Wstęp do sztucznej inteligencji

Laboratorium drugie - algorytmy genetyczne i ewolucyjne

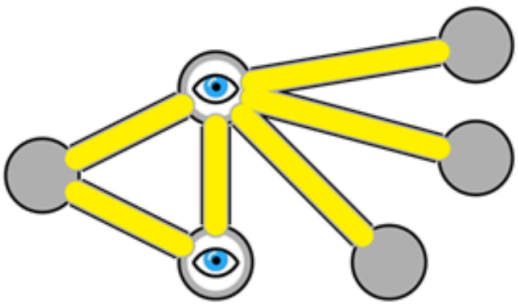
Krystian Kamiński nr 304013

Polecenie:

Tematem drugich ćwiczeń są algorytmy genetyczne i ewolucyjne. Państwa zadaniem będzie zaimplementować klasyczny algorytm ewolucyjny bez krzyżowania, z selekcją turniejową i sukcesją generacyjną. W raporcie należałoby wskazać jak zmiana liczby osobników w populacji wpływa na jakość uzyskanych rozwiązań przy ograniczonym budżecie. Warto również opisać zachowanie algorytmu dla różnych rodzajów danych wejściowych oraz wpływ zmiany parametrów. Przykładowe zbiory danych i/lub ich generatory należy samemu skonstruować na potrzebę zadania.

Osoby z nazwiskami na literę K

W ostatnim czasie doszło do cięć budżetowych w Wolsce i zmniejszono finansowanie policji tego wspaniałego kraju. W mieście X znajduje się n=25 placów (wierzchołki w grafie), z których można dotrzeć do sąsiadujących placów (istnieje krawędź w grafie). Policjant stojący na placu jest w stanie pokryć wszystkie krawędzie biegnące od tego placu do wszystkich jego sąsiadów. Gdzie powinniśmy umieścić policjantów by uzyskać jak największe pokrycie wszystkich ulic. (vertex cover problem)



Sugerowane grafy: graf pełny, graf dwudzielny, graf losowy (graf pełny z usuniętymi 50-70% krawędzi)

Założenia:

Selekcja turniejowa w moim przypadku zakłada istnienie turniejów o stałej liczbie osobników równej dwa.
Mutacja osobnika polega na zamianie pewnego wierzchołka z jednym z jego sąsiadów.
Osobnikiem w populacji jest zbiór wierzchołków w grafie, interpretowany jako zbiór punktów stacjonowania policji.

Spośród proponowanych grafów zdecydowałem się na „graf peny z usunitymi 50-70% krawdzi”

Implementacja:

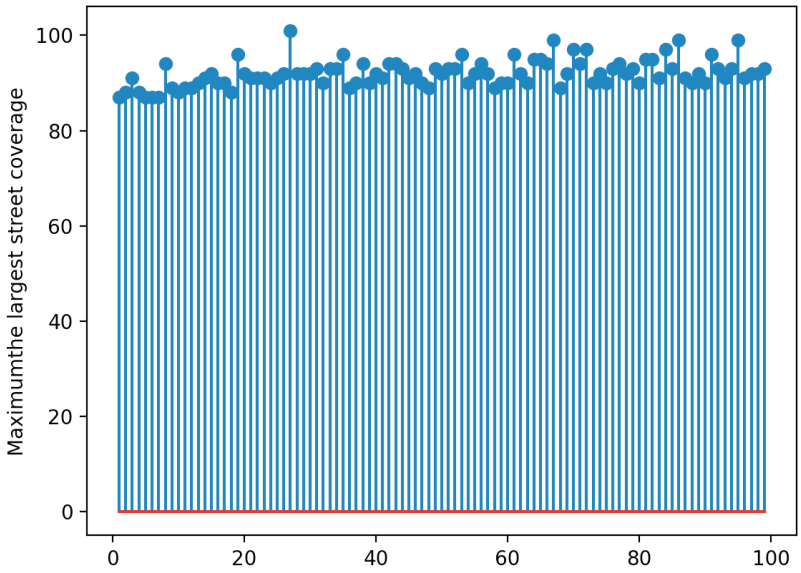
Cały program jest podzielony na następujące funkcje:

- random_graph - ma za zadanie utworzyć graf pełny z usuniętymi 50% krawędzi
- population_mark - zwraca ocenę całej populacji
- find_best - spośród wszystkich ocen zwraca maksymalną, wraz z przypisaniem do wierzchołków grafu
- selection - realizuje selekcję turniejową
- mutation - realizuje mutację
- evolution_algorithm - funkcja realizująca algorytm ewolucyjny zgodny z założeniami zadania
- main - zaimplementowane symulacje działania algorytmu ewolucyjnego wraz z przedstawieniami graficznymi

