Algorytm genetyczny w wersji Hollanda dla wektorów binarnych

Sprawozdanie, laboratorium numer 2 WSI

Rafał Kuśmierz 305858

Działanie zaimplementowanego algorytmu badałem na problemie znalezienia wektora 6 liczb całkowitych z zakresu [-4, 3] maksymalizujących funkcję f:

$$f(x) = -\frac{\sum_{i=1}^{6}\left(x_i^4 - 16x_i^2 + 5x_i\right)}{2}$$

Funkcja –(x^4 – 16x^2 + 5x)/2 dla liczb całkowitych z zakresu [-4, 3] największą wartość przyjmuje dla x = -3 i wynosi ona 39. Maksymalna wartość zadanej funkcji f(x) w zadanym zakresie [-4, 3] powinna wynieść 234 – potwierdza to zaimplementowany przeze mnie algorytm:

```
nowe naj. rozwiązanie w 1 generacji --> f([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1]) = 151.000
nowe naj. rozwiązanie w 1 generacji --> f([0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1]) = 160.000
nowe naj. rozwiązanie w 1 generacji --> f([0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1]) = 175.000
nowe naj. rozwiązanie w 2 generacji --> f([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1]) = 185.000
nowe naj. rozwiązanie w 2 generacji --> f([0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1]) = 194.000
nowe naj. rozwiązanie w 3 generacji --> f([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1]) = 200.000
nowe naj. rozwiązanie w 4 generacji --> f([0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1]) = 204.000
nowe naj. rozwiązanie w 8 generacji --> f([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1]) = 214.000
nowe naj. rozwiązanie w 10 generacji --> f([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1]) = 224.000
nowe naj. rozwiązanie w 13 generacji --> f([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]) = 234.000
Najlepsze uzyskane rozwiązanie --> f([-3, -3, -3, -3, -3, -3]) = 234.000000 w 13 iteracji
```

Funkcja najwyższą wartość przyjmuje dla wektora:

[0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]

Zgodnie z kodem Graya odpowiada on wektorowi liczb całkowitych równemu [-3, -3, -3, -3, -3, -3]

```
values_to_check = {
    "000": -4,
    "001": -3,
    "011": -2,
    "010": -1,
    "110": 0,
    "111": 1,
    "101": 2,
    "100": 3
}
```

Wpływ rozmiaru populacji na działanie zaimplementowanego algorytmu

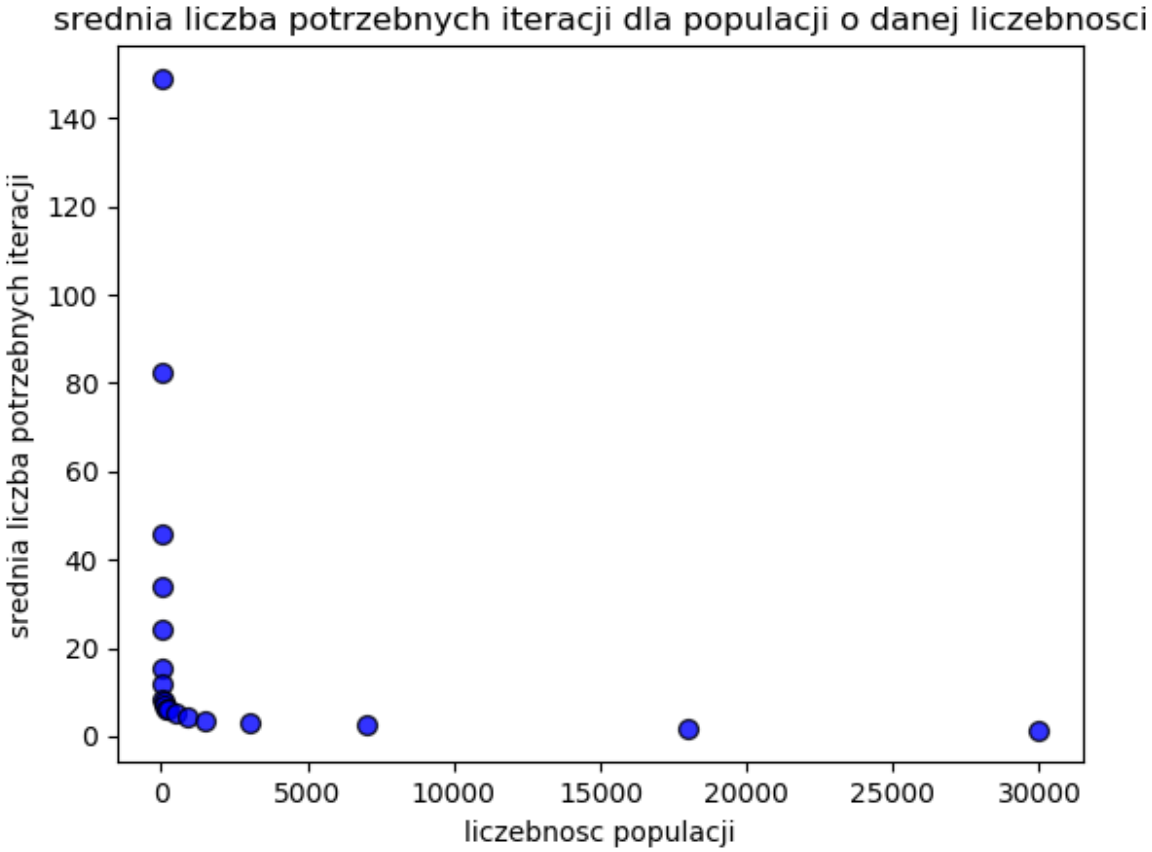
Dla następujących rozmiarów populacji:

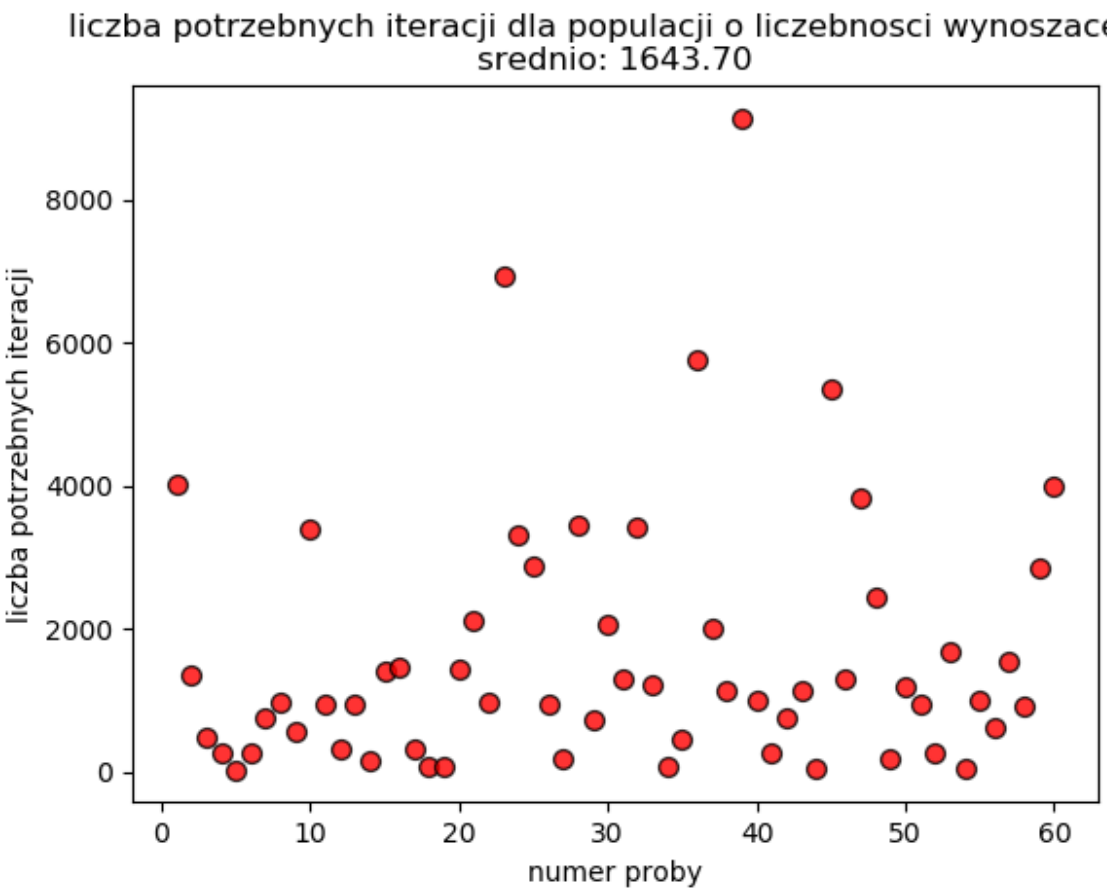
2, 4, 6, 8, 10, 14, 20, 36, 60, 84, 120, 160, 250, 500, 900, 1500, 3000, 7000, 18000, 30000

