UDC: 519.1:004.4 Info M: str. 25-30

GENERISANJE I REŠAVANJE VELIKIH SUDOKU ZAGONETKI SVOĐENJEM NA SAT PROBLEM
SOLVING AND GENERATING LARGE SUDOKU PUZZLES BY REDUCTION TO SAT

Mirko Stojadinović[1]

REZIME: Postoje dva osnovna problema vezana za Sudoku zagonetke: problem rešavanja i problem generisanja. U većini radova čija je tema rešavanje Sudokua, razmatraju se zagonetke dimenzije 9 x 9 i na njima se upoređuju nove sa postojećim metodama. Razlika u vremenima rešavanja za ovu dimenziju ne daje pravu sliku o efikasnosti različitih metoda jer se vreme rešavanja meri u delovima sekunde. Nekoliko metoda je razvijeno za rešavanje zagonetki većih dimenzija od 9 x 9: metoda paralelnog izvršavanja na više procesora, metoda simuliranog kaljenja i metoda koja koristi verovatnosne grafovske modele. U radu pokazujemo da je postojeće rešavanje Sudoku zagonetki svođenjem na Problem iskazne zadovoljivosti (SAT) značajno efikasnije od tri navedene metode. Glavni doprinos rada je unapređenje postojećeg algoritma za generisanje velikih i teških Sudoku zagonetki – one imaju jedinstveno rešenje, a uklanjanjem bilo kog početnog broja u generisanoj zagonetki svojstvo jedinstvenosti rešenja se gubi. Takođe, ispitujemo uticaj pravila preprocesiranja (i u procesu rešavanja i u procesu generisanja) koja se koriste za određivanje vrednosti nepoznatih polja pre svođenja na SAT. Preprocesiranje značajno ubrzava proces generisanja ali ne ubrzava proces rešavanja zagonetki. U radu je procenjen i pojedinačni značaj korišćenih pravila i prezentovane su neke od karakteristika generisanih zagonetki.

KLJUČNE REČI: Sudoku zagonetka, SAT, rešavanje, generisanje

ABSTRACT: Two problems related to Sudoku puzzles exist: the problem of solving and the problem of generating. In the most of the papers studying Sudoku, puzzles of size 9 x 9 are generated and solving methods are compared on these puzzles. The difference in solving times on these small-sized puzzles does not give a clear picture of the efficiency of different methods, as their solving times are measured in parts of the second. Few methods were developed for solving puzzles of greater size than 9 x 9: parallel solving, simulated annealing and method using probabilistic graphical models. We show that one well-known solving by reduction of Sudoku to SAT (Propositional satisfiability problem) significantly outperforms three mentioned methods. The main contribution of the paper is the improved existing algorithm for generating large and hard Sudoku puzzles – the property of the generated puzzles is uniqueness of the solution, and by removing any of the pre-filled numbers, this property is lost. We also evaluate the impact of the preprocessing rules (both in the process of solving and in the process of generating) used to deduce values of certain cells, prior to encoding to SAT. Preprocessing significantly speeds up the generating process but does not speed up solving of puzzles. We estimate the significance of the used rules and present some characteristics of the generated puzzles.

KEY WORDS: Sudoku puzzle, SAT, solving, generating

1 Ovaj rad je delimično finansiran od strane Ministarstva prosvete, nauke i tehnološkog razvoja, projekat 174021

1 UVOD

Postoje mnogi računarski-zasnovani pristupi za rešavanje realnih problema (planiranje, raspoređivanje, itd.). Poboljšanje pristupa za rešavanje zagonetki, igara i kombinatornih zadataka obično povlači i uspešniju primenu ovih pristupa u rešavanju realnih problema. Zato postoji veliko interesovanje za generisanje teških zagonetki, igara i kombinatornih zadataka i sve tri vrste problema su često deo poznatih i dosta korišćenih sistema, kao što su Minizinc 1.6 [11], Sugar [15], itd. Da bi se u nekoj oblasti izvršilo poređenje među razvijenim sistemima/rešavačima te oblasti i odredilo koji je efikasniji (i samim tim verovatno pogodniji za rešavanje realnih problema), redovno se održavaju takmičenja u tim oblastima (npr. SAT takmičenje[1], CSP takmičenje[2], itd.). Zagonetke, igre i kombinatorni zadaci obično čine značajan deo korpusa instanci na kojima se vrši poređenje sistema koji su učesnici tih takmičenja. Zato u procesu evaluacije sistema na takmičenjima možemo razlikovati dve faze: generisanje instanci na kojima će se vršiti poređenje sistema i samo poređenja tih sistema.

1 http://baldur.iti.kit.edu/sat2014/competitions.html

2 http://www.cril.univ-artois.fr/CPAI09

Sudoku zagonetke su doživele veliku popularnost u poslednjih nekoliko decenija. Zagonetka se sastoji od table dimenzije n x n koja je podeljena u blokove dimenzija m x m (m x $m = n$) na kojoj se nalaze početni brojevi koji se ne mogu ni menjati ni pomerati. Zagonetka je rešena kada je tabla popunjena tako da svaki red, kolona i blok sadrže različite brojeve od 1 do n.

