Algorytmy Optymalizacji Inspirowane Naturą
Projekt startowy

Jakub Wasilewski 263852

20.11.2025

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
- Algorytm zachłanny
- Symulowane wyżarzanie
- Algorytm ewolucyjny

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

Start: losowa permutacja klientów generowana `random_iterations` razy w jednym uruchomieniu. Każde rozwiązanie dekodowane i oceniane. Log: w każdej iteracji zapisujemy rozwiązanie najlepsze dotąd, bieżące, średnią i najgorsze (best/current/avg/worst).

4.3 Algorytm zachłanny

Start: wybieramy różne punkty startowe (rotacja po klientach). W każdej konstrukcji wybieramy najbliższego nieodwiedzonego klienta (nearest-neighbour), aż odwiedzimy wszystkich. Parametr `greedy_restarts` określa liczbę startów dla jednego uruchomienia (domyślnie liczba klientów). Log: po każdym restarcie zapisujemy best/current/avg/worst.

4.4 Symulowane wyżarzanie (SA)

Rozwiąznaie startowe: permutacja zachłanna od losowego węzła startowego. Sąsiedztwo: zamiana dwóch genów (swap). Akceptacja rozwiązania: jeśli koszt lepszy lub z prawdopodobieństwem $\exp(-\Delta/T)$. Parametry: temperatura początkowa T_0, minimalna $T_{\min}$, współczynnik chłodzenia α, liczba iteracji na temperaturę. Log: best/current/avg/worst w każdej iteracji, co pozwala obserwować skoki akceptacji gorszych ruchów.

4.5 Algorytm ewolucyjny (EA)

- Inicjalizacja: losowe permutacje (pop_size), każda dekodowana i oceniana.
- Selekcja: turniejowa, rozmiar `ea_tournament`; większa wartość zwiększa ciśnienie selekcyjne.
- Krzyżowanie: Ordered Crossover (OX) z prawdopodobieństwem Px; w przeciwnym razie kopiujemy rodzica.
- Mutacja: swap dwóch genów z prawdopodobieństwem Pm;
- Elitaryzm: kopiowanie najlepszych `ea_elites` osobników do następnego pokolenia bez zmian.
- Parametry: `ea_population`, `ea_generations`, `ea_crossover_rate`, `ea_mutation_rate`, `ea_tournament`, `ea_elites`.
- Log: best/avg/worst w każdym pokoleniu (populacja), co 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 per-run w CSV, 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=20 (iteracje=1000), greedy_runs=N (restarts=32), sa_runs=15 ($T_0 = 100$, $T_{\min} = 0{,}01$, $\alpha = 0{,}995$, iter/temp=200), ea_runs=15 (pop=100, gen=100, Px=0,7, Pm=0,1, tour=5, elites=1).

- Wizualizacje: `plot_logs.py` generuje wykresy single/combined/bar_best do `zadanego w zmiennej folderu/`.

7 Wyniki badań przed tuningiem

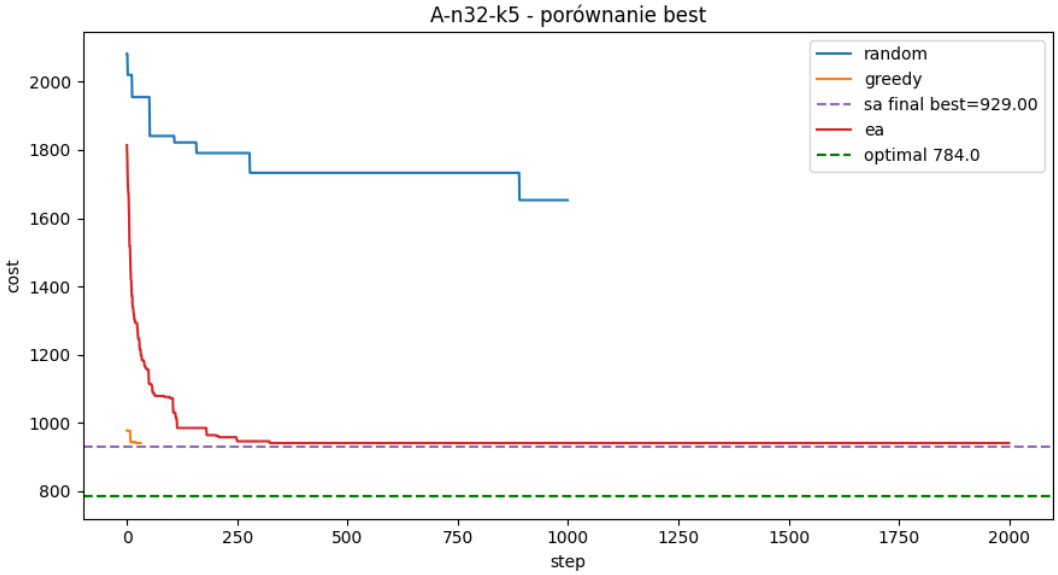
7.1 Tabela zbiorcza (logs_baseline/summary.csv)

