Univerzitet u Beogradu

Matematički fakultet

Aleksandar Đenić

Rešavanje nekih problema zadovoljenja ograničenja primenom egzaktnih i heurističkih metoda

master rad

Beograd

2011.

Mentor: **doc. dr Miroslav Marić**
Matematički fakultet u Beogradu

Članovi komisije: **prof. dr Duško Vitas**
Matematički fakultet u Beogradu

doc. dr Filip Marić
Matematički fakultet u Beogradu

Datum odbrane: _____

Rešavanje nekih problema zadovoljenja ograničenja primenom egzaktnih i heurističkih metoda

Apstrakt

Problemi zadovoljenja ograničenja (*Constraint Satisfaction Problems, CSP*) predstavljaju matematičke probleme definisane kao skup objekata čije stanje mora da zadovoljava niz ograničenja. *CSP* je definisan kao skup promenljivih $X_1, X_2, \dots, X_n$, i skup ograničenja $C_1, C_2, \dots, C_m$. Rešenje problema predstavlja skup dodela vrednosti promenljivima iz odgovarajućih domena, $\{X_1 = v_1, X_2 = v_2, \dots, X_n = v_n\}$, koji zadovoljava sva ograničenja. U ovom radu su rešavani

- problem magičnog kvadrata,
- problem binarnih sekvenci male autokorelacije,
- problem dizajna uravnoteženih nepotpunih blokova, i
- problem turnira u kojima se sve ekipe međusobno sastaju.

Predstavljene su egzaktne i heurističke metode rešavanja, kao i analiza i poređenje rezultata. Egzaktne tehnike su predstavljene kroz kreiranje modela za *Microsoft Solver Foundation*, *IBM ILOG CPLEX* i *SAT* rešavač, a heurističke kroz rešavanje problema metaheuristikama, genetskim algoritmom i metodom promenljivih okolina (*Variable Neighborhood Search*).

Sadržaj

Poglavlje 1

Uvod

Problemi zadovoljenja ograničenja (*Constraint Satisfaction Problem, CSP*) su matematički problemi definisani kao skup objekata čije stanje mora da zadovoljava niz ograničenja.

Definicija 1.1 *CSP* $\mathcal{P}$ je trojka $\langle X, D, C\rangle$ gde je X n-torka promenljivih $\langle x_1, x_2, \dots, x_n\rangle$, D odgovarajuća n-torka domena $\langle D_1, D_2, \dots, D_n\rangle$ takva da $x_i \in D_i$ i C m-torka ograničenja $\langle C_1, C_2, \dots, C_m\rangle$.

Svako ograničenje C_i predstavlja par $\langle R_{S_i}, S_i\rangle$ gde je S_i prostor promenljivih ograničenja C_i, a R_{S_i} relacija nad promenljivama prostora S_i. Drugim rečima oraničenje C_i predstavlja podskup dekartovog proizvoda domena promenljivih iz S_i.

Definicija 1.2 *Rešenje CSP*-a $\mathcal{P}$ je n-torka $A = \langle a_1, a_2, \dots, a_n\rangle$ gde $a_i \in D_i$ i svako C_j je zadovoljeno tako što R_{S_j} sadrži projekciju n-torke A na prostor S_j.

Neki *CSP*-ovi zahtevaju rešenje za koje će data funkcija cilja biti maksimalna. Detaljniji opis formalizacije *CSP*-ova dat je u [Ros06].

U određenom zadatku može se zahtevati pronalaženje skupa svih rešenja datog problema, utvrđivanje da li je taj skup neprazan ili pronalaženje nekog rešenja, ukoliko ono postoji. Ukoliko je skup rešenja prazan *CSP* je *nezadovoljiv*.

Klasa *CSP*-ova pokriva veliki broj problema veštačke inteligencije, operacionog istraživanja, planiranja, upravljanja lancem snabdevanja, računarske lingvistike i grafova. Problem zadovoljivosti u iskaznoj logici (*Satisfiability, SAT*), problem zadovoljivosti u teoriji (*Satisfiability Modulo Theories, SMT*), i programiranje prema skupu odgovora (*Answer Set Programming, ASP*) mogu se posmatrati kao podklase *CSP*-ova.

Primer 1.1

Neka su date tri celobrojne promenljive x_1, x_2, x_3 sa svojim domenima:

- $x_1 \in \{4, 7, 2\}, x_2 \in \{3, 4, 0\}$ i $x_3 \in \{4, 3, 9\}$.

Ukoliko postoje dva ograničenja:

- $C_1: X_1 + X_2 + X_3 = 10$,
- $C_2: X_1 > X_2$,

dobijaju se dva različita rešenja:

- $x_1 = 4, x_2 = 3, x_3 = 3$,
- $x_1 = 7, x_2 = 0, x_3 = 3$.

Ukoliko je dato i treće ograničenje:

- $C_1: X_1 + X_2 + X_3 = 10$,
- $C_2: X_1 > X_2$,
- $C_3: X_2 \neq X_3$,

dobija se samo jedno moguće rešenje:

- $x_1 = 7, x_2 = 0, x_3 = 3$.

CSP-ovi se obično rešavaju tehnikama pretrage, zaključivanja ili nekom od kombinacija ove dve tehnike. Ukoliko su svi domeni D_i konačni, postoji konačan *prostor pretrage* mogućih rešenja $\Omega = D_1 \times D_2 \times \ldots \times D_n$. Tada se za svaku n-torku prostora pretrage može ispitati da li predstavlja rešenje. Ovakav pristup naziva se tehnika nabrajanja i može biti poboljšana tehnikama zaključivanja i pretrage. Tehnike pretrage sistematski pretražuju Ω, često eliminišući podprostore sa nekim nezadovoljenim ograničenjem. Tehnike zaključivanja propagiraju lokalna ograničenja i na taj način eliminišu veliki broj podprostora prostora Ω koji sigurno ne sadrže rešenje. Uspeh obe tehnike zasnovan je na činjenici da je *CSP* konjuktivan (da bi bio rešen, sva ograničenja moraju da budu zadovoljena). Osnovne tri algoritamske tehnike pretrage su pretraga sa vraćanjem (*Backtracking*), dinamičko programiranje i lokalna pretraga.

Algoritmi za rešavanje *CSP*-ova mogu biti kompletni i nekompletni. *Kompletni*, ili *egzaktni*, algoritmi garantuju nalazak rešenja ukoliko postoji. Mogu se koristiti za pokazivanje da problem nema rešenje i za dokazivanje pronalaska optimalnog rešenja. Pretraga sa vraćanjem i dinamičko programiranje su u opštem slučaju primeri kompletnih algoritama. *Nekompletni*, ili *heuristički*, algoritmi ne mogu da dokažu da *CSP* nema rešenje ili da je pronađeno optimalno rešenje. Ipak nekompletni algoritmi su često efikasniji u nalaženju rešenja ukoliko ono postoji i mogu biti iskorišćeni za pronalazak aproksimacije optimalnog rešenja. Algoritmi lokalne pretrage su primer nekompletnih algoritama.

U tehnici pretrage sa vraćanjem grade se parcijalna rešenja, dodeljivanjem vrednosti promenljivima, dok se ne dostigne dodela nakon koje se parcijalno rešenje ne moze dosledno proširiti. Tada se poništava zadnji izbor i bira novi. To se izvršava na sistematski način koji garantuje ispitivanje svake mogućnosti. Tehnika pretrage sa vraćanjem poboljšava jednostavnu tehniku nabrajanja i testiranje svih kandidata za rešenje grubom silom tako što proverava zadovoljivost ograničenja pri svakom novom izboru, umesto da čeka generisanje kompletnog kandidata za rešenje. Proces pretrage sa vraćanjem često se predstavlja stablom pretrage, u kome svaki čvor (ispod korena) predstavlja izbor vrednosti za jednu promenljivu i svaka grana kandidata za parcijalno rešenje. Otkriće da parcijalno rešenje ne može biti prošireno odgovara odsecanju podstabla iz razmatranja.

Dinamičko programiranje predstavlja metodu za rešavanje složenih problema svođenjem na jednostavnije podprobleme. Da bi se rešio dati problem, rešavaju se različiti podproblemi i na osnovu njihovih rešenja se dobija celokupno rešenje. Često su mnogi podproblemi isti i nastoji se da se svaki podproblem računa samo jednom i tako smanji količina računanja. Da bi problemi mogli da se rešavaju dinamičkim programiranjem moraju da zadovolje osobine optimalne podstrukture i preklapajućih podproblema. Optimalna podstruktura znači da se rešenje datog problema može

dobiti računanjem rešenja podproblema. Optimalne podstrukture su obično rekurzivno opisane. Preklapajući podproblemi podrazumevaju da prostor podproblema treba da bude mali i da rekurzivni algoritam koji rešava sam problem treba da rešava i podprobleme.

Lokalna pretraga je jedna od osnovnih tehnika za rešavanje teško izračunljivih kombinatornih problema, uključujući i *CSP*-ove. Uprkos znatnom napretku egzaktnih algoritama, lokalna pretraga u mnogim slučajevima predstavlja jedini način za rešavanje velikih i složenih problema. Osnovna ideja lokalne pretrage je kreiranje slučajno generisanog kandidata rešenja koji se iterativno poboljšava uglavnom malim modifikacijama. Različite metode lokalne pretrage variraju u zavisnosti od načina na koji se postižu poboljšanja i od načina prevazilaženja situacija kada ne postoji direktno poboljšanje. *Stohastičke* metode lokalne pretrage koriste proizvoljnost da obezbede da proces pretrage ne stagnira sa nezadovoljavajućim kandidatom rešenja.

Ideja tehnika zaključivanja je da propagiraju lokalna ograničenja. Metoda pretrage sa vraćanjem vrlo često dovodi do nepotrebne pretrage. Nepotrebna pretraga je ponovljeno istraživanje podstabala koja se razlikuju u dodeli promenljivih beznačajnim za neuspeh podstabla. Uglavnom takvih beznačajnih dodela ima eksponencijalno mnogo. Nepotrebna pretraga je često najznačajniji faktor u vremenu izvršavanja pretrage sa vraćanjem. Do poboljšanja dolazi propagiranjem lokalnih ograničenja. Propagiranje ograničenja smanjuje domene promenljivih, pojačava početna ograničenja i kreira nova. To vodi do smanjenja prostora pretrage i čini problem jednostavnijim.

U biblioteci problema zadovoljenja ograničenja ([WSCsp]) izdvojeno je pedeset različitih problema razvrstanih u sedam kategorija pri čemu jedan problem može biti sadržan u više kategorija. Izdvojene kategorije su:

1. Planiranje (*Scheduling*),
2. Dizajn, konfiguracija i dijagnoza (*Design, Configuration and Diagnosis*),
3. Pakovanje i particionisanje (*Bin Packing and Partitioning*),
4. Dodela frekvencije (*Frequency Assignment*),
5. Kombinatorna matematika (*Combinatorial Mathematics*),
6. Igre i zagonetke (*Games and Puzzles*), i
7. Bioinformatika (*Bioinformatics*).

U sklopu ovog rada rešavani su:

- problem magičnog kvadrata (kategorije 5 i 6),
- problem binarnih sekvenci male autokorelacije (kategorije 4 i 5),
- problem dizajna uravnoteženih nepotpunih blokova (kategorije 2 i 5), i
- problem turnira u kojima se sve ekipe međusobno sastaju (kategorija 1).

U radu su problemi rešavani egzaktnim i heurističkim metodama rešavanja, a zatim je izvršena analiza dobijenih rezultata i njihovo poređenje.