Razvijen je veliki broj metoda za rešavanje Sudoku zagonetki pomoću papira i olovke. Takođe, postoji na desetine (ako ne i stotine) radova na temu rešavanja ovih zagonetki pomoću računara. Skoro svi radovi se odnose na rešavanje zagonetki dimenzije 9 x 9. Iako ove zagonetke mogu biti vrlo zahtevne kada se rešavaju pomoću papira i olovke, čak i jednostavni bektreking algoritmi na savremenim računarima rešavaju i najteže među njima za nekoliko sekundi. Postoji samo nekoliko pristupa razvijenih za rešavanje Sudoku zagonetki dimenzija većih od 9 x 9 i mi ćemo se u ovom radu baviti samo ovim zagonetkama.

Među postojećim metodama za rešavanje Sudoku zagonetki dimenzija većih od 9 x 9 razlikujemo egzaktne metode [3,6,7] i heuristike [8,10]. Jedan od načina rešavanja Sudoku zagonetki je *rešavanje svođenjem na SAT*, tj. *Problem iskazne zadovoljivosti* (Propositional satisfiability problem). Problem

se sastoji u ispitivanju postojanja vrednosti Bulovskih promenljivih takvih da je zadata logička formula zadovoljiva. Rešavanje svođenjem na SAT se sastoji iz dve faze: 1) *svođenja na SAT* – u ovoj fazi se ograničenja Sudoku problema prevode u logičku formulu 2) *rešavanja* – logička formula se prosleđuje nekom od SAT rešavača (besplatni programi koji su dostupni za preuzimanje sa Web-a) i ako rešavač pronađe rešenje, onda se ono prevodi u rešenje zadate zagonetke. Najpoznatije svođenje na SAT Sudoku zagonetki dimenzije 9 x 9 su razvili Linc i ostali [9]. Ovo svođenje je dalje unapređeno korišćenjem tehnika propagiranja ograničenja i Sudoku zagonetke različitih dimenzija su korišćene za testiranje [6]. Na žalost, autori nisu opisali svojstva korišćenih zagonetki, a efikasnost upotrebljenog načina rešavanja nije upoređena ni sa jednom drugom metodom (rešavanje Sudoku zagonetki nije u fokusu tog rada). Simulirano kaljenje je korišćeno za rešavanje zagonetki dimenzija do 25 x 25 [8], a ova metoda je u novom radu i unapređena [10]. Za rešavanje Sudoku zagonetki dimenzija do 16 x 16 korišćeno je i paralelno izvršavanje na više procesora istovremeno [3]. Verovatnosni grafovski modeli su upotrebljavani za rešavanje Sudoku zagonetki dimenzija do 16 x 16 [5].

Pre poređenja metoda na Sudoku zagonetkama, potrebno je generisati skup pogodnih zagonetki. Kao što je već pomenuto, zbog vremena rešavanja koje se obično meri delovima sekunde, poređenje metoda na zagonetkama dimenzije 9 x 9 ne daje pravu sliku njihove efikasnosti. Generisanje teških zagonetki većih dimenzija je otvoren problem i vrlo mali broj radova se do sada bavio ovom problematikom. Poznate su nam dve metode za generisanje velikih Sudoku zagonetki. Kod prve [8] se generišu zagonetke različitih nivoa popunjenosti ali ne postoji garancija jedinstvenosti rešenja. Druga metoda [3] garantuje jedinstvenost rešenja generisanih zagonetki.

Generisanje teških Sudoku zagonetki je od interesa i za sisteme opšte namene, koji rešavaju mnogo širi skup problema od Sudoku zagonetki. *Programiranje ograničenja* [16] je oblast koja se bavi izučavanjem i rešavanjem problema koji su zadati pomoću niza ograničenja nad promenljivama. Pre rešavanja bilo kog problema potrebno je modelirati taj problem, a pri modeliranju se određuju ograničenja koje rešenje problema mora da zadovoljava. Važno ograničenje pri modeliranju mnogih problema je *alldifferent*, i ono obezbeđuje da svi argumenti moraju da uzmu međusobno različite vrednosti. Za ovo ograničenje postoji veliko interesovanje naučne zajednice, pa je razvijen značajan broj algoritama i napisan veliki broj radova koji ga proučavaju (npr. jedan pregled radova koji se odnose ne *alldifferent je* obavio Jan van Hufe [17]). Da bi se poredila efikasnost algoritama razvijenih za *alldifferent*, potrebne su instance problema koji se modeliraju pomoću velikog broja pojavljivanja ovog ograničenja. Sudoku se lako i prirodno modelira samo pomoću *alldifferent* ograničenja o čemu je još 2005. godine pisao Simonis u jednog od prvih radova na temu modelovanja i rešavanja Sudoku zagonetki pomoću programiranja ograničenja [13]. Naime, uvode se n^2 promenljive sa istim domenom $\{1, ..., n\}$ gde svaka promenljiva odgovara jednom polju zagonetke i postavljaju se ograničenja da u svakom redu/koloni/bloku pripadajuće promenljive moraju uzimati međusobno različite vrednosti. Stoga je Sudoku pogodan problem za poređenje različitih algoritama koji se odnose na ovo ograničenje (npr. pogledati rad Bankovića i ostalih [1]).

