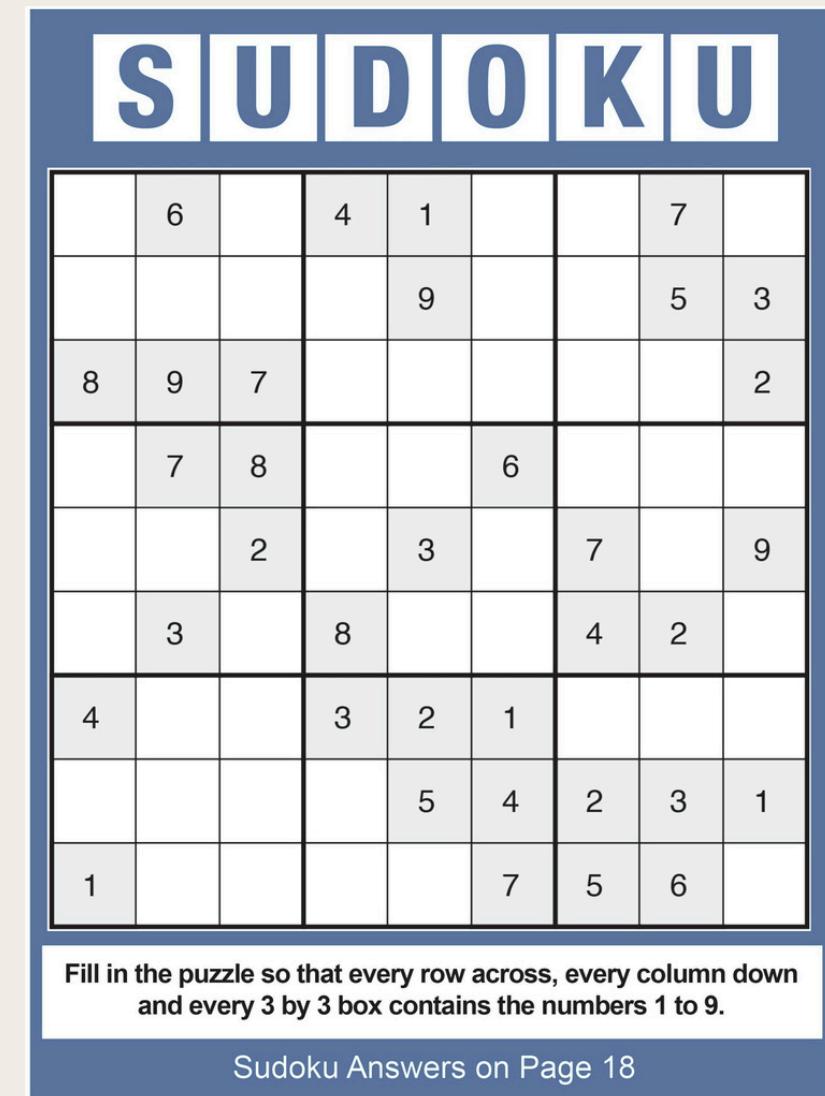




РАČУНАРСКА ИНТЕЛИГЕНЦИЈА

# SUDOKU PROBLEM

- NP-težak kombinatorni problem
- Cilj: razviti hibridni algoritam koji kombinuje prednosti Lokalne Pretrage (LS) i Programiranja Ograničenja (CP)
- Iterativna Lokalna Pretraga (ILS - Iterated Local Search): ponavlja lokalnu pretragu uz kontrolisane perturbacije
- Programiranje Ograničenja (CP - Constraint Programming): koristi ograničenja da popuni tablu logički
- Kombinovanjem se postiže ravnoteža između brzine i kvaliteta rešenja



# FORMALNA OGRANIČENJA

- **Zadatak:** popuniti  $N^2 \times N^2$  mrežu brojevima u opsegu od 1 do  $N^2$   
( $N$  - dimenzija zagonetke)

- **Tri ključna ograničenja:**

1. Ograničenje **reda**: Svaki broj od 1 do  $N^2$  mora se pojaviti tačno jednom u svakom redu

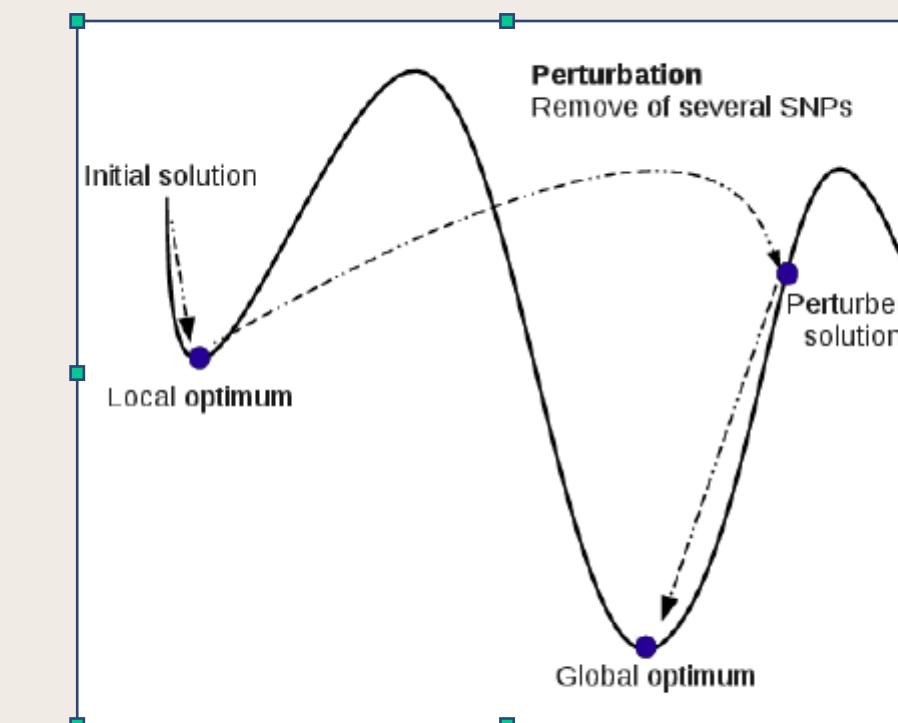
2. Ograničenje **kolone**: Svaki broj od 1 do  $N^2$  mora se pojaviti tačno jednom u svakoj koloni

3. Ograničenje **bloka**: Svaki broj od 1 do  $N^2$  mora se pojaviti tačno jednom u svakom  $N \times N$  bloku



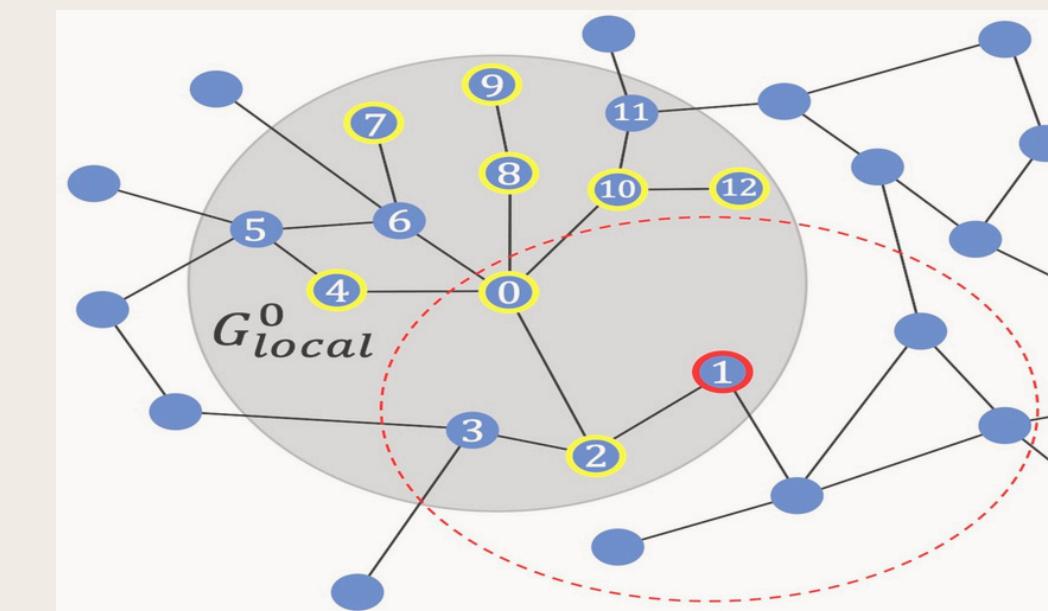
# ITERATIVNA LOKALNA PRETRAGA (ILS)

