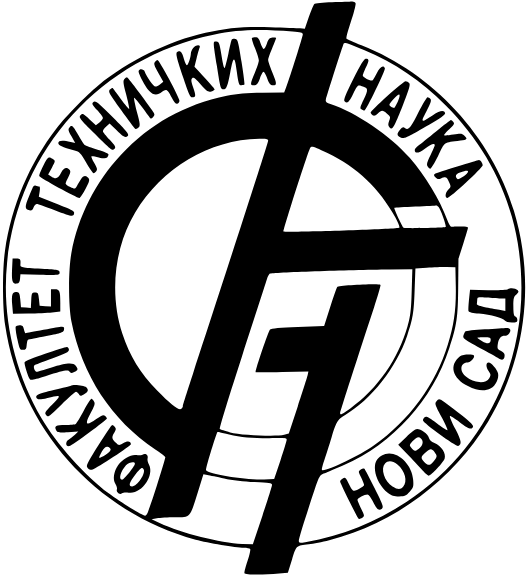
# PROJEKTNI ZADATAK IZ PRIMENJENIH TEORIJA IGARA

*Mladen Blizanac E2 87/2024  
Petar Popov E2 \_/2024*

**

Profesor: dr. Milan Rapaić

[PROJEKTNI ZADATAK IZ PRIMENJENIH TEORIJA IGARA 1](#_Toc211786456)

[**1.** **Koncenzus** 3](#_Toc211786457)

[**2.** **Korišćene tehnologije** 4](#_Toc211786458)

[**3.** **Osnovni koncenzus modeli** 5](#_Toc211786459)

[4. **Implementacija i evaluacija** 8](#_Toc211786460)

[**5.** **Zaključak** 16](#_Toc211786461)

[Ograničenja i prostor za napredak 16](#_Toc211786462)

## **Koncenzus**

Konsenzus problem zahteva da grupa distribuiranih procesa (agenta) postigne dogovor o jednoj vrednosti, čak i kada neki procesi ne odgovaraju ili daju netačne informacije. Konsenzus je paradigma u distribuiranim sistemima i koristi se u mnogim aplikacijama (najviše kao replikacija stanja, baza podataka, blockchain) kojima je potreban dosledan, konzistentan i istinit pogled na podatke. Na primer, u distribuiranim bazama podataka ili distribuiranom računarskom klasteru, konsenzus je potreban da se odredi kojim redosledom će se transakcije izvršavati ili koje stanje sistema će se replicirati. U praksi se često koristi princip **glasovanja po većini** – ako je npr. tri nezavisna računara rešavala isti zadatak, rešenje koje se pojavljuje kod najmanje dva od tri (većina) smatra se ispravnim.

U ovom projektu implementirani su i evaluirani osnovni konsenzus algoritmi u simuliranom multi-agent okruženju. To obuhvata postavljanje mreže agenata u kojoj svaki agent iterativno ažurira svoje stanje na osnovu informacija od susednih agenata, sve dok se ne postigne saglasnost. Klasični (linearan) konsenzus algoritam iterativno računa novu vrednost agenta $i$ kao ponderisanu sumu stanja njegovih suseda, npr. . Primenom adekvatnih težinskih koeficijenata obezbeđuje se konvergencija procena svakog agenta ka zajedničkoj vrednosti.

## **Korišćene tehnologije**

Za razvoj projekta i analizu rezultata u pythonu korišćeni su sledeći alati:

• **Mesa** - agent-bazirana simulacija; upravlja populacijom agenata i njihovim ciklusom *step/advance*.  
• **NetworkX** - generisanje i analiza grafova (*Erdos-Renyi*, *Watts-Strogatz*, *Barabasi-Albert*, *prsten*), kao i Laplasijana.  
• **NumPy** - numeričke operacije, slučajna inicijalizacija vrednosti agenata i dodavanje šuma.  
• **Pandas** - skladištenje rezultata simulacija, analiza i grupisanje po parametrima eksperimenta.  
• **Matplotlib** - vizuelizacija: vremenske serije, konvergencija i uporedni stubičasti grafici.  
• **tqdm** - prikaz trake napretka pri višestrukim eksperimentima (grid-search po parametrima).

