

Przykład.

Procedura rozwiązująca PROBLEM PLECAKOWY

$f(i, l) \quad i = 0, 1, \dots, n \quad l = 0, 1, 2, \dots, b$ (el. ze zb. $\{1, \dots, i\}$ umieszczane są w plecaku o rozm. l)

$f(0, l) = 0 \quad \forall l$

$f(i, 0) = 0 \quad \forall i$

$f(i, l) = f(i-1, l) \quad \text{jeśli } l < s(a_i) \quad (\text{tzn. gdy element jest większy od rozmiaru plecaka})$

$f(i, l) = \max\{f(i-1, l-s(a_i)) + w(a_i), f(i-1, l)\} \quad \text{jeśli } l \geq s(a_i)$

Optimum dla $f(n, b)$

$i \backslash l$	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	3	3	3	3	3	3
2	0	0	0	4	4	4	4	4	7	7	7
3	0	0	2	4	4	6	6	6	7	7	9
4	0	0	2	4	6	6	8	10	10	12	12
5	0	0	2	4	6	6	8	10	10	12	12

Obliczmy złożoność obliczeniową. Liczba elementarnych operacji, które należy wykonać zależy wielomianowo od liczby elementów tablicy określającej wartości $f(i, l)$, ta zaś jest $O(nb)$. Zauważmy, że **nie jest to jednak algorytm wielomianowy**.

c) Klasa problemów silnie NP-zupełnych.

Problemy NP-zupełne są trudne obliczeniowo. Dla niektórych z nich algorytmy oparte na programowaniu dynamicznym wykazują zadziwiająco korzystne właściwości obliczeniowe. Należy do nich np. PROBLEM PLECAKOWY.

Przykład.

Algorytmy takie jak przedstawione w powyższym przykładzie noszą nazwę **pseudowielomianowych** gdyż ich funkcja złożoności obliczeniowej jest ograniczona od góry przez wielomian zależący od dwóch zmiennych: rozmiaru instancji $N(I)$ oraz maksymalnych wartości występujących w tym problemie liczb $Max(I)$. W praktyce $Max(I)$ przyjmuje skończone wartości i algorytmy te mają korzystne własności z punktu widzenia czasu obliczeń. Ponieważ liczby są kodowane przy podstawie większej od 1, zatem długość łańcucha symboli użytego do zakodowania $Max(I)$ wynosi $\lfloor \log(Max(I)) \rfloor + 1$ i złożoność algorytmu wielomianowego byłaby $O(p(N(I), \lfloor \log(Max(I)) \rfloor + 1))$ a nie $O(p(N(I), Max(I)))$, jak w przykładzie.

Def. (problemów liczbowych) Algorytmy pseudowielomianowe można ewentualnie skonstruować jedynie dla **liczbowych** problemów (decyzyjnych) Π , to znaczy takich, dla których nie istnieje wielomian p taki, że $Max(I) \leq p(N(I))$ dla każdego $I \in D_\Pi$. Do problemów liczbowych należą m.in. problemy: PLECAKOWY, KOMIWOJAŻERA, PODZIAŁU ZBIORU NA TRÓJELEMENTOWE PODZBIORY (tzn. PROBLEM TRÓJPODZIAŁU).

Gdyby istniało ograniczenie $Max(I) \leq p(N(I))$, to otrzymalibyśmy

$$\begin{array}{c} \underbrace{O(p(N(I), Max(I)))}_{Max(I) \leq p(N(I))} \\ \downarrow \\ O(p1, N(I)) \quad - \text{wielomian} \\ \downarrow \\ P=NP \quad - \text{a to jest fałsz} \end{array}$$

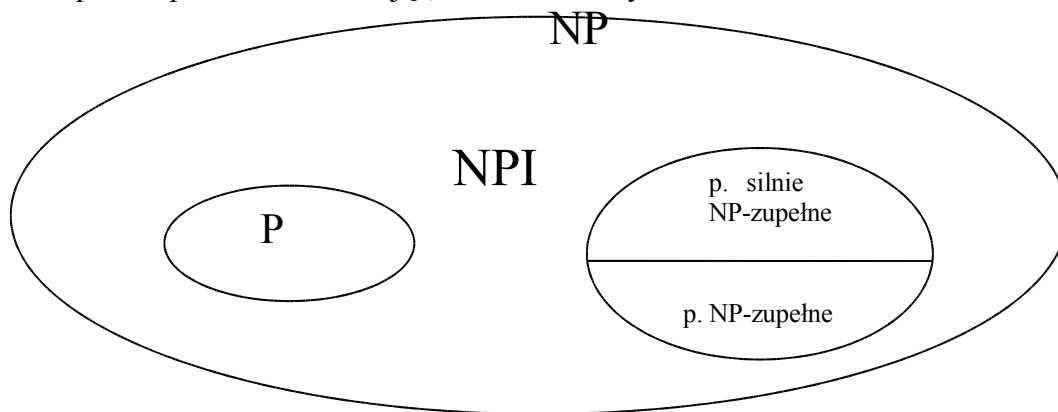
Nie są natomiast problemami liczbowymi np. problemy: SPEŁNIALNOŚCI, MAKSYMALNEGO SKOJARZENIA, KOLOROWANIA GRAFU.

