Sprawozdanie z laboratorium:
Metaheurystyki i Obliczenia Inspirowane Biologicznie

Część I: Algorytmy optymalizacji lokalnej, problem QAP

26 listopada 2014

Prowadzący: dr hab. inż. Maciej Komosiński

Autorzy:				
	Jacek Jankowski	inf99410	ISWD	jacek.j.jankowski@gmail.com
	Benedykt Jaworski	inf99893	ISWD	sth.jaworski@gmail.com

Zajęcia poniedziałkowe, 15:10.

Oświadczamy, że niniejsze sprawozdanie zostało przygotowane wyłącznie przez powyższych autorów, a wszystkie elementy pochodzące z innych źródeł zostały odpowiednio zaznaczone i są cytowane w bibliografii.

1 Quadratic Assignment Problem

1.1 Opis problemu

QAP jest problemem kombinatorycznym NP-trudnym. Nie istnieją algorytmy pozwalające na rozwiązanie tego problemu w czasie wielomianowym. QAP jest matematycznym modelem dla zadania umieszczenia połączonych ze sobą elementów na płytce drukowanej (elektronika) lub wyborze lokalizacji jednostek firmy (administracja). Problem można opisać w następujący sposób:

> Istnieje n jednostek i n lokalizacji. Dla każdej pary jednostek istnieje *dystans*, a dla każdej pary lokalizacji istnieje *przepływ* (nazywany również wagą). Przepływ może być np. ilością materiałów transportowanych pomiędzy magazynem a halą produkcyjną lub szerokością ścieżki na płytce drukowanej. Zadanie polega na przydzieleniu jednostkom lokalizacji w taki sposób, aby zminimalizować sumę iloczynów dystansów pomiędzy jednostkami i przepływów pomiędzy nimi.

Formalna definicja problemu QAP jest następująca:

Definicja 1. *Dane są dwa zbiory, P („jednostki") i L („lokalizacje"), równych rozmiarów, a także funkcja wagi $w : P \times P \to \mathbb{R}$ i funkcja odległości $d : L \times L \to \mathbb{R}$. Znajdź funkcję przypisującą jednostkom lokalizacje $f : P \to L$ w taki sposób, że funkcja kosztu:*

$$\sum_{a,b \in P} w(a,b) \cdot d(f(a),f(b)) \tag{1}$$

ma możliwie najmniejszą wartość.

Tak zdefiniowana funkcja kosztu intuicyjnie zachęca do umieszczania pozycji o dużym przepływie w niewielkiej odległości od siebie.

1.2 Reprezentacja i operator sąsiedztwa

Rozwiązaniem problemu jest permutacja jednostek. Kolejność jednostek w permutacji określa ich położenie i pozwala odczytać odległości z macierzy odległości. Użyty operator sąsiedztwa to sąsiedztwo 2-opt, zamieniające miejscami dwa elementy w permutacji.

Złożoność obliczeniowa uzyskania wartości kosztu aktualnej permutacji ma charakter $O(n^2)$, gdzie n to rozmiar permutacji. Zaimplementowaliśmy funkcję, która sprawdza o ile zmieni się koszt, jeżeli użyjemy operatora sąsiedztwa na danej permutacji. Funkcja ta ma złożoność liniową, co pozwoliło przyspieszyć działanie algorytmu.

Gdyby uwzględnić fakt, że większość problemów jest symetryczna, można by zmniejszyć liczbę wymaganych obliczeń dwukrotnie, uwzględniając w obliczeniach jedynie część macierzy nad główną przekątną definiujących koszty zamiast pełnych macierzy. Skupiliśmy się jednak na najbardziej ogólnej postaci QAP.

Rozmiar sąsiedztwa dla operatora 2-opt to $\frac{n \cdot (n-1)}{2}$.

1.3 Algorytmy

Do generowania losowości w algorytmach użyto generatora liczb pseudolosowych Marsenne Twister 19937, a jego wyjście było przekształcane w jednorodną dystrybucję w wymaganym przez nas przedziale. Ziarno generatora liczb losowych jest ustawiane na początku działania programu. Wartość jest taka sama dla każdego uruchomienia, co gwarantuje powtarzalność eksperymentów.

Liczba uruchomień algorytmu losowego była proporcjonalna do rozmiaru problemu. Na podstawie czasu wykresu czasu działania stwierdziliśmy, że gwarantuje to czas działania podobny do czasów działania algorytmów Greedy i Steepest.

1.3.1 Heuristic

Algorytm kieruje się heurystyką i ustawia najpierw najbliżej siebie jednostki o największych przepływach. W każdym kolejnym kroku dobierana jest spośród wszystkich pozostałych jednostek para o największym przepływie i umieszczana na niewykorzystanych do tej pory lokalizacjach, które są w najmniejszej odległości od siebie. Algorytm jest w pełni deterministyczny.

1.3.2 Random

Algorytm generuje losową permutację, z równomiernym rozkładem prawdopodobieństwa – każde możliwe rozwiązanie ma jednakowe prawdopodobieństwo wylosowania. Czas jego działania zależny jedynie od wielkości problemu.

1.3.3 Greedy

Algorytm Greedy wybiera losowego sąsiada i oblicza jego koszt. Jeżeli sąsied ma koszt niższy lub równy aktualnemu rozwiązaniu, to zastępuje on aktualne rozwiązanie. Jeżeli nie, algorytm sprawdza kolejnego sąsiada. Jeżeli żaden sąsiad nie ma kosztu równego lub niższego, algorytm kończy działanie. Aby zapobiec zapętleniu (przechodzieniu pomiędzy dwiema permutacjami o równej ocenie), stosujemy ograniczenie do 3 przejść do sąsiada o równym koszcie.

1.3.4 Steepest descent

Algorytm działa podobnie jak Greedy, z tym, że sprawdza wszystkich sąsiadów, i spośród nich wybiera tego o najniższej wartości funkcji celu. Aby zapobiec zapętleniu (przechodzieniu pomiędzy dwiema permutacjami o równej ocenie), stosujemy ograniczenie do 3 przejść do sąsiada o równym koszcie.

2 Eksperymenty

2.1 Instancje problemów

Eksperymenty uruchamialiśmy na następujących problemach:

1. bur26a
2. chr12a
3. chr15a
4. chr18a
5. chr20a
6. chr22a
7. chr25a
8. had14
9. nug17
10. ecs16a
11. els19
12. esc32
13. kra30a
14. nug21

Problemy te posiadają obliczone optimum, co ułatwiło uzyskanie jakości uzyskanych przez nas rozwiązań.

Problemy pochodzą z repozytorium QAPLIB[1]. Źródłem danych są m.in. czas pisania stenotypisty, macierze sąsiedztwa drzew czy testy układów elektronicznych.

Algorytmy Greedy oraz Steepest descent zostały uruchomione 200 razy, a wyniki i parametry działania zagregowane. Algorytm losowy został uruchomiony tyle razy, by czas jego działania odpowiadał rządem wielkości czasowi działania algorytmów Greedy i Steepest descent. Poniższe wykresy przedstawiają anlizę pracy i sprawności algorytmów.

2.2 Jakość działania jako funkcja rozmiaru problemu

2.2.1 Średnia jakość rozwiązania

Jakość działania liczona jest jako stosunek kosztu rozwiązania optymalnego do kosztu rozwiązania znalezionego przez algorytm. Wykres z Rys. 1 przedstawia wartości średnie jakości (punkty) oraz wartość 1 odchylenia standardowego (linie pionowe). Wyniki nie mają rozkładu normalnego, dlatego do interpretacji odchylenia standardowego powinniśmy podchodzić ostrożnie. Zakresy odchylenia standardowego mogą przekraczać wartość 1,0. Dla problemu esc32 wynika to ze skupienia jakości rozwiązań w okolicy rozwiązania optymalnego i niewielu rozwiązań o znacznie odbiegającej jakości.

Z wykresu odczytujemy, że średnia jakość działania algorytmu Random jest dla większości problemów niska i nie przekracza wartości 0,9. Dla 8 z 14 przypadków wartość średnia nie przekracza 0,5, więc znalezione rozwiązania mają co najmniej dwa razy większy koszt. Dwa najlepsze rozwiązania są dla problemów o rozmiarach 14 i 26. Widoczne jest jednak, że są to bardzo proste problemy – algorytmy Greedy i Steepest descent mają wartości średnie prawie równe optimum, z niewidocznym odchyleniem standardowym.

Algorytm korzystający z heurystyki w każdym przypadku znajduje lepsze lub równie dobre rozwiązanie, co średnie rozwiązanie algorytmu losowego.

