Sprawozdanie z laboratorium:
Uczenie maszynowe

Część I: Algorytmy optymalizacji lokalnej, problem QAP

14 stycznia 2021

Prowadzący: dr hab. inż. Maciej Komosiński

Autorzy: **Jarosław Warmbier** inf132148 ISWD jaroslaw.warmbier@student.put.poznan.pl

Zajęcia środowe, 13:30.

Oświadczam/y, że niniejsze sprawozdanie zostało przygotowane wyłącznie przez powyższych autora/ów, a wszystkie elementy pochodzące z innych źródeł zostały odpowiednio zaznaczone i są cytowane w bibliografii.

1 Opis problemu

Problem przypisania kwadratowego (ang. Quadratic Assignment Problem – QAP) został wprowadzony przez Koopmana i Beckmana w 1957 roku [3] jako model matematyczny przypisujący działalności ekonomiczne od odpowiednich miejsc, który uwzględniał odległości i przepływy pomiędzy obiektami oraz koszt umiejscowienia obiektu w określonej lokacji.

W niniejszej implementacji rezygnujemy z macierzy kosztów umiejscowienia danego obiektu w określonej miejscowości, stąd mamy dwie macierze [1]:

$A = (a_{ij})$, gdzie a_{ij} jest przepływem z fabryki i do fabryki k;

$B = (b_{jl})$, gdzie b_{jl} jest odległością z miejsca j do miejsca l;

$$\min_{\phi \in S_n} \left(\sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\phi(i)\phi(k)} \right) \tag{1}$$

Poszukując rozwiązania przypisania kwadratowego rozwiązujemy równanie (1), gdzie S_n jest zestawem wszystkich możliwych permutacji liczb $1, 2, \ldots, n$. Każdy z osobna iloczyn $a_{ik} b_{\phi(i)\phi(k)}$ jest kosztem transportu spowodowany przypisaniem obiektu i do lokalizacji $\phi(i)$ i obiektu k do lokalizacji $\phi(k)$. Niniejszy problem jest problem silnie NP-trudynm.

Współcześnie problem QAP znajduje zastosowanie w różnych technologiach m.in. transport, scheduling, elektronika (ang. *wiring problem*), obliczenia rozproszone, genetyka, chemia.

Przykładem występowania problemu definiowanego jako QAP jest problem okablowania tablic rozdzielczych. Mamy określoną liczbę modułów, które są połączone w pary przewodami. Celem jest znalezienie ułożenie modułów tak w tablicy rozdzielczej żeby długość przewodów była jak najmniejsza [2].

Po wstępnej analizie wybrano cztery zbiory o porównywalnej wielkości $n = 25, 26, 27$, natomiast charakteryzują się całkowicie odmiennym krajobrazem optymalizacji. Zbiór chr25a ma w tablicy z przepływami tylko 48 na 625 elementów niezerowych oraz skoki wartości w tablicy z odległościami nie są wielkie z nielicznymi wartościami znacznie niższymi od średniej. Zbiór danych nug25 ma całą wypełnioną tabelę przepływów wartościami od 0 do 8, a kolejna tablica ma 226 elementów zerowych pozostałe mają wartości różne od zera. Zbiór nug27 posiada tablice wartościami wypełniony od 0 do 10 z łączną liczbą zer równą 290. Natomiast zbiór bur26a posiada łącznie 138 wartości zerowych w jednej tablicy, a w drugiej nie ma żadnych.

2 Operator sąsiedztwa

Wykorzystanym operatorem sąsiedztwa jest operator 2-OPT, w którym sąsiedztwo jest generowane na podstawie zamiany dwóch elementów, zamiana kolejności podciągów powodowałaby dużą zmianę w funkcji celu (1).

3 Porównanie działania algorytmów

3.1 Krótki opis algorytmów

Algorytm Random za każdym razem wybiera losowy wektor poprzez wypełnienie wektora rozwiązania liczbami naturalnymi od 1 do n, gdzie n oznacza wielkość problemu, następnie iteruje kolejno po wszystkich kolejnych elementach wektora zmieniając element wektora wynikowego i z losowo wybranym elementem z zakresu $i + 1$ do n.