Dla dowolnego problemu decyzyjnego Π i dowolnego wielomianu p określonego dla liczb całkowitych, niech Π_p oznacza podproblem Π otrzymany przez ograniczenie D_Π tylko do tych instancji, dla których $Max(I) \leq p(N(I))$. (Π_p nie jest problemem liczbowym).

Def. Problem decyzyjny Π jest **silnie NP-zupełny**, jeśli należy do NP i istnieje wielomian p , określony dla liczb całkowitych, dla którego Π_p jest NP-zupełny.

Problem NP-zupełny+nieliczbowy=problem silnie NP-zupełny.

Jeśli problem Π jest silnie NP-zupełny, to istnienie dla niego algorytmu pseudowielomianowego jest równoważne istnieniu algorytmów wielomianowych dla wszystkich problemów NP-zupełnych czyli równości $P=NP$, jest więc równie mało prawdopodobne. Zakładając, że $P \neq NP$ mamy:



Aby wykorzystać **silną NP-zupełność** można posłużyć się pojęciem **transformacji pseudowielomianowej**, która pozwala uniezależniać się od znajdowania podproblemu Π_p .

Def. Pseudowielomianową transformacją problemu Π_2 do problemu Π_1 nazywamy funkcję $f: D_{\Pi_2} \rightarrow D_{\Pi_1}$ taką, że:

1. dla każdej instancji $I_2 \in D_{\Pi_2}$ odpowiedź brzmi „tak” \Leftrightarrow gdy dla $f(I_2)$ brzmi także „tak”,
2. funkcja f może być obliczona (przez DTM) w czasie ograniczonym od góry przez wielomian zależny od dwóch zmiennych: $Max_2(I_2)$ i $N_2(I_2)$,
3. istnieje wielomian q_1 taki, że dla każdego $I_2 \in D_{\Pi_2}$: $q_1(N_1(f(I_2))) \geq N_2(I_2)$,

istnieje wielomian dwóch zmiennych q_2 taki, że dla każdego $I_2 \in D_{\Pi_2}$:

$$Max_1(f(I_1)) \leq q_2(Max_2(I_2), N_2(I_2)).$$

Gdy porównamy powyższą definicję z definicją transformacji wielomianowej widzimy, że osłabiony został warunek wielomianowego czasu realizacji algorytmu obliczającego funkcję f . W zamian za to żąda się, by rozmiar instancji nie zmalał wykładniczo, a największa wartość liczbową zawartą w danych nie wzrosła wykładniczo.

Twierdzenie.

Można wykazać, że jeśli problem Π_2 jest silnie NP-zupełny, $\Pi_1 \in NP$ oraz istnieje **pseudowielomianowa transformacja** problemu Π_2 do Π_1 , to problem Π_1 jest **silnie NP-zupełny** [G.J.78]

Ponieważ problemy silnie NP-zupełne pozostałyby NP-zupełnymi nawet w przypadku kodowania jedynekowego, dlatego nazywa się je czasem **jedynekowo NP-zupełnymi**.

4. ANALIZA ZŁOŻONOŚCI I METODYKA ROZWIĄZYWANIA PROBLEMÓW OPTYMALIZACYJNYCH.

a) Analiza złożoności obliczeniowej problemów optymalizacyjnych.

Przeanalizujmy teraz wielomianową równoważność problemów decyzyjnych i optymalizacyjnych.

Def. Przez **problem przeszukiwania** π rozumiemy zbiór instancji D_π i zdefiniowany dla każdej instancji $I \in D_\pi$ zbiór rozwiązań $Z_\pi(I)$ (instancje i zbiór rozwiązań są skończone). Powiemy, że algorytm rozwiązuje problem przeszukiwania Π_1 jeśli dla każdej instancji $I \in D_\pi$ odpowiedź brzmi „nie” jeśli $Z_\pi(I) = \emptyset$, a w przeciwnym razie algorytm znajduje jedno z rozwiązań należących do zbioru $Z_\pi(I)$.

Każdy problem decyzyjny Π można łatwo przedstawić w postaci problemu przeszukiwania, definiując $Z_\pi(I) = \{ „tak” \}$ jeśli $I \in Y_\pi$ oraz $Z_\pi(I) = \emptyset$ jeśli $I \notin Y_\pi$.

Aby wykazać równoważność problemów przeszukiwania posłużmy się wielomianową **transformacją Turinga**.

Def. Wielomianową transformacją Turinga problemu przeszukiwania Π_1 do problemu przeszukiwania Π_2 (oznaczenie $\Pi_1 \alpha_T \Pi_2$), nazywać będziemy algorytm rozwiązujący problem Π_1 przy wykorzystaniu hipotetycznej procedury Π_2 przy czym czas wykonania algorytmu A przez DTM w czasie wielomianowym.

W przypadku istnienia takiej transformacji będziemy mówili, że problem Π_1 jest T-transformowalny przez problem Π_2 .

