Grafy i sieci - wykrywanie społeczności

Ireneusz Stanicki, Mateusz Śliwakowski,
Bartłomiej Truszkowski, Przemysław Woźniakowski

08/04/2020

Spis treści

1 Wstęp

1.1 Przedstawienie problemu

Badanie sieci społecznościowych jest podstawową dziedziną nauki o sieciach. Problem ten znacząco zyskał na znaczeniu, odkąd możliwy jest dostęp do olbrzymich zbiorów danych, dzięki działaniom takich firm jak Facebook, Google, czy Twitter. Szczególnie istotnym zagadnieniem, zarówno dla środowiska biznesowego jak i akademickiego, jest wyszukiwanie społeczności w grafach społecznościowych. Wykorzystuje się je w takich dziedzinach jak kryminalistyka, opieka zdrowotna, polityka, czy marketing[**ms-paper1**].

Grafem społecznościowym nazywamy taki graf, który reprezentuje relacje między jednostkami. Najczęściej spotykanym przykładem jest struktura, gdzie wierzchołki identyfikują osoby, a krawędzie - relacje między danymi osobami. Tą relacją może być znajomość, lecz nic nie stoi na przeszkodzie aby definiować ją dowolnie - np. pytaniem 'Czy dane osoby wymieniły ze sobą wiadomość?'.

Podstawowa idea wykrywania społeczności opiera się na znajdowaniu grup wierzchołków, dla których liczba połączeń w obrębie społeczności jest znacząco wyższa, niż liczba połączeń do wierzchołków spoza tej społeczności. Oczywiście definicja ta jest dosyć płynna - w tym problemie często algorytmy opiera się na pewnych heurystykach, które sprawdza się na rzeczywistych zbiorach danych. Weryfikacja takich rozwiązań nie jest prosta - często przeprowadzana jest ona manualnie, przez analizę wizualizacji.

1.2 Typowe podejścia

Problem znajdowania społeczności, ze względu na swoją popularność i mnogość praktycznych zastosowań, doczekał się wielu podejść teoretycznych.

Pierwszym, dość intuicyjnym podejściem jest wyszukiwanie społeczności skupione na wierzchołkach (node centric community detection). Zakłada ono znajdowanie grup wierzchołków z którym każdy spełnia pewne kryterium, jakim może być np: odpowiedni stopień, osiągalność (czyli odległość od innego, wskazanego wierzchołka) czy wzajemność (mutuality). Przy znajdowaniu grup opartych na wzajemności, kluczem jest znajdowanie klik (dla grafów nieskierowanych) lub pełnych grafów dwudzielnych (dla grafów skierowanych). Niestety, znalezienie tych struktur jest bardzo kosztowne czasowo i dla większych grafów okazuje się niepraktyczne (dla kliki mamy doczynienia z problemem NP-zupełnym).

Kolejnym sposobem jest rozpatrywanie grup opartych na osiągalności. W tym podejściu rozważa się kilka, podstawowych grup:

- $k-klika$ - maksymalny podgraf w którym odległość między dwoma dowolnymi wierzchołkami (w oryginalnym grafie, więc ścieżki mogą przechodzić przez wierzchołki spoza $k-kliki$) jest nie większa niż k, czyli: $d(i,j) \leq k \forall v_i, v_j$

- $k-klan$ - $k-klika$ dla której odległość między dwoma dowolnymi wierzchołkami w podgrafie (to jest, biorąc pod uwagę tylko krawędzie pomiędzy wierzchołkami z $k-klanu$) jest nie większa niż k. Każdy k-klan jest k-kliką.

- $k-klub$ - maksymalny podgraf, w których odległość między dwoma wierzchołkami w grupie (nie uwzględniając ścieżek przechodzących po wierzchołkach spoza grupy) nie przekracza k. Każdy k-klan jest k-klubem.

W przypadku grup opierających się na stopniach wierzchołków kluczowe są następujące struktury:

- $k-pleks$ - to minimalny podgraf zawierający n_s wierzchołków, z których każdy sąsiaduje z conajmniej n_s-k wierzchołkami z tego podgrafu. Innymi słowy każdy element ma co najwyżej k wierzchołków, z którymi nie jest połączony.

- $k-rdzenie$ - to struktury, w której każdy wierzchołek jest połączony z conajmniej k wierzchołkami ze struktury, czyli $d_s(i) \geq k \forall v_i \in V_s$

Definicje $k-pleksu$ i $k-rdzenia$ uzupełniają się. Te struktury są zwykle odporne na usuwanie krawędzi. Wiemy, że stopnie wierzchołków w rzeczywistych sieciach rozkładają się według prawa potęgowego, tzn. w grafie jest mało wierzch o dużych stopniach i wiele wierzchołków o mniejszych stopniach. Niestety grupy oparte na stopniach wierzchołków wymagają wielu wierzchołków o dużych stopniach, więc słabo nadają się one do analizy rzeczywistych sieci.

