Systemy operacyjne
22.10.2010

Złożoność obliczeniowa ciąg dalszy

Złożoność obliczeniowa – funkcja podająca zależność między czasem zużywanym przez ten algorytm na skonstruowanie rozwiązania dla konkretnego problemu, a rozmiarem problemy. Wejście: wielkość problemu Wyjście: czas obliczeń.

Algorytmy efektywne – liczba obliczeń rośnie w nich wielomianowo wraz ze wzrostem rozmiaru

Algorytmy nieefektywne – liczba obliczeń rośnie niewielomianowo, ale np. wykładniczo, problem jest wtedy nierozwiązywalny dokładnie

Algorytm wielomianowy – algorytm którego złożoność obliczeniowa jest rzędu O(p(k)), gdzie p – pewien wielomian, k – rozmiar problemu. Każdy algorytm którego funkcja złożoności nie może być tak ograniczona nazywa się **algorytmem wykładniczym** – choć funkcja złożoności obliczeniowej tych algorytmów nie musi być funkcją wykładniczą. Dojście do rozwiązania obarczone jest bardzo dużym czasem.

Dla zastosowań także w systemach komputerowych rezygnuje się często z **nieefektywnych optymalnych algorytmów** operacyjnych na rzecz **efektywnych algorytmów suboptymalnych** – przybliżających, aproksymacyjnych (heurystycznych)

Heurystyka – rozwiązanie problemu w sposób przybliżony

Metody naturalne, inteligencja obliczeniowa, sztuczna inteligencja: **metaheurystyki** (mają pewne ogólne sformułowania, które dostosowuje się do problemu).

Problemy które wymagają dla swego optymalnego rozwiązania algorytmu o czasie trwania wykładniczo zależnym od rozmiarów problemu (tzn. algorytmu nieefektywnego) nazwiemy problemami klasy **NP-zupełnej**. Natomiast klasę **P-zupełną** definiuje się jako obejmującą wszystkie problemy dla których istnieją algorytmy efektywne.

Przy konstruowaniu efektywnych algorytmów istotna jest cena odległości (w sensie wartości rozpatrywanego kryterium) od rozwiązania optymalnego dokonywana na drodze przeliczenia reprezentatywnej liczby przykładów. Do tej oceny niezbędna jest oczywiście znajomość rozwiązania optymalnego stąd nie można zrezygnować z konstruowania nieefektywnych algorytmów optymalnych.

Algorytmy suboptymalne (przybliżone, heurystyczne) mogą być algorytmami dedykowanymi (specjalizowanymi) do rozwiązywania specyficznego problemu operacyjnego, lub mogą być algorytmami tzw. metaheurystykami tzn. opartymi o ogólne sparametryzowane metody obliczeniowe w których dla danego specyficznego problemu są definiowane sposoby reprezentacji, obliczeń, kryteriów itd.
Takimi metaheurystykami są np. algorytmy genetyczne, sieci neuronowe, algorytm poszukiwania z tabu, algorytm symulowania wyżarzania, metyody podziału obliczeń.

Przykład problemu NP-zupełnego (niewielomianowego)
problem pakowania (w zagadnieniu przydziału pamięci)

n elementów o rozmiarach a_i, przyjmujemy, że ten rozmiar jest od (0 do 1), elementy te mamy włożyć do pudełek o jednostkowych pojemnościach tak, aby liczba pudełek była jak najmniejsza.

Aproksymacyjne algorytmy pakowania:

- według kolejności (FF – first fit), wkładanie do pudełka o możliwie najmniejszym numerze
- według dopasowania elementów (BF – best fit), dopasowanie do pudełka w którym została jak najmniejsza przestrzeń
- dopasowanie uporządkowanych elementów (FFD – first fit decreasing), wersja FF, nierosnące wartości elementów
- najlepsze dopasowanie uporządkowanych elementów (BFD – best fit decreasing), wersja BF, nierosnące wartości elementów