.** Skup svih automorfizama dizajna $\mathscr{D}$ zajedno s operacijom kompozicije funkcija je grupa automorfizama dizajna $\mathscr{D}$ i označavamo ju $\mathrm{Aut}(\mathscr{D})$. Bilo koja podgrupa $G$ pune grupe automorfizama $\mathrm{Aut}(\mathscr{D})$ je grupa automorfizama dizajna $\mathscr{D}$.

**Teorem 1.3.18.** Neka su $M = [m_{i,j}]$, odnosno $M' = [m'_{i,j}]$ matrice incidencije blokovnih dizajna $\mathscr{D}$, odnosno $\mathscr{D}'$. Dizajni $\mathscr{D}$ i $\mathscr{D}'$ su izomorfni ako i samo ako postoje permutacija $\alpha$ skupa $\{1,2,\ldots,v\}$ i permutacija $\beta$ skupa $\{1,2,\ldots,b\}$ takve da je

$$m_{i,j} = m'_{\alpha(i),\beta(j)},$$

za sve $i \in \{1,2,\ldots,v\}$, $j \in \{1,2,\ldots,b\}$.

**Teorem 1.3.19.** *(Fisherova nejednakost)* Ako postoji 2-$(v,k,\lambda)$ dizajn s $b$ blokova, tada je $b \geq v$.

**Definicija 1.3.20.** Nepotpuni $t$-$(v,k,\lambda)$ dizajn $\mathscr{D}$ je simetričan ako je $t \geq 2$ i ako ima jednak broj blokova i točaka, tj. $|\mathscr{P}| = |\mathscr{B}|$.

**Propozicija 1.3.21.** Ako je $t$-$(v,k,\lambda)$ simetričan dizajn, tada je $t = 2$.

**Napomena 1.3.22.** Za simetričan $(v,k,\lambda)$ dizajn $\mathscr{D}$ vrijedi $r = k$.

**Napomena 1.3.23.** Red simetričnog $(v,k,\lambda)$ dizajna je $n = k - \lambda$.

**Napomena 1.3.24.** Komplementaran dizajn 2-$(v,k,\lambda)$ dizajna $\mathscr{D}$ je dizajn s parametrima 2-$(v, v-k, v-2k+\lambda)$. Ako je $\mathscr{D}$ simetričan blokovni dizajn, tada je i njegov komplement simetričan dizajn.

**Definicija 1.3.25.** Projektivna ravnina reda $n$ je simetrični blokovni dizajn s parametrima 2-$(n^2+n+1, n+1, 1)$.

**Definicija 1.3.26.** Afina ravnina reda $n$ je dizajn s parametrima 2-$(n^2, n, 1)$.

**Napomena 1.3.27.** Afina ravnina nije simetričan dizajn.

**Definicija 1.3.28.** Simetričan dizajn s parametrima 2-$(v,k,2)$ je dvoravnina.

**Lema 1.3.29.** Ako je $M$ matrica incidencije simetričnog $(v,k,\lambda)$ dizajna, onda je

1. $MM^T = (k-\lambda)I + \lambda J$,
2. $M^T M = (k-\lambda)I + \lambda J$.

**Teorem 1.3.30.** *(Ryser)* U simetričnom 2-$(v,k,\lambda)$ dizajnu svaka dva bloka su incidentna s točno $\lambda$ zajedničkih točaka.

**Korolar 1.3.31.** Neka je $M$ matrica incidencije simetričnog 2-$(v,k,\lambda)$ dizajna. Tada je $M^T$ također matrica incidencije simetričnog 2-$(v,k,\lambda)$ dizajna.

**Korolar 1.3.32.** Dual simetričnog dizajna je također simetričan dizajn.

**Propozicija 1.3.33.** Neka je $M$ kvadratna $(0,1)$-matrica reda $v$ i neka su $k, \lambda \in \mathbb{N}$, $k \geq \lambda$. Ako je

$$MM^T = (k-\lambda)I + \lambda J,$$

onda je $M$ matrica incidencije simetričnog $(v,k,\lambda)$ dizajna.

**Teorem 1.3.34.** Neka je $X = \{x_1, x_2, \ldots, x_v\}$ skup i neka se skup $B$ sastoji od $k$-članih podskupova skupa $X$, tj. blokova, takvih da svaka dva različita podskupa imaju točno $\lambda$ zajedničkih elemenata. Tada je $(X, B, \mathscr{I})$ simetričan blokovni dizajn, pri čemu je $\mathscr{I}$ relacija pripadnosti.

Navest ćemo dva nužna uvjeta koje moraju zadovoljavati parametri $v,k$ i $\lambda$ da bi postojao simetrični $(v,k,\lambda)$ dizajn.

**Teorem 1.3.35.** *(Shützenberger)* Pretpostavimo da postoji simetrični $(v,k,\lambda)$ dizajn. Ako je $v$ paran broj, onda red dizajna $n$ mora biti kvadrat prirodnog broja.

**Teorem 1.3.36.** *(Bruck-Ryser-Chowla)* Pretpostavimo da postoji simetrični $(v,k,\lambda)$ dizajn. Ako je $v$ neparan broj, onda jednadžba

$$nx^2 + (-1)^{\frac{v-1}{2}} \lambda y^2 = z^2$$

mora imati cjelobrojno rješenje $(x,y,z)$, takvo da je $(x,y,z) \neq (0,0,0)$.

Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ blokovni dizajn i $G \leq \mathrm{Aut}(\mathscr{D})$. Grupa $G$ djeluje prirodno na skup točaka $\mathscr{P}$ i skup blokova $\mathscr{B}$ dizajna $\mathscr{D}$. Za to djelovanje mogu postojati fiksne točke, odnosno fiksni blokovi.

**Napomena 1.3.37.** Fiksni blokovi za djelovanje grupe $G$ na dizajn su disjunktne unije $G$-orbita točaka. Analogno vrijedi za fiksne točke pri djelovanju grupe $G$ na dizajn.

**Definicija 1.3.38.** Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ blokovni dizajn i $G \leq \mathrm{Aut}(\mathscr{D})$. Neka je $\mathscr{P}' \subseteq \mathscr{P}$ skup fiksnih točaka grupe $G$, a $\mathscr{B}' \subseteq \mathscr{B}$ skup fiksnih blokova grupe $G$. Fiksna struktura grupe $G$ je struktura $(\mathscr{P}', \mathscr{B}', \mathscr{I}')$, pri čemu je relacija $\mathscr{I}' = \mathscr{I}|_{\mathscr{P}' \times \mathscr{B}'}$.

### 1.3.1. Orbitne matrice blokovnih dizajna

Dokazi tvrdnji koje ovdje navodimo mogu se pronaći u [5] i [24].

**Definicija 1.3.39.** Dekompozicija $v \times b$ matrice $M$ je particija $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_m$ redaka od $M$ i particija $\mathscr{B}_1, \mathscr{B}_2, \ldots, \mathscr{B}_n$ stupaca od $M$.

Označimo sa $\nu_i = |\mathscr{P}_i|$, $i \in \{1, 2, \ldots, m\}$ te sa $\beta_j = |\mathscr{B}_j|$, $j \in \{1, 2, \ldots, n\}$. Označimo sa $M_{i,j}$ podmatricu dimenzija $\nu_i \times \beta_j$ koja je određena retcima iz $\mathscr{P}_i$ i stupcima iz $\mathscr{B}_j$.

Ako je suma elemenata u svakom retku matrice $M_{i,j}$ konstantna, za sve $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$, tada je dekompozicija taktička po retcima.

Ako je suma elemenata u svakom stupcu matrice $M_{i,j}$ konstantna, za sve $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$, tada je dekompozicija taktička po stupcima.

Dekompozicija matrice je taktička ako je taktička i po retcima i po stupcima.

**Definicija 1.3.40.** Neka je $M$ matrica nad poljem $F$. Rang matrice je broj linearno nezavisnih redaka (stupaca) te matrice s obzirom na operacije u polju $F$. Ako je $F$ polje reda $p$, rang matrice $M$ je $p$-rang.

**Propozicija 1.3.41.** Neka je $M$ $v \times b$ matrica ranga $v$. Za taktičku dekompoziciju matrice $M$ vrijedi:

$$0 \leq n - m \leq b - v.$$

**Definicija 1.3.42.** Neka je $\mathscr{D}$ dizajn s pripadnom matricom incidencije $M$. Particija skupa točaka i blokova dizajna $\mathscr{D}$ je taktička po točkama ako odgovara taktičkoj dekompoziciji matrice $M$ po retcima. Particija skupa točaka i blokova dizajna $\mathscr{D}$ je taktička po blokovima ako odgovara taktičkoj dekompoziciji matrice $M$ po stupcima. Particija skupa točaka i blokova dizajna $\mathscr{D}$ je taktička ako odgovara taktičkoj dekompoziciji matrice $M$.

**Propozicija 1.3.43.** Neka je $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_m$ particija skupa točaka, a $\mathscr{B}_1, \mathscr{B}_2, \ldots, \mathscr{B}_n$ particija skupa blokova dizajna $\mathscr{D}$. Tada je dekompozicija dizajna

1. taktička po točkama ako i samo ako je svaka točka iz $\mathscr{P}_i$ incidentna s istim brojem blokova iz $\mathscr{B}_j$, $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$,

2. taktička po blokovima ako i samo ako je svaki blok iz $\mathscr{B}_j$ incidentan s istim brojem točaka iz $\mathscr{P}_i$, $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$.

**Napomena 1.3.44.** Trivijalnim taktičkim dekompozicijama dizajna smatramo dekompoziciju u kojoj su klase točaka i klase blokova jednočlane te dekompoziciju koja se sastoji samo od jedne klase točaka i jedne klase blokova. Ostale taktičke dekompozicije su netrivijalne.

**Propozicija 1.3.45.** Neka grupa $G$ djeluje na dizajn $\mathscr{D}$. Tada $G$-orbite točaka i blokova daju taktičku dekompoziciju dizajna $\mathscr{D}$.

**Korolar 1.3.46.** Neka je zadan 2-$(v,k,\lambda)$ dizajn $\mathscr{D}$. Ako $G \leq \mathrm{Aut}(\mathscr{D})$ ima $m$ orbita točaka i $n$ orbita blokova, tada vrijedi:

$$0 \leq n - m \leq b - v.$$

**Korolar 1.3.47.** U simetričnom blokovnom dizajnu broj $G$-orbita točaka jednak je broju $G$-orbita blokova.

Neka je dan 2-$(v,k,\lambda)$ dizajn $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$. Označimo broj blokova dizajna $\mathscr{D}$ sa $b$. Neka je $G \leq \mathrm{Aut}(\mathscr{D})$. Neka su sa $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_m$ označene $G$-orbite točaka, a sa $\mathscr{B}_1, \mathscr{B}_2, \ldots, \mathscr{B}_n$ $G$-orbite blokova dizajna $\mathscr{D}$. Označimo duljine tih orbita sa:

$$\nu_i = |\mathscr{P}_i|, \quad i \in \{1,2,\ldots,m\},$$

$$\beta_j = |\mathscr{B}_j|, \quad j \in \{1,2,\ldots,n\}.$$

Vrijedi:

$$\sum_{i=1}^{m} \nu_i = v, \quad \sum_{j=1}^{n} \beta_j = b.$$

Za blok $x \in \mathscr{B}$ i točku $P \in \mathscr{P}$ označimo

$$\langle x \rangle = \{Q \in \mathscr{P} \,|\, (Q,x) \in \mathscr{I}\},$$

$$\langle P \rangle = \{y \in \mathscr{B} \,|\, (P,y) \in \mathscr{I}\}.$$

Budući da $G$-orbite točaka i blokova daju taktičku dekompoziciju dizajna $\mathscr{D}$, za $P \in \mathscr{P}_i$ i $x \in \mathscr{B}_j$ označimo sa

$$a_{i,j} = |\langle P \rangle \cap \mathscr{B}_j|,$$

$$b_{i,j} = |\langle x \rangle \cap \mathscr{P}_i|.$$

Tada vrijedi

$$\sum_{j=1}^{n} a_{i,j} = r, \quad i \in \{1,2,\ldots,m\},$$

$$\sum_{i=1}^{m} b_{i,j} = k, \quad j \in \{1,2,\ldots,n\}.$$

Za $a_{i,j}$ i $b_{i,j}$ vrijede i jednakosti iz sljedeće leme i propozicije.

**Lema 1.3.48.** Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ dizajn, $G \leq \mathrm{Aut}(\mathscr{D})$ te neka su $\nu_i$, $\beta_j$, $a_{i,j}$ i $b_{i,j}$, $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,n\}$ definirani kao ranije. Tada vrijedi:

1. $\nu_i a_{i,j} = \beta_j b_{i,j}$,
2. $\sum_{j=1}^{n} a_{s,j} b_{t,j} = \lambda \nu_t + \delta_{s,t}(r-\lambda), \quad s,t \in \{1,2,\ldots,m\}$,

gdje je $\delta_{s,t}$ Kroneckerov delta simbol.

**Propozicija 1.3.49.** Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ dizajn, $G \leq \mathrm{Aut}(\mathscr{D})$ te neka su $\nu_i$, $\beta_j$, $a_{i,j}$ i $b_{i,j}$, $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,n\}$ definirani kao ranije. Tada vrijedi:

1. $\sum_{j=1}^{n} a_{i,j} = r$,
2. $\sum_{j=1}^{n} \frac{\nu_t}{\beta_j} a_{s,j} a_{t,j} = \lambda \nu_t + \delta_{s,t}(r-\lambda), \quad s,t \in \{1,2,\ldots,m\}$,

gdje je $\delta_{s,t}$ Kroneckerov delta simbol.

**Definicija 1.3.50.** Točkovna orbitna matrica za parametre $(v,k,\lambda)$, distribuciju duljina orbita točaka $\nu = (\nu_1, \nu_2, \ldots, \nu_m)$ i distribuciju duljina orbita blokova $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ je svaka $m \times n$ matrica $A = [a_{i,j}]$ s elementima iz $\mathbb{N}_0$ koja ispunjava sljedeće uvjete:

1. $0 \leq a_{i,j} \leq \beta_j, \quad i \in \{1,2,\ldots,m\}, j \in \{1,2,\ldots,n\}$,
2. $\sum_{j=1}^{n} a_{i,j} = r, \quad i \in \{1,2,\ldots,m\}$,
3. $\sum_{i=1}^{m} \frac{\nu_i}{\beta_j} a_{i,j} = k, \quad j \in \{1,2,\ldots,n\}$,

4. $\displaystyle\sum_{j=1}^{n} \frac{\nu_t}{\beta_j} a_{s,j} a_{t,j} = \begin{cases} \lambda \nu_t, s \neq t, \\ \lambda(\nu_t - 1) + r, s = t, \end{cases}$

pri čemu je $\displaystyle\sum_{i=1}^{m} \nu_i = v, \quad \sum_{j=1}^{n} \beta_j = b, \quad r = \frac{v-1}{k-1}\lambda, \quad b = \frac{vr}{k}.$

**Definicija 1.3.51.** Blokovna orbitna matrica za parametre $(v,k,\lambda)$, distribuciju duljina orbita točaka $\nu = (\nu_1, \nu_2, \ldots, \nu_m)$ i distribuciju duljina orbita blokova $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ je svaka $m \times n$ matrica $B = [b_{i,j}]$ s elementima iz $\mathbb{N}_0$ koja ispunjava sljedeće uvjete:

1. $0 \leq b_{i,j} \leq \nu_i, \quad i \in \{1,2,\ldots,m\}, j \in \{1,2,\ldots,n\},$
2. $\displaystyle\sum_{i=1}^{m} b_{i,j} = k, \quad j \in \{1,2,\ldots,n\},$
3. $\displaystyle\sum_{j=1}^{n} \frac{\beta_j}{\nu_i} b_{i,j} = r, \quad i \in \{1,2,\ldots,m\},$
4. $\displaystyle\sum_{j=1}^{n} \frac{\beta_j}{\nu_s} b_{s,j} b_{t,j} = \begin{cases} \lambda \nu_t, s \neq t, \\ \lambda(\nu_t - 1) + r, s = t, \end{cases}$

pri čemu je $\displaystyle\sum_{i=1}^{m} \nu_i = v, \quad \sum_{j=1}^{n} \beta_j = b, \quad r = \frac{v-1}{k-1}\lambda, \quad b = \frac{vr}{k}.$

**Propozicija 1.3.52.** Neka je $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ simetričan blokovni dizajn, $G \leq \mathrm{Aut}(\mathscr{D})$ te neka su $a_{i,j}$, $b_{i,j}$, $\nu_i$, $\beta_j$, $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,n\}$, definirani kao ranije. Za elemente točkovne orbitne matrice $A = [a_{i,j}]$ dizajna $\mathscr{D}$ na koji djeluje grupa $G$ vrijede sljedeće jednakosti:

1. $\displaystyle\sum_{j=1}^{n} a_{i,j} = r, \quad i \in \{1,2,\ldots,m\},$
2. $\displaystyle\sum_{i=1}^{m} \frac{\nu_i}{\beta_s} a_{i,s} a_{i,t} = \lambda \beta_t + \delta_{s,t}(r - \lambda),$

gdje je $\delta_{s,t}$ Kroneckerov delta simbol.

U daljnjem radu koristit ćemo samo točkovne orbitne matrice za konstrukciju blokovnih dizajna $\mathscr{D} = (\mathscr{P}, \mathscr{B}, \mathscr{I})$ na koje djeluje grupa automorfizama $G \leq \mathrm{Aut}(\mathscr{D})$.

Konstrukcija blokovnih dizajna na koje djeluje odgovarajuća grupa automorfizama sastoji se od dva osnovna koraka:

1. konstrukcije orbitnih matrica blokovnih dizajna za djelovanje zadane grupe,
2. konstrukcije blokovnih dizajna za dobivene orbitne matrice iz prethodnog koraka, odnosno, indeksiranja orbitnih matrica.

# 1.4. GRAFOVI

Dokazi tvrdnji iz teorije grafova koje ovdje navodimo mogu se pronaći u [25].

**Definicija 1.4.1.** Graf je incidencijska struktura $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$, pri čemu je $\mathscr{V}$ neprazan skup, $\mathscr{V}$ i $\mathscr{E}$ su disjunktni skupovi te je svaki element skupa $\mathscr{E}$ incidentan s dva elementa skupa $\mathscr{V}$.

Elementi skupa $\mathscr{V}$ su vrhovi grafa, elemente skupa $\mathscr{E}$ bridovi grafa, a $\mathscr{I}$ je relacija incidencije grafa.

**Napomena 1.4.2.** Ako je brid $e$ incidentan s vrhovima $u$ i $v$, skraćeno zapisujemo $e = uv$.

**Definicija 1.4.3.** Graf $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ je jednostavan ako je svaki element skupa $\mathscr{E}$ incidentan s dva različita elementa skupa $\mathscr{V}$ i ako za bilo koja dva različita elementa $x$ i $y$ skupa $\mathscr{V}$ postoji najviše jedan element skupa $\mathscr{E}$ koji je incidentan i sa $x$ i sa $y$.

**Definicija 1.4.4.** Neka je $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ graf. Ako su skupovi $\mathscr{V}$ i $\mathscr{E}$ konačni skupovi, je graf $\mathscr{G}$ je konačan. U suprotnom je graf $\mathscr{G}$ beskonačan.

**Definicija 1.4.5.** Graf koji ima samo jedan vrh je trivijalan graf, a inače je netrivijalan.

Graf $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ je prazan ako je $\mathscr{E} = \emptyset$.

**Napomena 1.4.6.** Bridovi incidentni s dva različita vrha su pravi bridovi.

Općenito, grafovi mogu sadržavati i bridove koji su incidentni samo s jednim vrhom i takvi bridovi su petlje.

Grafovi mogu sadržavati i više bridova koji su incidentni s istim parom vrhova, tj. višestruke bridove.

Jednostavni grafovi su grafovi bez petlji i višestrukih bridova.

**Definicija 1.4.7.** Graf $\mathscr{G}' = (\mathscr{V}', \mathscr{E}', \mathscr{I}')$ je podgraf grafa $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$, u oznaci $\mathscr{G}' \subseteq \mathscr{G}$, ako je podstruktura grafa $\mathscr{G}$ kao incidencijske strukture.

**Definicija 1.4.8.** Dva vrha grafa $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ su susjedna ako su incidentna s istim bridom.

Dva brida grafa $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ su susjedna ako su incidentna s istim vrhom.

**Definicija 1.4.9.** Neka je $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ graf i $x \in \mathscr{V}$ vrh grafa. Broj bridova incidentnih s vrhom $x$ je stupanj vrha $x$ i označava $d_{\mathscr{G}}(x)$. Pri tome, petlje ubrajamo dva puta. Izolirani vrh je vrh stupnja 0.

**Teorem 1.4.10.** *(Lema o rukovanju)* Neka je $\mathscr{G}$ graf s $v$ vrhovima $x_1,\ldots,x_v$ i $\varepsilon$ bridova. Tada je

$$\sum_{i=1}^{v} d_{\mathscr{G}}(x_i) = 2\varepsilon.$$

**Definicija 1.4.11.** Put u grafu $\mathscr{G}$ je netrivijalan niz $x_0e_1x_1e_2\ldots x_r$ kojemu su svi vrhovi i svi bridovi međusobno različiti, pri čemu su $x_0,x_1,\ldots,x_r$ vrhovi grafa $\mathscr{G}$ i $e_i,\ i=1,2,\ldots,r$, bridovi koji su incidentni s vrhovima $x_{i-1}$ i $x_i$.

**Definicija 1.4.12.** Graf $\mathscr{G}$ je povezan graf ako za svaka dva vrha tog grafa postoji put koji ih povezuje.

**Definicija 1.4.13.** Matrica incidencije grafa $\mathscr{G}$ sa $v$ vrhova $x_1,x_2,\ldots,x_v$ i $\varepsilon$ bridova $e_1,e_2,\ldots,e_\varepsilon$ je $v\times\varepsilon$ matrica $M=[m_{i,j}]$ takva da $m_{i,j}\in\{0,1,2\}$ broji koliko su puta vrh $v_i$ i brid $e_j$ incidentni.

**Definicija 1.4.14.** Matrica susjedstva grafa $\mathscr{G}$ sa $v$ vrhova $x_1,x_2,\ldots,x_v$ je $v\times v$ matrica $M=[m_{i,j}]$ takva da je $m_{i,j}$ broj bridova incidentnih s vrhovima $x_i$ i $x_j$ .

**Napomena 1.4.15.** Matrica susjedstva grafa $\mathscr{G}$ je simetrična matrica (tj. $m_{i,j}=m_{j,i}$), čiji su elementi nenegativni cijeli brojevi.

Ako je graf $\mathscr{G}$ jednostavan, ova matrica sadrži samo nule i jedinice te nule na glavnoj dijagonali.

**Definicija 1.4.16.** Graf u kojem su svi vrhovi jednakog stupnja $k$ je $k$-regularan graf.

**Definicija 1.4.17.** Komplement $\mathscr{G}^C$ grafa $\mathscr{G}$ je komplementarna incidencijska struktura od $\mathscr{G}$.

**Napomena 1.4.18.** Ukoliko označimo s $M$ matricu reda $v$ susjedstva grafa $\mathscr{G}$, onda ćemo matricu susjedstva komplementa $\mathscr{G}^C$ tog grafa označavati sa $M^C$. Vrijedi:

$$M^C = J - M - I,$$

pri čemu je $I$ jedinična matrica reda $v$, a $J$ matrica reda $v$ čiji su svi elementi jedinice.

**Definicija 1.4.19.** Neka su zadana dva grafa: $\mathscr{G}_1=(\mathscr{V}_1,\mathscr{E}_1,\mathscr{I}_1)$ i $\mathscr{G}_2=(\mathscr{V}_2,\mathscr{E}_2,\mathscr{I}_2)$. Izomorfizam iz grafa $\mathscr{G}_1$ u graf $\mathscr{G}_2$ je izomorfizam tih grafova kao incidencijskih struktura. Analogno definiramo i automorfizam grafa $\mathscr{G}$.

### 1.4.1. Jako regularni grafovi

Dokazi tvrdnji koje ovdje navodimo mogu se pronaći u [10].

**Definicija 1.4.20.** Neka je $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ jednostavan $k$-regularan graf koji ima $v$ vrhova. Graf $\mathscr{G}$ je jako regularan graf s parametrima $(v,k,\lambda,\mu)$ ako je broj zajedničkih susjeda svaka dva različita vrha $x$ i $y$ jednak

- $\lambda$ ako su susjedni
- $\mu$ ako nisu susjedni.

**Napomena 1.4.21.** Kao oznaku za jako regularne grafove s parametrima $(v,k,\lambda,\mu)$ koristit ćemo SRG$(v,k,\lambda,\mu)$.

