Sprawozdanie z laboratorium:
Metaheurystyki i Obliczenia Inspirowane Biologicznie

Część I: Algorytmy optymalizacji lokalnej, problem ATSP

25 listopada 2013

Prowadzący: dr hab. inż. Maciej Komosiński

Autorzy:	**Dawid Wiśniewski**	inf94387	ISWD	wisniewski.dawid@gmail.com
	Paweł Rychły	inf94362	ISWD	pawelrychly@gmail.com

Zajęcia poniedziałkowe, 15:10.

1 Wstęp

Problem komiwojażera opisywany jest często jako problem wędrownego sprzedawcy zamierzającego odwiedzić pewien zbiór miast. Planując swoją podróż, usiłuje on znaleźć możliwie najkrótszą trasę, która kończyłaby się w punkcie startowym. Zakłada się, że każde miasto z wyjątkiem początkowego powinno zostać odwiedzone tylko jeden raz. Przyjmuje się również, że każda para miast połączona jest drogą o określonej długości. Opis ten stanowi jedynie ilustrację ogólniejszego zagadnienia. Zapisując Problem Komiwojażera w języku teorii grafów, można zdefiniować go jako problem znajdowania takiego cyklu Hamiltona w pełnym grafie ważonym, dla którego suma wag odwiedzonych krawędzi jest minimalna. W tak zdefiniowanym problemie można wyróżnić Asymetryczny problem Komiwojażera (ATSP). Zakłada się w nim, że odległość pomiędzy dwoma miastami A i B może być różna w zależności od tego, czy sprzedawca przemieszcza się z punktu A do B, czy też w kierunku przeciwnym. Zarówno symetryczna jak i asymetryczna wersja problemu komiwojażera znalazła bardzo szerokie zastosowanie w praktyce. Jej wykorzystanie nie ogranicza się jedynie do problemów związanych z transportem. Problem komiwojażera stosowany jest między innymi w produkcji elektroniki, gdzie optymalizuje się drogę lasera wypalającego obwody elektroniczne. W sieciach komputerowych zastosowany jest do optymalizacji tras routingu. Niestety opisywany problem jest problemem o wykładniczej złożoności obliczeniowej i należy do klasy problemów NP-trudnych.

2 Operator sąsiedztwa

W zaimplementowanych przez nas algorytmach wykorzystaliśmy operator sąsiedztwa 2-opt. W podejściu tym, każda permutacja oznaczająca kolejność odwiedzanych wierzchołków w grafie sąsiaduje z permutacjami utworzonymi poprzez zamianę miejscami dwóch liczb w sekwencji. Rozmiar sąsiedztwa (n), można obliczyć za pomocą wzoru:

$$n = \frac{l * (l-1)}{2}$$

gdzie l – liczba wierzchołków

3 Porównanie działania 8 algorytmów i rodzajów sąsiedztw na wszystkich instancjach problemów

w każdym z wykresów przyjęliśmy podobną konwencję nazewnictwa algorytmów występującą w legendach do nich. Oto objaśnienie:

greedy – algorytm wybierający dane rozwiązanie przegladając zadane sąsiedztwo wtedy, kiedy napotka lepsze od aktualnego

random-greedy – algorytm w pełni losowy (rzucający losową permutację) działający tak długo jak greedy

nearest neighbour – prosta heurystyka dobudowująca najbliższe miasto do rozwiązania random-walker-steepest - algorytm losowo skaczący po sąsiedztwie o czasie działania takim jak steepest

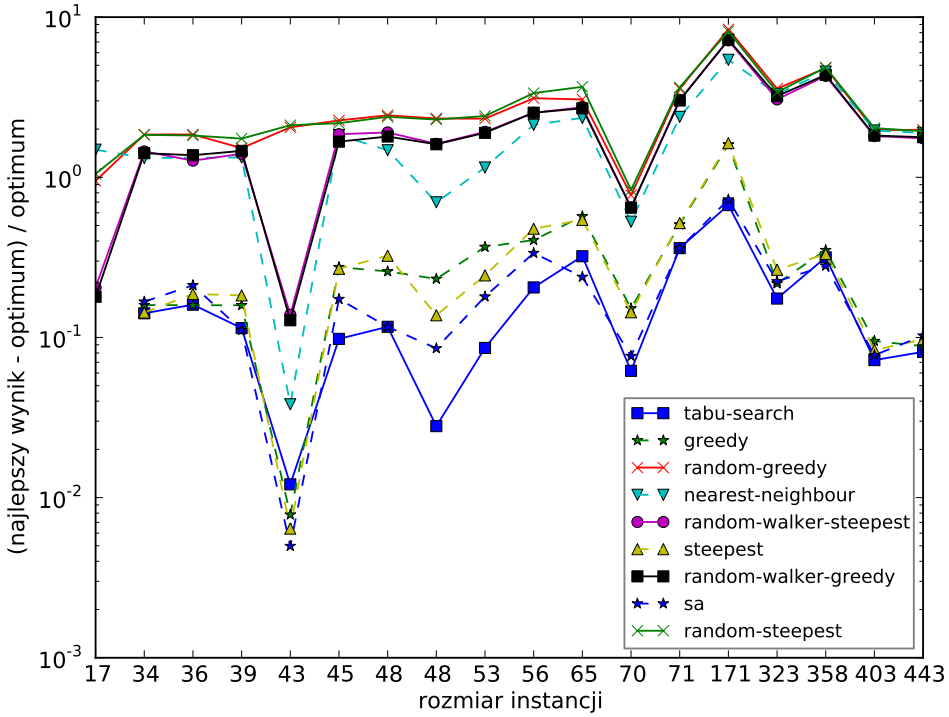
steepest – algorytm przeszukujący całe sąsiedztwo i wybierający najlepszego kandydata random-walker-greedy - algorytm losowo skaczący po sąsiedztwie o czasie działania takim jak greedy

