Sprawozdanie z ćwiczenia laboratoryjnego

Tomasz Mikołajewski

2014

Spis treści

1 Wprowadzenie

Niniejszy dokument powstał w ramach przedmiotu Projektowanie Algorytmów i Metod Sztucznej Inteligencji. Jest on rezultatem przeprowadzonych ćwiczeń w laboratorium oraz pracy wykonanej w domu.

Ćwiczenie polegało na znalezieniu ścieżki w grafie. Do wykonania postawionego zadania należało wykorzystać algorytm best-first-search. Celem wykonanego zadania było również określenie skuteczności wymienionego algorytmu. Przy okazji wykonywania ćwiczenia zaimplementowano 3 różne warianty algorytmu i badano ich skuteczność.

Uwaga: Dwa pozostałe algorytmy (depth-first-search i breadth-first-search) zostały już dokładnie opisane we wcześniejszym sprawozdaniu. Można je znaleźć w pliku grafy.pdf pod adresem:
https://github.com/Mikolajewski/Projekt1/tree/master/Pamsi/pamsi7

2 Kod programu

Program został oparty o wcześniej stworzone pliki zawierające kod potrzebny do wykonania *benchmark-u* podprogramu, czyli określenia czasu jaki potrzebny jest na jego wykonanie. Główny program zawiera funkcję otwierającą pliki z danymi oraz zapisującą rezultat operacji do pliku. Dodatkowo do programu wykonującego sprawdzenie algorytmów dodano metodę automatycznie tworzącą plik z danymi potrzebnymi do sprawdzenia algorytmu. Wszystkie ustawienia można zmieniać w pliku *konfiguracja.hh*. Zdecydowanie ułatwia to prace przy analizie zadanej struktury.

Zaimplementowano trzy wersje badanego algorytmu:

- Wyszukiwanie zawsze wybierające ścieżkę o najniższym koszcie.
- Wyszukiwanie bez rozważania wagi ostatniego kroku.
- Wyszukiwanie najlepszej drogi, czyli o najniższym koszcie.

Wszystkie wersje algorytmu zostały opisane w dalszej części sprawozdania.

3 Pomiary

Pomiary zostały podzielone na dwie części. Pierwszą z nich było wykonanie symulacji na małych ilościach danych (rzędu kilkudziesięciu) i sprawdzenie, jak radzą sobie poszczególne algorytmy w konkretnych sytuacjach. W tej części otrzymane wyniki były porównywane z grafami, które zostały wygenerowane przez program. Druga część pomiarów polegała na próbie określenia złożoności algorytmu. W tej części testowana była pierwsza z wersji. Na początku wykonano pomiar mający na celu sprawdzenie jaki wpływ na działanie algorytmu ma gęstość grafu. Następnie wykonano pomiary czasu wykonania algorytmu. Z racji dużej losowości pojedynczych prób, wynikającej z losowości powstających grafów, wykonano dużą ilość testów dla każdego z pomiarów (50). W ten sposób otrzymano rzetelne wyniki.

4 Wyniki pomiarów i obserwacje

Po przeprowadzeniu serii pomiarów otrzymano wyniki przedstawione w tabelach. Na podstawie wyników utworzono wykresy zamieszczone poniżej. Na końcu sprawozdania znajdują się wnioski wyciągnięte z załączonych wyników pomiarów.

4.1 Wyszukiwanie zawsze wybierające ścieżkę o najniższym koszcie.

Jest to rodzaj wyszukiwania najprostszy pod względem założeń. Algorytm ten zawsze wybiera ścieżkę o najniższym koszcie, a jeśli obrana droga nie doprowadzi go do celu to wybierana jest kolejna ścieżka wychodząca z ostatniego węzła (lub w przypadku braku innych krawędzi do sprawdzenia z poprzedniego węzła). Algorytm ten wykorzystuje listę na której znajdują się kolejne możliwe kierunki ruchu w grafie. W programie, na podstawie którego zostało wykonywane sprawozdanie, użyto listy LIFO. Aby było to możliwe dodawano kolejne ścieżki od najmniej korzystnej do najbardziej korzystnej po każdorazowej zmianie wierzchołka. Rezultat algorytmu jest opisany na podstawie poniższego grafu (Rysunek 1).