Doprinosi ovog rada su sledeći:

- Sprovedeni su testovi sa ciljem da se uporedi najpoznatije postojeće rešavanje Sudoku zagonetki svođenjem na SAT [6] sa drugim savremenim metodama razvijenim za rešavanje zagonetki većih dimenzija (paralelno izvršavanje na više procesora [3], simulirano kaljenje [10] i metoda koja koristi verovatnosne grafovske modele [5]). Koliko nam je poznato, ovo svođenje nije do sada detaljno testirano. Rezultati pokazuju da je rešavanje svođenjem na SAT značajno efikasnije na zagonetkama dimenzija 16 x 16 i 25 x 25 od tri postojeće metode.
- Preprocesiranje je izvršeno sa ciljem da se dobije kompaktnije svođenje na SAT. Sprovedeni su testovi sa ciljem da se ispita uspešnost preprocesiranja prilikom rešavanja.
- Unapređen je (u pogledu potrebnog vremena) jedan postojeći algoritam za generisanje Sudoku zagonetki proizvoljne dimenzije sa jedinstvenim rešenjem. Sa korišćenjem preprocesiranja, vreme potrebno za generisanje se dodatno značajno smanjuje. Predstavljene su neke od karakteristika generisanih instanci kao i procena korisnosti korišćenih pravila preprocesiranja. Naš sistem i generisane zagonetke su javno dostupni i time je omogućeno lako poređenje sa drugim sistemima u budućnosti.

2 REŠAVANJE SUDOKU ZAGONETKI SVOĐENJEM NA SAT I PREPROCESIRANJE

SAT problem.

Literal je ili Bulovska promenljiva ili njena negacija. Klauza je disjunkcija literala i razmatra se konačan skup klauza. SAT (Problem iskazne zadovoljivosti) se sastoji u ispitivanju da li je iskazna formula zadata konačnim skupom klauza (ovaj oblik se zove Konjunktivna normalna forma, skraćeno KNF) zadovoljiva, tj. da li postoji dodeljivanje vrednosti promenljivama tako da su sve klauze tačne. SAT je prvi problem za koji je pokazano da je NP-kompletan i on i danas zauzima centralno mesto u teoriji izračunljivosti. Postoje mnoge praktične primene ovog problema, npr. u oblasti hardverske i softverske verifikacije, planiranju, raspoređivanju, itd. Prethodnih decenija napravljen je ogroman napredak u tehnologiji SAT rešavanja čime je omogućeno da se danas rešavaju industrijski SAT problemi sa stotinama hiljada promenljivih i milionima klauza. Kada se neki problem rešava pomoću SAT tehnologije, onda je potrebno prvo modelirati taj problem, zatim odrediti svođenje na SAT i na kraju pokrenuti jedan od mnogih besplatnih i javno dostupnih SAT rešavača. Ovi rešavači su programi koji primaju iskaznu formulu (najčešće kao datoteku) u DIMACS formatu[3] i vraćaju vrednosti promenljivih ako je formula za-

3 ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/doc/satformat.dvi

dovoljiva, a odgovor da takve vrednosti promenljivih ne postoje u slučaju da je formula nezadovoljiva. SAT rešavači su prenosivi i jednostavno ih je pokrenuti na bilo kom današnjem računaru.

Primer: Neka je dat skup klauza $x \vee y' \vee z$, $x' \vee z'$ i $y \vee x$ (x' je negacija promenljive x). Potrebno je rešiti SAT problem, tj. odrediti vrednost promenljivama x, y i z tako da sve klauze budu tačne. Jedno od rešenja ovog problema je: $x = 1$, $y = 0$, $z = 0$ (0 odgovara vrednosti *netačno,* a 1 odgovara vrednosti *tačno*).

Svođenje problema Sudoku na SAT.

Najpoznatije svođenje na SAT Sudoku problema [6] se sastoji u tome da se za svako polje *(r, k)* (r i k su red i kolona, tj. $1 <= r <= n$, $1 <= k <= n$) i za svaku moguću vrednost v ($1 <= v <= n$) koja se smešta u zagonetku, uvodi Bulovska promenljiva $x_{r,k,v}$ (tačna je, tj. uzima vrednost 1 ako i samo ako je vrednost v smeštena na polje *(r, k)*). Skup ovako uvedenih promenljivih ćemo označiti sa *X*. Promenljive koje se odnose na početno popunjena polja i promenljive koje ne mogu da budu tačne jer je vrednost kojoj odgovaraju već smeštena u isti red/kolonu/blok se eliminišu (obe ove grupacije promenljivih imaju svoju istinitosnu vrednost već određenu). Koristeći ovu eliminaciju, ostaje skup preostalih (nepoznatih) promenljivih koji ćemo označiti sa *Y* (svaka preostala promenljiva $x_{r,k,v}$ je preimenovana u $y_{r,k,v}$). Cilj eliminacije promenljivih je da se smanji veličina instance koja će se proslediti SAT rešavaču, što može značajno ubrzati vreme rešavanja. Autori navedenog svođenja u svom prethodnom radu [7] navode da se ovom eliminacijom broj klauza može smanjiti i do 79 puta.

