SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 3617

# OPTIMIZACIJA POVEĆANJA PROPUSNOSTI KOMBINACIJSKIH MREŽA EVOLUCIJSKIM ALGORITMIMA

Dominik Šišejković

Zagreb, lipanj 2014.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 12. ožujka 2014.

# ZAVRŠNI ZADATAK br. 3617

Pristupnik **Dominik Šišejković**
Studij: Računarstvo
Modul: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Optimizacija povećavanja propusnosti kombinacijskih mreža evolucijskim algoritmima**

Opis zadatka:

Opisati postupak povećavanja propusnosti kombinacijskih mreža uporabom memorijskih elemenata. Definirati postupak povećavanja propusnosti kao kombinatorički optimizacijski problem. Ostvariti programski sustav za simulaciju propusnosti kombinatoričkih mreža. Ispitati uspješnost stohastičkih optimizacijskih algoritama na problemu povećavanja propusnosti. Posebnu pažnju posvetiti učinkovitosti optimizacije s obzirom na veličinu kombinacijske mreže. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Izv.prof.dr.sc. Domagoj Jakobović

Djelovođa:

Doc.dr.sc. Ivica Botički

Predsjednik odbora za završni rad modula:

Prof.dr.sc. Krešimir Fertalj

*Zahvaljujem mentoru prof. dr. sc. Domagoju Jakoboviću na strpljenju, razumijevanju, stalnoj potpori i izuzetnom vođenju prilikom izrade ovog rada.*

*Također hvala asistentu dipl. ing. el. Stjepanu Piceku na sugestijama, idejama i stalnoj suradnji.*

*Posebno hvala mojoj obitelji na bezuvjetnoj pomoći, odricanju i razumijevanju. Hvala vam što ste mi omogućili da ostvarim svoje ciljeve.*

# Sadržaj

# Uvod

Razvojem računala i računarske znanosti otvaraju se nove mogućnosti rješavanja problema razvojem algoritamskih postupaka primjenjivih na računalu. S obzirom na veliku brzinu rada današnjih računala, moguće je njihovom uporabom u jednoj sekundi istražiti milijune kombinacija potencijalnih rješenja. Ova činjenica je izrazito korisna ukoliko se rješava problem koji se može riješiti grubom silom, odnosno algoritmima iscrpne pretrage prostora rješenja. Nažalost, postoji čitav niz problema koji ne spadaju u tu kategoriju te je za njihovo rješavanje potrebno koristiti drugačije postupke.

Postavlja se pitanje kako pristupiti rješavanju izrazito kompleksnih problema koje ne možemo riješiti današnjom računalnom snagom u zadovoljavajuće kratkom konačnom vremenu? Jedan od odgovor krije se ideji primjene prirodom razvijenih algoritama evolucije.

Priroda je oduvijek predstavljala glavni izvor inspiracije čovjekovim postupcima pa tako i prilikom modeliranja generičkih algoritama rješavanja bilo kakvog problema optimizacije. Takav pristup nije bezrazložan. Prirodni procesi optimiraju sve životne segmente već preko 4 milijarde godina koristeći zapravo vrlo jednostavne, ali idejno snažne generičke algoritme. Preslikavanjem prirodom kovanih algoritama u računalno primjenjive algoritme pokazalo se kao izrazito uspješan način rješavanja kompleksnih problema.

U ovom završnom radu, nastoji se primjenom prirodom inspiriranih genetskih algoritama povećati propusnost kombinacijskih mreža raspodjelom dodatnih memorijskih elemenata na samom sklopu. Ideja povećanja propusnosti temelji se na povećanju brzine propagacije signala kroz strukturu sklopa ostvarenjem protočnog načina rada kombinacijske mreže što predstavlja svojevrsnu paralelizaciju sklopa. U okviru ovog rada često će biti spomenut termin mreže te se pri tome semantički referencira isključivo na kombinacijske mreže.

U prvom poglavlju ovog rada opisana je osnovna ideja evolucijskih algoritama sagledavajući pri tome kanonsku izvedbu genetskog algoritma te operatore potrebne za evoluciju. Drugo poglavlje posvećeno je sažetom opisu strukture i osnovnih svojstava kombinacijskih mreža. Treće poglavlje sagledava problematiku propusnosti mreža i izvedbu konkretne implementacije evaluatora. U posljednjem poglavlju osvrće se na analizu dobivenih rezultata evaluacije uz predočene statistike i izvedene zaključke za pojedine konfiguracije ispitnog primjera.

# 1. Evolucijski algoritmi

U umjetnoj inteligenciji, evolucijski algoritam (engl. *Evolutionary algorithm, EA*) predstavlja podskup tehnika evolucijskog računarstva, jedne od najbrže razvijajućih grana računarske znanosti. Evolucijski algoritmi spadaju u heurističke metode, odnosno algoritme koji pronalaze rješenja koja su zadovoljavajuće dobra, ali ne nude nikakvu garanciju da će uspjeti pronaći optimalno rješenje.

Osnovna inspiracija evolucijskih algoritama prodire iz procesa biološke evolucije uključujući selekciju, reprodukciju, križanje i mutaciju. Simulirajući suštinski model Darwinove evolucije na računalu, evolucijski algoritmi optimiziraju definiranu funkciju cilja nekog specifičnog problema.

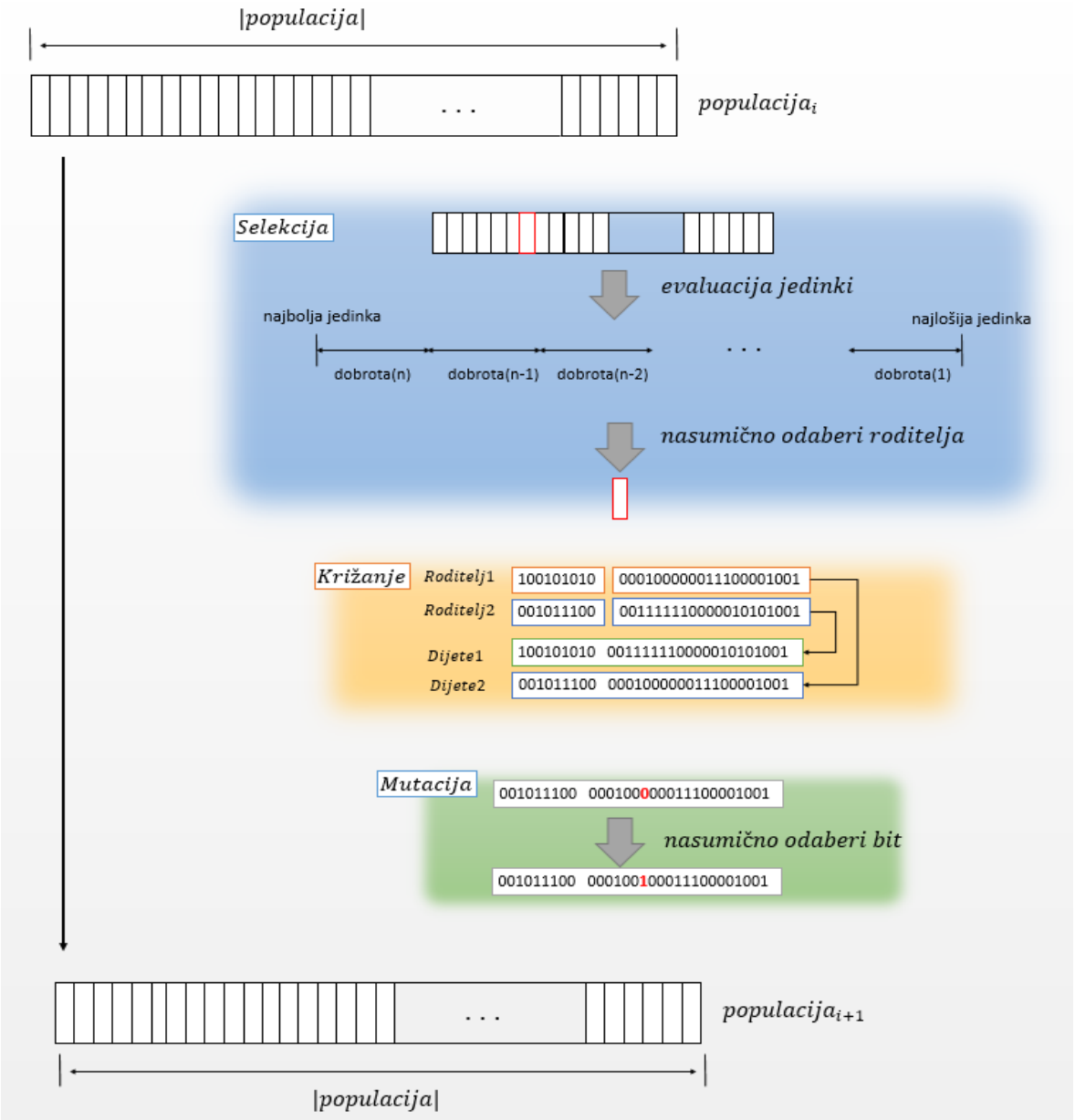
Početni uvjet korištenja evolucijskog algoritma jest preoblikovanje zadanog problema kao problem optimizacije funkcije cilja što predstavlja jednu od ljepota korištenja ovog pristupa jer omogućava neovisnost i općenitost algoritma u odnosu na sam problem.

## 1.1. Genetski algoritam

Svi entiteti u svijetu sadrže svojevrsne gene u sebi kao što i sam čovjek sadrži gene. Geni nekog entiteta sadrže vezu s samim svojstvima i ponašanjem entiteta. U trenutku preoblikovanja proizvoljnog problema u optimizacijski problem nailazimo na pojam jedinke kao reprezentaciju mogućeg rješenja specifičnog problema. Svaka jedinka u sebi sadrži vlastiti genetski materijal koji definira svojstva te jedinke. U okviru optimizacijskog problema, evaluacijom jedinke genetski se materijal preslikava u određenu razinu dobrote rješenja. Upravo u postojanju raznovrsnosti svojstava genetskog skupa neke jedinke leži osnovna ideja genetskog algoritma - prijenos korisnog genetskog materijala iz generacije u generaciju i reprodukciju novog materijala na temelju postojeće genetske osnove uz izvršavanje operacije mutacije.

Sagledavajući internu funkcionalnost, genetski algoritam predstavlja iterativnu proceduru koja se najčešće odrađuje nad populacijom konstantne veličine. Pod pojmom „genetski algoritam“ nije točno određen samo jedan algoritam, već niz algoritama koji se općenito odvijaju na sljedeći način:

Inicijalna populacija jedinki (rješenja, kromosoma) se generira nasumično ili heuristički. Tijekom svake iteracije tzv. generacije, evaluiraju se sve jedinke trenutne populacije, dodjeljujući im određenu vrijednost dobrote. S ciljem formiranja nove generacije, jedinke se odabiru operatorom selekcije te križaju što rezultira stvaranjem novih jedinki koje sadrže po dio genetskog materijala svakog od roditelja. Zbog povećanja raznovrsnosti, svim novonastalim jedinkama se genetska struktura dodatno izmjeni operatorom mutacije. Ovisno o vrsti genetskog algoritma, novonastale se jedinke ubacuju u novu generaciju ili odbacuju sukladno vrijednosti dobrote (*Slika 1*) [1].




**Slika 1.** Kanonska izvedba genetskog algoritma s binarnom reprezentacijom rješenja.

### 1.1.1. Selekcija

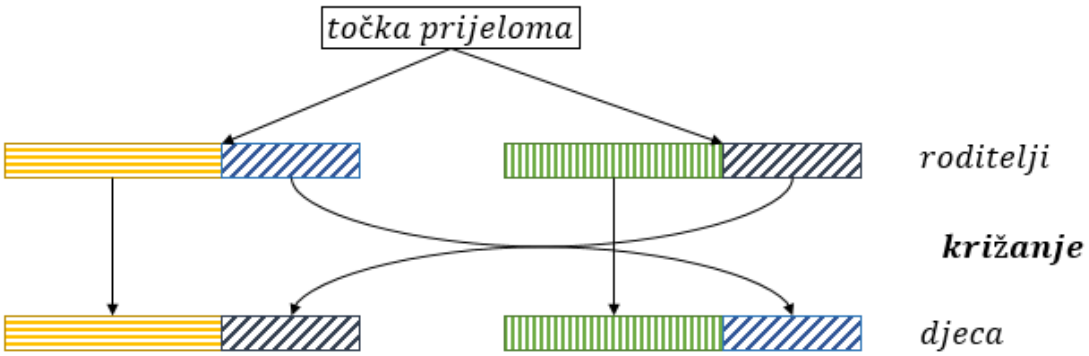
U trenutku generiranja jedne populacije s izračunatim i pridruženim vrijednostima dobrote svake jedinke, kreće postupak odabira jedinki koje će operatorom križanja dati nove jedinke s sličnim genetskim svojstvima. U standardnom genetskom algoritmu (SGA), vjerojatnost odabira neke jedinke za reprodukciju iz trenutne populacije proporcionalna je njezinoj dobroti [1]. U praksi postoje niz načina na koje je moguće odabrati prikladne roditeljske jedinke od kojih su najpopularnije sljedeće:

- proporcionalna selekcija (engl. roulette wheel selection),
- selekcija linearnim rangiranjem,
- turnirska selekcija.