Pojęcie to można też zdefiniować wykorzystując maszynę Turinga z wyrocznią (OTM). Wówczas wielomianową transformację Turinga problemu π_1 do problemu π_2 nazywamy program (algorytm) M , który dla alfabetu Σ spełnia warunek - dla każdej funkcji $\Sigma^* \rightarrow \Sigma^*$, która rozwiązuje π_2 , program M_g jest wielomianowym programem dla OTM, a funkcja f_M^g obliczana przez M_g rozwiązuje π_1 .

Def. Problem przeszukiwania π jest **NP-trudny** jeśli istnieje NP-zupełny problem π' , taki że $\pi' \alpha_T \pi$.

Zauważmy, że NP-trudnego problemu przeszukiwania Π nie będzie można rozwiązać w wielomianowym czasie, o ile zostanie wykazane, że $P=NP$. Jednakże, w przeciwieństwie do problemów NP-zupełnych odwrotne stwierdzenie nie jest prawdziwe i wykorzystanie równości klas $P=NP$ nie przesądza o możliwości rozwiązania problemów NP-trudnych w wielomianowym czasie.

Przeanalizujmy raz jeszcze kwestię wzajemnych zależności NP-zupełnych problemów decyzyjnych i odpowiadających im problemów przeszukiwania. Łatwo można zauważyć, że wykazując iż wielomianowy algorytm rozwiązujący problem przeszukiwania można wykorzystać do rozwiązania w wielomianowym czasie odpowiadającego mu problemu decyzyjnego podajemy jednocześnie transformację Turinga od problemu decyzyjnego do problemu przeszukiwania. Ponieważ ten pierwszy problem jest NP-zupełny, zatem problem przeszukiwania jest NP-trudny.

Okazuje się, że wiele problemów decyzyjnych i optymalizacyjnych jest związana jeszcze ściślej. Można bowiem wykazać, że problemy decyzyjne są także nie łatwiejsze niż odpowiedniki optymalizujące.

Rozpatrzmy twierdzenie.

Twierdzenie 6.

Optymalizacyjny problem plecakowy (sformułowany jako problem przeszukiwania) jest T-transformowalny do decyzyjnego PROBLEMU PLECAKOWEGO.

Dowód:

Zacznijmy od zdefiniowania problemu pośredniego między dwoma problemami wymienionymi w tezie twierdzenia, będącego rozszerzeniem problemu decyzyjnego.

ROZSZERZONY PROBLEM PLECAKOWY (RPP)

Parametry:

skończony zbiór elementów $a = \{a_1, \dots, a_n\}$, rozmiar $s(a_i)$ oraz wartość $w(a_i) \in \mathbb{Z}^+$ dla każdego elementu $a_i \in a$, stałe b, y oraz podzbiór $A_k \subset A$, $A_k = \{a_{[1]}, \dots, a_{[k]}\}$, gdzie $a_{[i]} \neq a_{[j]}$ dla $i \neq j$ oraz $a_{[i]} \in A$ dla $i = 1, 2, \dots, k$.

Pytanie:

Czy podzbiór A_k można rozszerzyć do podzbioru $A' \subset A$ takiego, że

$$\sum_{a_i \in A'} s(a_i) \leq b$$

$$\sum_{a_i \in A'} w(a_i) \geq b$$

$RPP \in NP$ stąd i z definicji NP-zupełności mamy $RPP \alpha PP$ ($PP \equiv$ problem plecakowy). Zatem $RPP \alpha_T PP$. Pozostaje wykazać, że $OPP \alpha_T RPP$ ($OPP \alpha_T DRPP$).

Założmy, że $P[A, s, w, \bar{A}_k, \bar{b}, y]$ jest procedurą rozwiązującą RPP w wielomianowym czasie

dla następujących parametrów: zbiór elementów A , funkcja rozmiarów s i wartości w elementów, podzbiór częściowy A_k oraz stałe b i y .

Dla dowolnej instancji OPP określmy optymalną (maksymalną) wartość y^* sumy wartości elementów tworzących podzbiór zbioru A , spełniający ograniczenie i zawierający element a_i . Wartość y_i^* jest zawarta między dwiema wartościami granicznymi:

$$y_{i \min} = \min \{w(a_i)\} \text{ a } y_{i \max} = n^* \min \{w(a_i)\}.$$

Poniższa procedura przeszukiwania binarnego dla określonego w powyższy sposób przedziału znajduje y_i^* .

1. Jeśli $y_{i \max} - y_{i \min} = I$ to podstaw $y_i^* := y_{i \min}$ i zakończ procedurę.

2. Podstaw $y_i := \lceil (y_{i \max} + y_{i \min})/2 \rceil$ i wykonaj procedurę $P[A, s, w, \{a_{ij}\}, b, y_i]$. Jeśli odpowiedź brzmi „tak”, to $y_{i \min} := y_i$ w przeciwnym razie $y_{i \max} := y_i$. Idź do kroku 1.

(Aby znaleźć y_i^* należało dokonać co najwyżej $\lceil \log_2 y_{i \max} \rceil$ razy procedurę $P[A, s, w, \{a_{ij}\}, \bar{b}, \bar{y}]$ dla różnych wartości y_i .)