Poglavlje 2

Tehnike rešavanja

Neka su dati *prostor rešenja* S, *funkcija cilja* $f: S \to \mathbb{R}$ i *dopustiv skup* $X \subseteq S$, koji je zadat nizom ograničenja nad prostorom rešenja S. Problem *matematičke optimizacije* ili *matematičkog programiranja* je definisan kao pronalaženje

$$\min \{f(x) \mid x \in X, \}.$$

U praksi se uglavnom koristi prostor rešenja koji je podskup od $\mathbb{R}^n$ i racionalna funkcija cilja.

Problem je *kombinatorni optimizacioni* ukoliko je S konačan skup. Ukoliko je $S = \mathbb{R}^n$ problem je *neprekidno optimizacioni*. Rešenje $x^* \in X$ je *optimalno* ako

$$f(x^*) \leq f(x), \forall x \in X.$$

Mnoge klase problema mogu da se posmatraju kao podklase problema matematičke optimizacije.

Kod problema *konveksnog programiranja* je potrebno da funkcija cilja bude konveksna i da dopustiv skup bude zadat konveksnim jednačinama i nejednačinama. Problemi *linearnog programiranja* su problemi konveksnog programiranja u kojima je funkcija cilja linerarna i dopustiv skup zadat linearnim jednačinama i nejednačinama. Ukoliko se u problemu linearnog programiranja zahteva da sve promenljive budu celobrojne onda problem nazivamo problem *celobrojnog programiranja*, a ukoliko se zahteva da samo neke promenljive budu celobrojne onda problem nazivamo problem *mešanog celobrojnog programiranja*.

Kod problema *kvadratnog programiranja* funkcija cilja osim linearnih može imati i kvadratne izraze, dok dopustiv skup mora biti zadat linearnim jednačinama i nejednačinama. Problemi kvadratnog programiranja mogu biti konveksni i nekonveksni u zavisnosti od funkcije cilja i dopustivog skupa.

Kod *CSP*-ova funkcija cilja je konstantna. Klase *SAT* i *SMT* problema su podklase klase *CSP*. U *SAT* problemima potrebno je odrediti da li se promenljivama date iskazne formule mogu dodeliti takve vrednosti, da formula daje tačan iskaz. *SMT* problemi predstavljaju proširenje *SAT* problema i kod njih je potrebno odrediti da li je data iskazna formula zadovoljiva u teoriji.

Egzaktni algoritmi za rešavanje optimizacionih problema, ukoliko postoje, pronalaze optimalno rešenje, zajedno sa dokazom njegove optimalnosti, ili pokazuju da ne postoji dopustivo rešenje. Međutim u praksi nije uvek moguće koristiti egzaktne algoritme jer je za velike probleme potrebno dosta vremena za rešavanje. U takvim slučajevima se često koriste heuristike. Opšte heurističke tehnike koje se koriste za rešavanje različitih klasa problema nazivaju se *metaheuristike*.

2.1 Egzaktne tehnike rešavanja

2.1.1 Metoda pretrage sa vraćanjem

Metoda pretrage sa vraćanjem prvi put je formalno uvedena 1965. u [Gol65]. Primenjuje se samo na problemima koji podržavaju koncept parcijalnih rešenja, za koja se može proveriti da li je moguće da budu dopunjena do dopustivog rešenja. Proces pretrage metode je opisan u poglavlju 1 ovog rada. Pretraga sa vraćanjem se primenjuje za rešavanje *CSP*-ova, u nekim tehnikama parsiranja i čini osnovu za izvršavanje logičkih programskih jezika, kao što je Prolog.

2.1.2 Simpleks metoda

Simpleks metoda je metoda za rešavanje problema linearnog programiranja. Ova metoda, koju ju je 1947. osmislio Džordž Dancig ([Dan63]), testira susedna temena dopustivog skupa tako da se za svaku novu tačku funkcija cilja poboljša ili ostane nepromenjena. Dopustiv skup je predstavljen kao višedimenzioni mnogoguao. Simpleks metoda je vrlo efikasna u praksi i za određene distribucije slučajnog ulaza konvergira u polinomijalnom vremenu ([Noc99] i [For02]). Ipak, najgora vremenska složenost simpleks metode je eksponencijalna, kao što je pokazano sa specijalno konstruisanim primerima u [Kle72].

2.1.3 Metode unutrašnje tačke

Metode unutrašnje tačke su metode koje rešavaju probleme linearnog i nelinearnog programiranja. Ove metode konstruišu niz dopustivih tačaka koje konvergiraju ka rešenju. Dopustiv skup je predstavljen kao višedimenzioni mnogoguao, a tačke koje se testiraju se nalaze u njegovoj unutrašnjosti za razliku od simpleks metode gde se nalaze na temenima. Metode unutrašnje tačke imaju polinomijalnu vremensku složenost. Karmarkar je u svom radu 1984. ([Kar84]) započeo istraživanje metoda unutrašnje tačke. U praksi, jedna od najboljih metoda unutrašnje tačke je prediktor-korektor ([Meh92]) metoda, koja je konkurentna simpleks metodi, pogotovo za probleme velikih dimenzija.

2.2 Metaheurističke tehnike rešavanja

2.2.1 Genetski algoritmi

Genetski algoritmi su metaheuristike koje imitiraju proces prirodne evolucije. Tvorac genetskih algoritama je Džon Holand, razvio ih je 60-ih i 70-ih godina prošlog veka sa grupom kolega i studenata, na Univerzitetu Mičigen ([Hol75]). Holandova originalna ideja nije bila da dizajnira algoritme koji rešavanju specifične probleme, već da proučava prirodni fenomen adaptacije simulacijom u računarskom sistemu. Daljim razvitkom Holandove teorije, genetski algoritmi su se pokazali kao veoma pogodni za rešavanje problema pretrage i optimizacije.

Osnovni biološki pojmovi vezani za evoluciju su:

- **Ćelija**
  Ćelija predstavlja osnovnu gradivnu jedinicu živih bića. Ćelijsko jedro sadrži hromozome jedinke.

- **Hromozomi**
  Sve genetske informacije su sačuvane u hromozomima jedinke. Hromozomi su podeljeni na više delova koji se nazivaju geni. Geni određuju osobine jedinke. Skup svih gena jedne vrste naziva se genom. U genetskim algoritmima je svaka jedinka predstavljena svojim hromozom, koji je u potpunosti određuje.

- **Reprodukcija**
  Reprodukcija je proces u kome jedinke, roditelji, proizvode svoje potomstvo. U genetskim algoritmima se pod reprodkukcijom uglavnom podrazumeva ukrštanje jedinki, gde dva roditelja ukrštanjem daju potomstvo.

- **Mutacija**
  Mutacije su iznenadne i spontane promene na genomu jedinke. U genetskim algoritmima mutacije uglavnom predstavljaju male slučajne promene na hromozomu jedinke.

- **Selekcija**
  Selekcija je proces biranja jedinki za reprodukciju. Prilagođenije jedinke imaju veću šansu za opstanak i reprodukciju. U genetskim algoritmima prilagođenost predstavlja verovatnoću da će jedinka dati potomstvo, a meri se kvalitetom jedinke u datoj populaciji jedinki.

Populacija jedninki je osnovni pojam genetskog algoritma. Jedinka predstavlja jednu tačku iz prostora rešenja i predstavljena je hromozomom. Najčešće je kodirana nizom bitova. Na početku algoritma zadaje se inicijalna populacija, uglavnom generisana slučajnim izborom. Zatim se iterativno, sve dok se ne ispune uslovi završetka algoritma, izvršavaju sledeći koraci:

- Selekcija: Biraju se pojedinci za reprodukciju. Izbor se vrši slučajno. Verovatnoća izbora jedinke je srazmerna njenoj prilagođenosti. Prilagođenije jedinke imaju veću šansu da daju potomstvo.
- Reprodukcija: Potomstvo se kreira ukrštanjem izabranih jedinki, nakon čega dolazi do mutacije.
- Ocena: Računa se prilagođenost novih jedinki.
- Zamena: Jedinke iz stare populacije se zamenjuju novim.

Prilagođenost zavisi od funkcije cilja, ali i od raznovrsnosti populacije. Da bi populacija ostala raznovrsna, ukoliko dođe do znatnog ponavljanja jedinki, ponovljenim jedinkama se prilagođenost postavlja na vrednost koju imaju najlošije jedinke u populaciji.

Kao operatori selekcije najčešće se koriste rulet selekcija i turnirska selekcija. Kod proste rulet selekcije verovatnoća odabira jedinke je direktno proporcionalna vrednosti njene funkcije cilja. Jedno od unapređenja je da se umesto vrednosti funkcije cilja koristi vrednost prilagođenosti jedinke. Kod turnirske selekcije se organizuju turniri u kojima se jedinke nadmeću za učešće u ukrštanju i davanju potomstva. Veličina turnira je broj jedinki koje učestvuju na turniru. Na slučajan način se iz populacije biraju jedinke za učešće u turniru i pobeđuje najprilagođenija jedinka. Jedno od unapređenja turnirske selekcije je fino gradirana turnirska selekcija gde veličina turnira nije fiksna nego se koristi realni parametar koji označava očekivanu prosečnu veličinu turnira.

U genetskim algoritmima ne dolazi uvek do ukrštanja jedinki. U zavisnosti od parametra koji predstavlja verovatnoću ukrštanja, nekad dolazi do ukrštanja, a nekad potomci ostaju identični roditeljima. Najčešće se koriste jednopoziciono, dvopoziciono, višepoziciono ili uniformno ukrštanje. Kod jednopozicionog ukrštanja se na slučajan način bira broj k koji predstavlja tačku ukrštanja jedinki. Ukoliko su $x_1x_2 \dots x_n$ i $y_1y_2 \dots y_n$ geni roditelja, potomci će imati gene $x_1x_2 \dots x_k y_{k+1} \dots y_n$ i $y_1y_2 \dots y_k x_{k+1} \dots x_n$. Operator dvopozicionog ukrštanja koristi dve tačke ukrštanja i roditelji razmenjuju genetske kodove između tih tačaka. Kod uniformnog ukrštanja se na slučajan način generiše binarni niz iste dužine kao genetski kod. Roditelji zatim razmenjuju gene na svim pozicijama na kojima clanovi niza imaju vrednost 0, dok na mestima gde članovi niza uzimaju vrednost 1 roditelji zadržavaju svoje gene.

Mutacijom dolazi do promena na jedinkama populacije. Verovatnoća mutacije je dosta mala. Ukoliko je određeni gen jednak u svim jedinkama populacije, verovatnoća mutacije tog gena se povećava. Ukoliko se kod jedinki kodiranih bitovima na istom mestu u celoj populaciji nalazi isti bit onda se on naziva zaleđen bit. Verovatnoća njegove mutacije se množi sa faktorom zaleđenog bita.

U konkretnim implementacijama se retko cela populacija zamenjuje novom. Uglavnom najbolje jedinke ostaju da žive i u sledećoj generaciji, a ukrštanjem se ne kreira cela populacija nego samo deo nje.

Genetski algoritam se izvršava iterativno, sa idejom da će generacije postajati vremenom sve kvalitetnije, a najbolja jedinka u populaciji u jednom trenutku dati tačno rešenje problema. Kriterijumi zaustavljanja mogu biti, na primer, maksimalni broj generacija, maksimalni broj ponavljanja najbolje jedinke, stepen sličnosti jedinki, ograničeno vreme izvršavanja ili dostizanje optimalnog rešenja.

Detaljnije opis genetskih algoritama dat je u [Siv08], [Mel98] i [Mar08].

2.2.2 Metoda promenljivih okolina

Metoda promenljivih okolina (*Variable Neighborhood Search, VNS*) je metaheuristika koja rešava probleme kombinatorne i globalne optimizacije. Zasniva se na sistematskim promenama okolina, kako u fazi traženja lokalnog minimuma, tako i u fazi izlaska iz njega. Metoda je nastala 90-ih godina prošlog veka, a tvorci su Nenad Mladenović i Pier Hansen ([Mla97]).

