Algorytmy optymalizacji lokalnej i globalnej, problem STSP

Karol Bonenberg <kbonenberg@cs.put.poznan.pl>

16.12.2009

1. Opis problemu

Problem komiwojażera (*ang. traveling salesman problem,TSP*) jest jednym z fundamentalnych problemów optymalizacji kombinatorycznej. Polega on na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. W wersji symetrycznej (*ang. symmetric traveling salesman problem,STSP*) waga przy przejściu między parą miast jest taka sama, niezależnie od kierunku w jakim ją przebywamy.

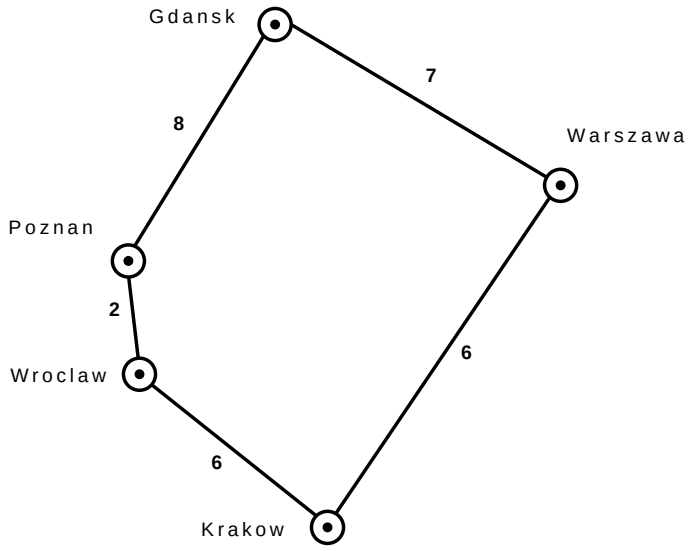
1.1 Sformułowanie problemu

Zbiór miast definiujemy jako zbiór n punktów geograficznych. Możemy go rozpatrywać jako pełny graf G o n wierzchołkach. Z każdą krawędzią *e(i, j)* kojarzymy wagę równą odległości euklidesowej pomiędzy miastami *i* oraz *j*. Wagę tę oznaczmy *w(i, j)*. Problem sprowadza się do wyznaczenia minimalnego cyklu Hamiltona w grafie G. Jako cykl Hamiltona rozumiemy ścieżkę wychodzącą z jednego wierzchołka, przechodzącą przez wszystkie pozostałe dokładnie raz i wracającą do początkowego wierzchołka. Koszt ścieżki z miasta i do j jest taki sam jak z j do i. Każde rozwiązanie R (droga) jest reprezentowane przez wektor, którego elementami są kolejne odwiedzane wierzchołki. R_i stanowi numer miasta odwiedzanego jako i-te. Koszt całej drogi można policzyc z poniższego równania:

$$C_R = \sum_{i=1}^{n-1} (C_{Ri}, R_{i+1}) + C_{Rn,R1} \tag{1}$$

1.2 Cechy problemu

Na wejściu dany jest zbiór punktów przestrzeni, które można przekształcić w pełny graf nieskierowany. W takim grafie znajduje się co najmniej jeden cykl Hamiltona. Ponieważ graf ma skończoną liczbę wierzchołków, to w zbiorze cykli Hamiltona istnieje taki (niekoniecznie jedyny), który posiada minimalna sumę wag krawędzi. Jest to problem NP-trudny, który wymaga przeanalizowania dużej ilości przypadków. Dla n miast mamy $\frac{(n-1)!}{2}$ rozwiązań.



Rysunek 1: Przykładowe rozwiązanie problemu STSP

2. Algorytmy

2.1 Implementacja

Funkcja odległości zrealizowana jest za pomocą kwadratowej macierzy $n \times n$, gdzie komórka i, j oznacza odległość między i-tą oraz j-tą lokalizacją.

Bieżące rozwiązanie problemu przechowywane jest w postaci permutacji o długości równej rozmiarowi problemu. Pojedyncza liczba w permutacji wskazuje na numer obiektu (miasta), natomiast pozycja na której się znajduje nie ma znaczenia, gdyż początek permutacji łączy się z końcem i jako element początkowy wynikowej trasy możemy obrać dowolny element permutacji.

Funkcja oceny odpowiada sumie wag (odległości) pomiędzy kolejnymi miastami (wzór 1).

2.2 Operator sąsiedztwa

Dla każdej permutacji można wygenerować permutację sąsiednią (sąsiada) powstałą poprzez zamianę dwóch różnych elementów miejscami. Dla każdej permutacji istnieje $\frac{n(n-1)}{2}$ różnych sąsiadów.

2.3 Algorytm „Random"

Zaczyna od zadanej permutacji (losowej), wybiera losowego sąsiada i przechodzi do niego. Robi tak aż wyczerpie przypisany sobie czas, lub liczbę przejść. Na potrzeby testów ograniczyliśmy jego czas działania do 100 ms.

2.4 Prosty Algorytm Heurystyczny „Heuristic"

Buduje rozwiązanie od zera wybierając kolejne miasta. Losowo wybiera pierwsze z miast, po czym wyszukuje(spośród pozostałych) miasta najmniej oddalonego i dodaje je jako następne za tym pierwszym. Następnie szuka miasta najmniej oddalonego od drugiego i tak dalej aż do zbudowania pelnego rozwiązania. Ma złożoność $O(\frac{n(n-1)}{2})$.

2.5 Algorytm „Greedy"

Rozpoczyna od zadanego rozwiązania (dla testów porównawczych od losowego) i przeszukuje kolejnych sąsiadów do momentu znalezienia pierwszego rozwiązania, które polepszy ocenę. Przechodzi do tego sąsiada i powtarza przeszukiwanie do momentu, gdy wsród sąsiadów nie ma żadnego, który polepszałby ocenę.

2.6 Algorytm „Steepest"

Analogicznie do algorytmu Greedy i Random może startować z zadanej permutacji, jednak dla porównania startuje z losowej. Przegląda całe sąsiedztwo wybierając najlepszego sąsiada i przechodząc do niego. Tę czynność powtarza do momentu w którym żaden z sąsiadów nie polepsza oceny.

2.7 Algorytm „Simulated Annealing"

W odróżnieniu od poprzednich algorytmów, SA oraz TS akceptują ruchy pogarszające aktualne rozwiązanie. Jest to konieczne, aby wyjść z minimów lokalnych i zbliżyć się do optimum globalnego. Wiąże się to oczywiście z ryzykiem wpadnięcia w cykle. Każdy z algorytmów rozwiązuje ten problem inaczej. Algorytm Simulated Annealing (Symulowane Wyrzażanie) czerpie ideę z fizycznego procesu wyżarzania. Temperatura w tym procesie posiada w algorytmie analogię do skłonności

do wykonania ruchu niepolepszającego rozwiązania (im wyższa tym większa skłonność). Poniższy wzór przedstawia prawdopodobieństwo przejścia z rozwiązania R do R', przy temperaturze t.

$$P_t(R') = e^{\frac{CR-CR'}{t}} \tag{2}$$

Algorytm rozpoczyna obliczenia w stanie wysokiej temperatury (prawdopodobieństwo zaakceptowania dowolnego ruchu jest wysokie i wynosi 95%). Dobór temperatury jest doświadczalny na podstawie próbki 1000 najbliższych sąsiadów rozwiązania początkowego. Wraz z upływem czasu (kolejnych iteracji) temperatura (a co za tym idzie - prawdopodobieństwo akceptacji dowolnego ruchu) spada, aż akceptowane są praktycznie tylko rozwiązania polepszajace jego jakość. Dzięki temu na początku przeszukiwane są rozwiązania mocno odległe od siebie, natomiast w końcowej fazie znajdowane jest minimum lokalne.

Warunkiem stopu jest przekroczenie maksymalnego czasu działania (5 min) lub maksymalnej liczby pogorszonych rozwiązań w serii (rozmiar instancji do potęgi drugiej).

Rozwiązaniem końcowym jest najlepsze, znalezione w czasie trwania przeszukiwania, rozwiązanie.

2.8 Algorytm „Tabu Search"

Działa podobnie do algorytmu Greedy. Aby uniknąć wpadnięcia w cykle stosuje listę tabu, która przechowuje ostatnio wykonane ruchy (elementy o których numerach zostały zmienione miejscami). Ruchy są pamiętane przez n/2 iteracji. Jeżeli ruch znajduje się na liście tabu, nie wykonuje się go. Dodatkowo algorytm został wzbogacony o tzw. „Masters List". Rozwiązanie to polega na tym, że w początkowej iteracji wybiera się n/2 najlepszych sąsiadów (zamian np. element 3 listy zamień z elementem 4) i wykonuje się kolejno odpowiadające im zamiany do momentu w którym zostaną wykonane wszystkie, lub ocena rozwiązania po zamianie byłaby gorsza niż ocena rozwiązania najgorszego z masters listy. Algorytm ten, podobnie jak SA, kończy pracę kiedy minie maksymalny czas (5 min), lub łącznie nastąpiło 10n kroków pogarszających rozwiązanie.

W celu przyspieszenia działania algorytmu lista tabu jest zaimplementowana jednoczesnie za pomocą macierzy (aby operacje sprawdzania, czy ruch znajduje się na liście wykonywać w czasie O(1)) oraz kolejki (aby operacje aktualizacji listy odbywały się w czasie O(1) — bez konieczności przeglądania macierzy).

3. Eksperyment

3.1 Opis eksperymentu

Aby zapewnić porównywalność uzyskanych rozwiązań eksperyment zaprojektowano w taki sposób, aby wszystkie obliczenia zostały wykonane w możliwie takich samych warunkach oraz żeby można było łatwo je powtórzyć. W tym celu cały proces zautomatyzowano, a uzyskane wyniki przedstawiono na wykresach. Badania ustandaryzowano w następujący sposób:

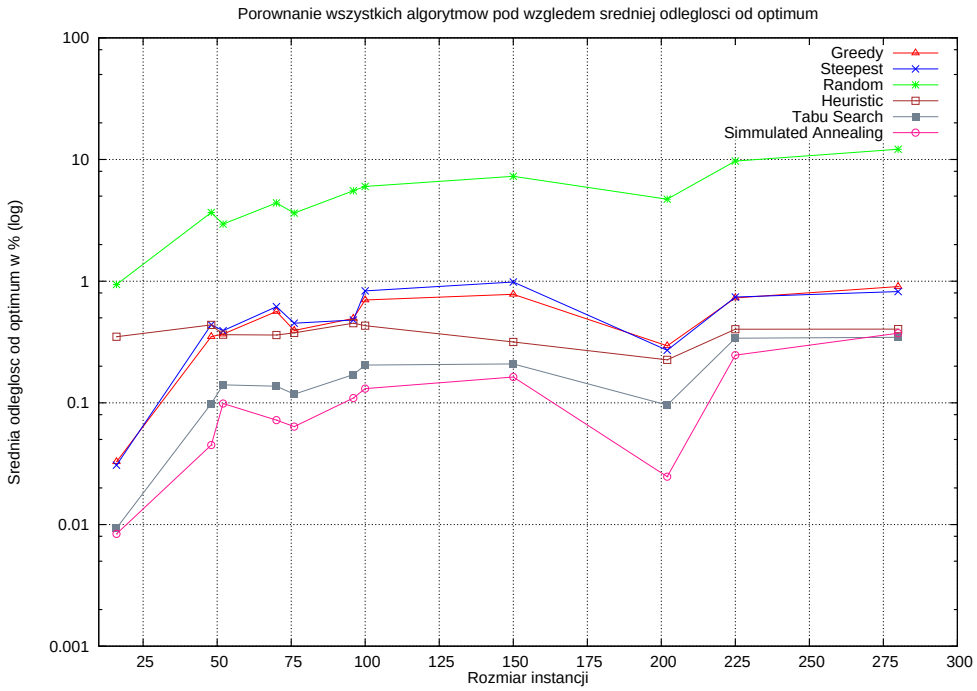
- wszystkie badania przeprowadzono na jednym komputerze w środowisku Windows 2003 Server na maszynie wirtualnej JVM
- wszystkie algorytmy zostały przetestowane na tych samych 11 instancjach problemu
- każdy algorytm był wykonywany 30 razy dla każdej instancji (4 * 11 * 30 = 1320 wykonań; zastosowano tu tryb multi-random)
- każde wykonanie wszystkich algorytmów (poza Heuristic) dla danej instancji startowało z tego samego, losowego rozwiązania
- wyniki 30-krotnego wykonania algorytmu na każdej instacji zostały uśrednione, zapamiętując przy tym wynik najlepszy, najgorszy, średni i wartość odchylenia standardowego

3.2 Przeprowadzone badania

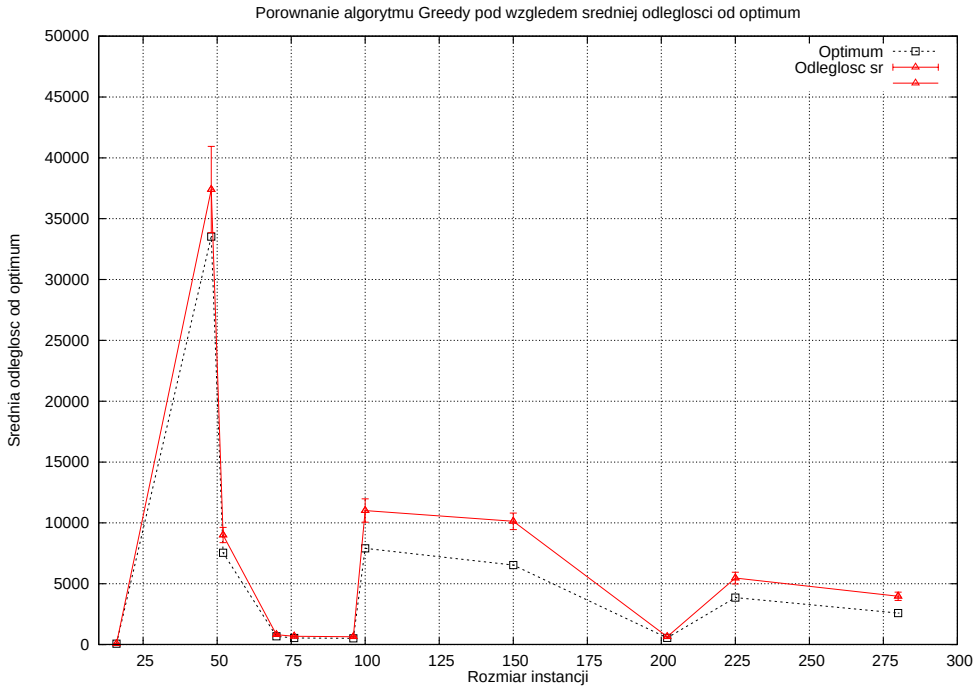
Odległość od optimum w funkcji rozmiaru instancji

Odległość od optimum jest liczona według poniższego wzoru (C_* jest to koszt rozwiązania optymalnego). Jego wynikiem jest procentowa odległość od najlepszego rozwiązania.

