Rectilinear Picture Compression

1. Uvod

1.1 Opis problema

Cilj projekta je implementacija algoritama za kompresiju slike metodom pravougaonika (Rectilinear Picture Compression). Slika se predstavlja kao matrica piksela, a svaki pravougaonik obuhvata grupu piksela slične boje. Na taj način se slika opisuje pomoću manjeg broja pravougaonika, što smanjuje količinu memorije potrebne za njeno skladištenje.

1.2 Motivacija

Standardni formati kompresije (kao JPEG) koriste frekvencijsku analizu, dok se ovde istražuje prostorni pristup zasnovan na segmentaciji slike u pravougaonike. Ova metoda je posebno zanimljiva jer se lako može interpretirati u kontekstu optimizacije i kombinatornih algoritama.

1.3 Cilj rada

Razviti i uporediti tri heuristička algoritma za kompresiju slike: lokalnu pretragu, simulirano kaljenje i RVNS (Random Variable Neighborhood Search). Cilj je pronaći kompromis između tačnosti rekonstrukcije (izražene pomoću MSE) i stvarne kompresije podataka.

---

2. Opis rešenja

2.1 Princip algoritama

- **Lokalna pretraga**: iterativno spaja susedne pravougaonike ako se time poboljšava funkcija cilja (smanjuje MSE). Kada više nema poboljšanja, algoritam se zaustavlja.
- **Simulirano kaljenje (Simulated Annealing)**: koristi nasumične poteze i temperaturni parametar koji kontroliše verovatnoću prihvatanja lošijih rešenja, kako bi se izbegli lokalni minimumi.
- **RVNS (Random Variable Neighborhood Search)**: menja veličinu okoline pretrage (parametar k) i pokušava spajanja na različitim nivoima detalja dok se ne pronađe poboljšanje.

2.2 Implementacija

Slika se učitava kao matrica boja (RGB vrednosti) dimenzija $H \times W \times 3$. Početna aproksimacija slike kreira se podelom slike na blokove veličine $block\_size \times block\_size$. Za svaki blok računa se prosečna boja:

```
def init3(colored_matrix, block_size=4):
    rectangles = []
    H, W, _ = colored_matrix.shape
    for i in range(0, H, block_size):
        for j in range(0, W, block_size):
            h = min(block_size, H - i)
            w = min(block_size, W - j)
            v = tuple(np.mean(colored_matrix[i:i+h, j:j+w].reshape(-1, 3), axis=0))
            rectangles.append([i, j, h, w, v])
```

```
    return rectangles, len(rectangles)
```

Algoritmi zatim prolaze kroz listu pravougaonika i odlučuju koje blokove mogu da spoje, u zavisnosti od sličnosti boja i prostorne blizine.

2.3 Funkcija cilja

Za svako rešenje računa se **srednja kvadratna greška (MSE)**:

$$MSE = \frac{1}{(H \times W)} \sum_{i}, j\left(I_{ij} - R_{ij}\right)^2$$

gde je (I) originalna, a (R) rekonstruisana slika.

---

3. Eksperimentalni rezultati

3.1 Opis eksperimenta

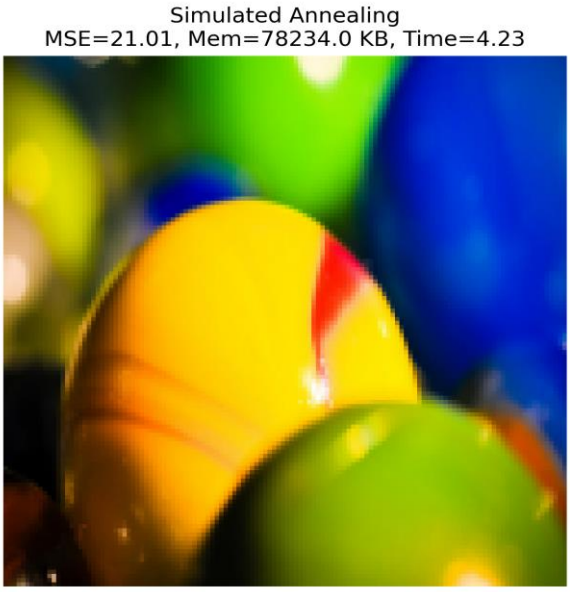
Testiranja su izvršena na slici rezolucije 256×256 piksela. Primenjeni su algoritmi simuliranog kaljenja i RVNS. Upoređeni su po vremenu izvršavanja, tačnosti (MSE) i stepenu stvarne kompresije (compression ratio) nakon binarnog zapisa.