Tabela 1: Zbiorcze statystyki (przed tuningiem)

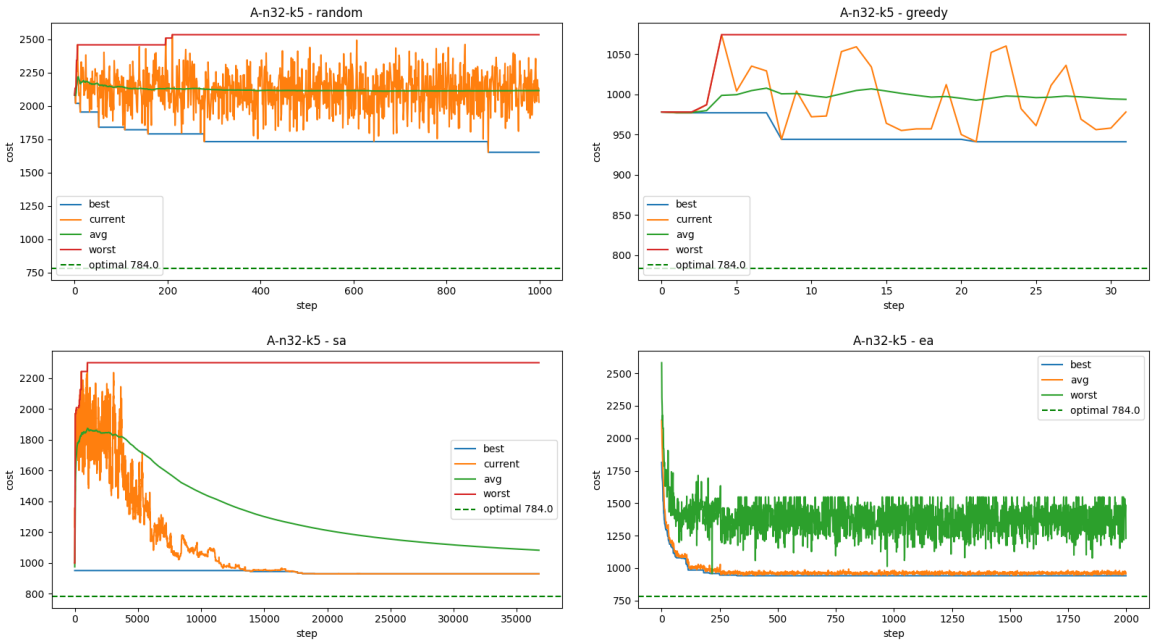
Instance	Optimal	Random Runs	Random Best	Random Worst	Random Avg	Random Std	Greedy Runs	Greedy Best	Greedy Worst	Greedy Avg	Greedy Std	EA Runs	EA Best	EA Worst	EA Avg	EA Std	SA Runs	SA Best	SA Worst	SA Avg	SA Std
A-n54-k7	1167	1000	2449	2848	2724.51	60.2820	54	1380	1380	1380	0.00	10	1409	1644	1500.80	81.8997	10	1273	1446	1376.30	44.8844
A-n39-k5	822	1000	1552	1865	1748.80	45.3239	39	920	920	920	0.00	10	969	1105	1049.90	48.8190	10	907	1004	945.10	32.2380
A-n32-k5	784	1000	1420	1781	1655.76	49.8083	32	941	941	941	0.00	10	941	1041	991.40	29.4761	10	842	949	895.70	29.4824
A-n37-k6	949	1000	1549	1880	1788.69	45.1863	37	1058	1058	1058	0.00	10	1039	1242	1119.90	57.5994	10	967	1097	1038.10	36.8658
A-n48-k7	1073	1000	2166	2549	2390.06	56.3475	48	1301	1301	1301	0.00	10	1246	1415	1345.00	52.5928	10	1188	1315	1241.40	35.7413
A-n60-k9	1354	1000	2834	3226	3104.78	64.5207	60	1524	1524	1524	0.00	10	1605	1805	1695.80	63.2784	10	1480	1604	1549.10	36.3496
A-n45-k6	944	1000	2077	2428	2303.96	55.3456	45	1119	1119	1119	0.00	10	1131	1400	1228.70	82.5022	10	1027	1160	1100.40	39.5505

