Algorytmy aproksymacyjne

Algorytm aproksymacyjny dla problemu optymalizacyjnego znajduje przybliżenie optymalnego rozwiązania.
Algorytm aproksymacyjny A ma *ograniczenie względne* $\rho(n)$ jeśli dla dowolnych danych rozmiaru n zachodzi:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$$

gdzie C – koszt rozwiązania znalezionego przez A, C^* – koszt optymalnego rozwiązania dla tych danych.
Dla maksymalizacji: $0 < C \leq C^*$ i $\rho(n)$ ogranicza C^*/C. Dla minimalizacji: $0 < C^* \leq C$ i $\rho(n)$ ogranicza C/C^*.
Błąd względny:

$$\frac{|C - C^*|}{C^*}$$

Algorytmy aproksymacyjne

Ograniczenie błędu względnego $\varepsilon(n)$:

$$\frac{|C - C^*|}{C*} \leq \varepsilon(n)$$

Zachodzi:

$$\varepsilon(n) \leq \rho(n) - 1$$

(*Dla minimalizacji:* $\varepsilon(n) = \rho(n) - 1$, *Dla maksymalizacji:* $\varepsilon(n) = (\rho(n) - 1)/\rho(n)$ *i* $\rho(n) \geq 1$.)

Schemat aproksymacji

Schemat aproksymacji – algorytm aproksymacyjny, który na wejściu otrzymuje dane problemu oraz wartość $\varepsilon > 0$ i wyznacza rozwiązanie o ograniczeniu błędu względnego ε.
Schemat aproksymacji jest *wielomianowy* jeśli dla ustalonego ε działa w czasie wielomianowym ze wzgl. na rozmiar danych.
Schemat aproksymacji jest *w pełni wielomianowy* jeśli działa w czasie wielomianowym ze wzgl. na rozmiar danych jak i ze wzgl. na $1/\varepsilon$.

Problem pokrycia wierzchołkowego

Optymalnym pokryciem wierzchołkowym jest najmniej liczne pokrycie wierzchołkowe.

```
Approx-Vertex-Cover(G)
C ← ∅
E' ← E[G]
while E' ≠ ∅ do
    wybierz dowolną (u, v) ∈ E'
    C ← C ∪ {u, v}
    usuń z E' krawędzie incydentne z u lub v
return C
```

Czas: wielomianowy. Po zakończeniu C jest pokryciem wierzchołkowym. (E' *się opróżni, gdy każda krawędź zostanie pokryta przez jakiś wierzchołek z* C)

Problem pokrycia wierzchołkowego

Tw. Ograniczenie względne `Approx-Vertex-Cover` wynosi 2.

D-d. Niech A – zbiór krawędzi wybranych w wierszu 4. Żadne dwie krawędzie z A nie mają wspólnego wierzchołka. Zatem najmniej liczne pokrycie wierzchołkowe G musi mieć $\geq |A|$ wierzchołków.
Po każdym wykonaniu wiersza 4 dodajemy do C dwa wierzchołki. Stąd $|C| = 2|A|$. $\Box$

Problem komiwojażera

Optymalna marszruta – cykl Hamiltona o najmniejszym koszcie. Zakładamy, że spełniona jest *nierówność trójkąta*:

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

dla dowolnych $u, v, w, \in V$.
W ogólnym przypadku (bez dodatkowych założeń) istnienie algorytmu aproksymacyjnego dla tego problemu implikuje $P = NP$ (co pokażemy później), a zatem jest mało prawdopodobne.

Approx-TSP-Tour

`Approx-TSP-Tour(`G`,`c`)`

`1` *wybierz* $r \in V[G]$ *jako "korzeń"*

`2` *zbuduj minimalne drzewo rozpinające* T *dla* G

`3` *niech* L *– wierzchołki* T *w kolejności preorder*

`4 return` L *(bez powtórzeń)*

Uwaga: W wierszu 2 można zastosować np. algorytm Prima, który w czasie wielomianowym znajduje minimalne drzewo rozpinające rozpoczynając od korzenia r.

Algorytmy konstrukcji minimalnych drzew rozpinających będą omawiane na wykładzie z teorii grafów.

Czas: wielomianowy.

Approx-TSP-Tour

Tw. `Approx-TSP-Tour` jest algorytmem aproksymacyjnym z ograniczeniem względnym 2, jeśli jest spełniona nierówność trójkąta.

D-d. Niech T – minimalne drzewo rozpinające G, Niech H^* – optymalna marszruta w G. Przez usunięcie jednej krawędzi z H^* otrzymujemy drzewo rozpinające G. Stąd $c(T) \leq c(H^*)$.

Niech W lista wierzchołków odwiedzanych przy pełnym przejściu drzewa. Każdą krawędź T przechodzimy w W dwukrotnie. Stąd $c(W) = 2c(T) \leq 2c(H^*)$.

