Algorytmy rozwiązujące problem QAP

Michał Kuźma i Michał Biernacki

11 listopada 2016

Opis problemu

QAP (Quadratic assignment problem) reprezentuje następujące zadanie:

Dane są zbiory n lokalizacji i n ośrodków. Każda para lokacji znajduje się w określonej odległości od siebie, a dla każdej pary ośrodków znany jest przepływ. Celem jest takie przypisanie ośrodków do lokalizacji, aby zminimalizować sumę iloczynów odległości i przepływów.

Problem wykorzystywany jest często do zamodelowania zadania rozmieszczenia fabryk (ośrodków) w zestawie znanych lokalizacji. Jako przepływy podane są wówczas interakcje, w jakie fabryki wchodzą wzajemnie (transport surowców, etc.).

Ponieważ problem należy do grupy NP-trudnych, nie jest znany algorytm, który pozwoliłby na znalezienie dokładnego rozwiązania w czasie wielomianowym. W celu osiągnięcia zadowalających wyników czasowych uzyskując dobre rozwiązanie, wykorzystuje się algorytmy heurystyczne i metaheurystyki.

Operator sąsiedztwa

W projekcie korzystano z operatora sąsiedztwa 2-OPT, który dla każdej permutacji zwraca sąsiedztwo złożone ze wszystkich permutacji uzyskanych przez zamianę dwóch pozycji miejscami.

Wykorzystanie tego operatora sprawia, że wielkość sąsiedztwa każdej permutacji wynosi n2 (gdzie n to długość permutacji).

Krótki opis zaimplementowanych algorytmów

Random Search

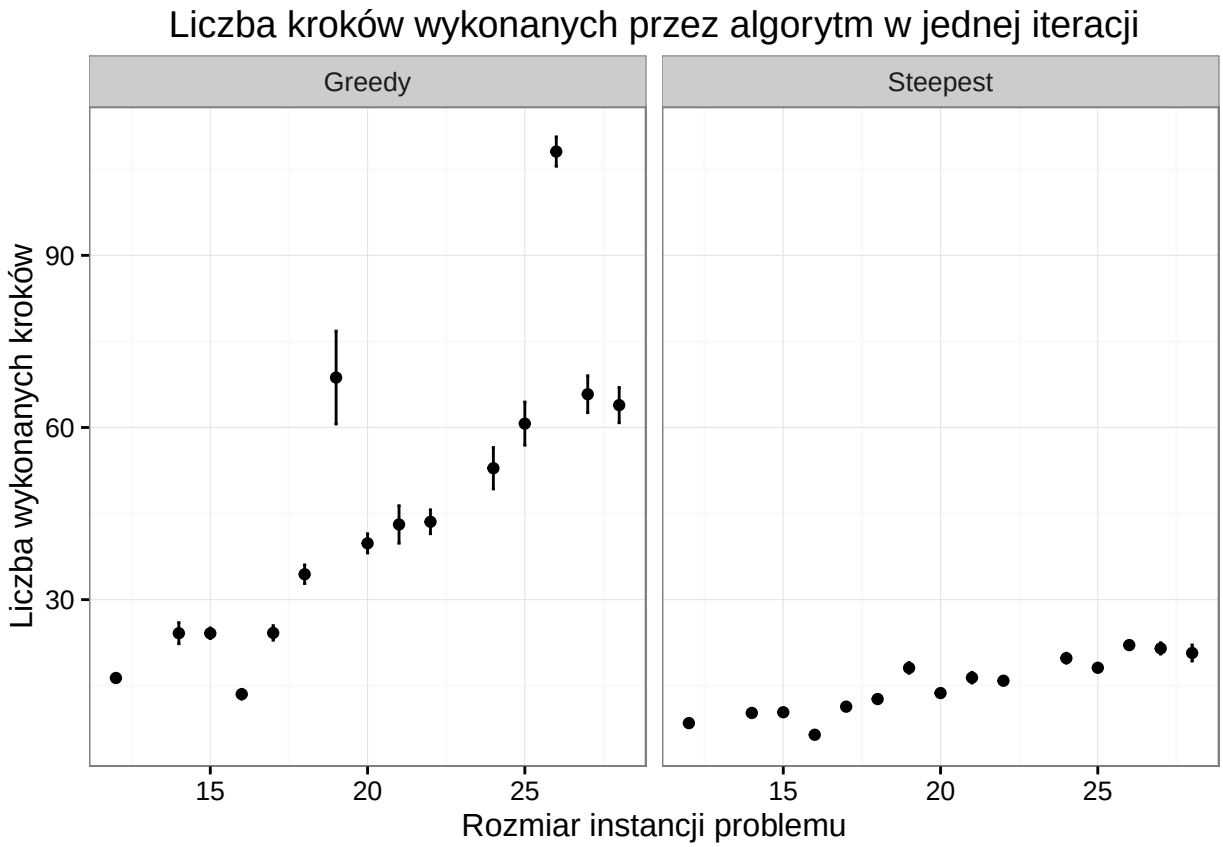
Algorytm przeszukiwania losowego (Random Search) jest najprostszym z wykorzystanych w projekcie. Przez określony czas losuje on rozwiązania i na koniec zwraca najlepsze z nich. Parametr czasowy stanowi jedyne kryterium stopu algorytmu.

Local Search

Algorytm przeszukiwania lokalnego (Local Search) wyszukuje lepsze rozwiązania w zbiorach sąsiedztwa aż do osiągnięcia optimum lokalnego. Nie daje jednak żadnej gwarancji odnalezienia optimum globalnego. Algorytm został zaimplementowany w dwóch wersjach różniących się sposobem wyboru sąsiada, do którego algorytm powinien przejść.

- *Greedy Local Search* wybiera pierwszego sąsiada, który jest lepszy od obecnie rozpatrywanego rozwiązania.
- *Steepest Local Search* przeszukuje całe sąsiedztwo wybierając najlepszego sąsiada i przechodzi do niego, jeśli jest lepszy od obecnego rozwiązania.

Przeprowadzono eksperyment zliczający, ilu sąsiadów oceniają obie wersje algorytmu, oraz ile kroków robią. Wyniki przedstawiono na poniższych wykresach. W celu lepszej czytelności wykresów, wybrano instancje o wielkości mniejszej, niż $n = 30$.



Powyższe wykresy pokazują, że w pełnej iteracji *Steepest* wykonuje znacznie mniej kroków od algorytmu *Greedy*, jednak ponieważ za każdym razem sprawdza wszystkich możliwych sąsiadów, liczba ocenionych przez niego rozwiązań jest większa dla większości instancji problemu. Algorytm *Greedy* wykazuje dużą niestabilność zarówno w liczbie wykonanych kroków, jak i ocenionych sąsiadów (Dla większości instancji odchylenia są większe, niż w *Steepest*).

Heurystyka

Algorytm heurystyczny buduje rozwiązanie w oparciu o predefiniowane reguły, które zgodnie z założeniem mają prowadzić do dobrego wyniku. Zaproponowana w projekcie heurystyka działa analogicznie do sortowania bąbelkowego. Po wylosowaniu permutacji startowej wybierany jest najkorzystniejszy w obecnym krajobrazie element na kolejne pozycje (0, 1, 2, . . .).

