Wykład 3: Klasy P, NP i Algorytmy heurystyczne

Nguyen Hung Son

7 marca 2006

Streszczenie

Klasa złożoności . . .
Klasa złożoności NP
Relacja między klasami . . .
Wielomianowa . . .
Definicje NP-trudności
Twierdzenie Cook'a
Więcej problemów NP- . . .
Algorytmy aproksymacji
Algorytmy . . .
Generator liczb losowych
Symulowane wychładzanie
Algorytm Metropolisa
Model matematyczny
Home Page
Title Page
Page 2 of 39
Go Back
Full Screen
Close
Quit

1. Klasa złożoności PTIME, PSPACE

- Niech $f(n)$ będzie rosnącą funkcją względem n, przez $\mathbf{TIME}(f(n))$ oznaczamy klasę takich problemów decyzyjnych π_L, dla których istnieją stałe $a, b \in \Re$ i maszyna Turinga M takie, że że dla każdego słowa w, maszyna M sprawdzi czy $w \in L$ w czasie krótszym niż $af(|w|) + b$.
- Zwykle interesują nas klasy $\mathbf{TIME}(n^k)$, $\mathbf{TIME}(n \log n)$, ...
- interesuje nas również klasa:

$$\mathbb{P} = \mathbf{PTIME} = \bigcup_k TIME(n^k)$$

  problemów rozwiązalnych w czasie wielomianowym.
- można analogicznie definiować klasę $\mathbf{PSCPACE}$

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 3 of 39
Go Back
Full Screen
Close
Quit

1.1. Przykłady

- Czy dwie liczby m, n są względnie pierwsze
- Czy m jest podzielny przez n?
- Czy dany graf jest 2-kolorowany?
- Czy dla trzech macierzy A, B, C zachodzi równość $AB = C$?
- ...

2. Klasa złożoności NP

- Język L należy do klasy $\mathbb{NP}$ wtedy i tylko wtedy, gdy istnieje maszyna Turinga $\mathbf{M}$ taka, że
  - Dla każdego słowa w istnieje certyfikat $C(w)$ w postaci ciągu bitów.
  - $\mathbf{M}$ rozpoznaje język

    $$L' = \{(w, C(w)) : w \in L\}$$

    zakładamy dodatkowo, że $(w', X) \notin L'$ dla każdego słowa $w' \notin L$.
  - $\mathbf{M}$ działa w czasie wielomianowym
- Innym słowem: jeśli mamy słowo $w \in L$ wraz z certyfikatem jego przynależności, to możemy szybko sprawdzić czy certyfikat jest prawidłowy.
- może być wiele certyfikatów dla słów z L, lecz nie ma poprawnego certyfikatu dla $w \notin L$.
- szukanie takich certyfikatów dla danego słowa w jest znacznie trudniejsze.

2.1. Przykłady

- Problem k-kolorowalności:

  DANE: Graf $G = (V, E)$, stała k;

  PYTANIE: Czy można kolowować V k kolorami w taki sposób, że każda krawędź miała różno-kolorowe końce?

- Problem komiwojażera: DANE: Lista miast i odległości między nimi, stała K;

  PYTANIE: Czy istnieje droga o długości $< K$ obiegająca wszytkie miasta?

Czasem warto rozpatrywać "problem odwrotny", np.

- Odwrotny problem komiwojażera:

  DANE: Lista miast i odległości między nimi, stała K;

  PYTANIE: Sprawdź, że nie istnieje droga o długości $< K$ obiegająca wszytkie miasta?

- Jeśli odwrotny problem należy do $\mathbb{NP}$, to mówimy, że dany problem należy do $co - \mathbb{NP}$;

3. Relacja między klasami złożoności

$$\mathbb{P} \subset \mathbb{NP} \subset \mathbf{PSPACE}$$

$$\mathbb{P} \subset co - \mathbb{NP} \subset \mathbf{PSPACE}$$

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 7 of 39
Go Back
Full Screen
Close
Quit

4. Wielomianowa transformacja problemów

- Mówimy, że język L_1 jest wielomianowo transformowalna do L_2 wtw, gdy istnieje funkcja

$$f : \Sigma_1^* \longrightarrow \Sigma_2^*$$

taka, że:

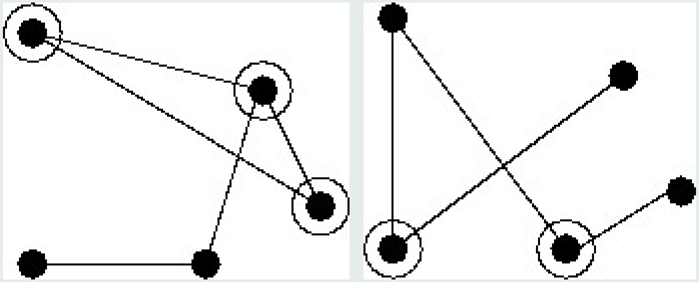
1. dla każdego słowa $w \in \Sigma_1^*$ zachodzi $w \in L_1 \Leftrightarrow f(w) \in L_2$
2. f działa w czasie wielomianowym.

- Zauważmy, że jeśli L_1 jest wielomianowo transformowalna do L_2, to

$$L_2 \in \mathbb{P} \Rightarrow L_1 \in \mathbb{P}$$

4.1. Przykład

Problem KLIKI jest redukowalny do problemu pokrycia wierzchołkami:



5. Definicje NP-trudności

Definicja *Dany język L nazywamy $\mathbf{NP}$-trudnym, jeśli każdy język z klasy $\mathbf{NP}$ jest wielomianowo transformowalny do L.*

Uwaga: jeśli uda się pokazać, że jakiś problem $\mathbf{NP}$-trudny należy do klasy $\mathbf{P}$, wówczas mamy $\mathbf{P} = \mathbf{NP}$.

Definicja *Jeśli dany język z klasy $\mathbf{NP}$ jest $\mathbf{NP}$-trudny, to nazywamy go $\mathbf{NP}$-zupełnym.*

Uwaga:

- jeśli uda się pokazać, że jakiś problem $\mathbf{NP}$-zupełny należy do klasy $\mathbf{P}$, wówczas mamy $\mathbf{P} = \mathbf{NP}$.
- Aby pokazać, że dany język jest $\mathbf{NP}$-zupełny, wystarczy transformować do niego wielomianowo pewien znany problem $\mathbf{NP}$-zupełny.

Klasa złożoności . . .
Klasa złożoności NP
Relacja między klasami . . .
Wielomianowa . . .
Definicje NP-trudności
Twierdzenie Cook'a
Więcej problemów NP- . . .
Algorytmy aproksymacji
Algorytmy . . .
Generator liczb losowych
Symulowane wychładzanie
Algorytm Metropolisa
Model matematyczny
Home Page
Title Page
Page 10 of 39
Go Back
Full Screen
Close
Quit

6. Twierdzenie Cook'a

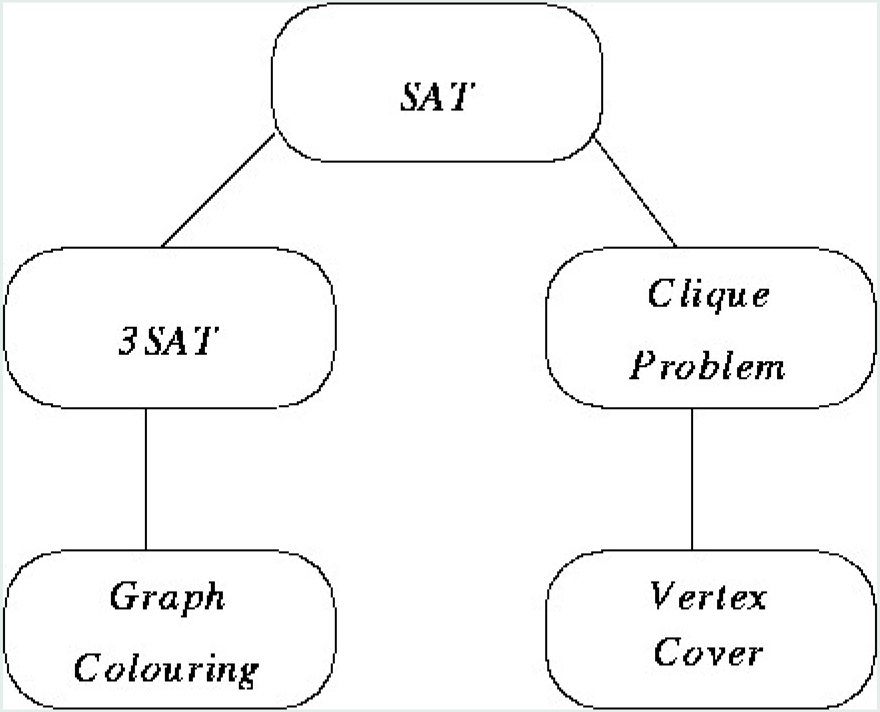
Problem SAT:
Dana: funkcja Boolowska $f(x_1, ..., x_n)$ zależna od n zmiennych.
Pytanie: "czy funkcja f jest spełnialna", tzn. czy istnieje takie wartościowanie zmiennych, dla którego f przyjmuje wartość 1.

