

Ewolucyjny malarz

Rafał Kwiatkowski, Franciszek Sioma

24 kwietnia 2020

1 Opis projektu

1.1 Cel projektu

Celem projektu *Ewolucyjny Malarz* było stworzenie aplikacji wykorzystującej algorytm ewolucyjny. Aplikacja przyjmuje na wejściu kolorowy obrazek, a na wyjściu zwraca możliwe jak najdokładniejsze odwzorowanie obrazu wejściowego przy użyciu prostokątów RGBA.

1.2 Przyjęte założenia

W algorytmie ewolucyjnym istotne jest to, aby zdefiniować czym będzie *Osobnik*. W naszym przypadku osobnikiem jest obraz stworzony z n prostokątów RGBA. Każdy prostokąt składa się z 8 wartości:

R kanał czerwony

G kanał zielony

B kanał niebieski

A kanał alpha (mówi o przezroczystości danego piksela)

X pozycja względem osi poziomej

Y pozycja względem osi pionowej

W szerokość

H wysokość

Kolejnym niejasnym punktem może być sposób obliczania funkcji przystosowania (J). W naszym przypadku funkcja ta będzie przyjmować wartości od 0 do 1 i będzie obliczana na podstawie porównywania piksel po pikselu obrazu stworzonego przez osobnika z obrazem oryginalnym.

1.3 Wkład autorów

- Obsługa obrazów - Rafał Kwiatkowski
- Algorytm Ewolucyjny - Franciszek Sioma
- Testy i eksperymenty - Rafał Kwiatkowski
- Dokumentacja - Franciszek Sioma

1.4 Decyzje projektowe

Algorytm zakłada dwa rodzaje krzyżowania osobników:

- uśrednianie
- interpolacja

W naszym programie postanowiliśmy umieścić oba te sposoby, by móc przebadć ich wpływ na końcowy wynik algorytmu. Taki sam zabieg dotyczy metod wybierania kolejnej generacji populacji. Zaimplementowane metody to:

- metoda μ najlepszych
- metoda koła ruletki
- metoda selekcji rankingowej

Algorytm zwraca najlepszego osobnika wraz z jego wartością funkcji dopasowania. Program wyświetla obraz reprezentowany przez tego osobnika.

1.5 Wykorzystane narzędzia i biblioteki

Do napisania aplikacji użyliśmy języka Python w wersji: 3.8, dokumentacja została stworzona przy użyciu języka Latex, a IDE z którego korzystaliśmy to Visual Studio Code. Użyliśmy również systemu kontroli wersji Git.

Spis użytych bibliotek znajduje się w pliku *requirements.txt*. Do najważniejszych z nich należy biblioteka Pillow wykorzystana do obsługi obrazów.

2 Uruchamianie aplikacji i odtworzenie wyników testów

W celu uruchomienia aplikacji należy wykonać komendę:

```
python app.py "simple_img.jpg" 30 20 45 1000 0.95 "inter" "best"
```

Wówczas aplikacja zostanie wykonana dla wielkości populacji równej 30, w każdym z osobników będzie 20 prostokątów, podpopulacja wybierana z populacji do krzyżowania będzie składać się z 45 osobników, maksymalna liczba iteracji

wyniesie 1000, a warunkiem stopu będzie dopasowanie na poziomie 95%. Algorytm użyje interpolacji przy krzyżowaniu, oraz użyje metody najlepszego dopasowania do wybrania kolejnej generacji populacji. Pozostałe dostępne wartości to "mean- uśrednianie dla krzyżowania oraz roulette" i "zanking" (odpowiednio metoda koła ruletki i selekcji rankingowej) dla wybierania.

3 Eksperymenty

Wykonaliśmy szereg testów, sprawdzających jak działa aplikacja w zależności od następujących parametrów:

- liczebność populacji
- liczebność populacji μ
- liczba prostokątów w osobniku
- strategia krzyżowania
- strategia wybierania kolejnej generacji populacji

3.1 Eksperymenty i wyniki

W celu uzyskania możliwie jak najbardziej prawdziwych wyników dla każdego przypadku testowego, każdy eksperyment z danymi parametrami był przeprowadzony 5 razy. Wszystkie testy zostały wykonane dla obrazu *simple_img.jpg* oraz dla z warunkami zakończenia 1000 iteracji lub dokładność 99%. Gdy nie było to przedmiotem eksperymentu to metodą krzyżowania było uśrednianie, a metodą wybierania - wybór najlepszych.

Wpływ liczebności populacji

Test wpływu liczebności populacji został przeprowadzony dla populacji o wielkościach: 2, 10, 15, 30, 40, 50, 100, liczbie prostokątów w osobniku równej 20, wielkości populacji μ będącej 1,5 raza większej od wielkości testowanej populacji.

Wpływ liczebności populacji μ

Test wpływu liczebności populacji μ został przeprowadzony dla stałej wielkości populacji λ równej 40 na podstawie której kolejne przypadki testowe był obliczane za pomocą wartości procentowych: 110%(44), 130%(52), 150%(60), 200%(80), 250%(100), 300%(120). Liczba prostokątów w osobniku była równa 20.

Wpływ liczby prostokątów w osobniku

Następny test został przeprowadzony dla następujących liczb prostokątów w osobniku: 10, 20, 50, 100, 200, 300. Wielkość populacji wynosiła 10, podpopulacji 15. Pozostałe parametry zostały niezmienione względem poprzedniego testu.

Wpływ użytej strategii krzyżowania

Dla populacji λ - 20, populacji μ - 30 i liczbie 20 prostokątów w osobniku testy zostały przeprowadzone dla uśredniania i interpolacji.

Wpływ użytej strategii wybierania kolejnej populacji

Dla populacji λ - 20, populacji μ - 30 i liczbie 20 prostokątów w osobniku testy zostały przeprowadzone dla wyboru najlepszych, ruletki i metody rankingu.

3.2 Wnioski