POLITECHNIKA WARSZAWSKA

Wydział Elektroniki i Technik Informacyjnych

Instytut Automatyki i Informatyki Stosowanej

Jakub Jarzyński

Numer albumu: 218787

Praca dyplomowa magisterska:

Analiza triad w serwisach społecznościowych

Praca pod kierunkiem
dra inż. Mariusza Kamoli

Przewodniczący komisji:

……

Ocena: ……

Data i podpis:

……

Warszawa, 2016

Analiza triad w serwisach społecznościowych

Serwisy społecznościowe gromadzą nigdy wcześniej niedostępne do analizy socjologicznej ilości danych. Informacje te są jednak przeważnie zbyt cenne, by udostępniać je w całości czy udzielać dostępu do dowolnych wartości statystycznych. Stosunkowo niewielkim nakładem pracy możliwe jest jednak umiejętne wykorzystanie dostępnych API do wyboru interesującego wycinka sieci użytkowników i badania jego dynamiki. Zrealizowane tutaj zadanie polegało na zaproponowaniu algorytmu tworzenia takiego wycinka i posłużenie się nim do pobrania sieci 100 tysięcy użytkowników Instagramu, by następnie w około tygodniowych odstępach czasu aktualizować lokalny obraz stanu połączeń między nimi. Uzyskane w ten sposób dane posłużyły następnie w analizie triad, czyli spisu szesnastu możliwych konfiguracji połączeń w ramach dowolnej trójki użytkowników, i weryfikacji powstałych jeszcze przed czasami dostępności danych o takiej skali teorii o ich dynamice. Jak się okazało, większość zachowań ludzi funkcjonujących w sieci społecznościowej daje się opisać podstawowymi prawami wywiedzionymi z teorii grafów.

Analysis of triads in social networks

The social networks store unprecendented amounts of data, crying out for sociological analysis. And though it tends to prove too valuable to any enterprise coming into its possession to ever be made completely open to scientifical research, the API's available allow extraction of certain sub-graphs of entire networks, which if done properly, can lead to interesting conclusions. This paper proposes an algorithm for such extraction and guides you through its example use with Instagram, where a network of 100 thousand users was chosen, saved to a local database, and then updated weekly to observe the dynamics of connections within the graph. The data was used for analysis of triads, i.e. sixteen possible states of directed connections within any group of three people. Triad-related theories dating back to when no data of such scale was available could now be put to test. The results showed that real people's behaviour regarding established and maintained relationships can usually be described in terms of elementary graph theory.

Spis treści

1 Wstęp

Skala danych dostępnych w serwisach społecznościowych jest bezprecedensowa dla analizy socjologicznej. Przed ich powstaniem i gwałtowną popularyzacją weryfikacja teorii o prawach rządzących zachowaniem grup ludzi i dynamiką relacji między nimi wymagała znacznych nakładów pracy. Do zdobycia potrzebnej próbki danych konieczne było zaangażowanie odpowiednio licznego zespołu, a później dostęp do odpowiednich środków na pokrycie kosztów kolekcji danych. Eksperyment musiał być często szczegółowo zaplanowany. Sama analiza danych stanowiła niewielką część całości.

We współczesnych serwisach społecznościowych, gdzie użytkownicy są jedynymi autorami wszystkich dostępnych treści, a z drugiej strony za sprawą udostępnianej o sobie masy informacji największym kapitałem firm obsługujących takie serwisy, wszystkie dane w przekraczających ludzkie pojęcie ilościach czekają tylko na analizę, o ile tylko firmy dostarczają do nich dostęp.

Zdobycie i przeanalizowanie danych w ilościach kiedyś w ogóle niedostępnych leży w takiej sytuacji w zasięgu pracującego bez niczyjej pomocy badacza o pewnych umiejętnościach technicznych. Przy dobrym pomyśle i rozsądnie zdefiniowanym celu możliwe jest również wydobywanie danych ze źródeł do tego nieprzewidzianych, a jednocześnie nienaruszające regulaminów formułowanych przez podmioty dostarczające dostęp do konkretnych danych. Taką właśnie próbą jest niniejsza praca.

1.1 Teza

Punktem wyjściowym było pytanie o prawa rządzące dynamiką sieci społecznościowej. W myśl zasady, że najwięcej informacji na temat węzłów sieci, czyli użytkowników, niesie grupa powiązań z innymi węzłami, za najbardziej wartościowe dane uznano listy znajomych, lub w przypadku połączeń skierowanych - obserwowanych i obserwujących. Nawiązywania połączeń jako takiego nie można jednak jeszcze analizować, potrzebne są konkretne ramy tej analizy. I tak, idąc od najmniejszych komórek w ramach sieci, czyli od pojedynczego węzła, dla którego nie istnieją połączenia, poprzez pary węzłów, czyli diady, dla których istnieją już połączenia, ale bez żadnego kontekstu, dochodzimy do najmniejszej komórki sieci społecznościowej niosącej już pewną informację o węzłach w niej zawartych, czyli do trójki użytkowników zwanej triadą.

Badania triad w sieciach społecznych sięgają lat 60-tych [2, 10, 11]. Istnieją, i zostały opisane w niniejszej pracy, teksty o różnych konfiguracjach połączeń, zarówno skierowanych i nieskierowanych, w ramach triad, ich zachowaniu w zależności od *siły* poszczególnych połączeń czy skalowaniu się zjawisk zachodzących w triadach na całe sieci. W poprzedzającym pracę rozeznaniu w zrealizowanych już badaniach zdaje się brakować jednak póki co weryfikacji tych dawno dostępnych teorii na gruncie współczesnych sieci społecznościowych.

Wybór serwisu do analizy padł na cechujący się prostą funkcjonalnością, ogromną popularnością, udostępniający potrzebne do realizacji zadania API, a co najważniejsze dla analizy triad oparty na połączeniach skierowanych: Instagram. Dostępne zapytania o listy połączeń między użytkownikami umożliwiły rozpoczęcie pracy nad realizacją zadania.

1.2 Środki realizacji

Dane o użytkownikach mediów społecznościowych są zawsze dostępne w pewnej formie. Na tę chwilę brakuje jednak jakiegokolwiek języka zapytań dla danych dotyczących struktury czy dynamiki jakiejkolwiek takiej sieci jako całości. Gromadzenie i komercjalizacja takich danych to przywilej firm je obsługujących. Możliwe jest natomiast, czego dowodzą uzyskane tutaj wyniki, wydobycie w nienaruszający zasad dostępu do API sposób dość istotnych porcji danych. Zasady takiego działania okazują się jednak dość nieoczywiste i zasługujące na opis formalny jako specyficzne algorytmy. Dane o strukturze połączeń w ramach grafu skierowanego tworzonego przez wszystkich użytkowników Instagramu nie są ogólnie dostępne. Dlatego w pierwszym kroku konieczne było opracowanie usystematyzowanej metody wyboru z całej tej sieci fragmentu możliwie spójnego i reprezentatywnego pod kątem właściwości triad.

W pracy przedstawiono projekt algorytmu *wycinania* fragmentu sieci społecznościowej o wysokiej gęstości połączeń, stanowiący pewną modyfikację podstawowych algorytmów przeszukiwania grafu, specyficzną dla sieci społecznościowych. Algorytm w każdym kolejnym kroku dołącza do wycinanego fragmentu węzeł o największym stopniu wejściowym, liczonym jedynie względem węzłów już dołączonych do tworzonego fragmentu. Oprócz opisu podstawowego przebiegu algorytmu wyliczone zostały modyfikacje i sposoby dostrajania parametrów algorytmu pozwalające na uzyskanie wycinka sieci o lepszych właściwościach. Zrealizowana implementacja posłużyła do wycięcia z sieci wszystkich użytkowników Instagramu grafu o 100 tysiącach węzłów, gdzie ograniczenie wynikało z dostępnych zapytań do wysłania do API w skali godziny i pożądanego maksymalnego odstępu między kolejnymi *migawkami* stanu tego wycinka sieci. Gdyby nie te ograniczenia, póki co nie stwierdzono powodów, dla których algorytm miałby być mniej skuteczny przy chęci uzyskania większego fragmentu sieci. Pożądane właściwości uzyskanego grafu, jak i późniejsze statystyki dynamiki triad potwierdziły jego użyteczność przy realizacji postawionego zadania.

Dla wybranych 100 tysięcy użytkowników Instagramu w tygodniowych odstępach prowadzono ponowne skanowanie połączeń. Wszystkie nowe zapisywano w lokalnej bazie wraz ze stemplami czasowymi. Na tej podstawie możliwe było każdorazowe wyszukiwanie nowych triad i wyznaczanie domknięć, poprzez które powstały, czyli poprzedzających stanów połączeń w ramach trójek użytkowników. Efekt końcowy zaś to statystyki dotyczące triad, liczebność poszczególnych typów oraz prawdopodobieństwa domknięć. Tym sposobem, nakładem pracy leżącym w zasięgu pojedynczego programisty-dyplomanta przeprowadzona została kolekcja, analiza i opracowanie wyników statystyk dotyczących grupy przeszło 100 tysięcy osób, co można śmiało uznać osiągnięciem bez precedensu w socjologii przed rozwojem sieci społecznościowych.

1.3 Wyniki

Wyniki statystyczne zostały zestawione z omówionymi, od dawna funkcjonującymi teoretycznymi własnościami triad. Nie stwierdzono silnych rozbieżności, lecz rzeczywiste wyniki zgodnie z przewidywaniami nie pokrywają się w całości z prawami wywiedzionymi z teorii grafów czy z badań prowadzonych na mniejszej próbce danych.

Analizę różnych uwarunkowań wpływających na zachowanie triad poza samą ich strukturą uznano za wykraczającą poza zakres tej pracy. Nie tylko z powodu pracy, jaka była by w związku z tym do wykonania, ale również przez ostrożność przed formułowaniem teorii przy niekompletnych i nie dość licznych danych. Żadne dane o aktywności użytkowników, jak komentarze, *hashtagi*, polubienia lub zamieszczane treści, nie niosą tak zwięzłej i jasnej informacji, jak połączenia.

Uzyskane dane z pewnością mogą posłużyć jednak do dalszej analizy pod kątem właściwości niezwiązanych z analizą triad i w ogóle ściśle z obranym tutaj kierunkiem. W najbliższym czasie w zamiarze autora pozostaje poszukiwanie dalszych możliwości ich interpretacji, najprawdopodobniej przy współpracy specjalistów w dziedzinie analizy sieci społecznych, a nie technicznych i algorytmicznych środków realizacji kolekcji i przetwarzania danych.

2 Serwisy społecznościowe

Analiza sieci społecznościowych jest dziedziną stosunkowo młodą, której znaczenie i popularność rosną jednak w stopniu porównywalnym z niewieloma innymi. Jej zakres w najprostszym ujęciu można nazwać analizą grafów przeniesioną na grunt danych dotyczących sieci społecznościowych. Takie określenie jest tyleż zwięzłe i trafne, co wymaga rozwinięcia i nie oddaje całej różnorodności zagadnień, z którymi się ona mierzy.

Jeszcze kilkanaście lat temu osoby zajmujące się sieciami społecznościowymi uznawane były za zupełnych teoretyków. Przed gwałtownym wzrostem popularności serwisów jak Facebook czy Twitter dostępne dane pochodziły ze źródeł o nieporównywalnie mniejszej liczbie możliwych do wydzielenia węzłów i parametrach mierzalnych dla tych węzłów.

W samym centrum analizy sieci społecznościowej stoi założenie, że połączenia występujące między pewnym użytkownikiem a innymi są najbardziej istotną i wartościową informacją, jaką można o nim zdobyć. Wszystko od osobowości, poprzez pochodzenie i wykształcenie, po poglądy znajduje odzwierciedlenie w zawieranych i podtrzymywanych kontaktach z innymi ludźmi. Same połączenia należy również rozróżniać, nie każde połączenie ma takie samo znaczenie. Najłatwiejszym i najczęściej niezawodnym sposobem rozpoznania wagi połączenia jest częstość komunikacji. Bardziej naturalnym i znajdującym odzwierciedlenie w rzeczywistości typem połączeń są z kolei połączenia asymetryczne.

Dostępne dziś w niespotykanej wcześniej skali dane stanowią bezprecedensową okazję do eksperymentalnej weryfikacji teorii budowanych w socjologii zarówno w skali mikro, jak i makro. Do tych pierwszych zalicza się na przykład sprawdzenie przełożenia liczby wspólnych znajomych na siłę więzi między danymi dwoma użytkownikami, w takim przypadku klasyfikowanych jako połączenia silne. Z drugiej strony równie istotne i równie interesujące dla socjologii są połączenia słabe, rzadziej aktywowane, za to bardzo istotne w skali makro, kiedy ma dojść do rozprzestrzenienia się informacji pomiędzy odrębnymi grupami bliskich znajomych. Poszukiwanie praw obowiązujących od skali mikro do makro jest tutaj szczególnie interesujące.

Rodzaj połączeń dominujących w danym serwisie społecznościowym w dużym stopniu decyduje o jego specyfice. Połączenia nieskierowane, wymagającego zatwierdzenia przez obie strony zazwyczaj niosą mniej informacji w porównaniu z połączeniami skierowanymi, gdzie relacja może być jednokierunkowa i wzajemność nie jest w żaden sposób wymuszana. Bardzo rzadko spotykamy się z wartościowaniem samego połączenia, to znaczy możliwością określenia połączenia jako pozytywnego dla przyjaciela, a negatywnego dla wroga. Kwestią otwartą pozostaje, jak często połączenia skierowane prawdopodobnie powstają z intencją śledzenia osoby nam wrogiej.

Skoro połączenia niosą najwięcej informacji o użytkownikach sieci społecznościowej i o samej sieci, to jak najlepiej je analizować? Ile połączeń rozważać jednocześnie? Jaka jest najmniejsza liczba węzłów wraz z krawędziami między nimi niosąca już pewną informację? Jedną z możliwych odpowiedzi jest triada, czyli 3 węzły, 4 możliwe konfiguracje połączeń nieskierowanych, 16 konfiguracji połączeń skierowanych, przy czym te drugie stanowią lepsze odwzorowanie relacji występujących w rzeczywistości.

Można uznać, że 90% wszystkich fabuł opiera się na relacjach w ramach diad, czyli z udziałem dwóch węzłów, lub triad, począwszy od XVI-wiecznych powieści, a skończywszy na scenariuszach współczesnych seriali i przebojach kinowych [1]. Osobiste bliskie kontakty z ludźmi również zamykamy najczęściej w w grupach do trzech osób. Rzut oka na otaczające nas grupki osób rozmawiających przy dowolnej okazji towarzyskiej również dostarczy przede wszystkim przykładów 2- i 3-osobowych.

Przy współczesnej skali rozwoju mediów społecznościowych w sposób naturalny ich badanie i w rezultacie łatwe wnioskowanie informacji o użytkownikach służy nie tyle celom naukowym, co komercyjnym. Wobec ostatnich rewolucji w Egipcie i Tunezji [1] znaczenie dziedziny wykracza jednak i poza ten zakres, zyskując kontekst globalny i polityczny. Prawidłowa analiza, interpretacja i zrozumienie danych dostępnych dzisiaj na wyciągnięcie ręki daje moc przewidywania przyszłości, a może nawet kreowania jej.

3 Triady

Przy analizie grafów, a do takiej analizy przeważnie sprowadzane jest badanie sieci społecznościowych, łatwo wydzielić dwie liczne grupy sposobów mierzenia pewnych związanych z nimi wartości. Pierwsza grupa to czynności zmierzające do określenia własności węzłów jako takich, zliczania ich stopni wejściowych i wyjściowych, rangi czy przechodniości. Druga z kolei obejmuje czynności prowadzące do poznania pewnych ogólnych mierzalnych wartości związanych ze strukturą połączeń w grafie. Dzieli się ona z kolei na operacje wychodzące od całościowej struktury grafu, a następnie wydzielające na podstawie pewnych kryteriów jego części składowe jako różnego rodzaju podgrafy, i te wychodzące od pojedynczych węzłów i w pewien oddolny sposób budujące coraz większe struktury. Do tej ostatniej podgrupy zalicza się analiza triad.

3.1 Formy analizy danych w sieci społecznościowej

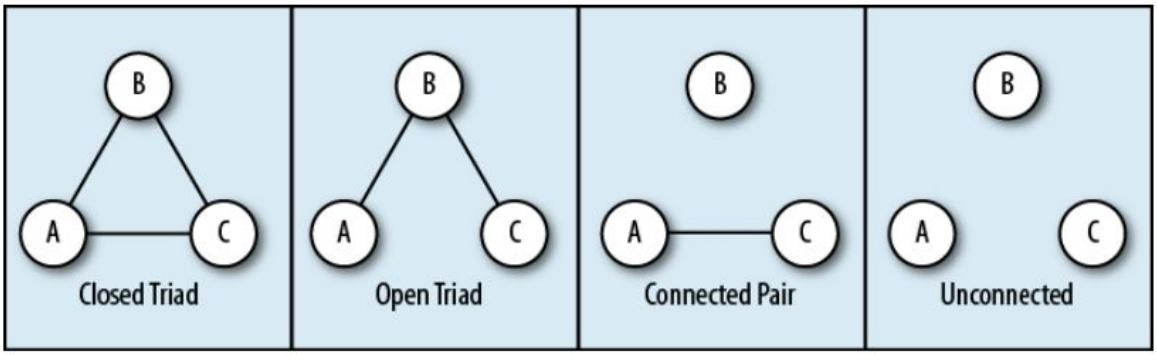
W miarę jak coraz więcej aktywności przenosi się do mediów społecznościowych, zmienia się również sposób korzystania z nich. Jak wiadomo, większość użytkowników gromadzi obecnie bardzo znaczące ilości połączeń, tworząc w ten sposób znacznie większe grupy znajomych, komunikujących się nie tylko poprzez prywatne wiadomości, ale i komentarze pod wpisami wspólnych znajomych. Przed dominacją tego typu mediów kręgi znajomych były mocniej zamknięte.

Pytanie jaki wpływ mają takie procesy na budowanie się sieci międzyludzkim w szerszym kontekście, zajmowało naukowców już od lat 90-tych, m.in. grupę prowadzoną przez B. Welliama [12]. Siła połączeń w sieci może przynieść nową perspektywę i odpowiedź na pytanie, jak aktywność w sieci na różnych portalach przenosi się poprzez połączenia o różnej sile. Przy setkach połączeń utrzymywanych przez znaczną część użytkowników można zastanawiać się, ile z nich wskazuje na faktyczne utrzymywanie kontaktu, a ile pozostaje nieaktywnych i właśnie aktywowanie ich może prowadzić do najciekawszych zjawisk możliwych do zaobserwowania w sieciach.

Naukowców nie od dziś zajmuje analiza siły połączeń i zjawiska prowadzące do nawiązywania nowych. Badania prowadzone przez grupę Camerona Marlow [5] polegały na zadaniu pytania, na ile poszczególne połączenia widoczne na profilu użytkownika świadczą o faktycznej komunikacji, a nie są tylko informacją widoczną na profilu. Przy pomocy dostępnych danych zostały wyróżnione trzy typy połączeń na podstawie obserwacji prowadzonych na przestrzeni jednego miesiąca: komunikacja dwustronna, komunikacja jednostronna, śledzenie aktywności drugiego człowieka.

3.2 Definicja i typy triad

Triadą nazywa się trzy konkretne wierzchołki grafu wraz ze wszystkimi łączącymi je krawędziami. Gdy te ostatnie są nieskierowane, istnieje 4 możliwych triad (rys. 3-1). O wiele ciekawsze przy analizie triad są grafy skierowane. Tam wyróżnia się 16 możliwych wariantów.