Należało znaleźć drogę pomiędzy g i k.

Znaleziona droga to: g->f->o->u->a->p->k Koszt drogi to 342.

4.2 Wyszukiwanie bez rozważania wagi ostatniego kroku.

Algorytm ten kończy swoje działanie, gdy z wierzchołka w którym się znajduje osiągalny jest wierzchołek docelowy. Koszt ostatniego kroku jest dodawany do ogólnego kosztu całej drogi, lecz w tym miejscu następuje wyłamanie się z ogólnej zasady algorytmu polegającej na wybieraniu zawsze drogi o najniższym koszcie. Niesie to ze sobą konsekwencje omówione w dalszej części sprawozdania.

4.2.1 Sytuacje przy których algorytm okazał się skuteczniejszy

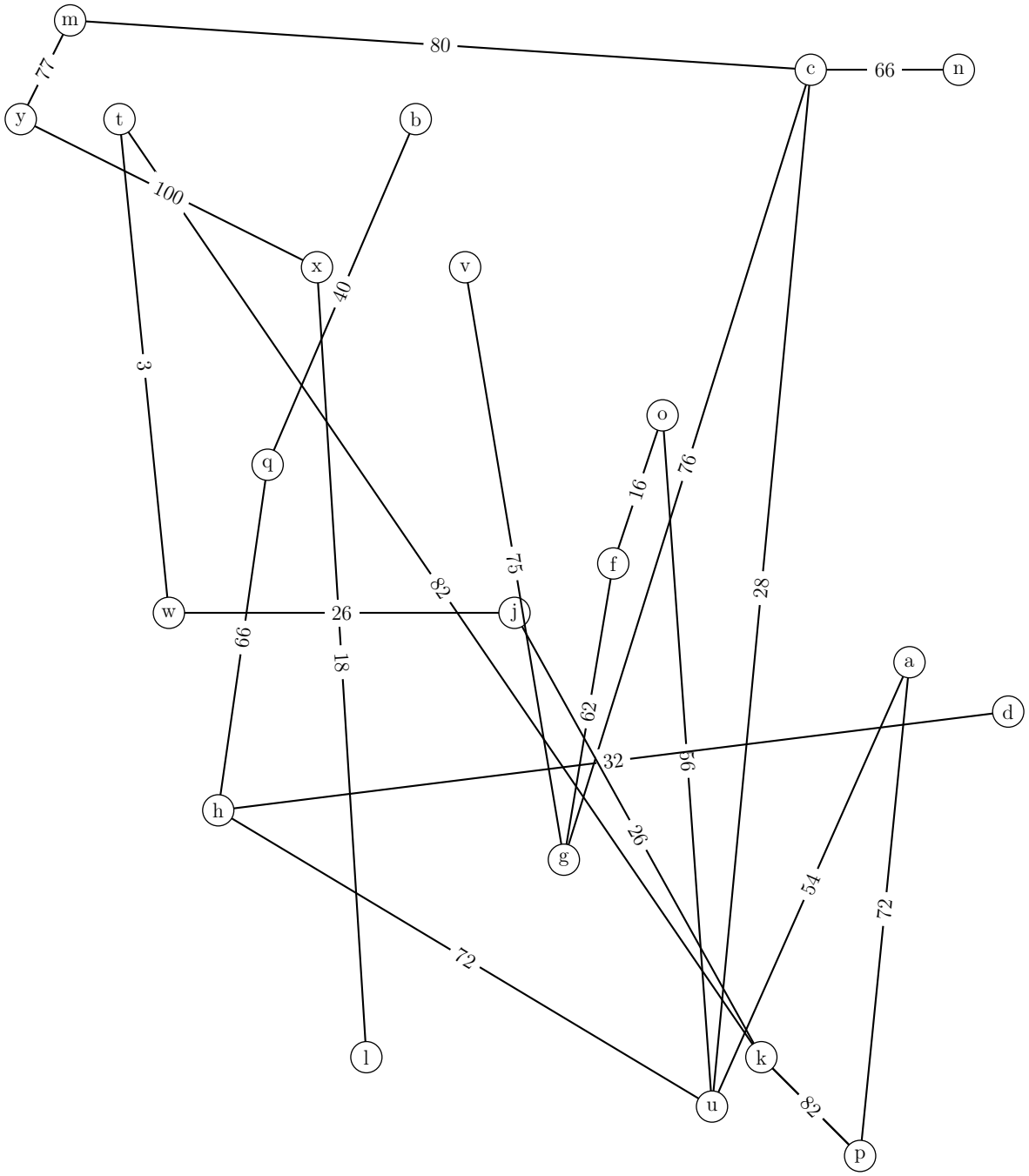
Jeżeli pomija się istotność ostatniego kroku to nie wykonuje się dodatkowych operacji polegających na dalszym przeszukiwaniu grafu. Oznacza to, że wyszukana ścieżka może być krótsza od znalezionej w algorytmie trzymającym się zasady wybierania najtańszej ścieżki do samego końca. Statystycznie krótsza ścieżka oznacza mniejszy koszt poniesiony na drodze pomiędzy punktami. Omawiana sytuacja przedstawiona jest na poniższym grafie (Rysunek 2).

