Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Katja Tuma

Generiranje in reševanje sudokuja

DIPLOMSKO DELO

UNIVERZITETNI BOLONJSKI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVA IN INFORMATIKE

Mentor: izr. prof. dr. Branko Šter

Ljubljana 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Generiranje in reševanje sudokuja

Tematika naloge:

Raziščite pogoste pristope generiranja, reševanja in ocenjevanja težavnosti sudoku mrež. Napišite računalniški program, ki zna generirati in reševati sudoku mreže ter oceniti njihov nivo težavnosti (za človeka). Izmerite in primerjajte časovno zahtevnost posameznih zgoraj omenjenih nalog.

Angleški naslov naloge: Generating and solving Sudoku puzzles

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Katja Tuma, z vpisno številko **63100285**, sem avtorica diplomskega dela z naslovom:

Generiranje in reševanje sudokuja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom izr. prof. dr. Branka Štera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 5. septembra 2014 Podpis avtorja:

Rada bi se zahvalila svojem mentorju izr. prof. dr. Branku Šteru za strokovno svetovanje, potrpežljivost in spodbudo pri nastajanju diplomskega dela.

Želim se zahvaliti tudi družini in prijateljem, brez katerih bi bila moja študentska leta bistveno slabša.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
BIB	Balanced incomplete block - designs	Modeli uravnoteženih nepopolnih blokov
CRME	Column, row and minigrid elimination	Stolpična, vrstična ter eliminacija znotraj škatel
SEG	Sum of entropy for all cells in the sudoku grid	Vsota entropij vseh polj v mreži
GA	Genetic algorithm	Genetski algoritem

Povzetek

V diplomskem delu so raziskani različni pristopi in algoritmi generiranja, ocenjevanja ter reševanja sudoku mreže. Cilj je ugotoviti, kateri pristopi in algoritmi so za posamezen del programa bolj učinkoviti ter kako se med seboj razlikujejo. Matematična uganka sudoku je neke vrste latinski kvadrat, katerega lastnosti narekujejo obliko rešitve problema ter posledično število možnih rešitev problema. Generiranje mrež lahko poteka na dva nasprotna si načina: generiranje s polnenjem prazne mreže ter generiranje z brisanjem vrednosti iz polne mreže. Generiranje mreže je s strani računalnika podoben problem kot reševanje mreže. Ocenjevanje težavnostne stopnje mreže je delikaten problem, saj je potrebno upoštevati tudi načine človeškega reševanja. S pomočjo prebrane literature in programske rešitve sem tekom diplomskega dela potrdila ugotovitve o časovni zahtevnosti reševanja sudoku mrež ter uporabnosti ocenjevanja težavnostne stopnje mreže s pomočjo entropije.

Ključne besede: Latinski kvadrat, sudoku, generator, reševalnik, ocenjevalnik.

Abstract

The thesis explores different approaches and algorithms for generating, grading and solving the sudoku grid. The aim of the thesis is to determine which approaches and algorithms are more efficient and how they differ from each other. Mathematical puzzle sudoku is a kind of Latin square, features of which dictate the form of a solution and, consequently, the number of possible solutions to the problem. Generating grids can be done in two opposite ways: generating with filling an empty grid and generating with deleting values from a full grid. Generating a grid is from a computer's perspective similar to solving one. Determining the difficulty level of a grid is a delicate problem, since it is necessary to consider human techniques of solving sudoku. With the help of listed literature and software solution I confirm, during the thesis, the findings on time complexity of solving sudoku grids and usefulness of grading sudoku grids by using information entropy.

Key words: Latin square, sudoku, generator, solver, grader.

Poglavje 1

Uvod

Ime “sudoku” nosi japonske korenine ter je sestavljeno iz dveh besed - “su” (število) in “doku” (edino), kljub temu pa prihaja ideja sudokuja in prve mreže iz Evrope. Sudoku je ena izmed najbolj priljubljenih miselnih igric iz osemdesetih let. Kljub temu, da je za izurjeno človeško oko večina mrež hitro rešljivih, spada reševanje mreže v množico NP-polnih problemov. Namen diplomskega dela je ugotoviti, kateri algoritmi ter tehnike reševanja, generiranja in ocenjevanja sudoku mreže so najbolj primerni za računanje z mrežo velikosti 81 polj. Polnjenje in reševanje sudoku mrež lahko učinkovito opravi požrešna metoda, za boljše rezultate in hitrost programa pa je potrebna implementacija bolj naprednih algoritmov. Ocenjevanje težavnostne stopnje sudoku mrež je delikaten problem, saj mora dober ocenjevalnik upoštevati človeške tehnike reševanja mrež.

Poglavje 2

Pravila igre

Reševanje sudoku mreže je precej preprosto. Cilj igre je izpolniti prazna polja igralnega polja tako, da so števke v posameznih vrsticah, stolpcih in enotah unikatne. Igralno polje je ponavadi sestavljeno iz kvadratne mreže s stranico dolžine devetih polj, vendar so mreže drugačnih dimenzij tudi veljavne. Vsekakor ni vseeno, kolikšno število polj sestavlja dimenzijo mreže, saj mrežo sestavljajo manjše enote ali t.i. škatle. Za potrebe diplomskega dela veljajo naslednje definicije.

Definicija 1. *Dimenzija sudoku mreže n je poljubna, vendar mora zanjo veljati enačba 2.1. Za vsako število n in njegov koren mora veljati, da je element množice celih števil.*

$$\forall n; (n \in Z) \wedge (\sqrt{n} \in Z) \tag{2.1}$$

Definicija 2. *V primeru, da sudoku uganko sestavlja mreža velikosti 9×9 polj, so možne rešitve za prazna polja števke 1,2,3...,9. Torej rešitve x predstavlja zaprti interval celih števil 2.2. Za vsako rešitev x posameznega polja velja, da je element celih števil, hkrati pa je večje ali enako 1 ter manjše ali enako dimenziji n.*

$$\forall x; (x \in Z) \wedge (x \in [1, n]) \tag{2.2}$$

Vsaki mreži pripada težavnostna stopnja, ki v grobem določa, koliko časa naj bi igralec potreboval za reševanje mreže. Ponavadi težavnostne stopnje množico vseh možnih mrež razdelijo na 4 množice: lahke, srednje težke, težke in zlobne (angl. evil). Težavnostne stopnje se razlikujejo po številu danih polj in postavitvi le-teh po igralnem polju.

Poglavje 3

Latinski kvadrat

Sudoku mreža je posebna vrsta latinskega kvadrata, razumevanje katerega je pomembno za nadaljnjo obravnavo izračunov števila možnih mrež s pomočjo kombinatoričnih operacij. Latinski kvadrat (angl. latin square) je polje velikost $n \times n$, pri čemer je n celo število. Vrednost v vrstici se pojavi natanko enkrat, prav tako v stolpcu, kot kaže skica 3.1. Ime je povzeto iz matematičnih zapisov Leonarda Eulerja, ki je tedaj uporabljal latinske znake za prikaz vrednosti v kvadratu.

$$\begin{bmatrix} A & B & C \\ C & A & B \\ B & C & A \end{bmatrix} \tag{3.1}$$

Definicija 3. *Za posamezen numeričen element v latinskem kvadratu velja*

$$a_{ij} \equiv i + j - 1 \pmod{n} \qquad i, j = 1, 2, 3, ..., n, \qquad n \in Z \tag{3.2}$$

Z drugimi besedami, posamezen element a_{ij} latinskega kvadrata določa modul vsote indeksov pozicije elementa z zamikom velikosti 1, pri čemer je n dimenzija kvadrata in element množice celih števil. Kot primer opazujmo izračun elementov kvadrata dimenzije 3:

$$\begin{aligned} a_{11} &= 1 + 1 - 1 \pmod{3} = 1 \\ a_{12} &= 1 + 2 - 1 \pmod{3} = 2 \\ a_{13} &= 1 + 3 - 1 \pmod{3} = 3 \\ &... \end{aligned} \tag{3.3}$$

Na ta način pridobljeni elementi tvorijo naslednjo rešitev:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \tag{3.4}$$

3.1 Lastnosti

Vsak latinski kvadrat lahko iz predstavitvene oblike pretvorimo v normirano (standardno ali reducirano). V prvem stolpcu normirane oblike latinskega kvadrata si vrednosti sledijo v naraščajočem zaporedju. S pomočjo preimenovanja vrednosti ter permutiranja vrstic in stolpcev je mogoče vsak latinski kvadrat normirati:

$$\begin{bmatrix} A & B & C \\ C & A & B \\ B & C & A \end{bmatrix} \mapsto \begin{bmatrix} A & B & C \\ B & C & A \\ C & A & B \end{bmatrix} \tag{3.5}$$

Množica vseh možnih latinskih kvadratov je razdeljena na podmnožice ekvivalenčnih razredov. V ekvivalenčnem razredu so latinski kvadrati, pri katerih je možen prehod iz enega v drugega s pomočjo preprostih operacij:

(a) permutiranja vrstic,

(b) permutiranja stolpcev,

(c) permutiranja vrednosti v kvadratu (angl. relabeling),

(d) preslikavo preko obeh diagonal,

(e) preslikavo čez (vertikalno in horizontalno) polovico,

(f) rotacijo za $0°, 90°, 180°$ ali $270°$.

Definicija 4. *Ekvivalenčna relacija je relacija, za katero velja zakon o refleksivnosti, simetričnosti in tranzitivnosti. Za elemente a,b,c v ekvivalenčni množici A velja*

$$\begin{aligned} &\forall a \in A; a = a \\ &\forall a, b \in A; (a = b) \implies (b = a) \\ &\forall a, b, c \in A; (a = b) \wedge (b = c) \implies c = a \end{aligned} \tag{3.6}$$

Latinska kvadrata iz primera 3.5 tako spadata v isti ekvivalenčni razred. Število vseh reduciranih latinskih kvadratov je torej število različnih ekvivalenčnih razredov. Z naraščanjem

Tabela 3.1: Število vseh in reduciranih latinskih kvadratov določene dimenzije.