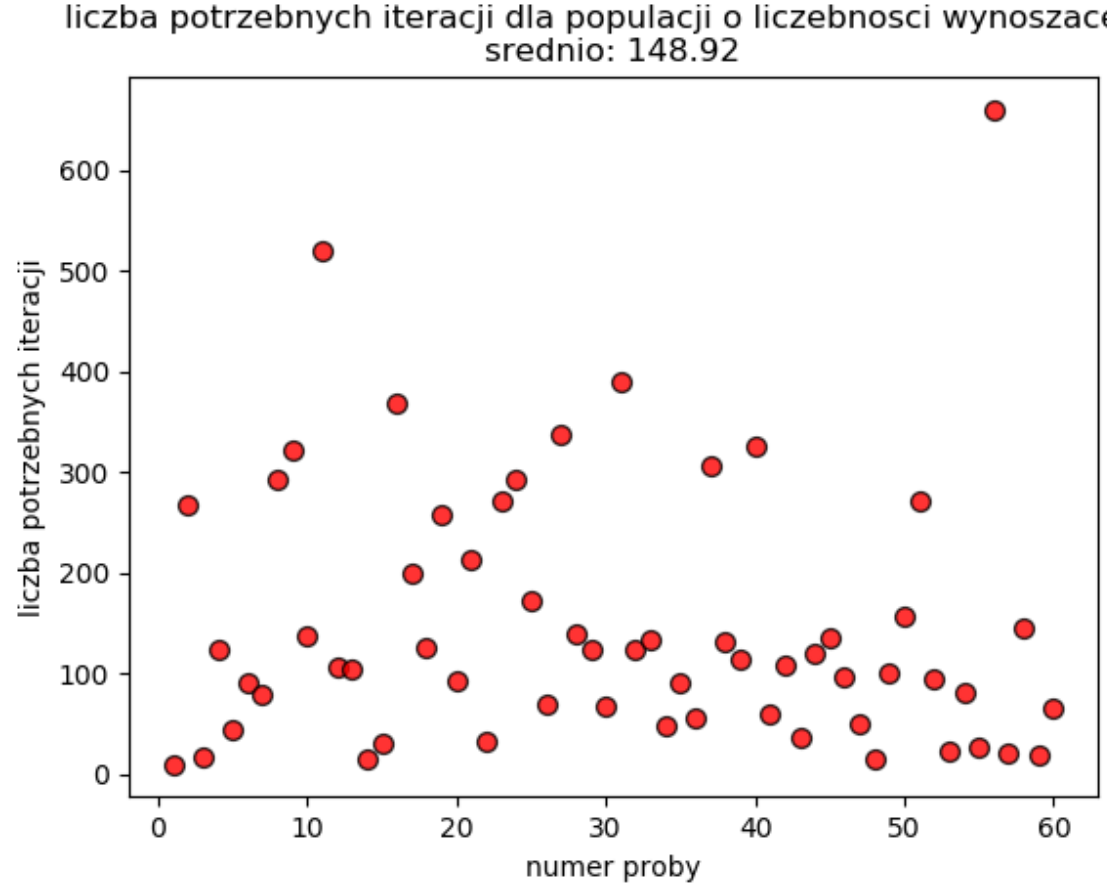
uruchomiłem algorytm po 60 razy. Dla każdego rozmiaru populacji wygenerowałem wykres przedstawiający ile iteracji algorytm potrzebował, aby osiągnąć maksymalną wartość funkcji. Od razu można zauważyć, że dla danego rozmiaru populacji ilość potrzebnych iteracji znacząco się waha. Po wyliczeniu odpowiednich średnich wygenerowałem zbiorczy wykres który doskonale obrazuje wpływ rozmiaru populacji na działanie algorytmu genetycznego. Dla populacji wynoszącej dwóch osobników do uzyskania maksymalnej wartości w skrajnym przypadku potrzeba było ponad 8000 iteracji. Średnio potrzeba było prawie 1650. Niewątpliwie wraz ze wzrostem rozmiaru populacji spada ilość potrzebnych generacji – już dla populacji o liczebności wynoszącej 120 zdarzył się przypadek, że wystarczyła zaledwie jedna, a dla populacji o 30 000 osobnikach nawet średnia wynosi bardzo mało - 1.45. Należy jednak pamiętać, że w przypadku tak ogromnej liczby pomimo 1 iteracji algorytm i tak działa o wiele dłużej. W moim przypadku najoptymalniejsze okazało się wybranie populacji o rozmiarze pomiędzy 120 a 300 osobników – liczba iteracji była stosunkowo niska, a algorytm szybko rozwiązywał zadany mu problem maksymalizacji.

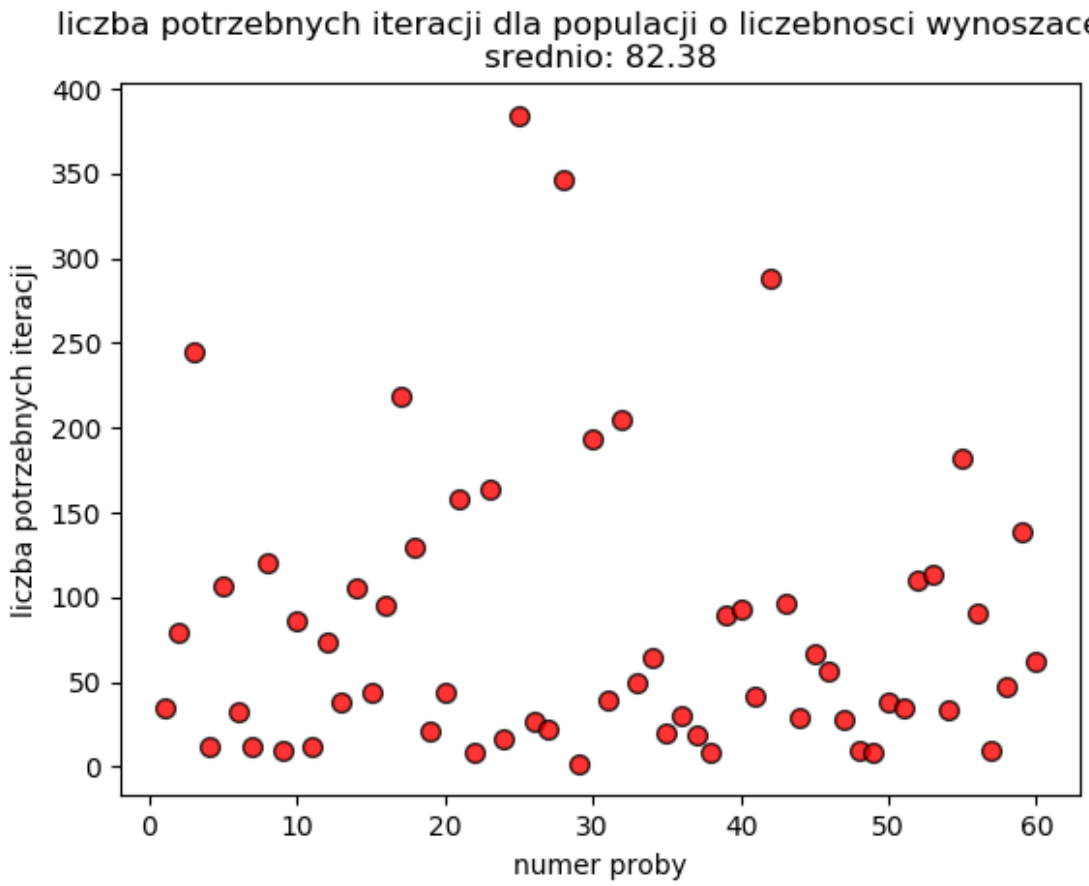
W moim przypadku udało się uzyskać prawidłową wartość maksymalną zadanej funkcji w każdej próbie. W niektórych przypadkach jednak algorytm potrzebował więcej czasu na osiągnięcie oczekiwanego rezultatu. Analizując działanie algorytmu miałem tą przewagę, że wiedziałem jaki jest docelowy wynik. W praktyce jednak w trakcie działania algorytmu nie można stwierdzić, czy aktualna maksymalna wartość jest największą wartością w ogóle, czy może uda się znaleźć jeszcze lepsze rozwiązanie. Dla populacji o dwóch osobnikach zdarzało się, że algorytm już po chwili osiągał docelową wartość. Zdarzyło się również, że przy ograniczeniu liczby iteracji do 1000, algorytm kończył pracę z odnalezioną znacznie mniejszą wartością maksymalną niż największa możliwa.

