

Kombinatoryczne problemy optymalizacyjne to problemy wyboru najlepszego rozwiązania z pewnego zbioru rozwiązań dopuszczalnych.

NP-optymalizacyjny problem Π składa się ze:

- zbioru *instancji* D_Π rozpoznawalnego w czasie wielomianowym; liczby w instancjach są wymierne; model obliczeń nie obsługuje liczb rzeczywistych nieskończonej precyzji; *rozmiarem* instancji $I \in D_\Pi$, oznaczanym $|I|$, nazywamy liczbę bitów potrzebnych do zapisania $|I|$ przy założeniu, że wszystkie liczby występujące w I są reprezentowane binarnie;
- zbioru *rozwiązań dopuszczalnych* $S_\Pi(I)$ dla każdej instancji $I \in D_\Pi$; zbiory $S_\Pi(I)$ są niepuste i każde rozwiązanie $s \in S_\Pi(I)$ ma rozmiar wielomianowy względem $|I|$; ponadto istnieje algorytm wielomianowy, który dla danej pary (I, s) stwierdza czy $s \in S_\Pi(I)$;
- wielomianowo obliczalnej *funkcji celu*, f_Π lub f (w literaturze *obj Π*), przyporządkowującej liczbę rzeczywistą $f(I, s)$ każdej parze (I, s) , gdzie I jest instancją Π , a s rozwiązaniem dopuszczalnym dla I ; funkcja celu ma często interpretację fizyczną jak *koszt*, *długość*, *waga* czy inna miara, w tym będąca kombinacją (często ważoną) różnych wielkości fizycznych.

- Dodatkowo o problemie Π wiemy, że może być *problemem minimalizacji* bądź *problemem maksymalizacji* (funkcji celu).
- Ograniczenie problemu Π do instancji, w których wszystkie koszty są jednostkowe, nazywamy *licznościową wersją* Π .
- *Rozwiązaniem optymalnym* problemu minimalizacji (maksymalizacji) jest takie rozwiązanie dopuszczalne, dla którego funkcja celu przyjmuje najmniejszą (największą) wartość. Przez $OPT_{\Pi}(I)$ (OPT jeśli problem i instancja są oczywiste) oznaczamy wartość funkcji celu dla rozwiązania optymalnego instancji I .

Na przykład:

- Instancja problemu pokrycia wierzchołkowego składa się z grafu nieskierowanego $G = (V, E)$ i funkcji kosztu określonej na zbiorze wierzchołków.
- Dopuszczalnym rozwiązaniem jest dowolny zbiór $S \subseteq V$, będący pokryciem G .
- Wartością funkcji celu rozwiązania S jest suma kosztów wierzchołków z S .
- Rozwiązaniem optymalnym jest rozwiązanie dopuszczalne o minimalnym koszcie.

Czy problem pokrycia wierzchołkowego jest problemem **NP**-trudnym i jakie są tego konsekwencje dla czasu jego rozwiązania?

Algorytm aproksymacyjny znajduje rozwiązanie, które jest „bliskie” optymalnemu, a do tego jest efektywny czasowo.

Niech Π będzie problemem minimalizacji (maksymalizacji) i niech $k : \mathbf{Z}^+ \rightarrow \mathbf{Q}^+$ spełnia $k \geq 1$ ($k \leq 1$).

Algorytm \mathcal{A} jest *algorytmem k -aproksymacyjnym dla Π* , jeżeli dla dowolnej instancji I , \mathcal{A} znajduje rozwiązanie dopuszczalne s takie, że $f_{\Pi}(I, s) \leq k(|I|) \cdot OPT(I)$ ($f_{\Pi}(I, s) \geq k(|I|) \cdot OPT(I)$). Ponadto wymagany czas działania \mathcal{A} musi być wielomianowy względem $|I|$.

Albo inaczej.