Rys. 3-1 Spis triad dla grafów nieskierowanych. W tym przypadku istnieją cztery możliwe warianty zestawu połączeń w ramach trójki wierzchołków. Źródło [1], str. 67

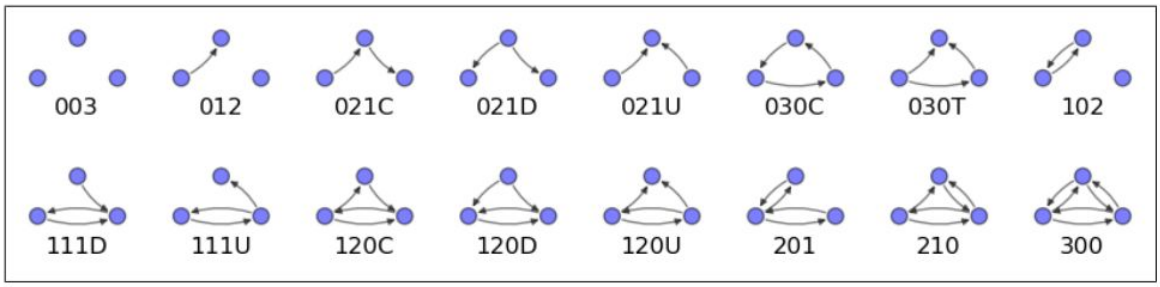
Triadą zamkniętą nazywamy podgraf pełny między węzłami w triadzie, otwartą z kolei taką triadę, w której pewnych połączeń brakuje. Niektóre otwarte triady uznawane są za niestabilne, niektóre za stabilne. Pewne konfiguracje połączeń niosą zatem informację o wysokim prawdopodobieństwie rychłego pojawienia się kolejnego połączenia, podczas gdy niektóre nie zapowiadają domknięcia triady, póki oczywiście z powodów niezależnych, niezapowiadanych samą strukturą triady, nie pojawi się jakieś nowe połączenie.

Kolejnym typem triad, najrzadziej obserwowanym w sieciach społecznościowych są te występujące w grafach ważonych z dwiema dopuszczalnymi wartościami krawędzi: + i -. Takie połączenia pozwalają na rozróżnienie relacji przyjacielskich i koleżeńskich od relacji wrogich. Między każdym węzłem triady istnieje wówczas określone połączenie dodatnie lub ujemne. W tej pracy nie zajmujemy się wagami krawędzi w triadach.

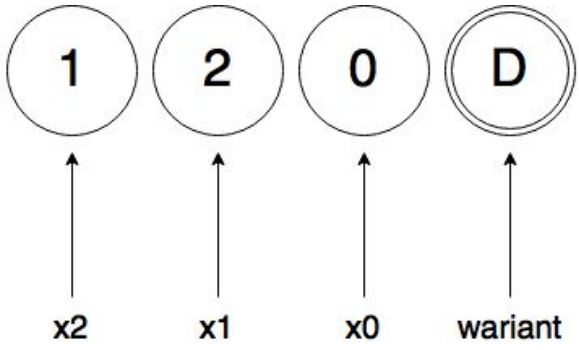
3.3 Triad census

Pojęciem *triad census* określa się badanie wszystkich triad, w których jednym z węzłów jest jeden obrany węzeł. Na podstawie wyników takiego pomiaru prawdopodobne jest określenie pozycji, jaką ten węzeł pełni w całości sieci.

Spis szesnastu skierowanych triad (rys, 3-2) uzupełnia jednoznaczny sposób identyfikacji każdej z nich poprzez trzy cyfry z ewentualną konieczną dla rozróżnienia wariantów literą na ostatniej pozycji (rys. 3-3). Trzy cyfry odpowiadają po kolei liczbom podwójnych, pojedynczych i pustych połączeń między dwoma węzłami w triadzie. Litery z kolei albo rozróżniają kierunek dwóch połączeń w triadzie (U - up, D - down), albo ich przechodni lub cykliczny charakter (C - circle, T - transitive).



Rys. 3-2 Spis triad dla grafów skierowanych. W tym przypadku istnieje szesnaście możliwych wariantów zestawów połączeń w ramach trójki wierzchołków. Źródło [1], str. 76



Rys. 3-3 Znaczenie kolejnych pól w etykiecie triady. Reprezentują one po kolei: liczbę podwójnych, pojedynczych i braku połączeń między parami wierzchołków. Ostatnie pole służy rozróżnieniu wariantów zależnie od skierowania połączeń.

3.4 Zastosowania

Analiza triad wśród połączeń między ludźmi jest interesująca nie tyle ze względów socjologicznych, nie jest więc nauką dla samej nauki, teoretyzowaniem, ale przekłada się na konkretne możliwe do pozyskania korzyści. Pewne luki w triadach, brak pewnych połączeń, może w sposób udowodniony w biznesie i polityce w sposób wyraźny prowadzić do korzyści, zwłaszcza dla tego węzła triady, który jest połączony z pozostałymi, nie połączonymi. Pośrednik w sprzedaży korzysta z tego, że podmioty, między którymi pośredniczy, się nie znają. Również w świecie finansów informacja, przede wszystkim w połączeniu z inną informacją z niezależnego źródła, decyduje o przewadze przy spekulacji. Równie ciekawych wniosków może dostarczyć analiza związków pewnych państw z supermocarstwami i państwami sąsiednimi, niekoniecznie znajdujących odzwierciedlenie w relacjach między nimi.

3.5 Konteksty analizy triad

W kontekście triad poza liczebnością poszczególnych typów, a nawet pod tym względem najwięcej pola do interpretacji dają powody domykania się triad. Poszukiwane są powody stabilności i niestabilności poszczególnych ich typów.

Domykanie się triad jest dość intuicyjne i łatwo znaleźć na nie przykłady we własnym otoczeniu. Prawdopodobieństwo nawiązania znajomości poprzez dwie osoby zdecydowanie wzrasta, kiedy mają jakiegoś wspólnego znajomego. Po pierwsze dlatego, że mogą się za jego pośrednictwem spotkać, po drugie ponieważ dzielenie znajomości często wskazuje już na pewien stopień podobieństwa między nieznającymi się osobami.

3.6 Własność domknięcia silnych triad

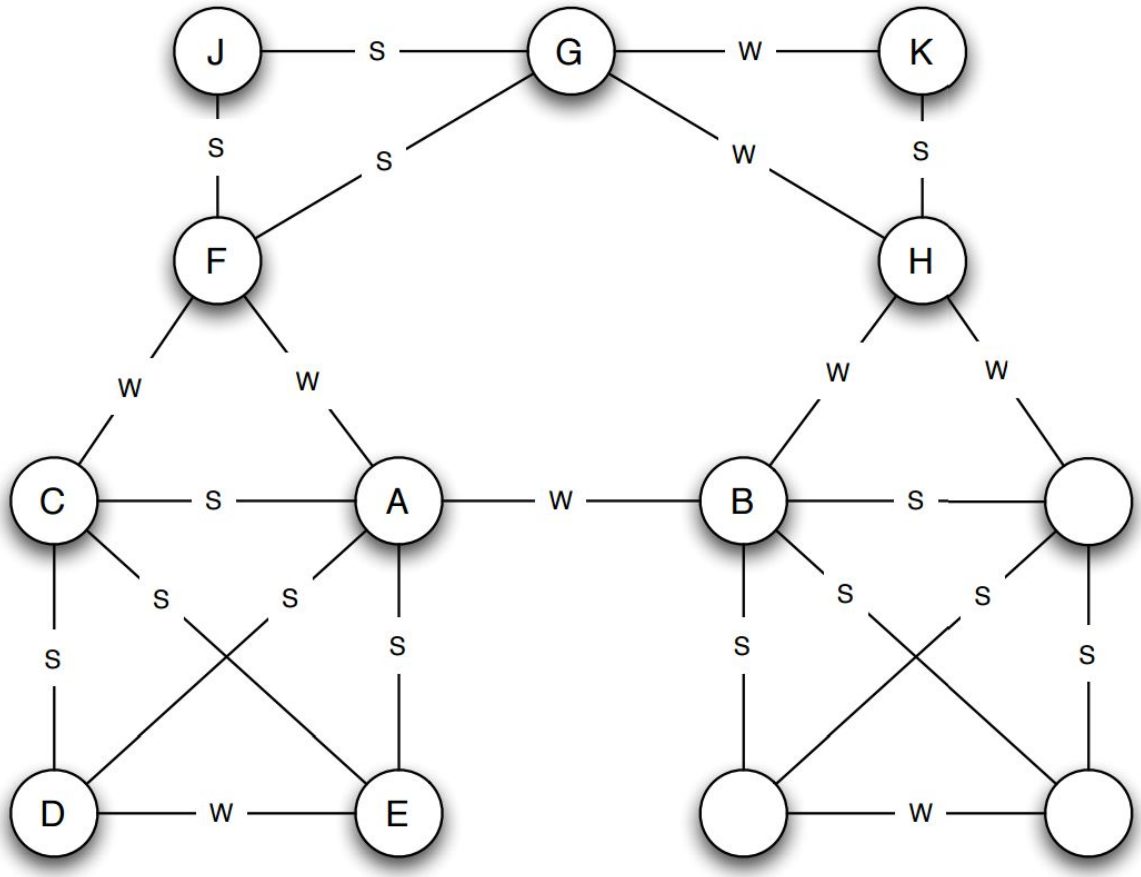
Należy pamiętać o konieczności rozróżnienia siły połączeń między węzłami. Chodzi tutaj o jakąś miarę bliskości relacji między osobami z już nawiązanym połączeniem. Nie jest ona łatwa do ścisłego zdefiniowania, chociażby dlatego, że informacja możliwa do zdobycia poprzez obserwację sieci nigdy nie będzie w pełni oddawała całej treści relacji między ludźmi. Na przykład ludzie pozostający w związku kohabitacyjnym mogą w sposób oczywisty wymieniać znacznie mniej wiadomości, ponieważ i tak spędzają ze sobą większość czasu.

Dla ułatwienia definicji miary siły połączeń można wprowadzić rozróżnienie jedynie na *silne* i *słabe* połączenia (rys. 3-4), gdzie pierwsze odpowiadają połączeniom między faktycznymi przyjaciółmi, natomiast te drugie między dalszymi znajomymi. Po przyjęciu takiej miary można podjąć próbę opisania wszystkich krawędzi w grafie reprezentującym strukturę sieci społecznościowej jako połączenia *silne* i *słabe*.

Taka dodatkowa informacja o triadach na pewno wzbogaci wyniki analizy domknięć triad występujących w badanej sieci. Wcześniej przytoczone argumenty o prawdopodobieństwie domknięć w triadach nabierają siły w przypadku *silnych* połączeń. Prowadzi to do stwierdzenia, że jeśli dla dwóch węzłów nawiązanie połączenia jest szczególnie prawdopodobne, jeżeli mają one *silne* połączenia z tym samym innym węzłem.

Przy nienawiązaniu połączenia w takiej sytuacji mówi się o naruszeniu właściwości domknięć triad, a samą triadę pozostającą w takim stanie uznaje się za niestabilną. Naturalnie taka zasada jest zbyt ogólna, aby można było stosować ją do wszystkich węzłów w analizowanej sieci. Stanowi jednak przydatne przybliżenie w dalszej analizie.

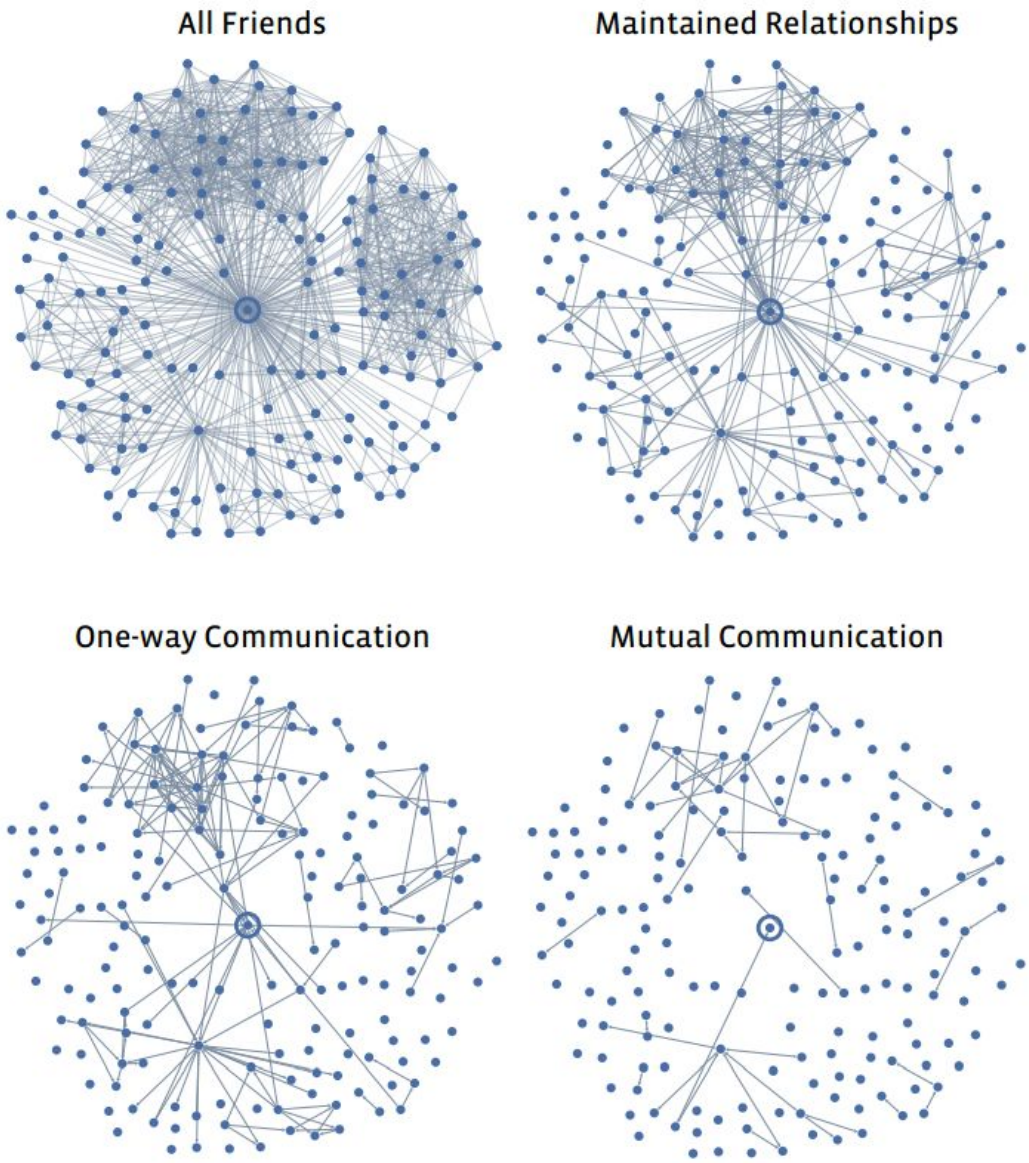
Taki podział zbliża nas do zrozumienia, jakie przełożenie na rzeczywiste relacje międzyludzkie ma cały zbiór połączeń widocznych na profilu. Wynika z tego kilka interesujących wniosków (rys. 3-5). Nawet osoby z bardzo licznymi listami znajomych faktycznie komunikowały się z podobną liczbą spośród nich i zazwyczaj było to kilkanaście osób [10]. Również liczba osób jedynie biernie obserwowanych była zawsze podobna i przeważnie nie przekraczała kilkudziesięciu. Ta druga obserwacja jest szczególnie interesująca i bezprecedensowa w kontekście sieci połączeń międzyludzkich. Dysproporcja między ludźmi, z którymi się komunikujemy, a tymi, których obserwujemy, znajduje odzwierciedlenie również w ogólnym sposobie odbioru świata, jak przyswajanie znacznie większej ilości informacji za pośrednictwem serwisów internetowych zamiast lektury dłuższych i bardziej monotematycznych powieści.



Rys. 3-4 Przykładowa sieć *silnych* i *słabych* połączeń. Połączenia spełniają własność *Domknięcia silnych triad* dla każdego z połączeń: jeśli węzeł ma *silne* połączenia z dwoma węzłami sąsiednimi, te węzły muszą być połączone co najmniej *słabym* połączeniem. Źródło [8], str. 52

Podobne badania były prowadzone na przykładzie Twittera, gdzie użytkownicy zamieszczają krótkie wiadomości lub jak wolą niektórzy *ćwierkają*. Można również śledzić profile innych użytkowników i przekazywać dalej ich wpisy. Te dwie ostatnie formy aktywności odpowiadają biernemu obserwowaniu pewnego szerszego grona ludzi na Facebooku lub bliższej komunikacji poprzez bezpośrednią wymianę wiadomości. Udostępnienie i skomentowanie wpisu to już bliższa komunikacja, jest widoczna dla obserwowanego.

Badania na ten temat prowadziła grupa: Huberman, Romero i Wu [7]. Dla każdego użytkownika sprawdzane było, wiadomości ilu innych użytkowników śledzi, oraz jako *silne* połączenia traktowano te z tymi, z którymi wymieniono podczas procesu obserwacji co najmniej dwie wiadomości. W ten sposób wyznaczono zależność liczby *silnych* połączeń od liczby połączeń w ogóle. Jak się okazało, podobnie jak na Facebooku nawet dla bardzo dużych zbiorów obserwowanych profili liczba *silnych* połączeń pozostawała niska.



Rys. 3-5 Cztery różne typy więzi między użytkownikami Facebooka. Taka klasyfikacja pomaga w zrozumieniu, jak duże listy znajomych przekładają się na właściwe wzorce aktywności. W lewym górnym rogu widoczna jest sieć wszystkich połączeń w ramach zbioru użytkowników. Pozostałe trzy obrazki ukazują stopniowy zanik połączeń w miarę nakładania coraz silniejszych ograniczeń na definicję połączenia między węzłami. Tym sposobem wyłania się również podział sieci na grupy, w ramach których utrzymywane jest faktycznie więcej połączeń. Źródło [8], str. 61

W ogólności okazuje się, że utrzymywanie *silnych* połączeń koniec końców wymaga inwestycji czasu i energii, ostatecznie rozbija się więc o ograniczoną liczbę godzin w dobie. Dlatego widać ogólnie panujące górne limity na tego typu połączenia. Z kolei liczby *słabych* połączeń nie podlegają już takim regułom i w ten sposób gromadzenie znacznych ich zbiorów jest dużo łatwiej osiągalne, a przede wszystkim w ogóle możliwe.

3.7 Homofilia w sieciach społecznych

Nawiązywanie się połączeń między ludźmi o pewnym wspólnym mianowniku samo w sobie jest cechą sieci społecznościowych. Wśród cech wspólnych łatwo wyróżnić pewne kategorie. Kryteria wyboru grupy towarzyskiej nazywa się *selekcją*, na przykład na podstawie rasowej czy innych cech wspólnych. Niektóre mogą być specyficzne dla ludzi, inne z kolei wynikają z zamieszkiwania tych samych terenów i wtedy mówi się o selekcji *implicite*, bo jest ona nieświadoma i niezależna od jednostek.

Atrybuty niezbywalne jak rasa zostają określone raz na zawsze dla każdego człowieka w momencie narodzin. Przez resztę życia mają one znaczący wpływ na sieć połączeń, której częścią stanie się człowiek.

Atrybuty zmienne są znacznie bardziej złożone. Jednostki mają wpływ na formowanie w ten sposób swojej sieci społecznej. Nasuwają się przykłady awansu społecznego, przeprowadzki do lepszej dzielnicy, zmiany otoczenia, ale również degradacji społecznej. Co wyróżnia tego typu procesy, to że wpływ jest dwustronny, to znaczy jednostka w pewnym stopniu decyduje o sieci tworzącej się wokół niej, z drugiej z kolei sieć, w ramach której się porusza, ma na nią mocny wpływ i determinuje jej postępowanie.