W nie jest cyklem Hamiltona, bo powtarzają się w nim wierzchołki. Korzystając z nierówności trójkąta możemy usuwać z W wierzchołki nie zwiększając kosztu W. Zatem po usunięciu powtórzeń z W dostajemy cykl Hamiltona H taki, że $c(H) \leq c(W)$. $\square$

Ogólny problem komiwojażera

Tw. Jeśli $P \neq NP$ i $\rho \geq 1$, to nie istnieje wielomianowy algorytm aproksymacyjny o ograniczeniu względnym ρ dla ogólnego problemu komiwojażera.

D-d. Załóżmy, że dla pewnego $\rho \geq 1$ istnieje wielomianowy algorytm aproksymacyjny A o ograniczeniu względnym ρ. (Można przyjąć, że ρ – całkowite.)
Pokażemy, jak wykorzystać A do rozwiązywania problemu `HAM-CYCLE`. Niech $G = (V, E)$ – dany graf. Tworzymy $G' = (V, V \times V)$ i funkcję kosztu:

$$c(u,v) = \begin{cases} 1, & \text{jeśli } (u,v) \in E \\ \rho \cdot |V| + 1, & \text{jeśli } (u,v) \notin E \end{cases}$$

Przekształcenie G w (G', c) – w czasie wielomianowym.

...

Ogólny problem komiwojażera

Jeśli w G jest cykl Hamiltona, to w (G', c) optymalna marszruta ma koszt $|V|$. Algorytm A dla (G', c) zwróci wtedy marszrutę o koszcie $\leq \rho \cdot |V|$.
Jeśli w G nie ma cyklu Hamiltona, to w (G', c) optymalna marszruta ma koszt $\geq (\rho \cdot |V| + 1) + (|V| - 1) > \rho \cdot |V|$ (*musi zawierać jakąś krawędź spoza* E). Algorytm A dla (G', c) zwróci wtedy marszrutę o koszcie $> \rho \cdot |V|$.
Zatem koszt marszruty zwróconej przez A rozstrzyga (w czasie wielomianowym) czy w G jest cykl Hamiltona.
$\Box$.

Problem pokrycia zbioru

Dane *problemu pokrycia zbioru* – para $(X, \mathcal{F})$, gdzie X – zbiór skończony, $\mathcal{F}$ – rodzina podzbiorów X taka, że każdy $x \in X$ należy do pewnego zbioru w $\mathcal{F}$. Mówimy, że $S \in \mathcal{F}$ *pokrywa* swoje elementy. Należy znaleźć najmniej liczną podrodzinę $\mathcal{C} \subseteq \mathcal{F}$, która *pokrywa* cały podzbiór (t.j. $X = \bigcup_{S \in \mathcal{C}} S$).
Problem pokrycia zbioru jest `NP-trudny`, bo jest *uogólnieniem* problemu pokrycia wierzchołkowego. (X *– zbiór krawędzi,* $\mathcal{F}$ *– zbiory krawędzi pokryte przez wierzchołki.*)

Greedy-Set-Cover

Strategia zachłanna: *W każdym kroku wybieramy zbiór z $\mathcal{F}$, który pokrywa najwięcej z jeszcze nie pokrytych wierzchołków.*

`Greedy-Set-Cover(`X`,`$\mathcal{F}$`)`

$U \leftarrow X$ $\triangleright$ U *– nie pokryte wierzchołki*

$\mathcal{C} \leftarrow \emptyset$

`while` $U \neq \emptyset$ `do`

    wybierz $S \in \mathcal{F}$, *maksymalizujący* $|S \cap U|$

    $U \leftarrow U \setminus S$

    $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$

`return` $\mathcal{C}$

Czas: wielomianowy

Greedy-Set-Cover – analiza

Oznaczenie: $H(d) = H_d$ (t.j. d-ta liczba harmoniczna: $H_d = \sum_{i=1}^{d} 1/i$). Przypomnienie: $H(n) \leq \ln(n) + 1$.

Tw. Ograniczenie względne algorytmu `Greedy-Set-Cover` wynosi:

$$H(\max\{|S| : S \in \mathcal{F}\}) \leq (\ln |X| + 1)$$

D-d. Niech S_i – i-ty zbiór wybrany przez `Greedy-Set-Cover`. Dodanie S_i do $\mathcal{C}$ – koszt jednostkowy, rozdzielany równo między elementy pokryte po raz pierwszy. Każdy element $x \in X$ dostaje koszt c_x (tylko raz). Jeśli x jest pierwszy raz pokryty przez S_i, to:

$$c_x = \frac{1}{|S_i \setminus (S_1 \cup \ldots \cup S_{i-1})|}$$

...