Algorytm aproksymacyjny \mathcal{A} dla problemu π znajduje w czasie wielomianowym rozwiązanie dopuszczalne, dla którego wartość funkcji celu jest bliska optymalnej, czyli różni się od OPT o co najwyżej o pewien z góry określony czynnik, tzw. *współczynnik aproksymacji*.

1. Kiedy algorytm aproksymacyjny jest lepszy, z punktu widzenia wielkości *współczynnika aproksymacji k* ?
 k bliższe 1, k dalsze od 1, $k = 1$.
2. Aby zagwarantować aproksymację z konkretnym współczynnikiem należy ...
3. i dlaczego jest to trudne?

Aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego. Jednak dla problemów **NP**-trudne jest nie tylko znalezienie rozwiązania optymalnego, ale również (a może nawet bardziej) obliczenie jego kosztu.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego. Jednak dla problemów **NP**-trudne jest nie tylko znalezienie rozwiązania optymalnego, ale również (a może nawet bardziej) obliczenie jego kosztu.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Algorytm aproksymacyjny dla licznosciowego problemu pokrycia wierzchołkowego

Niezbędne definicje.

Dla grafu $H = (U, F)$ podzbiór M zbioru krawędzi nazywamy *skojarzeniem*, jeżeli żadne dwie krawędzie z M nie mają wspólnego końca.

Skojarzeniem najliczniejszym nazywamy skojarzenie o największej możliwej mocy.

Skojarzeniem maksymalnym nazywamy skojarzenie maksymalne pod względem zawierania.

Skojarzenie maksymalne można znaleźć w czasie wielomianowym, zachłannie wybierając krawędzie i usuwając ich końca z grafu.

Można też znaleźć w czasie wielomianowym skojarzenie najliczniejsze, ale metoda(y) nie jest(są) już wtedy takie proste.

Rozmiar dowolnego maksymalnego skojarzenia w G jest ograniczeniem dolnym.

Dlaczego?

Algorytm aproksymacyjny dla licznosciowego problemu pokrycia wierzchołkowego

Niezbędne definicje.

Dla grafu $H = (U, F)$ podzbiór M zbioru krawędzi nazywamy *skojarzeniem*, jeżeli żadne dwie krawędzie z M nie mają wspólnego końca.

Skojarzeniem najliczniejszym nazywamy skojarzenie o największej możliwej mocy.

Skojarzeniem maksymalnym nazywamy skojarzenie maksymalne pod względem zawierania.

Skojarzenie maksymalne można znaleźć w czasie wielomianowym, zachłannie wybierając krawędzie i usuwając ich końca z grafu.

Można też znaleźć w czasie wielomianowym skojarzenie najliczniejsze, ale metoda(y) nie jest(są) już wtedy takie proste.

Rozmiar dowolnego maksymalnego skojarzenia w G jest ograniczeniem dolnym.

Dlaczego?

Dowolne pokrycie wierzchołkowe musi bowiem zawierać co najmniej jeden z końców każdej krawędzi skojarzenia.

Algorytm

$W \leftarrow \emptyset$

while G zawiera przynajmniej jedną krawędź do

wybierz dowolną krawędź (u, v) w G

$W \leftarrow W \cup u, v$

usuń wszystkie krawędzie pokryte przez u i v z G

end while

return W

Sprawdźmy, czy prawdziwe jest zatem twierdzenie, że powyższy algorytm jest algorytmem *2-aproksymacyjnym* dla licznosciowego problemu pokrycia wierzchołkowego?

Jeżeli M będzie znalezionym skojarzeniem, to wszystkie krawędzie G są pokryte wybranymi przez algorytm wierzchołkami.

Gdyby jakaś krawędź nie była pokryta, można by ją dodać do M , co przeczyłoby maksymalności M .

1. Jakie jest *dolne ograniczenie*?
2. Ile zatem jest krawędzi w M ?
3. Ile jest wierzchołków pokrywanych przez te krawędzie?
4. Jaki współczynnik aproksymacji ma powyższy algorytm?

