Podstawy sztucznej inteligencji
Sprawozdanie z projektu - przeszukiwanie "RB.S7 A może ślub?"

Radosław Tuzimek
Marcin Hanas

24 kwietnia 2020

1 Cel projektu oraz jego założenia

Celem projektu było zmodyfikowanie klasycznego algorytmu ewolucyjnego oraz zbadanie różnic w obu jego wersjach. Modyfikacja polegała na łączeniu osobników w pary (aż do śmierci) i obliczaniu funkcji przystosowania J jako średnia z wartości J obu osobników pary.

2 Implementacja algorytmów

2.1 Implementacja klasycznego algorytmu ewolucyjnego

Na potrzeby projektu został zaimplementowany klasyczny algorytm ewolucyjny, którego ogólny schemat prezentuje się następująco:

1. Wygeneruj P - populację μ osobników.
2. Wybierz przez losowanie ze zwracaniem T - λ-elementową populację tymczasową.
3. Reprodukuj z T λ-elementową populację potomną R, stosując mutację.
4. Utwórz nowe P jako μ osobników z największą wartością funkcji przystosowania J, wybranych z $P \cup R$.
5. Jeśli jest spełniony warunek stopu, zwróć najlepszego osobnika z P jako wynik. W przeciwnym wypadku wróć do punktu 2.

Algorytm został zaimplementowany w pliku `EvolutionaryAlgorithm.py`. Został on użyty do znajdowania minimum funkcji ze standardowego zestawu funkcji benchmarkujących `CEC2017`, zatem po pomnożeniu przez -1 (bo algorytm domyślnie maksymalizuje, a poszukiwane jest minimum) pełnią one rolę funkcji przystosowania J. Jako warunek kończący działanie algorytmu przyjęta została ilość wykonanych iteracji postaci:

$$\frac{10000 * \text{wymiarowość zadania}}{\text{rozmiar generowanej populacji}} \tag{1}$$

Algorytm zwraca osobnika z ostatniego pokolenia, dla którego wartość funkcji przystosowania jest najmniejsza.

2.2 Implementacja modyfikacji klasycznego algorytmu ewolucyjnego

Dokonano modyfikacji opisanego wyżej algorytmu ewolucyjnego w ten sposób, aby osobniki dobierane były w pary (aż do śmierci), a funkcja przystosowania liczona była zgodnie ze wzorem:

$$J(p) = J(q) = \text{mean}(J(p), J(q)) \tag{2}$$

gdzie p i q to osobniki tworzące parę. Uogólniony schemat algorytmu jest następujący:

1. Wygeneruj P - populację μ osobników, połączonych w pary.
2. Wybierz przez losowanie ze zwracaniem $\frac{\lambda}{2}$ par osobników stanowiących T - populację tymczasową.
3. Reprodukuj z T λ-elementową populację potomną R, stosując mutację każdego pojedynczego osobnika.
4. Połącz w pary R, poprzez losowanie bez zwracania.
5. Utwórz nowe P jako $\frac{\mu}{2}$ najlepszych par osobników (przyjmując, że funkcja przystosowania pary to średnia z wartości funkcji przystosowania J osobników tej pary) wybranych z $P \cup R$.
6. Jeśli jest spełniony warunek stopu, oblicz wartość J indywidualnie dla każdego osobnika i zwróć najlepszego jako wynik. W przeciwnym wypadku wróć do punktu 2.

Algorytm zaimplementowany został w pliku `ModifiedEvolutionaryAlgorithm.py`. Tak samo jak poprzedni algorytm, ten również użyto do znajdowania minimum funkcji ze standardowego zestawu funkcji benchmarkujących `CEC2017` (analogicznie są używane jako J po zmianie znaku). Przyjęto taki sam warunek kończący działanie algorytmu - ilość wykonanych iteracji wyliczany tym samym wzorem. Algorytm zwraca jednego osobnika z ostatniego pokolenia, dla którego wartość funkcji przystosowania (liczona standardowo, a nie jako średnia dla pary) jest najmniejsza.