- Metaheuristika koja se koristi za prevazilaženje osnovnog nedostatka standardnih metoda lokalnog pretraživanja - zaglavljivanje u **lokalnom minimumu**
- Gradi sekvencu lokalno optimalnih rešenja korišćenjem znanja stečenog u prethodnim iteracijama
- **Ključni koraci:**
  1. **Lokalna pretraga (LS)**
  2. **Perturbacija**
  3. **Kriterijum prihvatanja**



# LOKALNA PRETRAGA (LS)

- **Lokalno Pretraživanje** (Local Search): proces pronalaženja **lokalnog minimuma** polazeći od trenutne konfiguracije
- Prelazi sa trenutnog rešenja na susedno primenom lokalnih promena - pomeranja
- **Iterativni proces** - nastavlja se sve dok se ne pronađe rešenje koje zadovoljava kriterijume optimalnosti ili dok se ne prekorači definisano vremensko ograničenje
- **Cilj** - dostizanje lokalnog minimuma pre primene perturbacija
- Korišćena heuristika: **Min-Conflicts**



# MIN-CONFLICTS HEURISTIKA

Min-Conflicts algoritam postupa po sledećem principu:

1. Izabrati promenljivu koja je trenutno u konfliktu
2. Dodeliti joj onu vrednost koja minimizuje broj preostalih konfliktata sa ostalim promenljivim (u slučaju izjednačenja izbor se vrši nasumično)



# TABU SEARCH

**Tabu lista** - skladištenje nedavno izvršenih zamen s ciljem sprečavanja **cikličnog** kretanja kroz prostor pretraživanja

Pomeranja koja su na listi su **tabu** - zabranjena za razmatranje (zabrana se ignoriše ako pomeranje dovodi do smanjenja troška)

## Kriterijum prihvatanja:

1. Kandidati koji rezultiraju nižim troškom rešenja uvek se prihvataju
2. Kandidati koji dovode do većeg ili jednakog troška prihvataju se samo uz određenu verovatnoću

Proces se ponavlja dok se ne pronađe optimalno rešenje ili se ne dostigne ograničenje broja iteracija

# PERTURBACIJA

- **Primarna funkcija** - izvlačenje rešenja iz lokalnih optimuma koje je prethodno pronašao algoritam lokalnog pretraživanja (Min-Conflicts sa Tabu listom)
- **Hibridni mehanizam** - u proces ILS-a se uključuje Constraint Programming (CP)
- **Kontrolisano kvarenje** trenutnog najboljeg rešenja da bi se dobilo **novo** polazno rešenje
- Vrši se u dva koraka:
  1. **Pražnjenje ćelija** koje su u konfliktu
  2. **Primena CP** na namerno narušenu tablu kako bi se izvršila intenzivna pretraga i popunile preostale praznine
- Time se osigurava da sledeći poziv rutine lokalnog pretraživanja započne pretragu u novoj, delimično popravljenoj oblasti



# KRITERIJUM PRIHVATANJA

## 1 - Unutar Lokalnog Pretraživanja - Prihvatanje Poteza

- Primjenjuje na svakoj iteraciji unutar rutine Min-Conflicts heuristike pojačane Tabu listom, odlučuje da li će se najbolji dopušteni potez prihvati
- Kriterijum kombinuje elemente **Hill Climbinga** i **Simuliranog Kaljenja** (Simulated Annealing)
- Potez se prihvata ako je ispunjen bilo koji od sledećih uslova:

1. **Determinisano prihvatanje:** Ako predloženi potez dovodi do nižeg troška od trenutnog rešenja, potez se automatski prihvata

2. **Stohastičko prihvatanje:** Ako predloženi potez dovodi do većeg ili jednakog troška, potez se prihvata samo uz određenu verovatnoću (izlazak iz plitkih lokalnih optimuma)

3. **Aspiracijski kriterijum:** Ako tabu potez dovodi do rešenja koje je bolje od globalno najboljeg, prihvata se bez obzira na tabu status

## 2 - Unutar ILS Petlje - Prihvatanje Rešenja

- Deluje na nivou **celog** rešenja, nalazi se u glavnoj petlji
- Služi za održavanje globalno najboljeg rešenja tokom ukupnog procesa ILS-a



- Vraćeni lokalni optimum se prihvata kao **novo** globalno najbolje rešenje samo ako je trošak tog rešenja niži od troška trenutnog najboljeg rešenja
- Stohastika osigurava efikasno kretanje unutar regiona
- Klasični kriterijum garantuje da se zadržava najbolji rezultat tokom ukupnog procesa ILS-a

# KLASA SUDOKUSOLVER

Klasa služi kao temeljni model koji enkapsulira stanje Sudoku problema i definiše pravila igre

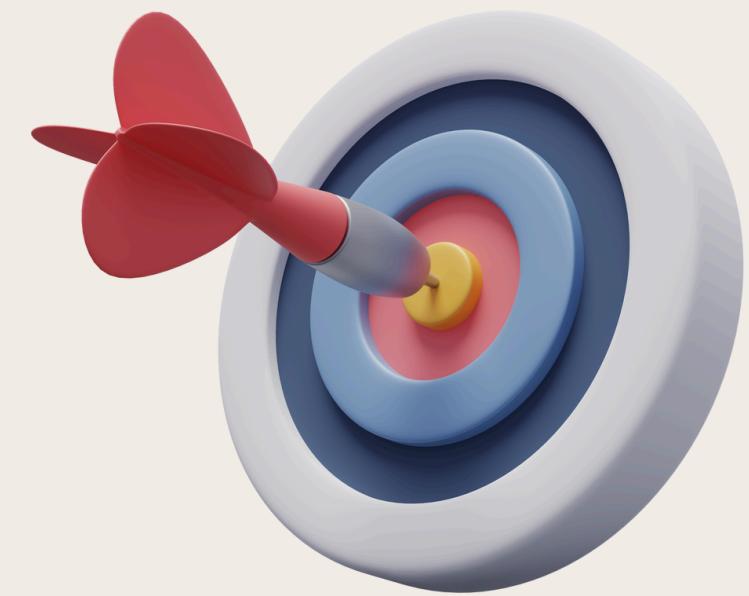
## Osnovni model:

- **self.grid** - radna tabla predstavljena kao NumPy matrica
- **self.N** - čuva dimenziju table ( $N \times N$ )
- **self.K** - čuva dimenziju bloka ( $K = \text{sqrt}(N)$ )
- **self.fixed\_mask**: binarna matrica koja označava koje ćelije su originalno zadate i moraju ostati fiksirane (True) tokom pretraživanja



# FUNKCIJA CILJA

- **Trošak rešenja** definiše se kao ukupan broj prekršaja pravila u redovima, kolonama i blokovima
- **Cilj** algoritma je da postigne trošak  **$C = 0$**
- **objective\_f (funkcija cilja)** - sumira konflikte iz sve tri dimenzije
- Ukupan trošak  **$C = C_{Row} + C_{Col} + C_{Block}$**   
(zbroji konflikata u redovima, kolonama i blokovima)