Sprawdźmy, czy prawdziwe jest zatem twierdzenie, że powyższy algorytm jest algorytmem *2-aproksymacyjnym* dla licznosciowego problemu pokrycia wierzchołkowego?

Jeżeli M będzie znalezionym skojarzeniem, to wszystkie krawędzie G są pokryte wybranymi przez algorytm wierzchołkami.

Gdyby jakaś krawędź nie była pokryta, można by ją dodać do M , co przeczyłoby maksymalności M .

1. Jakie jest *dolne ograniczenie*?
2. Ile zatem jest krawędzi w M ?
3. Ile jest wierzchołków pokrywanych przez te krawędzie?
4. Jaki współczynnik aproksymacji ma powyższy algorytm?

Algorytm jest *2-aproksymacyjny*

Niech Π będzie **NP**-trudnym problemem optymalizacyjnym z funkcją celu f_Π .

Mówimy, że algorytm \mathcal{A} jest *schematem aproksymacyjnym* dla Π , jeżeli dla wejścia (I, ϵ) , gdzie I jest instancją Π , a $\epsilon > 0$ jest parametrem opisującym dopuszczalny błąd, \mathcal{A} daje rozwiązanie s takie, że

- $f_\Pi(I, s) \leq (1 + \epsilon) \cdot OPT$, jeśli Π jest problemem minimalizacyjnym,
- $f_\Pi(I, s) \geq (1 - \epsilon) \cdot OPT$, jeśli Π jest problemem maksymalizacyjnym.

Mówimy, że \mathcal{A} jest *wielomianowym schematem aproksymacyjnym*, (*Polynomial Time Approximation Scheme*), jeżeli dla dowolnego ustalonego $\epsilon > 0$ czas działania \mathcal{A} jest wielomianowy ze względu na rozmiar instancji I .

Mówimy, że \mathcal{A} jest *w pełni wielomianowym schematem aproksymacyjnym*, (*Fully Polynomial Time Approximation Scheme*), jeżeli dla dowolnego ustalonego $\epsilon > 0$ czas działania \mathcal{A} jest wielomianowy ze względu na rozmiar instancji I oraz ze względu na $1/\epsilon$.

O ile $\mathbf{P} \neq \mathbf{NP}$, to istnienie algorytmu FPTAS jest w wypadku **NP**-trudnych problemów optymalizacyjnych najlepszą możliwą sytuacją.

Problem komiwojażera (*ang. Travelling Salesman Problem (TSP)*)

Problem. Dla danego grafu pełnego i nieujemnej funkcji kosztu określonej na jego krawędziach należy znaleźć najtańszy cykl przechodzący przez każdy z wierzchołków dokładnie raz.

Pytanie 1. Czy dla ogólnego TSP istnieją algorytmy aproksymacyjne?

Problem komiwojażera (*ang. Travelling Salesman Problem (TSP)*)

Problem. Dla danego grafu pełnego i nieujemnej funkcji kosztu określonej na jego krawędziach należy znaleźć najtańszy cykl przechodzący przez każdy z wierzchołków dokładnie raz.

Pytanie 1. Czy dla ogólnego TSP istnieją algorytmy aproksymacyjne?

Nie

Problem komiwojażera (*ang. Travelling Salesman Problem (TSP)*)

Problem. Dla danego grafu pełnego i nieujemnej funkcji kosztu określonej na jego krawędziach należy znaleźć najtańszy cykl przechodzący przez każdy z wierzchołków dokładnie raz.

Pytanie 1. Czy dla ogólnego TSP istnieją algorytmy aproksymacyjne?

Nie

Pytanie 2. Czy można to udowodnić?

Problem komiwojażera (*ang. Travelling Salesman Problem (TSP)*)

Problem. Dla danego grafu pełnego i nieujemnej funkcji kosztu określonej na jego krawędziach należy znaleźć najtańszy cykl przechodzący przez każdy z wierzchołków dokładnie raz.

Pytanie 1. Czy dla ogólnego TSP istnieją algorytmy aproksymacyjne?