**Teorem 1.4.22.** Neka je $\mathscr{G}$ jako regularan graf s parametrima $(v,k,\lambda,\mu)$. Tada je

$$k(k-\lambda-1) = \mu(v-k-1).$$

**Teorem 1.4.23.** Simetrična $(0,1)$-matrica $B$ reda $v$, s nulama na dijagonali, je matrica susjedstva jako regularnog grafa $\mathscr{G}$ s parametrima $(v,k,\lambda,\mu)$ ako i samo ako vrijedi

$$B^2 = (k-\mu)I + \mu J + (\lambda-\mu)B,$$

pri čemu je $I$ jedinična matrica reda $v$, a $J$ matrica reda $v$ čiji su svi elementi jedinice.

**Teorem 1.4.24.** Komplement jako regularnog grafa $\mathscr{G}$ s parametrima $(v,k,\lambda,\mu)$ je jako regularan graf $\mathscr{G}^C$ s parametrima

$$(v, v-k-1, v-2k+\mu-2, v-2k+\lambda).$$

Neka je $G$ grupa automorfizama jako regularnog grafa $\mathscr{G}$ s parametrima $(v,k,\lambda,\mu)$. Grupa $G$ djeluje na graf $\mathscr{G}$ te pri tom djelovanju mogu postojati fiksni vrhovi.

### 1.4.2. Orbitne matrice jako regularnih grafova

Dokazi tvrdnji koje ovdje navodimo mogu se pronaći u [3] i [4].

Neka je $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{I})$ jako regularan graf s parametrima $(v,k,\lambda,\mu)$. Neka je $G$ grupa automorfizama grafa $\mathscr{G}$. Pretpostavimo da se pod djelovanjem grupe $G$ skup vrhova grafa $\mathscr{G}$ particionira u $b$ orbita. Označimo $G$-orbite za to djelovanje sa $O_1, O_2, \ldots, O_b$. Duljinu orbite

$O_i$ označimo sa $n_i$, tj. neka je $|O_i| = n_i$, za sve $i = 1, 2, \ldots, b$. Označimo vrhove grafa na način da se očuva uređaj orbita, odnosno, neka za svaki vrh $v_i$ iz orbite $O_k$ i svaki vrh $v_j$ iz orbite $O_l$ vrijedi: ako je $k \leq l$, onda je $i \leq j$.

Orbite na skupu vrhova induciraju particiju redaka i stupaca matrice susjedstva $B$ grafa $\mathscr{G}$. Matrica susjedstva je, uz ovakvu particiju redaka i stupaca, particionirana matrica susjedstva za djelovanje grupe $G$. Podmatrica $B_{i,j}$ u particioniranoj matrici susjedstva je matrica susjedstva vrhova orbite $O_i$ i vrhova orbite $O_j$.

Neka je $B$ particionirana matrica susjedstva jako regularnog grafa $\mathscr{G}$ za djelovanje grupe $G$. Iz particionirane matrice susjedstva $B$ možemo dobiti stupčanu orbitnu matricu grafa $\mathscr{G}$ za djelovanje grupe $G$, u oznaci $C = [c_{i,j}]$, gdje je $c_{i,j}$ suma unutar jednog stupca podmatrice $B_{i,j}$. Iz particionirane matrice susjedstva $B$ možemo dobiti i retčanu orbitnu matricu grafa $\mathscr{G}$ za djelovanje grupe $G$, u oznaci $R = [r_{i,j}]$, gdje je $r_{i,j}$ suma unutar jednog retka podmatrice $B_{i,j}$.

Promatrajmo orbitu $O_i$ i orbitu $O_j$. Sa $x_j$ označimo predstavnika orbite $O_j$. Neka je $x_j$ susjedan s točno $u$ elemenata orbite $O_i$. Onda je, po definiciji automorfizma jako regularnog grafa, slika elementa $x_j$ susjedna s točno $u$ elemenata orbite $O_i$. Analogno dolazimo do toga da je svaki element orbite $O_j$ susjedan s točno $u$ elemenata orbite $O_i$.

Zbog navedenog dolazimo do toga da su $c_{i,j}$ i $r_{i,j}$ dobro definirani, odnosno ne ovise o izboru predstavnika orbite. Za $r$-ti redak (stupac) stupčane (retčane) orbitne matrice kažemo da odgovara orbiti $O_r$.

Matrica susjedstva jako regularnog grafa je simetrična matrica pa vrijedi da je

$$R = C^T.$$

Podmatrica $B_{i,j}$ ima $n_j$ stupaca i $n_i$ redaka, suma unutar svakog stupca je $c_{i,j}$, a suma unutar svakog retka je $r_{i,j}$ . Zbrojimo li sve elemente podmatrice $B_{i,j}$ dobivamo $r_{i,j}n_i$, odnosno $c_{i,j}n_j$, iz čega slijedi sljedeća jednakost:

$$r_{i,j} = c_{i,j}\frac{n_j}{n_i},$$

odnosno,

$$c_{j,i} = c_{i,j}\frac{n_j}{n_i}.$$

Neka je $W = B^2$. Particionirajmo matricu $W$ na podmatrice $W_{i,j}$ tako da se očuva uređaj orbita. Iz matrice $W$ možemo dobiti matricu $S = [s_{i,j}]$ reda $b$ za koju vrijedi da je $s_{i,j}$ suma svih elemenata podmatrice $W_{i,j}$.

**Teorem 1.4.25.** Neka je $B$ matrica susjedstva jako regularnog grafa $\mathscr{G}$ s parametrima $(v,k,\lambda,\mu)$ na kojeg djeluje grupa automorfizama $G$, pri čemu se skup vrhova particionira u $b$ orbita. Duljine orbita označimo sa $n_i$, $i = 1,2,\ldots,b$. Neka je $N$ dijagonalna matrica takva da je na $i$-tom dijagonalnom mjestu duljina $i$-te orbite. Onda vrijedi

$$CNR = S.$$

Neka je $W = B^2$. Imamo:

$$W_{i,j} = \delta_{i,j}(k-\mu)I + \mu J + (\lambda-\mu)B_{i,j},$$

gdje je $\delta_{i,j}$ Kroneckerov delta simbol. Dimenzije matrice $I$ u ovoj jednakosti su $n_i \times n_i$, a dimenzije matrice $J$ su $n_i \times n_j$. Vrijedi da je suma svih elemenata podmatrice $W_{i,j}$ jednaka

$$s_{i,j} = \delta_{i,j}(k-\mu)n_j + \mu n_i n_j + (\lambda-\mu)c_{i,j}n_j.$$

Na temelju do sada opisanog, definiramo retčane i stupčane orbitne matrice.

**Definicija 1.4.26.** Matrica $R = [r_{i,j}]$ dimenzija $b \times b$ čiji elementi zadovoljavaju sljedeće jednakosti:

1. $\displaystyle\sum_{j=1}^{b} r_{i,j} = \sum_{i=1}^{b} \frac{n_i}{n_j} r_{i,j} = k,$

2. $\displaystyle\sum_{s=1}^{b} \frac{n_s}{n_j} r_{s,i} r_{s,j} = \delta_{i,j}(k-\mu) + \mu n_i + (\lambda-\mu)r_{j,i},$

pri čemu je $0 \leq r_{i,j} \leq n_j$, $0 \leq r_{i,i} \leq n_i - 1$ i $\displaystyle\sum_{i=1}^{b} n_i = v$ je retčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,\ldots,n_b)$.

**Teorem 1.4.27.** Za retčanu orbitnu matricu $R = [r_{i,j}]$ za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,...,n_b)$ vrijedi da je

$$r_{i,j} = \frac{n_j}{n_i} r_{j,i}.$$

**Definicija 1.4.28.** Matrica $C = [c_{i,j}]$ dimenzija $b \times b$ čiji elementi zadovoljavaju sljedeće jednakosti:

1. $\sum_{i=1}^{b} c_{i,j} = \sum_{j=1}^{b} \frac{n_j}{n_i} c_{i,j} = k,$

2. $\sum_{s=1}^{b} \frac{n_s}{n_j} c_{i,s} c_{j,s} = \delta_{i,j}(k-\mu) + \mu n_i + (\lambda - \mu) c_{i,j},$

pri čemu je $0 \leq c_{i,j} \leq n_i$, $0 \leq c_{i,i} \leq n_i - 1$ i $\sum_{i=1}^{b} n_i = v$ je stupčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,\ldots,n_b)$.

**Teorem 1.4.29.** Za stupčanu orbitnu matricu $C = [c_{i,j}]$ za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,...,n_b)$ vrijedi da je

$$c_{i,j} = \frac{n_i}{n_j} c_{j,i}.$$

**Napomena 1.4.30.** Neka je $C = [c_{i,j}]$ stupčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,...,n_b)$. Njoj odgovarajuća retčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,...,n_b)$ je $R = C^T$.

**Teorem 1.4.31.** Neka su $N = [n_{i,j}]$ i $S = [s_{i,j}]$ matrice reda $b$ takve da je

$$n_{i,j} = \begin{cases} n_i, \, i = j, \\ 0, \, i \neq j, \end{cases}$$

$$s_{i,j} = \delta_{i,j}(k-\mu) n_j + \mu n_j n_i + (\lambda - \mu) c_{i,j} n_j,$$

pri čemu je $\sum_{i=1}^{b} n_i = v$.

$C = [c_{i,j}]$ je stupčana i $R = [r_{i,j}]$ je njoj odgovarajuća retčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1,n_2,\ldots,n_b)$ ako i samo ako je

$$CNR = S,$$

gdje je $C = R^T$ i $\sum_{i=1}^{b} c_{i,j} = k$, $0 \leq c_{i,j} \leq n_i$, $0 \leq c_{i,i} \leq n_i - 1$.

**Napomena 1.4.32.** Neka je $\mathscr{G}$ jako regularan graf s parametrima $(v,k,\lambda,\mu)$. Neka grupa automorfizama $G$ djeluje na vrhove jako regularnog grafa pri čemu particionira skup vrhova u $b$

orbita čije su duljine $n_1, n_2, \ldots, n_b$. Onda je stupčana (retčana) orbitna matrica grafa $\mathscr{G}$ također i stupčana (retčana) orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1, n_2, ..., n_b)$.

Obrat ne vrijedi, odnosno ukoliko je matrica $C$ stupčana (retčana) orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribuciju duljina orbita $(n_1, n_2, ..., n_b)$ to ne mora značiti da je to orbitna matrica jako regularnog grafa s parametrima $(v,k,\lambda,\mu)$.

# 2. GENETSKI ALGORITMI

U ovom poglavlju prikazat ćemo kratki povijesni pregled evolucijskog računanja i genetskih algoritama. Opisat ćemo kako su genetski algoritmi modelirani kao imitacija procesa prirodne evolucije. Definirat ćemo osnovne pojmove vezane uz genetske algoritme, kao što su prostor pretraživanja, krajolik dobrote, funkcija dobrote, križanje, mutacija i selekcija. Opisat ćemo na koji način radi jednostavni genetski algoritam i prikazati pseudokod takvog algoritma.

## 2.1. UVOD

Znanost proizlazi iz same ljudske želje da razumije i kontrolira svijet. Tijekom povijesti, mi, ljudi, smo postupno izgradili veliko zdanje znanja koje nam omogućuje da predvidimo, u različitoj mjeri, vrijeme, kretanje planeta, pomrčine Sunca i Mjeseca, tijek bolesti, uspon i pad ekonomskog rasta, faze jezičnog razvoja kod djece i veliki niz drugih prirodnih, društvenih i kulturnih fenomena. U novije vrijeme čak smo shvatili neke temeljne granice naše sposobnosti predviđanja. Kroz vrijeme, razvili smo sve složenija sredstva za kontrolu mnogih aspekata naših života i naših interakcija s prirodom, i naučili smo, često na teži način, u kojoj mjeri ostale aspekte nije moguće kontrolirati.

Pojava računala nedvojbeno je najrevolucionarniji razvoj u povijesti znanosti i tehnologije. Ova stalna revolucija duboko povećava našu sposobnost predviđanja i kontrole prirode na načine koji su prije stotinjak godina jedva mogli bili zamišljeni. Za mnoge, kruna postignuća ove revolucije bit će stvaranje - u obliku računalnih programa - novih vrsta inteligentnih bića, pa čak i novih oblika života. Ciljevi stvaranja umjetne inteligencije i umjetnog života mogu se pratiti od samih početaka kompjuterskog doba. Najraniji računalni znanstvenici - matematičari Alan Turing, John von Neumann, Norbert Wiener i drugi - bili su motivirani velikim dijelom vizijama o prožimanju računalnih programa inteligencijom, životnom sposobnosti samorepliciranja i prilagodljivom sposobnosti učenja i kontrole okoline. Ovi rani pioniri računalne znanosti bili su

zainteresirani za biologiju i psihologiju koliko i za elektroniku, a prirodne su sustave promatrali kao vodilju za postizanje svojih vizija. Stoga ne bi trebalo biti iznenađenje da su se od najranijih dana računala primjenjivala ne samo za proračun putanje projektila i dešifriranje vojnih kodova, već i za simuliranje modela mozga, oponašajući ljudsko učenje i biološku evoluciju. Ove biološki motivirane računalne aktivnosti su se povećavale i smanjivale tijekom godina, ali od ranih 1980-ih doživjele su ponovno oživljavanje u računalnim istraživanjima. Tako su se razvila područja neuronskih mreža, strojnog učenja i onoga što se danas naziva "evolucijsko računanje", čiji su primjer genetski algoritmi.

## 2.2. POVIJEST EVOLUCIJSKOG RAČUNANJA

U 1950-im i 1960-im godinama nekoliko je matematičara i računalnih znanstvenika neovisno proučavalo evolucijske sustave s idejom da se evolucija može koristiti kao alat za optimizaciju za inženjerske probleme. Ideja u svim tim sustavima bila je razviti populaciju kandidata rješenja za određeni problem, koristeći operatore inspirirane prirodnom genetskom varijacijom i prirodnom selekcijom.

Šezdesetih godina prošlog stoljeća Rechenberg [51], [52] uvodi "evolucijske strategije" (njem. *Evolutionsstrategie*), metodu koju je koristio za optimizaciju realnih parametara za uređaje kao što su aeroprofili. Ovu ideju je dalje sedamdesetih godina razvijao Schwefel [56], [57]. Fogel, Owens i Walsh [27] razvili su "evolucijsko programiranje", tehniku u kojoj su kandidati za rješenja zadanih problema bili predstavljeni kao konačni automati, koji su evoluirali nasumično mutirajući svoje dijagrame prijelaza stanja i birajući najsposobnije. Zajedno, evolucijske strategije, evolucijsko programiranje i genetski algoritmi čine okosnicu područja evolucijskog računanja.

Nekoliko drugih ljudi koji su radili 1950-ih i 1960-ih razvili su algoritme inspirirane evolucijom za optimizaciju i strojno učenje. Box [6], Friedman [30], Bremermann [8] i Reed, Toombs i Baricelli [53] svi su radili na ovom području, iako je njihovom radu dano malo ili nimalo pažnje ili praćenja koje su dobile evolucijske strategije, evolucijsko programiranje i genetski algoritmi. Osim toga, brojni evolucijski biolozi koristili su računala za simulaciju evolucije u svrhu kontroliranih eksperimenata, npr. Baricelli [1], [2]; Fraser [28], [29]; Martin i Cockerham [44].

Genetske algoritme (GA) izumio je John Holland 1960-ih, a razvili su ih Holland i njegovi studenti i kolege na Sveučilištu Michigan 1960-ih i 1970-ih. U suprotnosti s evolucijskim strategijama i evolucijskim programiranjem, Hollandov izvorni cilj nije bio dizajnirati algoritme za rješavanje specifičnih problema, već formalno proučavati fenomen prilagodbe kakav se javlja u prirodi i razviti načine na koje bi se mehanizmi prirodne prilagodbe mogli uvesti u računalne sustave. Hollandova knjiga *Adaptation in Natural and Artificial Systems* [33] iz 1975. predstavila je genetski algoritam kao apstrakciju biološke evolucije i dala teorijski okvir za prilagodbu prema genetskom algoritmu. Hollandov genetski algoritam je metoda za prelazak iz jedne populacije "kromosoma", zovemo ih također "jedinke", (npr. nizovi jedinica i nula ili "bitova") u novu populaciju korištenjem svojevrsne "prirodne selekcije" zajedno s genetikom inspirira-

nim operatorima križanja (engl. *crossover*), mutacija i inverzija. Svaki se kromosom sastoji od "gena" (npr. bitova), pri čemu je svaki gen primjerak određenog "alela" (npr. 0 ili 1). Operator selekcije na temelju sposobnosti (pogodnosti, dobrote - engl. *fitness*) kromosoma odabire one kromosome u populaciji kojima će biti dopušteno razmnožavanje, a kromosomi koji su sposobniji u prosjeku proizvode više "djece" (potomaka) od onih manje sposobnih. Križanje izmjenjuje poddijelove dvaju kromosoma, ugrubo oponašajući biološku rekombinaciju između dva jednokromosomska ("haploidna") organizma. Mutacija nasumično mijenja vrijednosti alela nekih pozicija u kromosomu, a inverzija obrće redoslijed dijela kromosoma, čime se preuređuje redoslijed u kojem su geni raspoređeni.

Hollandovo uvođenje algoritma temeljenog na populaciji s križanjem, inverzijom i mutacijom bila je velika inovacija. Rechenbergove evolucijske strategije počele su s "populacijom" od dvije jedinke, jednog roditelja i jednog djeteta, pri čemu je dijete mutirana verzija roditelja; populacije s više jedinki i križanje uvedeni su kasnije. Evolucijsko programiranje Fogela, Owensa i Walsha koristilo je samo mutaciju koja osigurava varijaciju. Štoviše, Holland je bio prvi koji je pokušao dati čvrste teorijske osnove računalnoj evoluciji. Njegovo teorijsko utemeljenje, utemeljeno na pojmu "shema", bila je osnova mnogih kasnijih teorijskih radova o genetskim algoritmima.

## 2.3. EVOLUCIJA I BIOLOŠKA TERMINOLOGIJA

Postavlja se pitanje zašto koristiti evoluciju kao inspiraciju za rješavanje računalnih problema. Za istraživače koji se bave evolucijskim računanjem, mehanizmi evolucije čine se prikladnima za brojne računalne probleme na mnogim poljima. Mnogi računski problemi zahtijevaju pretraživanje kroz ogroman broj mogućnosti rješenja. Jedan primjer je problem računalnog proteinskog inženjerstva, u kojem se traži algoritam koji će tražiti protein među golemim brojem mogućih sekvenci aminokiselina sa specificiranim svojstvima. Drugi primjer je traženje skupa pravila ili jednadžbi koje će predvidjeti uspone i padove financijskog tržišta, kao što je ono za stranu valutu. U evolucijskom računanju pravila su obično "prirodna selekcija" s varijacijama zbog križanja i/ili mutacija. Očekivano ponašanje takvih sustava je da se stvaraju visokokvalitetna rješenja za teške probleme i sposobnost prilagođavanja ovih rješenja pri suočavanju s promjenjivim okruženjem.

Biološka evolucija je privlačan izvor inspiracije za rješavanje ovakvih problema. Evolucija je, u suštini, metoda pretraživanja među golemim brojem mogućnosti "rješenja". U biologiji je ogroman skup mogućih genetskih sekvenci, a željena "rješenja" su visoko pogodni organizmi, tj. organizmi koji su vrlo dobro sposobni preživjeti i razmnožavati se u svom okruženju. Evolucija se također može vidjeti kao metoda za osmišljavanje inovativnih rješenja složenih problema. Na primjer, imunološki sustav sisavaca je čudesno razvijeno rješenje za problem invazije virusa i bakterija u tijelo. Gledano u ovom svjetlu, mehanizmi evolucije mogu potaknuti metode računalnog pretraživanja. Naravno, sposobnost biološkog organizma ovisi o mnogim čimbenicima - na primjer, koliko dobro može podnijeti fizikalne karakteristike svojega okoliša i koliko dobro se može natjecati s drugim organizmima oko sebe ili surađivati s njima. Kriteriji sposobnosti se neprestano mijenjaju kako stvorenja evoluiraju, tako da evolucija pretražuje konstantno promjenjivi skup mogućnosti. "Pravila" evolucije su izuzetno jednostavna: vrste se razvijaju pomoću slučajnih varijacija (preko mutacija, rekombinacija i drugih operatora), nakon čega slijedi prirodna selekcija u kojoj najsposobniji teže tome da prežive i razmnožavaju se, šireći tako svoj genetski materijal na buduće generacije. Iako su jednostavna, ova pravila smatraju se velikim dijelom odgovornima za izuzetnu raznolikost i složenost koju vidimo u biosferi.

Uvedimo dio biološke terminologije koja se koristi u kontekstu evolucijskog računanja i genetskih algoritama. Ovi biološki pojmovi koriste se u duhu analogije sa stvarnim pojmovima iz biologije, iako su entiteti na koje se odnose mnogo jednostavniji od pravih bioloških.

Svi živi organizmi sastoje se od stanica, a svaka stanica sadrži isti skup jednog ili više kromosoma - nizova DNK - koji služe kao "nacrt" za organizam. Kromosom može biti konceptualno podijeljen na gene - od kojih svaki kodira određeni protein. Vrlo ugrubo, može se zamisliti da gen kodira osobinu, kao što je, na primjer, boja očiju. Različite moguće "postavke" za osobinu (npr. plava, smeđa, zelena) nazivaju se aleli. Svaki gen se nalazi na određenom lokusu (poziciji) u kromosomu.

Mnogi organizmi imaju više kromosoma u svakoj stanici. Kompletna zbirka genetskog materijala (svih kromosoma zajedno) naziva se genom organizma. Pojam genotip odnosi se na određeni skup gena sadržanih u genomu. Kaže se da dvije osobe koje imaju identične genome imaju isti genotip. Genotip stvara, u fetalnom i kasnijem razvoju, fenotip organizma - njegove fizičke i mentalne karakteristike, kao što su boja očiju, visina, veličina mozga i inteligencija.

Organizmi čiji su kromosomi raspoređeni u parovima nazivaju se diploidni, a organizmi čiji su kromosomi nespareni se nazivaju haploidni. U prirodi je većina spolno razmnožavajućih vrsta diploidna, uključujući ljudska bića, od kojih svaki ima 23 para kromosoma u svakoj somatskoj (ne-zametnoj) stanici u tijelu. Tijekom spolnog razmnožavanja, dolazi do rekombinacije (ili križanja): u svakom roditelju geni se izmjenjuju između svakog para kromosoma kako bi formirali gametu (jedan kromosom), a zatim se gamete od dva roditelja uparuju kako bi stvorili puni skup diploidnih kromosoma. U haploidnom spolnom razmnožavanju geni se izmjenjuju između jednolančanih kromosoma dvaju roditelja. Potomstvo je podložno mutaciji, u kojoj se pojedinačni nukleotidi (elementarni dijelovi DNK) mijenjaju iz roditelja u potomstvo, a promjene često proizlaze iz pogrešaka tijekom kopiranja. Sposobnost organizma se obično definira kao vjerojatnost da će organizam doživjeti da se razmnožava (živost) ili kao funkcija broja potomaka koje organizam ima (plodnost).

U genetskim algoritmima pojam "kromosom" ili "jedinka" obično se odnosi na kandidata za rješenje problema. Najjednostavniji prikaz jedinke u računalu je niz bitova. "Geni" su ili pojedinačni bitovi ili kratki blokovi susjednih bitova koji kodiraju određeni element rješenja kandidata. Alel u takvom nizu bitova je ili 0 ili 1; za veće alfabete moguće je više alela na svakom lokusu. Križanje se obično sastoji od razmjene genetskog materijala između dva jednokromosomska haploidna roditelja. Mutacija se sastoji u zamjeni bita na nasumično odabranom odabranom lokusu (ili, za veće alfabete, zamjenu simbola na slučajno odabranom lokusu slučajno odabranim novim simbolom).

Većina primjena genetskih algoritama koristi haploidne jedinke, posebno jednokromosom-

ske jedinke. Genotip jedinke u genetskom algoritmu koji koristi nizove bitova jednostavno je konfiguracija bitova u kromosomu toga pojedinca. Pojam "fenotip" u kontekstu genetskih algoritama se rijetko koristi, ali ima područja u kojima postoji i genotipska i fenotipska razina (npr. kodiranje niza bitova neuronske mreže i same neuronske mreže).

# 2.4. PROSTOR PRETRAŽIVANJA I KRAJOLIK DOBROTE

Prostor pretraživanja željenog rješenja među kolekcijom kandidata za rješenje skraćeno nazivamo "prostor pretraživanja" (engl. *search space*). Pojam "prostor pretraživanja" može se odnositi na neku kolekciju kandidata za rješenja problema zajedno s "udaljenosti" između kandidata za rješenje. Za primjer, uzmimo jedan od najvažnijih problema u računalnom bioinženjeringu: prethodno spomenuti problem računalnog dizajna proteina. Pretpostavimo da želimo koristiti računalo za traženje proteina - niza aminokiselina - koji se može saviti u određeni trodimenzionalni oblik tako da se može koristiti, recimo, za borbu protiv određenog virusa. Prostor pretraživanja je kolekcija svih mogućih nizova proteina, tj. beskonačan skup mogućnosti. Da bi postao konačan, ograničimo pretragu na sve moguće sekvence duljine 100 ili manje, što je i dalje ogroman prostor za pretraživanje, budući da postoji 20 mogućih aminokiselina na svakoj poziciji u nizu. Ako 20 aminokiselina predstavimo slovima abecede, rješenja kandidata će izgledati ovako:

A G G M C G B L. . . .