Dimenzija (n)	Število reduciranih latinskih kvadratov $R(n)$	Število vseh latinskih kvadratov $A(n)$
1	1	1
2	1	2
3	1	12
4	4	576
5	56	161.280
6	9.408	812.851.200
7	16.942.080	61.479.419.904.000
8	$5,35 \times 10^{11}$	$1,09 \times 10^{20}$
9	$3,78 \times 10^{17}$	$5,52 \times 10^{27}$
10	$7,58 \times 10^{24}$	$9,98 \times 10^{36}$
11	$5,36 \times 10^{33}$	$7,77 \times 10^{50}$

dimenzije latinskega kvadrata narašča zelo hitro tudi število ekvivalenčnih razredov, njihovih elementov, ter nekoliko počasneje, število reduciranih kvadratov. Števila vseh latinskih kvadratov iz Tabele 3.1 so pridobljena z Enačbo 3.7, vendar ta ne velja za latinske kvadrate z večjo dimenzijo [2].

$$A(n) = n!(n-1)! \times R(n) \tag{3.7}$$

3.2 Uporaba latinskega kvadrata pri računskih problemih

Latinski kvadrat je na videz zelo preprosta podatkovna strukutra, vendar kljub temu nosi pomebno vlogo v matematičnem, računalniškem ter raziskovalnem svetu. V zgodnjih letih dvajsetega stoletja so se latinski kvadrati izkazali za uporabne na področju eksperimentalnega oblikovanja (angl. experimental design: BIB - balanced inclomplete block - designs). Opazujmo naslednji primer. *Predpostavimo, da želimo ugotoviti, katero izmed petih zdravil A, B, C, D, E pripomora k zdravljenju kronične bolezni. Na voljo imamo 5 pacientov, vsak izmed pacientov bo pripravljen pet tednov sodelovati pri poskusu, torej je eksperimentalna enota v tem primeru pacient-teden. Strukturo eksperimentalnih enot smiselno ureja kvadratna mreža, kjer vrstice predstavljajo pacienta, stolpci pa zdravilo. S pomočjo latinskega kvadrata lahko hitro ugotovimo, kako učinkovito organizirati zdravljenje posameznega pacienta tako, da bo vsak pacient deležen vseh zdravil, in da bo*

v vsakem tednu poskušenih vseh pet zdravil.

$$\begin{bmatrix} A & B & C & D & E \\ B & A & D & E & C \\ C & E & A & B & D \\ D & C & E & A & B \\ E & D & B & C & A \end{bmatrix} \tag{3.8}$$

Podobno lahko uporaba take podatkovne strukture pripomore pri problemu, kot je t.i. problem poroke, dodeljevanja nalog, potem pri reševanju sudoku mrež ter nenazadnje pri dodeljevanju časovnih rezin procesorske moči v večjih paralelnih računalniških sistemih.

Latinski kvadrat se pojavi tudi v teoriji grafov. Vsak končen graf je mogoče predstaviti v obliki ortogonalnega latinskega kvadrata, s pomočjo katerega se ugotavlja ujemanje grafov [10].

Za izračun števila možnih latinskih kvadratov ne obstaja enačba, ki bi veljala za vse primere. Do sedaj sta najbolj natančno enačbo, ki določa zgornjo in spodnjo mejo števila možnih latinskih kvadratov velikih dimenzij, zapisala van Lint in Wilson leta 1992 [6] (stran 186):

$$\sum_{k=1}^{n} (k!)^{n/k} \geq L(n) \geq \frac{(n!)^{2n}}{n^{n^2}} \tag{3.9}$$

Zgornja in spodnja meja števila možnih latinskih kvadratov se z naraščanjem dimenzije kvadrata oddaljujeta. Na tem področju znanosti je še prostor za nova odkritja.

Poglavje 4

Število možnih mrež velikosti 9x9

Preden razmišljamo o algoritmu, ki generira sudoku mrežo, raje pomislimo, koliko takih mrež sploh obstaja. Števila možnih mrež ni tako preprosto določiti, čeprav se zdi na pogled preprost kombinatoričen problem. V prejšnjem poglavju sem obravnavala lastnosti latinskega kvadrata, ki je neke vrste skupni imenovalec vsem matematičnim ugankam, kot so sudoku, kenken ipd., s tem, da ima sudoku dodatno omejitev. Izkaže se, da je število možnih latinskih kvadratov dimenzije 9 kar $5524751496156892842531225600 \approx 5.52 \times 10^{27}$, kar je primerljivo z oceno mase Zemlje v gramih[1]. Kot sem omenila v prejšnem poglavju, sudoku mreže združujejo različni ekvivalenčni razredi, število katerih je veliko manjše.

V nadaljevanju se bom sklicevala na posamezne enote (škatle) sudoku mreže z oznakami B1-B9, ki predstavljajo posamezne dele mreže, kot prikazuje Slika 4.1. Metoda izračuna števila možnih mrež je povzeta po članku *Mathematics of sudoku I* [4]. Število esencialno različnih sudoku mrež je pridobljeno postopoma s pomočjo:

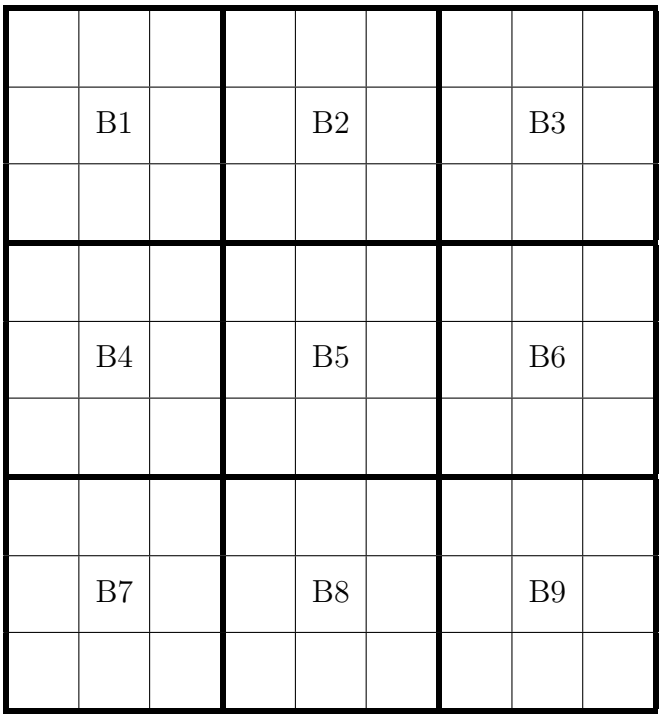
1. Preimenovanja vrednosti polj do standardne oblike B1,
2. Polnenja B2 in B3 ter redukcije,
3. Polnenja B4, B5, B6, B7, B8, B9 s požrešnim algoritmom.

4.1 Preimenovanje vrednosti polj

Vsaki veljavni rešitvi mreže lahko poljubni vrednosti zamenjamo, na primer vse enice zamenjamo s trojkami. Posledično lahko s preimenovanjem vsako mrežo preoblikujemo v standarno (normirano) obliko (Poglavje 3.1). S tem postopkom smo zmanjšali število rešitev R za faktor $9! = 362880$:

[1] 5.98×10^{27} g

$$R = \frac{N}{9!} = \frac{5.52 \times 10^{27}}{362880} \tag{4.1}$$



Slika 4.1: Enote sudoku mreže.

4.2 Polnenje in redukcija B2 in B3

4.2.1 Vsi možni kandidati B2, B3

Nadaljnji korak je zapolniti sledeči škatli mreže, ob predpostavki, da je mreža standardne oblike. Zgornja vrstica škatel B2 in B3 ima 20 možnih rešitev ($\frac{6!}{6\times 6}$):

4,5,6,7,8,9	4,6,8,5,7,9	7,8,9,4,5,6	5,7,9,4,6,8
4,5,7,6,8,9	4,6,9,5,7,8	6,8,9,4,5,7	5,7,8,4,6,9
4,5,8,6,7,9	5,6,7,4,8,9	6,7,9,4,5,8	4,8,9,5,6,7
4,5,9,6,7,8	5,6,8,4,7,9	6,7,8,4,5,9	4,7,9,5,6,8
4,6,7,5,8,9	5,6,9,4,7,8	5,8,9,4,6,7	4,7,8,5,6,9

V primeru, da izberemo prvo rešitev, lahko dopolnimo ostala polja s kombinacijo trojic iz škatle B1 (Slika 4.2), kar pomeni $3!^6$ možnih rešitev (vsaka podmnožica trojic je lahko postavljena na 3! različne načine). Enako velja za primer, ko zgornjo vrstico dopolnimo z obratnim seznamom 7,8,9,4,5,6.

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6

Slika 4.2: Primer prvih treh škatel sudoku mreže.

1	2	3	4	5	7	6	8	9
4	5	6	8	9	a	7	b	c
7	8	9	6	b	c	4	5	a

Slika 4.3: Spremenljivke a, b in c zavzemajo vrednosti $1, 2$ in 3 v poljubnem zaporedju.

Za ostale primere velja, da bodo trojice škatel B2 in B3 sestavljene iz dveh (ali več) trojic škatle B1. V primeru, da izberemo drugo rešitev, lahko ponazorimo polnenje škatel B2 in B3 s Sliko 4.3. V tem primeru je možnih rešitev nekoliko več, in sicer $3 \times 3!^6$. Skupno je torej možnih $2 \times 3!^6 + 18 \times 3 \times 3!^6 = 2612736$ rešitev za prve tri škatle sudoku mreže.

4.2.2 Redukcija

Do sedaj imamo seznam možnih rešitev dolžine 2612736 za prve tri škatle mreže. Za vsako rešitev prvih treh škatel je potrebno poiskati vse možne poti do polne mreže. Zadnji korak metode je računanje ostalih polj s požrešnim algoritmom, zato je potrebno dolžino seznama zmanjšati. Poleg operacije preimenovanja vrednosti polj lahko permutiramo vrstice in stolpce mreže. Nadaljnja redukcija je možna s pomočjo operacije permutiranja, leksikografske ter stolpične redukcije. Upoštevati je treba tudi, da lahko v poljubnem koraku uporabimo operacijo preimenovanja, saj je vedno možno s permutiranjem pridobiti standardno obliko B1.

Leksikografsko redukcijo sestavljata dva koraka: permutiranje stolpcev znotraj B2 in B3, dokler so zgornje vrednosti v naraščajočem zaporedju, nato pa morebitna zamenjava škatel B2 in B3 (tako, da je prva vrednost škatle B2 manjša od istoležeče vrednosti v škatli B3). V prvem koraku je možnih 6 permutacij stolpcev znotraj posamezne škatle, kar pomeni 6^2 rešitev z enakim številom načinov dopolnevanja. Drugi korak podvoji faktor redukcije, kar skupno zmanjša število rešitev prvih treh škatel na 36288.

