PORR Projekt

Algorytm świetlika (FA) i roju świetlików (GSO) do rozwiązania problemu plecakowego.

Cezary Dubiel, 251387
Tomasz Korzeniowski, 265753

19 grudnia 2017

1 Zadanie

Zdefiniować i rozwiązać dwa zadania o różnej złożoności. Zrealizować zadania w wersji sekwencyjnej i równoległej. Opisać przyśpieszenie obliczeń w zależności od stopnia zrównoleglenia (porównać obie metody - FA i GSO). Przedstawić graficznie zbieżność obu algorytmów w wersji sekwencyjnej - wartości funkcji celu w kolejnych iteracjach.

2 Podstawy teoretyczne

2.1 Problem plecakowy

Problem plecakowy polega na zapakowaniu maksymalnie cennego zbioru przedmiotów do plecaka, tak by nie przekroczyć jego ładowności. Można wyróżnić dwa podejścia do liczby pakowanych przedmiotów. W ogólnym podejściu zakładamy, że posiadamy wiele $(1, 2, \ldots)$ egzemplarzy każdego z N różnych przedmiotów. W bardziej rzeczywistych przypadkach posiadamy dokładnie jeden egzemplarz każdego z N różnych przedmiotów. W tym projekcie zajmiemy się drugim podejściem.

Zakładamy, że mamy do dyspozycji $n \in N$ przedmiotów. Każdy z tych przedmiotów jest warty c, a także posiada swoją wagę m. Naszym zadaniem jest zdecydowanie czy i które z przedmiotów załadujemy, tak by ich sumaryczna masa nie może przekroczyła ładowności plecaka M_{max}. Model matematyczny problemu plecakowego można przedstawić następująco

$$\max \sum_{i}^{N} c_i x_i \tag{1}$$

$$\sum_{i}^{N} m_i x_i \leqslant M_{max} \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

Zatem poszukujemy rozwiązania w postaci binarnego wektora x długości N:

$$x_i = \begin{cases} 1, & \text{gdy i-ty przedmiot zostanie zapakowany do placaka} \\ 0, & \text{w przeciwnym przypaku} \end{cases}$$

Jeśli przyjmiemy, że posiadamy sześć przedmiotów, które możemy spakować, a także znamy $C = [c_i]_{1\times N}$, $M = [m_i]_{1\times N}$ oraz wektor $x = (0, 1, 0, 0, 1, 1)$ to wiemy, że do plecaka został załadowany przedmiot drugi, piąty oraz szósty.

2.2 Algorytm świetlika

Algorytm świetlika (FA - ang. $fireflyalgorithm$) jest heurystyką inspirowaną zachowaniem społecznym świetlików (robaczków świętojańskich). Jego głównym zastosowaniem są zadania optymalizacji ciągłej z ograniczeniami. Może jednak być wykorzystywany także w problemach optymalizacji kombinatorycznej.

Fenomen świetlików polega na ich bioluminescencyjnej komunikacji. Każdy z nich cechuje się pewną atrakcyjnością, proporcjonalną do jasności z jaką świecą. Świetlik o większej jasności przyciąga inne osobniki (przyjmujemy obupłciowość świetlików) o mniejszej jasności. Zakładamy też, że świetliki nie umierają.

Na początku działania algorytmu losowana jest populacja świetlików. Każdy z nich reprezentuje pewne rozwiązanie problemu, czyli poszukiwany wektor x. Jego jasnością jest wartość funkcji celu dla danego rozwiązania. Następnie poszukiwany jest najlepszy ze świetlików (pod względem jasności - wartości funkcji celu), a pozostałe poruszają się w jego kierunku. Ruch i-tego świetlika zależy od wartości funkcji atrakcyjności β_i

$$\beta_i = \beta_0 e^{-\gamma r_{ij}^2} \tag{4}$$

gdzie β_0 (maksymalna atrakcyjność) oraz γ (współczynnik absorpcji) są ustalonymi na wstępie parametrami algorytmu, a r_{ij}^2 to odległość między świetlikami i oraz j.

Ruch świetlika i przyciąganego do bardziej atrakcyjnego świetlika j jest następujący

$$x_i = x_i + \beta_i(x_j - x_i) + u_i \tag{5}$$

gdzie u_i jest wektorem liczb losowych wygenerowanych z rozkładem równomiernym z przedziału [0,1]. Jeśli świetlik nie znajdzie się w sąsiedztwie innego świetlika o większej jasności wówczas wykonywane jest jego losowe przesunięcie u_i.

Algorytm świetlika można przedstawić w postaci pseudokodu

Algorytm 1 Algorytm świetlika (FA)

Require: C, M, k, β_i, γ
  wylosuj populację świetlików
  while nie osiągnięto warunku stopu **do**
    for i=1 to k **do**
      for j=1 to k **do**
        if $f(x_j) > f(x_i)$ **then**
          oblicz odległość r_{ij}
          zmodyfikuj atrakcyjność β_i
          wygeneruj losowy wektor u_i
          utwórz nowe rozwiązanie zgodnie z (5)
        end if
      end for
    end for
    wygeneruj losowy wektor u_b
    przesuń najlepszego świetlika $x_b = x_b + u_b$
  end while

$f(x_j)$ oznacza wartość funkcji celu dla świetlika x_j. Warunkiem stopu algorytmu może być zadana liczba iteracji. Do sprawdzenia zbieżności algorytmu warunkiem stopu będzie brak zmian rozwiązania (na pewnym poziomie) przed zadaną liczbę iteracji.

