Sprawozdanie z laboratorium:
Uczenie maszynowe

Część I: Algorytmy optymalizacji lokalnej, problem QAP

26 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-trudnym.

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 osiem zbiorów o różnych wielkości $n = 12, 25, 26, 27, 50$, które charakteryzują się odmiennym krajobrazem optymalizacji. W pierwszej tablicy najmniejszego zbioru danych chr12a istnieją 22 elementy niezerowe, która jest macierzą rzadką. Ze względu na najmniejszą przestrzeń rozwiązań, istnieje największe prawdopodobieństwo znalezienia optymalnego rozwiązania w porównaniu do pozostałych zbiorów.

Kolejnymi wykorzystanymi zbiorami o podobnej wielkości są: chr25a, nug25, bur26a, nug27. 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.

Problemy, które charakteryzują się największą przestrzenią rozwiązań to: tai50b, lipa 50a, lipa50b. Pomimo identycznej wielkości liczby możliwych rozwiązań charakteryzują się one różnymi krajobrazami optymalizacji. Pierwsza tablica zbioru lipa50a składa się z liczb: 0, 1, 2; których liczność wynosi kolejno: 91, 1173, 49. W zbiorze lipa50b obydwie tablice mają podobny rozkład i zmienność wartości. Natomiast zbiór tai50b zawiera, w której pierwsze tablica jest symetryczna, a druga niesymetryczna.

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. Kolejno odnajduje największą wartość w macierzy przepływów i łączy ją z minimalną wartością w macierzy odległości aż do zapełnienia wektora rozwiązania.

3.2 Jakość działania

Algorytmy Steepest i Greedy uruchomiono na 10 przebiegów algorytmów, natomiast czas działania algorytmów RandomWalk i Random jest równy czasowi jednemu uruchomieniu algorytmu Steepest dla danego zbioru (Tab. 1.

Tabela 1: Czasy trwania jednego uruchomienia algorytmu Steepest

Zbiór danych	bur26a	tai50b	nug27	chr25a	lipa50a	chr12a	nug25	lipa50b
Czas [s]	4,621	84,081	6,012	2,161	47,742	0,186	2,419	48,448

Wykorzystano miarę, w której znalezione najlepsze rozwiązanie dla zbioru jest punktem referencyjnym (2), s jest uzyskanym rozwiązaniem algorytmu, a s_{opt} jest rozwiązaniem najlepszym jakie znaleziono dla danego zbioru. Dodatkowo rezultat mnożymy przez 100.

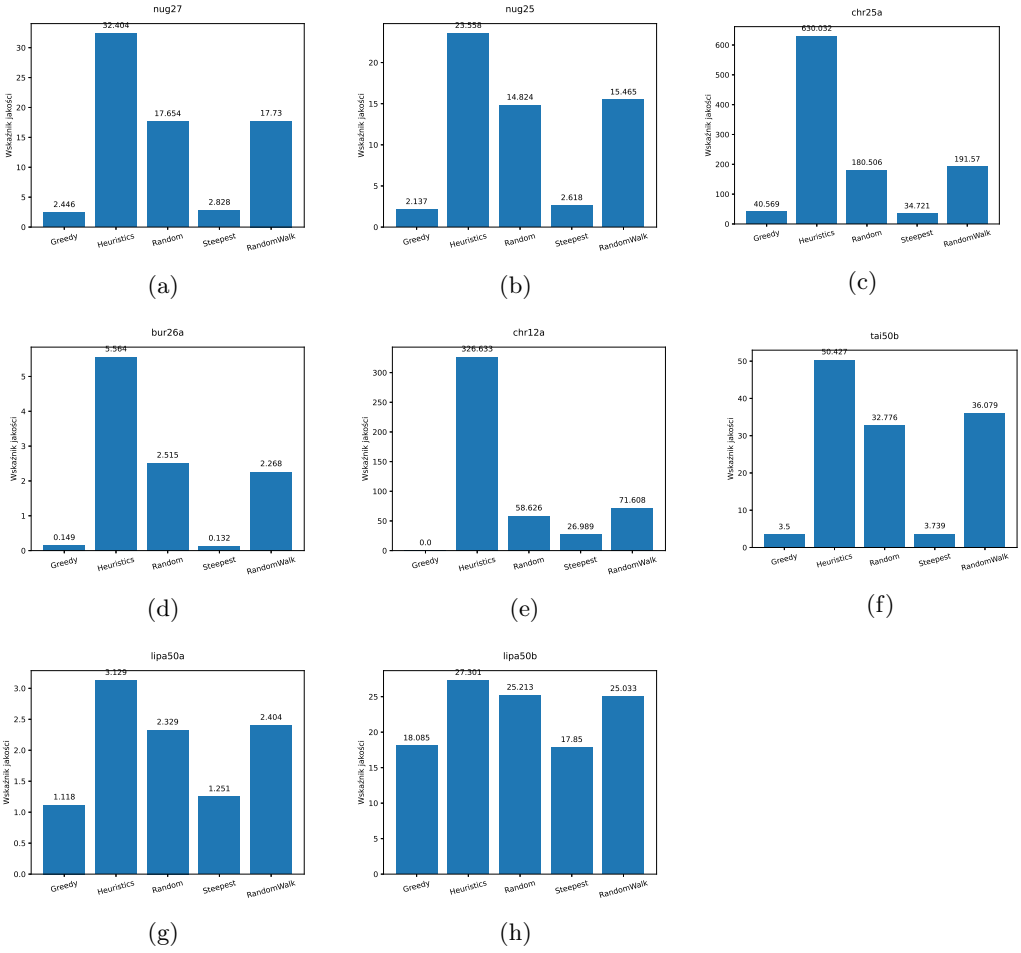
$$R = \frac{s - s_{opt}}{s_{opt}} \cdot 100 \tag{2}$$

