Wykład 1

Heurystyki przeszukiwania

Kazimierz Grygiel

O czym będzie ten wykład?

- Klasyfikacja algorytmów
  - Algorytmy dokładne (tradycyjne pojęcie algorytmu)
    Dla danej instancji problemu obliczają jego dokładne rozwiązanie
  - Schematy (algorytmy) aproksymacyjne
    Dla danej instancji problemu obliczają jego przybliżone rozwiązanie z zadaną dokładnością
  - Heurystyki
    Dla danej instancji problemu obliczają jego przybliżone rozwiązanie z nieznaną dokładnością, ale w akceptowalnym (kontrolowanym) czasie
- Tematem wykładu będą metody obliczeniowe oparte na różnych pomysłach heurystycznych, mające zastosowanie głównie w optymalizacji.

Podejście heurystyczne

- Etymologia: *heuriskein* (gr.) — znaleźć, odkryć (*heurisko* = znajduję, *eureka* = znalazłem)
- Potocznie: praktyczna, oparta na doświadczeniu, „inteligentna" reguła postępowania, która może drastycznie uprościć lub skrócić proces rozwiązywania problemu, gdy metoda rozwiązania nie jest znana lub jest zawiła i czasochłonna
- W algorytmice: „niepełnowartościowy" algorytm, który umożliwia znalezienie w akceptowalnym czasie przynajmniej „dostatecznie dobrego" przybliżonego rozwiązania problemu, choć nie gwarantuje tego *we wszystkich* przypadkach
- *Heurystyki przeszukiwania*: klasa uniwersalnych heurystyk (najczęściej zrandomizowanych) służących do przybliżonego rozwiązywania zadań wyszukiwania i optymalizacji

Home Page
Title Page
Page 3 of 21
Go Back
Full Screen
Close
Quit

Cele optymalizacji

- Pogląd „ortodoksyjny”

  Dążenie człowieka do perfekcji znajduje swój wyraz w teorii optymalizacji. Zajmuje się ona tym, jak opisać i osiągnąć Najlepsze, gdy wiemy już, jak mierzyć i zmieniać Dobre i Złe.

- Pogląd „pragmatyczny”

  Wyobraźmy sobie człowieka podejmującego decyzje, np. biznesmena. Na jakiej podstawie będziemy go oceniać? [...] Zazwyczaj osądzimy jego pracę pozytywnie, jeśli dokonuje on adekwatnego wyboru w dostępnym czasie i wykorzystując dostępne środki. Biznesmena ocenia się stosownie do jego zdolności konkurencyjnych. Czy produkuje lepszy model? Czy sprawniej dostarcza go na rynek? Czy zapewnił lepszą promocję? Nigdy nie oceniamy biznesmena na podstawie tego, czy osiągnął najlepszy możliwy wynik [...]. Tak więc [...] zbieżność do optimum nie jest celem ani w interesach, ani w większości innych zabiegów życiowych; wystarcza nam w zupełności, jeśli wypadamy lepiej od innych. [...] Głównym celem optymalizacji jest ulepszenie. Czy jesteśmy w stanie szybko osiągnąć dobry, satysfakcjonujący poziom wydajności?[1]

[1] D.E. Goldberg, *Algorytmy genetyczne i ich zastosowania*, WNT, Warszawa 2003

Zadania optymalizacyjne

- Dane:
  - niepusty zbiór S obiektów, zwanych *rozwiązaniami*
  - *funkcja oceny (jakości)* rozwiązań $f : S \to \mathcal{R}$
- Szukane:
  - *ekstremum globalne* (wartość optymalna) funkcji f (f^*)
  - *rozwiązanie ekstremalne, optimum*: rozwiązanie dla którego funkcja osiąga ekstremum (x^*)
- Warianty:

  maksymalizacja funkcji f: $\quad (x \in S)\ f(x^*) \geqslant f(x)$
  minimalizacja funkcji f: $\quad (x \in S)\ f(x^*) \leqslant f(x)$

- Problem złożoności: wiele praktycznych zadań optymalizacyjnych (np. *problem komiwojażera*) to zadania **trudne obliczeniowo**

Przykład: zadanie komiwojażera (TSP)