## Pomoćne funkcije za računanje konflikata

- **\_line\_conflicts** - izračunava minimalni broj izmena potreban za uklanjanje konflikata
- **\_row\_conflicts** - ukupan broj konflikata za redove
- **\_col\_conflicts** - ukupan broj konflikata za kolone
- **\_block\_conflicts** -ukupan broj konflikata za blokove

# KLASA ILS\_CP

- Klasa ILS\_CP je jezgro implementacije hibridnog algoritma Iterativne Lokalne Pretrage (ILS) u spremi sa Programiranjem Ograničenja (CP)

- Podklasa bazne klase SudokuSolver, čime nasleduje:

1. Osnovnu reprezentaciju Sudoku mreže (**self.grid**, **self.fixed\_mask**)

2. Funkciju cilja **objective\_f**

3. Osnovne metode za rad sa Sudoku ograničenjima

- Dodatni atributi:

- **self.best\_cost** - najmanji trošak

- **self.best\_grid** - čuva najbolje globalno rešenje

- **self.tabu\_list** - za lokalnu pretragu

```
procedure Iterated Local Search
     $s_0 = \text{GenerateInitialSolution}$ 
     $s^* = \text{LocalSearch}(s_0)$  % optional
    repeat
         $s' = \text{Perturbation}(s^*)$ 
         $s^{*'} = \text{LocalSearch}(s')$ 
         $s^* = \text{AcceptanceCriterion}(s^*, s^{*'})$ 
    until termination condition met
end
```

# KLJUČNE METODOLOŠKE FUNKCIJE

- Brzo spuštanje ka Lokalnom Optimumu

## \_min\_conflicts\_with\_tabu

- Sprovodi lokalnu pretragu
- Pronalazi potez koji najviše smanjuje broj konflikata
- Koristi **Tabu listu** i aspiracijski kriterijum da spreči ciklično kretanje

## perturb

- Sprovodi perturbaciju i hibridizaciju
- Nasumično prazni deo polja (**p\_rate**)
- Poziva CP fazu (**cp\_refinement**) sa vremenskim ograničenjem (**cp\_time\_limit**)
- Ako CP pronađe validno rešenje → prihvata se kao novo bolje stanje

## solve ils\_cp

- Glavna ILS petlja – kontroliše ceo proces
- Iterativno poziva LS i perturbaciju
- Ažurira najbolje rešenje
- Postepeno smanjuje faktor kvarenja (**empty\_factor \*= α**) radi fokusiranja pretrage

# HIBRIDNI PERTURB MEHANIZAM

- Na osnovu faktora kvarenja (**empty\_factor**), nasumično briše vrednosti ( $\text{grid}[i, j] = 0$ ) iz određenog broja konfliktnih polja
- Izbacivanje algoritma iz Lokalnog Optimuma u novi, neistražen region
- Instancira se novi CP rešavač (SudokuCP) sa **osakaćenom** mrežom
- CP rešavač pokušava da **popuni** preostale prazne ćelije unutar kratkog vremenskog roka (**cp\_time\_limit**)
- Ako CP pronađe rešenje i to rešenje je **validno** (po Sudoku pravilima) i **kompletno** (nema praznih polja), to predstavlja snažan skok ka rešenju
- Time se kompenzuju nedostatci ILS algoritma u finalnom podešavanju

# GLAVNA PETLJA (SOLVE\_ILS\_CP)

- **Funkcija:** Kontroliše ceo proces rešavanja
- Proces: **Iterativno** primenjuje:
  1. **Lokalnu Pretragu** (\_min\_conflicts\_with\_tabu)
  2. **Perturbaciju** (perturb - hibridni skok)
- **Adaptacija:** Faktor kvarenja se postepeno smanjuje (**empty\_factor \*= alpha**) što sužava pretragu
- **Kriterijum:** Neprestano se ažurira najbolje rešenje (**self.best\_cost**) i prati statistika

---

**Algorithm 2** Iterated local search for Sudoku

---

**Input:** puzzle, timeLimit, resetFactor,  $\alpha$

```
1: FIXCELLSUSINGARCCONSISTENCY(puzzle)
2:
3: FILLREMAININGCELLSRANDOMLY(puzzle)
4:
5: bestPuzzle  $\leftarrow$  puzzle
6: bestCost  $\leftarrow$  EVALUATE(puzzle)
7:
8: while bestCost > 0  $\wedge$  timeLimit not passed do
9:   puzzle  $\leftarrow$  MINCONFLICTSWITHTABULIST(puzzle)
10:
11:  cost  $\leftarrow$  EVALUATE(puzzle)
12:
13:  if bestCost > cost then
14:    bestCost  $\leftarrow$  cost
15:    bestPuzzle  $\leftarrow$  puzzle
16:  end if
17:
18:  if cost > 0 then
19:    Empty all unfixed cells in puzzle which are in conflict
20:
21:    Additionally empty relative amount of
22:    remaining unfixed cells defined by resetFactor
23:
24:    FORWARDCHECKINGSEARCH(puzzle)
25:
26:    FILLREMAININGCELLSRANDOMLY(puzzle)
27:
28:    resetFactor  $\leftarrow$  resetFactor  $\cdot \alpha$ 
29:  end if
30: end while
```

**Output:** bestPuzzle

---

# KLASA: SUDOKUCP

- Implementacija rešavanja Sudokua korišćenjem Constraint Programming (CP)
- **Nasleđeno:** Osnovna reprezentacija Sudoku mreže (`self.grid`)
- `self.random` - NumPy generator za inicijalizaciju
- `self.cp_vars` - instanca CP modela
- `self.cp_solver` - Instanca rešavača
- Koristi **Google OR-Tools** za efikasno pronalaženje rešenja pod zadatim ograničenjima
- **Cilj** - Pronaći validno rešenje Sudokua (ili doraditi parcijalno rešenje) poštujući sva pravila
- `_build_cp_model` - Modelovanje problema
  1. Kreira  $N \times N$  celobrojnih varijabli
  2. Koristi **AddAllDifferent** da osigura da su brojevi različiti u svakom redu, svakoj koloni i svakom bloku
- Fiksira sve **originalno zadate brojeve** kao konstante u CP modelu, smanjujući prostor pretrage



# KLASA: SUDOKUCP

- `_is_cell_nonconflicting` - Brza Provera Konflikta

1. Proverava da li vrednost (i,j) stvara **konflikt** sa susedima
2. Vraća **False** ako se vrednost ponavlja ili je 0
3. Koristi NumPy (`np.count_nonzero`) za brzu detekciju konflikata



## GLAVNA FUNKCIJA: CP\_REFINEMENT

- Drastično smanjuje broj slobodnih varijabli za rešavača
- Osvežava CP model i fiksira originalne brojeve i nekonfliktne vrednosti (ako je `fix_noncon` omogućen)
- Usmerava pretragu ka postojećem parcijalnom rešenju, ubrzavajući pronalaženje
- Postavlja **strogoo** ograničenje (`max_time_in_seconds`) - Osigurava da se CP faza završi brzo kada se koristi kao deo bržeg hibridnog algoritma (poput ILS)
- `solver.Solve(model)` - Ako je rešenje pronađeno (OPTIMAL/FEASIBLE), cela mreža se ažurira

# EKSPERIMENTALNO ISTRAŽIVANJE

## Prva serija eksperimenata - Eksperimenti validacije performansi i parametara

1. Potvrda **Skalabilnosti** (Brzina rešavanja većih problema)
  2. Ispitivanje Uticaja parametara **alpha**
  3. Ispitivanje Uticaja parametara **broj LS iteracija**
- **Metodologija:** Testiranje na instancama od **4×4** do **25×25** različitih težina

## Druga serija eksperimenata - Uporedni eksperiment (Verifikacija efikasnosti)

1. Stopa uspešnosti u odnosu na težinu problema
- **Metodologija:** testiranje **9×9** Sudokua sa težinom od **5%** do **100%** fiksnih celija