7.2 Wybrane wykresy

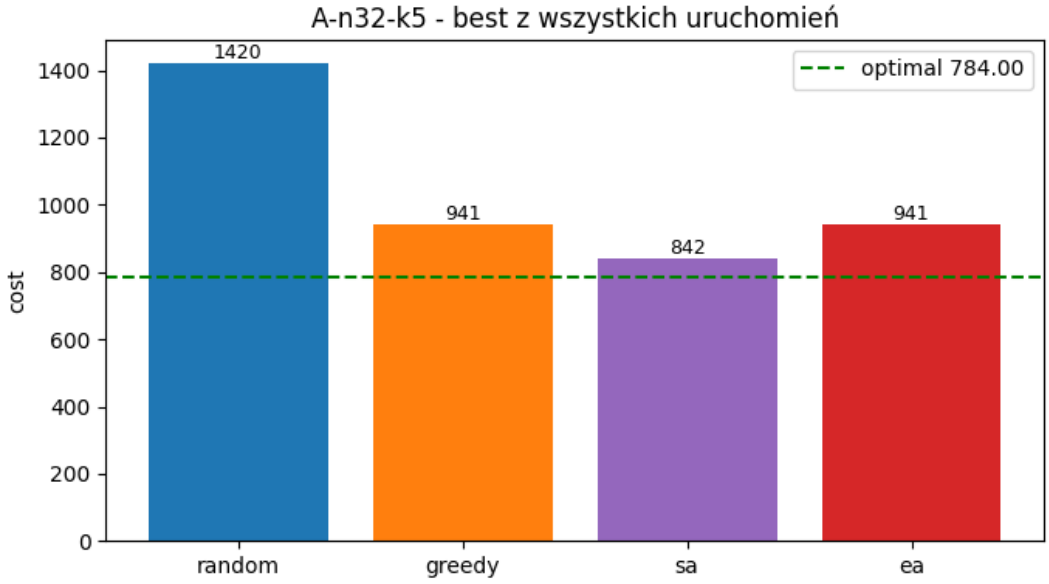
Poniżej trzy przykładowe zestawy wykresów (instancje A-n32-k5, A-n45-k6, A-n60-k9). Dla każdej instancji pokazano: (1) przebiegi best wszystkich algorytmów, (2) przebiegi pojedynczych algorytmów (best/current/avg/worst), (3) słupki najlepszych wyników z wielu uruchomień.



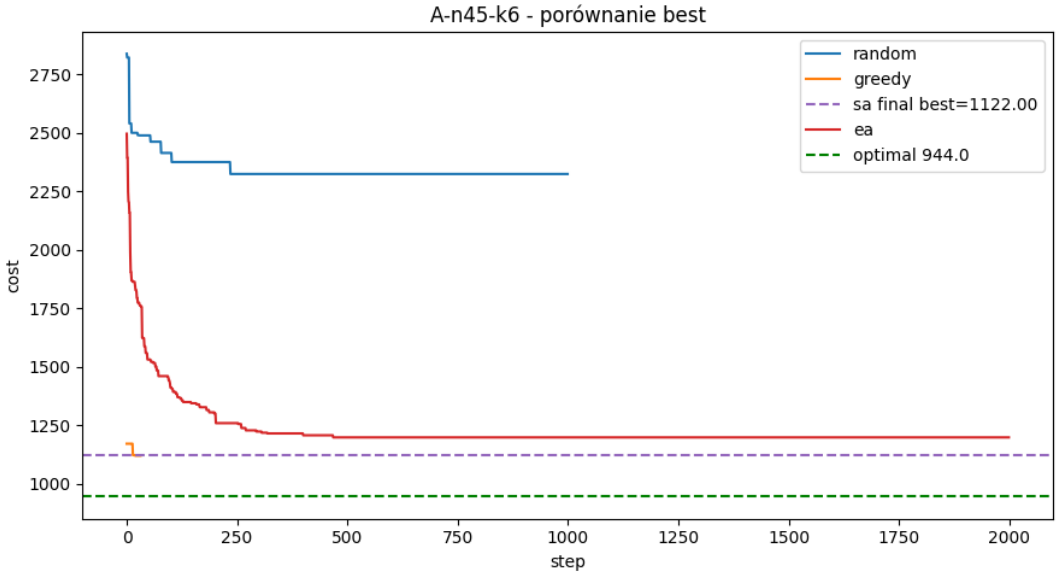
Rysunek 1: A-n32-k5: porównanie przebiegów najlepszych wyników.