Odchodząc od rozważania pojedynczych wierzchołków, kolejne podejście skupia się na ich całych grupach. W takim podejściu to nie każdy wierzchołek, musi spełniać pewne założenia, ale całe zbiory wierzchołków. Przykładem takich grupy mogą być grupy oparte na gęstości. Podgraf $G_s(V_s, E_s)$ jest γ gęsty (zwany również prawie-kliką) gdy $\frac{E_s}{V_S(V_s-1)/2} >= \gamma$. Wewnątrz takiego podgrafu, stopnie wierzchołków mogą znacznie się różnić, dlatego stosowane są one do dużych sieci.

Rozszerzając dalej obszar rozważań, dochodzimy do metod opartych na całych sieciach. Rozważają one nie pojedyncze grupy, a sieci jako jedną całość. Przykładem mogą być:

- grupy oparte na podobieństwie wierzchołków - biorą one pod uwagę np podobieństwo zbioru sąsiadów (podobieństwo strukturalne). Jest to jednak dosyć ograniczająca definicja, stosuje się więc też np. podobieństwo automorficzne, w którym dwa wierzchołki są podobne, gdy ich sąsiedzi

mogą być przemianowani tak by utworzyć izomorficzny graf. Wykrywanie podobieństw w dużych, praktycznych sieciach jest jednak dość trudne i bardzo skomplikowane obliczeniowo.

- grupy oparte na minimalnym przekroju - społeczności definiuje się jako podzbiór $C \subset V$, taki że $\forall v \in C, v$ ma co najmniej tak dużo krawędzi do wierzchołków z C jak do wierzchołków z $V \backslash C$.

- grupy oparte na modularności - modularność jest miarą, która bierze pod uwagę rozkład stopni w grafie dostosowując strukturę społeczności. Mierzy ona jak bardzo sieć różni się od równomiernie losowego grafu (*nullmodel*). Definiuje się ją następująco: $Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{d_i d_j}{2m}] \delta(s_i, s_j)$, gdzie d_i to stopień wierzchołka v_i, a s_i to społeczność do której należy wierzchołek v_i (podobnie d_j i s_j dla v_j), δ to delta Kroneckera (wartość 1 gdy v_i i v_j są w tej samej społeczność 0 wpw.). Generalnie celem jest dostosowanie struktury społeczności tak by zmaksymalizować modularność.

Ostatnim wartym wspomnienia podejściem jest wykrywanie społeczności oparte na hierarchiach. Wyróżnia się trzy typy podziału hierarchicznego:

- rozdzielające (divisive) hierarchiczne skupianie (clustering) - graf jest dzielony np. w oparciu o *edge betweeness*, które jest miarą tego jak wiele najkrótszych ścieżek pomiędzy dwoma wierzchołkami prowadzi przez daną krawędź. Pozwala ona wykrywać krawędzie, będące "mostami"między społecznościami, które w celu znalezienia podziału na społeczności są sukcesywnie usuwane.

- aglomeracyjne (agglomerative) hierarchiczne skupianie - opiera się na umieszczeniu wszystkich wierzchołków w oddzielnych społecznościach, które są następnie łączone, tak by zmaksymalizować modularność powstałego podziału. Ważne jest, by w tym podejściu łączyć przede wszystkim społeczności o podobnych rozmiarach, tak by uzyskać zbalansowany podział.

- wyszukiwanie struktur - bazuje na wyszukiwaniu hierarchii, które mają duże prawdopodobieństwo by utworzyć sieć.

Wszystkie te podejścia są jednak ściśle matematycznymi zagadnieniami. Używane w nich miary i struktury, często do obliczenia bądź znalezienia wymagają dużej mocy obliczeniowej[**pw-paper1**]. Dlatego w praktycznych zastosowaniach i algorytmach często stosuje się pewne uproszczenia, modyfikuje się te podejścia, a nawet łączy je by uzyskać najlepsze efekty, zarówno pod kątem czasu obliczeń, jak i jakości otrzymanych społeczności.

2 Algorytm Girvan-Newmana

2.1 Opis algorytmu

Metoda Girvan-Newmana jest metodą dzielącą - rozważanie problemu rozpoczyna się od całej sieci i z każdym krokiem zmniejsza ją poprzez usuwanie krawędzi. Każda z nich posiada specjalny parametr, który w literaturze funkcjonuje pod nazwą *edge betweenness* [**is-paper1**]. Określa on sumę najkrótszych ścieżek przechodzących przez daną krawędź. W przypadku tylko jednej takiej ścieżki parametr zwiększany jest o 1, natomiast gdy między wierzchołkami występuję więcej niż jedna najkrótsza ścieżka, to każdej z tych ścieżek przypisywana jest równa waga, tak by suma wag dla wszystkich była równa 1.