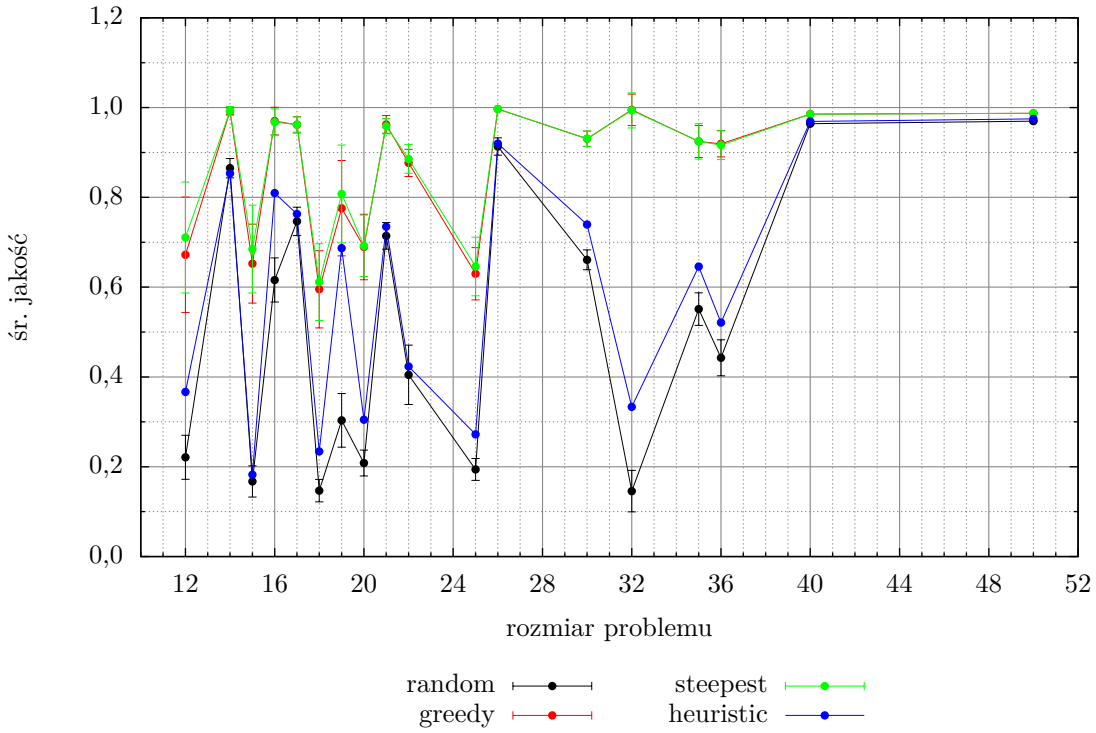
Średnie rozwiązania dla 200 uruchomień dla algorytmów Steepest descent oraz Greeedy mają bardzo podobne, nierozróżnialne wartości.

2.2.2 Jakość najlepszego znalezionego rozwiązania

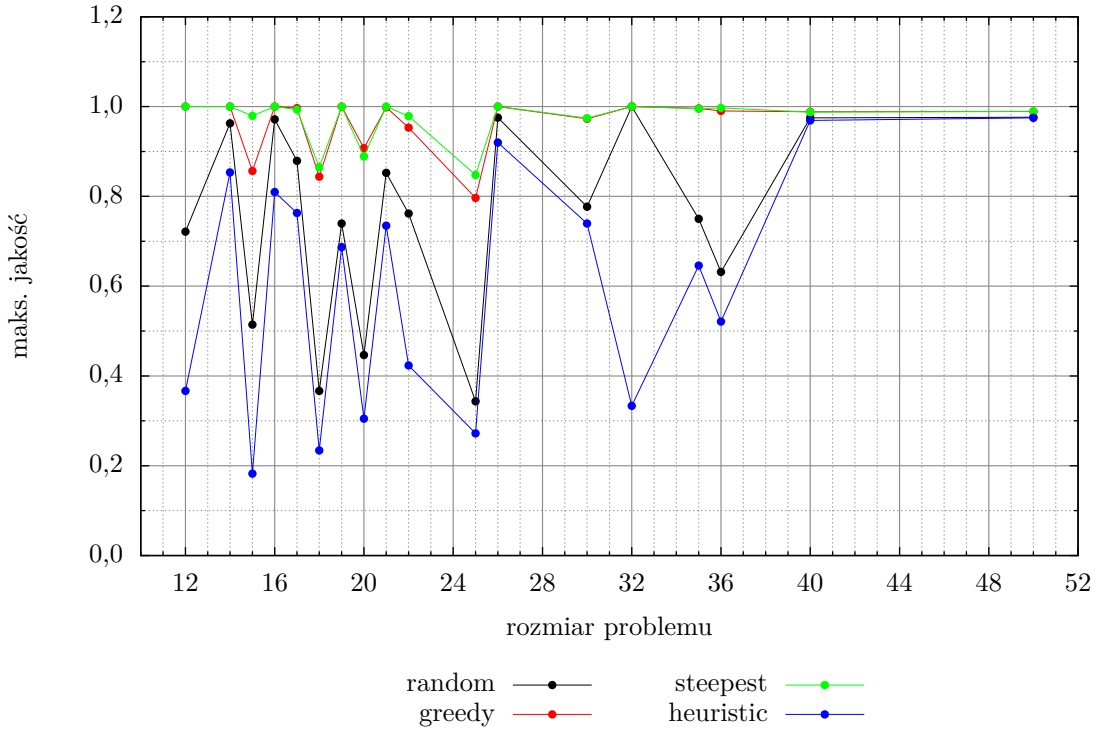
Wykres z Rys. 2 przedstawia jakość najlepszego rozwiązania, znalezionego przez algorytmy, dla poszczególnych instancji.

Algorytmy Greedy i Steepest descent zwracają bardzo podobne wyniki. Jeżeli udało się znaleźć optymalne rozwiązanie, to oba algorytmy je znajdowały. Dla problemów, gdzie nie udało się znaleźć optymalnego rozwiązania, Steepest descent osiągał lepsze rezultaty (np. instancje o wielkości 15, 22, 25). W jednym przypadku (problem chr20a) najlepsze rozwiązanie znalezione przez algorytm Greedy miało lepszą jakość.

[1] `http://www.opt.math.tu-graz.ac.at/qaplib/`



Rysunek 1: Wykres uśrednionej jakości rozwiązania od rozmiaru problemu



Rysunek 2: Wykres jakości najlepszego znalezionego rozwiązania od rozmiaru problemu

3 Czas działania w zależności od rozmiaru problemu

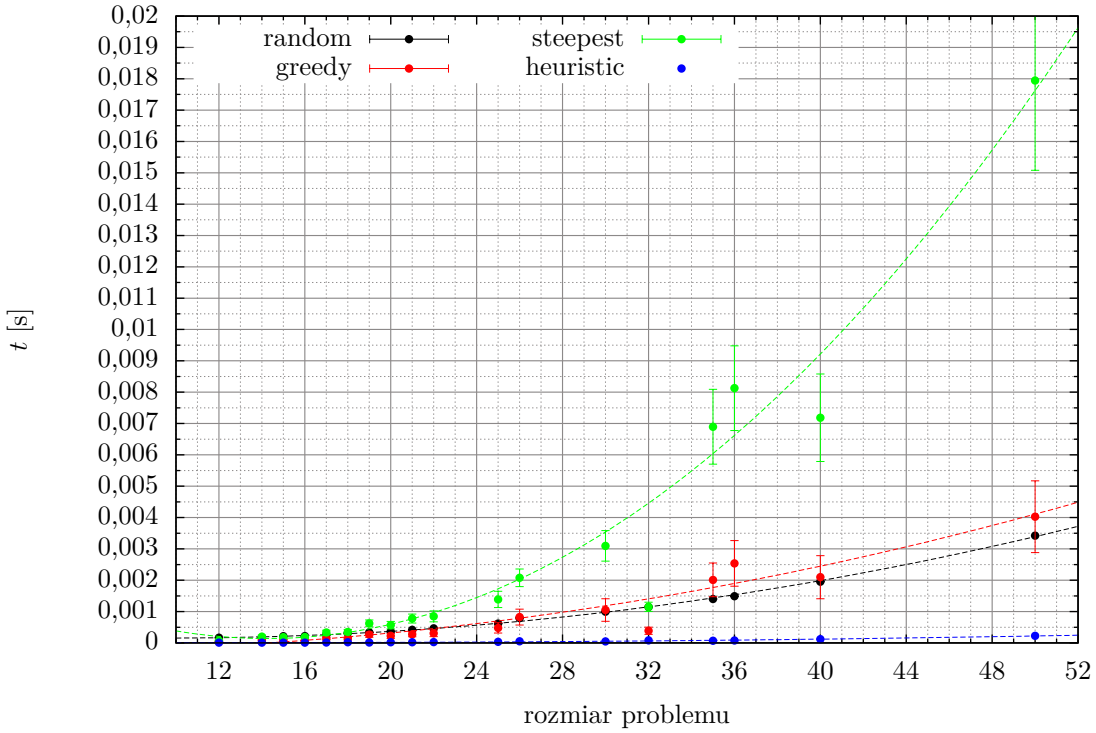
3.1 Random, Greedy i Steepest descent

Na wykresie z Rys. 3 przedstawiona jest średnia wartość czasu potrzebnego na znalezienie rozwiązania dla 200 uruchomień algorytmów Random, Greedy i Steepest descent i wartość jednego odchylenia standardowego dla wszystkich uruchomień.