random-steepest – algorytm losowy działający tak długo jak steepest

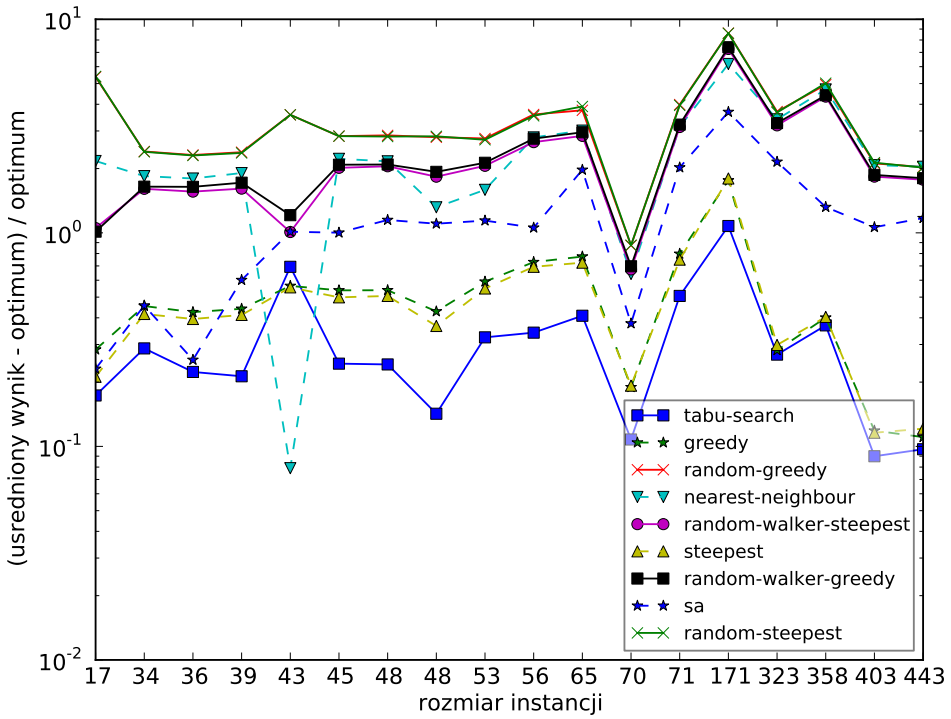
tabu-search – algorytm z zaimplementowanym przeszukiwaniem tabu

sa – algorytm symulowanego wyżarzania

3.1 Odległość od optimum.



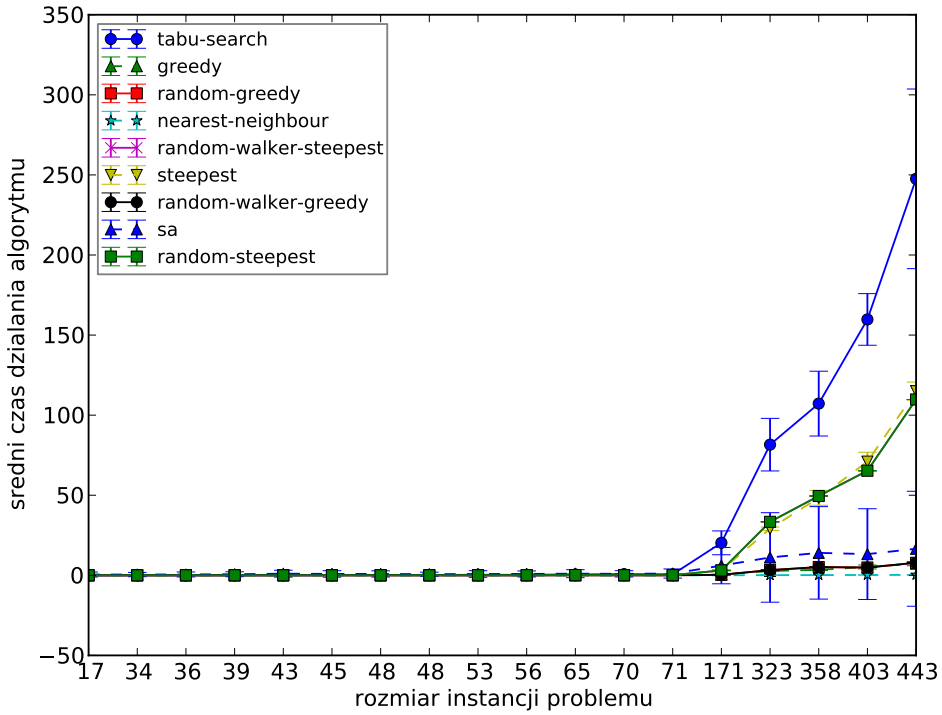
Rysunek 1: Odległości od optimum najlepszych znalezionych rozwiązań dla kilku przykładowych algorytmów.



Rysunek 2: Odległości od optimum wartości średnich ze znalezionych rozwiązań dla kilku przykładowych algorytmów.

Jak widać na załączonych wykresach w kontekście względnej odległości od optimum (liczonej jako odległość najlepszego wyniku od optimum podzielonego przez optimum) algorytmy greedy i steepest mają bardzo zbliżone rezultaty. Najlepsze wyniki osiągnął algorytm Tabu Search. Znalezione przez niego permutacje, okazały się najbliższe rozwiązaniu optymalnemu. Rezultaty osiągnięte przez algorytm Simulated annealing (SA), średnio były gorsze od tych znalezionych przez algorytmy Greedy i Steepest. Najlepsze rozwiązania znajdywane przez ten algorytm, były jednak porównywalne z algorytmem Tabu Search. Najgorszymi natomiast okazały się być algorytmy losowe, zwracające losową permutację, działające tak długo jak odpowiednio greedy i steepest. Widzimy więc, że proste heurystyki i algorytmy losowe wypadają znacznie gorzej niż podejścia greedy i steepest.

3.2 Porównanie czasu działania algorytmów.



Rysunek 3: Porównanie średnich czasów działania algorytmów.