Cecha ta zwraca od razu uwagę na fakt, który może stanowić główną wadę algorytmu - jego złożoność jest dość duża i może on mieć znaczne problemy wydajnościowe przy większych sieciach. Co iterację, po usunięciu krawędzi, konieczne jest ponowne wyliczenie wartości parametru dla każdej z krawędzi. To prowadzi do złożoności czasowej na poziomie $O(m^2n)$, gdzie m stanowi liczbę krawędzi, a n liczbę wierzchołków. Usprawnienia algorytmu można dokonać poprzez wyrzucenie w jednym kroku wszystkich krawędzi o najwyższej wadze. Co prawda nie zmieni to złożoności pesymistycznej algorytmu, jednak w większości przypadków znacznie go usprawni. Warto nadmienić, iż twórcy algorytmu nie sprecyzowali kolejności usuwania krawędzi o tej samej wadze, pozostawiając ten aspekt do rozwiązania przy implementacji.

Algorytm kończy swe działanie, gdy usunięta zostanie ostatnia krawędź. W czasie jego działania stworzony zostanie dendrogram, na podstawie którego będzie można odbudować podział społeczności na danym poziomie. Znalezienie optymalnej dekompozycji grafu stanowi w tym wypadku kluczowy problem. W tym przypadku analizowana metoda ukazuje swoje zalety. Co iterację można tutaj wyliczyć współczynnik modularności i na samym końcu określić poziom, dla którego był on największy, a następnie odtworzyć dany stan sieci. W ten sposób metoda Girvan-Newmana jest w stanie wskazać nam najlepszy podział sieci na społeczności.

Motywację przy wyborze tego algorytmu stanowił fakt, iż jest to najbardziej znana metoda związana z wykrywaniem społeczności, o której wzmianki można odnaleźć w niemal każdej publikacji naukowej z tej dziedziny. Zagłębiając się w tematykę warto rozpocząć od niego swoją pracę.

2.2 Modyfikacje dla społeczności nachodzących

2.2.1 CONGA

W tradycyjnym wykorzystaniu algorytmu Girvan-Newmana każda kolejna iteracja prowadzi do podziału sieci na mniejsze części, które razem zawierają

wszystkie wierzchołki grafu. Aby umożliwić wykrywanie społeczności nakładających się, musimy odnaleźć moment w którym dojdzie do podziału wierzchołka, tak aby znalazł się on w dwóch oddzielnych społecznościach. W algorytmie CONGA problem ten został rozwiązany poprzez wprowadzenie dodatkowego parametru dla każdego wierzchołka, w literaturze nazwanego *vertex betweenness* [**is-paper2**].

Analogia do nazwy *edge betweenness* wykorzystywanej w wersji podstawowej algorytmu nie jest przypadkowa. Parametr dla wierzchołka mówi nam bowiem o tym, ile najkrótszych ścieżek przechodzi przez ten wierzchołek. Dodatkowo dla każdego wierzchołka v wartość tą można obliczyć bazując na wyliczonych wcześniej parametrów każdej z krawędzi zawierającej v. Zważywszy na złożoność czasową całego algorytmu modyfikacja ta nie jest znacząco kosztowna.

Z podejściem CONGA związane są trzy główne pytania na temat podziału wierzchołka:

1. Kiedy wykonać je zamiast zwykłego usunięcia krawędzi?
2. Który wierzchołek podzielić?
3. Jak go podzielić?

O ile na 2 pytanie odpowiedź stanowi parametr *vertex betweenness*, tak pozostałe wymagają przyjęcia pewnych założeń. Określenie momentu podziału wierzchołka zamiast usunięcia krawędzi również wydaje się proste - klonowanie następuje wtedy, gdy największa wartość *vertex betweenness* przekroczy największą wartość *edge betweenness*. Odpowiedź na ostatnie pytanie stanowi już jednak o wiele poważniejszy problem i wymaga wprowadzenia dodatkowego parametru.

Pair betweeness wierzchołka v dla (u, w), gdzie u i w są sąsiadami v i $u \neq w$ jest to liczba najkrótszych ścieżek, które przechodzą zarówno przez krawędź (u, v) jak i (v, w). Obliczenie tego parametru dla każdej pary można przeprowadzić razem z wyliczaniem zwykłego *edge betweenness*. Aby móc wykonać podział konieczne jest stworzenie dodatkowej konstrukcji opartej na ww. parametrze. Konstrukcja ta jest grafem pełnym o liczbie wierzchołków równej liczbie sąsiadów v, dla którego każdy wierzchołek jest numerowany indeksem sąsiada v, a krawędzie (u, w) są oznaczane wartością parametru *pair betweenness* dla tej pary osiągniętym przy analizie z wierzchołka v.