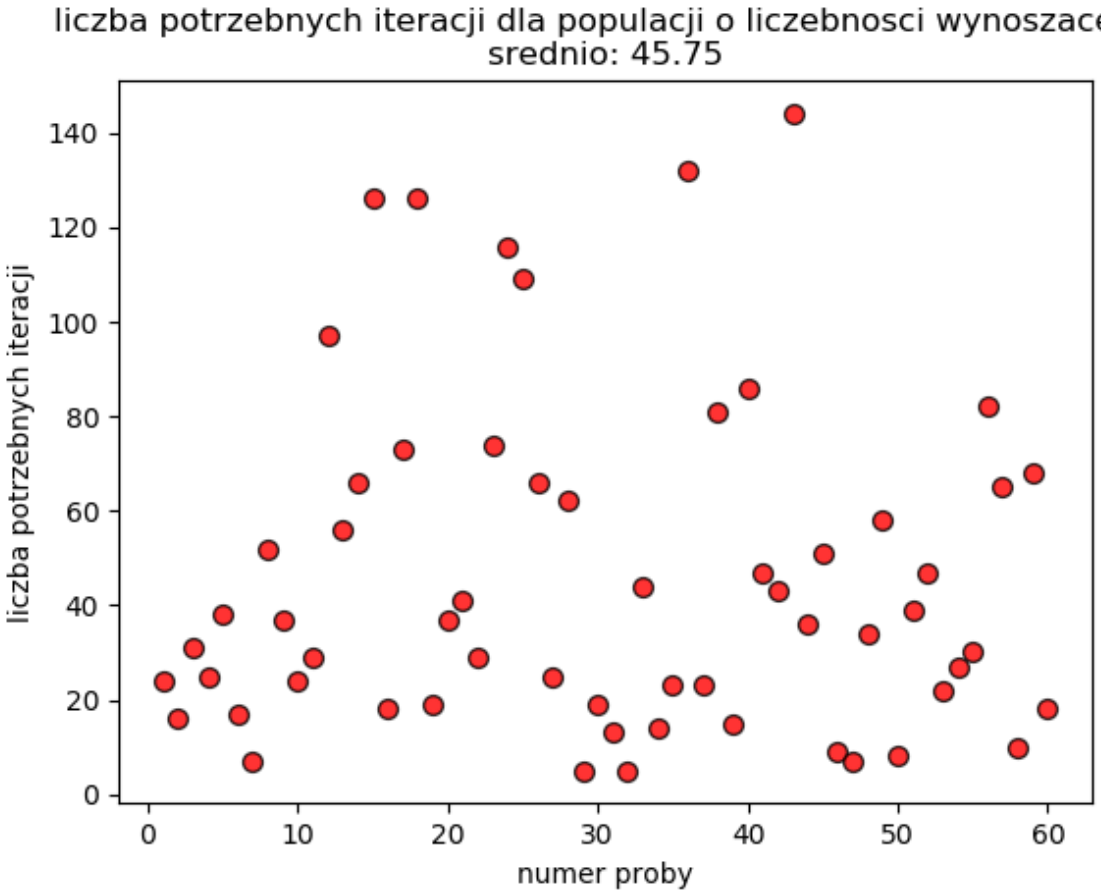
Zgodnie z przeprowadzoną analizą, można wysunąć wnioski, że aby zoptymalizować działanie algorytmu genetycznego w wersji Hollanda dla wektorów binarnych należy rozsądnie dobrać rozmiar populacji – tak, aby algorytm był w stanie przy sensownej liczbie iteracji rozwiązać zadany problem maksymalizacji z faktycznym najlepszym możliwym rezultatem, a jednocześnie – aby czas do tego potrzebny nie wydłużał się na próżno i pozostawał w akceptowalnym zakresie.

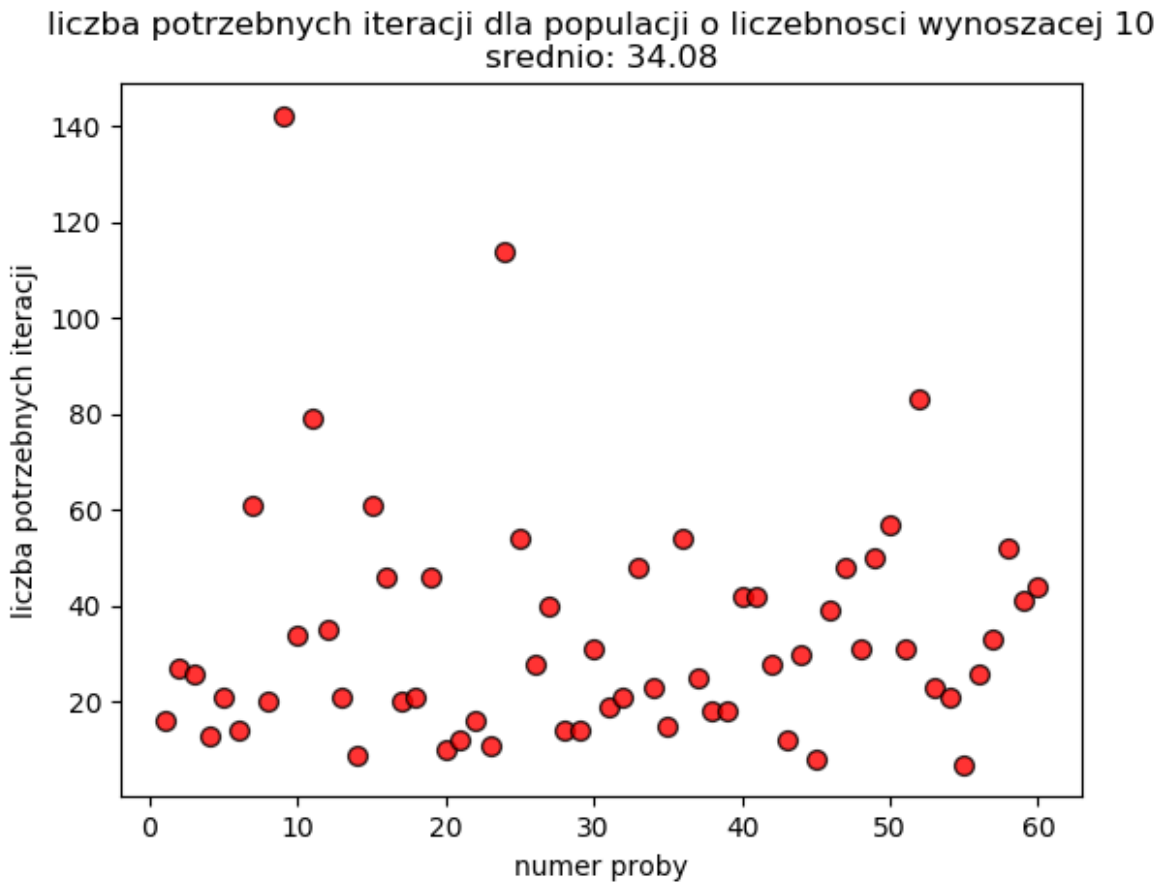

srednia liczba potrzebnych iteracji dla populacji o danej liczebnosci
srednia liczba potrzebnych iteracji
liczebnosc populacji
0
20
40
60
80
100
120
140
0
5000
10000
15000
20000
25000
30000

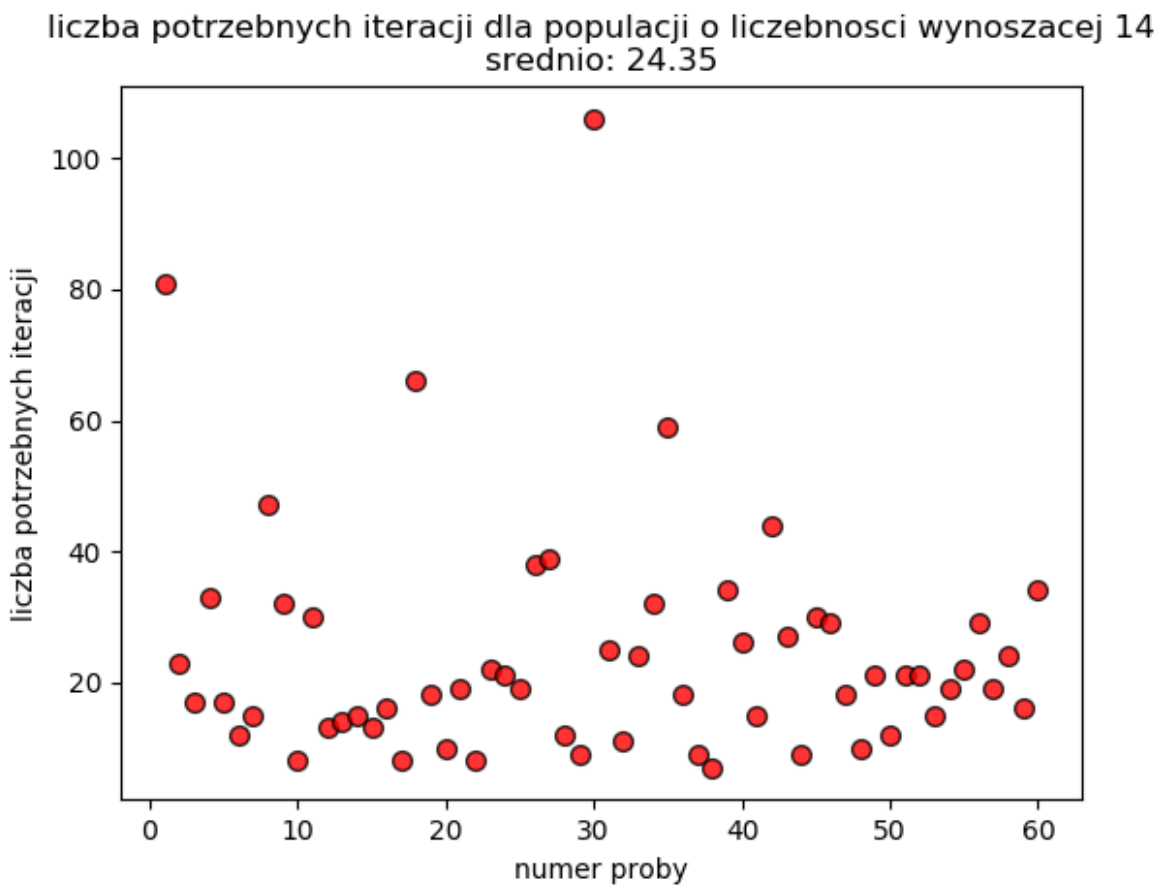

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 2
srednio: 1643.70
liczba potrzebnych iteracji
numer proby
0
2000
4000
6000
8000
0
10
20
30
40
50
60