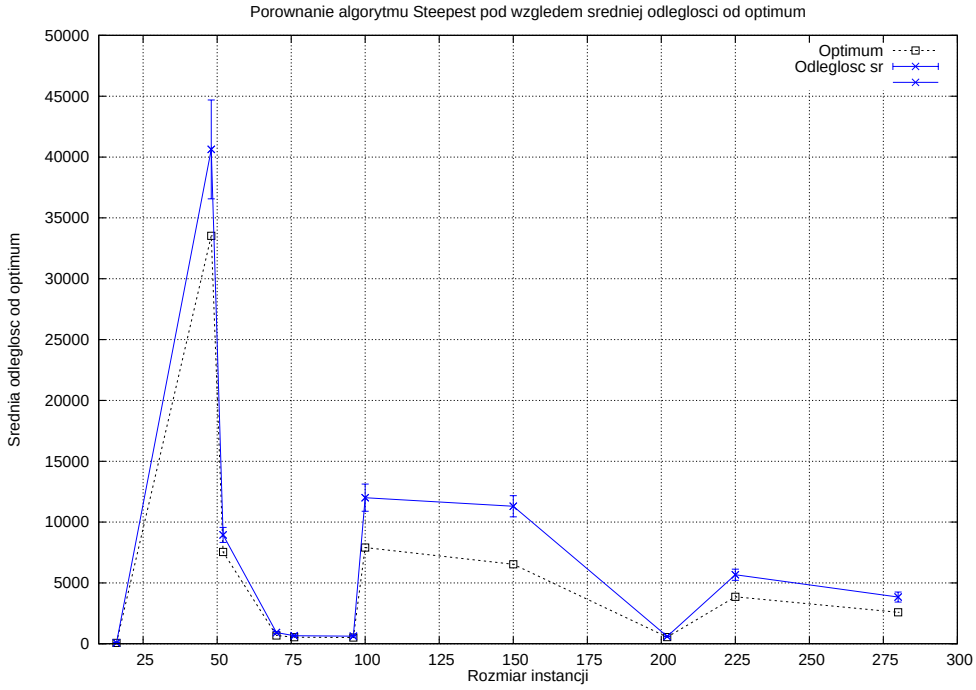
$$O_{opt} = \frac{C_R}{C_*} - 1 \tag{3}$$



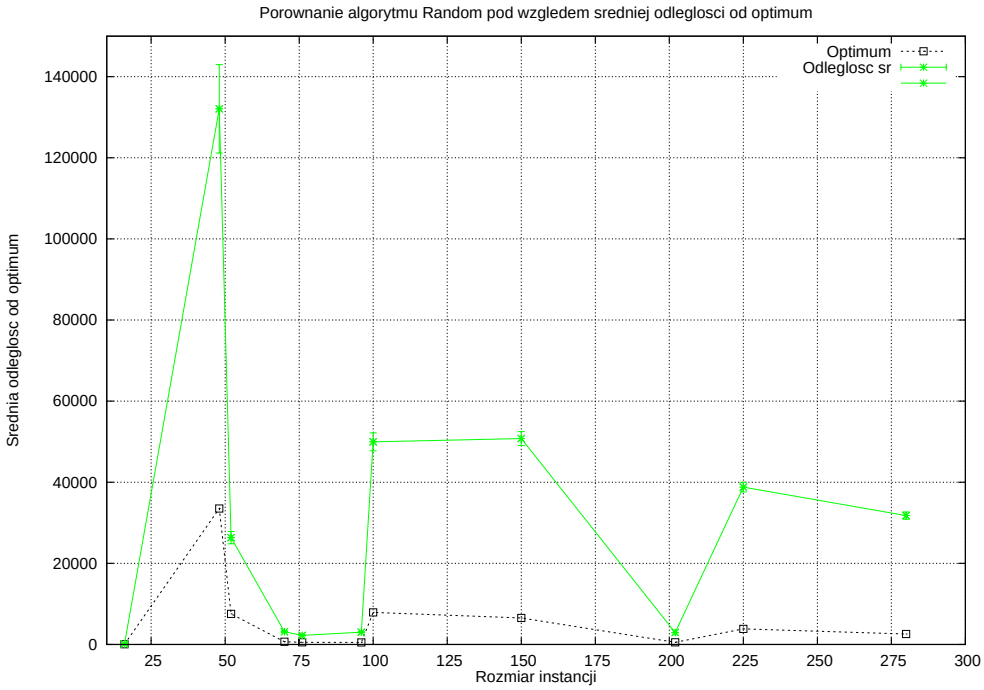
Rysunek 2: Porównanie wszystkich algorytmów pod względem odległości od optimum



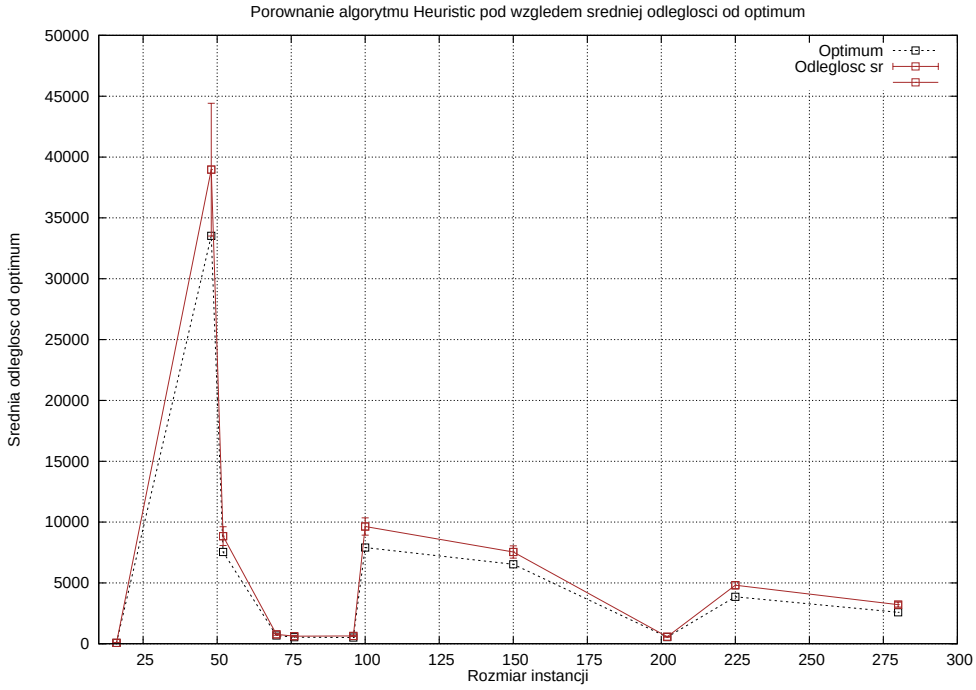
Rysunek 3: Wyniki algorytmu Greedy dla średniej odległości od optimum



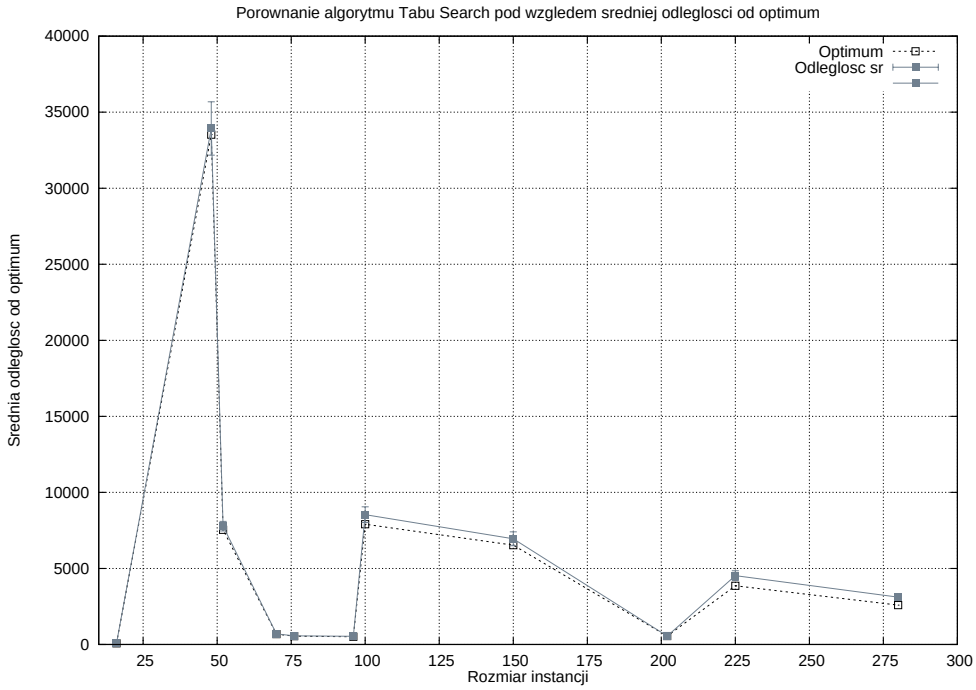
Rysunek 4: Wyniki algorytmu Steepest dla średniej odległości od optimum



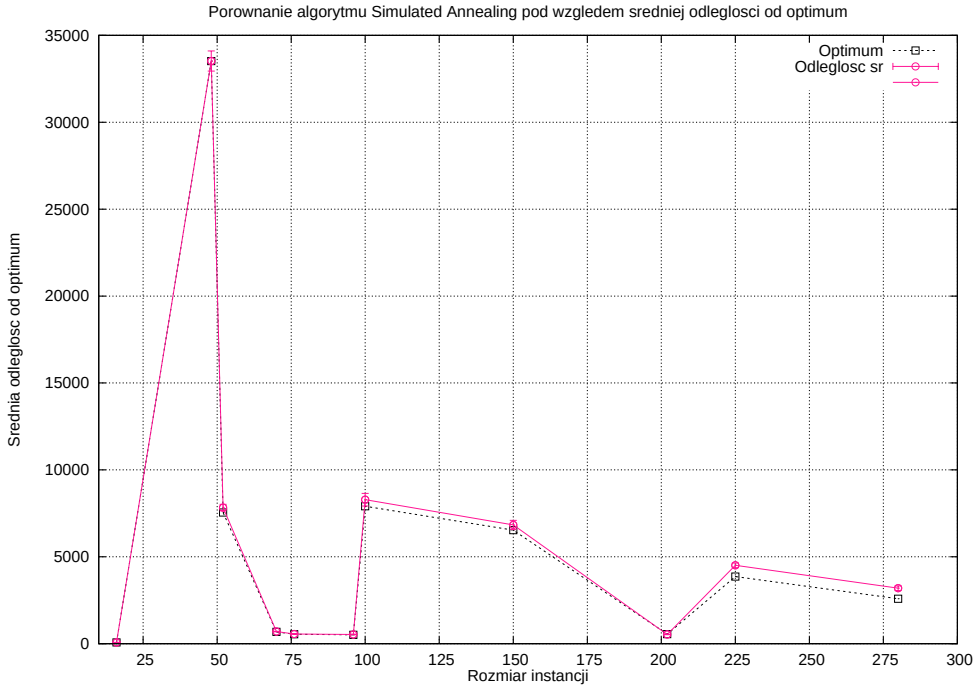
Rysunek 5: Wyniki algorytmu Random dla średniej odległości od optimum