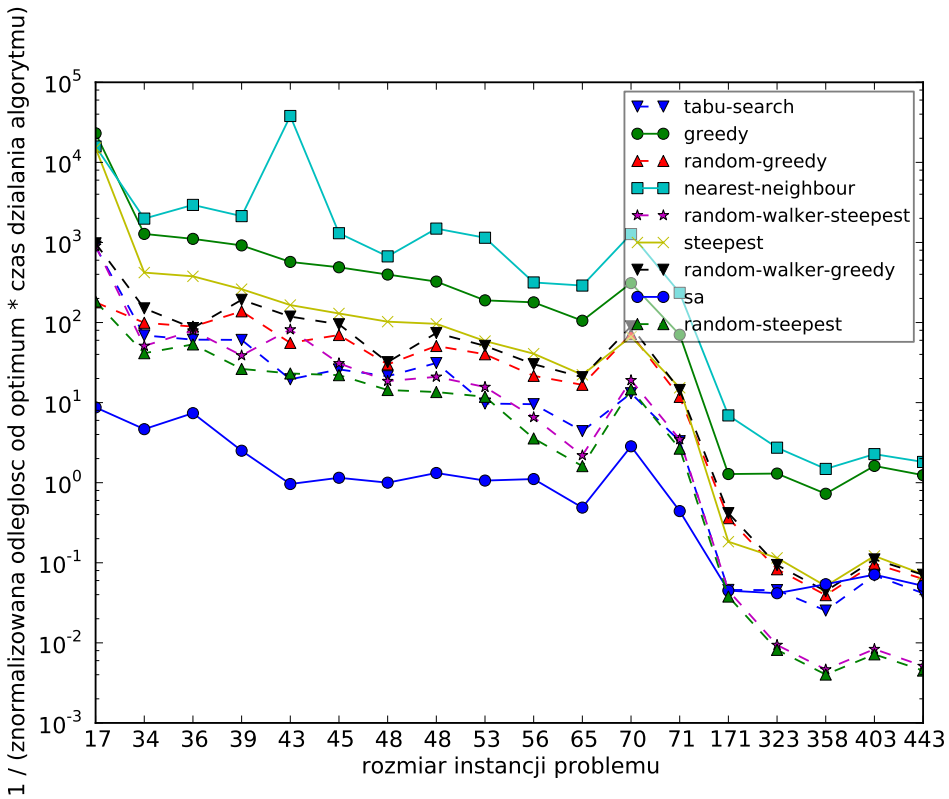
Jednakże dobre rezultaty obarczone są długimi czasami działania algorytmów Greedy, Steepest oraz Tabu Search. Wszystkie proste Heurystyki mają niewielki wpływ rozmiaru instancji na czas działania. Natomiast w przypadku wymienionych wcześniej algorytmów rozmiar odgrywa kluczową rolę w tej kwestii. Rozmiar instancji ma szczególnie duży wpływ na czas działania algorytmu Tabu Search. W tym przypadku, algorytm realizując każdy krok, musi przeszukać całą listę Tabu, której rozmiar również jest zależny od rozmiaru instancji (liczba elementów listy tabu jest równa liczbie wierzchołków podzielonej przez 4). Warto zwrócić uwagę na stosunkowo niewielkie czasy działania algorytmu SA.

3.3 Porównanie efektywności algorytmów.

W celu porównania efektywności działania algorytmów wprowadzono miarę:

$$Efektywnosc = \frac{1}{\frac{wynik-optimum}{optimum} * czas}$$

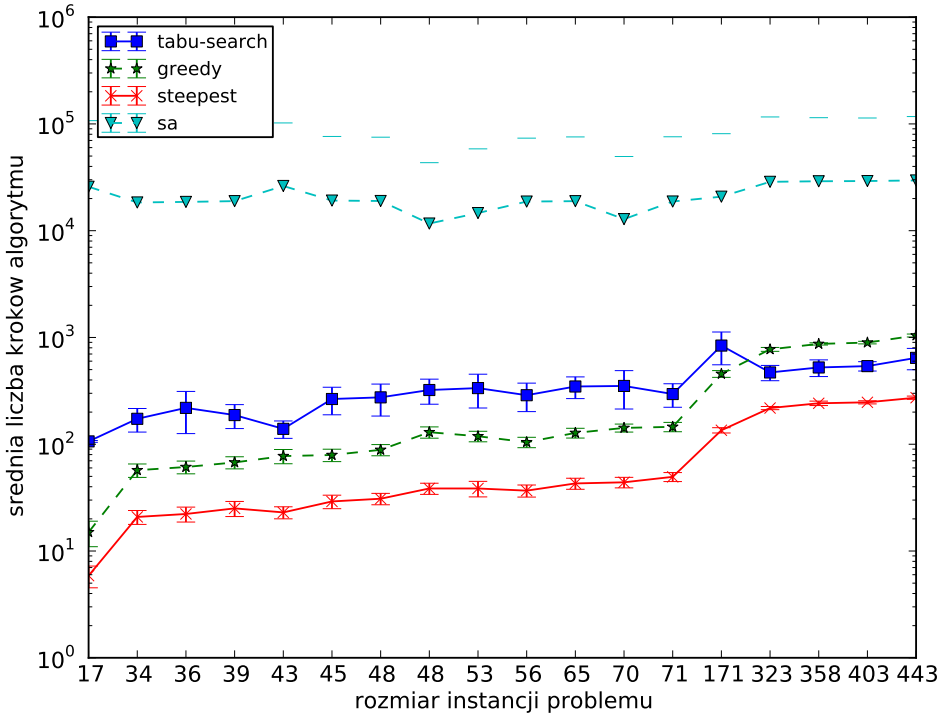
Została ona tak zaprojektowana aby efektywność była tym większa im mniejsza jest średnia odległość otrzymanych wyników od optimum oraz im krótszy jest czas działania algorytmu.



Rysunek 4: Porównanie efektywności algorytmów.

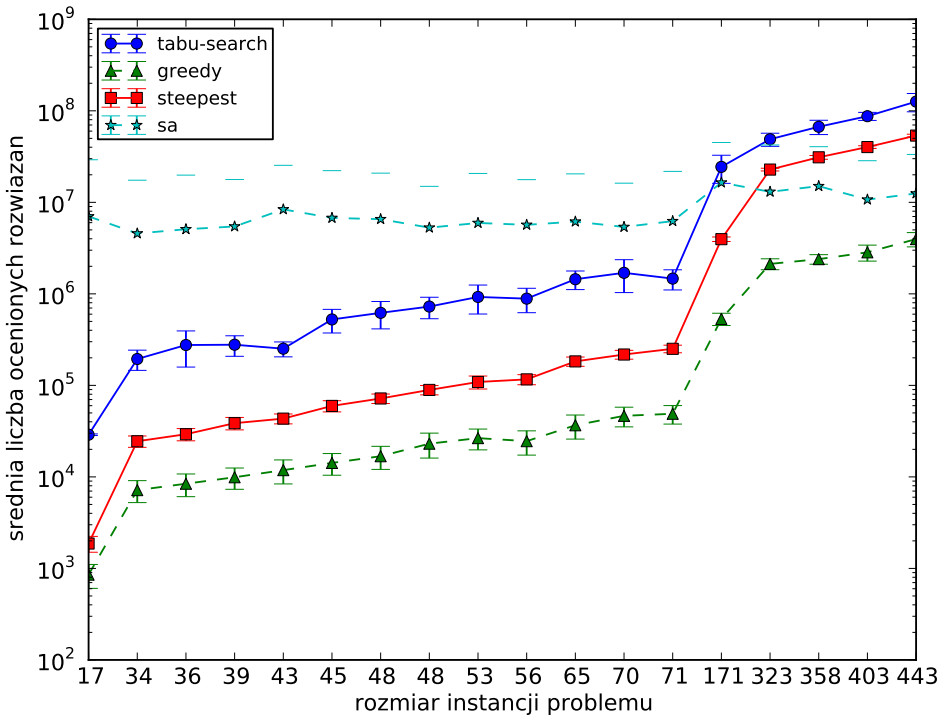
Powyższy wykres zawiera porównanie efektywności algorytmów. Jak można zauważyć najlepszą efektywność osiągnęły algorytmy Greedy, oraz heurystyka nearest neighbour. Najniższą efektywność osiągnęły algorytmy losowe oraz (w przypadku małych instancji problemu) algorytm SA.

