ALHE - Dokumentacja

Maciej Kapuściński, Sebastian Pietras

Treść zadania

SK.ALHE.12

Dla sieci o nazwie *india35* ze strony http://sndlib.zib.de/home.action zastosować algorytm mrówkowy (*Ant Colony*) do znalezienia n najlepszych (wg. ustalonej metryki) ścieżek w danej sieci. Porównanie z innym algorytmem będzie dodatkowym atutem.

Przyjęte założenia i doprecyzowanie treści

Sieć zostanie potraktowana jako graf ważony nieskierowany. Wagi zostaną przypisane krawędziom zgodnie z kolumną *module cost*, która znajduje się w pliku definiującym sieć.

Ścieżki będą wyszukiwane między wierzchołkiem początkowym a końcowym. Te wierzchołki będą podawane jako parametr programu.

Jako metryka zostanie wykorzystany koszt ścieżki. To znaczy, że najlepsza jest ta ścieżka, dla której suma wag jej krawędzi jest najmniejsza.

Implementacja zostanie wykonana z użyciem języka programowania Python.

Opis algorytmu

Znajdowanie jednej najlepszej ścieżki

Algorytm wysyła m mrówek (a raczej ich abstrakcyjnych reprezentacji) przez graf I razy, gdzie I to ustalona ilość iteracji, po czym zapamiętywana jest najkrótsza znaleziona przez mrówki ścieżka.

Mrówki, które znalazły ścieżkę zostawiają na niej feromony. Lepsze ścieżki powinny częściej zawierać większą ilość feromonu.

W każdym kroku mrówki, które nie dotarły jeszcze do wierzchołka końcowego, wybierają wierzchołek, do którego się przemieszczą. Brane pod uwagę są jedynie wierzchołki sąsiadujące z wierzchołkiem, w którym aktualnie znajduje się mrówka. Decyzja o wybranym wierzchołku zostanie podjęta losowo, lecz zgodnie z prawdopodobieństwami przypisanymi do każdej krawędzi.

Prawdopodobieństwo wyboru krawędzi jest opisane wzorem $p_{xy} = \frac{\tau_{xy}^{\alpha}\eta_{xy}^{\beta}}{\sum_{z \in N(x)} \tau_{xz}^{\alpha}\eta_{xz}^{\beta}}$, gdzie:

τ_{xy} - ilość feromonu na krawędzi xy

α - parameter kontrolujący wpływ τ

η_{xy} - "atrakcyjność" krawędzi xy (w naszym przypadku odwrotność wagi tej krawędzi)

β - parametr kontrolujący wpływ η

$N(x)$ - wierzchołki sąsiadujące z wierzchołkiem x

Gdy wszystkie mrówki znajdą ścieżkę lub przekroczą limit kroków, nastąpi aktualizacja feromonów na krawędziach grafu. Do wartości feromonu danej krawędzi będzie przypisywana wartość $\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^{k}$, gdzie:

τ_{xy} - wartość feromonu krawędzi xy

ρ - współczynnik parowania (zanikania) feromonów

m - liczba mrówek

$\Delta\tau_{xy}^{k}$ - ilość feromonu pozostawiona na krawędzi xy przez mrówkę k

$\Delta\tau_{xy}^{k}$ wynosi 0 jeśli mrówka k nie przeszła przez krawędź xy, a w przeciwnym przypadku $\frac{Q}{L_k}$, gdzie:

Q - współczynnik nakładania feromonów

L_k - koszt całej drogi mrówki k w tej iteracji

Znajdowanie n najlepszych ścieżek

Aby znaleźć $(i+1)$-szą najlepszą ścieżkę należy zablokować poprzednie i ścieżek w grafie, a następnie wyszukać w takim grafie najlepszą ścieżkę. Można tak iteracyjnie postępować aż do znalezienia n najlepszych ścieżek.

Zablokowanie ścieżek w grafie będzie polegać na usunięciu po jednej krawędzi z każdej ścieżki. Ta czynność zostanie powtórzona dla wszystkich możliwych doborów krawędzi. We wszystkich powstałych grafach zostanie wyszukana najlepsza ścieżka w danym grafie. Z wyszukanych ścieżek zostanie wybrana ta, która była ogółem najlepsza.

Uruchomienie programu

```
python -m antcolony [--n N] [--n_ants N_ANTS] [--n_iter N_ITER]
                    [--max_steps MAX_STEPS]
                    [--alpha ALPHA] [--beta BETA] [--ro RO] [--q Q]
                    graph start_node end_node
```

Argumenty:

- `graph` - ścieżka do pliku z definicją grafu
- `start_node` - wierzchołek początkowy
- `end_node` - wierzchołek końcowy
- `--n N` - liczba ścieżek do znalezienia (domyślnie 1)
- `--n_ants N_ANTS` - liczba "mrówek" w algorytmie mrówkowym (domyślnie 50)
- `--n_iter N_ITER` - liczba iteracji w algorytmie mrówkowym (domyślnie 10)
- `--max_step MAX_STEP` - maksymalna liczba kroków mrówki w jednej iteracji (domyślnie 1000)
- `--alpha ALPHA` - współczynnik α (domyślnie 0.5)
- `--beta BETA` - współczynnik β (domyślnie 1.2)
- `--ro RO` - współczynnik ρ (domyślnie 0.6)
- `--q Q` - współczynnik Q (domyślnie 10)

Program wypisze na standardowe wyjście ścieżki znalezione przez algorytm mrówkowy oraz, dla porównania, przez algorytm Dijkstry.

Wykorzystane narzędzia

- `Python`
- `NetworkX` (do reprezentacji grafu)
- `NumPy` (do operacji na macierzach)