Neka su dati prostor rešenja S, funkcija cilja $f: S \to \mathbb{R}$ i dopustiv skup $X \subseteq S$. Označimo sa $N_k (k = 1 \dots k_{max})$ niz unapred odabranih struktura okolina prostora S i sa $N_k(x)$ skup rešenja u k-

toj okolini tačke $x, x \in S$. Struktura okoline predstavlja oblik i vrstu okoline i može biti izračunata uz pomoć jedne ili više metrika definisanih na prostoru rešenja S. Optimalno rešenje x_{opt} (ili globalni minimum) je dopustivo rešenje u kom je dostignut minimum. Ako ne postoji rešenje $x \in N_k(x') \subseteq X$, takvo da je $f(x) < f(x')$, onda $x' \in X$ nazivamo lokalnim minimumom u okolini $N_k(x')$.

VNS je zasnovan na tri jednostavne činjenice:

1. lokalni minimum date okoline nije neophodno i lokalni minimum druge okoline koja ga sadrži;
2. globalni minimum je lokalni minimum, kada se posmatraju sve okoline;
3. u mnogim problemima, lokalni minimumi jedne ili više susednih okolina su relativno međusobno blizu.

Poslednja, empirijska činjenica podrazumeva da lokalni minimum često pruža neke informacije o globalnom minimumu. Moguće je, na primer, da više promenljivih uzima istu vrednost u oba rešenja. Međutim obično se ne zna koje su to promenljive. Pošto one ne mogu biti otkrivene unapred, potrebno je sprovesti organizovanu studiju okoline lokalnog minimuma, sve dok se ne nađe bolje rešenje.

VNS algoritam se sastoji od uprošćenog i osnovnog *VNS* algoritma. Uprošćeni *VNS* algoritam se prvi izvršava, i njegov rezultat se najčešće koristi kao početno rešenje osnovnog algoritma. Osnovni *VNS* u izvršavanju koristi algoritam lokalne pretrage.

Algoritam uprošćenog *VNS*-a:

Inicijalizacija. Izabrati niz struktura okolina N_k, $k = 1 \dots k_{max}$, koje će biti korišćene u pretrazi; izabrati inicijalno rešenje x; izabrati kriterijum zaustavljanja;
Ponavljati sledeće korake sve dok se ne zadovolji kriterijum zaustavljanja:

1. *Postaviti $k \leftarrow 1$;*
2. *Ponavljati sledeće korake sve dok je $k \neq k_{max}$:*
   a. *Mešanje. Izabrati proizvoljnu tačku x' u k-toj okolini tačke x ($x' \in N_k(x)$);*
   b. *Pomeriti se ili ne. Ukoliko je $f(x') < f(x)$ tada $x \leftarrow x'$ i $k \leftarrow 1$; inače $k \leftarrow k + 1$;*

Algoritam lokalne pretrage:

Inicijalizacija. Izabrati niz struktura okolina N_k, $k = 1 \dots k_{max}$, koje će biti korišćene u pretrazi; izabrati inicijalno rešenje x ili primeniti algoritam za prosleđenu tačku x;
Ponavljati sledeće korake sve dok se se dobijaju unapređenja:

1. *Postaviti $k \leftarrow 1$;*
2. *Ponavljati sledeće korake sve dok je $k \neq k_{max}$:*
    a. *Istraživanje okoline. Pretragom kroz sve tačke okoline $N_k(x)$ pronaći tačku x' takvu da je vrednost $f(x')$ minimalna;*
    b. *Pomeriti se ili ne. Ukoliko je $f(x') < f(x)$ tada $x \leftarrow x'$ i $k \leftarrow 1$; inače $k \leftarrow k + 1$;*

Algoritam osnovnog *VNS*-a:

Inicijalizacija. Izabrati niz struktura okolina N_k, $k = 1 \dots k_{max}$, koje će biti korišćene u pretrazi; izabrati inicijalno rešenje x ili primeniti algoritam za prosleđenu tačku x; izabrati kriterijum zaustavljanja;
Ponavljati sledeće korake sve dok se ne zadovolji kriterijum zaustavljanja:

1. *Postaviti $k \leftarrow 1$;*
2. *Ponavljati sledeće korake sve dok je $k \neq k_{max}$:*
    a. *Mešanje. Izabrati proizvoljnu tačku x' u k-toj okolini tačke x ($x' \in N_k(x)$);*
    b. *Lokalna pretraga. Primeniti lokalnu pretragu na tačku x' kao početno rešenje; ozančiti sa x'' dobijeni lokalni minimum;*
    c. *Pomeriti se ili ne. Ukoliko je $f(x'') < f(x)$ tada $x \leftarrow x''$ i $k \leftarrow 1$; inače $k \leftarrow k + 1$;*

Kriterijumi zaustavljanja mogu biti, na primer, potrošeno procesorsko vreme, maksimalan broj iteracija ili maksimalan broj iteracija između dva unapređenja.

Detaljniji opis metode promenljivih okolina dat je u [Gen10], [Han01] i [Mla97].

Poglavlje 3

Rešavači

3.1 Microsoft Solver Foundation

Microsoft Solver Foundation (*MSF*) je biblioteka funkcija za matematičko programiranje, modelovanje i optimizaciju, razvijena od strane *Microsoft DevLabs*-a ([WSMsfA]). U industriji se koristi kao pomoć kompanijama za povećavanje efikasnosti, maksimizaciju profita i upravljanje rizikom. Za rešavanje koristi metaheuristike, pretrage i tehnike kombinatorne optimizacije. *MSF* može da se koristi kao biblioteka u svakom programu koji se izvršava na *.NET Framework* platformi ([WSNet]). Postoje tri verzije biblioteke *Express*, *Standard* i *Enterprise*. Razlikuju se u podržanoj hardverskoj podršci i broju promenljivih i ograničenja koji se mogu definisati u jednom modelu. *Express* verzija je besplatna, dok su *Standard* i *Enterprise* komercijalne. Za potrebe ovog rada korišćena je *MSF Standard 3.1* verzija.

Problem se definiše preko modela, zadavanjem promenljivih sa njihovim domenima, zadavanjem ograničenja, ciljeva i podataka koji se obrađuju. Rešenje se dobija rešavanjem modela nekim od rešavača:

- Simpleks rešavač – rešava probleme mešanog celobrojnog programiranja,
- Rešavač koji koristi metodu unutrašnje tačke – rešava probleme linearnog i kvadratnog konveksnog programiranja,
- *CSP* rešavač – pronalazi rešenja problema zadovoljenja ograničenja,
- Kompaktan kvazi Njutnov rešavač – pronalazi lokalni minimum ili maksimum neograničene nelinearne funkcije,
- *SAT* rešavač – rešavač koji određuje zadovoljivost iskazne formule.

Za potrebe ovog rada korišćen je *CSP* rešavač. On u rešavanju koristi tehnike pretrage sa vraćanjem, lokalnu pretragu i metaheuristike. Detaljniji opis *MSF*-a dat je u [WSMsf].

3.2 IBM ILOG CPLEX

IBM ILOG CPLEX (*CPLEX*) je optimizacioni softverski paket za rešavanje problema matematičkog programiranja. *CPLEX* rešava probleme linearnog i kvadratnog programiranja. Za rešavanje koristi simpleks metodu i metode unutrašnje tačke. Problem se definiše zadavanjem matematičkog modela. Pruža interfejse ka C++, C#, Java, Python i Matlab programskim jezicima. *CPLEX* je komercijalan softver i za potrebe ovog rada korišćena je verzija *Teaching Edition 12.1*.

U radu je *CPLEX* korišćen kao primer linearnog rešavača za rešavanje *CSP*-ova. Detaljniji opis *CPLEX*-a dat je u [WSCpl].

3.3 Picosat

Kao *SAT* rešavač korišćen je *picosat*, razvojen od strane instituta za formalne modele i verifikaciju, u Lincu ([WSPic]). *Picosat* je otvorenog koda i slobodan za korišćenje. Za potrebe rada korišćena je verzija *picosat 936.*

3.4 GA Framework

Za rešavanje problema genetskim algoritmom korišćena biblioteka otvorenog koda *GA Framework*([WSGaf]). Biblioteka ima podršku za rad sa populacijom jedinki, funkcijom prilagođenosti, za razne vrste selekcije i ukrštanja i za mutaciju. U slučaju konkretnog problema potrebno je samo definisati osobine algoritma specifične problemu. Biblioteka podržava višeprocesorsko izvršavanje algoritma. U njenoj implementaciji se u iterativnim koracima izvršava:

1. sortiranje jedinki prema funkciji prilagođenosti;
2. selekcija jedinki za reprodukciju;
3. ukrštanje;
4. zamena najlošijih jedinki novim;
5. mutacija.

3.5 VNS

Za rešavanje problema *VNS* tehnikom rešavanja nije korišćena već implementirana biblioteka, već je za svaki problem implementiran specifičan *VNS* algoritam koji odgovara problemu. Osobine implementacije date su u sledećem poglavlju rada, u odeljcima vezanim za same probleme.

Poglavlje 4

Problemi

Za rešavanje svih problema genetskim algoritmom u ovom radu korišćena je populacija od 150 jedinki. Inicijalna populacija je slučajno generisana. Za funkciju prilagođenosti je korišćena normirana funkcija cilja na intervalu [0,1] (najbolja jedinka ima prilagođenost 1, najlošija 0). Ukoliko broj identičnih jedinki prelazi maksimalan broj dozvoljenih identičnih jedinki, jedinkama koje su višak se prilagođenost postavlja na nulu. Isto tako, ako broj jedinki sa istom vrednošću funkcije cilja prelazi maksimalan broj jedinki sa istom vrednošću funkcije cilja, višku se prilagođenost postavlja na nulu. Za selekciju je korišćena fino gradirana turnirska selekcija. Bira se 50 jedinki roditelja, koje ukrštanjem daju potomstvo, 50 novih jedinki. Prostor rešenja, funkcija cilja, kodiranje jedinke i operatori ukrštanja i mutacije su zavisni od problema koji se rešava, i biće opisani za svaki problem posebno.

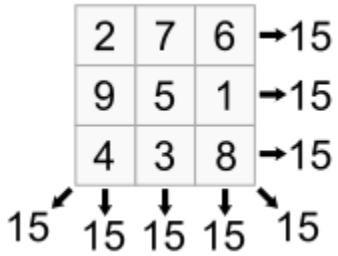
Za rešavanje svih problema *VNS* tehnikom u ovom radu korišćeno je da je uslov zaustavljanja uprošćenog *VNS*-a hiljadu izvršenih iteracija. Inicijalno rešenje za uprošćen *VNS* je slučajno izabrano, a inicijalno rešenje za osnovni *VNS* je rezultat uprošćenog *VNS*-a. Korišćen je isti niz struktura okolina u algoritmima uprošćenog *VNS*-a, lokane pretrage i osnovnog *VNS*-a.

4.1 Magični kvadrat

Magični kvadrat (*Magic Square, MS*) dimenzije n je matrica dimenzije $n \times n$ koja sadrži sve brojeve od 1 do n^2. Brojevi su tako raspoređeni da su sume svakog reda, kolone i dijagonale u kvadratu jednake. Problemi vezani za magični kvadrat su:

1. nalaženje jednog kvadrata date dimenzije,
2. pronalaženje svih kvadrata date dimenzije.

U ovom radu analiziran je prvi navedeni problem. Primer magičnog kvadrata dat na slici 1.



