[PSZT] RB.S14 A może na wyspę?

Adrian Brodzik Jakub Górka

2 grudnia 2019

Zadanie

Porównać standardowy algorytm ewolucyjny z jego wersją, w której zastosowano model wyspowy (ang. *island model*). W modelu tym stosuje się podział populacji na podgrupy. Każda podpopulacja rozwija się oddzielnie, ale co pewien czas występują migracje.

Teza

Algorytm genetyczny z modelem wyspowym jest szybszy i efektywniejszy niż klasyczny algorytm genetyczny z jedną wyspą.

Rozwiązanie

Model wyspowy działa podobnie jak klasyczny algorytm genetyczny z tą różnicą, że pod koniec każdej generacji może nastąpić migracja osobników do różnych wysp.

Osobniki początkowe są generowane losowo. Wybór N osobników, np. do krzyżowania, jest losowym wyborem ważonym, tzn. prawdopodobieństwo wybrania danego osobnika zależy od jego przystosowania. Nie ma gwarancji, że zostaną wybrani najlepsi. Taka implementacja ma zapewnić zachowanie różnorodności populacji.

Przyjęliśmy, że migracje występują dokładnie co X generacji. Poza tym jest szansa, że emigranci z jednej wyspy trafią do wyspy, z której wyruszyli, czyli migracja może się nie odbyć.

Algorithm 1 Sekwencyjny algorytm genetyczny z modelem wyspowym

```
wygeneruj wyspy

while not koniec do
  for all wyspy do
    wybierz dwóch osobników
    wykonaj krzyżowanie
    wykonaj mutacje
    oceń przystosowanie dzieci
    dodaj dzieci do populacji
    usuń nieprzystosowane osobniki z populacji
  end for

  if czas na migracje then
    for all wyspy do
      wybierz osobniki emigrujące
      wylosuj wyspę docelową
      zapisz emigrantów i wyspę docelową
    end for

    dodaj emigrantów do odpowiednich populacji wysp
  end if
end while
```

W rzeczywistości nie stosuje się algorytmu sekwencyjnego, tylko algorytm równoległy, który znacznie przyspiesza obliczenia. Każda wyspa dostaje swój proces, na którym tworzone są kolejne generacje populacji. Procesy są synchronizowane wtedy, gdy nadejdzie czas na migracje.

Algorytm został zaimplementowany w języku Python, wykorzystując biblioteki: `numpy`, `pandas`, `tqdm`, `multiprocessing`, `math`, `json`, itp.

Podział obowiązków

Adrian Brodzik	Jakub Górka
implementacja algorytmu	
optymalizacja funkcji Griewanka	optymalizacja funkcji Rastrigina
analiza wstępna	analiza szczegółowa
stylizacja dokumentacji	treść dokumentacji

Testowanie

Jako przykładowe problemy optymalizacyjne wybraliśmy funkcję Rastrigina

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) \right]$$

oraz funkcję Griewanka

$$f(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right).$$

Obie funkcje posiadają minimum globalne w punkcie $\mathbf{x} = \mathbf{0}$ oraz bardzo dużo ekstremów lokalnych. Złożoność (wymiar) problemu ustala parametr n.

Wykonaliśmy testy dla funkcji Griewanka dla $n = 20$ oraz funkcji Rastrigina dla $n = 20$ i $n = 40$. Zbadaliśmy kombinacje różnych parametrów:

- liczba wysp: $1, 2, 3, 4, 5, 10, 15, 20$
- liczba osobników jednej wyspy: $100, 500, 1000$
- prawdopodobieństwo mutacji pojedynczego bitu: $10\%, 1\%, 0.1\%$
- częstotliwość migracji (co X generacji): $100, 500, 1000$
- liczba osobników migrujących: $1, 2, 3, 4, 5, 10$

Wyniki testowe można odtworzyć, ustawiając odpowiednie parametry przy uruchamianiu pliku `main.py`. **Uwaga**: jeśli liczba wysp wynosi 1, należy zamienić funkcję `run_parallel` na `run_single_island`. Istnieje też możliwość dodania własnej funkcji przystosowania dekodującej genotyp osobników; przykłady znajdują się w `decoder.py` oraz `benchmark.py`.

Testowanie zostało przeprowadzone głównie w chmurze serwisu Kaggle. Celem było osiągnąć *przystosowanie* $= 0$. Dla testów z jedną wyspą ograniczenie czasowe wynosiło 9 godzin. Dla testów z więcej niż jedną wyspą ograniczenie czasowe wynosiło od 3 do 5 minut. Każdy problem testowy został zbadany dwa razy (dla dwóch różnych ziaren generatora liczb pseudolosowych). Wszystkie wyniki oraz użyte parametry zostały zapisane i załączone.