Razmak između dva niza ćemo definirati kao Hammingovu udaljenost, tj. broj pozicija na kojima se slova razlikuju. Na primjer, udaljenost između nizova

A G G M C G B L i M G G M C G B L

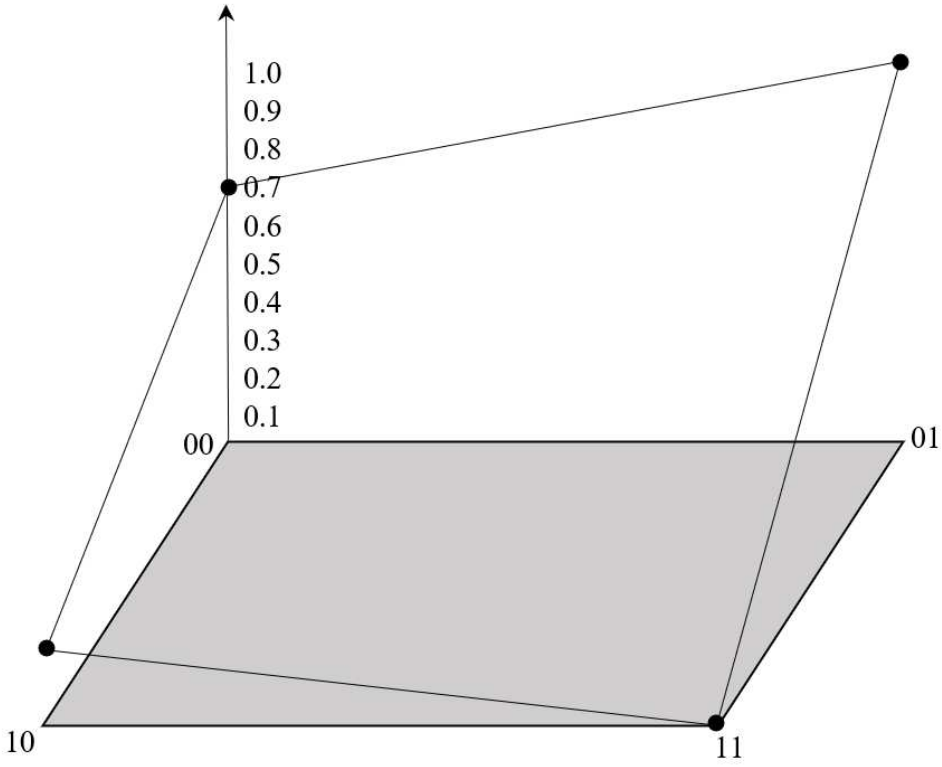
je 1, a udaljenost između nizova

A G G M C G B L i L B M P A F G A

je 8. Algoritam za pretraživanje ovog prostora je metoda za odabir kandidata za rješenje kojeg ćemo testirati u svakoj fazi pretraživanja. U većini slučajeva sljedeći kandidat za rješenje(a) kojeg će se testirati ovisit će o rezultatima testiranja prethodnih sekvenci; najkorisniji algoritmi podrazumijevaju da će postojati neka korelacija između kvalitete "susjednih" kandidata za rješenje - onih bliskih u prostoru. Genetski algoritmi podrazumijevaju da se visokokvalitetna "roditeljska" kandidatska rješenja iz različitih regija u prostoru mogu kombinirati putem križanja kako bi se, po mogućnosti, proizvela kandidatska rješenja "potomaka".

Drugi važan koncept je koncept "krajolika dobrote" (engl. *fitness landscape*). Pojam krajolika dobrote je izvorno definirao 1931. godine biolog Sewell Wright u [63] u kontekstu populacijske genetike, a podrazumijeva prikaz prostora svih mogućih genotipa zajedno s njihovim dobrotama.

Pretpostavimo, radi jednostavnosti, da je svaki genotip niz bitova duljine $l$ te da je udaljenost između dva genotipa njihova Hammingova udaljenost - broj mjesta na kojima se odgovarajući bitovi razlikuju. Također pretpostavimo da se svakom genotipu može dodijeliti realna vrijednost dobrote. Krajolik dobrote se može vizualizirati kao $(l+1)$-dimenzionalni prikaz u kojem je svaki genotip $l$-dimenzionalna točka, a njegova dobrota $f$ ucrtana je duž $(l+1)$-osi. Jednostavan krajolik za $l = 2$ prikazan je na slici 2.1.




Slika 2.1: Jednostavni krajolik dobrote za $l = 2$. Ovdje je $f(00) = 0.7$, $f(01) = 1.0$, $f(10) = 0.1$ i $f(11) = 0.0$.

Takve prikaze nazivamo krajolicima jer graf vrijednosti dobrote može formirati "brda", "vrhove", "doline" i druge značajke analogne onima u prirodnim krajolicima. Prema Wrightovoj formulaciji, evolucija uzrokuje kretanje stanovništva po krajolicima na određene načine, a "prilagodba" se može promatrati kao kretanje prema lokalnim vrhovima. "Lokalni vrh" ili "lokalni

optimum" nije nužno najviša točka u krajoliku, već bilo koji mali pomak od njega uzrokuje snižavanje dobrote. Na sličan način, u genetskim algoritmima operatori križanja i mutacije uzrokuju kretanje populacije po krajoliku definiranom funkcijom dobrote.

Ideja evolucije koja pomiče populacije u nepromjenjivim krajolicima je biološki nerealna iz nekoliko razloga. Na primjer, organizmu se ne može dodijeliti vrijednost dobrote neovisna o drugom organizmu u njegovom okolišu; tako će se, kako se populacija mijenja, dobrote pojedinih genotipova također mijenjati. Drugim riječima, u stvarnom svijetu "krajolik" se ne može odvojiti od organizama koji ga nastanjuju. Pojam krajolika dobrote postao je jedan od središnjih u proučavanju genetskih algoritama.

# 2.5. ELEMENTI GENETSKIH ALGORITAMA

Pokazalo se da ne postoji rigorozna definicija "genetskog algoritma" prihvaćena među znanstvenicima koji se bave evolucijskim računalstvom kojom bi se genetski algoritmi razlikovali od ostalih metoda evolucijskog računanja. Međutim, može se reći da većina metoda koje se nazivaju "genetskim algoritmima" ima barem sljedeće zajedničke elemente: populacije jedinki, selekciju prema dobroti, križanje u svrhu proizvodnje novog potomstva i slučajne mutacije novog potomstva.

Jedinke u populaciji genetskog algoritma često imaju oblik nizova bitova, pri čemu svaki lokus unutar jedinke ima dva moguća alela: 0 ili 1. Međutim, ovisno o potrebi mogu se koristiti i drugi oblici. Svaka se jedinka može smatrati točkom u prostoru pretraživanja kandidata za rješenje. Genetski algoritam obrađuje populacije jedinki, sukcesivno zamjenjujući jednu takvu populaciju drugom. Genetski algoritam najčešće zahtijeva funkciju dobrote koja svakoj jedinki u trenutnoj populaciji dodjeljuje ocjenu - dobrotu koja ovisi o tome koliko dobro ta jedinka rješava predmetni problem.

## 2.5.1. Primjeri funkcija dobrote

Navedimo dva primjera korištenja genetskog algoritma i načina na koje se može definirati funkcija dobrote. Prvi primjer je numeričke prirode.

**Primjer 2.5.1.** Jedna uobičajena primjena genetskog algoritma je optimizacija funkcije, gdje je cilj pronaći skup vrijednosti varijabli koje maksimiziraju funkciju više varijabli. Kao jednostavan primjer, recimo da želimo maksimizirati realnu funkciju s jednom varijablom

$$f(x) = x + |\sin(32x)|, \quad 0 \leq x \leq \pi.$$

Kao što je opisano u [54], ovdje su kandidati za rješenje vrijednosti varijable $x$, koje se mogu kodirati kao nizovi bitova koji predstavljaju realne brojeve. Izračun dobrote pretvara zadani niz bitova u realni broj $x$, a zatim evaluira funkcijsku vrijednost u toj točki. Dobrota niza bitova je vrijednost funkcije u toj točki.

Razmotrimo sada i jedan nenumerički primjer.

**Primjer 2.5.2.** Zadan je problem pronalaženja niza od 50 aminokiselina koji se može saviti u željenu trodimenzionalnu strukturu proteina. Genetski algoritam bi se mogao primijeniti na

ovaj problem pretraživanjem populacije kandidata za rješenja, od kojih je svako kodirano kao niz od 50 slova, kao što je npr.

IHCCVASASDMIKPVFTVASYLKNWTKAKGPNFEICISGRTPYWDNFPGI,

gdje svako slovo predstavlja jednu od 20 mogućih aminokiselina. Jedan od načina da definiramo dobrotu niza - kandidata jest kao negativnu vrijednost potencijalne energije niza u odnosu na željenu strukturu. Potencijalna energija je mjera fizikalnog otpora koju bi niz pružio kada bi bio prisiljen saviti se u željenu trodimenzionalnu strukturu - što je niža potencijalna energija, to je veća dobrota. Naravno da ne bismo željeli fizički prisiliti svaki niz u populaciji u željenu trodimenzionalnu strukturu i mjeriti njegov otpor - to bi bilo vrlo teško, ako ne i nemoguće. Umjesto toga, za zadani niz i željenu trodimenzionalnu strukturu (uz poznavanje relevantne biofizike), moguće je procijeniti potencijalnu energiju izračunavanjem nekih od sila koje djeluju na svaku aminokiselinu pa se cijeli izračun dobrote može napraviti korištenjem računala.

Ovi primjeri pokazuju dva različita konteksta u kojima su kandidati za rješenja problema kodirani kao apstraktne jedinke - nizovi simbola, s funkcijama dobrote definiranim na rezultirajućem prostoru nizova. Genetski algoritam je metoda za pretraživanje takvih krajolika dobrote u cilju pronalaženja visoko prikladnih nizova.

### 2.5.2. Operatori u genetskim algoritmima

Najjednostavniji oblik genetskog algoritma uključuje tri vrste operatora: selekciju, križanje pomoću jedne točke (engl. *one-point crossover*), i mutaciju. Definirat ćemo ih za kodiranje jedinki pomoću nizova bitova, a za druge načine kodiranja se i ovi operatori definiraju na specifičan način, ali po sličnom principu.

Operator **selekcije** odabire jedinke u populaciji za reprodukciju. Što je jedinka bolja, vjerojatnije je da će biti odabrana za reprodukciju.

Operator **križanja** nasumično bira lokus (poziciju) i razmjenjuje podnizove prije i poslije tog lokusa između dvije jedinke s ciljem stvaranja dvoje potomaka. Na primjer, nizovi 10000100 i 11111111 mogu se križati nakon trećeg lokusa i proizvesti potomke 10011111 i 11100100. Operator križanja ugrubo oponaša biološku rekombinaciju između dva jednokromosomska (haploidna) organizma.

Operator **mutacije** nasumično mijenja neke od bitova u jedinki. Na primjer, niz 00000100 može se mutirati na drugoj poziciji kako bi se dobilo 01000100. Mutacija se može dogoditi na

svakoj poziciji bita u nizu s nekom vjerojatnošću.

## 2.6. JEDNOSTAVNI GENETSKI ALGORITAM

Pretpostavimo da je jasno definiran problem koji treba riješiti i odabran prikaz kandidata za rješenje. Također, pretpostavimo da je definirana funkcija $f$ dobrote kojom ćemo evaluirati jedinke.

Jednostavni genetski algoritam radi na način koji opisujemo u nastavku.

1. Generiramo populaciju koja se sastoji od $n$ nasumičnih jedinki - kandidata za rješenje problema u odabranom prikazu.

2. Izračunamo dobrotu $f(x)$ svake jedinke $x$ u populaciji. Ukoliko postoji jedinka čija dobrota je zadovoljavajuća, odnosno koju možemo proglasiti rješenjem našeg problema, algoritam može stati.

3. Ponavljamo sljedeće korake dok se ne stvori $n$ potomaka:

   (a) Odaberemo par roditeljskih jedinki iz trenutne populacije pri čemu je vjerojatnost odabira jedinke proporcionalna dobroti te jedinke. Postoje razni načini za ovu selekciju, kao što su jednostavna selekcija (rulet selekcija, engl. *roullete wheel selection*), turnirska selekcija (engl. *tournament selection*), eliminacijska selekcija (engl. *steady-state selection*) i druge.

   (b) S vjerojatnošću $p_c$ ("vjerojatnost križanja" ili "stopa križanja", engl. *crossover probability, crossover rate*), križamo odabrani par roditelja na nasumično odabranoj poziciji (odabranoj s uniformnom vjerojatnošću) za formiranje dva potomka. Ako ne dolazi do križanja, formiraju se dva potomka koji su kopije svojih roditelja.

   Ovdje je stopa križanja definirana kao vjerojatnost da će se dva roditelja križati u jednoj točki. Postoje i verzije genetskih algoritama s križanjem u više točaka (engl. *multi-point crossover*).

   (c) Mutiramo dva potomka na svakoj poziciji s vjerojatnošću $p_m$ ("vjerojatnost mutacije" ili "stopa mutacije", engl. *mutation probability, mutation rate*) i smještamo rezultirajuće jedinke u novu populaciju.

   Ako je $n$ neparan, jedan novi član populacije može se nasumično odbaciti.

4. Zamijenimo trenutnu populaciju novom populacijom.

5. Vratimo se na korak 2.

Svaka iteracija ovog procesa naziva se generacija. Genetski algoritam obično iterira od 50 do 500 generacija, iako taj broj varira u ovisnosti o tome što njime želimo postići. Cijeli skup generacija zove se izvođenje algoritma (engl. *run*). Na kraju izvođenja često postoji jedna ili više jedinki koje su visoko prikladne, odnosno dovoljno dobre da budu rješenje zadanog problema. Postoje i problemi kod kojih se traži egzaktno rješenje kod kojeg nas neće zadovoljiti dovoljno dobra aproksimacija; u tom slučaju algoritam staje kada pronađemo upravo to rješenje ili kada je dosegnut unaprijed definirani broj generacija algoritma. Budući da slučajnost igra veliku ulogu u svakom izvođenju algoritma, dva izvođenja s različitim izvorima generiranih nasumičnih brojeva općenito će proizvesti različita ponašanja algoritma. Uz genetski algoritam se često bilježe i statistike, kao što je najbolja vrijednost dobrote dobivena u jednom izvođenju i redni broj generacije u kojoj se nalazi jedinka s tom dobrotom, uprosječene tijekom više različitih izvođenja genetskog algoritma za isti problem.

Jednostavan postupak koji je upravo opisan temelj je za većinu primjena genetskog algoritma. Postoji niz detalja koji se specificiraju u ovisnosti o problemu koji se rješava, kao što su veličina populacije i vjerojatnosti križanja i mutacije te uspjeh algoritma često uvelike ovisi o tim detaljima. Postoje i kompliciranije verzije genetskih algoritama, npr. genetski algoritmi koji rade na prikazima koji nisu nizovi ili genetski algoritmi koji imaju različite vrste križanja i mutacija.

# 3. KONSTRUIRANJE BLOKOVNIH DIZAJNA BEZ PRETPOSTAVLJENOG DJELOVANJA GRUPE AUTOMORFIZAMA KORIŠTENJEM GENETSKOG ALGORITMA

U ovom poglavlju ponovit ćemo istraživanje i rezultate I. Martinjaka i M.-O. Pavčevića iz članka [45] koji se bavi korištenjem genetskog algoritma za konstruiranje blokovnih dizajna bez pretpostavljenog djelovanja grupe automorfizama. U svome članku, predstavili su genetski algoritam za konstrukciju 2-$(v,k,\lambda)$ dizajna pretraživanjem prostora rješenja samo na temelju parametara $v$, $k$ i $\lambda$ tih dizajna bez ikakvih dodatnih uvjeta. Korištenjem ovog algoritma uspjeli su pronaći nove jednostavne dizajne s parametrima 2-$(14,4,6)$.

## 3.1. OPIS ALGORITMA

Osmišljeni algoritam varijacija je algoritma za konstrukciju 2-dizajna kakav je prikazan u [62]. Radi se o algoritmu koji koristi 4-turnirsku selekciju s križanjem jedinki u dvije točke. Cilj algoritma je pronaći 2-$(v,k,\lambda)$ dizajn $\mathscr{D}$ sa $b$ blokova, pri čemu je svaka točka incidentna sa $r$ blokova.

Bez smanjenja općenitosti, možemo označiti skup točaka $\mathscr{P}$ traženog dizajna $\mathscr{D}$ sa $\mathscr{P} = \{1,\ldots,v\}$. Svaku jedinku u populaciji predstavit ćemo kao $b \times k$ cjelobrojnu matricu, pri čemu je svaki redak $k$-člani podskup skupa točaka $\mathscr{P}$. Retci te matrice su geni u jedinkama. Cilj nam je da svaki redak te jedinke predstavlja po jedan od $b$ blokova traženog dizajna. Primijetimo

da će takva jedinka predstavljati 2-$(v,k,\lambda)$ dizajn $\mathscr{D}$ ako i samo ako se svaki dvočlani podskup skupa točaka $\mathscr{P}$ pojavljuje u točno $\lambda$ redaka (gena) te jedinke. Nikakve dodatne uvjete ne treba provjeravati. Veličinu populacije, odnosno, broj jedinki u populaciji označit ćemo sa POP.

Kako bismo "izmjerili" koliko određenoj jedinki nedostaje da bi bila traženi 2-$(v,k,\lambda)$ dizajn $\mathscr{D}$, izbrojat ćemo koliko se puta svaki dvočlani podskup skupa točaka $\mathscr{P}$ pojavljuje u retcima jedinke, pri čemu ćemo stati kada se dosegne $\lambda$. Funkciju dobrote definirat ćemo kao zbroj svih takvih pojavljivanja. U slučaju da promatrana jedinka predstavlja dizajn, izbrojat ćemo ukupno $b \cdot \binom{k}{2}$ parova, što je jednako $\binom{v}{2} \cdot \lambda$. Dakle, maksimalna (optimalna) vrijednost funkcije dobrote je

$$\binom{v}{2}\lambda = \frac{v(v-1)\lambda}{2}.$$

U slučaju da dobrota neke jedinke dostiže ovu vrijednost, ta jedinka predstavlja konstruirani traženi dizajn.

U svakoj iteraciji algoritma, populacija se dijeli u grupe od četiri jedinke (4-turnirska selekcija), pri čemu se iz svake grupe izdvajaju dvije najbolje jedinke, odnosno one koje u toj grupi imaju najveće vrijednosti funkcije dobrote. Kako bi ta raspodjela bila slučajna, prethodno se populacija jedinki na slučajan način permutira. Te dvije jedinke preuzimaju ulogu roditelja čiji se geni križaju kao što je prikazano na primjeru dvoravnine 2-(7,4,2) na slici 3.1.

$$\left[\begin{array}{cccc} 6 & 1 & 3 & 5 \\ 3 & 7 & 1 & 4 \\ \hline 7 & 2 & 5 & 1 \\ 3 & 2 & 6 & 7 \\ 4 & 1 & 6 & 7 \\ 1 & 3 & 2 & 6 \\ \hline 1 & 3 & 7 & 4 \end{array}\right] \qquad \left[\begin{array}{cccc} 5 & 7 & 2 & 3 \\ 7 & 1 & 4 & 5 \\ \hline 1 & 3 & 4 & 5 \\ 6 & 5 & 4 & 1 \\ 2 & 1 & 5 & 7 \\ 3 & 1 & 7 & 4 \\ \hline 7 & 5 & 2 & 3 \end{array}\right]$$

roditelj A roditelj B

$$\left[\begin{array}{cccc} 6 & 1 & 3 & 5 \\ 3 & 7 & 1 & 4 \\ \hline 1 & 3 & 4 & 5 \\ 6 & 5 & 4 & 1 \\ 2 & 1 & 5 & 7 \\ 3 & 1 & 7 & 4 \\ \hline 1 & 3 & 7 & 4 \end{array}\right] \qquad \left[\begin{array}{cccc} 5 & 7 & 2 & 3 \\ 7 & 1 & 4 & 5 \\ \hline 7 & 2 & 5 & 1 \\ 3 & 2 & 6 & 7 \\ 4 & 1 & 6 & 7 \\ 1 & 3 & 2 & 6 \\ \hline 7 & 5 & 2 & 3 \end{array}\right]$$

dijete A dijete B

Slika 3.1: Križanje u dvije točke.

Nadalje, postoje dva različita operatora mutacije koja se mogu primijeniti na djeci dobivenom križanjem. Kod jednog djeteta može se zamijeniti cijeli gen, kao slučajni $k$-član podskup skupa točaka $\mathscr{P}$, a kod drugog djeteta može se zamijeniti samo jedan element slučajno odabranog gena i to slučajno odabranom točkom iz $\mathscr{P}$ (vodeći računa o tome da se novi gen i dalje sastoji od $k$ različitih elemenata). Oba operatora mutacije primjenjuju se s određenom vjerojatnošću $p_m$, koja se, eksperimentalno utvrđeno, postavlja na jednaku vrijednost za oba operatora.

Pseudokod razvijenog algoritma, za kojeg je napisan pripadni računalni program za GAP [60], prikazan je u nastavku.

**Algoritam GA**

1. generiraj početnu populaciju
2. $c \leftarrow 0$
3. **while** ($f_{\text{best}} < f_{\text{design}}$ and $c < c_{\max}$)
4. permutiraj populaciju jedinki
5. $i \leftarrow 0$
6. **while** ($i \leq POP$)
7. odaberi dvije najbolje jedinke (roditelje) između $i$-te i $(i+3)$-će
8. križaj roditelje i stvori djecu
9. mutiraj djecu
10. zamijeni dvije najgore jedinke između $i$-te i $(i+3)$-će mutiranom djecom
11. $i \leftarrow i+4$
12. **end while**
13. evaluiraj dobrotu svake pojedine jedinke
14. $c \leftarrow c+1$
15. **end while**
16. **end**

## 3.2. OPTIMIZACIJA PARAMETARA

Da bi se algoritam učinio što učinkovitijim, potrebna je optimizacija ulaznih parametara na osnovu nekoliko eksperimentalnih izvođenja. Kao što su to učinili autori u [45], za različite trojke parametara $(v,k,\lambda)$, pustili smo da se algoritam izvodi dok se ne pronađe rješenje ili dok se ne dosegne maksimalan broj iteracija $c < c_{\max} = 100\,000$ i to 100 puta. Autori su ovdje eksperimentalno utvrdili da permutacija populacije jedinki prije 4-turnirske selekcije daje bolje rezultate, što smo mi od početka uklopili u našu verziju algoritma, s ciljem da jedinke koje su susjedne u jednoj generaciji ne bi ostale susjedne u drugoj. Također, ponovljen je i eksperiment s različitim veličinama populacije POP, gdje smo isprobavali veličine od 20 do 400 jedinki. Kao što je i očekivano, u slučajevima kada je bilo moguće doći do rješenja, kod populacija s manje jedinki izvođenje algoritma je brže po iteraciji, no potrebno je više iteracija kako bi se došlo do rješenja. Kod većih populacija je suprotno, iteracije se sporije izvode, no treba ih manje na putu do rješenja. Zaključak je da populacije od 40, 60, 80 ili 100 jedinki daju najbolje rezultate uzevši u obzir brzinu izvođenja i potreban broj iteracija pa je odabrana vrijednost $\text{POP} = 60$ za daljnje eksperimente.

Uz navedenu veličinu populacije, nadalje smo pokušali optimizirati i vjerojatnost mutacije $p_m$ s kojom bi bilo potrebno što manje iteracija koje bi vodile do prvog rješenja. Ovdje $p_m$ znači vjerojatnost da će dijete biti mutirano nakon križanja. Postupnim povećavanjem vjerojatnosti mutacije dobili smo, kao i autori, poboljšanje u vidu prosječnog broja iteracija koje vode do dizajna (#iteracija) i broja dobivenih rješenja unutar 100 izvođenja (#rješenja). U tablici 3.1 prikazujemo dio eksperimentalno dobivenih rezultata za vjerojatnosti $p_m$ od 95, 98 i 100 posto za ulazne parametre (6,3,2), (7,3,1), (7,2,4), (8,4,3) i (11,5,2), (19,3,1) i (21,3,1). Naš ponovljeni eksperiment daje slične rezultate kao i kod autora. Zaključak je da maksimalna vjerojatnost mutacije $p_m$ od 100% daje najbolje rezultate u broju dobivenih rješenja.

| $(v,k,\lambda)$ | $p_m$ | 95 | 98 | 100 |
|---|---|---|---|---|
| (6,3,2) | #iteracija | 68 | 61 | 57 |
| | #rješenja | 76 | 81 | 84 |
| (7,3,1) | #iteracija | 102 | 86 | 52 |
| | #rješenja | 71 | 87 | 91 |
| (7,2,4) | #iteracija | 98 | 79 | 74 |
| | #rješenja | 46 | 52 | 70 |
| (8,4,3) | #iteracija | 687 | 601 | 448 |
| | #rješenja | 40 | 45 | 67 |
| (11,5,2) | #iteracija | 4 220 | 4 003 | 3 804 |
| | #rješenja | 38 | 50 | 62 |
| (19,3,1) | #iteracija | 20 191 | 19 207 | 19 184 |
| | #rješenja | 6 | 9 | 13 |
| (21,3,1) | #iteracija | 63 488 | 60 074 | 61 879 |
| | #rješenja | 11 | 24 | 72 |

Tablica 3.1: Optimizacija vjerojatnosti mutacije.