Twierdzenie *Cook Problem SAT jest* $\mathbf{NP}$*-zupełny*

Szkic dowodu:

- SAT jest w klasie $\mathbf{NP}$ (łatwo);
- SAT jest $\mathbf{NP}$-trudny (trudniej i bardziej technicznie):
  - Dla dowolnego języka L z klasy $\mathbf{NP}$, istnieje maszyna Turinga $\mathbf{T}$ akceptuje L w sposób "niedeterministyczny".
  - Należy skonstruować formułę Boolowską $F(w, C(w), \mathbf{T})$ (w czasie wielomianowym) o tej własności, że $\mathbf{T}$ akceptuje słowo w wraz z certyfikatem $C(w)$, tzn. słowo $(w, C(w)$ wtedy, i tylko wtedy, gdy $F(w, C(w), \mathbf{T})$ jest spełnialna.
  - Cook skonstruował formułę $F(w, C(w), \mathbf{T})$ mającą $O(p^2(n))$ zmiennych i $O(p^3(n))$ klauzuli, gdzie $n = |w|$ i $p(n)$ jest złożonością maszyny $\mathbf{T}$ akceptującej język L.

7. Więcej problemów NP-zupełnych



Home Page
Title Page
Page 12 of 39
Go Back
Full Screen
Close
Quit

Twierdzenie *Problem 3-SAT jest* $\mathbf{NP}$*-zupełny*

dowód:

- Niech ϕ będzie formułą, której spełnialność chcemy sprawdzić;
- Jeśli w ϕ występuje klauzula zawierająca więcej niż 3 literały, to ją zastępujemy nowymi klauzulami według następującej reguły:

  $$(l_1 \vee l_2 \vee l3 \vee l4 \vee l5 \vee ...) \equiv (l_1 \vee l_2 \vee z_1) \wedge (\neg z_1 \vee l3 \vee l4 \vee l5...)$$

  gdzie z_1 to nowo wprowadzona zmienna;
- Klauzule nowej formuły mają co najwyżej 3 literały; Liczba tych nowych klauzuli jest wielomianowo większa od liczby klauzuli w ϕ.
- To oznacza, że SAT został wielomianowo zredukowany do 3-SAT.

Twierdzenie *Problem KLIKI jest* $\mathbf{NP}$*-zupełny*

dowód: Transformujemy SAT do problemu KLIKI

- Zakładamy, że mamy formułę:

  $$\psi = (l_{1,1} \vee l_{1,2} \vee \ldots l_{1,k_1}) \wedge (l_{2,1} \vee l_{2,2} \vee \ldots l_{2,k_2}) \wedge \ldots \wedge (l_{s,1} \vee l_{s,2} \vee \ldots l_{s,k_s})$$

  gdzie $l_{i,j}$ są literałami.