Detalji o pojedinim operatorima selekcije neće biti opisani u ovom radu.

### 1.1.2. Križanje

U kanonskoj izvedbi genetskog algoritma s binarnom reprezentacijom jedinke, operator križanja odabire dvije jedinke posredstvom operatora selekcije te izvršava rez jedinke (kromosoma) na nasumično odabranoj poziciji. Rekombinacijom novonastalih dijelova kromosoma nastaju dvije nove jedinke.




**Slika 2.** Križanje s jednom točkom prijeloma

Općenito postoji niz različitih operatora križanja, od koji je bitno spomenuti sljedeće [4]:

- križanje s jednom točkom prijeloma (*Slika 2*),

- križanje s $t$ – točaka prijeloma,
- uniformno križanje.

Izbor odgovarajućeg operatora križanja iznimno ovisi o reprezentaciji prostora pretraživanja.

### 1.1.3. Mutacija

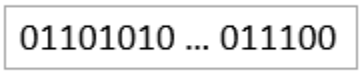
Mutacija omogućava izmjenu genetskog skupa jedinke za određenu vrijednost. U terminima prostora pretraživanja, mutacija označava po smjeru neodređene skokove u nešto drugačija područja prostora pretraživanja.

Operator mutacije u potpunosti ovisi o vrsti reprezentacije jedinke. U okviru jednostavne reprezentacije jedinke bitovnim nizom, mutacija se manifestira promjenom vrijednosti jednog ili više bitova u genotipu rješenja.

### 1.1.4. Problem reprezentacije

Izbor reprezentacije rješenja usko je vezan uz sam optimizacijski problem kojeg se rješava te predstavlja jedan od prvih koraka u rješavanju svakog optimizacijskog problema. Nije moguće svim prikazima predočiti prostor pretraživanja rješenja za bilo koji problem. Odabir reprezentacije može iznimno utjecati na uspješnost rada samog algoritma u odnosu na specifični problem. Prikaz rješenja možemo podijeliti u nekoliko najčešćih [4]:

- prikaz poljem ili vektorom brojeva,
- prikaz nizom bitova,
- prikaz permutacijama i matricama,
- prikaz složenijim i prilagođenim strukturama podataka,
- prikaz stablima.




**Slika 3.** Prikaz nizom bitova.

U okviru implementacije ovog rada korišten je isključivo prikaz nizom bitova (*Slika 3*).

# 2. Kombinacijske mreže

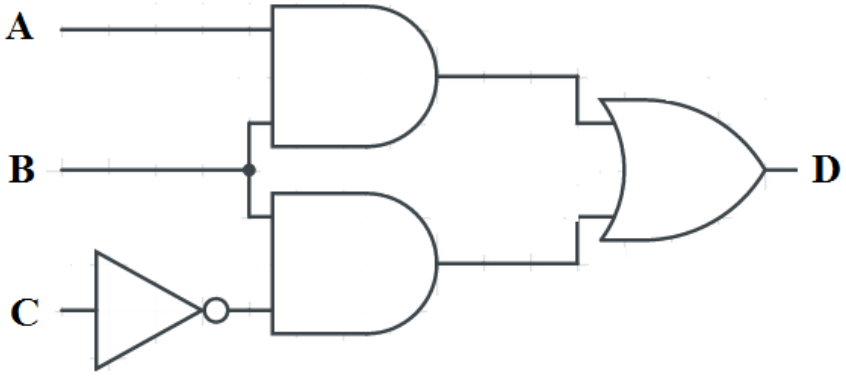
Suvremeni digitalni sustavi u mogućnosti su obrađivati širok spektar različitih podataka. Za čovjeka ti podaci se predstavljaju kao visoko apstraktni modeli podataka kao što su slika, video, zvuk, tekstualna datoteka i slično. Takav apstrahirani prikaz koristan je i razumljiv samo ljudskom poimanju dok računalo na svojoj najnižoj razini bilo koju nama apstraktnu predodžbu podataka obrađuje kao niz digitalnih signala, najčešće predstavljenih simbolima 0 i 1.

S ciljem ostvarenja neke funkcije kao sklopovski računalni sustav koriste se digitalne kombinacijske mreže. Svaka kombinacijska mreža, ovisno o stupnju složenosti, izgrađena je od više povezanih logičkih vrata čiji su izlazi jedinstveno definirani ulaznim skupom signala te je kombinacijom postojećih mreža moguće izgraditi složenije kombinacijske mreže.

Glavna uloga kombinacijskih mreža jest preslikavanje neke Booleove funkcije u konkretni sklop koji svojim sadržajem, rasporedom i načinom povezivanja osnovnih logičkih vrata ostvaruje zadanu funkciju.

Ponašanje kombinacijske mreže je bezmemorijsko. Postavljanjem određenih vrijednosti na ulazu rezultira pojavom signala na izlazu mreže nakon isteka određenog propagacijskog vremena koje je potrebno da se signal propagira od ulaza kroz sva logička vrata na jednom putu do izlaza mreže.

U nastavku je prikazan jednostavan primjer kombinacijske mreže sastavljen od osnovnih logičkih vrata I, ILI i NE:



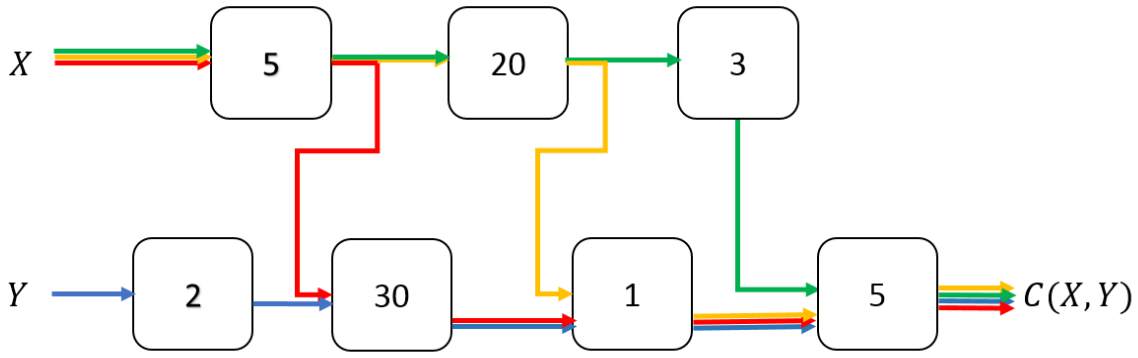

**Slika 4.** Realizacija Booleove funkcije $D = AB + B\bar{C}$.

Kod ovakvog prikaza mreže prirodno je i očekivano sagledavati smjer propagacije signala s lijeva na desno pri čemu su ulazi u ovom primjeru (*Slika 4*) označeni s $A$, $B$ i $C$ dok je jedini izlaz označen s $D$. Izlaz $D$ ostvaruje signal nakon određenog propagacijskog kašnjenja. Latencija čitavog sklopa, odnosno ukupno vrijeme potrebno da se signal propagira s ulaza na izlaz, ovisi o broju i vrsti logičkih vrata na najsporijem putu od ulaza do izlaza.

Praktične izvedbe kombinacijskih mreža mogu sadržavati iznimno veći broj logičkih vrata i razina nego što je to prikazano u primjeru (*Slika 4*).

Jedan od značajnijih svojstava svake kombinacijske mreže jest njezina propusnost. Propusnost se definira kao razina sposobnosti potpune propagacije podataka kroz mrežu u jedinici vremena. Definicija propusnosti usko je povezana s latencijom mreže što će biti objašnjeno u nastavku.

Na slici (*Slika 5*) prikazan je apstraktni model mreže u kojem su simboli osnovnih logičkih vrata (elemenata) apstrahirani jer u ovom trenutku nisu bitni. Iz zadanog primjera opisan je način računanja latencije i propusnosti mreže.




**Slika 5**. Apstraktna reprezentacije jednostavne kombinacijske mreže

U zadanom prikazu (Slika 5) jednostavne kombinacijske mreže označena su dva ulaza simbolima $X$ i $Y$ dok je izlaz prikazan kao funkcija ulaznih varijabli $C(X,Y)$. Svaki kvadrat predstavlja neki mogući logički element mreže, dok pridruženi broj označava vrijeme kašnjenja tog elementa. Za zadani primjer od ulaza do izlaza ukupno postoji 4 moguća puta, a to su svi mogući putevi između $X$ i $C$ te $Y$ i $C$ uzimajući u obzir smjer propagacije signala. Prema tome, putevi u suprotnom smjeru propagacije signala se ne uzimaju u izračun jer nisu validni putevi. Pregledavanjem svih puteva te sumacijom vremena kašnjenja svih pojedinih elemenata na

nekom putu, zaključujemo da vremenski najdulji put iznosi 41 vremenskih jedinica (označen crvenom bojom). Upravo taj put definira latenciju čitave kombinacijske mreže.

Propusnost sada možemo definirati kao recipročnu vrijednost latencije. Iz takvog odnosa latencije i propusnosti, slijedi da bilo kakvo smanjenje vremenski najduljeg puta u mreži rezultira povećanjem propusnosti. Za zadani primjer (Slika 5) propusnost iznosi $\frac{1}{41}$.

Općenito postoje nekoliko načina povećanja propusnosti kombinacijske mreže od kojih su najznačajniji povećanje propusnosti fizičkim paralelizmom i ostvarenjem protočnog načina rada [2].

U okviru ovog završnog rada biti će detaljnije analiziran samo postupak prilagodbe kombinacijske mreže za protočni način rada (treće poglavlje).

# 3. Optimizacija povećanja propusnosti

Kao što je spomenuto i u prethodnom poglavlju ovog rada, jedna od glavnih svojstava kombinacijske mreže jest upravo njezina propusnost. S ciljem povećanja iskoristivosti neke mreže moguće je izvršiti postupke svojevrsne paralelizacije prilagodbom mreže za protočni način rada. Pri tome rast iskoristivosti ovisi o tome koliko razina protočnosti (engl. *pipelining*) je korišteno i kolika su kašnjenja korištenih elementa. Taj problem biti će opisan nešto kasnije u okviru ovog poglavlja.

