Sprawozdanie z laboratorium:
Metaheurystyki i Obliczenia Inspirowane Biologicznie

Część I: Algorytmy optymalizacji lokalnej, problem QAP

28 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.

1.3.5 Simulated Annealing

Algorytm symulowanego wyżarzania działa podobnie do algorytmu Greedy. Różnica polega na tym, że rozwiązanie zostanie zaakceptowane z prawdopodobieństwem zależnym od jego kosztu i aktualnej temperatury. Funkcja prawdopodobieństwa wyboru rozwiązania ma postać $P(koszt, T) = e^{\frac{minKoszt - koszt}{T}}$, gdzie $minKoszt$ to koszt najlepszego znalezionego rozwiązania, a $koszt$ to koszt aktualanie ocenianego rozwiązania. Dla każdego rozwiązania losowana jest liczba z przedziału 0-1. Jeżeli ma ona wartość mniejszą od obliczonego prawodopodobieństwa, rozwiązanie jest akceptowane. W przeciwnym wypadku sprawdzany jest kolejny sąsiad. Kształt tej funkcji gwarantuje wybór rozwiązania o mniejszym koszcie z prawdopodobieństwem równym 100%. Warunkami stopu są maksymalna liczba kroków i brak poprawy wyniku przez liczbę kroków równą dziesięciokrotności rozmiaru sądziedztwa.

1.3.6 Tabu Search

Algorytm Tabu Search ocenia wszystkich sąsiadów, a następnie wybiera spośród nich $\frac{rozmiarProblemu}{10}$ rozwiązań. Zdecydowaliśmy się dla mniejszych problemów (rozmiar poniżej 40) na stałe wpisać wartość 3 jako liczbę najlepszych sąsiadów branych pod uwagę. Najlepsi sąsiedzi są sortowani, a następnie porównywani (w porządku rosnącym względem kosztu) z aktualnym najlepszym rozwiązaniem. Jeżeli nowe rozwiązanie jest lepsze, to jest ono zapisywane jako najlepsze i aktualne oraz dodawane do listy tabu. Jeżeli rozwiązanie jest gorsze i nie jest w liście tabu, to jest zapisywane jako aktualne i dodawane do listy tabu. Jeżeli żadne z rozwiązań nie było lepsze od najlepszego i wszystkie były w tabu, to algorytm kończy działanie. Warunkami stopu są także maksymalna liczba iteracji, liczba iteracji bez znalezienia lepszego rozwiązania lub brak możliwości ruchu z powodu ograniczeń tabu. Rozmiar tabu dobierany jest jako $\frac{rozmiarProblemu}{4}$.

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
15. tai35b
16. ste36c
17. lipa40a
18. lipa50a

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.

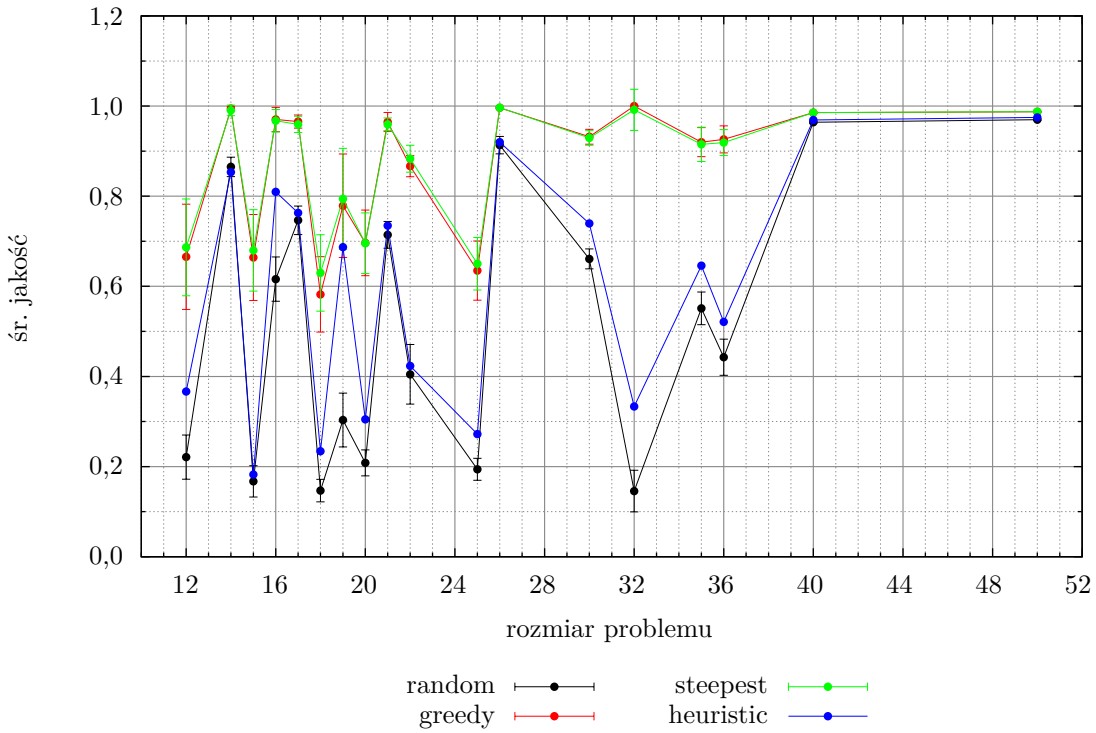
Algorytmy metaheurystyczne zostały uruchomione 30 razy, a wyniki i parametry działania zagregowane. Algorytm losowy został uruchomiony tyle razy, by czas jego działania odpowiadał rządem wielkości czasowi działania algorytmów Greedy i Steepest Descent.

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

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. Wykresy z Rys. 1 i Rys. 2 przedstawiają 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.



Rysunek 1: Wykres uśrednionej jakości rozwiązania od rozmiaru problemu

Z wykresu z Rys. 1 odczytujemy, że średnia jakość działania algorytmu Random jest dla większości problemów niska i nie przekracza wartości 0,9. Dla 9 z 18 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. Problemy lipa40a i lipa50a mają jedną z macierzy wypełnioną w większości jedynkami, a pozostałe liczby to 0 i 2, co sprawia, że nawet algorytmy losowy i heurystyczny znajdują tam bardzo dobre rozwiązania.

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

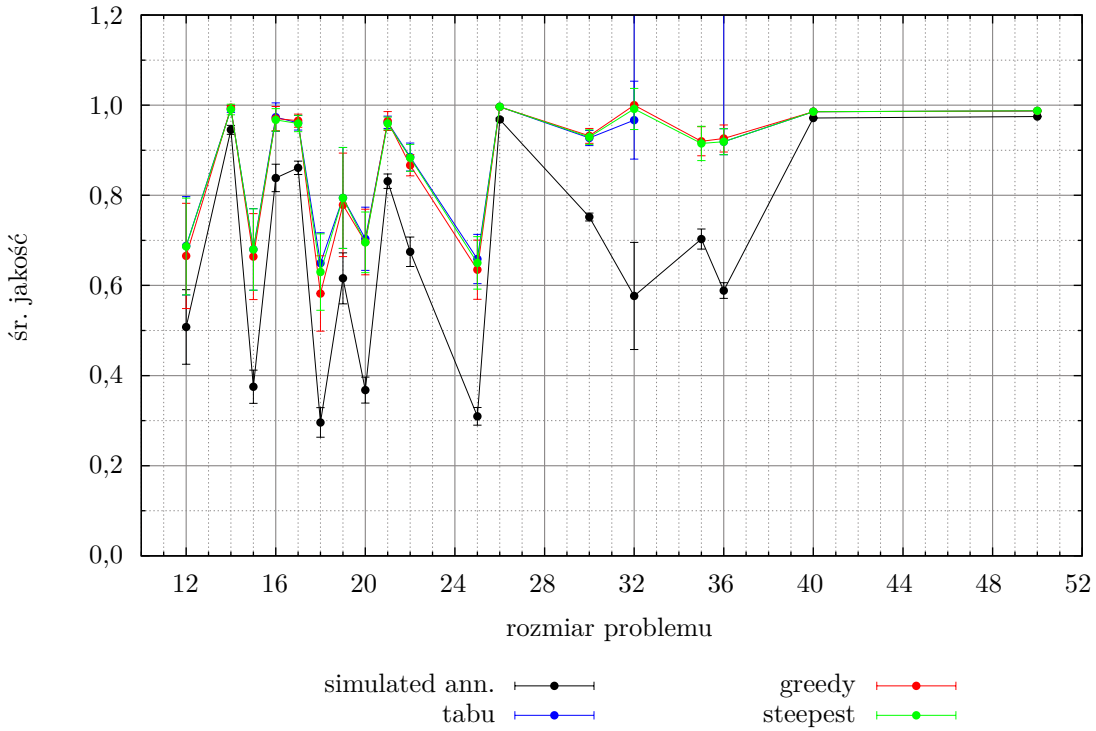
Średnie rozwiązania dla 30 uruchomień dla algorytmów Steepest Descent oraz Greeedy mają bardzo podobne, nierozróżnialne wartości.