2.3 Uruchomienie projektu

Na początku należy skompilować plik z oryginalną implementacją funkcji CEC17 napisaną w języku C. Dla systemu Ubuntu należy wykorzystując okno terminala otworzyć folder projektu i wpisać następujące polecenie:

```
gcc -fPIC -shared -lm -o cec17_test_func.so cec17_test_func.c
```

W efekcie tej operacji powinien w folderze projektu powstać plik cec17_test_func.so. Następnie należy uruchomić program testing.py.

2.4 Wykorzystane narzędzia i biblioteki

Programy napisano w języku python i wykorzystano do tego biblioteki `numpy`, `timei` dodatkowo do wizualizacji - `matplotlib`. Wykorzystano również bibliotekę funkcji benchmarkujących `CEC2017` wraz z modułem umożliwiającym wywołanie ich w pythonie.

3 Cele i tezy przeprowadzonych badań

W celu przeprowadzenia testów napisany został program wykonujący testy. Jego implementacja znajduje się w pliku testing.py. Oba algorytmy porównano poprzez użycie ich do znalezienia minimów funkcji benchmarkujących z biblioteki `CEC2017`. W pętli wybierano losowe populacje początkowe, takie same dla obydwu algorytmów, następnie uruchamiano oba algorytmy i zapisywano wyniki - znalezione przez oba algorytmy minima. Następnie wyznaczono średnie i odchylenia standardowe rozwiązań i obrazowano je na wykresach.
Działania te miały potwierdzić tezę, iż tradycyjny algorytm ewolucyjny znajdzie lepsze minimum globalne, natomiast modyfikacja umożliwia lepsze znajdowanie minimów lokalnych.

4 Wyniki eksperymentów

Eksperymenty zostały przeprowadzone dla 2 oraz 10-wymiarowego przypadku. We wszystkich eksperymentach wartość współczynnika λ wynosiła 30, a μ 20. Jako funkcje przystosowania J zostały wykorzystane funkcje: 5, 7, 9, 10 dla przypadku 2-wymiarowego oraz 5, 9, 12, 17 dla 10 wymiarów. Wyniki doświadczeń zostały przedstawione w tabelach jako wartości średnie z 25 niezależnych uruchomień algorytmów. Oba algorytmy otrzymały te same populacje początkowe, jednak przy każdym uruchomieniu były one inne. Liczbę iteracji wyliczano za pomocą opisywanego wcześniej wzoru, natomiast dla części uruchomień pomnożoną ją przez współczynnik $i = 0.1$ aby zobaczyć, jak szybko metody zbiegają do poszukiwanego wyniku.