Nie

Pytanie 2. Czy można to udowodnić?

Tak

Problem komiwojażera (*ang. Travelling Salesman Problem (TSP)*)

Problem. Dla danego grafu pełnego i nieujemnej funkcji kosztu określonej na jego krawędziach należy znaleźć najtańszy cykl przechodzący przez każdy z wierzchołków dokładnie raz.

Pytanie 1. Czy dla ogólnego TSP istnieją algorytmy aproksymacyjne?

Nie

Pytanie 2. Czy można to udowodnić?

Tak

Pytanie 3. W jaki sposób?

Twierdzenie

Dla żadnej funkcji $\alpha(n)$ obliczalnej w czasie wielomianowym (np. $\alpha(n) = \log n, \alpha(n) = n_3, \alpha(n) = 5^n, \alpha(n) = 2^{2^n}$) nie istnieje algorytm $\alpha(n)$ -aproxymacyjny dla TSP, o ile $\mathbf{P} \neq \mathbf{NP}$

Dowód.

Założmy przeciwnie, że algorytm \mathcal{A} może zostać użyty do rozwiązania *problemu cyklu Hamiltona* w czasie wielomianowym, czyli że $\mathbf{P} = \mathbf{NP}$.

Głównym elementem dowodu jest redukcja problemu Hamiltona do problemu komiwojażera. Redukcja ta przekształca n -wierzchołkowy graf G w pełny n -wierzchołkowy graf G' , taki, że:

1. jeśli G ma cykl Hamiltona, to koszt optymalnej trasy w G' jest równy n , oraz
2. jeśli G nie posiada cyklu Hamiltona, to koszt optymalnej trasy w G' jest większy niż $\alpha(n) \cdot n$.

Graf G' jest instancją (egzemplarzem) problemu komiwojażera. Jest to graf pełny, $G' = (V, E)$, gdzie

$$w(uv) = \begin{cases} 1 & \text{gdy } uv \in E \\ n \cdot \alpha(n) & \text{w p.p.} \end{cases}$$

Algorytm \mathcal{A} wykonany na grafie G' znajdzie:

ad. 1. rozwiązanie o koszcie $\leq \alpha(n) \cdot n$; $n = OPT$.

ad. 2. $> \alpha(n) \cdot n$; czyli każdy cykl Hamiltona zawiera co najmniej jedną krawędź o wadze $\alpha(n) \cdot n$.

Co zatem udowodniliśmy?

Algorytm \mathcal{A} wykonany na grafie G' znajdzie:

ad. 1. rozwiązanie o koszcie $\leq \alpha(n) \cdot n$; $n = OPT$.

ad. 2. $> \alpha(n) \cdot n$; czyli każdy cykl Hamiltona zawiera co najmniej jedną krawędź o wadze $\alpha(n) \cdot n$.

Co zatem udowodniliśmy?

Udowodniliśmy, że G ma cykl Hamiltona wtedy i tylko wtedy, gdy \mathcal{A} zwróci rozwiązanie o koszcie co najwyżej $\alpha(n) \cdot n$, co możemy sprawdzić w czasie wielomianowym.

Ponieważ (decyzyjny) problem cyklu Hamiltona jest **NP**-trudny, więc **P** = **NP**, co jest sprzeczne z założeniem.

□

Zatem, dla ogólnego TSP nie można stworzyć algorytmu aproksymacyjnego.

Pytanie. A dla jakiego można?

Krawędzie w grafie G nie spełniały nierówności trójkąta.

$$c(u, v) \leq c(u, w) + c(v, w)$$

Jeśli nierówność ta będzie spełniona, to problem TSP z ogólnego stanie się *metrycznym* TSP.

Nadal będzie **NP**-zupełny, jednak nie będzie już nieaproxymowalny.

Rozpatrzmy zatem najprostszy algorytm (k-apx_TSP) aproksymacyjny dla metrycznego TSP.

