Problem komiwojażera

travelling salesman problem, TSP

Zdefiniowanie problemu

Definicja bardzo formalna: znalezienie minimalnego cyklu Hamiltona w pełnym grafie ważonym 😊

Teraz po kolei wyjaśniamy pojęcia:

Graf pełny – graf w którym każdy wierzchołek połączony jest **bezpośrednią krawędzią** z każdym innym wierzchołkiem (jest to dużo silniejsze wymaganie niż spójność grafu, gdzie dopuszczalne były połączenia przez sąsiadów, oczywiście graf pełny z definicji jest spójny).

Graf pełny ważony – dość oczywista kwestia (każda krawędź w grafie pełnym ma jakąś wagę),

Cykl Hamiltona – zamknięta ścieżka łącząca **wszystkie** wierzchołki w grafie, dla której każdy wierzchołek odwiedzany jest **tylko raz** (skoro ścieżka łączy wszystkie wierzchołki to każdy odwiedzany jest **dokładnie** raz).

Symetryczny problem komiwojażera (STSP) - W tym problemie przyjmujemy, że graf jest nieskierowany (innymi słowy dla dowolnej pary wierzchołków A, B krawędź od A do B ma taką samą długość jak krawędź od B do A). Taki problem będziemy rozważać na zajęciach (dodatkowo – na naszych zajęciach przyjmiemy, że wagi są dodatnie).

Prostsza definicja: przyjmijmy że mamy grupę N miast, między każdą parą miast z naszej grupy istnieje droga która ma określoną długość (lub cena biletu). Interesuje nas znalezienie takiej trasy dla komiwojażera (zaczynającej się w dowolnie wybranym mieście i kończącej w tym samym mieście), która każde miasto odwiedzałaby dokładnie jeden raz przy czym jej długość (lub koszt przebycia) powinna być jak najmniejsza. Innymi słowy szukamy uporządkowanej listy miast (ich kolejność powinna minimalizować długość/koszt trasy między nimi).

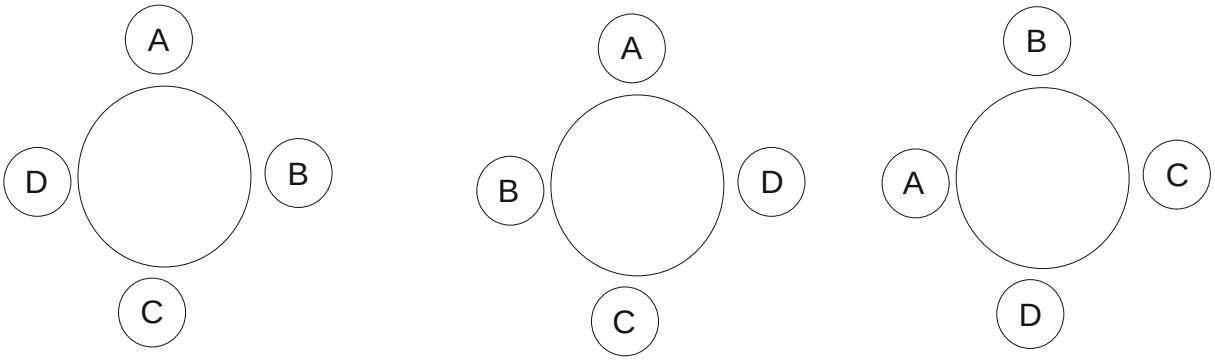
Złożoność obliczeniowa problemu

Wydawałoby się, że stosunkowo prostym (a przy tym ścisłym) rozwiązaniem problemu jest policzenie sumy krawędzi dla każdego możliwego cyklu Hamiltona i wybranie tego, dla którego suma jest najmniejsza. Zastanówmy się więc, ile takich cykli jest dla danego grafu.

(Poniższe wyprowadzenie to tylko ciekawostka, istotny jest tylko sam wzór i tabelka)

Jeśli chodzi o liczbę sposobów (X), w jaki można wybrać trasę będącą cyklem Hamiltona (niekoniecznie o najmniejszej sumie wag!) to jest to problem podobny do problemu znalezienia liczby sposobów posadzenia N osób przy okrągłym stole z N nienumerowanymi miejscami (przy czym wybierając sposób patrzymy na to jakich sąsiadów ma dana osoba).