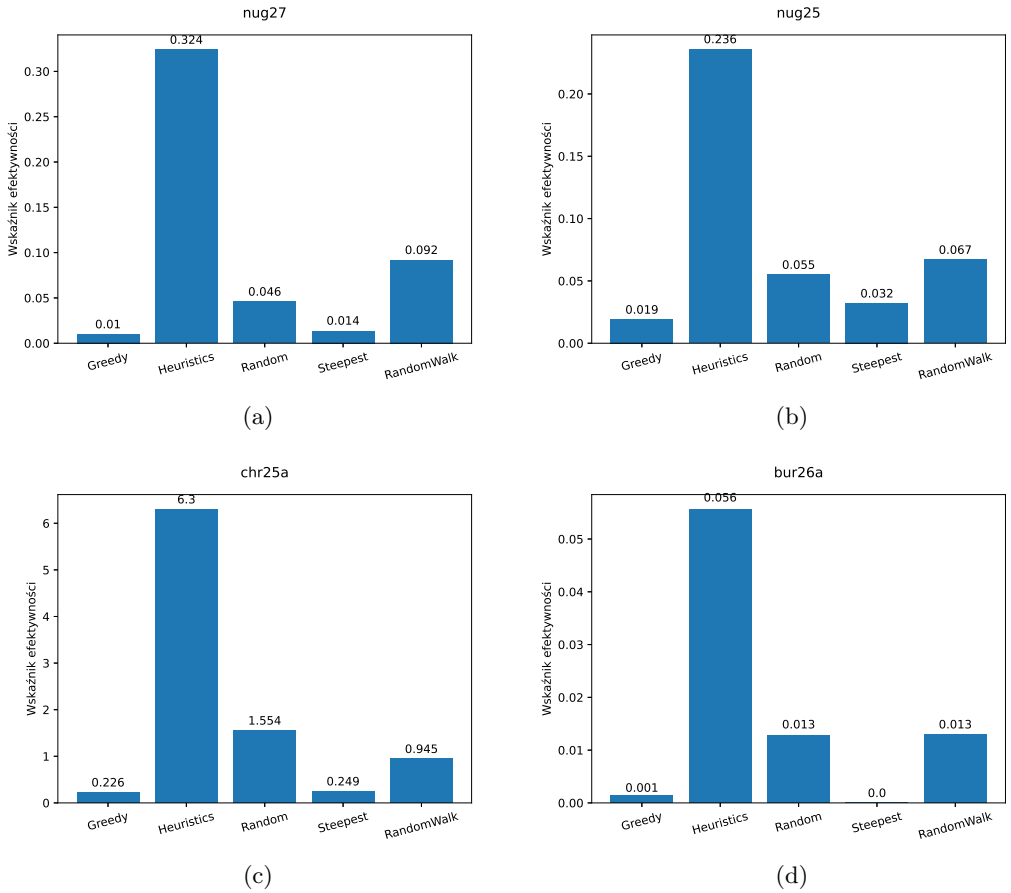
Natomiast algorytm RandomWalk wybiera losowy wektor, w którym w każdym kolejnym kroku zmienia dwa losowo wybrane elementy. Algorytmy losowe nie sprawdzają, czy już wcześniej sprawdzały dane rozwiązanie.

Algorytm Heuristics wypełnia rozwiązanie zerami, a następnie kolejno zaczyna zapełniać wektor, odnajdując największą wartość w macierzy przepływów i łączy je z minimalną wartością w macierzy odległości aż do zapełnienia wektora rozwiązania.

3.2 Jakość działania

Wszystkie algorytmy uruchomiono na 10 sekund, w tym czasie każdy algorytm przeszukiwał rozwiązania problemu. Wykorzystano miarę, w której znalezione najlepsze rozwiązanie jest punktem referencyjnym (2), s jest uzsykanym rozwiązaniem algorytmu, a s_{opt} jest rozwiązaniem najlepszym jakie znaleziono dla danego zbioru.