1. Znajdź minimalne drzewo rozpinające T grafu G .
2. Zastąp krawędzie T parami krawędzi; otrzymany graf \mathcal{L} będzie eulerowski.
3. Znajdź cykl Eulera \mathcal{T} w \mathcal{L} .
4. Wypisz trasę \mathcal{C} , która odwiedza wierzchołki G w kolejności ich pierwszych wystąpień w \mathcal{T} .

Algorytm jest k -aproksymacyjny.

Pytanie. Ile wynosi k ?

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy?

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Pytanie. Dlaczego zatem znalezienie MST da nam *ograniczenie dolne*?

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Pytanie. Dlaczego zatem znalezienie MST da nam *ograniczenie dolne*?

Ponieważ usunięcie dowolnej krawędzi z optymalnego rozwiązania problemu komiwojażera daje MST.

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Pytanie. Dlaczego zatem znalezienie MST da nam *ograniczenie dolne*?

Ponieważ usunięcie dowolnej krawędzi z optymalnego rozwiązania problemu komiwojażera daje MST.

Pytanie. Jaki jest koszt MST (w stosunku do OPT)?

Wiemy (z poprzedniego wykładu ☺), że aby zagwarantować aproksymację z konkretnym współczynnikiem należy porównać koszt rozwiązanie znajdowanego przez algorytm \mathcal{A} z kosztem rozwiązanie optymalnego.

Czego potrzebujemy?

Wskazówka: to coś (pojęcie/nazwa) pojawiło się w związku ze schematem $B\&B$.

Ograniczenia dolnego znalezione w czasie wielomianowym.

Pytanie. Dlaczego zatem znalezienie MST da nam *ograniczenie dolne*?

Ponieważ usunięcie dowolnej krawędzi z optymalnego rozwiązania problemu komiwojażera daje MST.

Pytanie. Jaki jest koszt MST (w stosunku do OPT)?

$$c(T) \leq OPT$$

Twierdzenie. *Algorytm $k\text{-apx_TSP}$ jest algorytmem 2-aproksymacyjnym dla metrycznego TSP.*

Dowód. Wiemy, że ...

jaki jest koszt MST w stosunku do OPT ?

Twierdzenie. *Algorytm $k\text{-apx_TSP}$ jest algorytmem 2-aproksymacyjnym dla metrycznego TSP.*

Dowód. Wiemy, że

$$c(T) \leq OPT$$

Ponieważ \mathcal{T} przechodzi każdą krawędź T dwukrotnie to ...

jak się ma koszt cyklu Eulera do kosztu MST?

Twierdzenie. *Algorytm $k\text{-apx_TSP}$ jest algorytmem 2-aproksymacyjnym dla metrycznego TSP.*

Dowód. Wiemy, że

$$c(T) \leq OPT$$

Ponieważ \mathcal{T} przechodzi każdą krawędź T dwukrotnie

$$c(\mathcal{T}) = 2 \cdot c(T)$$

Z nierówności trójkąta wynika, że w kroku 4. skracamy trasę, zatem

jak się ma koszt trasy do kosztu cyklu Eulera?

Twierdzenie. *Algorytm $k\text{-apx_TSP}$ jest algorytmem 2-aproksymacyjnym dla metrycznego TSP.*

Dowód. Wiemy, że

$$c(T) \leq OPT$$

Ponieważ \mathcal{T} przechodzi każdą krawędź T dwukrotnie

$$c(\mathcal{T}) = 2 \cdot c(T)$$

Z nierówności trójkąta wynika, że w kroku 4. skracamy trasę, zatem

$$c(\mathcal{C}) \leq c(\mathcal{T})$$

W związku z tym, ostatecznie otrzymujemy ...

