# Algorytmy Optymalizacji Inspirowane Naturą
# Projekt startowy

Jakub Wasilewski 263852

20.11.2025

## Spis treści

# 1 Sformułowanie zadania

Celem jest implementacja i przebadanie metaheurystyki Algorytmu Ewolucyjnego (EA) dla problemu cVRP oraz porównanie jej z metodami nieewolucyjnymi: algorytmem zachłannym (greedy), symulowanym wyżarzaniem (SA) i losowym przeszukiwaniem. Funkcja celu minimalizuje łączny koszt tras floty pojazdów o ograniczonej pojemności.

# 2 Sposób rozwiązywania zadania

Rozwiązania kodowane są jako permutacje klientów (jak w TSP). Dekoder dzieli permutację na trasy spełniające ograniczenie pojemności i liczy koszt sumując odległości (EUC_2D, zaokrąglenie). Metaheurystyki operują na permutacjach; ocena to łączny koszt z dekodera.

# 3 Metody użyte do rozwiązania zadania

- Algorytm losowy (random)
- Algorytm zachłanny (greedy)
- Symulowane wyżarzanie (SA)
- Algorytm ewolucyjny (EA)

# 4 Implementacja

## 4.1 Reprezentacja i dekoder

Każde rozwiązanie to permutacja klientów (magazyn pomijany). Dekoder przechodzi po permutacji, sumuje zapotrzebowanie i gdy pojemność jest przekroczona, rozpoczyna nową trasę. Koszt to suma odległości z/do magazynu i między kolejnymi klientami (EUC_2D, zaokrąglone).

## 4.2 Algorytm losowy

- Proste losowe przeszukiwanie przestrzeni rozwiązań bez heurystyk.
- W każdej iteracji generowana jest losowa permutacja klientów, która jest następnie dekodowana na rozwiązanie cVRP.
- **Parametry:** Liczba iteracji: 2000 (`random_iterations=2000`).
- **Log:** W każdej iteracji zapisywane są: najlepsze rozwiązanie dotąd (best), bieżące (current), średnia ze wszystkich (avg), najgorsze dotąd (worst).

## 4.3 Algorytm zachłanny

- Konstrukcja rozwiązania poprzez zachłanny wybór najbliższego nieodwiedzonego klienta (nearest-neighbour).
- Algorytm rozpoczyna od wybranego punktu startowego i iteracyjnie dodaje najbliższego klienta do trasy, aż wszyscy zostaną odwiedzeni. Dekoder automatycznie tworzy nowe trasy przy przekroczeniu pojemności.
- **Multi-start:** Wykonuje się wiele restartów z różnymi punktami startowymi (rotacja po klientach), by zwiększyć szansę na znalezienie dobrego rozwiązania.

- **Parametry:** Liczba restartów: `greedy_restarts` (domyślnie równa liczbie klientów).
- **Log:** Po każdym restarcie zapisywane są statystyki best/current/avg/worst.

### 4.4 Symulowane wyżarzanie (SA)

- Metaheurystyka inspirowana procesem wyżarzania metali - stopniowe "oziębianie" systemu pozwala na wyjście z minimów lokalnych.
- **Rozwiązanie startowe:** Losowa permutacja klientów (każde uruchomienie startuje z innego punktu przestrzeni rozwiązań).
- **Sąsiedztwo:** Operator swap - zamiana dwóch losowych pozycji w permutacji. Prosty operator pozwalający na eksplorację różnych konfiguracji tras.
- **Akceptacja rozwiązania:**
  - Jeśli nowe rozwiązanie jest lepsze (niższy koszt) - zawsze akceptowane.
  - Jeśli gorsze - akceptowane z prawdopodobieństwem $\exp(-\Delta/T)$, gdzie $\Delta$ to różnica kosztów, a $T$ to bieżąca temperatura.
  - Wysoka temperatura na początku pozwala akceptować gorsze ruchy (eksploracja), niska na końcu wymusza schodzenie do minimum (eksploatacja).
- **Schemat chłodzenia:** Temperatura obniżana geometrycznie: $T_{i+1} = \alpha \cdot T_i$, gdzie $\alpha$ to współczynnik chłodzenia (np. 0.995).
- **Parametry:** Temperatura początkowa $T_0$, minimalna $T_{\min}$, współczynnik chłodzenia $\alpha$, liczba iteracji na każdą temperaturę.
- **Kryterium stopu:** Algorytm kończy się gdy temperatura spadnie poniżej $T_{\min}$.
- **Log:** W każdej iteracji zapisywane są best/current/avg/worst, co pozwala obserwować charakterystyczne "skoki" przy akceptacji gorszych rozwiązań.

### 4.5 Algorytm ewolucyjny (EA)

- **Metoda:** Metaheurystyka inspirowana ewolucją biologiczną - populacja rozwiązań ewoluuje przez selekcję, krzyżowanie i mutację.
- **Inicjalizacja:** Populacja losowych permutacji (rozmiar `ea_population`), każda dekodowana i oceniana.
- **Selekcja:** Turniejowa - losujemy `ea_tournament` osobników i wybieramy najlepszego. Większy turniej zwiększa presję selekcyjną (silniejsze osobniki mają większą szansę reprodukcji).
- **Krzyżowanie:**
  - Operator Ordered Crossover (OX) - kopiuje segment od rodzica 1, uzupełnia brakujące elementy w kolejności z rodzica 2.
  - Operator Partially Mapped Crossover (PMX) - wymienia segmenty między rodzicami i naprawia konflikty przez mapowanie pozycji.
  - Oba operatory zachowują poprawność permutacji (każdy klient występuje dokładnie raz).
  - Stosowane z prawdopodobieństwem `ea_crossover_rate` (Px); w przeciwnym razie kopiowany jest rodzic bez zmian.

- **Mutacja:**
  - Swap - zamienia miejscami dwa losowo wybrane geny (klientów) w permutacji. Prosty operator lokalnej zmiany.
  - Inversion - odwraca kolejność genów w losowo wybranym segmencie permutacji. Większa zmiana niż swap.
  - Wprowadzają różnorodność genetyczną, pozwalają uciec z minimów lokalnych.
  - Każdy gen mutowany z prawdopodobieństwem `ea_mutation_rate` (Pm).
- **Elitaryzm:** Kopiowanie `ea_elites` najlepszych osobników do następnego pokolenia bez zmian - zapewnia że najlepsze rozwiązania nie zostaną utracone.
- **Parametry:** `ea_population`, `ea_generations`, `ea_crossover_rate`, `ea_mutation_rate`, `ea_tournament`, `ea_elites`, typ krzyżowania, typ mutacji.
- **Kryterium stopu:** Algorytm kończy się po wykonaniu `ea_generations` pokoleń.
- **Log:** W każdym pokoleniu zapisywane są statystyki całej populacji: best (najlepszy osobnik), avg (średnia populacji), worst (najgorszy osobnik) - pozwala śledzić zbieżność i różnorodność.

### 4.6 Konfiguracja i logi

Parametry w plikach `config_baseline.ini` / `config_tuning.ini` (katalogi danych/logów, liczba uruchomień per algorytm, ustawienia EA/SA/greedy/losowy). Program zapisuje logi dla każdego uruchomienia w plikach CSV, a zbiorcze statystyki w `summary.csv`. Skrypt `scripts/plot_logs.py` generuje wykresy pojedynczych przebiegów, zestawienia i wykres słupkowy najlepszych wyników.

## 5 Pliki wejściowe

Instancje z katalogu `inputs/`: A-n32-k5, A-n37-k6, A-n39-k5, A-n45-k6, A-n48-k7, A-n54-k7, A-n60-k9. Optymalne koszty z `optimal-solutions/*.sol`.

## 6 Procedura badawcza

- Uruchomienie: `./bin/vrp_runner config_baseline.ini`.
- Parametry bazowe: random_runs=1000 (iteracje=2000), greedy_runs=N (restarts=32), sa_runs=10 ($T_0 = 100$, $T_{\min} = 0{,}01$, $\alpha = 0{,}995$, iter/temp=20), ea_runs=10 (pop=100, gen=2000, Px=0,7, Pm=0,1, tour=5, elites=1, krzyżowanie: OX, mutacja: swap).
- Wizualizacje: `plot_logs.py` generuje wykresy single/combined/bar_best do `logs_*/` oraz `plots_*/`.

## 7 Wyniki badań przed strojeniem metaheurystyk

### 7.1 Wyniki zbiorcze

Tabela przedstawia statystyki wyników z wielu uruchomień każdego algorytmu. Dla każdego algorytmu przeprowadzono N niezależnych uruchomień, każde zwracające najlepsze znalezione rozwiązanie. Kolumny Best/Worst/Avg/Std oznaczają odpowiednio: najlepszy wynik ze wszystkich N uruchomień, najgorszy wynik, średnią arytmetyczną oraz odchylenie standardowe.