Najlepsze wyniki uzyskał algorytm Steepest w zbiorach danych: chr25a, bur26a, lipa50b; w pozostałych najlepsze wyniki uzyskał algorytm Greedy. Greedy w przypadku zbioru chr12a znalazł rozwiązanie optymalne. Algorytmy losowe charakteryzują się we wszystkich zbiorach podobną jakością. Najgorsze rezultaty uzyskał algorytm Heuristics, którego opis znajduje się wyżej (3.1, który przy zbiorze lipa50b osiąga przybliżone rezultaty do algorytmów losowych – różnica jest najmniejsza w porównaniu do pozostałych zbiorów.

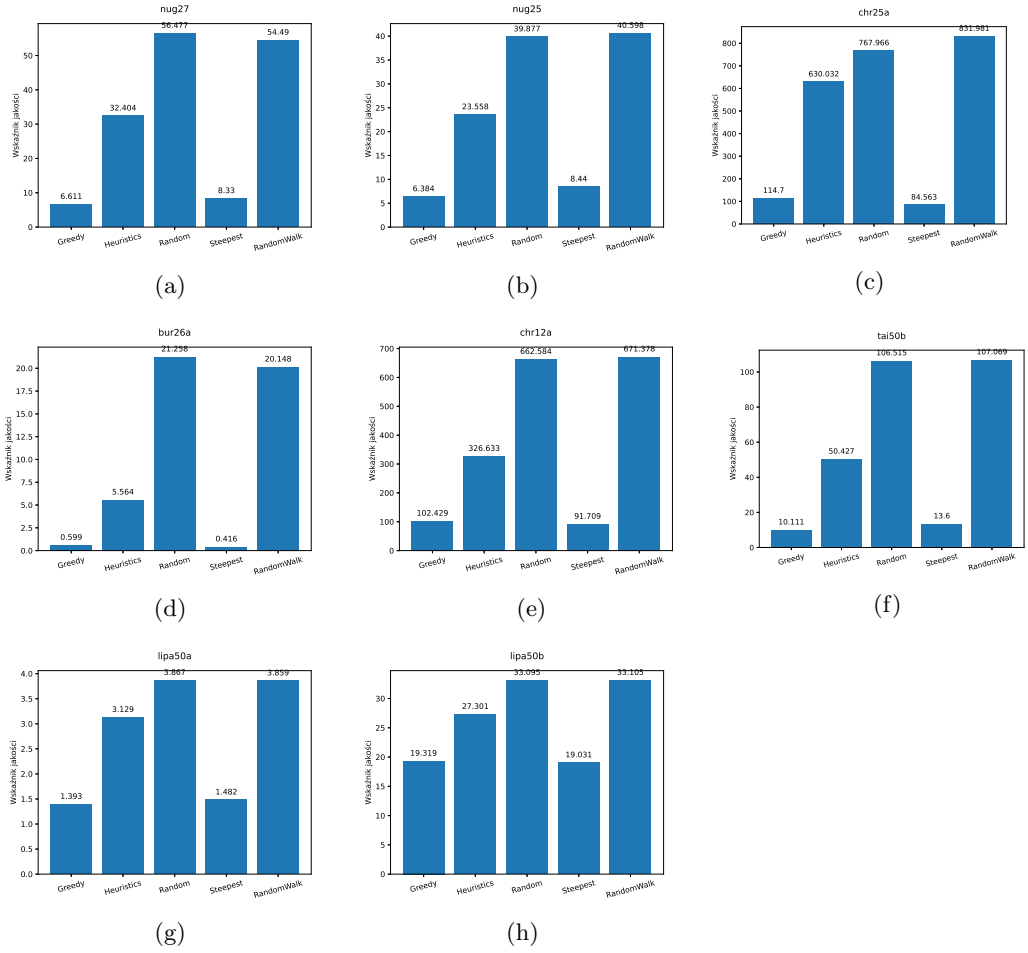
Na Rys. 2 znajdują się najgorsze rezultaty, które znaleziono w czasie przeszukiwania zbioru rozwiązań. Algorytmy przeszukiwania lokalnego charakteryzują się dużo mniejszym zakresem przeglądania gorszych jakościowo rozwiązań ze względu na ciągłe szukanie najlepszego rozwiązania w swoim sąsiedztwie. Metody przeszukiwania losowego RandomWalk i Random osiągają najgorsze maksymalne rozwiązania. Metoda Heurystyczna dla każdego zbioru znajduje lepsze rozwiązanie niż najgorsze znalezione przez algorytmy losowe.