U cilju dodatne eliminacije promenljivih, uvodimo vrlo jednostavna *pravila preprocesiranja* koja će biti primenjena na zagonetku, pre uvođenja Bulovskih promenljivih. Pravila odgovaraju zaključivanju koje se vrši pri rešavanju zagonetki pomoću papira i olovke. Za svako nepopunjeno polje, održava se lista svih vrednosti koje je moguće staviti na to polje. Primenjujemo sledeća pravila:

- *Jedna preostala vrednost.* Vrednost svakog polja na koje može da se smesti samo jedna vrednost v je određena, i vrednost ovog polja se postavlja na v.
- *Jedno preostalo polje.* Ako se neka vrednost v može smestiti na tačno jedno polje u nekom redu/koloni/bloku, onda se vrednost ovog polja postavlja na v.
- *Putna konzistentnost* (termin potiče iz rada [10] koji je već koristio ovo pravilo). Ako tri polja u istoj vrsti/koloni/bloku imaju svojstvo da se na dva od njih mogu smestiti samo vrednosti v_1 i v_2, a na treće samo vrednosti v_1, v_2 i v_3, onda se u treće polje smešta vrednost v_3 jer se vrednosti v_1 i v_2 moraju smestiti u prva dva polja.

Iterativno primenjujući navedena pravila dok se nijedno pravilo više ne može primeniti u zagonetki, a tek onda primenjujući opisanu eliminaciju promenljivih, dobija se još manji skup nepoznatih promenljivih *Z*, u odnosu na prethodno opisani skup *Y* (svaka preostala promenljiva $y_{r,k,v}$ je preimenovana u $z_{r,k,v}$). Ceo ovaj postupak se smatra *preprocesiranjem*.

Za proizvoljnih k Bulovskih promenljivih označenih sa c_1, ..., c_k ograničenje oblika $c_1 + ... + c_k = 1$ (tačno jedna od promenljivih c_1, ..., c_k je tačna) se zove *ograničenja kardinalnosti za najviše jedan* i postoji veliki broj načina da se ono efikasno prevede[4] u klauze SAT problema (npr., [2,6]). Na preostalim promenljivima *Z*, opisanim u prethodnom pasusu, uvode se ograničenja kardinalnosti za najviše jedan. Ograničenja koja se uvode i koja predstavljaju svođenje na SAT su:

- Za svako nepopunjeno polje, tačno jedna od pridruženih promenljivih kojima još nije određena vrednost je tačna.
- Za svaku vrednost i svaki red/kolonu/blok gde ta vrednost još nije smeštena, tačno jedna od svih pridruženih promenljivih toj vrednosti u tom redu/koloni/bloku je tačna. Ovo ograničenje se postavlja samo nad promenljivama u redu/koloni/bloku koje još nemaju dodeljenu vrednost, tj. samo na promenljivama skupa *Z*. Na primer, za Sudoku dimenzija 16 x 16, vrednost 3 i red 1, pri čemu u prvom redu samo polje 1 ima dodeljenu vrednost 2, postavlja se ograničenje $z_{1,2,3} + ... + z_{1,16,3} = 1$ (3 se nalazi na tačno jednom mestu u prvom redu), tj. promenljiva koja se odnosi na polje (1,1) se preskače pošto već ima dodeljenu vrednost 2.

Kao što je već rečeno, dobijena SAT formula se prosleđuje SAT rešavaču koji izračunava vrednosti promenljivih u slučaju zadovoljivosti formule (zagonetka ima rešenja) ili vraća odgovor da formula nije zadovoljiva (zagonetka nema rešenja).

3 GENERISANJE SUDOKU ZAGONETKI

Postojeći algoritam za generisanje zagonetki [3] polazi od rešenja zagonetke koje zovemo *početno rešenje*. Ovo rešenje se više puta može koristiti kao ulaz koristeći permutacije redova i kolona, permutacije kvadratnih redova i kolona (sastoje se iz blokova) i rotacijom table (ova pravila već su opisivana u ranijim radovima [3, 10]). Algoritam na slučajan način bira polje (*r, k*) i uklanja vrednost iz tog polja. Izmenjena zagonetka se rešava, sa dodatnim ograničenjem da se uklonjena vrednost ne može staviti na polje (*r, k*). Ako se pronađe rešenje izmenjene zagonetke, onda se uklanjanjem vrednosti broj rešenja povećao na barem dva (početno i pronađeno), pa se vrednost vraća u polje kako bi se zadržala jedinstvenost rešenja. Inače, vrednost ostaje uklonjena, ne uticajući na jedinstvenost rešenja. Ovaj proces se ponavlja dok se zagonetka ne isprazni do nivoa popunjenosti kada se uklanjanjem bilo koje vrednosti gubi jedinstvenost rešenja. Na ovaj način, vrlo teške zagonetke se generišu [3]. Primetimo da jedno generisanje podrazumeva veliki broj rešavanja.