Wyniki

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
15	100	0.001	1000	1	12000	0
15	100	0.001	1000	3	10000	0
15	100	0.001	1000	4	12000	0
10	100	0.01	1000	10	16000	0
10	100	0.001	1000	2	11000	0
15	100	0.001	1000	3	10000	0
10	100	0.001	1000	4	16000	-0.023
15	100	0.001	1000	2	12000	-0.024
15	100	0.001	1000	5	12000	-0.025
15	100	0.001	500	1	11500	-0.026

Tablica 1: Najlepsze wyniki funkcji Griewanka dla $n = 20$

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
1	100	0.001	0	0	80279	-0.394
1	1000	0.001	0	0	49840	-0.395
1	1000	0.01	0	0	98525	-0.411
1	500	0.01	0	0	49929	-0.418
1	500	0.001	0	0	105812	-0.426

Tablica 2: Najlepsze wyniki funkcji Griewanka dla $n = 20$ (jedna wyspa)

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
10	100	0.001	500	1	8500	0
10	100	0.001	500	2	8000	0
10	100	0.001	500	3	7500	0
10	100	0.001	500	4	7500	0
10	100	0.001	500	5	8000	0
10	100	0.001	500	10	8000	0
10	100	0.001	1000	2	8000	0
10	100	0.001	1000	3	8000	0
10	100	0.001	1000	10	8000	0
2	100	0.01	500	2	19500	0

Tablica 3: Najlepsze wyniki funkcji Rastrigina dla $n = 20$

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
1	1000	0.01	0	0	49822	-2.168
1	1000	0.001	0	0	33264	-4.002
1	1000	0.01	0	0	60437	-4.025
1	1000	0.001	0	0	99927	-4.975
1	100	0.01	0	0	49934	-5.969

Tablica 4: Najlepsze wyniki funkcji Rastrigina dla $n = 20$ (jedna wyspa)

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
10	100	0.001	1000	5	8000	-1.428
10	100	0.001	500	3	7500	-1.503
10	100	0.001	1000	10	8000	-1.594
10	100	0.001	500	1	7500	-1.969
10	100	0.001	500	2	7500	-2.276
10	100	0.001	1000	10	7000	-2.642
10	100	0.001	1000	5	7000	-2.887
10	100	0.001	1000	3	7000	-2.890
10	100	0.001	1000	4	7000	-3.192
10	100	0.001	1000	2	7000	-3.507

Tablica 5: Najlepsze wyniki funkcji Rastrigina dla $n = 40$

liczba wysp	liczba osobników	prawd. mutacji	częstotliwość migracji	liczba migrantów	liczba generacji	przystosowanie
1	1000	0.01	0	0	82823	-10.798
1	500	0.001	0	0	78153	-14.930
1	1000	0.001	0	0	51710	-15.149
1	500	0.01	0	0	49919	-15.835
1	500	0.01	0	0	84496	-15.879

Tablica 6: Najlepsze wyniki funkcji Rastrigina dla $n = 40$ (jedna wyspa)

Dyskusja

Problem o większej złożoności dla modelu z kilkoma wyspami uzyskuje lepsze wyniki przystosowania niż model klasyczny dla problemu o dwukrotnie mniejszej wartości parametru n. Oczywiście liczba wysp zależy od rozmiaru przeszukiwanej przestrzeni, np. dla funkcji Rastrigina $n = 20$ wystarczyło użycie 10 wysp.

W modelu klasycznym, pomimo dużej liczby generacji, uzyskane rozwiązanie jest znacznie gorsze od modelu wyspowego, gdzie liczba generacji dla każdej wyspy jest nawet 10 razy mniejsza. Prawdopodobieństwo mutacji w większości przypadków jest niskie. Mimo że losowe zmiany bitów pozytywnie wpływają na dywersyfikację populacji, to zwiększają ryzyko pogorszenia przystosowania danego osobnika. Rzadkie migracje pozwalają na uzyskanie lepszych wyników, gdyż wyspy potrzebują odpowiedniej liczby iteracji, aby skutecznie przeszukać własną wyspę (przestrzeń lokalną) zanim wyemigrują do innej.

Wnioski

Model wyspowy skuteczniej przeszukuje przestrzeń w celu znalezienia optimum niż model klasyczny. Dla jednej wyspy lepsze są większe zbiory populacji, gdyż pozytywnie wpływają na dywersyfikację oraz eksplorację przestrzeni. W przypadku wielu wysp małe populacje były wystarczające, gdyż każda wyspa zajmowała się eksploracją osobno. Im więcej użytych wysp w rozwiązaniu tym lepsza wartość przystosowania dla problemów o większej złożoności w porównaniu do modelu klasycznego. Liczba wysp powinna zależeć od rozmiaru przeszukiwanej przestrzeni. Zbyt duża ilość wysp czy osobników populacji może negatywnie wpłynąć na wyniki.

Realizacji projektu pozwoliła nam dokładnie przeanalizować wpływ poszczególnych parametrów na zachowanie algorytmu genetycznego z modelem wyspowym. Dzięki przeprowadzonym testom dowiedzieliśmy się, że zastosowanie wysp jest lepszym wyborem rozwiązywania problemów o większej złożoności w porównaniu do modelu klasycznego z jedną wyspą.

Literatura

[1] Yiyuan Gong and Alex Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011.

[2] Vijini Mallawaarachchi. Introduction to genetic algorithms, 2017.

[3] Darrell Whitley, Soraya Rana, and Robert Heckendorn. The island model genetic algorithm: On separability, population size and convergence, 1998.