$$R = \frac{s - s_{opt}}{s_{opt}} \tag{2}$$

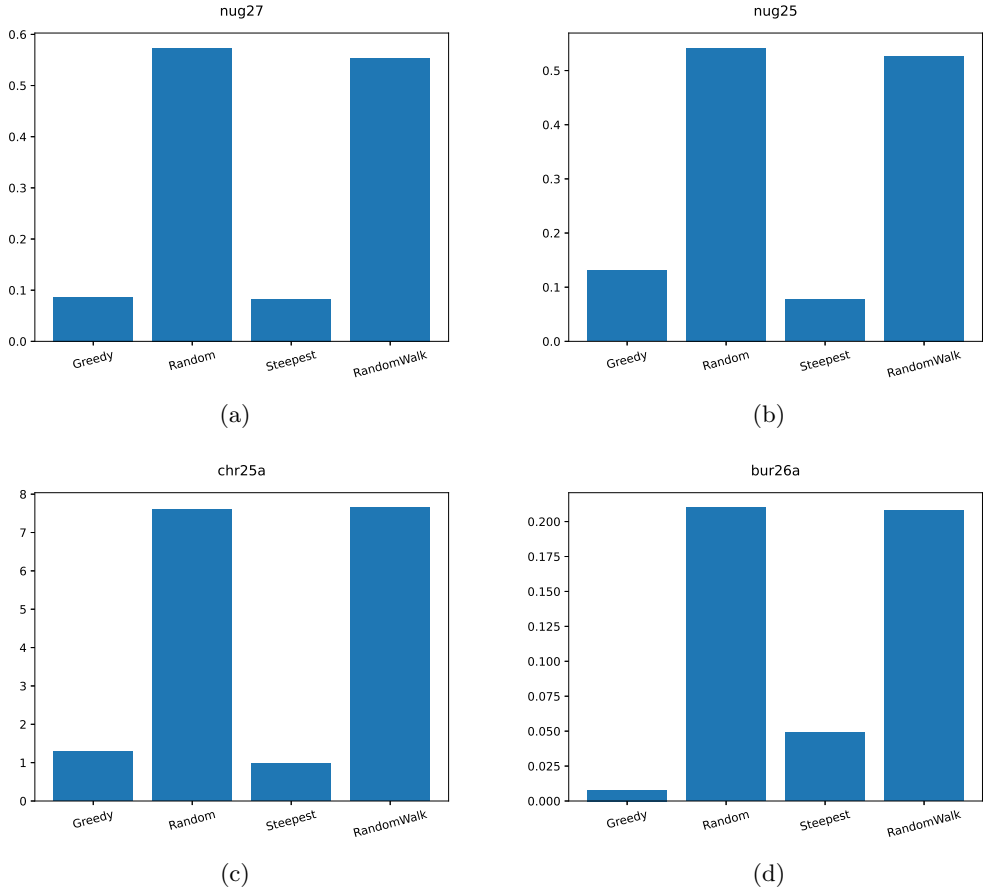


Rysunek 1: Minimalne wartości uzyskane przez algorytmy

Analizując najlepsze wyniki dla zbiorów na Rys. 1 wynika, że algorytmy, które charakteryzują się dużą losowością (Random, RandomWalk) w przeszukiwaniu zbioru rozwiązań

charakteryzują się znacznie gorszymi rezultatami. Wyniki dla obu algorytmów są niejednoznaczne lepsze wyniki uzyskał algorytm Random dla zbiorów nug27, chr25, a algorytm RandomWalk uzyskał znacznie lepsze rezultaty uzyskał dla zbiorów bur26a, nug27. Zbiory nug25 i chr25a charakteryzują się dużą liczbą zer w tablicach, stąd prawdopodobnie lepiej jest generować nowe rozwiązanie od zera, aniżeli przeszukiwać kolejnych sąsiadów wylosowanego zbioru początkowego.

Natomiast w przypadku algorytmów przeszukiwania lokalnego, algorytm Greedy uzyskał lepsze rozwiązania dla zbiorów nug27, chr25a, bur26a.

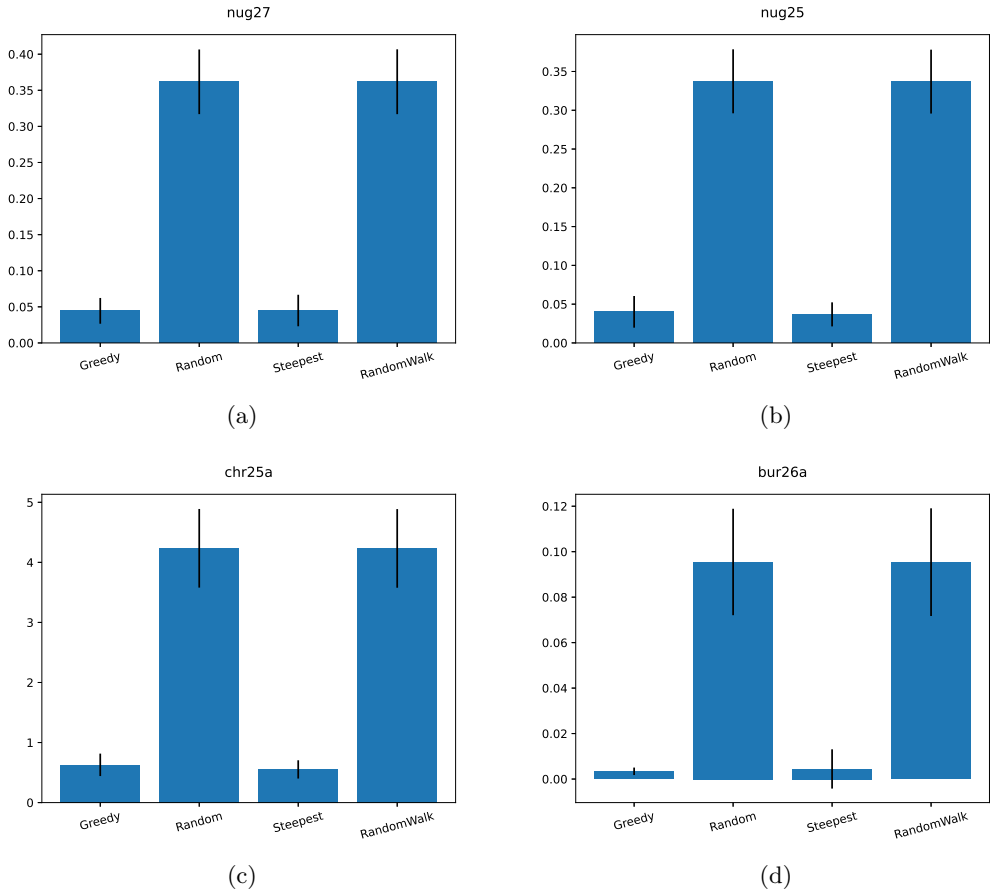


Rysunek 2: Maksymalne wartości uzyskane przez algorytmy

Na Rys. 2 znajdują się najgorsze rezultaty, które znaleziono w czasie przeszukiwania zbioru. Algorytmy przeszukiwania lokalnego uzyskały kilkakrotnie lepsze rezultaty.

Porównując średnie wyniki na Rys. 3 algorytmy przeszukiwania losowego uzyskują podobne rezultaty dla podanych zbiorów z porównywalnym odchyleniem standardowym.

Algorytmy przeszukiwania lokalnego charakteryzują się znacznie lepszymi średnimi wynikami przeszukiwania zbiorów, mniejszy zakres przeszukiwanych rozwiązań jest dla zbiorów nug26 i chr25a występuje przy algorytmie Steepest. Odchylenie standardowe dla algorytmów Steepest i Greedy jest mniejsze ze względu na to, że w danych wynikowych zostały uwzględnione tylko osiągnięte minima lokalne.