Twierdzenie. *Algorytm $k\text{-apx_TSP}$ jest algorytmem 2-aproksymacyjnym dla metrycznego TSP.*

Dowód. Wiemy, że

$$c(T) \leq OPT$$

Ponieważ \mathcal{T} przechodzi każdą krawędź T dwukrotnie

$$c(\mathcal{T}) = 2 \cdot c(T)$$

Z nierówności trójkąta wynika, że w kroku 4. skracamy trasę, zatem

$$c(\mathcal{C}) \leq c(\mathcal{T})$$

W związku z tym, ostatecznie otrzymujemy

$$c(\mathcal{C}) \leq 2 \cdot OPT$$

□

Współczynnik aproksymacji $3/2$

Możliwe jest poprawienie współczynnika aproksymacji do $3/2$

W tym celu należy skorzystać z dwóch twierdzeń

Twierdzenie 1. *W spójnym multigrafie istnieje cykl Eulera wtedy i tylko wtedy, gdy stopień każdego wierzchołka jest parzysty.*

Co więcej, jeśli cykl Eulera istnieje, można go znaleźć w czasie liniowym.

Twierdzenie 2. *Skojarzenie doskonałe o najmniejszym koszcie można znaleźć w czasie wielomianowym.*

Pozbywamy się wierzchołków stopnia nieparzystego. Niech V' oznacza zbiór wszystkich wierzchołków nieparzystych. Liczba $|V'|$ musi być liczbą parzystą, ponieważ suma stopni wszystkich wierzchołków MST jest parzysta. Jeżeli dodamy do tego drzewa *najtańsze doskonałe skojarzenie* w V' , wszystkie wierzchołki będą miały stopień parzysty. Otrzymany graf będzie eulerowski. Otrzymamy następujący algorytm.

Metryczny TSP - algorytm o współczynniku $3/2$

1. Znajdź minimalne drzewo rozpinające T grafu G .
2. Znajdź najtańsze skojarzenie doskonałe M w zbiorze wierzchołków nieparzystego stopnia w T . Dodaj M do T ; otrzymany graf jest eulerowski.
3. Znajdź cykl Eulera \mathcal{T} w \mathcal{L} .
4. Wypisz trasę \mathcal{C} , która odwiedza wierzchołki G w kolejności ich pierwszych wystąpień w \mathcal{T} .

Dowód na to, że pokazany algorytm jest algorytmem $3/2$ -aproxymacyjnym można znaleźć w książce „Algorytmy aproksymacyjne” - Vijay V. Vazirani, strona 31.

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Tak

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Tak

Pytanie 2. W jaki sposób?

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Tak

Pytanie 2. W jaki sposób?

Otóż problem TSP można rozpatrywać jako rozgrywający się w określonej przestrzeni. W d -wymiarowej przestrzeni euklidesowej. Jest to szczególny przypadek problemu TSP. Tak zwany *euklidesowy* TSP.

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Tak

Pytanie 2. W jaki sposób?

Otóż problem TSP można rozpatrywać jako rozgrywający się w określonej przestrzeni. W d -wymiarowej przestrzeni euklidesowej. Jest to szczególny przypadek problemu TSP. Tak zwany *euklidesowy* TSP.

Pytanie 3. Jak zatem wygląda rozwiązanie *euklidesowego* TSP?

Pytanie 1. Czy problem TSP można w jakiś sposób „uproszczyć”? Rozpatrywać go w inny sposób?

Tak

Pytanie 2. W jaki sposób?

Otóż problem TSP można rozpatrywać jako rozgrywający się w określonej przestrzeni. W d -wymiarowej przestrzeni euklidesowej. Jest to szczególny przypadek problemu TSP. Tak zwany *euklidesowy* TSP.

Pytanie 3. Jak zatem wygląda rozwiązanie *euklidesowego* TSP?

Skonstruowanie algorytmu rozwiązującego *euklidesowy* TSP nie jest proste. Stąd też nie będziemy się tym zajmować na wykładzie. Niemniej jednak, dla takiego problemu skonstruowano algorytm typu PTAS.

Sanjeev Arora w Polynomial time approximation scheme go Euclidian TSP and other geometric problems. In: *Proc. 37th IEEE Annual Symposium on Foundations of Computer Science*, pp. 2 – 11, 1996.

