Metaheuristics for Tetris – (using GA)

Autori:

- Filip Dramicanin 303/2023
- Vuk Vukmirovic 305/2023

1. Uvod

Ovaj projekat implementira **genetski algoritam (GA)** za igranje Tetrisa. Cilj je optimizacija parametara heuristike koji odlučuju kako će veštačka inteligencija rotirati i pozicionirati figure, sa krajnjim ciljem da se postigne što duže trajanje igre i osvoji što veći broj poena.

Za vizuelizaciju igre koristi se **Pygame**, koji omogućava grafički prikaz stanja tetris table i napredovanja algoritma. Rezultati treninga se čuvaju u **log fajlovima** (npr. results.csv, run_meta.json), a zatim analiziraju i grafički prikazuju pomoću Matplotlib i Pandas biblioteka.

Projekat je organizovan u više modula:

- **game/** – sadrži glavni kod, uključujući main.py za pokretanje treninga i gui2.py za vizuelizaciju igre.
- **ga/** – implementacija genetskog algoritma (inicijalizacija, selekcija, ukrštanje, mutacija, petlja evolucije). Ključni fajlovi su loop.py, selection.py, mutation.py i crossover.py.
- **tetris/** – implementacija osnovne logike same igre Tetris (simulacija padanja i postavljanja figura, detekcija linija, računanje poena).
- **plots/** – sadrži generisane grafike koji vizuelno prikazuju napredovanje algoritma, upoređuju metode selekcije i prikazuju evoluciju fitness-a.
- **config.py** – definisanje svih parametara genetskog algoritma (populacija, broj generacija, verovatnoća mutacije, tip selekcije itd.).

2. Opis algoritma

Genetski algoritam (GA) predstavlja metaheurističku tehniku inspirisanu procesom prirodne evolucije. U ovom projektu koristi se za optimizaciju parametara heuristike koja vodi odluke AI-ja u Tetrisu.

Osnovni koraci GA:

1. **Inicijalizacija populacije** – na početku se kreira populacija nasumičnih kandidata (genoma). Svaki genom predstavlja niz realnih brojeva koji definišu težine različitih heurističkih funkcija (npr. visina tornja, broj rupa, broj kompletiranih linija).
2. **Evaluacija (fitness funkcija)** – svaki genom se testira tako što odigra simulaciju igre. Kao fitness vrednost koristi se skor koji je postignut u toj partiji (broj oborenih linija i osvojenih poena).
3. **Selekcija** – biraju se najbolji kandidati koji će učestvovati u kreiranju nove generacije. U projektu su implementirane dve metode selekcije:
    - **Tournament selection** – nasumično se bira podskup populacije, a najbolji iz njega ide dalje.
    - **Roulette wheel selection** – verovatnoća izbora kandidata proporcionalna je njegovom fitness-u.
4. **Ukrštanje (crossover)** – dva roditelja se kombinuju kako bi se stvorila nova jedinka. U projektu se koristi jednostavan **single-point crossover**.
5. **Mutacija** – genomi potomaka se nasumično menjaju sa određenom verovatnoćom. Ovo omogućava dodatnu raznolikost i smanjuje rizik od prerane konvergencije.
6. **Elitizam** – najbolji genomi iz prethodne generacije se direktno prenose u novu, kako bi se osiguralo da najbolja rešenja ne budu izgubljena.

7. **Zaustavni kriterijum** – algoritam se izvršava unapred određen broj generacija ili dok se ne dostigne stagnacija (npr. nema poboljšanja tokom određenog broja iteracija).

Parametri genoma:

alpha[0] – rupe (holes)

- Broj praznih polja ispod popunjenih ćelija u kolonama.

alpha[1] – maksimalna visina (max height)

- Najviša popunjena ćelija na tabli, normalizovana po visini.

alpha[2] – neravnine (bumpiness)

- Zbir razlika u visinama između susednih kolona.

alpha[3] – kompletirane linije (lines)

- Bonus na osnovu broja linija oborenih u tom potezu (0, 1, 3, 5, 8).

alpha[4] – bunari (wells)

- Dubine uvučenih kolona između zidova ili blokova.

alpha[5] – kolone tranzicije (column transitions)

- Broj prelaza 0↔1 po kolonama (prazno/popunjeno).