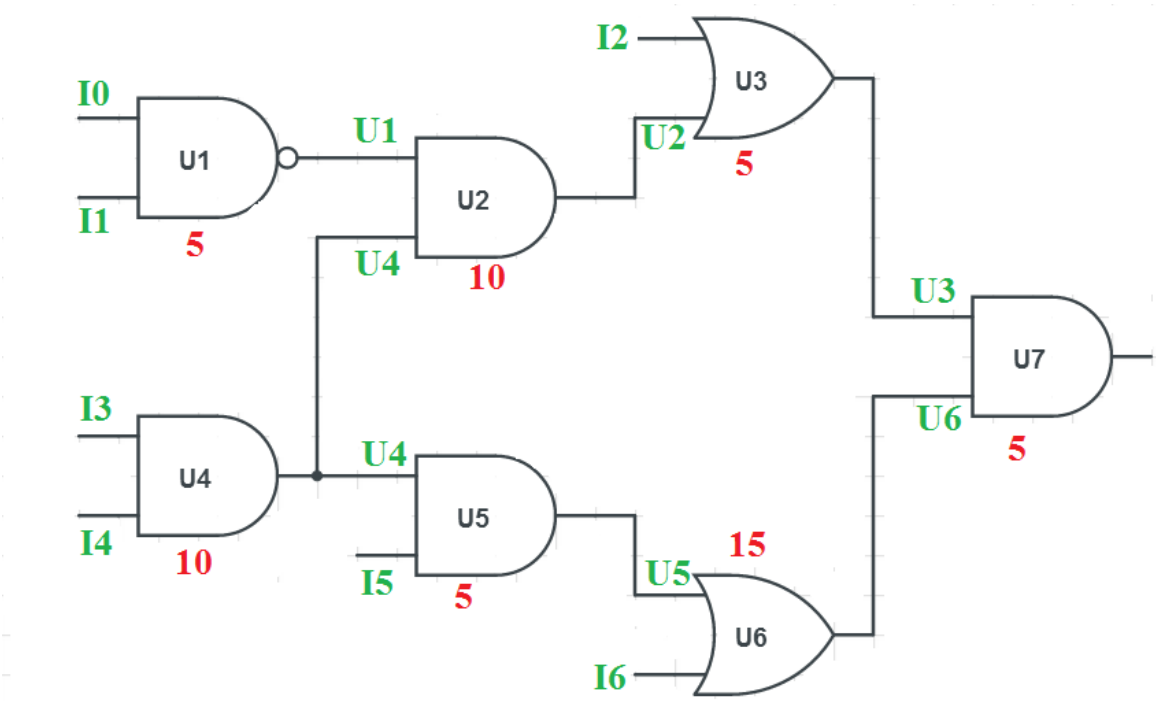
Sam postupak prilagodbe za protočni način rada naziva se još i protočna obrada (engl. *pipeline processing*) ili vremenski preklapajuća paralelna obrada (engl. *time-overlapped parallel processing*) [2].

Tehnika prilagodbe kombinacijske mreže za protočni način rada provodi se uvođenjem dodatnog kašnjenja na ispravnim mjestima u mreži čime se ostvaruju dodatne razine paralelne obrade podataka. Dodatno kašnjenje je pri tome ostvareno ubacivanjem memorijskih elemenata, odnosno bistabila (engl. *flip-flop*) u samu mrežu. Postavljanjem memorijskih elemenata na pogrešnim pozicijama poništava se sama struktura i funkcijska semantika početne mreže u kojoj nema ostvarena protočnost te se iz tog razloga mora poštivati pravilo ispravnosti kombinacijske mreže. Naime, kako bi se održala struktura funkcijski točno oblikovane mreže koja u sebi ima dodatne memorijske elemente potrebno je da svaki mogući put od ulaza do izlaza mreže ima identičan broj bistabila. Ukoliko je to pravilo zadovoljeno, novonastala mreža sigurno jest funkcijski točna i ostvaruje jednaku funkciju kao i početna mreža. Upravo u zadovoljavanju navedenog pravila leži težina optimizacije propusnosti kombinacijske mreže.

Kod prilagodbe mreže za protočni način rada moguće je ostvariti više razina paralelne obrade ovisno o tome koliko je nadodano bistabila u svaki mogući put mreže. Dodavanjem bistabila ostvaruje se vremenska podjela mreže na niz, ne nužno, jednakih dijelova. Tako oblikovana mreža tada se naziva N – razinska protočna kombinacijska mreža (engl. *n – stage pipeline circuit*) [2]. Po dogovoru, N razina označava postojanje N bistabila na svakom putu mreže uzimajući u obzir i posljednji izlazni bistabil. Međutim, prilikom pretrage prostora rješenja taj posljednji bistabil neće biti uzet u obzir jer on uvijek implicitno postoji u strukturi mreže.

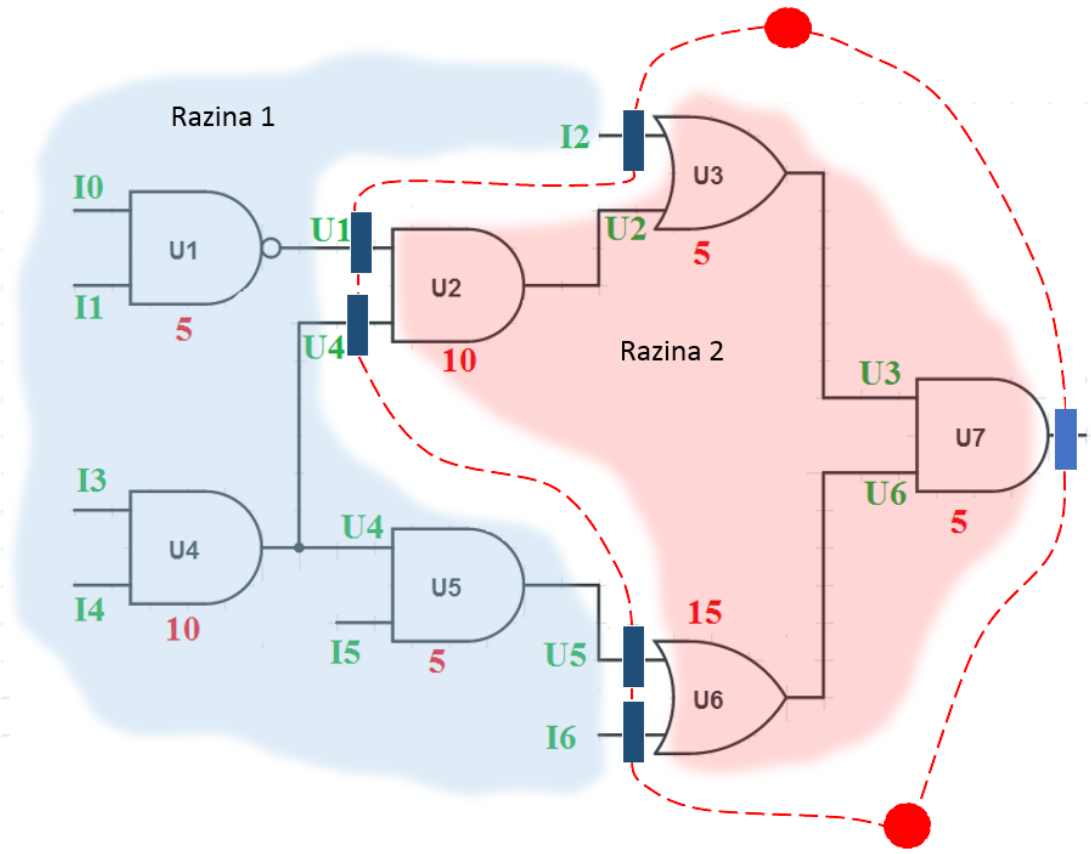
Jednostavan primjer pretvorbe početne mreže u 2 – razinsku protočnu kombinacijsku mrežu prikazan je u nastavku pri čemu su označeni nazivi (zeleno) i pridružena kašnjenja pojedinih elemenata (crveno). Propagacija signala sagledava se na slikama s lijeve na desnu stranu. Nazivi

pojedinih elemenata označeni su u formatu $Ui$ pri čemu je $i$ indeks elementa, dok su izravni ulazi mreže označeni u formatu $Ii$ pri čemu je $i$ indeks izravnog ulaza. Izlazni elementi mreže su svi oni elementi koji nisu ulazni niti jednom drugom elementu. Za navedeni primjer, jedini izlazni element je $U7$. Slika (*Slika 6*) predstavlja početnu mrežu u kojoj nema dodatnih memorijskih elemenata dok je na slici (*Slika 7*) prikazana odgovarajuća mreža s ostvarenom paralelnom obradom podataka.




**Slika 6.** Mreža bez memorijskih elemenata s sekvencijalnom obradom podataka.

Navedeni primjer (*Slika 6*) potrebno je pretvoriti u odgovarajući oblik koji podržava protočni način rada. Prema tome, potrebno je ispravno odabrati položaje na koje će biti dodani bistabili s ciljem ostvarenja dvije razine protočnosti. Pri tome je, kao što je već spomenuto, potrebno zadovoljiti pravilo ispravnosti mreže. Za navedeni primjer postoje dva moguća rješenja. Jedno od njih (ono vremenski bolje) označava postavljanje bistabila na sve ulaze elemenata $U2$ i $U6$ te na jedan ulaz elementa $U3$ koji je spojen na izravan ulaz $I3$. Bistabili su radi pojednostavljenja na slici (*Slika 7*) označeni kao tamno plavi popunjeni pravokutnici. Sama izvedba, vrste i način rada bistabila i korištenih elemenata neće biti obrađeni u okviru ovog rada kao ni sama funkcija kombinacijske mreže.




**Slika 7.** 2 – razinska mreža u protočnom načinu rada.

Na slici (*Slika 7*) prikazan je rezultat pretvorbe mreže u protočni način rada. Kao što je vidljivo iz primjera, dodani bistabili vremenski dijele mrežu na dvije razine paralelne obrade podataka. Prva razina (plavo područje) predstavlja obradu ulaznih podataka do trenutka nailaska svih signala na svim putevima do prvog niza bistabila nakon čega izlazni signali razine 1 postaju ulazni signali razine 2 (crveno područje) gledano s lijeva na desno. U trenutku predaje signala između razine 1 i 2, moguće je na ulaze razine 1 postaviti nove signalne vrijednosti čime se ciklus obrade ponavlja. Prilikom obrade drugog niza ulaznih signala u prvoj razini, paralelno se odvija obrada prethodno dospjelih signala u drugoj razini. Time je upravo ostvarena paralelna obrada podataka u kombinacijskoj mreži ne mijenjajući početnu funkcionalnost.

Potrebno je primijetiti već spomenuti dodatni bistabil na samom izlazu mreže, tj. na izlazu elementa $U7$ (pravokutnik svijetlije plave boje). Također samo vrijeme kašnjenja bistabila u ovom trenutku nije bitno.

Propusnost mreže u općenitom slučaju za mrežu bez dodatnih razina možemo izračunati prema sljedećem izrazu:

$$propusnost = \frac{1}{t_{max}}$$

pri čemu je $t_{max}$ vrijeme propagacije signala kroz vremenski najdulji put od ulaza do izlaza mreže. U ovom slučaju, latencija je jednaka vrijednosti $t_{max}$.

Nakon prilagodbe mreže, odnosno ostvarenja N razina paralelne obrade, propusnost se računa prema izrazu:

$$propusnost = \frac{1}{t_{\max\_razina}}$$

Pri čemu je $t_{\max\_razina}$ vrijednost vremenski maksimalne duljine puta u svim razinama paralelne obrade mreže. U ovom slučaju latencija se računa prema izrazu:

$$latencija = \frac{1}{propusnost} \cdot N$$

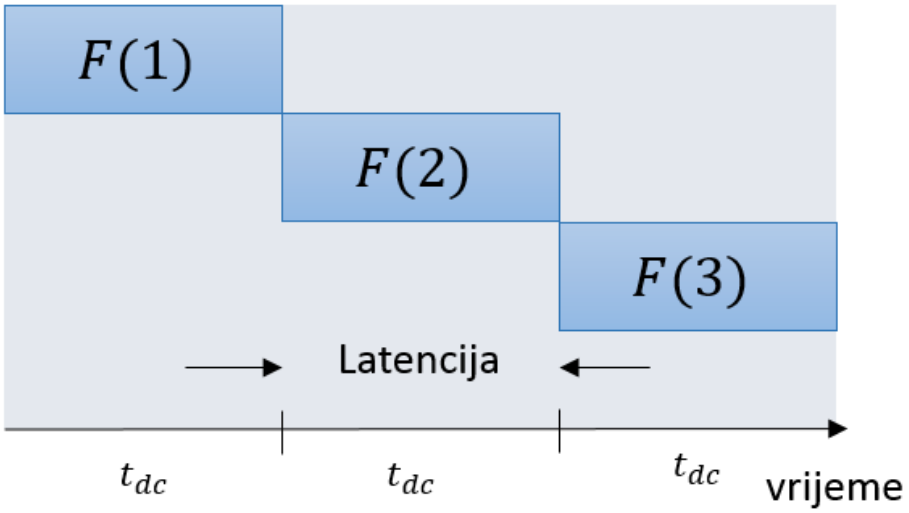
gdje je *N* broj ostvarenih razina u mreži [3].