Uz navedeni eksperiment za optimizaciju parametara mutacije, proveli smo, uz vjerojatnost mutacije $p_m = 100\%$ i eksperiment koji uspoređuje rezultate na temelju dvaju različitih početnih populacija za neke vrijednosti ulaznih parametara $(v,k,\lambda)$. Za svaku trojku parametara, prva populacija izgrađena je od jedinki sa slučajno generiranim genima, a kod druge ("poboljšane") populacije postavljen je dodatan uvjet, a to je da se skup gena koji čine jedinku mora sastojati od $k$-torki koje su sve međusobno različite. Za oba pristupa i za odabrane trojke ulaznih parametara dizajna (6,3,2), (7,3,1), (7,2,4), (8,4,3) i (11,5,2), (19,3,1) i (21,3,1) izmjereni su: najmanji broj iteracija potrebnih do rješenja (#min1, #min2) i prosječan broj iteracija do rješenja (#avg1,#avg2). Naš ponovljeni eksperiment pokazao je, kao i kod autora, da su rezultati s "poboljšanom" populacijom bolji samo u slučaju malih parametara, tako da smo nadalje početnu populaciju gradili sa slučajno generiranim genima. Rezultate prikazujemo u tablici 3.2.

| $(v,k,\lambda)$ | $\#min1$ | $\#avg1$ | $\#min2$ | $\#avg2$ |
|---|---|---|---|---|
| (6,3,2) | 12 | 54 | 4 | 28 |
| (7,3,1) | 17 | 50 | 8 | 41 |
| (7,2,4) | 22 | 73 | 6 | 201 |
| (8,4,3) | 70 | 465 | 147 | 1 341 |
| (11,5,2) | 226 | 3 971 | 392 | 5 111 |
| (19,3,1) | 2 439 | 20 325 | 3 595 | 23 588 |
| (21,3,1) | 9 492 | 60 324 | 11 593 | 72 505 |

Tablica 3.2: Usporedba dvaju različitih početnih populacija.

Naposlijetku, proveli smo test kojim smo pokušali konstruirati što je više moguće dizajna s fiksnim skupom parametara $(v,k,\lambda)$ na način da smo dozvolili izvođenje algoritma i nakon što pronađe prvo rješenje, sve dok se ne postigne maksimalan broj iteracija $c_{\max}$ koji je, kao i prije, postavljen na 100 000. Ukoliko se pronađe rješenje, ono se sprema u listu rješenja, a u populaciji zamjenjuje novom slučajnom jedinkom. Preostale jedinke u populaciji ostaju onakvima kakve su i bile u tom trenutku. Ova metoda testirana je na parametrima (8,4,12) i (11,5,2), međutim pokazalo se da je za svaku od ovih trojki parametara svako sljedeće rješenje izomorfno (ili jednako) prvom dobivenom rješenju. Zaključak je da jedna početna populacija ima tendenciju voditi uvijek k istom rješenju.

Još jedna od karakteristika ovog genetskog algoritma, kao uostalom i većine genetskih algoritama koji pokušavaju riješiti razne probleme iz mnogobrojnih područja znanosti, jest da algoritam kroz vrlo mali broj iteracija dolazi "blizu" rješenju (odnosno, u populaciji se pojavljuju jedinke s funkcijom dobrote bliskoj vrijednosti funkcije dobrote dizajna), međutim, nakon toga, u sljedećim iteracijama događaju se samo mala poboljšanja.

## 3.3. REZULTATI

Uz optimizirane ulazne parametara kao što je prethodno opisano, to jest, uz veličinu populacije POP $= 60$, vjerojatnost mutacije $p_m = 100\%$ i maksimalan broj iteracija $c_{\max} = 100\,000$, konstruirali smo blokovne dizajne s različitim parametrima $(v,k,\lambda)$. U tablici 3.3 za svaku trojku testiranih parametara prikazali smo minimalan broj iteracija potreban da bi se doseglo prvo rješenje (#min), prosječan broj iteracija do prvog rješenja (#avg) te ukupan broj dobivenih rješenja (#rješenja) dobivenih unutar 100 izvođenja algoritma. Pri tome, dobiveni broj rješenja unutar 100 izvođenja može biti i veći od 100 budući da je čest slučaj da se u istoj iteraciji algoritma pojave dvije ili više jedinki s ciljanom funkcijom dobrote koja karakterizira dizajn.

| $(v,k,\lambda)$ | #min | #avg | #rješenja |
|---|---|---|---|
| (8,4,3) | 82 | 468 | 88 |
| (8,4,15) | 231 | 997 | 91 |
| (10,3,2) | 145 | 720 | 98 |
| (10,5,4) | 1 678 | 41 107 | 83 |
| (12,4,3) | 5 978 | 75 778 | 44 |
| (21,3,2) | 5 177 | 58 687 | 71 |
| (7,3,1) | 19 | 51 | 102 |
| (13,4,1) | 201 | 2 060 | 101 |
| (21,5,1) | 44 441 | 99 280 | 3 |
| (7,2,4) | 25 | 68 | 107 |
| (11,5,2) | 212 | 4 005 | 100 |
| (9,3,1) | 63 | 201 | 105 |
| (13,3,1) | 224 | 4 123 | 98 |
| (15,3,1) | 427 | 13 434 | 93 |
| (19,3,1) | 5 873 | 54 223 | 80 |
| (21,3,1) | 6 001 | 66 236 | 59 |
| (25,3,1) | 16 710 | 97 201 | 11 |
| (27,3,1) | 64 278 | 99 405 | 1 |

Tablica 3.3: Primjeri konstruiranih 2-dizajna.

Iz tablice 3.3 možemo naslutiti da minimalan i prosječan broj iteracija do prvog rješenja ovise, kako o broju točaka dizajna $v$, tako i o ostalim parametrima. Dvoravnine, to jest, 2-$(v,k,2)$ dizajni, čine se "najrjeđima" među 2-$(v,k,\lambda)$ dizajnima, odnosno imaju najmanju gustoću rješenja u cjelokupnom prostoru rješenja. Na primjer, iako je broj točaka projektivne ravnine s parametrima 2-$(21,5,1)$ veći nego broj točaka dvoravnine s parametrima 2-$(16,6,2)$, unutar 100 000 iteracija algoritma izvedenog 100 puta, nismo uspjeli doseći tu dvoravninu, iako smo, kao što je navedeno u tablici, dobili tri rješenja za parametre (21,5,1).

Također, ispitali smo koje su grupe automorfizama dizajna dobivenih u tablici 3.3. Zamijetili smo da je većina tih grupa automorfizama trivijalna, pogotovo kod dizajna s većim parametrima.

Koristeći ovu metodu, autori su pronašli dvadeset do tada nepoznatih neizomorfnih dizajna s parametrima 2-(14,4,6). Svi ti dizajni imaju trivijalne grupe automorfizama. Budući da ti dizajni imaju 91 blok, teško je zamisliti da bi iscrpna potraga za takvim dizajnima mogla uroditi plodom, budući da je prostor pretraživanja jako velik, iako se parametri dizajna čine relativno malima.

# 4. KONSTRUIRANJE BLOKOVNIH DIZAJNA S PRETPOSTAVLJENOM GRUPOM AUTOMORFIZAMA KORIŠTENJEM GENETSKOG ALGORITMA

U ovom poglavlju opisat ćemo metodu konstrukcije blokovnih dizajna koja kombinira genetski algoritam i metodu za konstruiranje dizajna s pretpostavljenom grupom automorfizama korištenjem taktičke dekompozicije (tj. orbitnih matrica). Ovu metodu primijenit ćemo za konstrukciju novih Steinerovih sustava s parametrima $S(2,5,45)$ i novih simetričnih dizajna s parametrima (71,15,3).

## 4.1. UVOD

Konstrukcija blokovnih dizajna s određenim dopustivim parametrima se često pokušava izvesti za jedan određeni skup parametara uz pretpostavljanje nekih dodatnih ograničenja na strukturu dizajna kako bi pretraživanje bilo računalno izvedivo. Prirodno ograničenje je pretpostavka da određena grupa automorfizama djeluje na dizajn. Jedna od metoda konstruiranja blokovnih dizajna s pretpostavljenom grupom automorfizama je metoda koja koristi orbitne matrice. Ova metoda je opisana u [35] i uspješno su je koristili Z. Janko i drugi za konstrukciju blokovnih dizajna, na primjer u [18], [22], [36], [37], [38].

U cilju da se omogući konstrukcija blokovnih dizajna koja ne bi bila izvediva izvođenjem iscrpnog pretraživanja (engl. *exhaustive search*), predlažemo kombinaciju metode koja koristi orbitne matrice i genetskog algoritma. Koristeći ovaj pristup konstruiramo 35 novih Steinerovih sustava s parametrima $S(2,5,45)$ i 22 nova simetrična dizajna s parametrima (71,15,3). Heuris-

tičke metode korištene su i ranije za konstruiranje blokovnih dizajna i drugih kombinatornih struktura, na primjer u [31], [32], [39], [45], [46]. Međutim, koliko nam je poznato, ovo je prva primjena genetskog algoritma kao heurističke metode za konstrukciju blokovnih dizajna s pretpostavljenom grupom automorfizma, korištenjem orbitnih matrica.

## 4.2. KONSTRUKCIJA

Orbitne matrice se često koriste u konstrukciji blokovnih dizajna s pretpostavljenom grupom automorfizma. Konstrukcija dizajna koji dopuštaju djelovanje pretpostavljene grupe automorfizama sastoji se od sljedeća dva osnovna koraka (vidi [35]):

1. konstrukcije orbitnih matrica za pretpostavljenu grupu automorfizma,
2. konstrukcije blokovnih dizajna za orbitne matrice dobivene na ovaj način.

Drugi se korak često naziva *indeksiranjem* orbitnih matrica. Cilj drugog koraka konstrukcije je konstruirati matrice incidencije blokovnih dizajna koje odgovaraju orbitnim matricama dobivenim u prvom koraku. Indeksiranje orbitnih matrica obično se provodi metodom iscrpnog pretraživanja. Međutim, ponekad iscrpno pretraživanje nije izvedivo jer postoji previše struktura koje bi trebalo provjeriti, što je zahtjevno zbog ograničenih kapaciteta računalne memorije, a isto tako i samog vremena izvođenja pretraživanja. Kako bismo prevladali ovu prepreku, predlažemo korištenje genetskog algoritma u ovom koraku konstrukcije. Koliko nam je poznato, ovo je prvi put da se koristi heuristička metoda, odnosno genetski algoritam u izvođenju drugog koraka konstrukcije.

U procesu indeksiranja orbitnih matrica neke od orbitnih matrica uopće ne mogu producirati dizajne sa željenim parametrima, dok se iz drugih orbitnih matrica može dobiti više dizajna koji su međusobno neizomorfni. U ovom radu nećemo se baviti prvim korakom konstrukcije, odnosno konstrukcijom samih orbitnih matrica, već ćemo iz orbitnih matrica već poznatih dizajna pokušati konstruirati nove dizajne, neizomorfne s do tada poznatima.

### 4.2.1. Opis problema

Neka su $(v,k,\lambda)$ dopustivi parametri 2-dizajna. Neka je $G$ konačna grupa i neka je $(\nu_1,\nu_2,\ldots,\nu_m)$ distribucija duljina orbita točaka, a $(\beta_1,\beta_2,\ldots,\beta_n)$ distribucija duljina orbita blokova za djelovanje grupe $G$ na 2-$(v,k,\lambda)$ dizajn. Nadalje, neka je $M$ orbitna matrica za parametre $(v,k,\lambda)$ i distribuciju duljina orbita točaka $(\nu_1,\nu_2,\ldots,\nu_m)$ te distribuciju duljina orbita blokova $(\beta_1,\beta_2,\ldots,\beta_n)$. Naš cilj je konstruirati matricu incidencije 2-$(v,k,\lambda)$ dizajna $\mathscr{D}$ koji dopušta djelovanje grupe automorfizama izomorfne grupi $G$ uz distribuciju duljina orbita točaka $(\nu_1,\nu_2,\ldots,\nu_m)$ i distribuciju duljina orbita blokova $(\beta_1,\beta_2,\ldots,\beta_n)$, tako da to djelovanje producira orbitnu matricu $M$. U ovom radu pretpostavljat ćemo djelovanja cikličkih grupa prostog reda, odnosno grupa

izomorfnih sa $\mathbb{Z}_p, p \in \mathbb{P}$. Pri indeksiranju orbitne matrice $M$ koristit ćemo genetski algoritam, što znači da naše pretraživanje neće biti iscrpno. Drugim riječima, postoji mogućnost da ovim postupkom nećemo konstruirati sve neizomorfne dizajne koje je moguće konstruirati iz dane orbitne matrice $M$. Za početak ćemo testirati algoritam na primjerima dizajna s poznatim parametrima. To znači da ćemo nizom eksperimenata za različite parametre $(v,k,\lambda)$ poznatih 2-dizajna utvrditi optimalne vrijednosti parametara algoritma, kao što su veličina populacije, vjerojatnost za mutaciju i crossover, broj mutiranih i križanih gena itd.

### 4.2.2. Prikaz kandidata za rješenje

Prvi osnovni zadatak pri primjeni genetskog algoritma na specifični problem jest odrediti način prikaza kandidata za rješenje problema.

Za kandidate za rješenja u literaturi se koriste razni nazivi, od kojih su najučestaliji "kromosomi" ili, kako ćemo ih mi u daljnjem tekstu zvati "jedinke" (engl. *individuals*) koje čine populaciju. Kao što smo ranije naveli u poglavlju 2, najjednostavniji način prikaza jedinke jest u obliku niza bitova 0 ili 1, međutim, taj prikaz nije pogodan za rješavanje svih problema pa tako ni ovog našeg. U poglavlju 3, svaku jedinku u populaciji predstavili smo kao $b \times k$ cjelobrojnu matricu, pri čemu je svaki redak $k$-člani podskup skupa točaka $\mathscr{P}$. Retci te matrice su geni u jedinkama. Cilj nam bio je da svaki redak te jedinke predstavlja po jedan od $b$ blokova traženog dizajna.

U populaciji koju ćemo mi stvoriti, jedinke će biti matrice incidencije 1-dizajna koji dopuštaju djelovanje grupe $G$ s obzirom na orbitnu matricu $M$. Drugim riječima, jedinke su $(0,1)$-matrice dimenzija $v \times b$ kojima je suma elemenata u svakom pojedinom retku $r$, suma elemenata u svakom pojedinom stupcu $k$ i koje dopuštaju djelovanje grupe $G$ uz distribuciju duljina orbita točaka $(\nu_1, \nu_2, \ldots, \nu_m)$ i distribuciju duljina orbita blokova $(\beta_1, \beta_2, \ldots, \beta_n)$ iz kojih se može dobiti orbitna matica $M$. Cilj nam je iz početne populacije koja se sastoji od određenog broja takvih, na slučajan način generiranih jedinki, primjenom genetskog algoritma konstruirati jedinku koja predstavlja matricu incidencije 2-$(v,k,\lambda)$ dizajna.

Gen u ovakvoj jedinki čine retci matrice incidencije koji odgovaraju jednoj orbiti točaka. Bit jednog gena je podmatrica koja predstavlja presjek tog gena sa stupcima koji odgovaraju jednoj orbiti blokova. Bitovi, dakle, odgovaraju elementima orbitne matrice. Svaki bit određen je svojim prvim retkom, a ostali retci u bitu na jedinstven su način određeni djelovanjem grupe $G$ na prvi redak. Valja napomenuti da izraz "bit" ovdje koristimo jer je to najmanji mogući

podatak u jedinki koji se može izmijeniti. U našoj terminologiji "bit" može imati i više od dva stanja. Za neke elemente orbitne matrice bitovi su jednoznačno određeni (zvat ćemo ih fiksni bitovi), dok iz nekih drugih elemenata orbitne matrice postoji više mogućnosti za konstrukciju odgovarajućeg bita (zvat ćemo ih nefiksni bitovi). Isto tako, možemo govoriti i o fiksnim i nefiksnim genima.

**Primjer 4.2.1.** Za jedan 2-(11,5,2) dizajn uz pretpostavljeno djelovanje cikličke grupe $\mathbb{Z}_5$ dobije se orbitna matrica

| | 1 | 5 | 5 |
|---|---|---|---|
| 1 | 0 | 5 | 0 |
| 5 | 1 | 2 | 2 |
| 5 | 0 | 2 | 3 |

.

Budući da se radi o simetričnom dizajnu, distribucije duljina orbita blokova i točaka su jednake: (1,5,5). Grupa djeluje tako da postoje tri orbite točaka i tri orbite blokova.

Iz takve orbitne matrice, uz poznate distribucije duljne orbita blokova i točaka i pretpostavljeno djelovanje cikličke grupe $\mathbb{Z}_5$, na slučajan način generiramo jedinku. Jedna takva slučajno generirana jedinka je:

$$\left[\begin{array}{c|ccccc|ccccc}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
\hline
0 & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & 0 & 1 & 1 & 0 & 1 \\
0 & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & 1 & 0 & 1 & 1 & 0 \\
0 & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & 0 & 1 & 0 & 1 & 1 \\
0 & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & 1 & 0 & 1 & 0 & 1 \\
0 & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & 1 & 1 & 0 & 1 & 0
\end{array}\right].$$

U ovom prikazu podebljano je označen jedan gen, a ukošeno je označen jedan bit.

### 4.2.3. Odabir funkcije dobrote

Drugi osnovni zadatak pri primjeni genetskog algoritma na specifični problem jest definirati funkciju dobrote (engl. *fitness function*) koja će dobro opisivati pogodnost kandidata.

Testirali smo dvije različite funkcije dobrote koje smo smatrali pogodnima za ovaj problem. Na temelju eksperimenta, to jest usporedbe rada algoritma korištenjem svake od tih dviju funkcija zasebno, uz nepromijenjene ostale parametre algoritma, zaključili smo koju funkciju ćemo koristiti. Usporedbu rada ćemo prikazati kasnije, a u nastavku opisujemo funkcije dobrote koje smo testirali.

Prva funkcija dobrote definirana je na sličan način kao i u poglavlju 3, odnosno u [45], uz prilagodbu računanja dobrote za svaku pojedinu jedinku budući da se naš prikaz jedinke razlikuje od onog u [45]. Za svake dvije različite točke $P_i$ i $P_j$, definiramo da je $x_{i,j}$ broj zajedničkih pojavljivanja od $P_i$ i $P_j$ u zajedničkom bloku, to jest, skalarni produkt odgovarajućih redova matrice incidencije 1-dizajna kojom prikazujemo jedinku. Funkciju dobrote definiramo kao

$$\sum_{P_i,P_j\in\mathscr{P}} \min\{x_{i,j},\lambda\},$$

pri čemu je $\mathscr{P}$ skup točaka željenog dizajna. U 2-$(v,k,\lambda)$ dizajnu, svaki par točaka (a takvih parova ima $\binom{v}{2}$) pojavljuje se u točno $\lambda$ zajedničkih blokova. Jedinka će biti matrica incidencije 2-$(v,k,\lambda)$ dizajna ako i samo ako je vrijednost njezine funkcije dobrote

$$\binom{v}{2}\lambda.$$

Uz ovako definiranu funkciju dobrote, naš problem pronalaženja optimalnog rješenja, tj. dizajna, je maksimizacijski, odnosno prosječne vrijednosti dobrota jedinki tijekom generacija genetskog algoritma trebale bi rasti prema optimalnoj, odnosno maksimalnoj vrijednosti funkcije dobrote koja iznosi $\binom{v}{2}\lambda$.

Drugu funkciju dobrote ($x_{i,j}$ definiran kao i prije) definiramo kao

$$\sum_{P_i,P_j\in\mathscr{P}} (x_{i,j}-\lambda)^2.$$

Uz ovako definiranu funkciju dobrote, naš problem pronalaženja optimalnog rješenja, tj. dizajna, je minimizacijski, odnosno prosječne vrijednosti dobrota jedinki tijekom generacija genetskog algoritma trebale bi padati prema optimalnoj, odnosno minimalnoj vrijednosti funkcije dobrote koja iznosi 0.

Nakon eksperimentalne usporedbe ovih dviju funkcija, odlučili smo koristiti prvu opisanu funkciju dobrote.

### 4.2.4. Križanje

Križanje u genetskom algoritmu općenito znači zamjenu odgovarajućih gena dviju jedinki koje se križaju (nadalje ćemo takve jedinke zvati "roditelji"). U našem slučaju, to znači da ćemo zamijeniti retke na nekim pozicijama prvog roditelja s retcima na istim pozicijama u drugom roditelju i obratno. Na taj način dobivamo dvije nove jedinke, potomke roditelja (nadalje ćemo takve jednike nazivati "djeca"). Križanje možemo vršiti u jednoj, dvije točke ili u više točaka. Eksperimentalno smo utvrđivali vjerojatnost križanja $p_c$ te optimalan broj gena koje roditelji izmjenjuju.

**Primjer 4.2.2.** Za parametre 2-$(11,5,2)$ i pretpostavljeno djelovanje grupe $\mathbb{Z}_5$ s orbitnom matricom kao u primjeru 4.2.1, križamo jedan gen prvog roditelja s genom na odgovarajućoj poziciji drugog roditelja. Time dobivamo dvoje djece koji su mješavina genetskog materijala oba roditelja.

$$\left[\begin{array}{c|cccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \hline
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}\right]$$

roditelj A

$$\left[\begin{array}{c|cccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
\mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} \\
\mathit{1} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} \\
\mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} \\
\mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} \\
\mathit{1} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} \\ \hline
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}\right]$$

roditelj B

| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| *1* | *0* | *1* | *0* | *0* | *1* | *0* | *1* | *0* | *1* | *0* |
| *1* | *1* | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *0* | *1* |
| *1* | *0* | *1* | *0* | *1* | *0* | *1* | *0* | *0* | *1* | *0* |
| *1* | *0* | *0* | *1* | *0* | *1* | *0* | *1* | *0* | *0* | *1* |
| *1* | *1* | *0* | *0* | *1* | *0* | *1* | *0* | *1* | *0* | *0* |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

dijete A

| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** |
| **1** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

dijete B

### 4.2.5. Mutacija

Mutaciju provodimo na način da se jedan ili više bitova unutar gena jedinke zamijeni s novim, slučajno generiranim bitovima. To zapravo znači da na slučajan način permutiramo prvi redak bita, a ostali retci u bitu određeni su djelovanjem pretpostavljene grupe automorfizama. Također, kao i kod križanja, eksperimentalno smo utvrđivali vjerojatnost mutacije $p_m$, odnosno optimalan broj bitova nad kojima će se vršiti mutacija kod djece.

**Primjer 4.2.3.** Za parametre 2-$(11,5,2)$ i pretpostavljeno djelovanje grupe $\mathbb{Z}_5$ te orbitnu matricu iz primjera 4.2.1, mutiramo jedan bit u genu jedinke, dok preostali bitovi ostaju isti. Na primjer,

$$\left[\begin{array}{c|ccccc|ccccc} 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{array}\right] \longrightarrow \left[\begin{array}{c|ccccc|ccccc} 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{array}\right]$$

Primijetimo da je mutacija bita, odnosno podmatrice unutar jedinke koja je presjek redaka koji odgovaraju trećoj orbiti točaka sa stupcima koji odgovaraju trećoj orbiti blokova, zapravo određena mutacijom prvog retka te podmatrice. U orbitnoj matrici iz primjera 4.2.1 element na odgovarajućoj poziciji je 3, odnosno svaki blok iz treće orbite blokova incidentan je s 3 točke iz treće orbite točaka. To svojstvo mora biti očuvano i mutacijom. Dakle, mutacija zapravo znači preraspodjelu pozicija nula i jedinica u prvom retku te podmatrice, dok njihov broj ostaje očuvan. Ostali retci podmatrice na jedinstven su način određeni djelovanjem grupe $\mathbb{Z}_5$, odnosno oni su uzastopni ciklički pomaci tog prvog retka.

