Sprawozdanie z projektu- Algorytmy Genetyczne

Tobiasz Mańkowski 186924
Maksymilian Burdziej 189030

Temat: Zaimplementowanie algorytmu z podejściem ewolucyjnym, który rozwiązuje zadanie pchania wózka.

Cel Projektu

Celem projektu jest stworzenie programu rozwiązującego problem pchania wózka za pomocą samodzielnie zaimplementowanego algorytmu genetycznego (SGA – Simple Genetic Algorithm). Problem polega na znalezieniu sterowania, które maksymalizuje całkowitą drogę przebywaną przez wózek w zadanym czasie, jednocześnie minimalizując całkowity wysiłek związany z pchaniem.

Projekt ma na celu zarówno rozwiązanie konkretnego problemu, jak i demonstrację efektywności algorytmów genetycznych w optymalizacji dynamicznych procesów fizycznych.

Struktura kodu

Poniżej przedstawiono strukturę kodu algorytmu ewolucyjnego rozwiązującego zadanie pchania wózka, wraz z omówieniem jego poszczególnych elementów.

1. Model systemu

- **Funkcja `simulate_system(u, N)`**:
  - Definiuje model stanu układu dyskretnego, opisany równaniami różnicowymi:

$$x_1(k+1) = x_2(k),$$
$$x_2(k+1) = 2x_2(k) - x_1(k) + \frac{1}{N^2}u_k(k),$$

  - Funkcja przyjmuje jako parametry:
    - u – wektor sterowania,
    - N – liczba kroków.
  - Zwraca trajektorie:
    - x1 – położenie,
    - x2 – prędkość.

2. Funkcja celu

- **Funkcja `evaluate_fitness(u, N)`**:
    - Oblicza wartość wskaźnika jakości sterowania J:

$$J = x_1(N) - \frac{1}{2N}\sum_{k=0}^{N-1} u^2(k),$$

    - Maksymalizuje drogę x1(N) przy minimalizacji wysiłku (kary za sterowanie).
    - Parametry:
        - u – wektor sterowania,
        - N – liczba kroków.
    - Zwraca wartość J, która ocenia jakość sterowania.

3. Inicjalizacja populacji

- **Funkcja `initialize_population(population_size, N)`**:
    - Generuje początkową populację losowych wektorów sterowania u o ograniczonych wartościach do przedziału [−1,1].
    - Parametry:
        - `population_size` – liczba wektorów w populacji,
        - N – liczba kroków (długość wektora u).

4. Selekcja turniejowa

- **Funkcja `tournament_selection(population, fitness, tournament_size, num_selected)`**:
    - Realizuje selekcję turniejową:
        - Losowo wybiera daną liczbę osobników,
        - Zwycięzca (o najwyższej wartości J) jest dodawany do nowej puli.
    - Parametry:
        - `population` – aktualna populacja,
        - `fitness` – wartości dopasowania (J) dla osobników,
        - `tournament_size` – liczba osobników w turnieju,
        - `num_selected` – liczba osobników wybranych do nowej populacji.

5. Krzyżowanie

- **Funkcja `crossover(selected, population_size, N)`**:
    - Realizuje krzyżowanie jednopunktowe:
        - Dla dwóch losowych rodziców tworzy potomka, dzieląc genotypy w losowym punkcie podziału.
    - Parametry:
        - `selected` – wybrani osobnicy,
        - `population_size` – liczba osobników w nowej populacji,
        - N – liczba kroków (długość wektora u).

6. Mutacja

- **Funkcja `mutate(population, mutation_rate, N)`**:
  - Dodaje losowe wartości (mutacje) do wektorów sterowania z prawdopodobieństwem `mutation_rate`.
  - Zapewnia ograniczenie wartości u do przedziału [-1, 1].
  - Parametry:
    - `population` – populacja poddawana mutacji,
    - `mutation_rate` – prawdopodobieństwo mutacji,
    - N – liczba kroków.

7. Algorytm genetyczny

- **Funkcja `genetic_algorithm(N, population_size, generations, mutation_rate)`**:
  - Realizuje algorytm genetyczny w następujących krokach:
    - **Inicjalizacja populacji** – losowe wektory sterowania.
    - Następnie iteracyjnie:
      - **Ocena dopasowania** – obliczenie J dla wszystkich osobników.
      - **Selekcja turniejowa** – wybór najlepszych osobników.
      - **Krzyżowanie** – tworzenie nowej populacji.
      - **Mutacja** – wprowadzenie losowych zmian.
  - Parametry:
    - N – liczba kroków,
    - `population_size` – liczba osobników w populacji,
    - `generations` – liczba generacji,
    - `mutation_rate` – prawdopodobieństwo mutacji.
  - Zwraca:
    - Najlepszy wektor sterowania u,
    - Najwyższą wartość wskaźnika J,
    - Historię najlepszych i średnich wartości J w trakcie generacji.

