Algorytm
Artificial Benchmark for Community Detection

Bogumił Kamiński, Tomasz Olczak, Paweł Prałat, François Théberge

Szkoła Główna Handlowa w Warszawie, Ryerson University, Tutte Institute for Mathematics and Computing

25 października 2021

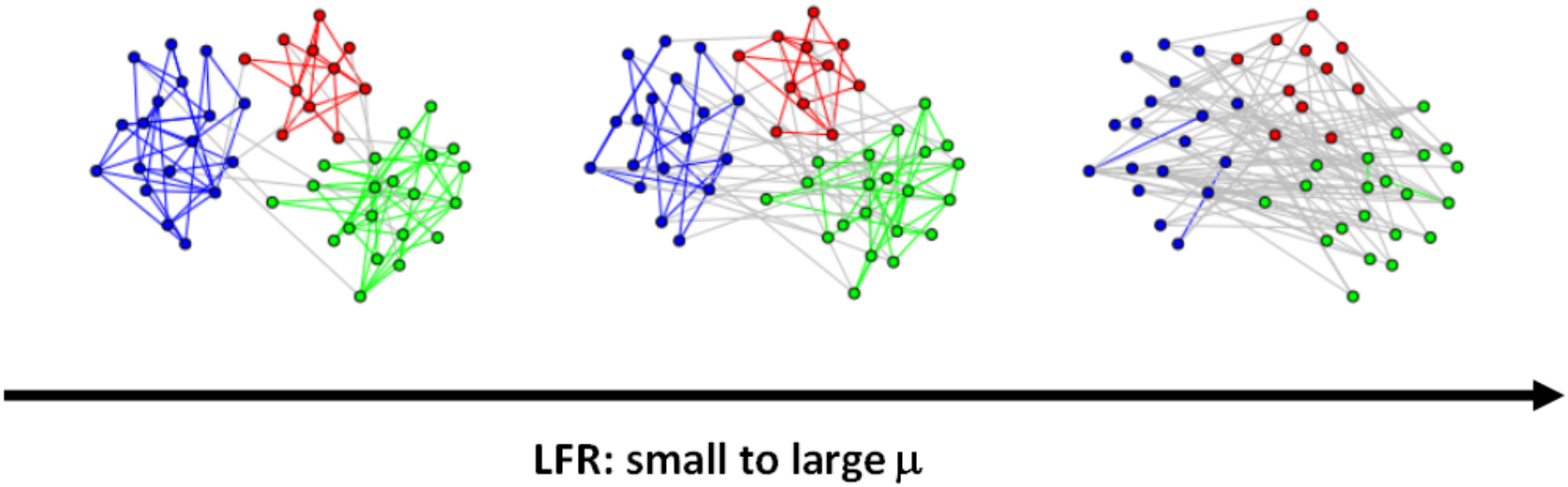
Cel pracy

1. Badania dotyczące efektów sieciowych stają mają coraz większe znaczenie teoretyczne i praktyczne.
2. Zastosowania: coraz więcej rynków, ze względu na rozwój Internetu, cechuje się występowaniem silnych efektów sieciowych; na takich rynkach uwzględnienie tych efektów jest istotne np. w celu prognozowania sprzedaży lub wykrywania nadużyć.
3. Teoria: modele objaśniające zachowanie podmiotów na takich rynkach.
4. Pojawia się potrzeba wykorzystania generatorów syntetycznych sieci społecznych.
5. W tym badaniu zajmujemy się generatorami sieci, które mają następujące oczekiwane cechy:
   - odtwarzają zadany rozkład stopni wierzchołków.
   - charakteryzują się występowaniem społeczności.

Standardowe podejście do generowania grafów ze społecznościami

1. Obecnie najpopularniejszy algorytm, który jest wykorzystywany w takich sytuacjach to podejście Lancichinetti–Fortunato–Radicchi (2008), obecnie praca ma prawie 3000 cytowań (algorytm LFR)
2. Podstawowe kroki procedury:
   - Wygenerowanie grafu z zadanym rozkładem stopni wierzchołków;
   - Wygenerowanie rozmiarów społeczności zgodnych z zadanym rozkładem;
   - Losowe przypisanie wierzchokłów do społeczności;
   - Losowe *przepinanie* krawędzi w grafie tak aby społeczności miały dostateczną gęstość połączeń wewnętrznych.

Przykładowe grafy LFR



Rysunek: Parametr μ steruje *odsetkiem* więzi wewnątrz społeczności dla pojedynczego węzła sieci.