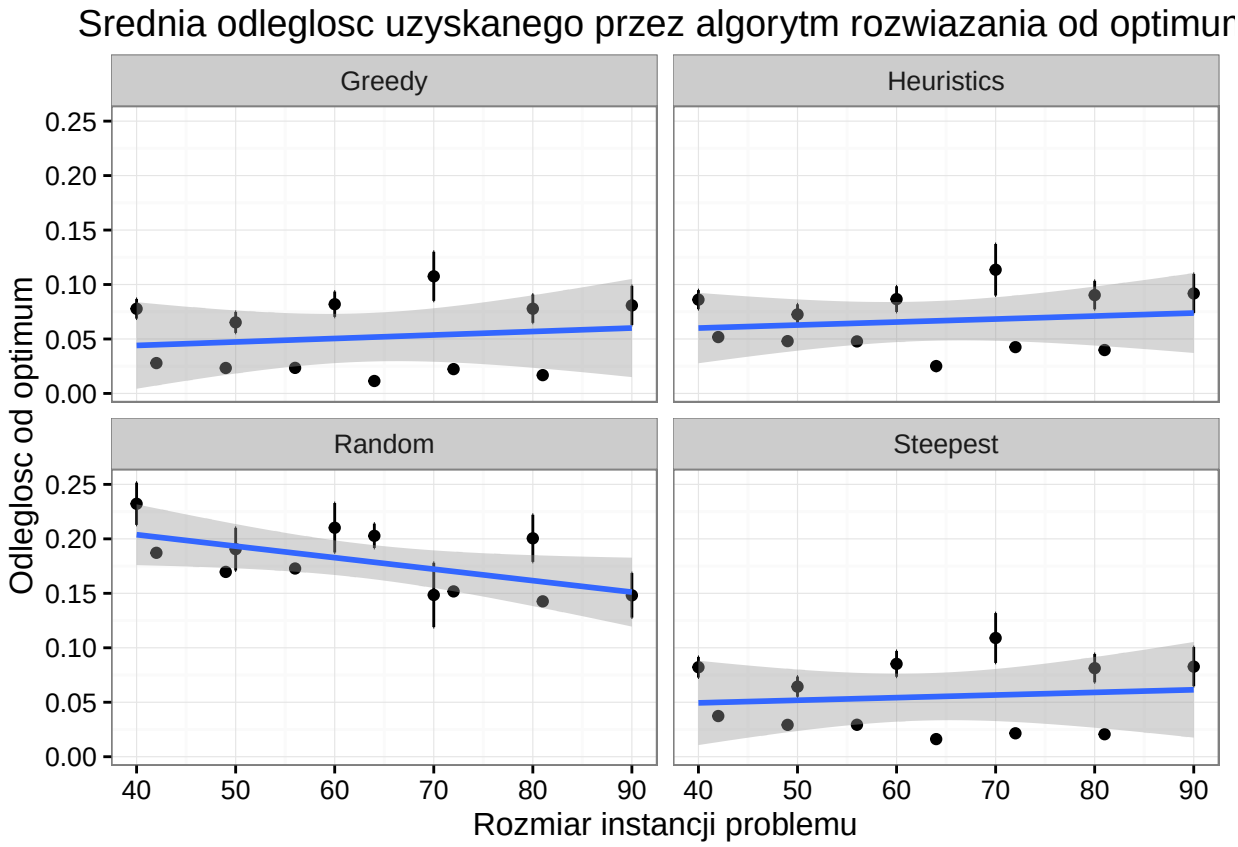
```
solution = randomSolution()
value = solution.getValue()

for (i = 0; i < n; i++) {
  for (j = i + 1; j < n; j++) {
    changeValue = valueOfChangingItemsAtPositions(i, j)
    if (changeValue < 0) {
      changeElementsAtPositions(i, j)
      value += changeValue
    }
  }
}
```

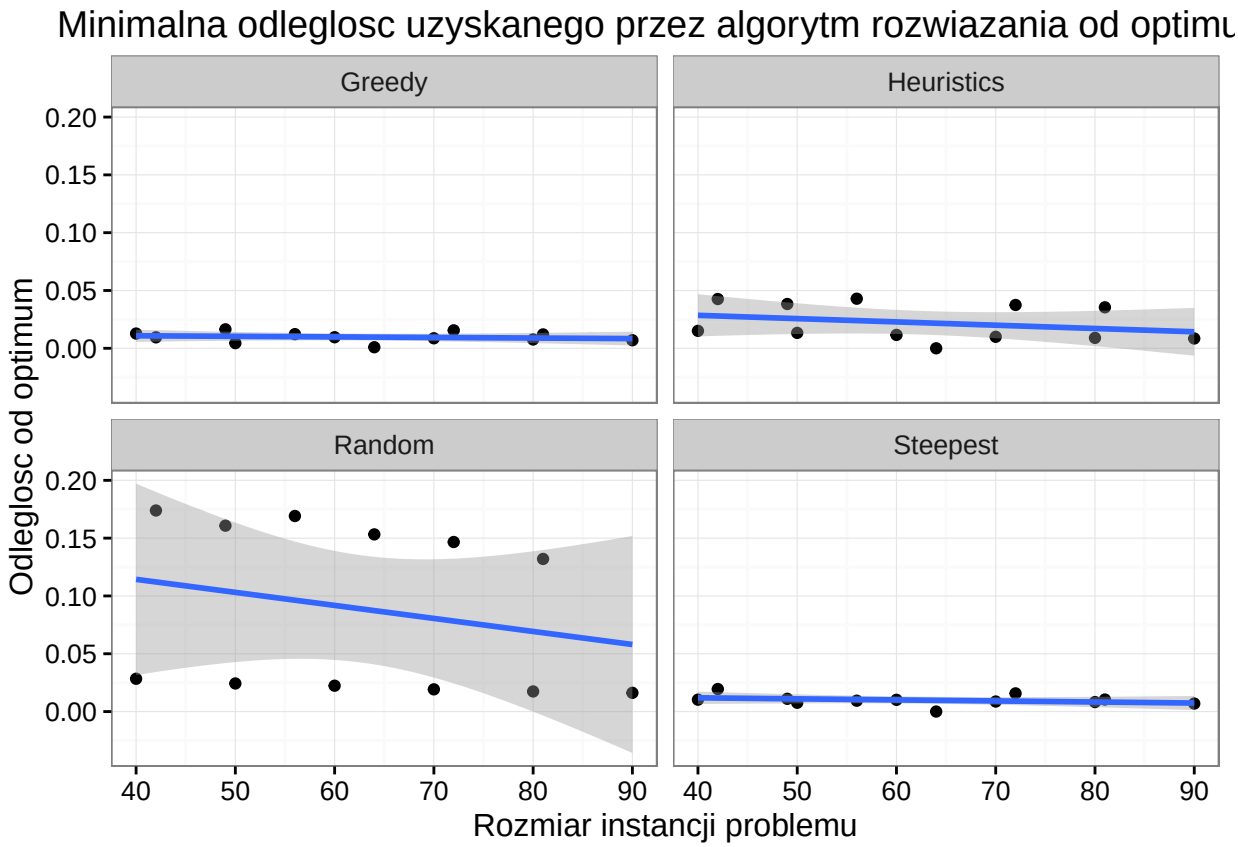
Porównanie odległości od optimum rozwiązań uzyskanych przez badane algorytmy

Do określenia odległości od optimum globalnego wykorzystano miarę opisaną wzorem:

$dist = \frac{Opt_L - Opt_G}{Opt_G}$

Dla każdej badanej instancji wykonano 10 pomiarów. Porównano wyniki średnie, oraz najlepsze. Wyniki pomiarów zostały przedstawione na poniższych wykresach.