Statyczna analiza połączeń i interakcji między użytkownikami nie dostarcza jeszcze informacji o tych zjawiskach. Lepiej jest to widoczne przy obserwacji dynamiki sieci. W pracy Cohena i Kandela [8] sugeruje się, że obydwa efekty są widoczne w obserwowanych przez nich danych, przy czym efekt wpływu środowiska pozostaje znacznie słabiej widoczny od efektu selekcji.

Podobny temat w swych badaniach podjęli Christakis i Fowler [9]. Na próbce 12 tysięcy użytkowników śledzili wskaźniki otyłości przez okres 32 lat. Badania wykazały, że otyli i nieotyli ludzie w nomenklaturze grafów łączą się w klastry zgodnie z zasadami homofilii.

Okazuje się, że określenie wpływu różnych czynników wpływających na powstawanie połączeń w sieci wymaga szczegółowej analizy. Prosty wniosek brzmi, że nawet jeśli ludzie zdają się podobni do osób ze swojego otoczenia, zwykle trudno określić powody takiego stanu rzeczy. Obserwacja i pomiary homofilii mogę stanowić w takim razie zarówno konstatację, jak i punkt wyjściowy badań.

3.8 Przynależność w sieciach społecznych

Omówione do tej pory sposoby interpretacji procesów rządzących kształtowaniem się struktur sieci społecznych wszystkie pochodziły jakoby spoza samej sieci. Można jednak również powody te umieścić w samej sieci.

Dobry przykład stanowią tutaj sieci łatwo reprezentowane przez grafy dwudzielne, gdzie graf dzieli się na zbiory wierzchołków niepołączonych, i tylko jednym z tych zbiorów są przeciętni użytkownicy, pozostałe zbiory to na przykład strony na Facebooku, które obserwują, grupy i fora, do których należą. Takie sieci nazywa się sieciami *przynależności*.

Nawet jeśli prowadzone badania nie skupiają się na tego typu sieciach, warto mieć świadomość możliwości takiej interpretacji struktury sieci i na przykład wyodrębnić i odrzucić węzły-*ogniska*, nie reprezentujące użytkowników równych innym użytkownikom.

3.9 Eksperymenty na sieciach społecznych

Przeprowadzone w latach 60-tych przez Newcomba [10] badania przypominają współczesne *reality show*. Na początku semestru siedemnastu studentów jednej rasy zamieszkało w domu bractwa studenckiego. W cotygodniowym cyklu mieli za zadanie oceniać swoje relacje z pozostałymi mieszkańcami domu.

Wyniki pokazały, że połączenia o asymetrycznej wycenie, gdzie wyceny dwóch stron mocno się różniły, były najmniej stabilne i trwały najkrócej. Z kolei stan połączeń o symetrycznych wycenach był znacznie bardziej stabilny. Najbardziej stabilne okazywały się zestawy połączeń w ramach triad.

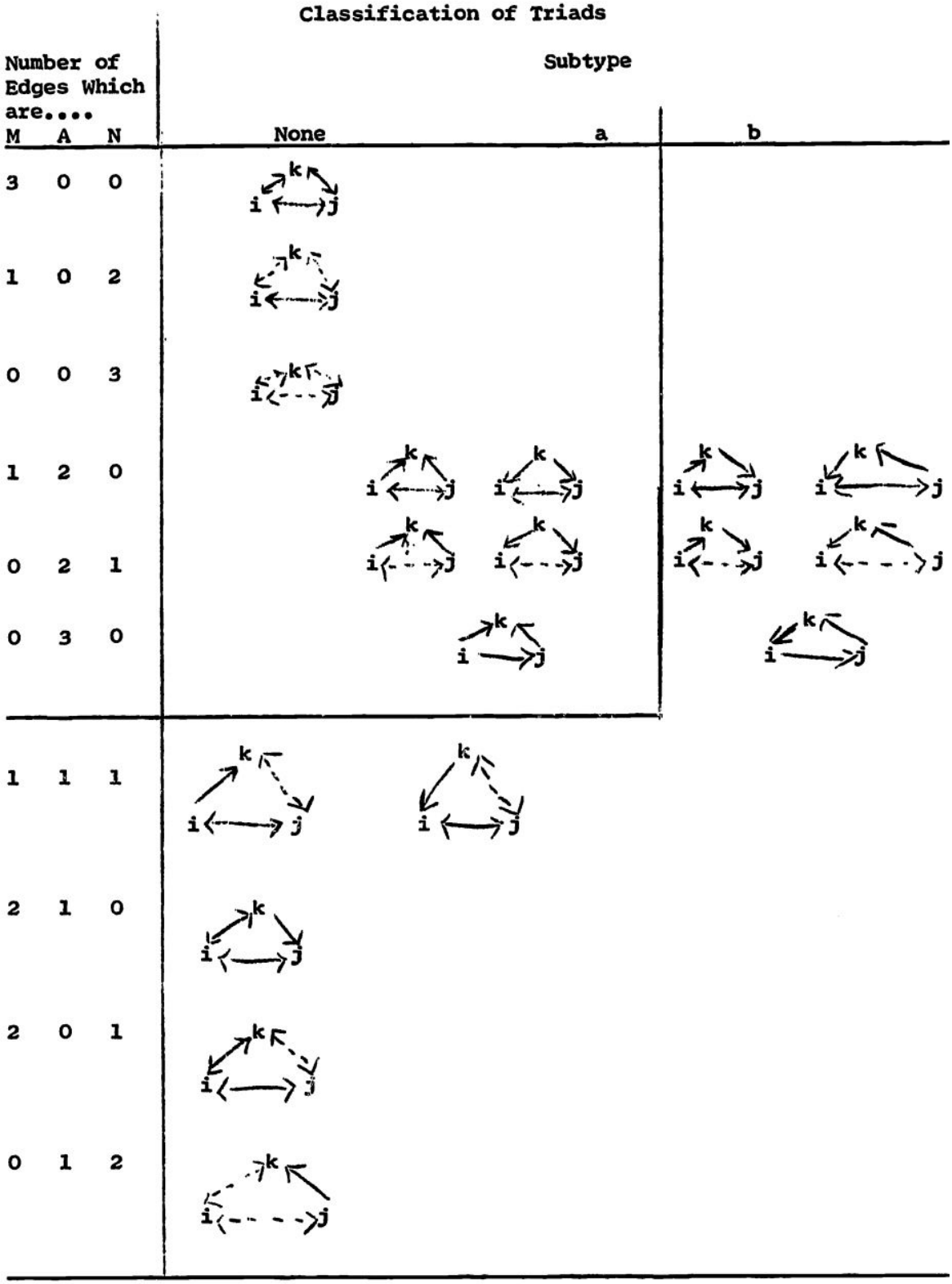
3.10 Stabilność triad

Praca Davisa i Leinhardta z 1967 roku [4] miała za cel weryfikację tezy Homana [11] o inherentnej tendencji małych grup społecznych do tworzenia klik. Badacze zaproponowali model teoretyczny oparty o analizę grafową i jego główne założenie - siedem *zakazanych* triad będących odstępem od przewidywanej domyślnej struktury sieci. Oczekiwany rozkład liczebności triad w sieci wywiedli z podstawowych własności grafów.

Teorię potwierdzali danymi zgromadzonymi z setek zastanych zbiorów danych, w tym trzydziestu szkół i trzydziestu grup osób dorosłych. Dane przeważnie potwierdzały zaproponowane teorie. Teorie Homana [11] zostały wówczas uznane za potwierdzone badaniami.

Davis i Leinhardt przedstawili katalog szesnastu triad występujących w grafach skierowanych (rys. 3-6). Na osi pionowej umieścili dziesięć możliwych triad z pominięciem zwrotu połączeń od węzła A. W środkowej strefie listy trzy typy (1-2-0, 0-2-1, 0-3-0, 1-1-1) zależą z kolei mocno od zwrotu połączeń wychodzących z tego węzła. Te triady podzielono na typy *a* i *b*.

Ogólna zasada modelu głosi, że triady poniżej linii poziomej i na prawo od linii pionowej (rys. 3-6) nie powinny istnieć. Następnie badacze wyjaśnili, dlaczego poszczególne typy są dozwolone lub niedozwolone.



Rys. 3-6 Spis i klasyfikacja triad w grafach skierowanych z 1967 roku. Źródło [7], str. 12

3.11 Grafowa interpretacja stabilności triad

Triady typu 3-0-0 są dozwolone i reprezentują trzy znające są osoby usytuowane na tym samym poziomie.

Triady 1-0-2 składają się z dwóch osób, *i* i *j*, w tej samej klice na tym samym poziomie, oraz trzeciej *k*, z innej kliki na tym poziomie.

Triady 0-0-3 składają się z trzech osób z różnych klik na tym samym poziomie.

Triady 1-2-0-a składają się z dwóch osób, *i* i *j*, z tej samej kliki na tym samym poziomie, oraz trzeciej *k* z wyższego lub niższego poziomu.

Triady 0-2-1-a składają się z dwóch osób, *i* i *j*, z różnych klik na tym samym poziomie, oraz trzeciej osoby *k* z wyższego lub niższego poziomu.

W triadach 2-1-0 *i* i *j* są w tej samej klice na tym samym poziomie. Relacja *M* między *i* a *k* wskazuje, że *k* jest również w tej klice, z kolei relacja *A* pomiędzy *k* a *j* wskazuje, że *k* jest na wyższym poziomie.

W triadach 2-0-1 *i* i *j* są w tej samej klice. Relacja *M* między i a k wskazuje, że k jest również w tej samej klice, z kolei relacja *N* między k a j wskazuje, że k jest w innej klice.

W triadach 0-1-2 dwie relacje *N* wskazują, że *i*, *j* i *k* są na tym samym poziomie, choć w innych klikach, z kolei relacja *A* między *j* a *k* wskazuje, że *k* sytuuje się na wyższym poziomie.

Triady typu 0-3-0-a składają się z osób z trzech różnych poziomów, przy czym *k* jest najwyżej, *i* najniżej, a *j* między nimi.

O tym, które triady uznaje się za zakazane, decydują sprzeczności występujące przy ich analizie zgodnej z powyższymi zasadami.

W triadach 1-2-0-b *i* i *j* są z tej samej kliki na tym samym poziomie, z kolei *k* jest na poziomie wyższym niż jeden, a niżej niż drugi z nich, co prowadzi do sprzeczności.

W triadach 0-2-1-b *i* oraz *j* są z różnych klik na tym samym poziomie, z kolei *k* jest na poziomie wyższym od jednego, a niższych od drugiego z nich, co prowadzi do sprzeczności.

W triadach 0-3-0-b mamy do czynienia z przykładem cykliczności uniemożliwiającej uporządkowanie według poziomów. Taka sytuacja również reprezentuje sprzeczność z przyjętym modelem.

W triadach 1-1-1, niezależnie od zwrotu połączeń między węzłami, *i* oraz *j* muszę znaleźć się w tej samej klice na tym samym poziomie, z drugiej strony połączenie skierowane między *i* a *k* wskazuje, że *k* jest na innym poziomie, podczas gdy relacja N wskazuje, że k jest na tym samym poziomie, co również prowadzi do sprzeczności z przyjętym modelem.

3.12 Relacja między jednostką a strukturą sieci

Analiza sieci społecznościowych dostarcza narzędzi do zestawienia teorii socjologicznych w dwóch różnych skalach: mikro i makro. Wielkoskalowe badania statystyczne jak i ilościowe służą weryfikacji modeli dotyczących mobilności społecznej, samoorganizacji społeczności czy struktury działań politycznych. W mniejszej skali dostarczają z kolei materiału do badań powiązań międzyludzkich w obrębie mniejszych grup.

Teza Ganovettera [2] brzmi, że to analiza procesów w sieciach międzyludzkich dostarcza najwięcej możliwości na znalezienie połączeń między procesami zachodzącymi w skalach mikro i makro. Podjął próbę analizy jednej konkretnej cechy sieci, która może nieść interesujące informacje, a jest możliwa do ścisłej analizy, czyli znaczenia *słabych* połączeń w sieci, ich znaczenia i wpływu na znaczące zmiany w strukturze sieci.

Zgodnie z tymi założeniami osobiste doświadczenia jednostek są mocno silnie warunkowane efektami wielkoskalowymi i w znacznym stopniu pozostają poza wpływem poszczególnych jednostek. Połączenie dwóch skal efektów miało być więc kluczowe dla dalszego rozwoju badań socjologicznych w tej dziedzinie.

3.13 Cel pracy

Głównym celem postawionym w tej pracy jest pozyskanie możliwie licznego zbioru danych niosącego możliwie pełną informację o dynamice połączeń w określonej sieci społecznościowej. Prześledzenie dynamiki pojawiania się nowych połączeń na wybranym wycinku sieci społecznościowej powinno doprowadzić do pozyskania danych w postaci możliwej do masowej analizy pod kątem triad obecnych na początku, pojawiających się w czasie, a w końcu statystyk dotyczących przejść między triadami. Współczesna dostępność danych o stanie rzeczywistych sieci społecznościowych ma w ten sposób przysłużyć się weryfikacji zestawu tez o zachowaniu się triad postawionych w przeszłości.

Triady to najmniejsza jednostka społeczna w sieci społecznościowej, dla której na podstawie stosunkowo prostych pomiarów możliwe jest wyznaczenie prawdopodobieństwa najbliższych dotyczących jej wydarzeń. Może więc stanowić najkrótszą drogę do przewidywania przyszłości w relacjach międzyludzkich lub nawet kształtowania ich.

4 Instagram

Niezbędne zdaje się słowo wyjaśnienia i uzasadnienia dokonanego wyboru sieci społecznościowej do analizy. Prace naukowe opisujące sieci społecznościowe zajmują się najczęściej serwisami o choć trochę poważniejszej zawartości niż nieprzebrany katalog zdjęć jedzenia i autoportretów robionych z ręki. Kierunek obrany w niniejszej pracy jest jednak nieprzypadkowy i twardo ugruntowany w szeregu bardzo korzystnych dla prowadzonego eksperymentu własności Instagramu, wyliczonych i objaśnionych poniżej.

4.1 Charakterystyka sieci

W cieszących się największą popularnością mediach społecznościowych jak Facebook czy LinkedIn nawiązanie połączenia wymaga obopólnej zgody. Wpływa to na faktyczne znaczenie znajomości na tych portalach. Zaproszenia do znajomości często nie wypada nie przyjąć. Z drugiej strony wielu użytkowników szczyci się liczbą nawiązanych połączeń i prowadzi pewnego rodzaju kolekcjonerstwo. W ten sposób obniża się ogólna wartość informacji o połączeniu między użytkownikami.

Równie istotna jest sama nomenklatura połączeń. Nazwanie połączeń *śledzeniem*, a nie *znajomością* sprzyja zawieraniu połączeń mimo niezawarcia znajomości w prawdziwym życiu. Osoby prowadzące profil publiczny zgadzają się na bycie obserwowanym przez wszystkich zainteresowanych, z kolei ci o ustawieniach prywatnych prawdopodobnie będą akceptować prośby o obserwowanie wyłącznie od osób, które chociaż pośrednio są im znane.

Przede wszystkim w porównaniu z występującymi na przykład w przypadku Facebooka czy LinkedIn połączeniami nieskierowanymi, do zawarcia których potrzebna jest obustronna akceptacja - połączenia na Instagramie są w całości nieskierowane. Oznacza to, że połączenie wychodzące z pewnego węzła i prowadzącego do pewnego innego węzła oznacza tylko fakt obserwowania przez pierwszy węzeł węzła drugiego. W takiej sytuacji węzeł drugi może, lecz nie musi, odwzajemnić zainteresowanie poprzez rozpoczęcie obserwowania swego obserwatora.

Przełożenie specyfiki połączeń na graf skierowany oznacza występowanie w badanej sieci społecznościowej pełnego zakresu szesnastu triad występujących w spisie. Wyniki eksperymentów niosą dzięki temu znacznie więcej informacji.

4.2 Zakres aktywności użytkowników

Drugim, równie istotnym argumentem przemawiającym za wyborem danego serwisu jest ogromna prostota jego budowy (rys. 4-1). Cała dopuszczalna działalność użytkownika sprowadza się niemalże wyłączeni do obserwowania innych użytkowników i samodzielnego publikowania zdjęć (rys. 4-2). Ponieważ nie istnieje inna możliwość niewyświetlania publikacji obserwowanego użytkownika niż zerwanie ze swojej strony połączenia skierowanego do reprezentującego go węzła, widoczne przy analizie sieci połączenie jednoznacznie świadczy o utrzymaniu kontaktu między dwoma osobami. W najgorszym przypadku publikacje jednej z nich są ze zniecierpliwieniem przewijane przez drugą, nie ma ona innego sposobu, aby przestać je widzieć.

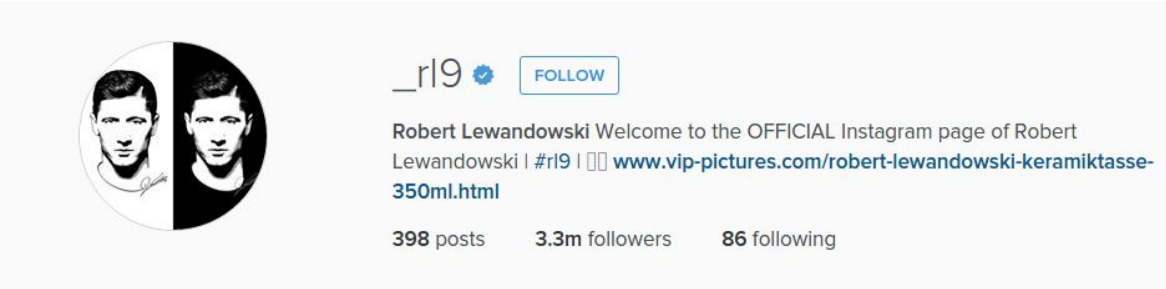
Poza publikowaniem i obserwowaniem użytkownikowi pozostaje możliwość komentowania lub polubienia zdjęć innych. Analiza danych statystycznych związanych z tymi zachowaniami może również prowadzić do ciekawych wniosków, jednak wykracza poza zakres tej pracy. Zgodnie z przedstawionym powyżej uzasadnieniem występowanie połączeń między użytkownikami stanowi wystarczający dowód na występowanie między nimi pewnej relacji.

Rys. 4-1 Zestaw podstawowych informacji o profilu użytkownika. Poza nazwą zawiera zdjęcie profilowe, liczbę zamieszczonych postów oraz połączeń wychodzących i wchodzących.

Rys. 4-2 Pojedynczy post na profilu użytkownika. Każdy post musi zawierać zdjęcie. Poza tym autor może zamieścić do niego opis, a w nim zawrzeć zestaw tagów. Wpis mogą polubić lub skomentować inni użytkownicy.

4.3 Zakłócenia w strukturze sieci

Przy analizie schematów zachowania się i domykania triad występuje potrzeba wstępnego oczyszczenia sieci z węzłów nieinteresujących, ponieważ zbyt różnych od pozostałych. Mowa tutaj przede wszystkim o profilach osób znanych (rys. 4-3), stron prowadzonych przez kluby czy restauracje. Oczywiste jest, że takie profile będą miały bardzo wysoki stopień wejściowy, a powstawanie i domykanie się triad z ich otoczenia nie będzie miało związku z ogólnymi tezami dotyczącymi tych procesów. Jak najwcześniejsze wyeliminowanie tych triad sprowadza się jednak do prostej operacji rozpoznania wysokiego rzeczywistego stopnia wejściowego i w rezultacie usunięcia ich ze zbioru węzłów analizowanego grafu.



Rys. 4-3 Podstawowe informacje na profilu sławnego polskiego sportowca. Widoczna jest mocna dysproporcja między połączeniami wychodzącymi a wchodzącymi.