3.4 Porównanie średniej liczby kroków oraz liczby ocenionych rozwiązań dla algorytmów steepest, greedy, tabu search i symulowanego wyżarzania.



Rysunek 5: Porównanie średniej liczby kroków algorytmów greedy i steepest w zależności od rozmiaru instancji problemu.

Jak można zauważyć na wykresie, algorytmy Steepest oraz Greedy, wykonują najmniej kroków. W przypadku obu algorytmów, ich liczba ograniczona jest przez monotoniczność przestrzeni rozwiązań. Większa liczba kroków, realizowana przez algorytm Greedy, spowodowana jest tym, że jego rozwiązania wolniej zbiegają do lokalnego optimum. Algorytmy Tabu Search oraz SA, realizują większą liczbę kroków. Duży wpływ na takie wyniki ma warunek stopu, ustalony dla algorytmów (liczba kroków algorytmu, bez poprawy wyniku). Dodatkowo, algorytmy te nie muszą zatrzymywać się w lokalnych optimach.

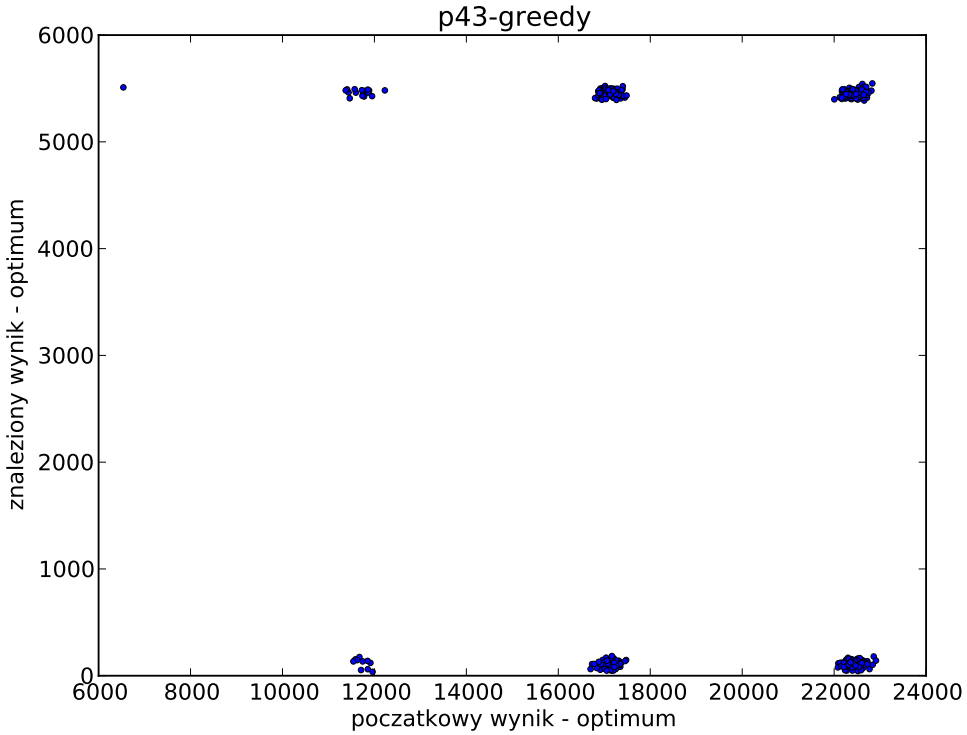


Rysunek 6: Porównanie średniej liczby ocen rozwiązań w zależności od rozmiaru instancji problemu – Algorytmy steepest, greedy, symylowane wyżarzanie oraz tabu search.

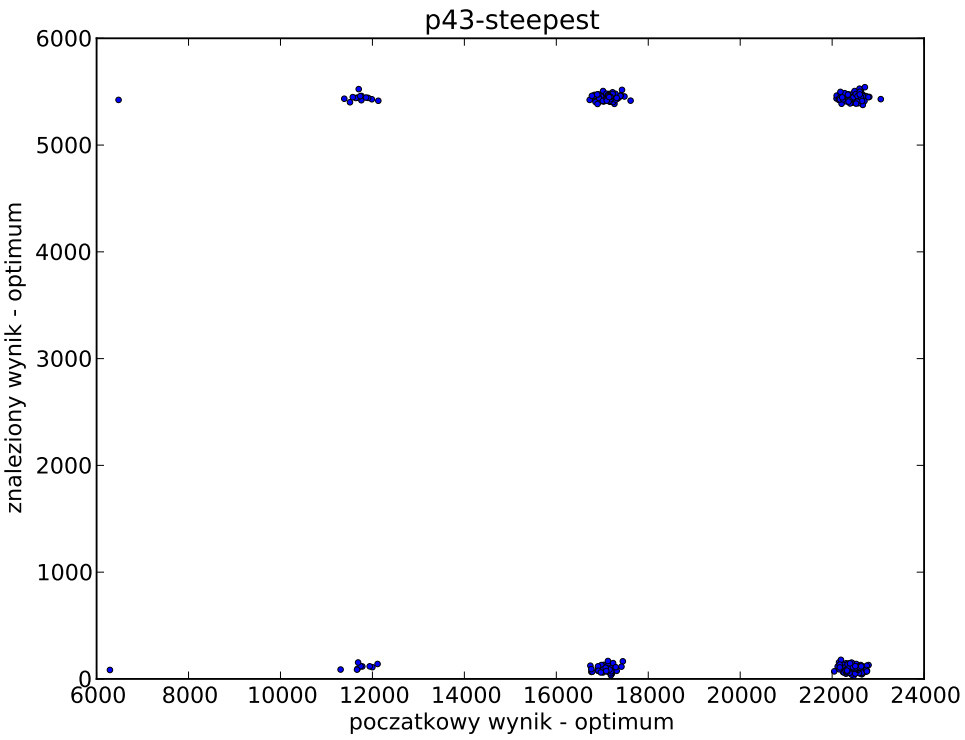
Rozbieżność tych wykresów jest dość oczywista. Steepest zawsze przegląda całe sąsiedztwo w poszukiwaniu optimum, greedy natomiast tylko jego fragment, próbując uzyskać poprawę wyniku. Algorytm Tabu Search, tworząc listę kandydatów ocenia podobnie jak algorytm Steepest, wszystkich swoich sąsiadów. Ponieważ algorytm Tabu Search realizuje średnio więcej kroków niż algorytm Steepest, w trakcie swojego działania dokonuje on większej liczby ocen rozwiązań.