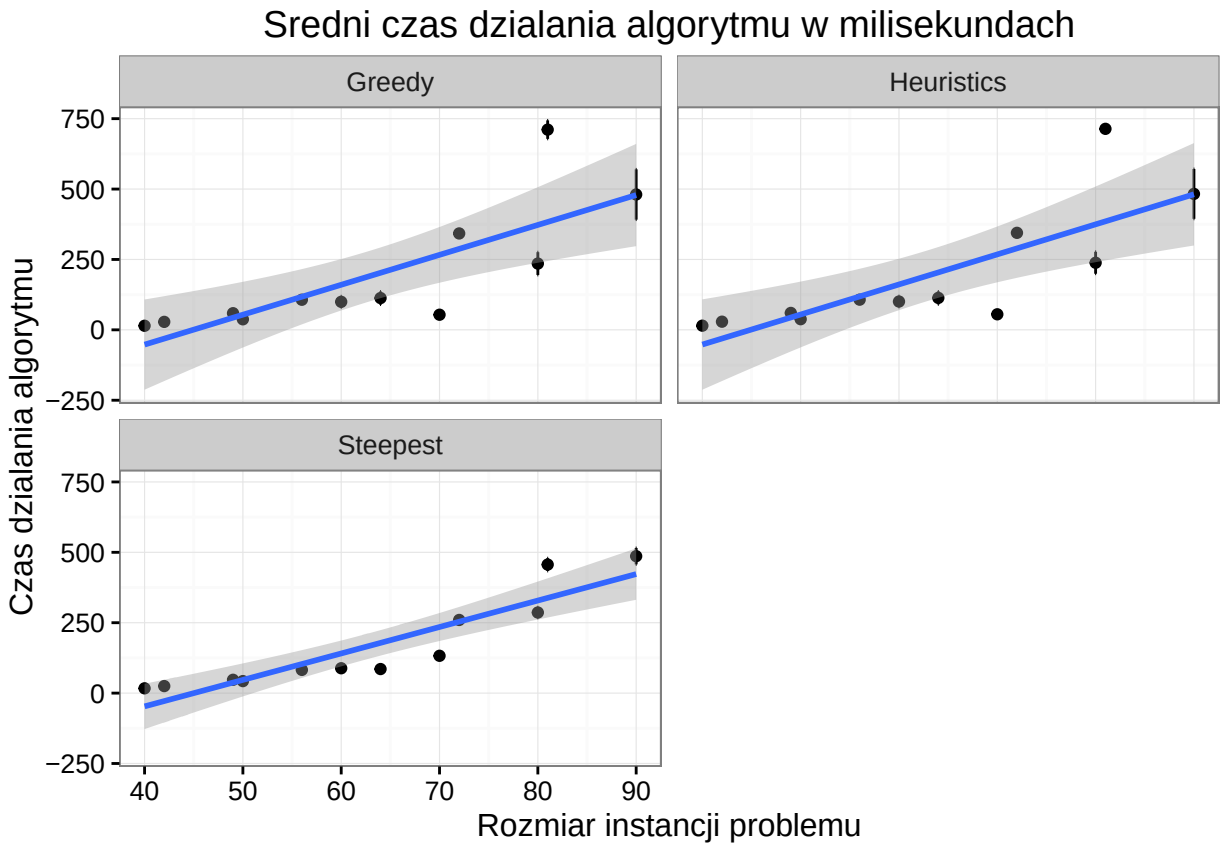
Zaobserwować można wysoką niestabilność uzyskanych wyników dla niektórych instancji problemu (duże odchylenia standardowe dla każdego algorytmu). Może to być spowodowane skomplikowaną przestrzenią rozwiązań.

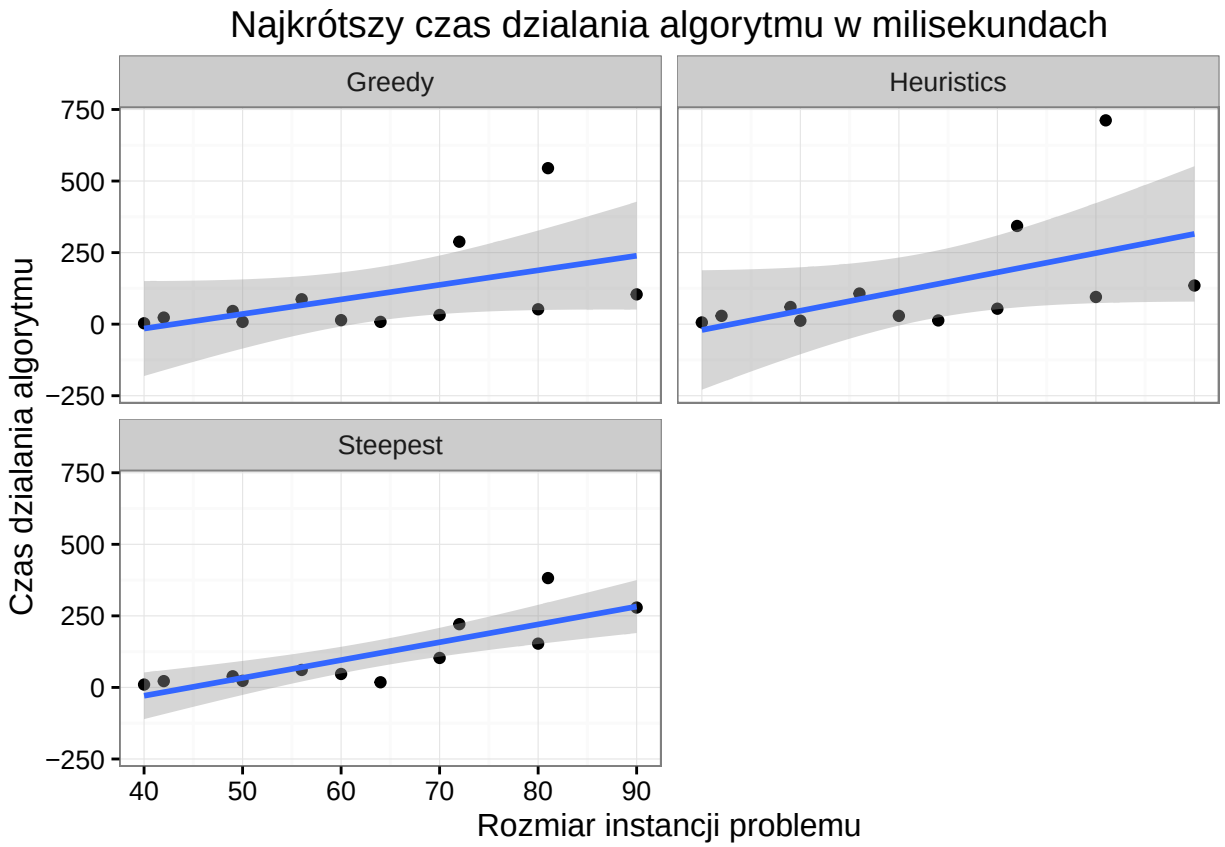
Metodą regresji liniowej, wyznaczono zależności między wielkością instancji, a względną odległością rozwiązania od optimum. Poza algorytmem losowym żaden nie wykazuje wyraźnej zależności pomiędzy wielkością instancji, a odległością znalezionego rozwiązania od optimum. Random wraz ze wzrostem wielkości instancji zwraca rozwiązania coraz bliższe optymalnemu. Jest to związane ze znacznie dłuższym czasem wykonywania algorytmu (takim, jak dla przeszukiwania lokalnego na danej instancji).