8. Wizualizacja wyników

Kod generuje i zapisuje wykresy ilustrujące wyniki optymalizacji:

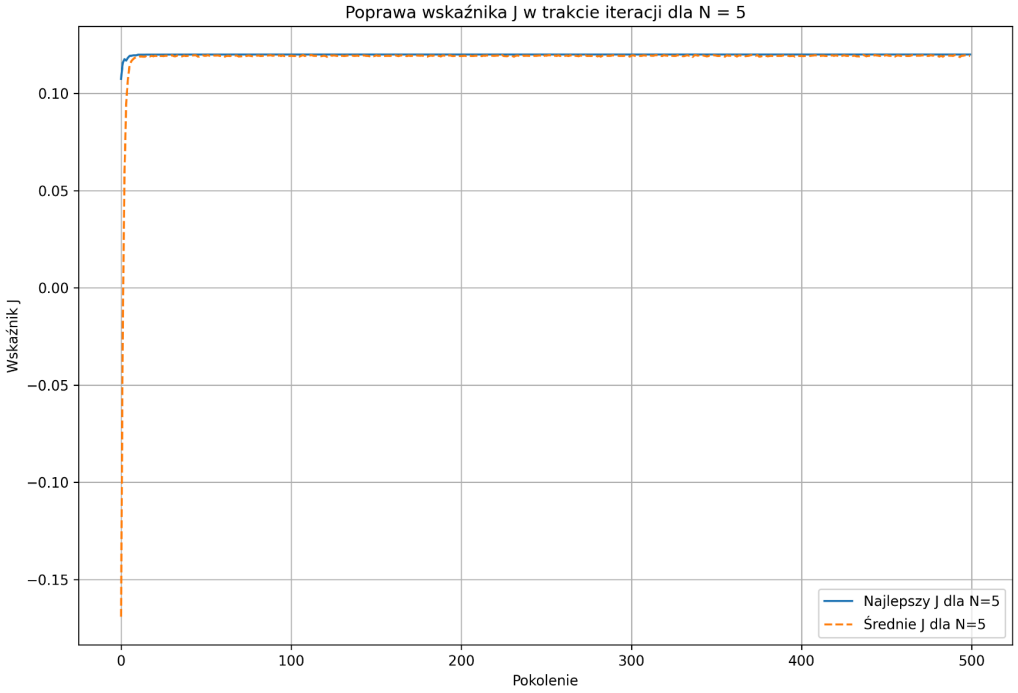
1. **Wskaźnik J w zależności od N**:
   - Najlepsze wartości J dla różnych długości wektora.
2. **Trajektorie sterowania u dla różnych N**:
   - Wektory sterowania u dla każdej wartości N.
3. **Trajektorie położenia x1**:
   - Położenie x1 w czasie dla różnych wartości N.
4. **Poprawa wskaźnika J w trakcie generacji**:
   - Historia najlepszych i średnich wartości J w każdej generacji.