Poleg permutiranja stolpcev je možno tudi permutiranje vrstic in stolpcev v B1, čeprav lahko to pokvari standardno obliko prve škatle. Po operaciji permutiranja je vedno možno

1	2	3	4	5	8	6	7	9
4	5	6	1	7	9	2	3	8
7	8	9	2	3	6	1	4	5

Slika 4.4: Vertikalna zamenjava označenih vrednosti ne vpliva na število različnih dopolnitev sudoku mreže.

preimenovati vrednosti do standardne oblike B1. Sedaj je za vsako izmed 36288 rešitev možnih 6 permutacij škatel B1-B3 ter 6 permutacij stolpcev znotraj posamezne škatle, kar je skupno $6^4 = 1296$ možnosti. Meritve so pokazale, da uporaba leksikografske redukcije po permutiranju bistveno zmanjša število možnih parov škatel B2 in B3. Vendar to še ni vse, saj lahko permutiramo tudi vrstice B1-B3 (6 permutacij), kar še nadaljno reducira število možnih konfiguracij škatel B2 in B3 na 416.

Dolžina seznama je do te stopnje že bistveno bolj obvladljiva, zato bi lahko na tem mestu prešli na zadnji korak izračuna števila možnih mrež velikosti 9×9. Vseeno je vredno omeniti stolpično redukcijo, katere razširjena metoda zmanjša število možnih rešitev B2 in B3 na 44. Na Sliki 4.4 sta v prvih dveh vrsticah vrednosti 8 in 9 na zrcalnih pozicijah. V primeru zamenjave takih vrednosti in po potrebi preimenovanja do standardne oblike B1 ima pridobljena rešitev enako število možnih dopolnitev, zato je ni potrebno upoštevati. Poleg omenjenih vrednosti pridejo v poštev tudi pari (4,1) v prvi in drugi vrstici ter (5,8), (6,9), (1,2) v drugi in tretji vrstici. Kandidati stolpične redukcije so v bistvu pravokotniki z enakimi vrednostmi v zgornji in spodnji vrstici, vendar v drugačnem zaporedju. Pri zamenjavi parov so kandidati pravokotniki velikosti 2×2, razširjena metoda stolpične redukcije pa upošteva tudi kandidate dimenzije $2 \times k$ oziroma $k \times 2$, kar zmanjša seznam konfiguracij B2 in B3 na samo 71 različnih možnosti.

S pomočjo dodatne redukcije možnih rešitev B1-B3 je dokazano [11], da je vsaka možna konfiguracija prvih treh škatel (množica A) v ekvivalenčni relaciji z eno izmed 44 edinstvenih rešitev (množica B):

$$\forall a \in A, \quad \exists b \in B, \quad b \equiv a \tag{4.2}$$

Za vsak element množice A velja, da obstaja taka kombinacija operacij, ki ga oblikuje v enega izmed elementov množice B.

4.3 Dopolnjevanje mreže

V zadnjem koraku je potrebno poiskati vse možne načine dopolnitve sudoku mrež iz množice B, katere moč je 44. Brez kakršnekoli redukcije, je vseh veljavnih mrež $6670903752021072936960 = 6.671 \times 10^{21}$. S pomočjo operacije preimenovanja, leksikografske in stolpične redukcije ter permutiranja ostane samo še $3546146300288 = 2^7 \times 27704267971$ veljavnih sudoku mrež.

Poglavje 5

Tehnike generiranja sudoku mrež

S pridobljenim znanjem iz prejšnih poglavij je pot do programske rešitve algoritmov generiranja, ocenjevanja in reševanja sudoku mreže lažja. Za razvoj programske rešitve je potrebna sistematična obdelava pristopov generiranja, ocenjevanja in reševanja sudoku mrež. V tem poglavju se bom osredotočila na generiranje mrež, čeprav se prepleta z ocenjevanjem in reševanjem.

Cilj je implementirati generator, ki izpolnuje naslednje pogoje:

(a) Pridobljene mreže imajo edinstveno rešitev,

(b) Generator je sposoben pridobiti sudoku mreže različnih težavnostnih stopenj,

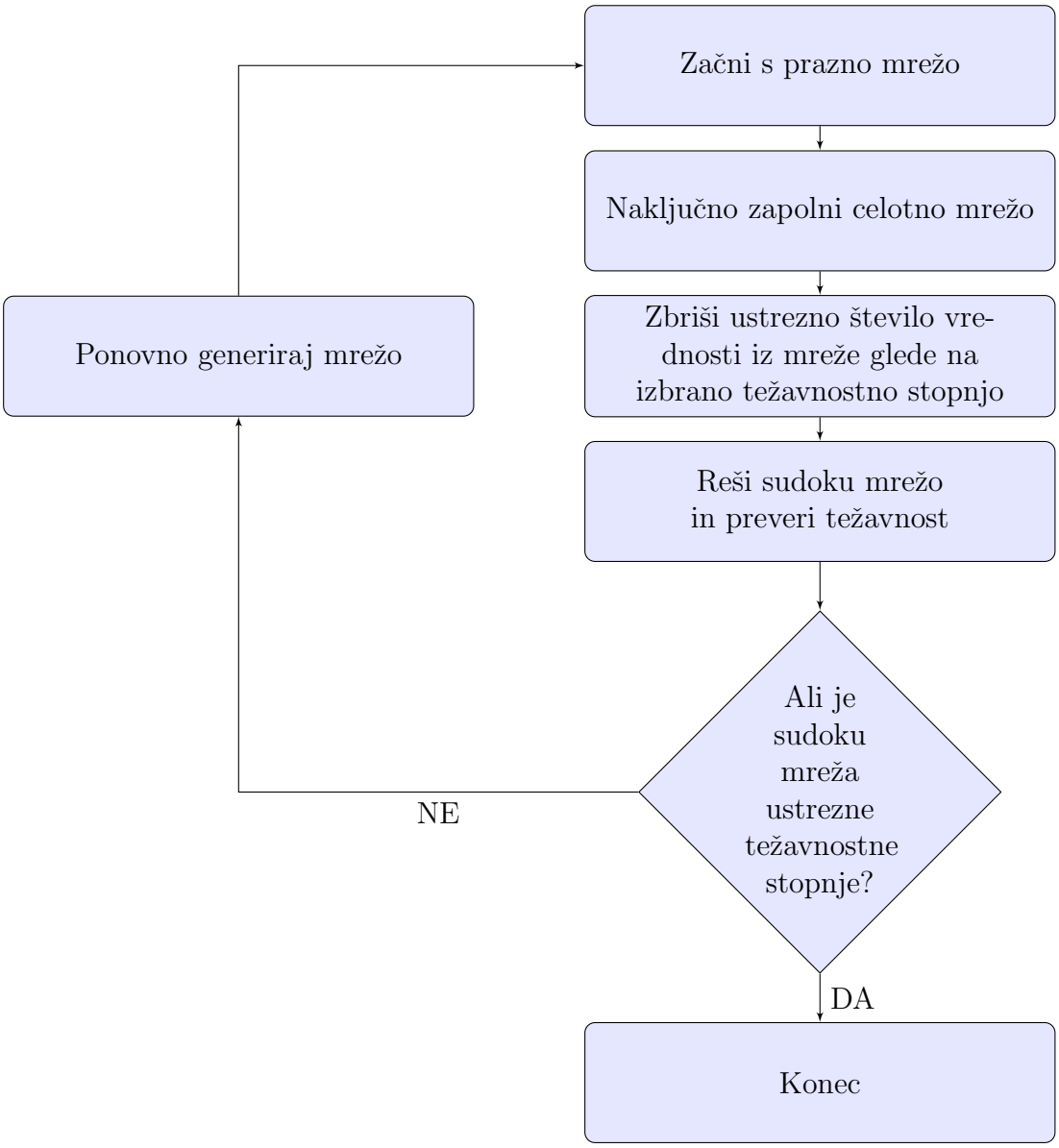
(c) Programska rešitev je hitra in učinkovita.

Za dosego tega cilja je najprej potrebno definirati programsko stukturo, nato pa je smiselno definirati vhodno in izhodno spremenljivko. Za namene diplomskega dela lahko predpostavimo, da je podatkovna struktura dvodimenzionalna tabela velikosti 9×9 polj. Vhodna spremenljivka je ponavadi prazna mreža, izhodna pa veljavna sudoku mreža, pripravljena za reševanje. Skupni imenovalec večine programskih rešitev generiranja je polnenje sudoku mreže. Z vidika programiranja sta tukaj dve možnosti: implementacija generatorja mrež z uporabo reševalnika ali postopno polnenje celotne mreže in naknadno brisanje vrednosti. Slednji algoritem predstavlja diagram poteka na Sliki 5.1.

5.1 Pridobivanje polne sudoku mreže

Za boljšo diferenciacijo sudoku mrež je algoritem za polnenje mrež tipično naključen.

Definicija 5. *Naključen algoritem je algoritem, ki poleg vhodne spremenljivke prejme naključni niz bitov, katere med izvajanjem uporabi za sprejemanje naključnih odločitev. Naključni algoritem lahko pri enakem vhodu v številnih ponovitvah vrne drugačne rezultate.*



Slika 5.1: Diagram poteka generiranja sudoku mrež [8].

Naključne algoritme klasificiramo v dve kategoriji. V prvi so t.i. *Las Vegas* algoritmi. Las Vegas algoritem bo vedno vrnil pravilno rešitev, ali pa ne bo vrnil nobene rešitve. V nasprotju z Las Vegas algoritmi so algoritmi druge kategorije, *Monte Carlo* algoritmi. Za njih je značilno, da vedno vrnejo rešitev, vendar je včasih ta rešitev napačna. Kljub temu so v nekaterih situacijah bolj primerni, saj je možno verjetnost pojavitve napačne rešitve zmanjšati z večkratnim izvajanjem algoritma z neodvisnimi naključnimi odločitvami [1].

Naključni algoritmi so zelo preprosti za razumevanje in implementacijo. Sledi primer naključnega polnenja mreže.