Slika 1. Magični kvadrat – 3 x 3

4.1.1 Microsoft solver foundation

Model magičnog kvadrata dimenzije n definisan za *MSF CSP* rešavač sadrži n^2 promenljivih, $x_{11}, x_{12}, \dots, x_{nn}$, gde svaka promenljiva predstavlja odgovarajuću vrednost u kvadratu. Svaka promenljiva uzima vrednost iz domena

$$x_{11}, x_{12}, \dots, x_{nn} \in \{1, 2, \dots, n^2\}.$$

Prvo ograničenje koje se uvodi je da sve promenljive moraju da budu međusobno različite. Ovaj tip modela sadrži ugrađeno odraničenje

$$Unequal(x_{11}, x_{12}, \dots, x_{nn}).$$

Zadaje se promenljiva b koja predstavlja vrednost potrebne sume svakog reda, kolone i dijagonale,

$$b = \frac{n \cdot (n \cdot n + 1)}{2}.$$

Zatim se dodaju ograničenja:

- suma svakog reda je b
  $$\sum_{j=1}^{n} x_{ij} = b, \qquad i \in \{1,2, \dots, n\},$$
- suma svake kolone je b
  $$\sum_{j=1}^{n} x_{ji} = b, \qquad i \in \{1,2, \dots, n\},$$
- suma svake dijagonale je b
  $$\sum_{i=1}^{n} x_{ii} = b,$$
  $$\sum_{i=1}^{n} x_{i(n-i+1)} = b.$$

Prilikom rešavanja ovog modela prvi put kada rešavač pronađe rešenje koje zadovoljava sva ograničenja, staje sa radom i vraća rezultat.

4.1.2 CPLEX

Model magičnog kvadrata dimenzije n definisan za *CPLEX* linearni rešavač sadrži n^2 promenljivih, sa domenom:

$$x_{11}, x_{12}, \dots, x_{nn} \in \{1, 2, \dots, n^2\},$$

slično *MSF CSP* rešavaču. U ovom modelu ne postoji ograničenje koje opisuje različitost promenljivih, pa se ona uvodi preko niza ograničenja:

$$|x_{ij} - x_{lk}| > 0, \qquad (i,j) \neq (l,k).$$

Ovakav način zadavanja ograničenja nije efikasan jer se za magični kvadrat dimenzije n zadaje $\frac{n^2 \cdot (n^2-1)}{2}$ različitih ograničenja. Zadaje se promenljiva b koja predstavlja vrednost potrebne sume svakog reda, kolone i dijagonale,

$$b = \frac{n \cdot (n \cdot n + 1)}{2}.$$

Model zahteva postojanje funkcije cilja koju je potrebno minimizovati ili maksimizovati. Iz tog razloga uvodi se pojam greške. Odstupanje zbira svake vrste, kolone i dijagonale od tražene vrednosti doprinosi funkciji greške. U trenutku kada su sva odstupanja jednaka nuli, pronađen je magični kvadrat. Funkcije grešaka po vrstama, kolonama i dijagonalama su zadate pomoćnim promenljivima e_1, e_1 i e_3, respektivno:

- ukupna greška po vrstama
  $e_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} |x_{ij} - b|,$
- ukupna greška po kolonama
  $e_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} |x_{ji} - b|,$
- greška dijagonala
  $e_3 = \left| \sum_{i=1}^{n} x_{ii} - b \right| + \left| \sum_{i=1}^{n} x_{i(n-i+1)} - b \right|.$

Rešavač minimizuje funkciju greške

$$f(x_{11}, x_{12}, \dots, x_{nn}) = e_1 + e_2 + e_3.$$

Kada rešavač naiđe na tačno rešenje ovog modela (ono koje nema grešku), računanje se nastavlja u cilju dokazivanja optimalnosti rešenja. Preciznije, algoritam proverava da možda ne postoji neko bolje rešenje koje će dati manju funkciju cilja. Iz ovog razloga, kao i zbog prevelikog broja zadatih ograničenja da bi se opisala različitost promenljivih, linearni rešavači nisu pogodni za dat tip problema.

4.1.3 Genetski algoritam

Kao prostor rešenja je korišćen skup S svih permutacija brojeva $\{1, 2, \dots, n^2\}$. Svaka tačka iz skupa S jednoznačno određuje matricu dimenzije $n \times n$, koja se dobija zapisivanjem brojeva redom po vrstama matrice. Funkcija cilja je definisana kao

$$f: S \to \mathbb{Z}$$

i ima vrednost sume odstupanja zbira svake vrste, kolone i dijagonale matrice od tražene vrednosti u magičnom kvadratu.

Potrebno je minimizovati funkciju cilja, odnosno naći onu permutaciju za koju je greška jednaka nuli. Ta permutacija predstavlja magični kvadrat dimenzije $n \times n$.

Svaka jedinka u populaciji kodira jednu permutaciju skupa S. Hromozom se sastoji od niza brojeva dužine n^2 koji predstavljaju traženu permutaciju.

Proces ukrštanja je definisan na sledeći način:

- Sa verovatnoćom 0.6 će jedan roditelj dati jednog potomka, drugi roditelj drugog (bez pravog ukrštanja), tako što će se zameniti dva slučajno odabrana broja u hromozomu.
- Sa verovatnocom 0.2 će oba potomka na mestima na kojima se roditelji poklapaju dobiti njihove vrednosti, a ostale vrednosti će biti slučajno raspoređene, tako da se očuva permutacija.
- sa verovatnoćom 0.2 će ćerka dobiti deo hromozoma majke i deo hromozoma oca, a sin drugi deo hromozoma majke i oca. U ovom slučaju može da se dobije nevalidna permutacija (ponavljanje nekih brojeva, a da drugi brojevi nisu uključeni) i neophodno je da se izvrsi validacija koja će popraviti jedinku.

Mutacija je definisana kao permutacija dva broja u hromozomu.

Korišćeni parametri su:

- maksimalan broj identičnih jedinki – 2,
- maksimalan broj jedinki sa istom vrednošću funkcije cilja – 20,
- očekivana prosečna veličina turnira – 5.2,
- verovatnoća ukrštanja – 0.8,
- verovatnoća mutacije – $0.5 \,/\, n^2$,
- faktor zaleđenog bita – 4.

4.1.4 Metoda promenljivih okolina

Prostor rešenja i funkcija cilja, redom S i f, koji su korišćeni u metodi promenljivih okolina su identični onima koji se koriste u genetskom algoritmu. Potrebno je minimizovati funkciju cilja, odnosno naći onu permutaciju za koju je greška jednaka nuli. Ta permutacija predstavlja magični kvadrat dimenzije $n \times n$.

Kao prva okolina date tačke x iz prostora rešenja koristi se skup tačaka koji se od tačke x razlikuje u tačno dva broja. Zamenom dva broja u datoj permutaciji se od tačke x dobija tačka x', i važi $x' \in N_1(x)$. Tačka iz druge okoline se dobija sa dve zamene, tačka iz k-te okoline sa k zamena.

Nakon zamene dva broja u permutaciji potrebno je izračunati novu vrednost funkcije cilja. Računanje vrednosti funkcije cilja se može optimizovati korišćenjem podatka da se nova tačka nalazi u okolini stare. Svakoj tački je pridružena informacija koja govori kolika je greška svakog

reda, kolone i dijagonale, kao i kolika je ukupna trenutna greška. Vrednosti u nizu grešaka redova gr i kolona gk kao i greške dijagonala gd_1, gd_2 mogu biti i pozitivne i negativne, u zavisnosti da li je suma veća ili manja od željene sume. Tada je ukupna greška g jednaka zbiru apsolutnih vrednosti svih grešaka po redovima, kolonama i dijagonalama.

Ukoliko se tačka x_{ij} menja sa tačkom x_{lk} onda se:

- od ukupne greške oduzima greška i-tog reda: $g = g - |gr_i|$,
- računa nova greška i-tog reda: $gr_i = gr_i - x_{ij} + x_{lk}$,
- ukupnoj grešci dodaje greška i-tog reda: $g = g + |gr_i|$,
- od ukupne greške oduzima greška l-tog reda: $g = g - |gr_l|$,
- računa nova greška l-tog reda: $gr_l = gr_l + x_{ij} - x_{lk}$,
- ukupnoj grešci dodaje greška l-tog reda: $g = g + |gr_l|$.

Na sličan način se manjaju greške kolona i dijagonala. Na taj način se funkcija cilja računa u vremenu $O(1)$, što u značajnoj meri poboljšava brzinu izračunavanja.

U algoritmu mešanja se izvršava k uzastopnih zamena parova tačaka u permutaciji. Broj koji je jednom bio zamenjen ne može u istoj iteraciji opet da se menja. Zamena je definisana na sledeci način:

- Sa verovatnoćom 0.9 se na slučajan način izabere broj u permutaciji za zamenu, i onda se traži najbolji koji će zameniti mesto sa njim.
- Sa verovatnoćom 0.1 se oba broja za zamenu biraju slučajnim izborom.

Korišćeni parametri su:

- u uprošćenom *VNS*-u $k_{max} = 1$,
- u lokalnoj pretrazi $k_{max} = 1$,
- u osnovnom *VNS*-u $k_{max} = n$.

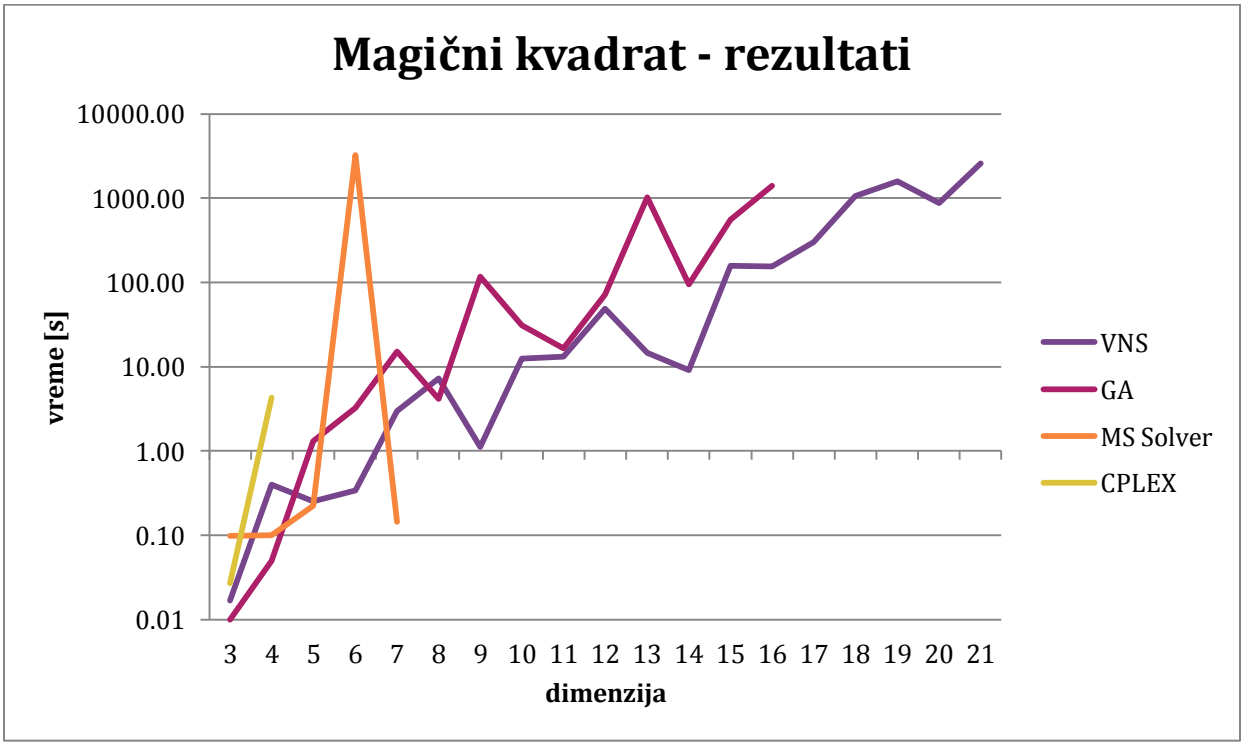
4.1.5 Rezultati

Sve četiri pomenute tehnike rešavanja su testirane za rešavanje problema magičnog kvadrata dimenzija od 3 do 21. Svi testovi su vršeni na računarima sa *Intel Core i7 860* procesorima i svaka instanca je puštana da radi do sat vremena. *CPLEX* rešavač je rešio dve najmanje instance, *MSF CSP* rešavač je rešio prvih pet instanci, genetski algoritam četrnaest, metoda promenljivih okolina devetnaest.