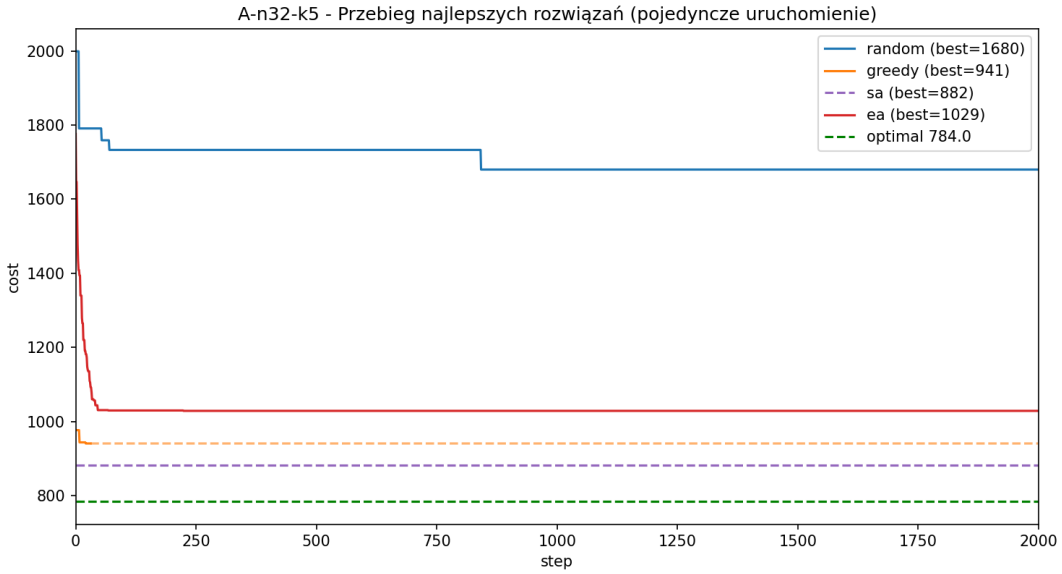
Tabela 1: Wyniki zbiorcze przed strojeniem metaheurystyk

| Instance | Optimal | Random Runs | Random Best | Random Worst | Random Avg | Random Std | Greedy Runs | Greedy Best | Greedy Worst | Greedy Avg | Greedy Std | EA Runs | EA Best | EA Worst | EA Avg | EA Std | SA Runs | SA Best | SA Worst | SA Avg | SA Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A-n32-k5 | 784 | 1000 | 1306 | 1745 | 1627.68 | 46.93 | 32 | 941 | 941 | 941 | 0.00 | 10 | 914 | 1066 | 991.5 | 48.55 | 10 | 830 | 937 | 889 | 36.10 |
| A-n37-k6 | 949 | 1000 | 1551 | 1866 | 1762.47 | 43.20 | 37 | 1058 | 1058 | 1058 | 0.00 | 10 | 1006 | 1131 | 1064.7 | 38.71 | 10 | 995 | 1066 | 1026.2 | 22.64 |
| A-n39-k5 | 822 | 1000 | 1518 | 1820 | 1720.1 | 43.49 | 39 | 920 | 920 | 920 | 0.00 | 10 | 935 | 1109 | 1015.2 | 51.17 | 10 | 862 | 1022 | 940.6 | 49.76 |
| A-n45-k6 | 944 | 1000 | 1986 | 2395 | 2268.04 | 55.15 | 45 | 1119 | 1119 | 1119 | 0.00 | 10 | 1116 | 1445 | 1236.6 | 116.18 | 10 | 1007 | 1252 | 1120.5 | 70.88 |
| A-n48-k7 | 1073 | 1000 | 2108 | 2472 | 2354.95 | 56.97 | 48 | 1301 | 1301 | 1301 | 0.00 | 10 | 1256 | 1452 | 1365.2 | 59.04 | 10 | 1208 | 1318 | 1257.4 | 30.73 |
| A-n54-k7 | 1167 | 1000 | 2395 | 2814 | 2689.17 | 59.84 | 54 | 1380 | 1380 | 1380 | 0.00 | 10 | 1380 | 1610 | 1509 | 73.11 | 10 | 1298 | 1436 | 1359.8 | 40.73 |
| A-n60-k9 | 1354 | 1000 | 2777 | 3193 | 3068.58 | 62.74 | 60 | 1524 | 1524 | 1524 | 0.00 | 10 | 1563 | 1823 | 1672.7 | 86.57 | 10 | 1501 | 1638 | 1570.3 | 43.11 |

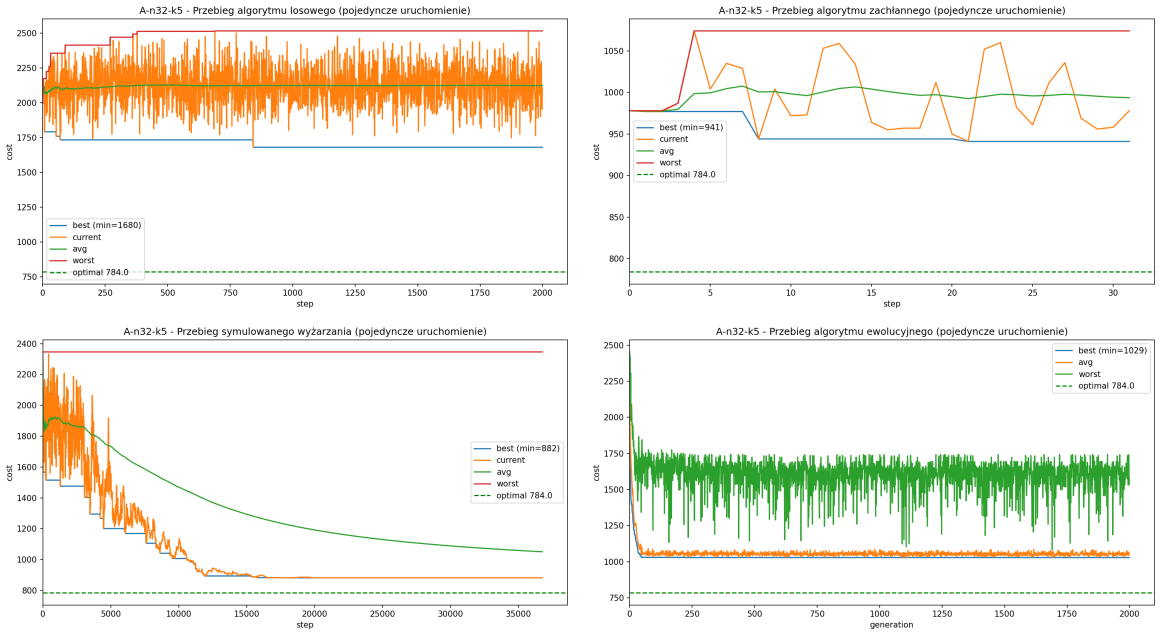
## 7.2 Wykresy

Poniżej przedstawiono wykresy dla trzech wybranych instancji. Dla każdej instancji pokazano:

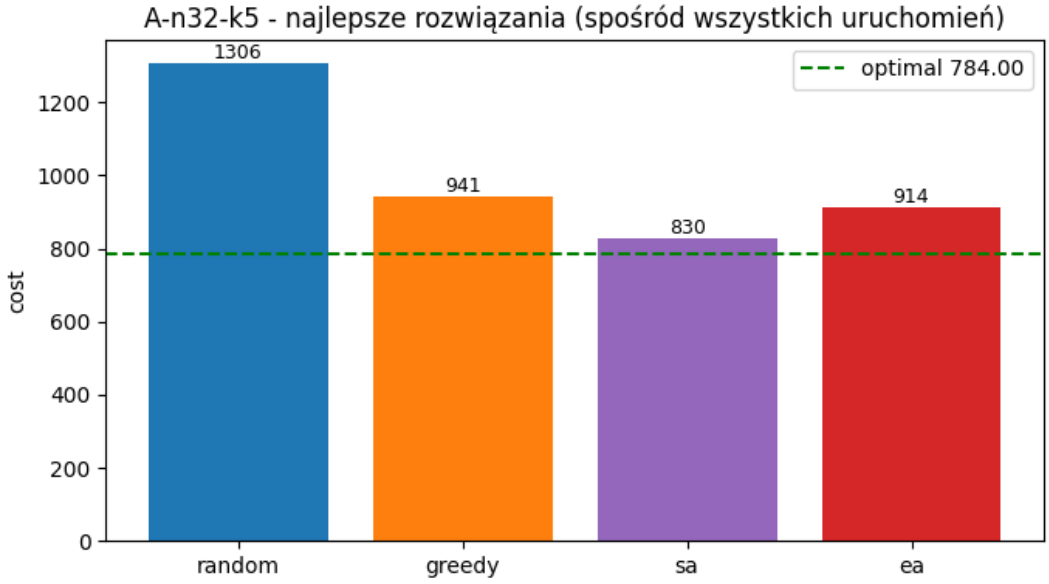
- Porównanie najlepszych przebiegów (best) wszystkich algorytmów (pojedyncze uruchomienie),
- Przebiegi: najlepsze (best) / bieżące (current) / średnie (avg) / najgorsze (worst) poszczególnych algorytmów (pojedyncze uruchomienie),
- Porównanie najlepszych wyników ze wszystkich uruchomień (wykres słupkowy).