Z wykresu z Rys. 2 odczytujemy, że Tabu Search osiąga wyniki porównywalne ze Steepest Descent oraz Greedy. Zdecydowanie gorszy okazuje się algorytm Simulated Annealing, którego wyniki są podobne do działania algorytmu losowego. Może to wynikać ze źle dobranych parametrów symulacji.

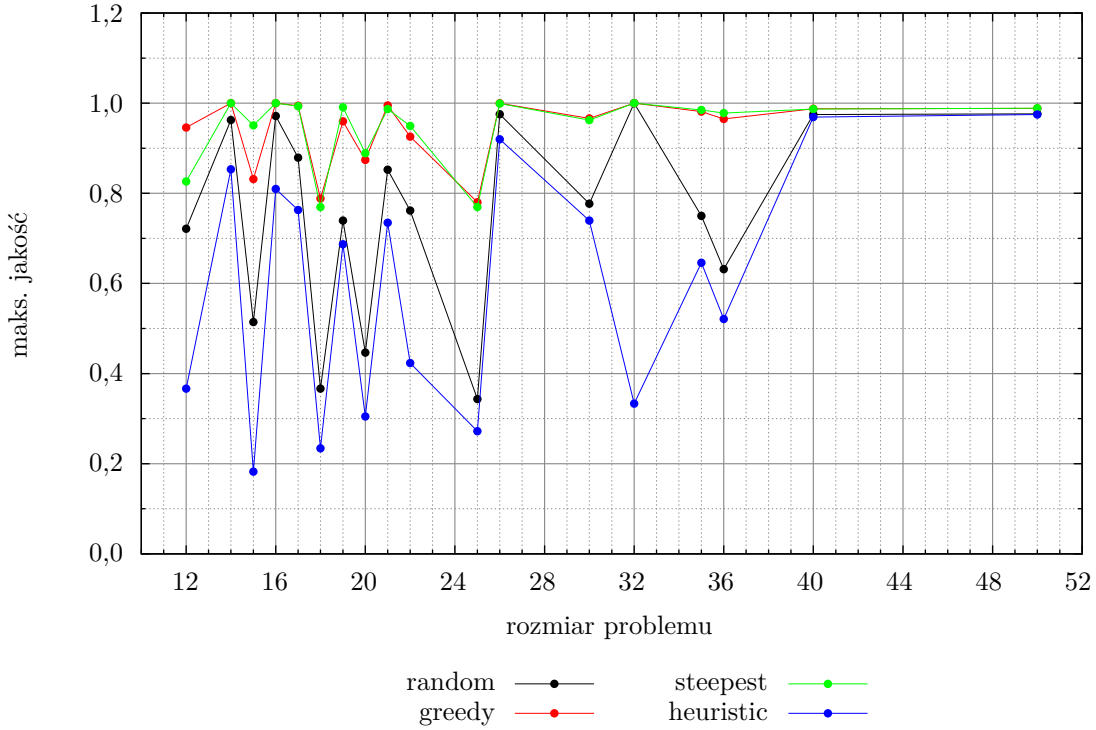
2.2.2 Jakość najlepszego znalezionego rozwiązania

Wykresy z Rys. 3 i z Rys. 4 przedstawiają jakość najlepszego rozwiązania, znalezionego przez algorytmy, dla poszczególnych instancji.

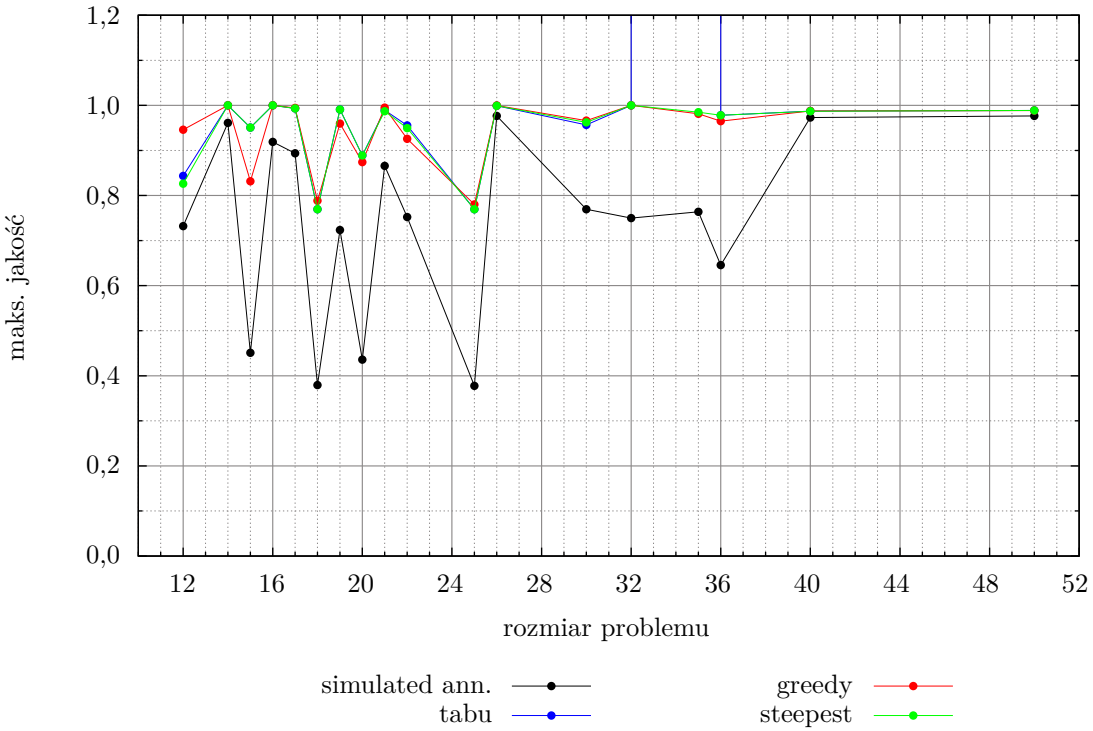
Algorytmy Greedy i Steepest Descent zwracają bardzo podobne wyniki. Jeżeli udało się znaleźć optymalne rozwiązanie, to oba algorytmy je znajdowały. Dla problemów, gdzie nie udało się znaleźć optymalnego rozwiązania, Steepest Descent osiągał lepsze rezultaty (np. instancje o wielkości 15, 22, 25). Algorytm Tabu działa porównywalnie do Steepest Descent i Greedy.



Rysunek 2: Wykres uśrednionej jakości rozwiązania od rozmiaru problemu



Rysunek 3: Wykres jakości najlepszego znalezionego rozwiązania od rozmiaru problemu



Rysunek 4: Wykres jakości najlepszego znalezionego rozwiązania od rozmiaru problemu

3 Czas działania w zależności od rozmiaru problemu

3.1 Random, Greedy, Steepest Descent i Simulated Annealing

Na wykresie z Rys. 5 przedstawiona jest średnia wartość czasu potrzebnego na znalezienie rozwiązania dla 30 uruchomień algorytmów Random, Greedy, Steepest Descent i Simulated Annealing oraz 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 algorytmy 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.