4 Jakość rozwiązania końcowego w zależności od jakości rozwiązania początkowego.

4.1 Instancja p43.

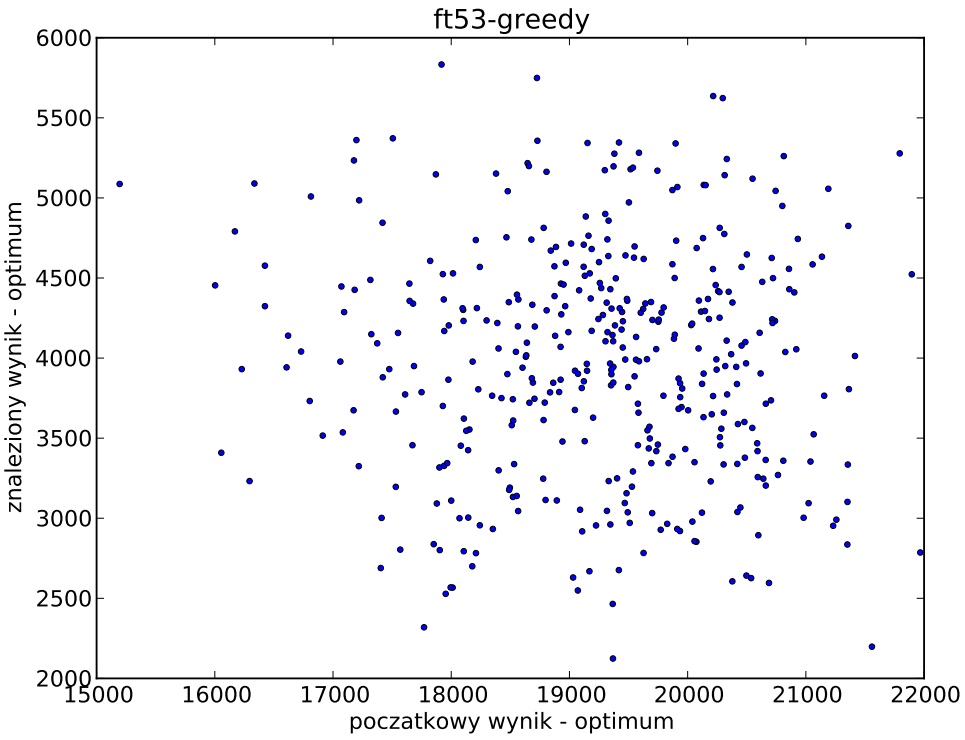


Rysunek 7: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja p43 - algorytm Greedy

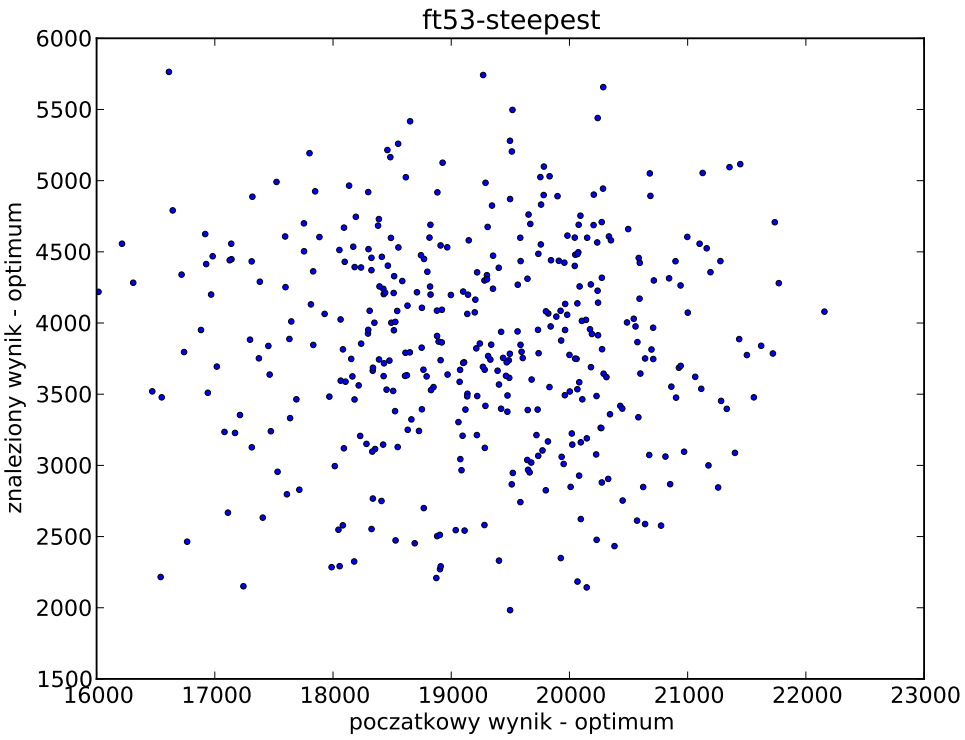


Rysunek 8: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja p43 - algorytm Steepest