Rysunek 6: Wyniki algorytmu Heuristic dla średniej odległości od optimum



Rysunek 7: Wyniki algorytmu Tabu Search dla średniej odległości od optimum

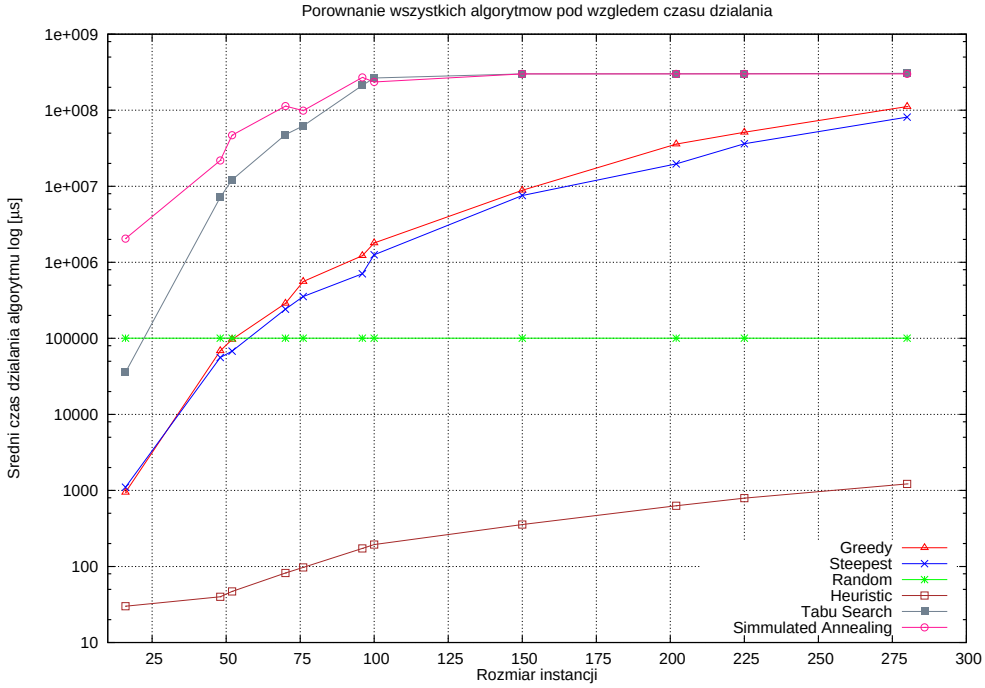


Rysunek 8: Wyniki algorytmu Simulated Annealing dla średniej odległości od optimum

- Spodziewano się, ze najgorzej w porównaniu wypadnie algorytm Random, nieco lepiej Heuristic, a o wiele lepiej, metaheurystyki, Tabu Search i Simmulated Annealing. Jak pokazuje rysunek 2 przewidywania nie do końca się sprawdziły. Rysunek 2 pokazuje, że rozwiązania najbliższe optimum na większości instancji uzyskiwał algorytm Simmulated Annealing, dalej był Tabu Search, Heuristic, metaheurystyki i na końcu Random.
- Algorytmy Greedy i Steepest uzyskują bardzo zbliżone do siebie wyniki na wszystkich instancjach i w niektórych przypadkach konkurują z Heuristic.
- Algorytm Random zgodnie z przewidywaniami na wszystkich instancjach uzyskuje wyniki najgorsze.
- Przy zastosowanym sąsiedztwie algorytmy przeszukiwania globalnego są wyraźnie lepsze.
- Odległość od optimum zależy raczej od konkretnego przykładu, niż rozmiaru instancji.

Czas działania

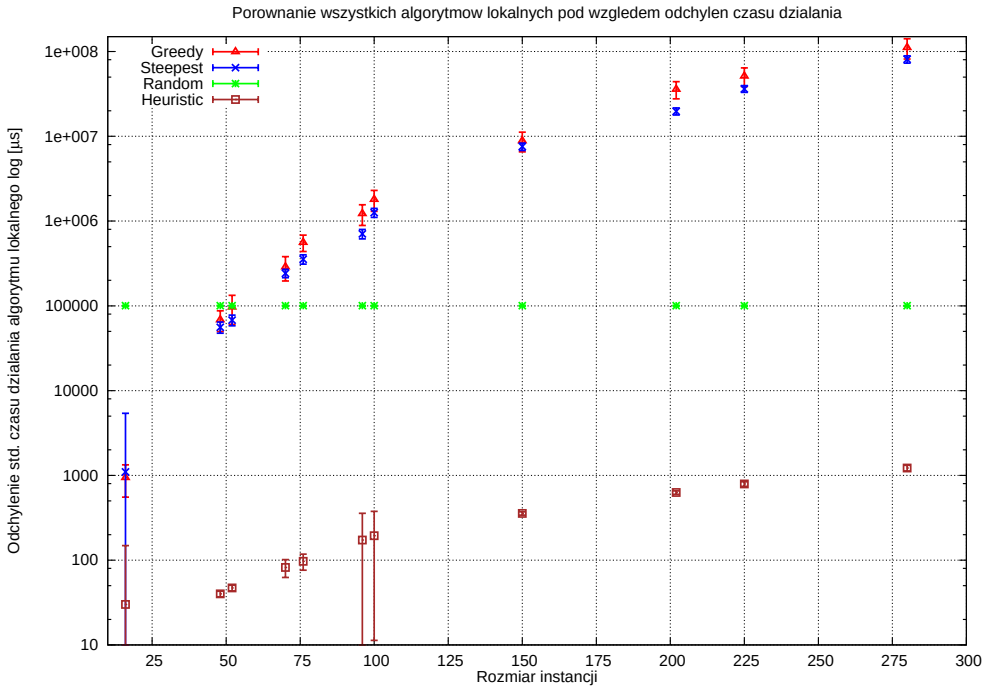
Na wykresie 10 przedstawiono czas działania w funkcji rozmiaru problemu. Można na nim porównać poszczególne algorytmy.



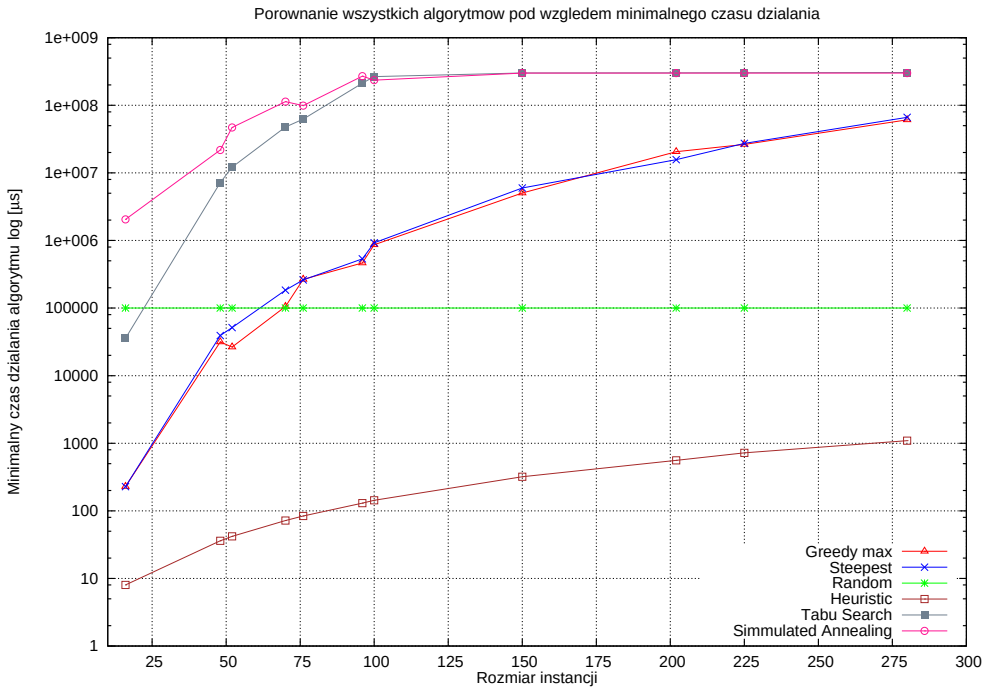
Rysunek 9: Porównanie wszystkich algorytmów pod względem średniego czasu działania

Spostrzeżenia:

- Czas działania wszystkich algorytmów (oprócz Random i Heuristic) rośnie w sposób wykładniczy wraz z rozmiarem instancji.
- Algorytm Steepest jest średnio nieco szybszy niż Greedy (szybciej znajduje jakieś minimum lokalne; szczególnie dla większych instancji). Minimalne czasy Greedy ma nieco mniejsze niż Steepest, lecz czasy maksymalne znacząco odstają.

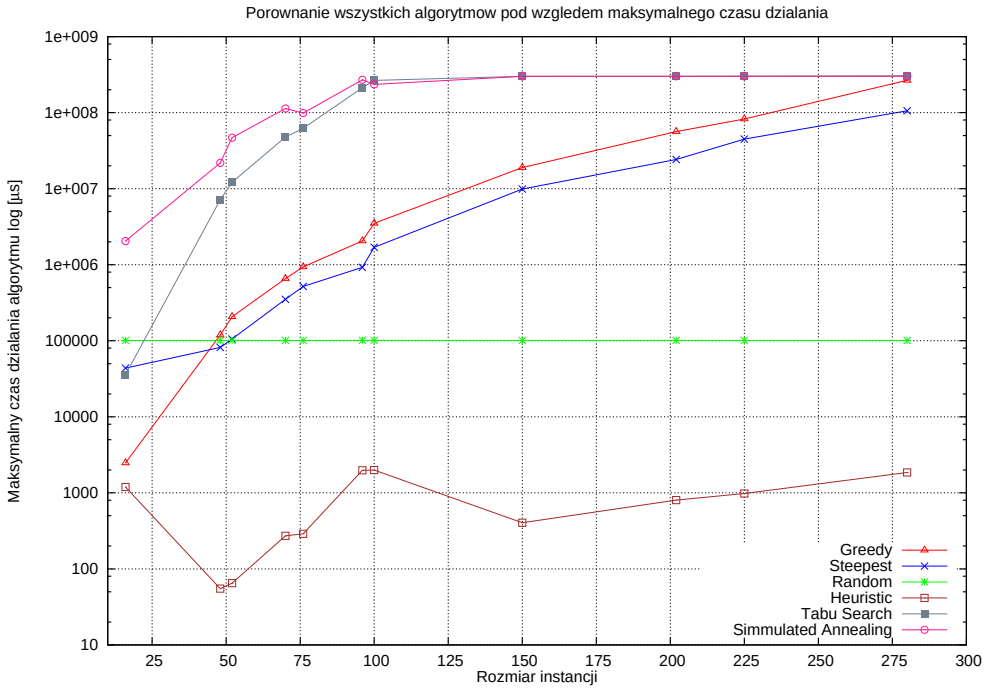


Rysunek 10: Porównanie wszystkich algorytmów pod względem średniego czasu działania (odchylenia standardowe)

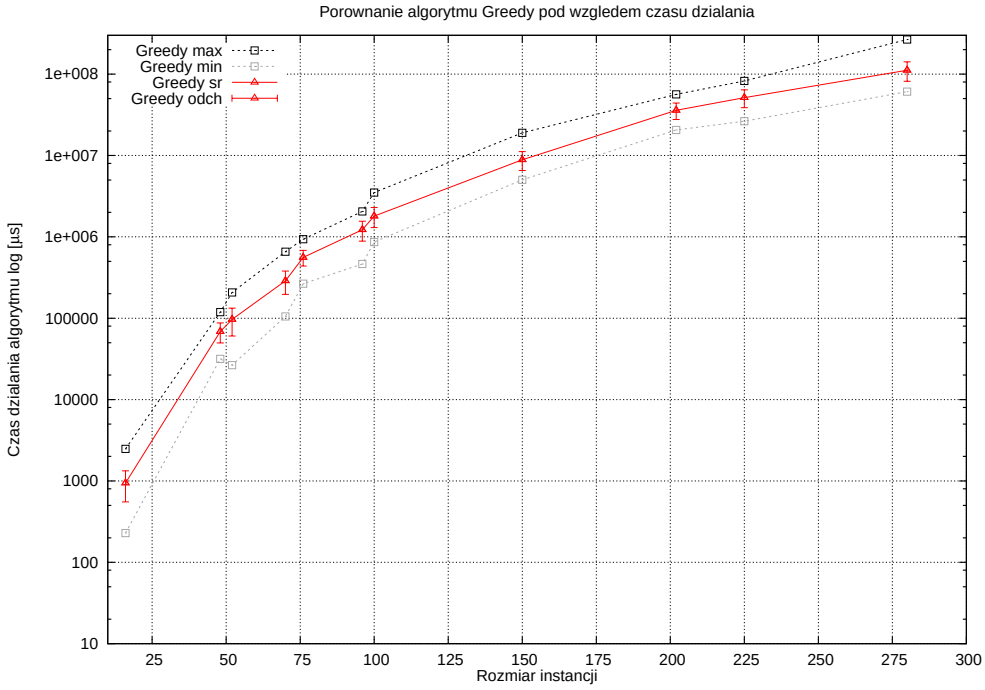


Rysunek 11: Porównanie wszystkich algorytmów pod względem minimalnego czasu działania

- Algorytm Heuristic jest co do wartości średniej najszybszym algorytmem na wszystkich instancjach.
- Najmniej stabilny czas działania ma Greedy.