U cilju ubrzanja generisanja zagonetki, dodajemo neke korake pre opisanog postupka. Ovi koraci predstavljaju prvi deo algoritma generisanja, a postupak opisan u prethodnom pasusu drugi deo. Algoritam koji smo razvili zovemo *Modifikovano Generisanje.*

4 Termin *efikasno prevođenje* ovde označava da su dobijene klauze "pogodne" za SAT rešavače, tj. da SAT rešavači imaju specijalno razvijene algoritme pomoću kojih efikasno rešavaju instance koje se sastoje iz tih klauza.

Pseudokod prvog dela generisanja je prikazan na Slici 1. U prvoj petlji se na slučajan način uklanjaju vrednosti iz kopije početnog rešenja, sve dok se procenat popunjenih polja ne smanji na neki stepen (npr. 20%). U ovom trenutku, skoro uvek postoji više rešenja zagonetke[5]. U drugoj petlji zagonetka se rešava uz uslov da se traži rešenje različito od početnog. Ako je novo rešenje pronađeno, lociraju se polja koja sadrže vrednosti različite od vrednosti početnog rešenja. Da bi se zabranilo novo rešenje, ova polja se popunjavaju vrednostima početnog rešenja. Koraci rešavanja zagonetke uz zabranu početnog rešenja i popunjavanja novim vrednostima se ponavljaju dok početno rešenje ne postane jedinstveno rešenje zagonetke (u generisanjima koje smo mi izvršavali procenat popunjenosti zagonetke u ovom trenutku generisanja je bio oko 50%).

Generiši_Sudoku_sa_jedinstvenim_rešenjem (rešenje)

```
{
    Kopiraj (zagonetka, rešenje);
    while (Izračunaj_procenat_popunjenosti (zagonetka) > 20)
      (i, j) = Vrati_slučajno_odabranu_popunjenu_poziciju (zagonetka);
      zagonetka [i][j] = prazna_pozicija;
    while (1)
      novo_rešenje = Reši_zabranjujući_rešenje (zagonetka, rešenje);
      if (novo_rešenje == UNSAT)
                break;
      Popuni_polja_koja_se_razlikuju (zagonetka, rešenje, novo_rešenje);
    return (zagonetka, rešenje);
}
```

Slika 1: *Generisanje zagonetke sa jedinstvenim rešenjem – na ulazu je popunjena tabla koja će predstavljati jedinstveno rešenje zagonetke koja će biti generisana.*

Prikazani algoritam je opšti i može se koristiti sa bilo kojom metodom rešavanja. Ime funkcije staviti u neki drugačiji font, ako je moguće isti kao onaj koji je korišćen za pseudokod sastoji se od već opisanog uvođenja Bulovskih promenljivih i preprocesiranja, generisanja SAT klauza, dodavanja klauze koja zabranjuje početno rešenje i pokretanja SAT rešavača. Na primer, ako su prazna polja na tabli samo (1,1), (2,2) i (3,3) a početno rešenje na ove pozicije smešta redom 4, 7 i 9, onda bi klauza koja zabranjuje to rešenje bila $z_{1,1,4}' \vee z_{2,2,7}' \vee z_{3,3,9}'$ (barem jedna od vrednosti 4, 7 i 9 nije na odgovarajućem polju, tj. novo rešenje na barem jednoj poziciji mora da se razlikuje od početnog, što je u ovom slučaju nemoguće).

Posle prvog dela algoritma primenjuje se drugi deo koji je već opisan na početku ove sekcije. Jedina modifikacija je da se polje (r, k) bira prolazeći redom kroz sva popunjena polja jedno po jedno, a ne na slučajan način kao u originalnom algoritmu. Na ovaj način se garantuje da će generisana zagonetka biti što je moguće manje popunjena. Cilj modifikacije je da se rešavanjem manjeg broja zagonetki ubrza proces generisanja. Pseudokod drugog dela generisanja je prikazan na Slici 2.

5 Prilikom vršenja eksperimenata čiji su rezultati prikazani u narednom poglavlju, vrlo su retki bili slučajevi kada nije postojalo više rešenja, i u tim slučajevima potrebno je dodatno isprazniti zagonetku.

Isprazni_Sudoku_do_manje_popunjenog (zagonetka, rešenje)

```
{
    while ( ((i, j) = Vrati_sledeće_popunjeno_polje (zagonetka)) != kraj)
      zapamćena_vrednost = zagonetka [i][j];
      zagonetka [i][j] = prazna_pozicija;
      novo_rešenje = Reši_zabranjujući_rešenje (zagonetka, rešenje);
      if (novo_rešenje != UNSAT)
                zagonetka [i][j] = zapamćena_vrednost;
    return (zagonetka, rešenje);
}
```

Slika 2: *Kreiranje manje popunjene zagonetke koja će imati jedinstveno rešenje.*

4 EKSPERIMENTALNI REZULTATI

Svi testovi su vršeni na višeprocesorskom računaru sa 48 procesora AMD Opteron(tm) CPU 6168 na 1.9Ghz, sa 2GB RAM memorije po procesoru, pod sistemom Linuks. Ograničenja kardinalnosti su svedena na SAT korišćenjem *komandant kodiranja* [6] implementiranih u okviru sistema *meSAT* [14]. Ovaj sistem rešava probleme svođenjem na SAT i pokretanjem SAT rešavača, pri čemu se svođenje može uraditi na više različitih načina. Rešavač MiniSat 2.2 [4] je korišćen za rešavanje SAT instanci. *Napomena:* Nadalje će se termin *rešavanje svođenjem na SAT* odnositi na prevođenje problema Sudoku u SAT formulu i pokretanje SAT rešavača MiniSat.