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

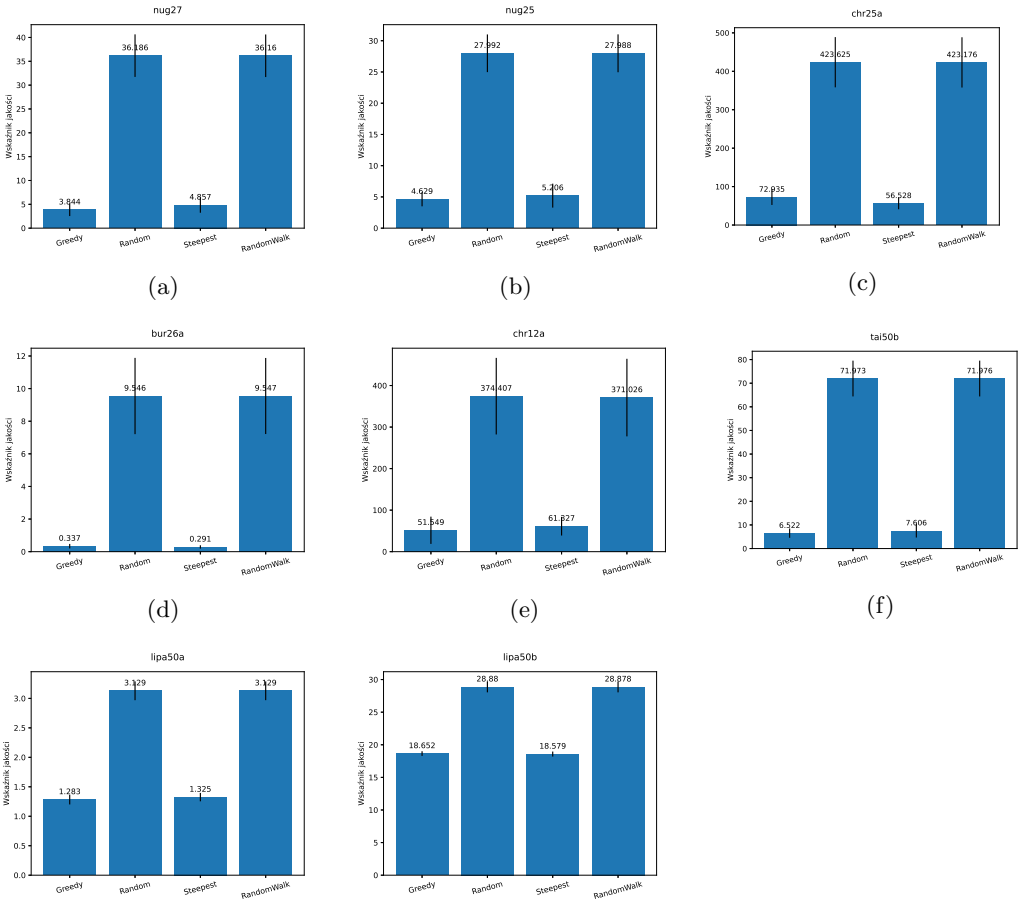
Algorytmy przeszukiwania lokalnego charakteryzują się znacznie lepszymi średnimi wynikami przeszukiwania zbiorów. Posiadają znacznie mniejsze odchylenie standardowe, ponieważ jako rezultat 10 przebiegów algorytmu podane są tylko osiągnięte minima lokalne bez całej historii przejrzanych rozwiązań. Żaden z algorytmów nie osiąga najlepszych średnich dla wszystkich zbiorów.