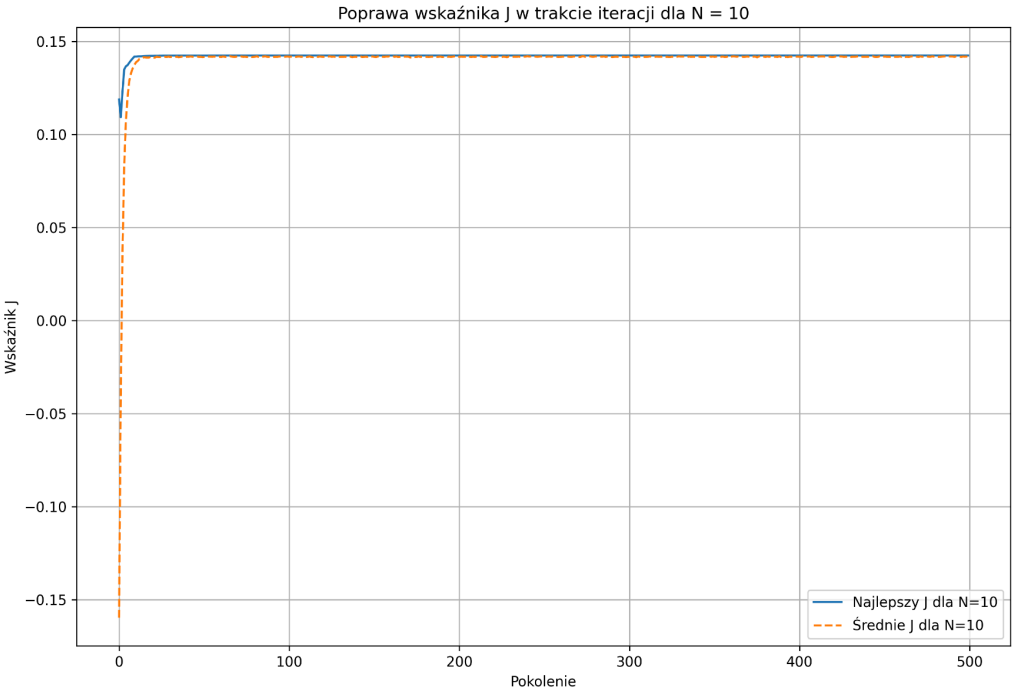
Rozwiązanie zadania, analiza wyników

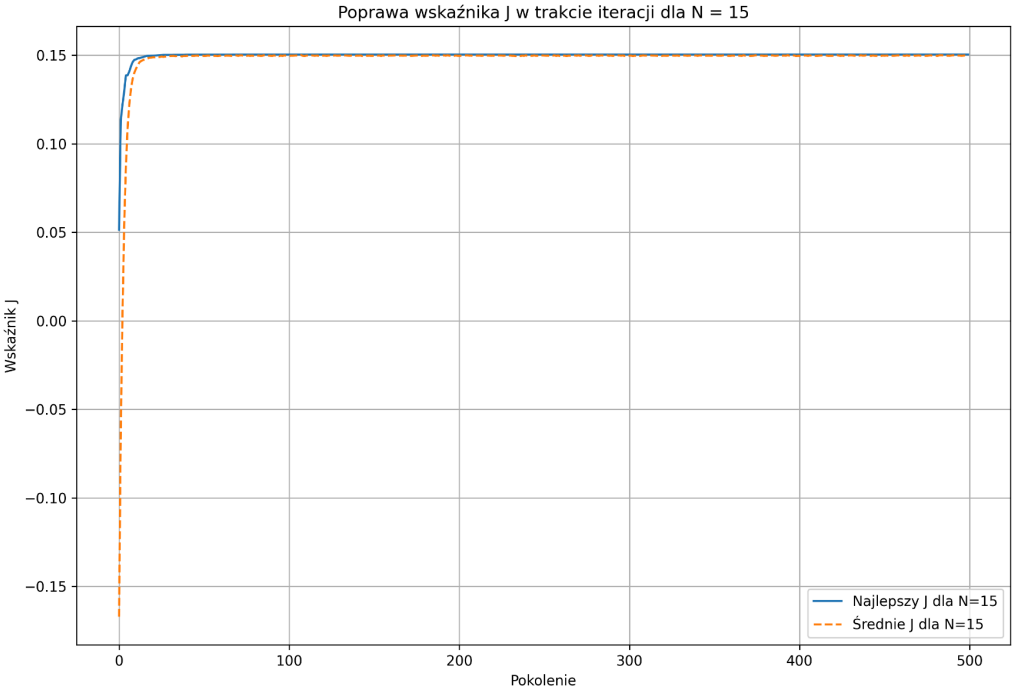
Wyniki otrzymane za pomocą algorytmu genetycznego porównywane są do wyników optymalnych, znalezionych za pomocą biblioteki scipy.optimize.

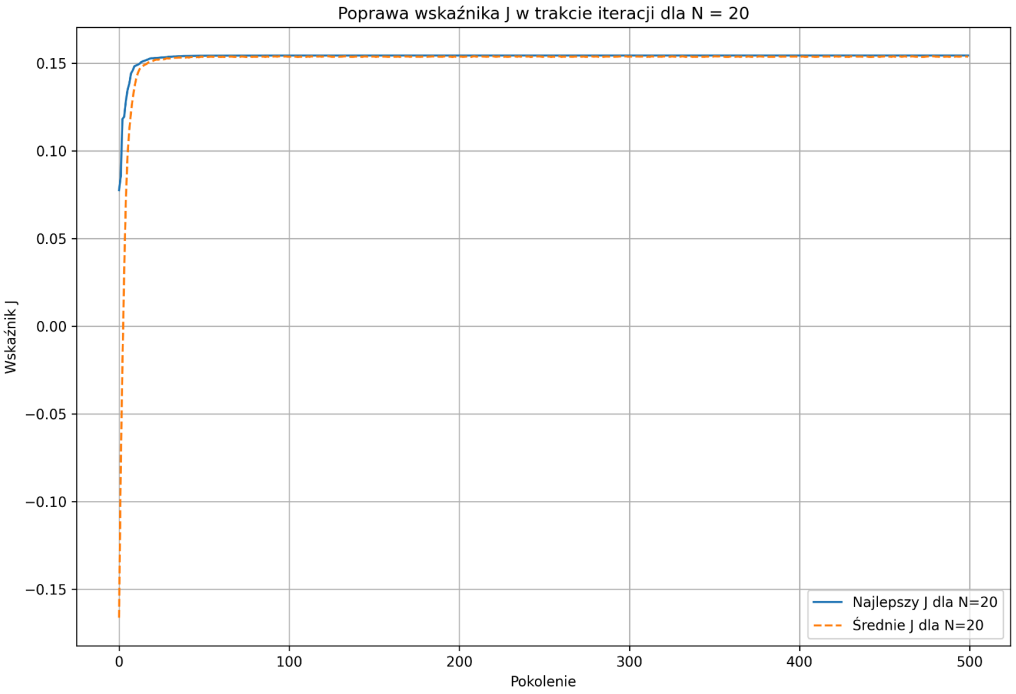
a. **Wykresy obrazujące poprawę wskaźnika J podczas iteracji**

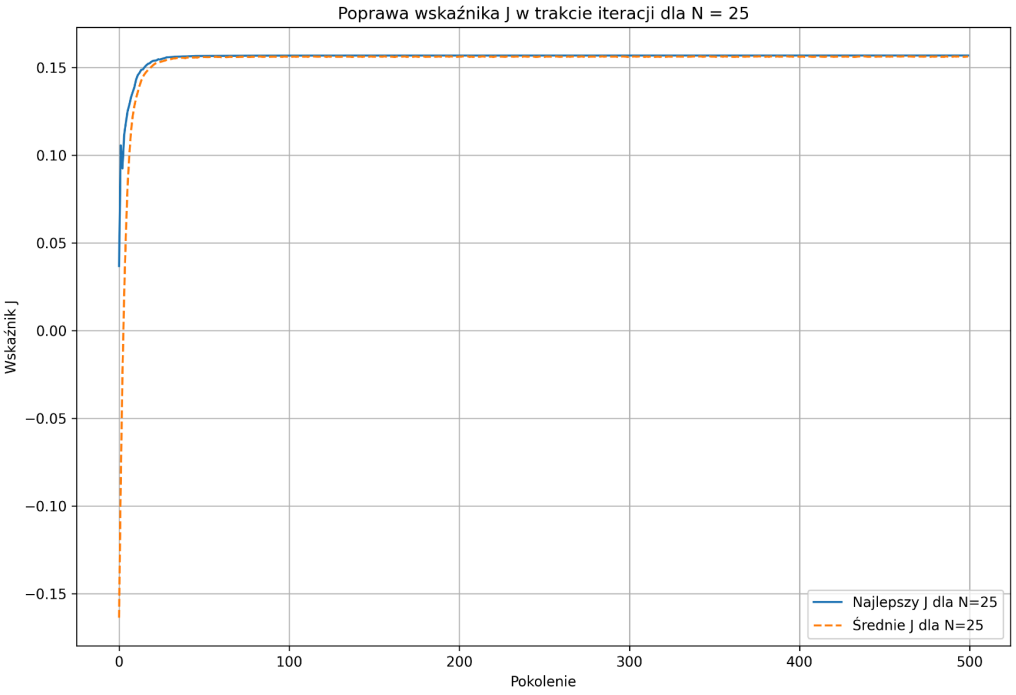
Poniżej przedstawiono wykresy obrazujące poprawę wskaźnika J w trakcie iteracji dla wybranych ilości kroków (5,10,15,20,25,30,35,40,45).