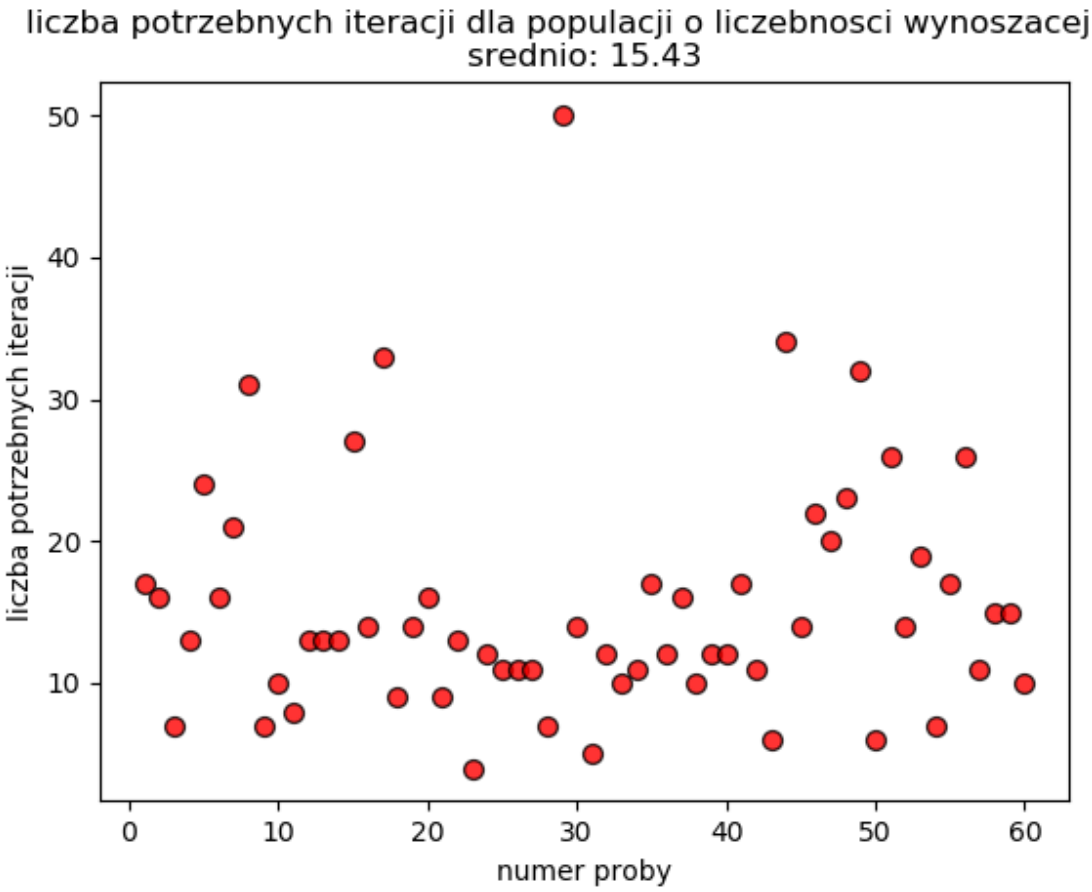

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 4
srednio: 148.92
liczba potrzebnych iteracji
numer proby
0
100
200
300
400
500
600
0
10
20
30
40
50
60

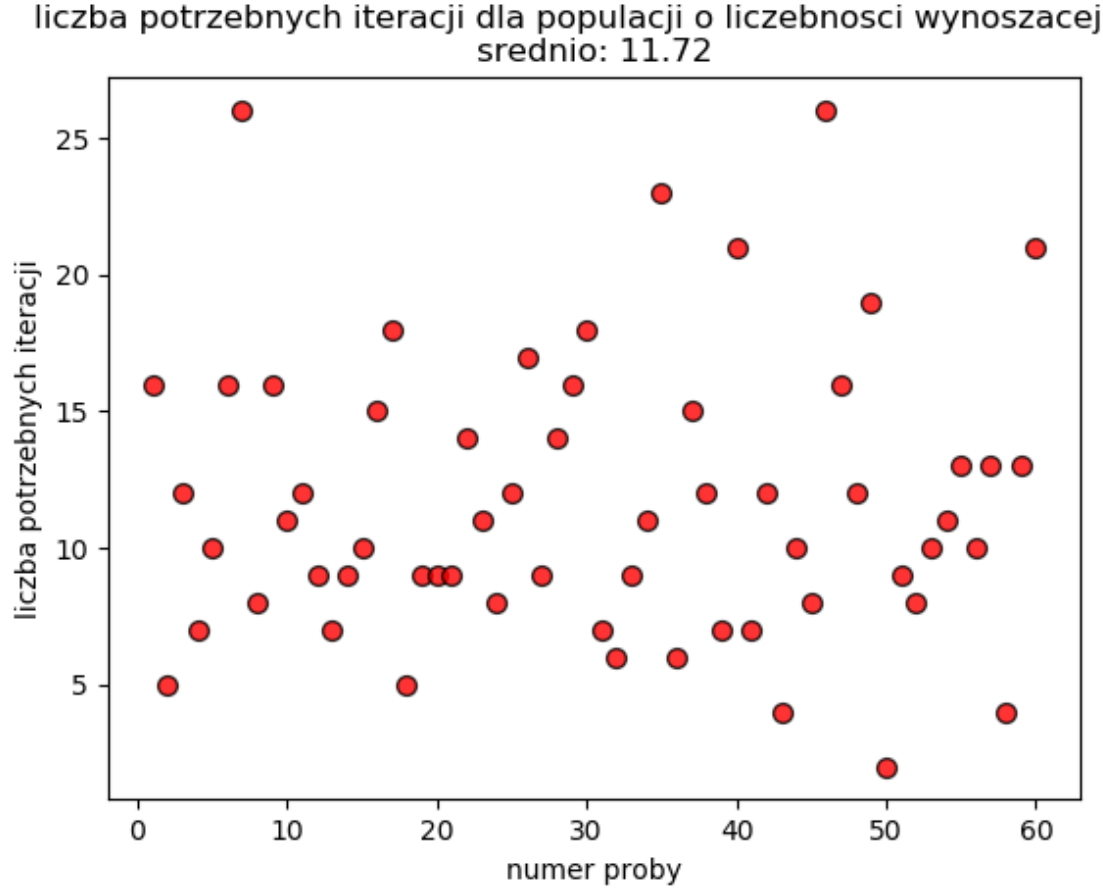

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 6
srednio: 82.38
liczba potrzebnych iteracji
numer proby
0
50
100
150
200
250
300
350
400
0
10
20
30
40
50
60

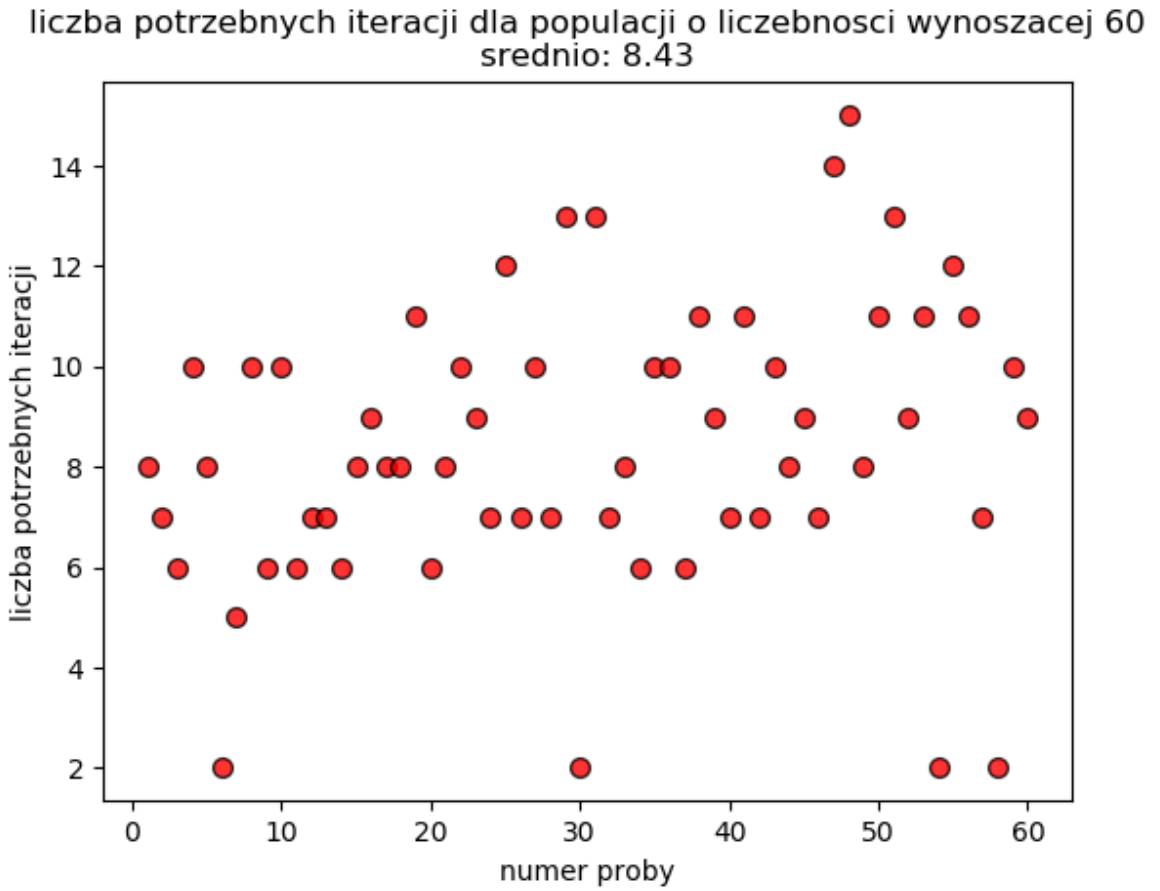

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 8
srednio: 45.75
liczba potrzebnych iteracji
numer proby
0
20
40
60
80
100
120
140
0
10
20
30
40
50
60

