WYKŁAD 8

Algorytmy zrandomizowane

PLAN WYKŁADU

- Generowanie liczb losowych
- Metody Monte Carlo i Las Vegas
  - przykłady zastosowań
- Przeszukiwanie losowe
  - metoda tabu
  - wychładzanie

ALGORYTMY ZRANDOMIZOWANE

Algorytmy, których działanie uzależnione jest od czynników losowych.

• Algorytmy typu **Monte Carlo**: dają (po pewnym czasie) wynik optymalny z prawdopodobieństwem bliskim 1

np. algorytm szukania pokrycia macierzy przez wielokrotny losowy wybór kolejnych kolumn (o ile pokrywają nowe wiersze macierzy)

• Algorytmy typu **Las Vegas**: dają zawsze wynik optymalny, a randomizacja służy jedynie poprawie szybkości działania.

np. zrandomizowany QuickSort

SKĄD WZIĄĆ LICZBY LOSOWE

Źródła „prawdziwej" losowości:
- zegar systemowy
- użytkownik (np. czas naciśnięcia klawisza, ruch myszki)
- przyrządy pomiarowe (np. szum wzmacniacza, licznik Geigera obok próbki promieniotwórczej)

W praktyce te źródła losowości służą do inicjowania ciągów liczb pseudolosowych w generatorach programowych.

Przykład:
generator Marsaglii (1991): $x_n = (x_{n-s} + x_{n-r} + c) \bmod M$,
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}

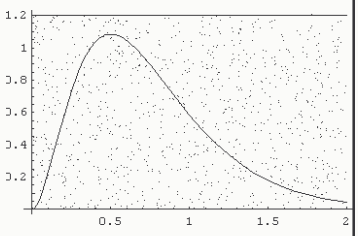
LICZBY LOSOWE O ZADANYM ROZKŁADZIE

- Metoda ogólna: **odwracanie dystrybuanty**.

Mamy wygenerować zmienną losową z rozkładu o znanej gęstości f(x).
Niech F(x) - dystrybuanta tego rozkładu. Wtedy:
wynik = $F^{-1}(u)$
jest szukaną zmienną losową, o ile *u* - wartość losowa z przedziału [0,1).

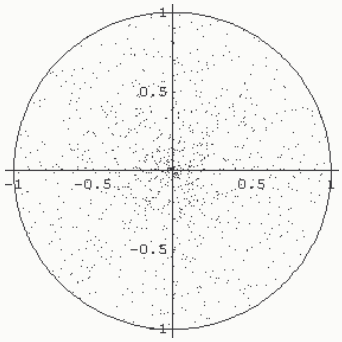
- **Metoda eliminacji**:

Znamy gęstość f(x) na pewnej dziedzinie D, jednak nie znamy F^{-1}. Niech M - ograniczenie górne f(x).



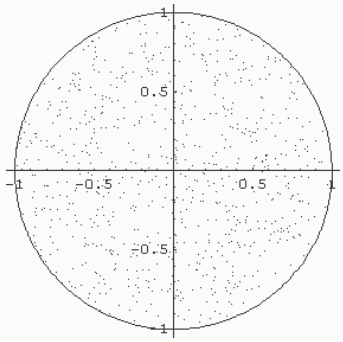

```
repeat
      u1=random(D)
      u2=random(0,M)
until u2<f(u1)
```

PRZYKŁAD - LOSOWANIE PUNKTÓW Z KOŁA



Metoda "biegunowa"

Losujemy współrzędne w układzie biegunowym (kąt i promień) z rozkładu jednostajnego.



Metoda eliminacji

Losujemy punkty z kwadratu i eliminujemy te, które leżą poza kołem.

LAS VEGAS - PRZYKŁADY

RandomQuickSort:

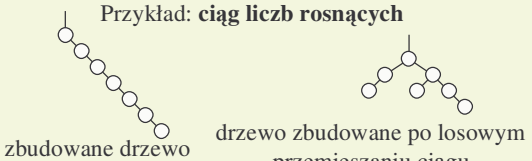
1. Mamy posortować zbiór S. Wybieramy z S losowy x.
2. Dzielimy S na zbiory S_1 - elementów mniejszych, niż x i S_2 - większych.
3. Rekurencyjnie sortujemy S_1 i S_2.

Średnia złożoność jest taka sama, jak w przypadku deterministycznym. Losowanie zapobiega zbyt częstemu pojawianiu się"złośliwych" danych.

Przykład pokrewny: budowa drzewa binarnego

Przed budową drzewa mieszamy losowo dane wejściowe.