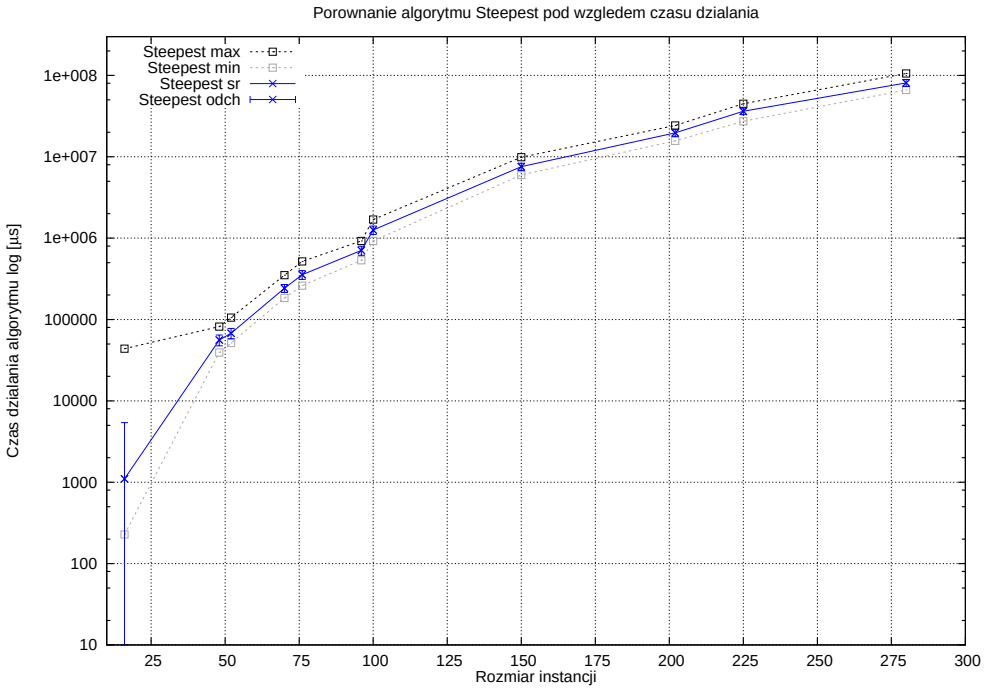


Rysunek 12: Porównanie wszystkich algorytmów pod względem maksymalnego czasu działania

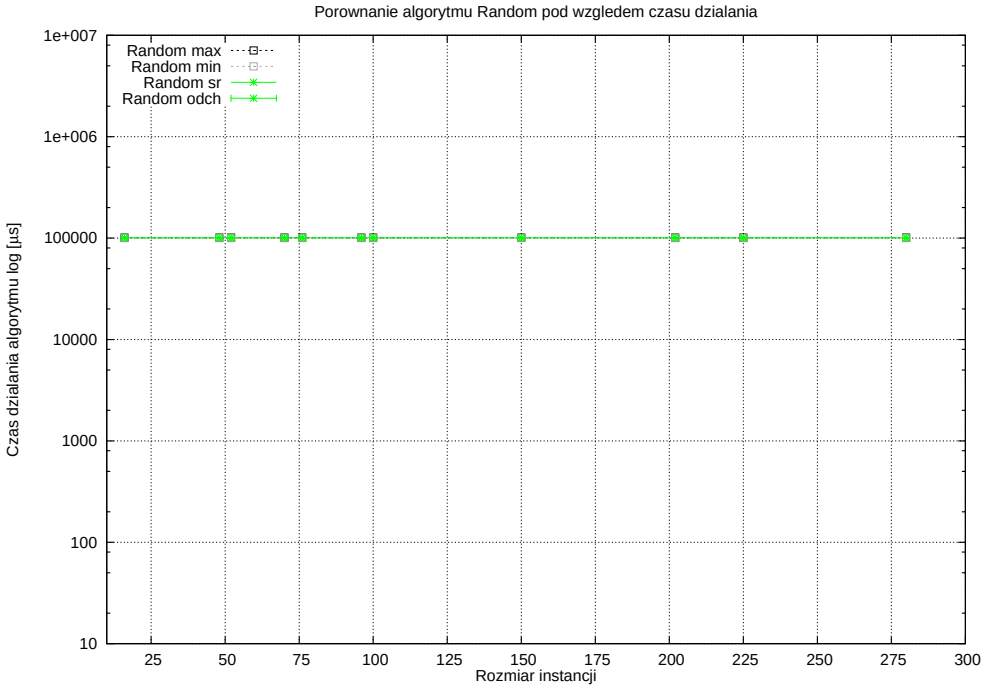


Rysunek 13: Wyniki algorytmu Greedy dla czasów działania

- Maksymalny czas działania algorytmu Heuristic dla niewielkich instancji wydaje się być nieco zaburzony (znacznie wyższy od średniego).
- Tabu Search i Simulated Annealing dla mniejszych instancji (n = 0 - 150) zatrzymują się wcześniej niż po maksymalnym czasie. Natomiast dla większych instancji ogranicza je maksymalny czas.

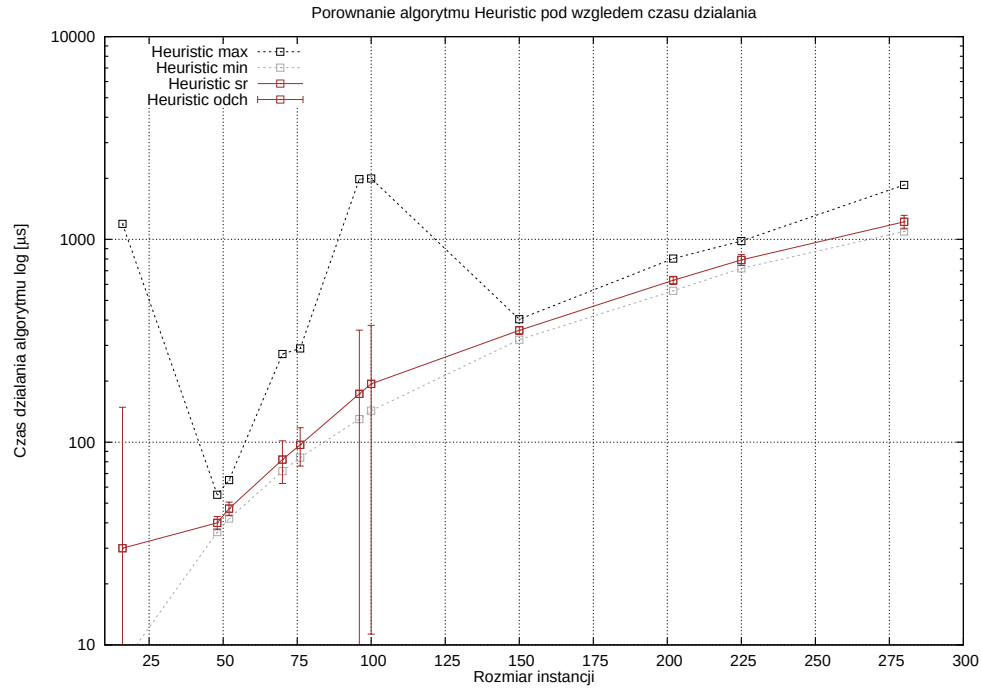


Rysunek 14: Wyniki algorytmu Steepest dla czasów działania

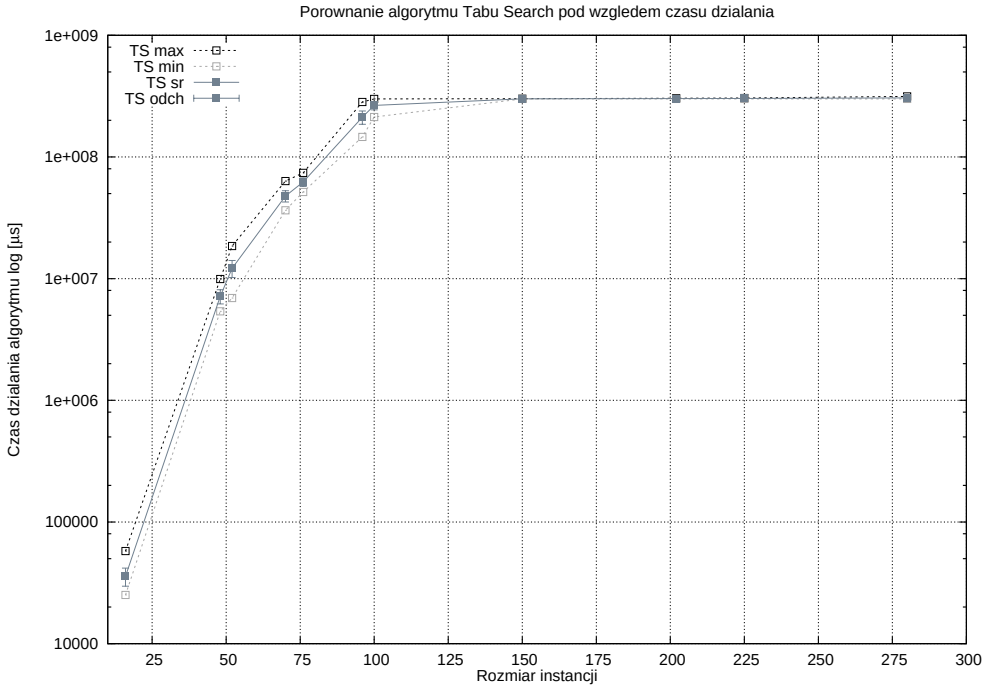


Rysunek 15: Wyniki algorytmu Random dla czasów działania

- Algorytmy przeszukiwania globalnego wykonują się zdecydowanie dłużej. Wynika to z kilku czynników:
  - Nie zadowalają one minimum lokalnego i pomimo jego znalezienia kontynuują poszukiwania.
  - Poprzez dopuszczenie możliwości wykonywania ruchów pogarszających dotychczas zna-

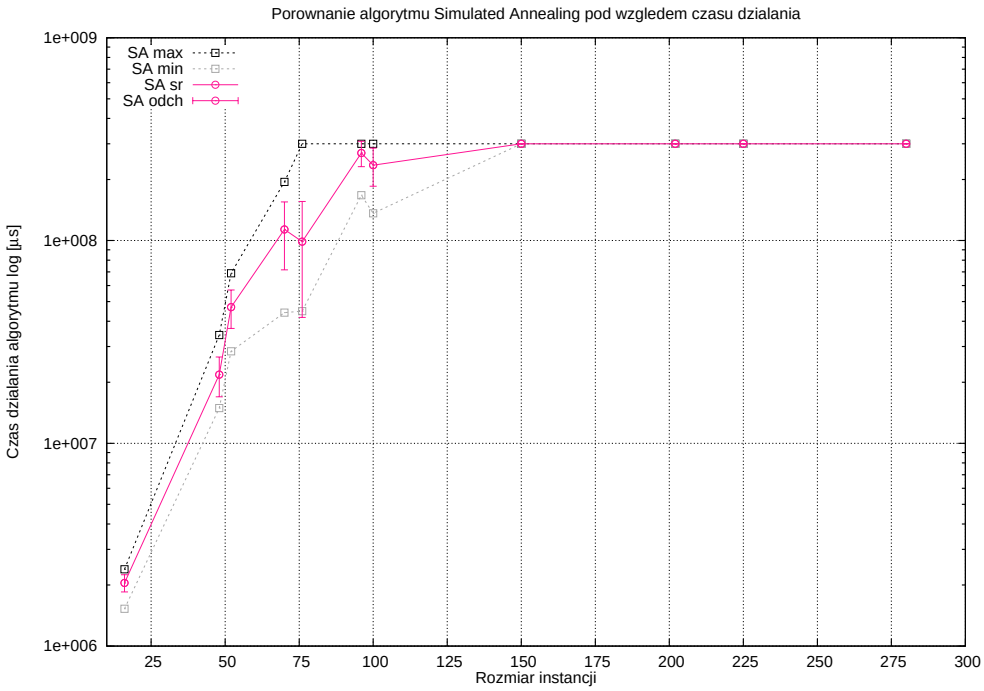


Rysunek 16: Wyniki algorytmu Heuristic dla czasów działania



Rysunek 17: Wyniki algorytmu Tabu Search dla czasów działania

lezione rozwiązanie, zwiększa się liczba sprawdzeń ocen rozwiązań sąsiednich.