4.2 Instancja ft53.

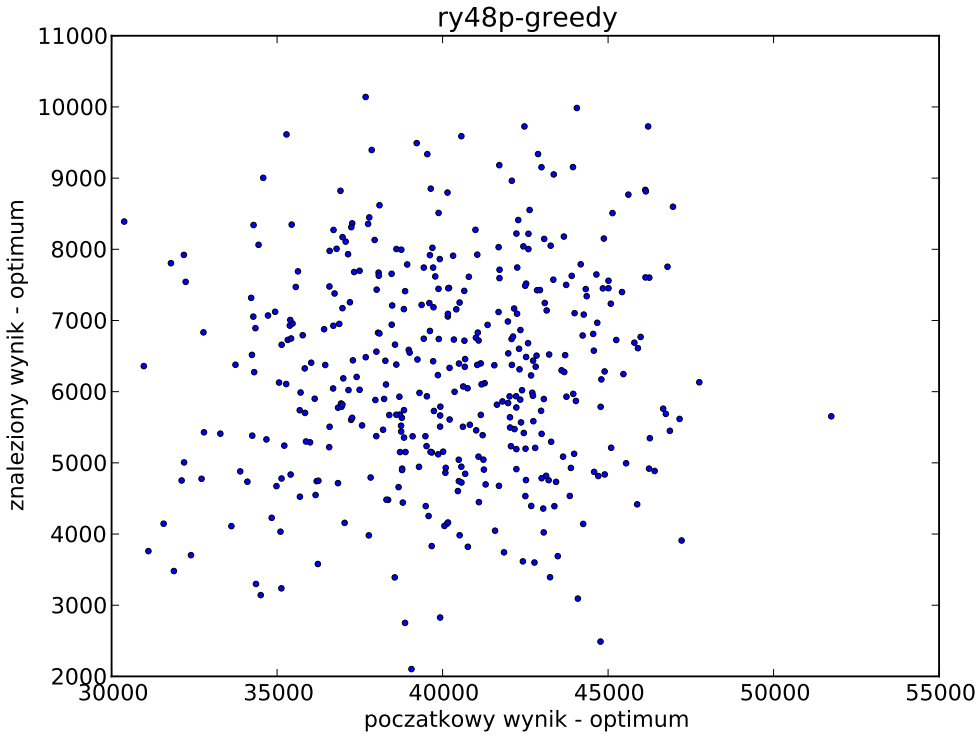


Rysunek 9: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja ft53 - algorytm Greedy

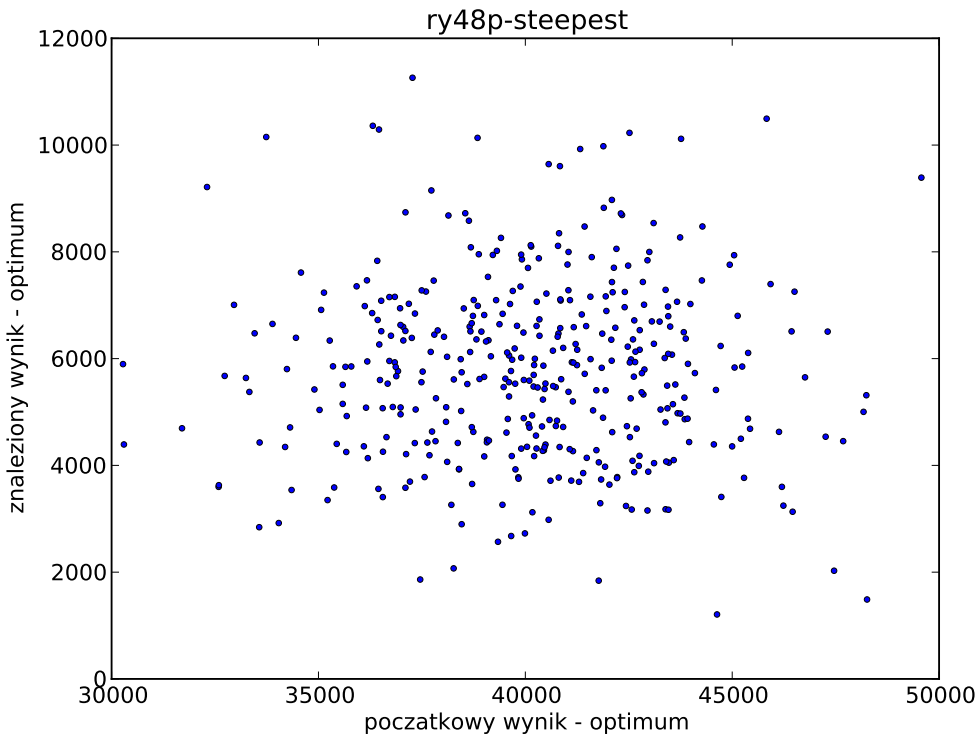


Rysunek 10: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja ft53 – algorytm Steepest

4.3 Instancja ry48p.



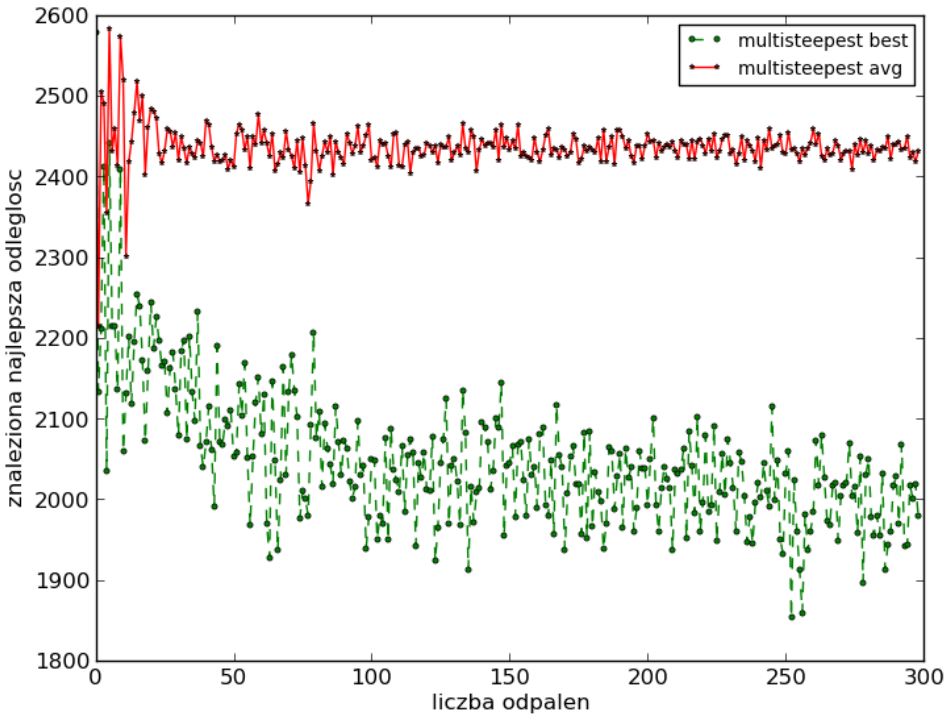
Rysunek 11: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja ry48p - algorytm Greedy