- Skonstruujemy graf $G_\psi = (V, E)$:

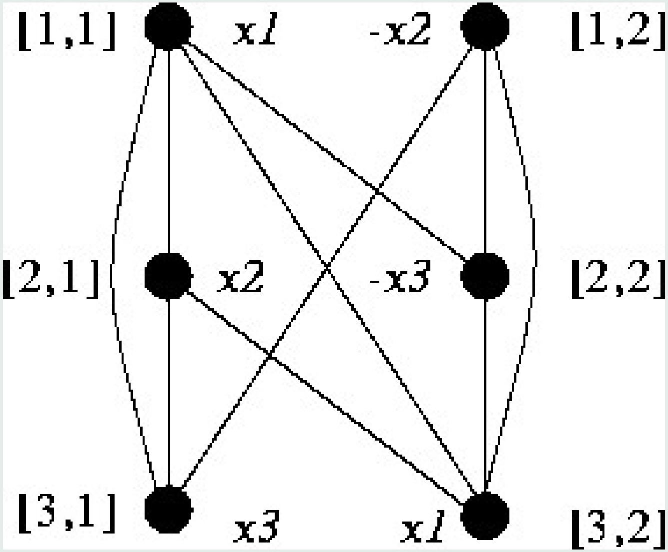
  $$V = \{[i,j] : 1 \le i \le s, 1 \le j \le k_i\}$$

  $$E = \{([i,j],[m,n]) : i \ne m, l_{i,j} \ne \neg l_{m,n}\}$$

- Łatwo można pokazać, że ψ jest spełnialny wtw, gdy G_ψ posiada klikę o rozmiarze s.

Przykład

$$\psi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_1 \vee x_3)$$



Twierdzenie *Problem Kolorowanie Grafu jest* **NP***-zupełny*

dowód: Sprowadzamy z problemu 3-SAT:

- Dana jest formuła $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ z k klauzulami i n zmiennymi: $x_1, \ldots, x_n$;
- Skonstuujemy graf $G_\psi = (V, E)$ z $3n + k$ wierzchołkami:

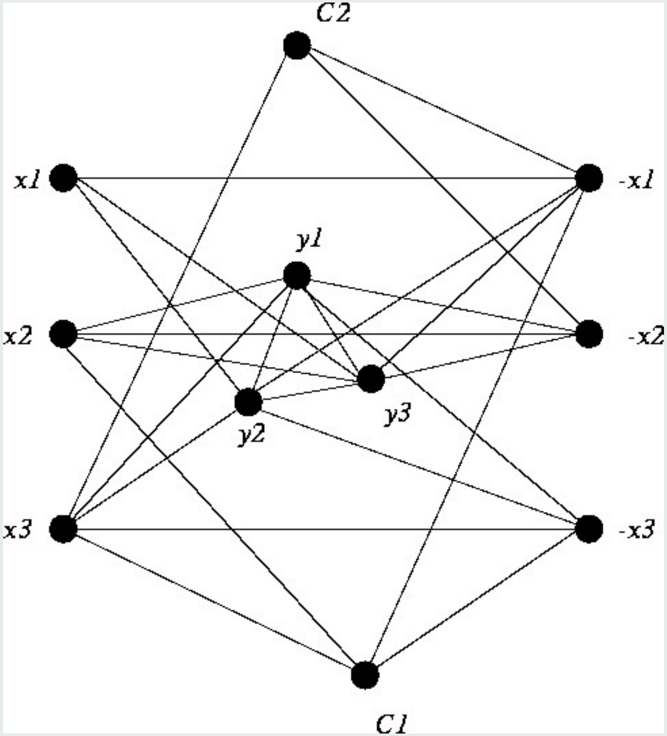
  $$V = \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\} \cup \{y_1, \ldots, y_n\} \cup \{C_1, \ldots, C_k\}$$

  które są połączone według nast. zasad:

  - Każda para $(x_i, \neg x_i)$ jest krawędzią;
  - Każda para (y_i, y_j) jest krawędzią;
  - Każda para (y_i, x_j) i $(y_i, \neg x_j)$ jest krawędzią, o ile $i \neq j$;
  - Każda para (C_i, x_j) lub $(C_i, \neg x_j)$ jest krawędzią, o ile x_j lub $\neg x_j$ nie występuje w C_i;
- ψ jest spełnialny wtw, gdy G_ψ jest $n + 1$ kolorowalny