Rysunek 18: Wyniki algorytmu Simulated Annealing dla czasów działania

Efektywność

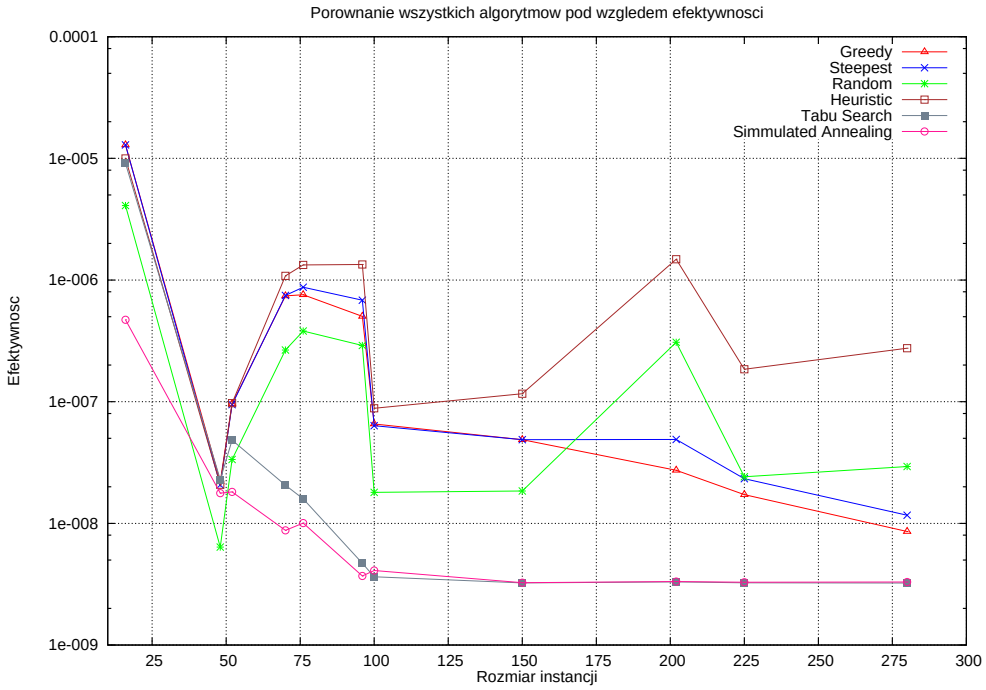
Jakość algorytmu jest proporcjonalna do jakości znajdowanych rozwiazań i odwrotnie proporcjonalna do czasu w jakim dokonują się obliczenia. Ponieważ najlepsza jakość jest wyznaczana przez małą odległość od optimum L_R, można zapisać (T_R jest to czas, w którym zostało znalezione rozwiazanie R). Podstawową jednostką czasu pomiarów jest mikrosekunda. Uznano, że różnicę w efektywności robią dopiero pojedyncze milisekundy, więc wartość czasu mnoży się przez 1000. W ten sposób rzeczywisty wpływ na ocenę mają milisekundy i sekundy, a odległość od optimum zyskuje większą wagę:

$$Q \approx \frac{1}{L_R + 1000 \cdot T_R} \tag{4}$$

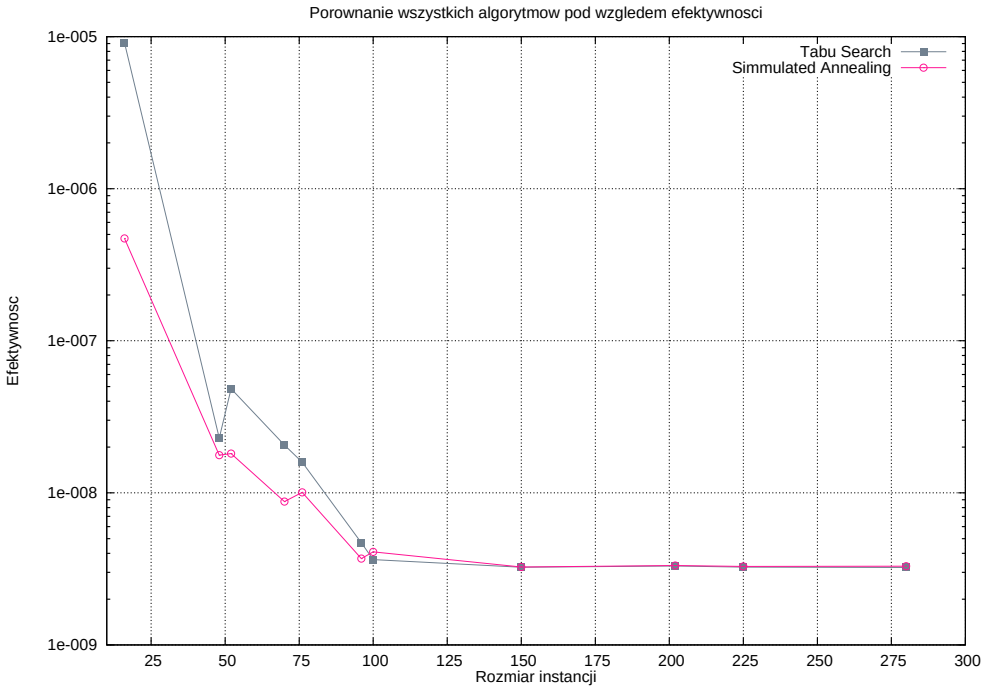
Problemem jest jednak to, że nie są to dokładne zależności, ale wykazują charakter losowy, do tego nieliniowy. Wykresy przedstawiające tą zależność przestawione są na rysunku 19.

Wnioski:

- Najefektywniejszym algorytmem okazał się Heuristic
- Algorytmy Greedy i Steepest osiągają dobre wyniki na mniejszych instancjach, wraz ze wzrostem rozmiaru problemów czas działania algorytmów znacznie się wydłuża
- Dla większych instancji sprawdza się algorytm Random, który czasem działania nadrabia słabą jakość wyników
- Algorytmy przeszukiwania globalnego ze względu na czas działania są mniej efektywne od algorytmów lokalnych.



Rysunek 19: Porównanie wszystkich algorytmów pod względem efektywności

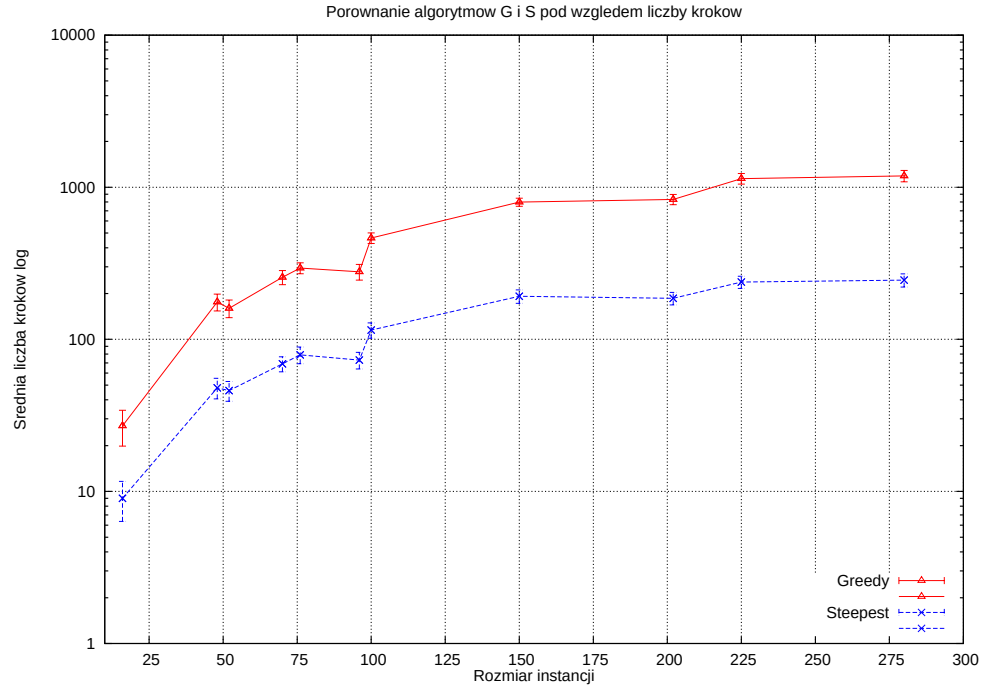


Rysunek 20: Porównanie wszystkich algorytmów pod względem efektywności

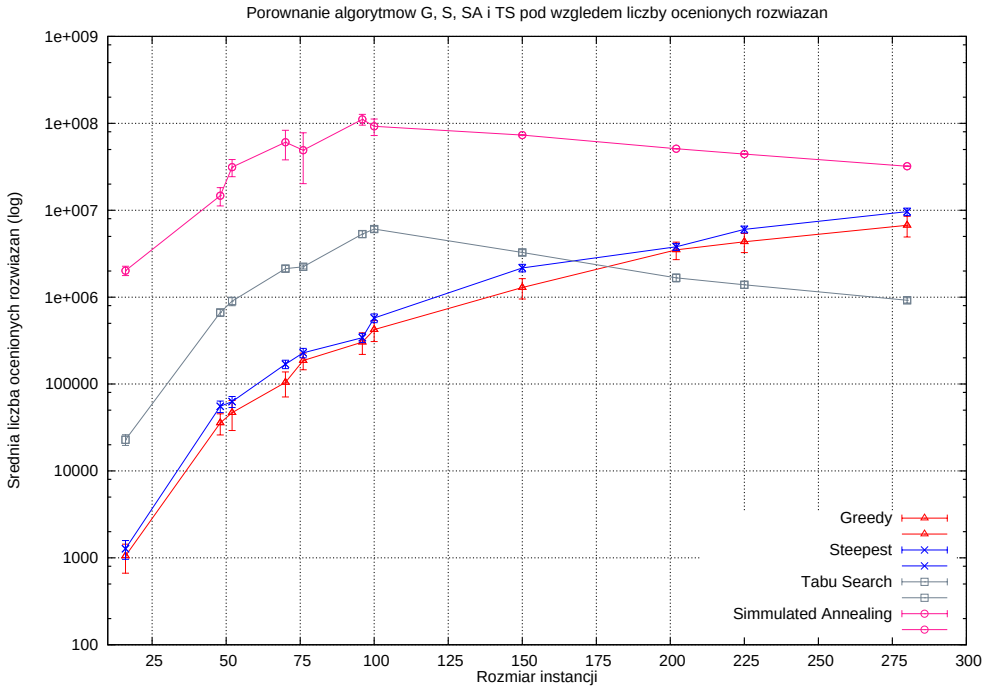
Liczba przejrzanych rozwiązań i liczba kroków (Greedy i Steepest)

Aby lepiej zrozumieć wyniki uzyskane przez algorytmy, zliczono i uśredniono liczbę wykonanych kroków i ocenionych rozwiązań. Mimo wszystko różnice między Greedy a Steepest nie są duże. Z osiągniętych wyników algorytmów wynika, że Steepest przegląda 1,5 – 3 razy więcej rozwiązań niż Greedy (co stanowi nawet 10 000 000 przejrzanych rozwiązań dla największych instancji), co potwierdza rysunek 22. Z rysunku 21 wynika z kolei, że algorytm Greedy wykonuje 2 – 6 razy więcej kroków niż Steepest(łacznie poniżej 1100 kroków). Porównanie między liczbą kroków a ich jakością (pierwszy lepszy sąsiad, czy najlepszy) wydaje się nie przeważać w żadną ze stron.

Algorytmy przeszukiwania globalnego sprawdzają zdecydowanie większe sąsiedztwo. Szczególnie jest to widoczne dla małych instancji, dla których SA przeszukuje o kilka rzędów wielkosci więcej rozwiązań. Zaskakujące jest, że od pewnego rozmiaru instancji, TS sprawdza najmniej rozwiązań z pośród wszystkich pozostałych algorytmów.