Postępowanie to powtarzamy dla każdego $a_i \in A$ i znajdujemy y_i^* .

W następnej fazie określamy maksymalną wartość $y_{i \max}^*$ spośród określonych uprzednio y_i^* oraz odpowiadający jej element a_i .

Następnie przechodzimy do wyznaczenia optymalnego podzbioru A^* . Oznaczmy przez $A_k \subset A$ rozszerzalny podzbiór elementów A , który można rozszerzyć przez dodanie pewnych elementów i utworzyć w ten sposób szukany podzbiór A^* . Zauważmy, że $\{a_{ij}\}$ jest rozszerzalnym podzbiorem, o ile $w(a_i) < y_{i \max}^*$. W tym przypadku musi istnieć co najmniej jeden element $a_j \in A - \{a_i\}$, taki, że $\{a_i, a_j\}$ jest rozszerzalnym podzbiorem lub $\{a_i, a_j\} = A^*$. W celu

znalezienia elementu a_j należy wykonać co najwyżej $n-2$ razy procedurę $P[A, s, w, \{a_i, a_j\}, b, y_{i \max}^*]$ dla różnych $a_j \in A - \{a_i\}$.

Powyższe postępowanie powtarzamy dla kolejnych rozszerzalnych podzbiorów A_k dopóty, dopóki nie skonstruujemy podzbioru A_k^* , dla którego zachodzi:

$$\sum_{a_i \in A_k^*} w(a_i) = y_{i \max}^*.$$

Konstrukcja A_k^* wymaga zatem $O(n^2)$ wykonań procedury p poza tymi, które zostały wykorzystane do określenia $y_{i \max}^*$.

Reasumując wykorzystaliśmy $OOP \propto_T RPP$.

Jeżeli wskażemy, że jakiś problem optymalizacyjny jest T-transformowalny do pewnego należącego do klasy NP, to z tego wynika iż może on być rozwiązany w wielomianowym czasie jeśli $P=NP$.

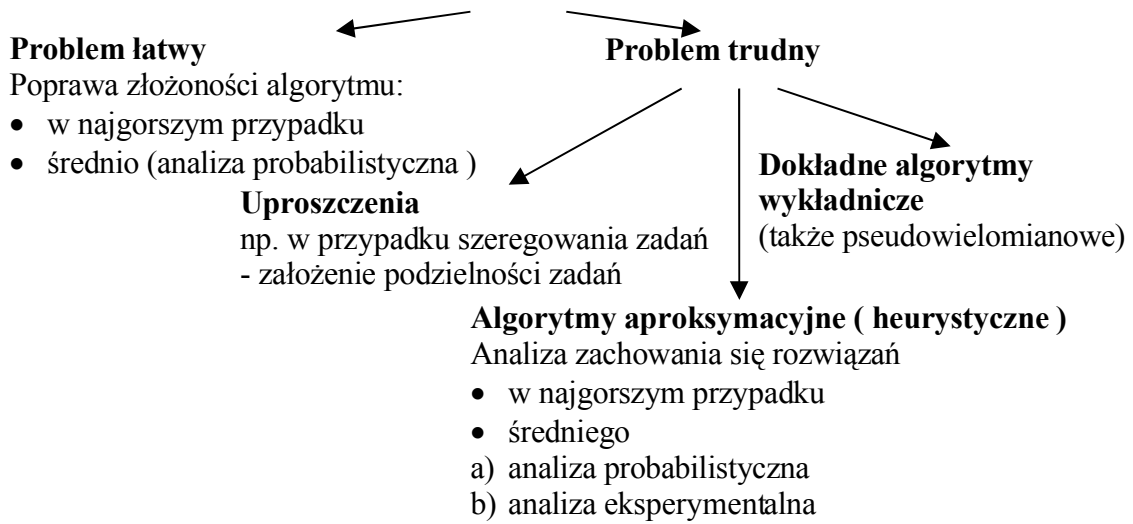
Def. Problem przeszukiwania Π_1 nazywać będziemy **NP-łatwym**, jeśli istnieje problem $\Pi_2 \in NP$ taki, że $\Pi_1 \propto_T \Pi_2$.

Def. Problemy przeszukiwania będące jednocześnie NP-trudnymi i NP-łatwymi nazywać będziemy problemami **NP-równoważnymi**.

Z rozważań tych wynika, że początkowe ograniczenie analizy złożoności obliczeniowej do problemów decyzyjnych nie spowodowało utraty ogólności, gdyż większość problemów optymalizacyjnych, których decyzyjne odpowiedniki są NP-zupełne, to problemy NP-równoważne.

Analogicznie do pojęcia NP-trudności definiuje się pojęcie **silnej NP-trudności**.