3.2 Algorytm heurystyczny

Na podstawie wykresu dla algorytmu heurystycznego, przedstawionego na rys 6, 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.

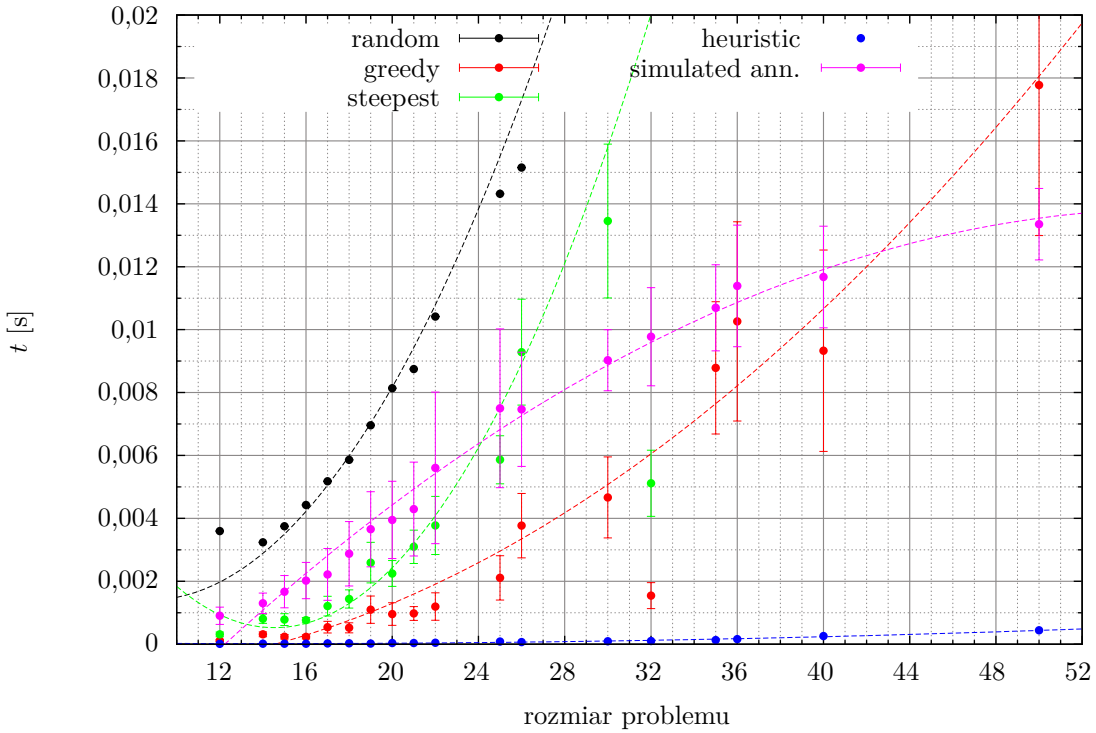
Czas obliczenia rozwiązania nie pokrywa się dokładnie z przybliżoną krzywą, bo w zależności od charakteru problemu, kolejne iteracje mogą potrzebować mniejszej liczby operacji na pamięci dla znalezienia minimalnej wartości przepływu (np. gdy wszystkie pozostałe do dyspozycji wartości są sobie równe –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

3.3 Tabu Search

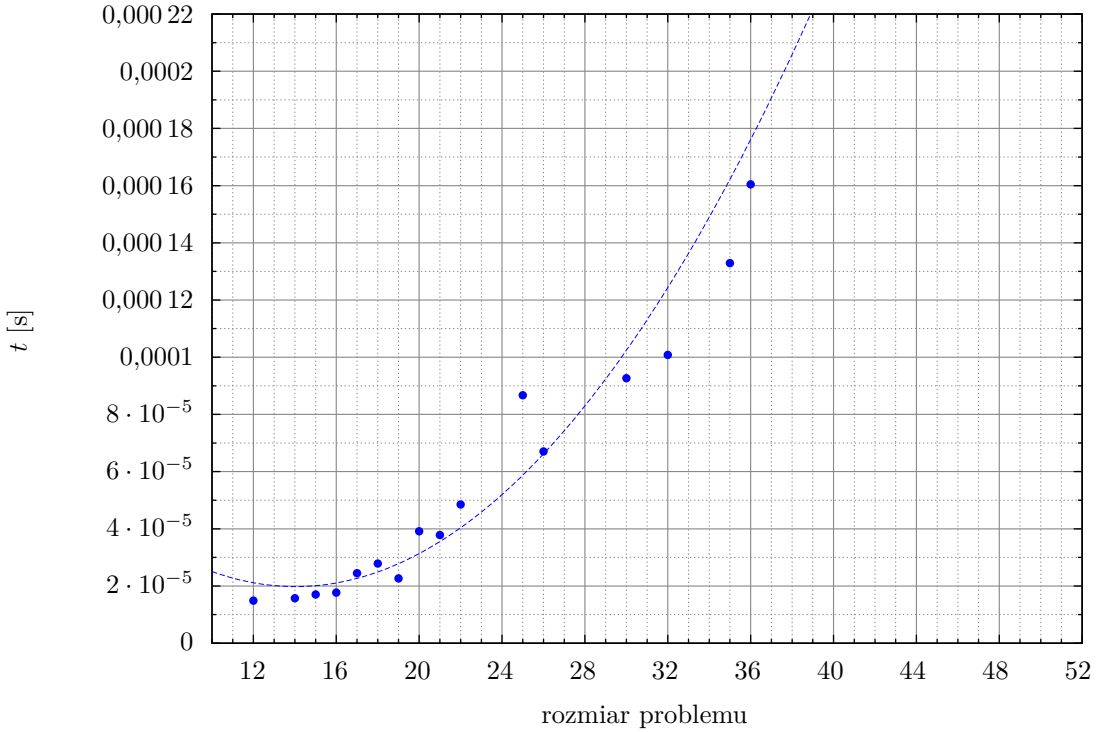
Implementacja Tabu Search wymagała oceny każdego z sąsiadów przy każdej iteracji, co okazało się bardzo kosztowne (ilustruje to wykres z Rys. 7). Algorytm wykonywał się wielokrotnie dłużej niż pozostałe, co czyni go w obecnej implementacji dużo mniej przydatnym niż zwykły local search, bowiem na analizowanych przykładach nie widać znaczącej poprawy jakości rozwiązania.