Podział wierzchołka na podstawie parametru *pair betweeness* jest zrealizowany metodą zachłanną. Warto zwrócić uwagę, iż modyfikacja ta wymagać będzie dodatkowej pamięci, a także zwiększy czas wykonania. Jest to też zdecydowanie najbardziej rozbudowany krok algorytmu CONGA.

2.2.2 CONGO

Optymalizacja algorytmu CONGA zrealizowana przez jego twórców [**is-paper3**]. Skupia się ona na analizie lokalnej problemu, odrzucając wszystkie ścieżki o długości większej niż parametr h (zazwyczaj dość niska wartość). Następnie problem aktualizacji wartości *edge betweenness* odbywa się również wyłącznie lokalnie, przez co czas wykonania algorytmu znacznie spada. W naszych działaniach pominiemy jednak tę metodę, gdyż podając ją analizie uznaliśmy, iż trudno znaleźć w niej miejsce do skutecznej modyfikacji.

2.3 Proponowane modyfikacje

Usprawnienia możliwe do zrealizowania oparte zostały o algorytm CONGA. Bazując na parametrach *edge* i *vertex betweenness* do głębszej analizy został wybrany krok sposobu podziału wierzchołka. Algorytm podstawowy bazuje na heurystyce i dodatkowym wykorzystaniu pamięci, a także dość skomplikowanym modelu obliczeniowym. W naszej modyfikacji chcielibyśmy wykorzystać dodatkowe własności grafu, które dla sieci społecznościowych stanowią cechy nieodłącznie z nimi kojarzone. Mowa tutaj o liczbie wspólnych znajomych, a także dacie zawarcia znajomości. Moduł podziału wierzchołka można wymienić na inny, korzystający z ww. cech. Dzięki temu w dość prosty sposób można zmodyfikować algorytm i skorzystać z dwóch sposobów podziału wierzchołka. Dzieląc wierzchołek przy pomocy liczby wspólnych znajomych, celem będzie stworzenie takiego podziału, który zmaksymalizuje liczbę wspólnych znajomych w obu zbiorach dla wierzchołka v i jego kopii. Bazując na dacie zawarcia znajomości można kierować się zasadą chronologii - daty mogą mieć odzwierciedlenie w danej sytuacji życiowej, a także grupie, w której w danym momencie się znajdujemy.

2.4 Oczekiwania

Korzystając z usprawnień opisanych w poprzedniej sekcji naszym głównym celem będzie poprawa parametru modularności. Każda dodatkowa wiadomość dotycząca krawędzi powinna prowadzić do lepszego podziału danej społeczności. Co za tym idzie algorytm szybciej powinien podzielić sieć na mniejsze części, przez co wykona mniej obliczeń w obrębie pełnego grafu, co przyspieszy jego działanie. Sam krok podziału wierzchołka również powinien zostać przyspieszony, a wykorzystana do tego pamięć znacznie zmniejszona i sprowadzona do informacji uzyskanych w fazie preprocessingu.

3 Label Propagation Algorithm

3.1 Opis algorytmu

Label Propagation Algorithm to algorytm stosunkowo prosty i szybki. Wybraliśmy go ze względu na jego interesujące własności oraz fakt, że został zaimplementowany w bilbliotece neo4j. Dokładnie został opisany w pracy 'Near linear time algorithm to detect community structures in large-scale networks'[**ms-paper2**]. Opiera się on na idei, że etykieta staje się dominująca w gęsto połączonych grupach, zaś nie jest propagowana przez rzadko połączone rejony. Ogólny schemat wygląda następująco:

1. Zainicjalizuj wierzchołki unikatowymi etykietami.
2. Dla każdego wierzchołka przypisz mu etykietę, która występuje najwięcej razy wśród jego sąsiadów. W razie remisu, wybierz losowo.
3. Powtarzaj krok 2, aż nie wystąpi żadna zmiana etykiety.

3.2 Modyfikacje dla społeczności nachodzących

3.2.1 COPRA

Aby umożliwić wykrywanie społeczności nachodzących wprowadzono parametr v - maksymalną ilość społeczności, do której może należeć wierzchołek[**ms-paper3**]. W każdym kroku algorytmu wierzchołkowi przypisujemy v najczęściej występujących wśród jego sąsiadów etykiet.

3.2.2 BMLPA

Zaproponowano modyfikację algorytmu COPRA, aby jego parametryzacja była bardziej uniwersalna[**ms-paper4**]. Wierzchołkowi przypisujemy tylko te etykiety, dla których iloraz $\frac{b}{b_{max}}$ jest większy bądź równy od nowo wprowadzonego parametru p, gdzie b to liczność danej etykiety wśród sąsiadów, b_{max} to liczność najczęściej występującej etykiety wśród sąsiadów. Algorytm ten wymaga wyznaczenia początkowych społeczności.