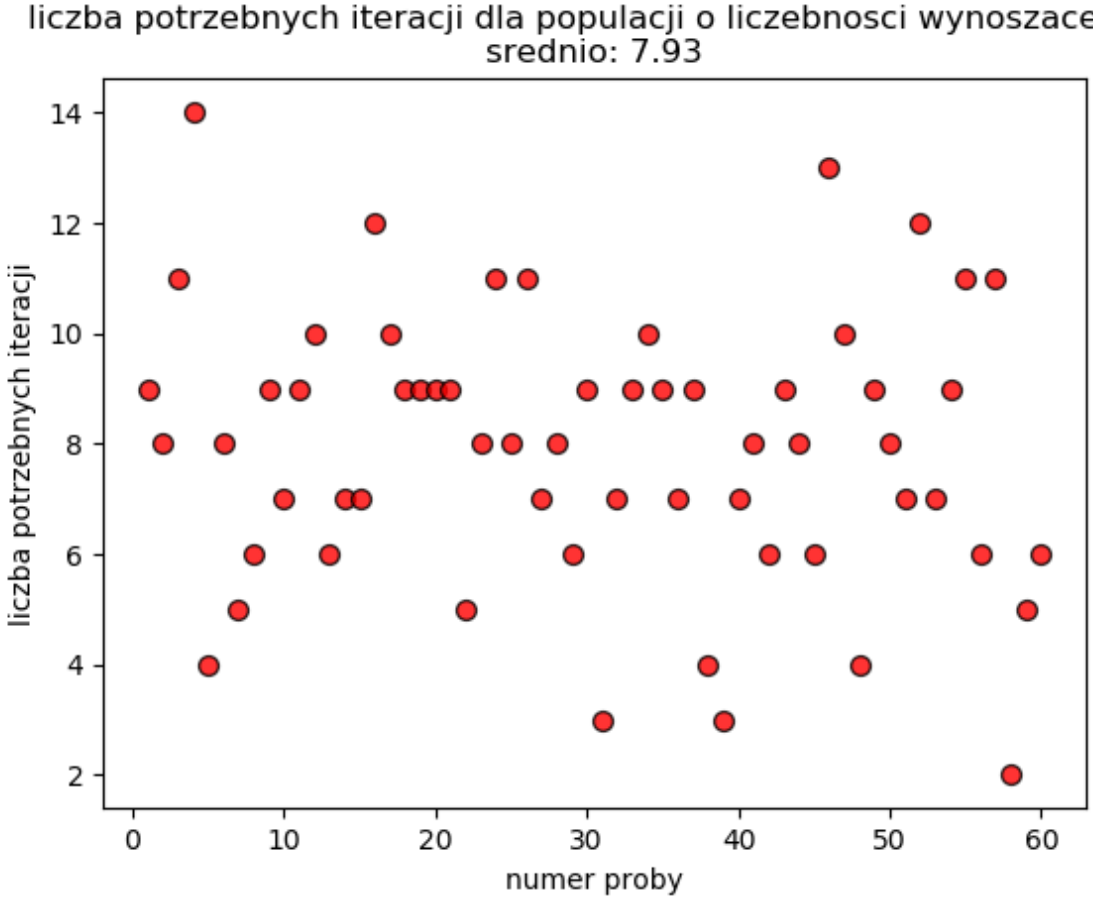

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 10
srednio: 34.08
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
20
40
60
80
100
120
140

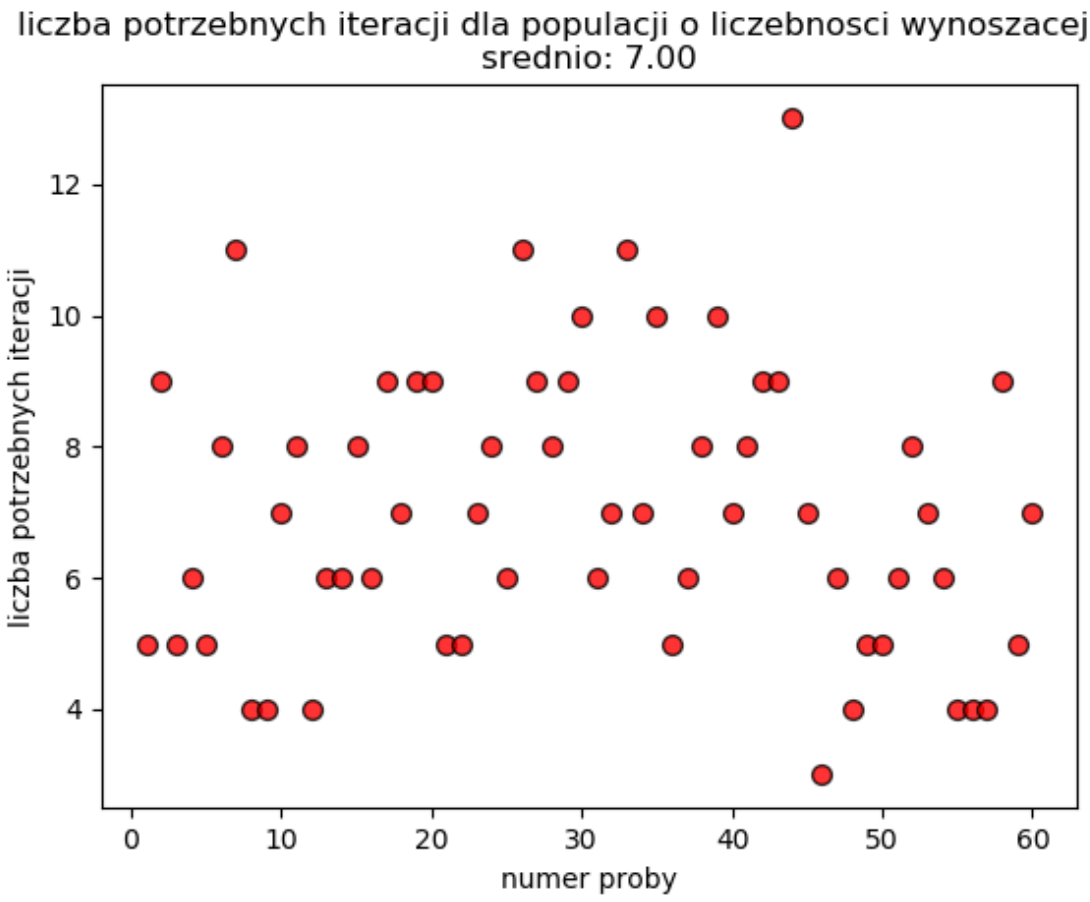

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 14
srednio: 24.35
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
20
40
60
80
100