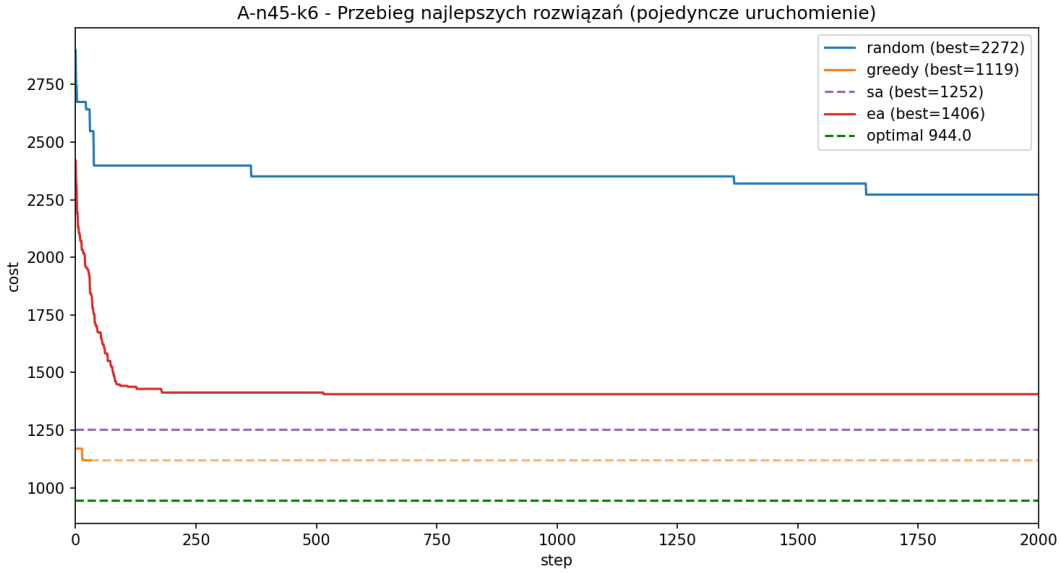

Rysunek 1: A-n32-k5: porównanie najlepszych przebiegów (best) wszystkich algorytmów (pojedyncze uruchomienie)



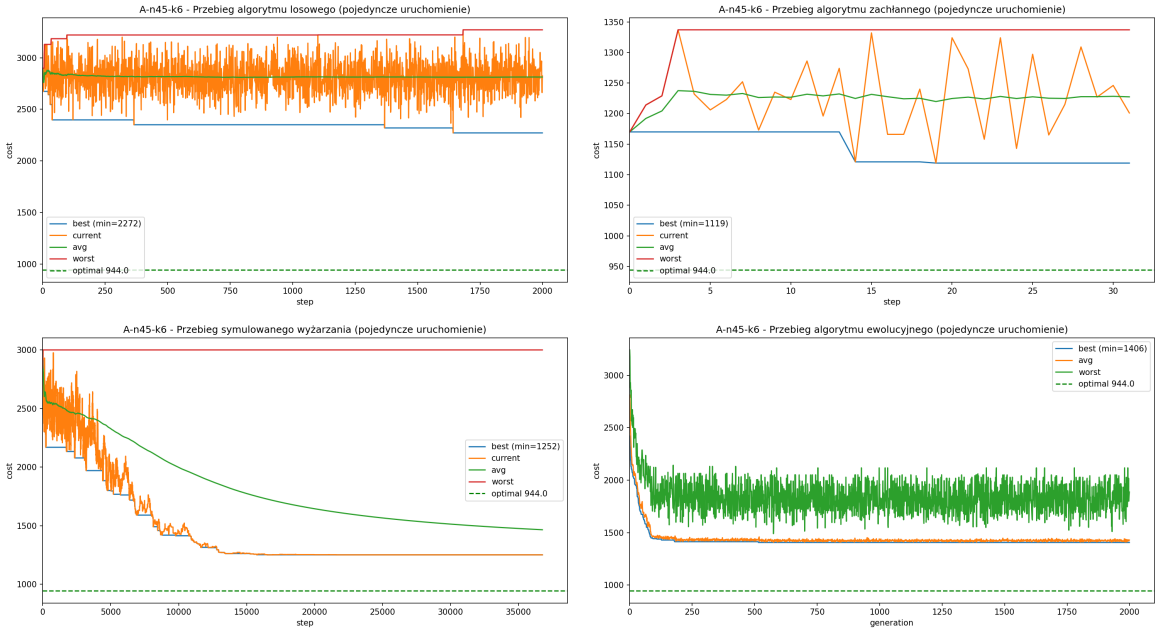

Rysunek 2: A-n32-k5: przebiegi poszczególnych algorytmów - najlepsze/bieżące/średnie/najgorsze (best/current/avg/worst) dla pojedynczego uruchomienia



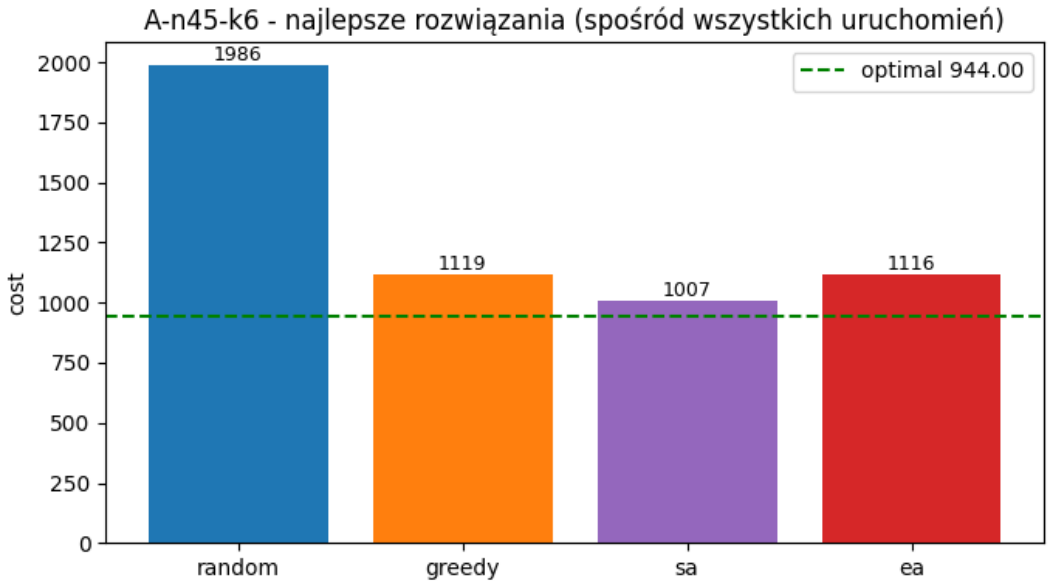

Rysunek 3: A-n32-k5: porównanie najlepszych wyników ze wszystkich uruchomień z linią optimum