Za navedeni primjer (*Slika 7*) vremenski najskuplji put ima vrijednost 20 (postoje dva takva puta) i nalazi se u drugoj razini. Uzimajući to u obzir u nastavku (*Tablica 1*) je reprezentativno prikazana usporedba izračuna za navedene primjere (*Slika 6* i *Slika 7*).

| **N (broj razina)** | **latencija** | **propusnost** |
|---|---|---|
| **1 (nema paralelne obrade)** | 35 | 0,028 |
| **2** | 40 | 0,05 |

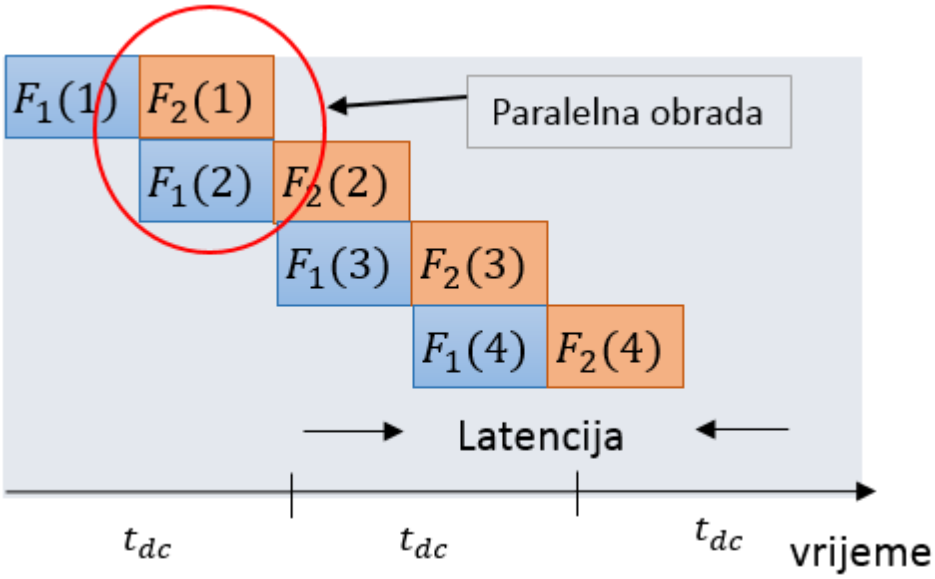
**Tablica 1.** Usporedba za reprezentativne primjere.

Bitno je primijetiti da se vrijednost propusnosti i ukupne latencije u paralelnom načinu rada povećala. Također, u idealnom slučaju (idealni memorijski elementi) ostvarenje dodatnih razina pozitivno utječe na povećanje maksimalne frekvencije rada kombinacijske mreže odnosno na povećanje moguće vrijednosti signala vremenskog vođenja, ili signala tak (engl. *clock*) jer je vrijeme najduljeg puta sigurno manje od najduljeg puta u mreži s samo jednom razinom. U realnoj situaciji potrebno je uzeti u obzir dodatna ograničenja koja se ne obrađuju u okviru ovog rada.




**Slika 8.** Propagacija signala u kombinacijskoj mreži.

Za kombinacijsku mrežu koja nema dodatne memorijske elemente svaki ulazni podatak se obrađuje u jednom prolazu kroz čitavu mrežu za vrijeme jednog ciklusa $t_{dc}$ u trajanju jednakom latenciji kombinacijske mreže (*Slika 8*).




**Slika 9.** Propagacija signala u 2 – razinskoj protočnoj mreži.

Uvođenjem dodatnih razina ukupni posao obrade se razlaže na manje dijelove koji se obrađuju paralelno. U idealnom slučaju u kojem zanemaruje vrijeme kašnjenja memorijskih elemenata

te za idealnu podjelu mreže na dvije vremenski identične razine u jednakom vremenskom ciklusu $t_{dc}$ obrađuje se dvostruko više ulaznih podataka (*Slika 9*).

Bitno je razumjeti da nije moguće za svaku kombinacijsku mrežu naći raspored bistabila koji ostvaruje dodatne razine. Teorijski je svaka mreža ograničena odozgo s maksimalnim brojem razina kojih je moguće ostvariti. Maksimalno ograničenje moguće je izračunati na temelju najkraćeg postojećeg puta u mreži od ulaza do izlaza. Pri tome se ne računa duljina po vremenskoj karakteristici već po samom broju elemenata na tom putu. Budući da uvijek mora biti zadovoljeno osnovno pravilo ispravnosti mreže tj. jednak broj bistabila na svim putevima, maksimalan broj bistabila na svim putevima ograničen je na maksimalan broj bistabila na najkraćem putu. To ograničenje i dalje ostaje teorijsko. Iako naizgled mreža podržava paralelnu obradu s $N$ razina paralelizacije, nije nužno da je sama izvedba u praksi i moguća. Također, moguće je nanizati više direktno povezanih bistabila za redom i time olabaviti gornje ograničenje, takav pristup neće biti obrađen u okviru ovog rada zbog načina reprezentacije kombinacijske mreže koji ne podržava direktno povezivanje više bistabila. Detaljnije o reprezentaciji rješenja opisano je slijedećim poglavljima.

## 3.1. Oblikovanje rješenja

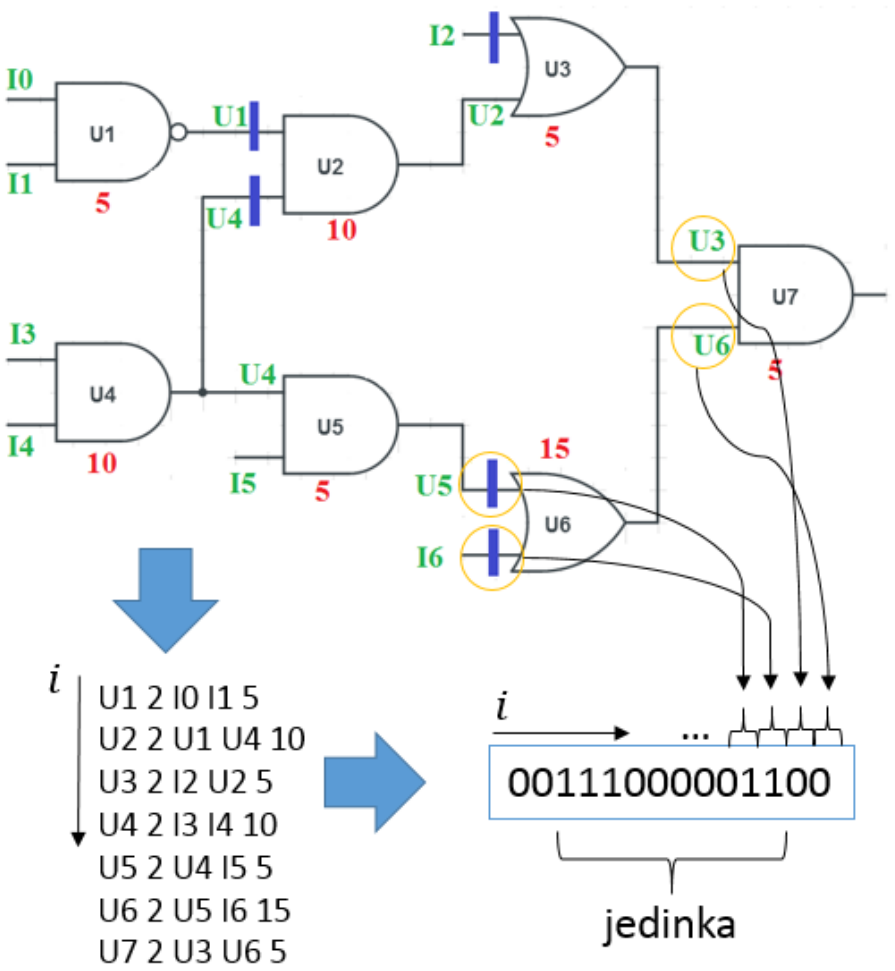
Već je spomenuta potreba preslikavanja realnog problema u optimizacijski problem kao osnova za oblikovanje prostora rješenja u okviru evolucijskih algoritama. U okviru ovog rada izrađen je evaluator reprezentacijskih jedinki koji za određeno rješenje, tj. neki raspored bistabila u mreži, izračunava vrijednost dobrote pojedine jedinke.

Problem preoblikovanja izražen je kao problem minimizacije vremenski maksimalnog puta u kombinacijskoj mreži. Detaljnije je opisano u nastavku.

### 3.1.1. Genotipska reprezentacija mreže

Jedno od osnovnih pitanja jest kako prikazati problem preoblikovanja mreže samom algoritmu s obzirom da algoritam vidi samo neku genotipsku reprezentaciju mreže ne poznajući njezino realno značenje. U okviru implementacije prirodno se nameće reprezentacija bitovnim nizom (engl. *bit string*).

Svaki logički element u mreži ima određeni broj ulaza na kojeg su spojeni ili drugi elementi ili izravni ulazi mreže. Na svakom ulazu bilo kojeg elementa moguće je postaviti ili izostaviti bistabil. Prema tome, ukoliko indeksiramo element, moguće je svaki njegov ulaz preslikati u vrijednost 0 (bistabil izostavljen) ili 1 (bistabil postavljen) na odgovarajućoj poziciji genotipske reprezentacije, tj. bitovnog niza. Indeksiranje elemenata određeno je redoslijedom zapisa pojedine definicije elemenata u ulaznoj datoteci koja opisuje samu mrežu pri čemu je podniz bitova koji predstavlja postojanje ili ne postojanje bistabila na nekom od ulaza pojedinog elementa također određeno redoslijedom navođenja u definicijskoj datoteci (*Slika 10*).




**Slika 10.** Preslikavanje mreže u odgovarajuću genotipsku reprezentaciju.

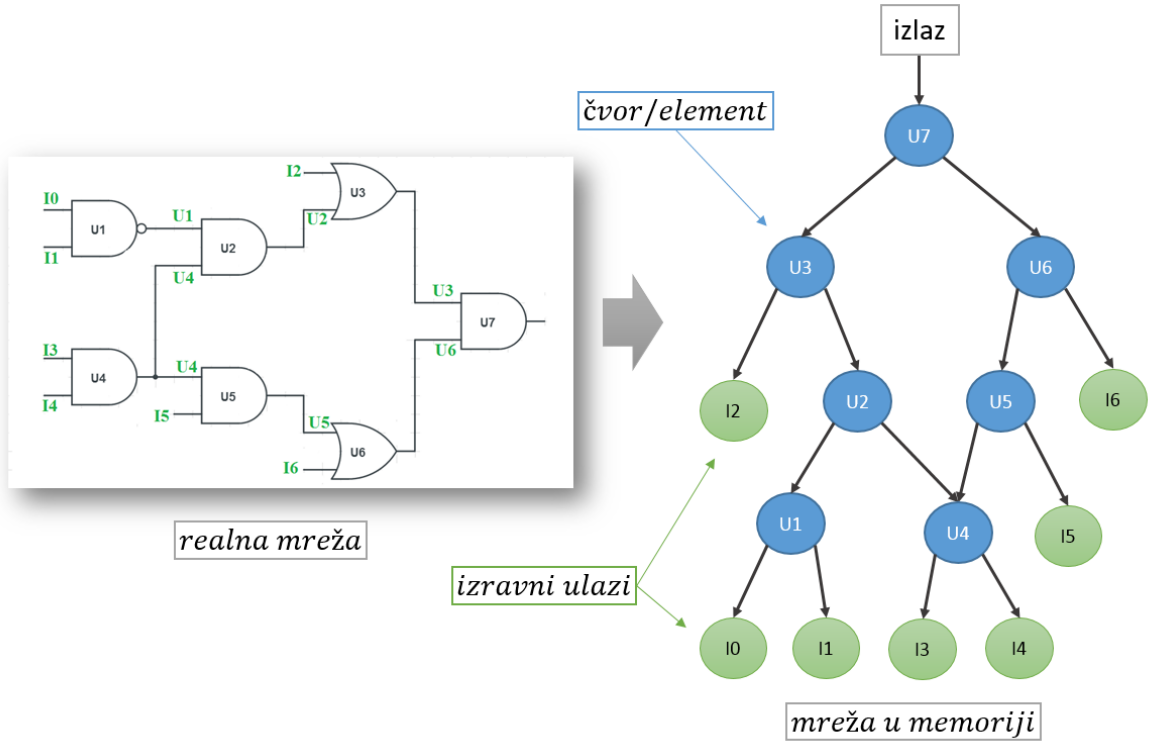
Ovako reprezentirana jedinka predstavlja vrlo jednostavan genotip za evoluciju te je pripremu mreže na temelju jedinke moguće odraditi u vrlo kratkom vremenu.