U tabeli 1 dati su svi dobijeni rezultati a njihov grafički prikaz dat je na grafikonu 1.

Dimenzija	VNS [s]	GA [s]	MSF [s]	CPLEX [s]
3	0.02	0.01	0.10	0.03
4	0.40	0.05	0.10	4.28
5	0.25	1.32	0.23	-
6	0.34	3.27	3271.94	-
7	3.00	15.09	0.14	-
8	7.28	4.17	-	-
9	1.13	117.23	-	-
10	12.44	31	-	-
11	13.23	16.55	-	-
12	49.13	72.39	-	-
13	14.56	1027.54	-	-
14	9.16	95.52	-	-
15	156.84	554.87	-	-
16	154.56	1405.69	-	-
17	302.16	-	-	-
18	1067.04	-	-	-
19	1596.38	-	-	-
20	870.74	-	-	-
21	2577.42	-	-	-

Tabela 1. Magični kvadrat – rezultati



Grafikon 1. Magični kvadrat – rezultati

4.2 Binarne sekvence male autokorelacije

Problem binarnih sekvenci male autokorelacije (*Low Autocorellation Binary Sequences, LABS*) ima mnoge praktične primene u oblasti komunikacija i elektrotehnike. Cilj je kreirati binarnu sekvencu S_i dužine n koja minimizuje autokorelacije promenljivih iz sekvence. Svaka promenljiva u sekvenci ima vrednost +1 ili -1. U zavisnosti od uslova, definišemo k-tu autokorelaciju:

1. Neperiodični (otvoreni) granični uslovi
   $$C_k = \sum_{i=0}^{n-k-1} S_i \cdot S_{i+k},$$
2. Periodični (ciklični) granični uslovi
   $$C_k = \sum_{i=0}^{n-1} S_i \cdot S_{(i+k) mod\ n}.$$

Cilj je minimizovati zbir kvadrata autokorelacija

$$\sum_{k=1}^{n-1} C_k^2.$$

U ovom radu je analiziran problem pri neperiodičnim graničnim uslovima, s obzirom na to da je pri periodičnim graničnim uslovima pronađeno ekzaktno analitičko rešenje.

4.2.1 Microsoft solver foundation

Model *LABS* problema dimenzije n definisan za *MSF CSP* rešavač sadrži n promenljivih, $x_0, x_1, \dots, x_{n-1}$, gde svaka promenljiva predstavlja odgovarajuću vrednost u sekvenci. Svaka promenljiva uzima vrednost iz domena

$$x_0, x_1, \dots, x_{n-1} \in \{-1, 0, 1\}.$$

Prvo ograničenje koje se uvodi je da svaka promenljiva mora da bude različita od nule:

$$x_i \neq 0, \qquad i \in \{0, 1, \dots, n-1\}.$$

Računaju se odgovarajuće autokorelacije:

$$C_k = \sum_{i=0}^{n-k-1} S_i \cdot S_{i+k}, \qquad k \in \{1, 2, \dots, n-1\},$$

i njihovi kvadrati:

$$D_k = C_k^2, \qquad k \in \{1, 2, \dots, n-1\}.$$

Rešavač minimizije funkciju:

$$f(D_1, D_2, \dots, D_{n-1}) = \sum_{i=1}^{n-1} D_i.$$

Prilikom rešavanja ovog problema rešavač dokazuje tačnost pronađenog rešenja, odnosno da je ono zaista minimalno. Tek kada dokaže korektnost staje sa radom i vraća rezultat. Vreme računanja prikazano u rezultatima ne prikazuje vreme za koje je rešavač došao do tačnog rešenja,već vreme za koje je rešavač dokazao njegovu korektnost.

4.2.2 Genetski algoritam

Kao prostor rešenja je korišćen skup S svih sekvenci dužine n u kojima se kao vrednosti pojavljuju jedino brojevi 1 i -1. Funkcija cilja je definisana kao

$$f: S \to \mathbb{Z}$$

i ima vrednost tražene sume kvadrata svih autokorelacija. Potrebno je minimizovati funkciju cilja.

Svaka jedinka u populaciji je niz binarnih brojeva dužine n. Nula u hromozomu kodira broj -1 u sekvenci, a jedinica broj 1. Korišćeno je standardno dvopoziciono ukrštanje.

Mutacija je definisana kao promena vrednosti jednog bita u hromozomu.

Korišćeni su parametri:

- maksimalan broj identičnih jedinki – 2,
- maksimalan broj jedinki sa istom vrednošću funkcije cilja – 20,
- očekivana prosečna veličina turnira – 5.2,
- verovatnoća ukrštanja – 0.9,
- verovatnoća mutacije – $1.0 \,/\, n$,
- faktor zaleđenog bita – 3.5.

4.2.3 Metoda promenljivih okolina

Kao prosor rešenja je korišćen skup S svih sekvenci dužine n u kojima se kao vrednosti pojavljuju jedino brojevi 1 i -1, slično genetskom algoritmu. Funkcija cilja je definisana kao

$$f: S \to \mathbb{Z}$$

i ima vrednost tražene sume kvadrata svih autokorelacija. Potrebno je minimizovati funkciju cilja.

Kao prva okolina date tačke x iz prostora rešenja koristi se skup tačaka koji se od tačke x razlikuje na samo jednom mestu u sekvenci. Promenom jednog broja ($1 \to -1$ ili $-1 \to 1$) se od tačke x, dobija tačka x', i važi $x' \in N_1(x)$. Tačka iz druge okoline se dobija sa dve promene, tačka iz k-te okoline sa k promena.

Nakon promene broja u sekvenci potrebno je izračunati novu vrednost funkcije cilja. Računanje vrednosti funkcije cilja se može optimizovati korišćenjem podatka da se nova tačka nalazi u okolini stare. Svakoj tački je pridružena informacija koja sadrži gornje trougaonu matricu m dimenzije $n \times n$, za koju važi $m_{ki} = x_i \cdot x_{i+k}$, i niz izračunatih autokorelacija c. Implementiran je sledeći algortitam pri promeni i-te vrednosti u sekvenci:

Promena i-te vrednosti u sekvenci:

za vrednosti broja k od 0 do $n-1$ *izvrši:*

1. *Ispitati uticaj broja sa i-te pozicije na k-tu korelaciju;*
2. *Ispitati uticaj broja sa* $(i-k)$*-te pozicije na k-tu korelaciju;*

$rezultat \leftarrow \sum_{k=1}^{n-1} c_k^2$;

Ispitati uticaj broja sa i-te pozicije na k-tu korelaciju:

ukoliko je $(k+i<n) \wedge (0<k<n) \wedge (0 \leq i < n-1)$ *onda:*

1. $c_k \leftarrow c_k - m_{ki}$;
2. $m_{ki} \leftarrow x_i \cdot x_{i+k}$;
3. $c_k \leftarrow c_k + m_{ki}$;

Na ovaj način se funkcija cilja računa u vremenu $O(n)$, što u velikoj meri poboljšava brzinu izračunavanja.

U algoritmu mešanja, u zavisnosti od parametra k, izvršava se k uzastopnih promena vrednosti slučajnog izabranog broja u sekvenci. Broj koji je jednom bio promenjen ne može u istoj iteraciji mešanja opet da se promeni.

Korišćeni parametri su:

- u uprošćenom *VNS*-u $k_{max} = 1$,
- u lokalnoj pretrazi $k_{max} = 1$,
- u osnovnom *VNS*-u $k_{max} = n \; div \; 2$.

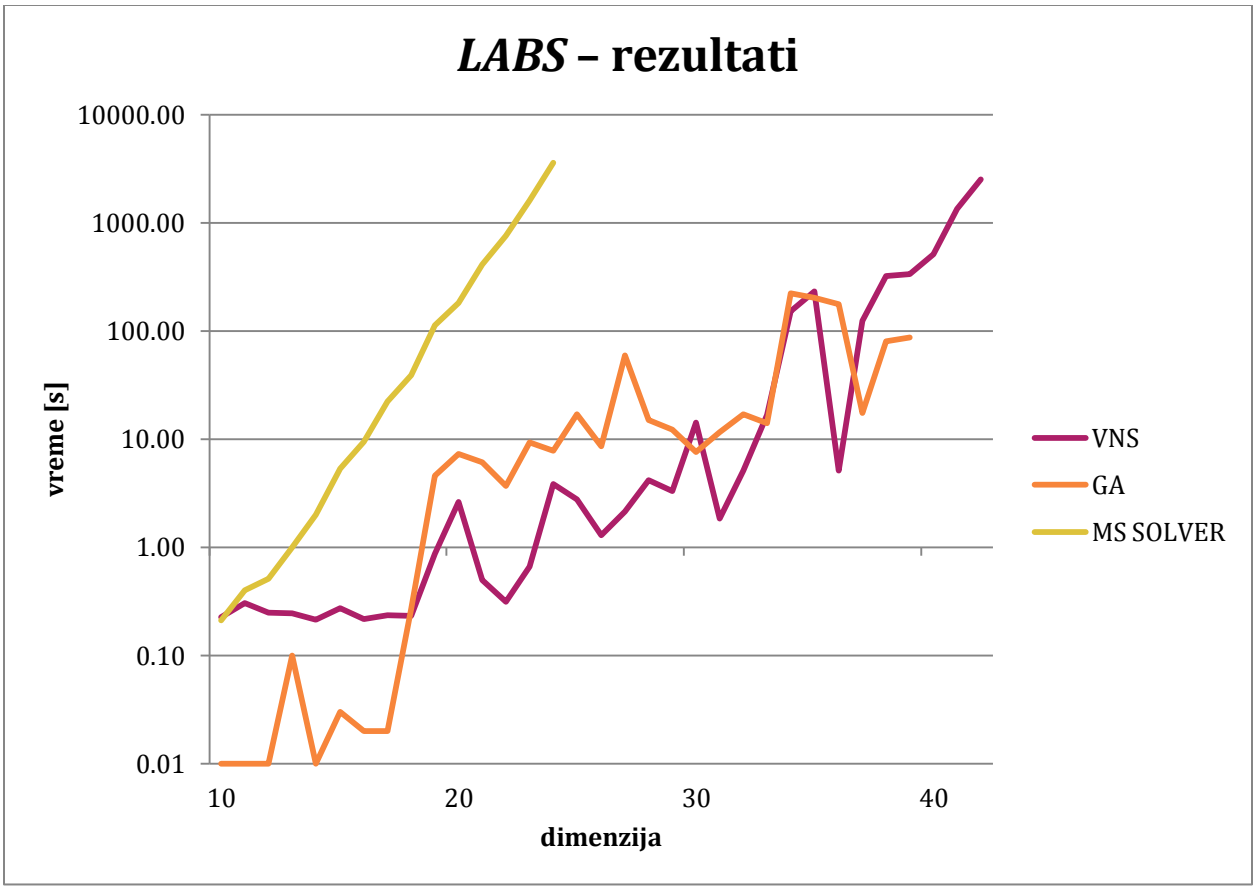
4.2.4 Rezultati

Sva tri tehnike rešavanja su testirane za rešavanje *LABS* problema dimenzija od 10 do 50. Svi testovi su vršeni na računarima sa *Intel Core i7 860* procesorima i svaka instanca je puštana da radi do sat vremena. *MSF CSP* rešavač je rešio prvih 15 instanci, genetski algoritam 30, metoda promenljivih okolina 37.