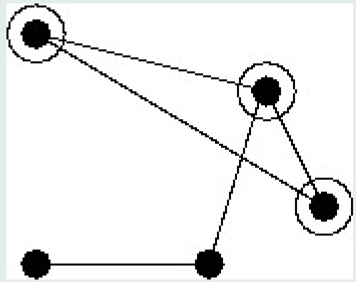
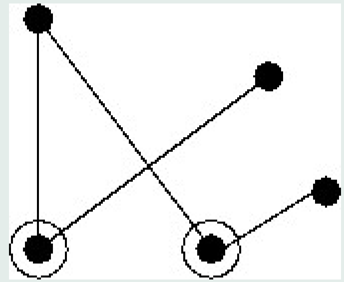
Przykład

$$\psi = (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee \neg x_3)$$



Twierdzenie *Problem pokrycia wierzchołkowego jest* $\mathbf{NP}$*-zupełny*

dowód: Sprowadzamy problem KLIKI do problemu pokrycia wierzchołkowego:

- Dany jest graf $G = (V, E)$ i stała k dla problemu kliki
- Skonstruujemy graf $G' = (V', E')$ w nast. sposób:

$$V' = V$$

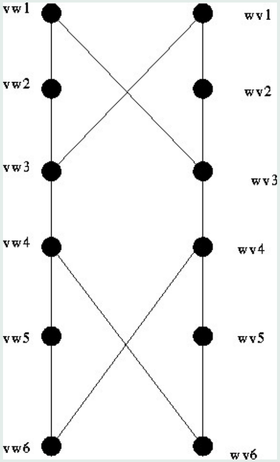
$$E' = \{(v_1, v_2) : (v_1, v_2) \notin E\}$$

- Trzeba pokazać, że G posiada klikę o rozmiarze k wtw, gdy G' ma pokrycie wierzchołkowe o rozmiarze $|V| - k$.

Twierdzenie *Problem Cyklu Hamiltona jest* $\mathbf{NP}$*-zupełny*

dowód: Sprowadzamy do niego problem pokrycia wierzchołkowego:

- Dany jest graf $G = (V, E)$ dla problemu pokrycia, skonstruujemy skierowany graf: $G' = (V', E')$ według nast. zasad:
  - Dla każdej krawędzi $e = (v, w)$ grafu G skonstruujemy podgraf:



  - wierzchołki o numerach 1 i 6 są połączone z innymi wierzchołkami;
  - w G' mamy k dodatkowych wierzchołków: $a_1, ..., a_k$
  - połączymy również każde a_i z $vw1$ i $vw6$
- Trzeba pokazać, że G posiada pokrycie o rozmiarze k wtw, gdy G' ma cykl Hamiltona.

7.1. Inne problemy NP-zupełne

- Problem podziału:
- Problem plecakowy:
- izomorfizm dwóch grafów:
- problem komiwojażera:

8. Algorytmy aproksymacji

Przykład: Problem szeregowania zadań:

Dane: n zadań wraz z czasem ich realizacji,
m identycznych maszyn.

Problem: Uszeregować te zadania w taki sposób, aby minimalizować okres realizacji wszystkich zadań.

Przykład rozwiązania: Algorytm "List scheduling":

Idea: Wybierz kolejne zadanie z listy zadań i uszereguj go do pierwszej wolnej maszyny.

Czy ten algorytm jest dobry?

PROBLEM: Co zrobić z problemami, które są NP-trudne?

- Zakładając, że instancje są losowo generowane, oszacuj óczekiwany wynik”. Np. ścieżka Hamiltona w grafach.
  Problem: jak ustalić właściwy rozkład prawdopodobieństwa?
- Wykonaj algorytm nad-wielomianowy z nadzieją, że czasem skończy on w rozsądnym czasie.
- Używaj jakiegoś *algorytmu heurystycznego* + pokaż, że działa dostatecznie dobrze (pod naszym nadzorem)

DEFINICJA PROBLEMU OPTYMALIZACJI:

- Dla każdej instancji I definiujemy:
  - $S(I)$ - zbiór możliwych rozwiązań;
  - $f : S(I) \rightarrow R$ - jakość rozwiązań;
- Problem maksymalizacji/minimalizacji $=$ szukanie w $S(I)$ rozwiązania maksymalizującego/mimimalizującego funkcję f;
- $OPT(I) = \arg\max_{\sigma \in S(I)} f(\sigma)$ – najlepsze rozwiązanie.
- Np. problem plecaku.

Założenie techniczne:

- dane wejściowe i wartości funkcji f są liczbami wymiernymi.
- $f(\sigma)$ jest wielomianowa wzg. $|\sigma|$.
- używamy kodu binarnego do obliczenia rozmiaru problemu.