Algorytmy *Greedy* i *Steepest* osiągają bardzo podobne średnie rozwiązania. Zaproponowana heurystyka zwraca nieznacznie gorsze rozwiązania od algorytmów przeszukiwania lokalnego, natomiast *Random* osiąga rozwiązania daleko gorsze od pozostałych (poza prostymi instancjami).

Porównanie czasów wykonywania algorytmów

Ponieważ warunkiem stopu algorytmu *Random* jest upłynięcie określonego czasu (średniego czasu wykonywania *Local Search*), jego wykresy zostały pominięte.


Sredni czas dzialania algorytmu w milisekundach
Greedy
Heuristics
Steepest
Czas dzialania algorytmu
750
500
250
0
−250
40
50
60
70
80
90
Rozmiar instancji problemu



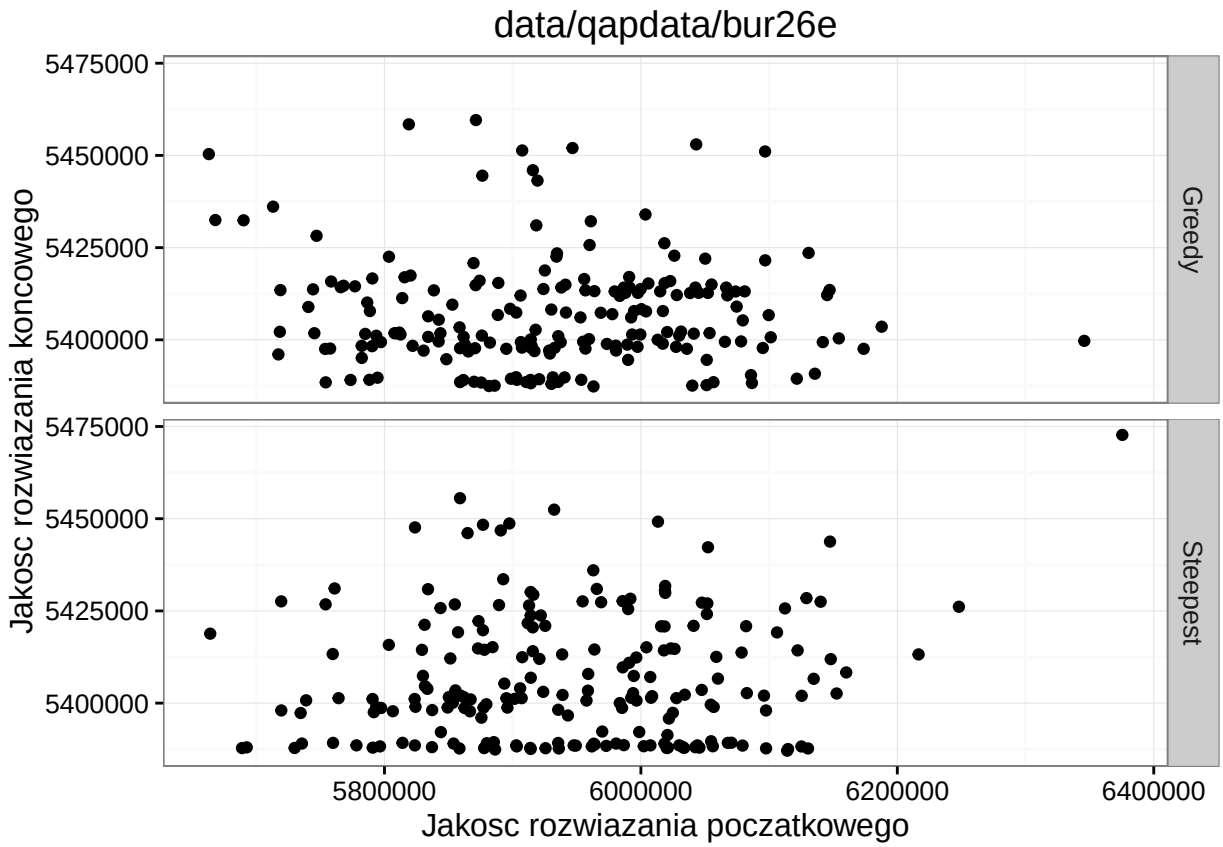
Przedstawione powyżej wykresy czasu trwania iteracji algorytmów pokazują wzrost czasu trwania algorytmu wraz ze wzrostem wielkości instancji (czego można się było spodziewać).