Wyzwania związane z wykorzystaniem algorytmu LFR

1. Wolne działanie dla bardzo dużych sieci.
2. Algorytm może generować anty-społeczności co zwykle jest niepożądane.
3. Założenie o stałej wartości parametru μ dla wszystkich wierzchołków prowadzi do nieintuicyjnych wyników dla społeczności o znacząco różnych rozmiarach.
4. Ze względu na swoją konstrukcję algorytm jest trudny do analizy teoretycznej.

W tej prezencacji przedstawię algorytm Artificial Benchmark for Community Detection (ABCD) który został opracowany w celu rozwiązania tych problemów.

Parametry wejściowe algorytmu **ABCD**

1. Liczba wierzchołków, n
2. Rozkład stopni wierzchołków $\mathbf{w} = (w_1, \ldots, w_n)$.
3. Liczba społeczności, k, i rozkład ich rozmiarów $\mathbf{s} = (s_1, \ldots, s_k)$ p.w. $\sum_{i=1}^{k} s_i = n$.
4. Parametr homogeniczności społeczności ξ (gdy $\xi = 0$ społeczności są rozłączne; gdy $\xi = 1$ społeczności nie mają wpływu na rozkład krawędzi).

Sposób generacji grafu **ABCD**

Pomysł polega na tym, że zamiast generować jeden graf będziemy generowali $k+1$ niezależnych grafów G_i ($i \in \{0, 1, \ldots, k\}$).

Z tego jeden graf G_0 modeluje połączenia między społecznościami, a grafy G_i, $i > 0$ połączenia wewnątrz społeczności.

Wynikowy graf jest sumą $k+1$ grafów G_i.

Podział stopni pomiędzy graf G_0 i pozostałe grafy

Parametr $\xi \in [0,1]$ kontroluje ile krawędzi trafia do grafu G_0.

Dzielimy wagi $\mathbf{w}$ na część odpowiedzialną za graf społeczności $\mathbf{y}$ oraz za graf G_0: $\mathbf{z}$:

$$\begin{aligned} \mathbf{y} = (y_1, \ldots, y_n) &= (1-\xi)\mathbf{w} = ((1-\xi)\cdot w_1, \ldots, (1-\xi)\cdot w_n), \\ \mathbf{z} = (z_1, \ldots, z_n) &= \xi\mathbf{w} = (\xi \cdot w_1, \ldots, \xi \cdot w_n). \end{aligned}$$

Przypisanie wierzchołków do społeczności

Problem: *ciężki* wierzchołek nie może trafić do *małej* społeczności.

Algorytm gwarantujący jednostajnie losowe i dopuszczalne przypisanie wierzchołków do społeczności:

1. Posortuj wierzchołki malejąco względem ich stopnia wewnątrz spłeczności: $\mathbf{y} = (y_1, \ldots, y_n)$.
2. Rozpatruj wierzchołki w kolejności od tego o największym stopniu; przypisuj wierzchołek do społeczności wybranego spośród społeczności które jednocześnie: (a) mają dostatecznie duży rozmiar, (b) nie są jeszcze w pełni obsadzone.
3. Prawdopodobieństwo przypisania wierzchołka społeczności powinno być proporcjonalne to liczby *wolnych miejsc*, które jescze pozostają w społeczności.

Wyniki testów wydajnościwych

Porównywane algorytmy:

1. LFR oryginalny, C++
2. LFR NetworKit (szybszy, ale posiadający niezgodną implementację), C++
3. ABCD, Julia

Wyniki:

1. Algorytm ABCD jest około 100x szybszy od oryginalnego LFR i około 10x sybszy od LFR NetworKit
2. Algorytm stosunkowo łatwo można zrównoleglić

Podsumowanie

1. Zaproponowany algorytm ABCD jest zarówno szybszy jak i ma lepsze własności niż dotychczas standardowo stosowany algortm LFR.
2. Obecne zastosowania algorytmu:
   - 2.1 modele wieloagentowe rynków uwzględniające efekty sieciowe i występowanie społeczności użytkowników.
   - 2.2 testowanie algorytmów mających za zadanie wykrywanie społeczności w grafach.
   - 2.3 testowanie algorytmów tworzących zanurzenia gratów w przestrzeniach R^n (np. na potrzeby prognozowania połączeń w sieci lub wykrywania anomalii).
3. Dasze badania:
   - 3.1 generowanie społeczności, które nie są rozłączne.
   - 3.2 uwzględnienie dotatkowych oczekiwań odnośnie cech generowanego grafu (np. współczynnik klastrowania, selektywność połączeń).
   - 3.3 generowanie hipergrafów.