### 4.2.6. Selekcija

Genetski algoritam koji smo razvili koristi 4-turnirsku selekciju (engl. *4-tournament selection*). Ova selekcija eksperimentalno se pokazala boljom od ostalih dviju metoda selekcije koje smo testirali, a to su jednostavna i eliminacijska selekcija. Kod 4-turnirske selekcije, veličina populacije mora biti višekratnik broja 4. Populacija se dijeli u podskupove od četiri jedinke, tj. 4-turnire, a na temelju funkcije dobrote dvije najbolje rangirane jedinke izdvajaju se kao roditelji na koje će djelovati operator križanja. Križanjem se dobiva dvoje djece na koje djeluje operator mutacije te tako stvorena djeca zamjenjuju dvije najlošije rangirane jedinke u 4-turniru. Prije stvaranja nove generacije, takvu populaciju jedinki permutiramo na slučajan način kako pri sljedećoj selekciji ne bismo odabirali opet iste roditelje u svakom 4-turniru.

Kod jednostavne selekcije (engl. *roulette wheel selection*) cilj je odabrati roditelje s vjerojatnosti za selekciju koja je proporcionalna njihovoj dobroti. Svim jedinkama $v_i$, $i = 1,2,\ldots,\text{POP}$, u populaciji izračunaju se vrijednosti dobrote $f(v_i)$ te se izračuna i ukupna dobrota populacije $D = \sum_{i=1}^{\text{POP}} f(v_i)$, kao zbroj pojedinačnih dobrota. Za svaku jedinku izračunamo njezinu kumulativnu dobrotu $q_k = \sum_{i=1}^{k} f(v_i)$, $k = 1,2,\ldots,\text{POP}$. Vjerojatnost selekcije $p_k$ jedinke $v_k$ definiramo kao $p_k = \frac{f(v_k)}{D}$ i ta je vjerojatnost proporcionalna dobroti jedinke $v_k$. Očito vrijedi $\sum_{i=1}^{\text{POP}} p_k = 1$. Potom generiramo slučajni realni broj $r \in [0,D]$ i potražimo $i$-tu jedinku za koju vrijedi $r \in [q_{i-1}, q_i]$. Nad tako selektiranim jedinkama vršimo križanje, dobivamo djecu i mutiramo ih.

Eliminacijska selekcija (engl. *steady-state selection*), za razliku od jednostavne selekcije, ne bira "dobre" jedinke, već "loše" koje treba eliminirati i reprodukcijom zamijeniti novima. Dakle, "loše" jedinke odumiru, a njih nadomještaju djeca nastala reprodukcijom roditelja, tj. preživjelih jedinki. Algoritam selekcije "loših" jedinki sličan je jednostavnoj selekciji, ali vjerojatnost selekcije je ovdje obrnuto proporcionalna dobroti jedinke. Umjesto funkcije dobrote treba definirati funkciju kazne $f'$:

$$f'(v_i) = \max\{f(v_i) \mid i = 1,2,\ldots,\text{POP}\} - f(v_i),\ i = 1,2,\ldots,\text{POP}.$$

Tada je vjerojatnost selekcije jedinke proporcionalna njezinoj kazni. Primijetimo da u ovom slučaju najbolje rangirana jedinka u trenutnoj populaciji ima kaznu 0. Analogno jednostavnoj selekciji, odabire se najlošija jedinka koja se eliminira iz populacije. Postupak se ponavlja unaprijed definirani broj puta, a križanjem preživjelih jedinki i mutacijom dobivene djece nadopunjavamo populaciju.

### 4.2.7. Pseudokod razvijenog algoritma

Navedimo pseudokod razvijenog algoritma koji se pokazao korisnim za konstrukciju blokovnih dizajna iz orbitnih matrica s pretpostavljenom određenom grupom automorfizama. Pripadni računalni program napisan je u programu GAP [60].

**Funkcija GA**

```
1.  NrOfCompleteResets ← 0
2.  NrOfPartialResets ← 0
3.  while NrOfCompleteResets < MaxNrOfCompleteResets
4.  StartingPopulation ← {}
5.     while NrOfPartialResets < MaxNrOfPartialResets
6.        Population ← GeneratePopulation(POP,StartingPopulation)
7.        f_best ← BestFitness(Population)
8.        f_bestNrOfRepeats ← 1
9.        while f_best < FitnessForDesign and NrOfGenerations < MaxNrOfGenerations
10.          Population ← Shuffle(Population)
11.          WorkOnPopulation(Population)
12.          povećaj NrOfGenerations
13.          f_best ← BestFitness(Population)
14.          if vrijednost f_best nije povećana then
15.             povećaj f_bestNrOfRepeats
16.          end if
17.          if f_bestNrOfRepeats  = f_bestNrOfRepeatsMax then
18.             StartingPopulation ← unaprijed određeni postotak StartingPercentage najbolje
                                     rangiranih jedinki
19.             provedi djelomično resetiranje populacije
20.          end if
21.       end while
22.       povećaj NrOfPartialResets
23.    end while
24.    povećaj NrOfCompleteResets
25. end while
26. if f_best=FitnessForDesign then
27.    izdvoji dobivena rješenja, tj. dizajne
28. end if
```

**Funkcija WorkOnPopulation**

1. $i \leftarrow 1$
2. **while** $i <$ POP
3. WorkOnFour($i$,Population)
4. $i \leftarrow i+4$
5. **end while**

**Funkcija WorkOnFour**

1. Parents $\leftarrow$ dvije najbolje rangirane jedinke između $i$-te i $(i+3)$-će jedinke
2. **while** Parent1 $=$ Parent2
3. MutatedParent2 $\leftarrow$ Mutation(Parent2)
4. **if** MutatedParent2 nije u Population
5. Parent2 $\leftarrow$ MutatedParent2
6. **end if**
7. **end while**
8. Children $\leftarrow$ Crossover(Parents)
9. **repeat**
10. MutatedChild1 $\leftarrow$ Mutation(Child1)
11. **until** nije MutatedChild1 u Population
12. Child1 $\leftarrow$ MutatedChild1
13. **repeat**
14. MutatedChild2 $\leftarrow$ Mutation(Child2)
15. **until** nije MutatedChild2 u Population
16. Child2 $\leftarrow$ MutatedChild2
17. dvije najlošije rangirane jedinke između $i$-te i $(i+3)$-će jedinke $\leftarrow$ Children

### 4.2.8. Opis pseudokoda razvijenog algoritma

Glavna funkcija koja se koristi u ovom algoritmu je funkcija GA. Algoritam se može pokrenuti do određenog broja unaprijed definiranih potpunih resetiranja (MaxNrOfCompleteResets), a djelomična resetiranja moguća su do određenog broja unaprijed definiranih djelomičnih resetiranja (MaxNrOfPartialResets). Na početku algoritma stvara se populacija određenog broja

jedinki (POP) kao nasumična populacija, koja zadovoljava ograničenja orbitne matrice. U svakoj iteraciji algoritma radimo na ovoj populaciji kako bismo stvorili jedinke s boljim svojstvima. Algoritam je postavljen da radi sve dok se ne pronađe jedinka koja zadovoljava naše kriterije (tj. matrica je incidencije dizajna), dok ne dođe do stagnacije (u tom slučaju vršimo djelomično resetiranje algoritma) ili dok se ne dosegne granica za broj potpunih resetiranja (MaxNrOfCompleteResets).

Dobrota svake pojedine jedinke evaluira se pomoću funkcije FitnessOfIndividual, a jedinka predstavlja matricu incidencije dizajna ako je njezina dobrota maksimalna moguća, odnosno, FitnessForDesign.

Djelomično resetiranje populacije podrazumijeva da se unaprijed definirani postotak (StartingPercentage) najbolje rangiranih jedinki u populaciji (StartingPopulation) zadržava u toj populaciji zajedno s novim nasumično generiranim jedinkama. Djelomično resetiranje populacije provodi se kako bi se izbjegla stagnacija, odnosno pojava da veći broj generacija zaredom postiže isti lokalni optimum (u našem slučaju lokalni maksimum s obzirom na to kako je definirana naša funkcija dobrote). Kako bismo otkrili stagnaciju, definiramo najbolju vrijednost funkcije dobrote (f_best) u jednoj generaciji kao maksimum dobrota svih jedinki u toj generaciji. Zatim algoritam detektira lokalni maksimum ako određeni unaprijed definirani broj generacija (f_bestNrOfRepeatsMax) stagnira u nekoj vrijednosti najbolje dobrote (f_best). Djelomično resetiranje također se može provesti ako se postigne unaprijed definirani maksimalni broj generacija MaxNrOfGenerations. Ako se nakon ovog unaprijed definiranog broja djelomičnih resetiranja ne dobije rješenje, provodi se potpuno resetiranje.

Jedinke, matrice incidencije 1-dizajna, se implementiraju kao blok matrice čiji su blokovi prethodno opisani bitovi i to na temelju orbitne matrice i pretpostavljene grupe automorfizama. Ove se matrice kreiraju polunasumično u sljedećem smislu: svaki element orbitne matrice se proširuje poštujući distribuciju blokova i točaka i ovo proširenje je ili jedinstveno (tj. ti su dijelovi orbitne matrice fiksni) ili se može proširiti na više od jednog načina. Odgovarajuća kombinacija proširenja svih unosa orbitne matrice može proizvesti matricu incidencije blok dizajna. Uzimajući u obzir mnoge moguće kandidate za matricu incidencije među tim proširenjima, u slučajevima kada se iscrpno pretraživanje ne može provesti, heuristički algoritmi su dobra opcija.

Funkcija NewIndividual stvara blok matricu polunasumično generiranih gena pomoću funkcije NewGene (svaki gen odgovara retku orbitne matrice koji pak odgovara predstavniku orbite

točaka). Ovi geni sastoje se od bitova (koji odgovaraju elementima u tom retku orbitne matrice) koji se generiraju pomoću funkcije NewBit. Gen se generira na način da zadovoljava očekivanu dobrotu gena (ExpectedGeneFitness). To znači da je prostor rješenja problema unaprijed sužen i sadrži samo one jedinke kod kojih se unutar svake orbite točaka svake dvije točke pojavljuju u točno $\lambda$ zajedničkih blokova.

Funkcija NewBit proširuje element orbitne matrice u matricu sa željenim brojem redaka i stupaca (poštujući distribucije duljina orbita). Prvi redak te matrice konstruira se nasumično, a svi ostali retci određeni su djelovanjem pretpostavljene grupe automorfizama. Elementi fiksnog dijela orbitne matrice se proširuju u jedinstvenu takvu matricu.

Funkcija GeneratePopulation generira novu populaciju od POP jedinki, bilo iz prazne populacije ili s već postojećim najbolje rangiranim jedinkama nakon djelomičnog resetiranja.

Selekcija koji se koristi u algoritmu je 4-turnirska selekcija (iz tog razloga POP mora biti višekratnik od 4) gdje se u svakoj iteraciji algoritma populacija dijeli u grupe od četiri pojedinca, među kojima su dvije bolje rangirane (s obzirom na funkciju dobrote) odabrane su kao roditelji za proces križanja. Križanjem tih dvaju roditelja nastaje dvoje djece koja se potom mutiraju i te dvije jedinke zamjenjuju dvije lošije rangirane jedinke u 4-turniru. Tijekom ovog postupka, ako se dogodi da su roditelji odabrani za križanje jednaki, jedan od roditelja se mutira sve dok ne postanu različiti kako bi se održala raznolikost u populaciji. Slično, kada se mutira dijete, to se čini tako da se rezultirajuća jedinka razlikuje od bilo koje druge jedinke u trenutnoj populaciji.

Nakon svake iteracije algoritma, cijela populacija se permutira funkcijom Shuffle kako bi se osiguralo da roditelji iz jedne generacije ne budu uvijek izabrani kao roditelji u sljedećoj generaciji.

# 4.3. TESTNI PRIMJERI I OPTIMIZACIJA ALGORITMA

## 4.3.1. Testni primjeri

Kao primjere za testiranje rada algoritma koristili smo orbitne matrice poznatih simetričnih i nesimetričnih dizajna koje navodimo u nastavku. Za svaki primjer dizajna odredili smo optimalnu vrijednost funkcije dobrote FitnessForDesign u odnosu na prvu funkciju dobrote definiranu u 4.2.3 (za drugu funkciju optimalna vrijednost je uvijek 0). Odredili smo pune grupe automorfizama $\mathrm{Aut}(\mathscr{D})$ za svaki od tih dizajna te moguće grupe $G$ prostog reda izomorfne podgrupama od $\mathrm{Aut}(\mathscr{D})$ koje djeluju na taj dizajn. Za svaku takvu grupu određena je orbitna matrica za djelovanje po jedne podgrupe od $\mathrm{Aut}(\mathscr{D})$ izomorfne sa $G$ na dizajn. Na temelju parametara dizajna i dobivene orbitne matrice, pokušali smo pomoću algoritma konstruirati dizajn s tim parametrima.

- projektivne ravnine reda $n$, 2-$(n^2+n+1, n+1, 1)$ dizajni (simetrični)

| $n$ | parametri | FitnessForDesign | $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | $G$ |
|---|---|---|---|---|
| 2 | 2-(7,3,1) | 21 | 168 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| 3 | 2-(13,4,1) | 78 | 5616 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| 4 | 2-(21,5,1) | 210 | 120960 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| 5 | 2-(31,6,1) | 465 | 372000 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| 7 | 2-(57,8,1) | 1596 | 5630688 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_7$ |

- Hadamardovi dizajni 2-$(4n+3, 2n+1, n)$, $n \geq 2$ (simetrični)

| $n$ | parametri | FitnessForDesign | $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | $G$ |
|---|---|---|---|---|
| 2 | 2-(11,5,2) | 110 | 660 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| 3 | 2-(15,7,3) | 315 | 20160 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| 4 | 2-(19,9,4) | 684 | 171 | $\mathbb{Z}_3$ |
| 5 | 2-(23,11,5) | 1265 | 660 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_{11}$ |
| 6 | 2-(27,13,6) | 2106 | 78 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_{13}$ |
| 7 | 2-(31,15,7) | 3255 | 9999360 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| 8 | 2-(35,17,8) | 4760 | 136 | $\mathbb{Z}_2$ |

- afine ravnine reda $n$, 2-$(n^2, n, 1)$ dizajni (nesimetrični)

| $n$ | parametri | FitnessForDesign | $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | $G$ |
|---|---|---|---|---|
| 3 | 2-$(9,3,1)$ | 36 | 432 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| 4 | 2-$(16,4,1)$ | 120 | 5 760 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| 5 | 2-$(25,5,1)$ | 300 | 12 000 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| 7 | 2-$(49,7,1)$ | 1 176 | 98 784 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_7$ |
| 8 | 2-$(64,8,1)$ | 2016 | 677 376 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_7$ |

- Steinerovi sustavi trojki, 2-$(6t+3,3,1)$ dizajni, $t \geq 2$ (nesimetrični)

| $t$ | parametri | FitnessForDesign | $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | $G$ |
|---|---|---|---|---|
| 2 | 2-$(15,3,1)$ | 105 | 60 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| 3 | 2-$(21,3,1)$ | 210 | 126 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_7$ |
| 4 | 2-$(27,3,1)$ | 351 | 486 | $\mathbb{Z}_2, \mathbb{Z}_3$ |

- Steinerovi sustavi trojki, 2-$(6t+1,3,1)$ dizajni, $t \geq 2$ (nesimetrični)

| $t$ | parametri | FitnessForDesign | $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | $G$ |
|---|---|---|---|---|
| 2 | 2-$(13,3,1)$ | 78 | 39 | $\mathbb{Z}_3$ |
| 3 | 2-$(19,3,1)$ | 171 | 3 | $\mathbb{Z}_3$ |
| 4 | 2-$(25,3,1)$ | 300 | 3 | $\mathbb{Z}_3$ |

Ovi početni primjeri služili su nam kako bismo ispitali ponašanje algoritma u odnosu na: veličinu populacije POP, maksimalan broj generacija MaxNrOfGenerations, vjerojatnost križanja $p_c$ i broja gena koji sudjeluju u križanju NrGenesForCrossover, način križanja, vjerojatnost mutacije $p_m$ i broja bitova koji se mutiraju NrBitsForMutation, maksimalan broj ponavljanja iste najbolje vrijednosti dobrote f_bestNrOfRepeatsMax koji sugerira stagnaciju, maksimalan broj djelomičnih i potpunih resetiranja MaxNrOfPartialResets i MaxNrOfCompleteResets te postotak najbolje rangiranih jedniki StartingPercentage koje se prenose u sljedeću populaciju nakon djelomičnog resetiranja. Također, algoritam je testiran i za dvije varijante funkcije dobrote opisane u 4.2.3 te za varijante operatora selekcije opisane u 4.2.6.

U seriji napravljenih eksperimenata, kao zadane vrijednosti parametara koristimo:

- veličinu populacije POP $= 100$,
- MaxNrOfGenerations $= 100\,000$,
- vjerojatnost mutacije $p_m = 100\%$,
- vjerojatnost križanja $p_c = 100\%$,
- broj gena koji sudjeluju u križanju NrGenesForCrossover $= 1$,
- broj bitova koji se mutiraju NrBitsForMutation $= 1$,
- f_bestNrOfRepeatsMax $= 100$,
- MaxNrOfPartialResets $= 10$,
- MaxNrOfCompleteResets $= 100$,
- StartingPercentage $= 10\%$.

Zadana funkcija dobrote je prva opisana funkcija u 4.2.3, selekcija 4-turnirska.

Eksperimentalno dobivene rezultate prikazat ćemo u nastavku. Kada algoritam pronađe dizajn sa željenim parametrima, dozvolili smo da traži i nova rješenja sve dok se ne postigne ukupno MaxNrOfCompleteResets $= 100$ potpunih resetiranja. Dobiveni broj rješenja unutar tih 100 resetiranja može biti i veći od 100 budući da se u istoj iteraciji algoritma mogu pojaviti dvije ili više jedinki s funkcijom dobrote FitnessForDesign.

### 4.3.2. Optimizacija vjerojatnosti mutacije $p_m$

Uz zadane vrijednosti ostalih parametara, mijenjali smo vjerojatnosti mutacije od 95, 98 i 100 posto i zabilježili prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja). Pretpostavljena grupa automorfizama koja djeluje na dizajne u tablici 4.1 je $\mathbb{Z}_2$. Rezultati pokazuju da se povećanjem vjerojatnosti mutacije u pravilu smanjuje potreban prosječan broj iteracija koje vode do prvog rješenja, a sukladno tome i povećava broj dobivenih rješenja. Na temelju provedenog eksperimenta zaključili smo da ćemo zadržati vjerojatnost mutacije $p_m = 100\%$, odnosno, mutirat će se svako dijete dobiveno križanjem.

| parametri | FitnessForDesign | $p_m$ | 95 | 98 | 100 |
|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 12 | 12 | 9 |
| | | #rješenja | 112 | 118 | 132 |
| 2-(31, 6, 1) | 465 | #iteracija | 58 | 49 | 45 |
| | | #rješenja | 87 | 91 | 103 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 295 | 280 | 251 |
| | | #rješenja | 48 | 47 | 51 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 214 | 207 | 204 |
| | | #rješenja | 50 | 51 | 53 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 387 | 343 | 322 |
| | | #rješenja | 39 | 41 | 44 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 918 | 887 | 842 |
| | | #rješenja | 28 | 30 | 30 |
| 2-(25, 5, 1) | 300 | #iteracija | 38 | 34 | 27 |
| | | #rješenja | 104 | 106 | 111 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 284 | 274 | 266 |
| | | #rješenja | 61 | 61 | 62 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 448 | 432 | 410 |
| | | #rješenja | 30 | 31 | 31 |
| 2-(21, 3, 1) | 210 | #iteracija | 143 | 138 | 120 |
| | | #rješenja | 50 | 54 | 55 |
| 2-(27, 3, 1) | 351 | #iteracija | 228 | 222 | 201 |
| | | #rješenja | 34 | 48 | 50 |

Tablica 4.1: Optimizacija vjerojatnosti mutacije.

### 4.3.3. Optimizacija broja mutiranih bitova

Najmanja mutacija koju možemo napraviti na jednoj jedinki jest mutacija jednog bita. Postavlja se pitanje bi li se mutacijom većeg broja bitova odjednom ubrzao rad genetskog algoritma, odnosno ubrzalo pretraživanje prostora rješenja.

**Primjer 4.3.1.** Za parametre 2-(19,9,4) dizajna i djelovanje grupe $\mathbb{Z}_3$ imamo, na primjer, orbitnu matricu

| | 1 | 3 | 3 | 3 | 3 | 3 | 3 |
|---|---|---|---|---|---|---|---|
| 3 | **0** | **0** | **3** | **0** | **0** | **3** | **3** |
| 3 | **1** | 1 | 2 | 2 | 1 | **0** | 2 |
| 3 | **1** | 1 | 2 | 1 | 2 | 2 | **0** |
| 3 | **1** | 2 | **0** | 1 | 1 | 2 | 2 |
| 3 | **0** | **3** | 2 | 1 | 1 | 1 | 1 |
| 3 | **0** | 1 | 1 | **3** | 1 | 2 | 1 |
| 3 | **0** | 1 | 1 | 1 | **3** | 1 | 2 |

iz koje možemo konstruirati, na primjer, jedinku

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | **0** | **0** | **0** | 0 | 1 | 1 |
| **1** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | **0** | **0** | **0** | 1 | 0 | 1 |
| **1** | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | **0** | **0** | **0** | 1 | 1 | 0 |
| **1** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | **0** | **0** | **0** |
| **1** | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | **0** | **0** | **0** |
| **1** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | **0** | **0** | **0** |
| **1** | 0 | 1 | 1 | **0** | **0** | **0** | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| **1** | 1 | 0 | 1 | **0** | **0** | **0** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| **1** | 1 | 1 | 0 | **0** | **0** | **0** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| **0** | **1** | **1** | **1** | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **0** | **1** | **1** | **1** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **0** | **1** | **1** | **1** | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **0** | 1 | 0 | 0 | 0 | 0 | 1 | **1** | **1** | **1** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| **0** | 0 | 1 | 0 | 1 | 0 | 0 | **1** | **1** | **1** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| **0** | 0 | 0 | 1 | 0 | 1 | 0 | **1** | **1** | **1** | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **0** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | **1** | **1** | **1** | 0 | 0 | 1 | 0 | 1 | 1 |
| **0** | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | **1** | **1** | **1** | 1 | 0 | 0 | 1 | 0 | 1 |
| **0** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | **1** | **1** | **1** | 0 | 1 | 0 | 1 | 1 | 0 |

.

Pritom su fiksni elementi orbitne matrice, odnosno, fiksni bitovi jedinke označeni podebljano. Od ukupno 49 bitova od kojih se sastoji jedinka, njih 19 je fiksno i 30 nefiksno. Mutiranje jednog bita ove jedinke značilo bi mutaciju otprilike 3.33% njezinih nefiksnih bitova. Broj nefiksnih bitova označimo sa NrNonFixBits, a postotak nefiksnih bitova koje ćemo mutirati sa NonFixBitsMutationPercentage. Tada ćemo potreban broj nefiksnih bitova NrBitsForMutation izračunavati kao najmanje cijelo njihovog umnoška, odnosno,

$$\text{NrBitsForMutation} = \lceil \text{NrNonFixBits} \cdot \text{NonFixBitsMutationPercentage} \rceil.$$

Na primjer, 1% i 2% ovdje znači da ćemo mutirati 1 bit, 5% da ćemo mutirati 2 bita, 10% da ćemo mutirati 3 bita itd. U tablici 4.2 navodimo kako se algoritam ponaša na testnim primjerima za postotke mutacije 1%, 5% i 10%. Pretpostavljena grupa automorfizama koja djeluje na dizajne u tablici je $\mathbb{Z}_2$. Rezultati pokazuju da se povećanje tog postotka, odnosno broja mutiranih bitova negativno odražava na efikasnost algoritma, tj. povećava se broj potrebnih iteracija i istovremeno smanjuje broj dobivenih rješenja. Prema tome, nadalje ćemo mutirati samo jedan bit.

| parametri | FitnessForDesign | % | 1 | 5 | 10 |
|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 9 | 20 | 64 |
| | | #rješenja | 132 | 107 | 48 |
| 2-(31, 6, 1) | 465 | #iteracija | 47 | 105 | 299 |
| | | #rješenja | 101 | 70 | 24 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 262 | 390 | 1 741 |
| | | #rješenja | 50 | 35 | 11 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 205 | 512 | 1 877 |
| | | #rješenja | 51 | 27 | 10 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 334 | 697 | 2 541 |
| | | #rješenja | 41 | 30 | 8 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 857 | 1 704 | 4 526 |
| | | #rješenja | 32 | 13 | 3 |
| 2-(25, 5, 1) | 300 | #iteracija | 29 | 68 | 207 |
| | | #rješenja | 115 | 75 | 20 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 278 | 465 | 1 305 |
| | | #rješenja | 60 | 47 | 13 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 420 | 1 080 | 3 553 |
| | | #rješenja | 28 | 16 | 7 |
| 2-(21, 3, 1) | 210 | #iteracija | 120 | 234 | 915 |
| | | #rješenja | 57 | 40 | 29 |
| 2-(27, 3, 1) | 351 | #iteracija | 200 | 447 | 1 103 |
| | | #rješenja | 48 | 28 | 11 |

Tablica 4.2: Optimizacija broja mutiranih bitova.