Analiza złożoności problemu kombinatorycznego (optymalizacyjnego)



Analiza trudnego problemu optymalizacyjnego

1.Osłabienie (uproszczenie) założeń przyjętych w problemie pierwotnym

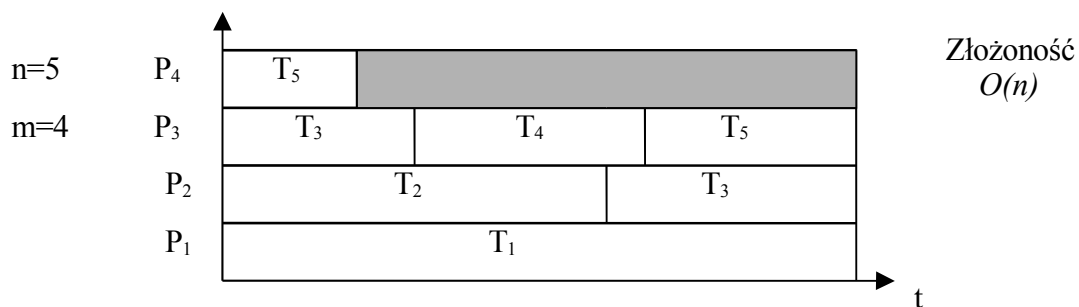
Na przykład w przypadku problemu szeregowania

- zezwolenie na przerywanie wykonywanych zadań
- przyjęcie jednostkowych czasów wykonywania zadań
- założenie pewnych typów grafów ograniczeń kolejnościowych (w przypadku zadań zależnych)

Przykład

ad a) **Algorytm McNaughton'a**

- Wyznacz $C_{\max}^* = \max \left\{ \max_j \{p_j\}, \frac{1}{m} \sum_{j=1}^n p_j \right\}$
- Rozpocznij wykonywanie dowolnego zadania na dowolnym procesorze w chwili $t=0$.
- Wybierz dowolne, nie uszeregowane jeszcze zadanie i rozpocznij jego wykonywanie na tym samym procesorze w chwili zakończenia poprzedniego zadania. Powtarzaj krok do momentu, w którym zostaną wybrane wszystkie zadania lub $t=C_{\max}^*$.
- Pozostałą część zadania przydziel do innego procesora w chwili $t=0$. Przejdź do kroku 3.



ad b) **Wykorzystanie algorytmów przybliżonych (aproksymacyjnych)**

NP-trudny problem optymalizacyjny

Optymalizacyjny alg. wykładniczy

- metoda podziału i ograniczeń
- programowanie dynamiczne
- całkowitoliczbowe programowanie liczbowe

Aproksymacyjny algorytm wielomianowy

Ocena dokładności algorytmów aproksymacyjnych jest to ocena oddalenia od optimum wartości rozwiązań konstruowanych przez te algorytmy

- ocena najgorszego przypadku:
- ocena zachowania się średniego: probabilistyczna, eksperymentalna.

1° Oszacowanie najgorszego przypadku

Def. Problem (kombinatoryczny) optymalizacji (optymalizacyjny) Π definiuje się przez podanie:

- 1 zbioru D_π instancji;
- 1 dla każdej instancji $I \in D_\pi$, skończonego zbioru $Z_\pi(I)$ potencjalnych rozwiązań;
- 1 funkcji f_π określonej dla każdej instancji $I \in D_\pi$ i każdego potencjalnego rozwiązania $z \in Z_\pi(I)$, wartość tego rozwiązania $f_\pi(I, z)$ będącego liczbą wymierną.

Jeżeli Π jest problemem minimalizacji (maksymalizacji)

Def. wówczas **optymalnym rozwiązaniem** dla instancji $I \in D_\pi$ nazywać będziemy potencjalne rozwiązanie $z^* \in Z_\pi(I)$ takie, że dla każdego innego rozwiązania $z \in Z_\pi(I)$ zachodzi:

$f_\pi(I, z^*) \leq f_\pi(I, z)$ [$f_\pi(I, z^*) \geq f_\pi(I, z)$]. Wartość funkcji f_π dla rozwiązania optymalnego z^* dla instancji I , oznaczać będziemy przez $OPT(I)$.

Def. Algorytm A nazywać będziemy **algorytmem aproksymacyjnym (suboptymalnym)** dla problemu Π , jeśli dla każdej instancji $I \in D_\pi$ znajduje on potencjalne rozwiązanie $z \in Z_\pi(I)$. (Dalej przez $A(I)$ oznaczać będziemy wartość $f_\pi(I, z)$ tego rozwiązania).

Def. Jeśli algorytm A znajduje rozwiązanie optymalne (tzn. $A(I) = OPT(I)$) dla każdej instancji $I \in D_\pi$, to nazywać go będziemy **algorytmem optymalizacyjnym** dla problemu π .

Wykorzystanie algorytmów aproksymacyjnych

Warunek konieczny stosowania: niski rząd funkcji złożoności obliczeniowej.

Warunek dostateczny: wystarczająca dokładność (odległość od optimum konstruowanych rozwiązań)

Analiza najgorszego przypadku

Niech Π będzie problemem minimalizacji (maksymalizacji), a $I \in D_\pi$, wówczas

$$S_A(I) = \frac{A(I)}{OPT(I)}; \quad (S_A(I) = \frac{OPT(I)}{A(I)})$$

gdzie $A(I)$ - wartość rozwiązania skonstruowanego przez algorytm dla instancji $I \in D_\pi$

$OPT(I)$ - wartość rozwiązania optymalnego dla I .

