Problemy optymalizacji dyskretnej
Algorytmy heurystyczne

`http://zajecia.jakubw.pl/nai`

ZADANIA OPTYMALIZACYJNE

Wiele problemów rozwiązywanych przez komputery ma postać **zadań optymalizacyjnych**:
„znaleźć wśród różnych możliwych rozwiązań takie, które najbardziej nam odpowiada”

Problemy:

- Jak ściśle zdefiniować „zadanie optymalizacyjne”?
- Jak określić stopień złożoności metody (algorytmu) poszukiwania rozwiązania?
- Które problemy zaliczyć do grupy „łatwych”, a które do grupy problemów „trudnych” (rozwiązywalnych tylko w przybliżeniu)?

ZADANIA OPTYMALIZACYJNE - DEFINICJA FORMALNA (przypomnienie)

Niech X - dowolny zbiór skończony (*przestrzeń stanów*)

Niech $f : X \rightarrow R$ - pewna rzeczywista funkcja na X (*funkcja celu*)

Zadanie optymalizacyjne polega na znalezieniu punktu x_0 ze zbioru X takiego, że:

$$f(x_0) = \max(f(x)), \qquad x \in X$$

lub

$$f(x_0) = \min(f(x)), \qquad x \in X$$

PRZYKŁADY (1)

Posortować *n* nazwisk alfabetycznie (rosnąco).

Przestrzeń stanów: wszystkie możliwe ustawienia *n* nazwisk.
Wielkość przestrzeni stanów: *n!* (**nie *n***).

Funkcja celu: np. liczba par sąsiadujących nazwisk ustawionych we właściwej kolejności (maksimum: *n-1*, co odpowiada właściwemu posortowaniu).

Każde dyskretne zadanie optymalizacyjne można rozwiązać przez przejrzenie wszystkich możliwości (wszystkich elementów przestrzeni stanów). Często jednak istnieją skuteczniejsze algorytmy (np. w przypadku sortowania).

PRZYKŁADY (2)

Znaleźć maksimum funkcji $f(x)=x^4 - 2x^2 + 3$ na przedziale $[0,10]$.

Przestrzeń stanów: wszystkie wartości x z przedziału $[0,10]$. Zakładając dokładność obliczeń do 6 cyfr znaczących, otrzymujemy 10^7 potencjalnych rozwiązań.

Funkcja celu: badana funkcja $f(x)$.

PRZYKŁADY (3) - PROBLEM KOMIWOJAŻERA

Dany jest graf G, którego krawędzie mają ustalone długości. Znaleźć najkrótszą drogę przechodzącą dokładnie raz przez wszystkie wierzchołki (o ile istnieje).

Przestrzeń stanów: wszystkie możliwe drogi przechodzące przez każdy wierzchołek dokładnie raz.

Wielkość przestrzeni stanów: co najwyżej $n!$, gdzie n - liczba wierzchołków.

Funkcja celu: łączna długość trasy.

PRZYKŁADY (4) - PROBLEM POKRYCIA MACIERZY

Dana jest macierz A={a_{ik}} o wartościach 0 lub 1. Znaleźć taki najmniejszy zbiór kolumn B, że w każdym *i*-tym wierszu macierzy A można wskazać wartość $a_{ik}=1$ taką, że kolumna *k* należy do B.

Przestrzeń stanów: wszystkie możliwe pokrycia kolumnowe macierzy A. Wielkość przestrzeni stanów: mniej, niż 2^n, gdzie *n* - liczba kolumn.

Funkcja celu: wielkość zbioru B.

ALGORYTM ZACHŁANNY (1)

Zasada ogólna działania: budujemy rozwiązanie "po kawałku", na każdym etapie konstruowania odpowiedzi podejmujemy taką decyzję, by lokalnie dawała ona największe zyski.

ALGORYTM ZACHŁANNY (2)

Przykład: problem wyboru zajęć.
Danych jest n zajęć, każde z nich zaczyna się i kończy o ustalonej godzinie. Znaleźć taki podzbiór zajęć, by żadne dwa nie kolidowały ze sobą, a jednocześnie by wybrać ich jak najwięcej.

Algorytm: jako pierwsze wybieramy to zajęcie, które kończy się najwcześniej. Następnie w kolejnych krokach wybieramy te, które nie kolidują z poprzednio wybranymi i kończą się możliwie najwcześniej.