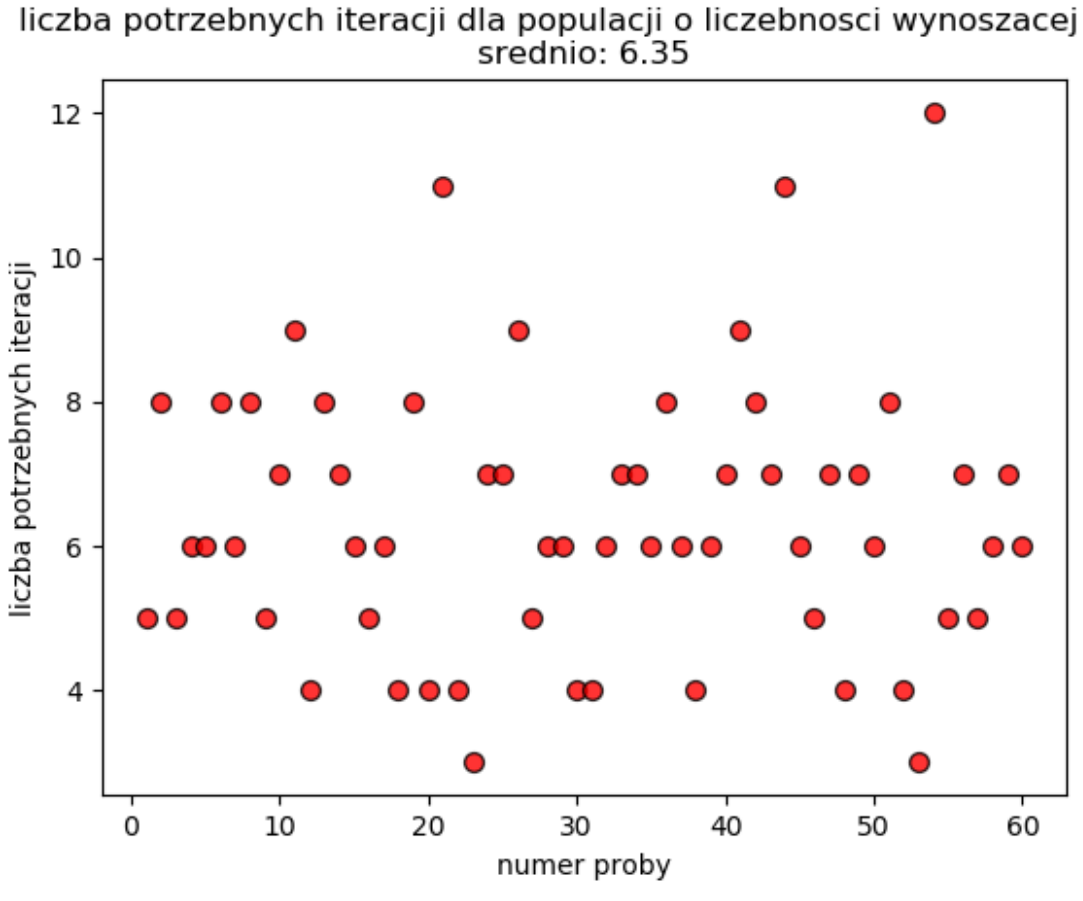

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 20
srednio: 15.43
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60

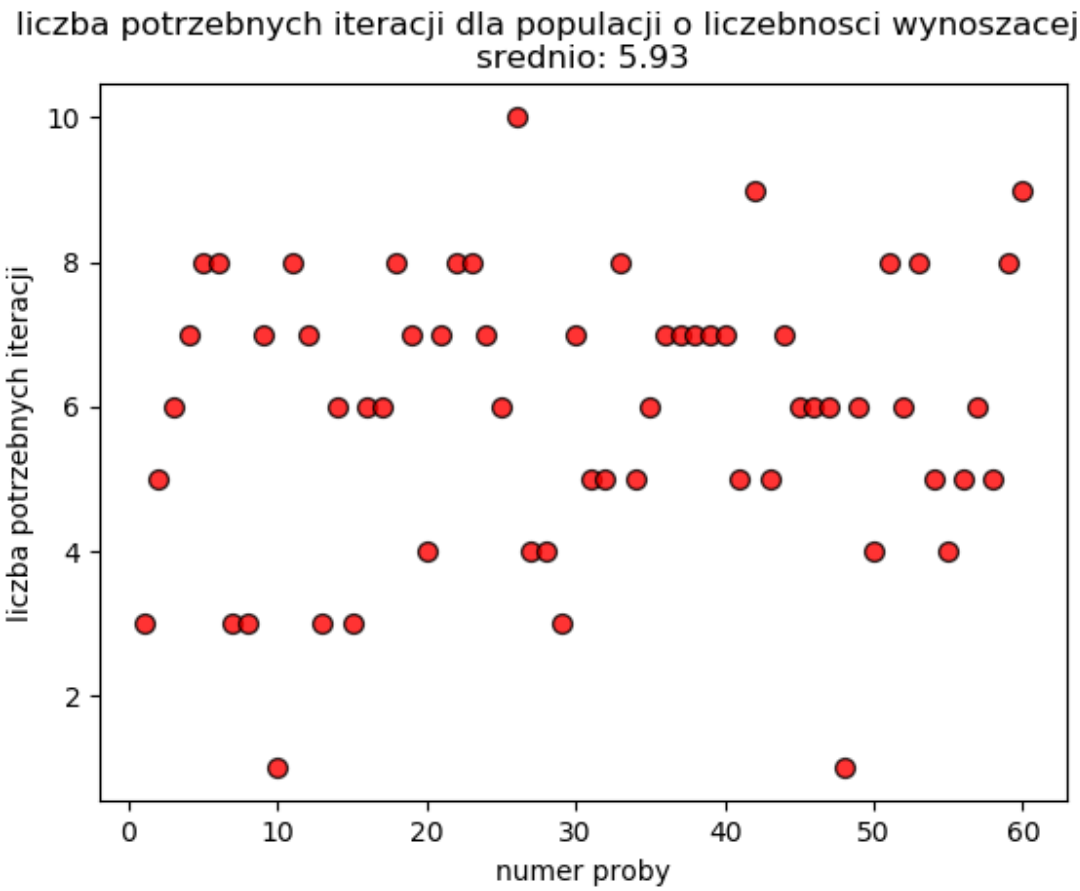

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 36
srednio: 11.72
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60

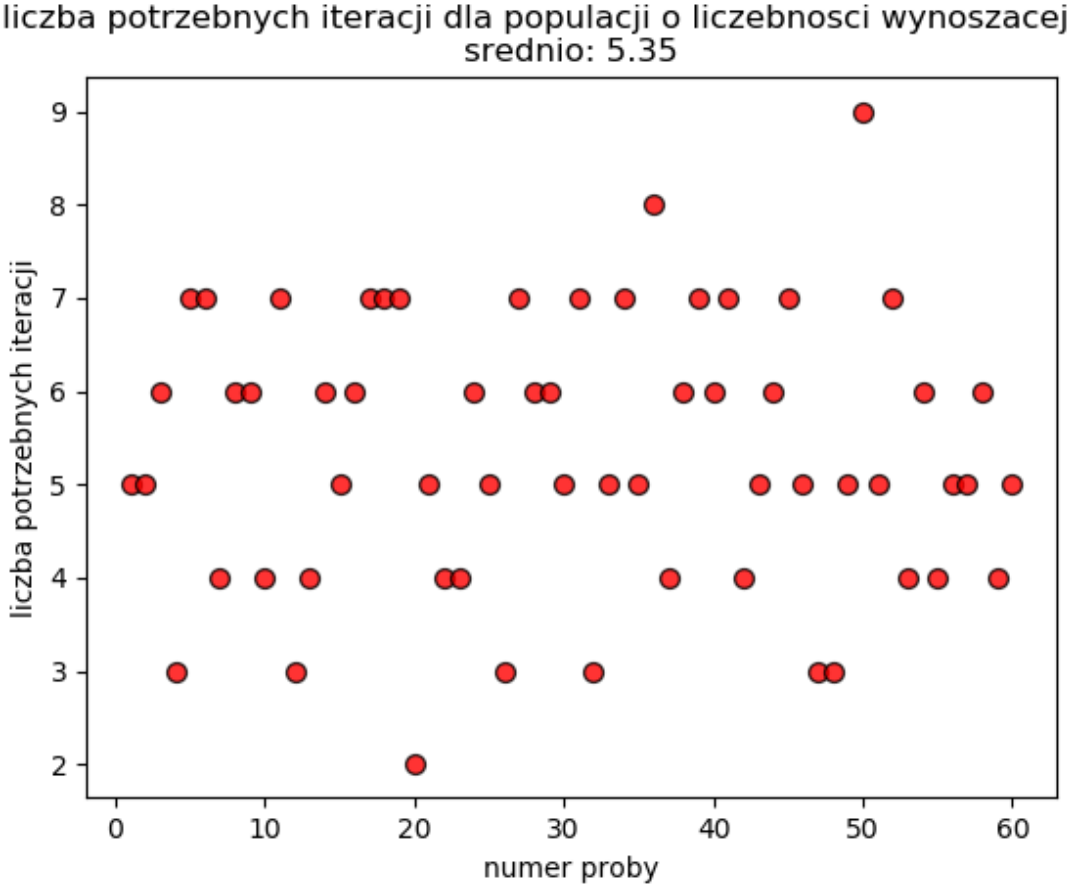

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 60
srednio: 8.43
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
2
4
6
8
10
12
14