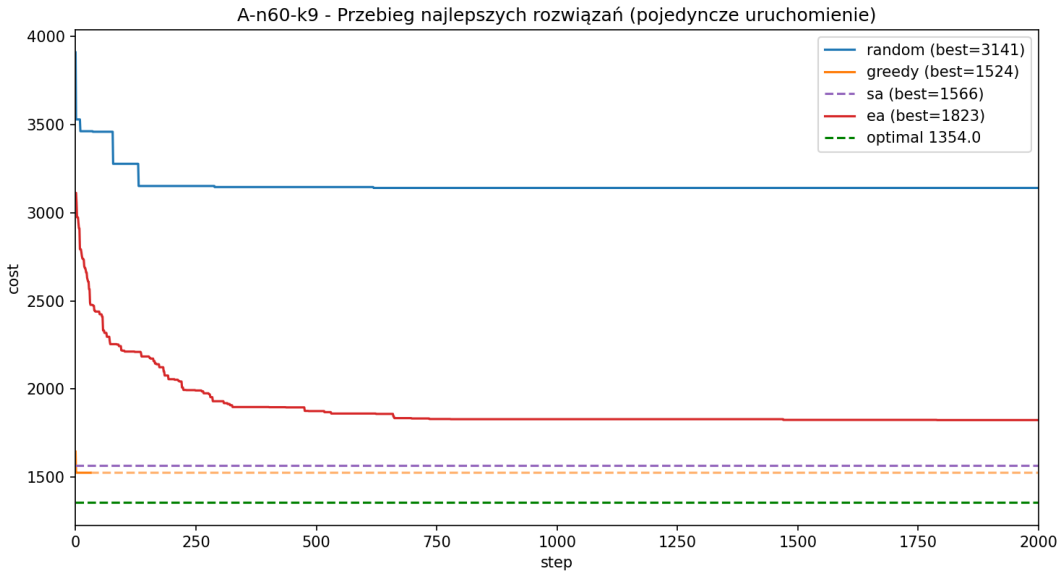

Rysunek 4: A-n45-k6: porównanie najlepszych przebiegów (best) wszystkich algorytmów (pojedyncze uruchomienie)



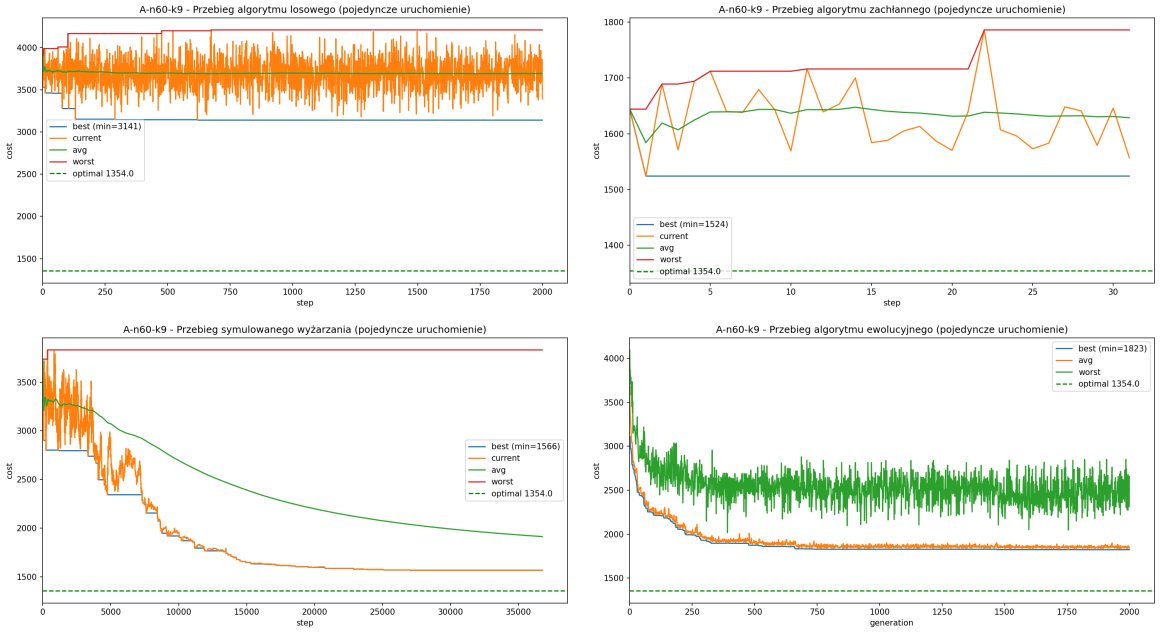

Rysunek 5: A-n45-k6: przebiegi poszczególnych algorytmów - najlepsze/bieżące/średnie/najgorsze (best/current/avg/worst) dla pojedynczego uruchomienia



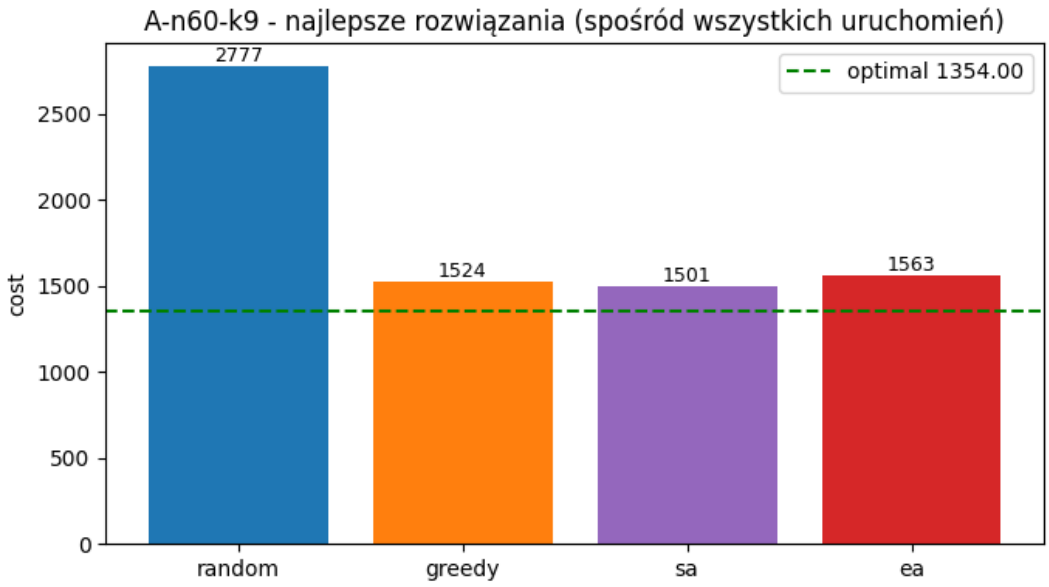

Rysunek 6: A-n45-k6: porównanie najlepszych wyników ze wszystkich uruchomień z linią optimum




Rysunek 7: A-n60-k9: porównanie najlepszych przebiegów (best) wszystkich algorytmów (pojedyncze uruchomienie)




Rysunek 8: A-n60-k9: przebiegi poszczególnych algorytmów - najlepsze/bieżące/średnie/najgorsze (best/current/avg/worst) dla pojedynczego uruchomienia




Rysunek 9: A-n60-k9: porównanie najlepszych wyników ze wszystkich uruchomień z linią optimum

### 7.3 Wnioski (etap bazowy)

- Greedy jest stabilny i często lepszy od niedostrojonych EA/SA przy obecnym krótkim budżecie obliczeń.
- SA w ustawieniach bazowych jest blisko optimum, wymaga więcej kroków/temperatur, by przebić greedy na trudniejszych instancjach.
- EA (OX+swap, mała populacja/pokolenia) przegrywa z greedy; potrzebne: większa populacja, bogatsze mutacje (inwersja/2-opt), lepsze krzyżowania (PMX/CX), inicjalizacja z greedy.
- Dalsze kroki: strojenie parametrów, dodanie nowych operatorów.

## 8 Wyniki badań po strojeniu

Przeprowadzono cztery etapy dostrajania parametrów algorytmów EA i SA, testując różne strategie optymalizacji. Każde strojenie koncentrowało się na innym aspekcie: lepsze operatory genetyczne, eksploatacja lokalna, czysta eksploracja oraz hybrydowe połączenie najlepszych cech.

### 8.1 Strojenie #1: Lepsze operatory + podstawowe wzmocnienie

#### 8.1.1 Cel i parametry

Celem pierwszego strojenia było wprowadzenie bardziej zaawansowanych operatorów genetycznych oraz wzmocnienie parametrów obu metaheurystyk. Kluczowe zmiany względem baseline:

**EA:**

- Zmiana operatorów krzyżowania: OX → PMX
- Zmiana operatorów mutacji: swap → inversion

- Populacja: 100 → 120
- Crossover rate: 0.7 → 0.85
- Mutation rate: 0.1 → 0.20
- Tournament: 5 → 3 (słabsza presja selekcyjna)
- Elites: 1 → 2

**SA:**

- Temperatura początkowa: 100 → 800 (8x wzrost)
- Iteracje na temperaturę: 20 → 150 (7.5x wzrost)
- Szacowana liczba kroków: ∼1800 → ∼16000

### 8.1.2 Wyniki

Tabela przedstawia statystyki wyników z 10 uruchomień EA i SA dla każdej instancji (format jak w sekcji baseline: Best/Worst/Avg/Std oznaczają odpowiednio najlepszy, najgorszy, średni i odchylenie standardowe wyników ze wszystkich uruchomień).

