Algorytmy ewolucyjne - raport

Maria Wyrzykowska

1 Opis rozpatrywanego zagadnienia

Rozpatrywanym zagadnieniem jest próba zrekonstruowania wybranego obrazu za pomocą ograniczonej liczby kół o różnej pozycji, rozmiarze, kolorze i przezroczystości. Do testowania algorytmów wybrano obraz 'Mona Lisa' Leonardo da Vinci'ego.
Obserwowanie działania algorytmów może być źródłem rozrywki. Innym potencjalnych zastosowaniem wyników obliczeń może być kompresja danych - dla przykładu, przedstawiając obrazek 300 x 300 px zakodowany w przestrzeni RGB za pomocą 500 kół, redukujemy zużycie pamięci o ponad 98%.
Oryginalny obrazek:

2 Definicja problemu

Przestrzenią poszukiwań są obrazy 299 x 299 px zakodowane w przestrzeni RGB.
Funkcję celu wyliczamy w następujący sposób:

$$F(individual) = \sum_{i=1}^{299} \sum_{j=1}^{299} \sum_{C \in \{R,G,B\}} (C_{i,j,orginal} - C_{i,j,individual})^2$$

gdzie $C_{i,j,orginal}$ oznacza wartość koloru C z kodowania RGB piksela o pozycji (i, j) orginalnego obrazka, a $C_{i,j,individual}$ - obrazka wygenerowanego przez algorytm.

Reprezentacja osobnika wygenerowanego przez algorytm to tablica, której każdy z elementów jest wektorem opisującym jedno koło jako $[x, y, z, q, R, G, B, A]$.
x, y - współrzędne lewego górnego rogu kwadratu w który wpisane jest koło
z, q - współrzędne prawego dolnego rogu kwadratu w który wpisane jest koło
R, G, B, A - kodowanie RGBA koloru koła.
Koła są nanoszone na początkowo czarny obrazek rozmiaru 299 x 299 px w kolejności występowania w tablicy z wykorzystaniem biblioteki PIL 8.1.

3 Opis użytych algorytmów ewolucyjnych oraz wyników ich działania

3.1 Metoda #1: mutacja dodająca losowe koła

Sposób działania algorytmu:

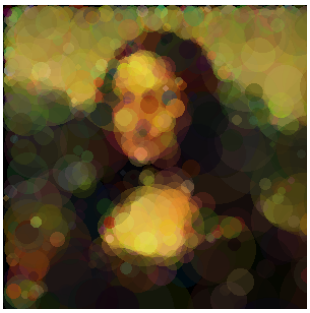
W każdej iteracji tworzymy 100 kopii rodzica i każdą z nich mutujemy. Mutacja z prawdopodobieństwem 0.5 dodaje losową elipsę, a w pozostałych przypadkach usuwa losową elipsę. Z prawdopodobieństwem 0.5 możemy także kolejny raz zmutować osobnika. Po zmutowaniu wszystkich dzieci, wybieramy najlepszego osobnika spośród dzieci i rodzica. Staje on się rodzicem w kolejnej iteracji.

Otrzymane wyniki:

Liczba iteracji	4000
Liczba rodziców	1
Liczba dzieci	100
Czas wykonania	37 min
Liczba kół	601
Wartość funkcji celu	74.110.852

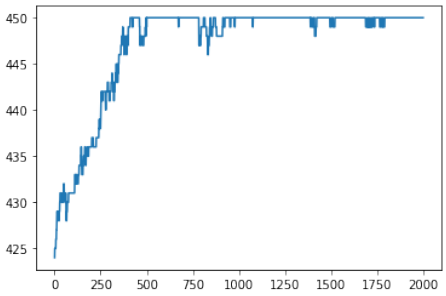
Zauważmy, że takie podejście w żaden sposób nie ogranicza maksymalnej liczby kół. Pozwala to dość szybko osiągnąć satysfakcjonujący wygląd, ale po dłuższym czasie może prowadzić do używania bardzo dużej liczby kół, więc jeśli chcielibyśmy zastosować tą metodę do kompresji danych, to musimy w jakiś sposób przeciwdziałać generowaniu dużej liczby kształtów.