Kao i svaka reprezentacija, bitovni niz sa sobom nosi i nekoliko ograničenja. Jedno od njih je upravo postojanje gornjeg ograničenja broja maksimalno mogućih razina paralelne obrade na temelju maksimalnog broja bistabila u najkraćem putu. Naime, ova reprezentacija pretpostavlja mogućnost postavljanja samo jednog bistabila na neki od ulaza mreže što rezultira nemogućnošću promjene gornjeg ograničenja. Ukoliko se inzistira na promjeni ograničenja uz korištenje ovako odabranog genotipa, potrebno je ili izvesti promjene u samoj implementaciji evaluatora ili uvesti promjene prilikom postupka preslikavanja mreže u jedinku. Budući da se najčešće želi izbjeći dodatne izmjene programske implementacije, prvi slučaj nije idealan te je jednostavnije uvesti promjenu prilikom preslikavanja pri čemu je jedno od mogućih rješenja

uvođenje pseudo tj. praznih elemenata s samo jednim ulazom u definicijsku datoteku pri čemu je njihova vrijednost kašnjenja jednaka nuli. Tako nadodani elementi za evaluator predstavljaju nove moguće pozicije bistabila bez potrebe za izmjenom fizičke mreže ili početne implementacije evaluatora.

### 3.1.2. Podatkovna struktura mreže u memoriji

Zbog same strukture kombinacijske mreže kao pogodna memorijska struktura izabrana je svojevrsna stablasta podatkovna reprezentacija (*Slika 11*). Svaki čvor predstavlja neka logička vrata mreže i svaki čvor ima određeni broj djece. Djeca nekog čvora označavaju konkretnu vezu između elemenata pri čemu svaki čvor poznaje isključivo svoje prethodnike, tj. djecu. Prema tome, niti jedan čvor nema izravnu referencu na roditeljski čvor iz razloga što to algoritamski nije potrebno te za velike mreže ne zauzima memorijski prostor dodatnim redundantnim informacijama.




**Slika 11.** Prikaz stablaste strukture kombinacijske mreže u memoriji.

Stablasta memorijska struktura pogodna je i iz razloga što ju je jednostavno generirati na temelju danog ulaza (*Slika 10*). Ako bolje pogledamo definicijsku datoteku mreže, možemo primijetiti da se izravno ne može očitati koji su točno elementi izlazni elementi mreže. Međutim taj dodatni podatak može se lako generirati kratkom analizom veza čvorova. Izlazni elementi su svi oni elementi koji nisu ulazni niti jednom drugom čvoru.

Postojanje bistabila na nekom čvoru manifestira se postojanjem vrijednosti 1 ili 0 u kolekciji za svaki ulaz nekog čvora pri čemu indeksi pojedinih vrijednosti odgovaraju indeksima u kolekciji koja sadrži reference na djecu čvora.

### 3.1.3. Algoritam izračuna dobrote

Evolucijski algoritmi, kao što je već spomenuto, svoju selekciju jedinki temelje na njihovoj dobroti. S ciljem ocjenjivanja nekog genotipa u okviru minimizacije vremenski maksimalnog puta u mreži, potrebno je definirati prikladan algoritam pronalaska maksimalnog puta u nekoj postojećoj razini paralelne obrade. Budući da se mreža u memoriji predstavlja svojevrsnom stablastom strukturom, kao osnovni način prolaska kroz sve moguće puteve strukture prirodno se nameće rekurzija koja je upravo i korištena u konkretnoj implementaciji evaluatora.

Osnovni problem koji se javlja prilikom pretrage maksimalnog puta jest pamćenje svih stanja u kojim se algoritam prethodno nalazio. Naime, potrebno je u strukturi pronaći sve moguće puteve u svim razinama trenutne konfiguracije mreže. Budući da se radi o stablastoj strukturi, u svakoj pojedinoj razini vrši se rekurzivni prolaz kroz sve puteve razine s ciljem pronalaska najduljeg puta. Prilikom prijelaza iz jedne razine u drugu važno je krenuti uvijek od kraja zadnje obrađene razine. Kako bi algoritam znao koja razina je sljedeća, potrebno je pamtiti zadnje posjećene elemente u mreži. Pojednostavljen algoritam pretrage najduljeg puta po razinama prikazan je u nastavku u obliku pseudokoda:

```
lastVisited = getAllOutputNodes();
while (lastVisited.size()>0) {
        lastVisitedTemp = [0]
        for (nextNode in lastVisited) {
            findMaxPath(lastVisitedTemp, nextNode, layerMax);
        }

        //osvježi listu zadnje posjećenih stanja
        lastVisited = lastVisitedTemp;

        //zapamti najdulji put
        if (layetMax > max) {
            max = layerMax;
        }
}
```

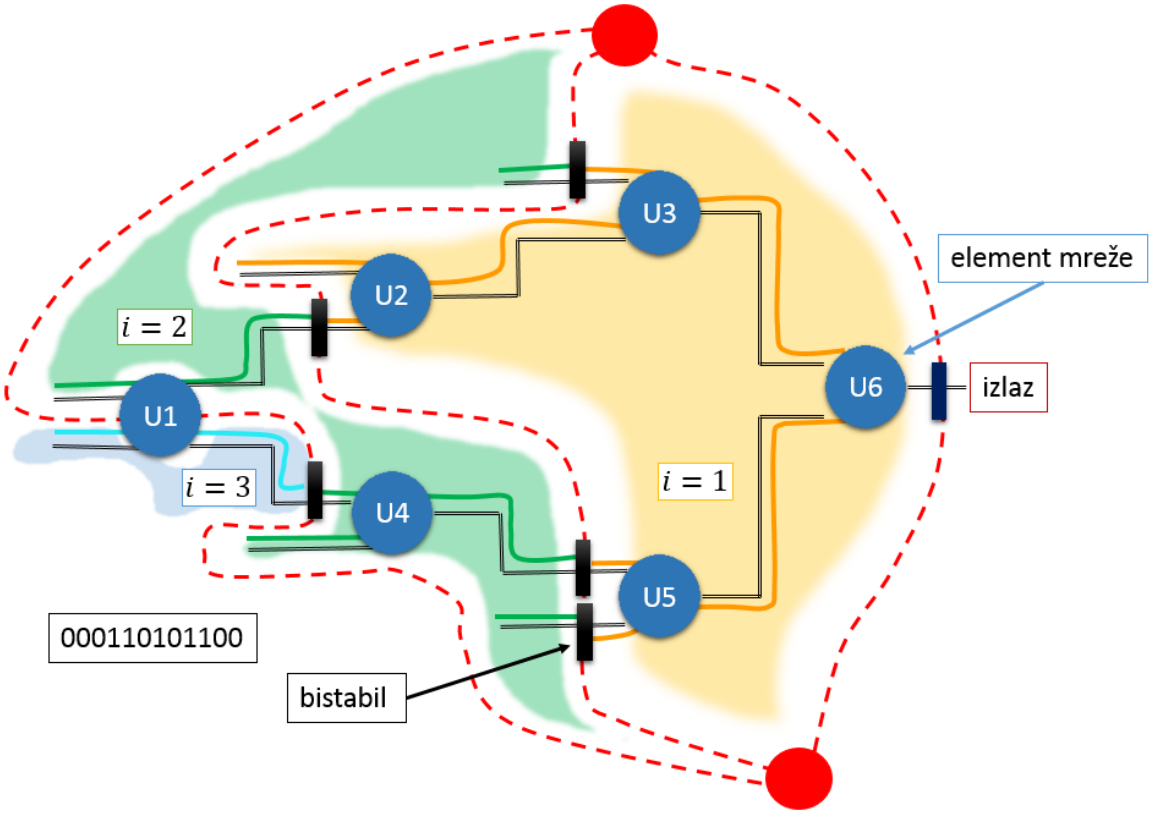
**Slika 12.** Osnovni algoritam prolaska po razinama uz pamćenje prethodnih stanja.

U zadanom algoritmu se iterativno poziva rekurzivna funkcija *findMaxPath* (*Slika 12*) pri čemu se prilikom svakog sljedećeg poziva predaje nova lista početnih stanja koji predstavljaju zadnje posjećene elemente, tj. granične elemente između dvije razine.

Zbog načina definicije mreže, implementacija se temelji na prolazu po razinama od izlaznih elemenata prema ulaznim. Semantika samog prolaza time ostaje nepromijenjena. Sukladno tome, početni elementi s kojima se kreće u rekurziju upravo jesu svi izlazni elementi mreže.

Sam rekurzivni algoritam (*findMaxPath*) prolaza temelji se na jednostavnom prolasku stablaste strukture od korijena, odnosno trenutno zadnje posjećenih elemenata, do listova tj. izravnih ulaza mreže ili sljedećih elemenata koji se nalaze nakon sljedećeg bistabila na nekom putu. Ovakvim iterativnim prolazom uz pamćenje prethodnih stanja, osiguran je prolaz svih mogućih puteva u svim razinama trenutne konfiguracije mreže.

Jednostavan prikaz iterativnog rekurzivnog prolaza prikazan je u nastavku (*Slika 13*). Na slici je prikazana neispravna mreža s genotipom 000110101100. Rekurzivni prolaz započinje od jedinog izlaznog elementa *U*6 te kroz tri iteracije određuje vremenski najdulji put u trenutnoj konfiguraciji mreže. Potrebno je još jednom naglasiti da se implicitni izlazni bistabil ne uzima u obzir prilikom rada algoritma.




**Slika 13.** Grafički prikaz iterativnog rekurzivnog algoritma kroz tri iteracije.

Algoritam iterativnog rekurzivnog prolaza ne predstavlja nužno vremenski zahtjevan dio ukupnog algoritma evaluatora budući da najčešće ne ulazi u duboku rekurziju. Tek manji dio puteva nema nikakvog bistabila što je sasvim očekivano jer postavljanje jednog bistabil u mreži automatski utječe na niz puteva kojima je ta pozicija samo jedan od elementarnih dijelova.

Vremenski iznimno zahtjevniji dio evaluacije događa se nakon samog pronalaska najduljeg puta. Tada je potrebno ocijeniti mrežu kao ispravnu ili neispravnu i sukladno tome modificirati dobrotu jedinke na neki definirani način. Ocjena ispravnosti temelji se na zadovoljavanju osnovnog pravila ispravnosti mreže spomenutog u prethodnim poglavljima. Budući da je potrebno prolaziti kroz sve moguće puteve mreže, neizbježno je rekurzivno proći sve dubine stablaste strukture što može predstavljati iznimno vremenski zahtjevan korak algoritma. Naime, broj puteva u mreži raste s ukupnim brojem elemenata u mreži i brojem djece svih pojedinih elemenata stablaste strukture. Sukladno tome raste i broj mogućih pozicija bistabila u mreži, odnosno veličina samog genotipa. Za male primjere vrijeme evaluacije ostaje zanemarivo dok se na već nešto većim primjerima događa kombinatorna eksplozija broja puteva pa tako i ukupno potrebnog vremena za evaluaciju jedinke.

Prilikom dodjeljivanja vrijednosti dobrote nekoj jedinci u obzir je uzet i broj puteva koji nemaju predodređeni broj bistabila, tj. $N-1$ bistabila, pri čemu je $N$ ciljni broj razina paralelne obrade. Sve se jedinke koje nemaju na svim putevima zadani broj bistabila kažnjavaju dodatnim promjenama vrijednosti dobrote. Implementacijski se dodjeljivanje dobrote vrši po sljedećem izrazu

$$dobrota = t_{max} + kazna \cdot M$$

pri čemu je $t_{max}$ vrijeme najduljeg puta, $M$ ukupan broj puteva koji nemaju zadani broj bistabila, a *kazna* je fiksna vrijednost koja može ovisiti o veličini mreže. Prilikom definiranja vrijednosti *kazne* od iznimne je važnosti da i najbolja jedinka koja ne zadovoljava pravilo ispravnosti bude vrednovana lošije od najlošije jedinke koja zadovoljava pravilo ispravnosti.