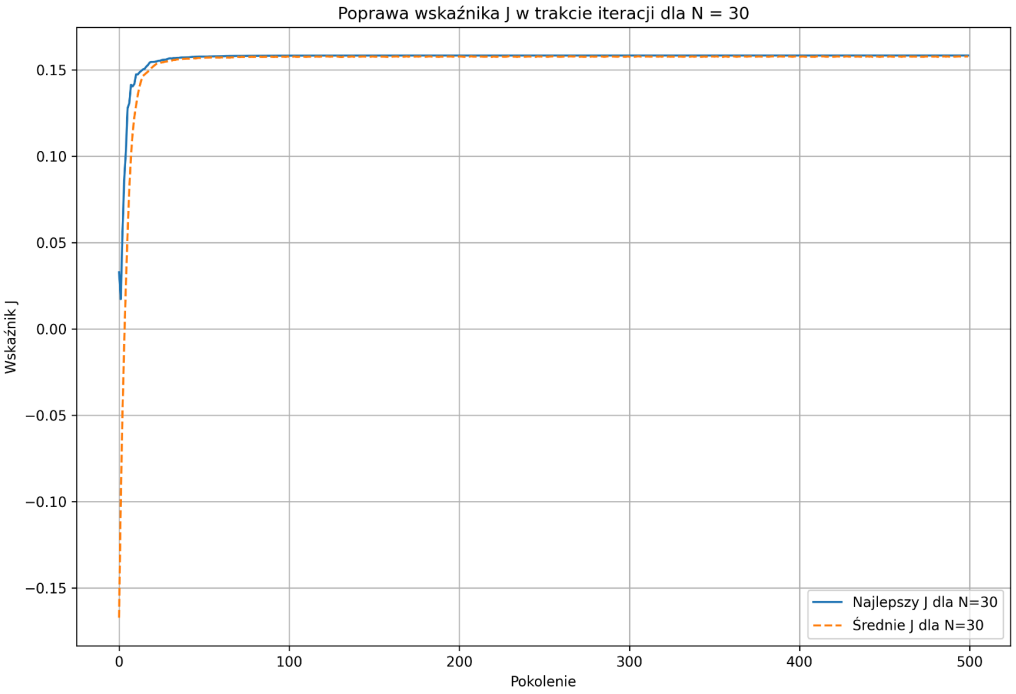


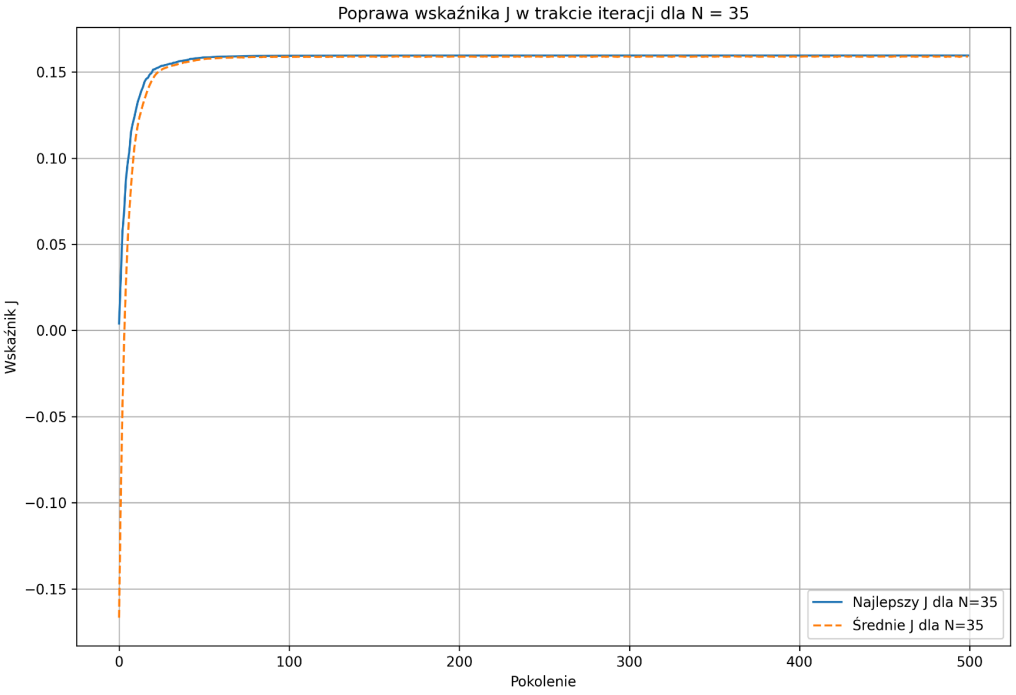

Poprawa wskaźnika J w trakcie iteracji dla N = 10
Wskaźnik J
Pokolenie
Najlepszy J dla N=10
Średnie J dla N=10
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500