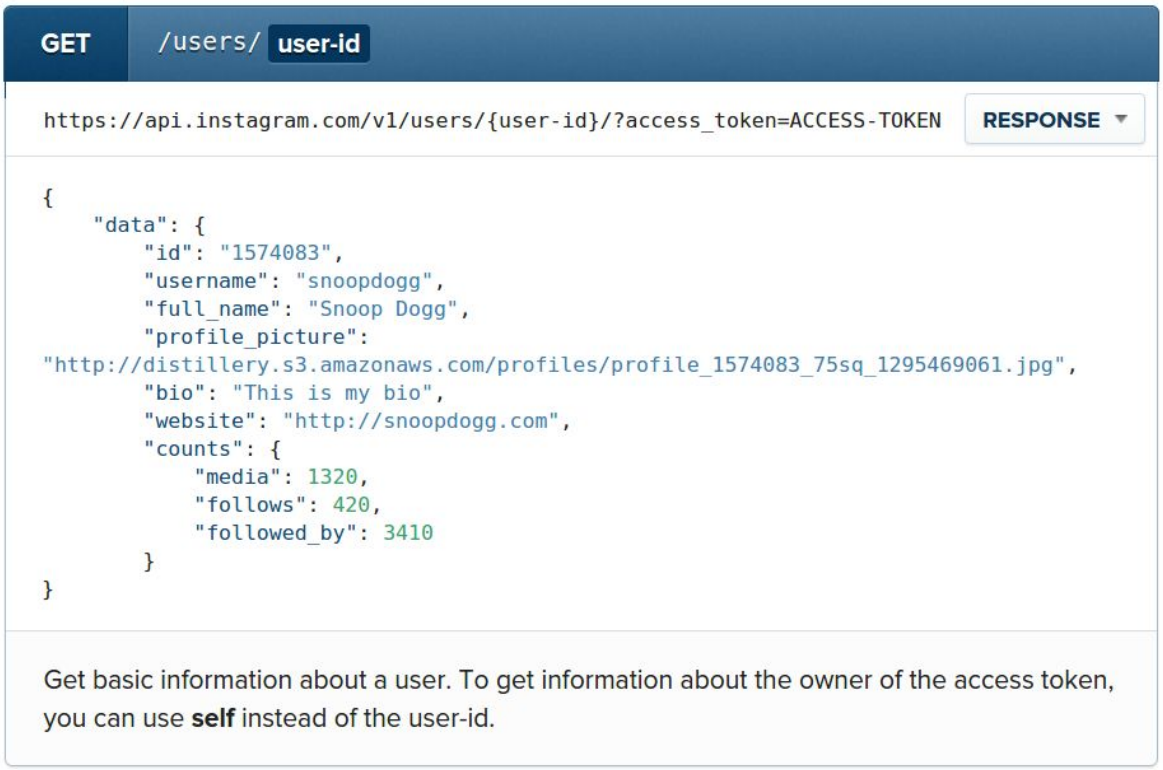
Celem jest znalezienie potwierdzenia dla procesów zachodzących w triadach w sieci społecznościowej. Badane domknięcia mają być uwarunkowane przede wszystkim specyficznymi typami triad i ich nieodłączną niestałością. Przy takiej tezie najkorzystniej jest wyeliminować wszystkie inne czynniki mogące wpływać na zachowanie triad. Dlatego najlepiej, żeby badana sieć była jak najbardziej jednolita pod względem demograficznym. Grupa zbliżona geograficznie, wiekowo i kulturowo będzie nawiązywała między sobą nowe połączenia przede wszystkim w oparciu o połączenia już istniejące i głównie z przyczyn towarzyskich.

4.4 API i narzędzia

Naturalnie okolicznością ostatecznie rozstrzygającą przy o wyborze sieci do podjęcia badań jest techniczna możliwość realizacji zadania, również stopień jego skomplikowania. Najgorszym możliwym scenariuszem byłaby konieczność zautomatyzowanego przemierzania stron HTML z profilami użytkowników. Takie rozwiązanie jest najbardziej skomplikowane w implementacji, również najmniej wydajne. Na szczęście większość sieci społecznościowych udostępnia różnego rodzaju publiczne API. Do rozważenia pozostaje więc, czy zbiór zapytań przygotowanych przez dany serwis jest wystarczające, ale również czy nie narzuca zbyt silnych ograniczeń na ich liczbę.

Dostęp do Instagramu odbywa się przez współcześnie chyba najbardziej popularne API typu REST. W największym skrócie idea polega na unikalnym identyfikowaniu każdego udostępnianego zasobu przez pewien możliwie czytelny adres URL. Dostępne zapytania pokrywają w ogóle większość zasobów dostępnych dla użytkowników serwisu (rys. 4-7). Można więc pobierać podstawowe informacje o profilach (rys. 4-5), wyświetlać ostatnie zdjęcia, eksplorować zdjęcia podpisane danymi tagami lub w końcu sprawdzać listę połączeń wychodzących i wchodzących dla danego węzła (rys. 4-6).

Każde wysyłane zapytanie musi być podpisane kluczem autoryzacyjnym, do uzyskania dla każdej zarejestrowanej aplikacji, jaka ma korzystać z dostępu przez API. Dla wysyłanych zapytań wprowadzone jest ograniczenie w liczbie 5 tysięcy w przeciągu godziny. Przy przekroczeniu tego limitu zapowiedzianymi konsekwencjami jest blokowanie kolejnych zapytań, a nawet całkowita blokada klucza, spod którego były one wysyłane.



Rys. 4-5 Struktura odpowiedzi na zapytanie o podstawowe informacje o profilu. W postaci JSON przekazywane są używane w realizacji pracy: nazwa użytkownika i liczby połączeń wchodzących i wychodzących.

Rys. 4-6 Struktura odpowiedzi na zapytanie o listę połączeń wychodzących dla profilu. W postaci JSON przekazywana jest lista ID profili śledzonych przez danego użytkownika.

User Endpoints

GET /users/**user-id**
GET /users/self/feed
GET /users/**user-id**/media/recent
GET /users/self/media/liked
GET /users/search

Relationship Endpoints

GET /users/**user-id**/follows
GET /users/**user-id**/followed-by
GET /users/self/requested-by
GET /users/**user-id**/relationship
POST /users/**user-id**/relationship

Media Endpoints
Comment Endpoints
Like Endpoints
Tag Endpoints
Location Endpoints
Geography Endpoints

Rys. 4-7 Punkty dostępowe REST API udostępniane przez Instagram. Te wykorzystywane w realizacji pracy zaznaczono kolorem granatowym. Źródło [3]

4.5 Możliwości wykorzystania

W zamyśle autorów API nie jest więc ono przygotowane na masowe pobieranie danych przez aplikacje stron trzecich. Z drugiej strony poza narzuconymi ograniczeniami i zakazem przechowywania treści zamieszczanych przez użytkowników w regulaminie nie pojawiają się zapisy wykluczające długofalowe pobieranie i przechowywanie informacji o strukturze sieci. O ile takie działania nie są więc ułatwione, ich realizacja nie pozostaje w sprzeczności z regulaminem. Przy pewnym istotnym nakładzie pracy programistycznej realizacja zadania leży w zasięgu możliwości pojedynczego programisty.

5 Algorytmy przeszukiwania

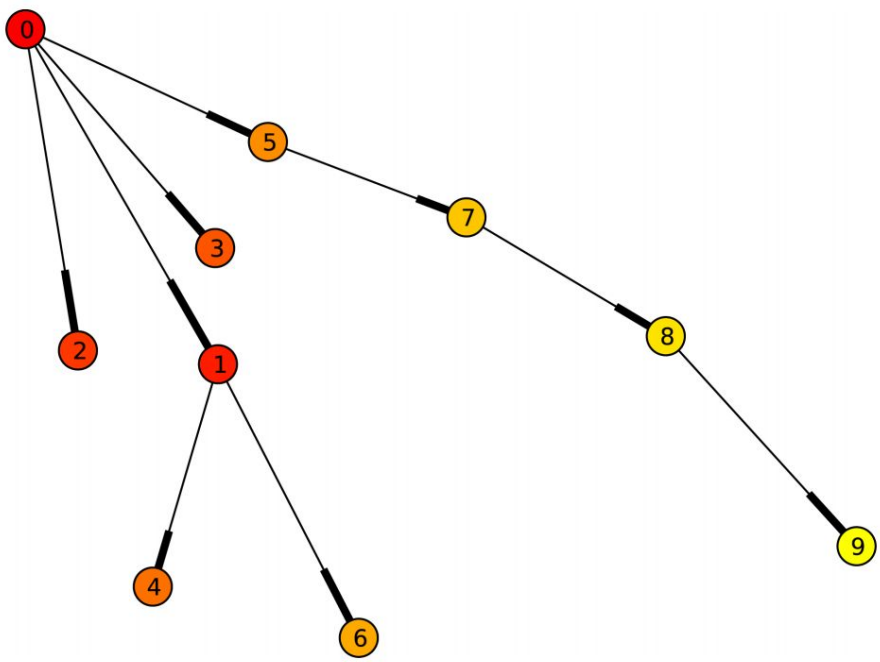
Jeśli ostatecznym wynikiem eksperymentu mają być wyniki badań dynamiki triad w sieci połączeń między użytkownikami, to ze względu na ograniczenia narzucone przez API i dostępną moc obliczeniową konieczne jest wyodrębnienie pewnego fragmentu grafu. Z jednej strony można by wybrać obserwowane węzły zupełnie losowo, co w sposób prowadziłoby do bardzo słabych wyników, biorąc pod uwagę liczbę użytkowników Instagramu. Jakie są zatem inne możliwości?

W sposób naturalny nasuwają się dwa podstawowe mechanizmy przeszukiwania grafów: przeszukiwanie wszerz (BFS) i w głąb (DFS).

5.1 Przeszukiwanie wszerz

Przeszukiwanie wszerz (rys. 5-1, algorytm 5-2) jest przeciwieństwem przeszukiwania w głąb, choć w gruncie rzeczy są bardzo podobne. W tym przypadku również przy wyborze węzłów do odwiedzenia jest również wyłącznie informacja o sąsiadach jak dotąd odwiedzonych węzłów, które nie zostały jeszcze odwiedzone. W przeciwieństwie do algorytmu DFC nowo odkryty węzeł nie zostanie odwiedzony od razu, a dopiero kiedy przyjdzie na niego kolej.

Zastosowania:
- odnajdowanie wszystkich połączonych węzłów w grafie,
- odnajdowanie najkrótszej ścieżki między węzłami,
- sprawdzanie dwudzielności grafu.



Rys. 5-1 Kolejność odkrywania węzłów podczas działania algorytmy przeszukiwania wszerz (BFS). Źródło [1], str. 31

```
zacznij w węźle n,
stwórz kolejkę Q,
oznacz n jako odwiedzony,
dodaj n do kolejki,
póki kolejka nie jest pusta:
      zdejmij n' z kolejki
      dla każdego sąsiada n' z kolejki:
            oznacz jako odwiedzony,
            dodaj do kolejki Q
```

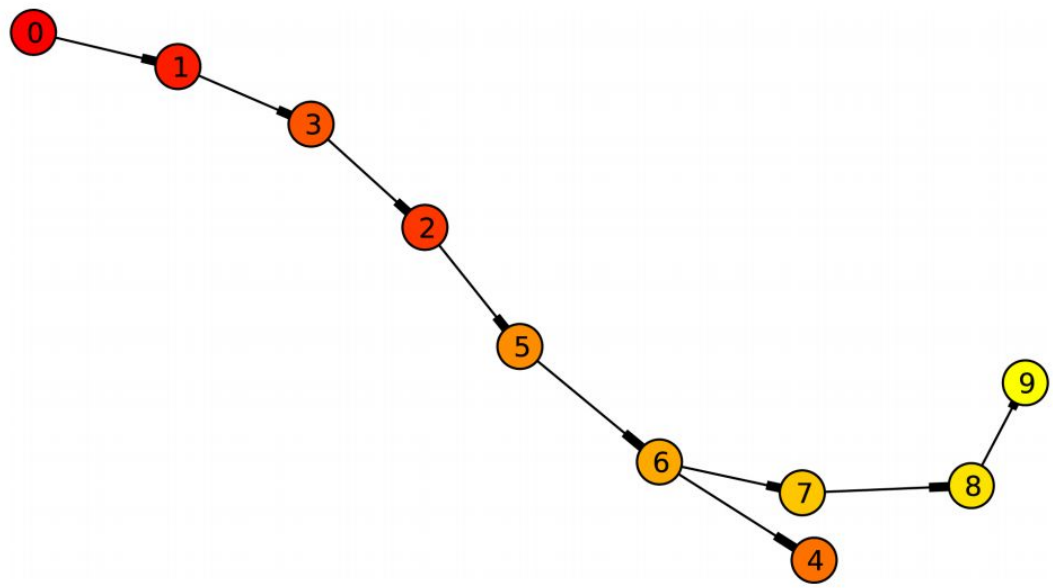
Algorytm 5-2 Skrócony opis algorytmu przeszukiwania wszerz (BFS)

5.2 Przeszukiwanie w głąb

Przeszukiwania w głąb (rys. 5-3, algorytm 5-4) przy poruszaniu się po grafie nie polega na żadnej informacji dodatkowej o odwiedzanych węzłach poza tym, jaki węzeł był ostatnio odwiedzony i do jakich kolejnych węzłów prowadzi. Sprowadza się do eksplorowania odkrytych ścieżek, dopóki jeszcze gdzieś prowadzą.

Zastosowania:

- znajdowanie spójnych składowych grafu,
- sprawdzanie istnienia ścieżki między dwoma węzłami grafu,
- spotykany w algorytmach brute force służących np. do przeszukiwania drzew gry.



Rys. 5-3 Kolejność odkrywania węzłów podczas działania algorytmy przeszukiwania w głąb (DFS). Źródło [1], str. 32

```
zacznij w węźle n,
oznacz n jako odwiedzony,
dla każdego nieodwiedzonego sąsiada n wykonaj rekurencyjnie
algorytm
```

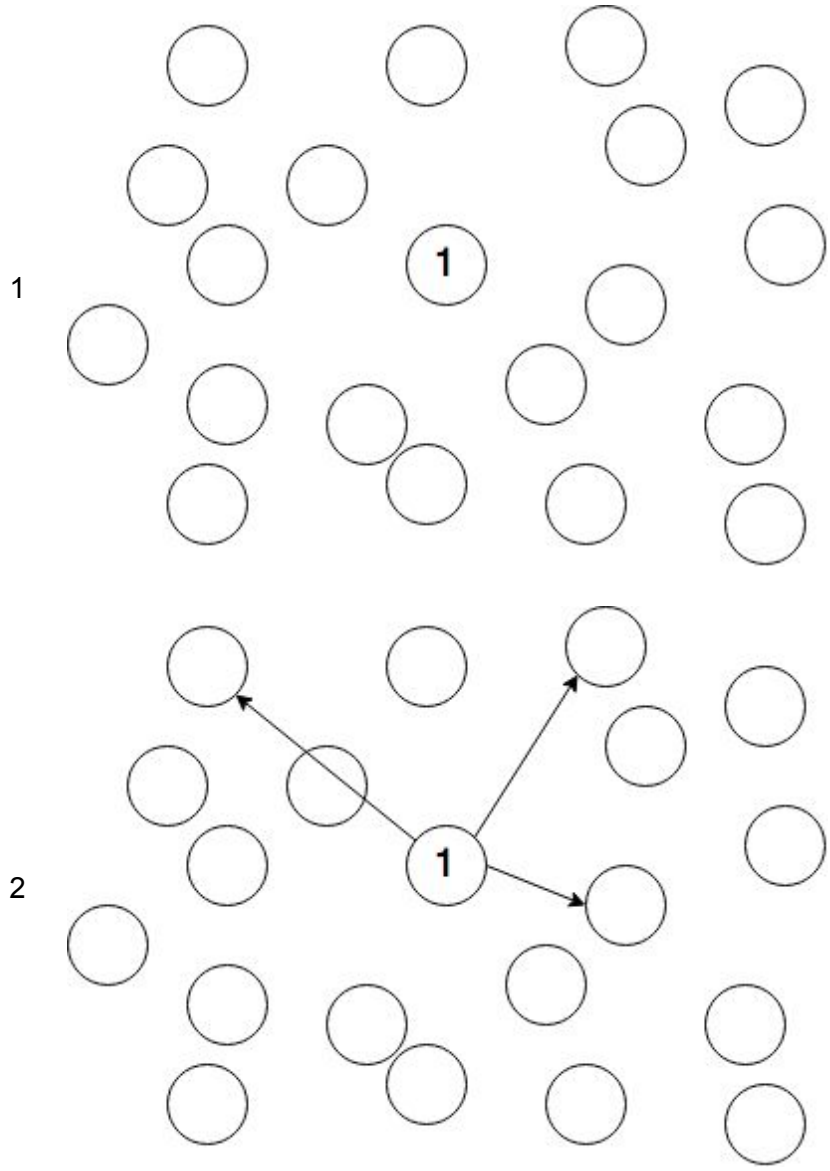
Algorytm 5-4 Skrótowy opis algorytmu przeszukiwania w głąb (DFS)

5.3 Możliwości zastosowania

Obydwa najpopularniejsze algorytmu przeszukiwania nie są jednak idealne dla postawionego zadania poszukiwania możliwe zagęszczonego fragmentu sieci społecznościowej. Nie biorą pod uwagę parametrów węzłów, ani liczonych względem całej sieci, ani względem węzłów już odwiedzonych. Zaproponowany i zrealizowany w dalszej części pracy algorytm stanowi pewną alternatywę dla poszukiwania wszerz, przy wyborze kolejnych węzłów do odwiedzenia biorącą jednak pod uwagę parametry z obu powyższych kategorii. Ten sposób postępowania i poruszania się po bardzo licznym grafie połączeń w sieci społecznościowej został zaprojektowany szczególnie z myślą o konkretnym zastosowaniu i nie zostały jak dotąd rozważone jego możliwe inne użycia.

6 Zaproponowany algorytm

Zaproponowany i zrealizowany algorytm (rys. 6-1 do 6-3, algorytm 6-4) wycinania możliwie reprezentatywnego fragmentu sieci społecznościowej o połączeniach skierowanych stanowi pewną modyfikację przeszukiwania wszerz, wzbogaconą o kryterium wyboru węzłów wartych odwiedzenia. W połączeniu z dodatkowymi modyfikacjami możliwymi dla specyficznej sieci badanej w tej pracy pozwolił na wydzielenie 50 tys. użytkowników zaliczających się przeważnie do podobnej grupy wiekowej i w większości zlokalizowanej w Warszawie, dzięki czemu połączonej dość zagęszczoną siecią połączeń i ciekawej pod kątem analizy dynamiki połączeń.



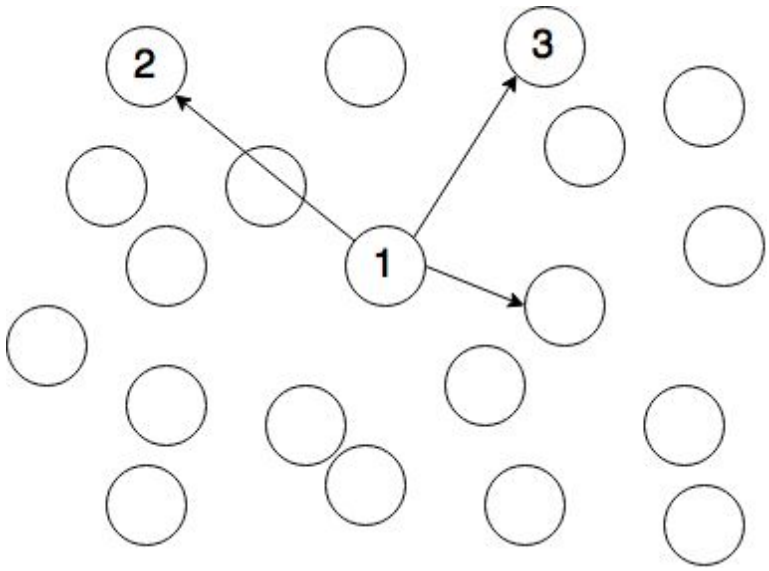
1 Dodanie węzła początkowego.

