Raport pracy domowej #5

Metody Losowe Optymalizacji Globalnej

Piotr Podbielski

11 maja 2020

1 Cel pracy domowej

Celem pracy domowej jest:

1. zaimplementowanie algorytmów: *Evolution Strategy*, *Biological Evolution* (pol. *algorytm ewolucyjny*), *Particle Swarm Optimization* oraz *Differential Evolution*,
2. porównanie działania powyższych algorytmów z poprzednio zaimplementowanymi algorytmami, w szczególności z algorytmem *Hill Climbing* (należy zwrócić uwagę na sprawiedliwe porównywanie metod jednopunktowych z populacyjnymi),
3. napisanie raportu zawierającego: (1) wykresy zbieżności algorytmów, (2) wnioski.

2 Przebieg wykonywania zadania

2.1 Zebranie zbioru wartości hiperparametrów dla poprzednio zaimplementowanych algorytmów

Wyniki w ramach poprzednich prac domowych były uzyskiwane z wykorzystaniem hiperparametrów, które były wybierane metodą empiryczną (manualnie). W ramach ostatniej pracy domowej, która ma na celu implementacje różnych algorytmów postanowiono dopisać moduł przeszukujący przestrzeń hiperparametrów w celu odnalezienia optymalnej konfiguracji algorytmów. Tablice 1, 2, 3, 4 ukazują wartości poszczególnych hiperparametrów dla *pewnych* algorytmów.

2.2 Zaimplementowanie algorytmu *Evolution Strategy*

Wszelkie algorytmy zaimplementowane w ramach tej pracy domowej zostały zaimplementowane na podstawie pseudokodów udostępnionych przez prowadzącego w [1]. Algorytm *Evolution Strategy* został zaimplementowany $1:1$. Tablica 5 przedstawia wartości optymalnych hiperparametrów uzyskanych poprzez uruchomienie algorytmu z 500 różnymi konfiguracjami. Każda konfiguracja przeliczana była 500 razy, a następnie końcowy wynik był uśredniany co pozwalało na porównanie różnych zestawów hiperparametrów.

W przypadku obu funkcji jako optymalna wartość parametru *Step mutation coefficient* została wybrana liczba 0.990, co oznacza, że w tym wypadku opłaca się modyfikować obszar przeszukiwań (*step*, będący odchyleniem standardowym) o małe wartości.

Tablica 1: Optymalne wartości hiperparametrów dla algorytmu *Hill Climbing*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Failures to reset	49	14
Step	0.211	0.030
Reset resets failures counter	False	False

Tablica 2: Optymalne wartości hiperparametrów dla algorytmu *Hill Climbing with Adaptive Step Size*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Number of particles	13	11

Tablica 3: Optymalne wartości hiperparametrów dla algorytmu *Bit Switch Hill Climbing*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Failures to reset	13	1
Step	8	2
Reset resets failure counter	True	False
Number of bits for grid mapping per dimension	6	15

Tablica 4: Optymalne wartości hiperparametrów dla algorytmu Bit Switch Hill Climbing with Variable Neighborhood Search.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Neighbor looks	350	800
Number of bits for grid mapping per dimension	16	29

Tablica 5: Optymalne wartości hiperparametrów dla algorytmu *Evolution Strategy*

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Improvements loop iterations	600	2000
Step	0.048	0.906
Step mutation coefficient	0.990	0.990

Tablica 6: Optymalne wartości hiperparametrów dla algorytmu *Biological Evolution*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Population	39	40
Crossover population	19	19
Mutation probability	1	0.004
Mutation step	0.471	1.670

Tablica 7: Optymalne wartości hiperparametrów dla algorytmu *Particle Swarm Optimization*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Number of particles	39	40
Omega	-0.084	-0.077
C1	1.071	0.903
C2	2.052	2.302

2.3 Zaimplementowanie algorytmu *Biological Evolution*

Algorytm *Biological Evolution* został zaimplementowany w następujący sposób:

- krzyżowanie — arytmetyczne uśredniające, dla każdego wymiaru osobno losowane dwa punkty,
- mutacja — arytmetyczna *Gauss*'a z *pewnym* prawdopodobieństwem jej wykonania,
- selekcja — elitarna.

Tablica 6 przedstawia wartości optymalnych hiperparametrów uzyskanych dla algorytmu *Biological Evolution* (pol. *Algorytmu Ewolucyjnego*).

Ciekawą pracą w przyszłości mogłoby być zaimplementowanie innych sposobów selekcji i porównanie wyników uzyskanych dla tych sposobów z wynikami umieszczonymi w wykładzie przez prowadzącego.

2.4 Zaimplementowanie algorytmu *Particle Swarm Optimization*

Algorytm *Biological Evolution* został zaimplementowany zgodnie z pseudokodem umieszczonym w wykładzie, z zaznaczeniem, że p_{best} to najlepszy punkt uzyskany do czasu t, a g_{best} to punkt będący najbliższym sąsiadem punktu, który obecnie jest ewaluowany. Najbliższy sąsiad wyznaczany jest z użyciem metryki euklidesowej.