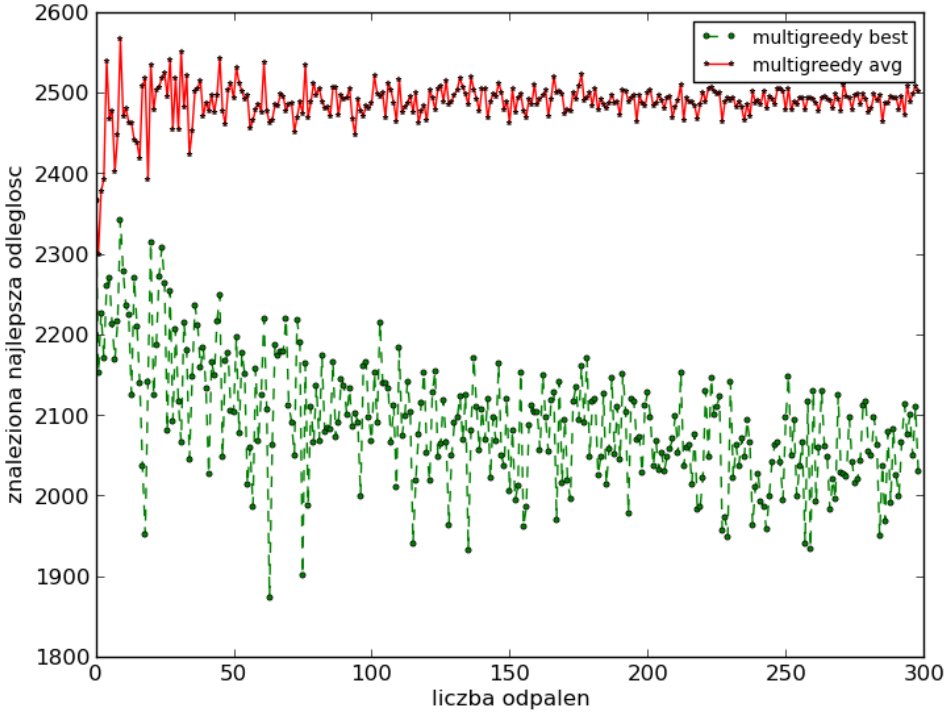
Rysunek 12: Odległości od optimum rozwiązań końcowych w zależności od odległości od optimum rozwiązań początkowych. Instancja ry48p – algorytm Steepest

Każdy wykres pokazuje nam, że zawsze kończymy w rezultacie lepszym niż ten z którego zaczynaliśmy podróż. Dwa z trzech przypadków nie daje ciekawych zależności, jednakże warto spojrzeć na p43. Tworzą się tam klastry ściągające podobne rozwiązania początkowe do podobnych rozwiązań końcowych.

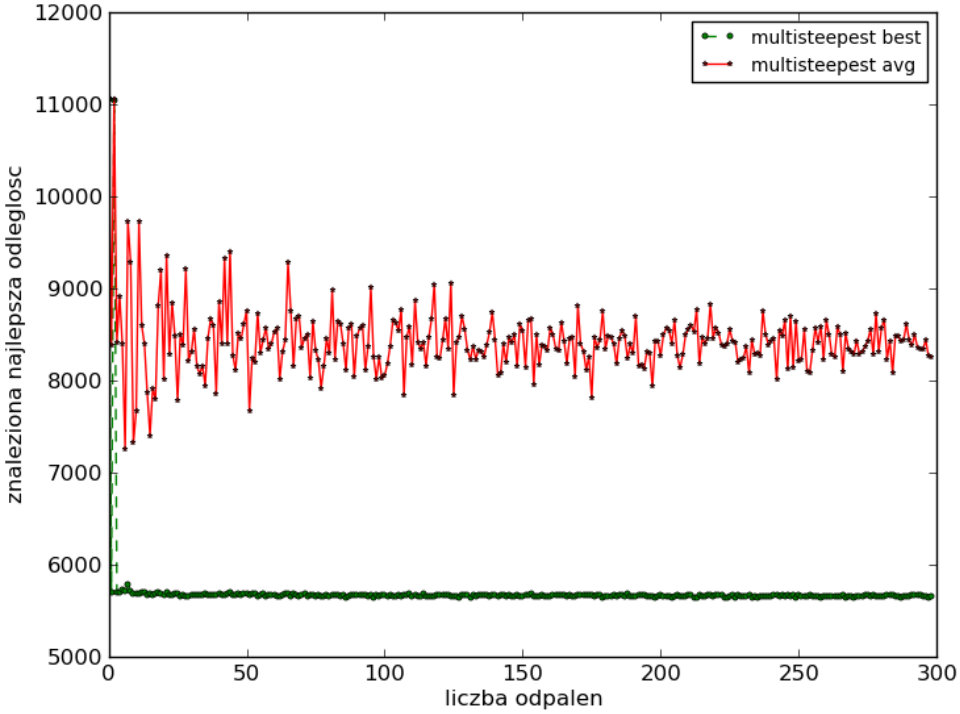
5 Zależność średnich i najlepszych rozwiązań od liczby restartów algorytmu.



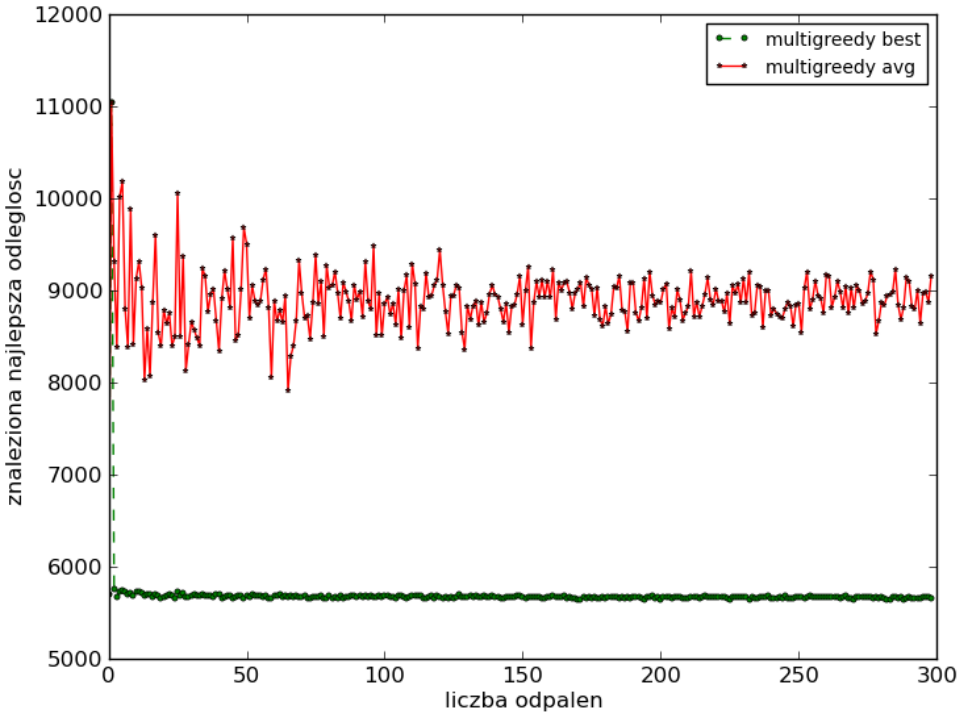
Rysunek 13: Zależność średnich i najlepszych rozwiązań od liczby restartów algorytmu - steepest dla pliku ftv44