# ULAZNE INSTANCE, PRVA SERIJA EKSPERIMENTA

ID	Dimenzija ( $N \times N$ )	Blok ( $K \times K$ )	Fiksnih ćelija	Napomena / Težina
1	$4 \times 4$	$2 \times 2$	100%	Lak (Već rešen)
2	$4 \times 4$	$2 \times 2$	25%	Srednje težak
3	$4 \times 4$	$2 \times 2$	0%	Težak
4	$9 \times 9$	$3 \times 3$	35%	Srednje težak
5	$9 \times 9$	$3 \times 3$	0%	Težak (Potpuno prazna)
6	$9 \times 9$	$3 \times 3$	10%	Težak
7	$16 \times 16$	$4 \times 4$	6%	Težak
8	$16 \times 16$	$4 \times 4$	0%	Težak (Potpuno prazna)
9	$25 \times 25$	$5 \times 5$	4%	Težak, Problem velikih dimenzija
10	$9 \times 9$	$3 \times 3$	N/A	Nerešiv problem
11	$16 \times 16$	$4 \times 4$	N/A	Nerešiv problem (Unutrašnji konflikt)

Parametar	Eksperiment 1	Eksperiment 2	Eksperiment 3
<b>Naziv</b>	Bazno testiranje	Optimizacija perturbacije	Optimizacija intenzifikacije
Maksimalno vreme (s)	300	300	300
Faktor kvarenja ( $\alpha$ )	0.995	<b>0.2</b>	0.995
Maks. LS poziva/iter.	5000	5000	<b>50</b>
Dužina Tabu liste	10	10	10
Verovatnoča prihvatanja	0.15	0.15	0.15
CP vremensko ogr.	15	15	15
Početni Empty Factor	0.2	0.2	0.2

# REZULTATI EKSPERIMENTA 1

ID	Veličina (N)	LS Iter.	Exec. Time (s)	Best Cost	Rešen	CP Poziva	CP Uspesi
1	4	0	0.0002	0	True	0	0
2	4	0	0.0001	0	True	0	0
3	4	0	0.0001	0	True	0	0
4	9	5000	18.4516	0	True	1	1
5	9	5000	18.5609	0	True	1	1
6	9	5000	18.7135	0	True	1	1
7	16	5000	58.0719	0	True	1	1
8	16	5000	58.6738	0	True	1	1
9	25	5000	144.056	0	True	1	1
10	9	1000000	37.252	2	False	200	1
11	16	1000000	58.043	90	False	200	1

## LAKI PROBLEMI (N=4):

Pronalazak rešenja **ekstremno brzo** (ispod 0.001s) i sa 0 LS iteracija

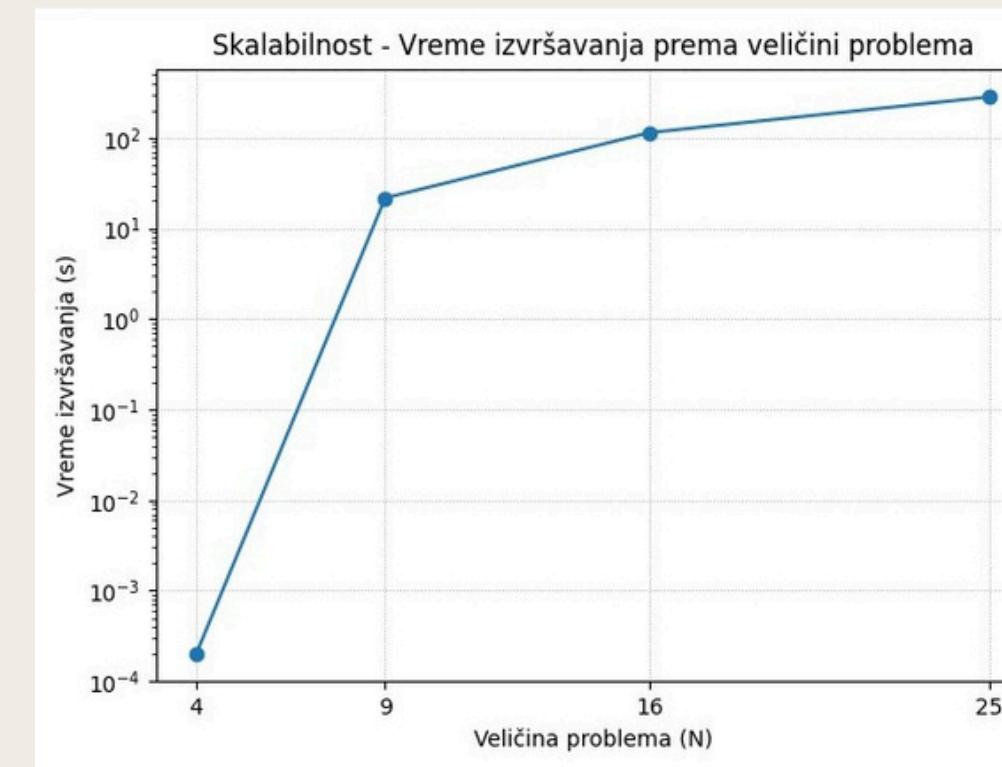
**Hibridizacija isključena:** CP poziva = 0 (hibridni deo algoritma se aktivira samo kada je to neophodno za složenije probleme)

## VEĆI PROBLEMI (N=9,16,25):

Rešenje je pronađeno sa samo **1 pozivom CP**

LS dovodi konfiguraciju do stanja skoro bez konflikata, CP služi kao finišer

# REZULTATI EKSPERIMENTA 1



Efikasnost za male probleme: Vreme izvršavanja za N=4 je izuzetno nisko

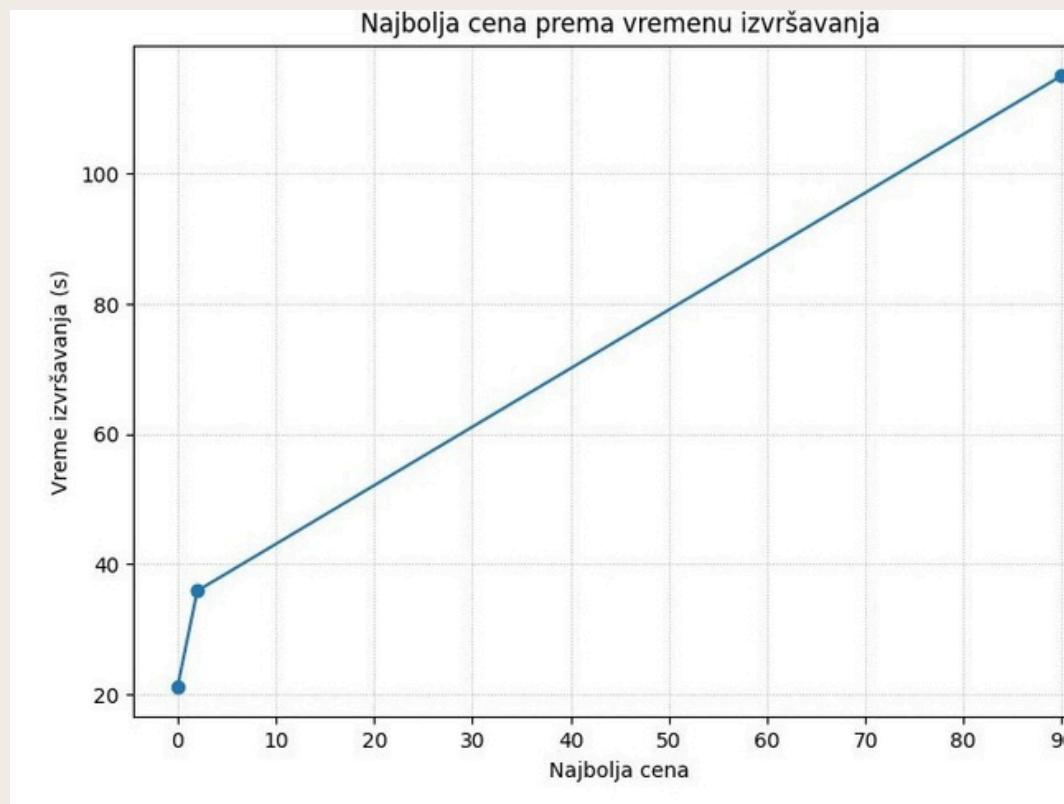
Vreme raste **superlinearno** sa dimenzijom ( $N=9 \approx 10s$ ,  $N=25 \approx 144s$ ). Ovo vizuelno potvrđuje da je problem **NP-težak**