- Dane:
  - zbiór n *miast* $1, 2, \ldots, n$
  - macierz *odległości* między miastami $\|d_{ij}\|, d_{ij} > 0$
- Szukane: najkrótsza *trasa* prowadząca przez wszystkie miasta, czyli permutacja π taka, że suma

$$\sum_{i=1}^{n} d_{\pi(i),\pi(i+1)}$$

  jest najmniejsza z możliwych $(\pi(n+1) = \pi(1))$
- Liczba potencjalnych rozwiązań: $\frac{(n-1)!}{2}$ (wariant symetryczny)
- Nie są znane algorytmy o czasie wielomianowym dla TSP (w ogólnym przypadku)

Home Page
Title Page
Page 6 of 21
Go Back
Full Screen
Close
Quit

TSP — ilustracja

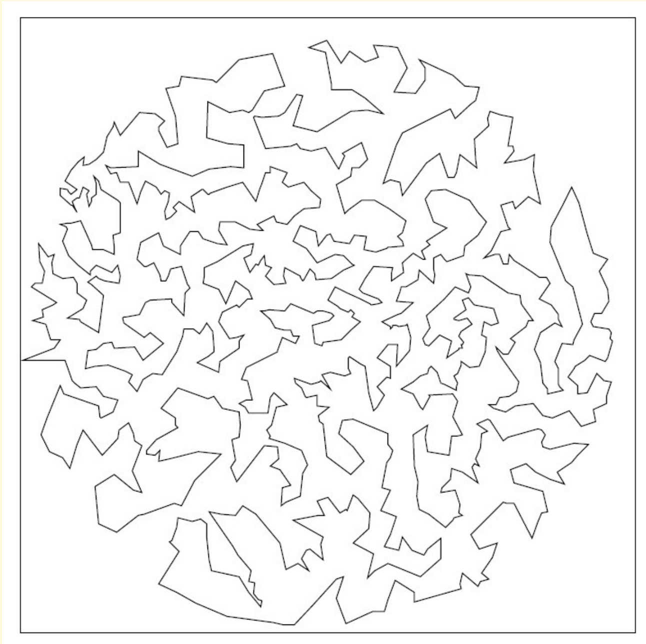


Fig. 1: Zadanie komiwojażera dla 1000 losowych miast na płaszczyźnie. Trasa bliska optymalnej.

Przykład:
optymalizacja parametryczna

- Jak znaleźć minimum globalne funkcji takiej, jak na poniższym wykresie, ale np. w 50 wymiarach?

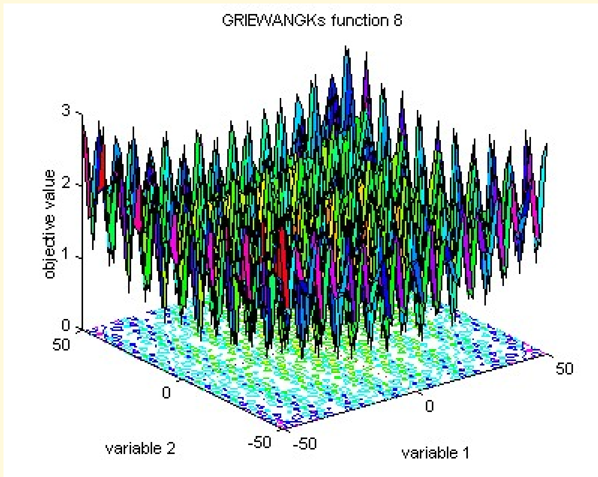


Fig. 2: Funkcja Griewangka w dwóch wymiarach.

Algorytm pełnego przeglądu

- Najprostszy koncepcyjnie algorytm dla problemów optymalizacyjnych ze *skończonym* zbiorem rozwiązań S
- Polega na sprawdzeniu wszystkich możliwych rozwiązań (*przeszukaniu* S)
- Wykorzystuje dwa moduły:
  - *generator rozwiązań*, który produkuje kolejne elementy zbioru S
  - *ewaluator*, który oblicza wartość funkcji oceny f dla bieżącego rozwiązania
- Pesymistyczna złożoność czasowa: $\Theta(|S|\,\mathrm{poly}(\log(|S|)))$
- Potencjalna heurystyka: przerywamy przeszukiwanie po spełnieniu zadanego *kryterium zatrzymania* (np. limitu liczby wygenerowanych rozwiązań) i zadowalamy się najlepszym znalezionym rozwiązaniem (*niepełny przegląd)*
- Pytanie: jak zbudować „najlepszy" generator?