Def. **Bezwzględne oszacowanie** S_A algorytmu aproksymacyjnego A zastosowanego do rozwiązania problemu Π :

$S_A = \inf \{ r \geq 1 : S_A(I) \leq r \text{ dla każdej instancji } I \in D_\pi \}$ - jest to największa wartość $S_A(I)$ dla jakiegokolwiek instancji tego problemu

Def. **Asymptotyczne oszacowanie** S_A^∞ algorytmu aproksymacyjnego A zastosowanego do rozwiązania problemu π :

$$S_A^1 = \inf \{ r \mid 1: \text{ dla pewnej naturalnej liczby } N, S_A(I) \leq r \text{ dla każdej instancji}$$

$I \in D_\pi$ spełniającej warunek $OPT(I) \leq N \}$ - dotyczy b. dużych rozwiązań

$S_A^\infty \leq S_A$. Im S_A^∞ (lub S_A) jest bliższe 1, tym lepszy jest dany algorytm aproksymacyjny.

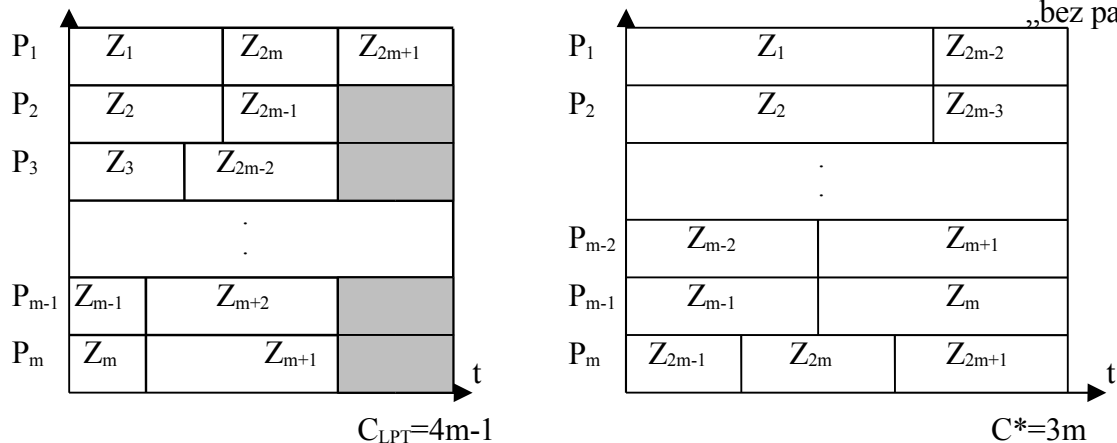
Przykład Algorytm LPT [Graham 69]

1. Ustaw zadania na liście w kolejności malejących czasów wykonania.
2. W momencie, w którym wolny staje się którykolwiek z procesorów, przydziel do niego pierwsze na liście nie przydzielone zadanie. Krok ten powtarzaj do momentu wykonania wszystkich zadań.

Twierdzenie [Graham '69]

$$S_{LPT} = \frac{4}{3} - \frac{1}{3m}$$

$n = 2m + 1, (p_1, p_2, \dots, p_n) = (\underbrace{2m-1, 2m-1, 2m-2, 2m-2, \dots, m+1, m+1, m, m, m}_{\text{zadania mające parami ten sam czas wykonania}}, \underbrace{m}_{\text{zadanie „bez pary”}})$



W najgorszym przypadku uszeregowanie LPT może być dłuższe o 33% od uszeregowania optymalnego. Gdy liczba zadań jest dłuższa oszacowanie algorytmem LPT znacznie się poprawia.

Twierdzenie 3 [Coffman, Sethi '76]

$$S_{LPT}(k) = 1 + \frac{1}{k} - \frac{1}{mk}$$

gdzie k jest najmniejszą liczbą zadań wykonywanych na jednym procesorze (algorytm jest dobry w przypadku zadań o zbliżonej długości, a nie, gdy jest jedno lub kilka zadań b. długich, a reszta krótkich).

Def. **Aproksymacyjnym schematem obliczeń** dla problemu optymalizacyjnego Π nazwiemy algorytm A , który dla danej $I \in D_\Pi$ oraz żądanej dokładności obliczeń $\varepsilon > 0$ znajduje potencjalne rozwiązanie $z \in Z_\Pi(I)$, takie, że $S_{A\varepsilon}(I) \leq I + \varepsilon$.

Def. A jest **wielomianowym aproksymacyjnym schematem obliczeń**, jeśli dla każdego $\varepsilon > 0$ jest algorytmem wielomianowym.

Przykład [Sahni '75]

Problem plecakowy można rozwiązać stosując następujący wielomianowy aproksymacyjny schemat obliczeń AP_k

- wygenerować wszystkie k -elementowe podzbiory zbioru elementów A ,
- rozszerzyć każdy z powyższych zbiorów elementami dokładanymi w kolejności nie rosnących $\frac{w(a_i)}{s(a_i)}$

Można wykazać, że:

$$S_{AP_k} \leq 1 + \frac{1}{k}$$

opisany schemat ma jednak dużą złożoność obliczeniową, gdyż zależy ona wykładniczo od k (liczba zbiorów k -elementowych $O(n^k)$).