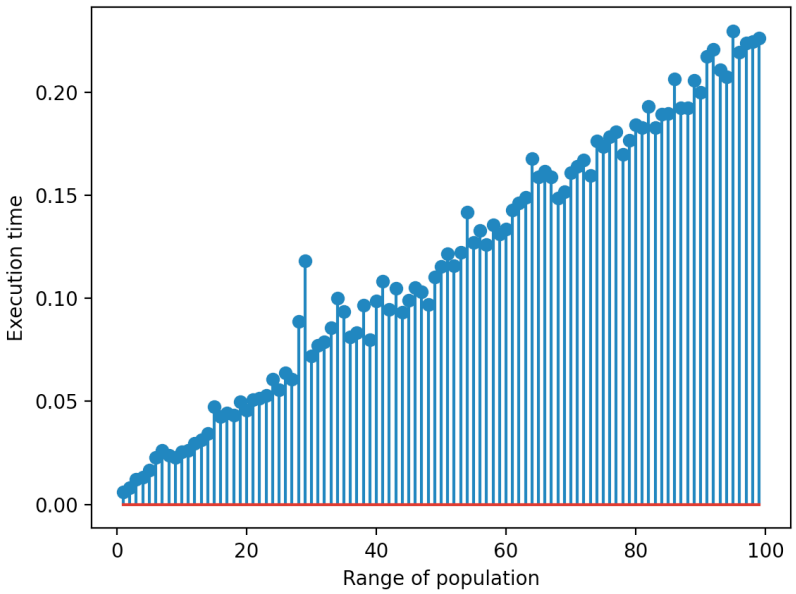
Selekcja turniejowa w moim przypadku zakłada istnienie turniejów o stałej liczbie osobników równej dwa.
Mutacja osobnika polega na zamianie pewnego wierzchołka z jednym z jego sąsiadów.
Osobnikiem w populacji jest zbiór wierzchołków w grafie, interpretowany jako zbiór punktów stacjonowania policji.
Spośród proponowanych grafów zdecydowałem się na „graf peny z usunitymi 50-70% krawdzi”

Wyniki:

Możemy zauważyć wzrost skuteczności maksymalizacji naszego zadania dla niewielkich liczności populacji.
Jednak dla większych liczności wartość maksimum już praktycznie się nie różni.