Do wyników dopasowaliśmy krzywe kwadratowe w postaci $ax^2 + bx + c$.

W regresji pominęliśmy problem o rozmiarze 32, ponieważ Greedy i Steepest descent ze względu na bardzo dużą liczbę zer w macierzach zatrzymywały się po kilku krokach i wyniki dla tego problemu bardzo odstawały od pozostałych.

Problemy posiadają różne krajobrazy rozwiązań. Ich charakter może być powodem, dla którego średnie rozwiązania nie układają się dokładnie na przybliżonej charakterystyce, a odchylenia standardowe mają duże wartości.



Rysunek 3: Wykres uśrednionego czasu wykonywania od rozmiaru problemu

3.2 Algorytm heurystyczny

Na podstawie wykresu dla algorytmu heurystycznego, przedstawionego na rys 4, widzimy, że regresja kwadratowa dość dobrze przewiduje czas działania dla badanych wielkości instancji. W rzeczywistości algorytm heurystyczny wykonuje zagnieżdżone trzy pętle typu for, zależne od wielkości problemu, co sugerowałoby zależność od potęgi trzeciej rozmiaru problemu.

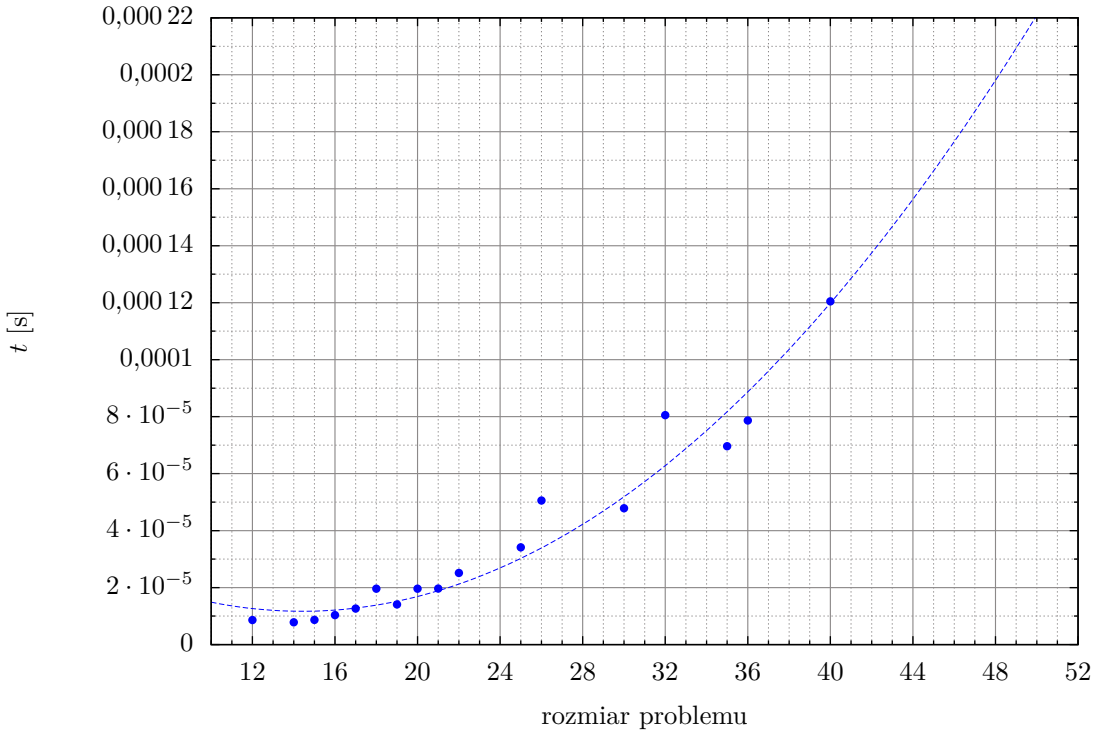
Czas obliczenia rozwiązania nie pokrywa się dokładnie z przybliżoną krzywą, bo w zależności od charakteru problemu, kolejne iteracje mogą potrzebować mniejszej liczby operacji na pamięci dla znalezienia minimalnej wartości przepływu (np. gdy wszystkie pozostałe do dyspozycji wartości są sobie równe – w efekcie czego algorytm pomija dużą część kopiowań wartości wewnątrz instrukcji warunkowych i czas działania jest krótszy).

4 Jakość rozwiązania w zależności od czasu działania

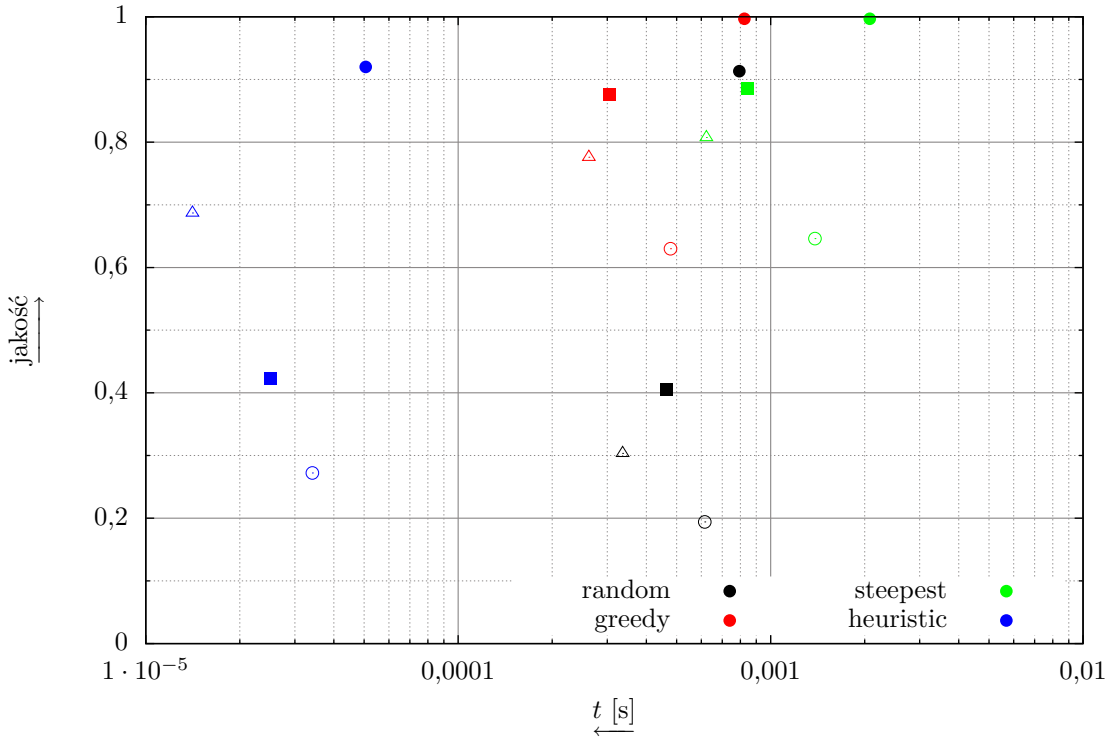
Na Rys. 5 i 6 przedstawiamy dwa wykresy, które pozwalają porównać algorytmy na podstawie czasu ich pracy i osiąganych rozwiązań. Na pierwszym wykresie dla czytelności przedstawiliśmy tylko 4 instancje problemów, drugi wykres zawiera punkty dla wszystkich instancji. Problemowi przypisany jest kształt znacznika, kolor indentyfikuje natomiast algorytm, który osiągnął dany wynik.

Algorytm heurystyczny jest zdecydowanie najszybszy, ale jakość jego rozwiązań jest niska. Tylko w przypadku problemu oznaczonego wypełnioną kropką jakość rozwiązania przekroczyła poziom 0,9. Algorytm losowy osiąga gorszcze wyniki niż heurystyczny, pomimo zdecydowanie dłuższego czasu pracy.