2 Dodanie połączeń wychodzących z węzła początkowego i wstawienie węzłów, do których prowadzą te połączenia, do kolekcji węzłów nieprzetworzonych.

Rys. 6-1 Pierwsze dwa kroki przebiegu algorytmu: dodanie i rozpoznanie połączeń wychodzących dla węzła początkowego.

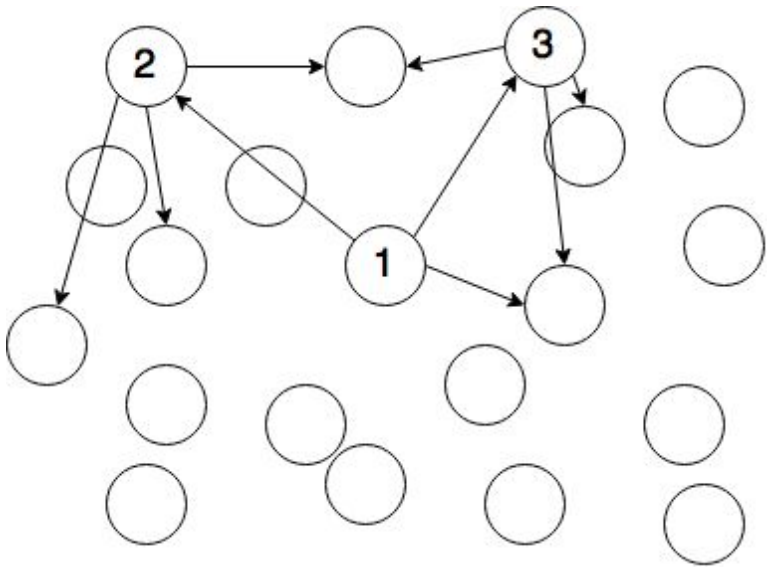
U podstaw działania algorytmu leży rekurencyjny sposób kolejkowania węzłów wartych odwiedzenia w najbliższej iteracji algorytmu. Jako miarę przyjmuje się stopień wejściowy węzłów już odkrytych, a jeszcze nie odwiedzonych, liczoną tylko z uwzględnieniem węzłów już odwiedzonych. Odczyt faktycznego stopnia wejściowego i wyjściowego wierzchołków pozwala na udoskonalenie algorytmu poprzez odrzucenie wierzchołków nieinteresujących w dalszej analizie, ale nie jest miarą, według której porządkuje się węzły do przetworzenia. Taki sposób przeszukiwania, wybierający do przetwarzania węzły najsilniej związane z już odwiedzonymi, prowadzi do stopniowego zagęszczania sieci połączeń w dotychczas wydzielonymi fragmencie. Kryterium stopu stanowi z kolei wyłącznie liczba przetworzonych węzłów.

3



Wstawienie do kolejki dwóch kolejnych węzłów do przetworzenia. Póki co wszystkie nieprzetworzone węzły mają stopień wejściowy 1, dlatego wybór tych dwóch węzłów jest losowy.

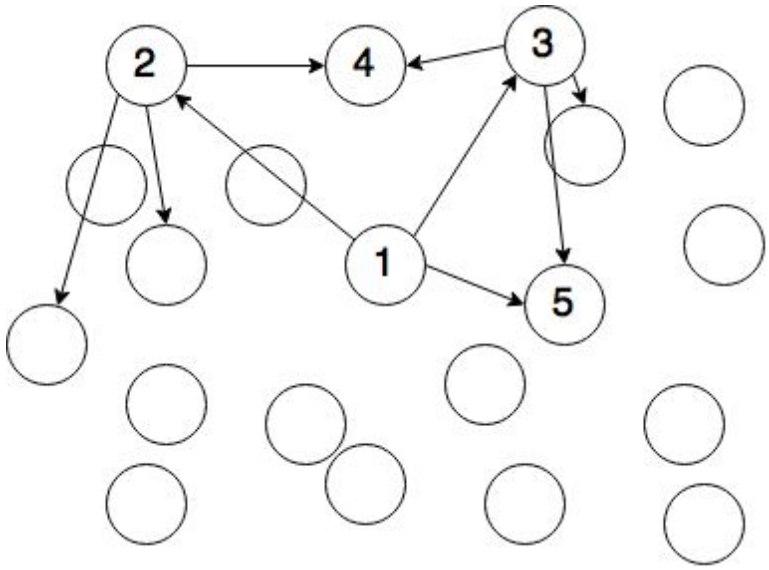
4



Dodanie połączeń wychodzących i węzłów, do których prowadzą te połączenia, dla dwóch węzłów z kolejki.

Rys. 6-2 Trzeci i czwarty krok algorytmu wycinania fragmentu sieci: sposób wyboru kolejnych węzłów do przetworzenia.

5



Załadowanie do kolejki kolejnych dwóch węzłów do odwiedzenia. Kryterium wyboru stanowi tak jak wcześniej stopień wejściowy liczony na podstawie rozpoznanych jak dotąd połączeń. Tym razem pojawia się już rozróżnienie na mocy tego kryterium.

Rys. 6-3 Piąty krok algorytmu wycinania fragmentu sieci: kolejna iteracja wyboru węzłów do przetworzenia.

```
wstaw węzeł początkowy do kolejki
póki liczba węzłów poniżej n:
      odwiedź wszystkie węzły z kolejki
      uzupełnij kolejkę

odwiedź węzeł:
      sprawdź, czy węzeł ma być przetwarzany
      dodaj do grafu wszystkie węzły śledzone przez węzeł
      dodaj wszystkie połączenia o zwrocie od węzła

uzupełnij kolejkę:
      posortuj węzły nieodwiedzone wg liczby śledzących
odwiedzonych węzłów
      zapełnij kolejkę pierwszymi tak posortowanymi węzłami
```

Algorytm 6-4 Skrócony zapis działania zaproponowanego algorytmu

6.1 Zasada działania

Przebieg algorytmu zaczyna się od wyboru jednego węzła początkowego. Jest on przetwarzany, tak samo jak każdy następny węzeł aż do wystąpienia kryterium stopu, poprzez dodanie do bazy węzłów jeszcze nieprzetworzonych wszystkich węzłów, do których prowadzą połączenia skierowane od przetwarzanego węzła. Już na tym etapie korzystnie jest rozpoznać wśród nich węzły niekorzystne do przyszłego przetwarzania, o czym będzie mowa później. W pierwszym przebiegu algorytmu niemożliwe jest rozróżnienie między rozpoznanymi a nie przetworzonymi na podstawie stopnia wejściowego liczone na podstawie już przetworzonych węzłów, bo taki był dotąd tylko jeden, wszyscy kandydaci do przetwarzania mają więc taki stopień wejściowy równy 1. Pierwsza kolejka węzłów do przetworzenia powstaje więc w sposób losowy.

W drugim przebiegu każdy węzeł z kolejki jest przetwarzany tak samo jak pierwszy węzeł w algorytmie. Wszystkie węzły, do których prowadzą połączenia wychodzące z procesowanego węzła, zostają dodane do bazy kandydatów do przetwarzania. Jak łatwo się domyślić, kiedy sieć, na której działamy nie jest reprezentowana przez graf zupełny, po drugim przebiegu nie wszystkie dodane do bazy węzły mają taki sam stopień wejściowy liczony na podstawie już przetworzonych węzłów. Dla niektórych kandydatów do przetwarzania wartość ta wynosi 2, dla pozostałych 1.

6.2 Warunek końcowy

W kolejnych przebiegach algorytmu rośnie rozbieżność stopnia wejściowego liczonego na podstawie węzłów już przetworzonych, przy jednoczesnym wzroście do pewnego momentu maksymalnej występującej jego wartości. Świadczy to o zagęszczaniu się sieci połączeń między już przetworzonymi węzłami, co jest pożądane i świadczy o sukcesie algorytmu.

O szybkości zbiegania algorytmu do możliwie gęstej sieci połączeń decyduje szereg parametrów. Manipulacja ich wartościami, nawet w trakcie działania algorytmu, może korzystnie wpływać na uzyskane rezultaty.

6.3 Dostrajanie

Pierwszym takim parametrem jest maksymalna długość kolejki węzłów do przetworzenia. W początkowych przebiegach algorytmu korzystne okazuje się skrócenie jej w stopniu prowadzącym do szybszego zwiększenia wariancji stopni wejściowych liczonych względem już przetworzonych węzłów. W przeciwnym wypadku, przy zbyt dużej wartości parametru, ryzykuje się niekontrolowany rozrost grafu już przetworzonych węzłów, który będzie trudny do zagęszczenia o wszystkie występującego w jego ramach połączenia.

Pozostałe parametry wpływają na to, jakie węzły zostaną z góry rozpoznane jako nieinteresujące przy dalszym przetwarzaniu. Zalicza się do nich węzły o zbyt dużym rzeczywistym, czyli liczonym względem całej sieci, a nie jak dotąd przetworzonych węzłów, stopniem wejściowym lub wyjściowym. Zbyt niski absolutny stopień wyjściowy jest również nieinteresujący.

6.4 Węzły - celebryci

Węzły o absolutnym stopniu wejściowym przekraczającym ustaloną wartość progową reprezentują z jednej strony przedsiębiorstwa prowadzące za pośrednictwem sieci społecznościowej działania marketingowa, a z drugiej osoby znane w skali wyższej niż powiedzmy pewne środowisko znajomych. W obu przypadkach identyfikacja jest banalna - niezwykle rzadkim jest, aby jedna osoba utrzymywała kontakty towarzyskie, a co za tym idzie miała profil śledzony przez tysiące osób. Dlatego odrzucenie takich profili wyłącznie na podstawie absolutnego stopnia wejściowego jest w pełni uzasadnione. To, że takie węzły są nieinteresujące dla zaplanowanej analizy, wynika z ich funkcjonowania w pewien sposób ponad lub poza siecią społeczną, połączenia skierowane do nich nie stanowią w najniższym stopniu odzwierciedlenia relacji międzyludzkich. Dodatkowo uwzględnienie ich podczas działania algorytmu byłoby zabójcze dla skuteczności algorytmu. Skoro wybór kolejnych węzłów do przetworzenia kierowany jest stopniem wejściowych liczonym na podstawie póki co przetworzonych węzłów, nie trzeba by długo czekać, aby w kolejce znalazł się piosenkarz lub piłkarz z innego kontynentu, poszerzając tym samym listę przetwarzanych węzłów o swoich znajomych.

6.5 Pozostałe sposoby filtrowania

Stopień wyjściowy węzła powyżej ustalonej wartości progowej również oznacza węzeł nieinteresujący przy analizie. Takie profile prowadzą osoby śledzące znacznie więcej osób, niż faktycznie znają. Podobnie jak w przypadku celebrytów i przedsiębiorstw, dla absolutnej wartości stopnia wyjściowego wierzchołka nietrudno wyznaczyć wartość graniczną. Bardzo rzadko obserwowanie 500 lub więcej profili będzie odzwierciedleniem relacji występujących w prawdziwym życiu. Również analogicznie do wcześniej omawianej grupy przetwarzania takich węzłów jako wartych analizy prowadziłoby znacznego pogorszenia własności algorytmu. Do bazy węzłów wartych przetworzenia dodawane by były węzły obserwowane tylko dla samego obserwowania, a w rzeczywistości zupełnie niezwiązane z powstającym wycinkiem sieci.

Ostatnim kryterium odrzucenia węzłów niewartych przetwarzania jest zbyt niski, a dokładnie zerowy absolutny stopień wyjściowy węzła. Dotyczy to osób niekorzystających faktycznie z serwisu społecznościowego, a co za tym idzie nieodpowiadających przyjętej założeniom o przełożeniu relacji ze świata rzeczywistego na kształt sieci połączeń w obserwowanej sieci.

6.6 Sposób powstania algorytmu

Do ostatecznego kształtu algorytmu wraz z modyfikacjami doprowadził szereg prób budowy gęstego wycinka sieci społecznościowej. Parametry sieci nie były badane inaczej niż przez sprawdzenie rezultatów dalszej analizy dynamiki sieci. Można powiedzieć, że propozycja potraktowania stopnia wejściowego w skali dotychczas przetworzonych węzłów jako kryterium wartościującego wywodzi się z intuicji i dostępnych narzędzi budowania sieci społecznościowej, to znaczy pobiegania połączeń wchodzących i wychodzących z przetwarzanego węzła.

7 Śledzenie dynamiki triad

Ostatecznym celem pracy było przedstawienie wyników badania dynamiki triad w wycinku sieci społecznościowej, a więc obserwacja w dłuższym przedziale czasu powstawania nowych połączeń między użytkownikami.

Aby było to możliwe, najpierw należało wyznaczyć zbiór wierzchołków do obserwacji w dalszych krokach. W tym celu został zaproponowany algorytm znajdujący możliwie korzystny dla dalszej analizy podgraf. Przez korzystny rozumiany jest zbiór możliwie zagęszczonej sieci połączeń, w domyśle stanowiący odzwierciedlenie pewnej stosunkowo licznej grupy społecznej o podobnej przeważającej lokalizacji geograficznej. Takie warunki umożliwiają wyznaczenie największej liczby interesujących triad, w pierwszym, jak i w kolejnych krokach badania dynamiki sieci.

7.1 Analizowane dane

Pierwszy krok, czyli wycięcie fragmentu sieci do dalszej analizy, ma dostarczyć zatem zbioru wierzchołków, który pozostanie niezmienny w ramach upływu czasu zbierania wyników, co może trwać dowolnie od kilku tygodni to kilku miesięcy czy nawet lat. Elementem zmiennym w czasie jest z kolei zbiór krawędzi z obserwowanym grafie. Wraz z upływem czasu oczekuje się istotnego wzrostu ich liczby. Między użytkownikami mają powstawać nowe połączenia, a co za tym idzie zupełnie nowe, lub domknięte względem wcześniejszych triady.

Kolejne odległe w czasie przebiegi służą więc do zachowywania swego rodzaju migawek obecnego stanu sieci. W każdym kolejnym uruchomieniu tak jak w pierwszym sprawdzane są wszystkie połączenia w ramach analizowanego zbioru wierzchołków, a do lokalnej bazy krawędzi są dopisywane nowe, które pojawiały się od poprzedniego przetwarzania. Dodanie do każdej krawędzi stempla czasowego reprezentującego pierwszy moment, kiedy została zaobserwowana, pozwala w dużym przybliżeniu i przy dużej niedokładności czynienie pewnych obserwacji o szybkości domykania się triad, zwłaszcza tam, gdzie podstawy teoretyczne sugerują, że jest to spodziewane.

Danymi na stałe przechowywanymi lokalnie są więc: zbiór wierzchołków i zbiór krawędzi wraz ze stemplami czasowymi.

Przetwarzanie tych dwóch zbiorów ma prowadzić do wyznaczenia wszystkich triad wszystkich typów występujących w badanym wycinku sieci. Nie dla wszystkich typów triad jest to możliwe, niektóre z nich są na tyle liczne, że ich uwzględnienie w repozytorium triad nie niosłoby żadnej informacji, fatalnie wpływając na wydajność pozostałych kroków eksperymentu. Dlatego po pobraniu każdej migawki-aktualizacji stanu połączeń między obserwowanymi profilami warto wyznaczyć tylko nowe triady wnoszące jakąś informację.

7.2 Jak badać dynamikę sieci?

Dodanie samych triad pozwala już na przedstawienie wyników w formie zestawienia liczebności poszczególnych ich typów, naturalnie w ramach zbioru triad wartych analizy. Dopiero kolejne kroki prowadzą jednak do rozpoznania ciekawszych procesów, począwszy od niestabilności niektórych triad, a na rozkładach prawdopodobieństwa czasu trwania konkretnych przejść skończywszy.

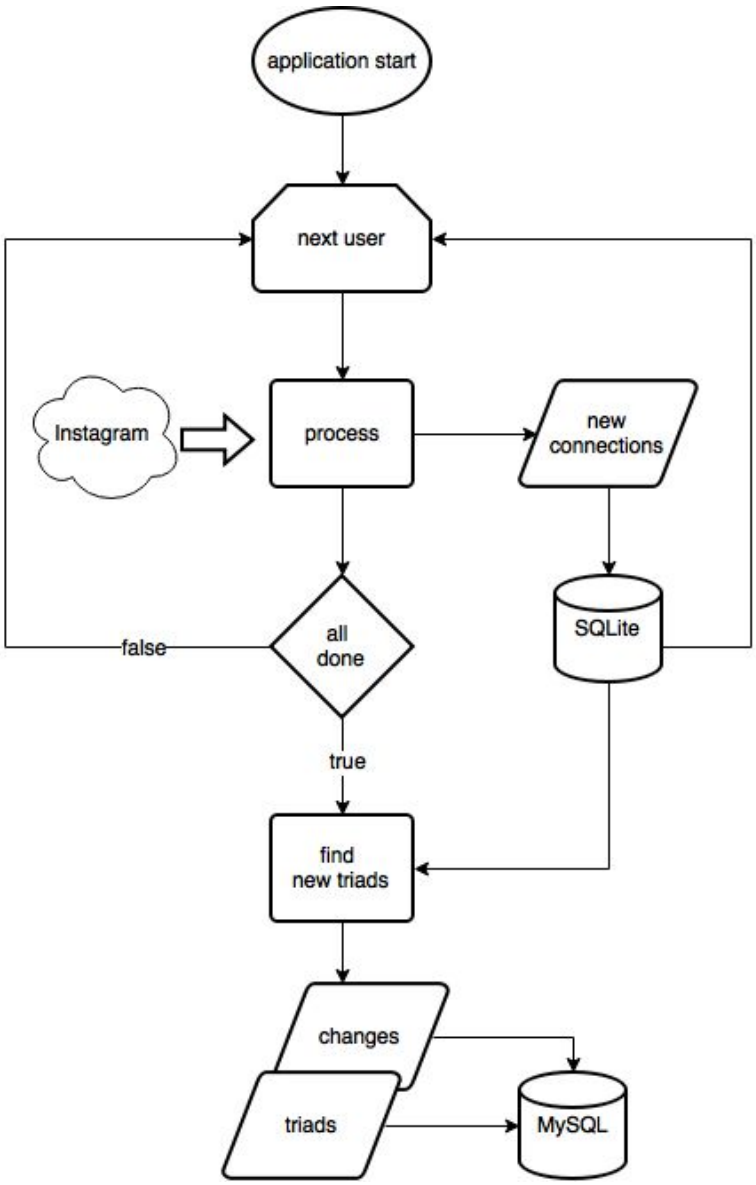
Rozpoznawanie nowych triad, czyli ich poszukiwanie w ramach połączeń o stemplu czasowych powyżej początku ostatniego uruchomienia, prowadzi do dodania do repozytorium pewnej liczby nowych nowych triad. Ponieważ każda jest reprezentowana przez zestaw połączeń między węzłami, łatwo rozpoznać wszystkie nowo dodane i poddać je dalszemu przetwarzaniu. Dla każdej nowej triady jest poszukiwana albo wcześniej istniejąca triada między trzema danymi węzłami, albo zestaw połączeń między tymi trzema węzłami sprzed ostatniego przebiegu. W tym momencie, przed zapisem przejść, identyfikowane i dodawane do repozytorium są triady wcześniej uznawane za nieistotne i zbyt liczne do przechowywania, a teraz potrzebne do reprezentacji poprzedniego stanu triady. Tylko w ten sposób uzyskana zostaje reprezentacja pełnego procesu domykania się triady, od pierwszego przebiegu aż do stanu aktualnego, konieczna przy dalszej analizie.

Efektem końcowym uruchomienia jest więc zbiór nowych połączeń, zbiór nowych triad i zbiór nowych przejść między triadami, wyznaczonych dla każdej nowej triady.