### 4.3.4. Optimizacija vjerojatnosti križanja $p_c$

Uz zadane vrijednosti ostalih parametara, mijenjali smo vjerojatnosti križanja od 90, 95 i 100 posto i zabilježili prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja). Pretpostavljena grupa automorfizama koja djeluje na dizajne u tablici 4.3 je $\mathbb{Z}_2$. Rezultati pokazuju da se po-

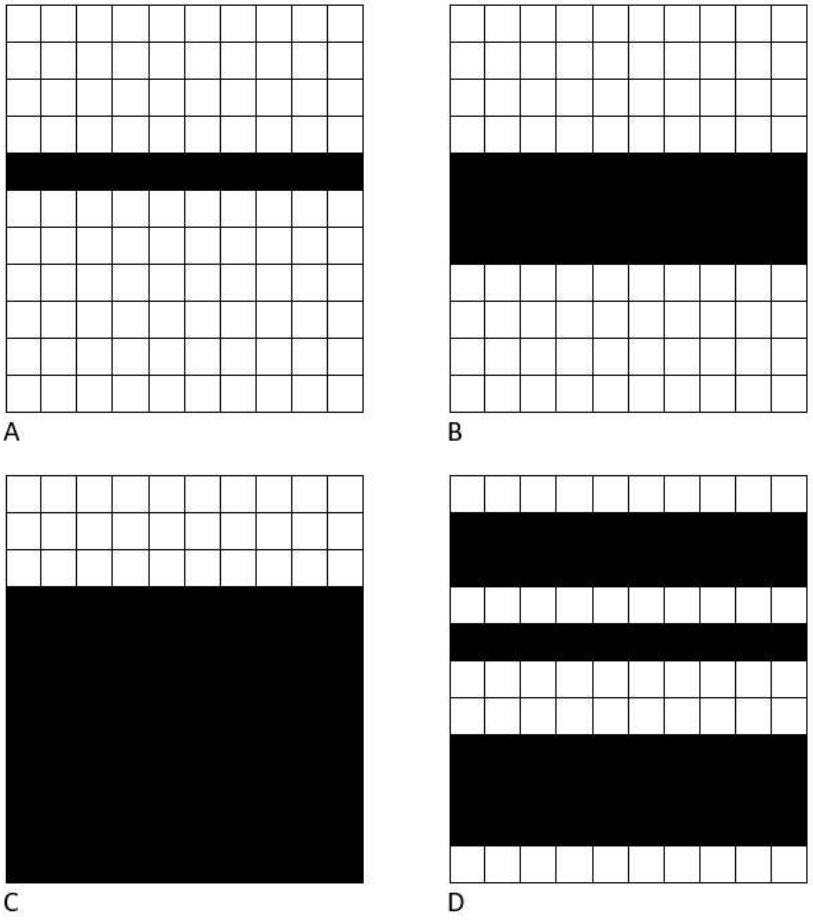
većanjem vjerojatnosti križanja u pravilu smanjuje potreban prosječan broj iteracija koje vode do prvog rješenja, a sukladno tome i povećava broj dobivenih rješenja. Na temelju provedenog eksperimenta zaključili smo da ćemo zadržati vjerojatnost križanja $p_c = 100\%$, odnosno, svaka dva roditelja će križati svoje gene.

| parametri | FitnessForDesign | $p_c$ | 90 | 95 | 100 |
|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 24 | 16 | 9 |
| | | #rješenja | 110 | 117 | 132 |
| 2-(31, 6, 1) | 465 | #iteracija | 64 | 52 | 45 |
| | | #rješenja | 80 | 85 | 103 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 307 | 267 | 251 |
| | | #rješenja | 41 | 44 | 51 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 222 | 217 | 204 |
| | | #rješenja | 47 | 49 | 53 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 389 | 367 | 322 |
| | | #rješenja | 34 | 39 | 44 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 924 | 885 | 842 |
| | | #rješenja | 26 | 28 | 30 |
| 2-(25, 5, 1) | 300 | #iteracija | 47 | 39 | 27 |
| | | #rješenja | 91 | 94 | 111 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 291 | 279 | 266 |
| | | #rješenja | 58 | 60 | 62 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 451 | 437 | 410 |
| | | #rješenja | 26 | 29 | 31 |
| 2-(21, 3, 1) | 210 | #iteracija | 151 | 146 | 120 |
| | | #rješenja | 50 | 52 | 55 |
| 2-(27, 3, 1) | 351 | #iteracija | 231 | 217 | 201 |
| | | #rješenja | 33 | 46 | 50 |

Tablica 4.3: Optimizacija vjerojatnosti križanja.

### 4.3.5. Optimizacija broja križanih gena i načina križanja

Najmanje križanje koje možemo provesti između dvije jedinke (roditelja) jest križanje jednog gena, odnosno, križanje u dvije uzastopne točke. Postavlja se pitanje bi li se križanjem većeg broja gena odjednom ubrzao rad genetskog algoritma, odnosno ubrzalo pretraživanje prostora rješenja. Neki od načina za križanje većeg broja gena su križanje u dvije točke koje nisu uzastopne (pri čemu obuhvaćamo 2 ili više gena), križanje u jednoj točki ili križanje u više točaka kao što je prikazano na slici 4.1.




Slika 4.1: Križanje jednog gena (A), križanje više gena u dvije točke (B), križanje u jednoj točki (C) i križanje u više točaka (D).

Jedinka iz primjera 4.3.1 ima ukupno 7 gena, od čega je 1 fiksan i 6 nefiksnih. Križanje jednog gena dviju jedinki ovog tipa značilo bi križanje otprilike 16.66% njezinih nefiksnih bitova. Broj nefiksnih gena označimo s NrNonFixGenes, a postotak nefiksnih gena koje ćemo mutirati s NonFixGenesCrossoverPercentage. Tada ćemo potreban broj nefiksnih gena NrGenesForCro-

ssover izračunavati kao najmanje cijelo njihovog umnoška, odnosno,

$$\text{NrGenesForCrossover} = \lceil \text{NrNonFixGenes} \cdot \text{NonFixGenesCrossoverPercentage} \rceil.$$

Na primjer, 10% ovdje znači da ćemo križati 1 gen, 25% da ćemo križati 2 gena, 50% da ćemo križati 3 gena itd. Primijetimo da je najveća moguća vrijednost postotka NonFixGenesCrossoverPercentage 50%. Naime, križanje $p$ posto gena dviju jedinki isto je što i križanje $100 - p$ posto preostalih gena. U tablici 4.4 navodimo kako se algoritam ponaša na testnim primjerima za postotke mutacije 10%, 25% i 50% i za križanje u više točaka, što odgovara križanju odgovarajućeg postotka na slučajno odabranim pozicijama gena u jedinkama. Isti eksperimenti provedeni su i za križanje 10%, 25% i 50% gena u dvije točke te križanje 10%, 25% i 50% gena u jednoj točki. Između ta tri načina križanja nije zamijećena značajna razlika pa u tablici prikazujemo rezultate samo za križanje u više točaka. Pretpostavljena grupa automorfizama koja djeluje na dizajne u tablici je $\mathbb{Z}_2$. Rezultati pokazuju da se povećanje tog postotka, odnosno broja križanih gena između dvaju roditelja negativno odražava na efikasnost algoritma, tj. povećava se broj potrebnih iteracija i istovremeno smanjuje broj dobivenih rješenja. Prema tome, nadalje ćemo križati samo jedan gen.

| parametri | FitnessForDesign | % | 10 | 25 | 50 |
|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 12 | 18 | 51 |
| | | #rješenja | 128 | 98 | 79 |
| 2-(31, 6, 1) | 465 | #iteracija | 44 | 92 | 230 |
| | | #rješenja | 104 | 64 | 59 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 258 | 345 | 1 101 |
| | | #rješenja | 47 | 31 | 13 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 201 | 482 | 1 640 |
| | | #rješenja | 46 | 34 | 12 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 313 | 617 | 1 917 |
| | | #rješenja | 40 | 32 | 10 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 841 | 1 711 | 3 974 |
| | | #rješenja | 31 | 16 | 5 |
| 2-(25, 5, 1) | 300 | #iteracija | 30 | 59 | 194 |
| | | #rješenja | 107 | 69 | 23 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 250 | 398 | 1 007 |
| | | #rješenja | 56 | 47 | 14 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 404 | 985 | 2 934 |
| | | #rješenja | 23 | 18 | 9 |
| 2-(21, 3, 1) | 210 | #iteracija | 113 | 202 | 874 |
| | | #rješenja | 54 | 39 | 32 |
| 2-(27, 3, 1) | 351 | #iteracija | 199 | 411 | 998 |
| | | #rješenja | 46 | 30 | 12 |

Tablica 4.4: Optimizacija broja križanih gena pri križanju u više točaka.

### 4.3.6. Usporedba dviju funkcija dobrote

Rad algoritma testirali smo i za dvije funkcije dobrote definirane u 4.2.3, odnosno, usporedili smo prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja) za testne dizajne i pretpostavljeno

djelovanje grupe $\mathbb{Z}_2$. Prva funkcija je, kao što je opisano u 4.2.3,

$$\sum_{P_i,P_j\in\mathscr{P}} \min\{x_{i,j},\lambda\},$$

a druga funkcija je

$$\sum_{P_i,P_j\in\mathscr{P}} (x_{i,j}-\lambda)^2.$$

Ostali parametri algoritma su ranije opisani zadani parametri. Eksperimentalno dobiveni rezultati u tablici 4.5 pokazuju da je pri korištenju prve funkcije dobrote algoritmu potreban manji prosječan broj iteracija do prvog rješenja. Nije zamijećena značajna razlika u broju dobivenih rješenja. Prema tome, u algoritmu ćemo koristiti prvu funkciju dobrote.

| parametri | FitnessForDesign | | prva funkcija | druga funkcija |
|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 9 | 9 |
| | | #rješenja | 132 | 134 |
| 2-(31, 6, 1) | 465 | #iteracija | 45 | 49 |
| | | #rješenja | 103 | 99 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 251 | 425 |
| | | #rješenja | 51 | 53 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 204 | 219 |
| | | #rješenja | 53 | 47 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 322 | 978 |
| | | #rješenja | 44 | 39 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 842 | 2 432 |
| | | #rješenja | 30 | 28 |
| 2-(25, 5, 1) | 300 | #iteracija | 27 | 42 |
| | | #rješenja | 111 | 104 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 266 | 567 |
| | | #rješenja | 62 | 58 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 410 | 2 001 |
| | | #rješenja | 31 | 30 |
| 2-(21, 3, 1) | 210 | #iteracija | 120 | 468 |
| | | #rješenja | 55 | 57 |
| 2-(27, 3, 1) | 351 | #iteracija | 201 | 1 251 |
| | | #rješenja | 50 | 51 |

Tablica 4.5: Usporedba dviju funkcija dobrote.

### 4.3.7. Usporedba operatora selekcije

Rad algoritma testirali smo i za različite odabire operatora selekcije definiranih u 4.2.6, odnosno, usporedili smo prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja) za testne dizajne i pretpostavljeno djelovanje grupe $\mathbb{Z}_2$. Ostali parametri algoritma su ranije opisani zadani

parametri. Eksperimentalno dobiveni rezultati u tablici 4.6 pokazuju da je pri korištenju 4-turnirske selekcije potreban najmanji prosječan broj iteracija do prvog rješenja, a i najveći je broj dobivenih rješenja.

| parametri | FitnessForDesign | selekcija | 4-turnirska | jednostavna | eliminacijska |
|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 9 | 24 | 17 |
| | | #rješenja | 132 | 86 | 91 |
| 2-(31, 6, 1) | 465 | #iteracija | 45 | 102 | 97 |
| | | #rješenja | 103 | 81 | 84 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 251 | 510 | 418 |
| | | #rješenja | 51 | 34 | 43 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 204 | 478 | 310 |
| | | #rješenja | 53 | 41 | 46 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 322 | 843 | 706 |
| | | #rješenja | 44 | 28 | 37 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 842 | 3 112 | 1 974 |
| | | #rješenja | 30 | 13 | 19 |
| 2-(25, 5, 1) | 300 | #iteracija | 27 | 91 | 78 |
| | | #rješenja | 111 | 84 | 91 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 266 | 876 | 570 |
| | | #rješenja | 62 | 31 | 49 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 410 | 1 983 | 912 |
| | | #rješenja | 31 | 12 | 20 |
| 2-(21, 3, 1) | 210 | #iteracija | 120 | 493 | 301 |
| | | #rješenja | 55 | 23 | 38 |
| 2-(27, 3, 1) | 351 | #iteracija | 201 | 597 | 317 |
| | | #rješenja | 50 | 20 | 31 |

Tablica 4.6: Usporedba operatora selekcije.

### 4.3.8. Optimizacija veličine populacije POP

Uz zadane vrijednosti ostalih parametara, mijenjali smo vrijednosti veličine populacije na POP = 20, POP = 40, POP = 100 i POP = 200 i zabilježili prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja). Pretpostavljena grupa automorfizama koja djeluje na dizajne u tablici 4.7 je $\mathbb{Z}_2$. Rezultati pokazuju očekivani ishod: ukoliko je populacija manja, iteracije algoritma se brže izvode, no potreban je veći broj iteracija za dobivanje rješenja nego što je to u slučaju veće populacije. Povećanjem broja jedinki u populaciji se, naravno, produljuje vrijeme izvođenja algoritma. Kod većih populacija, zbog veće količine genetskog materijala koji se obrađuje u svakom koraku, primijećeno je da se pojavljuje više optimalnih rješenja, iako su ona uglavnom ili ista ili izomorfna. Za zadanu vrijednost veličine populacije odlučili smo zadržati POP = 100 kao optimalnu vrijednost.

| parametri | FitnessForDesign | POP | 20 | 40 | 100 | 200 |
|---|---|---|---|---|---|---|
| 2-(21, 5, 1) | 210 | #iteracija | 30 | 16 | 9 | 6 |
| | | #rješenja | 101 | 106 | 132 | 138 |
| 2-(31, 6, 1) | 465 | #iteracija | 117 | 91 | 45 | 38 |
| | | #rješenja | 86 | 87 | 103 | 105 |
| 2-(57, 8, 1) | 1 596 | #iteracija | 550 | 398 | 251 | 214 |
| | | #rješenja | 38 | 44 | 51 | 59 |
| 2-(23, 11, 5) | 1 265 | #iteracija | 491 | 317 | 204 | 148 |
| | | #rješenja | 36 | 47 | 53 | 53 |
| 2-(27, 13, 6) | 2 106 | #iteracija | 601 | 428 | 322 | 298 |
| | | #rješenja | 39 | 43 | 44 | 46 |
| 2-(31, 15, 7) | 3 255 | #iteracija | 1 335 | 997 | 842 | 710 |
| | | #rješenja | 25 | 27 | 30 | 31 |
| 2-(25, 5, 1) | 300 | #iteracija | 51 | 41 | 27 | 21 |
| | | #rješenja | 87 | 97 | 111 | 132 |
| 2-(49, 7, 1) | 1 176 | #iteracija | 570 | 368 | 266 | 231 |
| | | #rješenja | 50 | 54 | 62 | 64 |
| 2-(64, 8, 1) | 2 016 | #iteracija | 831 | 599 | 410 | 355 |
| | | #rješenja | 21 | 24 | 31 | 31 |
| 2-(21, 3, 1) | 210 | #iteracija | 307 | 247 | 120 | 108 |
| | | #rješenja | 51 | 54 | 55 | 57 |
| 2-(27, 3, 1) | 351 | #iteracija | 454 | 314 | 201 | 183 |
| | | #rješenja | 34 | 39 | 50 | 52 |

Tablica 4.7: Optimizacija veličine populacije POP.

### 4.3.9. Novi operator mutacije MaxMutation

Nakon testiranja algoritma na primjerima koje smo vidjeli do sada, isti algoritam smo primijenili na konstrukciju još nekih dizajna, uz pretpostavljeno djelovanje involucije. U tablici 4.8 prikazujemo neke od dobivenih rezultata, za zadane vrijednosti parametara algoritma koje smo definirali prethodno. Kao i u prethodnim serijama eksperimenata, dozvolili smo maksimalno

100 potpunih resetiranja algoritma i zabilježili prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do 2-dizajna (#iteracija) i broj dobivenih rješenja unutar 100 resetiranja (#rješenja). Također, zabilježili smo i prosječno potrebno vrijeme (u minutama) pronalaska rješenja. Računalo na kojem se izvodio algoritam ima sljedeće specifikacije: procesor 11th Gen Intel(R) Core(TM) i7-1165G7 2.60GHz, 16GB RAM memorije frekvencije 2803MHz.

| parametri | FitnessForDesign | #iteracija | #rješenja | vrijeme |
|---|---|---|---|---|
| 2-(56,11,2) | 3 080 | 3 543 | 14 | 136.43 |
| 2-(79,13,2) | 6 162 | 5 393 | 8 | 238.23 |
| 2-(45,5,1) | 990 | 1 353 | 16 | 44.10 |

Tablica 4.8: 2-(56,11,2), 2-(79,13,2) i 2-(45,5,1) dizajni.

Neki od dobivenih 2-(45,5,1) dizajna, tj. Steinerovih sustava $S(2,5,45)$, iz tablice 4.8 bili su neizomorfni s do tada poznatim dizajnima pa smo htjeli ispitati možemo li na temelju poznatih dizajna i njihovih orbitnih matrica dobiti još novih dizajna. U tu svrhu, kako bismo smanjili broj potrebnih generacija i vrijeme dok se ne dostigne dizajn, napravili smo modifikaciju mutacije.

Razvijen je drugi tip operatora mutacije, nazvan MaxMutation, kako bi se potraga usmjerila prema boljim jedinkama. To je učinjeno jer smo primijetili da nakon određenog broja generacija, kada su dobrote jedinki u populaciji blizu željene dobrote za dizajn, osnovna mutacija jedinke često uzrokuje smanjivanje vrijednosti dobrote te jedinke, što znači da takve jedinke obično bivaju odbačene kao lošije rangirane jedinke u 4-turniru. To dovodi do stagnacije jer su preostali pojedinci previše slični (imaju veliki broj međusobno jednakih gena, što je posljedica križanja, a novi genetski materijal uveden mutacijom je odbačen). Stoga smo uveli operator MaxMutation kao ispravljački operator.

Operator MaxMutation koristi prilagođenu vrstu mutacije, koja neće mutirati bilo koji slučajni bit u jedinki, već će otkriti "problematične" gene u jedinkama i pokušati mutirati bitove u tim genima kako bi se stvorila jedinka s većom vrijednosti funkcije dobrote. Na taj se način prostor pretraživanja istražuje na usmjereniji način, prema kandidatima za rješenje s većom vrijednosti funkcije dobrote.

Kako bismo otkrili "problematične" gene, promatramo djelomične vrijednosti funkcije dobrote samo onih redaka u jedinkama koji odgovaraju dvama ili više redaka orbitne matrice. Ako se ova djelomična vrijednost funkcije dobrote razlikuje od očekivane u slučaju kada je-

dinka predstavlja matricu incidencije dizajna, to znači da bismo mutiranjem nekih bitova u tom dijelu pojedinca mogli povećati njegovu dobrotu, umjesto da je smanjimo mutiranjem slučajnog bita koji može biti pozicioniran u nekom dijelu jedinke koji je već blizu tog dijela u matrici incidencije.

Koristeći ovu drugu vrstu mutacije, poboljšali smo rad algoritma s obzirom na osnovni algoritam, kao što se može vidjeti iz tablice 4.9. Računalo na kojem se izvodio algoritam ima sljedeće specifikacije: procesor 11th Gen Intel(R) Core(TM) i7-1165G7 2.60GHz, 16GB RAM memorije frekvencije 2803MHz.

| parametri | FitnessForDesign | #iteracija | #rješenja | vrijeme |
|---|---|---|---|---|
| 2-(56,11,2) | 3 080 | 63 | 19 | 33.03 |
| 2-(79,13,2) | 6 162 | 98 | 11 | 42.92 |
| 2-(45,5,1) | 990 | 24 | 21 | 6.44 |

Tablica 4.9: 2-(56,11,2), 2-(79,13,2) i 2-(45,5,1) dizajni uz korištenje mutacije MaxMutation.

# 4.4. REZULTATI

Korištenjem algoritma koji koristi modificirani tip mutacije MaxMutation, konstruirali smo nove Steinerove sustave $S(2,5,45)$ i simetrične (71,15,3) dizajne.

## 4.4.1. Konstrukcija novih Steinerovih sustava $S(2,5,45)$

Prema našim saznanjima, do sada je bilo poznato 30 Steinerovih sustava $S(2,5,45)$ (vidi [40]). To su Steinerovi sustavi konstruirani u [12], [16], [41], [47] i oni koji su konstruirani metodom opisanom u [48]. Metodom koju predlažemo konstruirali smo 35 novih Steinerovih sustava $S(2,5,45)$ koji dopuštaju djelovanje grupe grupe reda dva. Matrice incidencije 30 ranije poznatih Steinerovih sustava $S(2,5,45)$ može se naći na [40]. Matrice incidencije 35 Steinerovih sustava $S(2,5,45)$ konstruiranih u ovom radu dostupne su na poveznici:

`https://github.com/TinZrinski/structures/blob/main/mat_inc_new_35_S(2,5,45).txt`

Tijekom pokretanja algoritma, parametri POP, MaxNrOfCompleteResets i MaxNrOfPartialResets postavljeni su na POP $= 100$, MaxNrOfCompleteResets $= 100$, a MaxNrOfPartialResets $= 10$. Vjerojatnosti za mutaciju i križanje su $p_m = p_c = 100\%$, a mutira se jedan bit i križa jedan gen. Za djelomično resetiranje, postotak najbolje rangiranih pojedinaca u populaciji koji se zadržavaju u populaciji postavljen je na StartingPercentage $= 10\%$.

U tablici 4.10 dajemo informacije o punim grupama automorfizama i 2-rangovima ranije poznatih Steinerovih sustava $S(2,5,45)$.

| $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | struktura $\mathrm{Aut}(\mathscr{D})$ | 2-rang | #neizomorfnih dizajna |
|---|---|---|---|
| 360 | $(\mathbb{Z}_{15} \times \mathbb{Z}_3) : Q_8$ | 45 | 1 |
| 160 | $(E_{16} : \mathbb{Z}_5) : \mathbb{Z}_2$ | 36 | 1 |
| 72 | $E_9 : Q_8$ | 45 | 3 |
| 72 | $E_9 : Q_8$ | 37 | 3 |
| 40 | $\mathbb{Z}_5 : Q_8$ | 45 | 1 |
| 32 | $E_{16} : \mathbb{Z}_2$ | 36 | 1 |
| 24 | $\mathbb{Z}_2 \times A_4$ | 38 | 3 |
| 8 | $Q_8$ | 45 | 1 |
| 8 | $Q_8$ | 37 | 2 |
| 6 | $S_3$ | 37 | 2 |
| 4 | $\mathbb{Z}_4$ | 37 | 6 |
| 2 | $\mathbb{Z}_2$ | 37 | 2 |
| 1 | $I$ | 37 | 4 |

Tablica 4.10: Prethodno poznati Steinerovi sustavi $S(2,5,45)$.

Koristeći opisani algoritam, uspjeli smo konstruirati 35 novih Steinerovih sustava $S(2,5,45)$ iz orbitnih matrica za djelovanje grupe $\mathbb{Z}_2$. Orbitne matrice koje smo koristili za ovaj proces su sve orbitne matrice za djelovanje involucije dobivene iz 26 poznatih Steinerovih sustava $S(2,5,45)$ koji imaju netrivijalnu grupu automorfizama, a zatim i iz novih koje smo konstruirali, na način da smo za Steinerov sustav $\mathscr{D}$ konstruirali orbitnu matricu za svaku klasu konjugiranosti involucija iz grupe $\mathrm{Aut}(\mathscr{D})$. Ukupno postoji 239 orbitnih matrica, 44 od 30 ranije poznatih dizajna i 195 od 35 novokonstruiranih dizajna. Algoritam je zaustavljen kada nakon 48 sati izvođenja nakon zadnjeg pronađenog novog dizajna nije više pronađen niti jedan drugi novi dizajn.

Informacije o punim grupama automorfizama i 2-rangovima dobivenih novih dizajna prikazane su u tablici 4.11.

| $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | struktura $\mathrm{Aut}(\mathscr{D})$ | 2-rang | #neizomorfnih dizajna |
|---|---|---|---|
| 32 | $E_{16} : \mathbb{Z}_2$ | 36 | 2 |
| 16 | $(\mathbb{Z}_4 \times \mathbb{Z}_2) : \mathbb{Z}_2$ | 39 | 4 |
| 16 | $(\mathbb{Z}_4 \times \mathbb{Z}_2) : \mathbb{Z}_2$ | 38 | 8 |
| 16 | $\mathbb{Z}_2 \times D_8$ | 38 | 2 |
| 8 | $E_8$ | 37 | 3 |
| 8 | $E_8$ | 38 | 7 |
| 8 | $E_8$ | 39 | 1 |
| 8 | $\mathbb{Z}_4 \times \mathbb{Z}_2$ | 39 | 4 |
| 4 | $E_4$ | 39 | 4 |

Tablica 4.11: Novi Steinerovi sustavi $S(2,5,45)$.