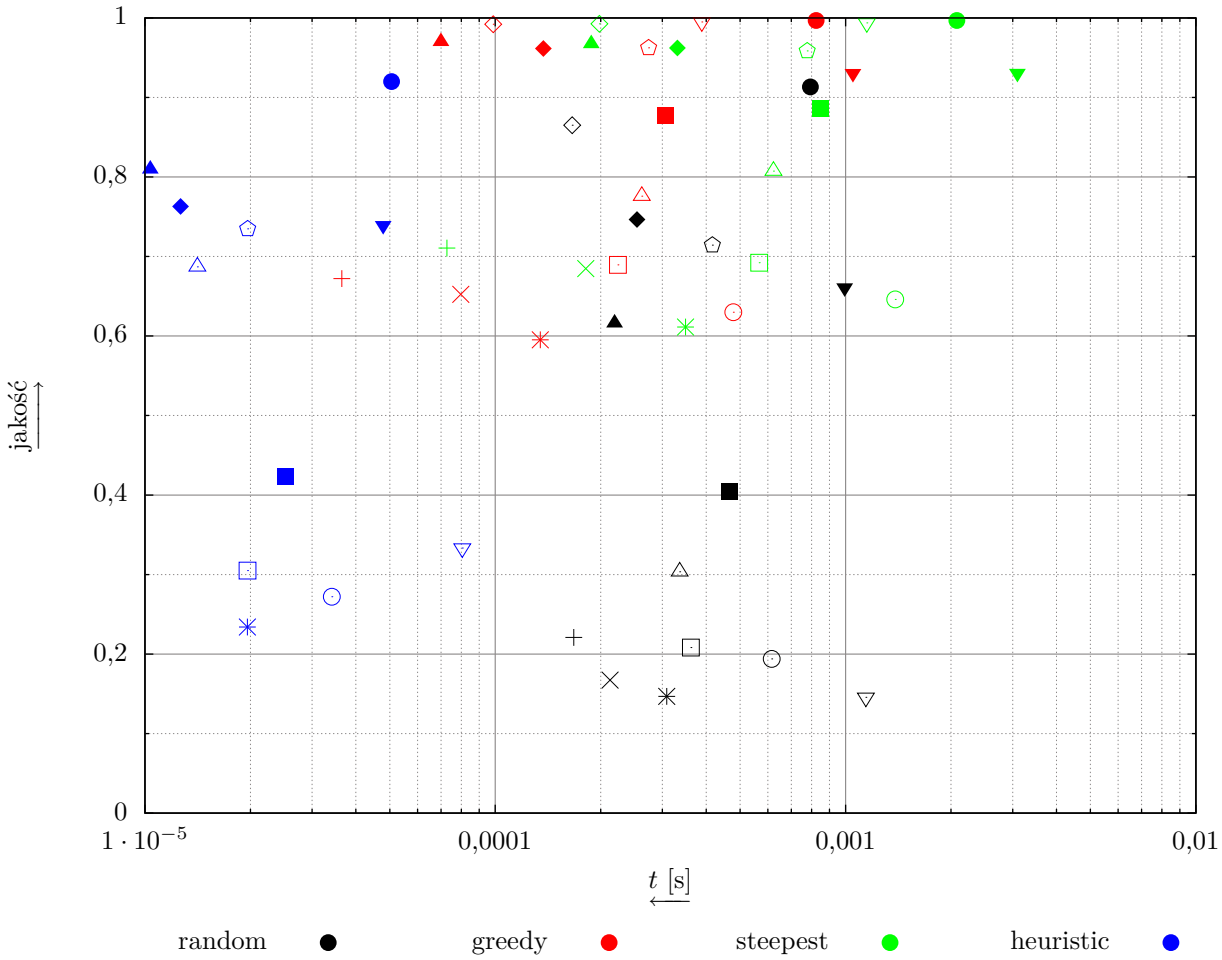
Algorytmy Greedy i Steepest osiągają porównywalne wyniki pod względem jakości. Rozwiązania oznaczone kolorem czerwonym znajdują się zawsze na lewo od odznaczonych na zielono, Greedy potrzebuje więc mniej czasu na osiągnięcie podobnego, co Steepest, rezultatu.



Rysunek 4: Wykres czasu wykonywania alg. heurystycznego od rozmiaru problemu



Rysunek 5: Wykres jakości rozwiązania w funkcji czasu rozwiązania dla wybranych problemów



Rysunek 6: Wykres jakości rozwiązania w funkcji czasu rozwiązania dla różnych problemów

5 Średnia liczba kroków w zależności od rozmiaru problemu

Średnia liczba kroków wykonywanych przez algorytmy zależy bardzo od krajobrazu rozwiązań. Z wykresu z Rys. 7 wywnioskowaliśmy, że można by dla algorytmu Steepest descent dopasować zależność między liczbą kroków a rozmiarem problemu (poza ekstremalnym przypadkiem problemu esc32).

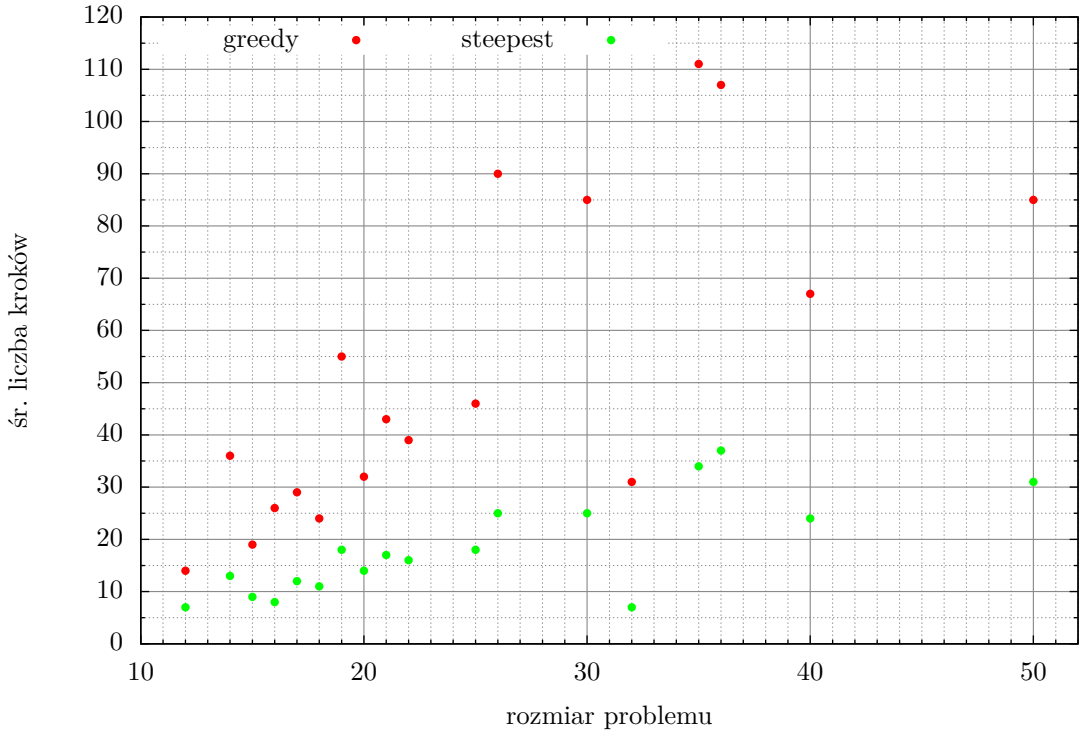
Dla algorytmu Greedy zależność nie jest tak łatwa do zbudowania. Wszystkie problemy wymagają większej liczby kroków niż w przypadku Steepest descent. Niektóre instancje, jak np. had14, els19 czy bur26a wymagały dużo więcej przejść pomiędzy sąsiadami. Algorytm dłużej więc „podróżował" po przestrzeni rozwiązań.

Biorąc pod uwagę, że dla tych problemów znalezione zostały rozwiązania optymalne (had14 i bur26a) lub bardzo bliskie optimum (els19), a odchylenia standardowe liczby kroków są bardzo małe, wnioskujemy, że krajobraz zawiera mniej lokalnych wypłaszczeń czy lokalnych minimów i bliższy jest sytuacji z globalną wypukłością.