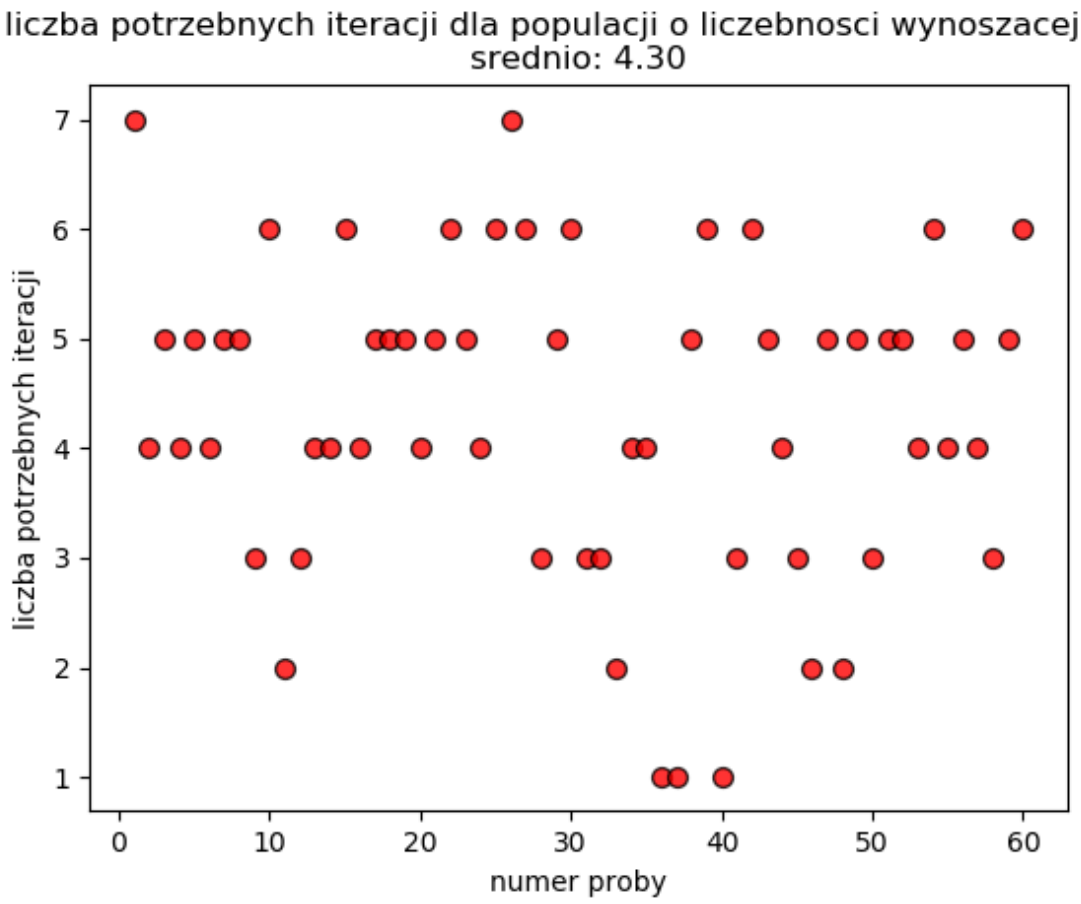

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 84
srednio: 7.93
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
2
4
6
8
10
12
14

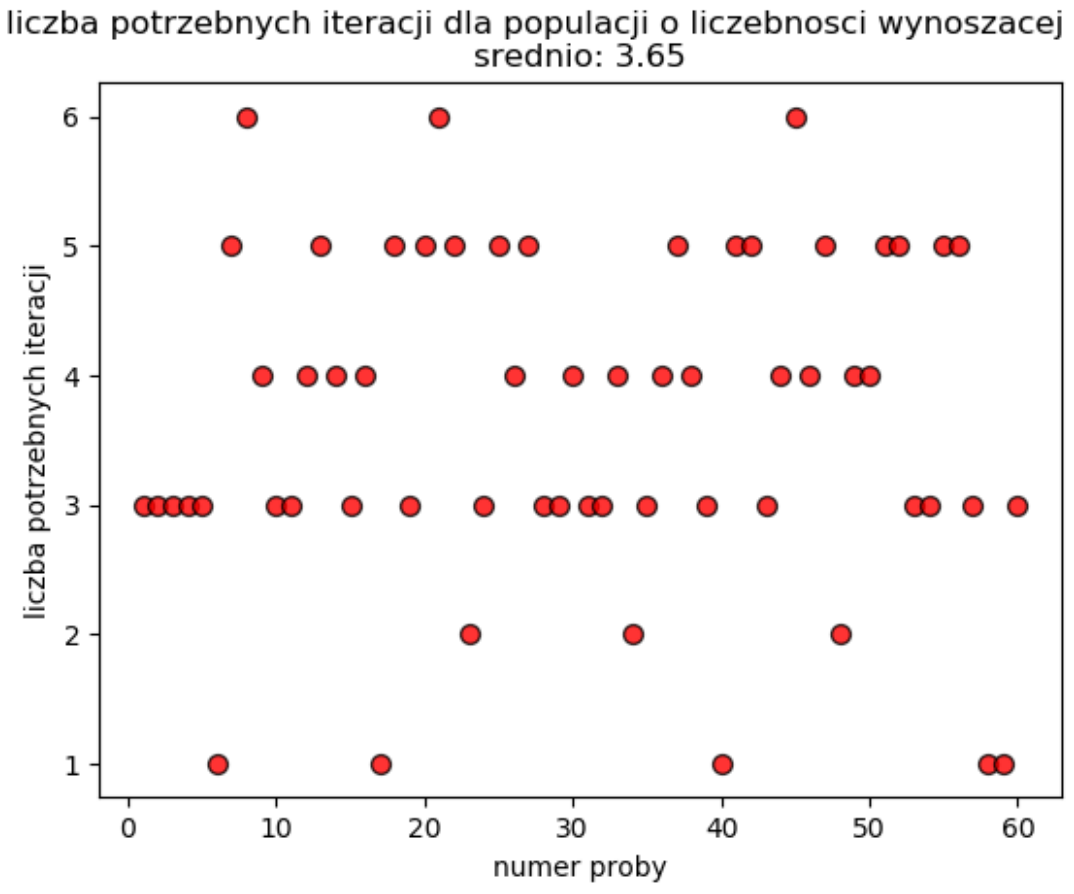

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 120
srednio: 7.00
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
4
6
8
10
12

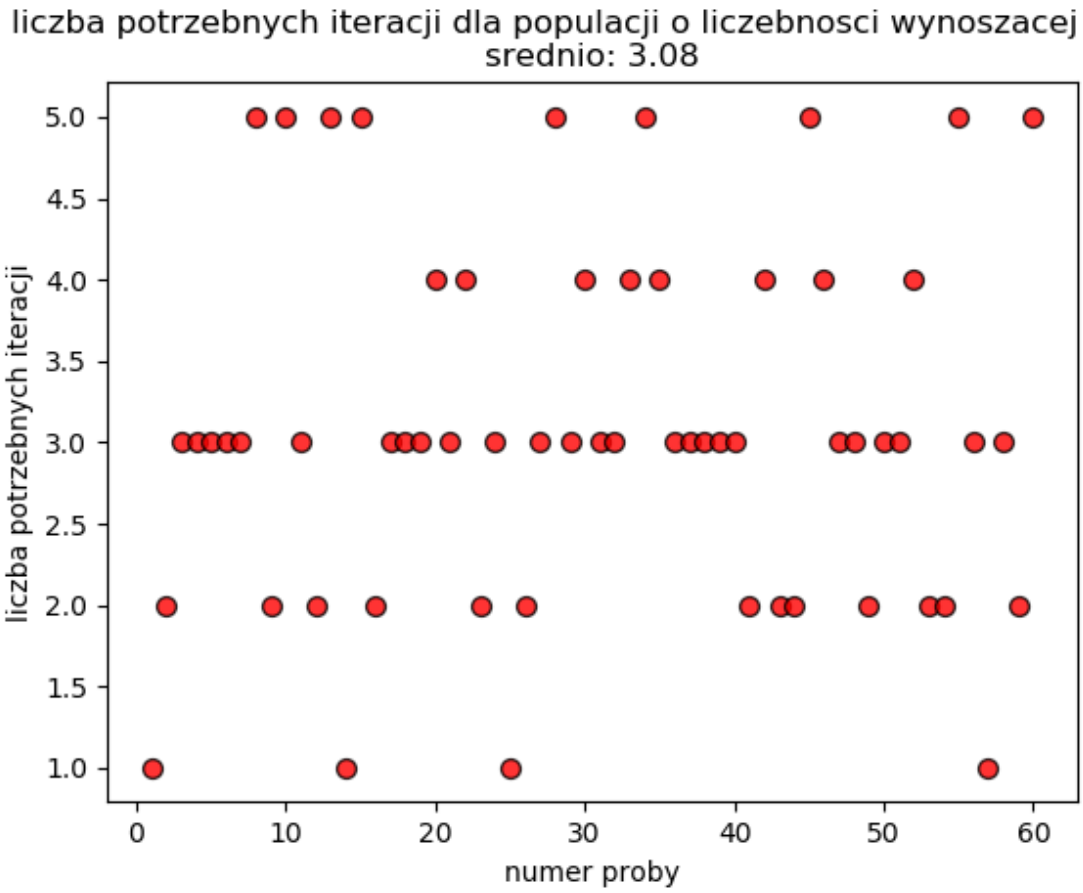

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 160
srednio: 6.35
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
4
6
8
10
12