4 Jakość rozwiązania w zależności od czasu działania

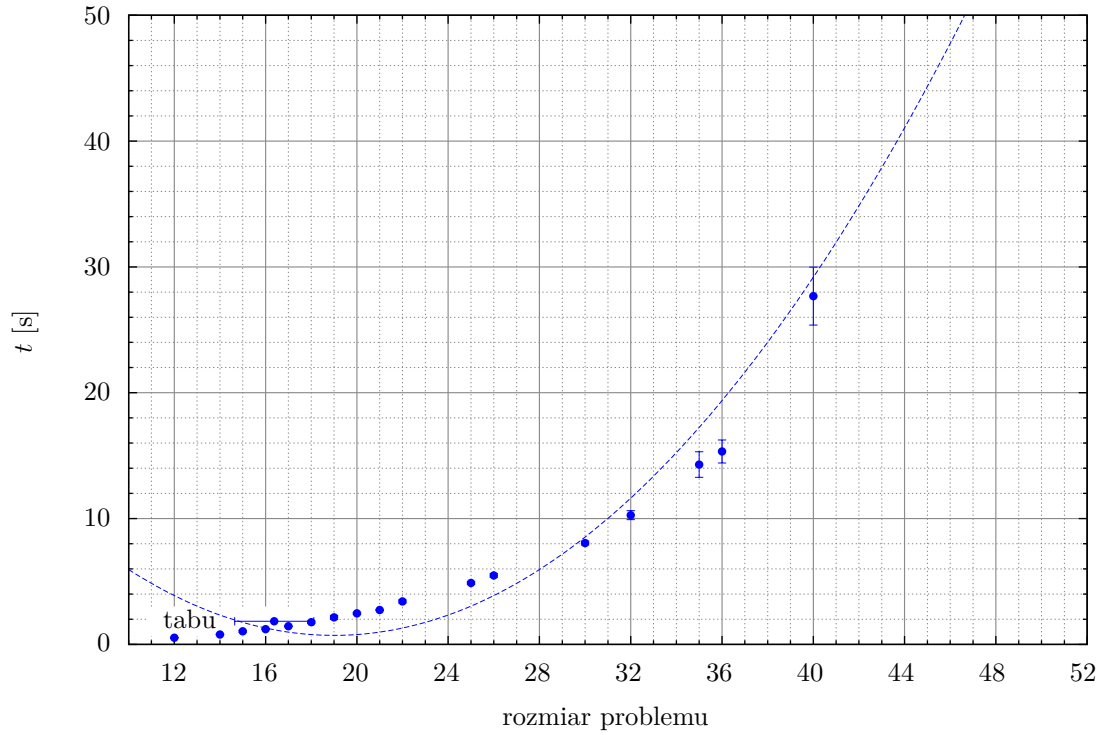
Na Rys. 8 i 9 przedstawiamy dwa wykresy, które pozwalają porównać algorytmy na podstawie czasu ich pracy i osiąganych rozwiązań dla 4 instancji problemów Wykresy z Rys. 9 i Rys. 10 zawierają punkty dla wszystkich instancji. Problemowi przypisany jest kształt znacznika, kolor indentyfikuje natomiast algorytm, który osiągnął dany wynik.



Rysunek 5: Wykres uśrednionego czasu wykonywania od rozmiaru problemu



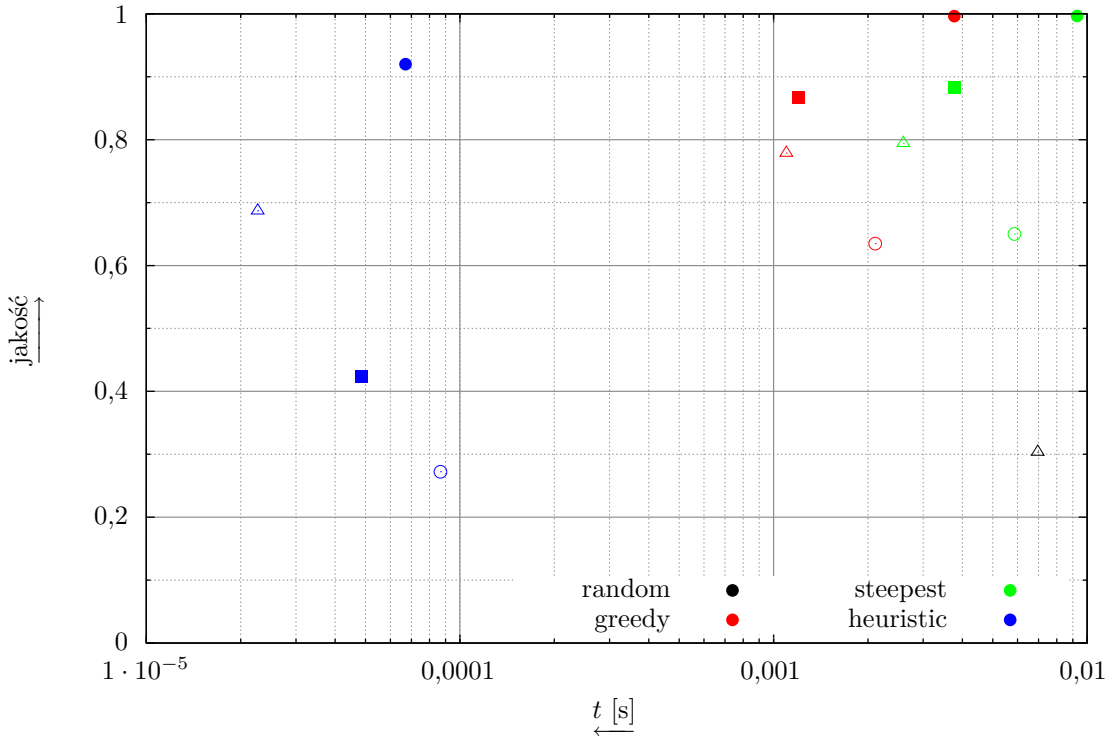
Rysunek 6: Wykres czasu wykonywania alg. heurystycznego od rozmiaru problemu



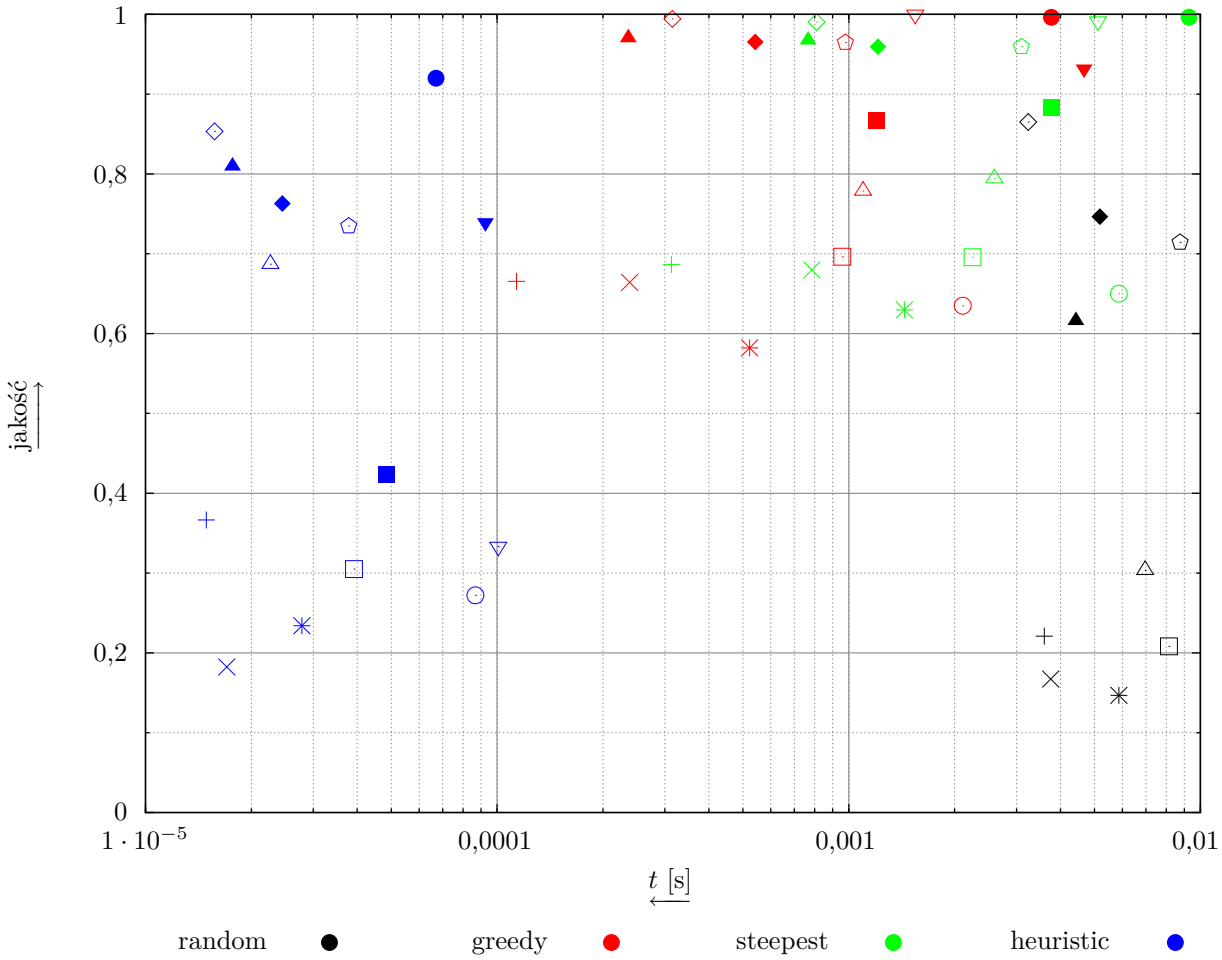
Rysunek 7: Wykres czasu wykonywania alg. Tabu Search od rozmiaru problemu

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, Steepest i Tabu Search osiągają porównywalne wyniki pod względem jakości. Greedy potrzebuje więc mniej czasu na osiągnięcie podobnego co Steepest rezultatu. Tabu Search pracuje kilkadziesiąt do kilkuset razy wolniej. Rozwiązania znajdowane przez Simulated Annealing są zdecydowanie gorsze niż przez pozostałe algorytmy. W porównaniu z Greedy czy Steepest Descent rozwiązania znajdowane przez SA nie są atrakcyjne (ze względu na czas pracy).