Tabela 2: Wyniki strojenia #1 - lepsze operatory (PMX + inversion)

| Instance | Optimal | Random Runs | Random Best | Random Worst | Random Avg | Random Std | Greedy Runs | Greedy Best | Greedy Worst | Greedy Avg | Greedy Std | EA Runs | EA Best | EA Worst | EA Avg | EA Std | SA Runs | SA Best | SA Worst | SA Avg | SA Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A-n32-k5 | 784 | 1000 | 1404 | 1793 | 1655.46 | 52.12 | 32 | 941 | 941 | 941 | 0.00 | 10 | 807 | 926 | 884.2 | 38.32 | 10 | 796 | 863 | 829.6 | 17.14 |
| A-n37-k6 | 949 | 1000 | 1602 | 1893 | 1788.62 | 46.86 | 37 | 1058 | 1058 | 1058 | 0.00 | 10 | 977 | 1057 | 1017.8 | 24.21 | 10 | 963 | 1013 | 989.8 | 17.74 |
| A-n39-k5 | 822 | 1000 | 1527 | 1851 | 1745.25 | 48.75 | 39 | 920 | 920 | 920 | 0.00 | 10 | 857 | 1017 | 921.8 | 54.01 | 10 | 838 | 941 | 878.2 | 31.29 |
| A-n45-k6 | 944 | 1000 | 1916 | 2434 | 2295.73 | 60.24 | 45 | 1119 | 1119 | 1119 | 0.00 | 10 | 1008 | 1216 | 1131.4 | 66.37 | 10 | 971 | 1098 | 1007.8 | 37.88 |
| A-n48-k7 | 1073 | 1000 | 2155 | 2526 | 2388.64 | 59.29 | 48 | 1301 | 1301 | 1301 | 0.00 | 10 | 1237 | 1353 | 1295.3 | 37.68 | 10 | 1120 | 1188 | 1162.1 | 21.15 |
| A-n54-k7 | 1167 | 1000 | 2453 | 2864 | 2722.84 | 60.67 | 54 | 1380 | 1380 | 1380 | 0.00 | 10 | 1250 | 1450 | 1361.1 | 62.44 | 10 | 1232 | 1317 | 1262 | 27.39 |
| A-n60-k9 | 1354 | 1000 | 2762 | 3243 | 3103.12 | 65.95 | 60 | 1524 | 1524 | 1524 | 0.00 | 10 | 1558 | 1753 | 1638.6 | 50.02 | 10 | 1390 | 1532 | 1473.9 | 38.56 |

### 8.1.3 Analiza wyników

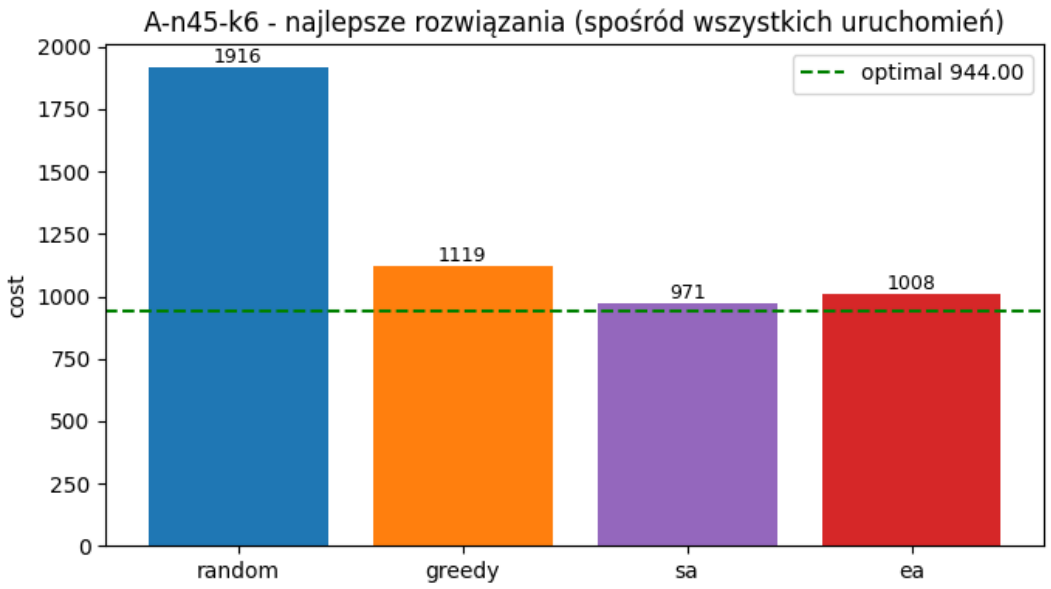
**Poprawa względem baseline:**

- **EA:** Średnia najlepszych wyników (z kolumny best dla wszystkich 7 instancji): 1167.1 → 1099.1 (poprawa o 5.8%). Wprowadzenie operatorów PMX i inversion znacząco poprawiło jakość rozwiązań.
- **SA:** Średnia najlepszych wyników: 1100.1 → 1044.3 (poprawa o 5.1%). Zwiększenie budżetu obliczeń pozwoliło na lepszą eksplorację przestrzeni.
- **EA vs Greedy:** EA zaczął osiągać lepsze wyniki niż greedy (np. A-n32-k5: 807 vs 941).

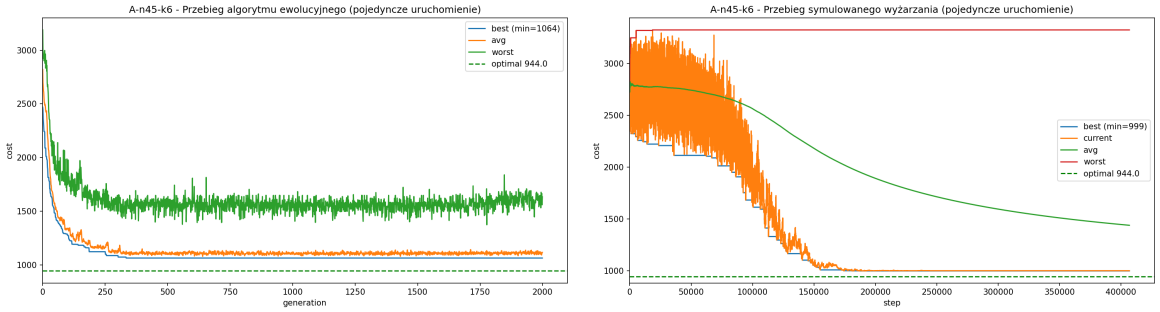
**Wnioski:**

- Operatory PMX i inversion są wyraźnie lepsze niż OX i swap dla problemu CVRP:
  - PMX lepiej zachowuje względne pozycje klientów (ważne dla struktury tras)
  - Inversion wprowadza większe zmiany niż swap, co poprawia eksplorację przestrzeni rozwiązań
- Zwiększenie budżetu obliczeń SA dało wyraźną poprawę bez ryzyka przedwczesnej zbieżności.
- Zmniejszenie presji selekcyjnej (tournament 5→3) pomogło w utrzymaniu różnorodności populacji.

### 8.1.4 Wykresy (A-n45-k6)




Rysunek 10: Strojenie #1 (A-n45-k6): porównanie najlepszych wyników ze wszystkich uruchomień




Rysunek 11: Strojenie #1 (A-n45-k6): przebiegi pojedynczych uruchomień EA i SA

## 8.2 Strojenie #2: Eksploatacja lokalna

### 8.2.1 Cel i parametry

Celem drugiego strojenia było przetestowanie czy eksploatacja lokalna (greedy seeding + 2-opt) poprawi wyniki EA. Wprowadzono mechanizmy szybkiego znajdowania dobrych rozwiązań poprzez:

**EA:**

- Populacja: 120 → 150
- Generacje: 2000 → 2500
- Crossover rate: 0.85 → 0.90
- Mutation rate: 0.20 → 0.15 (mniej eksploracji)
- Tournament: 3 → 5 (silniejsza presja)
- Elites: 2 → 3
- **Greedy init: 0.0 → 0.3** (30% populacji z greedy)
- **2-opt rate: 0.0 → 0.2** (lokalna poprawa)

**SA:**

- Temperatura początkowa: 800 → 1400
- Minimalna temperatura: 0.001 → 0.0005
- Cooling rate: 0.995 → 0.996
- Iteracje na temperaturę: 150 → 200
- Szacowana liczba kroków: ∼16000 → ∼65000