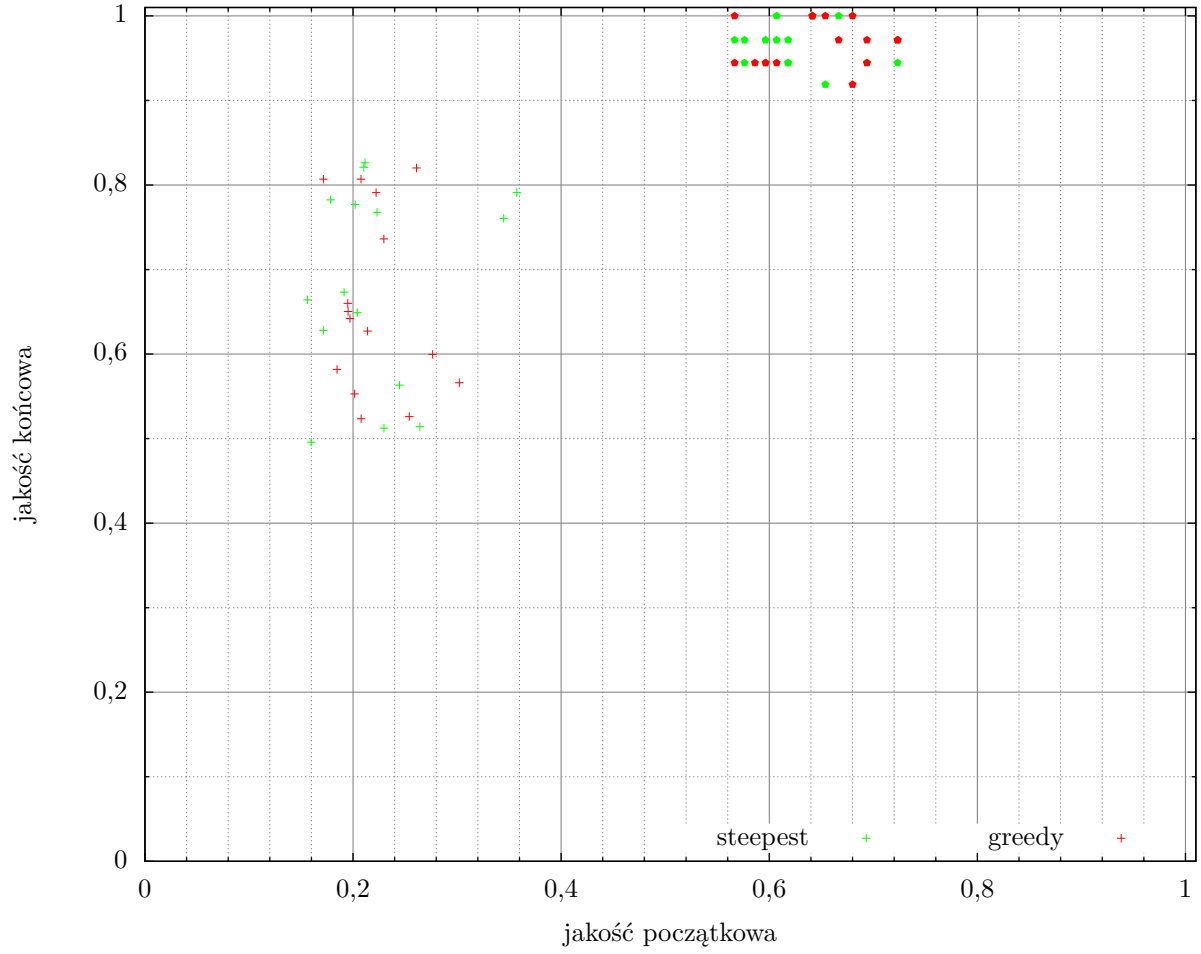
6 Jakość rozwiązania najlepszego w zależności od początkowego

Z wykresów przedstawionych na Rys. 8 i 9 widać, że występuje pewna zależność między jakością rozwiązania początkowego oraz ostatecznie znalezionego rozwiązania. Widać również, że punkty dotyczące danej instancji problemu skupiają się na niewielkim obszarze na wykresie. Oznacza to, że jakość początkowego, losowego rozwiązania bardzo zależy od testowanej instancji problemu.

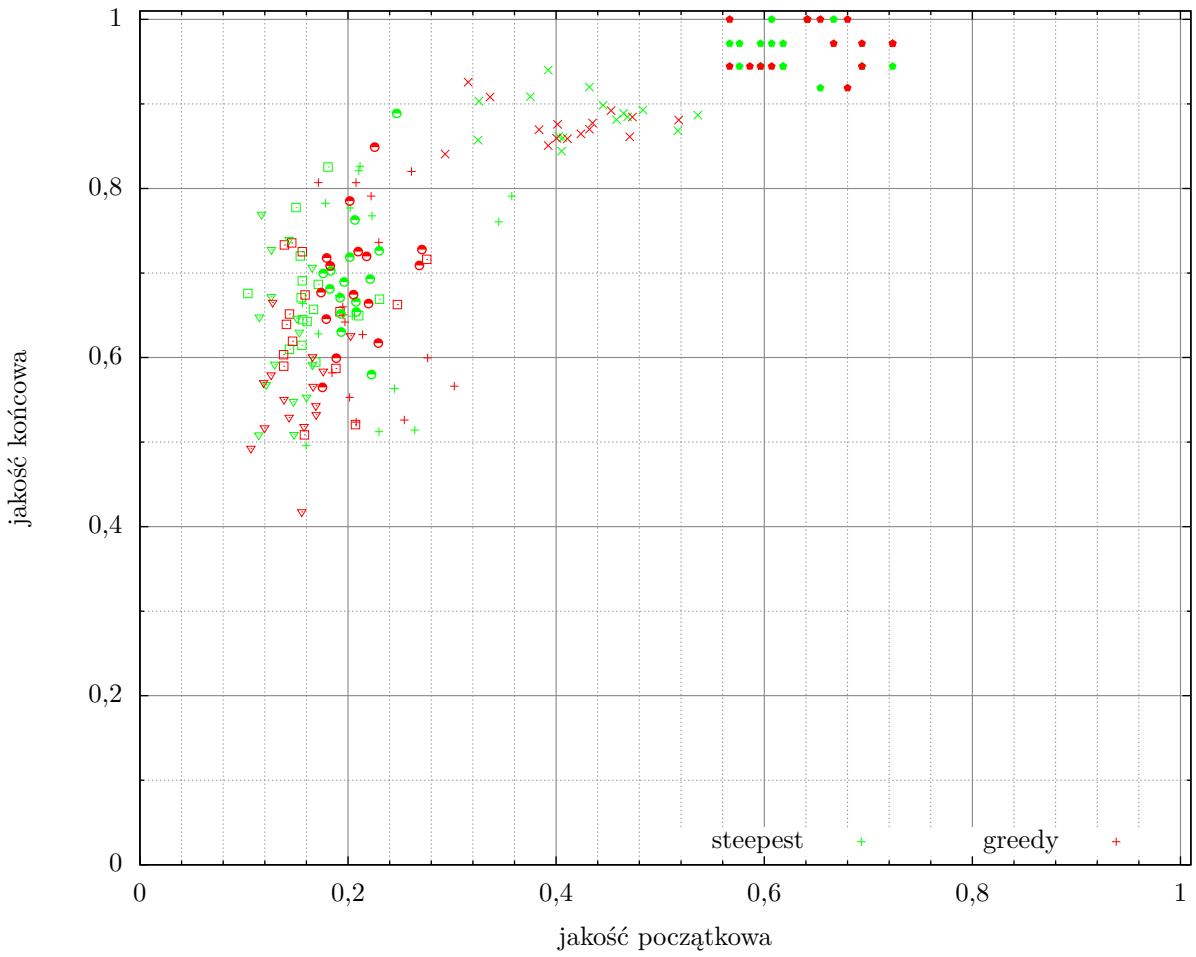
Na wykresie z Rys. 9 widać, że dla większości problemów jakość początkowego rozwiązania waha się między 0,08 a 0,22, a jedynie dla dwóch przykładowych problemów są to wartości powyżej 0,4, a jakość znalezionego rozwiązania końcowego dla większości problemów waha się dość istotnie. Trudno zatem ocenić, czy zależność między jakością rozwiązania początkowego a końcowego jest rzeczywistym związkiem przyczynowo-skutkowym (jeśli wysoka jakość początkowa, to wysoka jakość końcowa), czy też wysoka jakość rozwiązania końcowego dla wspomnianych dwóch problemów wynika tylko z ich charakteru.



Rysunek 7: Wykres liczby kroków od rozmiaru problemu



Rysunek 8: Wykres jakości rozwiązania najlepszego od rozwiązania początkowego dla wybranych problemów



Rysunek 9: Wykres jakości rozwiązania najlepszego od rozwiązania początkowego dla różnych problemów

7 Poprawa jakości znalezionego rozwiązania w kolejnych krokach

Wykres na Rys. 10 przedstawia jaką wielkość miary jakości miały rozwiązania w kolejnych krokach algorytmu Steepest descent dla problemu kra30a. Wykres z Rys. 11 przedstawia analogiczne dane dla algorytmu Greedy. Oba algorytmy zostały uruchomione 200 razy.

Z wykresu 10 widać, że Steepest descent dość szybko dochodzi do rozwiązania (liczba kroków bliska wielkości problemu), krzywa poprawy jakości ma charakter eksponencjalny – jakość wyraźnie, coraz wolniej z czasem, poprawia się do pewnej wartości granicznej bliskiej optimum. Nie zdarza się raczej, by algorytm zatrzymał się dłużej na jednej wartości i „podróżował" po rozwiązaniach o jednakowej ocenie. Dopiero pod koniec działania algorytmu, gdy ów nie znajduje już żadnych rozwiązań lepszych od aktualnego, pozostaje przez kilka kroków przed zakończeniem na stałej wartości jakości.

Z wykresu 10 widać, że Greedy, w przeciwieństwie do algorytmu Steepest descent, często zatrzymuje się na kilka kroków na jednej wartości jakości, a kształt poprawy, choć również przypomina krzywą wykładniczą, jest o wiele bardziej nieregularny. Liczba kroków wykonanych przez Greedy'ego jest kilkakrotnie większa od liczby kroków algorytmu Steepest descent. Jednak obliczenie przejścia w algorytmie Greedy jest mniej czasochłonne, więc, jak było widać we wcześniejszych sekcjach sprawozdania, ogólny czas wykonywania Greedy'ego jest krótszy.