Eksperimenti koji se odnose na rešavanje.

Najveći problem pri poređenju sa postojećim pristupima koji su razvijeni za rešavanje velikih Sudoku zagonetki je da najčešće ni izvorni kod a ni korišćene zagonetke nisu javno dostupni. Da bi omogućili drugima poređenje sa našim pristupom, sistem je zajedno sa svim korišćenim zagonetkama javno dostupan na adresi *http://jason.matf.bg.ac.rs/~mirkos/Sudoku.html*.

Generisali smo instance na isti način kao u već postojećem radu autora Maćado i ostali [10]. U tom radu je heuristički pristup koji je koristio *simulirano kaljenje* ostvario najbolje rezultate. Počevši od rešene zagonetke i fiksirane vrednosti p, generisanjem se vrednost svakog polja uklanja sa verovatnoćom p i na taj način dobija zagonetka koju treba rešiti. Generisanja se pokreću za različite vrednosti p, pri čemu p uzima vrednosti između 0.05 i 1.0, sa korakom 0.05. Što je vrednost p veća, to je manji broj popunjenih polja u zagonetki (1.0 odgovara potpuno praznoj zagonetki). Počevši od različitih rešenih zagonetki, za svaku od 20 vrednosti p generisali smo po 20 zagonetki. Na ovaj način je generisano po 400 zagonetki za svaku od dimenzija 16 x 16, 25 x 25 i 36 x 36. Tabela 1 prikazuje rezultate poređenja originalnog, najpoznatijeg rešavanja svođenjem na SAT [6] sa našom verzijom u kojoj je korišćeno preprocesiranje pre svođenja. Rezultati pokazuju da ne postoji bitna razlika između originalnog svođenja i onog koje koristi

preprocesiranje. Rešavanje originalnim svođenjem je čak rešilo jednu instancu više dimenzije 36 x 36, ali pošto je razlika vrlo mala, ona se može pripisati slučajnim varijacijama u rešavanju pomoću SAT rešavača[6].

Izvorni kod pristupa koji koristi simulirano kaljenje nije javno dostupan i naš cilj je bio da prevaziđemo problem poređenja tako što smo koristiti slabiji računar i veliki broj zagonetki generisanih na isti način kao u originalnom radu (opisano u prethodnom pasusu). Oba rešavanja svođenjem na SAT su rešila svaku od generisanih instanci dimenzije 25 x 25 za najviše 1.6 sekunde. Pošto je u originalnom radu [10] navedeno da simulirano kaljenje nije rešilo više od 80% zagonetki dimenzije 25 x 25 za vremensko ograničenje od 350 sekunde po zagonetki (na jačem računaru), možemo zaključiti da je rešavanje svođenjem na SAT značajno efikasnije od pomenute heuristike pri rešavanju Sudoku zagonetki.

Dimenzija	Broj instanci	Originalno svođenje	Svođenje sa preprocesiranjem
16 x 16	400	400 (61)	400 (61)
25 x 25	400	400 (163)	400 (158)
36 x 36	400	384 (7231)	383 (7216)

Tabela 1: *Poređenje rešavanja originalnog svođenja na SAT sa rešavanjem svođenjem na SAT koje koristi preprocesiranje. Vremensko ograničenje po zagonetki je* 300 *sekundi. U svakom polju su prikazani broj rešenih instanci i u zagradi ukupno utrošeno vreme u sekundama.*

Verovatnosni grafovski modeli su takođe upotrebljavani za rešavanje Sudoku zagonetki [5]. Efikasnost ovog pristupa u originalnom radu je testirana na zagonetkama različitih stepena popunjenosti – pristup nije uspeo da reši neke zagonetke dimenzije 16 x 16. Iako nismo uspeli da dobijemo zagonetke koje su originalno korišćene (veb stranica navedena u radu nije bila dostupna), rezultati u Tabeli 1 su dati na zagonetkama vrlo raznolikih stepena popunjenosti. Pošto su oba rešavanja svođenjem na SAT rešila sve zagonetke dimenzije 16 x 16 u proseku za 61/400=0.15 sekundi, zaključujemo da je rešavanje svođenjem na SAT značajno efikasnije.

Takođe smo poredili efikasnost rešavanja svođenjem na SAT sa paralelnim rešavanjem Sudoku zagonetki [3] na 20 zagonetki generisanih na način opisan u originalnom radu, pošto originalne zagonetke nisu javno dostupne. Paralelno rešavanje je rađeno na jačem računaru koji je koristio više jezgara istovremeno, a prosečno vreme na zagonetkama dimenzije 16 x 16 je mereno u sekundama. Oba rešavanja svođenjem na SAT su rešila svaku od 20 generisanih zagonetki i to u proseku za 0.11 sekundi, pa smatramo da su ovo značajno bolji rezultati u odnosu na paralelno rešavanje. Uzimajući u obzir da se rešavanje svođenjem na SAT izvršava samo na jednom a paralelno izvršavanje na više jezgara istovremeno, rezultati u korist rešavanja svođenjem na SAT su još ubedljiviji.