### 8.2.2 Wyniki

Tabela 3: Wyniki strojenia #2 (eksploatacja)

| Instance | Optimal | Random Runs | Random Best | Random Worst | Random Avg | Random Std | Greedy Runs | Greedy Best | Greedy Worst | Greedy Avg | Greedy Std | EA Runs | EA Best | EA Worst | EA Avg | EA Std | SA Runs | SA Best | SA Worst | SA Avg | SA Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A-n32-k5 | 784 | 1000 | 1428 | 1773 | 1655.2 | 52.79 | 32 | 941 | 941 | 941 | 0.00 | 10 | 811 | 889 | 847.4 | 22.60 | 10 | 807 | 880 | 835.9 | 19.81 |
| A-n37-k6 | 949 | 1000 | 1623 | 1890 | 1789.93 | 44.82 | 37 | 1058 | 1058 | 1058 | 0.00 | 10 | 970 | 989 | 976.8 | 6.85 | 10 | 956 | 1007 | 978.4 | 15.07 |
| A-n39-k5 | 822 | 1000 | 1567 | 1856 | 1748.07 | 46.63 | 39 | 920 | 920 | 920 | 0.00 | 10 | 834 | 858 | 844 | 7.39 | 10 | 830 | 875 | 850.4 | 14.45 |
| A-n45-k6 | 944 | 1000 | 2001 | 2435 | 2300.71 | 59.04 | 45 | 1119 | 1119 | 1119 | 0.00 | 10 | 992 | 1063 | 1021.2 | 26.05 | 10 | 960 | 1042 | 1000.1 | 23.85 |
| A-n48-k7 | 1073 | 1000 | 2183 | 2517 | 2388.07 | 57.76 | 48 | 1301 | 1301 | 1301 | 0.00 | 10 | 1136 | 1239 | 1190.5 | 31.67 | 10 | 1114 | 1173 | 1153.3 | 14.97 |
| A-n54-k7 | 1167 | 1000 | 2347 | 2866 | 2723.43 | 60.82 | 54 | 1380 | 1380 | 1380 | 0.00 | 10 | 1226 | 1282 | 1248.1 | 16.11 | 10 | 1201 | 1290 | 1252.1 | 25.09 |
| A-n60-k9 | 1354 | 1000 | 2880 | 3243 | 3106.46 | 62.77 | 60 | 1524 | 1524 | 1524 | 0.00 | 10 | 1376 | 1421 | 1394 | 12.17 | 10 | 1407 | 1490 | 1447.7 | 27.93 |

### 8.2.3 Analiza wyników

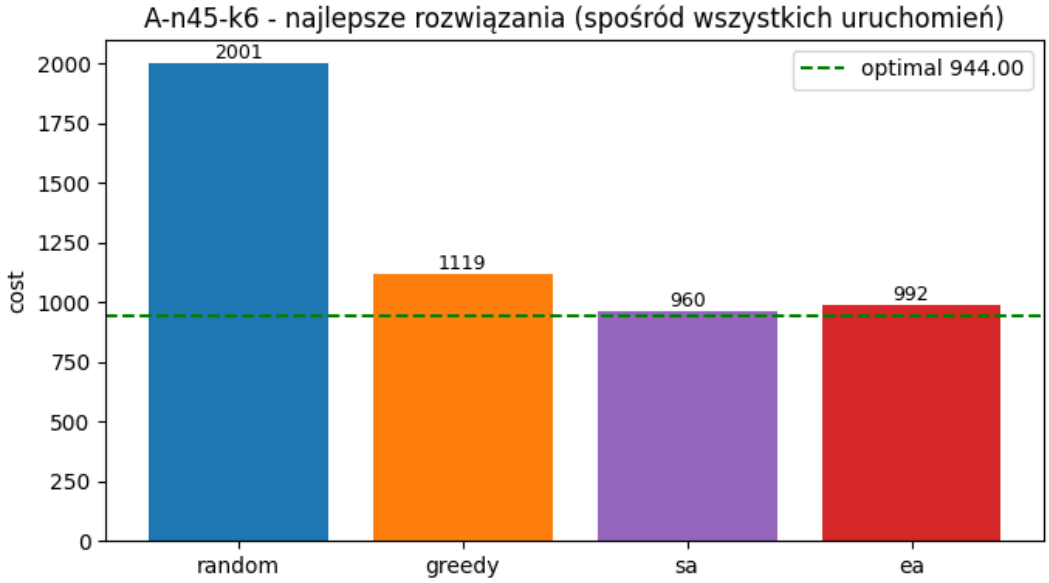
**Poprawa względem strojenia #1:**

- **EA:** Średnia najlepszych wyników (z kolumny best dla wszystkich 7 instancji): 1099.1 → 1049.3 (poprawa o 4.5%). Najlepszy wynik EA spośród wszystkich strojeń!
- **SA:** Średnia najlepszych wyników: 1044.3 → 1039.3 (poprawa o 0.5%).
- **Stabilność EA:** Zauważalnie mniejsze odchylenie standardowe (np. A-n54-k7: 62.44 → 16.11).

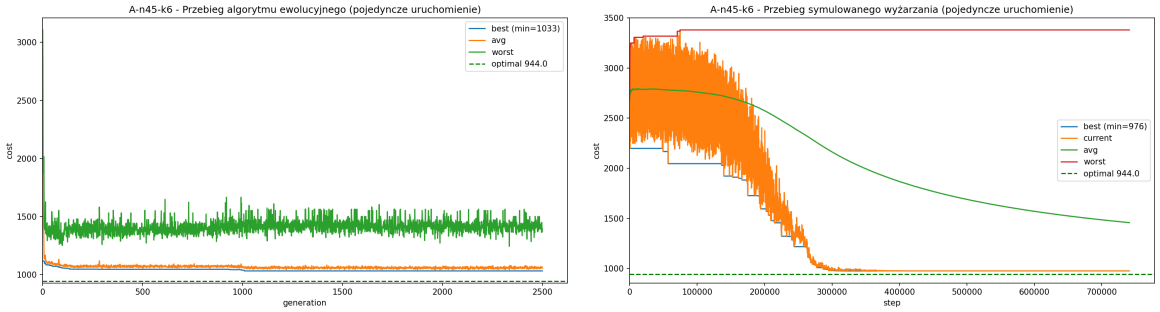
**Wnioski:**

- **Zaskakujący wynik:** Eksploatacja (greedy seeding + 2-opt) faktycznie pomogła EA osiągnąć najlepsze średnie wyniki.
- Greedy seeding (30% populacji) pozwolił na szybsze znalezienie dobrych obszarów przestrzeni rozwiązań.
- Lokalna poprawa 2-opt skutecznie dopracowywała rozwiązania przed oceną.
- Ryzyko: analiza przebiegów EA pokazuje, że silna eksploatacja (strojenie #2) prowadzi do przedwczesnej zbieżności - najlepszy wynik osiągnął wartość 1045 już w połowie eksperymentu i nie poprawił się w drugiej połowie, a avg zbliżyło się do best (różnica 1%), co wskazuje na utratę różnorodności populacji.

### 8.2.4 Wykresy (A-n45-k6)




Rysunek 12: Strojenie #2 (A-n45-k6): porównanie najlepszych wyników ze wszystkich uruchomień




Rysunek 13: Strojenie #2 (A-n45-k6): przebiegi pojedynczych uruchomień EA i SA

## 8.3 Strojenie #3: Czysta eksploracja

### 8.3.1 Cel i parametry

Trzecie strojenie testowało przeciwne podejście: czystą eksplorację bez mechanizmów eksploatacji. Celem było sprawdzenie czy wolniejsza, ale szersza eksploracja przestrzeni da lepsze wyniki niż eksploatacja ze strojenia #2.

**EA:**

- Populacja: 150 → 200 (większa różnorodność)
- Generacje: 2500 → 4000 (więcej czasu)
- Crossover rate: 0.90 → 0.75 (więcej czystej mutacji)
- Mutation rate: 0.15 → 0.30 (dwukrotnie wyższa)
- Tournament: 5 → 2 (minimalna presja)
- Elites: 3 → 1
- **Greedy init: 0.3 → 0.0** (bez greedy seeding)
- **2-opt rate: 0.2 → 0.0** (bez lokalnej poprawy)

**SA:**

- Temperatura początkowa: 1400 → 1800 (maksymalna)
- Minimalna temperatura: 0.0005 (bez zmian)
- Cooling rate: 0.996 → 0.9975 (najwolniejsze chłodzenie)
- Iteracje na temperaturę: 200 → 250 (maksimum)
- Szacowana liczba kroków: ∼65000 → ∼400000