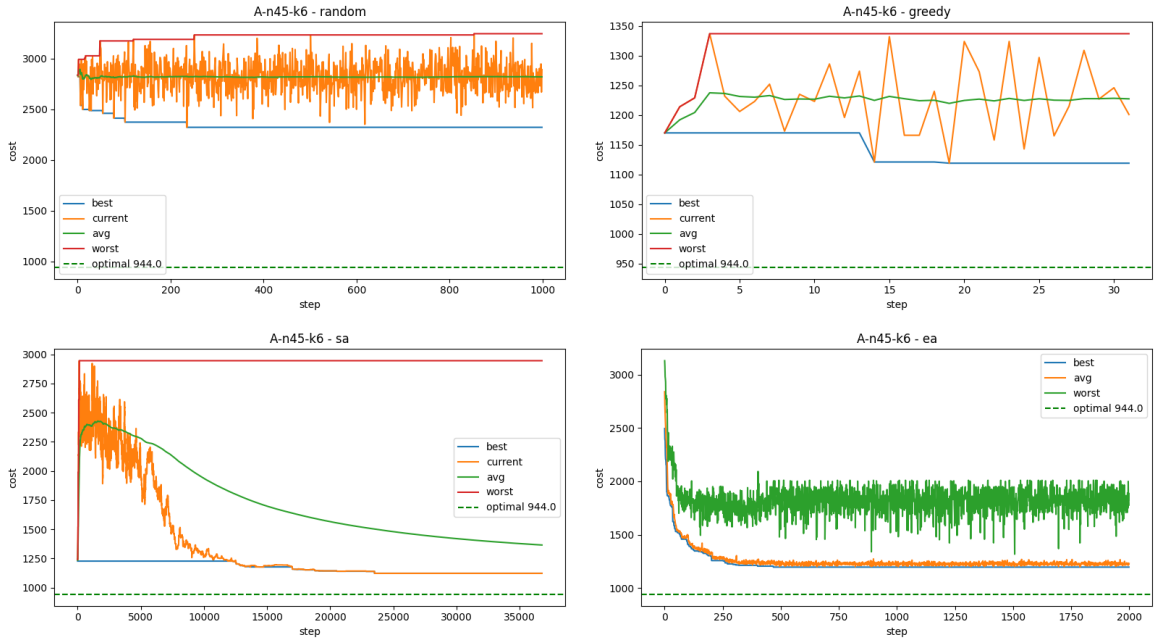
Rysunek 2: A-n32-k5: przebiegi pojedynczych algorytmów (best/current/avg/worst).