Algorytm zachłanny daje w tym przypadku zawsze rozwiązanie optymalne.

ALGORYTM ZACHŁANNY (3)

Problem komiwojażera.
W każdym kroku idziemy do najbliższego nieodwiedzonego miasta.

Problem pokrycia macierzy.
W każdym kroku wybieramy tę kolumnę, która pokrywa jak najwięcej dotychczas niepokrytych wierszy.

Problem plecakowy: mamy n przedmiotów, każdy o masie m_i i wartości w_i. Zmieścić w plecaku o ograniczonej pojemności M przedmioty o możliwie największej łącznej wartości.
Dodajemy kolejno te przedmioty, które mają największą wartość w przeliczeniu na masę, aż do wyczerpania się pojemności plecaka.

ALGORYTM WSPINACZKI

Schemat działania: startujemy z losowego punktu, przeglądamy jego sąsiadów, wybieramy tego sąsiada, który ma największą wartość ("idziemy w górę"), czynności powtarzamy do osiągnięcia maksimum lokalnego.

```
x0 = Random(X)
do
  max=x0
  for(x∈N(x0))
    if(f(x)>f(max)) max=x
  end for
  if(max=x0) break
  x0=max
while(1)
```

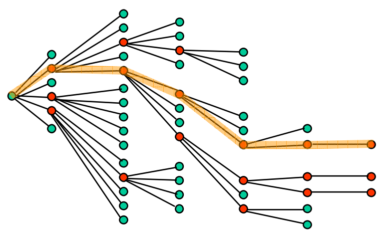
- Zalety: prosta implementacja, szybkie działanie.
- Wady: nieodporność na maksima lokalne, duża zależność wyniku od punktu startu
- Idea zbliżona do strategii zachłannej *(przeszukujemy przestrzeń gotowych rozwiązań, zamiast budować je stopniowo).*

PRZESZUKIWANIE WIĄZKOWE (1)

Schemat działania: budujemy rozwiązanie krok po kroku (jak w alg. zachłannym), zawsze zapamiętując k najlepszych rozwiązań i od nich startując w krokach następnych.

PRZESZUKIWANIE WIĄZKOWE (2)

Przykład: problem komiwojażera dla 7 miast (*k*=*3*)



1. Startujemy z losowego miasta.

2. Znajdujemy *k* miast najbliższych.

3. Startując z każdego z nich, liczymy odległości do miast dotąd nieodwiedzonych. Wybieramy *k* takich, by dotychczasowa długość drogi była minimalna.

4. Powtarzamy punkt 3. zapamiętując zawsze *k* najlepszych dróg.

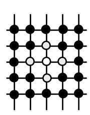
5. Gdy dojdziemy do ostatniego miasta, wybieramy najkrótszą z *k* zapamiętanych dróg.

SĄSIEDZTWO (1)

Zakładamy, że możemy na zbiorze X zdefiniować pojęcie “sąsiedztwa”: jeśli $x \in X$, to N(x) - zbiór (skończony) jego “sąsiadów”.

Taka definicja daje nam całkowitą dowolność w definiowaniu “sąsiadów”. W praktyce pojęcie to powinno być związane z konkretnym zadaniem.

Przykład: X - płaszczyzna. Za “sąsiadów” uznajemy punkty odległe w poziomie lub pionie o 0.01.

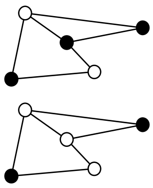


SĄSIEDZTWO (2)

Przykład: problem pokrycia wierzchołkowego.

W danym grafie G znaleźć taki najmniejszy zbiór wierzchołków B, że każda krawędź grafu kończy się na jednym z wierzchołków z B.

Przestrzeń stanów X - zbiór wszystkich potencjalnych pokryć (podzbiorów zbioru wierzchołków).



Za dwa pokrycia sąsiednie można uznać takie, które różnią się jednym wierzchołkiem.

DWA PODEJŚCIA

Zasada zachłanna:
tworzymy rozwiązanie stopniowo, na każdym kroku wybierając drogę maksymalizującą (lokalnie) funkcję jakości rozwiązania częściowego.

Przeszukiwanie sąsiedztwa:
definiujemy strukturę sąsiedztwa (określamy, które kompletne rozwiązania uznamy za sąsiednie, tzn. podobne) i badamy przestrzeń stanów przeskakując od sąsiada do sąsiada.