3.3 Proponowane modyfikacje

3.3.1 Zrównoleglenie

Algorytm LPA jest algorytmem o złożoności liniowej, dlatego nie ma tutaj zbyt dużego pola na optymalizacje wydajnościowe. Interesującym doświadczeniem, może być jednak zaimplementowanie algorytmu oraz jego modyfikacji na procesory graficzne (np. używając biblioteki CUDA). Doświadczenia takie były już przeprowadzane ([**ms-paper5**], [**ms-paper6**]), jednak w naszej pracy zyskalibyśmy porównanie nie tylko między CPU i GPU, lecz również między innymi metodami.

3.3.2 Użycie roli

Znaczącą poprawę wyników LPA można uzyskać wybierając społeczności początkowe, zamiast inicjalizować wszystkie wierzchołki unikatowymi społecznościami. Jednym z pomysłów jest użycie idei ról w grafie do wyboru początkowych społeczności[**ms-paper7**]. Chcemy zastosować tę ideę do inicjalizacji algorytmu BMLPA.

3.3.3 Użycie parametrów krawędzi

Dla rozpatrywanych grafów możemy do krawędzi przypisać interesujące wartości, np. datę zawarcia znajomości, czy liczbę wspólnych znajomych. Chcemy użyć tych parametrów jako wag przy wyborze etykiet w kroku algorytmu. Przykłady:

- W przypadku remisu wybieramy tę etykietę, dla której znajomości były zawierane w podobnym czasie. Matematycznie problem ten można sprowadzić do wyboru etykiety z najmniejszą wariancją dla parametrów jej krawędzi.
- Dodajemy wagę do wierzchołka - liczbę wspólnych znajomych dzieloną przez maksymalną wspólną liczbę znajomych z sąsiadem, a następnie używamy jej razem z wagami, zgodnie z ideą algorytmu BMLPA.

3.4 Oczekiwania

Jeśli chodzi o równoległą implementację, oczekujemy, że będzie ona znacząco szybsza od jakiejkolwiek innej metody w naszej pracy. W przypadku modyfikacji z punktów 3.3.2 i 3.3.2, chcemy uzyskać rezultaty bardziej wiarygodne, pozbawione gigantycznych społeczności, bez nienaturalnych rozkładów.

4 Overlapping Community Detection by Local Community Expansion

4.1 Opis algorytmu

Algorytm ten wykorzystuje podejście znalezienia wielu małych lokalnych społeczności, a następnie połączeniu ich w większe, nakładające się. Składa się on z trzech kroków:

4.1.1 Wyznaczenie lokalnych społeczności

- Zainicjuj lokalne społeczności zbiorem pustym
- Dla każdej pary wierzchołków (u, v), że oba nie należą jeszcze do tej samej społeczności wykonuj:
  - Zainicjuj tymczasową społeczność *tmp* jako $\{u, v\}$
  - Rozpatrz każdego wspólnego sąsiada u i v, jeżeli modularność społeczności wraz z nim jest większa od modularności bez niego, dodaj go do społeczności
  - Jeżeli w *tmp* znajduje się przynajmniej 5 elementów, dodaj *tmp* do zbioru lokalnych społeczności

Jest to jedna z metod wyznaczania lokalnych społeczności, zaprezentowana w [**bt-paper1**]. Najważniejszą rolę w niej odgrywa w niej modularność. Jest ona wyznaczana ze wzoru:

$M = \frac{M_{in}}{M_{out}} = \frac{\frac{1}{2}\sum_{ij} A_{ij}\theta(i,j)}{\sum_{i,j} A_{ij}\lambda(i,j)}$,

gdzie A_{ij} jest macierzą sąsiedztwa grafu, θ i λ zaś funkcjami. $\theta(i, j)$ zwraca 1, jeśli oba wierzchołki i i j należą do rozpatrywanej społeczności, w przeciwnym wypadku zwraca 0. $\lambda(i, j)$ zwraca 1 jeśli tylko jeden z wierzchołków należy do społeczności, w przeciwnym wypadku 0. Można więc powiedzieć, że θ odpowiada bitowej operacji AND, zaś λ bitowej operacji XOR.

4.1.2 Łączenie społeczności

- Zainicjuj zbiór wyjściowych społeczności C' zbiorem lokalnych społeczności
- Dla każdej społeczności C z C' wykonuj:
  - Znajdź zbiór społeczności połączonych z C
  - Przyłącz ten zbiór do C
  - Usuń z C' wszystkie sieci należące do tego zbioru
  - Uaktualnij licznik społeczności we wszystkich wierzchołkach z C'

W wyniku wyżej opisanego algorytmu można skonstruować społeczności nakładające się na siebie, jako że wiele lokalnych społeczności może zawierać te same wierzchołki. Należy jednak zdefiniować, kiedy dwie społeczności są ze sobą połączone. W [**bt-paper1**] proponowana jest metoda, polegająca na liczeniu ważonego wyniku nakładania się WOS, danego wzorem:
$WOS(C_i, C_j) = \alpha \frac{|V_i \cap V_j|}{min(|V_i|,|V_j|)} + (1-\alpha)\frac{|E_i^{in} \cap E_j^{out}|}{min(E_i^{in}, E_j^{out})}$,
gdzie α jest ustalonym parametrem między 0, a 1, V określa zbiór wierzchołków danej społeczności, a E^{in} i E^{out} to zbiory krawędzi odpowiednio wewnętrznych (o obu końcach w społeczności) i zewnętrznych (o tylko jednym końcu wewnątrz). Dane społeczności C_i i C_j sąsiadują ze sobą jeśli $WOS(i,j)$ jest większy od ustalonej na początku stałej β.