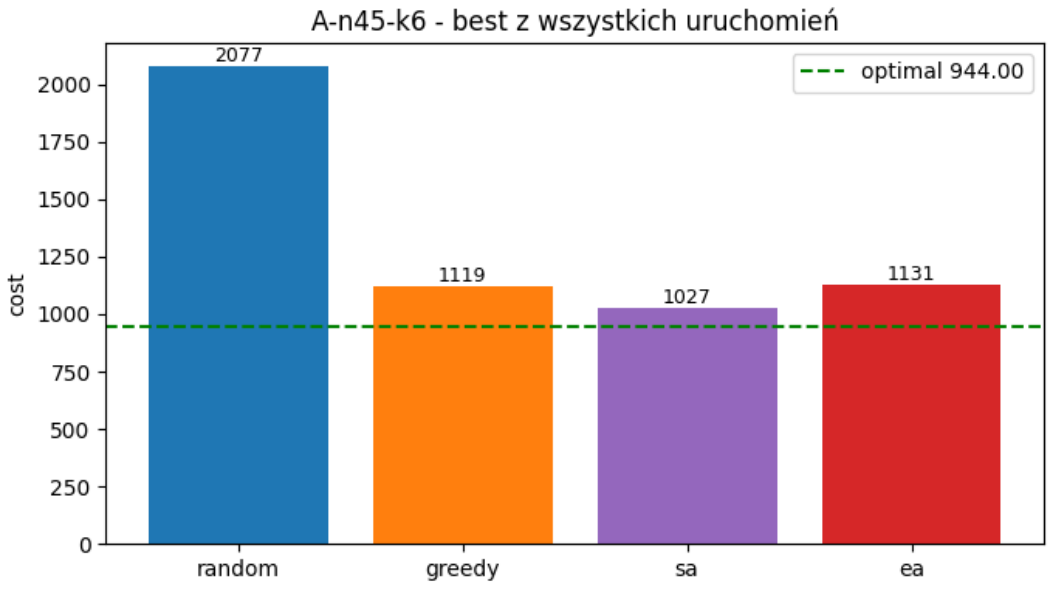
Rysunek 3: A-n32-k5: najlepsze wyniki ze wszystkich uruchomień (losowy/greedy/SA/EA) z linią optimum.



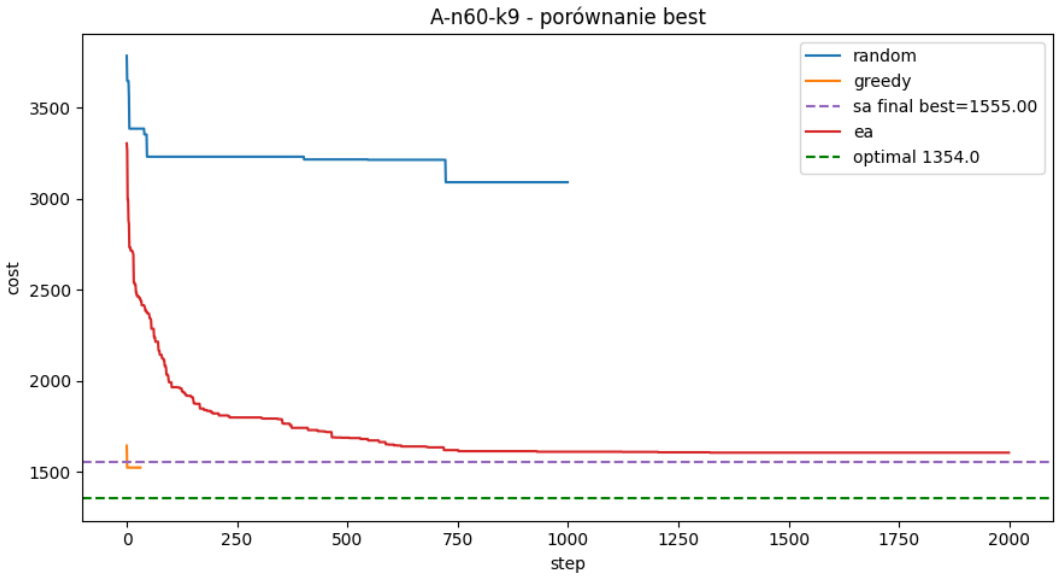
Rysunek 4: A-n45-k6: porównanie przebiegów najlepszych wyników.