Algoritem 1 Las Vegas: Metodi izberiMesto() in izberiKandidata() sta naključni

```
1:  procedure napolniMrežo(mreža)
2:      if mrežaJePolna() then
3:          return mreža
4:      mesto ← izberiMesto()
5:      kandidat ← izberiKandidata()
6:      mreža[mesto] ← kandidat
7:      if preveriPravilnost(mreža) then
8:          napolniMrežo(mreža)
9:      else
10:         mreža[kandidat] = 0
        return mreža
```

Algoritem 1 uporablja sestopanje (angl. backtracking). Tak algoritem poišče vse (ali nekatere) rešitve računskega problema s pomočjo postopnega grajenja kandidatov delnih rešitev v končne. V trenutku, ko algoritem ugotovi, da je kandidat neprimeren, se vrne na prejšni korak. Drugače rečeno, v primeru "slabe" veje rekurzije se algoritem vrne na prejšnje vozlišče in od tod nadaljuje z računanjem.

5.2 Brisanje vrednosti

V drugem koraku generiranja sudoku mrež ima program shranjeno rešitev problema, ki mora ostati tudi edina končna rešitev mreže. Najbolj naiven in preprost način brisanja vrednosti sestavlja naključni izbor polja, brisanja vrednosti v polju ter preverjanje števila možnih rešitev mreže. Primer takega generiranja prikazuje Algoritem 2.

Pri brisanju vrednosti je smiselno vpeljati določene izboljšave, ki pohitrijo program. Ker je ob vsakem brisanju vrednosti potrebno preveriti število rešitev, je dobro zmanjšati število rekurzivnih klicev metode brisanja vrednosti. Število rekurzivnih klicev lahko zmanjšamo z rezanjem odločitvenih dreves (angl. pruning optimization). Tovrstna optimizacija sugerira, da če v odločitvenem drevesu trenutno vozlišče na zadošča robnim pogojem, tudi podrejena vozlišča ne zadoščajo robnim pogojem ter so lahko eliminirana iz množice rešitev. Vsaka iteracija brisanja vrednosti predstavlja novo vozlišče v drevesu.

Algoritem 2 Brisanje vrednosti

```
1: procedure BRIŠIVREDNOSTI(MREŽA)
2:     if dovolj vrednosti zbrisanih then
3:         return mreža
4:     mesto ← izberiMesto()
5:     shranjenaVrednost ← mreža[mesto]
6:     mreža[mesto] ← 0
7:     if številoRešitev > 1 then
8:         mreža[mesto] ← shranjenaVrednost
9:     else
10:        brišiVrednosti(mreža)
       return mreža
```

$$\begin{bmatrix} 0 & 2 & . \\ 3 & 4 & . \\ . & . & . \end{bmatrix} \mapsto \begin{bmatrix} 0 & 0 & . \\ 3 & 4 & . \\ . & . & . \end{bmatrix} \mapsto \begin{bmatrix} 0 & 0 & . \\ 0 & 4 & . \\ . & . & . \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & . \\ 0 & 4 & . \\ . & . & . \end{bmatrix} \mapsto \begin{bmatrix} 0 & 2 & . \\ 0 & 0 & . \\ . & . & . \end{bmatrix}$$

Slika 5.2: Primer rezanja odločitvenih dreves.

Opazujmo hipotetičen primer zaporednih vozlišč v drevesu rešitev na Sliki 5.2. V prvem koraku je bila zbrisana vrednost v zgornjem levem kotu. Po brisanju prve vrednosti se je izkazalo, da ima mreža še vedno eno rešitev. Nato je bila v drugem koraku zbrisana dvojica, vendar se je izkazalo, da ima taka mreža več možnih rešitev. Ob upoštevanju optimizacije rezanja vozlišč, so vse rešitve, ki vsebujejo enak vzorec zbrisanih vrednosti, eliminirane. Zbrisana vrednost se vrne na svoje mesto in program izbere alternativno mesto za brisanje [13].

Poleg rezanja odločitvenih dreves je možna izboljšava metode za izbor naslednjega kandidata za brisanje v mreži. Možno je namreč brisanje vsake druge vrednosti, brisanje s sprehajanjem po mreži (v obliki črke "S") ali brisanje vrednosti z leve proti desni in nato zvrha navzdol. Izkaže se, da je uporaba posamezne metode najbolj učinkovita pri določeni težavnostni stopnji (Tabela 5.1) [13].

5.3 Generiranje z reševanjem

Poleg opisanega pristopa h generiranju sudoku mrež, je možno vhodno prazno mrežo programsko polniti, dokler ne obstaja edinstvena rešitev mreže (Algoritem 3). Algoritem

Tabela 5.1: Učinkoviti načini brisanja vrednosti iz matrike pri posameznih težavnostnih stopnjah.

Težavnostna stopnja	Optimalna metoda brisanja vrednosti
Zelo lahka	Naključna metoda izbire
Lahka	Naključna metoda izbire
Srednja	Izbira vsake druge vrednosti
Težka	Izbira s sprehajanjem po mreži (v obliki črke "S")
Zlobna	Brisanje z leve proti desni ter zvrha navzdol

je v principu isti in zanj veljajo enake izboljšave, saj gre zgolj za obratno logiko. Reševanje sudoku mreže je ozko grlo v programski rešitvi, zato je pomembno, da je ta del programa čim bolj učinkovit. Več o reševanju sudoku mrež v poglavju 7.

Algoritem 3 Reševanje prazne mreže

```
1:  procedure REŠI(MREŽA)
2:      if dovolj vrednosti vstavljenih then
3:          return mreža
4:      mesto ← izberiMesto()
5:      kandidat ← izberiKandidata()
6:      mreža[mesto] ← kandidat
7:      if preveriPravilnost(mreža) then
8:          reši(mreža)
9:      else
10:         mreža[mesto] = 0
        return mreža
```

5.4 Generiranje s permutiranjem

Včasih ni potrebno, da je generator sudoku mrež popolnoma naključen. Ob tej predpostavki je lahko generator bistveno hitrejši, če upoštevamo dejstvo: Veljavna sudoku mreža se s pomočjo operacij permutiranja in preslikav spremeni v drugo, prav tako veljavno, sudoku mrežo. Operacije, ki omogočajo take transformacije mrež so sledeče:

(a) permutiranje vrstic,

(b) permutiranje stolpcev,

(c) preslikava preko obeh diagonal,

(d) preslikava čez (vertikalno in horizontalno) polovico.

Pravzaprav gre za podmnožico operacij, ki definirajo ekvivalenčni razred latinskega kvadrata (Poglavje 3.1). Slaba stran takega generatorja je, da je število pridobljenih mrež na ta način omejeno z velikostjo ekvivalenčnega razreda.

Algoritem 4 Generiranje s permutiranjem

```
1: procedure PERMUTIRAJ()
2:     mreža ← prvaVeljavnaMreža()
3:     i ← številoPermutacij
4:     while i ≠ 0 do
5:         mreža ← naključnoPermutiraj(mreža)
6:         i ← (i − 1)
       return mreža
```

Poglavje 6

Ocenjevanje težavnostne stopnje

Dober algoritem ocenjevanja težavnostne stopnje mora upoštevati v prvi vrsti število podanih vrednosti, nato porazdelitev podanih vrednosti po mreži ter tehnike človeškega reševanja sudoku mreže. Nekatere tehnike reševanja so kompleksne, njihova uporaba pri ocenjevanju pa ne prinese vedno veliko informacij o težavnosti, zato jih včasih ni potrebno upoštevati.

6.1 Tehnike reševanja sudoku mrež

Pristop k reševanju je seveda odvisen od osebe, vendar je nekaj tehnik splošno znanih. Zelo lahke sudoku mreže so rešljive kar s tehniko prostega očesa (angl. eyeballing technique), medtem ko težje zahtevajo pripisovanje števk in nekoliko več razmisleka. Tehnike so uvrščene glede na težavnost, kot prikazuje Tabela 6.1.

6.1.1 Lažje in srednje težke tehnike reševanja sudoku mrež

Ob opazovanju vrstice, stolpca ali škatle sudoku mreže je možno, da je zaradi porazdelitve števk v okolici samo eno mesto, ki ustreza določeni števki. Pri lažjih mrežah se to pogosto zgodi in je na ta način pravzaprav možno pridobiti celotno rešitev. Sicer je potrebno uporabiti bolj napredne tehnike reševanja, kot so na primer opazovanje vrst kandidatov,

Tabela 6.1: Najbolj pogoste tehnike reševanja sudoku mrež.

Zahtevnost tehnike	Tehnika reševanja
Lažje tehnike	*Edino mesto, Edini kandidat*
Sredje težke tehnike	*CRME*
Napredne tehnike	*Goli pari/trojice, Skriti pari/trojice*
Mojsterske tehnike	*X-krilo, Riba mečarica, Vsiljene verige*
Najtežje tehnike	*Nishio, Ugibanje*

dvojnih parov ter večvrstično tehniko. Včasih je tehniko reševanja težje razložiti kot uporabiti, vendar lažjih tehnik reševanja tu ne bom opisovala, saj so podrobneje obravnavane v poglavju 7.

Tehnika golih in skritih parov ter trojic je uporabna, kadar se na različnih mestih v vrstici, stolpcu ali škatli nahaja enak par oziroma trojica kandidatov. Zato bo na enem izmed teh mest vrednost zagotovo izpolnjena, in je ni potrebno upoštevati med kandidati na ostalih mestih vrstice. Primer 7.7 iz poglavlja o reševanju sudoku mrež prikazuje primer take porazdelitve v škatli.

6.1.2 Težje tehnike reševanja

Poleg opisanih tehnik reševanja obstaja mnogo drugih, kot so tehnika vsiljenih verig ter Nishio, ki temeljijo na ugibanju ter opazovanju sprememb v mreži. Za namene diplomskega dela ni potrebno poznavanje vseh najtežjih tehnik, zato so omenjene zgolj kot zanimivost.

Ena izmed bolj zahtevnih tehnik reševanja je t.i. tehnika "X-krilo", ki je uporabna, kadar obstajata dve vrstici v mreži z istim kandidatom v skupnih stolpcih. Z drugimi besedami, polja, kjer se nahaja kandidat, tvorijo oglišča pravokotnika. Na Sliki 6.1 so kandidati relevantnih polj obarvani z modro. Za razumevanje X-krila sedaj recimo, da je v polju (3,4) kandidat 6. Taka izbira vpliva na obarvana polja tako, da ostane v diagonalnem oglišču (tj. v polju (9,9)) le šestica (Slika 6.2). Podobno se zgodi ob izbiri šestice v polju (9,4), zato povezana diagonalna oglišča tvorijo obliko črke "X". Znotraj oglišč pravokotnika še vedno ne moremo potrditi nobenega kandidata, vendar lahko iz množic kandidatov istih vrstic in stolpcev eliminiramo vse šestice. Na Sliki 6.3 lahko s pomočjo te tehnike eliminiramo le dva kandidata, vendar s tem potrdimo osmico v polju (9,1).