W przedstawianym grafie należało znaleźć drogę z p do w.

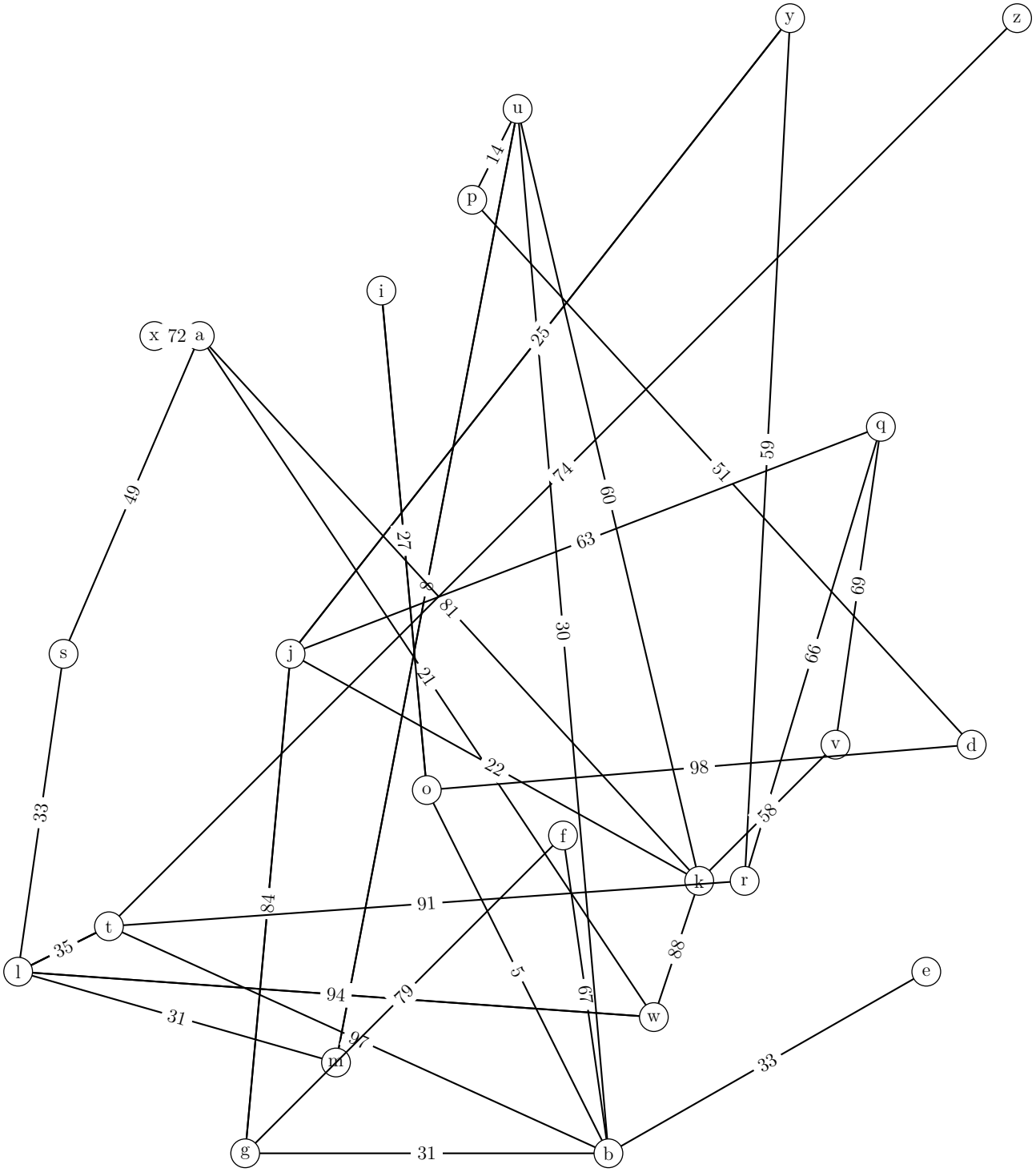
Dla podstawowej wersji wyszukiwania (z uwzględnianiem kosztu ostatniego kroku) znaleziono drogę: p->u->m->l->s->a->w Łączny koszt tej drogi to: 156