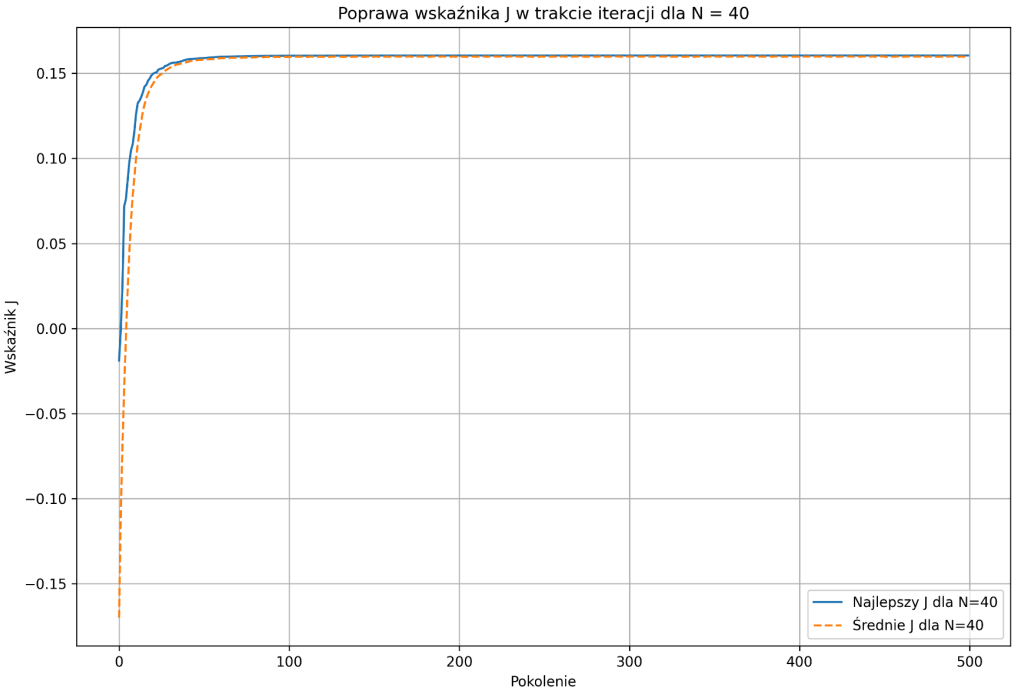

Poprawa wskaźnika J w trakcie iteracji dla N = 15
Wskaźnik J
Pokolenie
Najlepszy J dla N=15
Średnie J dla N=15
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500

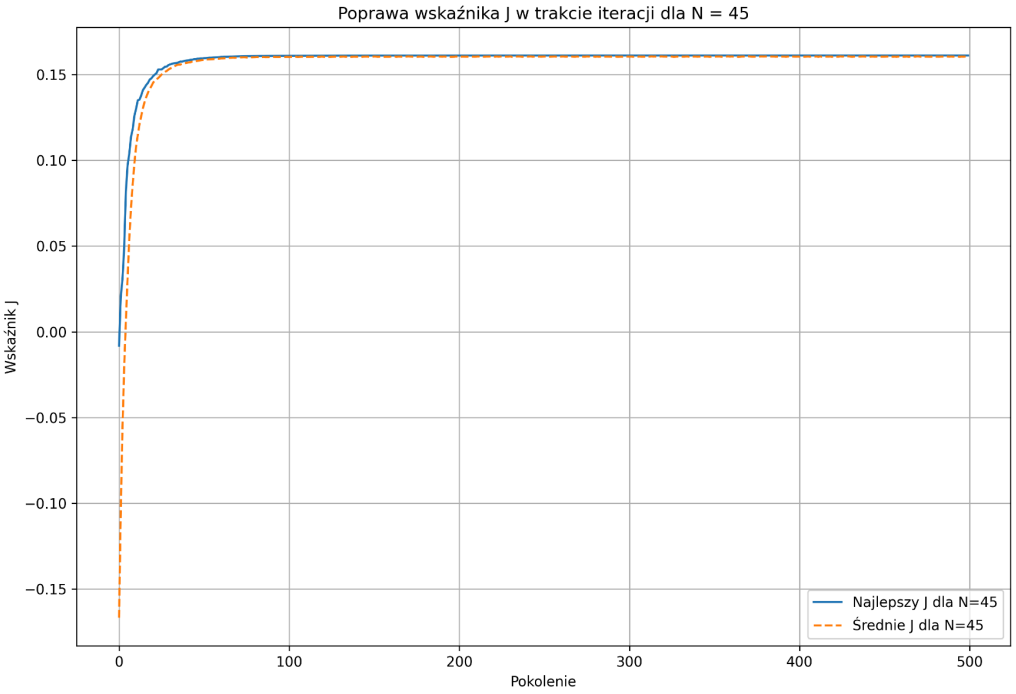

Poprawa wskaźnika J w trakcie iteracji dla N = 20
Wskaźnik J
Pokolenie
Najlepszy J dla N=20
Średnie J dla N=20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500


Poprawa wskaźnika J w trakcie iteracji dla N = 25
Wskaźnik J
Pokolenie
Najlepszy J dla N=25
Średnie J dla N=25
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500


Poprawa wskaźnika J w trakcie iteracji dla N = 30
Wskaźnik J
Pokolenie
Najlepszy J dla N=30
Średnie J dla N=30


Poprawa wskaźnika J w trakcie iteracji dla N = 35
Wskaźnik J
Pokolenie
Najlepszy J dla N=35
Średnie J dla N=35


Poprawa wskaźnika J w trakcie iteracji dla N = 40
Wskaźnik J
Pokolenie
Najlepszy J dla N=40
Średnie J dla N=40
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500