Rysunek 1: Minimalne wartości uzyskane przez algorytmy



Rysunek 2: Maksymalne wartości uzyskane przez algorytmy



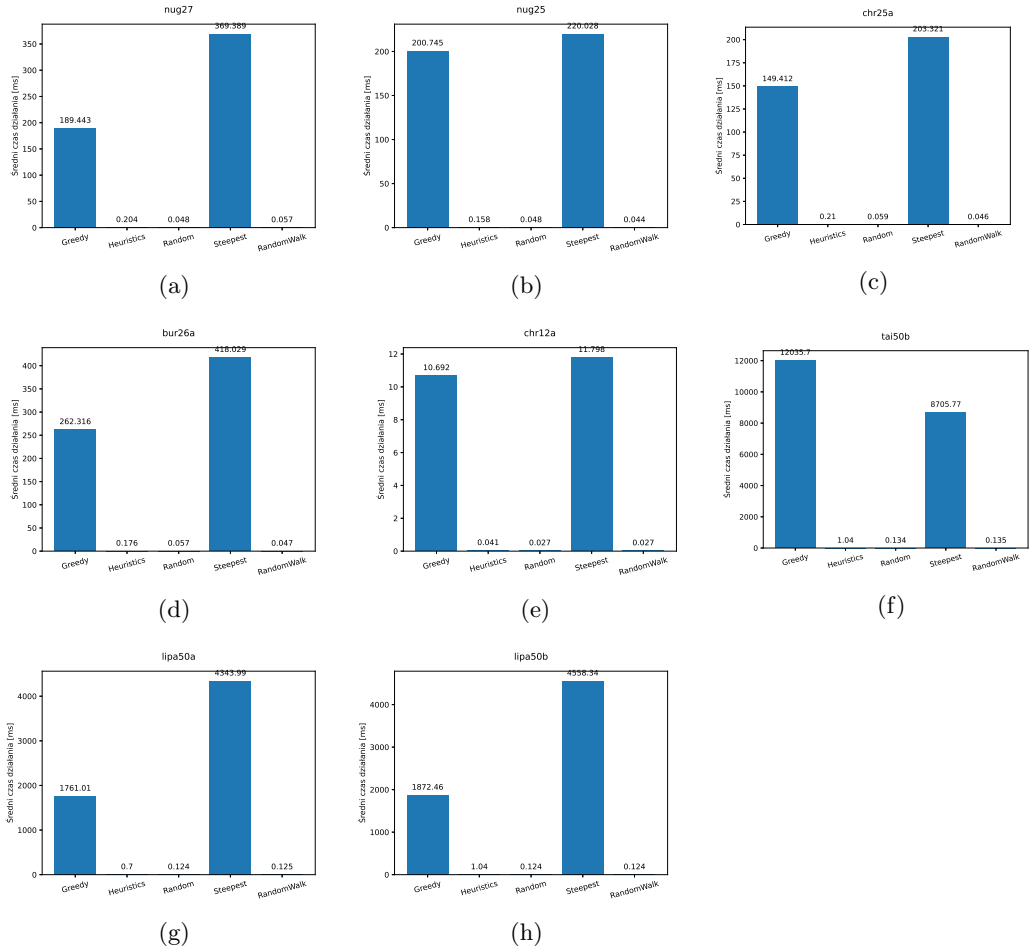
Rysunek 3: Średnie wartości uzyskane przez algorytmy

3.3 Czas działania

Średnie czasy działania algorytmów zaprezentowane na Rys. 4 przedstawiają, że czasy generowania kolejnych rozwiązań dla algorytmów Random i RandomWalk są porównywalne i osiągają wartości kilkadziesiątkrotnie niższe wartości od przeszukiwania lokalnego. Niemniej jednak, algorytm RandomWalk powinien być szybszy. Algorytm Random za każdym razem generuje nową permutację rozwiązania, a algorytm RandomWalk tworzy tylko na początku nowe rozwiązanie, a każde kolejne jest losową zamianą dwóch miejsc. Czas działania algorytmu Heuristics jest większa od czasu działania algorytmów losowych.

Szczególnym przypadkiem jest zbiór tai50b (Rys. 4f dla którego Greedy ma znacznie wyższy średni czas działania i uzyskuje lepsze wyniki (Rys. 3f). Prawdopodobnie jest to uzasadnione krajobrazem optymalizacji, w którym Greedy schodzi szukając minimum lokalnego okrążając je.

Algorytm Steepest ma średnio dłuższy czas działania, jest to uzasadnione przeglądaniem zawsze całego sąsiedztwa.