Jeżeli zastosuje się algorytm pomijający wagę ostatniego kroku to otrzyma się ścieżkę: p->u->m->l->w Łączny koszt tej drogi to: 147

Rysunek 1: Najprostszy algorytm



Rysunek 2: Skuteczniejszy



4.2.2 Sytuacje przy których algorytm okazał się mniej skuteczny

Nie zawsze koszty poszczególnych ścieżek w grafie są do siebie zbliżone. Możliwe jest wystąpienie sytuacji w której jedna z krawędzi odchodząca od wierzchołka docelowego ma o wiele większy koszt (wagę) od innych wychodzących z pola celu. Jeżeli w tej sytuacji algorytm pomijający istotność ostatniego kroku wyznaczy ścieżkę przez omawianą krawędź o wysokim koszcie, to może się okazać, że wybrana droga jest mniej korzystna od dłuższej, lecz prowadzącej przez krawędzie o niższych wagach. Omawiana sytuacja przedstawiona jest na poniższym grafie (Rysunek 3).

Zadaniem postawionym w tym grafie było przejście z l do e

Algorytm bez sprawdzania ostatniego elementu wyznaczył drogę: l->p->e Łączny koszt tej drogi to 119

Algorytm sprawdzający ostatnie z połączeń wyznaczył drogę: l->p->o->g->e Łączny koszt tej drogi to 114