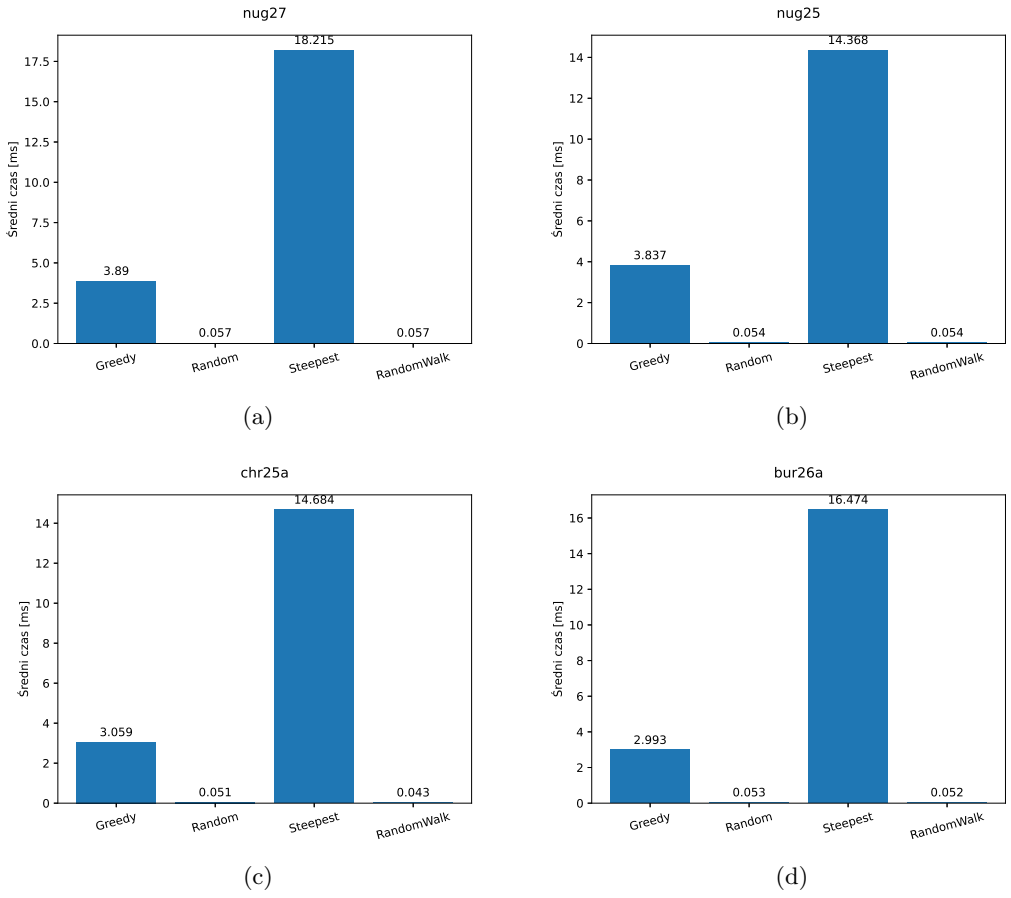


Rysunek 3: Średnie wartości uzyskane przez algorytmy

3.3 Czas działania

Średnie czasy działania algorytmów zaprezentowane na Rys. 4 przedstawiają, że czas generowania kolejnych rozwiązań dla algorytmów Random i RandomWalk są porównywalne i osiągają wartości kilkadziesiąt-krotnie niższe wartości od przeszukiwania lokalnego.

Średni czas działania algorytmu Steepest jest kilkakrotnie dłuższy, w algorytmie Steepest jest przeszukiwane całe sąsiedztwo w poszukiwaniu najlepszego sąsiada, natomiast w przypadku algorytmu Greedy kolejnym sąsiadem jest pierwszy sąsiad, który jest lepszy od poprzedniego. Stąd, wynikają takie różnice w czasach przy przeszukaniu kolejnych rozwiązań.