Home Page
Title Page
Page 9 of 21
Go Back
Full Screen
Close
Quit

Inspiracje

- Pomysł: szukać wskazówek wykorzystując wiedzę zdobytą na wcześniejszym etapie
- Tę ideę poznajemy już w przedszkolu (niżej instrukcja dla przedszkolanki):

  Ciepło-zimno

  Reguły są proste: Najpierw pokazujemy dziecku, jakimi przedmiotami/obrazkami będziemy się dziś bawić. Potem pytamy je, co chciałoby, żebyśmy najpierw schowali. Kiedy mamy już wybraną zabawkę/ilustrację, dziecko na chwilę odwraca się i zamyka oczy, a my ukrywamy tę rzecz. Maluch próbuje ją znaleźć, a my mu pomagamy mówiąc „Ciepło”, gdy jest coraz bliżej miejsca, w którym jest ukryta, albo „Zimno”, gdy się od tego miejsca oddala.

- Jest to esencja heurystycznego przeszukiwania — z następującym zastrzeżeniem: informacja, którą zdobywamy, może być zwodnicza (tj. możemy otrzymywać fałszywe wskazówki)

Heurystyki ślepego przeszukiwania

- Prototyp: algorytm pełnego przeszukiwania
- Modyfikacje:
  - Generator ze sprzężeniem zwrotnym (oparty na historii):

$$\left.\begin{array}{cccc} x_0, & x_1, & \ldots, & x_k \\ y_0, & y_1, & \ldots, & y_k \end{array}\right\} \longrightarrow x_{k+1}$$

    gdzie $y_i = f(x_i)$
  - Kryterium zatrzymania (potencjalnie nieskończony ciąg rozwiązań)
- Przypadek szczególny — *algorytm enumeracyjny*:
  - niepowtarzalność rozwiązań ($x_i \neq x_j$ dla $i \neq j$)
  - „naturalne" kryterium zatrzymania (sprawdzenie wszystkich rozwiązań)
- Ograniczenia praktyczne: heurystyki z ograniczoną pamięcią

  Można zapamiętywać explicite pewien podzbiór rozwiązań lub „skompresowaną metainformację" o wygenerowanych rozwiązaniach, np. w postaci pewnych rozkładów prawdopodobieństwa

Złożoność heurystyk przeszukiwania

- Tradycyjna teoria złożoności nie daje się zastosować:
  - w ogólnym przypadku nie jesteśmy w stanie stwierdzić, czy w chwili zatrzymania algorytmu przeszukiwania optimum zostało znalezione
  - nie znamy kosztu działania generatora ani ewaluatora
- Jak porównywać (teoretycznie) różne heurystyki przeszukiwania?
- *Złożoność czarnoskrzynkowa*: koszt algorytmu = liczba kroków potrzebnych do wygenerowania optimum
- Czy istnieją algorytmy, które dla pewnych *klas* funkcji generują rozwiązania optymalne *średnio szybciej* niż inne?
- *No Free Lunch Theorem for Search* (D. Wolpert, W. McReady):

  Dla klasy $\mathcal{F}$ wszystkich funkcji $f : S \to U$ (S, U skończone) odpowiedź jest negatywna!
- Kontrowersje interpretacyjne

Uściślenia

- Model: Rozważamy dowolne (pod)klasy funkcji z X do Y, gdzie X, Y — skończone
- Algorytm: deterministyczna heurystyka przeszukiwania
- Złożoność czarnoskrzynkowa algorytmu A

  dla ustalonej funkcji f: $M_A(f)$

  średnia dla klasy funkcji $\mathcal{F}$:

$$M_A(\mathcal{F}) = \frac{1}{\mid \mathcal{F} \mid} \sum_{f \in \mathcal{F}} M_A(f)$$

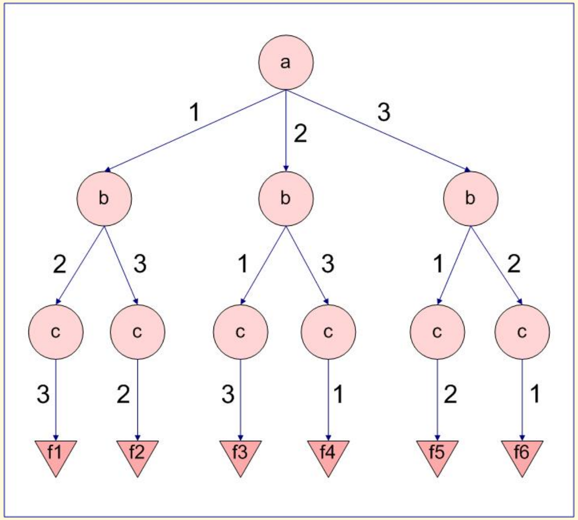
- Twierdzenie o NFL

  Jeśli $\mathcal{F}$ — klasa wszystkich funkcji z X do Y, to dla dowolnych dwóch algorytmów A_1, A_2 zachodzi równość $M_{A_1}(\mathcal{F}) = M_{A_2}(\mathcal{F})$