4.3 Wyszukiwanie najlepszej drogi, czyli o najniższym koszcie.

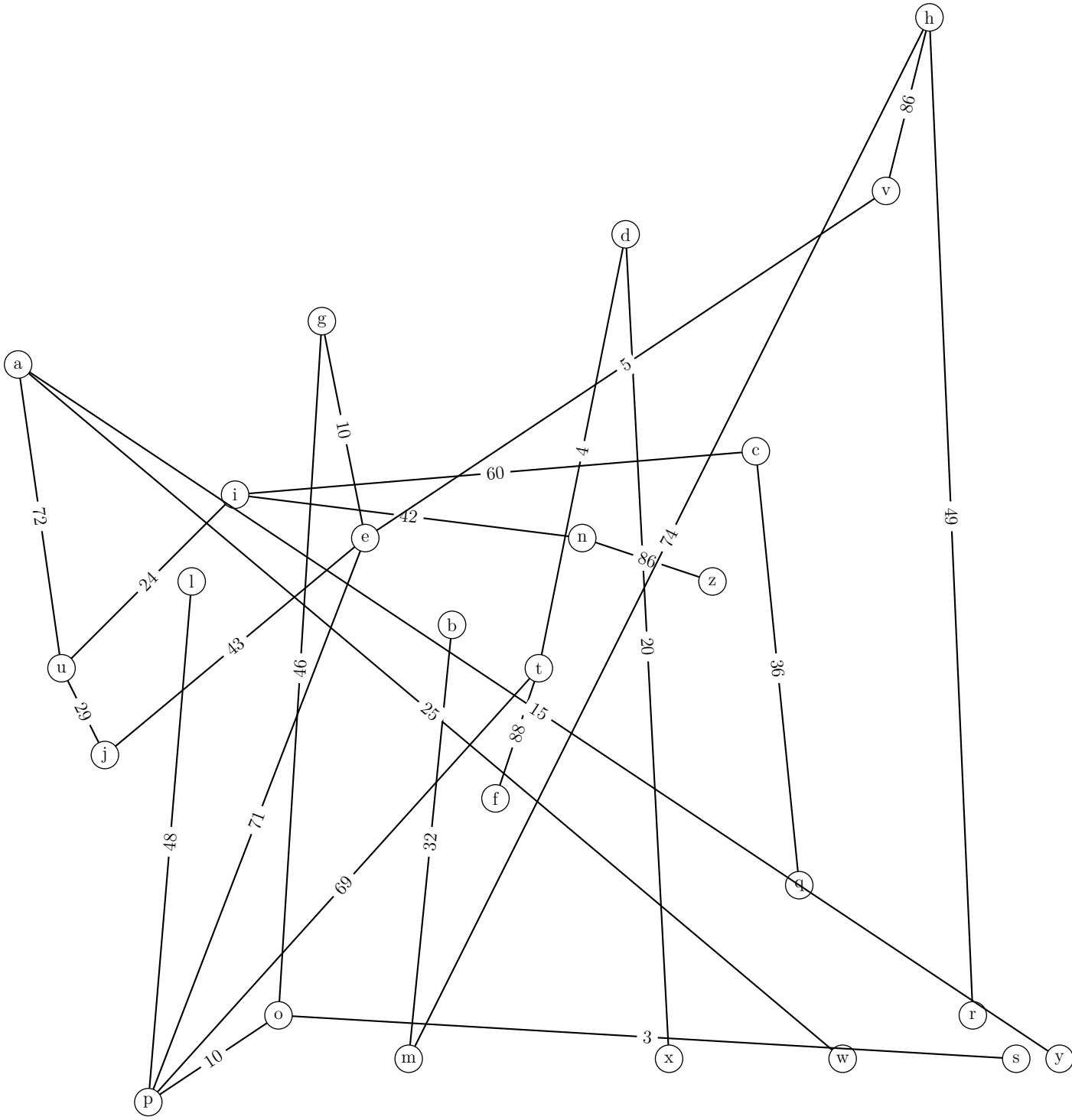
Omawiane wcześniej algorytmy działały lepiej lub gorzej w zależności od konkretnego przypadku. Nigdy jednak nie można było mieć pewności, że wyznaczona ścieżka posiada najniższy koszt z możliwych. Ten problem nie występuje w trzeciej wersji omawianego algorytmu.

Zmiana względem wcześniejszych polega na tym, że algorytm zapamiętuje koszt jaki musiał ponieść przemieszczając się po grafie w określone miejsce. Następnie do tego kosztu dodawana jest waga następnej krawędzi i tak otrzymany rezultat dodawany jest do listy priorytetowej na której znajdują się możliwe do wyboru ścieżki ułożone w sposób posortowany. Zapewnia to otrzymanie najtańszej z możliwych ścieżek (pod względem kosztów). Zalety z przeprowadzenia tego typu wyszukiwania omówiono na podstawie poniższego grafu (Rysunek 4).