Otrzymany obrazek:



3.1.1 Próby ograniczenia liczby kół: ścisły limit

Oczywistą metodą ograniczenia liczby kół jest narzucenie ścisłego limitu - po jego osiągnięciu mutacja może tylko usuwać koła, aż ich liczba zmniejszy się i znów będziemy pozwalać na ich dodawanie. Przy takich samych parametrach jak wyżej i ogarniczeniu liczby kół do 450 otrzymaliśmy wartość funkcji celu równą 84.757.413. Limit liczby kół został osiągnięty około 2400 iteracji, a ich liczba od 2000 iteracji prezentowała się następująco:



3.1.2 Próby ograniczenia liczby kół: większy minimalny rozmiar koła

Inną metodą ograniczenia liczby kół jest zwiększenie ich minimalnego rozmiaru, przez co ich dodawanie będzie spowolnione i utrudnione. We wszystkich pozostałych metodach minimalna długość boku kwadratu w który wpisane jest koło wynosi 5 px. Po zwiększeniu tej wartości do 8 px i pozostawieniu reszty parametrów bez zmian otrzymaliśmy wartość funkcji celu równą 76.023.231 i liczbę kół wynoszącą 554, zatem około 50 mniej niż bez prób jej zmniejszenia.

3.1.3 Próby ograniczenia liczby kół: mniejsze prawdopodobieństwo dodania koła przy przekraczaniu limitu

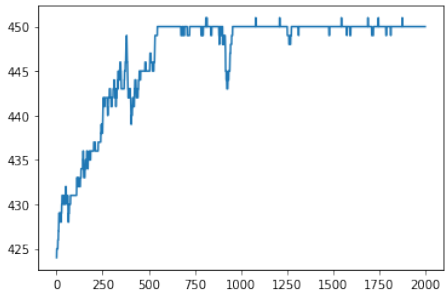
Kolejną zastosowaną metodą ograniczenia liczby kół jest zmniejszenie prawdopodobieństwo dodania koła przez mutację, a zwiększenia prawdopodobieństwa usunięcia koła w zależności od tego jak bardzo przekraczamy limit. Prawdopodobieństwa zmieniano o $0,01 * koła - ponad - limit$. Okazuje się, że to dość słaba metoda. Po ustaleniu limitu na 450 kół i pozostawieniu reszty parametrów bez zmian, otrzymana wartość funkcji celu wynosi 82.703.473, a liczba kół 499. Maksymalna możliwa do osiągnięcia liczba kół to 500, ponieważ wtedy prawdopodobieństwo dodania nowego koła wyniesie 0. Okazuje się, że przy większej liczbie iteracji osiągniemy tą wartość, ponieważ dodawanie odpowiednich nowych kół nam się opłaca. W efekcie otrzymamy podobne wyniki jakie dostalibyśmy ustalając ścisły limit na 500 kół, ale wolniej.

3.1.4 Próby ograniczenia liczby kół: funkcja celu karająca za przekraczanie limitu kół

Ostatnią wypróbowaną metodą jest zmiana funkcji celu na taką, która kara osobniki przekraczające limit liczby kół:

$$F'(individual) = F(individual) * max(1, \frac{liczba-kół}{limit-kół})$$

To rozwiązanie sprawdziło się najlepiej - otrzymana wartość (oryginalnej) funkcji celu przy limicie liczby kół równym 450 wynosi 81.617.456, a ostateczna liczba kół - 450. Jest to lepszy wynik niż w przypadku twardego limitu, ponieważ pozwalamy na dodanie kół ponad limit gdy znacznie poprawiają one osobnika.



3.1.5 Podsumowanie metody #1

Nazwa	Liczba kół	Wartość funkcji celu
Bez ograniczeń na liczbę kół	601	74.110.852
Ścisły limit liczby kół	450	84.757.413
Większy minimalny rozmiar kół	554	76.023.231
Mniejsze prawdopodobieństwo dodania koła	499	82.703.473
Funkcja celu karająca dużą liczbę kół	450	81.617.456

3.2 Metoda #2: mutacja zmianiająca pozycję i kolor kół

Sposób działania algorytmu:

Zaczynamy od rodzica składającego się ze stałej liczby zupełnie losowych kół. W każdej iteracji tworzymy niewiele kopii rodzica i każdą z nich mutujemy. Mutacja polega na wybraniu pewnych kół każdego osobnika (każde koło wybieramy z takim samym **prawdopodobieństwem**), osobno tych którym będziemy zmieniać pozycję i kolor.

```
size = len(dna)
mask_color = np.random.choice([0, 1],
             size = size,
             p=[1 - mutation_probability, mutation_probability])
mask_position = np.random.choice([0, 1],
                size = size,
                p=[1 - mutation_probability, mutation_probability])
```

Każdą z wartości R, G, B, A (dla kół, którym mutujemy kolor), współrzędnych (x, y) lewego górnego rogu kwadratu w który wpisane jest koło i długości boku tego kwadratu (dla kół, którym mutujemy pozycję) zmieniamy o losowe wartości z zakresu (**-moc**,**moc**) Dodatkowo, mutacja dba aby mutowane koło nie zniknęło, co mogłoby nastąpić w wyniku przesunięcia jeg pozycji poza obrazek, wyzerowania długości boku kwadratu w którego jest ono wpisane czy ustawienia jego wartości A na odpowiednio małą. Po zmutowaniu wszystkich dzieci, wybieramy najlepszego osobnika spośród dzieci i rodzica. Staje on się rodzicem w kolejnej iteracji. Wykonujemy znacznie większą niż w przypadku mutacji #1 liczbę iteracji, ale pracujemy na mniejszej populacji.