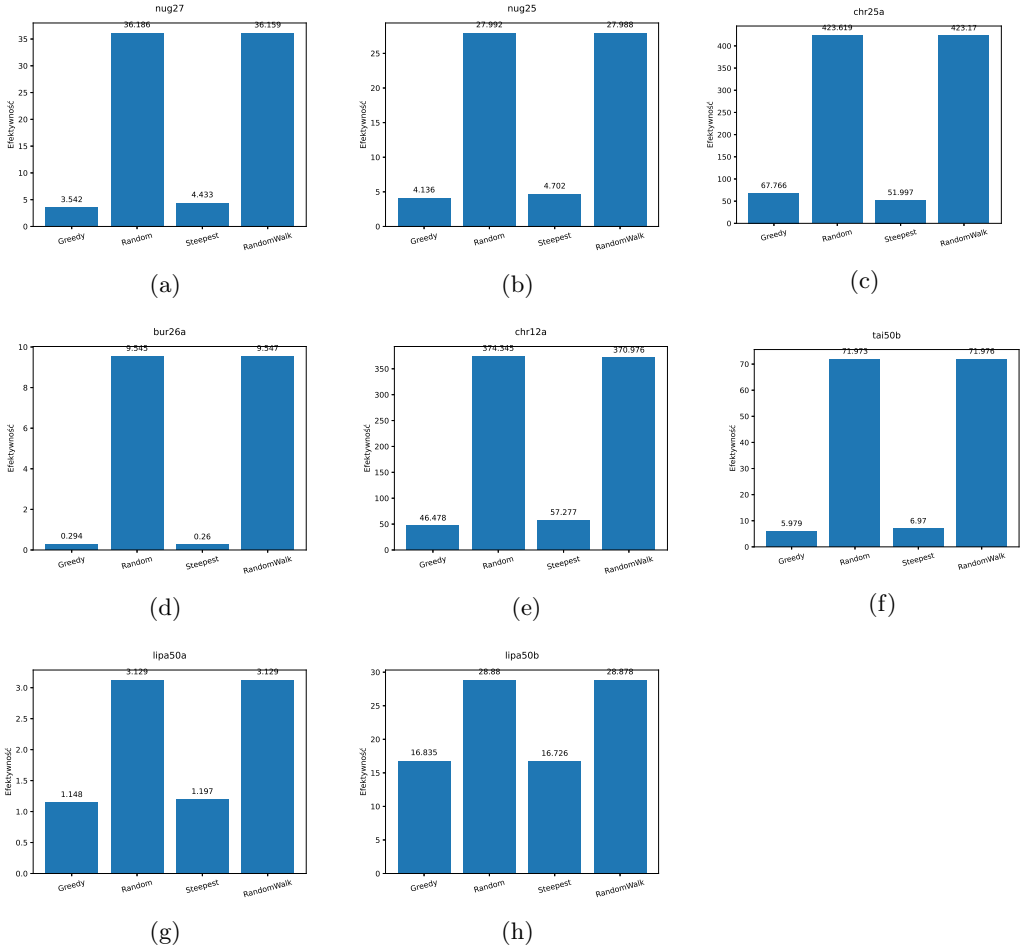


Rysunek 4: Średnie czasy uzyskane przez algorytmy

3.4 Efektywność algorytmów

W celu obliczenia efektywności w czasie dla każdego rezultatu uzyskanego podczas pracy algorytmu wyznaczyliśmy jego relatywny wskaźnik przy pomocy wzoru (2), następnie obliczyliśmy pole pod wykresem w czasie jego działania, a następnie uzyskana wartość pola została podzielona przez czas działania algorytmu.

Im mniejsza wartość tego wskaźnika tym lepsze wyniki uzyskuje algorytm w całym czasie działania. Określa średnią jakość wszystkich wyników w czasie.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Rysunek 5: Wydajność algorytmów w czasie

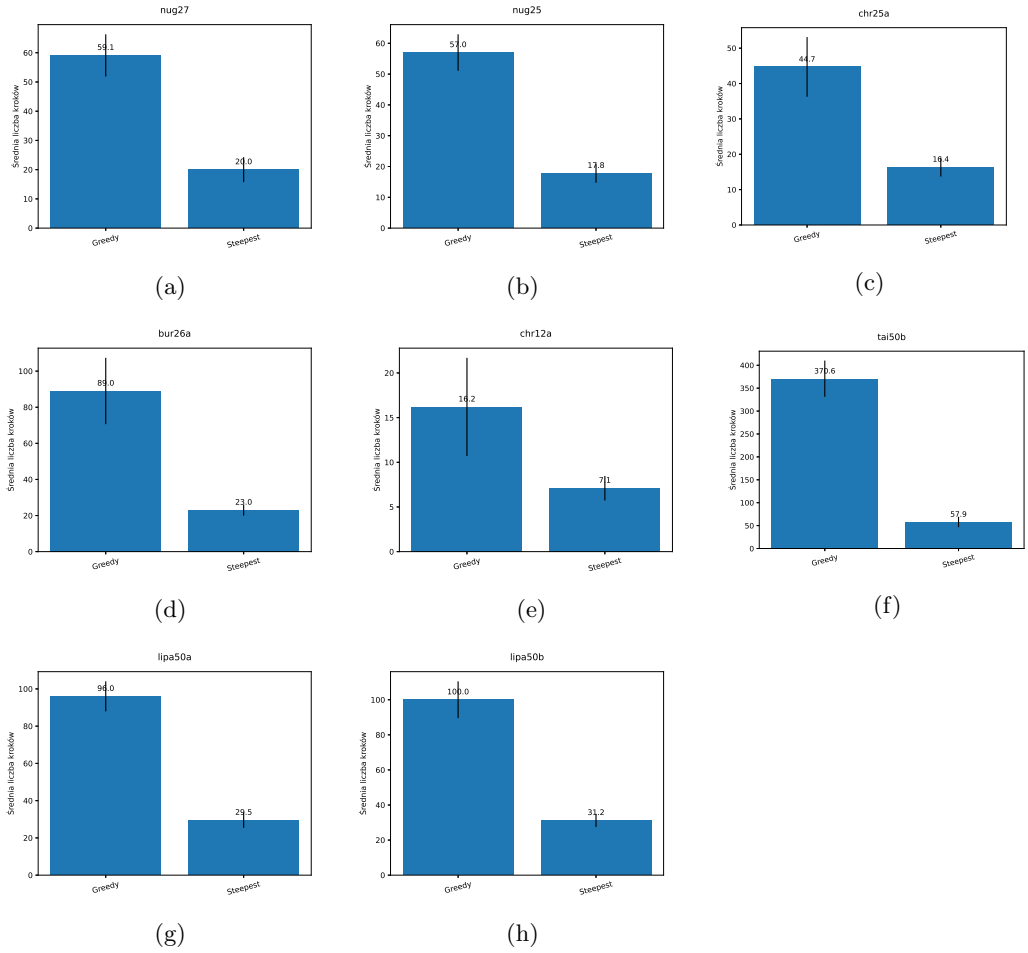
Nie uwzględniono algorytmu Heuristics, pomimo dużo gorszego rezultatu od pozostałych algorytmów (Rys. 1) uzyskałby bardzo wysoką efektywność ze względu na bardzo niski czas działania (Rys. 4).