Rysunek 8: Wykres jakości rozwiązania w funkcji czasu rozwiązania dla wybranych problemów



Rysunek 9: 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. 11 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.

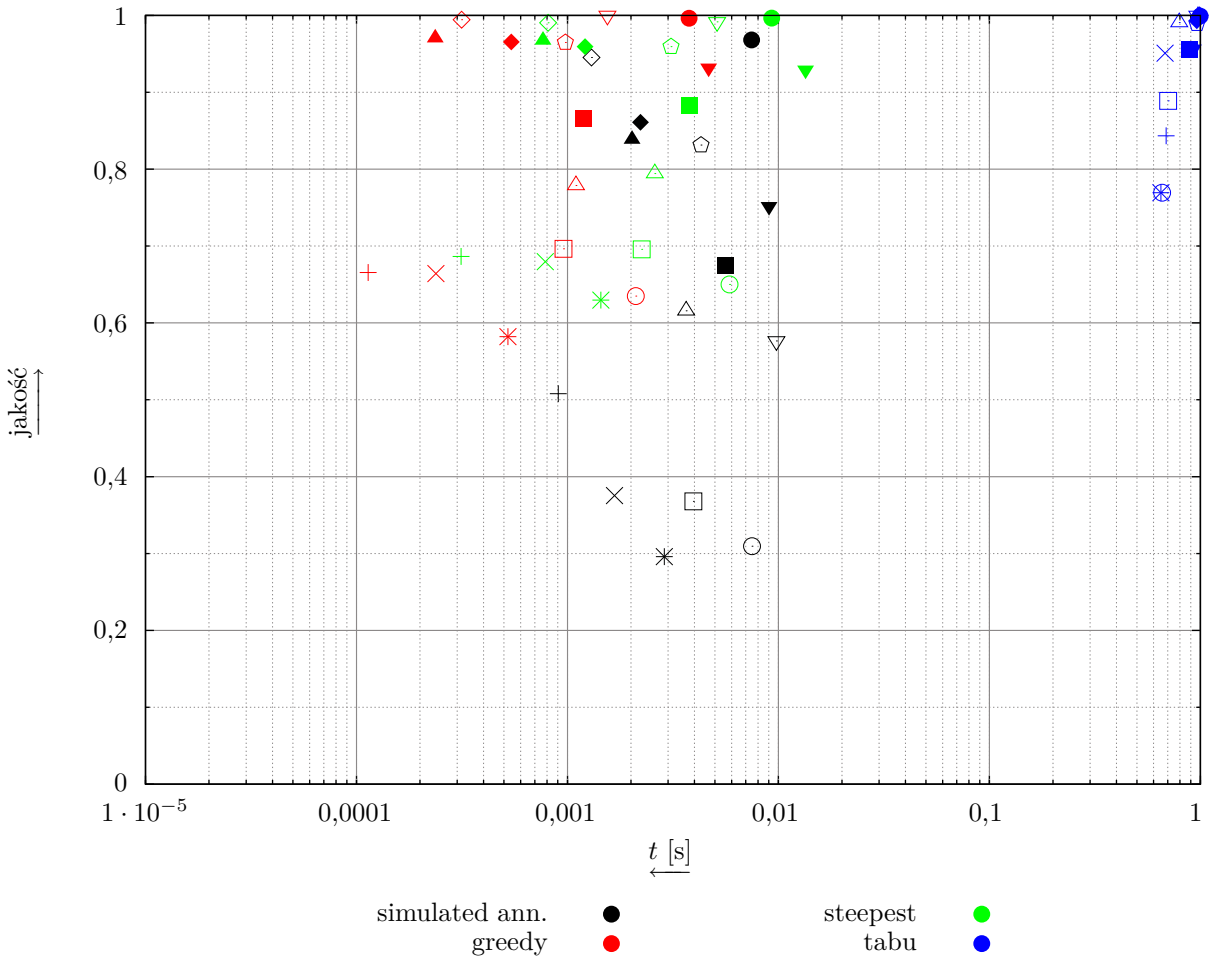
Biorąc pod uwagę, że dla tych problemów znalezione zostały rozwiązania optymalne (had14 i bur26a) lub bardzo bliskie optimum (els19), a odchylenia standardowe liczby kroków są bardzo małe, wnioskujemy, że krajobraz zawiera mniej lokalnych wypłaszczeń czy lokalnych minimów i bliższy jest sytuacji z globalną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

Wyniki działania algorytmów TS i SA przedstawione są na wykresie z Rys.12. Liczba kroków potrzebnych przez algorytm SA do zgłoszenia rozwiązania jest liniowo zależna od rozmiaru problemu. Może to wynikać z faktu, że temperatura spada po liczbie kroków zależnej od rozmiaru problemu. Przy pewnej temperaturze prawdopodobieństwo przejścia jest niewielkie i algorytm zatrzymuje się. Tabu Search praktycznie zawsze osiągał maksymalną liczbę (5000) kroków. Może to wynikać z narzuconych warunków - brak poprawy przez liczbę kroków równą dziesięciokrotności rozmiaru sąsiedztwa, która to liczba przewyższała odgórne ograniczenie na 5000 kroków.

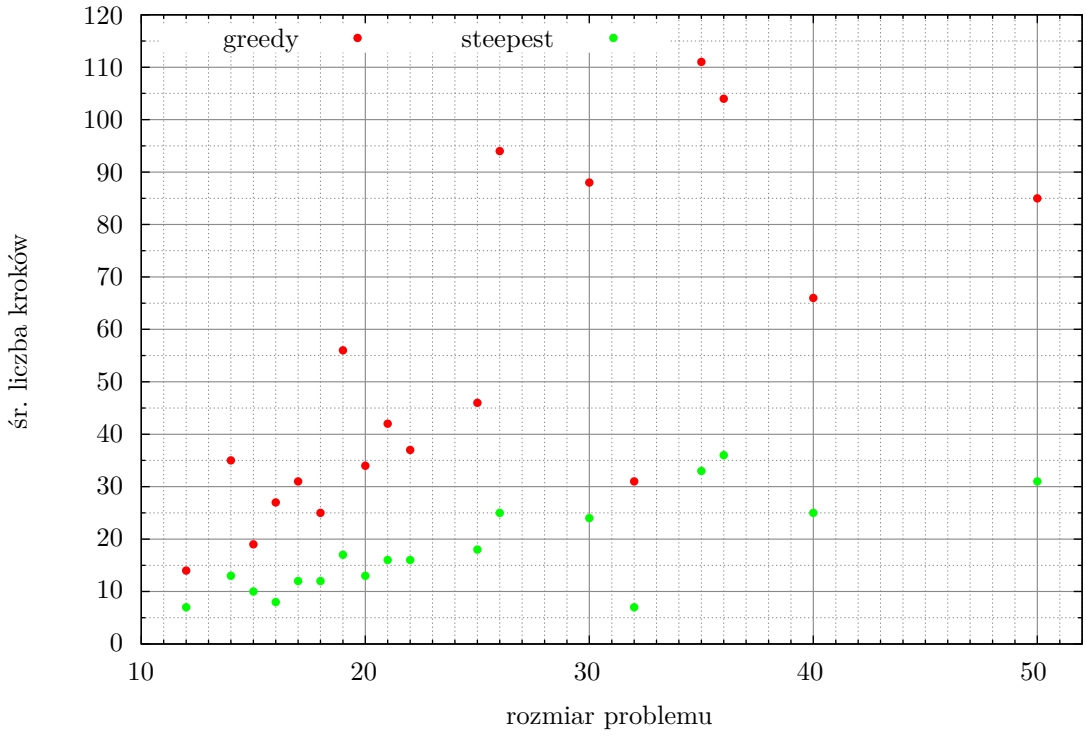
6 Jakość rozwiązania najlepszego w zależności od początkowego

Z wykresów przedstawionych na Rys. 13 i 14 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.