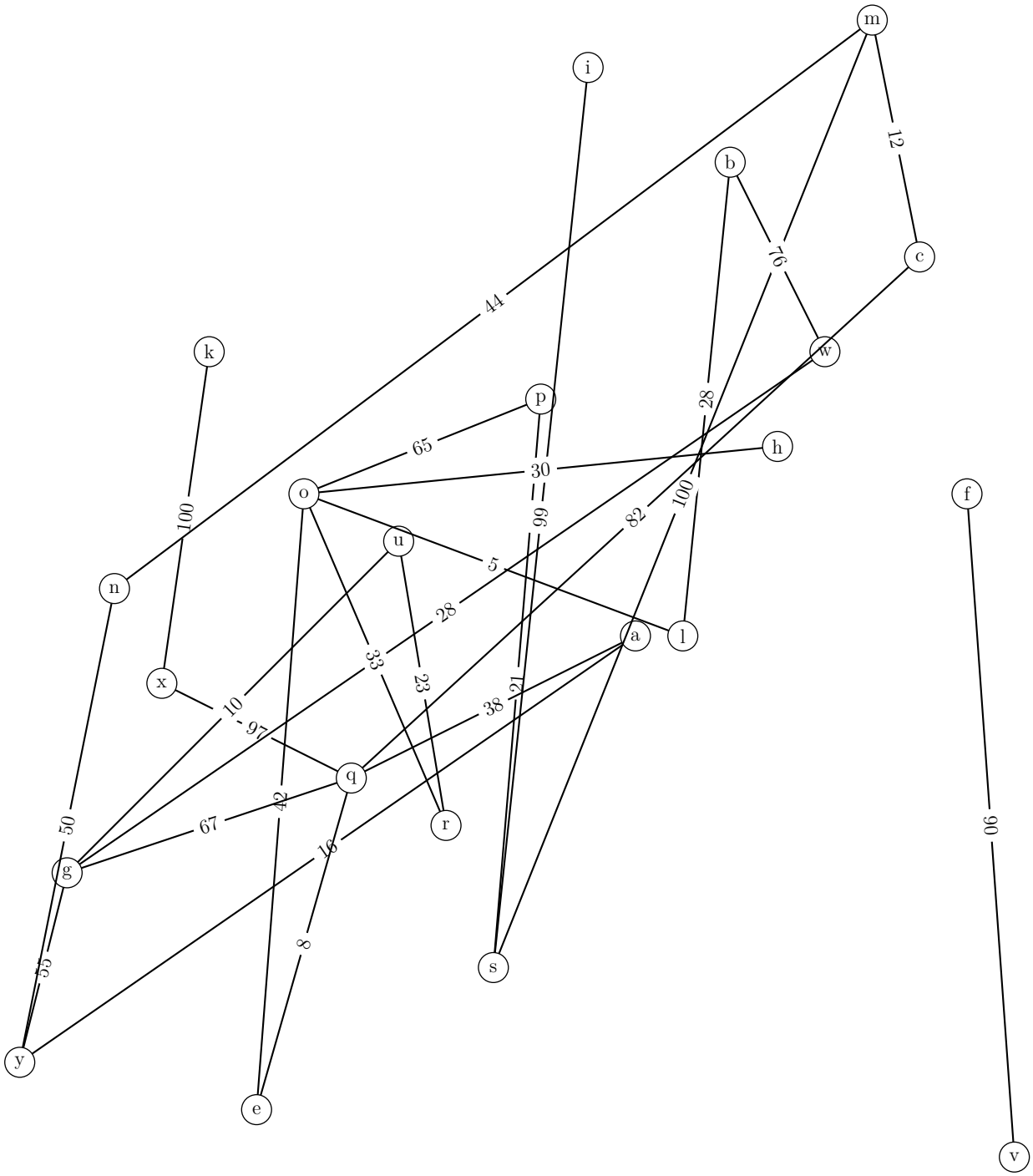
Droga powstała na podstawie algorytmu wybierającego zawsze ścieżkę o najmniejszym koszcie oraz algorytmu pomijającego wagę ostatniego połączenia wskazały ścieżkę: p->s->m->c->q->a o koszcie 253. W tym momencie u Czytelnika mogły się pojawić wątpliwości dlaczego algorytm wybrał właśnie tą drogę, ponieważ wydaje się że istnieją krawędzie o mniejszym koszcie, które pominął. Jednakże jest to tylko złudzenie, ponieważ ścieżki o niższych kosztach są początkami rozbudowanych cykli bądź *ślepych zaułków*.

Algorytm wyszukujący najlepszą ścieżkę wskazał drogę: p->o->e->q->a o koszcie 153.