Algorytmy przeszukiwania losowego uzyskują mają najgorszą efektywność. Greedy lepsze rezultaty od Steepest'a w 5 na 8 zbiorów, gorszą efektywność uzyskuje dla zbioru chr25a, bur26a, lipa50b. Interesujący jest zbiór tai50b, w którym pomimo znacznie dłuższego czasu działania od Steepest'a (Rys. 4f) jego efektywność jest lepsza ze względu na dużo lepszą jakość rozwiązań (Rys. 3f).

3.5 Średnia liczba kroków

Greedy wykonuje zawsze znacznie większą liczbę kroków, co najmniej dwa razy więcej, ponadto Steepest ma w badanych zbiorach zawsze mniejsze odchylenie standardowe.

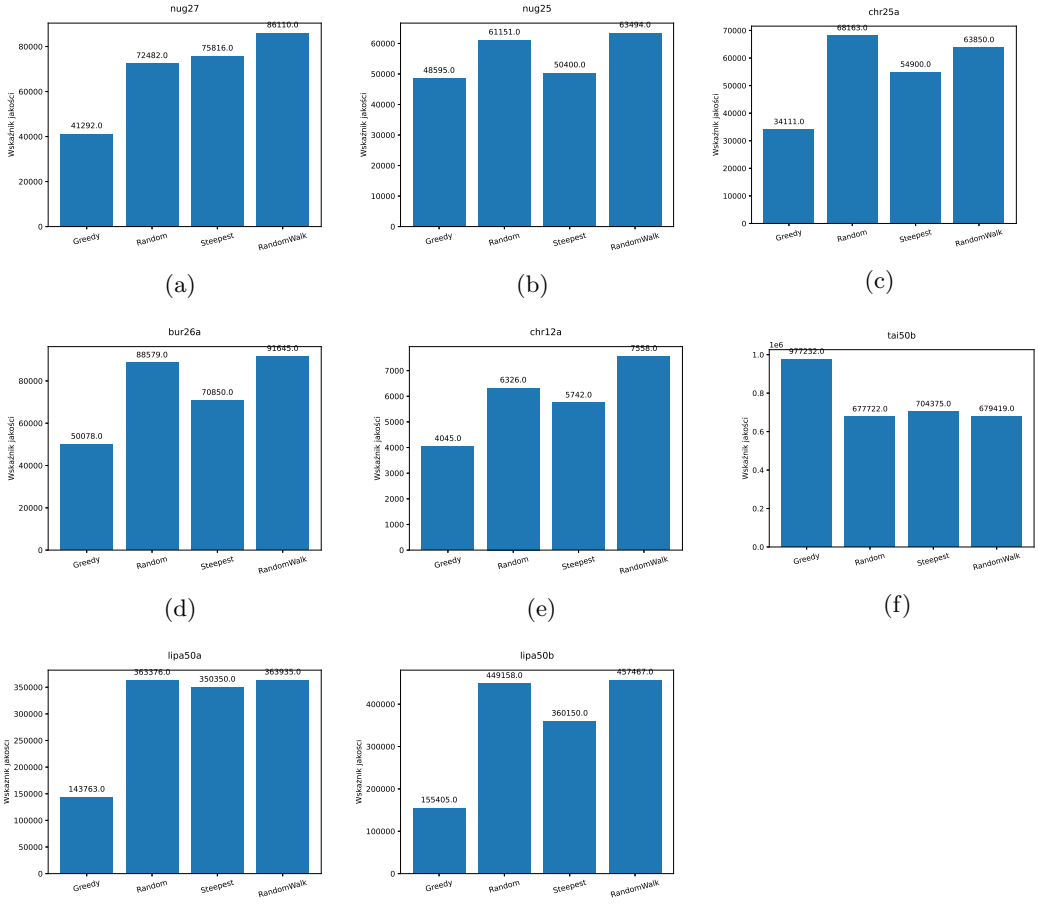
Większa liczba kroków przez Greedy'ego jest uzasadniona wybieraniem pierwszego lepszego sąsiada do następnego kroku, a Steepest przegląda całe sąsiedztwo.