Greedy-Set-Cover – analiza

Algorytm znajduje pokrycie o całkowitym koszcie $|\mathcal{C}|$, rozdzielonym między elementy zbioru X.
Optymalne pokrycie $\mathcal{C}^*$ też pokrywa X, więc:

$$|\mathcal{C}| = \sum_{x \in X} c_x \leq \sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x$$

Fakt: Dla dowolnego $S \in \mathcal{F}$ mamy:

$$\sum_{x \in S} c_x \leq H(|S|)$$

Z *Faktu* i wcześniejszej nierówności wynika teza Tw.:

$$|\mathcal{C}| \leq \sum_{S \in \mathcal{C}^*} H(|S|) \leq |C^*| \cdot H(\max\{|S| : S \in \mathcal{F}\})$$

Greedy-Set-Cover – analiza

D-d Faktu: Dla $S \in \mathcal{F}$ niech $u_0 = |S|$ oraz dla $i = 1, \ldots, |\mathcal{C}|$:

$$u_i = |S \setminus (S_1 \cup \ldots \cup S_i)|$$

Niech k najmniejszy indeks taki, że $u_k = 0$. Zatem S – pokryty przez S_1,...,S_k. Wówczas dla $i = 1, \ldots, k$ zachodzi $u_{i-1} \geq u_i$ (*oczywiste*), a S_i pokrywa po raz pierwszy $u_{i-1} - u_i$ elementów z S. Zatem:

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{|S_i \setminus (S_1 \cup \ldots \cup S_{i-1})|}$$

Zauważmy, że (z zachłannego wyboru S_i):

$$|S_i \setminus (S_1 \cup \ldots \cup S_{i-1})| \geq |S \setminus (S_1 \cup \ldots \cup S_{i-1})| = u_{i-1}$$

...

Greedy-Set-Cover – analiza

Stąd

$$\sum_{x \in S} c_x \le \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}}$$

Dla liczb naturalnych $a < b$, mamy:

$$H(b) - H(a) = \sum_{i=a+1}^{b} 1/i \ge (b-a)\frac{1}{b}$$

To też zachodzi dla $a = b > 0$, bo $0 \ge 0 \cdot \frac{1}{b}$. Stąd:

$$\sum_{x \in S} c_x \le \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i)) = H(u_0) - H(u_k) = H(|S|) - H(0)$$

co kończy dowód *Faktu*, bo $H(0) = 0$. $\square$

Problem sumy zbioru

Wersja decyzyjna `SUBSET-SUM`: *Czy dla danych (S, t), gdzie S skończony podzbiór liczb naturalnych, istnieje podzbiór S o sumie $= t$?*
(Problem `NP-zupełny`.)
Wersja optymalizacyjna: *Dla danych (S, t) znaleźć podzbiór S o jak największej sumie ale nie przekraczającej t.*
Zakładamy, że $|S| = n$ i $S = \{x_1, \ldots, x_n\}$.

wykładniczy `Exact-Subset-Sum`

```
Exact-Subset-Sum(S, t)
n ← |S|
L_0 ← ⟨0⟩
for i ← 1 to n do
    L_i ← Merge-Lists(L_{i−1}, L_{i−1} + x_i)
    usuń z L_i elementy większe od t
return największy element L_i
```

$L_{i-1} + x_i$ – *kopia* L_{i-1}*, gdzie do każdego elementu dodano* x_i*.*

`Merge-Lists`(L, L') *scala dwie posortowane listy w czasie* $\Theta(|L| + |L'|)$ *w posortowaną listę długości* $|L| + |L'|$ *i usuwa wszystkie powtórzenia.*

Maksymalny możliwy rozmiar listy L_i: 2^n.

Czas: wykładniczy, chociaż w szczególnych przypadkach wielomianowy (*gdy* t *lub wszystkie elementy* S *– wielomianowe ze wzgl. na* $|S|$).

Schemat aproksymacji

Podamy w pełni wielomianowy schemat aproksymacji.
Skrócenie listy L przez δ, $0 < \delta < 1$ – usuwanie elementów z L, tak aby w otrzymanej liście L' dla każdego usuniętego y istniał $z \leq y$ taki, że $\frac{y-z}{y} \leq \delta$ (co jest równoważne $(1-\delta)y \leq z \leq y$).