Fitness funkcija kombinuje ove parametre u linearnu kombinaciju:

*fitness = -(a0*holes_n + a1*max_h_n + a2*bump_n) + a3*line_n + a4*wells_n - a5*trans_n*

3. Implementacija

Struktura projekta organizovana je u nekoliko foldera i fajlova:
- game/: sadrži glavni kod i GUI implementaciju
- ga/: implementacija genetskog algoritma (selekcija, mutacija, loop...)
- tetris/: logika simulacije Tetrisa
- plots/: vizuelizacija rezultata

3.1 game/ — ulazna tačka za trening i orkestracija

- **main.py**
    - Ulaz u program: učita konfiguraciju iz config.GAConfig, postavi seed (ga.utils.set_seed) i pokrene evolucioni loop (ga.loop.run_ga).
    - Zadaje parametre GA: pop_size, genome_size, gens, elitism, mutation_rate, tip selekcije (preko ga.selection.make_selector), stagnation_patience, immigrant_fraction.
    - Snima snapshotove najboljeg rešenja:
        - best_ever.json – globalno najbolje tokom celog treninga (gen, fitness, genome).
        - best_of_last_gen.json – najbolje u poslednjoj generaciji.
    - Po završetku, pokreće kratku simulaciju najboljeg genoma da generiše **sheets.txt** (frejmove za replay u GUI-u).
- (Pomoćno) **config.py** (na rootu repo-a, ali logički spada uz game):

- Sadrži dataclass GAConfig: brojevi generacija, veličina populacije/genoma, stope mutacije, izbor selekcije, seed, kao i write_frames_steps i log_file.

3.2 ga/ — jezgro genetskog algoritma

- **loop.py**
  - Funkcija **run_ga()**: glavna evolutivna petlja.
    - Inicijalizuje populaciju (Individual), evaluira ih pozivom na simulaciju (prosleđuješ tetris.sim.simulate_game).
    - Sortira po fitness-u, primenjuje **elitizam**, pravi **nove jedinke** ukrštanjem i mutacijom, re-evaluiše.
    - Prati **best_this_gen**, **best_overall**, **stagnation**; ubrizgava imigrante ako stagnira (immigrant_fraction).
    - Čuva **snapshot** najboljeg: best_snapshot.json (opciono sa sekvencom figura za reprodukciju).
    - Vraća (best_individual, fitness_history) i **ispisuje log po generaciji** (best/mean/genome).
- **individual.py**
  - Klasa **Individual**:
    - Drži **genome** (lista dužine genome_size), **fitness**.
    - evaluate(simulate_game): pokreće simulaciju i upisuje fitness.
    - clone() za duboku kopiju najboljih jedinki (elitizam, poređenje).
- **selection.py**
  - **make_selector(kind, k=None, ...)**: vrati funkciju selekcije u zavisnosti od vrednosti iz konfiguracije.
  - Implementirano:
    - **Tournament selection** (sa k): nasumično uzorkuje k jedinki i vraća najbolju.
    - **Roulette wheel**: verovatnoća $\propto$ fitness.
  - lako proširivo (dodaj npr. boltzmann, rank, SUS…).
- **mutation.py**

- **mutate(individual, rate)**: prolazi kroz genome i za svaku komponentu sa verovatnoćom ‘rate’ pravi malu promenu

- **crossover.py**
  - **single_point(p1, p2)**: klasičan 1-tačka crossover – vraća dvoje dece spajanjem delova genoma.
- **utils.py**
  - **set_seed(seed)**: deterministočnost (Python/NumPy random).

3.3 tetris/ — simulacija igre i heuristika/fitness

- **sim.py** (najbitniji deo za performanse GA jer definiše evaluaciju):
  - **simulate_game(alpha, max_steps=None, write_frames=False, piece_sequence=None, capture=False, ...)**
    - Prima **genome alpha** (6 težina), pokreće partiju u headless modu.
    - Ako je max_steps zadat, ograniči broj koraka; ako je piece_sequence prosleđen, koristi determinističku sekvencu figura (za reproducibilnost najboljeg).
    - Ako je write_frames=True, upisuje frejmove (stanja tabele + “NEXT: …” + score) u **sheets.txt** (GUI kasnije prikazuje).
    - Ako je capture=True, vraća i meta-info (npr. pieces_used) da se snimi u snapshot.
    - Vraća **fitness** (skor) – to koristi GA.
  - **calculate_fitness(field, alpha)**
    - Kombinuje **6 normalizovanih feature-a** stanja table sa težinama alpha (vidi “Opis algoritma”):
      - rupe (holes_n), max visina (max_h_n), neravnine (bump_n), linije (line_n), bunari (wells_n), kolone tranzicije (trans_n).
    - Formula:
      fitness = -(a0*holes_n + a1*max_h_n + a2*bump_n) + a3*line_n + a4*wells_n - a5*trans_n
  - **Pomoćne funkcije** (primeri koje si naveo):
    - **_well_sum(field)** – računa zbir dubina bunara po kolonama (uz cap).