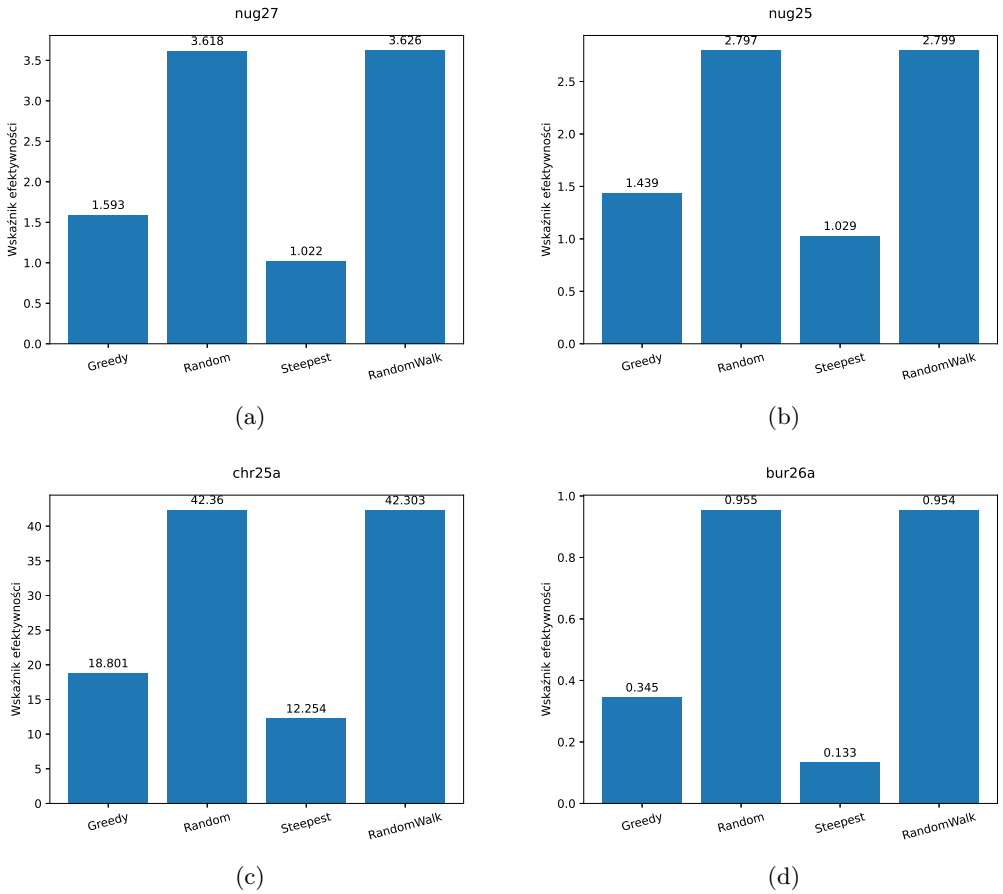
Rysunek 4: Średnie wartości uzyskane przez algorytmy

3.4 Efektywność algorytmów

W celu obliczenia efektywności w czasie dla każdego rezultatu uzyskanego podczas pracy algorytmu obliczyliśmy jego relatywny wskaźnik przy pomocy wzoru (2), następnie obliczyliśmy pole pod wykresem w czasie działania 10 sekund programu przy pomocy wykorzystaniu metody trapezowej.

Wykorzystano tę miarę, ponieważ w łatwy sposób jesteśmy w stanie rozróżnić, które algorytmy w całym procesie działania. Im mniejsza wartość tego wskaźnika tym lepsze wyniku

uzyskuje algorytm w całym czasie działania. Uwzględniana jest jakość wszystkich rozwiązań, a nie pojedynczego najlepszego. Bardzo dobre wyniki w tym przypadku są ważne przy zastosowaniach online.



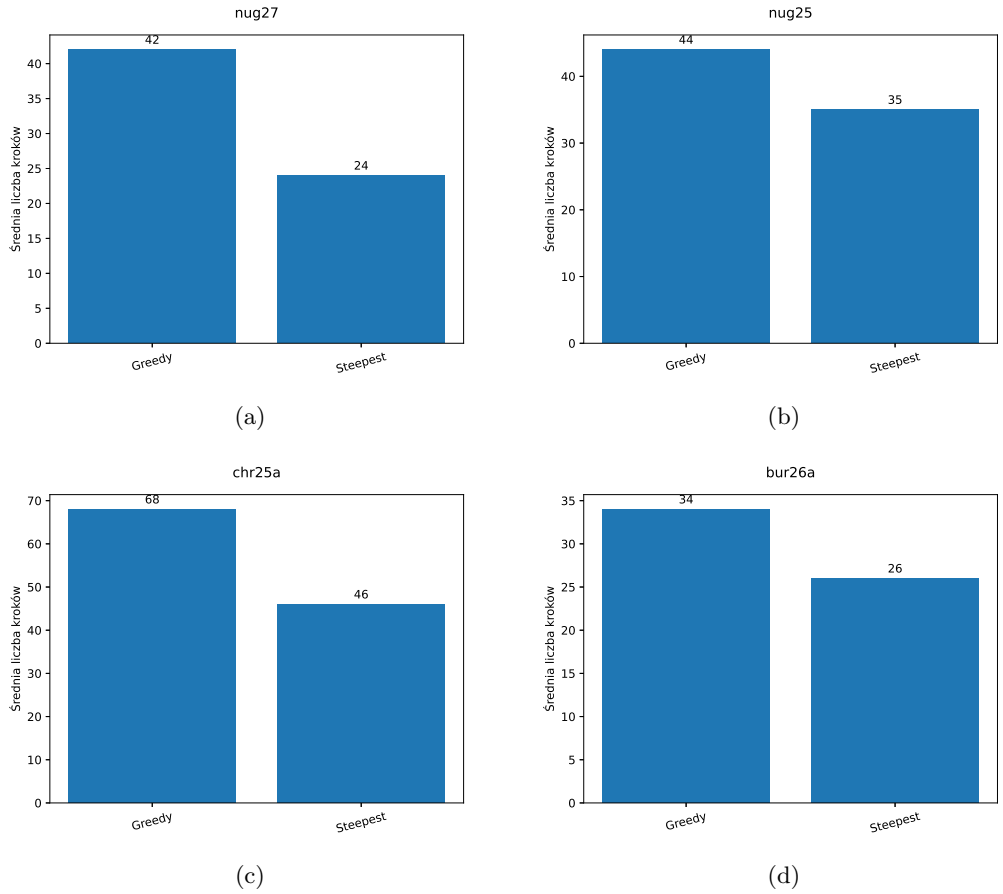
Rysunek 5: Wydajność algorytmów w czasie

Z wyników przedstawionych na Rys. 5 dostrzegamy, że najlepsze wyniki uzyskuje algorytm Steepest, ponieważ w każdym kolejnym kroku wybiera najlepszego sąsiada. Natomiast, algorytm Greedy nie przegląda całego sąsiedztwa, a wybiera pierwszego sąsiada, który osiąga wartość lepszą od bieżącego.