Drzewa decyzyjne

Przykład (drzewo decyzyjne dla algorytmu prostego przeglądu):
$S = \{a, b, c\}$, $U = \{1, 2, 3\}$, $\mathcal{F} = \{f : S \to U\}$
$|\mathcal{F}| = 3^3 = 27$.
Rozważmy tylko funkcje różnowartościowe (jest ich $3! = 6$)

	a	b	c
f_1	1	2	3
f_2	1	3	2
f_3	2	1	3
f_4	2	3	1
f_5	3	1	2
f_6	3	2	1



Home Page
Title Page
Page 14 of 21
Go Back
Full Screen
Close
Quit

Właściwości drzew decyzyjnych

- Terminologia:
  - ścieżka pełna: ścieżka rozpoczynająca się w korzeniu i kończąca się w liściu
  - sygnatura ścieżki: ciąg $(y_1, \ldots, y_k)$, $y_i \in U$ etykiet krawędzi tworzących ścieżkę
- Fakty: dla klasy $\mathcal{F}$ wszystkich funkcji $f : S \to U$
  - każdy algorytm enumeracyjny jest jednoznacznie wyznaczony przez swoje drzewo decyzyjne
  - każda ścieżka pełna wyznacza pewną funkcję $f : S \to U$ i każda funkcja $f : S \to U$ odpowiada pewnej ścieżce pełnej
  - wszystkie drzewa decyzyjne mają *ten sam zbiór sygnatur ścieżek*
  - złożoność czarnoskrzynkowa algorytmu enumeracyjnego dla funkcji f jest funkcją sygnatury odpowiedniej ścieżki
- Wniosek: Średnia złożoność czarnoskrzynkowa dla klasy $\mathcal{F}$ *nie zależy od algorytmu* (NFL!)

Home Page
Title Page
Page 15 of 21
Go Back
Full Screen
Close
Quit

Zrandomizowane heurystyki przeszukiwania

- Algorytm *zrandomizowany* wykorzystuje do obliczeń *generator liczb pseudolosowych*, symulując rozmaite zdarzenia losowe
- W heurystykach przeszukiwania randomizacja odgrywa istotną rolę
- Schemat formalny działania zrandomizowanego generatora rozwiązań:

$$\left.\begin{array}{cccc} x_0, & x_1, & \ldots, & x_k \\ y_0, & y_1, & \ldots, & y_k \end{array}\right\} \longrightarrow \pi_{k+1} \rightsquigarrow x_{k+1}$$

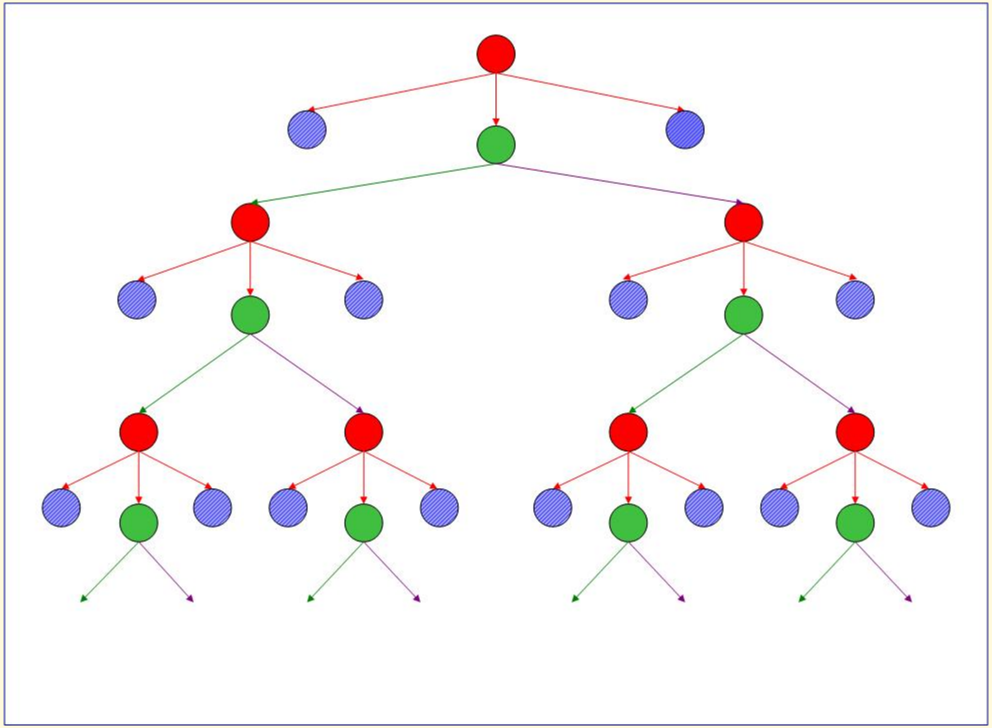
  gdzie $y_i = f(x_i)$ oraz π_{k+1} jest rozkładem prawdopodobieństwa na S, służącym do wylosowania rozwiązania x_{k+1}
- Uwaga: W rzeczywistości „losowanie” rozwiązania nie wymaga *jawnej* znajomości rozkładu; występuje on tylko w opisie teoretycznym