Pri tehniki ribe mečarice gre za podoben pristop kot pri X-krilu. Kadar izbira kandidata v določenem polju mreže verižno vpliva na druga polja, je včasih možno kandidate v posameznih stolpcih izločiti. Za lažjo predstavo prikazuje Slika 6.4 tako porazdelitev, kjer so relevantna polja obarvana z modro. Če opazujemo, kako so porazdeljene le štirice v vrsticah ribe mečarice, vidimo, da sta možna le dva scenarija. Na Sliki 6.5 sta obarvana z modro in rdečo barvo. V obeh primerih pa so vrstice in stolpci ribe mečarice zasedeni s štirico. To pomeni, da lahko v množici kandidatov istih vrstic in stolpev eliminiramo vse štirice (Slika 6.6). Tako porazdelitev je v mreži težje opaziti, saj je potrebno opazovati več vrstic in stolpcev hkrati.

Tehnika vsiljenih verig zelo dobro deluje, ko v mreži ostane veliko kandidatov v parih. Včasih se zgodi, da oba kandidata določenega polja verižno povzročita potrditev drugega polja. Pri tehniki vsiljenih verig je torej potrebno ugibanje kandidatov ter opazovanje drugih polj. Ko tudi vsiljene verige odpovejo, igralcu ne preostane drugega kot reševanje

9	6 8	2 4	2 4	5	1	7	3	6 8
1	4 6	7	3	9	8	2	4 6	5
5	3 4 8	2 3 4	2 4	7	6	4 8	9	1
8	1	6 9	7	2	4	3	5	6 9
2	3 4	3 4 9	1	6	5	4 8 9	4 8	7
4 6	7	5	9	8	3	4 6	1	2
4 6	2	1	5	3	7	4 6 8 9	4 6 8	4 6 8 9
7	5	8	6	4	9	1	2	3
3	9	4 6	8	1	2	5	7	4 6

Slika 6.1: Primer porazdelitve mreže, kjer je mogoče uporabiti tehniko X-kril.

9	6 8	2 4	2 4	5	1	7	3	6 8
1	4 6	7	3	9	8	2	4 6	5
5	3 4 8	2 3 4	2 4	7	6	4 8	9	1
8	1	6	7	2	4	3	5	~~6~~ 9
2	3 4	3 4 9	1	6	5	4 8 9	4 8	7
4 6	7	5	9	8	3	4 6	1	2
4 6	2	1	5	3	7	4 6 8 9	4 6 8	4 6 8 9
7	5	8	6	4	9	1	2	3
3	9	~~6~~ 4	8	1	2	5	7	~~4~~ 6

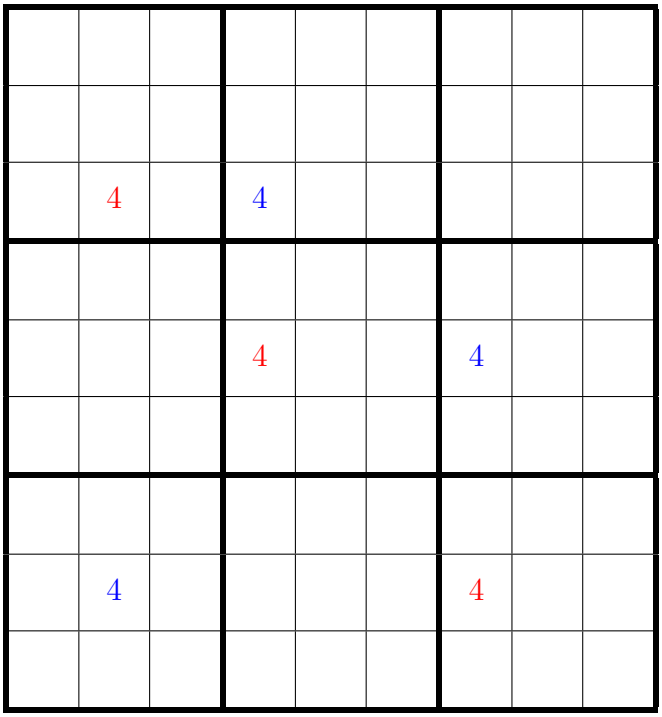
Slika 6.2: Izbira šestice v polju (3,4) in njen vpliv na diagonalno oglišče (9,9).

9	6 8	2 4	2 4	5	1	7	3	~~6~~ 8
1	4 6	7	3	9	8	2	4 6	5
5	3 4 8	2 3 4	2 4	7	6	4 8	9	1
8	1	6 9	7	2	4	3	5	6 9
2	3 4	3 4 9	1	6	5	4 8 9	4 8	7
4 6	7	5	9	8	3	4 6	1	2
4 6	2	1	5	3	7	4 6 8 9	4 6 8	4 6 8 9
7	5	8	6	4	9	1	2	3
3	9	4 6	8	1	2	5	7	4 6

Slika 6.3: Uporaba tehnike reševanja X-krila.

1	9	5	3	6	7	2	4	8
2 4	7	8	1 2 4	5	1 4	3	6	9
3	2 4	6	2 4	9	8	1	5	7
2 6	1 2 6	3	7	8	1 4	5	9	2 4
7	1 2	9	1 4	2 3	5	4 8	3 8	6
5	8	4	9	2 3	6	7	1	2 3
8	3	2	5	4	9	6	7	1
9	4 6	7	6 8	1	3	4 8	2	5
4 6	5	1	6 8	7	2	9	3 8	3 4

Slika 6.4: Primer porazdelitve mreže, kjer je mogoče uporabiti tehniko ribe mečarice.



Slika 6.5: Možni scenariji porazdelitev štiric v vrsticah ribe mečarice.

mreže z ugibanjem [3].

6.2 Ocenjevalnik

Algoritem ocenjevanja težavnostne stopnje sudoku mreže mora na učinkovit način oceniti posamezno sudoku mrežo. Igralci imajo lahko različne zmogljivosti pri prepoznavanju vzorcev in logičnem sklepanju, zato je potrebno pri ocenjevanju upoštevati tudi reševalne tehnike. V splošnem so ocenjevalni razredi razdeljeni glede na število praznih polj v mreži, kot prikazuje Tabela 6.2. Več kot je praznih polj v mreži, težja bo pot do rešitve. Kljub temu obstajajo izjeme, pri katerih to ne velja.

Tabela 6.2: Porazdelitev težavnostnih stopenj glede na število praznih polj v mreži.

Razred	Število praznih polj
Lahko	40 do 45
Srednje	46 do 49
Težko	50 do 53
Zelo težko	54 do 58

Drugi pomemben faktor pri ocenjevanju težavnosti mreže je porazdelitev podanih vrednosti. Običajno so sudoku mreže z enakomerno porazdeljenimi vrednostmi lažje, medtem ko so mreže s podanimi vrednostmi v gručah težje. Ocenjevalnik mora upoštevati

1	9	5	3	6	7	2	4	8
2 4	7	8	1 2 ~~4~~	5	1 4	3	6	9
3	2 4	6	2 4	9	8	1	5	7
2 6	1 2 6	3	7	8	1 4	5	9	2 4
7	1 2	9	1 4	2 3	5	4 8	3 8	6
5	8	4	9	2 3	6	7	1	2 3
8	3	2	5	4	9	6	7	1
9	4 6	7	6 8	1	3	4 8	2	5
4 6	5	1	6 8	7	2	9	3 8	3 4

Slika 6.6: Uporaba tehnike reševanja ribe mečarice.

tudi čas, ki je potreben za reševanje mreže, zato je smiselno med ocenjevanjem meriti čas programske rešitve mreže. Čas programskega reševanja je lahko dober pokazatelj težavnosti, sploh če reševalnik uporablja prej opisane tehnike reševanja (Poglavje 6.1). V vsakem primeru je dobro, da ocenjevalnik torej upošteva, katere tehnike so potrebne pri reševanju mreže in kolikokrat je potrebna njihova uporaba, dokler ne pride do rešitve. Program za ocenjevanje težavnostne stopnje lahko na primer uporablja globalni števec, ki uteži posamezno mrežo. Vsakič, ko je neko polje rešljivo z določeno tehniko, ocenjevalnik poveča števec točk. Tehnike se razlikujejo v težavnosti, zato si tudi težje zaslužijo večje število točk. Mreža z večjim števcem je označena kot težja, saj je v procesu reševanja potrebna uporaba kompleksnejših tehnik reševanja.

Tabela 6.3: Težo tehnik reševanja sudoku mrež in uporabnost takšne implementacije podbrobneje opisuje vir [8].

Tehnika reševanja	Število točk
Tehnika s prostim očesom [1]	1
Edini kandidat v škatli	2
Edini kandidat v vrstici	2
Edini kandidat v stolpcu	2
Goli par v stolpcu	3
Goli par v vrstici	3
Goli par v škatli	3
Gola trojica v stolpcu	4
Gola trojica v vrstici	4
Gola trojica v škatli	4
Eliminacija z grobo silo	5
Backtracking z grobo silo[2]	5

6.2.1 Ocenjevanje s pomočjo entropije

Koncept entropije se je začel v fizikalnih znanostih uveljavljati v devetnajstem stoletju, predvsem na področju termodinamike in razvoja statistične fizike. Okoli leta 1850 je Rudolf Clausius predstavil koncept termodinamičnega sistema in trdil, da prihaja v poljubnem nepovratnem procesu do naraščajoče izgube energije izven meja sistema. Clausius je z nadaljevanjem svoje raziskave postopoma uvedel pojem entropije in njene lastnosti.

Po preteku stoletja se je poleg termodinamične pojavila informacijska entropija. Leta 1948 je Claude Shannon, v času svoje zaposlitve pri Bell Telephone Laboratories, matematično preučeval statistično naravo izgube informacije v komunikaciji preko telefonskih

[1] V angleški literaturi je ta tehnika imenovana "Eyeballing technique" ali kar CRME (Column, Row, Mini-grid Elimination).

