Genetic Algorithm

Opis, implementacja i zastosowanie w problemie komiwojażera

Wstęp

Algorytm genetyczny (*ang. Genetic Algorithm*) jest metodą optymalizacji inspirowaną biologicznym procesem ewolucji. Wywodzi się z prac Johna Hollanda z lat 70. XX wieku i należy do szerszej rodziny algorytmów ewolucyjnych. [1]

Idea metody polega na iteracyjnym ulepszaniu populacji osobników (potencjalnych rozwiązań) poprzez operacje imitujące zjawiska biologiczne, takie jak: selekcja, krzyżowanie oraz mutacja. Dzięki temu osobniki lepiej przystosowane mają większą szansę na przekazanie swoich cech potomstwu, co docelowo prowadzi do stopniowej poprawy jakości rozwiązań.

1. Opis algorytmu

Algorytm genetyczny składa się z następujących kroków:

- **Inicjalizacja** – generowanie początkowej populacji osobników (potencjalnych rozwiązań), zazwyczaj w sposób losowy, z uwzględnieniem dziedziny problemu.
- **Ocena** – wyznaczanie wartości funkcji przystosowania (*ang. fitness function*) dla każdego osobnika w populacji, która stanowi miarę jakości reprezentowanego przez niego rozwiązania.
- **Selekcja** – wybór osobników przeznaczonych do reprodukcji na podstawie ich wartości przystosowania.
- **Krzyżowanie** – wymiana fragmentów materiału genetycznego pomiędzy dwojgiem rodziców w celu wytworzenia potomstwa dziedziczącego cechy obojga.
- **Mutacja** – losowa modyfikacja genotypu potomka z określonym prawdopodobieństwem, umożliwiająca eksplorację przestrzeni rozwiązań oraz zapobiegająca przedwczesnej zbieżności algorytmu do optimum lokalnego i stagnacji.
- **Sukcesja** – formowanie nowej populacji poprzez zastąpienie części lub wszystkich osobników pokolenia rodzicielskiego nowymi potomkami.
- **Kryterium stopu** – algorytm iteruje kroki od 2 do 6 aż do spełnienia przyjętego kryterium zakończenia, którym może być osiągnięcie maksymalnej liczby generacji, stagnacja wartości funkcji przystosowania lub znalezienie rozwiązania spełniającego wymagania jakościowe.

[1] Wikipedia - Algorytm genetyczny

2. Zastosowanie w problemie komiwojażera

Problem komiwojażera, znany również jako problem wędrownego sprzedawcy (*ang. Traveling Salesman Problem*, TSP), polega na wyznaczeniu najkrótszej trasy przechodzącej przez każde z n zadanych miast dokładnie jeden raz i powracającej do punktu początkowego. Główną trudność stanowi wzrost liczby możliwych tras wraz z liczbą miast – dla symetrycznej wersji problemu[2] wynosi ona $\frac{(n-1)!}{2}$, co czyni pełny przegląd wszystkich rozwiązań niepraktycznym już przy umiarkowanych wartościach n. Właśnie dlatego algorytm genetyczny może stanowić atrakcyjne podejście do tego problemu.

Populacja początkowa składa się z osobników, których genotypem jest n-elementowa permutacja liczb od 1 do n. Każda liczba reprezentuje konkretne miasto, a kolejność elementów określa kolejność ich odwiedzania, np.:

$$[2, 5, 1, 4, \ldots, n] \;\rightarrow\; \text{trasa: } 2 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow \ldots \rightarrow n \rightarrow 2$$

Funkcja przystosowania dla osobnika x_i przyjmuje postać odwrotności długości reprezentowanej trasy:

$$f(x_i) = \frac{1}{D(x_i)}, \tag{1}$$

dzięki czemu krótsze trasy uzyskują wyższą wartość przystosowania, a algorytm dąży do jej maksymalizacji.