## **Osnovni koncenzus modeli**

Projekat uključuje sledeće protokole: **Metropolis** (ponderisano u proseku sa Metropolis-Hastings težinama), **Simple Average** (DeGroot) - klasičan linearni model konsenzusa sa jednakim težinama suseda, i **Gossip** - asinhrono ažuriranje na slučajno izabranom edgu u svakoj iteraciji.

Svi ovi protokoli ciljaju na usaglašavanje skalarne vrednosti svih agenata (npr. senzorska merenja ili bilo koji drugi numericki podatak) pod pretpostavkom potpuno povezane mreže i dovoljno male šum smetnje.

U determinističkoj sinhronoj varijanti (DeGroot/Metropolis), stanje vektora vrednosti u trenutku *t* označimo sa

Ažuriranje je linearno:

gde je stohastična matrica težina zasnovana na susedstvu grafa, a šum (Gaussian noise).

Za **Metropolis** težine, tipično:

**Topologija mreže**

• **Erdos-Renyi**   
• **Prsten** (*cycle graph*)  
• **Watts-Strogatz** (*small-world*) sa parametrima   
• **Barabasi-Albert** (*scale-free*) sa parametrom

Topologija mreže utiče na brzinu konvergencije algoritama konsenzusa.  
Jak teorijski prediktor brzine konvergencije je **spektralni zazor** (druga najmanja sopstvena vrednost Laplasijana grafa), označena kao

Formalno, ako je , Laplasijan grafa, definisan kao

gde je dijagonalna matrica stepena čvorova, a matrica susedstva,  
onda su sopstvene vrednosti Laplasijana uređene kao

Vrednost (poznata i kao **algebraic connectivity**) meri povezanost grafa:  
što je veća , to je mreža bolje povezana i tipično brže dostiže konsenzus.  
Projekat implementira merenje ove veličine iz sopstvenih vrednosti Laplasijana:

**Nepouzdanosti i zli agenti**

• **Packet loss** - svaki prenos se može odbaciti sa verovatnoćom ,  
što znači da agent u tom slučaju zadržava svoju staru vrednost (ne prima informaciju od suseda).

• **Byzantine agenti** - mogu slati tendenciozne, lažne ili ekstremne vrednosti ka susedima.  
U projektu je implementirana jednostavna politika:  
ako je agent Byzantine, njegova izveštajna vrednost je fiksna velika konstanta (npr. 10.0).

Prisustvo byzantine agenata ruši pretpostavke linearne konvergencije i može onemogućiti postizanje istinskog konsenzusa.  
Standardne robusne šeme, poput **W-MSR** (Weighted-Mean-Subsequence-Reduced) algoritma, oslanjaju se na **trimovanje ekstremnih vrednosti** – svaka jedinka odbacuje do najvećih i najmanjih vrednosti iz svog lokalnog skupa pre nego što izračuna prosek.

U ovom projektu je implementirana **naivna** byzantine politika samo radi poređenja…

## **Implementacija i evaluacija**

**src/consensus/model.py**

Klasa **ConsensusModel** postavlja strukturu grafa, kreira agente (po jedan za svaki čvor mreže), računa matricu težina **W** (za protokole *Metropolis* i *Simple Average*),  
i implementira različite režime ažuriranja vrednosti agenata.

* **Metropolis / Simple Average** - sinhrono *DeGroot* ažuriranje, gde svaki agent računa:

uz opciono dodavanje Gaussovog šuma .

* **Gossip** - u svakoj iteraciji bira se nasumični edge/ivica ; oba kraja ivice ažuriraju vrednosti sa korakom :

Ostali čvorovi ostaju nepromenjeni.

Uvedeni su parametri:

* **noise\_std** () - standardna devijacija šuma,
* **p\_drop** - verovatnoća odbacivanja poruke (*simulacija gubitka paketa*).

Za vizantijske/lazne agente, metoda byzantine\_value vraća ekstremnu vrednost (npr. **10.0**) koja kvari proseke suseda i usporava ili onemogućava konvergenciju.

**src/consensus/agent.py**

Klasa **ConsensusAgent** čuva sledeće atribute:

* *value* - trenutna vrednost agenta,
* *next\_value* - vrednost nakon sledećeg koraka,
* oznake *is\_stubborn* i *is\_byzantine* koje definišu tip ponašanja.

U sinhronom režimu agent računa **ponderisani zbir preko svih suseda** koristeći matricu težina **W** iz modela:

a zatim poziva *advance()* da commit-uje novu vrednost.  
U asinhronom režimu (gossip) ažuriranje koordinira sama klasa modela.