Prezentacja algorytmu znajduje się na stronie zoo.iiar.pwr.wroc.pl/pea.html

Algorytm Arora miał złożoność

$$O\left(n^{2d}(\log n)^{O\left((\sqrt{d}/\epsilon)^{d-1}\right)}\right)$$

Algorytm Arora miał złożoność

$$O\left(n^{2d}(\log n)^{O\left((\sqrt{d}/\epsilon)^{d-1}\right)}\right)$$

Niezależnie, ten sam wynik uzyskał **J.S.B. Mitchell** Guillotine subdivisions approximate polynomial subdivision:
a simple polynomial-time approximation scheme for geometric TSP, k -MST, and related problems. *SIAM Journal on Computing*, 28 : 1298 – 1309, 1999

Algorytm Arora miał złożoność

$$O\left(n^{2d}(\log n)^{O\left((\sqrt{d}/\epsilon)^{d-1}\right)}\right)$$

Niezależnie, ten sam wynik uzyskał **J.S.B. Mitchell** Guillotine subdivisions approximate polynomial subdivision: a simple polynomial-time approximation scheme for geometric TSP, k -MST, and related problems. *SIAM Journal on Computing*, 28 : 1298 – 1309, 1999

W 1998 pojawił się algorytm o lepszej złożoności czasowej

$$O\left(n\left(\log(n) + 2^{\text{poly}(1/\epsilon)}\right)\right)$$

S. Rao i **W.D. Smith** Approximating geometrical graphs vis ”spanners” and ”banyans”. In *Proc. 30th A CM Symposium on the Theory of Computing*, pp. 540 – 550, 1998.

Algorytm FPTAS dla problemu plecakowego

Punktem wyjścia do tworzenia algorytmów FPTAS dla problemów, które nie są silnie **NP**-zupełne (takim jest problem plecakowy), jest istnienie algorytmu pseudowielomianowego.

Niech P będzie maksymalnym zyskiem z zabrania pojedynczego przedmiotu.

$$P = \max_{a \in S} \text{profit}(a)$$

Wtedy nP jest ograniczeniem górnym wartości rozwiązania problemu plecakowego.

Dla dowolnego $i \in \{1, \dots, n\}$ i $p \in \{1, \dots, nP\}$ niech

$S_{i,p}$ będzie najmniejszym podzbiorem zbioru a_1, \dots, a_i dającym zysk p ,

$A(i, p)$ oznacza rozmiar zbioru $S_{i,p}$ (jeśli $S_{i,p}$ nie istnieje, to $A(i, p) = \infty$).

Korzystając z wzoru rekurencyjnego

$$A(i+1, p) = \left\{ \begin{array}{ll} \min \left\{ A(i, p), \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1})) \right\} & \text{jeśli } \text{profit}(a_{i+1}) \leq p \\ A(i, p) & \text{w p.p.} \end{array} \right\}$$

obliczamy wszystkie $A(i, p)$ w łącznym czasie $O(n^2P)$.

Korzystając z wzoru rekurencyjnego

$$A(i+1, p) = \begin{cases} \min \left\{ A(i, p), \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1})) \right\} & \text{jeśli } \text{profit}(a_{i+1}) \leq p \\ A(i, p) & \text{w p.p.} \end{cases}$$

obliczamy wszystkie $A(i, p)$ w łącznym czasie $O(n^2P)$.

Pytanie 1. Co uzyskamy jeśli zyski odpowiadające przedmiotom będą ograniczone wielomianową funkcją n ?

jaki algorytm uzyskamy?

Korzystając z wzoru rekurencyjnego

$$A(i+1, p) = \begin{cases} \min \left\{ A(i, p), \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1})) \right\} & \text{jeśli } \text{profit}(a_{i+1}) \leq p \\ A(i, p) & \text{w p.p.} \end{cases}$$

obliczamy wszystkie $A(i, p)$ w łącznym czasie $O(n^2P)$.