Kao dodani pristup u evoluciji rješenja korištena je tehnika fiksiranja bistabila u reprezentaciji mreže. Ovakav pristup iziskuje postojanje rješenja koje zadovoljava pravilo ispravnosti mreže. Ukoliko takvo rješenje postoji, moguće je preslikati položaje bistabila tog rješenja u memorijsku strukturu i fiksirati mjesta koja imaju bistabil. Tijekom evolucije se fiksirana mjesta više ne smiju mijenjati. Time se zapravo postiže podjela mreže na ispravne razine te zasebnu manipulaciju pojedinih dijelova mreže. Također važno je uočiti da se implementacijom dinamički prilagođenog genotipa efektivno smanjuje prostor pretraživanja rješenja sa $2^D$na $2^{D-I}$ mogućih rješenja gdje je $D$ veličina inicijalnog genotipa, a $I$ broj već zauzetih pozicija mreže. Ovakvim pristupom teži se iz rješenja koje predstavlja N-razinsku mrežu dobiti barem N+1-razinsku mrežu. Uspješnost te tehnike komentirana je u poglavlju rezultata ovog rada.

Nešto drugačiji pristup izračuna dobrote neke jedinke može se ostvariti dinamičkim praćenjem izmjena nastalih promjenom konfiguracije mreže na temelju evaluacije nove jedinke uz memorijsko očuvanje svakog mogućeg puta u mreži. Ovaj pristup omogućio bi dinamičku nerekurzivnu izmjenu samo onih puteva koji su zaista i izmijenjeni zbog promjene jedinke. Time bi se teorijski mogao ubrzati postupak ocijene dobrote. No zbog moguće velikog broja puteva pristup iziskuje iznimno veliku memorijsku potrošnju te izgradnju kompleksnijeg evaluatora koji je u stanju pratiti promjene svakog puta tijekom evaluacije. Zbog prevelikog memorijskog opterećenja, takav algoritam nije korišten u okviru ovog rada.

### 3.1.4. Problemi implementacije

Tijekom razvoja evaluatora i korištenih algoritama pokazalo se nekoliko problema i ograničenja na koje je potrebno obratiti pažnju.

Osnovni problem korištenog rekurzivnog algoritma jest potreba za prolazom iznimno velikog broja puteva za nešto veće mreže. Rješenje bi bilo korištenje nerekurzivnog prolaza stablaste strukture. U praksi postoje algoritmi koji to omogućavaju pa čak i s linearnom složenošću ($O(n)$). Međutim ti algoritmi rade isključivo nad stablima kod kojih čvor ima isključivo jednog roditelja i svaki čvor ima isključivo dvoje djece. Niti prvo niti drugo pravilo ne mora biti zadovoljeno u kombinacijskom mreži te se zbog toga takvi algoritmi ne mogu koristiti. Opcionalno, moguće je formirati vlastiti stog te uz pomoć drugih potpornih struktura simulirati rekurzivni prolaz pomoću petlje. No i takav pristup se pokazao sporijim zbog opsežnog korištenja potpornih struktura.

Nadalje, kod iterativnog rekurzivnog algoritma potrebno je obratiti pažnju na kolekciju koja čuva reference na zadnje posjećene elementa. U nekoj razini sasvim je moguće da velik broj puteva završava u jednakom čvoru. Ukoliko se ne osigura jedinstvenost elemenata u kolekciji zadnje posjećenih čvorova, algoritam će bespotrebno iterativno prolaziti kroz razine koje je već obradio. Jednostavna solucija je korištenje neke izvedbe sučelja koje ne podržavaju višestruke jednake elementa (engl. *set*).

## 3.2. Primjena evolucije

Iz prethodno navedenog opisa problema propusnosti, naizgled se čini kao da je relativno jednostavno pronaći odgovarajuće pozicije na koje je potrebno postaviti bistabile. Međutim, navedeni reprezentativni primjer (*Slika 6*) s tek devet puteva i četrnaest mogućih pozicija bistabila daleko je od realnih problema te za ovako male mreže nije potrebno koristit evolucijske algoritme već je dovoljno iscrpnom pretragom pretražiti prostor svih mogućih rješenja. U realnim situacijama broj mogućih puteva i pozicija bistabila iznimno je veći. U okviru ovog rada glavni aspekt optimizacije propusnosti i ostvarenja paralelnog izvođenja usmjeren je na realnu S-box kombinacijsku mrežu koja predstavlja jednu od osnovnih komponenata simetričnih algoritama kriptiranja te služi za izvršavanje supstitucije. S-box mreža sadrži, uz ograničenja reprezentacije rješenja, 432 moguće pozicije bistabila. Kao što je spomenuto u dijelu genotipske reprezentacije mreže, 432 moguće pozicije bistabila rezultira jedinkom veličine 432 vrijednosti od kojih svaka može biti 0 ili 1 čime prostor pretraživanja sadrži ukupno $2^{432}$ moguća rasporeda bistabila u S-box kombinacijskoj mreži. Ukoliko bi prostor rješenja pretraživali tehnikom grube sile i ukoliko bismo imali računalo koje bi jednu kombinaciju moglo provjeriti u jednoj nanosekundi ($1 \cdot 10^{-9}$ sekundi), za provjeru svih kombinacija trebali bismo $3{,}51 \cdot 10^{113}$godina! Kako bismo bolje razumjeli koliko je ovo vremena, dobro je spomenuti da je svemir star oko $1{,}37 \cdot 10^{10}$ godina. Pri tome je u realnom okruženju zbog inherentne složenosti evaluacije jednog rješenja, ukupno potrebno vrijeme iznimno veće.

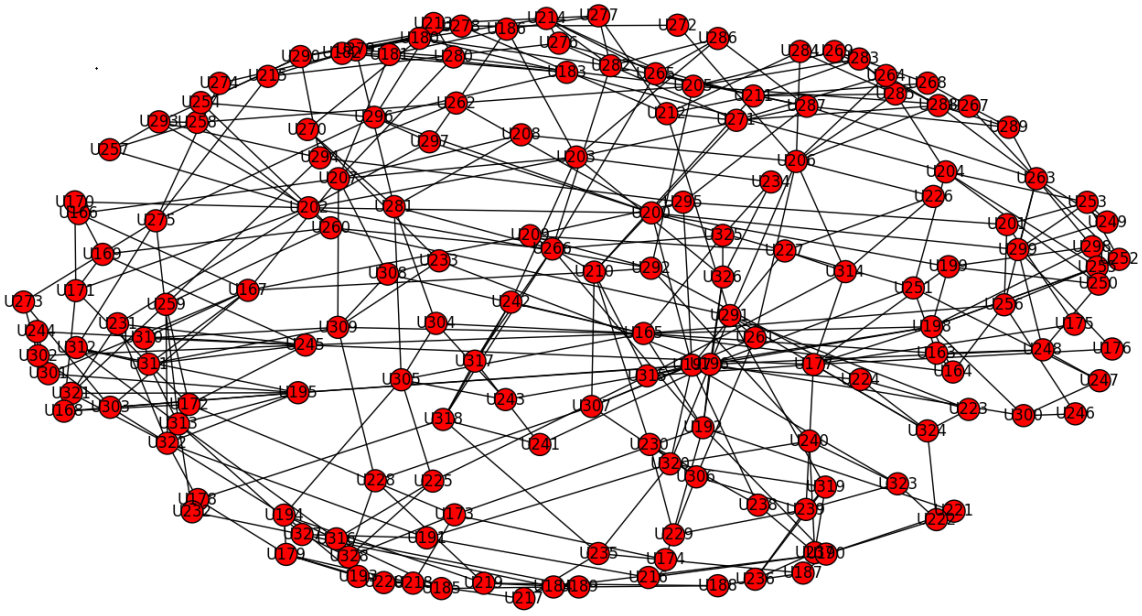
U prethodnom poglavlju spomenuta je kombinatorna eksplozija broja puteva sukladno povećanju broja elemenata i broja djece pojedinih elemenata. Prikaz eksplozije broja puteva u odnosu na broj mogućih pozicija bistabila predstavljen je u sljedećoj tablici:

| **Broj pozicija (veličina genotipa)** | **Broj puteva u mreži** |
|---|---|
| **12** | 8 |
| **21** | 29 |
| **121** | 475 |
| **217** | 902 |

| | |
|---|---|
| **432 (S-box)** | 8 023 409 |

**Tablica 2.** Broj puteva za različite veličine genotipa.

Važno je napomenuti da točnu vezu između veličine genotipa i ukupnog broja puteva nije moguće direktno izvesti. Ukupan broj puteva ovisi i o načinu povezivanja elemenata unutar mreže. Iz tog razloga navedeni podaci mogu poslužiti samo kao reprezentacijske vrijednosti.



**Slika 14.** Prikaz čitave S-box mreže.

Također uvidom u tablicu (*Tablica 2*) u zadnjem retku je dana statistika za S-box mrežu koja je korištena u realnom testiranju kao dio ovog rada. Možemo primijetiti da je ukupan broj puteva veći od osam milijuna, što jako utječe na vrijeme izvođenja samog algoritma. U poglavlju 3.1.3. spomenuta je potreba za rekurzivnim prolazom algoritma kroz sve puteve mreže. To upravo znači da je potrebno za svaku pojedinu evaluaciju jedinke proći preko osam milijuna puteva rekurzivno! Prikaz S-box mreže iz koje je vidljiva kompleksnost uslijed razgranate povezanosti elemenata predočen je na slici (*Slika 14*).

# 4. Rezultati

Zbog težine samog problema optimizacije propusnosti S-box mreže, vrijeme potrebno za izvršavanje dovoljnog broja pokusa je iznimno. U skladu s vremenskom ograničenošću, u nastavku su prikazani postignuti rezultati evolucije rješenja.

Ukoliko drugačije nije napomenuto, sva rješenja su dobivena korištenjem eliminacijskog genetskog algoritma (engl. *steady-state genetic algorithm*).

## 4.1. Izvedba implementacije

Evaluator jedinki izgrađen je korištenjem objektno orijentiranog jezika *C++*. Implementacija evaluatora integrirana je kao evaluacijska komponenta za korištene algoritme u sklopu postojećeg radnog okvira *ECF* (*Evolutionary Computation Framework*).

## 4.2. Ispitni primjer

### 4.2.1. Početna statistika

Osnovni ispitni primjer je realna S-box mreža opisana u prethodnom poglavlju primjene evolucije (3.2). Za početak potrebno je pregledati sve početne statistike, tj. statistike mreže koja ima samo jednu razinu rada i nema dodatnih memorijskih elemenata u strukturi (*Tablica 3*).

| **S-box mreža (*Slika 16)*** | |
|---|---|
| **Ukupan broj puteva** | 8023409 |
| **Vrijeme propagacije kroz najdulji put** | 3848,86 ps |
| **Maksimalan broj razina protočnosti** | 4 |
| **Veličina genotipske reprezentacije** | 432 |

| | |
|---|---|
| **Broj logičkih vrata:** | 166 |
| **Broj izravnih ulaza:** | 29 |

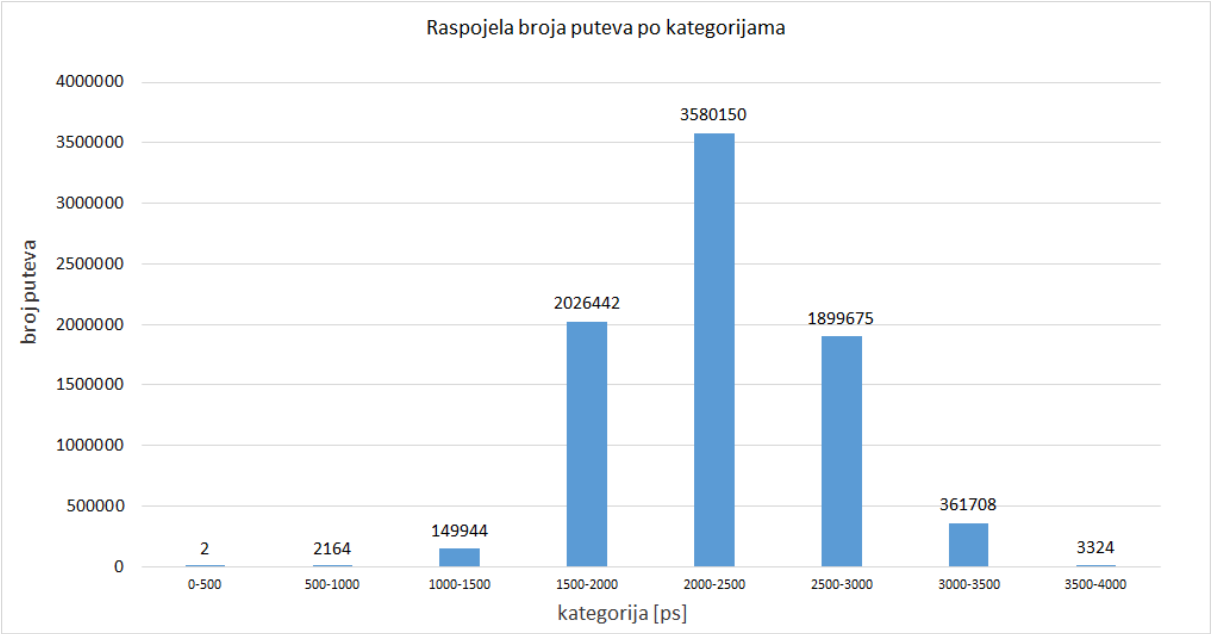
**Tablica 3.** Statistika za inicijalnu mrežu.