Selekcja realizowana jest metodą ruletkową, w której prawdopodobieństwo wyboru osobnika jest proporcjonalne do jego wartości funkcji przystosowania:

$$p(x_i) = \frac{f(x_i)}{\sum_{j=1}^{|P|} f(x_j)}. \tag{2}$$

Ruletka jest „kręcona“ tyle razy, ile osobników ma zostać wybranych do reprodukcji. Należy wspomnieć, że metoda ta jest wrażliwa na rozkład wartości funkcji przystosowania – duże różnice mogą prowadzić do dominacji pojedynczych osobników.

Krzyżowanie realizowane jest za pomocą operatora *Order Crossover* (OX), który zachowuje względną kolejność genów z obu rodziców – własność kluczową w TSP, ponieważ klasyczne operatory jednopunktowe lub dwupunktowe mogą generować niepoprawne trasy zawierające powtórzenia lub pominięcia miast. Procedura OX przebiega następująco:

1. Losowo wybierane są dwa punkty podziału chromosomu, wyznaczające segment do skopiowania.
2. Segment z pierwszego rodzica kopiowany jest do potomka na odpowiadających pozycjach.
3. Pozostałe pozycje uzupełniane są genami z drugiego rodzica w kolejności ich występowania, z pominięciem genów już obecnych w potomku.

Mutacja polega na losowej zamianie pozycji dwóch genów (miast) w chromosomie potomka (*ang. swap mutation*).

[2]Tj. przy założeniu, że dla dowolnych miast X i Y zachodzi $D(X, Y) = D(Y, X)$.

3. Implementacja

Algorytm zaimplementowano w języku Python z wykorzystaniem biblioteki `NumPy` do operacji wektorowych.

```
def genetic_algorithm(cost_fun, p_0, p_m, p_c, t_max, coords):
    t = 0
    fitnesses = fitness_fun(cost_fun, p_0, coords)
    current_population = p_0

    x_best, o_best = find_best(p_0, fitnesses)
    while t < t_max:

        children = reproduction(current_population, fitnesses)
        new_population = cross_and_mute(children, p_m, p_c)
        fitnesses = fitness_fun(cost_fun, new_population, coords)

        x_loc_best, o_loc_best = find_best(new_population, fitnesses)
        if o_loc_best >= o_best:
            o_best = o_loc_best
            x_best = x_loc_best

        current_population = new_population
        t += 1

    return x_best, o_best
```

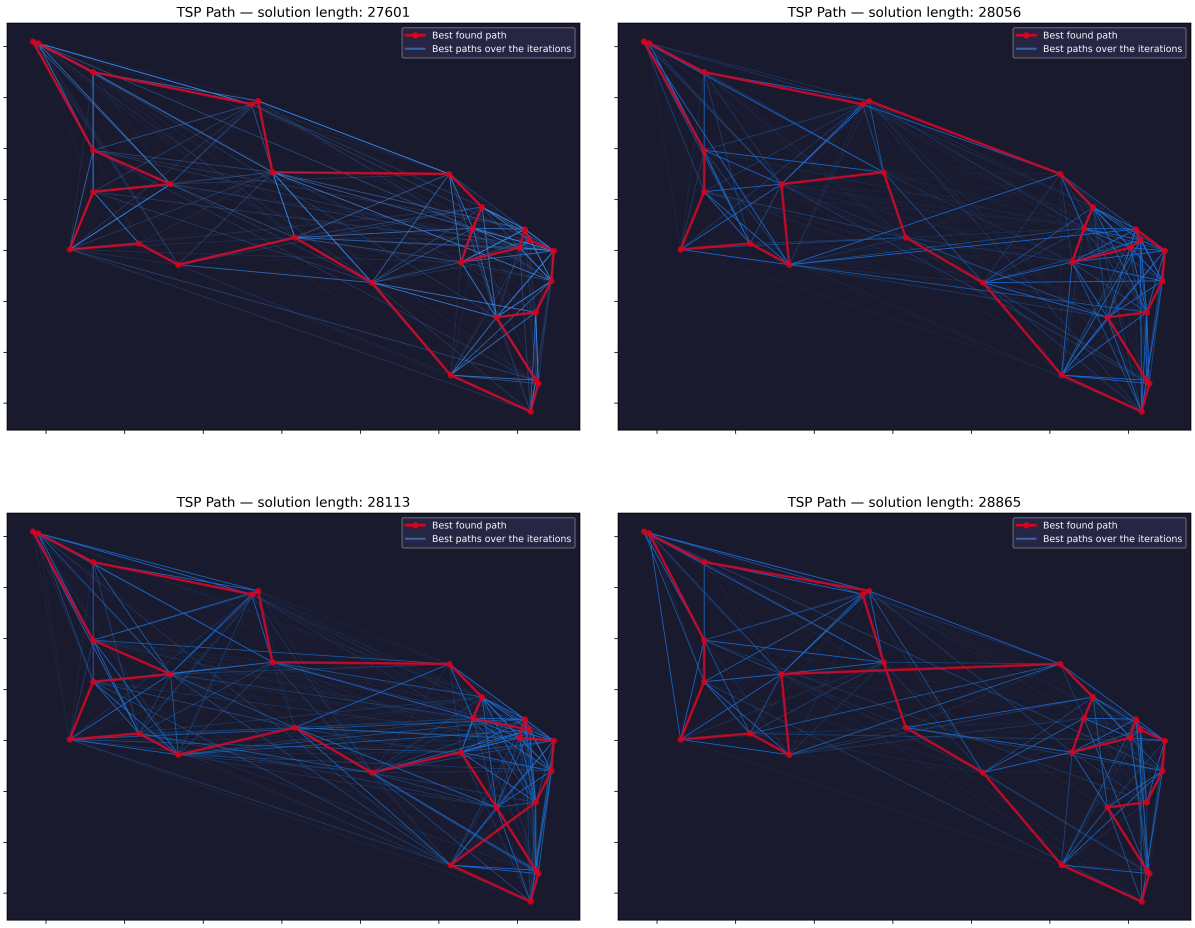
Funkcja przyjmuje argumenty:

- `cost_fun` – funkcja kosztu; w rozważanym przypadku oblicza długość trasy s_i i służy do wyznaczania wartości funkcji przystosowania,
- `p_0` – populacja początkowa P_0,
- `p_m` – prawdopodobieństwo mutacji $p_m \in [0, 1]$,
- `p_c` – prawdopodobieństwo krzyżowania $p_c \in [0, 1]$,
- `t_max` – maksymalna liczba iteracji,
- `coords` – lista przechowująca współrzędne miast.

Funkcja zwraca najlepszą znalezioną ścieżkę `x_best` oraz jej ocenę `o_best`.

4. Algorytm w działaniu

Przykładowe przebiegi działania algorytmu dla 29 miast Sahary Zachodniej [3] przedstawiono za pomocą biblioteki `Matplotlib` na rysunkach poniżej. Czerwona ścieżka oznacza najlepsze znalezione rozwiązanie, natomiast niebieskie ścieżki to najlepsze rozwiązania z poprzednich iteracji. Poniższe wyniki uzyskano, wywołując funkcję z parametrami:
P_0 - 1000 osobników, $p_m = 0.03$, $p_c = 0.3$, $t_{max} = 3000$ iteracji.



Jak widać, mimo uruchamiania algorytmu z identycznymi parametrami, wyznaczane są różne rozwiązania. Wynika to z losowości procesów selekcji, krzyżowania i mutacji, co stanowi naturalną cechę algorytmów genetycznych. Zjawisko to ma istotne znaczenie metodologiczne: porównanie algorytmów nie może opierać się na pojedynczych przebiegach, lecz wymaga wielokrotnych eksperymentów i analizy statystycznej uzyskanych wyników.

W związku z tym wszystkie kolejne przeprowadzone przez mnie testy porównawcze parametrów i modyfikacji algorytmu bazują na wielu niezależnych uruchomieniach. Zebrane statystyki przedstawione na kolejnych stronach są uśredniane w celu uzyskania jak najbardziej obiektywnych wniosków.