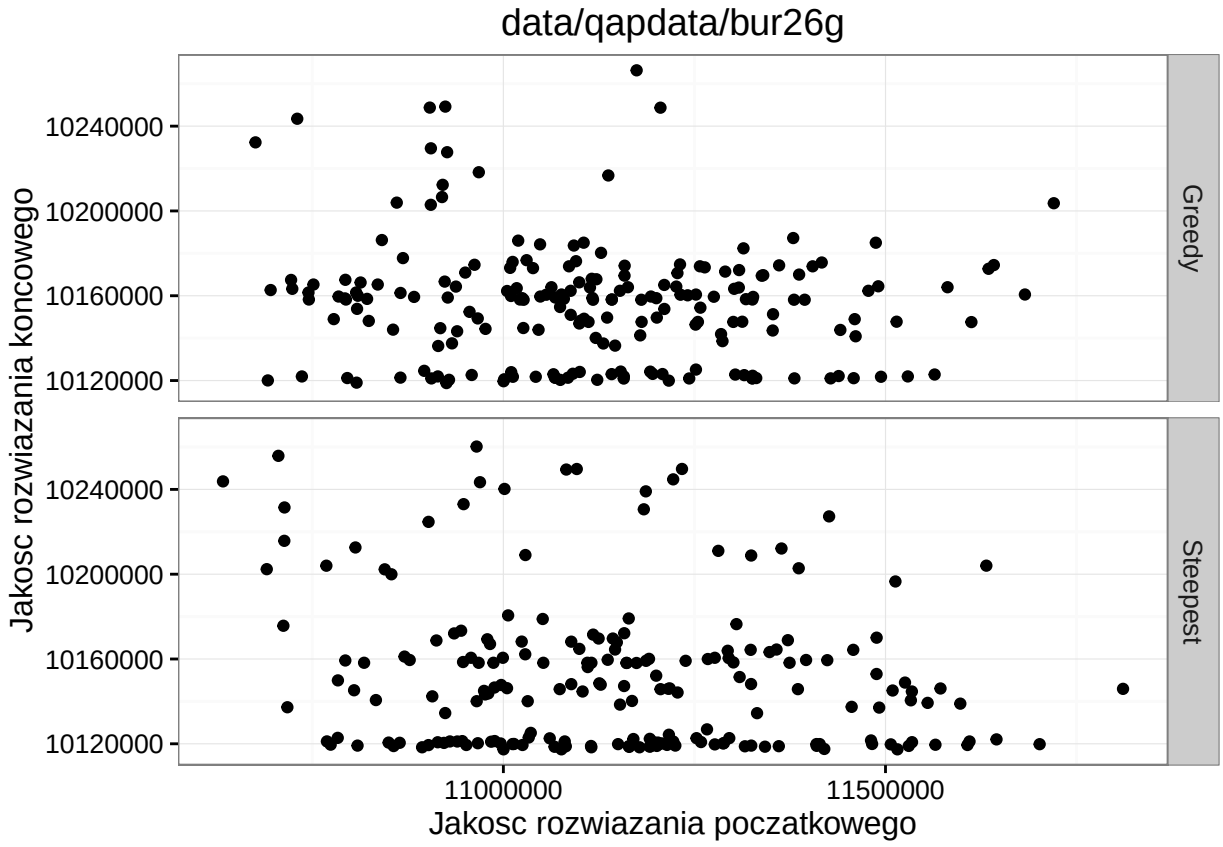
Wykresy średniego czasu działania pokazują rosnącą niestabilność mierzonego czasu (która nie jest jednak zależna jedynie od wielkości instancji). Najmniejsze odchylenia zanotowano dla algorytmu *Steepest LS*, co było do przewidzenia. W każdym kroku oceni on taką samą liczbę sąsiadów (co zajmie tyle samo czasu), więc czas trwania algorytmu zależy wyłącznie od liczby wykonanych kroków. Ponieważ dla danej instancji wykonuje on zawsze mniej więcej tyle samo kroków (potwierdzają to niewielkie odchylenia standardowe na wykresie rozmiar instancji / liczba kroków), liczba odwiedzonych sąsiadów, a więc i czas trwania algorytmu cechują się wysoką stabilnością.

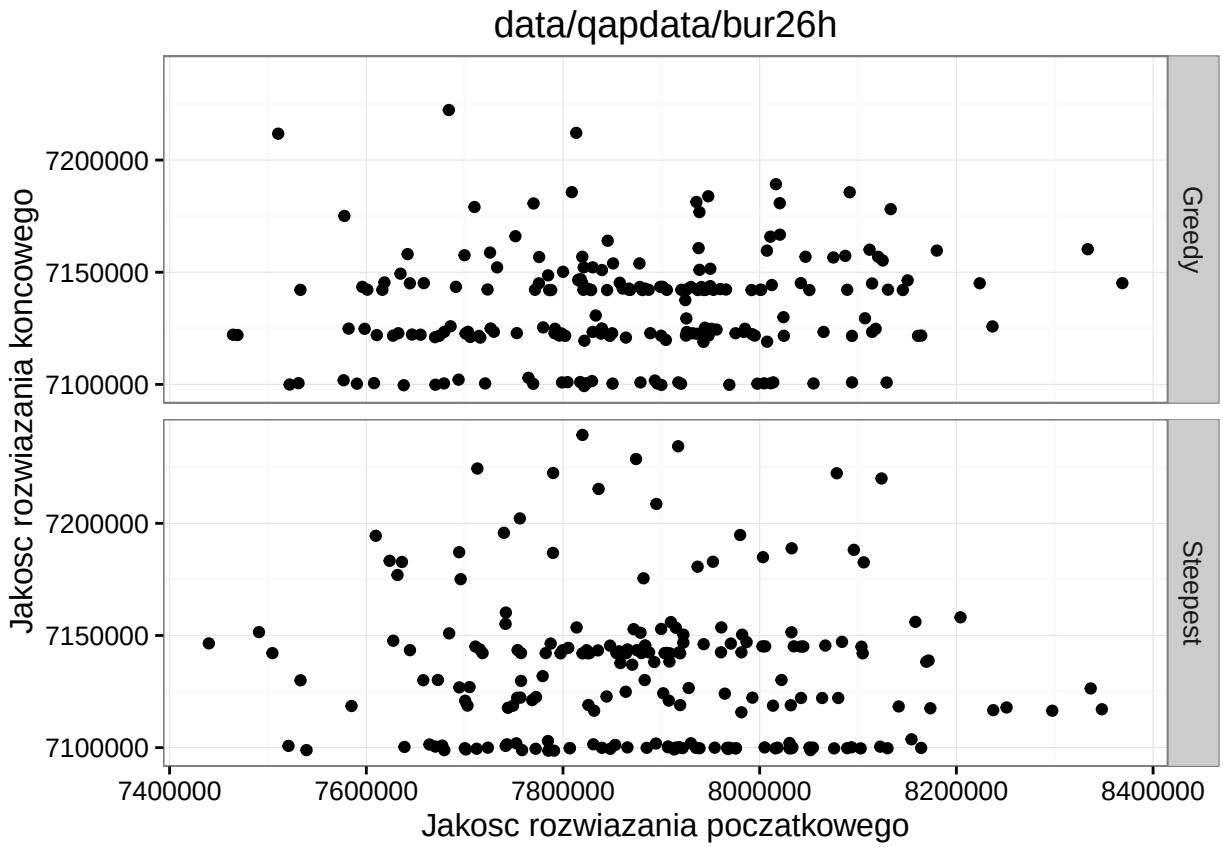
Trudno jednoznacznie określić, który z algorytmów przeszukiwania lokalnego działa szybciej. Dla niektórych instancji krótszy czas osiągnął *Steepest*, dla innych *Greedy*. Jest to silnie związane z kształtem przestrzeni rozwiązań, a nie bezpośrednio z wielkością instancji.