        - **_column_transitions(field)** – broji 0↔1 prelaze po kolonama.
        - (Tipično postoje i helperi za izračunavanje visina kolona, bumpiness, pronalazak “rupa”, detekciju kompletiranih linija, generisanje i rotacije tetromina, validaciju pozicije i sl.)
    - Interno, simulate_game na svakom potezu iterira kroz **sve moguće (kolona × rotacija)** pozicije trenutne figure, ocenjuje poziciju preko calculate_fitness, bira najbolju i spušta figuru → ažurira tablu → nastavlja sa sledećom figurom dok se ne izgubi ili ne istekne max_steps.

3.4 plots/ — analiza rezultata i grafici

- **Ulazni fajlovi (logovi)**:
    - **Tournament_selection.txt / roullete_selection.txt** sirovi logovi po generacijama; linije poput:
      Gen k: best_this_gen = ..., mean_this_gen = ..., genome = [...]
- **Skripte:**
    - Parser koji iz logova izvlači nizove: gen, best, mean, best_ever, genome.
    - Plotovi (matplotlib):
        - **Linijski grafici**: evolucija best_this_gen i mean_this_gen kroz generacije.
        - **Upoređivanje selekcija**: tournament (plavo) vs roulette (crveno) za 25/50/100 gen
        - **Distribucije**: histogram best_this_gen vrednosti
        - **Bar-plot**: poređenje “best-ever” između selekcija.
        - **Korelaciona mapa**

3.5 Pomoćni fajlovi (na rootu)

- **best_ever.json** – najbolji rezultat tokom celog treninga (gen, fitness, genome, opcionalno label).
- **best_of_last_gen.json** – najbolje rešenje iz poslednje generacije (korisno ako želiš baš “poslednje stanje”).
- **run_meta.json** – “tekući” meta-log (generacija/fintess/genome) koji može da čita GUI/plot skripta.
- **sheets.txt** – frejmovi odigrane partije (matrice stanja + NEXT: + score) za vizuelni replay u GUI-u.

4. GUI

Grafički deo projekta realizovan je pomoću **pygame** biblioteke i služi za vizuelni prikaz partije koju je odigrao genetski algoritam. GUI čita frejmove iz fajla sheets.txt, koji nastaje kada se pozove simulacija sa uključenim opcijom write_frames=True. Na osnovu tih frejmova se na ekranu prikazuje tabla dimenzija 10×20, sa obojenim ćelijama za postavljene blokove, a poslednja ubačena figura se dodatno ističe crvenim okvirom.

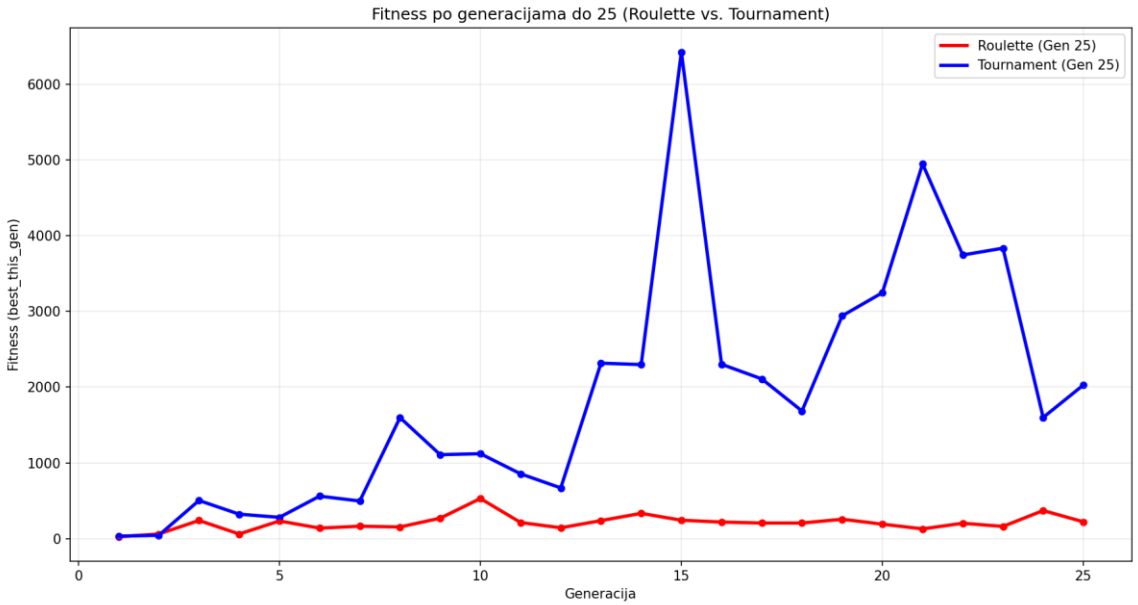
Sa desne strane nalazi se panel na kojem se prikazuje sledeća figura (preview box), trenutni skor, kao i dugme **Stop** kojim se može prekinuti reprodukcija. Na vrhu ekrana je header u kome su prikazani osnovni podaci o treningu – broj generacije, vrednost fitness-a i vrednosti genoma, preuzeti iz fajlova run_meta.json ili best_ever.json.