[3] National TSPs - West Sahara

5. Wpływ prawdopodobieństwa mutacji

Prawdopodobieństwo mutacji p_m stanowi jeden z kluczowych parametrów algorytmu genetycznego, determinujący równowagę między *eksploracją* (przeszukiwaniem nowych obszarów przestrzeni rozwiązań) a *eksploatacją* (doskonaleniem już znalezionych rozwiązań). Zbyt niska wartość p_m może prowadzić do przedwczesnej zbieżności algorytmu do optimum lokalnego i utknięcie w nim, natomiast zbyt wysoka - do destrukcji dobrych rozwiązań i utraty zdolności do zbieżności.

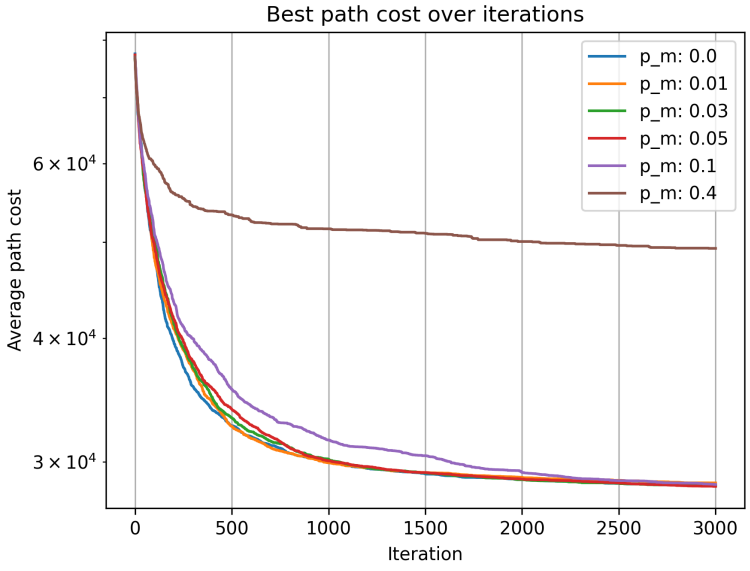
Metodyka badania

W celu zbadania wpływu prawdopodobieństwa mutacji przeprowadzono serię eksperymentów dla wartości $p_m \in \{0.0, 0.01, 0.03, 0.05, 0.1, 0.4\}$. Każdy wariant uruchomiono 40 razy z parametrami: P_0 - 1000 osobników, $p_c = 0.3$, $t_{max} = 3000$ iteracji. Wyniki zebrano w tabeli i wykresie poniżej.

Pogrubione wartości oznaczają najlepsze wyniki wśród umiarkowanych wartości p_m, natomiast *kursywa* wskazuje na wyraźnie gorsze rezultaty przy nadmiernej mutacji.

p_m	Średni koszt	Najlepszy	Najgorszy	Mediana
0.0	28403.25	**27601.17**	30248.07	28365.85
0.01	28563.76	**27601.17**	30144.90	28498.44
0.03	28378.78	**27601.17**	30769.54	28087.47
0.05	**28332.79**	**27601.17**	**29651.96**	**28042.22**
0.1	28460.46	**27601.17**	30715.10	28228.30
0.4	*49283.12*	*44125.76*	*52825.87*	*49578.17*

Ze względu na dużą rozbieżność wyników funkcji kosztu, w celu lepszej wizualizacji różnic, wykres posiada logarytmiczną skalę osi Y:



Analiza wyników

Na podstawie zebranych danych można sformułować następujące obserwacje:

1. **Najlepsze znalezione rozwiązanie** Dla badanych wartości p_m najlepszy uzyskany koszt wynosi 27601.17. Wynik ten jest bardzo bliski wartościom raportowanym w literaturze dla tej instancji (różnica wynika z braku zaokrąglania odległości), jednak ze względu na stochastyczny charakter algorytmu nie można jednoznacznie potwierdzić osiągnięcia minimum globalnego. Trasa o tej długości została zaobserwowana w sekcji 4. (Algorytm w działaniu, lewy górny rysunek).

2. **Najlepszą jakość rozwiązań uzyskano dla $p_m = 0.05$.** Dla tej wartości uzyskano jednocześnie najniższy średni koszt (28332.79), najniższą medianę (28042.22) oraz najniższy wynik w kolumnie „najgorszy" (29651.96), co wskazuje na korzystny kompromis między jakością i stabilnością.