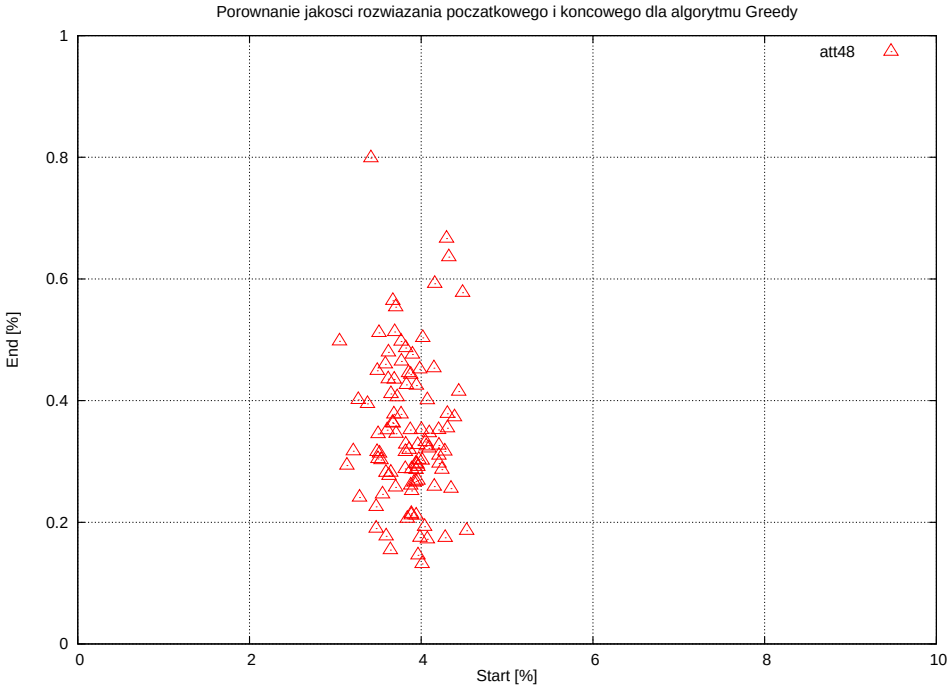
Rysunek 21: Średnia ilość kroków dla algorytmów Greedy i Steepest



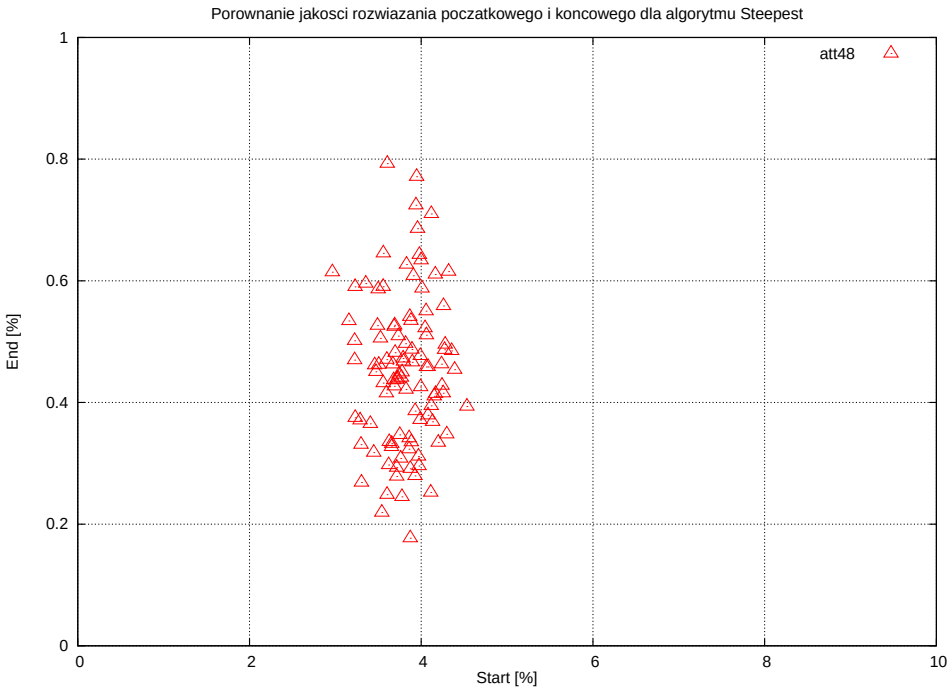
Rysunek 22: Średnia ilość przejrzanych rozwiązań dla algorytmów Greedy, Steepest, Tabu Search i Simulated Annealing

Wpływ jakości rozwiązania początkowego na jakość rozwiązania końcowego

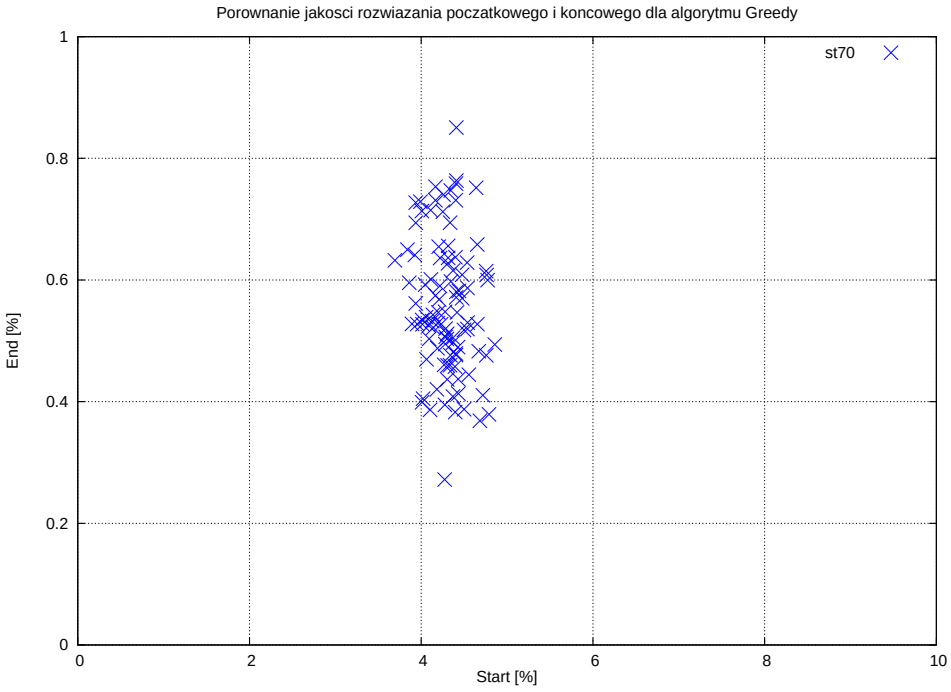
Na wykresach 23-28 przedstawione są wyniki po stu iteracjach algorytmów Steepest i Greedy dla problemów att48, st70, gr96. Otrzymaliśmy w wyniku skupienie o owalnym kształcie, z czego nie wynika, żeby istniała jakaś zależność między jakością rozwiązania początkowego, a jakością końcowego. Startowe rozwiązania są dość mocno skupione, tzn. mają podobną odległość od optimum. Było to spowodowane tym, że każde uruchomienie algorytmu było dla losowego rozwiązania startowego. Możnaby w tym miejscu próbować stopniowo polepszać rozwiązania początkowe, żeby mieć większą różnorodność na osi jakości rozwiązania początkowego.



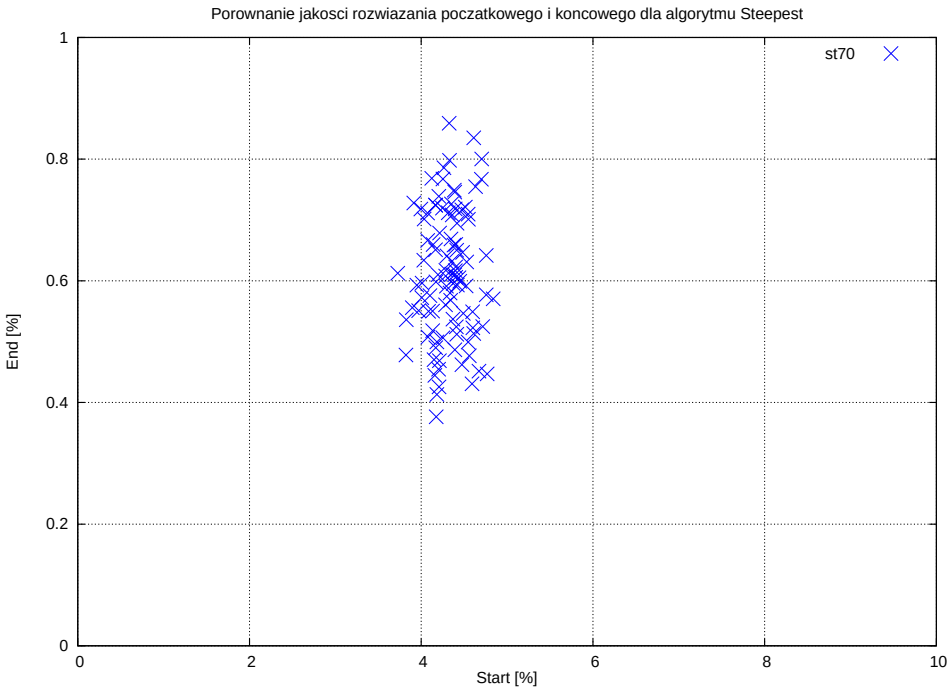
Rysunek 23: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Greedy (100 powtórzeń) dla instancji att48



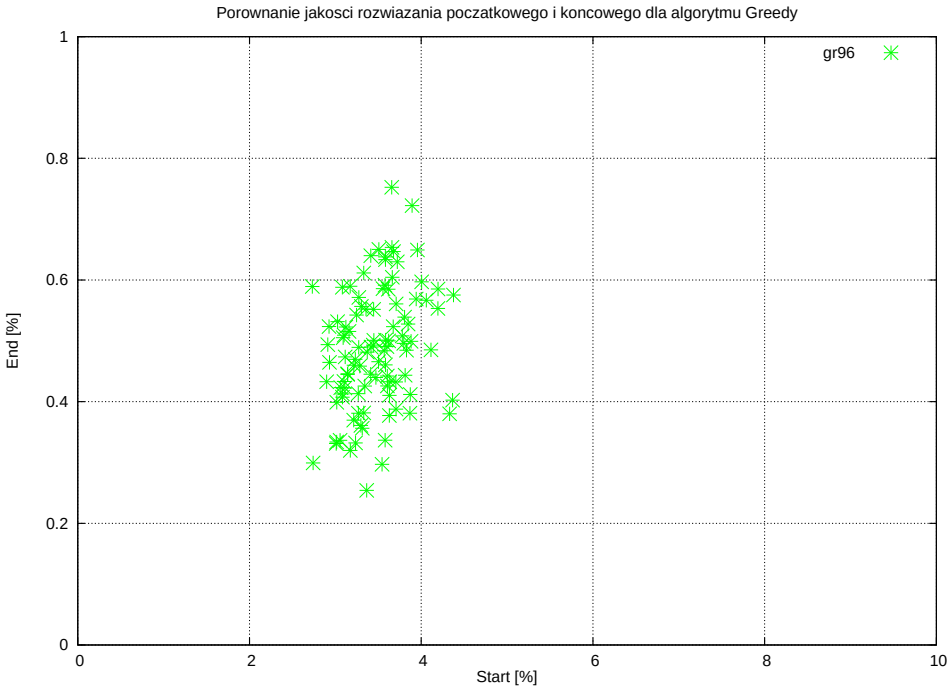
Rysunek 24: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Steepest (100 powtórzeń) dla instancji att48



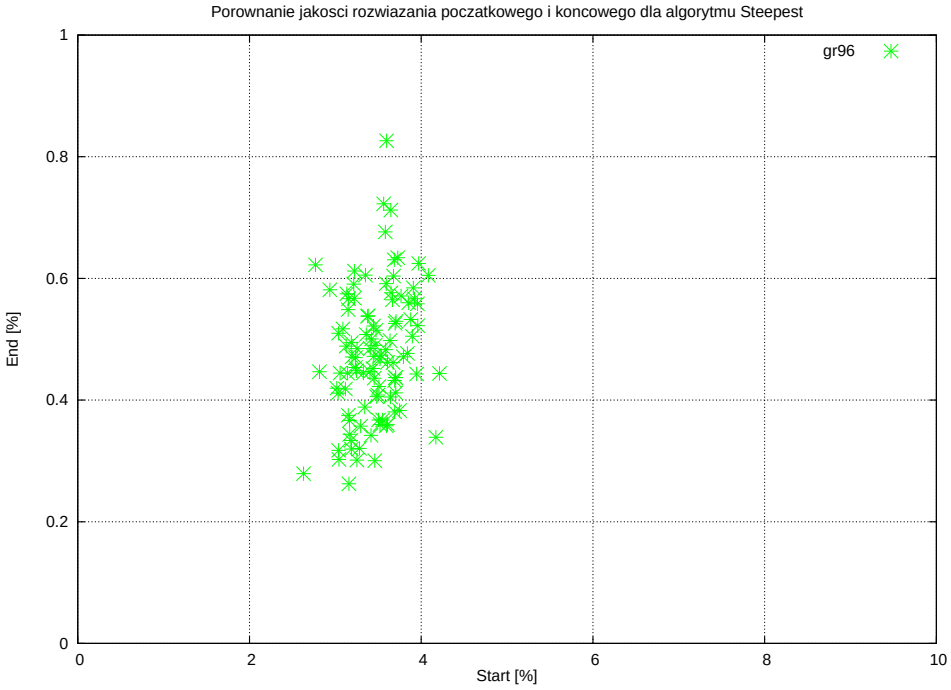
Rysunek 25: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Greedy (100 powtórzeń) dla instancji st70



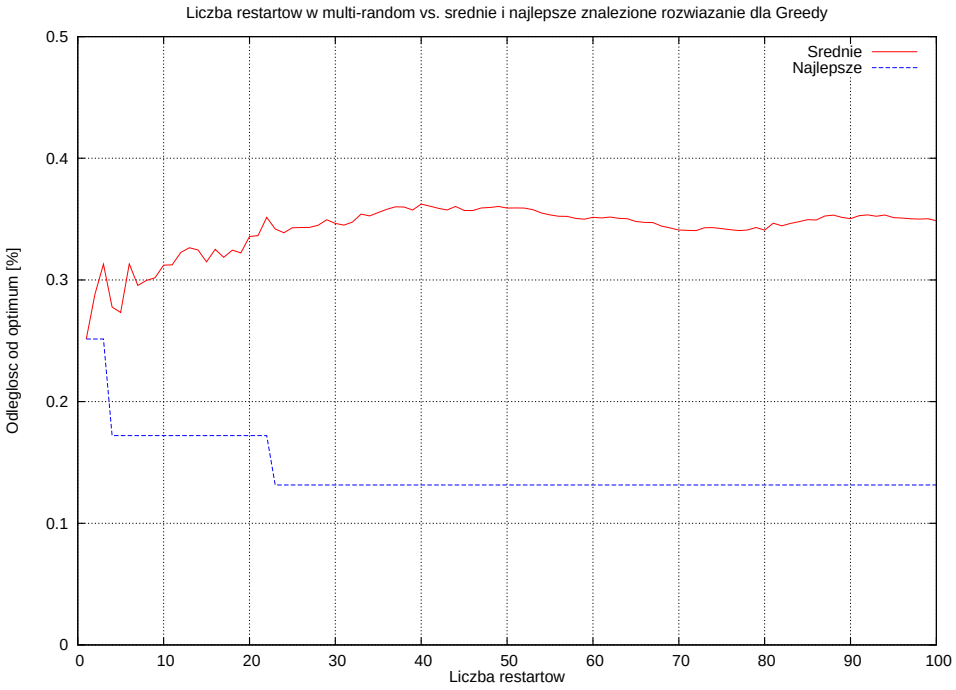
Rysunek 26: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Steepest (100 powtórzeń) dla instancji st70



