

# SUDOKU

(Mladen Ilić E1 28/2023)

Prilikom rešavanja prvog ispitnog zadatka od nas se zahteva da koristimo SA algoritam (algoritam simuliranog kaljenja) koji je inspirisan termodinamičkim procesom kaljenja metala. Koristi se za pronalaženje globalnog optimuma u velikom prostoru rešenja.

Ukratko ovaj algoritam se može opisati u sledeća četiri koraka:

1. Prilikom inicijalizacije, početno rešenje se bira nasumično i postavlja se početna temperatura.
2. Nakon toga izvršava se broj iteracija koje smo zadali na početku i iz trenutnog rešenja se generiše novo rešenje u susedstvu prethodnog. Ako je novo rešenje bolje, tj. ima niži *cost* od prethodnog, usvaja se. U slučaju da novo rešenje nije bolje, prihvatata se sa verovatnoćom  $e^{-\Delta E/T}$ , gde je  $\Delta E$  razlika u vrednostima *cost*-a novog i starog rešenja.
3. Hlađenje je korak u kom se temperatura smanjuje prema određenom obrascu (faktor hlađenja se kreće u rasponu od 0 do 1).
4. Četvrti korak se odnosi na zaustavljanje koje se dešava nakon što temperatura padne ispod unapred definisane granice ili dok se ne izvrši određeni broj iteracija.

U nastavku će biti ukratko opisana struktura koda koji sam koristio za rešavanje ovog zadatka a koji se zasniva na fajlovima koji su preuzeti sa sledećeg linka: <http://yarpiz.com/223/ypea105-simulated-annealing>.

- **CreateModel.m** – unutar ovog fajla je kod funkcije koji učitava vrednosti iz tekstualnih fajlova.
- **SudokuCost.m** – *cost* funkcija prolazi kroz sve vrste i kolone i traži brojeve koji nedostaju i vraća vrednost *cost*-a koji predstavlja ukupan broj nedostajućih brojeva kako bi sudoku ukrštenica bila tačno rešena. Sudoku je rešen kad je vrednost *cost*-a jednaka 0.
- **CreateRandomSolution.m** – funkcija koja nasumično popunjava sva prazna polja (polja sa brojem 0) iz učitane sudoku ukrštenice.
- **CreateNeighbor.m** – ova funkcija generiše novo susedno rešenje tako što zameni vrednosti dve nasumično izabrane ćelije koje nisu fiksirane. Prvo pronađe sve nefiksirane ćelije, zatim izabere jednu nasumično i pronađe drugu nefiksiranu ćeliju u istom 3x3 regionu, i na kraju zameni njihove vrednosti kako bi kreirala novu tablu.
- **PlotSudokuGrid.m** – funkcija koja se koristi kako bi se grafički prikazala nerešena sudoku ukrštenica učitana iz tekstualnog fajla, a takođe i ukrštenica nakon rešavanja.
- **sa.m** – fajl koji sadrži implementaciju celog SA algoritma i u njemu se inicijalizuju svi potrebni parametri (broj iteracija, inicijalna temperatura, faktor hlađenja) i pozivaju sve prethodno opisane funkcije.
- **main.m** – u ovom fajlu se startuje SA.

Na sledećoj slici može da se vidi primer u kom je SA algoritam pronašao rešenje sudoku ukrštenice u iteraciji 134 i to primećujemo jer je vrednost *cost*-a jednak 0. Takođe u nastavku će biti i prikazan deo koda u kojem su inicijalizovane vrednosti parametara.

Iteration 129: Best Cost = 2  
 Iteration 130: Best Cost = 2  
 Iteration 131: Best Cost = 2  
 Iteration 132: Best Cost = 2  
 Iteration 133: Best Cost = 2  
 Iteration 134: Best Cost = 0

#### %% SA Parameters

MaxIt = 350;      % Maximum Number of Iterations  
 MaxSubIt = 20;    % Maximum Number of Sub-iterations  
 T0 = 0.04;        % Initial Temp. default T0 = 0.025  
 alpha = 0.99;     % Temp. Reduction Rate

Izgled sudoku ukrštenice pre i posle rešavanja:

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

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