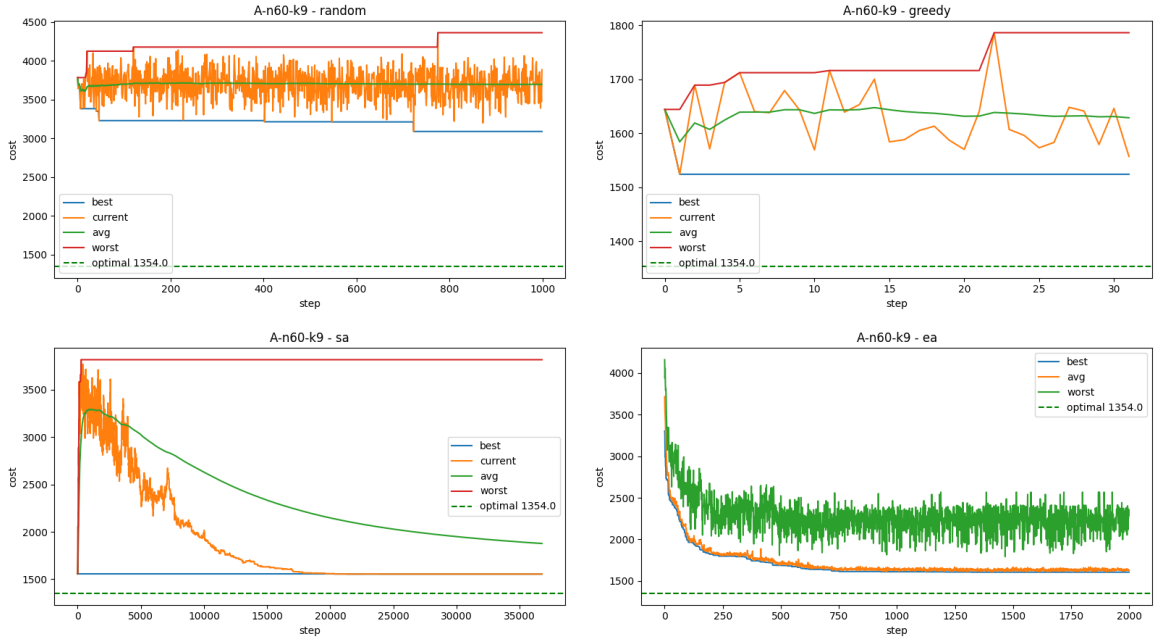
Rysunek 5: A-n45-k6: przebiegi pojedynczych algorytmów.



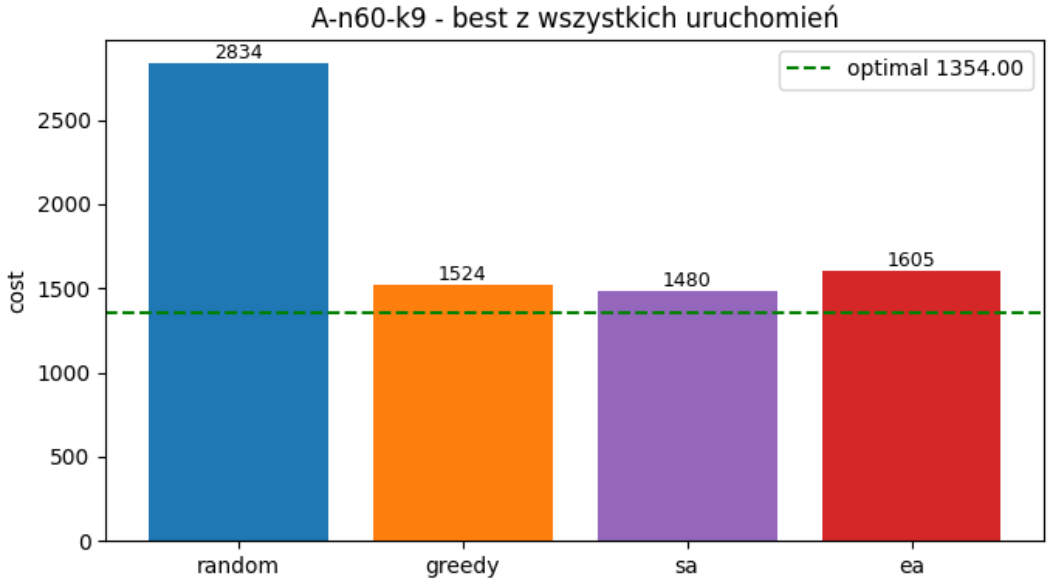
Rysunek 6: A-n45-k6: najlepsze wyniki ze wszystkich uruchomień z linią optimum.



Rysunek 7: A-n60-k9: porównanie przebiegów najlepszych wyników.



Rysunek 8: A-n60-k9: przebiegi pojedynczych algorytmów.



Rysunek 9: A-n60-k9: najlepsze wyniki 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 tuningu

TO DO (do uzupełnienia po dodaniu operatorów i uruchomieniu `config_tuning.ini`).