**src/utils/graph\_utils.py**

Funkcija **create\_graph** generiše graf prema zadatim parametrima:

* *Erdos-Renyi (G(n,p))*,
* *Watts-Strogatz (k,p)*,
* *Barabasi-Albert (m)*,
* ili *prsten (ring)* topologiju.

Funkcija **spectral\_gap(G)** računa **spektralni zazor** grafa - drugu najmanju sopstvenu vrednost Laplasijana, označenu kao :

Ova veličina predstavlja **algebrajsku povezanost grafa** i dobar je prediktor brzine konvergencije algoritama konsenzusa.

**src/experiments/runner.py**

Funkcija **single\_run(config)**:

* instancira model prema konfiguraciji (N, graph\_type, protocol, alpha, noise\_std, p\_drop, seed, max\_steps, tol),
* pokreće simulaciju do **konvergencije** ili do zadatog broja koraka,
* vraća rezultate i metrike:
  + broj koraka do konsenzusa,
  + vrednost ,
  + protokol, topologiju, seed, vreme izvršavanja, itd.

Funkcija **sweep\_and\_save(output\_csv, param\_grid, repeats)** iterira kroz više konfiguracija i semena (repeticija), pokreće single\_run više puta, sabira rezultate u CSV fajl, i vraća DataFrame.

**src/viz/plotting.py i src/viz/plots.py**

Ovi moduli sadrže funkcije za vizualizaciju rezultata simulacija:

* **Vremenske serije** - prikazuju srednju vrednost i opseg () kroz vreme,
* **Grafici** - prikazuju prosečan broj koraka do konvergencije (sa standardnom devijacijom) grupisano po topologiji ili protokolu.

Fajlovi sa vizualizacijama se automatski čuvaju u folderu *visualizations*

**Pokretanje**

Glavni ulaz u projekat je fajl *run\_experiment.py* (u root folderu projekta).  
On definiše **param\_grid** - kombinacije topologija i protokola, uz tipične parametre poput:

* *N* = 50
* *max\_steps*
* *tol*
* *noise\_std* = 0.0
* *p\_drop* = 0.0
* različite vrednosti parametra protocol (*metropolis*, *simple\_avg*, *gossip*).

Skripta poziva funkciju *sweep\_and\_save()* sa više ponavljanja (repeats) i belezii rezultate u *data/experiment\_baseline.csv* fajlu, ne bi li posle generisala grafike unutar foldera *visualizations*

Pokretanje projekta nije komplikovano. Prvo je potrebno instalirati sve potrebne bilbioteke i pakete, a koji se nalaze unutar fajla *requirements.txt.* To se može odraditi korišćenjem terminal komande pip install -r requirements.txt pod uslovom da se lokalno nalazi instaliran pip paket. Zatim se pokrece glavni fajl komandom python run\_experiment.py ili preko ODE user Interface-a (u ovom slučaju Visual Studio Code-a, Run→Run Without Debugging). Posle pokretanja, pojavljuje se bar u terminal koji nam govori porcentualno koliki je deo procesa završen. Završetkom procesa stvaraju se ili ažuriraju stari grafici i vrednosti unutar foldera data i visualizations.

**Definicija konvergencije i metrike**

Konvergencija u kodu je definisana preko opsega vrednostiagenata ili norme promene stanja između iteracija.

* Definicija po opsegu:
* Alternativna definicija po normi:

Parametar tol (tolerancija) određuje prag ispod kog se sistem smatra konvergentnim.

Iz rezultata se čuvaju sledeće osnovne metrike:

* *convergence\_step* - broj iteracija do dostizanja praga konvergencije,
* *graph\_type, protocol, N, seed* - deskriptivni parametri eksperimenta,
* *spectral\_gap* - spektralni zazor grafa, tj. druga najmanja sopstvena vrednost Laplasijana:

**Evaluacija rezultata**

**(a) Topologija:**  
Grafovi sa većim *spektralnim zazorom* (npr. *Erdos-Renyi* sa većim , *Watts-Strogatz* sa većim ili ) obično konvergiraju brže.  
C*ycle graph* ima najmanji zazor i najsporiji konsenzus.  
*Barabasi-Albert* mreže često konvergiraju brže od prstena, ali sporije od gustih *ER* mreža.