Rysunek 6: Średnia liczba kroków

3.6 Liczba przejrzanych rozwiązań

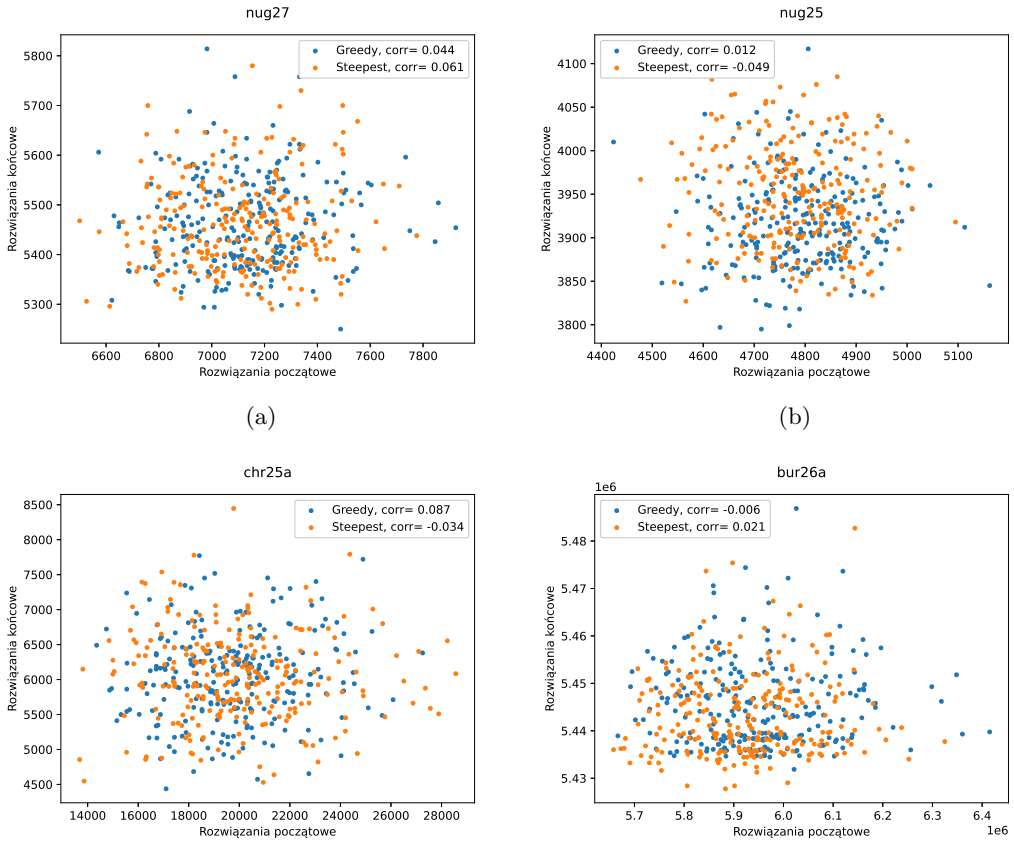
W liczbie sprawdzonych rozwiązań nie zostało uwzględnione czy dane rozwiązanie wystąpiło kolejny raz. Algorytmy przeszukiwania losowego sprawdziły największą liczbę rozwiązań w tym samym czasie co Steepest ze względu na mniejszą złożoność obliczeniową. Greedy odwiedził najmniejszą liczbę rozwiązań ze względu na najkrótszy czas pracy, tylko w przypadku zbioru tai50b Greedy przejrzał więcej, uzasadnione jest to dłuższym czasem pracy.



Rysunek 7: Liczba przejrzanych rozwiązań

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.