U tabeli 2 dati su svi dobijeni rezultati i grafički su prikazani na grafikonu 2.

Dimenzija	VNS [s]	GA [s]	MSF [s]
10	0.23	0	0.21
11	0.31	0.01	0.40
12	0.25	0.01	0.52
13	0.24	0.1	1.00
14	0.21	0	2.00
15	0.27	0.03	5.33
16	0.22	0.02	9.45
17	0.24	0.02	22.42
18	0.23	0.27	39.00
19	0.87	4.57	112.57
20	2.64	7.33	182.73
21	0.50	6.09	412.76
22	0.31	3.67	758.20
23	0.66	9.38	1610.19
24	3.87	7.78	3581.50
25	2.78	16.99	-
26	1.29	8.57	-
27	2.15	59.11	-
28	4.15	15.01	-
29	3.31	12.28	-
30	14.16	7.64	-
31	1.85	11.54	-
32	5.15	16.99	-
33	16.71	14.08	-
34	152.06	223.8	-
35	233.64	201.3	-
36	5.15	177.54	-
37	123.62	17.57	-
38	320.02	80.36	-
39	337.23	86.68	-
40	510.65	-	-
41	1343.60	-	-
42	2503.09	-	-
43	-	-	-
44	-	-	-
45	914.81	-	-
46	967.11	-	-
47	1647.00	-	-
48	-	-	-
49	-	-	-
50	279.84	-	-

Tabela 2. *LABS* – rezultati



Grafikon 2. *LABS* – rezultati

4.3 Dizajn uravnoteženih nepotpunih blokova

Dizajn uravnoteženih nepotpunih blokova (*Balanced Incomplete Block Designs, BIBD*) je standardni kombinatorni problem teorije dizajna. Prvobitno je korišćen za dizajn statističkih eksperimenata a kasnije su pronađene druge primene, na primer u kriptografiji. Predstavlja specijalan slučaj uopštenog problema dizajna blokova, koji uključuje probleme Latinskog kvadrata (*Latin Square problems*).

BIBD se definiše kao uređenje v različitih objekata u b blokova takvih da svaki blok sadrži k različitih objekata, svaki objekat se pojavljuje u tačno r različitih blokova, i svaka dva različita ovjekta se pojavljuju zajedno u tačno $lambda$ blokova.

Drugi način definisanja *BIBD*-a je preko matrica binarnih brojeva, dimenzije $v \times b$, sa tačno r jedninica u svakom redu, k jedinica u scakoj koloni, gde je skalarni proizvod svaka dva različita reda jednak $lambda$. Na taj način *BIBD* je definisan petorkom celobrojnih parametara $(v, b, r, k, lambda)$. Na primer rešenje problema za parametre (7,7,3,3,1) je

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

4.3.1 Microsoft solver foundation

Model *BIBD* problema za parametre $(v, b, r, k, lambda)$ definisan za *MSF CSP* rešavač sadrži $v \cdot b$ promneljivih, $x_{11}, x_{12}, \dots, x_{vb}$. Svaka promenljiva predstavlja odgovarajuću vrednost u matrici i uzima vrednost iz domena

$$x_{11}, x_{12}, \dots, x_{vb} \in \{0, 1\}.$$

Uvedena su sledeća ograničenja:

- suma svakog reda je r
  $$\sum_{j=1}^{b} x_{ij} = r, \qquad i \in \{1,2, \dots, v\},$$
- suma svake kolone je k
  $$\sum_{j=1}^{v} x_{ji} = k, \qquad i \in \{1,2, \dots, b\},$$

- skalarni proizvod svaka dva reda je $lambda$
  $\sum_{n=1}^{b} x_{in} \cdot x_{jn} = lambda, \qquad i, j \in \{1,2, \dots, v\}.$

Prilikom rešavanja ovog modela prvi put kada rešavač pronađe rešenje koje zadovoljava sva ograničenja, staje sa radom i vraća rezultat.

4.3.2 *SAT*

Model problema definisan za SAT rešavač sadrži $v \cdot b$ promneljivih, $x_{11}, x_{12}, \dots, x_{vb}$. Svaka promenljiva uzima vrednost iz domena

$$x_{11}, x_{12}, \dots, x_{vb} \in \{T, F\}.$$

T (tačno) kodira broj 1 u matrici, dok F (netačno) kodira broj 0. Ograničenja uvedena za SAT model su:

- suma svakog reda je r
  $Card(x_{i1}, x_{i2}, \dots, x_{ib}) = r, \qquad i \in \{1,2, \dots, v\},$
- suma svake kolone je k
  $Card(x_{1i}, x_{2i}, \dots, x_{vi}) = k, \qquad i \in \{1,2, \dots, b\},$
- skalarni proizvod svaka dva reda je $lambda$
  $Card(x_{i1} \wedge x_{j1}, x_{i2} \wedge x_{j2}, \dots, x_{ib} \wedge x_{jb}) = lambda, \qquad i, j \in \{1,2, \dots, v\}.$

Funkcija $Card(a_1, a_2, \dots, a_n)$ vraća broj promenljivih sa vrednošću T u nizu promenljivih $a_1, a_2, \dots, a_n$.

Prilikom rešavanja ovog modela prvi put kada rešavač pronađe rešenje koje zadovoljava sva ograničenja, staje sa radom i vraća rezultat.

4.3.3 Genetski algoritam

Kao prostor rešenja je korišćen skup S svih matrica $v \times b$ u kojima se kao vrednosti pojavljuju jedino brojevi 0 i 1. Funkcija cilja je definisana kao

$$f: S \to \mathbb{Z}$$

i ima vrednost

$$f(x_{11}, x_{12}, \dots, x_{vb}) = \sum_{i=1}^{v} \left| \sum_{j=1}^{b} x_{ij} - r \right| + \sum_{i=1}^{b} \left| \sum_{j=1}^{v} x_{ji} - k \right| + \sum_{i=1}^{v} \sum_{j=i+1}^{v} \left| \sum_{n=1}^{b} x_{in} \cdot x_{jn} - lambda \right|,$$

koja predstavlja sumu odstupanja zbira svake vrste, zbira svake kolone i zbira svakog skalarnog proizvoda dva različita reda.

Potrebno je minimizovati funkciju cilja, odnosno naći onu matricu za koju je greška jednaka nuli. Ta matrica predstavlja rešenje problema.

Svaka jedinka u populaciji je niz binarnih brojeva dužine $v \cdot b$. Jedinka kodira matricu tako što se binarni brojevi hromozoma jedinke rasporede redom po vrstama matrice. Korišćeno je standardno dvopoziciono ukrštanje.

Mutacija je definisana kao promena vrednosti jednog bita u hromozomu.

Korišćeni su parametri:

- maksimalan broj identičnih jedinki – 1,
- maksimalan broj jedinki sa istom vrednošću funkcije cilja – 10,
- očekivana prosečna veličina turnira – 5.4,
- verovatnoća ukrštanja – 0.85,
- verovatnoća mutacije – 0.4 / $v \cdot b$,
- faktor zaleđenog bita – 3.5.

4.3.4 Metoda promenljivih okolina

Kao prostor rešenja je korišćen skup S svih matrica $v \times b$ u kojima se kao vrednosti pojavljuju tačno $v \cdot r$ jedinica, a ostalo su nule. Vrednost funkcije cilja je ista kao i u genetskom algoritmu za ovaj problem, samo se računa na manjem domenu. Cilj je naći onu matricu za koju funkcija cilja ima vrednost nula.

Kao prva okolina date tačke x koristi se skup tačaka koji pripada prostoru rešenja, a od tačke x se razlikuju na tačno dva mesta u matrici. Tako se promenom dva broja u tački x, jedne jedinice u nulu i jedne nule u jedinicu, mogu dobiti sve tačke iz prve okoline tačke x, $N_1(x)$. Tačke iz druge okoline se dobijaju sa dve promene, tačke iz k-te okoline sa k promena.

Nakon promene brojeva u matrici potrebno je izračunati novu vrednost funkcije cilja. Računanje vrednosti funkcije cilja se može optimizovati korišćenjem podatka da se nova tačka nalazi u okolini stare. Svakoj tački je pridružena informacija koja sadrži niz grešaka redova gr (odstupanja zbira redova od vrednosti r), niz grešaka kolona kr (odstupanja od vrednosti k), marticu grešaka skalarnih proizvoda redova m (odstupanja od vrednosti $lambda$) i trenutni rezultat r. Implementiran je sledeći algortitam pri promeni vrednosti brojeva x_{ij} i x_{nm} u matrici:

Promena vrednosti brojeva x_{ij} i x_{nm} u matrici:

Promena vrednosti broja x_{ij} u matrici;

Promena vrednosti broja x_{nm} u matrici;

Promena vrednosti broja x_{ij} u matrici:

$x_{ij} \leftarrow |x_{ij} - 1|$;

Promeniti vrednost greške i-tog reda gr_i i osvežiti rezultat r;

Promeniti vrednost greške j-te kolone gk_j i osvežiti rezultat r;

Promeniti vrednost greške skalarnih proizvoda svih redova sa i-tim redom u matrici m i osvežiti rezultat r;

Na ovaj način se funkcija cilja računa u vremenu $O(v)$, što u velikoj meri poboljšava brzinu izračunavanja.

U algoritmu mešanja se, u zavisnosti od parametra k, izvršava k uzastopnih promena vrednosti dva izabrana broja iz matrice. Broj koji je jednom bio promenjen ne može u istoj iteraciji mešanja opet da se promeni.

Promena je definisana na sledeći način:

- Sa verovatnoćom 0.9 se na slučajan način izabere broj u matrici za zamenu, i onda se traži najbolji koji će zameniti mesto sa njim.
- Sa verovatnoćom 0.1 se oba broja za zamenu biraju na slučajan način.

Korišćeni su parametri:

- u uprošćenom *VNS*-u, $k_{max} = 1$,
- u lokalnoj pretrazi $k_{max} = 1$,
- u osnovnom *VNS*-u, $k_{max} = \min(20, v \cdot b \; div \; 2)$.

4.3.5 Rezultati

Sva četiri tehnike rešavanja su testirane za rešavanje *BIBD* problema, na ukupno 85 instanci. Svi testovi su vršeni na računarima sa *Intel Core i7 860* procesorima i svaka instanca je puštana da radi do deset minuta. *MSF CSP* rešavač je rešio 14 instanci, *SAT* rešavač svih 85 instanci, uz najmanje prosečno vreme izvršavanja, genetski algoritam 65, metoda promenljivih okolina 44.