Algorytmy losowe uzyskują podobne parametry.

3.5 Średnia liczba kroków

Rysunek 6 określa średnią liczbę kroków wykonanych w danych zbiorach danych, uruchomiono algorytmy na czas 10 sekund. Jeden krok jest uważany jako osiągnięcie maksimum lokalnego. Większą liczbę kroków uzyskuje algorytm Greedy ze zwzględu na sposób przeszukiwania sąsiadów.

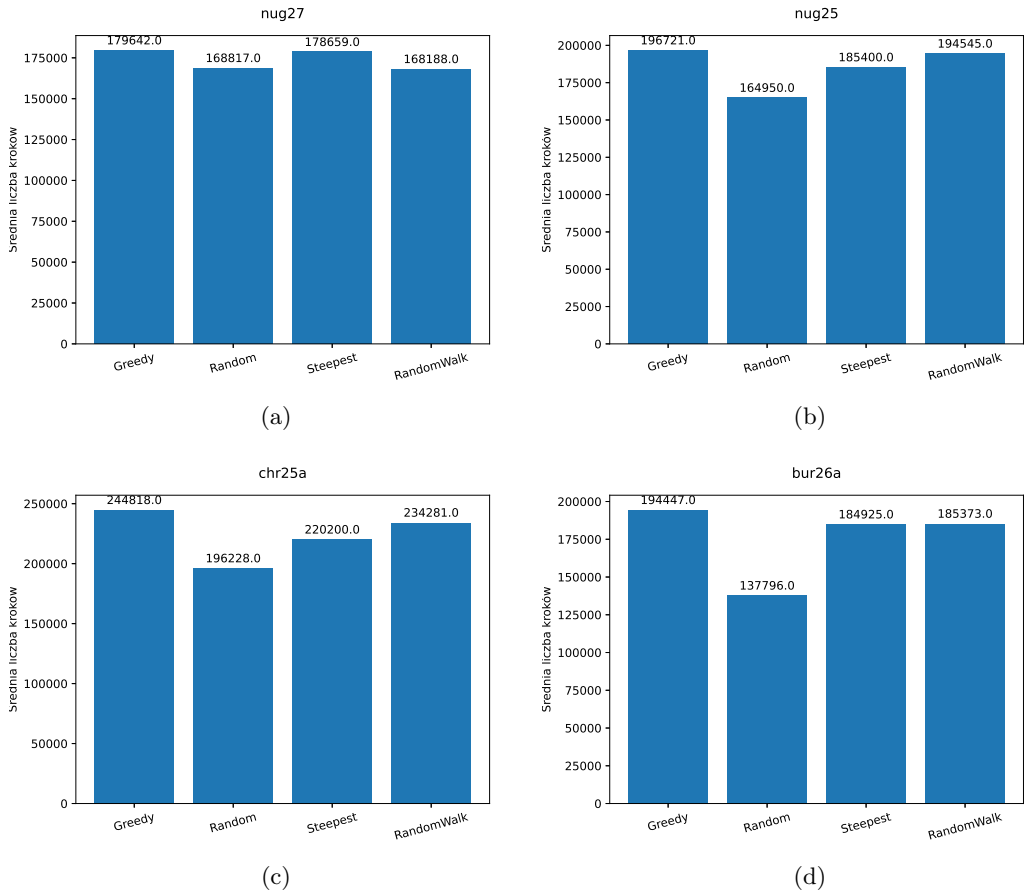


Rysunek 6: Wydajność algorytmów w czasie

Algorytm Greedy idzie zawsze w kierunku pierwzsego przeszukanego wzrost stąd jego liczba kroków jest większa.