**(b) Protokol:**  
Sinhroni protokoli (*Metropolis*, *DeGroot*) uglavnom konvergiraju brže po iteraciji nego asinhroni *gossip* protokol,  
ali *gossip* ima manji komunikacioni trošak po koraku jer u svakoj iteraciji ažurira samo dve jedinke (jednu ivicu).

**(c) Šum i gubici:**  
Veće vrednosti parametara *noise\_std* i *p\_drop* usporavaju konvergenciju i mogu izazvati stohastično oscilovanje oko konsenzusne vrednosti.  
To se vidi kao zadržavanje oscilacija u grafikonima opsega ili srednje vrednosti.

**(d) Byzantine agenti:**  
Čak i mali udeo byzantine agenata sa ekstremnim vrednostima (npr. konstanta 10.0) može sprečiti postizanje pravog konsenzusa ili uzrokovati dosta nepreciznu tačku stabilizacije.  
U ovom projektu je implementirana jednostavna, nesažeta byzantine politika bez robusnih zaštita, što omogućava da se efekat lako uoči na histogramima i vremenskim serijama.

A graph of a graph showing a green and blue bar

AI-generated content may be incorrect.

*Konvergiranje po topologijama*

*A graph with a blue line

AI-generated content may be incorrect.*

*Smanjenje greske vremenom, L2*

*A graph with colored dots and numbers

AI-generated content may be incorrect.*

*Spektralni zazor (spectral gap) i brzina konvergencije*

*A graph of a graph showing different colored squares

AI-generated content may be incorrect.*

*Vreme izvršavanja po topologiji i protokolu*

## **Zaključak**

### Ograničenja i prostor za napredak

• Uvesti dodatni robusni protokol **W-MSR** (*Weighted-Mean Subsequence Reduced*) sa parametrom (maksimalan broj byzantine agenata).  
Ovaj protokol koristi trimovanje ekstremnih vrednosti i omogućava robusni konsenzus čak i u prisustvu byzantine agenata.  
Poređenje sa postojećim protokolima (*Metropolis*, *DeGroot*, *Gossip*)

• Modelovanje *tvrdoglavih* **(**stubborn**)** agenata koji zadržavaju sopstvenu informaciju i samo delimično ažuriraju vrednosti prema susedima (možda sa nekim alpha članom tromosti kao kod SARSA ili Q Learninga). U kodu već postoji oznaka *is\_stubborn*, ali je potrebno dovršiti logiku za kontrolisanu fuziju podataka i ograničen uticaj.

• Kalibracija parametra alpha u *Gossip* protokolu - istražiti adaptivni izbor koraka alpha u zavisnosti od stepena čvora ili broja komunikacionih iteracija.  
Dodatno, može se razmotriti prioritetni izbor edge (npr. one koje povezuju vrhove visokog stepena) radi ubrzanja difuzije informacija.

• Sprovesti eksperimente sa heterogenim šumom, dinamičnim grafovima i kašnjenjima komunikacije,  
čime bi se približila simulacija stvarnim mrežnim uslovima.

• Proširiti teorijsku analizu: istražiti vezu između brzine konvergencije i izraza

gde je matrica prelaznih verovatnoća ili težina komunikacije među čvorovima.

Projekat uspešno demonstrira uticaj topologije mreže, izbora protokola i prisustva  
šuma, gubitaka poruka i byzantine agenata na dinamiku procesa konsenzusa.

Protokoli *Metropolis* i *DeGroot* konvergiraju pouzdano na povezanim grafovima bez vizantijskih agenata, a brzina konvergencije u velikoj meri zavisi od *spektralnog zazora* grafa, tj. od vrednosti .

Sa druge strane, *Gossip* protokol, iako sporiji po ukupnom broju iteracija, ima znatno manji trošak komunikacije po koraku i pogodan je za distribuirane i asinhone sisteme.

Za okruženja sa vizantijskim agentima ili gubicima komunikacije, potrebno je uvesti robusne protokole poput *W-MSR*, koji omogućavaju stabilnu konvergenciju i u prisustvu ekstremnih vrednosti.

Zahvaljujući svojoj modularnoj i preglednoj strukturi, ispitivanje ovih osnovnih konvenzusa predstavlja dobar korak za dalji razvoj distribuiranih konsenzus algoritama.