Algorytmy zrandomizowane

http://zajecia.jakubw.pl/nai

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)

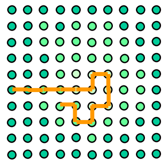
• Algorytmy typu **Las Vegas**: dają zawsze wynik optymalny, a randomizacja służy jedynie poprawie szybkości działania.

np. zrandomizowany QuickSort

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 oparta na sąsiedztwie.

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 "sąsiedztwa" 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.

Metody (często) oparte na sąsiedztwie.

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}

Generator z niektórych kompilatorów C/C++ (np. GCC):
$x_n = (\mathit{1103515245}\ x_{n-1} + \mathit{12345}) \bmod 2^{32}$
$\mathrm{rand}() = (x_n / 2^{16}) \bmod 2^{15}$

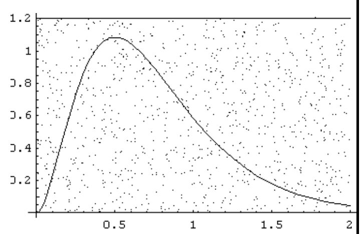
LICZBY LOSOWE O ZADANYM ROZKŁADZIE

- 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).

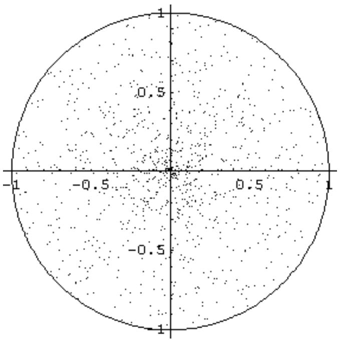
- **Metoda eliminacji**:

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

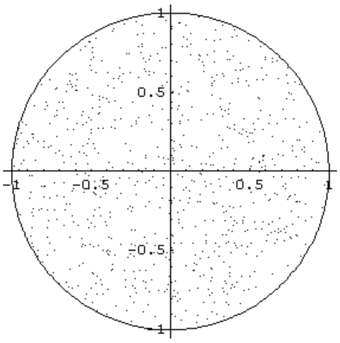



```
repeat
      u1=random(D)
      u2=random(0,M)
until u2<f(u1)
```

PRZYKŁAD - LOSOWANIE PUNKTÓW Z KOŁA



Metoda “biegunowa”



Metoda eliminacji

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

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

LAS VEGAS - PRZYKŁADY

RandomQuickSort:

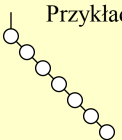
1. Mamy posortować zbiór S. Wybieramy z S losowy x.
2. Dzielimy S na zbiory S_1 - elementów mniejszych, niż x i S_2 - większych.
3. Rekurencyjnie sortujemy S_1 i S_2.

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

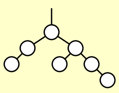
Przykład pokrewny: budowa drzewa binarnego

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)