3.6 Liczba przejrzanych rozwiązań

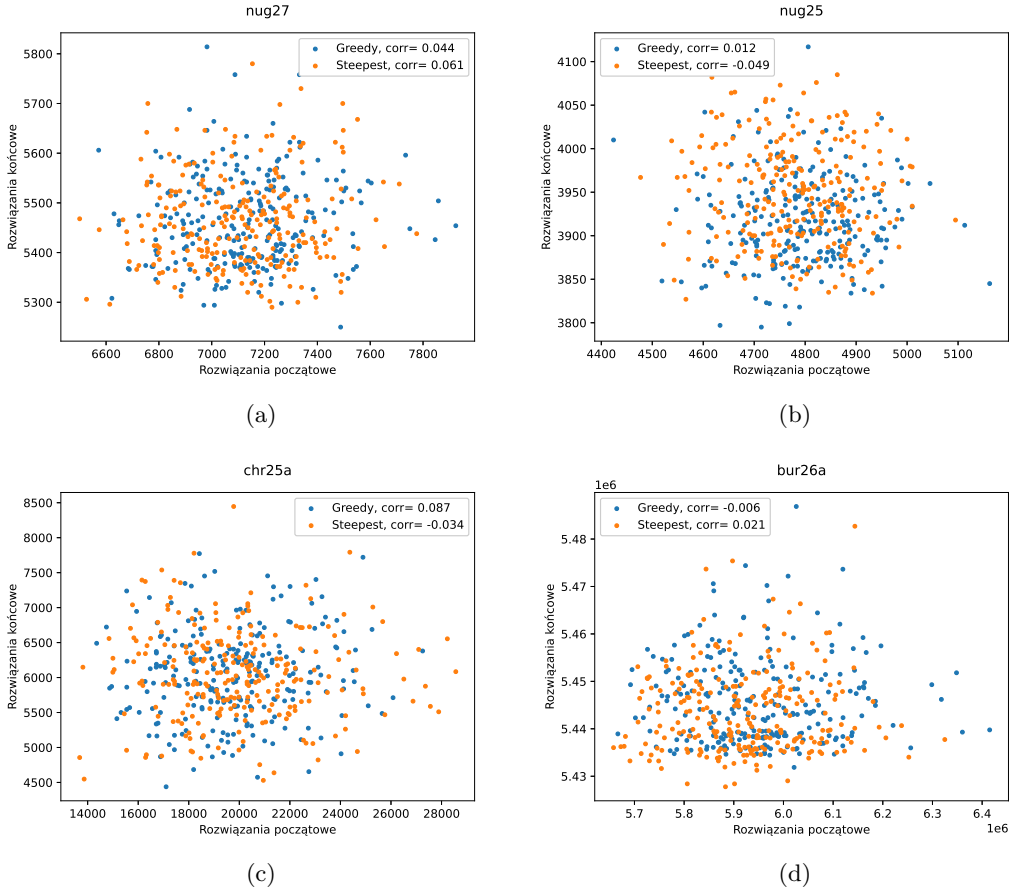
Wykresy na rysunku 7 dostrzegamy, że najmniejszą liczbę rozwiązań odwiedził algorytm Random, ze względu na jego złożoność, gdzie za każdym razem jest generowane nowe rozwiązanie.



Rysunek 7: Przejrzane rozwiązania algorytmów

4 Rozwiązania początkowe vs. rozwiązania końcowe

Na Rysunku 8 wykreślono wykresy, które informują o rozwiązaniach początkowych względem rozwiązań końcowych. Korelacja jest bliska zeru, zatem rozwiązanie początkowe nie ma wpływu na rozwiązanie końcowe przy algorytmach przeszukiwania lokalnego.

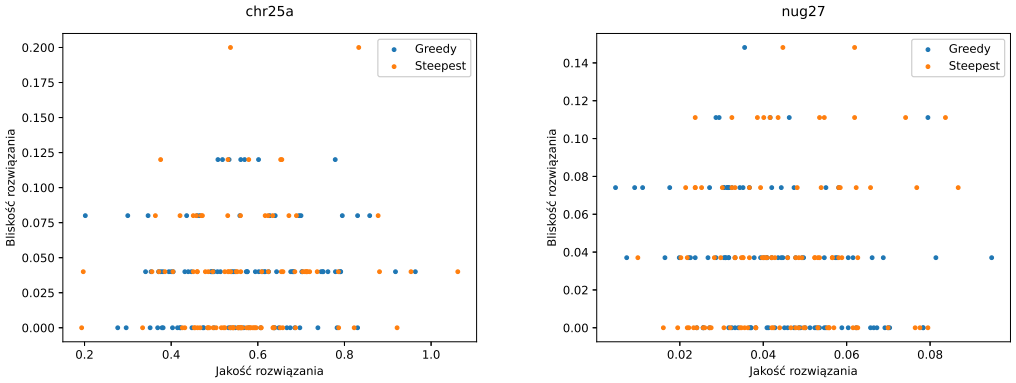


Rysunek 8: Rozwiązania końcowe vs. rozwiązania początkowe

5 Podobieństwo rozwiązań

Miara podobieństwa w odniesieniu do rozwiązania optymalnego to iloraz liczby miejsc permutacji zgodnych z rozwiązaniem optymalnym podzielonym przez długość permutacji tzn. 1 oznacza całkowitą zgodność, a 0 oznacza, że wszystkie liczby w kolejnych miejscach różnią się od rozwiązania optymalnego.

Uruchomiono algorytmy Steepest i Greedy dla zbiorów chr25a i nug27 uzyskując po 100 maksimów lokalnych dla każdego algorytmu.