Innymi słowy poniższe sposoby rozstawienia osób są równoważne (liczą się jako jeden)



Tak czy inaczej wyjdzie nam, że ilość sposobów dla N wierzchołków (N osób) wynosi:

$$X = \frac{(N-1)!}{2}$$

Bierze się to stąd, że pierwsza osoba zajmuje dowolne miejsce (nie ma to wpływu na wynik) druga ma (N-1) możliwości, trzecia (N-2), przedostatnia 2 możliwości a ostatnia 1 - stąd licznik (N-1)! . Dwójka w mianowniku bierze się z „symetrii" problemu (lewy i środkowy przykład na rysunku powyżej). W tabelce poniżej policzymy sobie liczbę sposobów dla kilku wybranych N

N	Ile krawędzi w grafie	Liczba sposobów X
3	3	1
6	15	60
10	45	$1.8*10^5$
20	190	$6.1*10^{16}$
30	435	$4.4*10^{30}$
40	780	$1.0*10^{46}$
60	1770	$0.7*10^{80}$
100	4950	$4.7*10^{155}$
We wszechświecie jest 10^{80} atomów 😊		

Jeśli chodzi o ścisłe rozwiązania to jedne z lepszych rozwiązań za pomocą programowania dynamicznego ma złożoność obliczeniową $O(N^2 2^N)$.

N	Ile krawędzi w grafie	$O(N^2 2^N)$.
10	45	$1*10^5$
20	190	$4*10^8$
30	435	$1*10^{12}$
40	780	$2*10^{15}$
60	1770	$4*10^{21}$
100	4950	$1*10^{34}$
We wszechświecie jest 10^{80} atomów 😊		

Algorytm zachłanny – najbliższego sąsiada (nearest neighbour (NN) algorithm)

Czego potrzebujemy

a) Listy, w której będziemy trzymać wierzchołki wchodzące w skład cyklu (ważne, by ta lista zachowywała kolejność dodawania).
b) Zmiennej, która przechowuje aktualny wierzchołek

c) Opcjonalnie możemy mieć też listę, w której będziemy trzymać krawędzie wchodzące w skład cyklu (jest to opcjonalne, suma wag krawędzi z tej listy to suma wag cyklu Hamiltona)
d) Opcjonalnie możemy mieć też tablicę (booleanów), w której zapisujemy, które wierzchołki są już „odwiedzone" czyli wchodzą w skład cyklu. Dzięki temu możemy szybko sprawdzić (czas O(1)) czy dany wierzchołek wchodzi w skład cyklu czy też nie (gdy sprawdzamy czy jest na liście wierzchołków odwiedzonych to zajmuje to czas O(N)).

Algorytm

1) Bierzemy wierzchołek startowy (dowolny) i oznaczamy go jako aktualny
2) Szukamy krawędzi z najniższą wagą z niego wychodzącej, dodajemy wierzchołek do startowy do cyklu. (opcjonalnie oznaczamy wierzchołek startowy jako odwiedzony oraz dodajemy znalezioną krawędź do listy krawędzi wchodzących w skład cyklu). Wierzchołek, do którego prowadzi znaleziona krawędź, traktujemy jako aktualny. Aktualny wierzchołek dodajemy do cyklu (po tym kroku w cyklu będą dwa wierzchołki), opcjonalnie aktualny wierzchołek zaznaczamy jako odwiedzony.
3) Dopóki długość listy z zawartością cyklu **nie** jest równa liczbie wszystkich wierzchołków w grafie (pętla while, opcjonalnie można sprawdzać czy jeszcze został jakikolwiek nieodwiedzony wierzchołek)

   Szukamy krawędzi o najniższej wadze wychodzącej z **aktualnego** wierzchołka i **NIE prowadzącej do wierzchołka już dodanego do cyklu**. Opcjonalnie znalezioną krawędź dodajemy do listy krawędzi cyklu. Następnie jako aktualny wierzchołek

oznaczamy ten, do którego prowadzi znaleziona przez nas krawędź. Aktualny wierzchołek dodajemy do cyklu (już ten nowy). Opcjonalnie aktualny wierzchołek dodajemy do listy odwiedzonych.