8 Implementacja śledzenia dynamiki triad

Uruchomienie programu w trybie śledzenia dynamiki sieci ma następować cyklicznie w kilkudniowych odstępach czasu. Odbywa się, gdy zbiór węzłów do śledzenia jest już gotowy i ustalony. Nie będzie zmieniał się przy kolejnych uruchomieniach.

Wykonanie (rys. 8-1, algorytm 8-2) zaczyna się od pobrania i zapisu w bazie SQLite wszystkich nowych połączeń między użytkownikami. Następnie wyszukuje się triad powstałych w związku z pojawieniem się nowych połączeń. Wszystkie one zapisywane są w bazie MySQL wraz z encjami reprezentującymi ich rodowód, czyli jakie domknięcie spowodowało powstanie danej triady.



Rys. 8-1 Schemat blokowy wykonania aplikacji śledzącej zmiany we fragmencie sieci.

```
dodaj nowe połączenia do bazy
zapisz do bazy nowe triady

odwiedź każdą nową triadę:
      spróbuj sklasyfikować poprzednika triady
      jeśli się uda, dodaj poprzednik do bazy
      jeśli nie, znajdź poprzednik w bazie
      zapisz przejście

zapisz czas zakończenia obiegu programu
```

Algorytm 8-2 Skrócony zapis kolejnych kroków pełnego wywołania aplikacji śledzącego dynamikę fragmentu sieci społecznościowej.

8.1 Migawki stanu sieci

Ponieważ Rest API udostępniane przez Instagram nie daje dostępu do czasów pojawiania się połączeń, kolejnych migawek stanu badanego fragmentu sieci nie można pobierać w sposób przyrostowy. Zamiast tego za każdym razem konieczne jest pobranie informacji o wszystkich co do jednego połączeniach obecnych w tej sieci, a zapisanie do lokalnego repozytorium połączeń wyłącznie nowych, nie rozpoznanych przy poprzednim przebiegu programu, odpowiednio oznaczając je aktualnym stemplem czasowym.

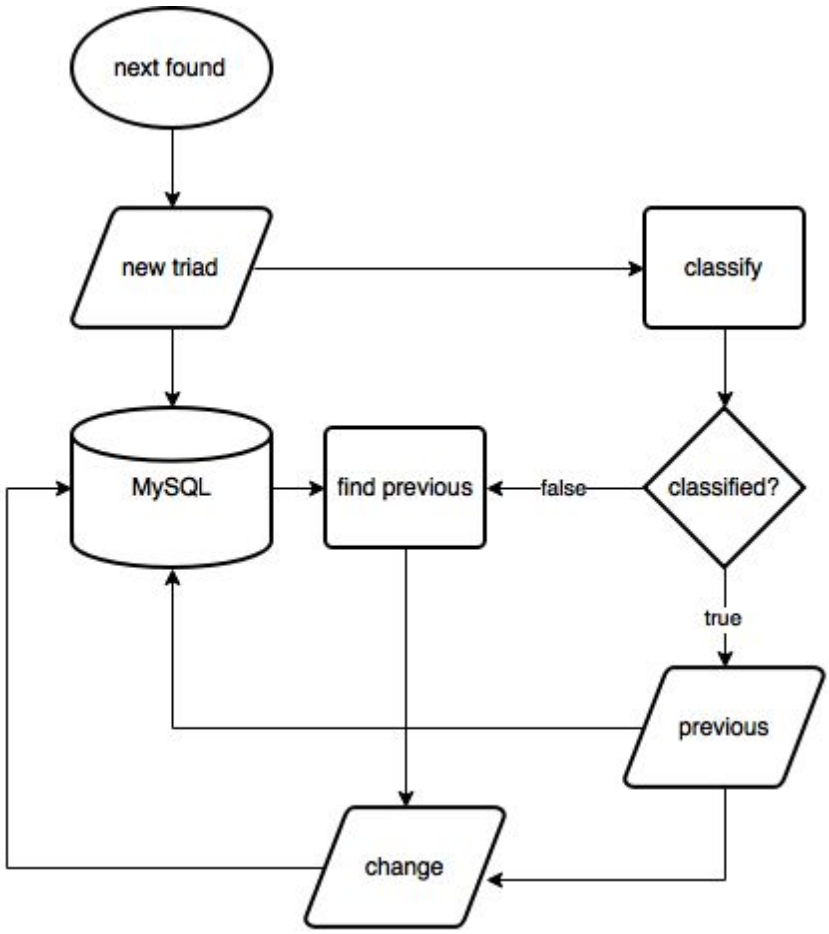
Ta część wykonania w tym trybie jest najbardziej czasochłonna, jej szybkość ogranicza bowiem dostępna liczba zapytań wysyłanych przez godzinę - 5 tysięcy. Tak jak przy wykonaniu wyboru węzłów do dalszej analizy, pobranie informacji o jednym węźle (algorytm 8-3) jest przeważnie bardziej kosztowne, niż byłoby w przypadku jednego wystarczającego zapytania i odpowiedzi.

```
pobierz węzeł z kolejki

odwiedź węzeł:
      pobierz typ węzła
      jeśli zmienił się na prywatny, zapisz informację w bazie
      pobierz listę wszystkich połączeń wychodzących
      zapisz w bazie wszystkie nowe połączenia
```

Algorytm 8-3 Skrócony zapis sposobu przetwarzania śledzonego węzła.

8.2 Odkrywanie triad

Odkrywanie triad (rys. 8-4) polega na wykonaniu reprezentujących je, przeważnie dosyć złożonych zapytań SQL (listing 8-5), zwracających trójki węzłów tworzących triady danych typów. Od razu w treści zapytań brane jest pod uwagę, by chociaż jedno z połączeń w ramach triady należało do nowo dodanych. W ten sposób najłatwiej wyłuskać tylko najnowsze triady, które wymagają dodania do repozytorium i dalszego przetwarzania.



Rys. 8-4 Schemat blokowy przetwarzania nowych triad rozpoznanych w wywołaniu aplikacji śledzącej dynamikę połączeń w sieci. Po próbie programowego sklasyfikowania poprzednika triady, które wynik zwraca tylko dla typów nieprzechowywanych w bazie, albo poprzednik zostaje dodany do bazy, albo wyszukany wśród triad funkcjonujących w bazie. Na koniec do bazy trafia zapis przejścia między triadami.

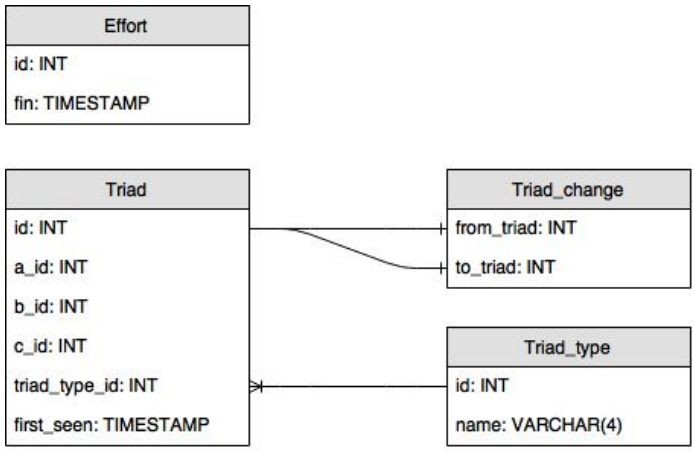
```
SELECT ab.follower_id AS a_id,
       bc.follower_id AS b_id,
       ca.follower_id AS c_id
           FROM following AS ab
LEFT OUTER JOIN following AS ba ON ab.follower_id = ba.followee_id
                               AND ab.followee_id = ba.follower_id
           JOIN following AS bc ON bc.follower_id = ab.followee_id
LEFT OUTER JOIN following AS cb ON bc.follower_id = cb.followee_id
                               AND bc.followee_id = cb.follower_id
           JOIN following AS ca ON bc.followee_id = ca.follower_id
                               AND ab.follower_id = ca.followee_id
LEFT OUTER JOIN following AS ac ON ca.follower_id = ac.followee_id
                               AND ca.followee_id = ac.follower_id
WHERE a_id < b_id
 AND ( b_id < c_id
    OR a_id < c_id )
 AND ba.follower_id IS NULL
 AND cb.follower_id IS NULL
 AND ac.follower_id IS NULL
 AND (   ab.first_seen > (SELECT max(fin) FROM effort)
      OR bc.first_seen > (SELECT max(fin) FROM effort)
      OR ca.first_seen > (SELECT max(fin) FROM effort)
     )
```

Listing 8-5 Zapytanie SQL wykonywane na bazie użytkowników i połączeń między nimi zwracające wszystkie nowe triady typu 030C. Ostatnie linie porównuję stemple czasowe połączeń w triadzie z czasem zakończenia poprzedniego wykonania programu.

8.3 Zapis triad do bazy

Zapis triad do bazy to kolejna operacja wymagająca czasu, ze względu na liczbę wierszy do dodania do tabel. Dla wyszczególnionego zbioru 50 tysięcy użytkowników po pierwszym poszukiwaniu triad w sieci ich liczba przekroczyła 30 milionów. Przy takiej liczebności danych na znaczne przyspieszenie zapisu pozwala użycie mechanizmów *bulk insert*. W tym celu dane do wpisania do bazy są dzielone na paczki o ustalonym maksymalnym rozmiarze i zapisywane do bazy grupowo. Wydzielenie paczek, a nie na przykład jednej paczki ze wszystkimi triadami danego typu, pozwala na monitorowania postępów zapisu, który przy tej skali może potrwać i ponad godzinę, poprzez logowanie liczby dotychczasowych zakończonych powodzeniem zapisów pomiędzy przetwarzaniem kolejnych paczek.

Po zakończeniu zapisów triad wszystkich typów do bazy (schemat na rys. 8-6) kolejny krok ma doprowadzić do zidentyfikowania przejść między triadami, jakie doprowadziły do ich powstania. Innymi słowy dla każdej nowo odkrytej i zapisanej triady ma zostać znaleziona triada, która istniała przed nią.



Rys. 8-6 Schemat bazy triad i przejść między nimi.

W procesie rozpoznawania triad na podstawie nowo dodanych połączeń nie są brane pod uwagę wszystkie możliwe triady występujące w spisie długości 16 pozycji. Niektóre z nich występowały by zbyt licznie i ze względu na małą liczbę połączeń są nieistotne. Ich znaczenie wzrasta dopiero przy poszukiwaniu triad poprzednich względem właśnie zidentyfikowanych. Rozwiązaniu tej sytuacji służy implementacja klasyfikatora (listing 8-7) poszukującego w momencie zapytania o poprzedzającą triadę przede wszystkim wśród połączeń występujących przed ostatnią aktualizacją połączeń między śledzonymi węzłami, czy dana trójka węzłów nie tworzyła którejś z triad nierozpoznawanych przez zapytania SQL. Dopiero jeśli taki poprzednik nie zostanie zidentyfikowany, poszukuje się go wśród wcześniej rozpoznanych i zapisanych w bazie triad.

```
@staticmethod
def classify(followings):
    if len(followings) == 0:
        return "003"
    if len(followings) == 1:
        return "012"
    if len(followings) != 2:
        return None
    if followings[0][1] == followings[1][0]:
        if followings[0][0] == followings[1][1]:
            return "102"
        else:
            return "021C"
    if followings[0][0] == followings[1][1]:
        if followings[0][1] == followings[1][0]:
            return "102"
        else:
            return "021C"
    if followings[0][0] == followings[1][0]:
        return "021D"
    if followings[0][1] == followings[1][1]:
        return "021U"
    else:
        return None
```

Listing 8-7 Fragment implementacji realizujący funkcję klasyfikatora poprzedników triad. Wszystkie widoczne typy nie zaliczają się do wyszukiwanych zapytaniami SQL, ponieważ są zbyt liczne i przenoszą niewiele informacji.

8.4 Odkrywanie domknięć

Wszystkie rozpoznane przejścia między triadami gromadzone są jako oddzielne encje. Można uznać je za drugie pochodne węzłów i połączeń między nimi stemplowanych czasowo, gdzie pierwszą pochodną są rozpoznane triady. Ich zapis i przechowywanie w bazie sprzyja dalszej analizie zjawisk występujących między triadami w badanej sieci w skali dłuższego czasu eksperymentu. Kolejne zapytania można już wykonywać na tabelach triad i przejść między nimi, zyskując w ten sposób znaczny skok wydajności.

8.5 Moduły implementacji

Wszystkie operacje na bazie MySQL związane ze śledzeniem dynamiki sieci zostały zamknięte w osobnym odpowiedzialnym za to module, co sprzyja czytelności samej logiki zawartej w implementacji mechanizmów i algorytmów. Do tego modułu przekazywane jest jedno połączenie do bazy, co z miejsca eliminuje problemy związane z wielodostępem, aplikacja jest ogółem rzecz biorąc stricte jednowątkowa. Wprowadzenie większej liczby wątków nie wprowadzałoby zwiększenia wydajności, ta jest bowiem ograniczona z jednej strony dostępnym limitem zapytań do API, z drugiej zaś szybkością masowych zapisów do bazy danych.

8.6 Kontrola kolejnych wykonań aplikacji

W śledzeniu dynamiki sieci istotne okazuje się wprowadzenie rozróżnienia między kolejnymi obiegami algorytmu, czyli kolejnymi, przeważnie oddalonymi o parę dni w czasie, uruchomieniami implementacji. Najbardziej oczywistym, prostym w implementacji i niezawodnym rozwiązaniem było wprowadzenie dodatkowej tabeli z wierszami odpowiadającymi czasom zakończenia kolejnych uruchomień implementacji. Przed każdym zakończeniem pełnego obiegu algorytmu do tej tabeli wstawiany jest zatem nowy wiersz.

Tym sposobem przykładowo wyróżnienie nowych połączeń, dodanych w obecnym obiegu algorytmu, które nie istniały w bazie jeszcze przed zakończeniem poprzedniego obiegu, sprowadza się do porównania dla wszystkich wpisów w tabeli połączeń ich stempli czasowych ze stemplem czasowym wiersza oznaczającego ostatnie zakończenie pełnego wykonania implementacji.

Tak jak przy wyborze fragmentu sieci, i wykonanie mające na celu wykonanie kolejnej migawki zmieniającej się sieci połączeń trwa długo, dla sieci z 50 tysięcy węzłów prawie dwie doby, dlatego i również w tym przypadku odpowiednie logowanie postępów w przetwarzaniu węzłów jest jedynym sposobem na monitorowanie na bieżąco postępów czynionych przez aplikację. Dla kroku aktualizacji sieci połączeń wpisy w logu programu są identyczne jak dla wycinania fragmentu sieci. Z kolei po przejściu do zapisu nowych triad, dzięki podziałowi dużych liczb nowych triad na paczki danych w efekcie co kilka minut w logu pojawia się wpis o zakończonym sukcesem zapisie pewnej liczby danych. Ostatni krok, czyli wyznaczanie domknięć, jakie zaszły w związku z pojawieniem się nowych triad, jest możliwy do monitorowania poprzez śledzenie liczby zidentyfikowany przejść z triad wcześniej już zidentyfikowanych podawanej oddzielnie względem przejść, dla których poprzednik nie znajdował się w bazie jako nieinteresujący, a w obecnej sytuacji zostaje ona o niego uzupełniona.

9 Wyniki

Na efekt końcowy badań składają się wszystkie zgromadzone dane o dynamice obserwowanego fragmentu sieci wraz z danymi pochodnymi względem nich, czyli statystykami dotyczącymi liczebności triad i przejść między nimi.

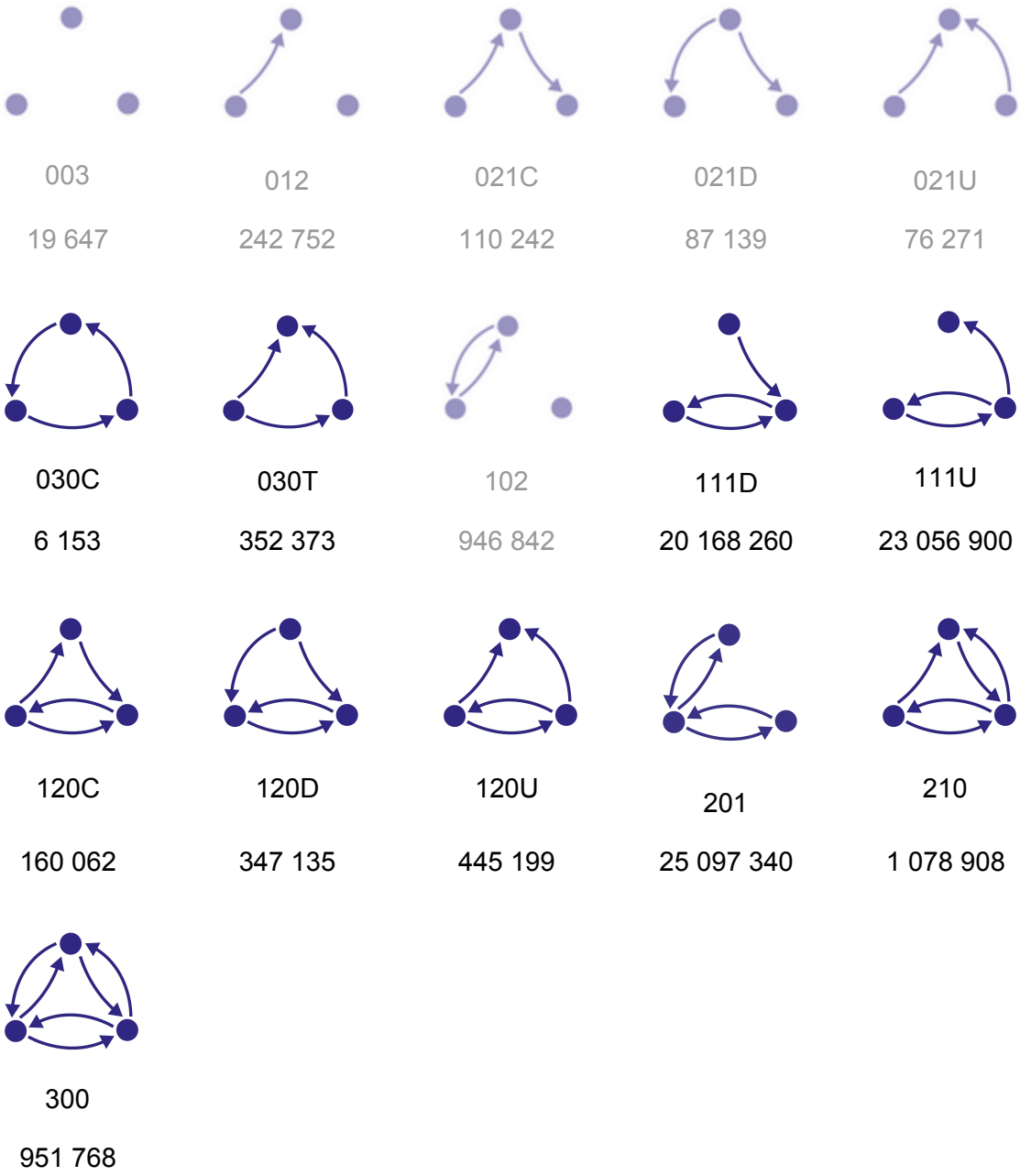
W procesie budowania fragmentu sieci do analizy, czyli iteracyjnego doboru węzłów zgodnie z zaprojektowanym i dostrojonym algorytmem, uzyskana została sieć ponad 80 tysięcy użytkowników o przeciętych stopniach wejściowych i wyjściowych. Do takiego stanu doprowadziło odrzucenie już na poziomie budowania sieci węzłów o niekorzystnych właściwościach.