Przykład: **ciąg liczb rosnących**



zbudowane drzewo

drzewo zbudowane po losowym przemieszaniu ciągu

MONTE CARLO

- Metody, które dają z pewnym prawdopodobieństwem dobry wynik
- Prawdopodobieństwo można zwiększać powtarzając obliczenia wielokrotnie

  (tej cechy nie mają alg. deterministyczne)
- Algorytmy numeryczne
  - całkowanie

    (liczymy średnią wartość funkcji w losowych punktach)
  - sprawdzanie, czy AB=C dla macierzy A,B,C

    (zamiast mnożyć A i B, co jest kosztowne, losujemy wektor *x* i badamy, czy A(Bx) = Cx, dla wielu różnych x)

PRZESZUKIWANIE LOSOWE

Najprostsza metoda: z przestrzeni stanów *S* losujemy wiele obiektów *x* i wybieramy ten, dla którego *f(x)* ma wartość największą.

Metoda bardziej złożona: po wylosowaniu np. 1000 obiektów zapamiętujemy 100 najlepszych i kolejnych losowań dokonujemy z "otoczenia" tych 100.

(Przykład - problem komiwojażera: losujemy 1000 dróg, wybieramy 100 najkrótszych, następnie losowo je modyfikujemy, otrzymując kolejne 1000 przykładów itd.)

Metoda hybrydowa: wybieramy 100 najlepszych z 1000 losowych, następnie dla każdego z nich uruchamiamy algorytm wspinaczki.

MONTE CARLO - PRZYKŁAD

Minimalne pokrycie kolumnowe macierzy zerojedynkowej A.

Algorytm:

- Losujemy permutację (kolejność) kolumn.
- Niech B będzie zbiorem wszystkich kolumn - załóżmy, że pokrywają one A.
- Usuwamy z B pierwszą kolumnę (według ustalonej kolejności).
- Sprawdzamy, czy wszystkie wiersze nadal są pokryte. Jeśli nie, przywracamy usuniętą ostatnio kolumnę.
- Powtarzamy operacje dla kolejnych kolumn.

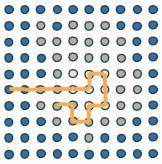
Algorytm nie musi dać globalnie optymalnego wyniku - błąd może być dowolnie duży.
Istnieje jednak taka permutacja kolumn, że algorytm znajdzie rozwiązanie optymalne - a więc, czekając odpowiednio długo, zawsze je otrzymamy.

1 1 0 0 0
1 0 1 0 0
1 0 0 1 0
1 0 0 0 1

PRZESZUKIWANIE TABU

Schemat działania: przeglądamy przestrzeń stanów (rozwiązań), przechodząc zawsze do tego sąsiada, dla którego wartość funkcji oceny jest największa (jak w algorytmie wspinaczki), o ile nie znajduje się on na liście punktów zabronionych. Na liście tej przechowujemy k ostatnio odwiedzonych punktów (np. k=1000).

Przykład: maksymalizacja funkcji dwuwymiarowej (na siatce o ustalonej dokładności).



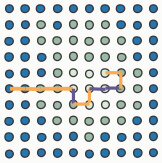
Wersja zrandomizowana:
Z prawdopodobieństwem 1/2 wybieramy najlepszego (dozwolonego) sąsiada, z prawd. 1/4 - drugiego z kolei, z prawd. 1/8 - trzeciego itd.

METODA WYCHŁADZANIA

(Simulated annealing)

*Schemat działania: startujemy z losowego punktu, wybieramy **losowo** nowy punkt z sąsiedztwa aktualnego, jeśli ma większą wartość - idziemy tam, jeśli mniejszą - z pewnym prawdopodobieństwem albo tam idziemy, albo testujemy następnego sąsiada.*

```
x0 = Random(X)
do
  x=Random(N(x0))
  if(f(x)>f(x0)) x0=x
  else
    if(Random()<P(x0,x))
      x0=x
while(time_out)
```



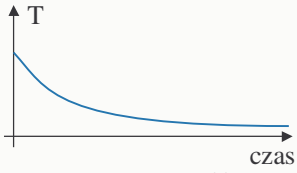
TEMPERATURA W ALG. WYCHŁADZANIA

Prawdopodobieństwo akceptacji gorszego stanu:

$$P(x_0, x) = e^{-\frac{|f(x_0) - f(x)|}{T}}$$

gdzie T - pewien parametr (“temperatura”)

Prawdopodobieństwo zależy od temperatury (im niższa, tym mniejsze), oraz od spadku wartości funkcji f.



Temperatura zwykle zmniejsza się w czasie.

- Zalety: odporność na maksima lokalne.
- Wady: wolniejsza zbieżność.