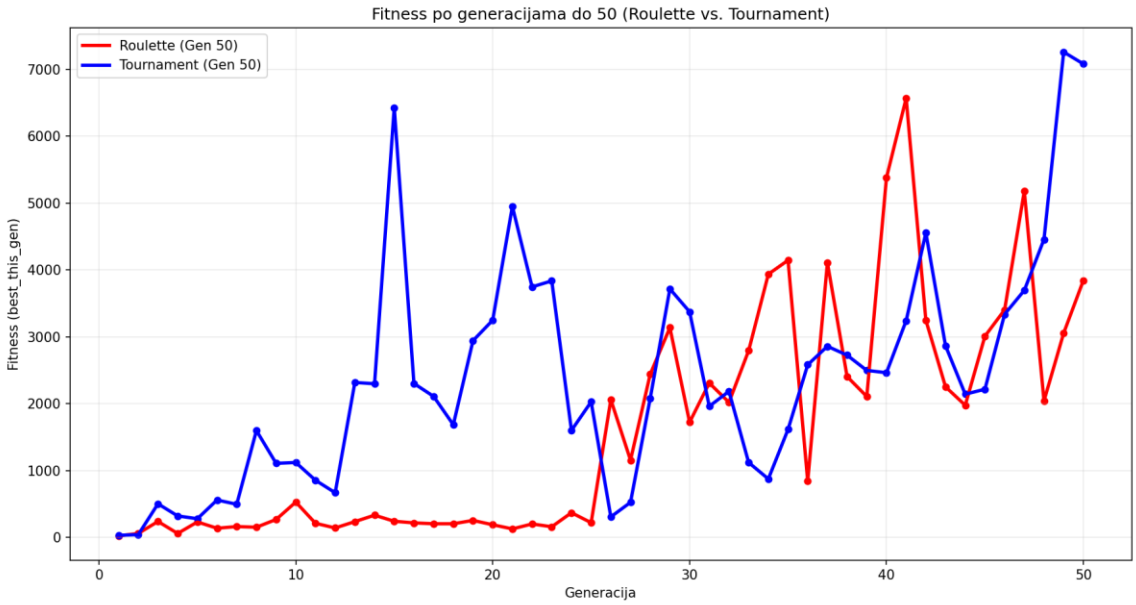
Tokom rada GUI prolazi kroz sve linije fajla sheets.txt i ažurira ekran u realnom vremenu, sa pauzom između frejmova da bi animacija bila pregledna. Na ovaj način moguće je vizuelno pratiti kako algoritam donosi odluke, kako se tabla puni i brišu linije, kao i koliko je uspešan dati genom.