[2] V primeru, da je pri ugibanju z grobo silo algoritem naredil napako, je potrebno prestopanje.

linij. Istega leta je objavil članek *A Mathematical Theory of Communication* [12], kjer definira informacijsko entropijo sledeče:

$$H = -K \sum_{i=1}^{k} p(i) * \log_2 p(i) \tag{6.1}$$

V enačbi informacijske entropije je K pozitivna konstanta, $p(i)$ verjetnost i-tega izida in k število možnih izidov poskusa. Informacijska ali Shannonova entropija je v informatiki količina, ki meri negotovost izida poskusa, povezanega s slučajno spremenljivko. Informacijska entropija je merilo za količino informacije, ki jo pridobimo s poznavanjem vrednosti slučajne spremenljivke. Informacijsko entropijo tipično označuje črka H, njeno količino pa običajno izraža enota bit, zato se pri izračunu uporablja logaritem z bazo 2. V primeru, da je entropija izražena v tritih ali nitih, je tudi baza logaritma drugačna (3 ali e).

V primeru meta poštenega kovanca je entropija meta enaka 1, saj je verjetnost posameznega izida 1/2. To pomeni, da lahko rezultat izida predstavimo samo z enim bitom:

$$H = -1/2 * \log_2 1/2 - 1/2 * \log_2 1/2 = 1/2 + 1/2 = 1 \tag{6.2}$$

Lastnosti entropije bolje ponazori primer meta dveh poštenih kock. V tem primeru so verjetnosti možnih izidov poskusa različne, kar pomeni, da ima met kock večjo informacijsko entropijo. Z drugimi besedami, za predstavitev izida poskusa potrebujemo minimalno 4 bite:

$$\begin{aligned} H = -1/36 * \log_2 1/36 - 2/36 * \log_2 2/36 - 3/36 * \log_2 3/36 - \\ 4/36 * \log_2 4/36 - 5/36 * \log_2 5/36 - 6/36 * \log_2 6/36 - \\ 5/36 * \log_2 5/36 - 4/36 * \log_2 4/36 - 3/36 * \log_2 3/36 - \\ 2/36 * \log_2 2/36 - 1/36 * \log_2 1/36 = 3.2744 \end{aligned} \tag{6.3}$$

Sudoku mreža je pravzaprav polje naključnih spremenljivk, katerih vrednosti so medsebojno odvisne. Če je sudoku mreža v celoti prazna, je entropija take mreže maksimalna. Več kot je podanih začetnih vrednosti, manjša je mera negotovosti mreže. Entropija rešene mreže je minimalna, saj so vse vrednosti v mreži določene. Ker je pri izračunu entropije potrebno upoštevati entropijo posameznega polja, lahko postopek opišemo kot sledi.

Definicija 6. *Informacijska entropija sudoku polja je mera negotovosti polja, katere vrednost ponazarja ulomek x/9, pri čemer x ponazarja količino pozitivno podporne informacije, oziroma število kandidatov polja trenutnega stanja mreže.*

Definicija 7. *Informacijska entropija sudoku mreže je mera negotovosti mreže, katere*

	3	6	7					4
	7		6		4		2	
		1			8			
3						6		
	1	7				3	9	
		2						8
7			1			2		
	6		5		2		3	
4					3	5	6	

Slika 6.7: Primer lahke mreže z entropijo 0.243 bitov, s pomočjo katere program napačno oceni težavnost mreže (težka mreže).

vrednost ponazarja ulomek SEG/81, pri čemer SEG predstavlja vsoto entropij vseh posameznih polj v mreži.[3]

Entropija sudoku mreže lahko dobro oceni težavnostno stopnjo, saj vsebuje informacijo o številu praznih polj in njihovi porazdelitvi po mreži. Izkaže se, da je polja z manjšo entropijo v postopku reševanja bolje obravnavati pred ostalimi. V skladu s predstavljenimi rezultati vira [5] je razvidno, da je ocenjevanje s programom na osnovi merjenja entropije mreže mogoče, vendar pride v nekaterih primerih do zmote (Slike 6.7 - 6.10). Program ocenjuje na osnovi pravil, ki sledijo:

Tabela 6.4: Pravila ocenjevanja težavnostne stopnje s pomočjo entropije.

Entropija mreže	Težavnostna stopnja mreže
$SEG > 0.26$	Zelo težka - zlobna
$0.26 \geq SEG > 0.23$	Težka
$0.23 \geq SEG > 0.19$	Srednja
$SEG \leq 0.19$	Lahka

[3]Potrebno je upoštevati, da definiciji o entropiji sudoku mreže veljata za dimezijo mreže 9, vendar sta prenosljivi tudi na druge velikosti mreže.

		5				7		8
		7					5	
1			7	9	5		3	2
			2			8		6
				6				
4		3			8			
5	7		1	3	2			9
	9					2		
2		6				5		

Slika 6.8: Primer srednje težke mreže z entropijo 0.246 bitov, s pomočjo katere program napačno oceni težavnostno stopnjo mreže (težka mreža).

				4			7	5
	1			6	7			3
8			3	5		1		
			4	1				
	6	2				8		
				7				
4				9		3		6
		1	7		4			
9								

Slika 6.9: Primer težke mreže z entropijo 0.303 bitov, s pomočjo katere program napačno oceni težavnostno stopnjo mreže (zelo težka mreža).

3						6		
			9			1	7	3
		7						5
					6			8
1			5					
	5		3				1	4
8	1							
6				9		8		
9		4	8	5		2		

Slika 6.10: Primer zelo težke mreže z entropijo 0.288 bitov, s pomočjo katere program pravilno oceni težavnost mreže.

Poglavje 7

Reševanje sudoku mreže

Algoritem programskega reševanja sudoku mreže spada v množico NP-polnih problemov, vendar obstajajo za mreže velikosti 81 polj zelo učinkoviti programi, ki v kratkem času pridobijo rešitev ne glede na težavnostno stopnjo mreže. S stališča računske kompleksnosti problema je odločitveni problem NP-poln, kadar je hkrati NP in NP-težek. Sama kratica "NP" v tem kontekstu pomeni "nondeterministic polynomial time", oziroma nedeterminističen polinomski čas. Čeprav je mogoče katerokoli rešitev NP-polnega problema hitro (v polinomskem času) preveriti, ne obstaja učikovit način pridobivanja njegovih rešitev. Torej, čas potreben za pridobivanje rešitev NP-polnega problema, z uporabo kateregakoli znanega algoritma, s povečevanjem problema hitro narašča.

Za reševanje NP-polnih problemov se v mnogih primerih uporabi tehnike za pohitritev pridobivanja rešitev. Izbira tehnike je odvisna od narave problema in stopnje učinkovite aplikativnosti tehnike na problem. Najbolj pogoste so opisane v Tabeli 7.1.

Tabela 7.1: Tehnike reševanja NP-polnih problemov.

Tehnika	Opis
Aproksimacija	Namesto iskanja optimalne rešitve program izvaja iskanje "skoraj" optimalne rešitve.
Uporaba naključnih metod	Z uporabo naključnih metod program izboljša povprečen čas reševanja s tem, da dopusti malo verjeten neuspeh pri iskanju rešitve.
Restrikcija	Z uporabo restrikcije strukture vhodnih podatkov je mogoča uporaba bolj učinkovitih algoritmov.[1]
Parametrizacija	Program je pogosto bolj učinkovit pri iskanju rešitev, kadar so določeni parametri vhodnih podatkov fiksirani.
Hevrističen pristop	Rešitev pridobljena s hevrističnimi metodami ni nujno optimalna, vendar je časovna kompleksnost programa obvladljiva.[2]

[1] Primer restrikcije vhodnih podatkov je omejitev na ravninske grafe pri procesiranju grafov.
[2] Aproksimacija je le ena izmed hevrističnih metod.

Brez uporabe hevrističnih metod ali drugih pohitritev je požrešen reševalnik sudoku mrež časovno zahteven (Algoritem 5). V nadaljevanju sledi obravnava tehnik reševanja, katerih implementacija pripomore k bistveno boljši časovni kompleksnosti programa.

Algoritem 5 Algoritem reševanja z grobo silo

```
1: procedure REŠI(MREŽA)
2:     if mreža je polna then
3:         return mreža
4:     mesto ← naslednjePraznoMesto()
5:     while i ≠ dimenzija do
6:         mreža[mesto] ← i
7:         if preveriPravilnost(mreža) then return reši(mreža)
8:         else
9:             mreža[mesto] = 0
10:        i ← i + 1
       return null
```

7.1 Tehnika CRME

V poglavju ocenjevanja težavnostnih stopenj je bilo omenjenih nekaj tehnik reševanja v skladu s težavnostno stopnjo mrež, kot na primer tehnika s prostim očesom, goli pari in trojice ter med drugim tudi najbolj zahtevne tehnike reševanja, kot je Nishio. Imena nekaterih tehnik se v literaturi razlikujejo, čeprav gre za isti postopek reševanja ("Edino mesto in edini kandidat" je tudi tehnika s "prostim očesom" oziroma CRME). Kratica CRME izvira iz angleškega jezika in pomeni "Column, Row, Minigrid Elimination", oziroma eliminacija stolpcev, vrstic in škatel mreže. Sudoku mreže najlažje težavnostne stopnje so ponavadi rešljive kar samo z uporabo te tehnike.

Slika 7.1 prikazuje primer porazdelitve podanih vrednosti v mreži. Ta mreža sicer ni veljavna, vendar je za namene razlage tehnike reševanja bolj primerna. Program se seveda izvaja pravilno, če je mreža veljavna. Ko program najde prazno polje, pregleda, katere vrednosti se nahajajo v isti vrstici, stolpcu in škatli. Nato eliminira kandidate, ki se že nahajajo v okolju (tj. isti vrstici, stolpcu ali škatli) praznega polja. V tem primeru je prvo prazno polje (1,2), za katero hitro ugotovi, da je zaradi skoraj polne vrstice edini kandidat število 9. Po končani CRME eliminaciji program hrani kandidate za vsako prazno polje in tista z edinim kandidatom so lahko izpolnjena (Slika 7.2). Pomembno je tudi, da program celotno pregledovanje mreže ob vsaki vstavljeni vrednosti ponovi, saj nova vrednost lahko vpliva na ostala polja v vrstici, stolpcu in škatli. S pomočjo CRME eliminacije lahko program pri lažjih sudoku mrežah pride do končne rešitve. Sicer je potrebna uporaba bolj zahtevnih tehnik reševanja. Čeprav velikokrat uporaba samo

4		3	7	8	5	1	2	6
2								
1								
	5	8						
6	1	4						
7		9						
9								
5								

Slika 7.1: Delno napolnjena sudoku mreža.

