Wykład 2: Algorytmy heurystyczne

Nguyen Hung Son
son@mimuw.edu.pl

Streszczenie

1. 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.

1.1. 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?

1.2. Absolutny wsp. aproksymacji

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

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

1.3. Względny wsp. aproksymacji

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

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

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

2. 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;

2.1. 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:

2.2. 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;

3. 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. $x_n = (x_{n-2} + x_{n-21} + c) \bmod 6$, okres 10^{16}
  - np. $x_n = (x_{n-22} - x_{n-43} - c) \bmod 2^{32} - 5$, okres 10^{414}

3.1. 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);
```

4. 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 !!!

5. 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

5.1. 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 $\mathcal{C} : R \longrightarrow \mathbb{R}^+$. 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.

5.2. 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

6. 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: $\mathbf{P}(c_k) = [P_{ij}(c_k)]$

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

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

6.2. Łańcuch Markowa

Kiedy istnieje stan stacjonarny?

Lemat (o istnieniu) *Jeśli łańcuch Markowa o macierzy przejść* $[\mathbf{P}_{ij}(c_k)]$ *jest skończony, nieprzywiedlny, acykliczny i jednorodny 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.*

6.3. 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)

6.4. 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