3. **W zakresie $p_m \in [0.0, 0.1]$ różnice są umiarkowane, ale systematyczne.** Rozstęp średnich kosztów wynosi 230.97, a najlepsze wyniki pojawiają się dla $p_m = 0.05$, co sugeruje, że umiarkowana mutacja poprawia efektywność przeszukiwania. Brak informacji o pełnym rozkładzie uniemożliwia ocenę istotności statystycznej, dlatego różnice należy interpretować jako tendencje empiryczne. Jednocześnie brak mutacji nadal daje dobre rezultaty, co wskazuje na dominującą rolę krzyżowania i selekcji.

4. **Wysoka mutacja ($p_m = 0.4$) działa niekorzystnie.** Względem najlepszego wariantu ($p_m = 0.05$) średni koszt rośnie o ok. 74%, a mediana o ok. 77%, co wskazuje na istotne zaburzenie procesu optymalizacji oraz ograniczenie zdolności do utrwalania korzystnych struktur w populacji.

5. **Wyniki są zgodne z literaturą dotyczącą doboru mutacji w GA.** Mutacja powinna być niewielka, ale niezerowa - wspiera różnorodność populacji, natomiast zbyt wysoka pogarsza jakość rozwiązań. [4]

Podsumowując, dla badanej instancji TSP najbardziej uzasadniony jest wybór $p_m \in [0.03, 0.1]$, ze szczególnym wskazaniem na $p_m = 0.05$, które w obecnej próbie daje jednocześnie najlepszą jakość średnią i największą odporność na niekorzystne przebiegi.

Dobrym kierunkiem dalszych prac byłoby zastosowanie algorytmów genetycznych opartych na adaptacyjnym doborze parametrów, które dynamicznie dostosowywałyby p_m w trakcie działania, np. zwiększając go w początkowych fazach eksploracji, a następnie stopniowo zmniejszając w miarę zbliżania się do potencjalnych rozwiązań optymalnych. [5]

[4] Baeldung: Genetic Algorithms - Crossover and Mutation Probability.
[5] A Rank based Adaptive Mutation in Genetic Algorithm - Avijit Basak

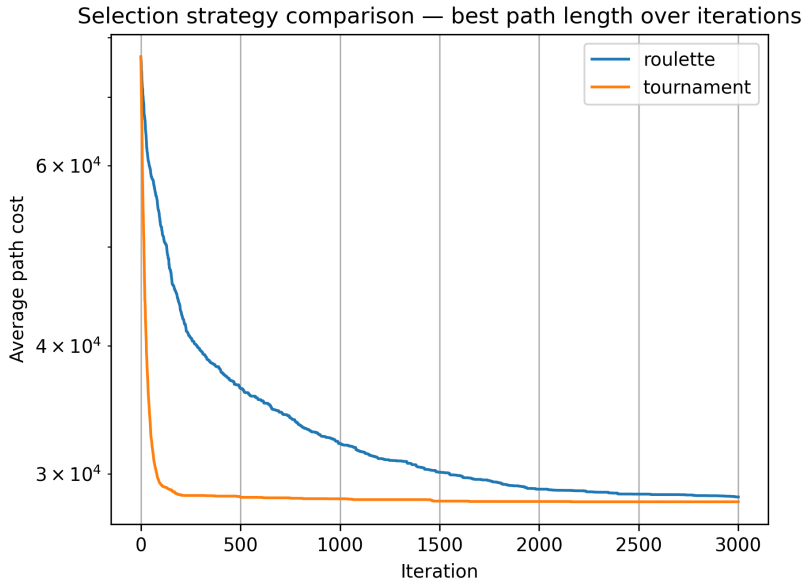
6. Rodzaj selekcji a zachowanie algorytmu

Selekcja odpowiada za wybór osobników przekazywanych do kolejnych pokoleń, a tym samym wpływa na tempo i kierunek działania algorytmu.

Selekcja ruletkowa przypisuje każdemu osobnikowi prawdopodobieństwo proporcjonalne do jego wartości funkcji przystosowania, przez co nawet słabsze rozwiązania mogą zostać wybrane. Z kolei **selekcja turniejowa** polega na losowaniu niewielkiej grupy osobników (w tej pracy trzech) i wyborze najlepszego z nich, co w praktyce wzmacnia preferencję dla lepszych rozwiązań.