4	9	3	7	8	5	1	2	6
2								
1								
3	5	8						
6	1	4						
7	2	9						
8								
9								
5								

Slika 7.2: Mreža po eliminaciji CRME.

te metode ne reši celotne mreže, kljub temu zapolni nekaj polj, kar olajša nadaljnje reševanje.

7.2 Iskanje osamelcev

Tehnika iskanja osamelcev (angl. lone ranger technique) išče tistega kandidata, ki se med vsemi ostalimi pojavi le enkrat v vrstici, stolpcu ali škatli. Iz Slike 7.3 je razvidno, da se kandidat 8 pojavi le enkrat in sicer v drugem polju. Iz Slik 7.4 in 7.5 pa lahko razberemo osamelce, ki se pojavijo v škatli in stolpcu. Ko program najde takega kandidata, lahko to polje takoj izpolni. Če program izmenično izvaja CRME eliminacijo in iskanje osamelcev, lahko že v nekaj korakih pride do rešitve. V naprotnem primeru se posluži iskanju golih parov in trojic.

1	3 4 8	5	9	6	7	2	3 4	3 4

Slika 7.3: Vrstica, kjer se kandidat 8 pojavi le enkrat.

6	2 4 5	7	2 3 8 9	2 3 8 9	1	3 4 5	3 4 5	3 4 5
2 4 5	8	2 4 5	7	2 3	2 3	1 3 4 5	6	9
9	3	1	6	4	5	2	8	7

Slika 7.4: V tretji škatli se pojavi kandidat 1 enkrat.

7.3 Goli pari in trojice

Tehnika iskanja golih parov in trojic ter kombinacija prej omenjenih tehnik ponavadi zadostuje tudi za reševanje zelo zahtevnih sudoku mrež. Program zopet opazuje vrstice, stolpce in škatle, vendar tokrat išče osamljene pare ali trojice enakih kandidatov znotraj opazovanega območja. Za lažjo predstavo ponazarja Slika 7.6 primer zgornjega dela mreže. V poljih (5,2) ter (6,2) se nahaja par kandidatov 2,3. V primeru, da bi v polje (5,2) vpisali dvojko, bi to povzročilo vpis trojke v sosednje polje in obratno. Zato lahko program predpostavi, da se na ostalih mestih vrstice, kandidata iz para ne moreta pojaviti ter ju lahko eliminira. Podobno velja za škatlo, v kateri se nahajata, zato program eliminira tudi kandidate znotraj druge škatle.

Pri iskanju trojic program deluje podobno, le da v tem primeru išče tri pare ali trojice kandidatov, ki se skupaj pojavijo trikrat znotraj opazovanega območja (Slika 7.7). Trojice v sudoku mreži so redko tako očitne. Bolj verjetno se pojavijo v obliki, prikazani na Sliki 7.8, kjer izpolnitev enega polja rezultira v pridobitvi golega para.

4 9	4 5 9	4 5 9	4 5 6 9	4 5 6 8	3	2	7	1

Slika 7.5: Vrstica, kjer se kandidat 8 pojavi le enkrat.

6	2 4 5	7	2 3 8 9	2 3 8 9	1	3 4 5	3 4 5	3 4 5
2 4 5	8	2 4 5	7	2 3	2 3	1	6	9
9	3	1	6	4	5	2	8	7

Slika 7.6: Primer zgornjega dela mreže, kjer par kandidatov povzroči eliminacijo dvojic v drugi vrstici in škatli.

1	2 4 6 7	9
2 4 6	3	8
2 4 6	2 4 6	5

Slika 7.7: Primer škatle mreže, kjer trojica kandidatov povzroči eliminacijo števil 2,4 in 6 v polju (2,1).

1 2 3	1 2 3	1 2
x	x	x
x	x	x

Slika 7.8: Primer pojavitve trojice (1 2) znotraj škatle.

7.4 Eliminacija z grobo silo

Z uporabo omenjenih tehnik je programsko mogoče rešiti večino težjih mrež, vendar je pri zlobnih sudoku mrežah vseeno potrebno uporabiti tehniko ugibanja, oziroma eliminacijo z grobo silo. Ponavadi je smiselno izbrati polje, ki vsebuje čim manj kandidatov, saj bo verjetnost uspešnega ugibanja tako maksimalna. Nato program naključno izbere enega ter skuša rešiti mrežo do konca. V primeru zmote se mora program vrniti nazaj ter izbrati drugega kandidata. Ponavadi je tudi najbolj zlobna mreža do tega koraka že bistveno bolj polna, zato je uporaba te tehnike obvezna na koncu, ko ostale že omagajo. Po eliminaciji z grobo silo program nadaljuje reševanje s prej opisanimi tehnikami, saj so hitrejše.

7.5 Alternativni pristopi k reševanju sudoku mrež

Sudoku mrežo je programsko mogoče reševati na različne načine. Poleg opisanega reševanja s pomočjo uporabe človeških tehnik reševanja, je med drugim možna implementacija Knuthovega algoritma X ali uporaba genetskih algoritmov. Uporaba alternativnih pristopov k reševanju vodi v nova spoznanja o zastavljenem problemu, zato jih je vredno omeniti.

7.5.1 Reševanje z uporabo genetskih algoritmov

Genetski algoritmi (GA) so optimizacijske metode, ki uporabljajo kot model računanja principe Darwinove evolucije. Rešitev problema je predstavljena v obliki individualnih enot, sestavljenih iz kromosomov, ki vsebujejo več genov. Konceptualno se individualne enote pri GA od naravnih razlikujejo po tem, da izhajajo deterministično iz kromosomov. Z drugimi besedami, vpliv fenotipa nima teže pri oblikovanju rešitev. Uspešnost posameznih individualnih enot ocenjuje evaluacijska ("fitness") funkcija, ki se zaveda omejitev ter kriterijev uspešnosti rešitve. Bolj kot je osebek uspešen, z večjo verjetnostjo bo izbran za starša novih osebkov. Prostor za novo generacijo je zagotovljen z odstranjevanjem najslabše ocenjenih osebkov iz populacije. Metode GA kreirajo z uporabo križanja in mutacije nove osebke. Pri operaciji križanja so novi geni kromosoma izbrani s pomočjo ene izmed naslednjih metod: enotočkovno, dvotočkovno in uniformno križanje. Pri operaciji mutacije pa so novi geni kromosoma spremenjeni naključno ali z uporabo predefinirani strategije. Strategija GA sledi Darwinovem principu "Preživetje najmočnejšega".

Reševanje sudoku mrež je kombinatorični optimizacijski problem, zato pride v poštev uporaba ustrezno prilagojenega genetskega algoritma. To pomeni, da operacije mutacij in križanj producirajo osebke v skladu s pravili reševanja sudoku mreže (Poglavje 2). Poleg tega morajo vse operacije nad mrežo ohraniti vnaprej dane vrednosti v mreži, zato so v programu statične. Ponavadi je sudoku mreža predstavljena v obliki enodimenzionalnega

$$\begin{array}{cc} \begin{bmatrix} 2 & 3 & 6 & 8 & 5 & 4 & 9 & 7 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 6 & 0 & 0 & 0 & 0 & 7 & 0 \end{bmatrix} \\ \text{Blok v polju mreže} & \text{Polje statičnih vrednosti} \\ \begin{bmatrix} 2 & 9 & 6 & 8 & 5 & 4 & 3 & 7 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 3 & 6 & 8 & 5 & 4 & 9 & 7 & 1 \end{bmatrix} \\ \text{Veljavna mutacija} & \text{Neveljavna mutacija} \end{array} \tag{7.1}$$

Slika 7.9: Primer zamenjalne mutacije nad blokom polja. Vsakič, ko se izvede operacija mutiranja, je potrebna primerjava s pomožnim poljem statičnih vrednosti. Dokler je mutacija neveljavna program ponavlja naključni izbor kandidatov za zamenjavo.

ali dvodimenzionalnega polja velikosti 81 elementov. Za kratek opis operacij nad mrežo predpostavimo, da program deluje nad enodimenzionalnem poljem[3], razdeljenim na 9 blokov velikosti 9 elementov glede na škatle, ki si sledijo z leve proti desni in od zgoraj navzdol. Operacija križanja povroči zamenjavo celotnih blokov med dvema individualnima enotama. Operacija mutacije pa zadeva posamezne bloke mreže. Najbolj pogoste operacije mutacij so zamenjalna mutacija, 3-zamenjalna mutacija in mutacija z vstavljanjem. Slika 7.9 prikazuje primer zamenjalne mutacije, aplicirane na bloku mreže. Rezultati [9] kažejo, da je reševanje sudoku mrež z uporabo genetskih algoritmov sicer mogoče, vendar ni tako učinkovito. Cilj reševanja sudoku mreže je pridobiti točno določeno (edino) rešitev mreže. To pa ni tipičen optimizacijski problem, za katerega obstaja množica rešitev, zato reševanje potrebuje preveč generacij in je program neučinkovit. Obstajajo bolj učinkoviti načini reševanja sudoku mrež, kot je na primer reševanje z uporabo učinkovite implementacije algoritma X s t.i. plesočimi povezavami dvojno povezanega seznama, podrobnosti katere opisuje Donald E. Knuth v članku “Dancing links” [7].

[3]Reprezentacija mreže ter operacije križanja in mutacije so povzete iz vira [9].

Poglavje 8

Programska rešitev

Program je s pripadajočimi metodami napisan v programskem jeziku Java. Implementacija programa sudoku je zaradi narave problema možna v vseh programskih jezikih, zato je moja izbira temeljila na osebnih preferencah značilnosti jezika. V skladu z nameni diplomskega dela sem implementirala generator, ocenjevalnik ter reševalnik sudoku mrež velikosti 81 polj.

8.1 Izbrani algoritmi ter struktura programa

Programsko rešitev sestavlja sedem javanskih razredov: TestSudoku.java, Sudoku.java, Generator.java, GenPermutate.java, Solver.java, Grade.java ter pomožni razred Helper.java. Sledi tabela z opisom posameznega razreda (Tabela 8.1).

8.2 Rezultati in meritve

Rezultat programske rešitve sudoku je poljubno število veljavnih sudoku mrež z edinstveno rešitvijo. Sudoku mreže so težavnostno ocenjene glede na stopnjo entropije mreže. Zaradi časovne zahtevnosti reševanja sudoku mrež, je program sposoben generirati le mreže z do štiriinpetdeset praznih polj (Slika 8.1).