Zależność jakości rozwiązania końcowego od jakości rozwiązania początkowego (algorytmy przeszukiwania lokalnego)

Przeprowadzono eksperyment porównujący jakość rozwiązania początkowego z jakością rozwiązania końcowego w algorytmach przeszukiwania lokalnego. Wyniki zilustrowano poniższymi wykresami.


data/qapdata/bur26e
Jakosc rozwiazania koncowego
5475000
5450000
5425000
5400000
Greedy
Steepest
5800000
6000000
6200000
6400000
Jakosc rozwiazania poczatkowego


data/qapdata/bur26g
Jakosc rozwiazania koncowego
10240000
10200000
10160000
10120000
Greedy
Steepest
11000000
11500000
Jakosc rozwiazania poczatkowego

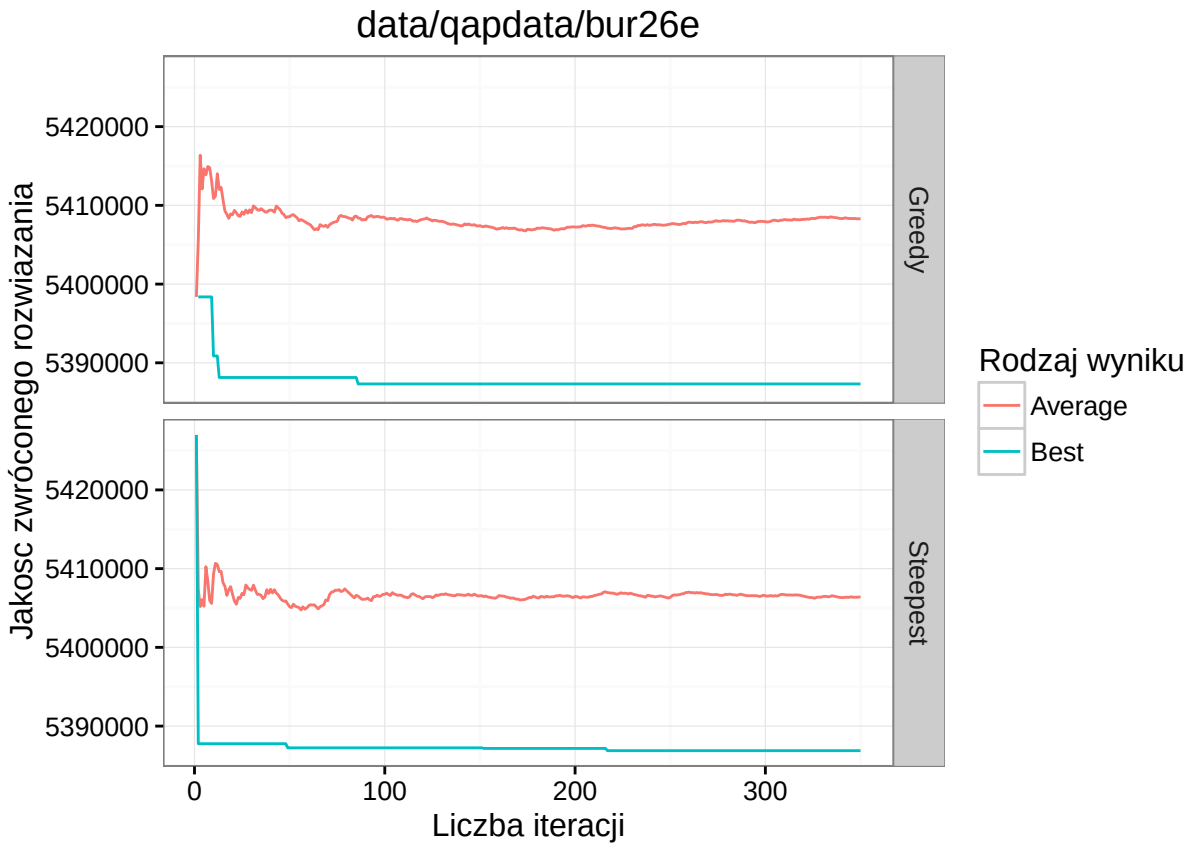


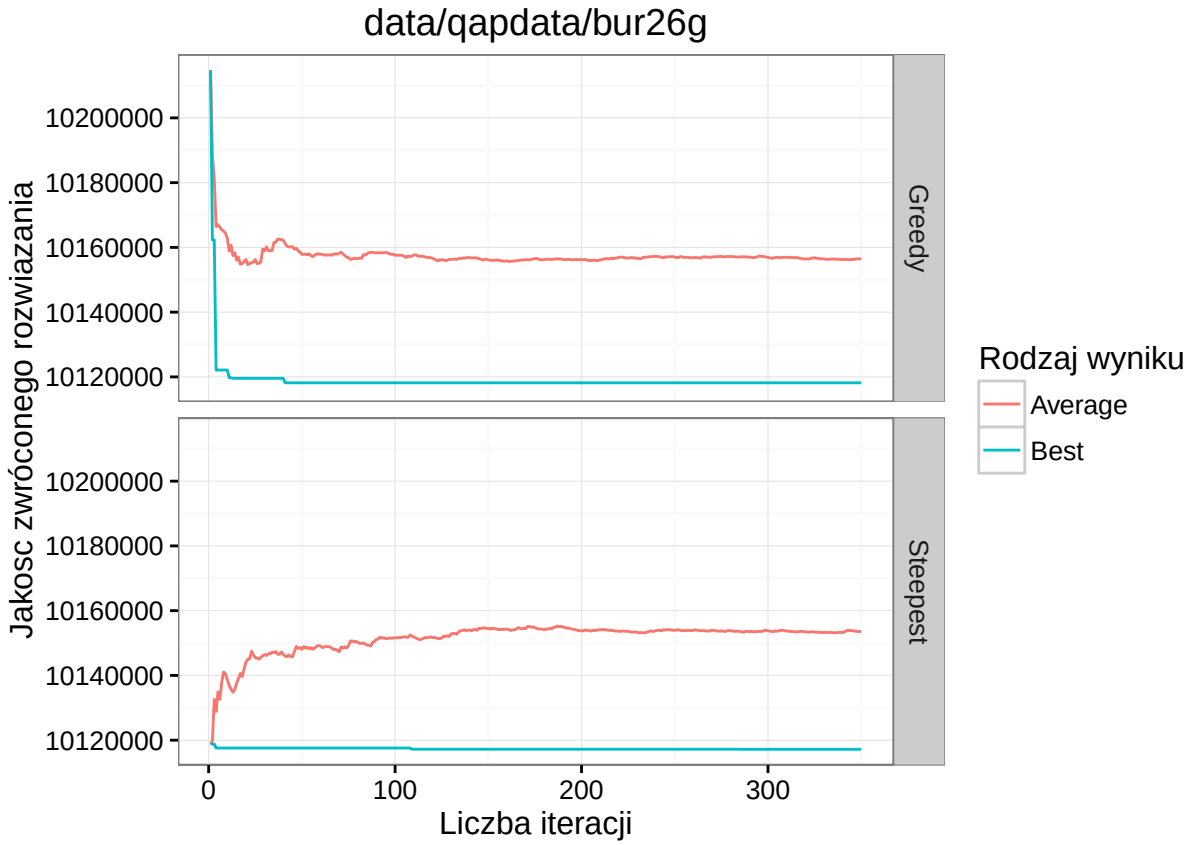
Z powyższych wykresów wynika, że nie istnieje wyraźna zależność między jakością rozwiązania początkowego, a końcowego. Pokazują one natomiast charakter badanych instancji problemu, w których określone rozwiązania występują częściej od innych (poziome "linie" na wykresach).