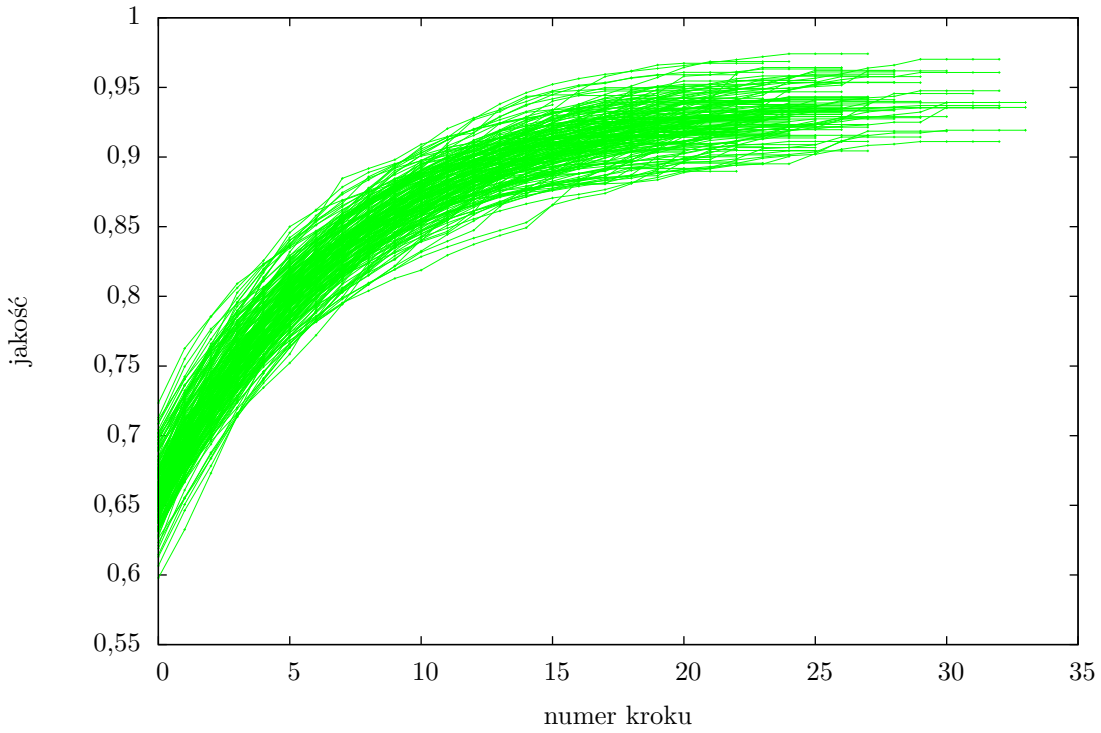
Oba algorytmy mają dość duży rozrzut jakości rozwiązania początkowego (startują od podobnych rozwiązań losowych, o jakościach od 0,6 do ok. 0,75, różnica 0,15), i oba zbiegają się do niezbyt mocno rozrzuconych pod względem miary jakości rozwiązań końcowych – w analizowanym przypadku wartości między 0,9 a 0,95, różnica 0,05.

8 Zależność jakości od liczby powtórzeń

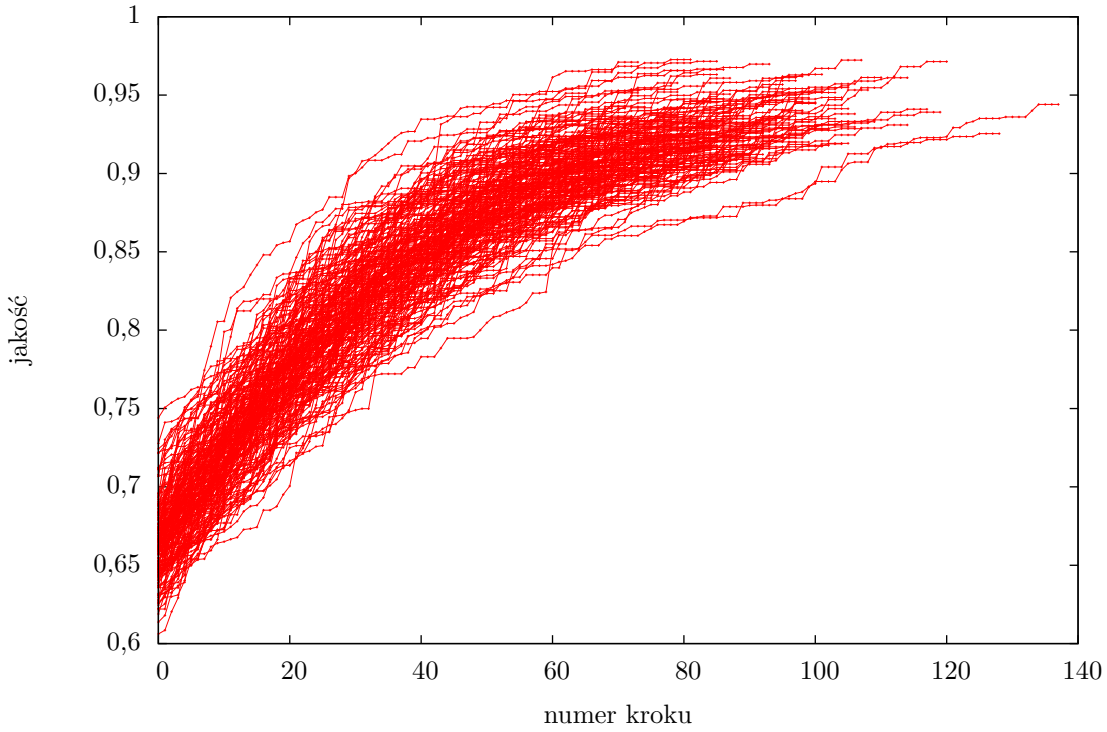
Z wykresu na Rys. 12 widać, że...POPRAWIĆ

9 Wnioski

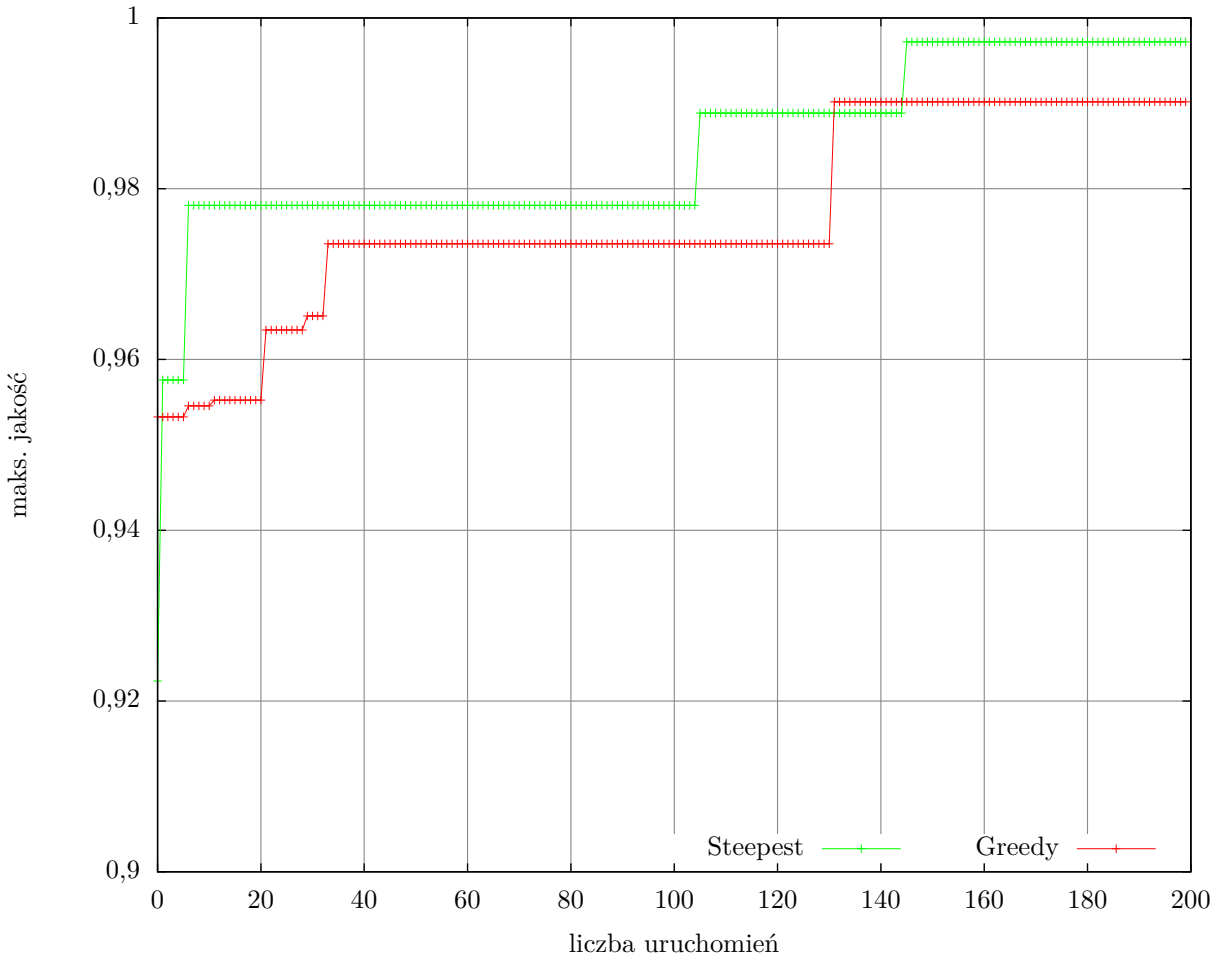
Głównym pytaniem, jakie zadaliśmy sobie po wykonaniu doświadczeń było: który algorytm jest najlepszy? Problem z odpowiedzią polega na tym, że nie można określić pojedynczego kryterium oceny algorytmów. W naszej analizie weźmiemy pod uwagę dwa kryteria: jakość średniego i najlepszego rozwiązania oraz czas pracy.



