SK6. Atak na sieć (II)

Raport wstępny do projektu w ramach kursu "Grafy i Sieci" (GIS)

Patryk Kocielnik, Jan Kumor, 5.04.2018r.

Opiekun projektu

dr inż. Sebastian Kozłowski

Opis zadania

Dane są dwie sieci: euklidesowa i losowa (ER) o mniej więcej takiej samej liczbie wierzchołków i krawędzi. Porównać prawdopodobieństwa powodzenia ataku na losowe krawędzie tych sieci (udany atak to taki, który prowadzi do rozspójnienia sieci).

Planowane wykorzystanie narzędzi w projekcie

- Środowisko rozwiązania: stacja robocza pod kontrolą systemu GNU/Linux,
- Język implementacji rozwiązania: Python,
- Narzędzie do wizualizacji wyników i referencyjnej weryfikacji rozwiązań: Graph-Tool - biblioteka dla języka Python,

Składniki rozwiązania

1. Moduł generacji sieci: euklidesowych oraz losowych, o zadanej liczbie wierzchołków.
   - Sposób wywołania: `graph = generate_graph(graph_type, vertex_probability)`,
   - Rezultat wywołania: `graph` jako dwuwymiarowa macierz sąsiedztwa (`int * int`) opisująca wygenerowany graf,
   - Podstawą komponentu będzie moduł generacji sieci z pakietu Graph-Tools [1].
2. Filtr usuwający z grafu losowo wybraną krawędź
   - sposób wywołania: `new_graph = break(graph)`.
   - rezultat wywołania: `new_graph` jako graf pozbawiony losowo wybranej krawędzi.

3. Analizator spójności sieci
    - Sposób wywołania: consistency_degree(graph),
    - Rezultat wywołania: n (`int`) jako liczba oznaczająca stan spójności grafu wejściowego: 0 - niespójny, 1 - spójny,

Interfejs aplikacji

Interfejsem aplikacji będzie konsola tekstowa. Motywacją tego podejścia jest łatwość łączenia aplikacji z interfejsem tekstowym w filtry, które później wykorzystać można do analizy bardziej złożonych struktur.

Przebieg eksperymentu

Liczbę iteracji k ustal na wartość z przedziału od 1 do 25. Liczbę wierzchołków v ustal na należącą do zbioru V_num: 10, 100, 1000, 10000, 100000, Liczbę krawędzi ustal na należącą do zbioru E_num: 10, 100, 1000, 10000, 100000.

1. Powtórz dla k przypadków:
2. Wygeneruj graf o zadanym typie, liczbie wierzchołków `v` i liczbie krawędzi `e`,
3. Usuń z grafu losowo wybraną krawędź,
4. Sprawdź, czy nastąpiło rozspójnienie grafu.
5. Oblicz iloczyn: $rozspójnień/ataków$

Generowanie grafów losowych

Generowanie grafu losowego będzie podzielone na dwa etapy.

Pierwszym etapem będzie przyjęcie zadanej liczby wierzchołków oraz zadanej gęstości grafu i obliczenie z nich docelowej liczby krawędzi $q_target$dla grafu wyjściowego. Drugi etap polegał będzie na wygenerowaniu grafu o n wierzchołkach połączonych losowo q_target krawędziami.

Algorytm ten przyjmuje dwa argumenty: liczbę wierzchołków n oraz współczynnik prawdopodobieństwa wystąpienia krawędzi n.

Grafy euklidesowe generowane będą poprzez weryfikację, czy dany losowo wygenerowany graf posiada własności grafu euklidesowego. Wygenerowane grafy nie spełniające tego warunku będą odrzucane.

Złożoność obliczeniowa algorytmu:

$(n * (n - 1))/2$

Oczekiwana liczba krawędzi:

$(n * (n - 1) * p)/2$

Spodziewany średni stopień wierzchołka:

$(n-1) * p$

Weryfikacja spójności grafu

Pseudokod algorytmu DFS [2], który zostanie wykorzystany do badania spójności grafów, przedstawiony został poniżej :

1. Utwóż tablicę `visited` o `n` elementach,
2. Tablicę `visited` wypełnij wartościami `false`,
3. Utwórz pusty stos `S`,
4. Inicjuj licznik odwiedzonych wierzchołków,
5. Rozpocznij przejście DFS od wierzchołka 0,
6. Wierzchołek oznacz jako odwiedzony,
7. Przechodź przez graf dopóki stos `S` nie jest pusty, wykonując następujące kroki:

- Pobierz wierzchołek ze stosu,
- Pobrany wierzchołek usuń ze stosu,
- Zwiększ licznik odwiedzonych wierzchołków,
- Przejrzyj kolejnych sąsiadów,
  - Szukaj do sąsiadów jeszcze nie odwiedzonych,
  - Odznacz sąsiada jeśli jeszcze nie odwiedzony,
  - Umieść sąsiada na stosie. Jeśli wszystkie wierzchołki zostały odwiedzone, graf jest spójny. W przeciwnym wypadku, graf jest niespójny.

Złożoność czasowa algorytmu wynosi $O(E+V)$

Schemat testów

Testy zostaną przeprowadzone w następujący sposób:

1. Wygenerowany zostanie zestaw grafów testowych o podanych wcześniej parametrach,
2. Z każdego z grafów zostanie usunięta losowo wybrana krawędź,
3. Spójność grafu zostanie sprawdzona i zapisana,
4. Obliczone zostanie prawdopodobieństwo rozspójnienia grafu P jako iloraz: $rozspójnień/ataków$

Uwagi sprawdzającego

> Nie bardzo rozumiem “Główny algorytm programu”. Proszę sprecyzować w spr.2