## PONAŠANJE NA NEREŠIVIM PROBLEMIMA

Algoritam **troši sve resurse** (1.000.000 LS Iteracija) i 200 CP poziva, ali **završava neuspešno**

Potvrda da se algoritam **ne zaglavljuje**, već troši resurse u intenzivnom, ali neuspešnom pokušaju CP Perturbacije

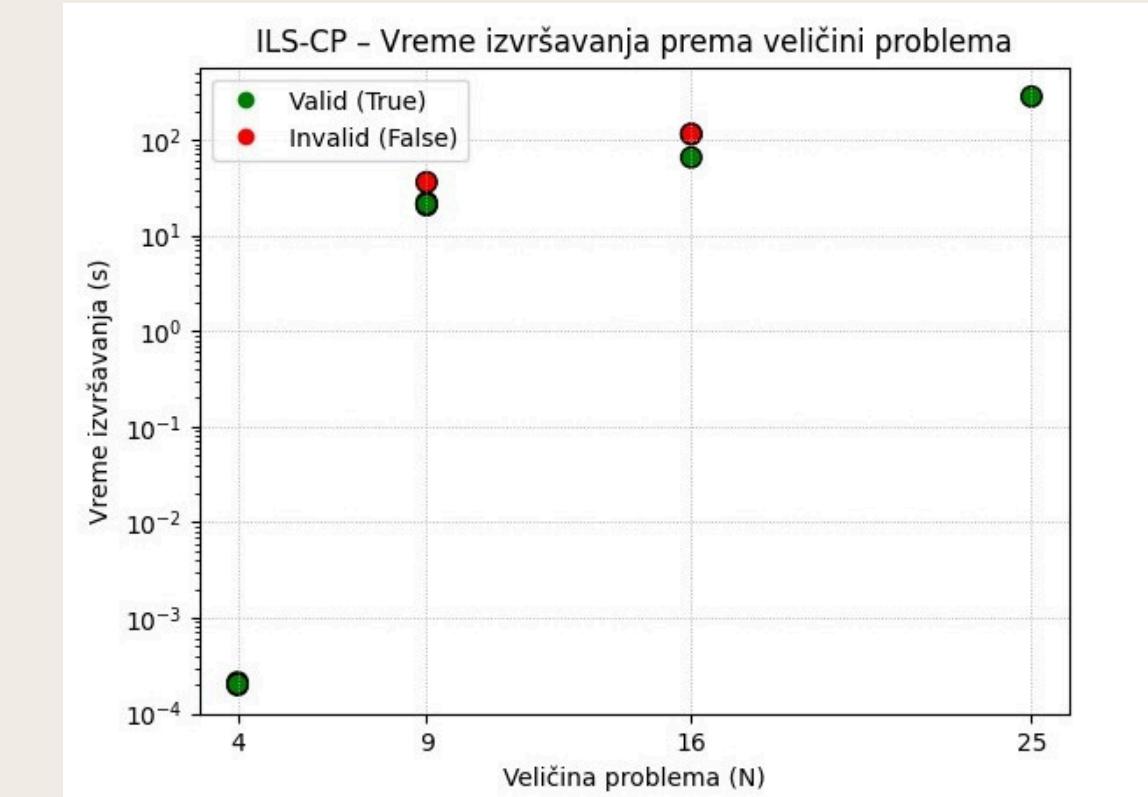
# REZULTATI EKSPERIMENTA 1



**Medijana** je robustan pokazatelj jer na nju ne utiču ekstremne vrednosti, za razliku od proseka

Korelacija između performansi i težine problema: što je Best Cost dalje od nule to je potrebno više vremena za njegovo neuspešno rešavanje

Algoritam prolazi kroz brojne neuspešne cikluse perturbacije i intenzifikacije pre nego što se zaustavi



Instance koje su uspešno rešene (Valid) prate superlinearni trend rasta vremena

Nerešeni problem zahteva više vremena nego rešeni (crvena tačka je iznad zelene)

Algoritam troši dodatno vreme u fazi Diverzifikacije ( CP Perturbacija) u pokušaju da se izvuče iz lokalnog optimuma. Dostiže pri tome maksimalan broj iteracija ili vreme

