Filip Rurak, 244960

Algorytmy Metaheurystyczne

Referat – rozwiązanie problemu komiwojażera z użyciem roju cząstek

Spis treści

1 Opis problemu

Problem komiwojażera, z ang. Traveling Salesman Problem (TSP), jest zagadnieniem optymalizacyjnym, przedstawiającym podróżnego (komiwojażera), chcącego odwiedzić wszystkie miasta na podanej mapie dokładnie raz, robiąc to jak najmniejszym kosztem / najkrótszym czasem / w najmniejszej odległości.

2 Podejście heurystyczne

Do rozwiązania problemu komiwojażera zostanie użyta heurystyka – rój cząstek, z ang. Particle Swarm Optimalization (PSO). Jest to metoda minimalizująca daną funkcję wejściową, używając danych cząstek poruszających się po przestrzeni z zadaną własną zmienną pozycją i prędkością. Za pomocą formuł matematycznych, każda cząstka zbioru będzie poruszać się w kierunku najlepszego (najmniejszego) rozwiązania.

Prędkość i nowa pozycja cząstki obliczane są w następujący sposób:

$$v_{n+1} = v_n \cdot w + \varphi 1 \cdot (p - x_n) + \varphi 2 \cdot (g - x_n)$$

$$x_{n+1} = x_n + v_{n+1}$$

Gdzie x_n jest aktualnym położeniem cząstki, p jest najlepszym położeniem w jakim się znajdowała, g jest najlepszym globalnym położeniem wszystkich cząstek, a $w, \varphi 1, \varphi 2$ są prawdopodobieństwem dla których te wartości mają się zerować.

TSP nie jest jednak funkcją ciągłą, więc cząstki nie mogą zawsze wykonać ruch przy określonej prędkości. Z tego powodu położenie cząstki i jej prędkość będą wyglądały w następujący sposób:

- Aktualne położenie pos będzie przedstawione jako lista $[c_1, c_2, c_3, \dots, c_n]$, gdzie kolejne wyrazy listy oznaczają kolejne miasta do odwiedzenia. Każdy wyraz c_n oprócz pierwszego i ostatniego jest unikalny.
- Aktualna prędkość v będzie wyglądała jako ciąg par, gdzie każda kolejna para oznacza, które miejsca w ciągu pos zostaną zamienione.

Przykład:

$$pos = [0, 1, 2, 3, 4, 5, 0]$$

$$v = [(1, 2), (2, 4)]$$

To pozycja pos po zmianie dla danej prędkości będzie wynosić:

$$pos = [0, 2, 4, 3, 1, 5, 0]$$

Cała metoda będzie bazować na algorytmach przedstawionych w następnym punkcie.

3 Opis algorytmów

Zostaną dodane trzy ważne algorytmy, które będą potrzebne do implementacji podejścia heurystycznego.

Pierwszym jest funkcja $seq(pos1, pos2)$, która dla dwóch pozycji cząstek, $pos1$ i $pos2$ policzy prędkość (sekwencję) v dla której musi poruszać się cząstka $pos2$ aby znaleźć się w $pos1$. Będzie to ciąg par, oznaczających kolejne zamiany wartości miejsc $pos2$.

Drugim jest funkcja $addSeq(pos, v)$ modyfikująca daną pozycję pos dla danej prędkości v. Ciąg wyjściowy będzie takiej samej długości jak ciąg wejściowy, jednak ze zmienionymi pozycjami.

Ostatnim algorytmem będzie funkcja $elim(p, v)$, która z zadanym prawdopodobieństwem p będzie kolejno usuwać wyrazy danej prędkości.

Dla tak przedstawionych algorytmów, główne funkcja metody PSO na obliczanie prędkości i pozycji:

$$v_{n+1} = v_n \cdot w + \varphi 1 \cdot (p - x_n) + \varphi 2 \cdot (g - x_n)$$

$$x_{n+1} = x_n + v_{n+1}$$

Zostaną przedstawione jako:

$$v_{n+1} = elim(w, v_n) + elim\big(\varphi 1, seq(p, x_n)\big) + elim\big(\varphi 2, seq(g, x_n)\big)$$

$$x_{n+1} = addSeq(x_n, v_{n+1})$$

Głowna funkcja programu, dopóki nie przekroczy ustalonego maksymalnego czasu t, dla każdej cząstki wykonuje powyższe operacje, a następnie sprawdza, czy otrzymany kosz przejścia komiwojażera dla danej ścieżki (pozycji cząstki) nie jest mniejszy niż aktualne minimum. Jeśli tak to najmniejsza ścieżka i minimalny koszt przejścia są aktualizowane.

4 Dane wejściowe / Dane wyjściowe

Dane wejściowe będą ustalone podobnie jak zadanie 2 z listy 1. W pierwszej linii wejścia będą umieszczone, oddzielone spacją liczby zmiennoprzecinkowe $t, n, s, w, \varphi 1, \varphi 2$, gdzie t jest limitem czasu na wykonanie algorytmu, n jest liczbą miast do odwiedzenia, s jest ilością cząstek, jaką algorytm ma użyć, a $w, \varphi 1, \varphi 2$ są parametrami wejściowymi kontrolującymi prędkość cząstek. W kolejnych n liniach będą znajdowały się odległości pomiędzy miastami. Zostały użyte następujące ograniczenia:

- Odległość z miasta A do A zawsze wynosi 0
- Odległość z miasta A do B nie musi być taka sama jak z miasta B do A
- Jeśli między miastami nie ma połączenia, to odległość musi być ustawiona na -1

Dane wyjściowe będą przedstawiały kolejne otrzymane wyniki, ścieżkę podaną dla danego wyniku oraz liczbę iteracji algorytmu. Następna linia będzie drukowana tylko jeśli nowy znaleziony wynik jest optymalniejszy niż poprzedni.

5 Wyniki

Dla $t = 1, n = 5, s = 5, w = 0.2, \varphi 1 = 0.9, \varphi 2 = 1$ oraz dla odległości przedstawionych poniżej:

	c1	c2	c3	c4	c5
c1	0	1	5	1	6
c2	1	0	5	1	100
c3	7	20	0	4	21
c4	1	7	4	0	11
c5	50	50	55	50	0

Otrzymano następujące wyniki:

```
Minimalna sciezka: [0, 4, 1, 3, 2, 0] odleglosc: 68 iteracje: 2

Minimalna sciezka: [0, 4, 1, 2, 3, 0] odleglosc: 66 iteracje: 4


Najkrotsza znaleziona sciezka: [0, 4, 1, 2, 3, 0] odleglosc: 66
```

Co jest najmniejszą odległością dla danej mapy miast zaczynając od miasta c1.

6 Źródła

1. K.P. Wang et al. PARTICLE SWARM OPTIMIZATION FOR TRAVELING SALESMAN PROBLEM
2. Sarman K. Hadia, Arjun H. Joshi, Chaitalee K. Patel
   Solving City Routing Issue with Particle Swarm Optimization
   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.7026&rep=rep1&type=pdf