Rysunek 10: Wykres jakości rozwiązania w kolejnych krokach 200 przebiegów algorytmu Steepest descent



Rysunek 11: Wykres jakości rozwiązania w kolejnych krokach 200 przebiegów algorytmu Greedy



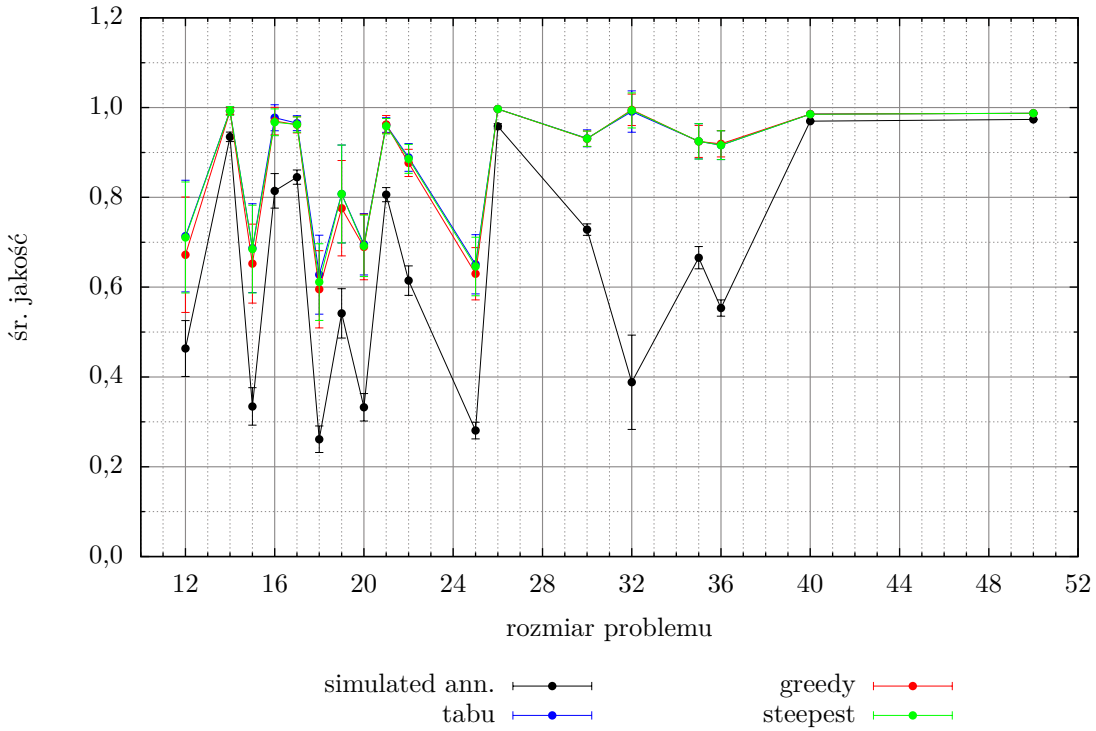
Rysunek 12: Wykres maksymalnej jakości rozwiązania dla kolejnych 200 uruchomień algorytmów Greedy i Steepest descent na problemie ste36c

Algorytm heurystyczny pracuje wielokrotnie szybciej niż algorytmy Steepest descent i Greedy. Otrzymywane rozwiązania mają koszt średnio dwa razy większy od minimalnego dla testowanych instancji. Jeżeli priorytetem jest krótki i przewidywalny czas pracypotrzebny na znalezienie rozwiązania przed rozpoczęciem pracy, to algorytm heurystyczny jest najlepszym wyborem.

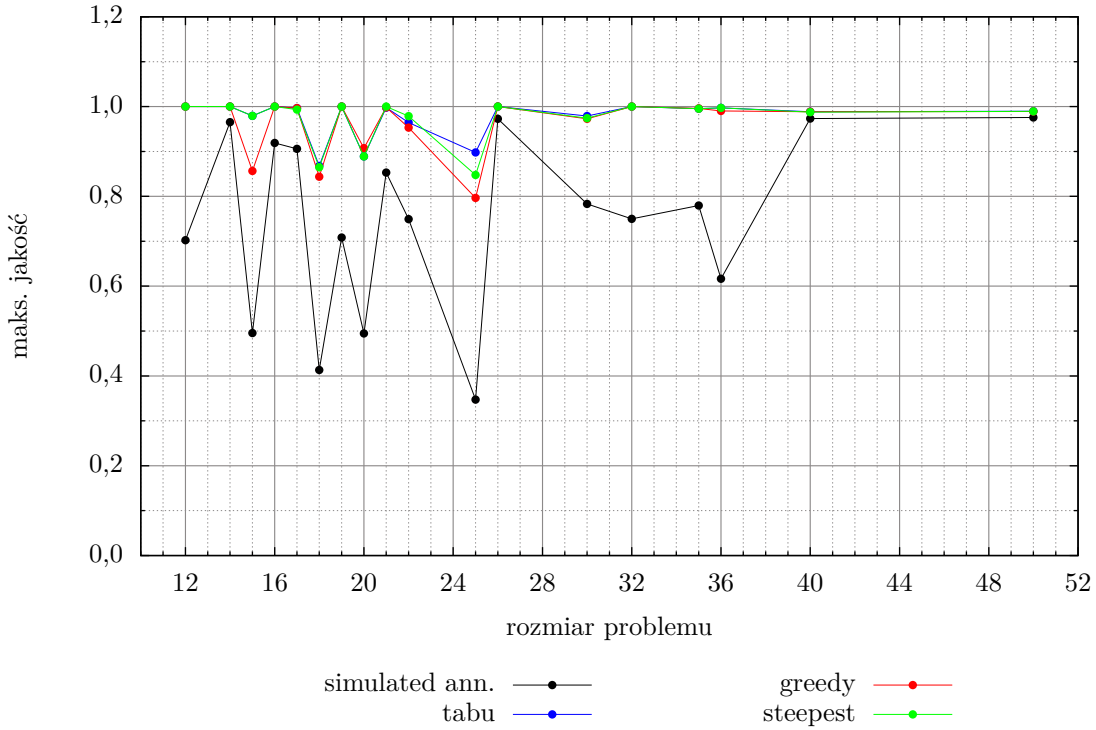
Algorytm losowy pracował w czasie podobnym do czasu pracy algorytmów Steepest descent oraz Greedy, ale otrzymywał średnio gorsze rezultaty niż o wiele szybszy heurystyczny. Wyjątkami od tej reguły są zadania o wielkościach 14 i 26, jednak Steepest descent i Greedy na tych zadaniach osiągają prawie zawsze optymalne rozwiązania. Podsumowując, uważamy, że algorytm losowy może służyć jako baza do porównania jakości działania bardziej rozbudowanych algorytmów, bowiem dobrze obrazuje rodzaj przestrzeni rozwiązań. Jeżeli znalezienie rozwiązania jest proste, to jego wartość średnia jest wyższa niż dla bardziej skomplikowanych krajobrazów funkcji kosztu.

Algorytmy Greedy i Steepest descent zachowują się bardzo podobnie. Zauważyliśmy, że Greedy wykonuje wielokrotnie więcej kroków, ale jego czas pracy jest zawsze krótszy. Jakość osiąganych przez niego rozwiązań jest porównywalna z jakością wyników algorytmu Steepest descent. Z uwagi na te wyniki uważamy, że należy uznać algorytm Greedy za lepszy od algorytmu Steepest, ponieważ osiąga praktycznie jednakowe wyniki w znacząco krótszym czasie.