Otrzymane wyniki dla różnych parametrów mutacji:

Liczba iteracji	50000
Liczba rodziców	1
Liczba dzieci	5
Czas wykonania	40 min
Liczba kół	450

Prawdopodobieństwo \ Moc	3	5	10
0.005	49.254.074	41.689.254	38.005.935
0.02	65.698.614	56.064.915	57.182.649
0.05	100.236.710	89.275.864	94.784.437

Jak widać otrzymane wyniki są w najlepszym wypadku niemalże o połowę niższe niż w przypadku metody #1. Nie natykamy się także na problem wzrastającej liczby kół, ponieważ ta jest tu od początku ustalona.

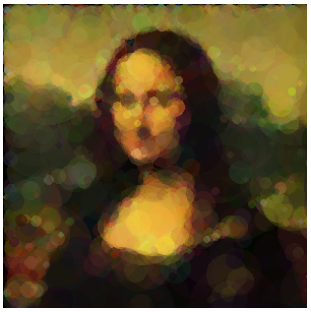
Otrzymany obrazek (dla prawdopodobieństwa = 0.005, mocy = 10):

4 Wyniki końcowe

Do ostatecznych testów wybrałam metodę #1 bez ograniczania liczby kół, z ograniczaniem liczby kół przez funkcję celu karającą ich nadmiar oraz metodę #2 z optymalnymi parametrami mutacji. Wyniki prezentują się następująco:

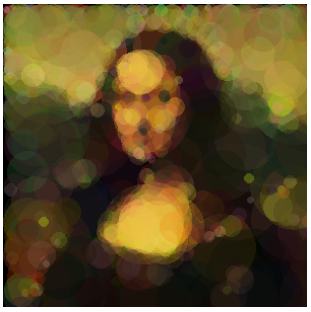
Metoda #1 bez ograniczania liczby kół

Liczba iteracji	15000
Liczba rodziców	1
Liczba dzieci	100
Czas wykonania	ok 2h40min
Liczba kół	977
Wartość funkcji celu	48.294.558



Metoda #1 z funkcją celu karającą nadmiar kół

Liczba iteracji	15000
Liczba rodziców	1
Liczba dzieci	100
Czas wykonania	2h30min
Limit liczby kół	500
Wartość funkcji celu	61.944.361



Metoda #2 z mutacją o prawdopodobieństwie = 0.005 i mocy = 10

Liczba iteracji	150000
Liczba rodziców	1
Liczba dzieci	5
Czas wykonania	1h50min
Liczba kół	500
Wartość funkcji celu	31.266.953

5 Podsumowanie

Podsumowując, najlepsze wyniki pod względem wartości funkcji celu i liczby kół otrzymujemy korzystając z metody #2. Osobiście uważam jednak, że metoda #1, pomimo osiągania gorszych wyników, daje ciekawe efekty wizualne i warto było ją wypróbować.
Projekt daje wiele perspektyw dalszego rozwoju.
Moglibyśmy wypróbować podejście, w którym w mądry sposób dodajemy nowe koła co daną liczbę iteracji, pozwalając algorytmowi ustabilizować się przed dodaniem kolejnego kształtu. Po pierwszych testach ta matoda wydaje się jednak bardziej czasochłonna niż dotychczas opisane.
Warto byłoby przyspieszyć także działanie algorytmów. Aktualnie najbardziej czasochłonne jest rysowanie obrazka dla danego chromosomu i liczenie wartości funkcji celu. Przyspieszenie byłoby możliwe przez zastosowanie biblioteki CUDA do obliczeń wielowątkowych i/lub napisanie własnej biblioteki do rysowania, zoptymalizowanej pod nasze algorytmy.
Ciekawe efekty wizualne mogłoby dać zastosowanie innych niż koła kształtów. Przedstawianie na przykład rysów twarzy za pomoca kół jest trudne, a wykorzystanie trójkątów mogłoby to ułatwić.