`Trim(`L`,`δ`)` $\triangleright$ L – *posortowana*

$m \leftarrow |L|$
$L' \leftarrow \langle y_1 \rangle$
last$\leftarrow y_1$
for $i \leftarrow 2$ **to** m **do**
    if *last*$< (1-\delta)y_i$ **then**
        dołącz y_i *na koniec* L'
        last$\leftarrow y_i$
return L' $\triangleright$ L' – *skrócenie* L

Schemat aproksymacji

`Approx-Subset-Sum(`S`,`t`,`ε`)`

1 $n \leftarrow |S|$

2 $L_0 \leftarrow \langle 0 \rangle$

3 **for** $i \leftarrow 1$ **to** n **do**

4 $\quad L_i \leftarrow$`Merge-Lists(`L_{i-1}`,`$L_{i-1} + x_i$`)`

5 $\quad L_i \leftarrow$`Trim(`L_i`,`ε/n`)`

6 $\quad$ *usuń z L_i elementy większe niż t*

7 *niech z – największy na L_i*

8 **return** z.

Tw. `Approx-Subset-Sum` jest w pełni wielomianowym schematem aproksymacji dla problemu sumy zbioru.

Niech P_i zbiór wszystkich możliwych sum podzbiorów zbioru $\{x_1, \ldots, x_i\}$. Zachodzi $L_i \subseteq P_i$. Zatem z zwracane w wierszu 8 jest sumą pewnego podzbioru S.

...

Schemat aproksymacji

Pozostaje pokazać, że $\frac{|C-C^*|}{C^*} \leq \varepsilon$ (*co dla problemu maksymalizacji jest równoważne:* $C^*(1-\varepsilon) \leq C$) oraz że czas jest wielomianowy ze wzgl. na rozmiar danych i $1/\varepsilon$.

Oszacowanie błędu: Skracając L_i wprowadzamy między pozostające wartości a wartości przed skróceniem błąd względny ε/n. Zatem przez indukcję można pokazać, że dla każdego $y \in P_i$ takiego, że $y \leq t$, na L_i istnieje z takie, że: $(1-\varepsilon/n)^i y \leq z \leq y$.

Jeśli $y^* \in P_n$ oznacza optymalne rozwiązanie, to istnieje $z \in L_n$ takie, że: $(1-\varepsilon/n)^n y^* \leq z \leq y^*$.

Ponieważ pochodna funkcji $f(x) = (1-\varepsilon/x)^x$ jest > 0, $(1-\varepsilon/n)^n$ rośnie wraz z n, czyli dla $n > 1$

$(1-\varepsilon) < (1-\varepsilon/n)^n$ skąd $(1-\varepsilon)y^* \leq z$.

...

Schemat aproksymacji

Oszacowanie czasu: Pokażemy, że $|L_i|$ jest wielomianowe ze wzgl. na rozmiar danych i $1/\varepsilon$. Po skróceniu kolejne elementy z i z' z L_i spełniają $z'/z > 1/(1-\varepsilon/n)$. Ponieważ największy na L_i jest $\leq t$, więc $|L_i|$ jest co najwyżej:
$\log_{1/(1-\varepsilon/n)} t = \frac{\ln t}{-\ln(1-\varepsilon/n)}$
Korzystamy z nierówności $\frac{x}{x+1} \leq \ln(x+1)$, dla $x > -1$.
(*Można ją wyprowadzić korzystając z nierówności* $z+1 \leq e^z$ *– ćw.*)
Podstawmy $x = \frac{\varepsilon}{n-\varepsilon}$. (*Wtedy* $x > 0 > -1$ *dla rozsądnych* n *i* ε.)
Z jednej strony:

$$\frac{x}{x+1} = \frac{\frac{\varepsilon}{n-\varepsilon}}{\frac{\varepsilon}{n-\varepsilon}+1} = \frac{\frac{\varepsilon}{n-\varepsilon}}{\frac{\varepsilon+(n-\varepsilon)}{n-\varepsilon}} = \frac{\frac{\varepsilon}{n-\varepsilon}}{\frac{n}{n-\varepsilon}} = \frac{\varepsilon}{n}$$

...

Schemat aproksymacji

Z drugiej strony:

$$\ln(x+1) = \ln\left(\frac{\varepsilon}{n-\varepsilon}+1\right) = \ln\left(\frac{n}{n-\varepsilon}\right) =$$

$$-\ln\left(\frac{n-\varepsilon}{n}\right) = -\ln\left(1-\varepsilon/n\right)$$

Czyli $\frac{\varepsilon}{n} \leq -\ln\left(1-\varepsilon/n\right)$ lub równoważnie $\frac{1}{-\ln(1-\varepsilon/n)} \leq \frac{n}{\varepsilon}$.
Stąd $|L_i| \leq \frac{\ln t}{-\ln(1-\varepsilon/n)} \leq \frac{n \ln t}{\varepsilon}$, co jest wielomianem ze wzgl. na n jak i $1/\varepsilon$ (oraz $\lg t$ – liczbę bitów do reprezentacji t).
Czas działania `Approx-Subset-Sum` jest wielomianem ze wzgl. na największą długość $|L_i|$. $\square$