2.3 Algorytm roju świetlików

Algorytm roju świetlików (GSO - ang. *glowwormswarmoptimization*), w przeciwieństwie do FA, oparty jest na zachowaniach społecznych nieskrzydlatych świetlików. Przejawia się to zdolnością do zmiany intensywności emisji lucyferny, czyli wydawaniem blasków o różnym natężeniu. Od algorytmu świetlika, GSO różni się także ze względu na możliwość podziału świetlików na podgrupy. Pozwala to osiągać wiele lub nawet wszystkie optima lokalne.

Różna jest także reprezentacja sąsiedztwa świetlików. Zmienna sąsiedztwa r_d jest związana z zakresem radialnym r_s, tak aby $0 < r_d \leqslant r_s$. Wynika z tego, że świetlik i bierze pod uwagę świetlika j jako sąsiada, jeśli j jest wewnątrz obszaru sąsiedztwa rozwiązania i oraz poziom lucyferny jest wyższy dla j-tego niż i-tego.

Algorytm rozpoczyna się od losowego rozmieszczenia świetlików, a także przypisaniu im wielkość lucyferny równą 0. Przebieg algorytmu składa się z 3 faz głównych. W fazie pierwszej następuje modyfikacja poziomu lucyferny, czyli wyznaczenie wartości funkcji celu. W oryginalnej wersji algorytmu odejmuje się część lucyferny w celu symulacji mechanizmu rozkładu tej substancji w czasie. Ze względu na to, że przedmioty przynoszą stały zysk wynikający z ich zapakowania, zrezygnujemy z takiego pomniejszania.

Kolejnym krokiem jest faza ruchu. Wyznaczane jest prawdopodobieństwo przemieszczenia świetlika i w kierunku j w chwili t zgodnie ze wzorem

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \tag{6}$$

$$N_i(t) = \{j : d_{ij}(t) < r_d(t); l_i(t) < l_j(t)\} \tag{7}$$

gdzie $N_i(t)$ to zbiór sąsiadów świetlika i, a $d_{ij}(t)$ odległość między świetlikiem i oraz j.

Następnie świetliki są przemieszczane zgodnie z

$$x_i(t+1) = x_i(t) + s\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \tag{8}$$

gdzie $s > 0$ oznacza rozmiar kroku przemieszczenia.

Ostatnią fazą jest modyfikacja zakresu sąsiedztwa.

$$r_d(t+1) = \min\left\{r_s, \max\left\{0, r_d(t) + \beta(n_t - |N_i(t)|)\right\}\right\} \tag{9}$$

gdzie β to pewna stała, a n_t jest parametrem używanym do sterowania liczbą świetlików w sąsiedztwie.

Algorytm 2 Algorytm roju świetlików (GSO)

Require: C, M
  wylosuj populację świetlików
  while nie osiągnięto warunku stopu **do**
    zmodyfikuj poziom lucyferny każdego świetlika
    przemieść każdego świetlika (7)
    for każdego świetlika $j \in N_i(t)$ **do**
      wyznacz prawdopodobieństwo przemieszczenia (6)
      przemieść świetliki (8)
      zmodyfikuj zakres sąsiedztwa (9)
    end for
  end while

3 Wybrane technologie

Implementacja algorytmów została wykonana w języku C++ 11. Do implementacji i testów wykorzystano zintegrowane środowisko programistyczne Code Blocks .

Do realizacji wersji równoległej algorytmów z pamięcią wspólną wykorzystano środowisko OpenMP.

3.1 Sprzęt

Procesor	Intel Core i7-4770K 3.5 GHz, 4 rdzenie, 8 wątków (korzystanie z hiper-wątków)
Pamięć RAM	16 GB
System operacyjny	Windows 7

4 Wyniki

Dla każdego z algorytmów wykonano 10 symulacji rozwiązania problemów o różnych złożonościach (liczbie możliwych elementów do spakowania). Symulacje te zostały wykonane w wersji sekwencyjnej oraz równoległej dla czterech i ośmiu wątków. Większej liczby wątków nie ma sensu rozważać, ponieważ używany sprzęt nie wystarczy do większego zrównoleglenia obliczeń.

4.1 FA

Tabela 1: Czas i przyśpieszenie wykonania algorytmu świetlika (FA)

złożoność	sekwencyjnie	4 wątki		8 wątków	
		czas	przyśpieszenie	czas	przyśpieszenie
24	0,0753906	0,17625	0,42775	0,0847009	0,89008
120	0,215828	0,46174	0,46742	0,302861	0,71263
600	1,27815	1,23022	1,03896	0,747475	1,70996
960	1,51908	1,1197	1,35668	0,698841	2,17371

4.2 GSO

Tabela 2: Czas i przyśpieszenie wykonania algorytmu roju świetlików (GSO)

złożoność	sekwencyjnie	4 wątki		8 wątków	
		czas	przyśpieszenie	czas	przyśpieszenie
24	0,0963501	0,3246547	0,29678	0,3521662	0,27359
120	0,245403	0,237385	0,91779	0,214741	1,14279
600	0,813467	0,612114	1,32895	0,661251	1,23019
960	1,30946	0,892147	1,46776	0,711577	1,84022

Literatura

[1] http://www-users.mat.uni.torun.pl/~henkej/knapsack.pdf

[2] http://pe.org.pl/articles/2014/6/37.pdf

[3] http://home.agh.edu.pl/~slukasik/pub/021_Lukasik_KAEiOG2011(presentation).pdf

[4] https://eti.pg.edu.pl/documents/176546/25263566/SZ_wyklad2.pdf