Algorytm ten daje wyniki średnio o 25 % gorsze (patrząc na długość cyklu) niż algorytm dokładny. Ta wartość jest zachowana, jeśli wierzchołki są w miarę równomiernie rozmieszczone na płaszczyźnie (a wagi krawędzi to odległości między wierzchołkami). Warto jednak mieć na uwadze, że można tak dobrać graf, że algorytm ten zwróci nam najgorszy z możliwych cykli Hamiltona (o największej sumie wag krawędzi).

Złożoność obliczeniowa:

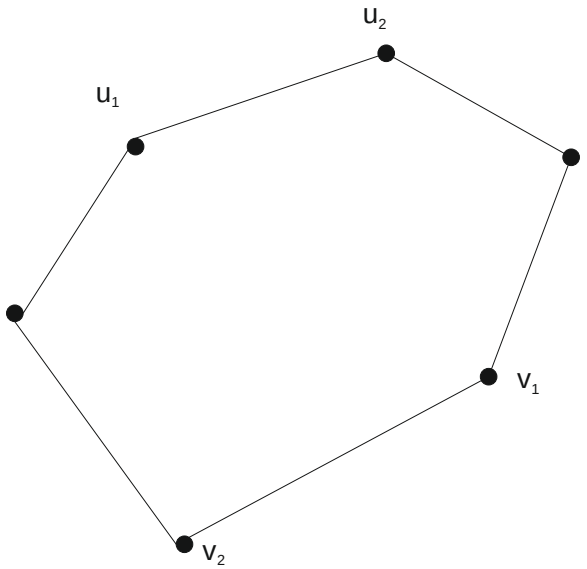
W każdym kroku pętli while dodajemy do cyklu jeden wierzchołek. Szukanie krawędzi z najniższą wagą wychodzącej z danego wierzchołka ma złożoność O(N). Zatem całkowita złożoność algorytmu to $O(N^2)$.

Proste ulepszenie

Uruchommy ten sam algorytm N razy, zaczynając od każdego wierzchołka w grafie, i jako wynik weźmy cykl z najmniejszą sumą wag. Złożoność takiego algorytmu będzie rzędu $O(N^3)$, średnio wynik (suma krawędzi w cyklu) będzie bodaj o 15% gorszy od dokładnego przy równomiernym rozmieszczeniu wierzchołków.

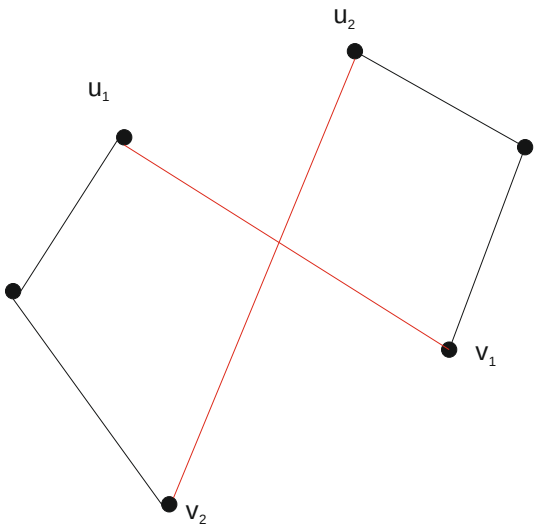
Ulepszenie 2-opt

Musimy mieć jakieś rozwiązanie wejściowe (cykl Hamiltona). Może to być np. wynik algorytmu zachłannego. Może też być to losowe uporządkowanie wierzchołków grafu (shuffle). Na rysunku poniżej mam wierzchołki grafu oraz nasz cykl wejściowy.



Musimy wybrać cztery wierzchołki, przy czym wszystkie muszą być różne ($u_1 \neq u_2 \neq v_1 \neq v_2$). O tym jak je konkretnie wybrać będzie za chwilę. Warto skupić się na tym że u_1 musi