Def. Algorytm A jest **w pełni wielomianowym aproksymacyjnym schematem obliczeń**, jeśli jego funkcja złożoności obliczeniowej jest ograniczona od góry przez wielomian zależny od $N(I)$ oraz $(1/\varepsilon)$.

Przykład [Ibarra, Kim '77]

Problem plecakowy można rozwiązać stosując następujący w pełni wielomianowy schemat obliczeń B_k :

- przekształcamy instancję I problemu plecakowego w I' przez przyjęcie $w'(a_i) = \lfloor w(a_i)/K \rfloor$, K - pewna stała,
- rozwiążmy problem plecakowy dla instancji I' wykorzystując programowanie dynamiczne, w którym wyznaczać się będzie funkcję $g(i,j) = 1 \Leftrightarrow$, gdy spośród elementów $\{1, \dots, i\}$ można wybrać podzbiór o wartości j .

Złożoność tej procedury jest $O\left(p\left(n \frac{W}{K}\right)\right)$, gdzie $W = \max\{w(a_i)\}$,

- przyjmując $K = W / [(k+1) * n]$ można wykazać, że $S_{B_k}(I) \leq 1 + \frac{1}{k}$, a złożoność obliczeniowa jest $O(p, (n^2 k))$, gdzie k jest pewną stałą całkowitą, dodatnią.

Algorytmy aproksymacyjne nie konstruują rozwiązań o takiej samej dokładności (o tym samym oszacowaniu) dla różnych problemów, których decyzyjne odpowiedniki wielomianowo transformują się do siebie. Jest to przyczyną istnienia problemów, dla których można skonstruować algorytmy aproksymacyjne o całkiem dobrych oszacowaniach jak na przykład PROBLEM PAKOWANIA dla którego $S_A \approx 1,2$ dla najlepszych algorytmów, a także problemów, dla których nie istnieją, jak na razie, algorytmy aproksymacyjne o oszacowaniu $S_A < 1$, jak na przykład problem znajdowania MAKSYMALNEGO ZBIORU NIEZALEŻNEGO W GRAFIE. Istnieją jednak z tego samego powodu problemy które w sensie istnienia wielomianowych algorytmów aproksymacyjnych o oszacowaniach bliskich jedności są łatwiejsze nawet od wspomnianego PROBLEMU PAKOWANIA.

NP-trudność problemów aproksymacji.

Def. Najlepsze osiągalne asymptotyczne oszacowanie dla problemu optymalizacyjnego Π definiuje się jako $S_{min}(\Pi) = \inf\{r \geq 1 : \text{istnieje wielomianowy, aproksymacyjny schemat obliczeń dla problemu } \Pi \text{ taki, że } S_A^\infty = r\}$

Z poprzedzających rozważań wynika, że istnieją problemy, dla których $S_{min}(\Pi) = \infty$, $1 < S_{min}(\Pi) = \infty$, $S_{min}(\Pi) = 1$.

W ostatnim przypadku ($S_{min}(\Pi) = 1$) możemy mieć do czynienia z następującymi przypadkami:

- dla Π istnieje wielomianowy aproksymacyjny schemat obliczeń,
- dla Π istnieje w pełni wielomianowy aproksymacyjny schemat obliczeń,
- Π można rozwiązać za pomocą wielomianowego algorytmu aproksymacyjnego A , dla którego $|A(I) - OPT(I)| \leq k$ dla pewnej stałej k .

Jakość aproksymacji jest najlepsza w ostatnim przypadku.

Dla wielu problemów nie istnieje określony powyżej typ aproksymacji o ile nie zostanie udowodniona równość $P = NP$.

Twierdzenie.

Jeśli $P \neq NP$ to nie istnieje wielomianowy aproksymacyjny algorytm A rozwiązujący problem plecakowy w sposób taki, że $|A(I) - OPT(I)| \leq k$

Twierdzenie [Garey, Johnson '75]

Niech Π oznacza problem optymalizacyjny. Jeśli istnieje wielomian q taki, że

$$\forall_{I \in D_\Pi} OPT(I) < q[N(I), Max(I)]$$

to istnienie w pełni wielomianowego aproksymacyjnego schematu obliczeniowego dla Π pociągałoby za sobą istnienie pseudowielomianowego algorytmu optymalizacyjnego dla Π .

Wniosek:

Niech Π będzie problemem optymalizacyjnym o całkowitoliczbowych rozwiązaniach, spełniającym warunki powyższego twierdzenia. Jeśli problem Π jest silnie NP-trudny, to istnienie w pełni wielomianowego aproksymacyjnego schematu obliczeń pociągałoby za sobą $P=NP$.

Twierdzenie.