U početnoj statistici važno je napomenuti da je teorijski maksimalan broj razina paralelne obrade izračunat ograničenjem maksimalne razine paralelizacije najkraćeg puta. Prema tome teorijski maksimalan broj razina paralelne obrade za S-box mrežu je četiri što označava postojanje tri bistabila na svakom putu mreže ne uključujući posljednji bistabil na svim izlazima mreže.

Također prikladno je pregledati statistiku broja puteva kategoriziranih po vrijednosti propagacije signala za mrežu bez paralelizacije (*Tablica 4, Slika 15*). Pri podjeli uzeta je u obzir granulacija intervala po vrijednosti 500 od 0 $ps$ do vrijednosti 4000 $ps$, budući da je maksimalna vremenska propagacija iznosa 3848,86 $ps$.

| **Interval podjele po vremenu propagacije (ps)** | **Broj puteva** |
|---|---|
| **0 – 500** | 2 |
| **500 – 1000** | 2164 |
| **1000 – 1500** | 149944 |
| **1500 – 2000** | 2026442 |
| **2000 – 2500** | 3580150 |
| **2500 – 3000** | 1899675 |
| **3000 – 3500** | 361708 |
| **3500 – 4000** | 3324 |

**Tablica 4.** Statistika raspodjele broja puteva po vrijednosti propagacije signala.




**Slika 15.** Histogram raspodjele broja puteva po vrijednosti propagacije signala za početnu mrežu.

Iz tabličnog prikaza podjele puteva po duljini (*Tablica 4*) dobro je primijetiti da relativno velik broj puteva spada u kategoriju najduljih puteva. Upravo to sugerira nemogućnost ručnog pomaganja evolucijskom algoritmu pri čemu bi se najduljim putevima inicijalno pridijelili bistabili. Takav pristup je neprikladan iz razloga prevelike povezanosti čvorova u mreži. Dodavanjem samo jednog bistabila na nekoj poziciji u mreži, potencijalno može promijeniti velik broj puteva. Ova tehnika mogla bi se uzeti u obzir za mreže kod kojih je ukupan broj najduljih puteva izrazito malen.

### 4.2.2. Pokusi za N = 2

U ovom poglavlju prikazane su statistike dobivenih rezultata za S-box mrežu s razinom paralelne obrade $N = 2$ što podrazumijeva postojanje samo jednog bistabila na svim putevima isključujući posljednji bistabil na izlazima mreže.

#### 4.2.2.1 Varijabilna vjerojatnost mutacije

U okviru ovog rada izvršeni su pokusi za varijabilnu vjerojatnost mutacije pri čemu su korištene vrijednosti mutacija od 0,3 do 0,6. Za sve pokuse korištena je fiksna vrijednost populacije (25

jedinki) i broj evaluacija (30 000). Svi pokusi izvršeni su ukupno 10 puta i od ukupno 40 pokretanja samo 2 su rezultirali ispravnim rješenjem za vrijednosti vjerojatnosti mutacije 0,4 i 0,6. Ovakvo ponašanje nažalost ne sugerira utjecaj vjerojatnosti mutacije na rad algoritma, odnosno uspješnost generiranja ispravnih rješenja te bi za konkretnije rezultate bilo potrebno izvršavati mnogo veći broj pokusa koji vremenski nadilaze ovaj završni rad.

#### 4.2.2.2 Varijabilan algoritam

Bitno je sagledati različite vrste algoritama na jednakom problemu. I za ovaj optimizacijski problem testirano je više algoritama. Usporedba uspješnosti izvršena je usporedbom najbolje generiranih jedinki na fiksnom broju evaluacija za svaki algoritam posebno.

Za usporedbu uz osnovni genetski algoritam korišteni su i sljedeći algoritmi:

1. genetski algoritam s proporcionalnom selekcijom,
2. algoritam genetskog kaljenja.

Svi pokusi izvršeni su ukupno 10 puta uz fiksan broj evaluacija (10 000). Niti jedan od pokusa do ovog trenutka nije generirao ispravno rješenje te na temelju dobivenih rezultata nije moguće zaključiti uspješnost rada pojedinog algoritma za zadani problem.

#### 4.2.2.3 Pronađena rješenja

Pokusi su u konačnici generirali ispravna rješenja za početnu S-box mrežu. Sva dobivena rješenja ostvaruju dvije razine paralelizacije. Sva prikazana rješenja u nastavku generirana su evaluacijom uz sljedeće parametre:

| Parametar | Vrijednost |
|---|---|
| **Algoritam** | Eliminacijski genetski algoritam |
| **Vjerojatnost mutacije** | 0,45 |
| **Veličina populacije** | 20 – 30 jedinki |
| **Uvjet zaustavljanja** | Izvršen maksimalan broj evaluacija |

| | |
|---|---|
| **Broj evaluacija** | 50 000 – 100 000 |

**Tablica 5.** Korišteni parametri uz koje su generirana ispravna rješenja.

Sjetimo se da je vrijednost vremenskog kašnjenja najduljeg puta početne mreže jednaka 3848,86 $ps$, što rezultira propusnošću od $2{,}598 \cdot 10^{-4}$ i latencijom jednakom duljini tog puta. Sada možemo primijetiti da niti jedno rješenje ne dijeli mrežu na dva jednaka dijela, već se mreža dijeli na dvije vremenski nejednako podijeljene razine.

U nastavku je prikazan postotak promjene propusnosti i latencije za sve postignute ispravne rezultate za $N$ = 2 u odnosu na inicijalnu mrežu.

| **Vrijeme propagacije za najdulji put [$ps$]** | **Propusnost mreže $\cdot 10^8$ [$s^{-1}$]** | **Postotak povećanja [%]** |
|---|---|---|
| 2793,62 | 3,57 | 37,77 |
| 2826,52 | 3,53 | 36,17 |
| 2942,42 | 3,39 | 30,81 |
| 3075,61 | 3,25 | 25,14 |
| 3155,11 | 3,16 | 21,98 |
| 3223,02 | 3,10 | 19,41 |
| 3247,64 | 3,08 | 18,51 |
| 3448,59 | 2,89 | 11,60 |
| 3489,24 | 2,86 | 10,30 |
| 3523,16 | 2,83 | 9,24 |
| 3676,83 | 2,72 | 4,67 |

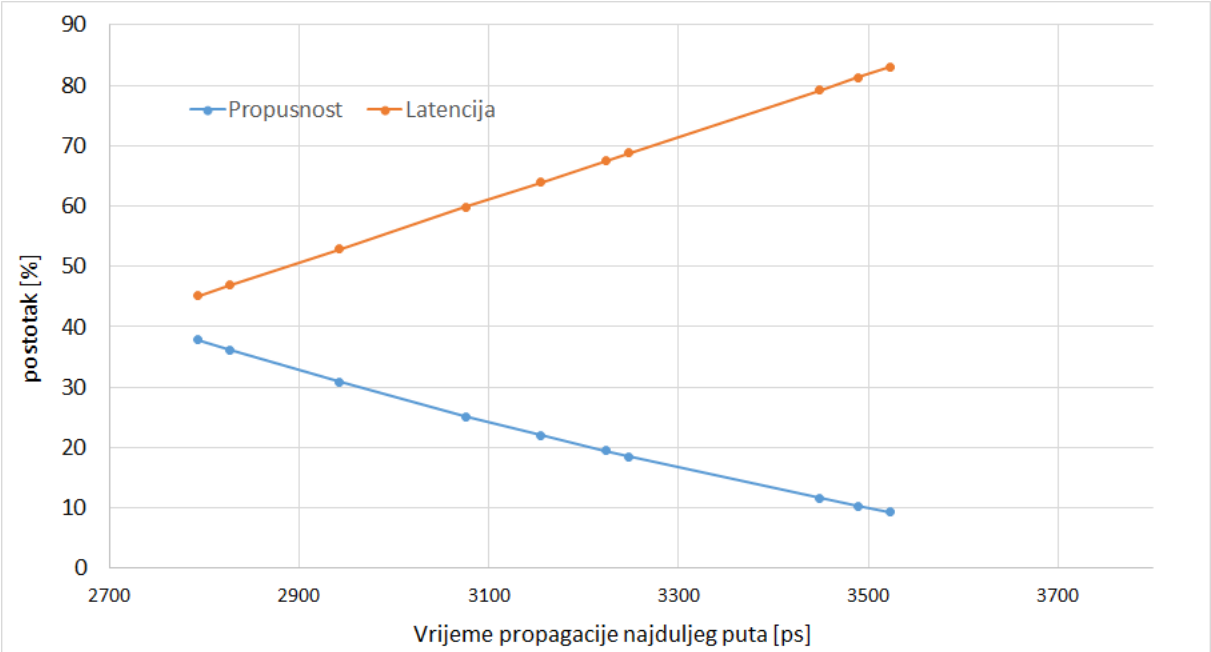
**Tablica 6.** Postotak povećanja propusnosti mreže za N = 2 u odnosu na inicijalnu mrežu.

| Vrijeme propagacije za najdulji put [$ps$] | Latencija mreže [$ps$] | Postotak povećanja [%] |
|---|---|---|
| 2793,62 | 5587,24 | 45,16 |
| 2826,52 | 5653,04 | 46,87 |
| 2942,42 | 5884,84 | 52,90 |
| 3075,61 | 6151,22 | 59,81 |
| 3155,11 | 6310,22 | 63,95 |
| 3223,02 | 6446,04 | 67,47 |
| 3247,64 | 6495,28 | 68,75 |
| 3448,59 | 6897,18 | 79,20 |
| 3489,24 | 6978,48 | 81,31 |
| 3523,16 | 7046,32 | 83,07 |
| 3676,83 | 7353,66 | 91,06 |

**Tablica 7.** Postotak povećanja latencije mreže za N = 2 u odnosu na inicijalnu mrežu.

Prikazani podaci u navedenim tablicama (*Tablica 6, Tablica 7*) jasno predočavaju ostvarenje osnovnog cilja paralelizacije kombinacijske mreže – povećanje propusnosti. Možemo uočiti da se propusnost za dobivena rješenja povećava smanjenjem vremenski najduljeg puta u nekoj od dvije ostvarene razine u mreži. Međutim, dodavanje dodatnih razina paralelne obrade u mrežu negativno utječe na ukupnu latenciju mreže koja se uslijed povećanja broja razina također povećava.