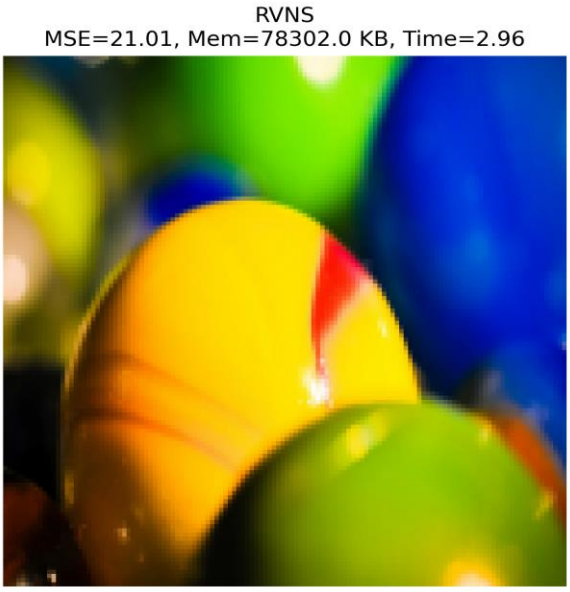
3.2 Poređenje algoritama

Algoritam	MSE	Vreme (s)	Original (KB)	Kompresovana (KB)	Compression ratio
Simulirano kaljenje	21.01	4.23	192.0	76.4	2.51x
RVNS	21.01	2.96	192.0	76.5	2.51x

3.3 Vizuelni rezultati

Slike ispod prikazuju rezultat kompresije korišćenjem oba algoritma:





Simulirano kaljenje generiše grublju aproksimaciju slike (manje pravougaonika), dok RVNS postiže sličan MSE ali kraće vreme izvršavanja. Kod obe metode, kompresija slike u binarnom formatu sa `struct` i `gzip` zapisom daje približno 2.5× smanjenje veličine fajla.

3.4 Kompresioni odnos i binarno kodiranje izlaza

Da bi se postigla stvarna kompresija, pravougaonici se binarno pakuju u fajl pomoću `struct` modula i gzip kompresije. Svaki pravougaonik se čuva pomoću 11 bajtova: 4 koordinatne vrednosti po 2 bajta (uint16) i tri komponente boje po 1 bajt (uint8). Time se izbegava Python overhead i značajno smanjuje veličina izlaza.

Ovaj pristup omogućava realno poređenje sa veličinom originalne slike i daje pokazatelj koliko heuristička kompresija uspeva da smanji memorijsko zauzeće bez gubitka osnovne strukture slike.

---

4. Zaključak

RVNS se pokazao kao efikasniji algoritam za ovaj problem jer uspeva da postigne istu grešku uz kraće vreme izvršavanja. Oba algoritma daju približno isti nivo kompresije (oko 2.5×) pri vrlo sličnom kvalitetu slike (MSE≈21). Ova metoda pokazuje da heurističke tehnike mogu da pruže konkurentne rezultate čak i kod jednostavnih prostornih modela slike.

---

5. Literatura

1. Mladenović, N., & Hansen, P. (1997). *Variable Neighborhood Search*. Computers & Operations Research, 24(11), 1097–1100.
2. Lissovoi, A., Oliveto, P. S., & Warwicker, J. (2018). *Simple Hyper-heuristics Control the Neighbourhood Size of Randomised Local Search Optimally for LeadingOnes*. arXiv preprint arXiv:1801.07546.
3. Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th Edition). Pearson Education.