Pytanie 1. Co uzyskamy jeśli zyski odpowiadające przedmiotom będą ograniczone wielomianową funkcją n ?

jaki algorytm uzyskamy?

Wielomianowy (?)

Korzystając z wzoru rekurencyjnego

$$A(i+1, p) = \begin{cases} \min \left\{ A(i, p), \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1})) \right\} & \text{jeśli } \text{profit}(a_{i+1}) \leq p \\ A(i, p) & \text{w p.p.} \end{cases}$$

obliczamy wszystkie $A(i, p)$ w łącznym czasie $O(n^2P)$.

Pytanie 1. Co uzyskamy jeśli zyski odpowiadające przedmiotom będą ograniczone wielomianową funkcją n ?

jaki algorytm uzyskamy?

Wielomianowy (?)

Pytanie 2. Dlaczego?

Korzystając z wzoru rekurencyjnego

$$A(i+1, p) = \begin{cases} \min \left\{ A(i, p), \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1})) \right\} & \text{jeśli } \text{profit}(a_{i+1}) \leq p \\ A(i, p) & \text{w p.p.} \end{cases}$$

obliczamy wszystkie $A(i, p)$ w łącznym czasie $O(n^2 P)$.

Pytanie 1. Co uzyskamy jeśli zyski odpowiadające przedmiotom będą ograniczone wielomianową funkcją n ?

jaki algorytm uzyskamy?

Wielomianowy

Pytanie 2. Dlaczego?

ponieważ mielibyśmy wtedy $O(n^2 \cdot \text{poly}(n))$

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Dokładność rozwiązania ϵ .

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Dokładność rozwiązania ϵ .

Pytanie 5. Od czego zatem otrzymane liczby są wielomianowo zależne?

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Dokładność rozwiązania ϵ .

Pytanie 5. Od czego zatem otrzymane liczby są wielomianowo zależne?

...od n i $1/\epsilon$, a to pozwala na znalezienie rozwiązania o wartości co najmniej $(1 - \epsilon) \cdot OPT$ w czasie wielomianowym ze względu na n i $1/\epsilon$.

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Dokładność rozwiązania ϵ .

Pytanie 5. Od czego zatem otrzymane liczby są wielomianowo zależne?

...od n i $1/\epsilon$, a to pozwala na znalezienie rozwiązania o wartości co najmniej $(1 - \epsilon) \cdot OPT$ w czasie wielomianowym ze względu na n i $1/\epsilon$.

Pytanie 6. Jakiego typu będzie to algorytm? ...gdzie rozwiązanie jest wielomianowo zależne od n i $1/\epsilon$?

Pytanie 3. Jak sprowadzić zyski ($profit(a_i)$) do wartości wielomianowych?

Pomijamy końcowe bity liczb $profit(a_i)$, dzięki czemu są one wielomianowe.

Pytanie 4. Co określa liczba pominiętych bitów? od czego może być/jest zależna ta liczba?

Dokładność rozwiązania ϵ .

Pytanie 5. Od czego zatem otrzymane liczby są wielomianowo zależne?

...od n i $1/\epsilon$, a to pozwala na znalezienie rozwiązania o wartości co najmniej $(1 - \epsilon) \cdot OPT$ w czasie wielomianowym ze względu na n i $1/\epsilon$.

Pytanie 6. Jakiego typu będzie to algorytm? ...gdzie rozwiązanie jest wielomianowo zależne od n i $1/\epsilon$?

Algorytm FPTAS.

Algorytm FPTAS dla problemu plecakowego (PP)

Dla danego $\epsilon > 0$, niech $K = \frac{\epsilon P}{n}$.

Dla każdego przedmiotu a_i niech $profit'(a_i) = \lfloor \frac{profit(a_i)}{K} \rfloor$.

Korzystając z PD , znajdź najkorzystniejszy zbiór S' dla PP z zyskami $profit(a_i)$.

Wypisz S'

Algorytm działa w czasie

$$O\left(n^2 \lfloor \frac{n}{\epsilon} \rfloor\right)$$