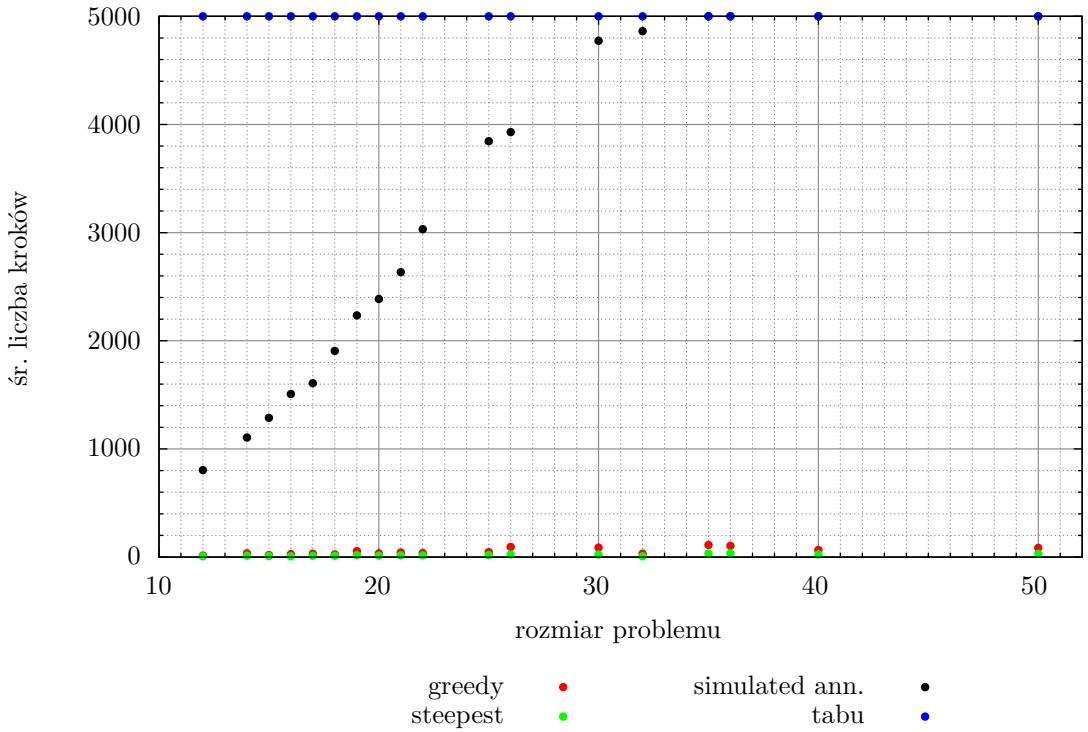
Na wykresie z Rys. 14 widać, że dla większości problemów jakość początkowego rozwiązania waha się między 0,08 a 0,22, a jedynie dla dwóch przykładowych problemów są to wartości powyżej 0,4, a jakość znalezionego rozwiązania końcowego dla większości problemów waha się dość istotnie. Trudno zatem ocenić, 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



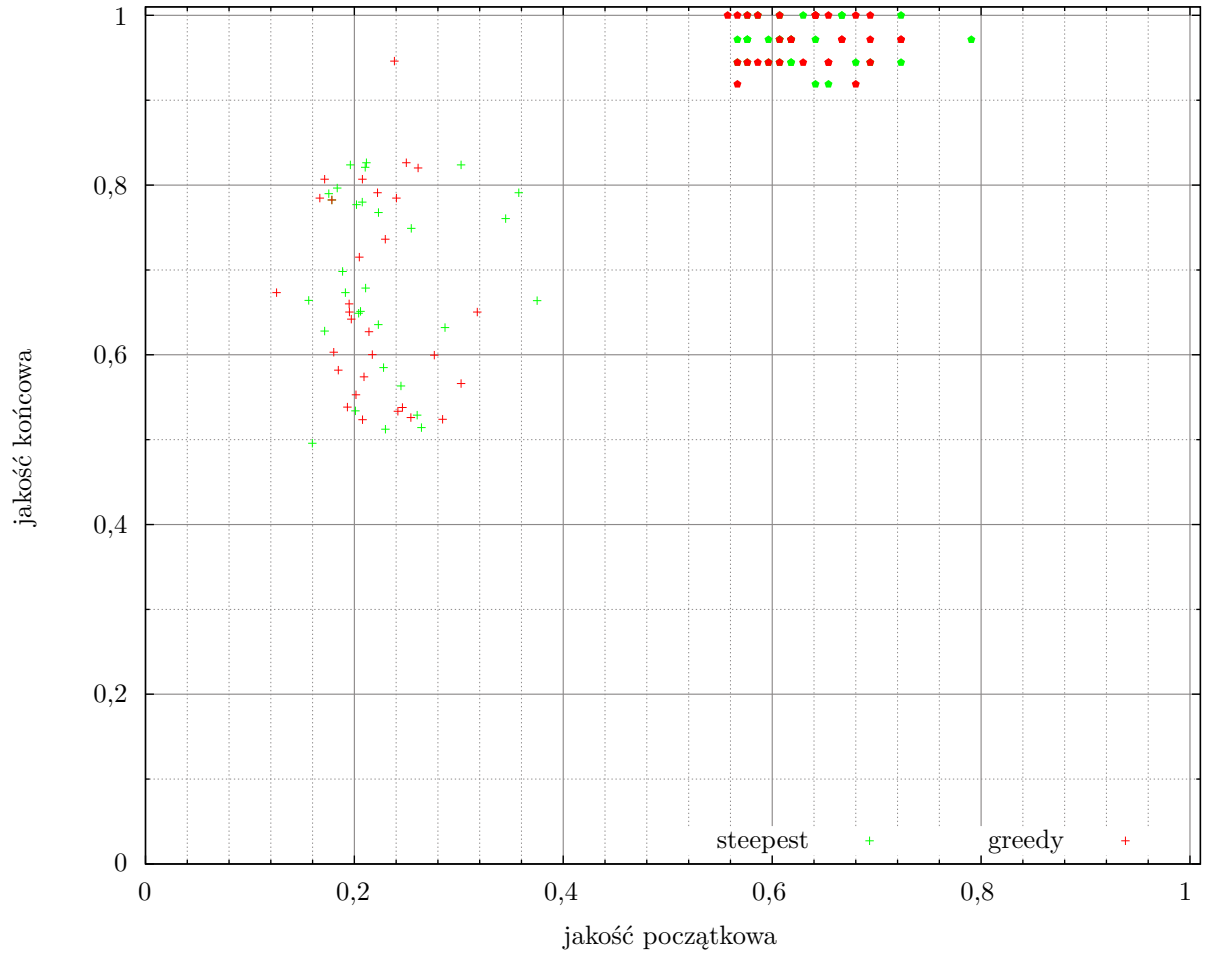
Rysunek 10: Wykres jakości rozwiązania w funkcji czasu rozwiązania dla różnych problemów