  - dane wejściowe i wartości funkcji f są liczbami wymiernymi.
  - $f(\sigma)$ jest wielomianowa wzg. $|\sigma|$.
  - używamy kodu binarnego do obliczenia rozmiaru problemu.

8.1. NPtrudny problem optymalizacji

- problem optymalizacji Π jest NP-trudny, jeśli można "redukować" jakiś inny, NP-trudny problem decyzyjnego do niego.
- czyli istnieje "wielomianowa transformacja Turinga".

Algorytm aproksymacji:

- każdy algorytm, który zwraca "rozsądne rozwiązanie";
- chcemy mieć dobry algorytm. Jak mierzyć?

8.2. Absolutny wsp. aproksymacji

Def.: Algorytm $\mathbb{A}$ ma absolutny wsp. aproksymacji k, jeśli dla każdej instancji I

$$|\mathbb{A}(I) - OPT(I)| \leq k$$

8.3. Względny wsp. aproksymacji

Def.: Algorytm $\mathbb{A}$ ma względny wsp. aproksymacji α, jeśli dla każdej instancji I mamy:

$$\mathbb{A}(I) \leq \alpha \times OPT(I) \qquad \text{dla problemu minimalizacji}$$

$$\mathbb{A}(I) \geq \alpha \times OPT(I) \qquad \text{dla problemu maksimalizacji}$$

9. Algorytmy zrandomizowane

Są to algorytmy, których działania jest uzależnione od pewnych czynników losowych. Istnieją 2 typy randomizacji:

- **Monte Carlo:** typ algorytmów dających wynik optymalny z prawdopodobieństwem bliskim 1 (po dostatecznie długim czasie działania);
- **Las Vegas:** typ algorytmów dających zawsze wynik optymalny, a randomizacja służy jedynie przyspieszeniu algorytmu;

9.1. Przykład metody Las Vegas

1. zrandomizowany QuickSort:
   - Wybierzmy losowy element x ze zbioru S;
   - Podzielmy S na podzbiory S_1 zawierający elementy mniejsze od x, i S_2 zawierający elementy większe od x;
   - rekurencyjnie stosujemy powyższe kroki dla S_1 i S_2

   uwaga: złożoność średnia jest taka sama, jak w przypadku QuickSort, ale losowość elementu dzielącego zapobiega pojawieniu złośliwych danych.
2. drzewo BST:

9.2. Przykład metody Monte Carlo

1. Sprawdzić czy $\mathbf{AB} = \mathbf{C}$, gdzie $\mathbf{A}, \mathbf{B}, \mathbf{C}$ są macierzami rzędu $n \times n$:
   - Dokładny algorytm mnożenia macierzy działa w czasie n^3;
   - zrandomizowany algoprytm: "Losowo wybierz kilka wektorów x i sprawdź, czy $\mathbf{A}(\mathbf{B}x) = \mathbf{C}x$";
   - jeśli nie zachodzi równość, to bardzo szybko można znaleźć kontr-przykłady!
2. Całkowanie: Liczymy średnią wartość funkcji w losowych punktach;
3. Problem komiwojażera: Losujemy kolejną permutację;

10. Generator liczb losowych

- Źródła prawdziwej losowości:
  - zegar systemowy;
  - użytkownik komputera (czas naciśnięcia klawisza, ruch myszki);
  - przyrządy pomiarowe (szumy, licznik Geigera próbki promieniotwórczej)
- W praktyce te źródła służą do inicjalizacji ciągów liczb pseudolosowych w generatorach programowych. Np.
  - Generator Marsaglii (1991):

    $$x_n = (x_{n-s} + x_{n-r} + c) \mathrm{mod} M$$

    gdzie $c = 1$ jeśli poprzednia suma przekroczyła M, $c = 0$ w p.p.;
  - np. $x_n = (x_{n-2} + x_{n-21} + c)$ mod 6, okres 10^{16}
  - np. $x_n = (x_{n-22} - x_{n-43} - c)$ mod $2^{32} - 5$, okres 10^{414}

10.1. Liczby losowe o zadanym rozkładzie

- **Metoda ogólna:** Odwracanie dystrybuanty:

  Mamy generować zm. losową X o gęstości $f(x)$. Niech $F(x)$ będzie dystrybuantą tego rozkładu. Wówczas $X = F^{-1}(u)$ będzie zmienną o żądanym rozkładzie, gdzie u jest zm. o rozkładzie jednostajnym na przedziale $[0, 1]$;

- Metoda eliminacji:

  Znamy gęstości $f(x)$ na dziedzinie D, jednak nie znamy F^{-1}. Niech M będzie ograniczeniem górnym rozkładu $f(x)$;

```
repeat
    u1 = random(D);
    u2 = random(0,M);
until u2 < f(u1);
```

11. Symulowane wychładzanie

Simulated Annealing: "Symulowane wychładzanie"

- Analogia do fizyki
- Monte Carlo Annealing
- Probabilistyczna metoda wspinaczkowa
- Statystyczne schłodzenie
- Stochastyczna relaksacja

Cechy charakterystyczne:

- Znajduje rozsądne rozwiązanie niezależnie od punktu startowego
- Kontrolowany czas obliczeń
- Łatwe w użyciu: zaczynamy od rozwiązania nie koniecznie optymalnego, a algorytm wykonuje resztę roboty !!!

12. Algorytm Metropolisa

- **Wychładzanie:** proces termiczny do otrzymania materiału o niskim stanie energii.
- **2-etapowy proces:**
  1. Zwiększamy temperaturę do maksymalnej (temp. topnienia)
  2. Bardzo powoli obniżamy temperaturę dopóki cząstki materialne rozmieszczają się w stanie krystalicznym.
- **Ciecz:** – chaotyczny ruch cząstek, wysoki stan energii wewnętrznej.
- **Kryształ:** – cząstki są układane w uporządkowanej strukturze, niski stan energetyczny
- Stan krystaliczny można otrzymać jeśli maks. temp. jest dostatecznie wysoka i schłodzenie jest dostatecznie wolne. W p.p. materiał będzie zamrożony do meta-stabilnego stanu (lokalne minimum)
- Szybkie schłodzenie $\longrightarrow$ hartowanie $\longrightarrow$ stan meta-stabilny

12.1. Pierwsze pomysły

(1953, Metropolous, Rosenbluth, Teller and Teller)
Problem optymalizacji:

- Zbiór możliwych konfiguracji (stanów, rozwiązań)

$$R = \{k_1, k_2, ..., k_N\}$$

- Każda konfiguracja k_i ma otoczenie $R_i \subset R$.
- Funkcja kosztu $\mathcal{C} : R \to \mathbb{R}^+$. Koszty mogą być interpretowane jako energie. Zamiast $\mathcal{C}(k_i)$ piszemy $\mathcal{C}_i$.
- Stan k_j jest akceptowany z k_i z prawdopodobieństwem

$$P_{accept}(j,i) = \begin{cases} 1 & \text{jeśli } \; C_j - C_i \leq 0, \\ e^{-\frac{C_j - C_i}{c}} & \text{wpp.} \end{cases}$$

gdzie c jest temperaturą układu.

Home Page
Title Page
◀◀ ▶▶
◀ ▶

Go Back
Full Screen
Close
Quit

12.2. Schemat Algorytmu

begin

  INITIALIZE(c_0,konfiguracja początkowa K_0);

  M := 0;

  repeat

    repeat

      LOSUJ nową konfigurację $K_j \in R_i$;

      $\triangle C_{ij} = C_j - C_i$;

      if $(\triangle C_{ij} \leq 0)$ **then** $K_i := K_j$;

      else if $\left(e^{\frac{-\triangle C_{ij}}{c_M}} > random(0,1)\right)$

        then $K_i := K_j$;

    until "Równowaga" jest prawie osiągalna

    $c_{M+1} := f(c_M)$;

    $M := M + 1$;

  until Kryterium STOPU zachodzi (system zamrożony)

end

13. Model matematyczny

1. Stany $R = \{1, 2, ..., N\}$
2. Macierz generacji: $\mathbf{G}(c_k) = [G_{ij}(c_k)]$

$$G_{ij}(c_k) = \frac{\chi_{R_i}(j)}{|R_i|} = \text{p-wo generowania } j \text{ z } i$$