Tablica 7 przedstawia wartości optymalnych hiperparametrów uzyskanych dla algorytmu *Particle Swarm Optimization* (pol. *Optymalizacja Rojem Cząstek*).

2.5 Zaimplementowanie algorytmu *Differential Evolution*

Podczas implementacji algorytmu *Biological Evolution* dwie rzeczy były istotne:

1. jaki zakres parametru F przeszukiwać,
2. w jaki sposób zaimplementować krzyżowanie.

Ze względu na brak doprecyzowania w wykładzie, zdecydowano się przeszukiwać zakres od 0 do 1 dla parametru F. Krzyżowanie zaś zaimplementowano na zasadzie podmiany wartości danego wymiaru pomiędzy dwoma punktami x i v.

Tablica 8 przedstawia wartości optymalnych hiperparametrów uzyskanych dla algorytmu *Differential Evolution*.

2.6 Wygenerowanie wykresów

Zgodnie z prośbą prowadzącego, wygenerowano wymagane wykresy.

Tablica 8: Optymalne wartości hiperparametrów dla algorytmu *Differential Evolution*.

Nazwa	**Wartość dla funkcji**	
	Rosenbrocka	**Rastrigina**
Population	19	34
F	0.511	0.505

2.6.1 Wykresy zbieżności

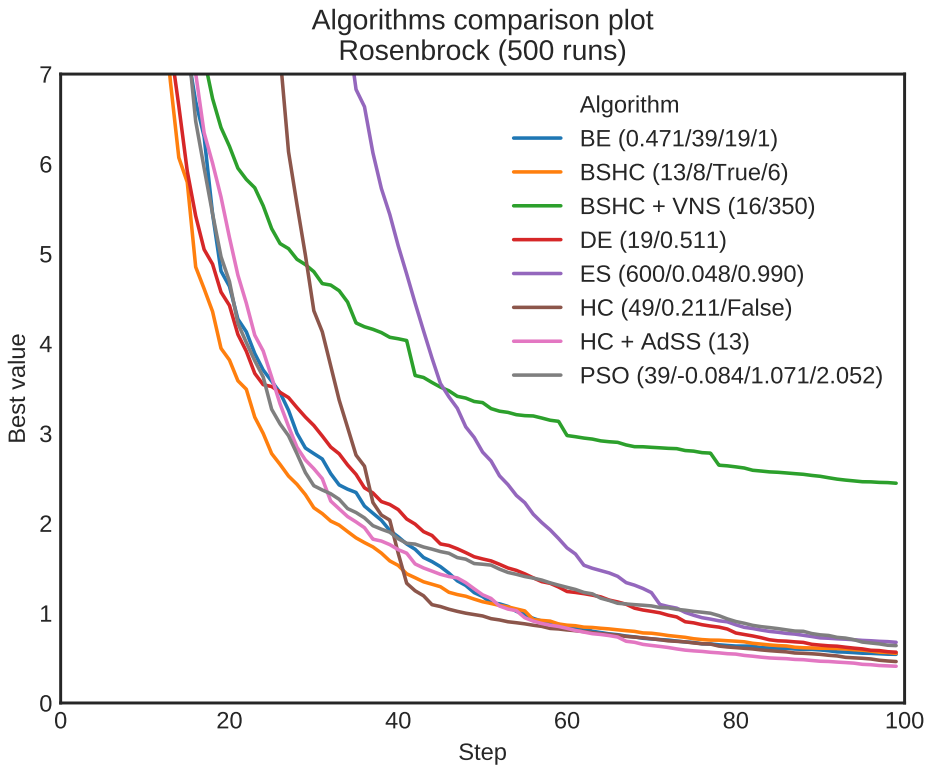
Wykresy zbieżności algorytmów wykonano dla funkcji *Rosenbrock*'a oraz *Rastrigin*'a. Zostały na nich przedstawione porównania algorytmów:

- *Biological Evolution* (BE),
- *Bit Switch Hill Climbing* (BSHC),
- *Bit Switch Hill Climbing with Variable Neighbourhood Search* (BSHC + VNS).
- *Differential Evolution* (DE),
- *Evolution Strategy* (ES),
- *Hill Climbing* (HC),
- *Hill Climbing with Adaptive Step Size* (HC + AdSS),
- *Particle Swarm Optimization* (PSO).

Z porównania wyłączono algorytmy *Grid Search*, *Monte Carlo* oraz *Simulated Annealing*.