Brak zależności między jakością początkowego i końcowego rozwiązania wynika ze złożonego kształtu powierzchni rozwiązań. Algorytmy przeszukiwania lokalnego znajdują lokalne optimum, a nie mamy żadnej gwarancji, że w okolicach dobrego rozwiązania będzie dobre optimum lokalne. Jedyną gwarancją daną nam przez algorytmy przeszukiwania lokalnego jest fakt, że rozwiązanie końcowe będzie zawsze nie gorsze od początkowego.

Multi-random local search: Zależność uzyskanego rozwiązania od liczby restartów

Dla dwóch algorytmów przeszukiwania lokalnego (*Greedy* i *Steepest*) przeprowadzono eksperyment sprawdzający zależność jakości uzyskanego rozwiązania od ilości restartów (ponownych przeszukiwań lokalnych zaczynających od losowo wybranych punktów początkowych). Wyniki przedstawiono na poniższych wykresach.


data/qapdata/bur26e
Jakosc zwróconego rozwiazania
5420000
5410000
5400000
5390000
Greedy
Steepest
0
100
200
300
Liczba iteracji
Rodzaj wyniku
Average
Best


data/qapdata/bur26g
Jakosc zwróconego rozwiazania
10200000
10180000
10160000
10140000
10120000
Greedy
Steepest
0
100
200
300
Liczba iteracji
Rodzaj wyniku
Average
Best

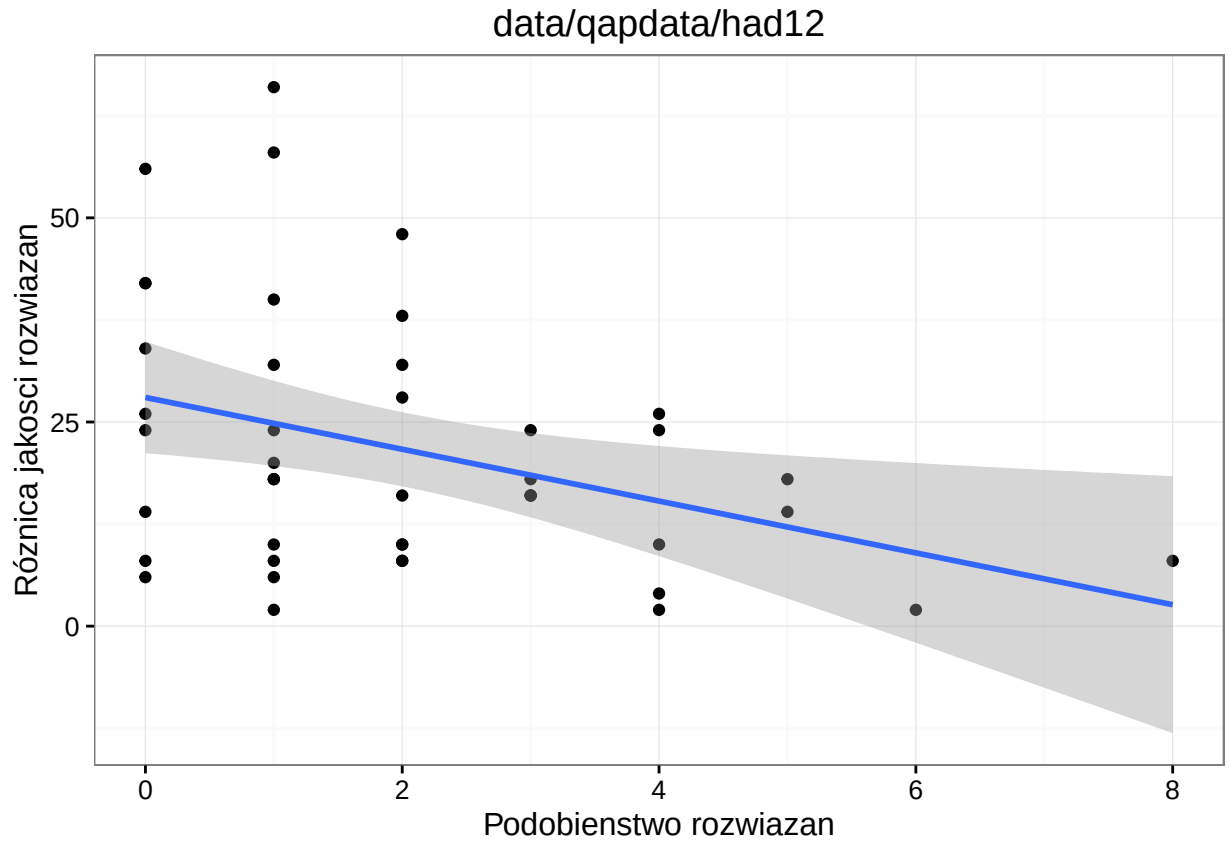
Zamieszczone wykresy pokazują, że algorytmy stosunkowo szybko (już po 2 - 3 iteracjach) osiągają rozwiązanie bliskie końcowemu (gdzie za końcowe uznajemy takie, które nie zmienia się przez kilkadziesiąt iteracji). Dalsze iteracje pomagają natomiast w coraz wolniejszym tempie poprawiać uzyskane rozwiązanie. Algorytm *Steepest* szybciej osiąga końcowe rozwiązanie. Sugeruje to, że efektywniej eksploruje on przestrzeń. Wybieranie za każdym razem najlepszego sąsiada może więc prowadzić do większej różnorodności znajdowanych optimów lokalnych po wielokrotnym uruchomieniu algorytmu. Więcej znalezionych optimów lokalnych zwiększa natomiast prawdopodobieństwo znalezienia optimum globalnego.