Rysunek 27: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Greedy (100 powtórzeń) dla instancji gr96



Rysunek 28: Odległość od rozwiązania początkowego vs. jakość rozwiązania końcowego Steepest (100 powtórzeń) dla instancji gr96



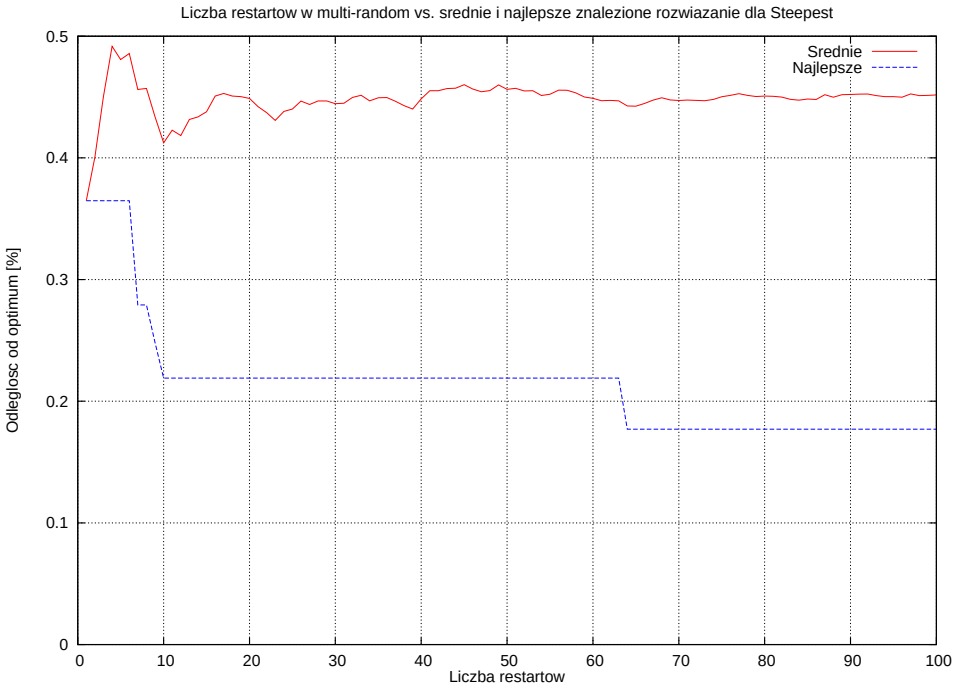
Rysunek 29: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Greedy dla instancji att48

Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie

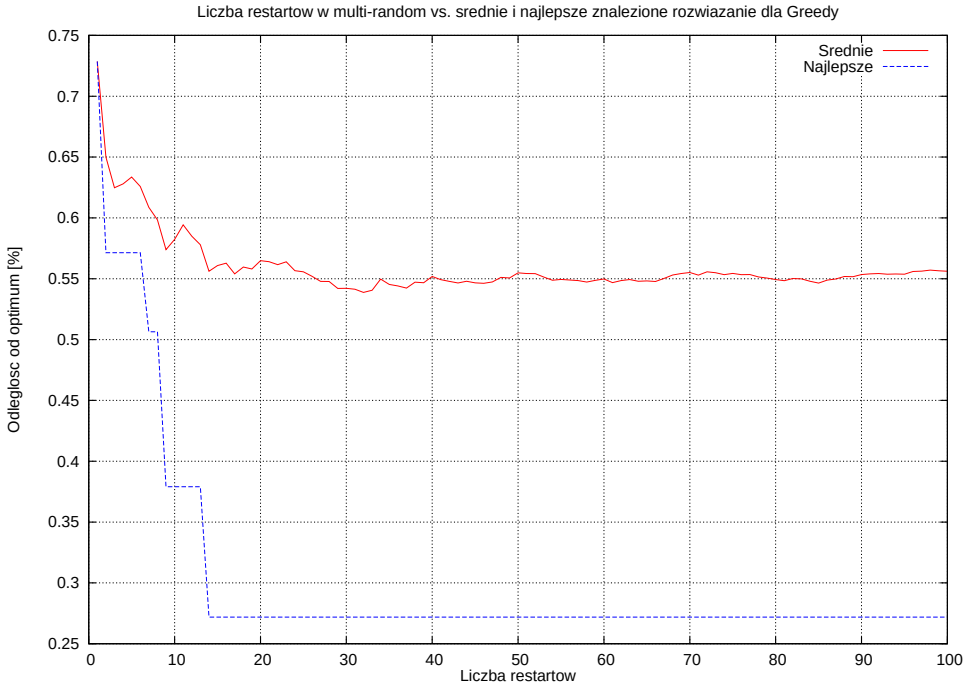
Na wykresach 29-34 przedstawiona jest jakość średniego i najlepszego rozwiązania w funkcji liczby iteracji. Jakość rozwiązania przedstawiona jest, tak jak w przypadku poprzednich wykresów, w formie procentowej odległości od optimum. Każda iteracja była rozpoczynana od losowego rozwiązania.

Wnioski:

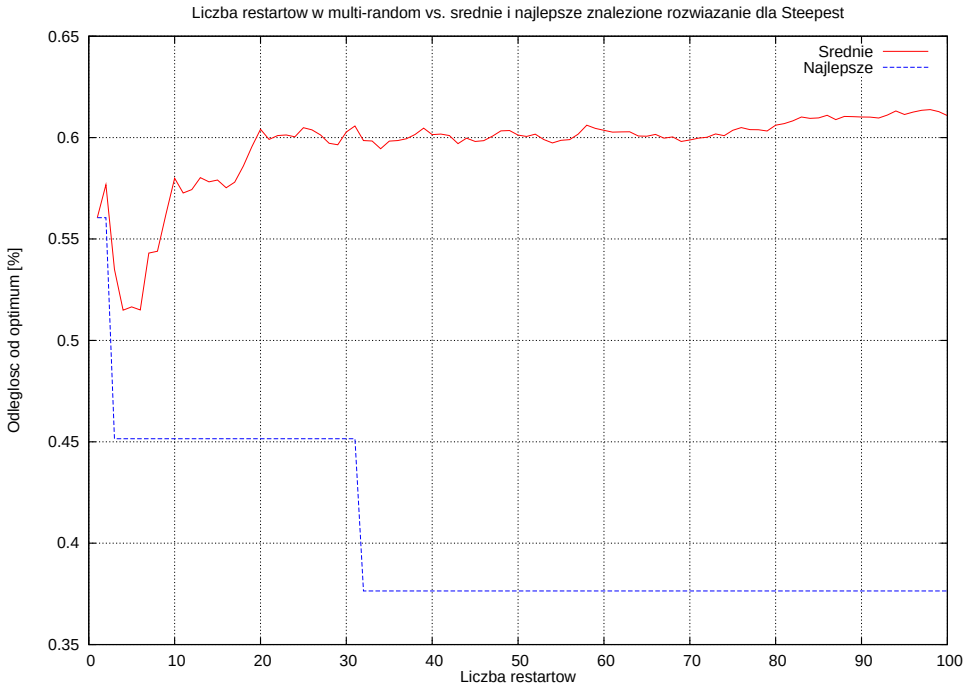
- Jakość średniego rozwiązania po wielu iteracjach obu algorytmów dochodzi do pewnego progu, dla którego kolejne uruchamianie algorytmów nie polepsza go znacząco
- Dopóki nie znajdziemy rozwiązania optymalnego, istnieje niezerowe prawdopodobieństwo, że znajdziemy lepsze rozwiązanie
- Dla ciągu uruchomień od raz do 100 razy algorytmu Greedy dla poszczególnych instancji problemów możemy określić graniczną liczbę uruchomień po której nie znajdujemy lepszego rozwiązania
- Podobnie możemy uczynić dla algorytmu Steepest
- Graniczna liczba uruchomień dla żadnego z algorytmów nie jest zależna od rozmiaru instancji, ale od samej instancji problemu



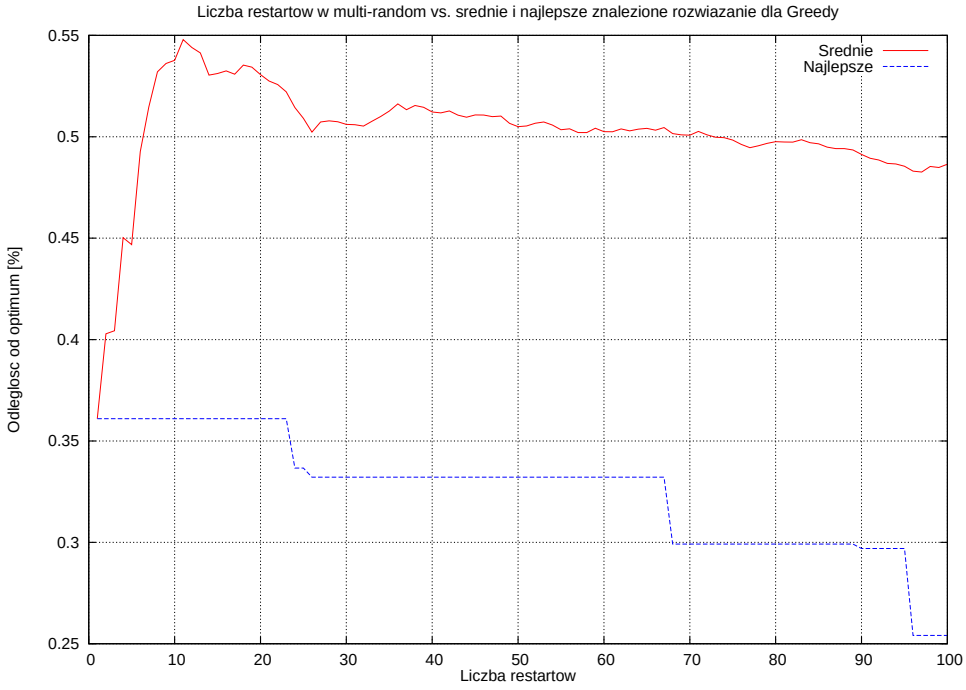
Rysunek 30: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Steepest dla instancji att48



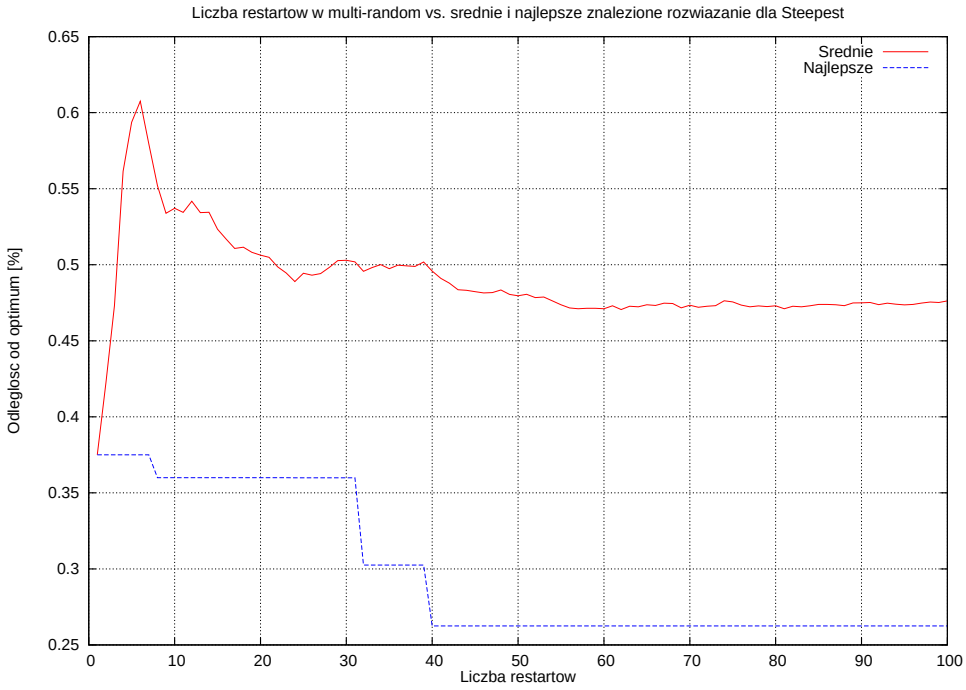
Rysunek 31: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Greedy dla instancji st70