# REZULTATI EKSPERIMENTA 2

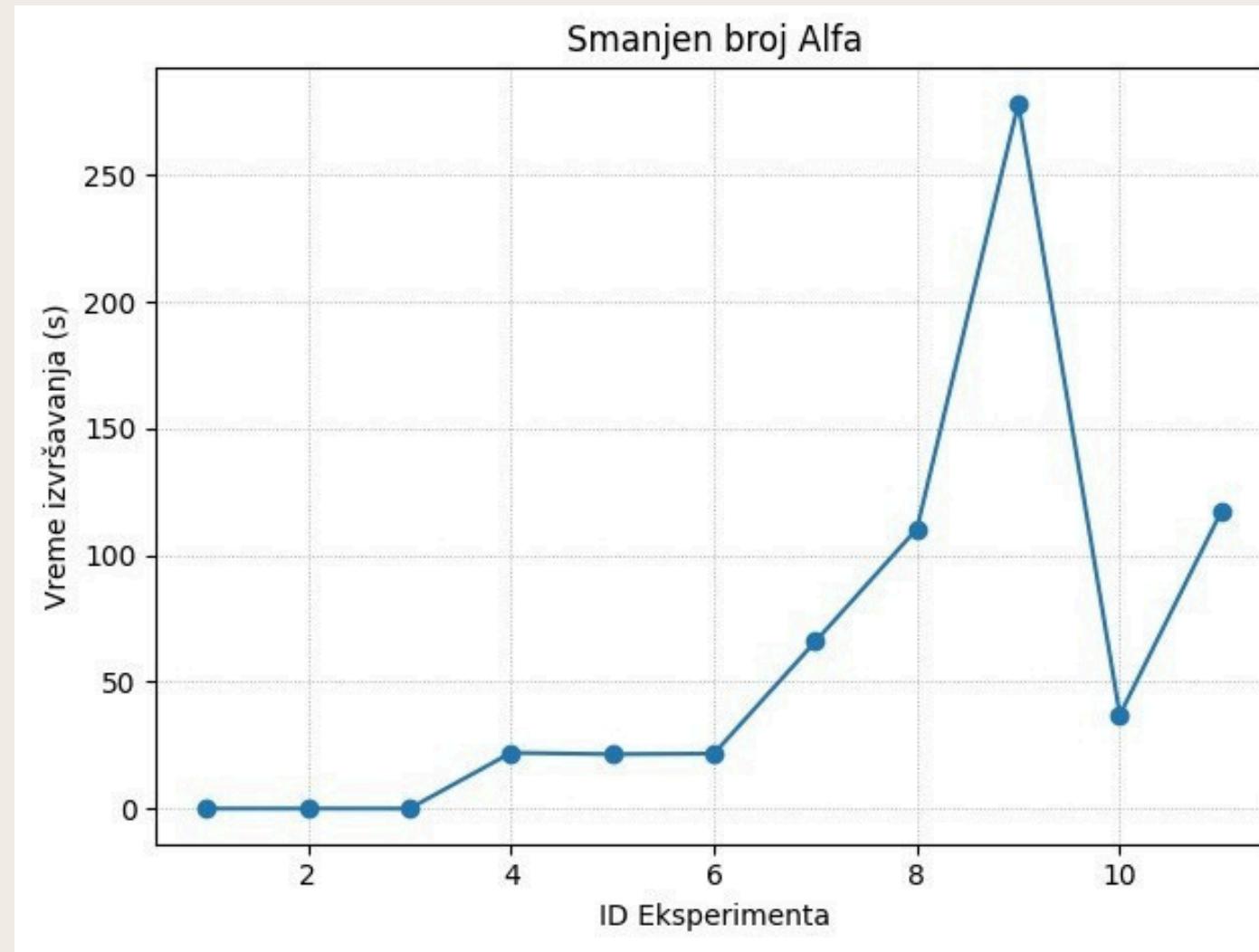
## UTICAJ PARAMETRA ALPHA

Instanca	(N)	Total Iter	LS Iter	Vreme	Best Cost	CP Pozivi	CP Uspeha
1-3	4	0	0	$\approx 0.0002$	0	0	0
4-6	9	1	5000	21.46 – 22.03	0	1	1
7-8	16	1	5000	65.94 – 110.35	0	1	1
9	25	1	5000	278.219	0	1	1
10	9	200	10000	37.004	2	200	1
11	16	200	10000	117.197	90	200	1

Promena strategije sa **Intenzifikacija** (pretraživanje rešenja blizu trenutnog optimuma) na **Eksploraciju** (skakanje daleko od trenutne lokacije)

Promena sa male perturbacije (**alpha = 0.995**) na agresivna diversifikacija (**alpha = 0.2**)

# REZULTATI EKSPERIMENTA 2



**Execution Time (s)** - Gotovo nepromenjen za probleme Problem\_Size=4, 9, 16

**Best Cost** - Obe strategije (Intenzifikacija i Agresivna diversifikacija) su zaglavljene u **istim** lokalnim optimumima

Drastičan skok na **≈275s** (najteža instanca, Problem\_Size=25), rešena na limitu postavljenih resursa

# REZULTATI EKSPERIMENTA 3

## SMANJEN BROJ LS ITERACIJA

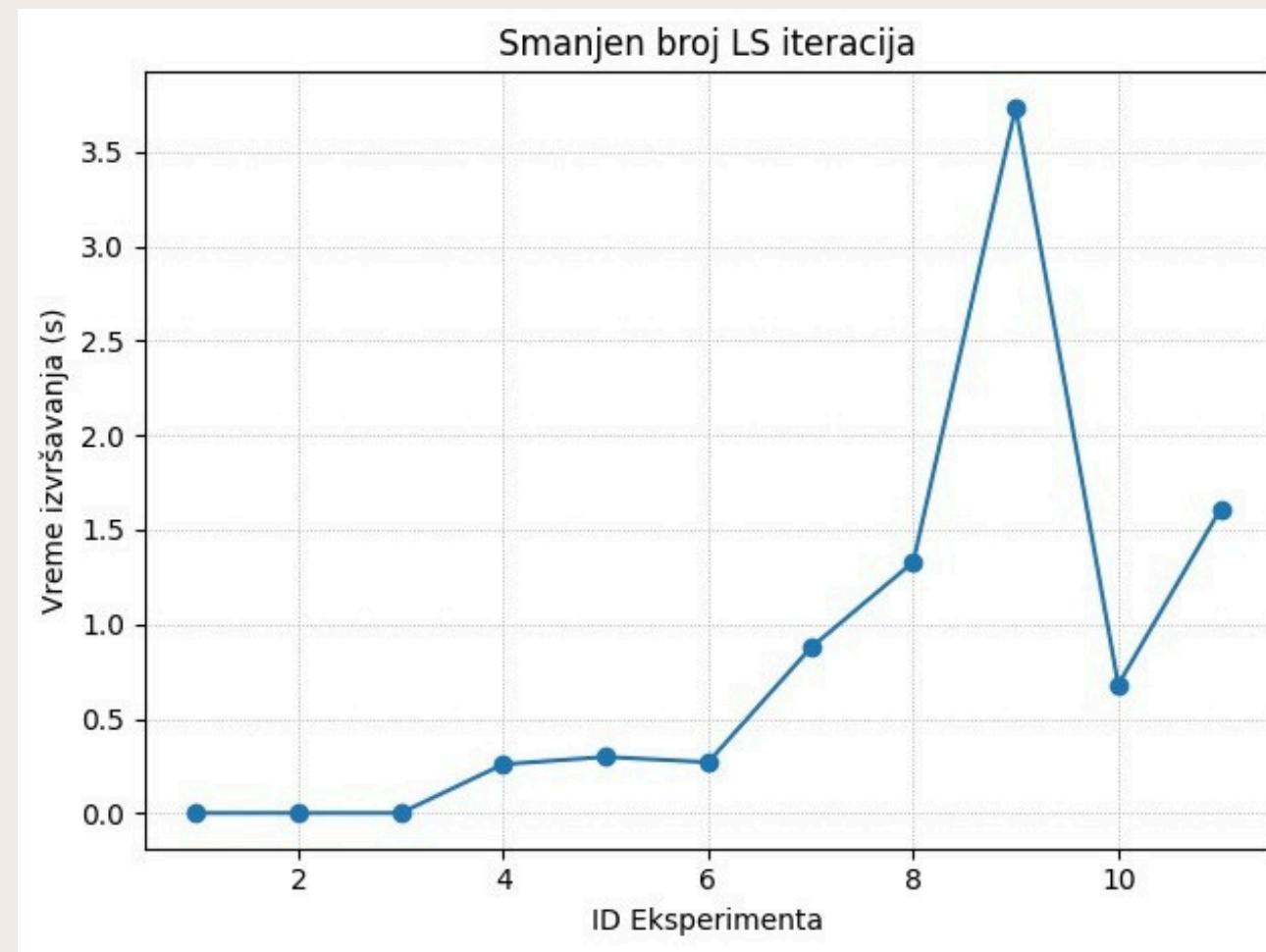
Iteracija	(N)	LS Iter.	$\alpha$	Vreme	Best Cost	Validno	CP Pozivi
1-3	4	0	0.995	$\approx 0.0002$	0	True	0
4-6	9	50	0.995	0.2584 – 0.2975	0	True	1
7-8	16	50	0.995	0.8752 – 1.3291	0	True	1
9	25	50	0.995	3.7345	0	True	1
10	9	2500	0.995	0.6796	2	False	50
11	16	2500	0.995	1.6111	90	False	50

Smanjenje iteracija za **100** puta rezultuje smanjenjem vremena za **≈80** puta.

LS\_Iterations je **ključni faktor performansi**

Brzo dostiže kriterijum zaustavljanja, pre nego što CP perturbacija i LS pretraga imaju priliku za kvalitativnu diversifikaciju

# REZULTATI EKSPERIMENTA 3



Maksimalno vreme opada sa  $\approx 275\text{s}$  na  $\approx 3.7\text{s}$

Vreme izvršavanja raste **linearno** i predvidivo sa veličinom problema, dostižući **vrhunac** od  $\approx 3.7\text{s}$  za najveći rešivi problem ( $N=25$ )

# ULAZNE INSTANCE, DRUGA SERIJA EKSPERIMENTA

ID Instanca	Kategorija Težine (%)	Br Fiksnih Ćelija	Tip instanca
1-2	5%	4	Ekstremno teška
3-4	10%	8	Ekstremno teška
5-6	15%	12	Teška
7-8	20%	16	Teška
9-10	25%	20	Teška
11-12	30%	24	Srednja
13-14	40%	32	Srednja
15-16	50%	40	Laka
17-18	60%	48	Laka
19-20	70%	56	Laka
21-22	80%	64	Vrlo laka
23-24	90%	72	Trivijalna
25-26	100%	81	Trivijalna (Rešena)

Parametar	Vrednost	Opis
ILS_CYCLES	200	Ukupan broj Iterativnih Lokalnih Pretraga (petlji).
LS_ITERATIONS_PER_CYCLE	2000	Broj iteracija lokalne pretrage po jednom ILS ciklusu.
ALPHA ( $\alpha$ )	0.995	Faktor opadanja za kontrolu veličine perturbacije.
ACCEPTANCE_PROB	0.15	Verovatnoća prihvatanja lošijeg rešenja.
TABU_SIZE	10	Veličina liste zabranjenih poteza (Tabu lista).
CP_LIMIT	4.0	Vremenski limit (u sekundama) za CP Perturbaciju.
EMPTY_FACTOR_INIT	0.2	Početni faktor pražnjenja ćelija u perturbaciji.