4.1 Wyniki dla przypadku o dwóch wymiarach

4.1.1 Badanie wartości funkcji minimalizowanej J dla znalezionych rozwiązań x

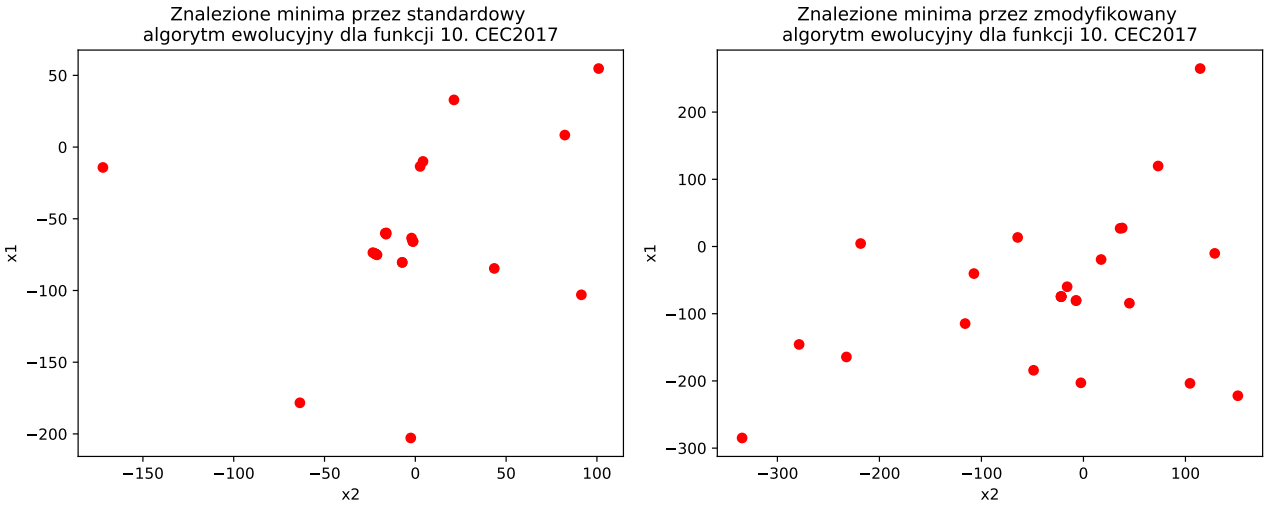
Numer funkcji CEC17 - 5				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	501.145683942	0.711288060113	0.385621s
Zmodyfikowany	0.1	504.015005916	3.00403896108	0.387041s
Klasyczny	1	501.244739711	1.23062833619	0.900762s
Zmodyfikowany	1	501.666929176	1.31276889007	0.889829s

Numer funkcji CEC17 - 7				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	702.400731606	1.16664031747	0.392011s
Zmodyfikowany	0.1	706.676187867	4.05499771782	0.390203s
Klasyczny	1	702.238957075	0.845648648011	0.875833s
Zmodyfikowany	1	703.35444433	1.8620683633	0.874793s

Numer funkcji CEC17 - 9				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	900.084663454	0.5211192721	0.391984s
Zmodyfikowany	0.1	1491.61187859	1918.93622232	0.393039s
Klasyczny	1	900.0	0.0	0.873511s
Zmodyfikowany	1	2274.86757577	6735.16134859	0.875345s

Numer funkcji CEC17 - 10				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	1055.13597702	65.6752967276	0.393054s
Zmodyfikowany	0.1	1157.16650418	105.298673382	0.372636s
Klasyczny	1	1045.03379819	56.3478343989	0.911762s
Zmodyfikowany	1	1159.59050551	169.611257627	0.902432s

4.1.2 Wykresy znalezionych rozwiązań x dla funkcji 10. CEC17



(a) (b)

4.1.3 Odchylenia standardowe znalezionych rozwiązań x dla każdego wymiaru

	Algorytm klasyczny		Algorytm zmodyfikowany	
Numer funkcji CEC17	x_1	x_2	x_1	x_2
5	12.42218023	15.03739447	17.37277135	16.40941158
7	20.97196532	19.80113369	22.50731096	21.48869428
9	2.37132039e-07	4.00745915e-07	132.70415326	57.30163968
10	50.91059485	53.71926847	122.08264096	113.42282208

4.2 Wyniki dla przypadku o dziesięciu wymiarach

4.2.1 Badanie wartości funkcji minimalizowanej J dla znalezionych rozwiązań x

Numer funkcji CEC17 - 5				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	3776.3073170882244	9878.668240976807	2.212275s
Zmodyfikowany	0.1	10291.516124698695	21957.543718202414	2.134550s
Klasyczny	1	753.674788910718	417.031409915584	19.470040s
Zmodyfikowany	1	872.0828867970008	418.55630886340197	19.616160s

Numer funkcji CEC17 - 9				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	1422212.7902651012	1907674.8188996406	1.995400s
Zmodyfikowany	0.1	4582174.848836722	4096087.177693227	1.939350s
Klasyczny	1	1072960.4852246507	1537293.658331386	19.634160s
Zmodyfikowany	1	2004383.0841611815	2540051.755257249	19.312720s