Rysunek 11: Wykres liczby kroków od rozmiaru problemu

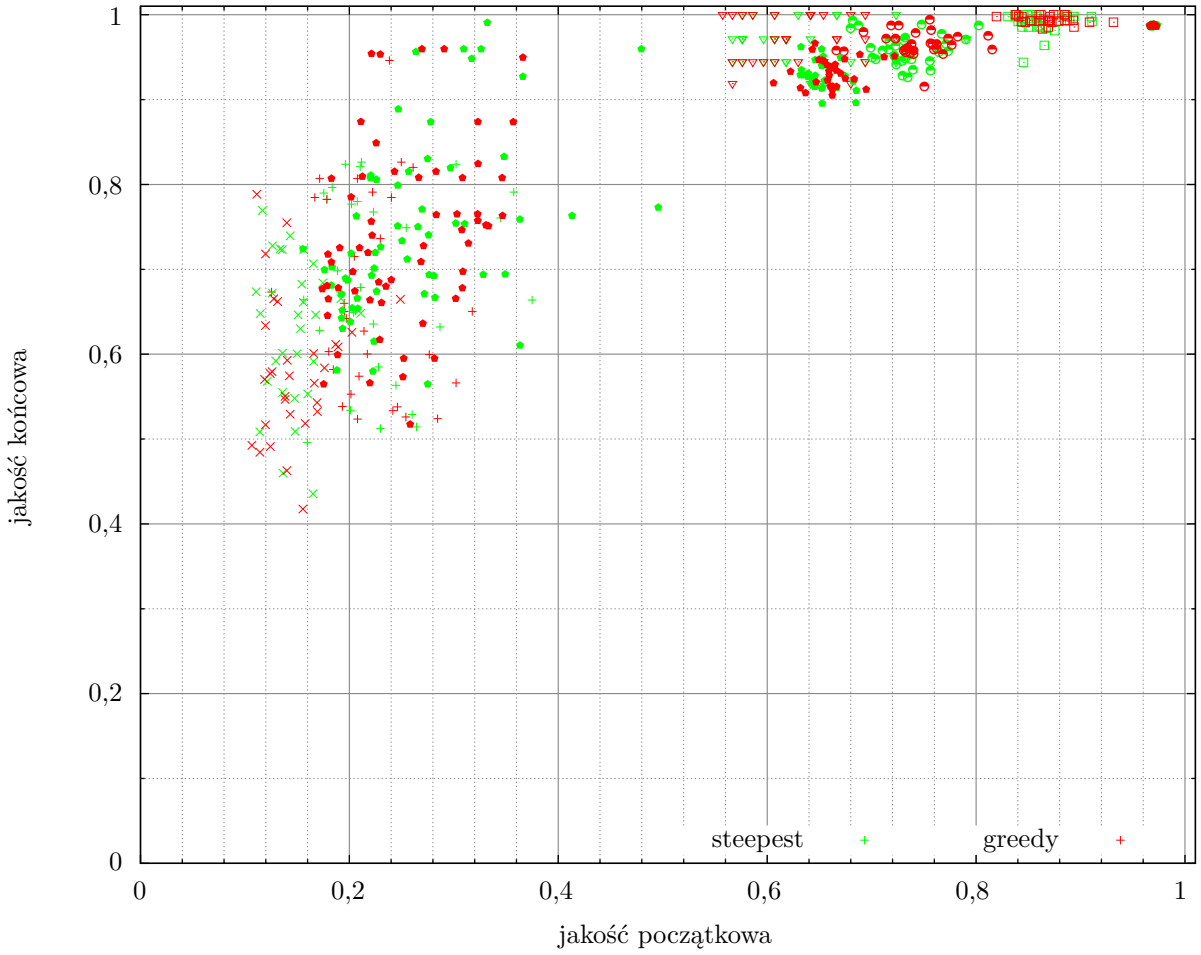


Rysunek 12: Wykres liczby kroków od rozmiaru problemu



Rysunek 13: Wykres jakości rozwiązania najlepszego od rozwiązania początkowego dla wybranych problemów

problemów wynika tylko z ich charakteru.



Rysunek 14: Wykres jakości rozwiązania najlepszego od rozwiązania początkowego dla różnych problemów

7 Poprawa jakości znalezionego rozwiązania w kolejnych krokach

Wykres na Rys. 15 przedstawia jaką wielkość miary jakości miały rozwiązania w kolejnych krokach algorytmu Steepest Descent dla problemu kra30a. Wykres z Rys. 16 przedstawia analogiczne dane dla algorytmu Greedy. Oba algorytmy zostały uruchomione 30 razy.

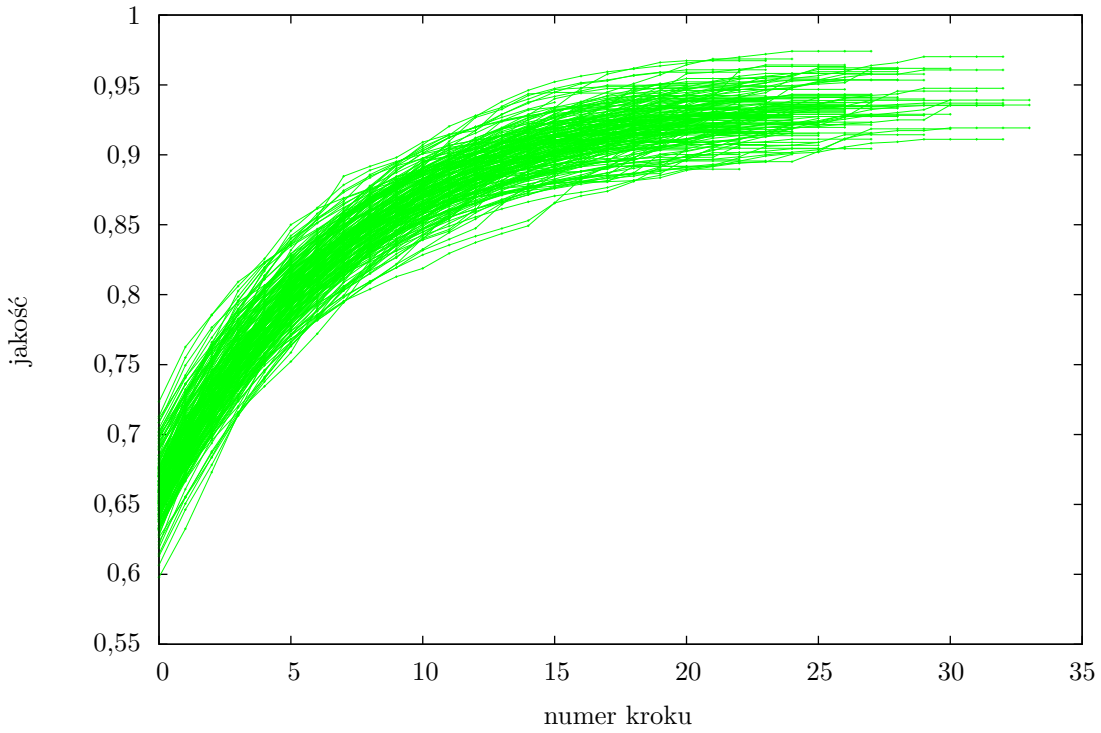
Z wykresu 15 widać, że Steepest Descent dość szybko dochodzi do rozwiązania (liczba kroków bliska wielkości problemu), krzywa poprawy jakości ma charakter eksponencjalny – jakość wyraźnie, coraz wolniej z czasem, poprawia się do pewnej wartości granicznej bliskiej optimum. Nie zdarza się raczej, by algorytm 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 15 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.