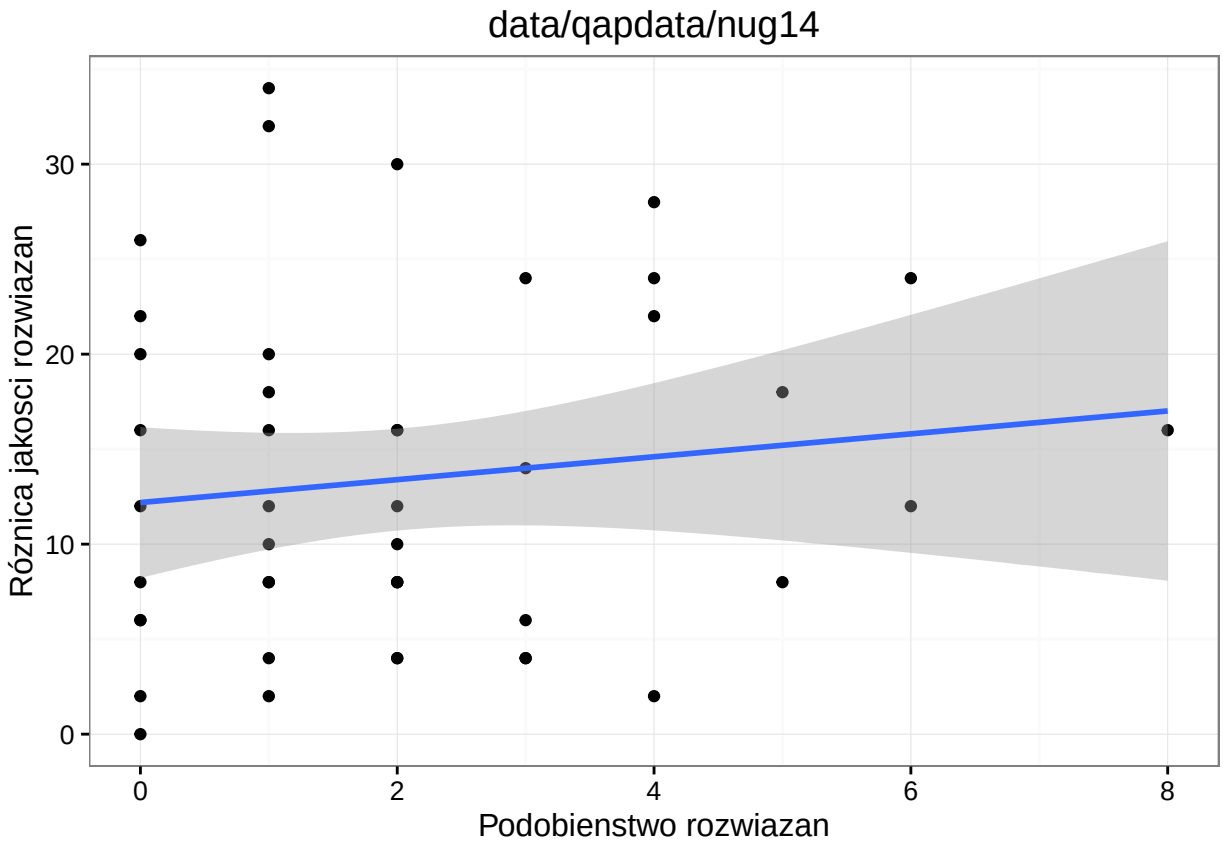
Ocena podobieństwa znajdywanych rozwiązań lokalnie optymalnych

Określono podobieństwo między rozwiązaniami znalezionymi przez algorytmy przeszukiwania lokalnego. Podobieństwo zostało zdefiniowane, jako liczba pozycji, na których rozwiązania mają równe wartości. Przykładowo:

Permutacja 1	Permutacja 2	Podobieństwo
2 9 10 1 11 5 4 0 7 6 3 8	7 9 1 10 11 6 4 0 2 5 3 8	6
2 9 10 1 11 5 4 0 7 6 3 8	2 9 10 1 11 4 5 6 7 0 3 8	8
3 2 13 12 1 5 6 4 7 0 9 11 10 8	3 13 12 1 0 5 2 4 7 8 9 11 6 10	6
3 13 2 5 9 1 12 4 8 11 0 6 7 10	5 13 3 2 9 4 12 6 8 11 0 1 7 10	8
10 8 7 11 9 1 12 2 6 5 0 13 3 4	10 8 11 2 9 7 12 4 13 5 0 1 6 3	6

Sporządzono wykresy próbując zbadać zależność między podobieństwem, a różnicą w jakości dla par rozwiązań. Badanie przeprowadzono na dwóch niewielkich instancjach.





Wykres dla pierwszej z badanych instancji przedstawia relację malejącą (im bardziej podobne rozwiązania, tym mniejsza różnica w ich jakości), podczas, gdy drugi nie przejawia wyraźnej relacji omawianych wielkości. Rozwiązania znalezione dla drugiej instancji są również mniej podobne do siebie wzajemnie - dla instancji problemu *had12* pary rozwiązań mają podobieństwo sięgające nawet 10 (na maksimum 12), natomiast dla *nug14* najwyższe podobieństwo wyniosło 6/14.

Wnioski

W ramach zadania zaimplementowano i przetestowano 4 algorytmy rozwiązujące problem QAP: *Random Search*, *Greedy Local Search*, *Steepest Local Search* oraz autorski algorytm heurystyczny. Do przeszukiwania przestrzeni rozwiązań skorzystano z sąsiedztwa 2-OPT, które dla każdego rozwiązania zwraca n2 jego sąsiadów.

W pierwszej kolejności porównano liczbę wykonanych kroków oraz sprawdzonych sąsiadów przez algorytmy *Greedy LS* i *Steepest LS*. Okazało się, że jakkolwiek *Steepest* wykonuje znacznie mniej kroków dla każdej badanej instancji, liczba sprawdzonych przez niego sąsiadów w pełnej iteracji jest zazwyczaj wyższa od "konkurenta".

Względna odległość znalezionego rozwiązania od optimum globalnego jest niezależna od wielkości instancji, a dla algorytmu *Random Search* jest ona nawet tym mniejsza, im większa jest instancja problemu. Jest to jednak w dużej mierze znacznym wydłużeniem czasu działania RS dla większych instancji.

Czas wykonywania algorytmów rośnie wraz ze wzrostem wielkości instancji. Jest to spodziewany wniosek, jednak należy zauważyć, że od tej reguły są wyjątki i długość permutacji nie jest jedynym kryterium determinującym czas wykonywania algorytmu. Najbardziej stabilny czas wykonywania wykazuje *Steepest Local Search*, co jest zrozumiałe biorąc pod uwagę, że w każdym kroku przeszukuje całe dostępne sąsiedztwo (które ma równą liczność w każdym punkcie przestrzeni rozwiązań).

Porównanie czasu wykonywania algorytmów *Greedy* i *Steepest* prowadzi do wniosku, że żaden z nich nie jest jednoznacznie lepszy od drugiego. Zawsze znajdzie się instancja, dla której *Greedy* zakończy się szybciej oraz taka, dla której to *Steepest* osiągnie lepszy czas.

Nie można wskazać wyrażnej zależności między jakością rozwiązania początkowego, a końcowego w algorytmach przeszukiwania lokalnego. Jedyną pewną zależnością jest, że zwrócone rozwiązanie będzie się charakteryzowało jakością nie gorszą od początkowego.

Multi-random Local Search (wielokrotne uruchamianie przeszukiwania lokalnego w losowych punktach) stanowi bardzo sprawne ulepszenie algorytmów LS. Z przeprowadzonych eksperymentów wynika, że dobre rozwiązanie osiągane jest stosunkowo szybko. Wielokrotne uruchamianie *LS* pozwala na znalezienie większej liczby optimów lokalnych, a w konsekwencji zwiększa szansę znalezienia rozwiązania optymalnego globalnie.

Na koniec oceniono podobieństwo znajdowanych przez *LS* rozwiązań oraz podjęto próbę znalezienia relacji między podobieństwem rozwiązań, a różnicą ich jakości. Wykorzystano miarę równą liczbie pozycji, na których permutacje mają równe wartości. Zauważono, że dla niektórych instancji może występować wyraźna zależność między podobieństwem rozwiązań, a różnicą ich jakości.

Napotkane trudności

Przy wykonywaniu zadania nie natrafiono na żadne istotne trudności.