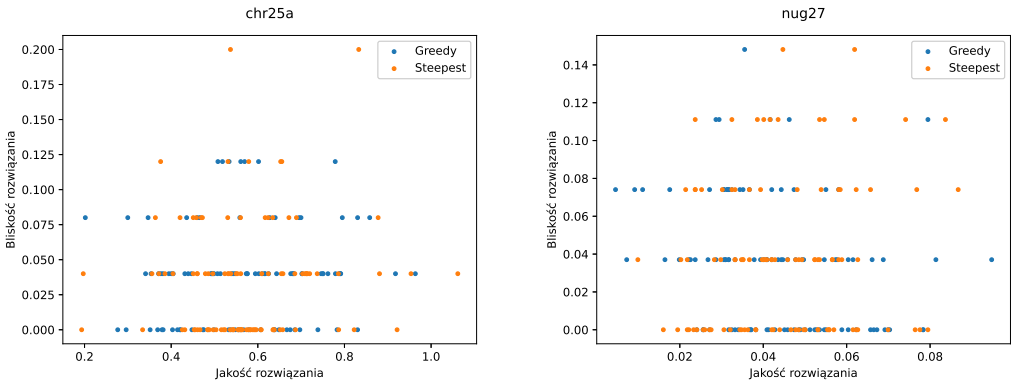


Rysunek 8: Rozwiązania końcowe vs. rozwiązania początkowe

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

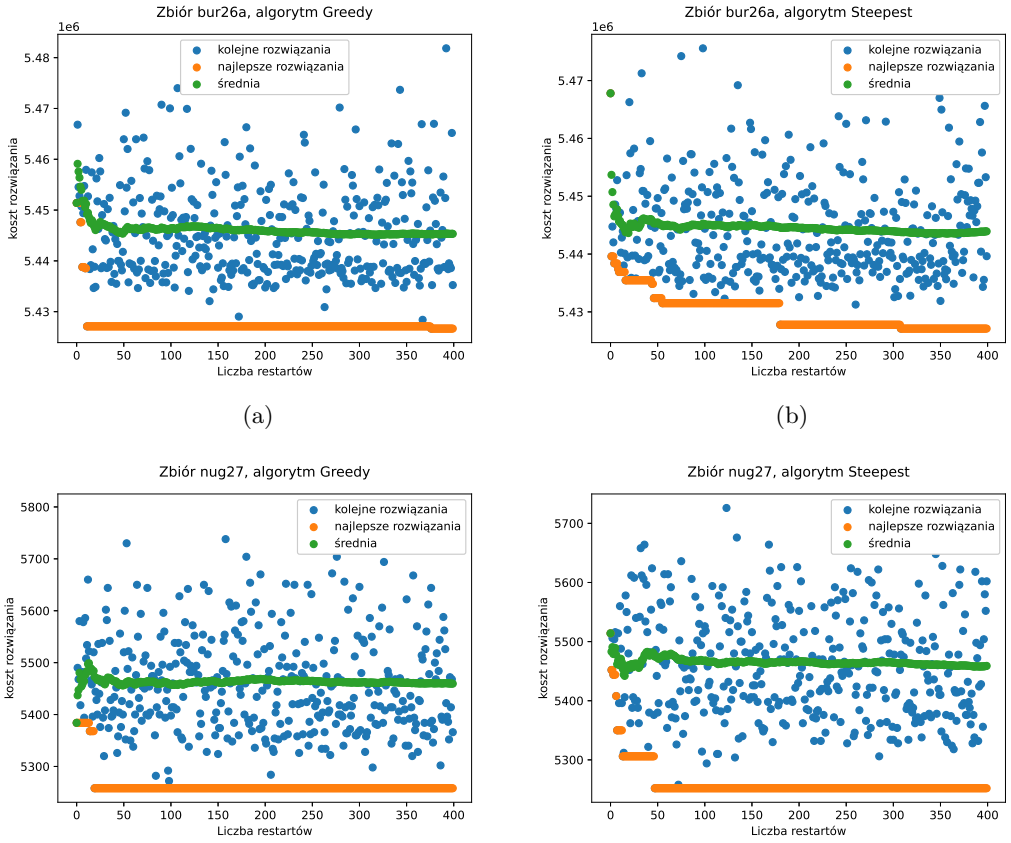
W przypadku zbioru chr25a rozwiązania, które nie są podobne do optymalnego (bliskość rozwiązania równa 0) mieszczą się w zakresie od 0,2 do około 0,95 jakości rozwiązania. Najlepsze rozwiązania osiągają jakość bliską 0,2 i charakteryzują się podobieństwem poniżej 0,1, stąd wynika, że bliskość rozwiązania nie ma znaczenia na jego jakość. Ponadto, dwa rozwiązania, których podobieństwo jest równe 0,2 posiadają niską jakość powyżej 0,5.

Najlepsze rozwiązanie w zbiorze nug27 ma miarę podobieństwa poniżej 0,08 i jego jakość jest równa poniżej 0,01. Rozwiązania z największym podobieństwem posiadają jakość powyżej 0,03.

Znalezione rozwiązania z największym prawdopodobieństwem osiągają dużo niższą jakość, zatem bliskość rozwiązania nie wpływa na poprawę wyniku. Nie znaleziono żadnej zależności pomiędzy wykorzystaniem danego algorytmu, a jakością, bliskością 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 do n, gdzie n ponownie oznacza numer kolejnego restartu.



Rysunek 10: Rozwiązania algorytmu w funkcji liczby restartów

W przypadku zbioru bur26a średnia jakość rozwiązania stabilizuje się w momencie uzyskania około 60 restartu algorytmu. W przypadku Greedy'ego dostrzegamy wykładniczy spadek do około 20 restartu, a w przypadku Steepest'a dostrzegamy osiągnięcie minimum lokalnego z ponownym odbiciem i pogorszeniem się średniej jakości. Najlepsze rozwiązanie w przypadku algorytmu Greedy znaleziono przy około 375 restarcie, ale od około 10 restartu nie było żadnej poprawy jakości. Steepest mniej więcej regularnie znajdował lepsze rozwiązania do około 55 restartu, a kolejne lepsze rozwiązania zostały znalezione po średnio 100 restartach.

W zbiorze nug27 algorytmy przeszukiwania lokalnego Steepst i Greedy, znalazły najlepsze rozwiązania kolejno po około 20 i 45 restartach, kolejne restarty nie poprawiły jakości najlepszego rozwiązania. W przypadku Greedy'ego na początku średnia rozwiązań ulegał

pogorszeniu do momentu około 20 restartu, następnie szybko po około 30 restarcie ustabilizowała się.

Nie ma potrzeby zbyt długiego ciągłego restartowania algorytmów ze względu na to, że kolejne restarty nie wpływają znacząco na poprawę jakości rozwiązania, a koszt pracy algorytmu jest wysoki.

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. Zbyt długie restartowanie algorytmu nie wpływa znacząco na poprawę jakości rozwiązania.

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.