4.1.3 Przyłączanie wierzchołków izolowanych

- Niech U będzie zbiorem takich wierzchołków, które aktualnie nie należą do żadnej społeczności. Dla każdego $u \in U$, wykonaj:
  - Rozpatrz wszystkich sąsiadów u i społeczności, do których należą. Dla każdej takiej społeczności oblicz modularność wraz z u oraz bez niego. Jeżeli modularność tej społeczności wraz z u jest większa, dodaj u do tej społeczności.

Jako, że w pierwszym kroku odrzuca się wszystkie społeczności mające mniej niż 5 wierzchołków, istnieje istotne prawdopodobieństwo, że w grafie pozostanie wiele wierzchołków izolowanych. Krok ten ma na celu zmniejszyć tę liczbę, jednak warto zauważyć, że mimo wykonania go, nie wszystkie wierzchołki muszą należeć do jakiejś społeczności. Modularność jest wyznaczania jak w kroku pierwszym.

4.1.4 Złożoność obliczeniowa

1. Krok pierwszy ma złożoność $O(m \cdot d)$, gdzie m jest liczbą krawędzi w grafie, a d średnim stopniem wierzchołków. Jest tak ponieważ początkowych społeczności będzie rzędu m, a wspólnych sąsiadów rzędu d.

2. Krok drugi ma złożoność $O(k_1^2)$, gdzie k_1 to liczba społeczności znalezionych w pierwszym kroku. Oczywiście $k_1 < m$, ale można oczekiwać, że zazwyczaj będzie $k_1 << m$.

3. Krok trzeci ma złożoność $O(n \cdot d)$, ponieważ rozważanych jest najwyżej n wierzchołków (zazwyczaj istotnie mniej) i każdy średnio ma stopień rzędu d.

Z powyższych rozważań wynika, że przy założeniu $n \approx m$, złożoność pesymistyczna jest kwadratowa, ale zazwyczaj będzie bliska liniowej.

4.2 Proponowane modyfikacje

4.2.1 Użycie innej miary niż modularność

Wierzchołki do lokalnych społeczności można dodawać za pomocą innych miar niż modularność M. Może być to na przykład funkcja f z [**bt-paper2**].

4.2.2 Inny sposób wybierania lokalnych społeczności

W dotychczasowej wersji algorytmu za początek społeczności brane są pary sąsiadujących wierzchołków, niebędących jeszcze w jednej społeczności. Zamiast tego można rozpatrzeć połączone ze sobą trójkąty, bądź większe kliki, których wszystkie wierzchołki nie są jeszcze w jednej społeczności. To rozwiązanie jest wolniejsze z powodu konieczności znalezienia wszystkich takich klik, ale istnieje szansa, że tak zbudowane społeczności będą lepiej dobrane.

4.3 Oczekiwania

Od metody oczekujemy tego, że po jej implementacji uda się uzyskać wysokiej jakości podział na społeczności. Od modyfikacji oczekujemy, że wprowadzenie ich zmieni w istotnym stopniu uzyskany wcześniej rezultat. Zastosowanie innej miary powinno być porównywalne co do wydajności w zestawieniu z podstawową wersją algorytmu, ale znalezione społeczności powinny się zauważalnie różnić. Inny sposób wybierania lokalnych społeczności zapewne będzie zaś sporo mniej wydajny, ale powinien skutkować zdecydowanie lepszej jakości rezultatem, to znaczy ściślejszymi powiązaniami wśród społeczności.

5 Algorytm Louvain

5.1 Opis algorytmu

Algorytm ten został zaproponowany przez profesora Vincenta Blondela z Uniwersytetu w Louvain. Jest to tzw. algorytm zachłanny (greedy algorithm), który dąży w każdym kroku do zmaksymalizowania modularności. Modularność, jako miara podziału sieci na społeczności już pojawiła się w tej pracy (**??**), ale przypomnijmy: $Q = \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{d_i d_j}{2m}]\delta(s_i, s_j)$. W tym algorytmie jednak, o wiele ważniejsza od wartości modularności jest jej różnica w każdym kroku, bo to na jej podstawie będziemy przenosić wierzchołki pomiędzy społecznościami:

$$\Delta Q = [\frac{\Sigma_{in} + 2k_{i,in}}{2m} - (\frac{\Sigma_{tot} + k_i}{2m})^2]$$

gdzie k_i to stopień wierzchołka i, k_{in} to suma wag krawędzi między wierzchołkiem i i wierzchołkami ze społeczności, do której trafia i, Σ_{in} to suma wag krawędzi wewnątrz tej społeczności, a Σ_{tot} to suma wag krawędzi wewnątrz społeczności z której usuwamy wierzchołek i.
Schemat algorytmu:

1. Każdy wierzchołek umieść w oddzielnej jednoelementowej społeczności.
2. Dla każdego wierzchołka przenieś go do sąsiadującej społeczności, tak by maksymalnie zwiększyć modularność (czyniąc to zachłannie). Jeśli nie da się tego wykonać, pozostaw go w obecnej. Jeśli w tym kroku nie nastąpi żadna zmiana, algorytm kończy działanie.
3. Każdą powstałą społeczność, scal do jednego wierzchołka. Krawędzie pomiędzy wierzchołkami wewnątrz społeczności zamieniają się w pętle o odpowiedniej wadze, a krawędzie z kilku wierzchołków ze społeczności do jednego spoza łączą się w jedną o zsumowanej wadze.
4. Powróć do drugiego kroku.

Wynikiem działania algorytmu jest graf, w którym każdy wierzchołek reprezentuje społeczność w oryginalnym grafie. Wszystkie węzły, które w czasie działania algorytmu zostały scalone do jednego wierzchołka, należą do jednej społeczności.

Wybraliśmy ten algorytm do naszych rozważań, ponieważ poza dość prostym schematem działania, cechuje się też dobrą szybkością obliczeń (złożoność $O(nlogn)$), oraz wysoką jakością otrzymanych wyników (w porównaniu do innych typowych rozwiązań). Z powodu tych własności jest jednym z algorytmów zaimplementowanych w bibliotece neo4j.

5.2 Modyfikacje dla społeczności nachodzących

Z racji, że algorytm Louvain jest algorytmem zachłannym, zależnie od kolejności iterowania po wierzchołkach, może on dawać różne wyniki pod względem

powstałych społeczności. Ten fakt zostanie wykorzystany do znajdowania społeczności nachodzących.

Modyfikacja dla społeczności nierozłącznych opiera się na wykonaniu n razy algorytmu dla społeczności rozłącznych, dla różnych permutacji wierzchołków. Na podstawie każdego otrzymanego zbioru społeczności konstruowana jest macierz $n \times m$ gdzie n - liczba wierzchołków w grafie, m - liczba społeczności, a element macierzy $A_{n,m}$ oznacza ilość krawędzi wychodzących z wierzchołka n do wierzchołków ze społeczności m (zakładamy, że wierzchołki są numerowane od 1 do n, społeczności od 1 do m). Następnie macierze dla każdego przebiegu algorytmu są sumowane, a wartości są następnie normalizowane. W tak powstałej macierzy, jeżeli wartości w n wierszu i m kolumnie przekracza wartość progową $\lambda = \frac{1}{ov+1}$ (gdzie ov jest wartością dobraną podczas implementacji algorytmu) wierzchołek n należy do społeczności m.

Następnie w ostatnim kroku, niektóre z powstałych społeczności mogą zostać połączone na podstawie modularności zaproponowanej przez Vincenzo Nicosie z Uniwersytetu w Catanii [**pw-paper2**] . Jest to jednak krok bardzo kosztowny ze względu na czas obliczeń (złożoność $O(n^3)$), a przynosi on stosunkowo niewielką poprawę ostatecznej jakości wyniku. Algorytm dokładniej wytłumaczony jest w artykule: Ą Fast Algorithm for Overlapping Community Detection"[**pw-paper3**].

5.3 Proponowane modyfikacje

W naszym projekcie skupimy się przede wszystkim na próbach optymalizacji ostatniego kroku, oraz porównaniu jego wpływu na ostateczny wynik. W celach optymalizacji możemy wykorzystać heurystyki oraz dodatkową wiedzę na temat sieci. Możemy na przykład, nie rozważać do scalenia społeczności, które przekroczą żądaną przez nas wielkość, lub ilość ich wspólnych wierzchołków nie jest wystarczająca. Ponadto modularność pozwoli w bardzo naturalny sposób wykorzystać dodatkowe informacje (np. w przypadku znajomych, datę zawarcia znajomości) jako wagę krawędzi. Powinno to dać ciekawe efekty, gdyż starzy znajomi powinni tworzyć rdzenie społeczności, i tak stworzone grupy mogą znacznie różnić się od społeczności otrzymanych bez tej modyfikacji.