Numer funkcji CEC17 - 12				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	1477512999.3220253	1746326890.9104261	1.888600s
Zmodyfikowany	0.1	40590619371.44869	60344093115.606544	1.891100s
Klasyczny	1	116938215.54399878	118338142.0784286	21.630000s
Zmodyfikowany	1	119172961.44551265	94743949.32221259	21.365120s

Numer funkcji CEC17 - 17				
Rodzaj algorytmu	Wsp. i	Średnia wartość J	Odchylenie std.	Średni czas wykonania
Klasyczny	0.1	5072.244624680641	12831.759346522747	2.107075s
Zmodyfikowany	0.1	89759.21800013508	333193.32968420937	2.037150s
Klasyczny	1	3701.4216665410804	6937.023552841023	20.448000s
Zmodyfikowany	1	99571.46605075363	421771.09173422446	19.996960s

5 Omówienie wyników i wnioski

Na podstawie wyników można zauważyć, że:

1. klasyczny algorytm szybciej zbiega do rozwiązania
2. w prawie wszystkich przypadkach algorytm klasyczny dał mniejsze wartości funkcji dla punktu, który był minimalizowany - był zatem lepszy do znajdowania minimum globalnego,
3. znajdowane przez algorytm klasyczny punkty miały mniejsze odchylenie standardowe (przyjmowały bardziej "podobne" do siebie wartości) - wskazuje to na większy nacisk tego algorytmu na eksploatację,
4. algorytm zmodyfikowany często był nieznacznie szybszy od klasycznego - mogło to być spowodowane koniecznością sortowania dwukrotnie mniejszego wektora (wartości J dla par). Możemy jednak wnioskować, że modyfikacja nie wpłynęła znacząco na czas wykonania algorytmu,
5. algorytm standardowy jest szybciej zbieżny niż zmodyfikowany.

Zbieżność algorytmu została sprawdzona dodatkowymi testami. Wynika z nich, że dla dziesięciokrotnie mniejszej ilości iteracji algorytm klasyczny znajdował rozwiązanie dużo bliższe ostatecznemu niż algorytm zmodyfikowany. Na rozbieżności w wynikach mogły wpłynąć inne populacje początkowe dla przypadków z dziesięciokrotnie mniejszą ilością iteracji.
Powyższe obserwacje mogą świadczyć o tym, że modyfikacja algorytmu spowodowała, że zaczął się on skupiać w większym stopniu na eksploracji, czyli poszukiwaniu nowych okolic w celu znalezienia minimum globalnego, niż na eksploatacji, czyli przeszukiwaniu obszarów w pobliżu minimów lokalnych. Można to zauważyć na wykresie znalezionych przez algorytmy rozwiązań - punkty dla algorytmu zmodyfikowanego są bardziej rozłożone w przestrzeni. Potwierdzeniem tego również mogą być wartości odchyleń standardowych znalezionych rozwiązań, są one zawsze mniejsze dla algorytmu klasycznego, oraz wolniejsza zbieżność algorytmu.

6 Wkład autorów

Poniżej wyszczególnione zostały zadania zrealizowane przez każdego członka zespołu:

1. Wspólne przeanalizowanie problemu
2. Zaimplementowanie algorytmów - Marcin Hanas
3. Dokończenie implementacji algorytmów, przetestowanie oraz poprawa błędów - Radosław Tuzimek
4. Zaimplementowanie programu testującego - Radosław Tuzimek
5. Poprawki w programie testującym - Marcin Hanas
6. Przeprowadzenie testów dla przypadku o dwóch wymiarach - Marcin Hanas
7. Przeprowadzenie testów dla przypadku o dziesięciu wymiarach - Radosław Tuzimek
8. Wspólne przeanalizowanie wyników, wyciągnięcie wniosków oraz sporządzenie sprawozdania.

Podczas realizacji zadania nauczyliśmy się implementacji algorytmów ewolucyjnych, oraz w jaki sposób przeprowadzać testy algorytmów z wykorzystaniem funkcji benchmarkujących.