### 8.3.2 Wyniki

Tabela 4: Wyniki strojenia #3 (czysta eksploracja)

| Instance | Optimal | Random Runs | Random Best | Random Worst | Random Avg | Random Std | Greedy Runs | Greedy Best | Greedy Worst | Greedy Avg | Greedy Std | EA Runs | EA Best | EA Worst | EA Avg | EA Std | SA Runs | SA Best | SA Worst | SA Avg | SA Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A-n32-k5 | 784 | 1000 | 1472 | 1766 | 1654.27 | 50.15 | 32 | 941 | 941 | 941 | 0.00 | 10 | 814 | 873 | 843.5 | 17.51 | 10 | 801 | 852 | 817.2 | 16.78 |
| A-n37-k6 | 949 | 1000 | 1577 | 1888 | 1789.28 | 45.09 | 37 | 1058 | 1058 | 1058 | 0.00 | 10 | 975 | 1057 | 1008.1 | 25.67 | 10 | 949 | 1018 | 972.8 | 19.96 |
| A-n39-k5 | 822 | 1000 | 1507 | 1879 | 1749.35 | 46.68 | 39 | 920 | 920 | 920 | 0.00 | 10 | 833 | 908 | 873.6 | 27.88 | 10 | 839 | 888 | 853.8 | 12.79 |
| A-n45-k6 | 944 | 1000 | 2011 | 2438 | 2303.01 | 56.27 | 45 | 1119 | 1119 | 1119 | 0.00 | 10 | 953 | 1077 | 1022.8 | 37.16 | 10 | 973 | 1045 | 1003.1 | 21.33 |
| A-n48-k7 | 1073 | 1000 | 2100 | 2528 | 2390.14 | 57.34 | 48 | 1301 | 1301 | 1301 | 0.00 | 10 | 1112 | 1262 | 1169.1 | 43.45 | 10 | 1118 | 1171 | 1142.7 | 17.11 |
| A-n54-k7 | 1167 | 1000 | 2476 | 2872 | 2723.63 | 60.37 | 54 | 1380 | 1380 | 1380 | 0.00 | 10 | 1221 | 1342 | 1288.8 | 37.52 | 10 | 1190 | 1240 | 1218.2 | 16.19 |
| A-n60-k9 | 1354 | 1000 | 2797 | 3252 | 3104.53 | 63.49 | 60 | 1524 | 1524 | 1524 | 0.00 | 10 | 1475 | 1586 | 1502.3 | 30.94 | 10 | 1391 | 1473 | 1413.5 | 27.02 |

### 8.3.3 Analiza wyników

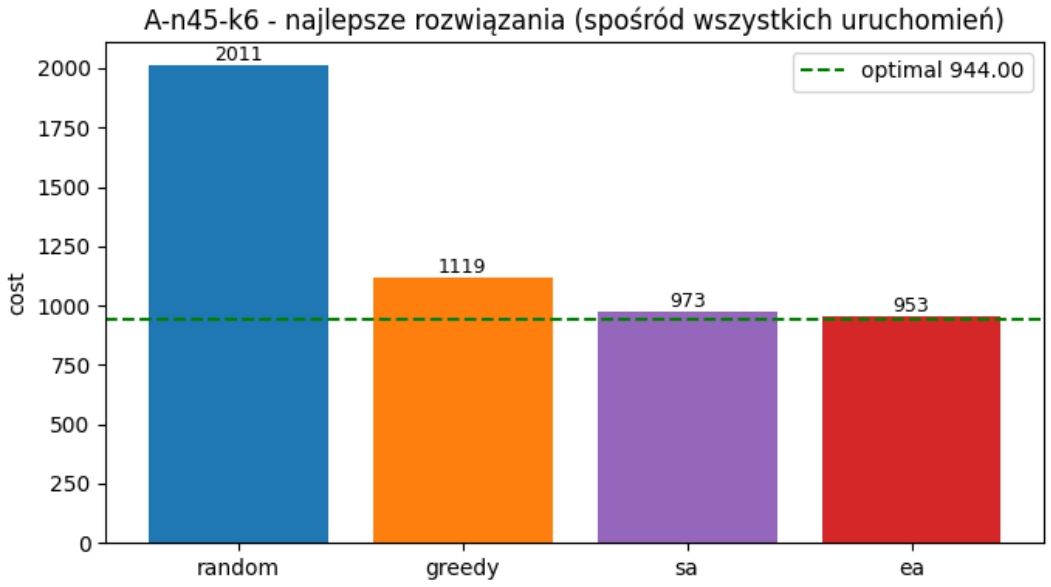
**Porównanie ze strojeniem #2:**

- **EA:** Średnia najlepszych wyników (best): 1049.3 → 1054.7 (pogorszenie o 0.5%). Czysta eksploracja była nieznacznie gorsza.
- **SA:** Średnia najlepszych wyników (best): 1039.3 → 1037.3 (poprawa o 0.2%). Najlepszy wynik SA!
- **Stabilność:** EA ma większe odchylenie standardowe (eksploracja vs eksploatacja).

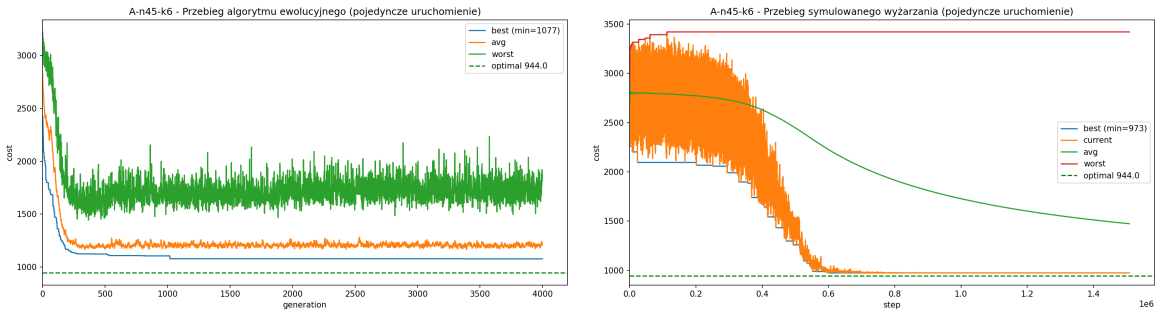
**Wnioski:**

- Czysta eksploracja dała nieznacznie gorsze wyniki niż eksploatacja w (strojeniu #2) dla EA. Jedyny zysk z eksploracji zauważono dla średnio trudnych instancji.
- SA skorzystał na maksymalnych parametrach - długi budżet obliczeń pozwolił na dogłębną eksplorację.
- EA: brak greedy seeding oznaczał start z gorszych rozwiązań, co wymagało więcej pokoleń na osiągnięcie dobrych wyników.
- Wysoka mutacja (0.30) mogła zakłócać konwergencję do optymalnych rozwiązań.

### 8.3.4 Wykresy (A-n45-k6)




Rysunek 14: Strojenie #3 (A-n45-k6): porównanie najlepszych wyników ze wszystkich uruchomień




Rysunek 15: Strojenie #3 (A-n45-k6): przebiegi pojedynczych uruchomień EA i SA

## 8.4 Strojenie #4: Hybrydowe połączenie

### 8.4.1 Cel i parametry

Czwarte strojenie łączy najlepsze cechy eksploatacji (strojenie #2) i eksploracji (strojenie #3). Na podstawie wyników poprzednich eksperymentów zaprojektowano zbalansowane parametry:

**EA:**

- Populacja: 180 (między 150 a 200)
- Generacje: 3500 (między 2500 a 4000)
- Crossover rate: 0.82 (między 0.90 a 0.75)
- Mutation rate: 0.23 (między 0.15 a 0.30)
- Tournament: 3 (między 5 a 2)
- Elites: 2 (między 3 a 1)
- **Greedy init: 0.15** (połowa ze strojenia #2)
- **2-opt rate: 0.10** (połowa ze strojenia #2)

**SA:**

- Parametry jak w strojeniu #3 (maksymalne, udowodniony sukces)
- Temperatura: 1800, min: 0.0005, cooling: 0.9975, iter/temp: 250