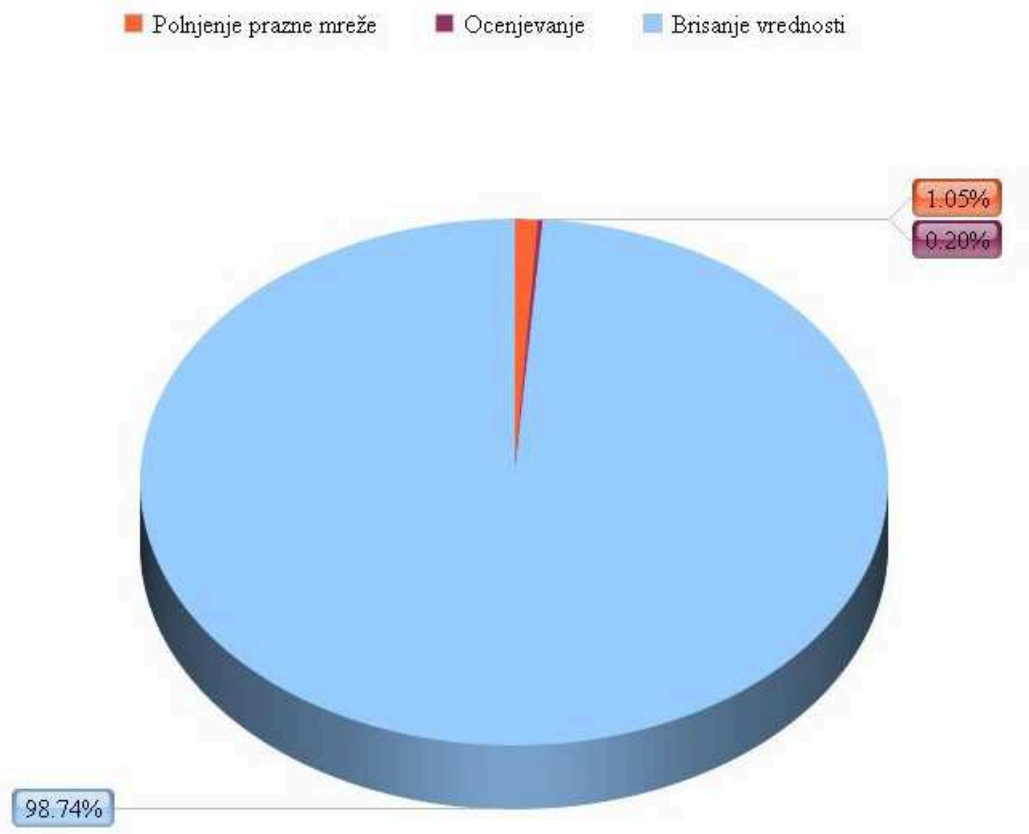
Meritve programa so osredotočene na časovno komponento problema. Za namene merjenja porabljenega časa, je bilo glede na število praznih polj opravljenih 500 iteracij metod generiranja, ocenjevanja ter reševanja sudoku mreže. Meritve so pokazale, da je povprečen čas porabljen za sudoku mrežo, ne glede na težavnostno stopnjo, med 4 in 5 ms. Sama sem težje mreže v povprečju reševala štirideset minut, medtem ko sem lažje rešila v desetih minutah. Nadaljnja analiza programa je pokazala, da porabi večino časa metoda brisanja vrednosti, ki ob vsaki zbrisani vrednosti reši mrežo (Slika 8.2). Največ časa torej potrebuje program, ko rešuje sudoku mrežo. Na Sliki 8.3 je prikazan graf

Tabela 8.1: Opis posameznega razreda programske rešitve sudokuja.

Ime razreda	Opis razreda
TestSudoku.java	Glavni razred, v katerem je deklarirana glavna metoda, nova instanca razreda sudoku s pripadajočo dimenzijo ter pomožne spremenljivke za merjenje časa.
Sudoku.java	Razred, ki vsebuje konstruktor s pripadajočo dimenzijo, ter metodo generatePuzzles, ki kot argument sprejme število mrež in število praznih polj v mreži.
Generator.java	Razred, ki vsebuje dve metodi: makePuzzle ter deleteValues. Metoda makePuzzle naključno napolni prazno sudoku mrežo ter s tem pridobi edino rešitev problema. Metoda deleteValues zbriše zahtevano število vrednosti iz pridobljene rešitve.
GenPermutate.java	Razred, ki vsebuje metode permutiranja mrež ter metodo za izračun prvotne mreže.
Solver.java	Razred, ki vsebuje metodo oneSolution, ki kot argument sprejme sudoku mrežo in število možnih rešitev. Reševalnik je implementiran s pomočjo rekurzivnega sestopanja.
Grade.java	Razred, ki vsebuje metodi, potrebni za izračun entropije sudoku mreže.
Helper.java	Razred, ki vsebuje pomožne metode za izpisovanje mreže, iskanje praznih polj v mreži ter preverjanje skladnosti mreže s pravili.

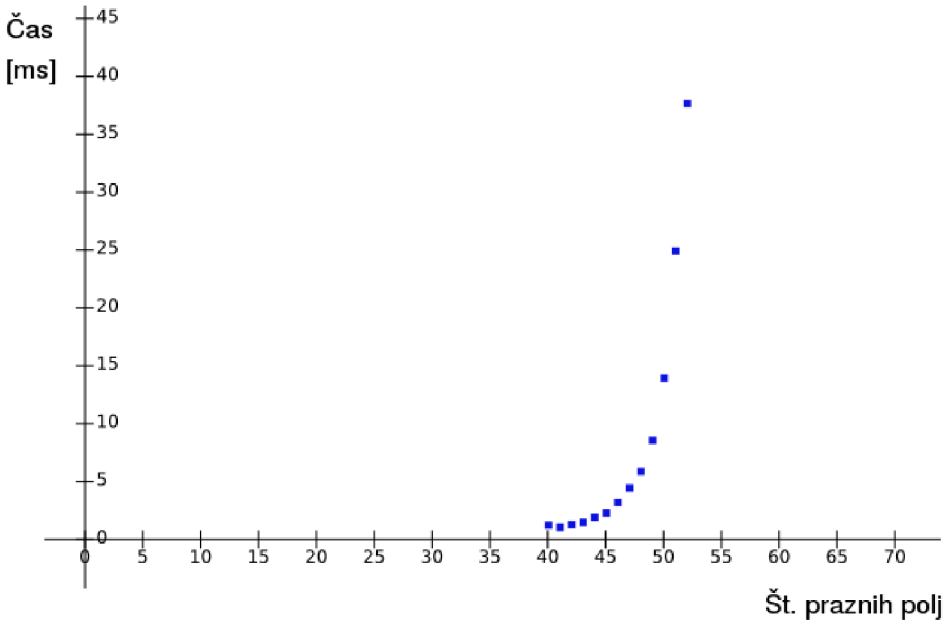
	2							
				7				
	8	6		2				3
		1					7	2
5		8		1	7	6		
			3					4
	5			6	3	8	2	7
								6
7	6						4	5

Slika 8.1: Primer pridobljene sudoku mreže lahke težavnostne stopnje.



Slika 8.2: Delež porabljenega časa.

povprečno porabljenega časa za mrežo v odvisnosti od števila praznih polj. Časovna kompleksnost programa torej ekpsonentno narašča z povečevanjem števila praznih polj v mreži.



Slika 8.3: Graf porabljenega časa generiranja, reševanja ter ocenjevanja v odvisnosti od števila praznih polj.

Poglavje 9

Zaključek in ugotovitve

Tekom izdelave diplomskega dela sem podrobno spoznala lastnosti algoritmov ter tehnik reševanja sudoku mreže. Uvodna poglavja ter poglavja o reševanju, generiranju in ocenjevanju sudoku mrež se nanašajo na raziskane obstoječe rešitve problema. Implementacija lastnega programa mi je omogočila nadaljno razumevanje težavnosti problema. Implementacija generiranja s pomočjo permutacij je omogočila hitro pridobivanje veljavnih mrež, vendar znotraj istega ekvivalenčnega razreda. Zato je bilo bolj primerno implementirati generiranje z brisanjem vrednosti. Kot je bilo predvideno, je reševanje sudoku mreže z rekurzivnim sestopanjem nezadostno za hitro pridobivanje rešitev mrež zlobne težavnostne stopnje. Nadaljno sem z implementacijo ocenjevanja s pomočjo entropije ugotovila, da prihaja pri ocenjevanju lažjih mrež do težav, zaradi t.i. "ozkega grla" mreže. V postopku reševanja mreže se velikokrat zgodi, da postane po vnosu določene vrednosti mreža precej preprosta. Zaradi tega prihaja pri ocenjevanju z entropijo do napak in je potrebno upoštevati človeške tehnike reševanja.

Literatura

[1] M. H. Alsuwaiyel. *Algorithms: Design Techniques and Analysis*. London: World Scientific Publishing Company, 1999.

[2] Estimation lemma. Latin square — Wikipedia, the free encyclopedia, 2014. [Online; dostop 1-Julij-2014].

[3] Estimation lemma. Techniques for solving sudoku, 2014. [Online; dostop 1-Avgust-2014].

[4] Bertram Felgenhauer and Frazer Jarvis. Mathematics of sudoku i. *Mathematical Spectrum*, 2006.

[5] Junhong Zhang Gaoshou Zhai. Solving sudoku puzzles based on customized information entropy. *International Journal of Hybrid Information Technology*, 2013.

[6] R.M. Wilson J.H. Van Lint. *A course in combinatorics*. Cambridge, 1992.

[7] Donald E. Knuth. Dancing links. *Millenial Perspectives in Computer Science*, 2000.

[8] Wei-Meng Lee. *Programming Sudoku*. Apress, 2006.

[9] Timo Mantere and Janne Koljonen. Solving and rating sudoku puzzles with genetic algorithms. In *New Developments in Artificial Intelligence and the Semantic Web, Proceedings of the 12th Finnish Artificial Intelligence Conference STeP*. Citeseer, 2006.

[10] Anthony B. Evans Paul Erdös. Representations of graphs and orthogonal latin square graphs. *Journal of Graph Theory*, 1989.

[11] Ed Russell and Frazer Jarvis. Mathematics of sudoku ii. *Mathematical Spectrum*, 2006.

[12] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.

[13] LI Yong-zhuo XUE Yuan-hai, JIANG Biao-bin. Sudoku puzzles generating: from easy to evil. *Mathematics in Practice and Theory*, 2009.