Niech Π będzie problemem minimalizacji o całkowitoliczbowych dodatnich rozwiązaniach. Przypuśćmy, że dla pewnej stałej $K \in \mathbb{N}$ problem decyzyjny: „Czy dla $I \in D_\Pi$ zachodzi $OPT(I) \leq K$?” jest NP-zupełny. Jeśli $P \neq NP$, to nie istnieje wielomianowy algorytm aproksymacyjny A dla Π taki, że $S_A < 1 + \frac{1}{K}$, dla Π nie istnieje też wielomianowy aproksymacyjny schemat obliczeń.

Przykład.

Problem 3-kolorowania grafu jest NPc, dlatego, o ile $P \neq NP$, to nie istnieje algorytm A taki, że $S_A < \frac{4}{3}$, nie istnieje też wielomianowy schemat obliczeń.

21 Analiza zachowania się średniego

a) Analiza probabilistyczna

Założenie:

parametry instancji rozpatrywanego problemu Π generowane są z pewnego rozkładu prawdopodobieństwa F . Przyjmuje się, że generowane wartości poszczególnych parametrów są realizacjami niezależnych zmiennych losowych o takich samych rozkładach.

Np.: dla PROBLEMU PLECAKOWEGO zarówno wartości elementów $w(z_1), w(z_2), \dots$ jak i ich rozmiary $s(a_1), s(a_2), \dots$, są niezależnymi zmiennymi losowymi generowanymi zazwyczaj z rozkładu jednorodnego w przedziale $[0, 1]$.

Dla tak określonej instancji I_n , gdzie n jest liczbą generowanych wartości parametrów, przeprowadza się probabilistyczną analizę wartości, która pozwala wyrazić w prosty sposób asymptotyczną wartość optimum funkcji celu $OPT(I_n)$ za pomocą parametrów problemu.

Następnie dokonuje się probabilistycznej oceny badanego algorytmu aproksymacyjnego A , traktując wartość rozwiązania konstruowanego przez ten algorytm jako zmienną losową $A(I_n)$. Wyróżnia się dwie miary oceny:

Def. pierwszą jest **błąd względny**

$$w_n = \frac{A(I_n) - OPT(I_n)}{OPT(I_n)}$$

Def. drugą miarą, silniejszą, jest **błąd bezwzględny**

$$b_n = A(I_n) - OPT(I_n) \quad \text{- minimalizacja}$$

W przypadku obu miar bada się ich zbieżność (convergence) do 0.

Wyróżnia się trzy rodzaje zbieżności:

Def. Najsilniejsza **zbieżność prawie na pewno** (almost sure convergence) dla ciągu zmiennych losowych y_n (np. ciąg wartości w_n lub b_n) zbieżnych do stałej c (jest to wartość optymalna), oznacza że:

$$\Pr\left\{\lim_{n \rightarrow \infty} y_n = c\right\} = 1$$

Def. Pociąga ona za sobą słabszą **zbieżność według prawdopodobieństwa** (convergence in probability), która oznacza, że

$$\forall \varepsilon > 0 \lim_{n \rightarrow \infty} \Pr\{|y_n - c| > \varepsilon\} = 0$$

Z tej ostatniej zbieżności wynika pierwsza, gdy dodatkowo spełniony jest warunek

$$\forall \varepsilon > 0 \sum_{j=1}^{\infty} \Pr\{|y_j - c| > \varepsilon\} < \infty$$

Def. Trzeci rodzaj zbieżności to **zbieżność według wartości średniej** (convergence in expectation) zachodzi gdy:

$$\lim_{n \rightarrow \infty} |E(y_n) - c| = 0$$

gdzie $E(y_n)$ jest wartością średnią y_n i również pociąga za sobą zbieżność według prawdopodobieństwa.

Z powyższych definicji wynika, że z punktu widzenia analizy probabilistycznej, algorytm jest najlepszy, gdy jego błąd bezwzględny jest prawie na pewno zbieżny do 0. Mówimy wówczas o **asymptotycznej optymalności algorytmu**.

Przykład

Niech $m=2$, a czasy wykonywania zadań są generowane z rozkładu równomiernego z przedziału $[0,1]$.

Twierdzenie. [Coffman, Fredericksen, Lueker '84]

$$\frac{n}{4} + \frac{1}{4(n+1)} \leq E(C_{\max}^{LPT}) \leq \frac{n}{4} + \frac{e}{2(n+1)}$$

Biorąc pod uwagę, że $\frac{n}{4}$ jest dolnym ograniczeniem wartości $E(C_{\max}^*)$ otrzymujemy

$$\frac{E(C_{\max}^{LPT})}{E(C_{\max}^*)} < 1 + O\left(\frac{1}{n^2}\right)$$

Zatem, wraz ze wzrostem n , $E(C_{\max}^{LPT})$ zdąża do wartości optymalnej nie wolniej niż $1 + O\left(\frac{1}{n^2}\right)$ zdąża do 1.

b) Analiza eksperymentalna

Porównuje się na drodze eksperymentu obliczeniowego rozwiązanie, w sensie wartości kryterium, konstruowane przez algorytm aproksymacyjny i optymalizacyjny, przy czym porównanie to powinno dotyczyć szerokiej, reprezentatywnej próbki instancji.