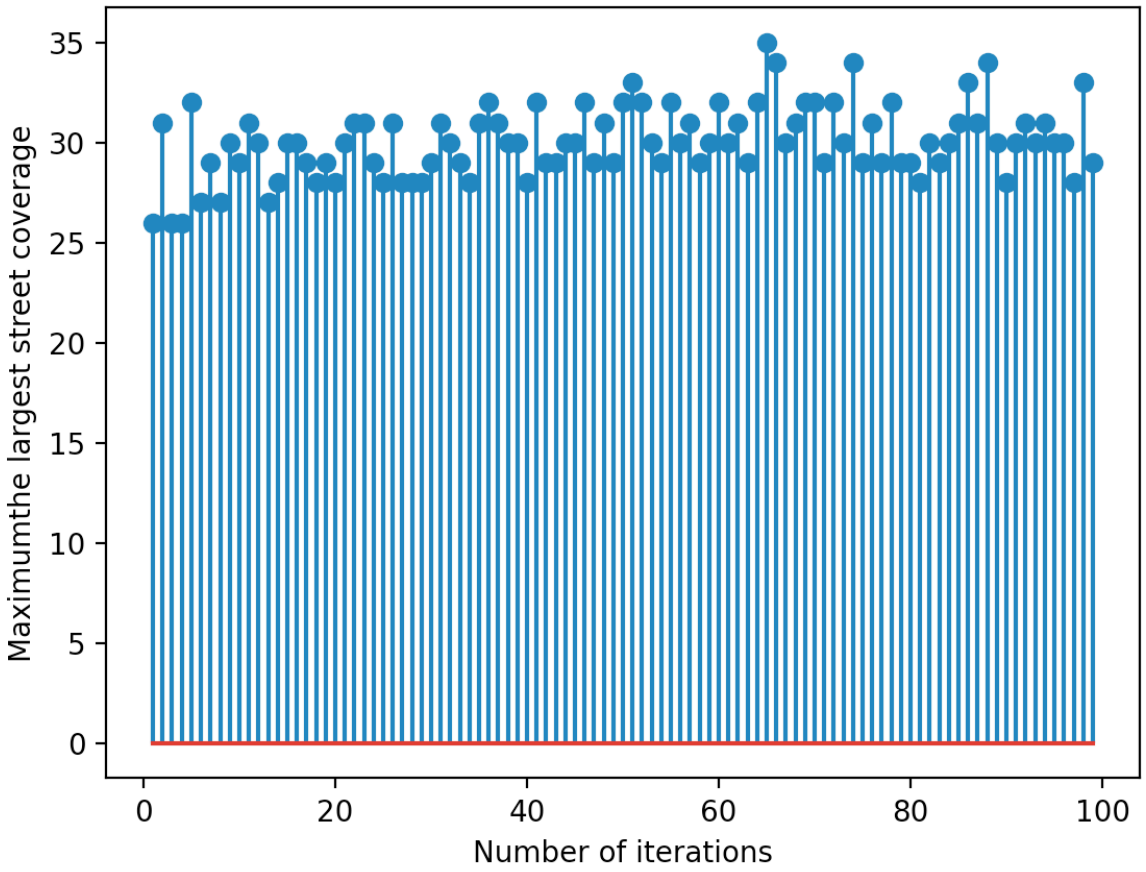


Otrzymana średnia wartość to 92.5



Otrzymana średnia wartość to 0.17 s

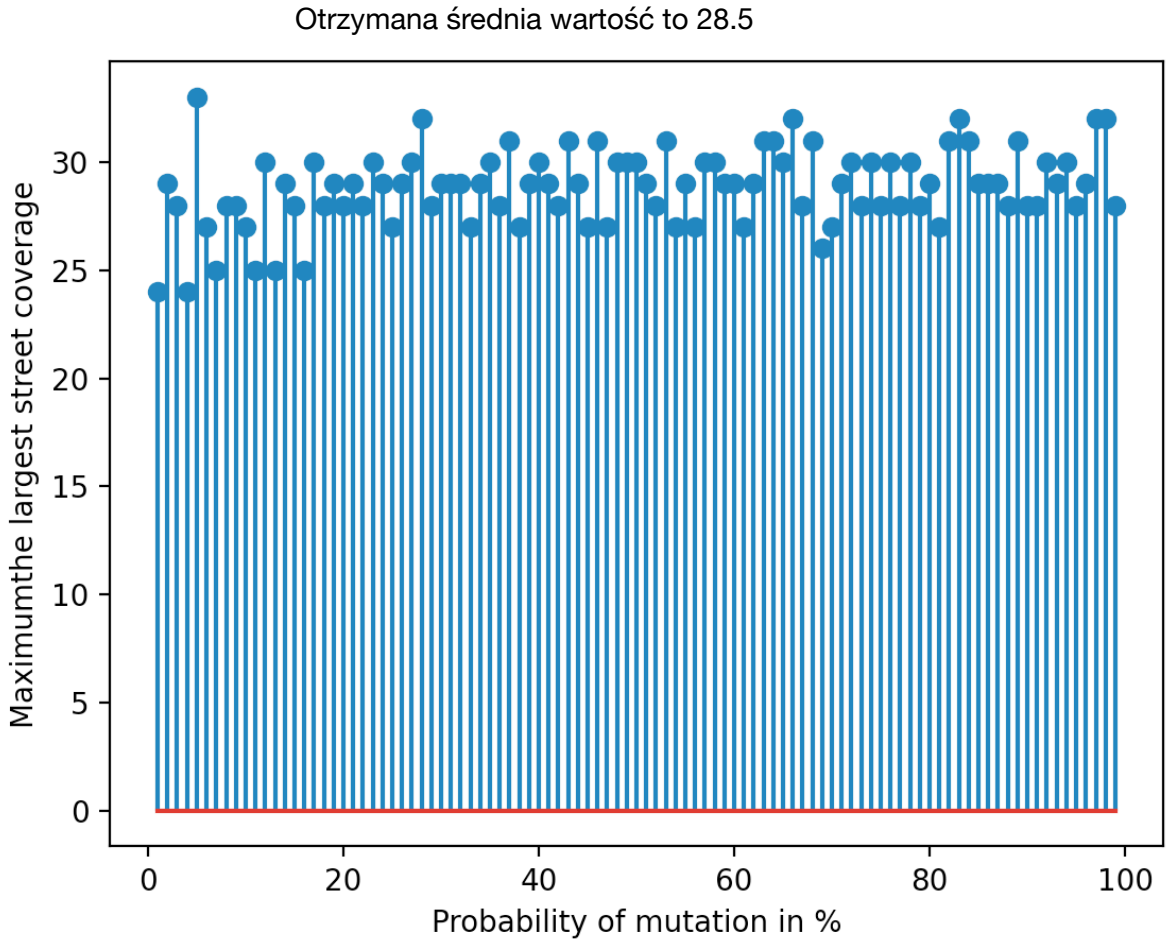
Otrzymana średnia wartość to 30.1



Czas wykonywania algorytmu ewolucyjnego jest wprost proporcjonalny do liczności populacji

Na kolejnym wykresie widzimy, że wraz ze wzrostem liczby pokoleń w algorytmie ewolucyjnym, otrzymujemy coraz bardziej satysfakcjonujący nas rezultat pokrycia ulic.

Analogicznie jak w poprzednim przypadku, wraz ze wzrostem prawdopodobieństwa mutacji wzrasta wartość wykrytego maksimum.



Ze względu na dosyć małą liczbę wierzchołków równa 25, maksimum lokalne jest znajdowane bardzo szybko, przez co nawet dla różnych wartości parametrów, t_max, p_mut, wykryte wartości maksimum różnią się nieznacznie.

Wykres przedstawia wartość maksima dla danej liczby dostępnych policjantów.
Wykres przedstawia przybliżenie funkcji max = n*(n-1)/4