W celu porównania obu metod, skumulowałem 40 uruchomień z parametrami: P_0 - 1000 osobników, $p_m = 0.05$, $p_c = 0.3$, $t_{max} = 3000$ iteracji. Wyniki zebrano w tabeli i na wykresie poniżej.

Metoda selekcji	Średni koszt	Najlepszy	Najgorszy	Mediana
Ruletkowa	28331.95	**27601.17**	29656.66	28042.26
Turniejowa	**28053.41**	**27601.17**	**28917.53**	**27891.68**



Porównanie metod

Z tabeli wynika, że selekcja turniejowa osiąga lepsze wyniki. Różnice w średnim koszcie wynoszą ok. 278, natomiast są wyraźniejsze w najgorszym przypadku (ok. 739), co wskazuje na większą stabilność tej metody.

Na wykresie widać również różnicę w tempie zbieżności – selekcja turniejowa osiąga niski poziom kosztu już po około 200 iteracjach, podczas gdy metoda ruletkowa wymaga znacznie większej liczby iteracji dla porównywalnych rezultatów. Różnice te są konsekwencją mechanizmu działania obu metod: silniejsza presja selekcyjna w turniejowej przyspiesza poprawę populacji, natomiast ruletkowa, dopuszcza większy udział słabszych osobników.

Podsumowanie

W pracy przedstawiono algorytm genetyczny oraz jego zastosowanie do rozwiązania problemu komiwojażera na instancji 29 miast Sahary Zachodniej. Zaimplementowany algorytm wykorzystuje reprezentację permutacyjną rozwiązań, operator krzyżowania OX (Order Crossover) oraz mutację typu swap, co pozwala zachować unikalność odwiedzanych miast i jednocześnie efektywnie eksplorować przestrzeń rozwiązań.

Przeprowadzone eksperymenty wykazały istotny wpływ parametrów algorytmu na jakość uzyskiwanych rozwiązań. Prawdopodobieństwo mutacji p_m odgrywa kluczową rolę w balansowaniu między eksploracją a eksploatacją przestrzeni rozwiązań – zbyt niska mutacja może prowadzić do utknięcia w optimum lokalnym, natomiast zbyt wysoka degraduje jakość populacji. W badanej instancji zadowalający zakres wyniósł $p_m \in [0.03, 0.1]$, ze szczególnym wskazaniem na $p_m = 0.05$, co zapewniało jednocześnie dobrą jakość średnią, stabilność wyników oraz odporność na niekorzystne przebiegi.

Porównanie metod selekcji wykazało przewagę selekcji turniejowej nad ruletkową – zapewniała ona szybszą konwergencję, wyższą stabilność populacji oraz lepsze wyniki końcowe dla badanej instancji TSP. Jednocześnie trzeba brać pod uwagę, że większa presja selekcyjna może prowadzić do szybszej zbieżności, ale zwiększa ryzyko utraty różnorodności populacji. Wyniki te potwierdzają, że mechanizm presji selekcyjnej ma istotne znaczenie dla efektywności algorytmu genetycznego.

Najlepszy uzyskany koszt trasy (27601.17) jest bliski wartościom literaturowym, co świadczy o poprawności implementacji i skuteczności algorytmu jako heurystyki do problemów kombinatorycznych. Jednocześnie stochastyczny charakter metody podkreśla konieczność wielokrotnych eksperymentów i analizy statystycznej wyników dla rzetelnej oceny wyników.

Perspektywy dalszego rozwoju obejmują m.in. implementację elitaryzmu, adaptacyjny dobór parametrów p_m i p_c, zastosowanie bardziej zaawansowanych operatorów krzyżowania, takich jak PMX, oraz eksperymenty z algorytmami ewolucyjnymi. Takie modyfikacje mają szansę zwiększyć skuteczność algorytmu, poprawić stabilność wyników i umożliwić jego zastosowanie do większych instancji problemu komiwojażera.