Przy wykonywaniu kolejnych migawek obserwowanej sieci zaobserwowane ponad 70 milionów triad oraz prawie dwa miliony przejść między nimi. Dane ilościowej dotyczące zebranych statystyk zestawiono w tabeli 9-1 Należy pamiętać, że nie wszystkie typy triad zaliczały się do obserwowanych. Typy zbyt liczne i nieuznawane w istocie za triady, były gromadzone jedynie jako poprzedniki nowych triad pojawiających się w sieci na podstawie wykonywania kolejnych migawek stanu sieci.

Liczba użytkowników	82 708
Liczba połączeń	2 199 726
Liczba triad	73 146 991
Liczba domknięć	1 809 155

Tabela 9-1 Ogólne statystki ilościowe opisujące dane zgromadzone i przechowywane w toku realizacji niniejszej pracy

9.2 Statystyki liczebności triad



Rys. 9-2 Statystyki liczebności szestanstu typów triad. Wyszarzono typy niezliczane w całości.

9.3 Graficzna prezentacja domknięć

Danymi pochodnymi względem samego zestawienia połączeń i liczebności triad są statystyki dotyczące obserwowanych przejść, czyli domknięć, między poszczególnymi typami triad.

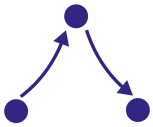
Końcowy, przedstawiony poniżej rezultat i wynik badań prowadzonych w ramach pracy, stanowi zestawienie uzyskanych statystyk co do przejść między typami triad. Na rysunkach od 9.3 do 9.8 połączone zostały trzy reprezentacje: graficzna, procentowa i liczbowa, tak aby analiza wyników była łatwiejsza i bardziej intuicyjna.

Dla każdego typu triady, który został sklasyfikowany jako poprzednik dla nowych triad, zliczone zostały typy triad, jakie po nim następowały. Typ stanowiący poprzednik znajduje się w pierwszym wierszu każdego elementu zestawienia.

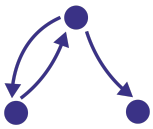
Dla każdego typu następującego po danym poprzedniku została przygotowana reprezentacja graficzna odpowiadająca poprzednikowi, tak aby z miejsca widoczne było, jaka krawędź pojawiła się w triadach typu poprzednika, co doprowadziło do domknięcia do innego typu.

Dla każdego domknięcia obserwowanego po danym poprzedniku przedstawiono procentowy udział danego typu wśród następników. Co istotne, pominięte zostały typy triad, dla których ta wartość procentowa była ułamkowa, uznając je za nieistotne.

Pod każdą wartością procentową przedstawiono reprezentowaną przez nią wartość liczbową, dzięki czemu łatwiejsza jest orientacja, ze statystykami jakiej skali mamy do czynienia.

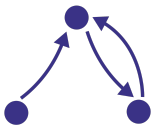


021C



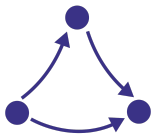
111U

54.27%
(59 831)



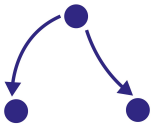
111D

42.54%
(46 898)

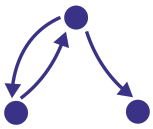


030T

2.22%
(2 449)

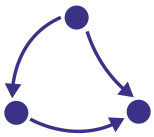


021D



111U

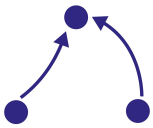
95.17%
(82 926)



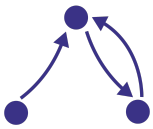
030T

2.82%
(2 461)

Rys 9-3 Statystyki domknięć triad typów 021C i 021D

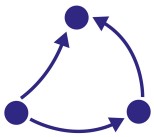


021U



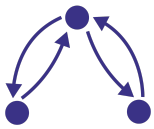
111D

90.82%
(69 268)



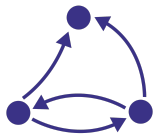
030T

4.40%
(3 355)



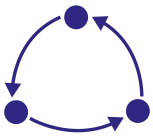
201

3.46%
(2 636)

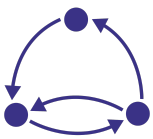


120U

1.24%
(945)

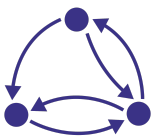


030C



120C

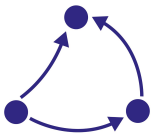
97.19%
(173)



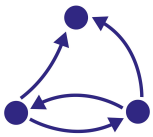
210

2.81%
(5)

Rys 9-4 Statystyki domknięć triad typów 021U i 030C

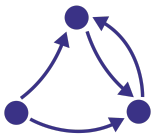


030T



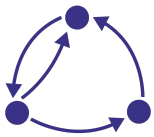
120U

50.04%
(2 444)



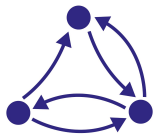
120D

25.86%
(1 263)



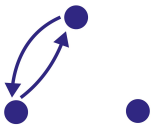
120C

18.28%
(893)

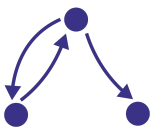


210

5.63%
(275)

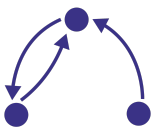


102



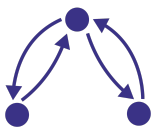
111U

36.45%
(345 079)



111D

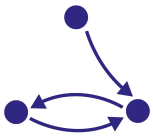
33.67%
(318 843)



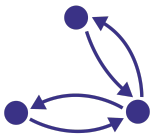
201

29.34%
(277 829)

Rys 9-5 Statystyki domknięć triad typów 030T i 102

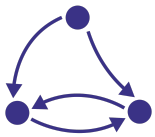


111D



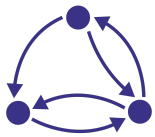
201

92.22%
(126 156)



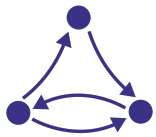
120D

4.20%
(5 746)



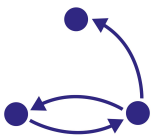
210

2.00%
(2 731)

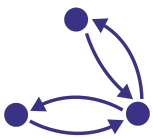


120C

1.51%
(2 070)

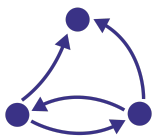


111U



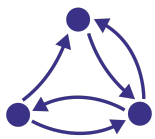
201

91.25%
(130 792)



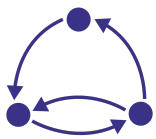
120U

4.99%
(7 156)



210

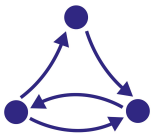
2.16%
(3 098)



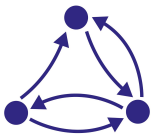
120C

1.32%
(1 887)

Rys 9-6 Statystyki domknięć triad typów 111D i 111U

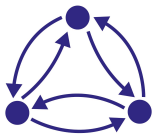


120C



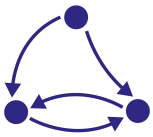
210

98.52%
(2 927)

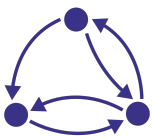


300

1.48%
(44)

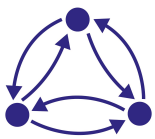


120D



210

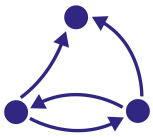
97.18%
(3 891)



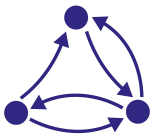
300

2.82%
(113)

Rys 9-7 Statystyki domknięć triad typów 120C i 120D

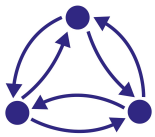


120U



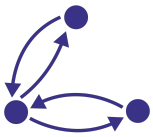
210

86.52%
(3 317)

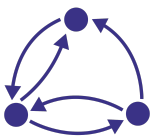


300

13.48%
(517)

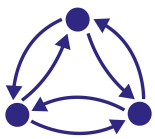


201



210

61.55%
(12 359)



300

38.45%
(7 722)

Rys 9-8 Statystyki domknięć triad 120U i 201

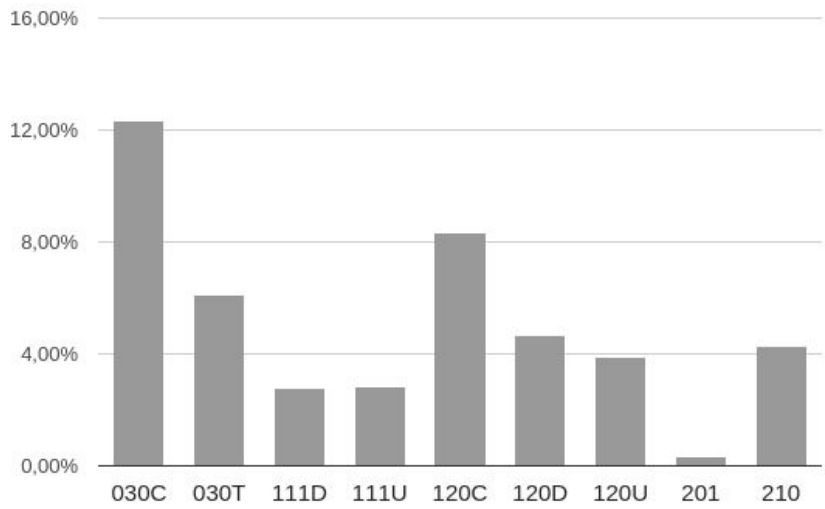
9.4 Czasy domknięć triad

Dla każdego domknięcia triady zaobserwowanej po raz pierwszy w późniejszym niż pierwsze wykonanie programu, czyli dla której jesteśmy w stanie stwierdzić z dokładnością co do tygodnia, że wcześniej jej nie było, wyznaczony został czas w tygodniach od jej pojawienia się do zniknięcia. Następnie dla poszczególnych typów triad sporządzono histogramy czasów życia triad (rys. od 9-10 do 9-14, wykresy słupkowe).

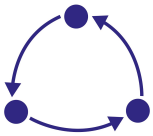
Informację uzupełniającą wobec rozkładu czasów życia triad stanowi stosunek liczy triad, które od czasu pierwszego ich zaobserwowania pozostały w danym stanie przez cały okres prowadzenia obserwacji (rys. od 9-10 do 9-14, wykresy kołowe).

Obliczenia te wykonano tylko dla typów triad wyznaczanych w całości na badanym grafie. Dla pozostałych typów wartości byłyby przekłamane.

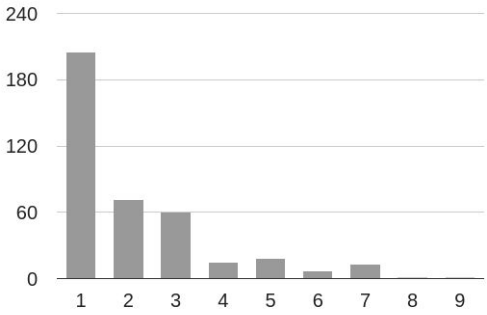
Dla wszystkich triad wyszukiwanych w całości wyznaczono procentową liczbę domknięć w skali wszystkich rozpoznanych triad danego typu. Wartości zestawiono na rys. 9-9.

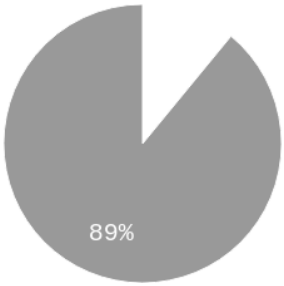


Rys. 9-9 Zestawienie udziału triad domkniętych wśród triad zaobserwowanych dla typów rozpoznawanych w całości

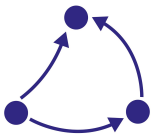


030C

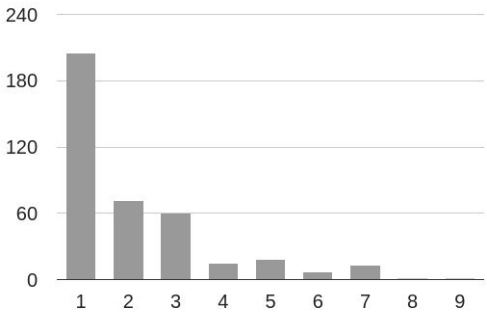


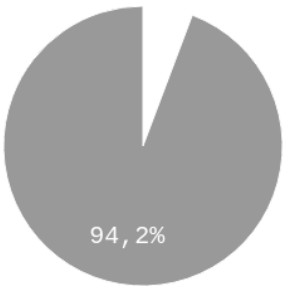


6 108
6 967



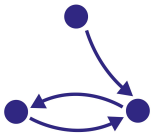
030T



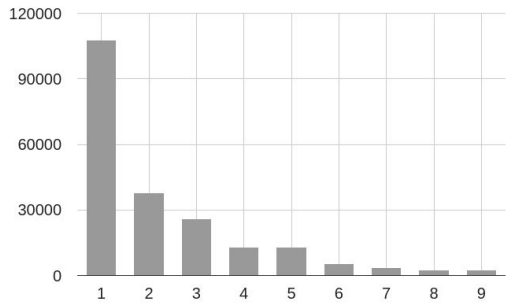


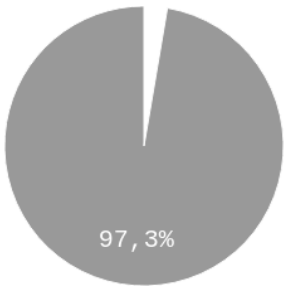
355 597
378 727

Rys. 9-10 Statystyki czasów domknięć dla triad typów 030C i 030T

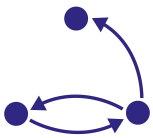


111D

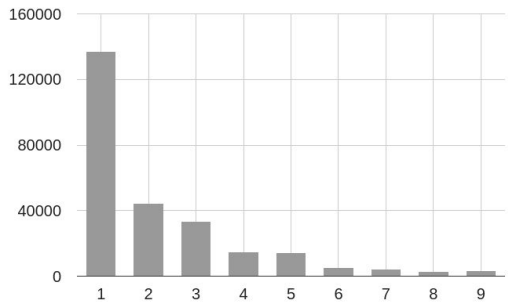


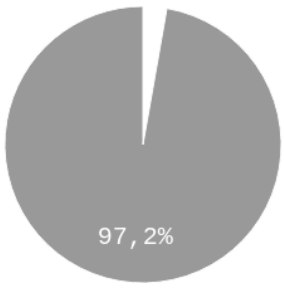


21 181 550
21 782 845



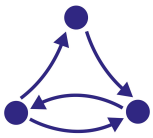
111U



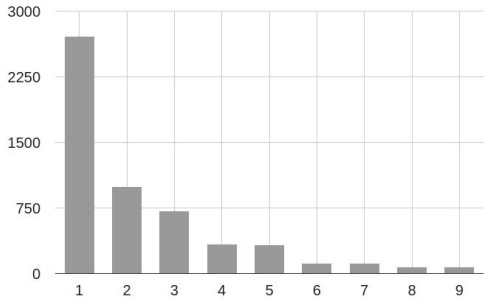


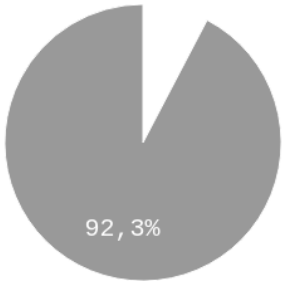
24 438 262
25 154 374

Rys. 9-11 Statystyki czasów domknięć dla triad typów 111D i 111U

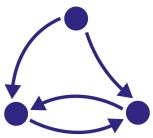


120C

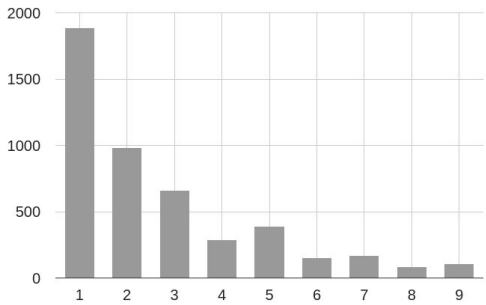


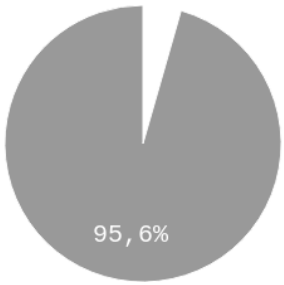


$$\frac{162\ 212}{176\ 928}$$



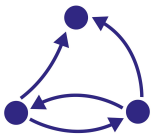
120D



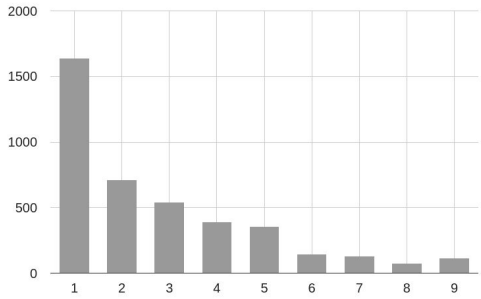


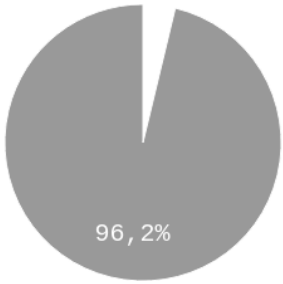
$$\frac{358\ 615}{376\ 109}$$

Rys. 9-12 Statystyki czasów domknięć dla triad typów 120C i 120D

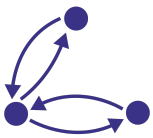


120U

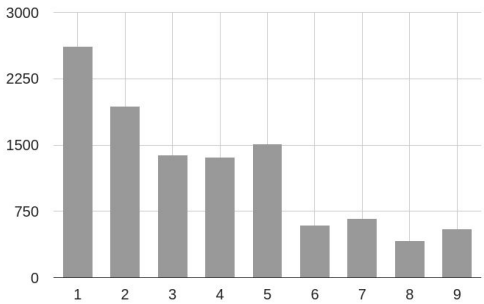


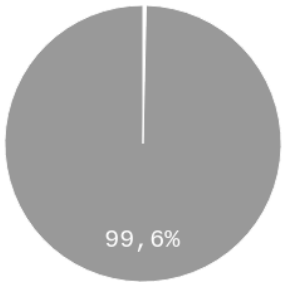


460 080
478 786



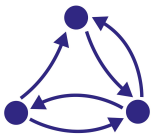
201



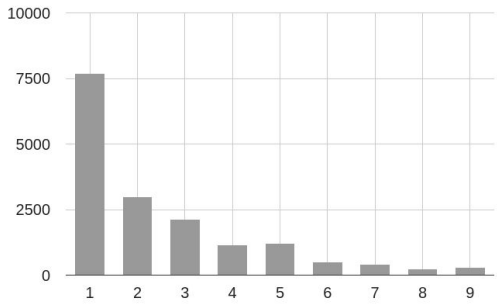


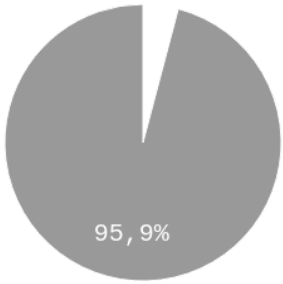
26 613 374
26 710 825

Rys. 9-13 Statystyki czasów domknięć dla triad typów 120U i 201



210





$$\frac{1\ 118\ 633}{1\ 168\ 784}$$

Rys. 9-14 Statystyki czasów domknięć dla triad typu 210

9.5 Analiza statystyk liczebności

Triady typu 003, 012 i 021C uznawane za niepołączone są zbyt liczne, by wyszukiwać je w tworzonym grafie. Dlatego w zestawieniu ilościowym należą do typów wyszarzonych, a ich liczba to w rzeczywistości jedynie liczba triad innych typów, dla których były poprzednikami.

Triady typu 021D, 021U i 102 to te spośród triad uznawanych za triady z lukami strukturalnymi, które również nie były wyszukiwane. Ich liczby również reprezentują wyłącznie liczbę triad, dla których stanowiły poprzedniki.