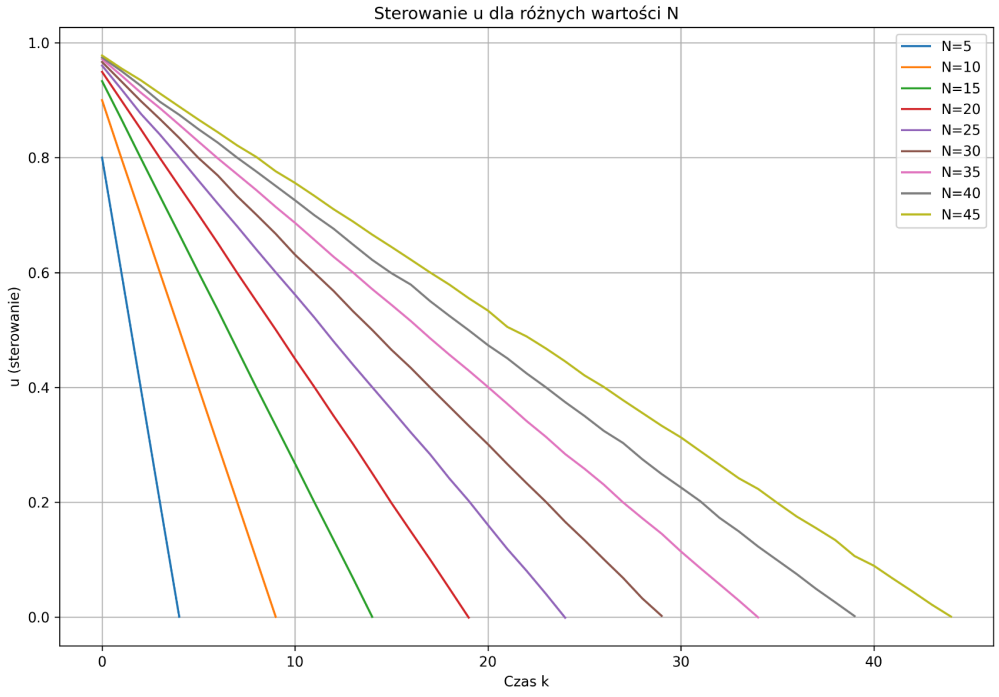

Poprawa wskaźnika J w trakcie iteracji dla N = 45
Wskaźnik J
Pokolenie
Najlepszy J dla N=45
Średnie J dla N=45
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
400
500

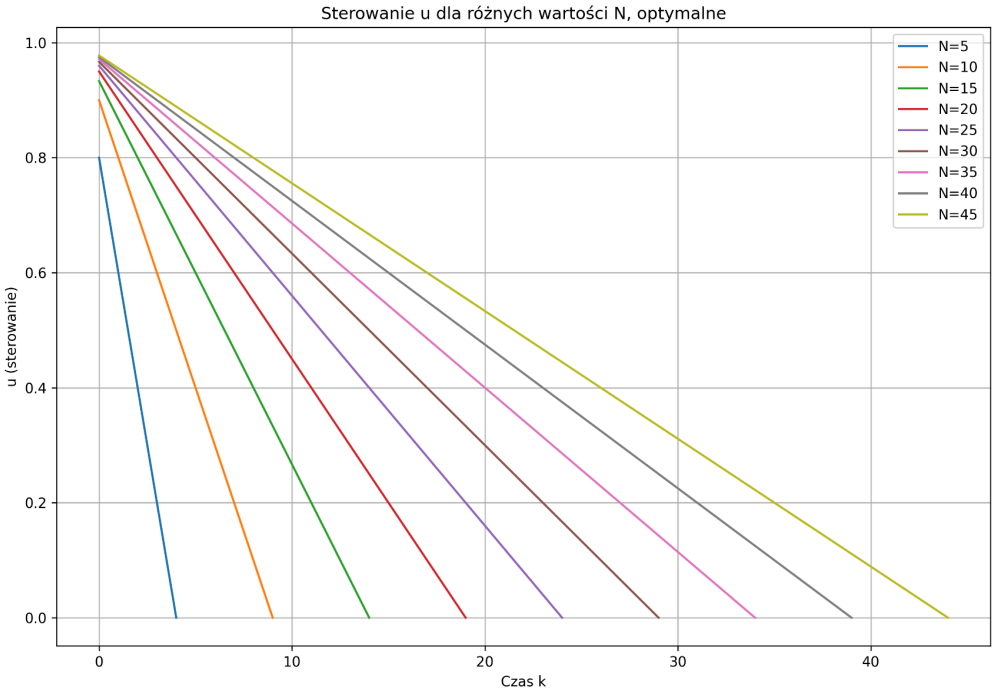
W każdym z powyższych przypadków algorytm szybko znajduje optymalne rozwiązanie zadanego problemu. Najszybciej jest to osiągane w przypadku małych wartości N, z racji na mniejszą ilość możliwości w przypadku krótszego wektora sterowania.

b. **Porównanie wektorów sterowania w zależności od N**

Poniższy wykres prezentuje znalezione przez algorytm genetyczny wektory sterowania dla konkretnych N.