3. Macierz akceptacji: $\mathbf{A}(c_k) = [A_{ij}(c_k)]$

$$A_{ij}(c_k) = \begin{cases} 1 & \text{jeśli } C_j - C_i \leq 0, \\ e^{-\frac{C_j - C_i}{c_k}} & \text{wpp.} \end{cases}$$

4. Stochastyczna macierz przejść: $\mathbf{P}(c_k) = [P_{ij}(c_k)]$

$$P_{ij}(c_k) = \begin{cases} G_{ij}(c_k) \cdot A_{ij}(c_k) & \text{jeśli } i \neq j, \\ 1 - \sum\limits_{l \neq i} P_{il}(c_k) & \text{wpp.} \end{cases}$$

13.1. Zbieżność lokalna SA

Twierdzenie 1: Łańcuch Markowa o macierzy przejść $[\mathbf{P}_{ij}(c_k)]$ jest stacjonarny, tzn.

$$\mathbf{q}_i = \lim_{k \leftarrow \infty} Pr(\{X(k) = i | X(0) = j\})$$

istnieje niezależnie od j, oraz

$$q_i(c) = \frac{1}{N_0(c)} \cdot e^{-\frac{C(i)}{c}}$$

gdzie $N_0(c) = \sum_{j \in R} e^{-\frac{C(i)}{c}} < \infty$,

13.2. Łańcuch Markowa

Kiedy istnieje stan stacjonarny?

Lemat (o istnieniu) *Jeśli łańcuch Markowa o macierzy przejść* $[\mathbf{P}_{ij}(c_k)]$ *jest <u>skończony</u>, <u>nieprzywiedlny</u>, <u>acykliczny</u> i <u>jednorodny</u> to istnieje jednoznaczny rozkład stacjonarny* $\mathbf{q} = (q_i)_{i \in R}$

Lemat *Jeśli* $\forall_{ij} q_i \cdot P_{ij} = q_j \cdot P_{ji}$ *to* $\mathbf{q} = (q_i)_{i \in R}$ *jest jednoznacznym rozkładem stacjonarnym dla ŁM.*

13.3. Dowód zbieżności lokalnej

1. Jednorodność, skończoność ...
2. Nieprzywiedlność:

$$\forall_{i,j\in R}\exists_{k\geq 1}\left(\mathbf{P}^k\right)_{ij} > 0$$

   wynika z wyboru otoczeń.

3. Jeśli łańcuch Markowa (ŁM) o macierzy przejść $\mathbf{P}_{ij}(c_k)$ jest nieprzywiedlny oraz $(\exists_{j\in R} P_{jj} > 0)$ to ŁM jest acykliczny tzn

$$\forall_{i\in R} NWD(\{n \in \mathbf{N} - \{0\} : (\mathbf{P}^n)_{ii} > 0\}) = 1$$

4. Wystarczy pokazać, że rozkład

$$q_i(c) = \frac{1}{N_0(c)} \cdot e^{-\frac{C(i)}{c}} \tag{1}$$

   gdzie $N_0(c) = \sum_{j\in R} e^{-\frac{C(i)}{c}} < \infty$, jest stacjonarny. (wynika z lematu 2)

13.4. Zbieżność globalna SA

Twierdzenie *Algorytm "Simulated Annealing" jest asymptotycznie zbieżny (tzn.*

$$\lim_{k\to\infty} Pr(\{X(k) \in R_{opt}\}) = 1$$

Dowód

Jeśli rozkład stacjonarny jest określony wzorem (1) to

$$\lim_{k\to\infty} q_i(c_k) = \lim_{k\leftarrow\infty} Pr(\{X(k) = i\}|c_k) = q_i^*$$

jest prawdopodobieństwem osiągania stanu i w ∞. Wówczas

$$q_i^* = \lim_{c\to 0^+} q_i(c_k) = \frac{1}{|R_{opt}|} \cdot \chi_{R_{opt}(i)}$$

Literatura

[1] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi, "Optimization by Simulated Annealing", Science, 220, No. 4598. 498-516, May 1983.

[2] N. Metropolis, A. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines", J. Chem. Phys. 21 1087, 1953.

[3] David E Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 1989

[4] Emile Aarts and Jan Korst, "Simulated Annealing and Boltzmann Machines", John Wiley & Sons, 1990