5.4 Oczekiwania

Oczekujemy, że dodatkowe usprawnienia pozwolą zniwelować, albo chociaż zmniejszyć negatywny wpływ jaki na czas obliczeń ma ostatni etap. Sama metoda, w artykule w którym została przedstawiona, opisywana jest w samych superlatywach. Autorzy zapewniają o jej wyższości nad innymi typowymi algorytmami. Oczekujemy, że dzięki naszym eksperymentom uda nam się skonfrontować ją z innymi metodami i ewentualnie usprawnić ją.

6 Porównanie

6.1 Pod kątem szybkości działania

Aby uzyskać miarodajne rezultaty, planujemy każdy algorytm uruchamiać na tych samych zbiorach danych oraz tej samej maszynie. Dla problemu społeczności nachodzących oczekujemy, że najwolniejszy będzie Algorytm Girvana-Newmana, ze względu na swoją złożoność obliczeniową. Następnie Algorytm Louveina, przez bardzo kosztowny ostatni krok wyznaczania społeczności nachodzących. Najszybszy powinien okazać się algorytm LPA, ze względu na niemal liniową złożoność, zaś na drugim miejscu spodziewamy się algorytmu OCDLCE.

6.2 Pod kątem jakości rozwiązania

W tym wypadku skupimy się na analizie wizualizacji rozwiązania. Sprawdzimy, czy powstałe społeczności mają naturalne dla problemu rozłożenie. Zamierzamy również badać, czy w rozwiązaniu nie pojawiły się zbiory gigantyczne, zbyt obszerne przecięcia lub zbiory rozłączne. Będziemy także sprawdzali, czy otrzymana liczba społeczności jest zbliżona do wzorcowego wyniku.

7 Projekt z dziedziny zainteresowań osobistych

Naszą pracę nad projektem chcieliśmy wzbogacić o coś, czego zazwyczaj nie kojarzymy z nauką, a raczej z chwilą odpoczynku. Oczy skierowaliśmy w stronę NBA - najlepszej ligi koszykarskiej na świecie, która skupia przed odbiornikami miliony odbiorców na całym świecie. Ta dość wąska grupa wybitnych koszykarzy będzie stanowić część naszej analizy.

Pomysł narodził się podczas obserwacji pomeczowego rytuału przybijania piątek. Oglądając go, można zwrócić uwagę, iż dla niektórych zawodników drużyn przeciwnych nie jest to tylko standardowy gest. Zamienia się on często w dłuższe rozmowy, a przecież mówimy często o zawodnikach, którzy jeszcze przed chwilą byli w stanie roszarpać się o najmniejszą różnicę zdań na parkiecie. Cofając się jednak rok czy dwa lata wcześniej odnajdziemy tę dwójkę, grającą razem w jednej drużynie. To stamtąd wywodzą się zapewne pomeczowe rozmowy i uśmiechy, chociaż dzieli ich obecnie kolor koszulki. Pytanie jednak, co bardziej identyfikuje społeczności koszykarzy - wspólna wielopokoleniowa tradycja klubowa, podobny staż w lidze czy może podobny poziom sportowy? Na to pytanie postaramy się odpowiedź przy pomocy naszych algorytmów. Problem jest o tyle ciekawy, iż w lidze przez 73 lata jej funkcjonania przewinęło sie mnóstwo typów zawodników - od tych wiernych jednej drużynie przez całą karierę, po takich, którzy zdążyli zwiedzić 15 miast w ciągu 15 lat swojej kariery.

Dane do stworzenia struktury grafu zostały przez nas pobrane z ogólnodostępnej witryny basketball-reference.com. Ilość danych na temat NBA, jaką można tam odnaleźć, jest ogromna. My skupiliśmy się na zbiorze 4800 koszykarzy, którzy będą stanowić wierzchołki naszego grafu. Dwa wierzchołki będą w relacji, jeśli kiedykolwiek udało im się zagrać wspólnie w jednej drużynie. Ponadto każda znajomość została wzbogacona informacją na temat daty jej zawarcia, a także liczbą wspólnych znajomych - ta została wygenerowana na podstawie struktury grafu, gdyż na stronie taka informacja nie jest dostępna.

8 Uwagi końcowe

Nie ulega wątpliwości, że wykrywanie społeczności i analiza sieci społecznościowych jest i staje się coraz bardziej przydatnym narzędziem. Pracują nad nim potężne zespoły, składające się z najlepszych matematyków i naukowców z całego świata, zatem naszym celem jest przede wszystkim zgłębienie tematu, aby zrozumieć działanie typowych metod i algorytmów, a następnie spróbować je wykorzystać i dostosować do postawionego przed nami problemu. Oczekujemy, że efekty mogą nie być, tak spektakularne jak byśmy tego chcieli, ale mamy nadzieję, że mimo wszystko uda nam się dokonać jakiś ciekawych obserwacji, które przybliżą nam temat wykrywania społeczności oraz problemów, z którymi spotykają się ludzie zajmujący się tym na co dzień.