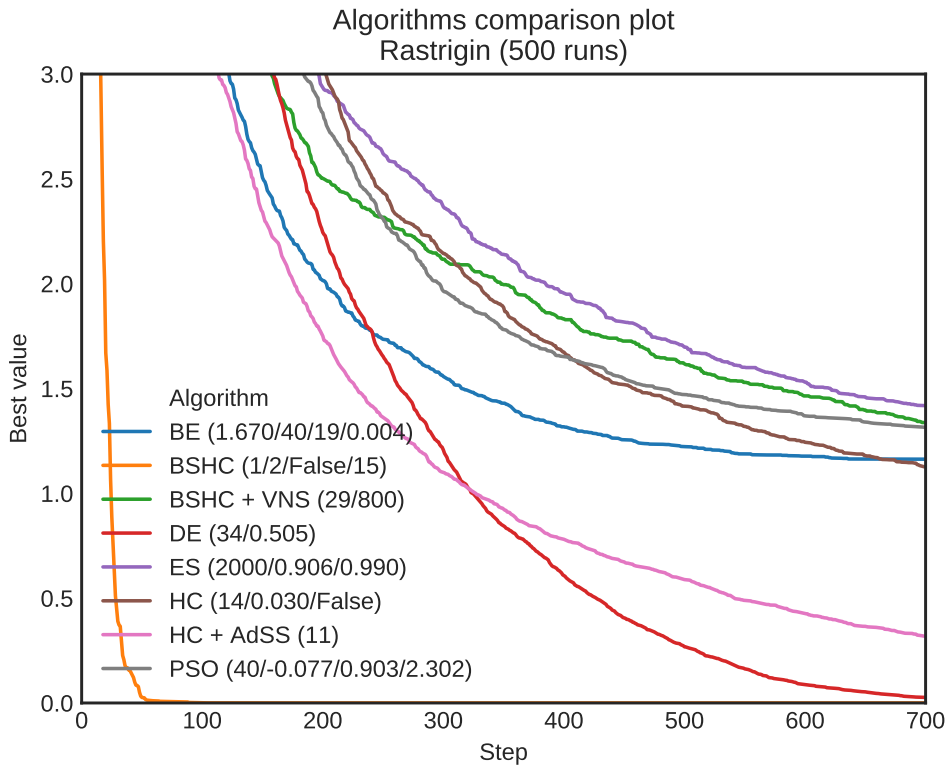
Podczas generowania wykresów zrezygnowano ze wstąg obrazujących wartość odchylenia standardowego, ponieważ zaciemniały one obraz. Wyniki dla każdego algorytmu uśredniane są z 500 wykonań tego algorytmu dla tego samego zestawu parametrów. Dzięki tak dużej liczbie wywołań ukryte wartości odchylenia standardowego są redukowane.

Wykres dla funkcji *Rosenbrock*'a został przedstawiony na rysunku 1.



Rysunek 1: Wykres porównujący wyniki optymalizacji z użyciem różnych algorytmów. Optymalizacja wykonana została dla funkcji *Rosenbrock*'a. Linie przedstawiają wartość średnią z 500 wywołań procesu optymalizacji.

Wykres dla funkcji *Rastrigin*'a został przedstawione na rysunku 2.



Rysunek 2: Wykres porównujący wyniki optymalizacji z użyciem różnych algorytmów. Optymalizacja wykonana została dla funkcji *Rastrigin*'a. Linie przedstawiają wartość średnią z 500 wywołań procesu optymalizacji.

3 Implementacja

Ze względu na objętość kodu napisanego w ramach 5. pracy domowej oraz ogólną dobrą jakość kodu, kod implementacji dostępny jest na portalu *GitHub* w repozytorium VictorAtPL/random-global-optimization-methods.

4 Wnioski

Analizując rysunki 1 oraz 2 ciężko jest jednoznacznie stwierdzić, czy metody wielopunktowe (populacyjne) mają przewagę nad metodami jednopunktowymi. Na wykresie dla funkcji *Rastrigin*'a widać, że metoda jednopunktowa BSHC zbiega szybciej niż inne metody. Dla funkcji *Rosenbrock*'a, ta sama metoda jednopunktowa BSHC zbiega najszybciej, ale nie do optimum globalnego. Do optimum globalnego zbiega zaś metoda HC + AdSS wprowadzająca do metody jednopunktowej, jaką jest *Hill Climbing*, elementy metod populacyjnych do kontroli kroku skoków.

W przypadku metod losowej optymalizacji globalnej pozytywnym aspektem jest to, że metoda może być prosta, a hiperparametry dla niej możemy odnaleźć pół-automatycznie (na przykład dzięki bibliotece `HyperOpt` w j. Python). Niestety nadal nie rozstrzygniętą w ramach zajęć kwestią jest pytanie, czy metody te nadadzą się do bardziej skomplikowanych funkcji, niż dwuwymiarowe funkcje *Rastrigin*'a i *Rosenbrock*'a.

Bibliografia

[1] Michał Okulewicz. *Populacyjne algorytmy optymalizacji globalnej*. [Online; accessed 11. May 2020]. Kw. 2020. URL: http://www.mini.pw.edu.pl/~okulewiczm/downloads/mlog/MLOG_03_ES_EA_PSO_DE.pdf.