Równoważny opis

- Każdej zrandomizowanej heurystyce przeszukiwania (dla pewnej klasy funkcji $\mathcal{F}$) odpowiada *zrandomizowane drzewo decyzyjne*
- Zrandomizowane drzewo decyzyjne można scharakteryzować, określając rozkład prawdopodobieństwa na pewnym zbiorze *deterministycznych drzew decyzyjnych*
- Zatem zrandomizowana heurystyka przeszukiwania to w istocie pewna rodzina deterministycznych heurystyk przeszukiwania z określonym na niej rozkładem prawdopodobieństwa

	$\ldots$	A_j	$\ldots$
$\vdots$			
f_i		$M_{A_j}(f_i)$	
$\vdots$			

Zrandomizowane drzewo decyzyjne



Komentarze i uzupełnienia

- Ponieważ twierdzenie NFL jest prawdziwe dla *każdego* algorytmu enumeracyjnego, więc jest prawdziwe również dla *zrandomizowanego* algorytmu enumeracyjnego
- Twierdzenie NFL można przenieść na przypadek dowolnych „niezdegenerowanych" heurystyk przeszukiwania, jeśli koszt mierzyć liczbą *różnych* rozwiązań wygenerowanych do osiągnięcia optimum
- Czy twierdzenie NFL zachodzi dla węższych klas funkcji?
- Schumacher, Vose, Whitley udowodnili, że:

  > Twierdzenie NFL zachodzi dla klasy funkcji $\mathcal{F}$ wtedy i tylko wtedy, gdy $\mathcal{F}$ jest zamknięta ze względu na permutacje

- Def.: Klasa $\mathcal{F}$ jest zamknięta ze względu na permutacje wtw, gdy dla dowolnej permutacji $\pi : X \to X$ jeśli $f \in \mathcal{F}$, to $(\pi f) \in \mathcal{F}$, gdzie $(\pi f)(x) = (f(\pi^{-1}x))$
- Większość interesujących klas funkcji *nie jest* zamknięta ze względu na permutacje (co pozostawia nam nadzieję na skonstruowanie wydajnej heurystyki dla danej klasy, o ile potrafimy ją dobrze scharakteryzować)

Przykład: Funkcje wskaźnikowe elementu

- Niech $\mid S \mid = n$. Klasa funkcji

$$\mathcal{F} = \{f_a : S \to \{0,1\} \mid a \in S\}$$

gdzie

$$f_a(x) = \begin{cases} 1 & \text{gdy } x = a \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

jest zamknięta ze względu na permutacje

- Są to tzw. „igły w stogu siana"
- Średnia złożoność czarnoskrzynkowa algorytmu prostego przeglądu w klasie $\mathcal{F}$ jest równa

$$M_0(\mathcal{F}) = \frac{1}{n}\sum_{i=1}^{n} i = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}$$

i tak samo jest dla dowolnego ślepego enumeratora

Literatura źródłowa

Droste S., Jansen Th., Wegener I. (2003) *Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization* Electronic Colloquium on Computational Complexity, Report No. 48

Schumacher C. Vose M.D., Whitley L.D. (2001) *The No Free Lunch and Problem Description Length* Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) pp. 565–570

Wolpert D. H., McReady W. G. (1996) *No Free Lunch Theorems for Search* Technical Report SFI-TR-95-02-010 Santa Fe Institute