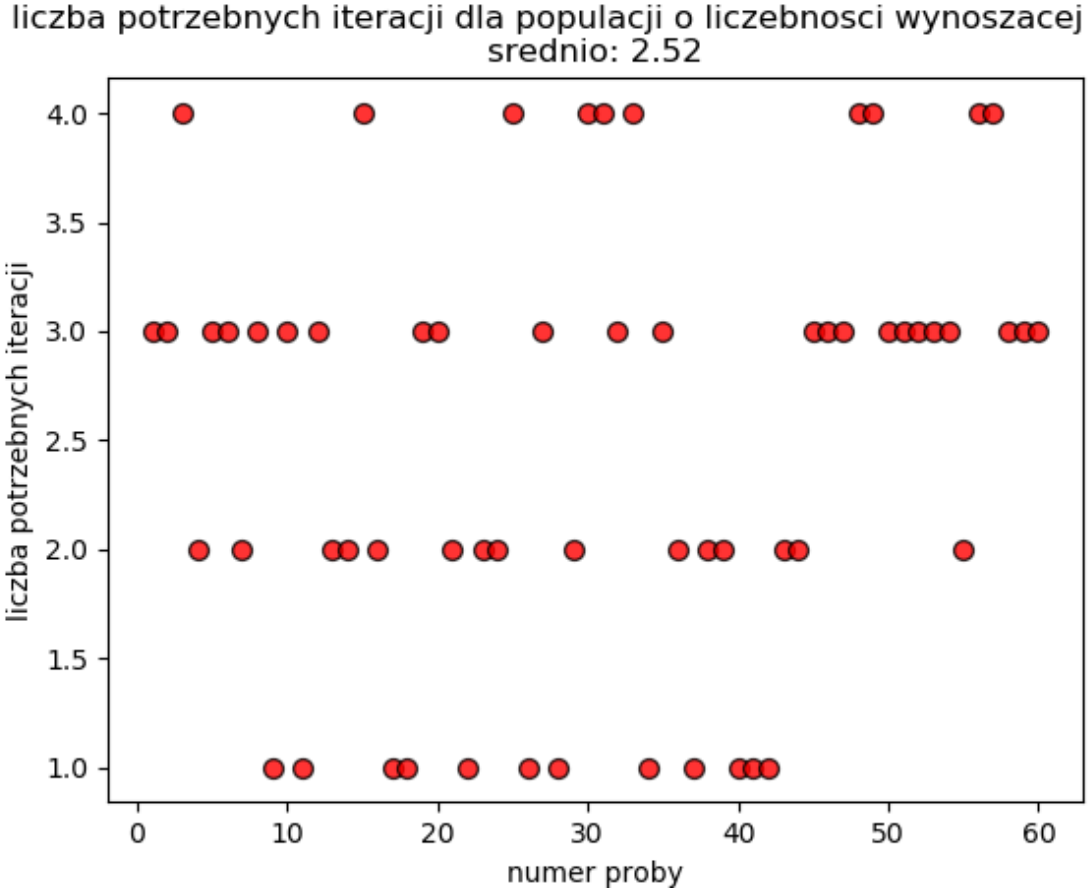

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 250
srednio: 5.93
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
2
4
6
8
10

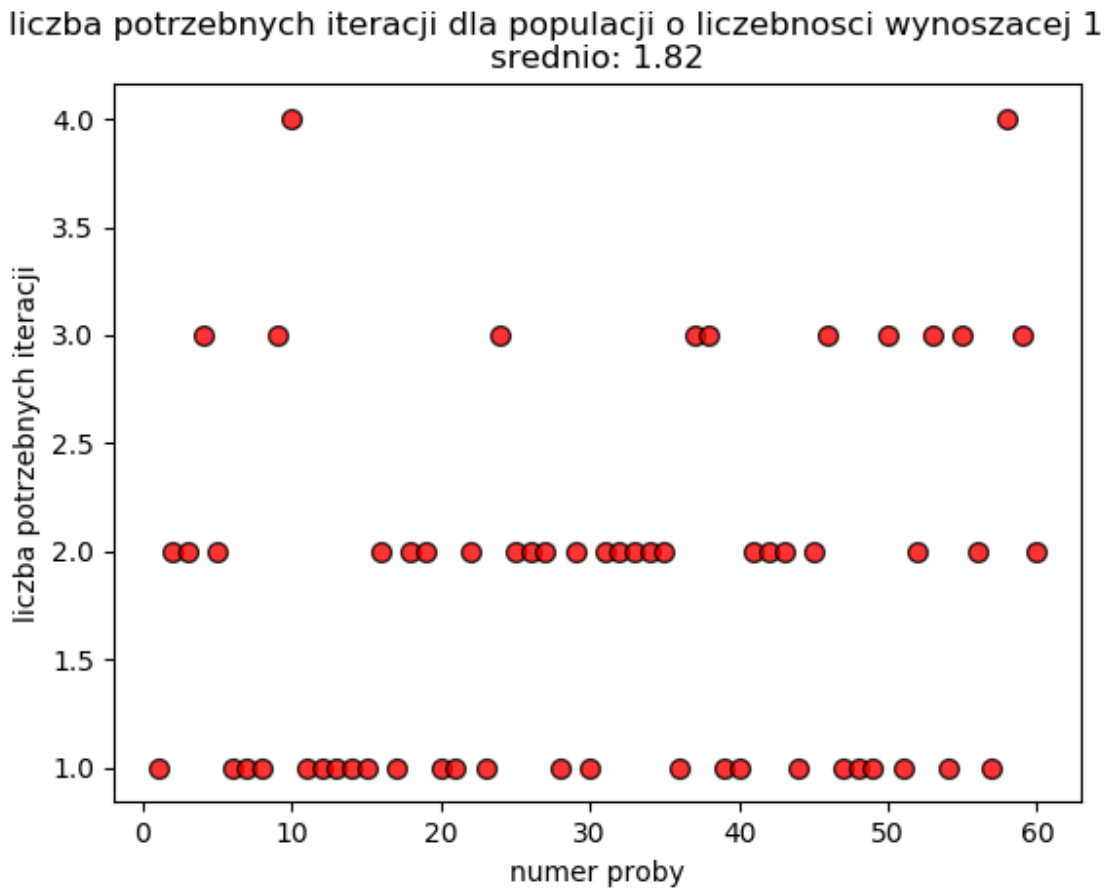

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 500
srednio: 5.35
liczba potrzebnych iteracji
numer proby
0
10
20
30
40
50
60
2
3
4
5
6
7
8
9

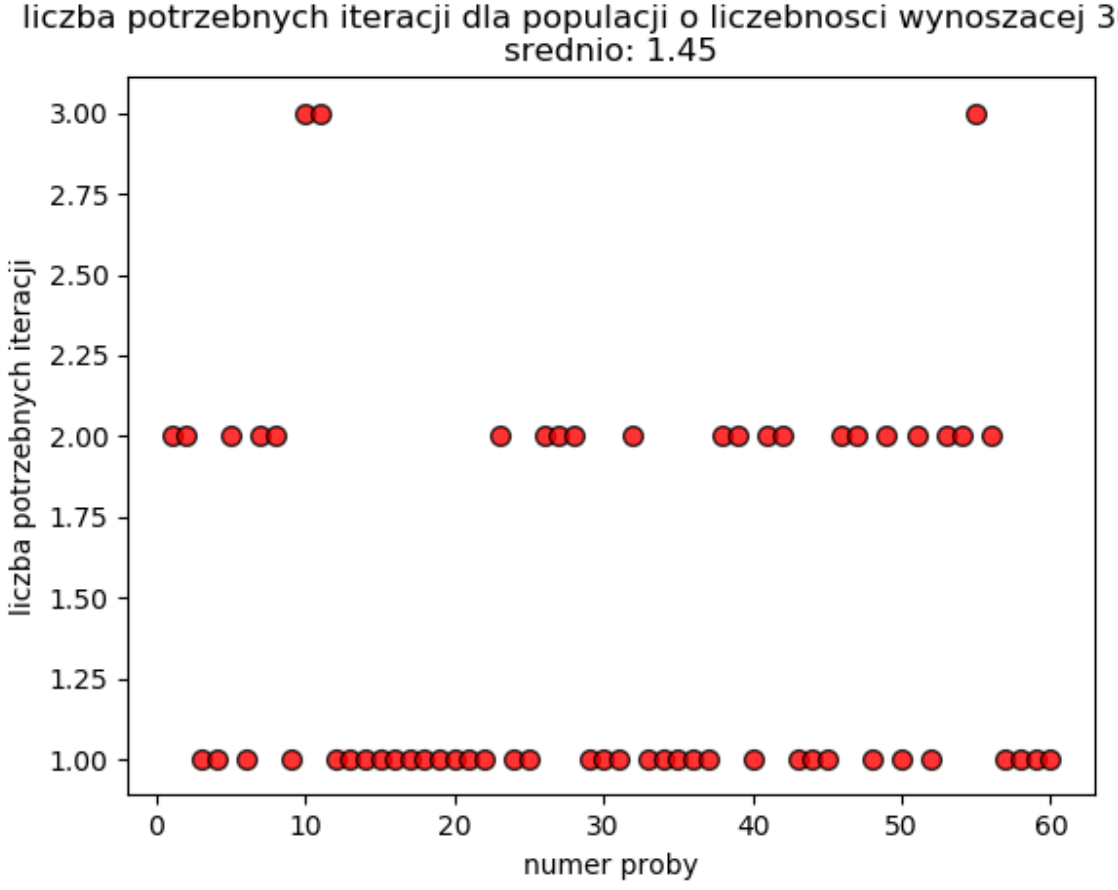

liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 900
srednio: 4.30
liczba potrzebnych iteracji
numer proby
1
2
3
4
5
6
7
0
10
20
30
40
50
60


liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 1500
srednio: 3.65
liczba potrzebnych iteracji
numer proby
1
2
3
4
5
6
0
10
20
30
40
50
60


liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 3000
srednio: 3.08
liczba potrzebnych iteracji
numer proby
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
0
10
20
30
40
50
60


liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 7000
srednio: 2.52
liczba potrzebnych iteracji
numer proby
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
10
20
30
40
50
60


liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 18000
srednio: 1.82
liczba potrzebnych iteracji
numer proby
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10
20
30
40
50
60


liczba potrzebnych iteracji dla populacji o liczebnosci wynoszacej 30000
srednio: 1.45
liczba potrzebnych iteracji
numer proby
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
10
20
30
40
50
60