### 8.4.2 Wyniki

Tabela 5: Wyniki strojenia #4 (hybryda)

| Instance | Optimal | Random Runs | Random Best | Random Worst | Random Avg | Random Std | Greedy Runs | Greedy Best | Greedy Worst | Greedy Avg | Greedy Std | EA Runs | EA Best | EA Worst | EA Avg | EA Std | SA Runs | SA Best | SA Worst | SA Avg | SA Std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A-n32-k5 | 784 | 1000 | 1414 | 1767 | 1656.6 | 49.36 | 32 | 941 | 941 | 941 | 0.00 | 10 | 829 | 866 | 838.4 | 11.83 | 10 | 801 | 875 | 826.8 | 21.66 |
| A-n37-k6 | 949 | 1000 | 1601 | 1904 | 1787.76 | 46.74 | 37 | 1058 | 1058 | 1058 | 0.00 | 10 | 970 | 983 | 974.4 | 4.01 | 10 | 953 | 992 | 971.8 | 12.87 |
| A-n39-k5 | 822 | 1000 | 1483 | 1856 | 1745.98 | 48.33 | 39 | 920 | 920 | 920 | 0.00 | 10 | 834 | 851 | 840.7 | 5.62 | 10 | 835 | 876 | 853.2 | 14.77 |
| A-n45-k6 | 944 | 1000 | 2079 | 2430 | 2301.06 | 57.05 | 45 | 1119 | 1119 | 1119 | 0.00 | 10 | 962 | 1048 | 994.9 | 21.35 | 10 | 947 | 1026 | 983.7 | 21.87 |
| A-n48-k7 | 1073 | 1000 | 2125 | 2517 | 2385.71 | 60.70 | 48 | 1301 | 1301 | 1301 | 0.00 | 10 | 1154 | 1207 | 1186.6 | 14.75 | 10 | 1107 | 1183 | 1146.1 | 21.62 |
| A-n54-k7 | 1167 | 1000 | 2490 | 2885 | 2726.76 | 60.02 | 54 | 1380 | 1380 | 1380 | 0.00 | 10 | 1206 | 1257 | 1235.3 | 16.91 | 10 | 1190 | 1286 | 1231.2 | 32.70 |
| A-n60-k9 | 1354 | 1000 | 2855 | 3238 | 3108.23 | 61.58 | 60 | 1524 | 1524 | 1524 | 0.00 | 10 | 1374 | 1423 | 1398.6 | 18.32 | 10 | 1380 | 1474 | 1414.9 | 25.57 |

### 8.4.3 Analiza wyników

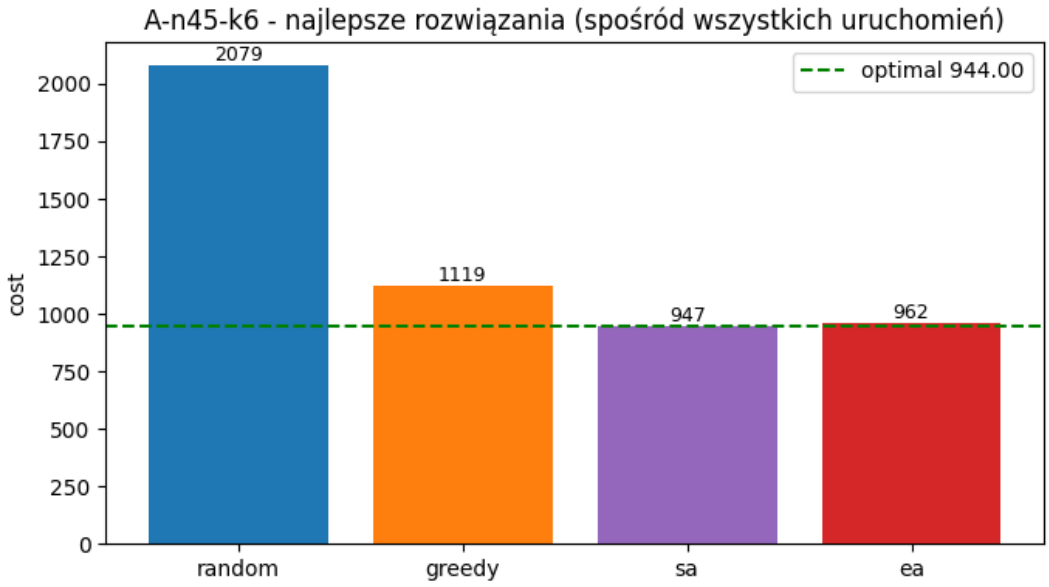
**Porównanie z najlepszymi wynikami:**

- **EA:** Średnia najlepszych wyników (z kolumny best dla wszystkich 7 instancji): 1048.1 - między strojeniem #2 (1049.3) a #3 (1054.7).
- **SA:** Średnia najlepszych wyników: 1041.0 - nieznacznie gorszy niż strojenie #3 (1037.3).
- **Stabilność EA:** Bardzo niskie odchylenia standardowe (np. A-n37-k6: 4.01), najlepsze spośród wszystkich strojeń.

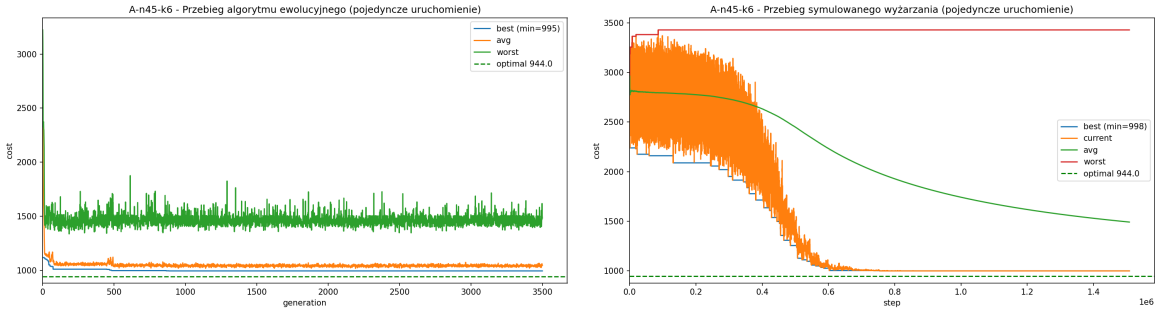
**Wnioski końcowe:**

- Hybrydowe podejście osiągnęło bardzo stabilne wyniki (niskie std dev) dzięki zbalansowaniu eksploracji i eksploatacji.
- Umiarkowana eksploatacja (greedy 15%, 2-opt 10%) okazała się dobrym kompromisem.
- Najlepsze absolutne wyniki EA pochodzą ze strojenia #2 (eksploatacja), ale kosztem większej zmienności. Zwiększenie budżetu obliczeniowego w podejściu hybrydowym mogłoby przynieść poprawę.
- SA osiągnął plateau w strojeniu #3 - dalsze zwiększanie parametrów prawdopodobnie nie przyniesie poprawy.

### 8.4.4 Wykresy (A-n45-k6)




Rysunek 16: Strojenie #4 (A-n45-k6): porównanie najlepszych wyników ze wszystkich uruchomień




Rysunek 17: Strojenie #4 (A-n45-k6): przebiegi pojedynczych uruchomień EA i SA

## 8.5 Podsumowanie strojeń

Tabela 6: Porównanie średnich najlepszych wyników (best) spośród wszystkich strojeń

| Strojenie | EA Best (śr.) | SA Best (śr.) | EA std | SA std |
|---|---|---|---|---|
| Baseline | 1167.1 | 1100.1 | 64.7 | 41.3 |
| #1 (operatory) | 1099.1 | 1044.3 | 48.1 | 27.4 |
| #2 (eksploatacja) | **1049.3** | 1039.3 | 17.6 | 20.5 |
| #3 (eksploracja) | 1054.7 | **1037.3** | 31.5 | 18.6 |
| #4 (hybryda) | 1048.1 | 1041.0 | **13.3** | 21.6 |

**Kluczowe obserwacje:**

1. Wprowadzenie operatorów PMX i inversion (strojenie #1) dało największy skok jakości (5.8% dla EA).
2. Eksploatacja (strojenie #2) wygrała z czystą eksploracją (strojenie #3) w EA, mimo że strojenie #3 miało 60% więcej generacji (4000 vs 2500). Pokazuje to, że balans między eksploracją a eksploatacją jest kluczowy - zbyt słaba presja selekcyjna i brak lokalnej poprawy prowadzą do gorszych wyników nawet przy dłuższym czasie obliczeń.
3. SA konsekwentnie poprawiał się z każdym strojeniem, osiągając optimum w strojeniu #3.
4. Hybrydowe podejście (strojenie #4) dało najbardziej stabilne wyniki przy minimalnie gorszej średniej.

**Najlepszy algorytm dla CVRP:**

W przeprowadzonym badaniu Symulowane Wyżarzanie (SA) okazało się najbardziej skuteczną metaheurystyką dla CVRP, osiągając konsekwentnie lepsze wyniki od Algorytmu Ewolucyjnego we wszystkich konfiguracjach (średnia przewaga 1-6%). Szczególnie imponujący był wynik w konfiguracji baseline, gdzie SA osiągnęło lepsze rezultaty przy 5x mniejszym budżecie ewaluacji (37,000 vs 200,000). W tym miejscu należy wspomnieć, że różne budżety obliczeniowe w strojeniach utrudniają bezpośrednie porównanie jakości algorytmów. W strojeniach zaawansowanych (#2-#4) różnice jakości się zmniejszyły - SA miało nieznaczną przewagę, natomiast korzystało z większego budżetu obliczeniowego. Niemniej jednak, SA wyróżnia się prostotą implementacji, mniejszą liczbą hiperparametrów oraz brakiem kosztownych operacji genetycznych, co w całokształcie czyni je praktycznym wyborem dla problemów CVRP.