6 Rešavanje pomoću SAT rešavača uvek ima određenu dozu nepredvidivosti i na njega se može uticati trivijalnim promenama na ulaznim instancama (npr. promenom redosleda klauza). Zato postoji određena doza slučajnosti pri SAT rešavanju. Statistički pristup poređenju SAT rešavača dali su Nikolić i ostali [12].

Eksperimenti koji se odnose na generisanje.

Poredili smo originalni algoritam za generisanje velikih zagonetki sa našim unapređenim algoritmom (oba opisana u poglavlju 3). Naš algoritam smo koristili sa i bez preprocesiranja. Za svaki algoritam smo generisali po 100 zagonetki dimenzija 16 x 16 i 25 x 25. Prosečno vreme generisanja je prikazano u Tabeli 2. Naš algoritam bilo bez ili sa preprocesiranjem pre svođenja na SAT se pokazao značajno efikasnijim od originalnog algoritma. Generisanje uz pomoć preprocesiranja se pokazalo efikasnijim od generisanja bez preprocesiranja. Generisanje pomoću bilo kog od tri algoritma za zagonetke dimenzije 36 x 36 nije bilo završeno ni posle više sati, pa nisu prikazani rezultati za ovu dimenziju. Potrebno je razviti efikasniji algoritam za generisanje zagonetki ove i većih dimenzija u budućnosti.

Dimenzija	Originalno generisanje	Modifikovano generisanje	Modifikovano generisanje sa preprocesiranjem
16 x 16	845	17	13
25 x 25	2916	372	283

Tabela 2: *Poređenje originalnog algoritma za generisanje sa dve njegove modifikacije, jednom koja ne koristi preprocesiranje i drugom koja ga koristi. Za svaku dimenziju i svaki algoritam dato je prosečno vreme generisanja u sekundama na* 100 *zagonetki.*

Karakteristika	Originalno generisanje	Modifikovano generisanje	Modifikovano generisanje sa preprocesiranjem
Popunjenost zagonetke	36 / 43	35 / 41	35 / 41
Broj rešavanja svođenjem na SAT	257 / 626	135 / 319	127 / 311
Broj korišćenja pravila *Jedna preostala vrednost*	-	-	402 / 343
Broj korišćenja pravila *Jedno preostalo polje*	-	-	2973 / 9375
Broj korišćenja pravila *Putna konzistentnost*	-	-	26 / 7

Tabela 3: *Neki podaci generisanja zagonetki dimenzija* 16 x 16 / 25 x 25. *Za svaku dimenziju date su prosečne vrednosti za* 100 *generisanja.*

Tabela 3 prikazuje prosečne vrednosti nekih podataka vezanih za proces generisanja. Broj rešavanja svođenjem na SAT i samim tim i broj pokretanja SAT rešavača je značajno smanjen u odnosu na originalni algoritam (skoro duplo). Razlog tome je da je posle prvog dela procesa generisanja procenat popunjenosti bio oko 50% - rad na tome da se ova zagonetka u drugom delu što je moguće više isprazni podrazumeva značajno manji broj zagonetki koje je potrebno rešiti, u poređenju sa pražnjenjem pune zagonetke kao u originalnom algoritmu. U slučaju kada se koristi preprocesiranje, broj rešavanja svođenjem na SAT je manji nego u slučaju kada se preprocesiranje ne koristi – razlog je što se samo pomoću preprocesiranja neke zagonetke u potpunosti rešavaju. Od korišćenih pravila preprocesiranja najviše je korišćeno pravilo *Jedno preostalo polje* dok je najmanje korišćeno pravilo *Putna konzistentnost*. Ako bi se od navedena 3 pravila koristilo samo jedno, onda pravilo

Jedno preostalo polje ima ubedljivo najveći uticaj na smanjenje vremena generisanja (ovo je potvrđeno eksperimentalnim rezultatima koji ovde nisu prikazani). Međutim, korišćenje sva 3 pravila ima prednost da izračunavanje vrednosti nekog polja u zagonetki pomoću jednog od pravila povlači moguće izračunavanje vrednosti nekog drugog polja pomoću nekog drugog pravila. Na primer, ako bi se koristilo samo pravilo *Jedna preostala vrednost* pri rešavanju zagonetki dimenzija 25 x 25, onda bi broj primena tog pravila bio značajno smanjen, sa prosečnih 343 datih u Tabeli 3 na prosečnih 164. Iz ove činjenice sledi da pri primeni pravila preprocesiranja treba koristiti kombinacije različitih pravila, a detaljnija procena efikasnosti različitih kombinacija jeste jedan mogući pravac daljeg rada.

5 ZAKLJUČCI

U radu smo razmatrali dva problema vezana za Sudoku zagonetke: rešavanje i generisanje.

Poredili smo najpoznatije postojeće rešavanje Sudoku zagonetki svođenjem na SAT sa pristupima razvijenim za rešavanje velikih Sudoku zagonetki. Iako nisu korišćene iste zagonetke kao u originalnim radovima, one su generisane na isti način i eksperimenti pokazuju da je rešavanje svođenjem na SAT značajno efikasnije od tri postojeće metode za rešavanje velikih Sudoku zagonetki.