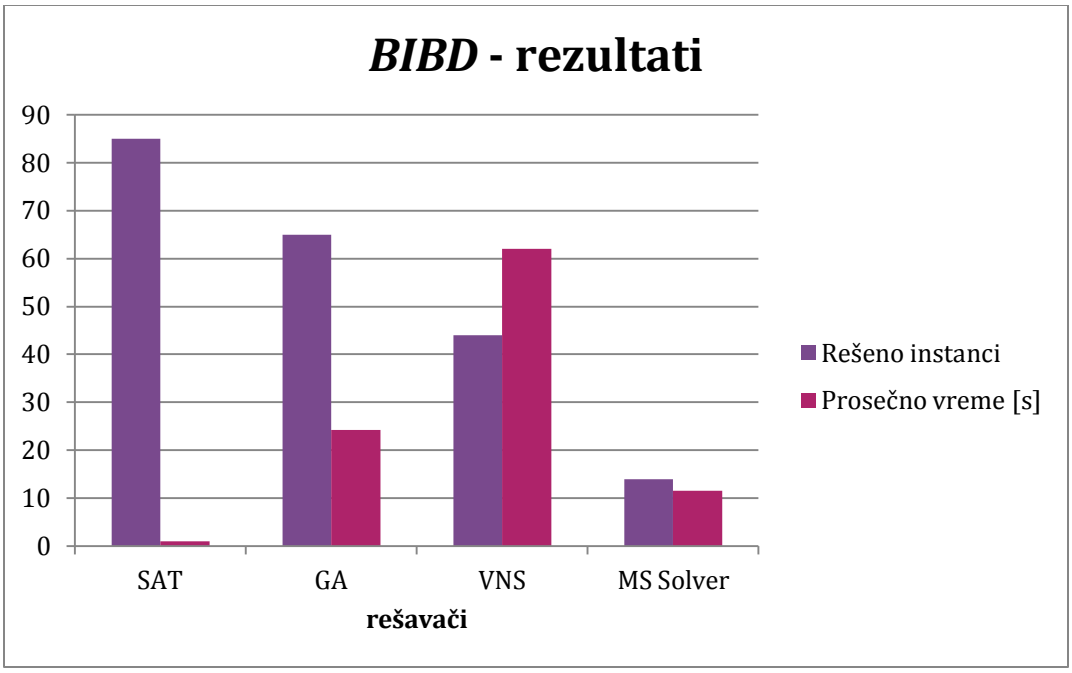
U tabeli 3 dati su svi dobijeni rezultati i grafički su prikazani na grafikonu 3.

v b r k lambda	SAT	GA	VNS	MS Solver
6 10 5 3 2	0.00	0.03	0.20	0.08
6 20 10 3 4	0.00	0.07	0.18	67.08
6 30 15 3 6	0.00	0.21	2.94	-
6 40 20 3 8	0.00	0.25	0.86	-
6 50 25 3 10	0.10	0.47	3.29	-
6 60 30 3 12	0.10	0.42	5.79	-
6 70 35 3 14	0.10	0.71	7.91	-
6 80 40 3 16	0.50	0.66	4.53	-
7 14 6 3 2	0.00	0.09	5.94	0.10
7 21 9 3 3	0.00	0.12	0.81	2.67
7 28 12 3 4	0.00	0.22	0.58	-

7 35 15 3 5	0.00	0.46	2.59	-
7 42 18 3 6	0.00	0.97	0.45	-
7 49 21 3 7	0.00	0.44	4.05	-
7 56 24 3 8	0.00	0.77	7.39	-
7 63 27 3 9	0.10	0.73	7.33	-
7 7 3 3 1	0.00	0.02	0.15	0.08
7 84 36 3 12	0.10	0.97	21.66	-
8 14 7 4 3	0.00	0.27	7.02	90.00
8 28 14 4 6	0.00	5.03	6.48	-
8 42 21 4 9	0.10	0.85	1.51	-
8 56 21 3 6	0.10	0.9	20.15	-
8 56 28 4 12	0.10	1.65	15.78	-
8 70 35 4 15	0.50	3.05	12.93	-
9 12 4 3 1	0.00	6.95	1.29	0.09
9 18 8 4 3	0.00	0.45	17.61	-
9 24 8 3 2	0.00	10.28	1.10	-
9 36 12 3 3	0.00	1.19	25.28	-
9 36 16 4 6	0.10	3.52	-	-
9 48 16 3 4	0.00	2.91	149.38	-
9 54 24 4 9	0.20	2.87	112.54	-
9 60 20 3 5	0.10	3.58	15.53	-
9 72 32 4 12	1.40	2.56	113.91	-
9 90 40 4 15	1.00	7.85	57.76	-
10 15 6 4 2	0.00	0.45	19.89	0.10
10 18 9 5 4	0.00	0.63	-	-
10 30 12 4 4	0.00	7.56	16.14	-
10 30 9 3 2	0.00	13.3	14.13	-
10 36 18 5 8	0.10	145.95	-	-
10 45 18 4 6	0.20	2.75	281.38	-
10 54 27 5 12	1.30	14.81	278.96	-
10 60 24 4 8	0.20	7.13	225.89	-
10 72 36 5 16	0.80	74.02	-	-
10 75 30 4 10	0.60	5.78	582.52	-
10 90 36 4 12	0.70	9.48	-	-
11 11 5 5 2	0.00	0.28	2.81	0.12
11 22 10 5 4	0.00	-	-	-
11 33 15 5 6	0.10	2.33	-	-
11 44 20 5 8	0.70	41.98	520.45	-
11 55 15 3 3	0.10	66.01	-	-
11 55 20 4 6	0.10	27.38	-	-
11 55 25 5 10	1.40	14.73	-	-
11 66 30 5 12	3.00	65.41	-	-
11 77 35 5 14	10.70	17.5	-	-
11 88 40 5 16	29.50	18.96	-	-

12 22 11 6 5	0.10	-	-	-
12 33 11 4 3	0.10	37.65	-	-
12 44 11 3 2	0.00	24.54	-	-
12 44 22 6 10	2.90	-	-	-
12 66 22 4 6	0.80	161.61	-	-
12 66 33 6 15	11.90	177.15	-	-
13 13 4 4 1	0.00	1.17	39.86	0.08
13 26 12 6 5	2.90	-	-	-
13 26 6 3 1	0.00	-	-	-
13 26 8 4 2	0.00	-	-	-
13 39 12 4 3	0.00	130.1	-	-
13 39 15 5 5	0.20	186.02	-	-
13 52 12 3 2	0.00	126.28	-	-
13 52 16 4 4	0.10	-	-	-
13 52 24 6 10	2.10	-	-	-
13 65 20 4 5	0.10	93.02	-	-
15 15 7 7 3	0.00	32.44	-	0.68
15 35 7 3 1	0.00	-	-	-
15 42 14 5 4	0.50	-	-	-
15 63 21 5 6	3.20	-	-	-
16 16 6 6 2	0.00	-	113.63	0.14
16 20 5 4 1	0.00	-	-	0.13
16 24 9 6 3	0.20	-	-	-
16 40 10 4 2	0.10	-	-	-
16 48 15 5 4	2.70	-	-	-
16 60 15 4 3	0.20	-	-	-
19 19 9 9 4	0.30	-	-	-
21 21 5 5 1	0.00	6.34	-	0.18
25 30 6 5 1	0.10	-	-	-
31 31 6 6 1	0.20	-	-	-

Tabela 3. *BIBD* – rezultati



Grafikon 3. *BIBD* rezultati

4.4 Turniri u kojima se sve ekipe međusobno sastaju

Turniri u kojima se sve ekipe međusobno sastaju (*Round-Robin Tournaments, RRT*) je problem organizacije turnira sa n timova. Turnir traje $(n-1)$-nu nedelju, svaka nedelja je podeljena na $n/2$ termina u kojima se igraju utakmice. U svakom terminu se sastaju tačno dva tima. Turnir mora da zadovoljava sledeća ograničenja:

- svaki tim igra jednom nedeljno,
- svaki tim igra najviše dva puta u istom terminu za vreme trajanja turnira, i
- svaki tim igra po jednu utakmicu sa svakim drugim timom.

Primer turnira dat je u tabeli 4.

	nedelja 1	nedlja 2	nedelja 3	nedelja 4	nedelja 5	nedelja 6	nedelja 7
termin 1	0 – 1	0 – 2	4 – 7	3 – 6	3 – 7	1 – 5	2 – 4
termin 2	2 – 3	1 – 7	0 – 3	5 – 7	1 – 4	0 – 6	5 – 6
termin 3	4 – 5	3 – 5	1 – 6	0 – 4	2 – 6	2 – 7	0 – 7
termin 4	6 – 7	4 – 6	2 – 5	1 – 2	0 – 5	3 – 4	1 – 3

Tabela 4. *RRT* – 8 ekipa

4.4.1 Microsoft solver foundation

Model *RRT* problema sa n timova definisan za *MSF CSP* rešavač sadrži matricu promenljivih $x_{11}, x_{12}, \dots, x_{n(n-1)}$, dimenzije $n \times (n-1)$. Svaka promenljiva predstavlja po jedan tim, kolona u matrici predstavlja odgovarajuću nedelju, a vrsta odgovarajući termin. Tačnije vrste $(2 \cdot i)$ i $(2 \cdot i + 1)$ predstavljaju i-ti termin, a u preseku sa kolonom j dobijaju se dva tima koja igraju utakmicu j-te nedelje u i-tom terminu. Svaka promenljiva uzima vrednost iz domena

$$x_{11}, x_{12}, \dots, x_{n(n-1)} \in \{0,1, \dots, n-1\}.$$

Prvo ograničenje koje se uvodi je da sve promenljive u svakoj koloni moraju da budu međusobno različite, jer jedna ekipa ne može da igra dvaput jedne nedelje

$$Unequal(x_{1i}, x_{2i}, \dots, x_{ni}), \qquad i \in \{1,2, \dots, n-1\}.$$

Definiše se niz b dužine $n/2$ takav da i-ti član niza predstavlja niz timova koji igraju u i-tom terminu

$$b_i = \left(x_{(2i)1}, x_{(2i)2}, \dots, x_{(2i)(n-1)}, x_{(2i+1)1}, x_{(2i+1)2}, \dots, x_{(2i+1)(n-1)}\right), \qquad i \in \{1,2, \dots, n/2\}.$$

Zatim se za svaki termin i svaki tim uvodi niz logičkih vrednosti, čiji članovi imaju vrednot tačno ukoliko ukoliko se se u datom terminu dati tim pojavljuje na odgovarajućem mestu u matrici

$$c_{ij} = (b_i[1] = j, b_i[2] = j, \dots, b_i[2 \cdot n - 2] = j), \qquad i \in \{1,2, \dots, n/2\}, j \in \{0,1, \dots, n-1\}.$$

Uvodi se niz ograničenja koji ograničavaju da jedan tim ne može da igra više od dva puta u istom terminu

$$AtMostMofN\big(2, c_{ij}\big), \qquad i \in \{1,2, \dots, n/2\}, j \in \{0,1, \dots, n-1\}.$$

Ovim modelom i dalje nisu zadata sva potrebna ograničenja. Potrebno je jos zadati ograničenje koje govori da su sve utakmice različite, tačnije da dva tima ne mogu da se sastanu dva puta. Da bi se uvelo takvo ograničenje potrebno je kreirati pojam utakmice. Pre svega potrebno je uvesti da ako se sastaju ekipe $x_{(2i)j}$ i $x_{(2i+1)j}$ tada važi

$$x_{(2i)j} < x_{(2i+1)j}.$$

Tada je moguće napraviti niz d svih odigranih utakmica, tako što se utakmica ekipa $x_{(2i)j}$ i $x_{(2i+1)j}$ kodira brojem

$$x_{(2i)j} \cdot n + x_{(2i+1)j}.$$

Na kraju potrebno je ubaciti ograničenje da su sve utakmice medjusobno različite

$$Unequal\big(d_1, d_1, \dots, d_{(n-1)\cdot n/2}\big).$$

Prilikom rešavanja ovog modela prvi put kada rešavač pronađe rešenje koje zadovoljava sva ograničenja, staje sa radom i vraća rezultat.

4.4.2 Genetski algoritam

Za rešavanje *RRT* problema genetskim algoritmom uveden je skup *U* svih utakmica koje treba da budu odigrane na turniru. Kao prostor rešenja je korišćen skup *S* svih permutacija skupa *U*.

Svaka tačka iz skupa S jednoznačno određuje jedan turnir tako što se utakmice redom, po vrstama, zapišu u tabelu turnira. Tabela turnira predstavlja raspored odigravanja utakmica po terminima i nedeljama (primer je dat u tabeli 4 za $n = 8$). Funkcija cilja je definisana kao

$$f: S \to \mathbb{Z}$$

i ima vrednost greške koju daje data permutacija. Greška se računa kao zbir prekoračenja pojavljivanja svih timova u terminima i nedeljama. Ako se tim pojavi $i > 1$ puta u jednoj nedelji onda je prekoračenje $i - 1$. Ako se tim pojavi $j > 2$ puta u jednom terminu onda je prekoračenje $j - 2$.