Rysunek 32: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Steepest dla instancji st70



Rysunek 33: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Greedy dla instancji gr96



Rysunek 34: Liczba restartów w multi-random vs. średnie i najlepsze znalezione rozwiązanie Steepest dla instancji gr96

4. Wnioski

- Nieopłacalne jest startowanie algorytmów Greedy i Steepest z losowego miejsca. Zatrzymują się one wtedy najczęściej w mało atrakcyjnym optimum lokalnym.
- Nieopłacalne jest również wykorzystanie algorytmu losowego, nawet jeśli damy mu bardzo dużo czasu. Rozwiązania znajdowane przez ten algorytm podczas testów nigdy nie znalazło sie w poblizu optimum.
- Biorąc pod uwagę, że algorytmy przeszukiwania lokalnego zdecydowanie lepiej działają startując z bardzo dobrego rozwiązania, warto byłoby zastosowić sią nad jakąś bardziej zaawansowaną heurystyką.
- Algorytm Steepest i Greedy maja zbliżone wyniki, jednak Steepest jest trochę szybszy i stabilniejszy (patrz: Rysunek 11).
- Algorytm TS radzi sobie zdecydowanie lepiej (znajduje lepsze rozwiązania) od algorytmów przeszukiwania lokalnego.
- Algorytm SA daje najlepsze rezultaty. Okupione jest to niestety długim czasem działania i koniecznoscią ustalenia wielu parametrów. Parametry te, jak np. temperatura początkowa, musiały być indywidualnie dobrane do każdej instancji problemu.
- Głównym kryterium oceny algorytmu optymalizacji powinna być jakość znajdowanych rozwiazań w przypadku średnim. Bardzo ważne jest także odchylenie od tej średniej, a ścislej działanie algorytmu w przypadkach pesymistycznych. Nie bez znaczenia pozostaje czas działania algorytmu, a także czy, dając algorytmowi więcej czasu, możemy liczyć na poprawę rozwiazania. W przypadku TS i SA jest pewność, że więcej czasu to lepsze rozwiązanie.

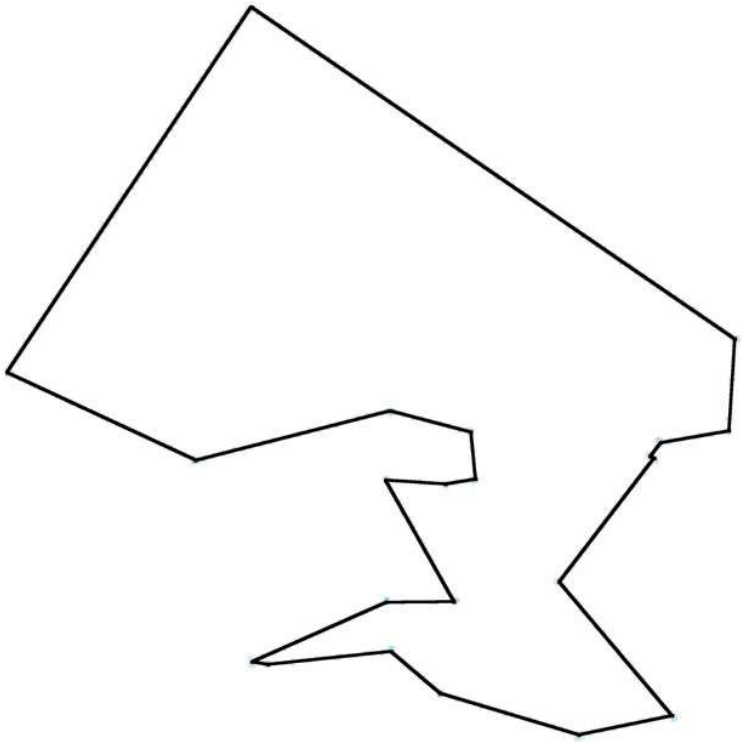
Wizualizacje rozwiązań są przedstawione na rysunkach 35 - 39.

5. Możliwe ulepszenia

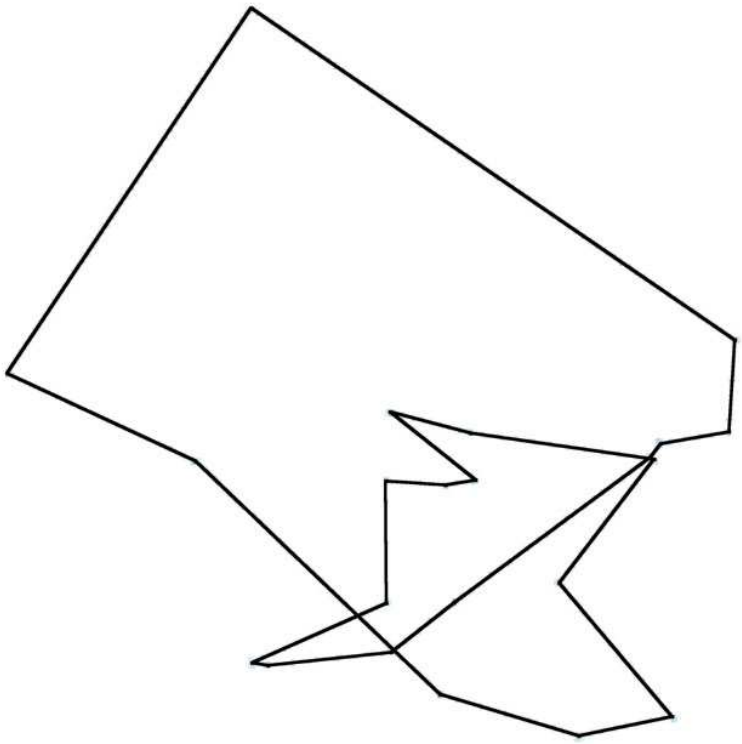
Zaproponowano następujące ulepszenia, potencjalnie poprawiające wyniki:

- W przypadku algorytmów G i S nie liczyć sumy odległości dla każdego miasta, a jedynie różnicę między danym rozwiązaniem a rozwiązaniem sąsiednim w którym wskutek zamiany miejscami dwóch miast zmieniają się wartości czterech odległości. Mogłoby to skrócić czas działania tych algorytmów.
- Można sprawdzić jak zachowają się algorytmy G i S, gdy będą startować nie od losowego rozwiązania, ale od uzyskanego przez heurystykę.
- Można sprawdzić jak wpłynie zmiana sąsiedztwa na inne (np. na sąsiedztwo w którym sąsiada uzyskuje się przez wyciągnięcie jednego miasta z listy i wstawienie go w innym miejscu). Dodatkowo ciekawym eksperymentem mogłoby być sprawdzenie jak wpłynie zmiana sąsiedztwa w trakcie obliczeń (np gdy Greedy, lub Steepest dojdą do optimum lokalnego).
- Można, dla algorytmu Greedy, celem szybszego osiągnięcia rozwiązań o lepszej ocenie, ustalić minimalne polepszenie, o które permutacja musi być lepsza od aktualnej. To polepszenie może się stopniowo zmniejszać, lub przestanie być brane pod uwagę gdy żaden sąsiad go nie spełni.
- Dla algorytmu SA można by zrobić testy jak wpływa dobór parametrów działania (temperatura początkowa, współczynnik spadku temperatury) na wynik.
- Algorytm TS również mozna by sprawdzić pod kątem doboru parametrów (rozmiar listy tabu, rozmiar listy Masters List).

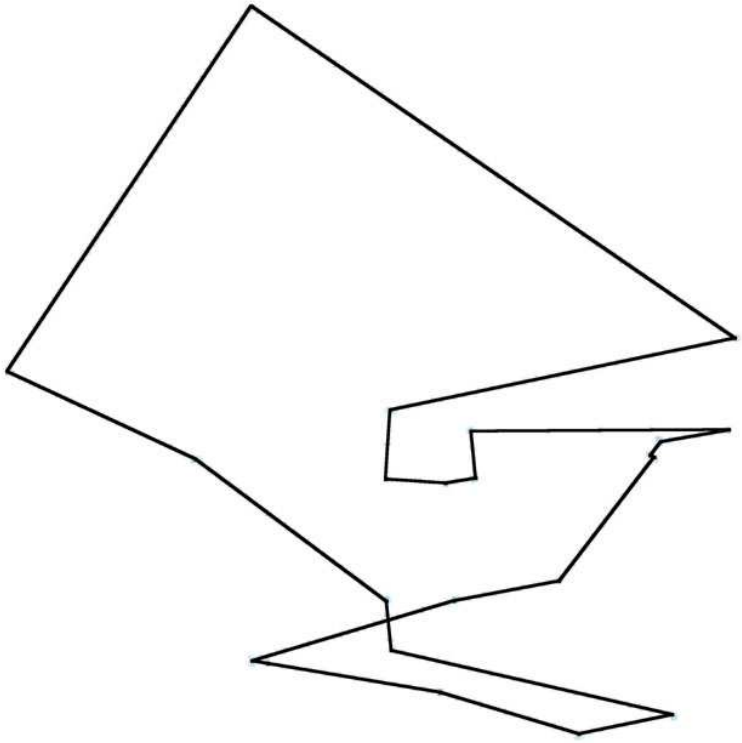
6. Pozostałe wykresy



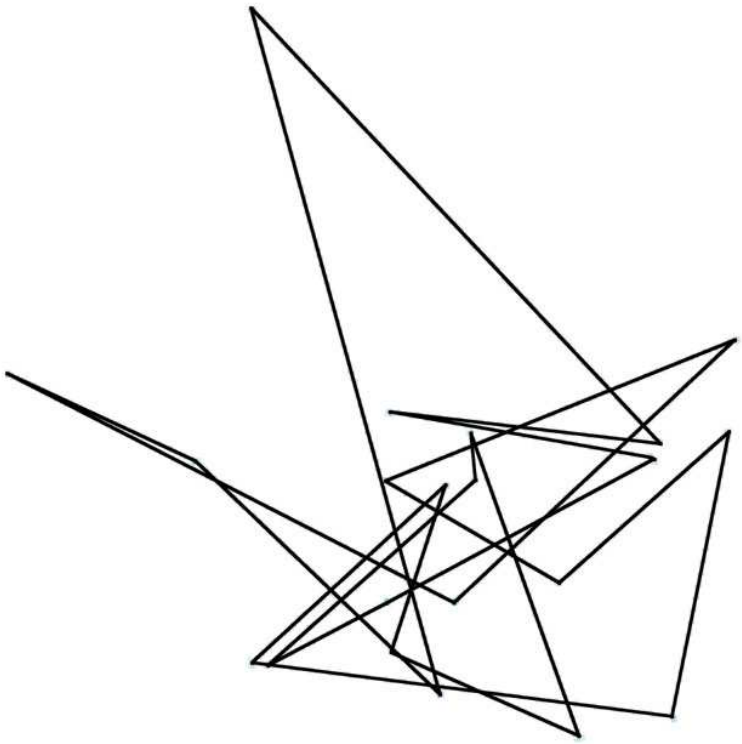
Rysunek 35: Wizualizacja rozwiązania optymalnego dla problemu ulysses22



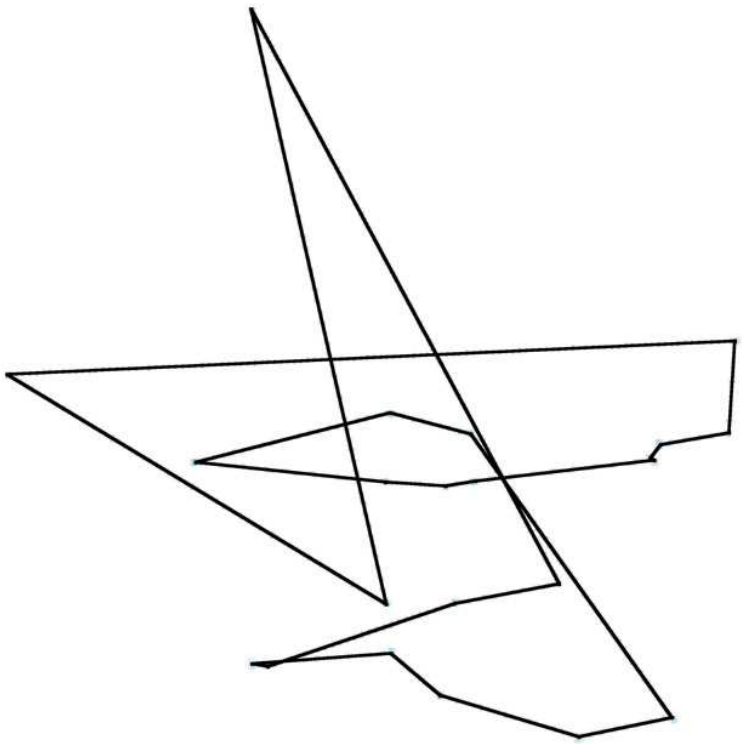
Rysunek 36: Wizualizacja rozwiązania Greedy dla problemu ulysses22



Rysunek 37: Wizualizacja rozwiązania Steepest dla problemu ulysses22



Rysunek 38: Wizualizacja rozwiązania Random dla problemu ulysses22



Rysunek 39: Wizualizacja rozwiązania Heuristic dla problemu ulysses22