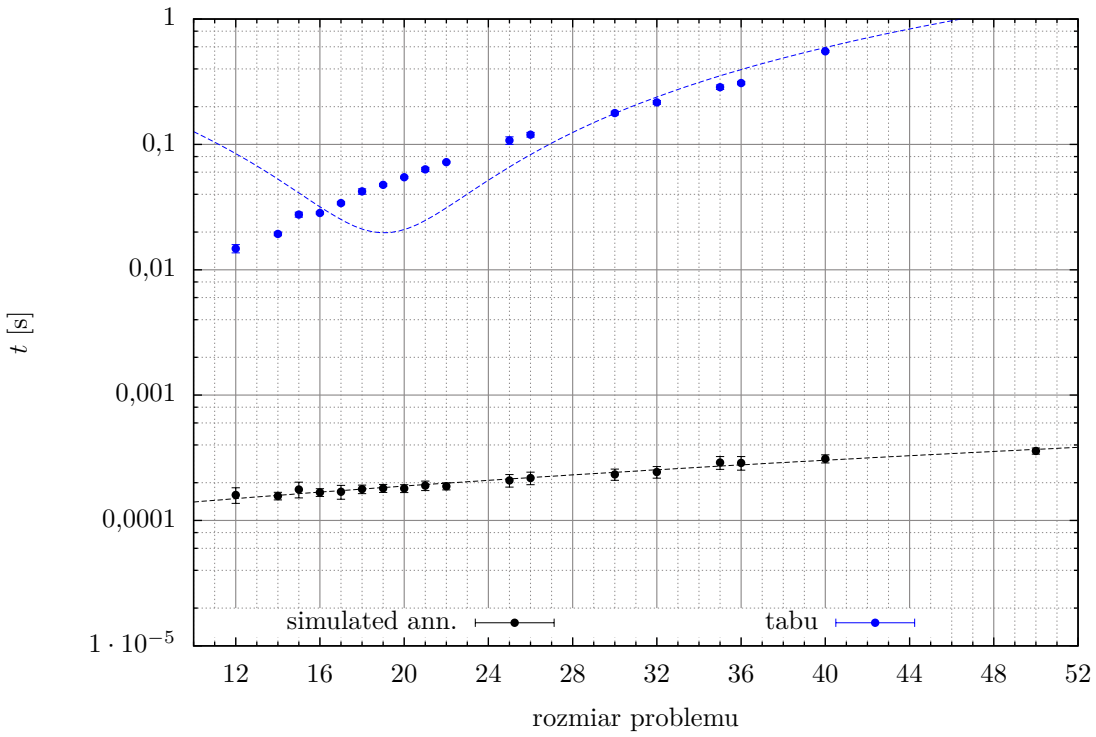
10 METAHEURYSTYKI



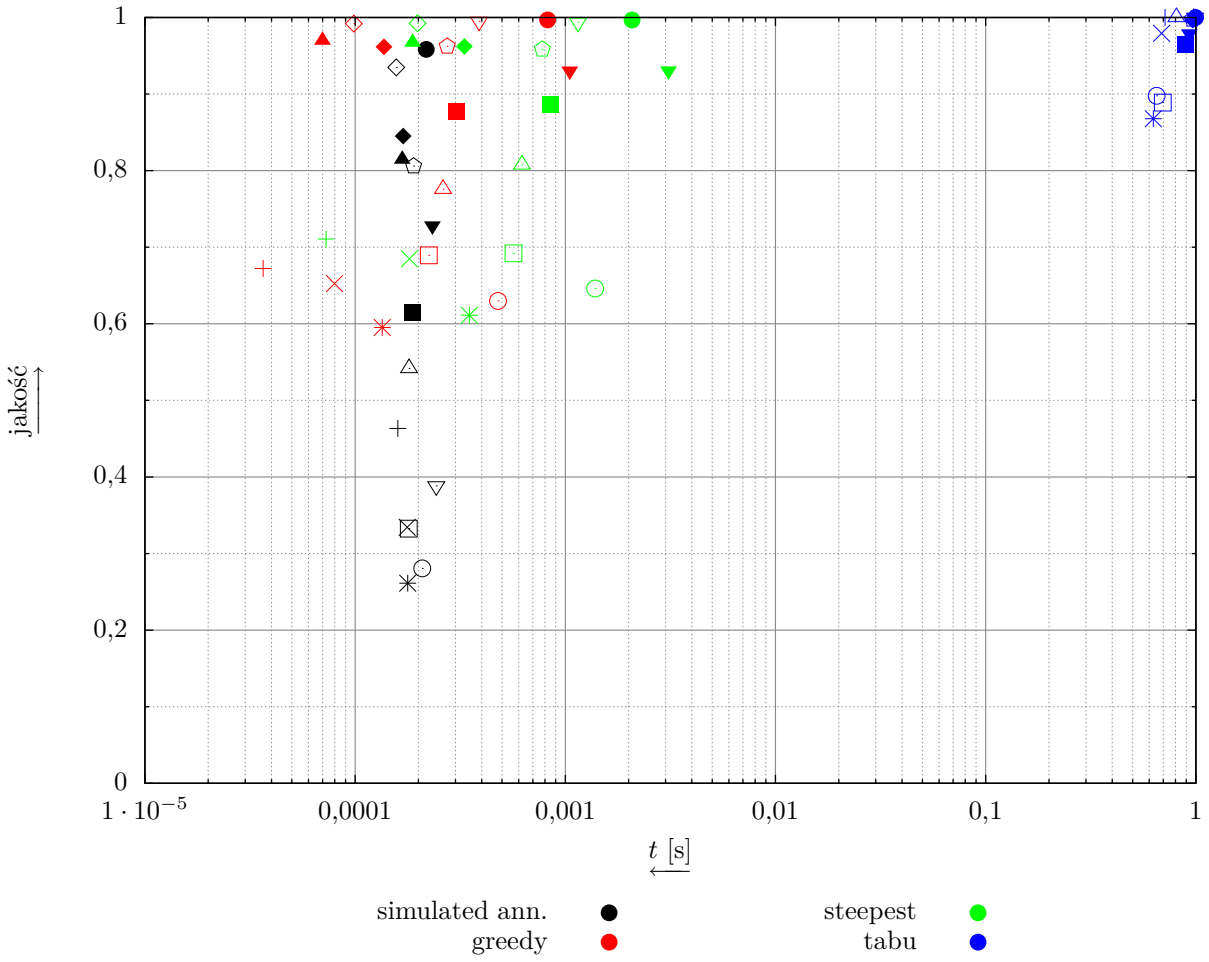
Rysunek 13: Wykres uśrednionej jakości rozwiązania od rozmiaru problemu



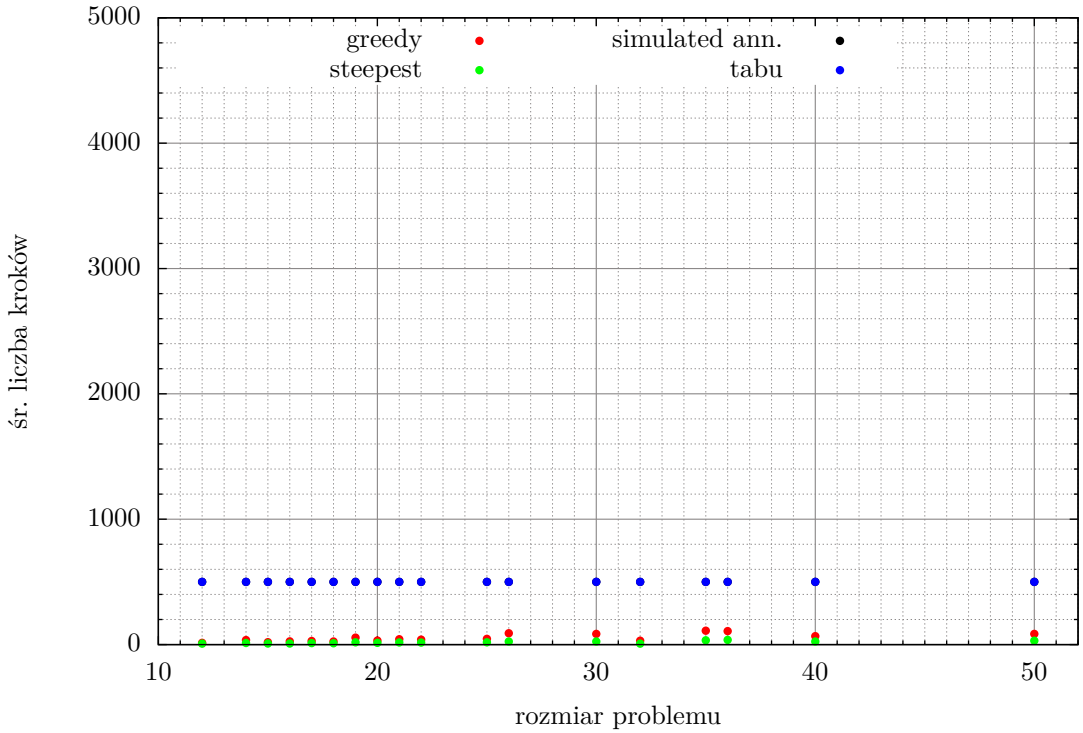
Rysunek 14: Wykres jakości najlepszego znalezionego rozwiązania od rozmiaru problemu



Rysunek 15: Wykres uśrednionego czasu wykonywania od rozmiaru problemu



Rysunek 16: Wykres jakości rozwiązania w funkcji czasu rozwiązania dla różnych problemów



Rysunek 17: Wykres liczby kroków od rozmiaru problemu