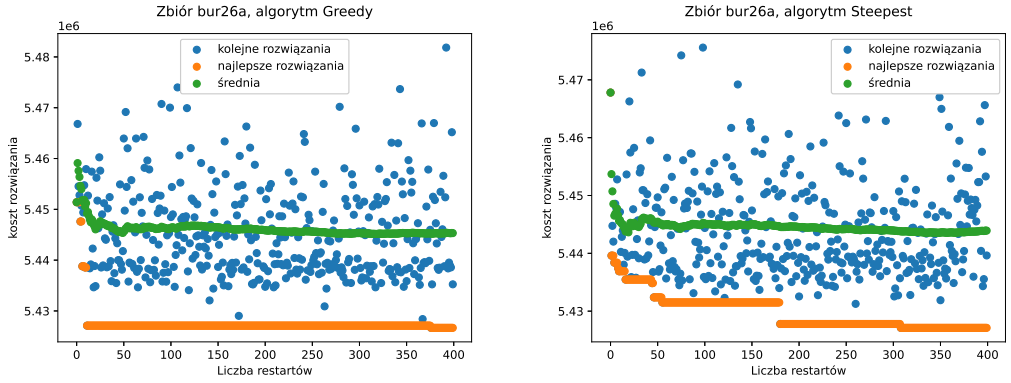


(a) (b)

Rysunek 9: Podobieństwo-jakość rozwiązań

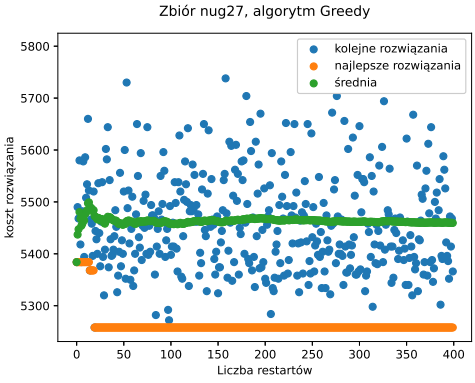
6 Rozwiązania w funkcji liczby restartów

Uruchomiliśmy 400 razy algorytmy Greedy i Steepst na dwóch zbiorach danych bur26a i nug27. Na wykresach Rysunek 10 niebieskimi kropkami oznaczono kolejne rozwiązania, które uzyskano przy restartowaniu algorytmów. Pomarańczowymi kropkami oznaczono najlepsze rozwiązanie od zera do n, gdzie n oznacza numer restartu. Zielonymi kropkami oznaczono średnie rozwiązanie dla zakresu od 0 n, gdzie n ponownie oznacza numer kolejnego restartu.

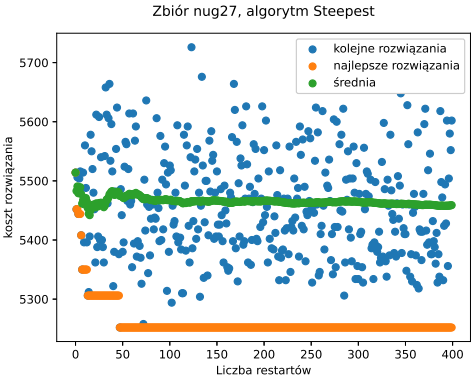


(a)

(b)



(c)



(d)

Rysunek 10: Rozwiązania algorytmu w funkcji liczby restartów

7 Wnioski

Algorytmy przeszukiwania lokalnego Greedy i Steepest osiągają najlepsze wyniki. Prosta heurystyka, która została zaprojektowana osiąga najgorsze rezultaty. Nie ma zdecydowanego zwycięzcy we wszystkich zbiorach danych. Algorytmowi Steepest udało się znaleźć rozwiązani optymalne dla zbioru bur26a. Średnie wyniki Greedy i Steepest są na porównywalnym poziomie.

Czas osiągania minimum lokalnego przez algorytm Steepest i Greedy różnią się kilkakrotnie, przy czym Greedy uzyskuje mniejszą wartość, ponieważ nie przeszukuje za każdym razem całego sąsiedztwa.

Efektywność w czasie algorytmów Rys. 5 wskazuje, że najlepsze wyniki uzyskuje algorytm Steepest, ale w tym samym czasie generuje mniejszą liczbę rozwiązań niż Greedy. Algorytm losowe uzyskują podobną efektywność.

W liczbie przejrzanych rozwiązań dostrzegamy, że najwolniejszy algorytmem jest Random. Miara nie uwzględniła, czy dane rozwiązanie zostało już sprawdzone przez dany algorytm. Algorytm Steepest przegląda nieznacznie mniejszą liczbę rozwiązań.

Pomiędzy rozwiązaniami początkowymi, a końcowymi minimami lokalnymi nie dostrzega się żadnej korelacji, stąd, rozwiązanie początkowe nie wpływa na końcowy wynik.

Rozwiązania, które są podobne do optymalnego uzyskują gorsze wyniki – Rysunek 9.

Dla zbioru nug27 czas po którym można byłby wyłączyć dalsze działanie algorytmów to około 50 przeszukanych rozwiązań lokalnych. W przypadku zbioru bur26a algorytm Greedy osiągnął bardzo dobry na samym początku i ten wynik został poprawiony dopiero po 350 zrestartowaniu algorytmu. Natomiast Steepest poprawiał swój wynik aż do około 320 restartu algorytmu.

Literatura

[1] Dell'Amico M. i Martello S. Burkard R. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Society for Industrial and Applied Mathematics, 2009.

[2] Kadłuczak P. Kwiecień J. i Filipowicz B. Chmiel W. A comparison of nature inspired algorithms for the quadratic assignment problem. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 65(4):513–522, 2017.

[3] Beckmann M. Koopmans T.C. Assignment problems and the location of economic activities. *Econometrica*, pages 53–76, 1957.