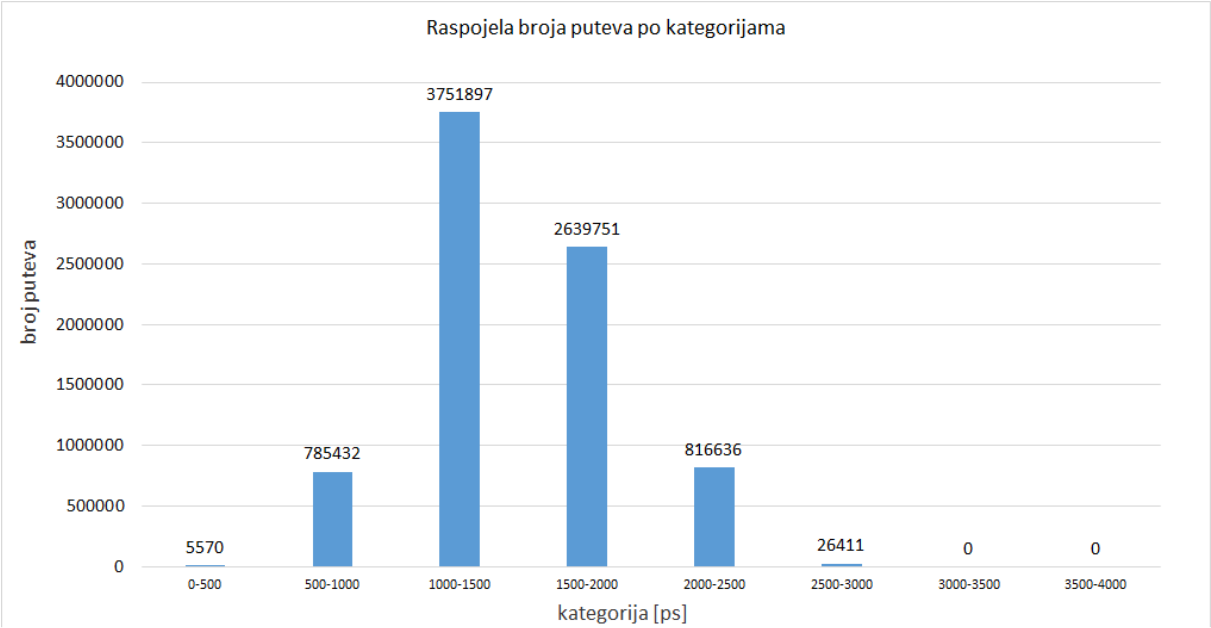
Važno je napomenuti da nisu sva rješenja jednaka na genotipskoj razini već se neka od njih razlikuju iako im je vrijednost dobrote jednaka. Prema tome nisu sva rješenja individualno prikazana u tablici već samo postignute vrijednosti dobrote.




**Slika 16.** Graf odnosa vremena propagacije najduljeg puta i postotnog povećanja propusnosti i latencije.

Iz prikaza (*Slika 16*) možemo vidjeti da bi daljnjim izračunom vrijednosti funkcije postotka povećanja propusnosti, optimum funkcije propusnosti bio postignut upravo za vrijednost jednakoj polovici vrijednosti latencije inicijalne mreže. To semantički odgovara takvoj raspodjeli bistabila koja bi mrežu dijelila na dva vremenski identična dijela. Nažalost, do ovog trenutka takvo se rješenje nije uspjelo evoluirati.

Uspješnost rada genetskog algoritma očituje se u i broju skraćenih puteva mreže. Kao usporedbu u odnosu na početnu raspodjelu (*Slika 15*) zgodno je pogledati prikaz u nastavku.




**Slika 17.** Histogram raspodjele broja puteva po vrijednosti propagacije signala za najbolje postignuto rješenje.

Histogram (*Slika 17*) jasno predočava uspješnost evolucije rješenja. Broj puteva u kategorijama od 2000 do 4000 $ps$ drastično je pao u usporedbi s prikazom na slici 15 te se broj kraćih puteva iznimno povećao, posebno u kategoriji od 1000 do 1500 $ps$.

Tijekom izvođenja pokusa bitno je naglasiti da postavljanje faktora stagnacije izrazito utječe na uspješnost pronalaska rješenja. Kroz pokuse u nekim slučajevima korišten je faktor stagnacije 30. Primijećeno je da postavljanjem faktora stagnacije na spomenutu vrijednost vjerojatnost postizanja rješenja pada na 0%. Uz postavljen faktor čak ni pokusi s 100 000 ili 200 000 evaluacija nisu generirali niti jedno ispravno rješenje (uz veličinu generacije od 20 ili 30 jedinki), dok su pokusi bez faktora stagnacije, ali uz uvjet maksimalnog broja evaluacija kao glavni kriterij zaustavljanja, postizali uspješnost od čak 40% i u uz manji broj evaluacija (50 000).

### 4.2.3. Pokusi za N > 2

Do trenutka pisanja ovog rada izvršeni su pokusi za $N = 3$ i $N = 4$ pri čemu je za oba slučaja korištena veličina populacije od 20 do 30 jedinki te vjerojatnost mutacije 0,45. Za slučaj $N = 3$ u kojem svaki put u mreži ima točno dva bistabila, ne uzimajući u obzir izlazne bistabile, dobivena su dva rješenja opisana u tablicama u nastavku (*Tablica 8, Tablica 9*).

Najbolje postignuto rješenje s tri razine paralelne obrade ima najkraći put s vremenskim kašnjenjem jednakim 2918,92 $ps$. Iako je ta vrijednosti doista i manje od latencije inicijalne mreže, takvo rješenje daje pozitivno povećanje propusnosti od 31,85% i izrazito negativno povećanje ukupne latencije mreže koja je sada čak 127,51% veće od početne. Ta informacija sugerira potrebu za pronalaskom rješenja koje bi mrežu ravnomjernije podijelilo uz manju cijenu povećanja latencije mreže.

| **Vrijeme propagacije za najdulji put [$ps$]** | **Propusnost mreže $\cdot 10^8$ [$s^{-1}$]** | **Postotak povećanja [%]** |
|---|---|---|
| 2918,92 | 3,42 | 31,85 |
| 2931,59 | 3,41 | 31,28 |

**Tablica 8.** Postotak povećanja propusnosti mreže za N = 3 u odnosu na inicijalnu mrežu.

| **Vrijeme propagacije za najdulji put [$ps$]** | **Latencija mreže [$ps$]** | **Postotak povećanja [%]** |
|---|---|---|
| 2918,92 | 8756,76 | 127,51 |
| 2931,59 | 8794,77 | 128,50 |

**Tablica 9.** Postotak povećanja latencije mreže za N = 3 u odnosu na inicijalnu mrežu.

Pokusi za $N = 4$ do ovog trenutka također nisu generirali ispravna rješenja pri čemu je bitno naglasiti da takva rješenja teorijski mogu postojati, uzimajući u obzir spomenuto gorenje

ograničenje mreže. Međutim njihovo postojanje u praksi nije nužno te temeljem rezultata izvršenih pokusa nije moguće potvrditi postojanje istih.

#### 4.2.3.1 Tehnika fiksiranja bistabila

Kao što je već spomenuto, jedna moguća tehnika s kojom se teorijski smanjuje prostor pretraživanja za rješenje koja predstavljaju veći broj razina paralelizacije od $N = 2$ je tehnika fiksiranja bistabila spomenuta u poglavlju opisa algoritma izračuna dobrote (3.1.3).

Postupak je obrađen korištenjem dostignutih ispravnih rješenja s dvije razine paralelne obrade, međutim do ovog trenutka navedena tehnika nije uspjela evoluirati niti jedno ispravno rješenje.

### 4.2.4. Zaključak rezultata

Budući da evaluacija rješenja povlači vremenski zahtjevnu obradu mreže, do ovog trenutka sakupljen je djelomično ograničen skup rezultata na temelju kojih nije direktno moguće ostvariti statistički korisne zaključke. Međutim ipak je jedan zaključak moguće definirati. S obzirom na vremensko ograničenje izvođenja pokusa i vremensku zahtjevnost izvođenja evaluatora, evoluciju rješenja uz fiksan broj evaluacija najpouzdanije je vršiti s manjom populacijom rješenja. Razlog tome je što se u ograničenom vremenskom okviru u manjoj generaciji veći dio vremena troši na djelovanje genetskih operatora koji u konačnici i izvode samu evoluciju rješenja. Ukoliko se bira populacija s većim brojem jedinki, genetskih operatori će biti manje utjecajni jer se tada uloženo vrijeme troši upravo na evaluaciju svih jedinki u populaciji, dok manji dio vremena otpada na izvođenje operatora. Za slučaj obavljenih pokusa, prilikom evolucije postavljena je fiksna vrijednost populacije od 20 do 30 jedinki ovisno o pokretanju.

# 5. Zaključak

Optimizacija povećanja propusnosti inherentno predstavlja kompleksan kombinatorički problem te poseban izazov za evoluciju genetskim algoritmom. Iako vremenski zahtjevni, pokusi su u konačnici pokazali da je moguće generirati pravilan raspored bistabila na zadanoj kombinacijskoj mreži s ostvarenjem dvije i tri razina paralelne obrade.

Ovakav pristup optimizacije propusnosti nudi dobru i relativno jednostavnu tehniku prilagodbe mreže uz prihvatljivu cijenu vremena kojeg je potrebno uložiti. Pri tome je zgodno spomenuti da je samim postupkom evolucije i primjenom razvijenih algoritama moguće generirati dodatne informacije o samoj mreži, uključujući realna ograničenja koja mogu biti od izrazite važnosti za daljnje postupke i analizu optimizacije.

Budući da se radi o kompleksnoj memorijskoj strukturi podataka, jednostavan bitovni niz predstavlja prikladan način reprezentacije jedinke. Tako oblikovana reprezentacija omogućuje ne samo jednostavnu manipulaciju tijekom same evolucije algoritma, već i donosi nove mogućnosti potencijalnog ručnog namještanja konfiguracije mreže s ciljem smanjenja prostora pretraživanja i početnog usmjeravanja algoritma.

Daljnja poboljšanja u sklopu implementacije evaluatora bi se ponajprije fokusirala na razvoj vremenski efikasnijeg algoritma evaluacije te omogućavanje automatizirane izmjene parametara s ciljem pomaka ograničenja broja teorijski maksimalne razine paralelne obrade. Također, fokus poboljšanja teži razvoju deterministčkih algoritama za dinamičko osvježavanje puteva tijekom evaluacije te oblikovanjem prilagođenijeg genotipa i odgovarajućih genetskih operatora s ciljem što efikasnijeg i usmjerenijeg pretraživanja prostora rješenja.

Kao zaključak bitno je spomenuti da osnova izgrađena ovim radom nudi mogućnosti daljnje analize i razvoja u nadi za generiranjem boljih i efikasnijih rješenja optimizacije propusnosti kombinacijskih mreža.

# 6. Literatura

[1] Affenzeller Michael, Winkler Stephan, Wagner Stephan, Beham Andreas: „Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications“, Taylor & Francis Group, 2009.

[2] Research Institute for Nanodevice and Bio Systems, Hiroshima University, s interneta, http://www.rnbs.hiroshima-u.ac.jp/RCNS/lecture/pdf/HJM_H20/OHP_CMOS_6(H20-5-16).pdf. Preuzeto 16.05.2014.

[3] MIT, Computation Structures, 6.004 Spring 2014, s interneta, http://6004.mit.edu/Fall12/tutprobs/pipeline.html. Preuzeto 16.05.2014.

[4] Čupić Marko, „Prirodom inspirirani optimizacijski algoritmi. Metaheuristike.“, http://java.zemris.fer.hr/nastava/pioa/knjiga-0.1.2013-07-12.pdf. Preuzeto 16.05.2014.

# 7. Sažetak

U ovom radu opisana je problematika optimizacije povećanja propusnosti kombinacijskih mreža korištenjem evolucijskih algoritama, ponajprije genetskog algoritma. Sagledava se problematika samog postupka paralelizacije kombinacijskih mreža s obzirom na svojstva i pripadna ograničenja mreže. Opisani su razvijeni algoritmi korišteni za evaluaciju jedinki te navedeni prijedlozi mogućih drugačijih pristupa samom problemu. Ostvaren je programski sustav koji predstavlja evaluator genotipske reprezentacije rješenja koji na temelju zadane jedinke definiranim postupkom određuje vrijednost preslikane konfiguracije mreže. Evaluacijom testova uz varijabilne parametre dokazana je mogućnost ostvarenja rješenja zadanog problema ovim pristupom rješavanja.

**Ključne riječi:** kombinacijska mreža, propusnost, latencija, evolucijski algoritmi, genetski algoritam, optimizacija, ECF, metaheuristika.

The main issue described within this bachelor's thesis is the problem of optimizing throughput of combination networks by using evolutionary algorithms, primarily the genetic algorithm. The issue of the parallelization process of combination networks is being described with respect to the properties of given networks and the associated limitations. A description of developed algorithms used for the evaluation of individuals is given together with proposals of possible different approaches to the defined problem. A software system has been implemented for the evaluation of the genotypic representation of solutions by calculating the value of a specified configuration of a network based on a given individual. The possibility to generate solutions by using the described problem-solving approach was proved by evaluating tests with variable parameters.

**Keywords:** combination network, throughput, latency, evolutionary algorithms, genetic algorithm, optimization, ECF, metaheuristic.