U tablici 4.12 prikazujemo podatke o ukupnom broju (#rješenja) dobivenih Steinerovih sustava $S(2,5,45)$ korištenjem opisanog algoritma, minimalnom (#min), maksimalnom (#max) i prosječnom broju potrebnih iteracija (#avg) te minimalnom ($t_{\min}$), maksimalnom ($t_{\max}$) i prosječnom vremenu ($t_{\text{avg}}$) dobivanja rješenja (u minutama). Računalo na kojem se izvodio algoritam ima sljedeće specifikacije: procesor 11th Gen Intel(R) Core(TM) i7-1165G7 2.60GHz, 16GB RAM memorije frekvencije 2803MHz.

| parametri | FitnessForDesign | #rješenja | #min | #max | #avg | $t_{\min}$ | $t_{\max}$ | $t_{\text{avg}}$ |
|---|---|---|---|---|---|---|---|---|
| 2-$(45,5,1)$ | 990 | 9 245 | 3 | 119 | 24 | 0.28 | 58.61 | 6.51 |

Tablica 4.12: Rezultati algoritma za Steinerove sustave $S(2,5,45)$.

### 4.4.2. Konstrukcija novih simetričnih (71,15,3) dizajna

Simetrični (71,15,3) dizajni, tj. troravnine reda dvanaest, imaju najveći broj točaka među poznatim simetričnim $(v,k,3)$ dizajnima, a pitanje egzistencije simetričnih $(v,k,3)$ dizajna s $v > 71$ još uvijek je neodgovoreno. Do na izomorfizam, do sada je bilo poznato 148 simetričnih (71,15,3) dizajna, od kojih je 146 koji dopuštaju automorfizam reda šest kao što je opisano u [20], a dva međusobno dualna dizajna koji imaju $E_8$ kao punu grupu automorfizma dobiveni su iz kodova kao što je opisano u [19]. Matrice incidencije 146 simetričnih (71,15,3) dizajna koji dopuštaju automorfizam reda šest može se naći na [55].

Metodom koju predlažemo konstruirali smo 22 nova simetrična (71,15,3) dizajna, a matrice incidencije tih dizajna mogu se pronaći na poveznici:

`https://github.com/TinZrinski/structures/blob/main/mat_inc_new_22_(71,15,3).txt`

U tablici 4.13 dajemo informacije o punim grupama automorfizama i 2-rangovima do sada poznatih simetričnih (71,15,3) dizajna.

| $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | struktura $\mathrm{Aut}(\mathscr{D})$ | 2-rang | #neizomorfnih dizajna |
|---|---|---|---|
| 336 | $(E_8 : F_{21}) \times \mathbb{Z}_2$ | 27 | 2 |
| 336 | $(E_8 : F_{21}) \times \mathbb{Z}_2$ | 25 | 4 |
| 168 | $E_8 : F_{21}$ | 35 | 1 |
| 168 | $E_8 : F_{21}$ | 33 | 2 |
| 48 | $E_4 \times A_4$ | 29 | 4 |
| 48 | $E_4 \times A_4$ | 27 | 14 |
| 48 | $E_4 \times A_4$ | 25 | 8 |
| 42 | $F_{21} \times \mathbb{Z}_2$ | 35 | 2 |
| 42 | $F_{21} \times \mathbb{Z}_2$ | 33 | 4 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 35 | 1 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 34 | 16 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 33 | 2 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 32 | 10 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 31 | 34 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 30 | 2 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 29 | 18 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 28 | 4 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 27 | 1 |
| 24 | $A_4 \times \mathbb{Z}_2$ | 25 | 1 |
| 24 | $S_3 \times E_4$ | 32 | 8 |
| 24 | $S_3 \times E_4$ | 30 | 8 |
| 8 | $E_8$ | 33 | 2 |

Tablica 4.13: Do sada poznati simetrični (71,15,3) dizajni.

Koristeći isti algoritam kao i za konstrukciju novih Steinerovih sustava $S(2,5,45)$ s istim operatorom MaxMutation i istim ulaznim parametrima, uspjeli smo konstruirati 22 nova simetrična $(71,15,3)$ dizajna iz orbitnih matrica za djelovanje grupe $\mathbb{Z}_2$. Orbitne matrice koje smo koristili za ovaj proces su orbitne matrice dobivene iz 148 ranije poznatih simetričnih $(71,15,3)$ dizajna i orbitne matrice dobivene iz novih konstruiranih dizajna. Ukupno postoji 868 orbitnih matrica, 602 od 148 ranije poznatih dizajna i 266 od 22 nova konstruirana dizajna. Kao i prije, algoritam je zaustavljen kada nakon 48 sati izvođenja nakon zadnjeg pronađenog novog dizajna nije više pronađen niti jedan drugi novi dizajn.

Informacije o punim grupama automorfizama i 2-rangovima dobivenih novih dizajna prikazane su u tablici 4.14.

| $\lvert\mathrm{Aut}(\mathscr{D})\rvert$ | struktura $\mathrm{Aut}(\mathscr{D})$ | 2-rang | #neizomorfnih dizajna |
|---|---|---|---|
| 16 | $E_{16}$ | 29 | 4 |
| 16 | $E_{16}$ | 27 | 6 |
| 16 | $E_{16}$ | 25 | 4 |
| 8 | $E_8$ | 34 | 2 |
| 8 | $E_8$ | 31 | 2 |
| 8 | $E_8$ | 29 | 4 |

Tablica 4.14: Novi simetrični $(71,15,3)$ dizajni.

U tablici 4.15 prikazujemo podatke o ukupnom broju (#rješenja) dobivenih simetričnih $(71,15,3)$ dizajna korištenjem opisanog algoritma, minimalnom (#min), maksimalnom (#max) i prosječnom broju potrebnih iteracija (#avg) te minimalnom ($t_{\min}$), maksimalnom ($t_{\max}$) i prosječnom vremenu ($t_{\mathrm{avg}}$) dobivanja rješenja (u minutama). Računalo na kojem se izvodio algoritam ima sljedeće specifikacije: procesor 11th Gen Intel(R) Core(TM) i7-1165G7 2.60GHz, 16GB RAM memorije frekvencije 2803MHz.

| parametri | FitnessForDesign | #rješenja | #min | #max | #avg | $t_{\min}$ | $t_{\max}$ | $t_{\mathrm{avg}}$ |
|---|---|---|---|---|---|---|---|---|
| 2-$(71,15,3)$ | 7 455 | 213 | 17 | 306 | 105 | 86.07 | 434.40 | 195.19 |

Tablica 4.15: Rezultati algoritma za simetrične $(71,15,3)$ dizajne.

# 5. KONSTRUIRANJE JAKO REGULARNIH GRAFOVA S PRETPOSTAVLJENOM GRUPOM AUTOMORFIZAMA KORIŠTENJEM GENETSKOG ALGORITMA

U ovom poglavlju opisat ćemo metodu konstrukcije jako regularnih grafova koja kombinira genetski algoritam i metodu za konstruiranje jako regularnih grafova s pretpostavljenom grupom automorfizama korištenjem orbitnih matrica. Ovu metodu primijenit ćemo za konstrukciju novih jako regularnih grafova s parametrima $\mathrm{SRG}(96,19,2,4)$ i $\mathrm{SRG}(96,20,4,4)$.

## 5.1. UVOD

M. Behbahani u svom doktoratu [3] izrađenom pod mentorstvom C. Lama, uvodi orbitne matrice jako regularnih grafova i razvija algoritam za konstrukciju orbitnih matrica jako regularnih grafova za djelovanje automorfizma prostog reda. Iako se orbitne matrice blokovnih dizajna uspješno primjenjuju od 1980-ih godina za konstrukciju blokovnih dizajna, u radu Behbahanija i Lama [4] prvi su put definirane i korištene orbitne matrice jako regularnih grafova. Direktna konstrukcija matrice susjedstva jako regularnog grafa često predugo traje čak i za grafove s malo vrhova. Da bi se konstrukciju grafova učinilo mogućom, može se koristiti metoda konstrukcije uz pomoć orbitnih matrica koju su uveli Behbahani i Lam. Najprije se pretpostavi djelovanje određene grupe automorfizma i odrede sve moguće distribucije duljina orbita. Zatim se, pomoću u tu svrhu razvijenih algoritama i na njima temeljenih računalnih programa, konstruiraju

odgovarajuće orbitne matrice jako regularnih grafova. Nakon toga se iz dobivenih orbitnih matrica konstruiraju matrice susjedstva jako regularnog grafa. M. Maksimović i D. Crnković su u [43] i u [17] napravili generalizaciju navedenog rezultata, odnosno izradili algoritme i računalne programe za konstrukciju orbitnih matrica jako regularnih grafova za djelovanje grupe automorfizama čiji red nije nužno prost broj i konstruirali nove jako regularne grafove iz dobivenih orbitnih matrica. Konstrukcija jako regularnih grafova iz orbitnih matrica uspješno je korištena i u radu D. Crnkovića, A. Švob i V. D. Toncheva [23].

U cilju da se omogući konstrukcija jako regularnih grafova koja ne bi bila izvediva izvođenjem iscrpnog pretraživanja, predlažemo kombinaciju metode koja koristi orbitne matrice i genetskog algoritma. Koliko nam je poznato, ovo je prva primjena genetskog algoritma za konstrukciju jako regularnih grafova s pretpostavljenom grupom automorfizma, korištenjem orbitnih matrica.

## 5.2. KONSTRUKCIJA

Konstrukcija jako regularnih grafova koji dopuštaju djelovanje pretpostavljene grupe automorfizama slična je konstrukciji blokovnih dizajna s pretpostavljenom grupom automorfizma. Sastoji se od sljedeća dva osnovna koraka (vidi [17], [43]):

1. konstrukcije orbitnih matrica za pretpostavljenu grupu automorfizma,
2. konstrukcije jako regularnih grafova za orbitne matrice dobivene na ovaj način.

Cilj drugog koraka konstrukcije, indeksiranja, je konstruirati matrice susjedstva jako regularnih grafova koje odgovaraju orbitnim matricama dobivenim u prvom koraku. Indeksiranje orbitnih matrica obično se provodi metodom iscrpnog pretraživanja. Međutim, kao i u slučaju blokovnih dizajna, ponekad iscrpno pretraživanje nije izvedivo jer postoji previše struktura koje bi trebalo provjeriti, što je zahtjevno zbog ograničenih kapaciteta računalne memorije, a isto tako i samog vremena izvođenja pretraživanja. Kako bismo prevladali ovu prepreku, predlažemo korištenje genetskog algoritma u ovom koraku konstrukcije, na sličan način kao u poglavlju 4.

Neke od orbitnih matrica koje se koriste u procesu indeksiranja ne mogu producirati jako regularne grafove sa željenim parametrima, dok se iz drugih orbitnih matrica može dobiti više jako regularnih grafova koji su međusobno neizomorfni. Kao i kod blokovnih dizajna, nećemo konstruirati orbitne matrice za određeni dopustivi skup parametara jako regularnog grafa, već ćemo iz orbitnih matrica već poznatih jako regularnih grafova pokušati konstruirati nove jako regularne grafove, neizomorfne s do tada poznatima.

### 5.2.1. Opis problema

Neka su $(v,k,\lambda,\mu)$ dopustivi parametri jako regularnog grafa. Neka je $G$ konačna grupa i neka je $(n_1,n_2,\ldots,n_b)$ distribucija duljina orbita vrhova za djelovanje grupe $G$ na jako regularan graf s parametrima $(v,k,\lambda,\mu)$. Nadalje, neka je $M$ retčana orbitna matrica za parametre $(v,k,\lambda,\mu)$ i distribucija duljina orbita vrhova $(n_1,n_2,\ldots,n_b)$. Naš cilj je konstruirati matricu susjedstva jako regularnog grafa $\mathscr{G}$ s parametrima $(v,k,\lambda,\mu)$ koji dopušta djelovanje grupe automorfizama izomorfne grupi $G$ uz distribuciju duljina orbita vrhova $(n_1,n_2,\ldots,n_b)$, tako da to djelovanje producira orbitnu matricu $M$. U ovom radu pretpostavljat ćemo djelovanja cikličkih grupa prostog reda, odnosno grupa izomorfnih sa $\mathbb{Z}_p, p \in \mathbb{P}$. Pri indeksiranju orbitne matrice $M$ koristit ćemo

genetski algoritam, što znači da naše pretraživanje neće biti iscrpno. Drugim riječima, postoji mogućnost da ovim postupkom nećemo konstruirati sve neizomorfne jako regularne grafove koje je moguće konstruirati iz dane orbitne matrice $M$. Za početak ćemo testirati algoritam na primjerima jako regularnih grafova s poznatim parametrima. Slično kao i kod algoritma za konstrukciju blokovnih dizajna, nizom eksperimenata za različite parametre $(v,k,\lambda,\mu)$ poznatih jako regularnih grafova utvrdili smo optimalne vrijednosti parametara algoritma za konstrukciju jako regularnih grafova.

### 5.2.2. Prikaz kandidata za rješenje

Slično kao i u slučaju blokovnih dizajna, pri konstrukciji jako regularnih grafova $\mathrm{SRG}(v,k,\lambda,\mu)$, jedinke će biti matrice susjedstva $k$-regularnih grafova s $v$ vrhova koji dopuštaju djelovanje grupe $G$ s obzirom na retčanu orbitnu matricu $M$. Drugim riječima, jedinke su $(0,1)$-matrice dimenzija $v \times v$ s nulama na glavnoj dijagonali, kojima je suma elemenata u svakom retku i u svakom stupcu jednaka $k$ i koje dopuštaju djelovanje grupe $G$ uz distribuciju duljina orbita vrhova $(n_1,n_2,\ldots,n_b)$ iz kojih se može dobiti orbitna matica $M$. Cilj nam je iz početne populacije koja se sastoji od određenog broja takvih, na slučajan način generiranih jedinki, primjenom genetskog algoritma konstruirati jedinku koja predstavlja matricu susjedstva jako regularnog grafa $\mathrm{SRG}(v,k,\lambda,\mu)$.

Gen u ovakvoj jedinki je unija redaka i stupaca matrice susjedstva koji odgovaraju jednoj orbiti vrhova. Naime, gen ne može biti samo redak koji odgovara određenoj orbiti vrhova jer matrica susjedstva mora biti simetrična.

Bit je podmatrica koja predstavlja presjek redaka i stupaca matrice susjedstva koji odgovaraju orbitama dvaju vrhova. Bitovi, dakle, odgovaraju elementima orbitne matrice. Bit u matrici susjedstva koji se nalazi iznad glavne dijagonale matrice susjedstva dolazi u paru sa svojom simetričnom slikom u odnosu na glavnu dijagonalu matrice susjedstva. Svaki bit iznad glavne dijagonale matrice susjedstva određen je svojim prvim retkom, a ostali retci u bitu na jedinstven su način određeni djelovanjem grupe $G$ na prvi redak. Bilo koja promjena koja se događa u jednom genu ili u jednom bitu matrice susjedstva mora očuvati simetričnost te matrice. Za neke elemente orbitne matrice bitovi su jednoznačno određeni (zvat ćemo ih fiksni bitovi), dok iz nekih drugih elemenata orbitne matrice postoji više mogućnosti za konstrukciju odgovarajućeg bita (zvat ćemo ih nefiksni bitovi). Isto tako, možemo govoriti i o fiksnim i nefiksnim genima.

**Primjer 5.2.1.** Za Petersenov graf, to jest, jako regularan graf s parametrima $\mathrm{SRG}(10,3,0,1)$, za pretpostavljeno djelovanje grupe $\mathbb{Z}_3$ dobije se retčana orbitna matrica s distribucijom duljina orbita vrhova $(1,3,3,3)$

$$\begin{array}{c||c|ccc} & 1 & 3 & 3 & 3 \\ \hline\hline 1 & 0 & 3 & 0 & 0 \\ \hline 3 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 2 \\ 3 & 0 & 1 & 2 & 0 \end{array}.$$

Iz takve orbitne matrice, s distribucijom duljina orbita vrhova $(1,3,3,3)$ uz djelovanje cikličke grupe $\mathbb{Z}_3$, na slučajan način generiramo jedinku. Jedna takva slučajno generirana jedinka je:

$$\left[\begin{array}{c|ccc|ccc|ccc} 0 & 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \textit{1} & \textit{0} & \textit{0} \\ 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \textit{0} & \textit{1} & \textit{0} \\ 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \textit{0} & \textit{0} & \textit{1} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \hline 0 & \textit{1} & \textit{0} & \textit{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 \\ 0 & \textit{0} & \textit{1} & \textit{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & \textit{0} & \textit{0} & \textit{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 \end{array}\right].$$

Ovdje smo jedan gen prikazali podebljano, a jedan bit i njegov simetrični par u odnosu na glavnu dijagonalu prikazali smo ukošeno.

### 5.2.3. Odabir funkcije dobrote

Kao i kod algoritma za konstrukciju blokovnih dizajna, i ovdje smo testirali dvije različite funkcije dobrote koje smo smatrali pogodnima za ovaj problem.

Definirajmo prvu moguću funkciju dobrote. Za svaka dva različita vrha $v_i$ i $v_j$ neka je $x_{i,j}$ broj njihovih zajedničkih susjeda, to jest, skalarni produkt redaka matrice susjedstva koji

odgovaraju tim dvama vrhovima. Funkciju dobrote definiramo kao

$$\sum_{\substack{v_i,v_j\in\mathscr{V}\\ v_i\neq v_j}} \begin{cases} \min\{x_{ij},\lambda\}, & \text{ako su } v_i \text{ i } v_j \text{ susjedni,}\\ \min\{x_{ij},\mu\}, & \text{ako } v_i \text{ i } v_j \text{ nisu susjedni.}\end{cases}$$

Za ovako definiranu funkciju dobrote, jedinka će biti matrica susjedstva jako regularnog grafa $\mathrm{SRG}(v,k,\lambda,\mu)$ ako i samo ako je vrijednost njezine funkcije dobrote jednaka

$$\frac{vk}{2}\lambda+\frac{v(v-k-1)}{2}\mu.$$

Naime, svaki od $v$ vrhova $k$-regularnog grafa ima $k$ susjeda, što znači da imamo ukupno $\frac{vk}{2}$ susjednih parova vrhova. Svaki takav susjedni par vrhova ima $\lambda$ zajedničkih susjeda. S druge strane, svaki od $v$ vrhova $k$-regularnog grafa ima $v-k-1$ nesusjedni vrh, što znači da imamo ukupno $\frac{v(v-k-1)}{2}$ nesusjednih parova vrhova. Svaki takav nesusjedni par vrhova ima $\mu$ zajedničkih susjeda.

Uz ovako definiranu funkciju dobrote, problem pronalaženja optimalnog rješenja, tj. matrice susjedstva jako regularnog grafa je maksimizacijski problem. Vrijednosti dobrota jedinki tijekom generacija trebale bi rasti prema maksimalnoj vrijednosti $\frac{vk}{2}\lambda+\frac{v(v-k-1)}{2}\mu$.

Drugu moguću funkciju dobrote definiramo ovako:

$$\sum_{\substack{v_i,v_j\in\mathscr{V}\\ v_i\neq v_j}} \begin{cases} (x_{ij}-\lambda)^2, & \text{ako su } v_i \text{ i } v_j \text{ susjedni,}\\ (x_{ij}-\mu)^2, & \text{ako } v_i \text{ i } v_j \text{ nisu susjedni.}\end{cases}$$

Za ovako definiranu funkciju dobrote, očito je da će jedinka biti matrica susjedstva jako regularnog grafa $\mathrm{SRG}(v,k,\lambda,\mu)$ ako i samo ako je vrijednost njezine funkcije dobrote jednaka nuli.

Uz ovako definiranu funkciju dobrote, problem pronalaženja optimalnog rješenja, tj. matrice susjedstva jako regularnog grafa je minimizacijski problem. Vrijednosti dobrota jedinki tijekom generacija trebale bi padati prema minimalnoj vrijednosti 0.

Nakon eksperimentalne usporedbe ovih dvaju funkcija, odlučili smo koristiti prvu opisanu funkciju dobrote.

### 5.2.4. Križanje

Križanje se i u slučaju jedinki u populaciji nad kojom radi algoritam za konstrukciju jako regularnih grafova definira na način da se zamjenjuju dva ili više gena na odgovarajućim pozicijama dvaju jedinki.

**Primjer 5.2.2.** Na primjeru parametara Petersenovog grafa (primjer 5.2.1) mogli bismo, na primjer, križati jedan gen između dvije jedinke na sljedeći način:

$$\left[\begin{array}{c|ccc|ccc|ccc} 0 & 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 1 \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \hline 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 \end{array}\right]$$

roditelj A

$$\left[\begin{array}{c|ccc|ccc|ccc} 0 & 1 & 1 & 1 & \mathit{0} & \mathit{0} & \mathit{0} & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & \mathit{1} & \mathit{0} & \mathit{0} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & \mathit{0} & \mathit{1} & \mathit{0} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \mathit{0} & \mathit{0} & \mathit{1} & 1 & 0 & 0 \\ \hline \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} \\ \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{1} & \mathit{0} \\ \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{1} \\ \hline 0 & 0 & 0 & 1 & \mathit{1} & \mathit{1} & \mathit{0} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \mathit{0} & \mathit{1} & \mathit{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \mathit{1} & \mathit{0} & \mathit{1} & 0 & 0 & 0 \end{array}\right]$$

roditelj B

$$\left[\begin{array}{c|ccc|ccc|ccc} 0 & 1 & 1 & 1 & \mathit{0} & \mathit{0} & \mathit{0} & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & \mathit{0} & \mathit{1} & \mathit{0} & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & \mathit{0} & \mathit{1} & \mathit{0} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & \mathit{0} & \mathit{0} & \mathit{1} & 0 & 0 & 1 \\ \hline \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} \\ \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{1} & \mathit{0} \\ \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & \mathit{1} \\ \hline 0 & 1 & 0 & 0 & \mathit{1} & \mathit{1} & \mathit{0} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \mathit{0} & \mathit{1} & \mathit{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \mathit{1} & \mathit{0} & \mathit{1} & 0 & 0 & 0 \end{array}\right]$$

dijete A

$$\left[\begin{array}{c|ccc|ccc|ccc} 0 & 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 1 & 0 & 0 \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \hline 0 & 0 & 0 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 \end{array}\right]$$

dijete B

### 5.2.5. Mutacija

Mutaciju provodimo na način da se jedan ili više bitova unutar jedinke zamijeni s novim, slučajno generiranim bitovima. To zapravo znači da na slučajan način permutiramo prvi redak bita, a ostali retci u bitu određeni su djelovanjem pretpostavljene grupe automorfizama. Pri tome, mutacija svakog bita uzrokuje i odgovarajuću mutaciju bita koji je njemu simetričan u

odnosu na glavnu dijagonalu matrice susjedstva, budući da matrica i nakon mutacije mora biti simetrična.

**Primjer 5.2.3.** Mutiramo jedan bit jedinke iz primjera 5.2.1 i bit koji je njemu simetričan u odnosu na glavnu dijagonalu matrice susjedstva:

$$\left[\begin{array}{c|ccc|ccc|ccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 1 \\ \hline
0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0
\end{array}\right]
\longrightarrow
\left[\begin{array}{c|ccc|ccc|ccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 \\ \hline
0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0
\end{array}\right]$$

Mutacija bita određena je mutacijom njegovog prvog retka. Mutacija zapravo znači preraspodjelu pozicija nula i jedinica u prvom retku te podmatrice, dok njihov broj ostaje očuvan. Ostali retci u bitu na jedinstven su način određeni djelovanjem grupe $\mathbb{Z}_3$, odnosno oni su uzastopni ciklički pomaci tog prvog retka.

### 5.2.6. Selekcija

Kao i kod konstrukcije blokovnih dizajna, u algoritmu za konstrukciju jako regularnih grafova također smo pokušali koristiti 4-turnirsku, jednostavnu i eliminacijsku selekciju.

# 5.3. TESTNI PRIMJERI I OPTIMIZACIJA ALGORITMA

## 5.3.1. Testni primjeri

Kao primjere za testiranje rada algoritma koristili smo orbitne matrice jako regularnih grafova koje navodimo u tablici 5.1. Matrice susjedstva tih grafova mogu se pronaći na [58]. Za svaki primjer jako regularnog grafa odredili smo optimalnu vrijednost funkcije dobrote FitnessForSRG u odnosu na prvu funkciju dobrote definiranu u 5.2.3 (za drugu funkciju optimalna vrijednost je uvijek 0). Odredili smo pune grupe automorfizama $\mathrm{Aut}(\mathscr{G})$ za svaki od tih jako regularnih grafova te moguće grupe $G$ prostog reda izomorfne podgrupama od $\mathrm{Aut}(\mathscr{G})$ koje djeluju na taj jako regularan graf. Za svaku takvu grupu određena je orbitna matrica za djelovanje po jedne podgrupe od $\mathrm{Aut}(\mathscr{G})$ izomorfne s $G$ na jako regularan graf. Na temelju parametara jako regularnog grafa i dobivene orbitne matrice, pokušali smo pomoću algoritma konstruirati jako regularan graf s tim parametrima.