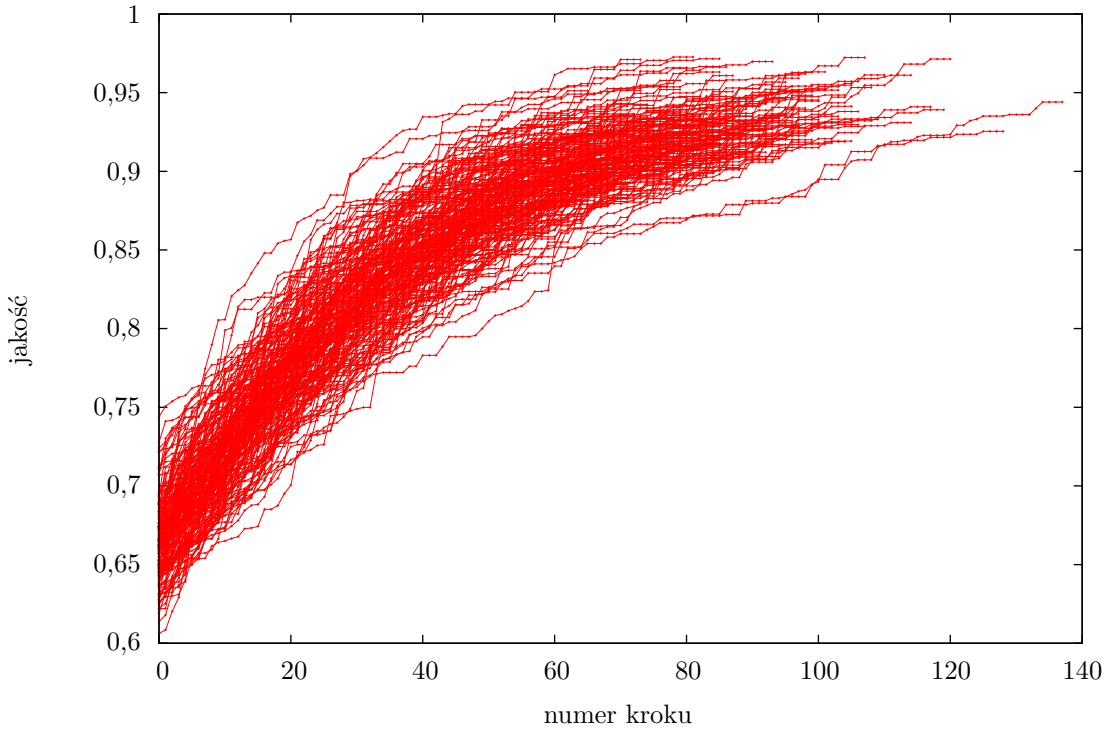
Oba algorytmy mają dość duży rozrzut jakości rozwiązania początkowego (startują od podobnych rozwiązań losowych, o jakościach od 0,6 do ok. 0,75, różnica 0,15), i oba zbiegają się do niezbyt mocno rozrzuconych pod względem miary jakości rozwiązań końcowych – w analizowanym przypadku wartości między 0,9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

8 Zależność jakości od liczby powtórzeń

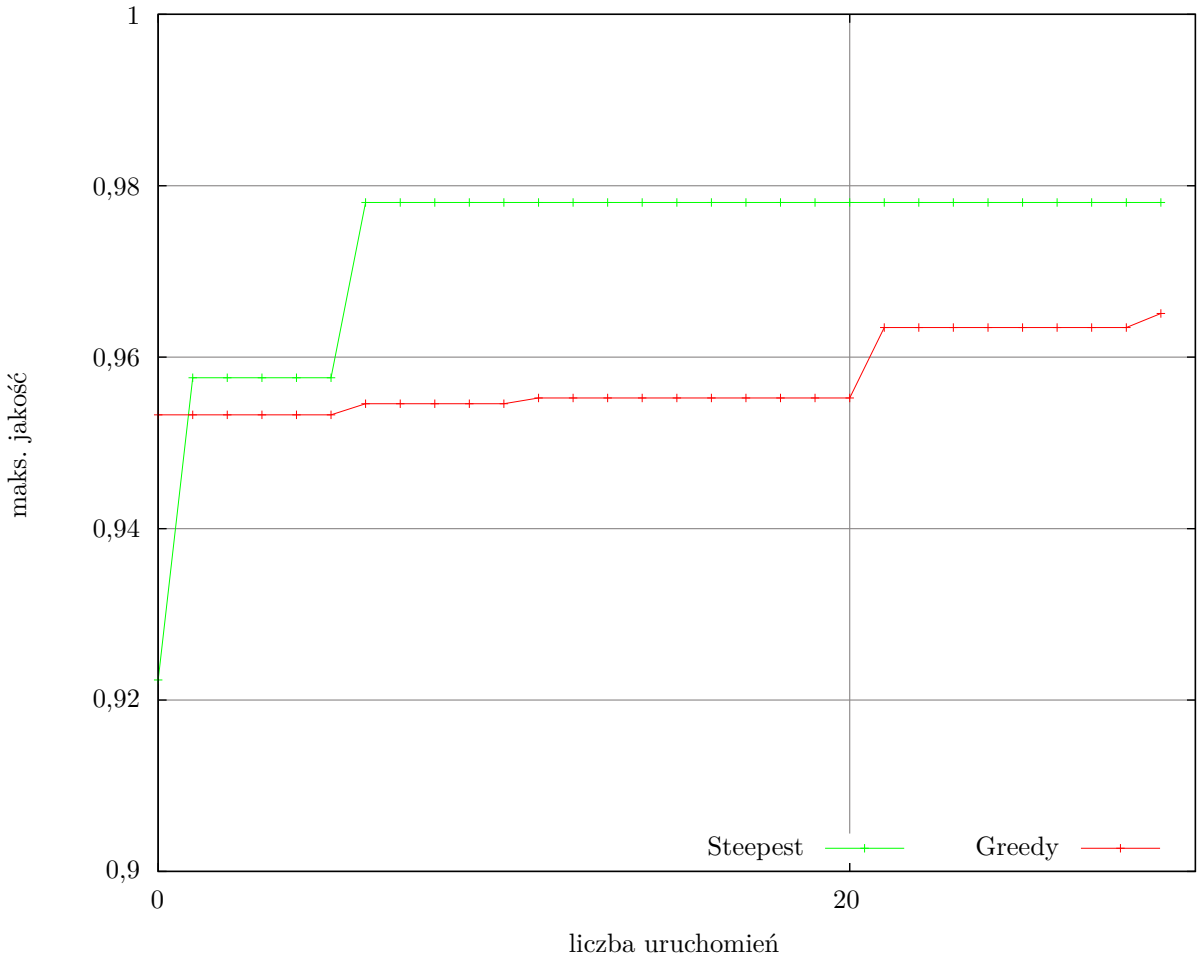
Z wykresu na Rys. 17 widać, że...POPRAWIĆ



Rysunek 15: Wykres jakości rozwiązania w kolejnych krokach 30 przebiegów algorytmu Steepest Descent



Rysunek 16: Wykres jakości rozwiązania w kolejnych krokach 30 przebiegów algorytmu Greedy



Rysunek 17: Wykres maksymalnej jakości rozwiązania dla kolejnych 30 uruchomień algorytmów Greedy i Steepest Descent na problemie ste36c

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.

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.

Wyniki zwracane przez algorytm Tabu Search są bardzo podobne do wyników działania Greedy i Steepest Descent. Ze względu na dobór funkcji jakości jako ilorazu $\frac{kosztOptymalnegoRozwizania}{kosztZnalezionegoRozwizania}$ na podstawie wykresów trudno ocenić różnicę pomiędzy znajdowanymi rozwiązaniami. Czas działania jest zdecydowanie dłuższy ze względu na konieczność oceny wszystkich sąsiadów. Żaden z analizowanych przypadków nie był na tyle „interesujący", aby Tabu Search znalazł lepszy wynik od podstawowych algorytmów local search.

Algorytm Simulated Annealing zwracał rozwiązania z kosztem przewyższającym algorytmy Greedy oraz Steepest, a czasami nawet rozwiązania znalezione przez algorytm losowy były lepsze. Winą obciążamy zły dobór parametrów. Trudno ocenić, która część zawiniła: inicjacja temperatury, prędkość wygaszania czy sama funkcja oceny prawdopodobieństwa.