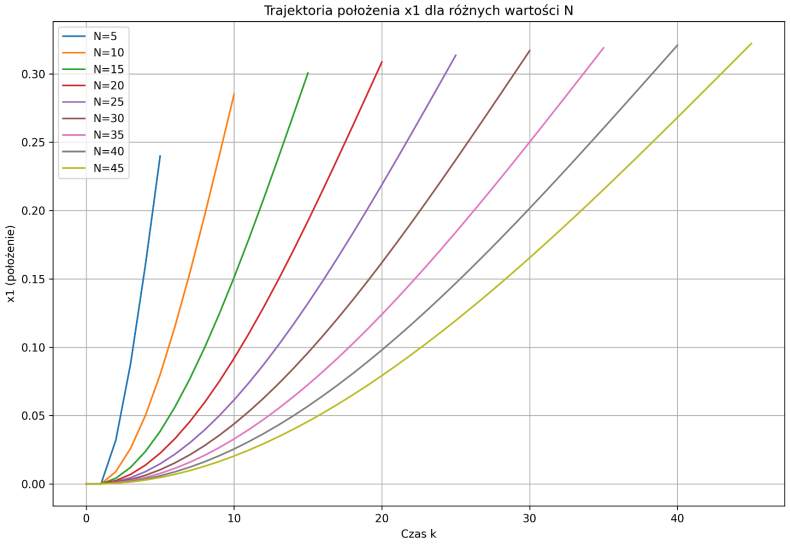
Poniższe wykresy prezentują znalezione przez algorytm genetyczny wektory sterowania dla konkretnych N. Na pierwszym wykresie widoczny jest wynik wyznaczony przy użyciu algorytmu genetycznego, natomiast na drugim prezentowane jest rozwiązanie optymalne. Rozwiązania nieznacznie różnią się od siebie, w przypadku rozwiązania optymalnego linie na wykresie są idealnie proste.

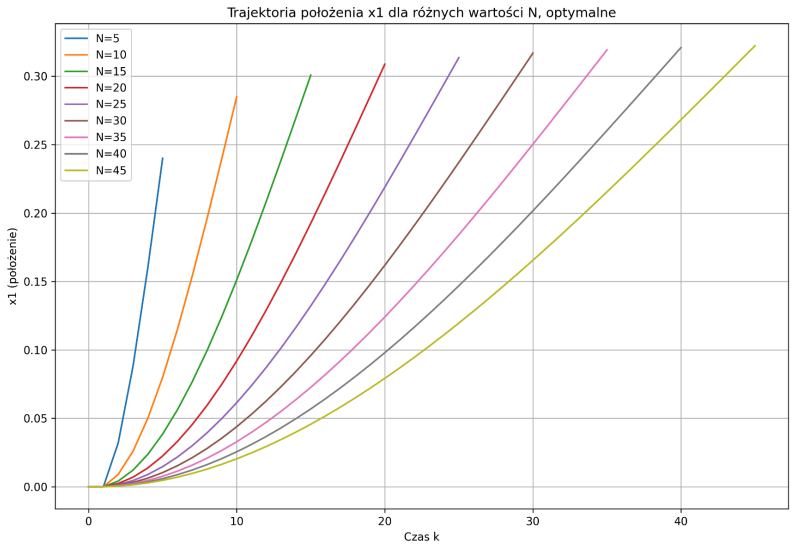




c. **Położenie wózka w czasie**

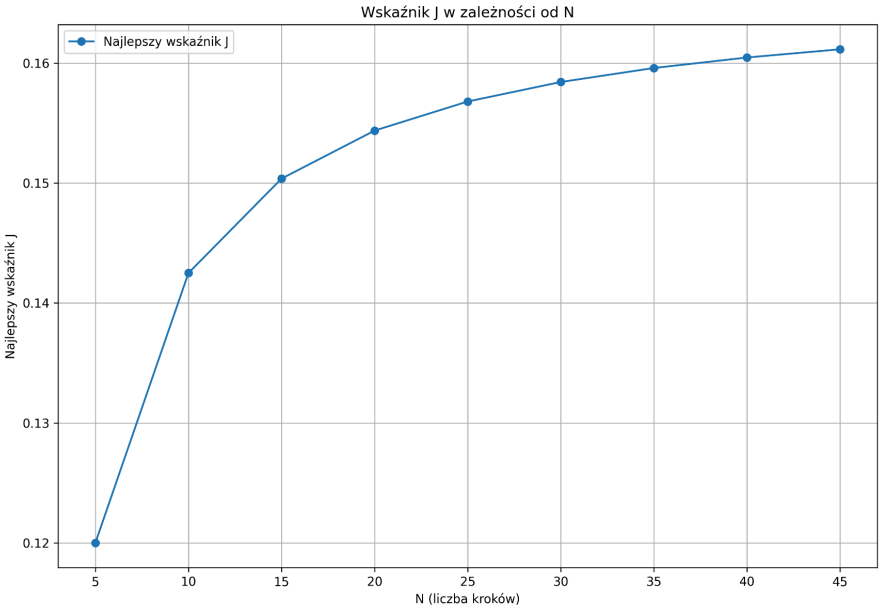
Poniższe wykresy prezentują zmianę trajektorii położenia w czasie dla różnych wartości N. Na wykresie pierwszym widoczna jest zmiana trajektorii położenia przy użyciu algorytmu genetycznego, natomiast na drugim prezentowane jest rozwiązanie optymalne. Oba rozwiązania nieznacznie się różnią.