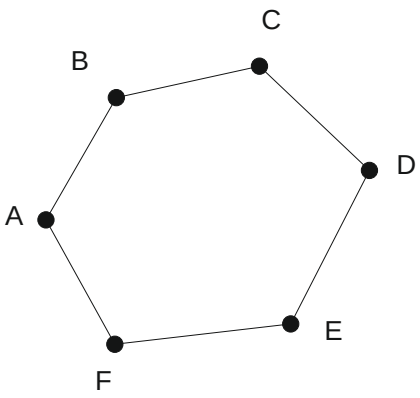
sąsiadować w cyklu z u_2 oraz v_1 musi sąsiadować z v_2. Zatem tak naprawdę musimy wybrać dwa indeksy wierzchołków z cyklu (i oraz j) które określą nam $u_1 \, oraz \, v_1$ natomiast $v_2 \, oraz \, v_2$ będą kolejnymi wierzchołkami (oczywiście i oraz j muszą spełniać pewne warunki by zachodziło $u_1 \neq u_2 \neq v_1 \neq v_2$). Pojedyncza modyfikacja polega na tym by rozciąć (usunąć z cyklu) krawędzie łączące u_1u_2 oraz v_1v_2 i dodać do cyklu krawędzie u_1v_1 oraz u_2v_2 (jak pokazano na kolejnym rysunku)



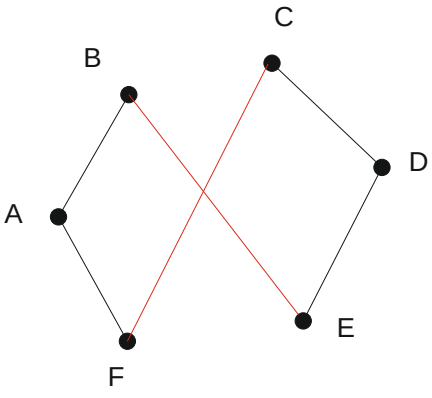
Algorytm 2-opt polega na tym, by (dla danego cyklu wejściowego) sprawdzić każdą dopuszczalną kombinację i oraz j (można je wybrać na $O(N^2)$ sposobów) i sprawdzić, która dała najlepszy rezultat (największe zmniejszenie sumy wag w cyklu).

Algorytm ten można uruchamiać wiele razy (w kolejnej iteracji na wyniku poprzedniej) co jeszcze bardziej poprawi wynik.

Jak w praktyce należy wykonać korektę cyklu w algorytmie 2-opt



ABCDEF



ABEDCF

Na rysunku powyżej (po lewej) mamy wejściowy cykl ABCDEF. Załóżmy, że chcemy przerwać krawędzie pomiędzy 1 i 2 oraz 4 i 5 wierzchołkiem (numeracja wierzchołków od 0 (wierzchołek A) do 5 (wierzchołek F)). W takiej sytuacji do wierzchołka będącego początek pierwszej rozrywanej krawędzi

włącznie kolejność w cyklu się nie zmienia (AB). Następnie musimy iść do pierwszego wierzchołka w drugiej rozrywanej krawędzi (czyli u nas E) i licząc od niego włącznie ODWRÓCIĆ kolejność wierzchołków w wejściowym cyklu, aż dojdziemy do drugiego wierzchołka pierwszej rozrywanej krawędzi włącznie (czyli do C). Z tego przyczynku dostaniemy fragment cyklu EDC. W ostatnim kroku wracamy do wierzchołka będącego końcem drugiej rozrywanej krawędzi i kopiujemy pozostały fragment wejściowego cyklu bez zmian (czyli u nas to będzie tylko F bo cykl się skończył). Sumarycznie dostaniemy zatem cykl ABEDCF.

Zadania na dziś.

Plan minimum 😊

1) Dla zadanego grafu
   a) znaleźć cykl Hamiltona o niskiej sumie wag za pomocą algorytmu zachłannego (wersja podstawowa) startując od wierzchołka o indeksie i=0 (oprócz samego cyklu – czyli kolejności wierzchołków- proszę podać sumę wag krawędzi dla znalezionego cyklu)
   b) to samo co w podpunkcie a) tylko stosując optymalizację, polegającą na uruchamianiu programu z dowolnego wierzchołka

Fajnie by było gdyby się udało:

2) zaimplementować algorytm opt-2 i poprawić nim wynik z 1b) (wynikiem tego będzie cykl + jego waga).

To są ostatnie zajęcia, dodatkowo problem jest dość trudny (czasochłonny, zwłaszcza jeśli bierzemy pod uwagę także modyfikacje 2-opt), dlatego dzisiaj pracujemy nie indywidualnie ale jako zespół.