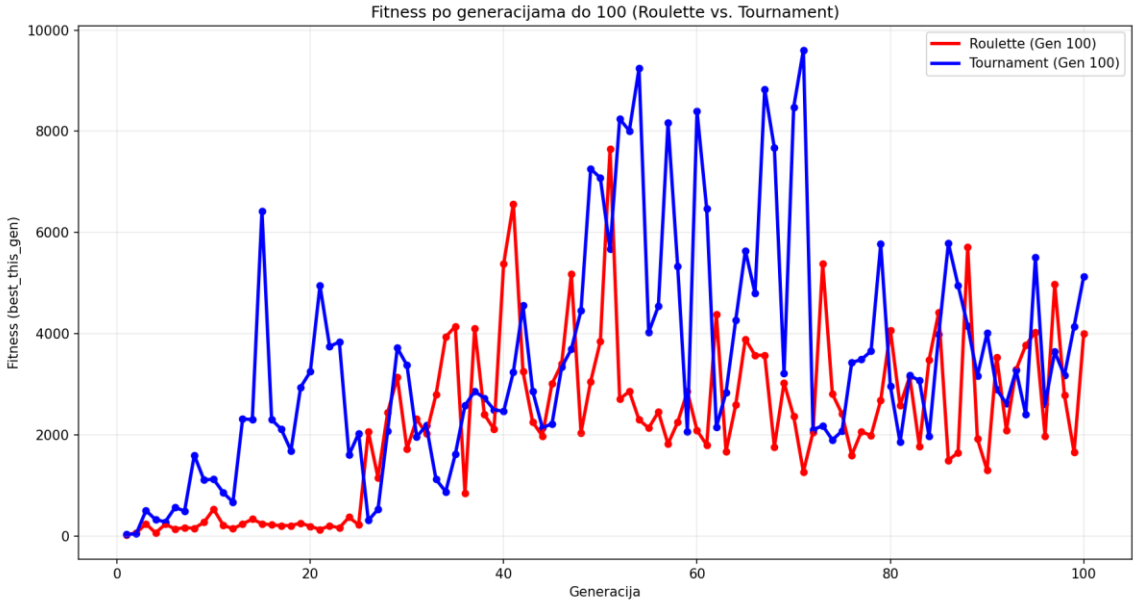
Za potrebe dokumentacije dovoljno je ubaciti screenshot GUI prozora, gde se vidi tabla, desni panel sa skorom i preview-om sledeće figure, i gornji header sa meta informacijama.

4. Rezultati

Eksperimenti su rađeni za 25, 50, i 100 generacija sa različitim metodama selekcije (tournament i roulette). Rezultati pokazuju da je tournament selekcija u proseku davala stabilnije i više vrednosti fitness funkcije, dok je roulette selekcija pokazivala veće varijacije.
Grafici evolucije fitness-a i poređenja selekcija nalaze se u folderu plots/




Fitness po generacijama do 50 (Roulette vs. Tournament)
Roulette (Gen 50)
Tournament (Gen 50)
Fitness (best_this_gen)
Generacija
0
1000
2000
3000
4000
5000
6000
7000
0
10
20
30
40
50


Fitness po generacijama do 100 (Roulette vs. Tournament)
Roulette (Gen 100)
Tournament (Gen 100)
Fitness (best_this_gen)
Generacija
0
2000
4000
6000
8000
10000
0
20
40
60
80
100

5. Zaključak

Projekat pokazuje kako se genetski algoritam može primeniti na kompleksne probleme kao što je igranje Tetrisa. Definisani su heuristički parametri koji mere kvalitet pozicije na tabli, a GA je korišćen da optimizuje njihove težine kroz selekciju, ukrštanje, mutaciju i elitizam. Na ovaj način AI uči da donosi bolje odluke pri rotaciji i postavljanju figura, čime se povećava skor i produžava trajanje igre.

Implementacija je modularna – deo za simulaciju igre, deo za sam algoritam, deo za analizu rezultata i deo za vizuelizaciju. To omogućava jednostavno proširivanje i eksperimentisanje sa različitim konfiguracijama, novim selekcionim metodama ili dodatnim heuristikama. Grafički prikaz igre kroz GUI daje jasan uvid u ponašanje algoritma, dok rezultati i generisani grafici pokazuju napredak tokom generacija i razlike između metoda selekcije.

Potencijalna proširenja uključuju dodavanje drugih optimizacionih tehnika (npr. simulirano kaljenje, PSO), testiranje većih populacija i broja generacija, kao i dodatno prilagođavanje heuristika koje bolje hvataju kompleksnost Tetrisa.

6. Reference

- Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
- Pygame Documentation – https://www.pygame.org/docs/
- Matplotlib Documentation – https://matplotlib.org/stable/contents.html
- Pandas Documentation – https://pandas.pydata.org/docs/
- Tetris Wiki (heuristike i AI pristupi) – https://tetris.fandom.com/wiki/Tetris_Wiki
- GitHub – Open-source implementacije Tetris AI projekata za poređenje i inspiraciju.