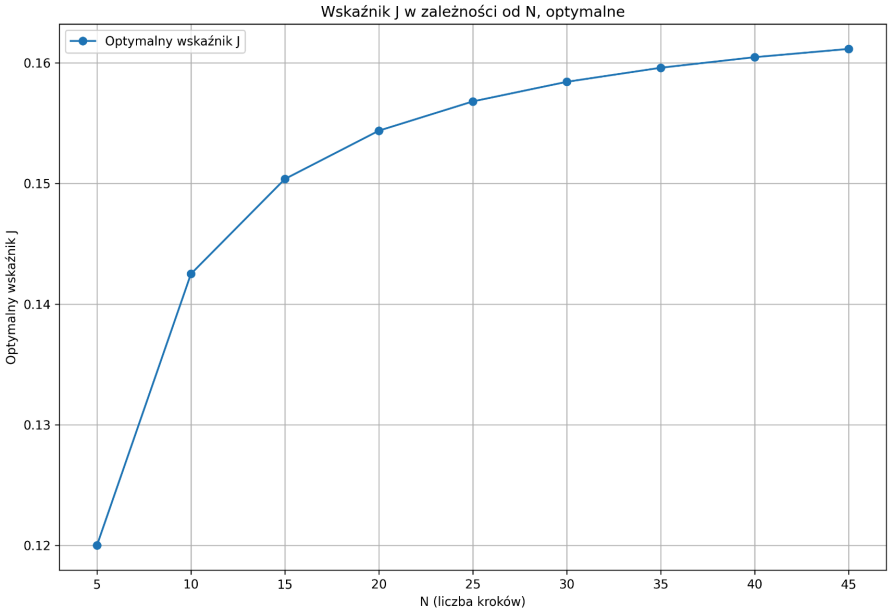




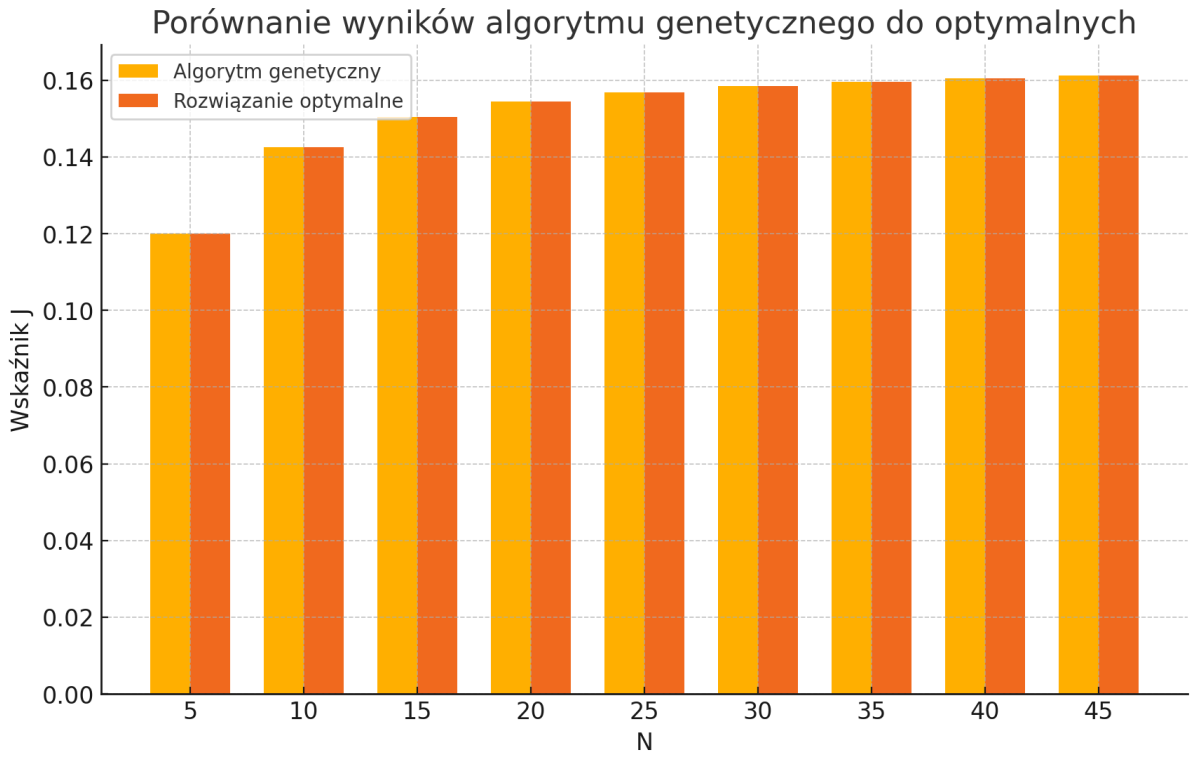
d. **Wartość wskaźnika J w zależności od N**

Poniższe wykresy prezentuje wartość wskaźnika J w zależności od liczby N. Można zauważyć że wykres przybiera kształt krzywej logarytmicznej. Jest to związane z właściwościami funkcji J, w której przy mniejszych wartościach N różnice we wskaźniku J są bardziej znaczące niż w przypadku większych, bardziej dokładnych sterowań. Pierwszy z nich prezentuje działanie zaimplementowanego algorytmu genetycznego, natomiast drugi pokazuje rozwiązanie optymalne.





e. Porównanie wyników algorytmu genetycznego do optymalnych



N	SGA	Optymalne	Różnica
5	0,11999990	0,12000000	9,5E-08
10	0,14249977	0,14250000	2,3E-07
15	0,15037024	0,15037037	1,3E-07
20	0,15437456	0,15437500	4,4E-07
25	0,15679962	0,15680000	3,8E-07
30	0,15842532	0,15842593	6,1E-07
35	0,15959069	0,15959184	1,2E-06
40	0,16046805	0,16046875	7E-07
45	0,16115169	0,16115226	5,7E-07

Wnioski

Analiza wyników jednoznacznie wskazuje, że algorytm genetyczny skutecznie rozwiązuje problem optymalnego sterowania wózkiem. Otrzymane wyniki są niemal identyczne z rozwiązaniami optymalnymi, co potwierdza efektywność zastosowanego podejścia ewolucyjnego.