# REZULTATI EKSPERIMENTA

Kategorija	SR (%)	Avg. LS poteza	Avg. Vreme	Ukupni uspesi	Pokretanja
5% Fiksirano	100.00	2000	14.04	4	4
10% Fiksirano	100.00	2000	14.38	4	4
15% Fiksirano	100.00	2000	14.29	4	4
20% Fiksirano	100.00	2000	14.49	4	4
25% Fiksirano	100.00	2000	14.48	4	4
30% Fiksirano	100.00	2000	14.43	4	4
40% Fiksirano	100.00	2000	14.57	4	4
50% Fiksirano	100.00	2000	14.21	4	4
60% Fiksirano	100.00	2000	14.17	4	4
70% Fiksirano	100.00	500	3.54	4	4
80% Fiksirano	100.00	0	0.00	4	4
90% Fiksirano	100.00	0	0.00	4	4
100% Fiksirano	100.00	0	0.00	4	4

Kategorije težine od **5%** do **100%** fiksnih ćelija (gustina)

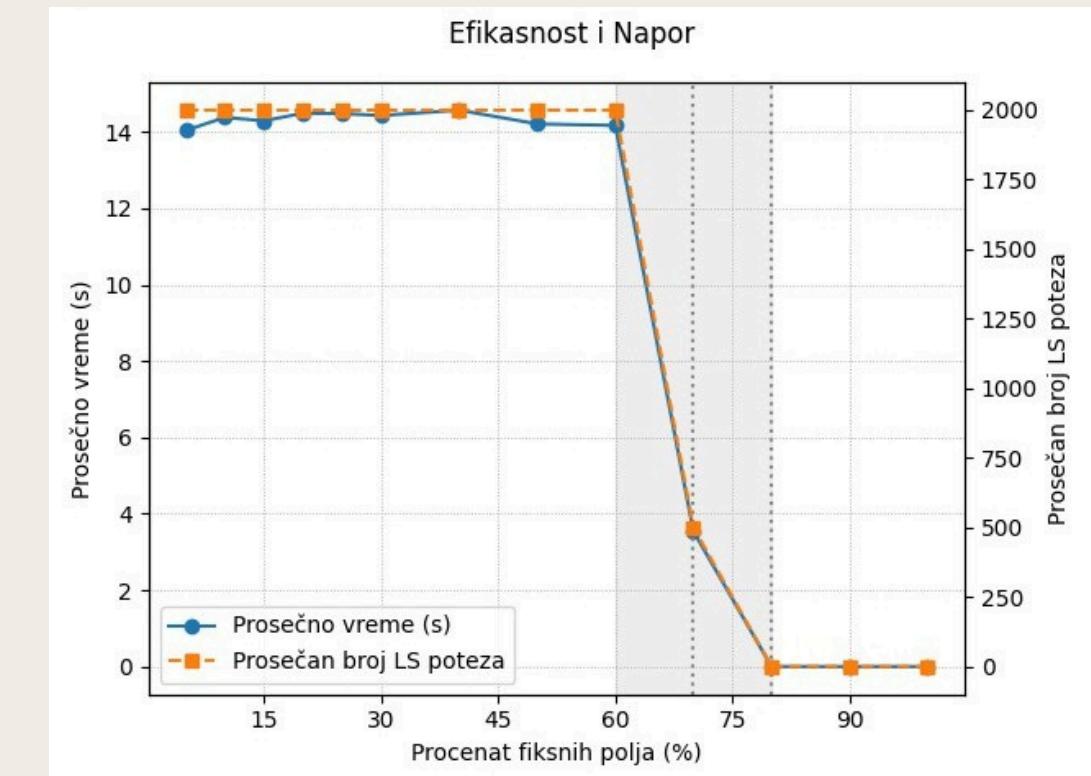
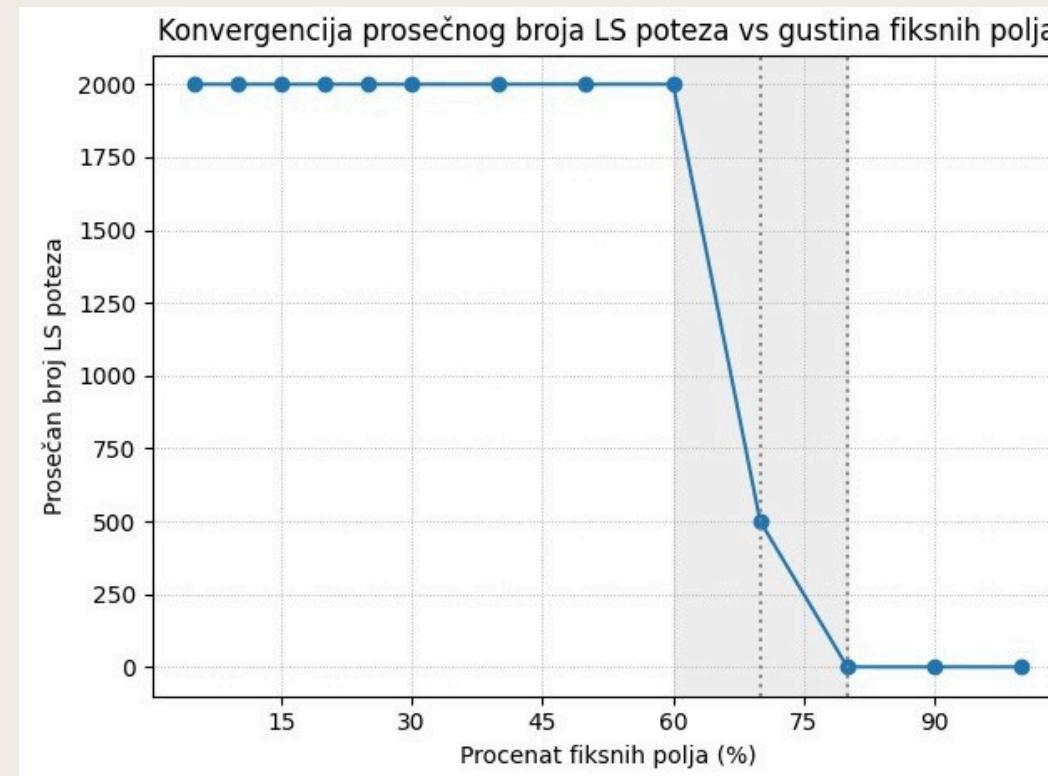
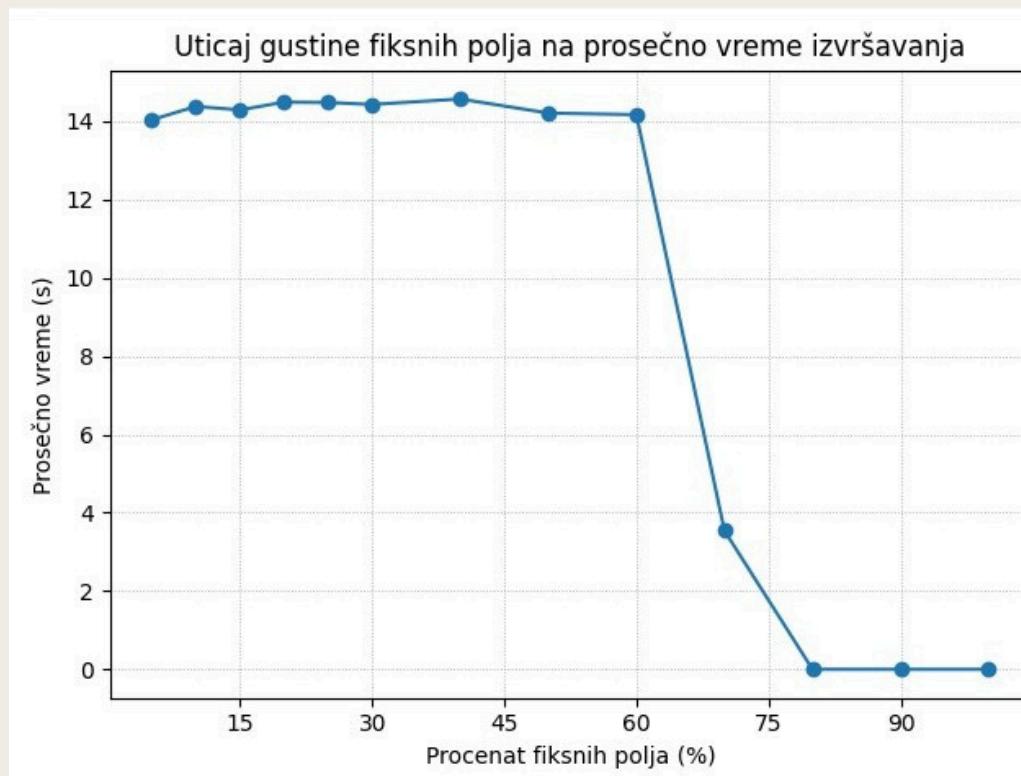
Što je **manji** procenat fiksnih ćelija (npr. 5% - 10%) problem je **teži**