Unapredili smo postojeći algoritam za generisanje Sudoku zagonetki dimenzija većih od 9 x 9 koje imaju jedinstveno rešenje, a uklanjanjem vrednosti sa bilo kog polja, jedinstvenost rešenja bi se izgubila. Unapređeni algoritam značajno skraćuje vreme potrebno za generisanje.

Prikazali smo vrlo jednostavna pravila preprocesiranja koja su iskorišćena da se smanji veličina formule koja je dobijena svođenjem Sudoku problema na SAT. U procesu rešavanja uvedena pravila ne unapređuju efikasnost originalnog rešavanja svođenjem na SAT ali u procesu generisanja doprinose značajnom poboljšanju efikasnosti. Dati su i podaci vezani za stepen upotrebe korišćenih pravila preprocesiranja tokom generisanja. Naizmenično korišćenje različitih pravila ima prednost da se ona mogu naizmenično primenjivati, što ima za posledicu poboljšanje efikasnosti. Moguć pravac daljeg rada je procenjivanje kombinacija pravila koje daju najbolje rezultate.

U budućnosti ćemo pokušati da efikasno rešimo i generišemo još veće Sudoku zagonetke. Inkrementalno rešavanje koje omogućava da se SAT rešavačima proslede nove klauze i da on nastavi sa rešavanjem (bez pokretanja rešavača ispočetka) takođe deluje kao interesantan pravac daljeg rada. Planiramo da razvijemo nova pravila preprocesiranja, specijalno razvijena za zagonetke većih dimenzija od 9 x 9. Ova pravila bi mogla da ubrzaju kako proces rešavanja, tako i proces generisanja Sudoku zagonetki.

ZAHVALNOST

Autor želi da se zahvali Filipu Mariću na korisnim komentarima prvih verzija rada.

LITERATURA

[1] Milan Banković and Filip Marić. An alldifferent constraint solver in smt. In *8th International Workshop on Satisfiability Modulo Theories*, 2010.

[2] Jingchao Chen. A new sat encoding of the at-most-one constraint. In Proceedings of the 9th *International Workshop on Constraint Modelling and Reformulation*, 2010.

[3] Alton Chiu, Ehsan Nasiri, and Rafat Rashid. Parallelization of sudoku. 2012. Technical report. http://individual.utoronto.ca/rafatrashid/Projects/2012/SudokuReport.pdf

[4] Niklas Een and Niklas Sorensson. An extensible sat-solver. In *SAT*, volume 2919 of Lecture Notes in Computer Science, pages 502–518. Springer, 2003.

[5] Sheehan Khan, Shahab Jabbari, Shahin Jabbari, and Majid Ghanbarinejad. Solving sudoku using probabilistic graphical models. 2014. Technical report. http://www.ece.ualberta.ca/~-madjid/Files/Publications/TR081231.pdf

[6] Will Klieber and Gihwon Kwon. Efficient cnf encoding for selecting 1 from n objects. In Proc. *International Workshop on Constraints in Formal Verification*, 2007.

[7] Gihwon Kwon and Himanshu Jain. Optimized cnf encoding for sudoku puzzles. In Proc. *13th International Conference on Logic for Programming Artificial Intelligence and Reasoning* (LPAR2006), pages 1–5, 2006.

[8] Rhyd Lewis. Metaheuristics can solve sudoku puzzles. *J. Heuristics*, volume 13, issue 4, pages 387–401, 2007.

[9] Ines Lynce and Joel Ouaknine. Sudoku as a sat problem. In *ISAIM*, 2006.

[10] Marlos C. Machado and Luiz Chaimowicz. Combining metaheuristics and csp algorithms to solve sudoku. In *SBGames*, pages 124–131. IEEE, 2011.

[11] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *CP*, volume 4741 of Lecture Notes in Computer Science, pages 529–543. Springer, 2007.

[12] Mladen Nikolić. Statistical methodology for comparison of sat solvers. In SAT, volume 6175 of Lecture Notes in Computer Science, pages 209–222. Springer, 2010.

[13] Helmut Simonis. Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages. 13-27. 2005.

[14] Mirko Stojadinović and Filip Marić. mesat: multiple encodings of CSP to SAT. *Constraints*, volume 19, issue 4, pages 380–403, 2014.

[15] Naoyuki Tamura and Mutsunori Banbara. Sugar: A csp to sat translator based on order encoding. In *Proceedings of the third constraint solver competition*, pages 65–69, 2008.

[16] Mirko Vujošević. Programiranje ograničenja. *Info M*, volume 44, pages 4-10, 2012.

[17] Willem Jan van Hoeve. The alldifferent constraint: A survey. *CoRR*, cs.PL/0105015, 2001.

Mirko Stojadinović, asistent na Matematičkom fakultetu u Beogradu, Katedra za Računarstvo i informatiku
Kontakt: mirkos@matf.bg.ac.rs
Oblasti interesovanja: programiranje ograničenja, mašinsko učenje, SAT rešavači