U tablici 5.2 navodimo prosječan broj iteracija (zaokružen na najbliži cijeli broj) koje vode do jako regularnog grafa (#iteracija) i broj dobivenih rješenja unutar 100 potpunih resetiranja (#rješenja) za pretpostavljeno djelovanje grupe $\mathbb{Z}_2$ na one jako regularne grafove iz tablice 5.1 koji to dozvoljavaju. Kada algoritam pronađe dizajn sa željenim parametrima, dozvolili smo da traži i nova rješenja sve dok se ne postigne ukupno $\mathrm{MaxNrOfCompleteResets} = 100$ potpunih resetiranja. Dobiveni broj rješenja unutar tih 100 resetiranja može biti i veći od 100 budući da se u istoj iteraciji algoritma mogu pojaviti dvije ili više jedinki s funkcijom dobrote FitnessForSRG.

| parametri | FitnessForSRG | $G$ |
|---|---|---|
| $(10,3,0,1)$ | 30 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(10,6,3,4)$ | 30 | $\mathbb{Z}_3$ |
| $(16,5,0,2)$ | 160 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(16,6,2,2)$ | 240 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| $(17,8,3,4)$ | 476 | $\mathbb{Z}_2$ |
| $(21,10,3,6)$ | 945 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| $(25,8,3,2)$ | 700 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(25,12,5,6)$ | 1 650 | $\mathbb{Z}_2$ |
| $(26,10,3,4)$ | 1 170 | $\mathbb{Z}_2$ |
| $(27,10,1,5)$ | 1 215 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(28,12,6,4)$ | 1 848 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(29,14,6,7)$ | 2 639 | $\mathbb{Z}_3$ |
| $(35,18,9,9)$ | 5 355 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| $(36,10,4,2)$ | 1 620 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(36,14,4,6)$ | 3 276 | $\mathbb{Z}_2$ |
| $(36,14,7,4)$ | 3 276 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| $(36,15,6,6)$ | 3 780 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |
| $(37,18,8,9)$ | 5 661 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| $(40,12,2,4)$ | 2 640 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| $(45,12,3,3)$ | 2 970 | $\mathbb{Z}_2, \mathbb{Z}_3$ |
| $(49,12,5,2)$ | 3 234 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_7$ |
| $(50,21,8,9)$ | 10 500 | $\mathbb{Z}_2, \mathbb{Z}_7$ |
| $(64,18,2,6)$ | 9 792 | $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ |

Tablica 5.1: Testni primjeri jako regularnih grafova.

| parametri | FitnessForSRG | #iteracija | #rješenja |
|---|---|---|---|
| $(10,3,0,1)$ | 30 | 7 | 116 |
| $(16,5,0,2)$ | 160 | 15 | 98 |
| $(16,6,2,2)$ | 240 | 23 | 94 |
| $(17,8,3,4)$ | 476 | 56 | 81 |
| $(21,10,3,6)$ | 945 | 291 | 49 |
| $(25,8,3,2)$ | 700 | 212 | 64 |
| $(25,12,5,6)$ | 1 650 | 412 | 47 |
| $(26,10,3,4)$ | 1 170 | 310 | 51 |
| $(27,10,1,5)$ | 1 215 | 375 | 43 |
| $(28,12,6,4)$ | 1 848 | 455 | 40 |
| $(35,18,9,9)$ | 5 355 | 801 | 19 |
| $(36,10,4,2)$ | 1 620 | 410 | 41 |
| $(36,14,4,6)$ | 3 276 | 733 | 28 |
| $(36,14,7,4)$ | 3 276 | 745 | 26 |
| $(36,15,6,6)$ | 3 780 | 756 | 23 |
| $(37,18,8,9)$ | 5 661 | 812 | 20 |
| $(40,12,2,4)$ | 2 640 | 688 | 32 |
| $(45,12,3,3)$ | 2 970 | 706 | 29 |
| $(49,12,5,2)$ | 3 234 | 739 | 23 |
| $(50,21,8,9)$ | 10 500 | 1 238 | 15 |
| $(64,18,2,6)$ | 9 792 | 987 | 19 |

Tablica 5.2: Prikaz broja potrebnih iteracija i broja dobivenih rješenja za djelovanje grupe $\mathbb{Z}_2$.

### 5.3.2. Optimizacija parametara

Kao i u poglavlju 4, proveli smo različite serije eksperimenata koji se tiču optimizacije ulaznih parametara grafova, odabira funkcije dobrote i selekcije. Na temelju dobivenih eksperimenata zaključili smo da ćemo koristiti prvu opisanu funkciju dobrote u 5.2.3, 4-turnirsku selekciju te ostale parametre kao i u slučaju konstrukcije blokovnih dizajna.

U seriji napravljenih eksperimenata, kao zadane vrijednosti parametara koristimo:

- veličinu populacije POP $= 100$,
- MaxNrOfGenerations $= 100\,000$,
- vjerojatnost mutacije $p_m = 100\%$,
- vjerojatnost križanja $p_c = 100\%$,
- broj gena koji sudjeluju u križanju NrGenesForCrossover $= 1$,
- broj bitova koji se mutiraju NrBitsForMutation $= 1$,
- FitnessForSRGNrOfRepeatsMax $= 100$,
- MaxNrOfPartialResets $= 10$,
- MaxNrOfCompleteResets $= 100$,
- StartingPercentage $= 10\%$.

# 5.4. KONSTRUKCIJA NOVIH SRG$(96,19,2,4)$ I SRG$(96,20,4,4)$

Poznati jako regularni grafovi s 96 vrhova s parametrima SRG$(96,19,2,4)$ i SRG$(96,20,4,4)$ konstruirani su u [7], [9] i [50]. U [61] može se pronaći ukupno 63 takva grafa, od kojih je 11 s parametrima SRG$(96,19,2,4)$ (tablica 5.3) i 52 s parametrima SRG$(96,20,4,4)$ (tablica 5.4). Redovi punih grupa automorfizama jako regularnih grafova SRG$(96,19,2,4)$ su u rasponu od 96 do 9 216, a jako regularnih grafova SRG$(96,20,4,4)$ od 96 do 138 240. F. Ihringer je u [34] nedavno objavio mnoge nove SRG$(96,19,2,4)$ i SRG$(96,20,4,4)$ koji uglavnom imaju grupe automorfizama malih redova (uključujući i trivijalne).

| $\lvert\mathrm{Aut}(\mathscr{G})\rvert$ | #neizomorfnih grafova |
|---|---|
| 9 216 | 1 |
| 1 536 | 1 |
| 768 | 3 |
| 288 | 1 |
| 96 | 5 |

Tablica 5.3: 11 poznatih SRG$(96,19,2,4)$.

| $\lvert\mathrm{Aut}(\mathscr{G})\rvert$ | #neizomorfnih grafova |
|---|---|
| 138 240 | 1 |
| 11 520 | 2 |
| 7 680 | 1 |
| 3 072 | 2 |
| 1 536 | 2 |
| 768 | 12 |
| 576 | 1 |
| 384 | 4 |
| 192 | 6 |
| 96 | 21 |

Tablica 5.4: 52 poznata SRG$(96,20,4,4)$.

Opisanim genetskim algoritmom i korištenjem orbitnih matrica poznatih jako regularnih grafova SRG$(96,19,2,4)$ uz pretpostavljeno djelovanje involucije, konstuirali smo nove jako regularne grafove s tim parametrima. Kao i u slučaju blokovnih dizajna, algoritam je zaustavljen kada nakon 48 sati izvođenja nakon zadnjeg pronađenog novog jako regularnog grafa nije više pronađen niti jedan drugi novi jako regularni graf. Konstruirali smo 21 SRG$(96,19,2,4)$ od kojih niti jedan nije izomorfan s grafovima iz tablice 5.3. Korištenjem programa GAP [60] provjerili smo jesu li ti grafovi izomorfni s nekim od jako regularnih grafova SRG$(96,19,2,4)$ koje je konstruirao F. Ihringer u [34] te smo zaključili da njih 15 nije izomorfno niti s jednim, a 6 je izomorfno s nekima od njih. Rezultate prikazujemo u tablici 5.5.

| $\lvert\mathrm{Aut}(\mathscr{G})\rvert$ | struktura $\mathrm{Aut}(\mathscr{G})$ | #neizomorfnih grafova |
|---|---|---|
| 8 | $E_8$ | 2 |
| 4 | $\mathbb{Z}_4$ | 1 |
| 4 | $E_4$ | 6 |
| 2 | $\mathbb{Z}_2$ | 12 |

Tablica 5.5: SRG$(96,19,2,4)$

Opisanim genetskim algoritmom i korištenjem orbitnih matrica poznatih jako regularnih grafova SRG$(96,20,4,4)$ uz pretpostavljeno djelovanje involucije, konstuirali smo nove jako regularne grafove s tim parametrima. Algoritam je zaustavljen kada nakon 48 sati izvođenja nakon zadnjeg pronađenog novog jako regularnog grafa nije više pronađen niti jedan drugi novi jako regularni graf. Konstruirali smo 22 SRG$(96,20,4,4)$ koji nisu izomorfni s grafovima iz tablice 5.4. Korištenjem programa GAP [60] provjerili smo jesu li ti grafovi izomorfni s nekima od jako regularnih grafova SRG$(96,20,4,4)$ u [34] te smo zaključili da niti jedan od njih nije izomorfan niti kojem od njih. Rezultate prikazujemo u tablici 5.6.

| $\lvert\mathrm{Aut}(\mathscr{G})\rvert$ | struktura $\mathrm{Aut}(\mathscr{G})$ | #neizomorfnih grafova |
|---|---|---|
| 16 | $\mathbb{Z}_2 \times D_8$ | 1 |
| 8 | $E_8$ | 3 |
| 4 | $E_4$ | 8 |
| 2 | $\mathbb{Z}_2$ | 10 |

Tablica 5.6: SRG$(96,20,4,4)$

U tablici 5.7 prikazujemo podatke o ukupnom broju (#rješenja) dobivenih jako regularnih grafova SRG$(96,19,2,4)$ i SRG$(96,20,4,4)$ korištenjem opisanog algoritma, minimalnom (#min), maksimalnom (#max) i prosječnom broju potrebnih iteracija (#avg) te minimalnom ($t_{\min}$), maksimalnom ($t_{\max}$) i prosječnom vremenu ($t_{\text{avg}}$) dobivanja rješenja (u minutama). Računalo na kojem se izvodio algoritam ima sljedeće specifikacije: procesor 11th Gen Intel(R) Core(TM) i7-1165G7 2.60GHz, 16GB RAM memorije frekvencije 2803MHz.

| parametri | FitnessForSRG | #rješenja | #min | #max | #avg | $t_{\min}$ | $t_{\max}$ | $t_{\text{avg}}$ |
|---|---|---|---|---|---|---|---|---|
| $(96,19,2,4)$ | 16 416 | 1 109 | 1 797 | 5 069 | 2 912 | 21.12 | 59.59 | 34.23 |
| $(96,19,2,4)$ | 18 240 | 1 232 | 1 931 | 11 311 | 3 613 | 22.60 | 132.36 | 42.28 |

Tablica 5.7: Rezultati algoritma za SRG$(96,19,2,4)$ i SRG$(96,20,4,4)$.

Grafovi koje smo konstuirali su dostupni na poveznicama:

`https://github.com/TinZrinski/structures/blob/main/96_19_2_4.txt`,

`https://github.com/TinZrinski/structures/blob/main/96_20_4_4.txt`.

# ZAKLJUČAK

Predmet istraživanja ove doktorske disertacije je konstrukcija blokovnih dizajna i jako regularnih grafova uz pretpostavku djelovanja određene grupe automorfizama korištenjem modificiranih genetskih algoritama.

I. Martinjak i M.-O. Pavčević su u [45] koristili modificirani genetski algoritam u svrhu nepotpune potrage za blokovnim dizajnima bez dodatne pretpostavke djelovanja grupe automorfizama. U ovoj disertaciji uvođenjem pretpostavke djelovanja grupe automorfizama uz korištenje genetskog algoritma omogućuje se konstrukcija blokovnih dizajna s većim brojem točaka i blokova, a također i jako regularnih grafova s relativno velikim brojem vrhova.

U poglavljima 4 i 5 razvijeni su algoritmi i pripadni računalni programi za nepotpunu pretragu modificiranim genetskim algoritmima koji služe za konstrukciju blokovnih dizajna i jako regularnih grafova iz orbitnih matrica dobivenih za pretpostavljeno djelovanje grupe automorfizama. Na ovaj način može se izvršiti nepotpuna potraga za blokovnim dizajnima i jako regularnim grafovima na temelju orbitnih matrica u slučajevima kada to nije moguće izvršiti iscrpnom pretragom.

Upotrebom tih algoritama, odnosno računalnih programa, konstruirali smo nove blokovne dizajne, točnije nove Steinerove sustave s parametrima $S(2,5,45)$ i nove simetrične dizajne s parametrima $(71,15,3)$ te nove jako regularne grafove s parametrima $(96,19,2,4)$ i $(96,20,4,4)$.

# BIBLIOGRAFIJA


[1] Baricelli, N. A.: *Numerical testing of evolution theories*. ACTA Biotheoretica, 16:143–182, 1957. ↑ 31.

[2] Baricelli, N. A.: *Symbiogenetic evolution processes realized by artificial methods*. Methodos, 9, no. 35-36:143–182, 1957. ↑ 31.

[3] Behbahani, M.: *On strongly regular graphs*. Disertacija, Concordia University, 2009. ↑ 2, 24, 90.

[4] Behbahani, M. i Lam, C.: *Strongly regular graphs with non-trivial automorphisms*. Discrete Mathematics, 311:132–144, 2011. ↑ 2, 24, 90.

[5] Beker, H. J., Mitchell, C. J. i Piper, F. C.: *Tactical decompositions of designs*. Aequationes Mathematicae, 25:123–152, 1982. ↑ 18.

[6] Box, G. E. P.: *Evolutionary operation: A method for increasing industrial productivity*. Journal of the Royal Statistical Society, C 6, no. 2:80–101, 1957. ↑ 31.

[7] Braić, S., Golemac, A., Mandić, J. i Vučičić, T.: *Graphs and symmetric designs corresponding to difference sets in groups of order 96*. Glasnik Matematički, 45:1–14, 2010. ↑ 3, 102.

[8] Bremermann, H. J.: *Optimization through evolution and recombination*. Proceedings of the Conference on Self-Organizing Systems, stranice 93–106, 1962. ↑ 31.

[9] Brouwer, A. E., Koolen, J. H. i Klin, M.H.: *A root graph that is locally the line graph of the Petersen graph*. Discrete Mathematics, 264:13–24, 2003. ↑ 3, 102.

[10] Brouwer, A. E. i Van Maldeghem, H.: *Strongly regular graphs*. Cambridge University Press, 2022. ↑ 24.

[11] Cameron, P. J.: *Permutation Groups*. Cambridge University Press, 1999. ↑ 5.

[12] Colbourn, C. i Steiner, A.: *A Steiner 2-design with an automorphism fixing exactly $r+2$ points*. Journal of Combinatorial Designs, 7:375–380, 1999. ↑ 2, 85.

[13] Corneil, D. G. i Mathon, R.: *Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations*. Annals of Discrete Mathematics, 2:1–32, 1978. ↑ 2.

[14] Crnković, D.: *Konstrukcije nekih novih simetričnih nacrta kvadratnog reda*. Disertacija, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Zagreb, 1998.

[15] Crnković, D. i Dumičić Danilović, D.: *Finding unitals in symmetric designs using a modified genetic algorithm*. Mathematica Pannonica, 24:183–196, 2013. ↑ 2.

[16] Crnković, D., Dumičić Danilović, D., Rukavina, S. i Šimac, M.: *On some new Steiner 2-designs $S(2,5,45)$*. Utilitas Mathematica, 111:281–308, 2019. ↑ 2, 85.

[17] Crnković, D. i Maksimović, M.: *Construction of strongly regular graphs having an automorphism group of composite order*. Contributions to Discrete Mathematics, 15:22–41, 2020. ↑ 3, 91, 92.

[18] Crnković, D. i Pavčević, M.-O.: *Some new symmetric designs with parameters (64,28,12)*. Discrete Mathematics, 237:109–118, 2001. ↑ 2, 53.

[19] Crnković, D. i Rukavina, S.: *New symmetric (71,15,3) designs*. Bulletin of the Institute of Combinatorics and its Applications, 94:79–94, 2022. ↑ 2, 87.

[20] Crnković, D., Rukavina, S. i Simčić, L.: *On triplanes of order twelve admitting an automorphism of order six and their binary and ternary codes*. Utilitas Mathematica, 103:23–40, 2017. ↑ 2, 87.

[21] Crnković, D. i Zrinski, T.: *Constructing block designs with a prescribed automorphism group using genetic algorithm*. Journal of Combinatorial Designs, 30:515–526, 2022.

[22] Crnković, D. i Švob, A.: *New symmetric 2-(176,50,14) designs*. Discrete Mathematics, 344:112623, 2021. ↑ 2, 53.

[23] Crnković, D., Švob, A. i Tonchev, V. D.: *Strongly regular graphs with parameters (81,30,9,12) and a new partial geometry*. Journal of Algebraic Combinatorics, 53:253–261, 2021. ↑ 3, 91.

[24] Dembowski, P.: *Verallgemeinerungen von Transitivitätsklassen endlicher projektiver Ebenen*. Mathematische Zeitschrift, 69:59–89, 1958. ↑ 18.

[25] Diestel, R.: *Graph Theory, Fourth edition*. Springer-Verlag, 2010. ↑ 22.

[26] Dumičić Danilović, D.: *Poopćenje i profinjenje nekih algoritama za konstrukciju blokovnih dizajna i istraživanje njihovih podstruktura*. Disertacija, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Zagreb, 2014.

[27] Fogel, L. J., Owens, A. J. i Walsh, M. J.: *Artificial Intelligence through Simulated Evolution*. Wiley, 1966. ↑ 31.

[28] Fraser, A. S.: *Simulation of genetic systems by automatic digital computers: I. Introduction*. Australian Journal of Biological Science, 10:484–491, 1957. ↑ 31.

[29] Fraser, A. S.: *Simulation of genetic systems by automatic digital computers: II. Effects of linkage on rates of advance under selection*. Australian Journal of Biological Science, 10:492–499, 1957. ↑ 31.

[30] Friedman, G. J.: *Digital simulation of an evolutionary process*. General Systems Yearbook, 4:171–184, 1959. ↑ 31.

[31] Galinier, P. i Jaumard, B.: *On triplanes of order twelve admitting an automorphism of order six and their binary and ternary codes*. Computers and Operations Research., 33:955–970, 2006. ↑ 2, 54.

[32] Gibbons, P. B. i Östergård, P. R. J.: *Computational methods in design theory*. Handbook of Combinatorial Designs (C. J. Colbourn and J. H. Dinitz, eds.), 2nd ed. Chapman & Hall/CRC, Boca Raton, FL, 2007. ↑ 2, 54.

[33] Holland, J. H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975. ↑ 31.

[34] Ihringer, F.: *Switching for small strongly regular graphs*. Australasian Journal of Combinatorics, 84(1):28–48, 2022. ↑ 3, 102, 103, 104.

[35] Janko, Z.: *Coset enumeration in groups and constructions of symmetric designs, Combinatorics '90 (Gaeta,1990)*. Annals of Discrete Mathematics, 52:275–277, 1992. ↑ 2, 53, 55.

[36] Janko, Z.: *The existence of symmetric designs with parameters (189, 48, 12)*. Journal of Combinatorial Theory, Series A, 80:334–338, 1997. ↑ 2, 53.

[37] Janko, Z.: *The existence of symmetric designs with parameters (105, 40, 15)*. Journal of Combinatorial Designs, 7:17–19, 1999. ↑ 2, 53.

[38] Janko, Z. i Van Trung, T.: *Construction of a new symmetric block design for (78,22,6) with the help of tactical decompositions*. Journal of Combinatorial Theory, Series A, 40:451–455, 1985. ↑ 2, 53.

[39] Koubi, S., Mata-Montero, M. i Shalaby, N.: *Using directed hill-climbing for the construction of difference triangle sets*. IEEE Transactions on Information Theory, 51:335–339, 2005. ↑ 2, 54.

[40] Krčadinac, V.: *Steiner 2-designs*. Pristupljeno 20. rujna 2022. `https://web.math.pmf.unizg.hr/~krcko/results/steiner.html`. ↑ 2, 85.

[41] Krčadinac, V.: *Construction and classification of finite structures by computer*. Disertacija, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Zagreb, 2004. ↑ 2, 85.

[42] Lang, S.: *Undergraduate algebra*. Springer, 2005. ↑ 5.

[43] Maksimović, M.: *Orbitne matrice jako regularnih grafova*. Disertacija, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Zagreb, 2015. ↑ 3, 91, 92.

[44] Martin, F. G. i Cockerham, C. C.: *Artificial Intelligence through Simulated Evolution*. Biometrical Genetics, stranice 35–45, 1960. ↑ 31.

[45] Martinjak, I. i Pavčević, M.-O.: *Modified Genetic Algorithm for BIBD Construction*. Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces, stranice 647–652, 2009. ↑ 2, 3, 44, 48, 54, 58, 105.

[46] Mathon, R.: *Computational methods in design theory*. Computational and Constructive Design Theory, Mathematics and Its Applications (W. D. Wallis, ed.). Kluwer Academic Publishers, 1996. ↑ 2, 54.

[47] Mathon, R. i Rosa, A.: *Some results on the existence and enumeration of BIBDs*. Tech. Report Math., 125-Dec-1985. McMaster University, Hamilton ON, 1985. ↑ 2, 85.

[48] Mezőfi, D. i Nagy, G. P.: *New Steiner 2-designs from old ones by paramodifications*. Utilitas Mathematica, 288:114–122, 2021. ↑ 2, 85.

[49] Mitchell, M.: *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[50] Muzychuk, M.: *A generalization of Wallis-Fon-Der-Flaass construction of strongly regular graphs*. Journal of Algebraic Combinatorics, 25:169–187, 2007. ↑ 3, 102.

[51] Rechenberg, I.: *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, 1965. ↑ 31.

[52] Rechenberg, I.: *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann - Holzboog, 1973. ↑ 31.

[53] Reed, J., Toombs, R. i Barricelli, N. A.: *Simulation of biological evolution and machine learning*. Journal of Theoretical Biology, 17:319–342, 1967. ↑ 31.

[54] Riolo, R.: *Survival of the fittest bits*. Scientific American, 267:114–116, 1. ↑ 39.

[55] Rukavina, S.: *2-(71,15,3) designs*. Pristupljeno 20. rujna 2022. https://www.math.uniri.hr/~sanjar/structures/. ↑ 87.

[56] Schwefel, H. P.: *Evolutionsstrategie und numerische Optimierung*. Disertacija, Tehničko sveučilište u Berlinu, Berlin, 1975. ↑ 31.

[57] Schwefel, H. P.: *Numerische Optimierung von Computer - Modellen mittels der Evolutionsstrategie*. Birkhäuser, 1977. ↑ 31.

[58] Spence, E.: *Strongly Regular Graphs on at most 64 vertices*. Pristupljeno 20. rujna 2022. http://www.maths.gla.ac.uk/~es/srgraphs.php. ↑ 98.

[59] Stinson, D. R.: *Combinatorial Designs with Selected Applications*. Springer-Verlag, 2004. ↑ 14.

[60] The GAP Group: *GAP - Groups, Algorithms, and Programming, Version 4.11.1*, 2021. https://www.gap-system.org. ↑ 46, 62, 103, 104.

[61] Vučičić, T.: *DifSets96*. Pristupljeno 20. rujna 2022. https://mapmf.pmfst.unist.hr/~vucicic/DifSets96/. ↑ 102.

[62] Wallis, W. D.: *Computational and Constructive Design Theory*. Kluwer Academic Publishers, 1996. ↑ 44.

[63] Wright, S.: *Evolution in Mendelian populations*. Genetics, 16:97–159, 1931. ↑ 37.

# Životopis

Tin Zrinski rođen je 10. siječnja 1993. godine u Rijeci gdje je završio osnovnu i srednju školu. Diplomirao je na Diplomskom studiju Diskretna matematika i primjene Odjela za matematiku Sveučilišta u Rijeci u lipnju 2016. godine. Iste godine upisao je Zajednički sveučilišni poslijediplomski doktorski studij matematike na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu. Zaposlen je kao asistent na Fakultetu za matematiku Sveučilišta u Rijeci. Član je Društva matematičara i fizičara Rijeka, Alumni kluba Fakulteta za matematiku te Seminara za konačnu matematiku u Rijeci, na kojem je održao niz seminara. Koautor je znanstvenog članka: Crnković, D. i Zrinski, T.: *Constructing block designs with a prescribed automorphism group using genetic algorithm.* Journal of combinatorial designs, 30: 515–526, 2022.