Potrebno je minimizovati funkciju cilja, odnosno naći onu permutaciju za koju je greška jednaka nuli. Ta permutacija predstavlja regularan turnir.

Svaka jedinka u populaciji kodira jednu permutaciju skupa S. Hromozom se sastoji od niza brojeva dužine $(n-1)\cdot n/2$ koji predstavljaju traženu permutaciju.

Proces ukrštanja i mutacije je isti kao što je prikazan na problemu magičnog kvadrata rešavanog genetskim algoritmom.

Korišćeni su parametri:

- maksimalan broj identičnih jedinki – 2,
- maksimalan broj jedinki sa istom vrednošću funkcije cilja – 20,
- očekivana prosečna veličina turnira – 5.2,
- verovatnoća ukrštanja – 0.85,
- verovatnoća mutacije – $0.5 \,/\, n^2$
- faktor zaleđenog bita – 4.

4.4.3 Metoda promenljivih okolina

Prostor rešenja i funkcija cilja, redom S i f, koji su korišćeni u metodi promenljivih okolina su identični onima koji su prikazani u genetskom algoritmu, za problem *RRT*. Potrebno je minimizovati funkciju cilja, odnosno naći onu permutaciju za koju je greška jednaka nuli. Ta permutacija predstavlja regularan turnir.

Kao prva okolina date tačke x iz prostora rešenja koristi se skup tačaka koji se od tačke x razlikuje u tačno dva broja. Zamenom dva broja u datoj permutaciji se od tačke x dobija tačka x', i važi $x' \in N_1(x)$. Tačka iz druge okoline se dobija sa dve zamene, tačka iz k-te okoline sa k zamena.

Nakon zamene dva broja u permutaciji potrebno je izračunati novu vrednost funkcije cilja. Računanje vrednosti funkcije cilja se može optimizovati korišćenjem podatka da se nova tačka nalazi u okolini stare. Svakoj tački je pridružena informacija koliko se puta svaki tim pojavio u svakom terminu, koliko se puta svaki tim pojavio svake nedelje i trenutni rezultat funkcije cilja. Uz pomoć ovih informacija zamenom dva broja u permutaciji, slično kao i kod problema magičnog kvadrata, se nova vrednost funkcije cilja računa u vremenu $O(1)$, što u značajnoj meri boboljšava brzinu izračunavanja.

U algoritmu mešanja se izvršava k uzastipnih zamena parova tačaka u permutaciji. Broj koji je jednom bio zamenjen ne može u istoj iteraciji opet da se menja. Zamena je definisana na sledeci način:

- Sa verovatnoćom 0.9 se na slučajan način izabere broj u permutaciji za zamenu, i onda se traži najbolji koji će zameniti mesto sa njim.
- Sa verovatnoćom 0.1 se oba broja za zamenu biraju na slučajan način.

Korišćeni su parametri:

- u uprošćenom *VNS*-u, $k_{max} = 1$,
- u lokalnoj pretrazi $k_{max} = 1$,
- u osnovnom *VNS*-u, $k_{max} = \min(10, n * (n-1)\ div\ 4)$.

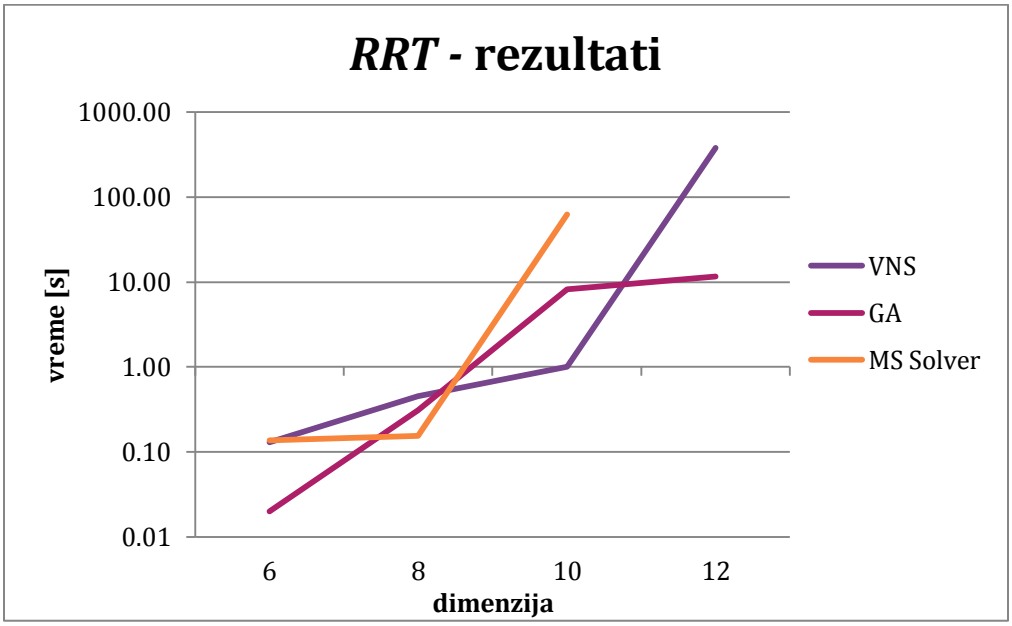
4.4.4 Rezultati

Sva tri tehnike rešavanja su testirane za rešavanje *RRT* problema dimenzija 6, 8, 10 i 12. Svi testovi su vršeni na računarima sa *Intel Core i7 860* procesorima i svaka instanca je puštana da radi do sat vremena. *MSF CSP* rešavač je rešio tri instance, genetski algoritam i metoda promenljivih okolina četiri.

U tabeli 5 dati su svi dobijeni rezultati i grafički su prikazani na grafikonu 4.

Dimenzija	GA [s]	VNS [s]	MS Solver [s]
6	0.02	0.13	0.14
8	0.31	0.45	0.15
10	8.2	1.01	61.95
12	11.54	378.20	-

Tabela 5. *RRT* – rezultati



Grafikon 4. *RRT* – rezultati

Poglavlje 5

Zaključak

U radu je opisana klasa problema zadovoljenja ograničenja. Predstavljena su i rešavana četiri problema iz te klase. Opisane su tri egzaktne tehnike rešavanja i dve metaheuristike. Implementirana su četiri modela za *MSF CSP* rešavač, jedan model za *CPLEX* i jedan za *SAT* rešavač. Implementirana su četiri programa zasnovana na genetskim algoritmima (uz pomoć *GA Framework* biblioteke), kao i četiri programa zasnovana na metodi promenljivih okolina.

Svaki problem je rešavan egzaktnim i heurističkim metodama i analizirani su rezultati. Dva puta su bile uspešnije metaheuristike, jednom egzaktni rešavači, a u jednom od problema su rezultati su dosta slični u obe tehnologije.

Nakon analize rezultata ne može se precizno utvrditi koju tehniku treba koristiti za klasu problema zadovoljenja ograničenja, ili neku njenu podklasu. Ono što se može na osnovu dobijenih rezultata zaključiti je da je za jednostavne probleme linearnog programiranja uglavnom bolje koristiti egzaktne rešavače, dok je za složene probleme linearnog i nelinearnog programiranja velikih dimenzija bolje koristiti heuristike. Takođe nije dovoljno za neki problem doneti odluku da li će se rešavati heuristikama ili egzaktnim rešavačima. Ima mnogo različitih egzaktnih rešavača, kao i različitih heuristika, koje mogu da daju značajno drugačije rezultate. Potrebno je testirati vise opcija, proveriti koji je rešavač najbolji za problem. Konkretno kada je testiran *BIBD* problem, testovi su puštani na *picosat* i *cryptominisat* rešavačima (*cryptominisat* je na zadnjem *SAT* turniru dao najbolje rezultate i pokazao se kao najbolji). *Picosat* je instance izračunao i po nekoliko hiljada puta brže od *cryptominisat*-a. Isti zaključak se može izvesti i iz rezultata koje su dale heuristike. Na dva problema je genetski algoritam bio uspešniji, a na dva problema *VNS*.

Dalja unapređenja rada i dalje istraživanje moguće je realizovati na neki od sledećih načina:

- testirati druge egzaktne i heurističke tehnike,
- testirati druge probleme zadovoljenja ograničenja,
- testirati drugu klasu problema,
- kreirati i testirati hibridnu tehniku rešavanja problema kao kombinaciju egzaktnih i heurističkih tehnika.

Jedna od mogućnosti za kreiranje hibridne tehnike je kombinacija *VNS* i *SAT* tehnika. Pronalazak okoline u kojoj se nalazi bolji lokalni minimum je čest problem koji se javlja u *VNS* metodi. Pretraga okolina oko trenutnog lokalnog minimuma *SAT* rešavačem bi mogla da doprinese efikasnijem rešavanju problema.

Za potrebe testiranja tehnika rešavanja u radu su korišćeni neki već implementirani rešavači i biblioteke kojima se zadaje model problema koji se rešava, ali su i neki rešavači implementirani u

potpunosti. Može se zaključiti da je implementaciono jednostavnije zadati model već kreiranom rešavaču ili biblioteci. Međutim implementacija je samo tehnički deo realizacije koji u većini slučajeva ne predstavlja veći problem. Kreiranje samog modela je najbitniji deo realizacije i neophodan je u svakom pristupu rešavanja problema.

Poglavlje 6

Literatura

[Dan63] **Dantzig G. B.,** "Linear Programming and Extensions", NJ: Princeton University Press (1963).

[For02] **Forsgren A., Gill P. E., Wright M. H.,** "Interior Methods for Nonlinear Optimization.", SIAM Vol. 44, pp. 525-597 (2002).

[Gen10] **Gendreau M., Potvin J.-Y.**, "Handbook of Metaheuristics, Second Edition", Springer (2010).

[Gol65] **Golomb S., Baumert L.,** "Backtrack programming", J. ACM, Vol. 12, pp. 516–524 (1965).

[Han01] **Hansen P., Mladenović N.**, "Variable neighborhood search: Principles and applications", European Journal of Operational Research, Vol. 130, pp. 449-467 (2001).

[Hol75] **Holland J.H.**, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor (1975).

[Kar84] **Karmarkar N.,** "A New Polynomial-Time Algorithm for Linear Programming", Combinatorica 4, pp. 373-395 (1984).

[Kle72] **Klee V., Minty G. J., Shisha O.,** "How Good is the Simplex Algorithm?", New York: Academic Press, pp. 159-175 (1972).

[Mar08] **Marić M.**, "Rešavanje nekih NP-teških hijerarhijsko-lokacijskih problema primenom genetskih algoritama", Doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet (2008).

[Meh92] **Mehrotra S.,** "On the Implementation of a Primal-Dual Interior Point Method", SIAM J. Optimization 2, pp. 575-601 (1992).

[Mel98] **Melanie M.**, "An Introduction to Genetic Algorithms", The MIT Press (1998).

[Mla97] **Mladenović N., Hansen P.**, "Variable neighborhood search", Computer & Ops Res. Vol. 24, No. 11. pp. 1097-1100 (1997).

[Noc99] **Nocedal J., Wright S. J.,** "Numerical Optimization", New York: Springer-Verlag (1999).

[Ros06] **Rossi F., Beek P. v., Walsh T.**, "Handbook of Constraint Programming", Foundations of Artificial Intelligence (2006).

[Siv08] **Sivanandam S.N., Deepa S.N.**, "Introduction to Genetic Algorithms", Springer (2008).

[WSCpl] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[WSCsp] http://www.csplib.org/

[WSGaf] http://code.google.com/p/gaframework/

[WSMsfA] http://msdn.microsoft.com/en-us/devlabs/hh145003

[WSMsfB] http://solverfoundation.com/about.aspx

[WSNet] http://www.microsoft.com/net

[WSPic] http://fmv.jku.at/picosat/