Rysunek 3: Mniej skuteczny



Rysunek 4: Najlepsza droga



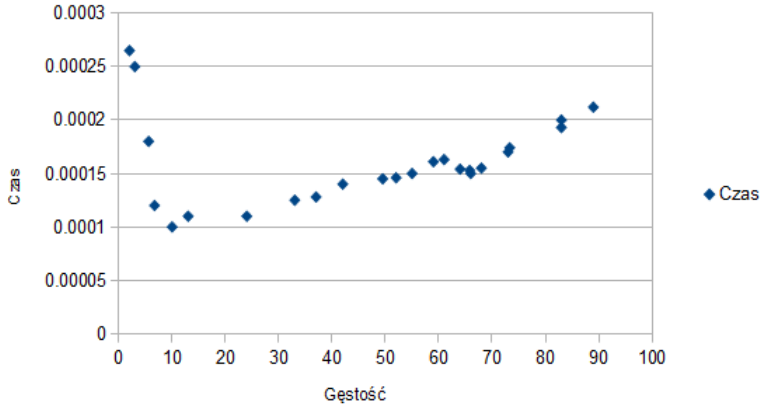
Rysunek 5: Zmiana gęstości grafu

Gęstość	Czas
83	0.0002
73	0.00017
55	0.00015
24	0.00011
13	0.00011
3	0.00025
5.6	0.00018
10	0.0001
6.7	0.00012
2	0.000265
66	0.00015
42	0.00014
33	0.000125
37	0.000128
49.5	0.000145
61	0.000163
59	0.000161
65.8	0.0001529
64	0.000154
68	0.000155
73.3	0.000174
89	0.000212
52	0.000146
83	0.000193

4.4 Pomiar czasu w zależności od gęstości grafu.

Zauważono, że na szybkość wykonywania algorytmu zdecydowanie wpływa gęstość grafu. Postanowiono przeprowadzić testy w tym kierunku. Otrzymane wyniki przedstawiono w tabeli (Rysunek 5)

Jak widać na załączonym wykresie (Rysunek 6) Zmiany gęstości grafu gęstość ma duże znaczenie. Dla wartości mniejszych od 10 procent szybkość zdecydowanie spada wraz ze wzrostem gęstości. Długie czasy dla niewielkiej ilości krawędzi są spowodowane faktem, iż często nie można znaleźć połączenia pomiędzy punktami w grafie i algorytm wykonuje przeszukiwanie zupełne. Wraz ze wzrostem ilości krawędzi prawdopodobieństwo występowania małych podgrafów maleje i dzięki temu czas również się zmniejsza. Dla gęstości grafu większych niż 10 procent czas rośnie wraz z przybywaniem elementów. Jest to najprawdopodobniej związane z przeprowadzaniem zapisów i odczytów z listy do której dodawane są kolejne punkty od których należy sprawdzać ścieżkę.



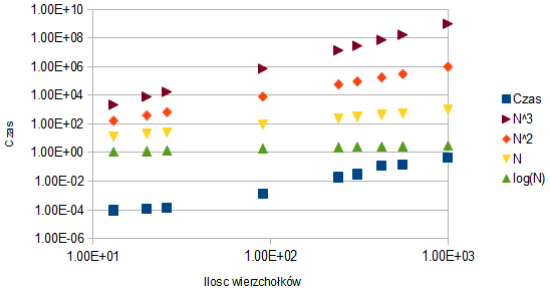
Rysunek 6: Zmiany gęstości grafu

Rysunek 7: Złożoność obliczeniowa

Ilość wierzchołków	Czas
26	0.000146
20	0.000121
13	0.000094
90	0.001312
306	0.030253
552	0.145129
240	0.019125
420	0.11745
992	0.439

4.5 Złożoność obliczeniowa algorytmu

Ze względu na zależność opisaną w poprzednim punkcie postanowiono wykonywać symulacje dla określonej gęstości grafu. Przyjęto gęstość wynoszącą około 50 procent. Następnie wykonano pomiary zamieszczone w tabeli na Rysunek 7



Rysunek 8: Złożoność algorytmu

5 Wnioski

Na podstawie danych, grafów i wykresów można wyciągnąć następujące wnioski:

- w poszukiwaniu najkorzystniejszej drogi w grafie dużą rolę odgrywa rodzaj zastosowanego algorytmu. Nawet niewielka zmiana schematu działania prowadzi do zupełnie różnych wyników
- gęstość grafu ma duży wpływa na czas wykonywania algorytmu
- przy małych gęstościach grafu algorytm często wykonuje przeszukanie zupełne
- jako że szukanie drogi w grafie jest uzależnione od doboru punktu startu i końca, duży nacisk należy położyć na ilość wykonywanych powtórzeń, aby otrzymać obiektywny wynik.
- im mniejsza gęstość grafu tym korzystniejsze jest wykorzystanie algorytmu pomijającego wagę ostatniej ścieżki.
- złożoność obliczeniowa algorytmu ulega zmianie. Dla niewielkiej ilości wierzchołków jest liniowa, a wraz ze wzrostem rozmiaru problemu przechodzi w kwadratową.