Rysunek 14: Zależność średnich i najlepszych rozwiązań od liczby restartów algorytmu - greedy dla pliku ftv44



Rysunek 15: Zależność średnich i najlepszych rozwiązań od liczby restartów algorytmu - steepest dla pliku p43



Rysunek 16: Zależność średnich i najlepszych rozwiązań od liczby restartów algorytmu – greedy dla pliku p43

Eksperyment przeprowadzono poprzez niezależne uruchamianie algorytmu dla 1..300 restartów i obserwowanie jakie wyniki najlepsze oraz średnie udało nam się uzyskać. Zauważyć możemy, że algorytmy są mocno zależne od zadanej instancji. Przykład p43 obrazuje nam jak bardzo często trafiamy w okolice optimum, natomiast ftv44 czasami nawet dla znacznej liczby uruchomień nie trafia w dobre wyniki. Dla wielu odpaleń widać stabilizowanie się średnich. Dla przypadku ftv44 widać silny trend poprawiania wyników do około 100 uruchomień algorytmu steepest, dla greedy taka zależność nie jest tak zauważalna. Widać jednak, że powtarzanie uruchomień prowadzi do prawdopodobnego poprawienia rezultatu, kosztem natomiast jest czas działania.

6 Obiektywna ocena podobieństwa znajdowanych rozwiązań lokalnie optymalnych dla dwóch wybranych instancji

	1	2	3	4	5	6	7	8	9	10
1	1	0.15	0.16	0.14	0.17	0.16	0.16	0.16	0.15	0.13
2	0.15	1	0.16	0.18	0.15	0.17	0.16	0.15	0.16	0.17
3	0.16	0.16	1	0.17	0.16	0.17	0.19	0.16	0.17	0.17
4	0.14	0.18	0.17	1	0.19	0.19	0.17	0.16	0.15	0.17
5	0.17	0.15	0.16	0.19	1	0.16	0.16	0.16	0.15	0.15
6	0.16	0.17	0.17	0.19	0.16	1	0.2	0.18	0.15	0.17
7	0.16	0.16	0.19	0.17	0.16	0.2	1	0.18	0.19	0.14
8	0.16	0.15	0.16	0.16	0.16	0.18	0.18	1	0.18	0.18
9	0.15	0.16	0.17	0.15	0.15	0.15	0.19	0.18	1	0.16
10	0.13	0.17	0.17	0.17	0.15	0.17	0.14	0.18	0.16	1

Tabela 1: Macierz podobieństwa dziesięciu rozwiązań stanowiących optima lokalne - zbiór rbg443

	1	2	3	4	5	6	7	8	9	10
1	1	0.17	0.17	0.15	0.16	0.18	0.15	0.16	0.16	0.17
2	0.17	1	0.16	0.13	0.16	0.17	0.17	0.15	0.16	0.15
3	0.17	0.16	1	0.16	0.2	0.18	0.17	0.15	0.17	0.17
4	0.15	0.13	0.16	1	0.17	0.15	0.19	0.18	0.16	0.17
5	0.16	0.16	0.2	0.17	1	0.17	0.17	0.15	0.17	0.14
6	0.18	0.17	0.18	0.15	0.17	1	0.15	0.15	0.17	0.15
7	0.15	0.17	0.17	0.19	0.17	0.15	1	0.14	0.16	0.16
8	0.16	0.15	0.15	0.18	0.15	0.15	0.14	1	0.16	0.17
9	0.16	0.16	0.17	0.16	0.17	0.17	0.16	0.16	1	0.17
10	0.17	0.15	0.17	0.17	0.14	0.15	0.16	0.17	0.17	1

Tabela 2: Macierz podobieństwa dziesięciu rozwiązań stanowiących optima lokalne - zbiór rbg403

Powyższe tabele zawierają oceny podobieństwa permutacji będących rozwiązaniami znalezionymi za pomocą dziesięciokrotnego odpalenia algorytmu steepest dla zbiorów rbg443 oraz rbg403. Jako miarę podobieństwa przyjęto liczbę wystąpień takich samych par sąsiadów w obu sekwencjach, podzieloną przez długość permutacji. Jak można zauważyć, podobieństwo pomiędzy znalezionymi rozwiązaniami jest niewielkie. Rozwiązania różnią się od siebie w bardzo dużym stopniu.

7 Wnioski

- Algorytmy heurystyczne i losowe osiągają znacznie gorsze wyniki niż algorytmy przeszukiwania lokalnego. Czas ich działania jest jednak znacznie mniej uzależniony od rozmiaru instancji problemu.

-]Algorytmy greedy oraz steepest osiągają podobne wyniki. Algorytm greedy jest jednak dużo bardziej efektywny. Liczba porównywanych przez niego rozwiązań, rośnie wolniej wraz ze wzrostem rozmiaru instancji niż w przypadku algorytmu steepest.
- Stosunkowo dobre wyniki osiąga prosta heurystyka polegająca na budowaniu wyniku w oparciu o najbliższego sąsiada. Mimo, że odległości rozwiązań od optimum, są znacznie większe niż w przypadku algorytmów steepest i greedy, czas działania tej heurystyki jest znacznie mniejszy niż w przypadku tych algorytmów.
- Wielokrotne odpalanie algorytmów przeszukiwania lokalnego, zwiększa szanse na znalezienie wyniku bliższego globalnemu optimum.
- Wyniki dla symulowanego wyżarzania w dużej mierze podyktowane są parametrami jakie zadamy algorytmowi. Należy uważnie je dobrać, aby uzyskać zarówno dobre efekty jak i czas działania.
- Symulowane wyżarzanie jest pochodną algorytmu greedy, które cechuje się większym czasem działania i które w ekstremalnym przypadku (obniżenia temperatury) możemy sprowadzić do algorytmu greedy. Jednak ustawianie odpowiedniej temperatury prowadzić może do poszukania lepszego optimum, kosztem lokalnego pogorszenia wyników.

8 Napotkane trudności

9 Uzasadnienie wprowadzanych ulepszeń, propozycje udoskonaleń i ich spodziewane efekty

9.1 Propozycje udoskonaleń

- Sprawdzenie działania operatora sąsiedztwa 3-opt. Prawdopodobnie większe sąsiedztwo znacznie wpłynęłoby na czas poszukiwania rozwiązania przez algorytmy greedy i steepest. Jednocześnie algorytmy te, przeszukiwałyby większą przestrzeń rozwiązań co pozytywnie wpłynęłoby na odległość wyników od optimum.