Povećana sloboda izbora - visok rizik od zaglavljivanja u lokalnim minimumima

Za **veći** procenat rešenja su **laka/trivijalna**

Drastično sužen prostor pretrage, smanjena sloboda izbora.

# REZULTATI EKSPERIMENTA



Grafik Efikasnosti i Napora pokazuje gotovo **savršenu konvergenciju** krivih (Vreme i LS Potezi), potvrđujući **direktnu proporcionalnost** između izvršenih operacija i utrošenog CPU vremena

**Kritičan prelaz** se dešava između **60% i 70%** fiksiranih ćelija

**Trijivijalna zona** je na 80% i više ćelija, nulta potrošnja, obe krive padaju na nulu

# POREĐENJE SA REFERENTNIM RADOM

Kriterijum	Ref Rad [1]	Naš eksperiment
Cilj	Statistička evaluacija	Verifikacija implementacije
Opseg	$9 \times 9$	$9 \times 9$
Skup Instanci	1200	26
Kategorizacija	20 kategorija	13 kategorija
Broj Pokretanja	60 instanci	4 instance
Ključne Metrike	SR, Avg.Time	SR, Avg.Time , Avg.LS poteza

Težina (%)	SR (%)	SR Ref (%)	Avg Time(s)	Avg Time Ref (s)	Razlika (%)
30	100	96	0.87	1.12	22.3
35	100	90	1.05	1.45	27.6
40	100	57	1.80	2.30	21.7
Prosek	100	81	1.24	1.62	23.9

Implementirani algoritam, iako testiran na manjem setu podataka, nudi snažan **kvalitativni dokaz** robusnosti i efikasnosti

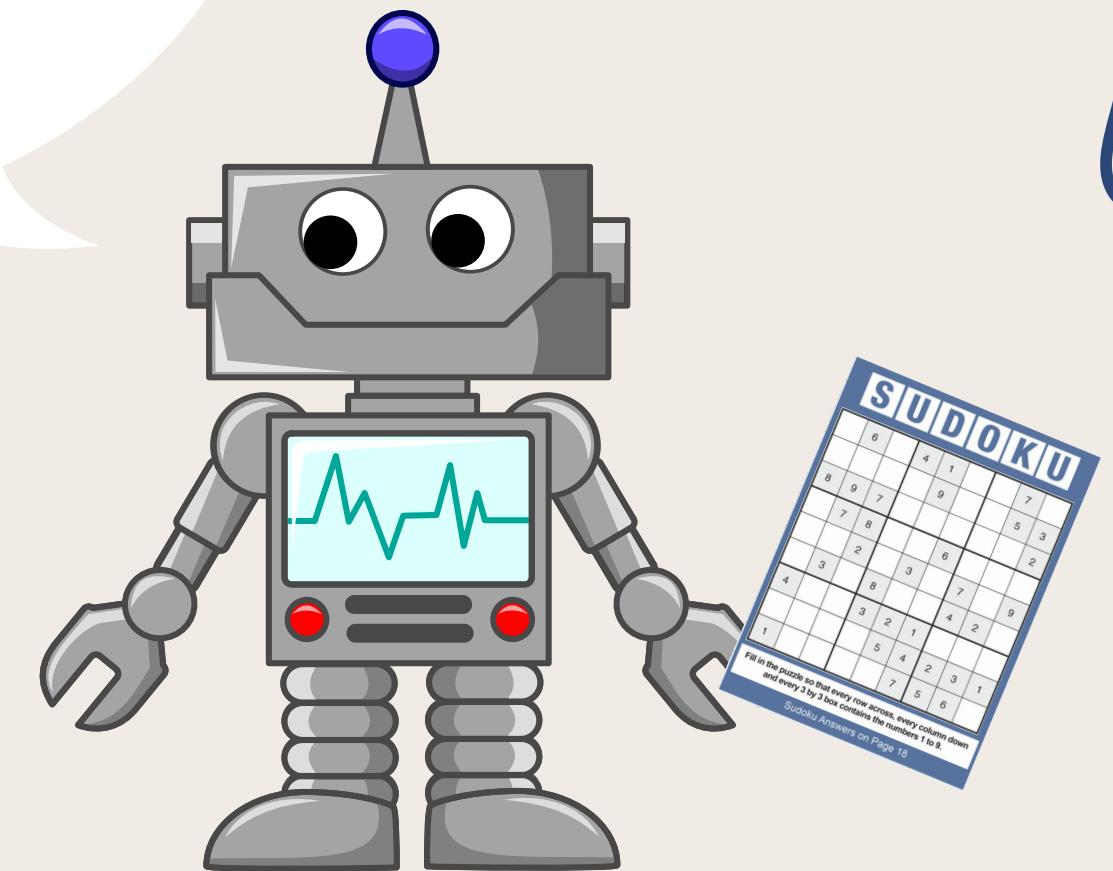
Ostvaruje superiornu **stopu uspešnosti** i kraće **prosečno vreme** na testiraniminstancama

# ZAKLJUČAK

- Uspešno demonstrirana **efikasnost** i **robustnost** hibridnog algoritma
- **SR = 100%** na svim testiranim kategorijama težine (od **5%** do **90%** fiksiranih ćelija)
- **Stabilnost:** Vreme izvršavanja **pravilno konvergira** pokazujući robustnost i predvidiv rad nezavisno od promena u ulaznim podacima
- CP komponenta - Dokazana kao ključna za uspešno izvlačenje pretrage iz lokalnih optimuma
- Smanjen **broj LS poteza** dovodi do drastičnog smanjenja **vremena** izvršavanja
- **Kvalitativna potvrda** poredjenjem sa referentnim radom



HVALA NA  
PAŽNJI



**Marija Sitarica 4003/2024**  
**Mihajlo Savić 4018/2024**