Triady 030C oraz 030T należą do klasyfikowanych przez zastane teorie jako niestabilne. W związku z tym ich obserwowana liczba powinna być niższa niż innych, stabilnych typów. Triady 030C faktycznie wyróżniają się niską liczebnością, czego nie można już powiedzieć o typie 030T. Tutaj liczebność jest zbliżona przynajmniej stopniem wielkości do typów uznawanych za stabilne.

Triady typu 120C są uznawane za niestabilne. Według zastanych teorii powinny być mało liczne. Okazuje się, że ich faktyczna liczebność jest zbliżona, przynajmniej jeśli chodzi o rząd wielkości, do wielu triad uznawanych za stabilne lub domknięte.

Oba typy triad 111, 111D i 111U, należące do triad stabilnych i zamkniętych, są licznie reprezentowane w badanym zbiorze danych. W porównaniu do wszystkich wcześniej wymienionych typów, gdzie żaden nie przekraczał liczebności pół miliona, te dwa typy są reprezentowane przez dziesiątki milionów przykładów.

Oba typy triad 120, 120D i 120U, należące do triad stabilnych i zamkniętych, są licznie, choć mniej licznie niż triady typu 111, reprezentowane w przetwarzanym zbiorze danych. Ich liczebność na poziomie setek tysięcy przykładów nie schodzi poniżej rzędu wielkości któregokolwiek rozważanego typu niedomkniętego.

Triady typu 201, klasyfikowane jako domknięte, należą do najbardziej licznych w przetwarzanym zbiorze danych. Ich liczba sięga dziesiątek milionów przykładów, co daje im miejsce wśród trzech najliczniejszych typów.

Dwa typy triad 210 i 300, z których pierwsza różni się o jedno brakujące połączenie od drugiego, reprezentującego z kolei komplet połączeń między trójką użytkowników, klasyfikowane jako triady domknięte, należą do bardzo licznych, choć nie najliczniejszych w przetwarzanym zbiorze danych, z liczebnością o rząd wielkości niższą od trzech najliczniej reprezentowanych typów.

9.6 Analiza statystyk domknięć

Należy pamiętać, że pomiary nowych połączeń w sieci nie były dokonywane w sposób ciągły. W wynikach często brakuje więc obserwacji stanów pośrednich, jakie trwały pomiędzy triadami widocznymi w wynikach. Przykładowo dla triady typu 201 jako poprzednika prawie równie liczne jak domknięcie do 210 jest domknięcie od razu do pełnej triady 300. Naturalnie przejście o dwa połączenia nie było w takim wypadku natychmiastowe, świadczy to jednak o niższej stabilności konfiguracji 210.

Czasy domknięć osiągają najbardziej płaski rozkład dla typu 201, gdzie między dwoma parami użytkowników istnieje komplet połączeń, a między jedną parą nie ma żadnego połączenia. Stabilna triada 201 domyka się podobnie jak pozostałe typy najczęście po tygodniu od powstania, ale już nieznacznie rzadziej po tygodniu.

Pośród typów 030C i 030T zachodzi znacząca różnica w liczebności i w sposób naturalny również między liczebnością domknięć. Dodatkowo mniej liczny typ 030C ma wyższy udział triad domkniętych wśród rozpoznanych - 12%. Typ 030C okazuje się bardzo rzadki i mocno niestabilny.

Wśród typów trzech typów triad 120, czyli wariantów D, U oraz C, wszystkie zdecydowanie prowadzą przeważnie do domknięcia do 210. Wariant 120U wyróżnia się jednak mniejszą stabilnością następującego po nim wariantu 210, bo częstsze okazują się obserwacje od razu pełnej triady 300 jako domknięcia. Rozkład czasów domykania się tych typów jest również zbliżony, najczęstsze są czasy do 1 tygodnia, a wartości dla kolejnych czasów dążą asymptotycznie do zera.

Dla wszystkich typów triad 021, czyli wariantów D, U oraz C, statystyki przejść są podobne, choć dla wersji 021C ponad 90% wyników rozkłada się między dwie możliwości miejsca, w którym może pojawić się nowe połączenie. Ta różnica względem dwóch pozostałych konfiguracji wynika z faktu niesymetryczności poprzednika w wariancie C. Przy pominięciu tego podziału dla wszystkich typów 021 widoczna jest zatem podobna sytuacja.

Dla obydwu typów 021 rzadkie jest pojawienie się połączenia między niepołączonymi węzła, dominuje natomiast uzupełnienie o połączenie zwrotne któregoś z nawiązanych już połączeń.

Również na przykładzie domknięć poprzednika 030T widoczna jest dominująca tendencja do domykania się połączeń po kolei. Rozkład procentowy prawdopodobieństw domknięć wskazuje tutaj na pewne uporządkowanie najbardziej prawdopodobnych połączeń, jakie się pojawią, natomiast najrzadsze jest pojawienie się od razu dwóch nowych połączeń.

Inaczej jest w przypadku domknięć poprzednika 102. Widoczny jest prawie równy rozkład między sytuacjami, gdzie ktoś z pary powiązanej dwukierunkową relacją zaczyna kogoś obserwować, zaczyna być przez kogoś obserwowany, czy w końcu nawiązuje obopólną relację z kimś z zewnątrz. Nie obserwuje się z kolei w istotnej skali, aby między drugą postacią z początkowej pary a nowym kontaktem pierwszej z nich od razu powstało połączenie. Można powiedzieć, że ludzie nawiązują zupełnie nowe kontakty na własną rękę, a dalszy rozwój triady wymaga czasu.

W zestawieniu udziałów triad domkniętych wśród rozpoznanych z rys. 9-9 widać najczęściej domykające się trzy typy: 030C, 120C i 030T. Wszystkie należą do triad niedomkniętych i zawierających luki strukturalne.

Analogicznie triady najrzadziej się domykające, czyli 111D, 111U oraz 201, są klasyfikowane jako triady zamknięte. Dla tych typów rozkłady czasowe znacznie się różnią. Dla dwóch pierwszych zdecydowana większość kolejnych przejść ma miejsce w ciągu tygodnia od pojawienia się. Różnica wobec 201 wynika z ogólnej mniejszej liczby połączeń w tych typach.

9.7 Wynik zestawienia z teorią

Wyniki pokazują, że w uzyskanym zbiorze danych nie odnaleziono zaprzeczenia dla zastanych teorii dotyczących stabilności triad. Choć brakuje dokładnego odwzorowania klasycznej klasyfikacji triad w ich liczebności, na przykład w trójce najliczniejszych triad 111D, 111U oraz 201 nie ma żadnej uznawanej za niestabilną. Z kolei najmniej liczny typ triady 030C jest w tejże klasyfikacji zaliczany do triad niestabilnych, co również świadczy o odzwierciedleniu tradycyjnej klasyfikacji w wynikach badań na sieci połączeń w Instagramie.

Na pewno zgromadzone dane, w szczególności ze statystykami domknięć triad, mogą stanowić wartościowy materiał do badań mechanizmów sieci społecznych przez specjalistów tej dziedziny, niekoniecznie związanych z inżynierią oprogramowania. Uzyskane nie tak wysokim nakładem pracy jednej osoby statystyki dostarczają znaczących, liczonych w dziesiątkach lub nawet setkach tysięcy próbek danych o domknięciach triad.

Takie wyniki uzyskane przy użyciu ogólnodostępnego API Instagramu, przewidzianego prawdopodobnie do innych celów, są możliwe do uzyskania w skończonym i nie tak długim czasie przez jednego pracującego samodzielnie programistę. W czas sprzed gwałtownego rozwoju serwisów społecznościowych uzyskanie takich danych wymagało setek dni pracy zespołów badaczy.

10 Podsumowanie

Zaproponowana modyfikacja znanych algorytmów przeszukiwania grafów spełniła swe zadanie i pozwoliła na realizację wyboru grupy przeszło 100 tysięcy użytkowników Instagramu niewyróżniających się stopniem wejściowym ani wyjściowym, czyli połączonych w większości z osobami, które znają i obserwują, realizując w ten sposób *słabe* połączenia.

Przygotowano program wykonujący migawki zmieniającej się sieci połączeń w ramach tak wyznaczonej grupy użytkowników. Zarówno dla początkowego stanu grafu, jak dla każdej aktualizacji wyznaczono triady. Dzięki stemplom czasowym zapisywanym w lokalnej bazie danych dla wszystkich obserwowanych połączeń możliwa jest czasowa analiza dynamiki sieci. W szczególności wyznaczone zostały wszystkie zaobserwowane przejścia między typami triad, czyli widoczne między kolejnymi skanami sieci zmiany konfiguracji połączeń w ramach trójek osób.

Efekt końcowy to uzyskany zbiór danych o dynamice fragmentu sieci, sprowadzalny do kolejnych migawek grafu skierowanego i przechowywany w postaci relacyjnej bazy danych ze stemplami czasowymi. Kwestią otwartą są przyszłe możliwości interpretacji tego zbioru danych, niekoniecznie związane z samą analizą triad. W ramach realizacji tej pracy pracę z danymi zakończono na analizie domknięć triad w zależności od ich typów, wyznaczeniu liczebności poszczególnych typów i prawdopodobieństw przejść między konkretnymi konfiguracjami. Te wyniki zostały zestawione z zebranymi wcześniej i również opisanymi teoretycznymi własnościami szesnastu możliwych triad dla grafów skierowanych, do których zalicza się Instagram.

Największym wyzwaniem praktycznym okazał się sposób pobierania danych poprzez API, Instagramu zgodnie z narzuconą przez firmę polityką danych dostępnych w ciągu godziny. Nieoczywisty był również zestaw zapytań rozpoznających w pobranych i zapisanych w bazie encjach wszystkie interesujące typy triad oraz przejścia między nimi.

Z kolei największym wyzwaniem w ogóle było element twórczy, czyli projekt algorytmu wyboru fragmentu sieci, zrealizowany metodą empiryczną i tak samo dostrojony dla uzyskania interesującego w dalszej części zbioru wierzchołków.

10.1 Wnioski

Wyniki statystyk liczebności i domknięć triad nie wykazują zdecydowanych sprzeczności z teorią wywiedzioną z własności grafów. Można by więc uznać, że teza o silnym wpływie samej struktury sieci społecznej, w ramach której poruszają się jednostki, na ich zachowanie, znajduje tutaj potwierdzenie. W zachowaniu użytkowników Instagramu znajdują odbicie prawa ogłoszone na długo przed powstaniem serwisów społecznościowych. Zachowanie to nie jest więc przypadkowe i całkowicie zależne od konkretnej jednostki.

Dane zebrane w toku realizacji pracy niosą najpewniej więcej możliwości analizy w przyszłych publikacjach. Ich największą wartością jest całościowej uchwycenie dynamiki wybranego według ścisłych reguł opisanych zaproponowanym algorytmem fragmentu sieci w określonym czasie prowadzenia obserwacji. Informacja o przebiegu powstawania połączeń między użytkownikami nie jest udostępniana przez API dostarczane przez Instagram.

Przy postawieniu pewnej nowej tezy o czynnikach warunkujących zachowanie się triad innej niż sama ich struktura możliwe jest w tym momencie weryfikacja na podstawie danych o aktywności użytkowników łatwo dostępnych poprzez API. Przy skali dostępnych współcześnie w serwisach społecznościowych informacji te prawa tylko czekają na odkrycie.

Literatura

[1] M. Tsvetovat, A. Kouznetsov. *Social Network Analysis for Startups*, rozdz. 1, 2, 4. 2011

[2] M. S. Granovetter. *The strength of weak ties*. 1973

[3] http://instagram.com/developer/endpoints/. 2016 (data dostępu: luty 2016)

[4] J. A. Davis, S. Leinhardt. *The structure of positive interpersonal relations in small groups*. 1967

[5] D. Easley, J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, rozdz. 3, 4, 5. 2010

[6] C. Marlow, L. Byron, T. Lento, I. Rosenn. *Maintained relationships on Facebook*. 2009

[7] B. A. Huberman, D. M. Romero, F. Wu. *Social networks that matter: Twitter under the microscope*. 2009

[8] D. B. Kandel. *Homophily, selection, and socialization in adolescent friendships*. 1978

[9] N. A. Christakis, and J. H. Fowler. *The spread of obesity in a large social network over 32 years*. 2007

[10] T. Newcomb. *The acquaintance process*. 1961

[11] G. C. Homans, *The Human Group*. 1950

[12] B. Wellman, J. Salaff, D. Dimitrova, L. Garton, M. Gulia, C. Haythornthwaite. *Computer networks as social networks: Collaborative work, telework, and virtual community*. 1996

Dodatek A: Implementacja wyboru fragmentu sieci

Poza samym REST API Instagram udostępnia biblioteki dla języków Python i Ruby. Najważniejszym kryterium przy doborze zestawu narzędzi, w tym języka, była skuteczność, z jaką będzie możliwe szybkie zaimplementowanie algorytmu wycięcia fragmentu sieci i analizy domykania się triad, wszystko z ograniczeniami narzucanymi przez API. Przenaszalność czy inne cechy języka nie były tak istotne. Spośród dwóch dostępnych bibliotek wybór padł na lepiej znany język Python.

A.1 Warstwa danych

Jako repozytorium danych najbardziej naturalnym wyborem okazała się baza danych o strukturze relacyjnej. Natura przechowywanych danych jest wyjątkowa bliska modelowi relacyjnemu. Poza encjami węzłów sieci, czyli profili użytkowników, pozostałe encje stanowią reprezentację połączeń między węzłami lub ich pochodne - zidentyfikowane triady jako zestawy połączeń i w końcu zaobserwowane przejścia między triadami jako powiązanie triady sprzed i po domknięciu.

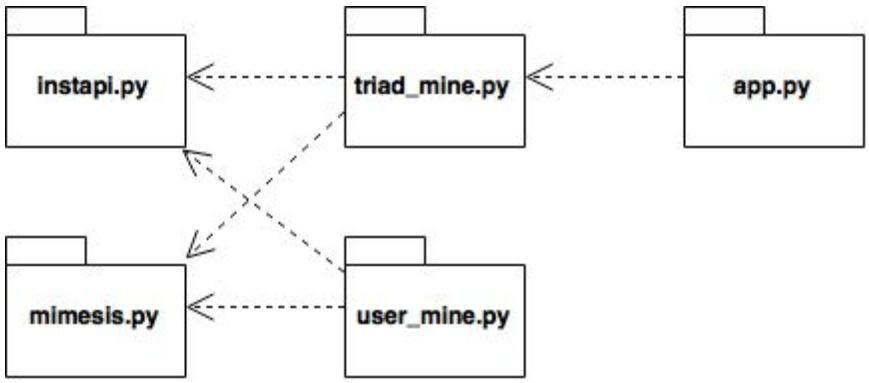
Sam wybór implementacji bazy relacyjnej był już bardziej problematyczny. Z początku wybrana ze względu na dużą przenaszalność baza SQLite nie spełniła wymagań wydajnościowych przy wykonaniu bardziej złożonych zapytań i masowego zapisu wielomilionowego zbioru zidentyfikowanych triad. W trosce o łatwość tworzenia kopii zapasowych wyniku zbierania danych poprzez API, czyli węzłów wraz z połączeniami, których zdobycie niesie ze sobą duży koszt czasu i których odtworzenie jest niemożliwe, ponieważ API nie udostępnia danych historycznych o zmianach w relacjach między użytkownikami - dla tych dwóch tabel bazy relacyjnej opłacało się jednak pozostać przy technologii SQLite. Druga baza miała posłużyć do przechowywania danych wtórnych, czyli możliwych do odtworzenia na podstawie informacji o węzłach i historii połączeń między nimi. W tym celu posłużono się technologią MySQL, oferującą znaczy wzrost wydajności, zwłaszcza przy zastosowaniu udostępnianych mechanizmów indeksowania. W ten sposób zapisano encje reprezentujące triady i występujące w czasie przejścia między nimi.

A.2 Wybrane technologie implementacji

Dla języka Python i użytych baz danych dostępne są implementacje systemów ORM, umożliwiających automatyczne mapowanie encji w bazie danych na obiekty zdefiniowanych klas. W tym celu wykorzystano technologię SQLAlchemy. Tabele zawierające użytkowników i połączenia między nimi są w ten sposób reprezentowane jako obiekty klas *User* i *Following*. Prowadzi to do zwiększenia w pewnym stopniu czytelności i łatwości implementacji.

We fragmentach odpowiedzialnych za znajdowanie typów triad wśród nowo zaobserwowanych połączeń zapytania były na tyle złożone, że łatwiej było przedstawić je jako zwykłe pliki SQL. Podobnie przy reszcie odwołań do bazy danych związanych z śledzeniem zmian wśród triad zrezygnowano z mapowania encji na obiekty. W zamian posłużono się biblioteką mysql udostępniającą funkcjonalność podobną do sterowników JDBC.

Podział aplikacji na moduły przedstawiono na rys. A-1. Cały kod odpowiedzialny za operacje na dwóch bazach danych zamknięto w module *mimesis*. Podobnie wszystkie operacje na Rest API z użyciem biblioteki przygotowanej przez Instagram znalazły się w module *instapi*, implementacja algorytmu wyboru węzłów do śledzenia w *user_mine*, w końcu mechanizm odpowiedzialny za odkrywanie triad i ich domknięć w *triad_mine*.



Rys. A-1 Graficzna reprezentacja zależności między modułami implementacji.

A.3 Ograniczenia na liczbę zapytań

Częścią wspólną modułów korzystających z modułu łączności z API *instapi*, czyli *user_mine* i *triad_mine*, jest sposób działania zoptymalizowany pod względem maksymalnej dozwolonej liczby zapytań (listingi A-2 do A-5) wysyłanych w skali godziny. Konieczne było wprowadzenie licznika wysyłanych zapytań, zdefiniowanego jako *hour_shots*, i kontrola tego licznika względem dozwolonych zapytań pozostałych w ciągu upływającej obecnie godziny - *ammo*.

Sposób działania *user_mine* i *triad_mine* polega na automatycznym cogodzinnym wpisywaniu do modułu *instapi* liczby dostępnych zapytań, a następnie wysyłania ich seriami o określonej długości, między którymi robione są pauzy o takiej długości, aby w skali godziny serie były rozłożone stosunkowo równomiernie. Przeważnie pozwala to na unikanie odpowiedzi o błędach ze strony API poprzez symulowanie naturalnego, nie maszynowego rozkładu zapytań w czasie. Po upływie godziny dostępna liczba zapytań jest ponownie automatycznie uzupełniania, a licznik zapytań wykonanych w ciągu aktualnej godziny zerowany.

```
def get_paginated(self, request, arg):
    results, next = self.risk_private(request, arg)
    LOG.debug("paginated 1st shot: {}".format(len(results)))
    self.shoot()
    while next:
        more_results, next = request(with_next_url=next)
        LOG.debug("paginated next shot: {}".format(len(more_results)))
        self.shoot()
        results.extend(more_results)
    return results
```

Listing A-2 Fragment implementacji realizujący stronicowanie zapytań do API. Takie zapytania poza wynikiem zwracają również referencję do kolejnej strony wyników (`next`). Wszystkie wysyłane zapytania rejestruje się w liczniku zapytań (`shoot`).

```
def search(self, username):
    LOG.debug('issue user search: {}'.format(username))
    self.shoot()
    response = self.api.user_search(username)
    return response[0] if len(response) > 0 else None
```

Listing A-3 Fragment implementacji realizujący zapytanie ID profilu na podstawie nazwy użytkownika. Krok używany tylko na samym początku tworzenia fragmentu sieci, gdy podany zostaje węzeł początkowy. Wszystkie wysyłane zapytania rejestruje się w liczniku zapytań (`shoot`).

```
def info(self, id):
    LOG.debug("issue user info: {}".format(id))
    response = self.risk_private(self.api.user, id)
    self.shoot()
    return UserInfo(response)
```

Listing A-4 Fragment implementacji realizujący zapytanie o podstawowe informacje o profilu. Wszystkie wysyłane zapytania rejestruje się w liczniku zapytań (`shoot`).

```
def followees(self, id):
    LOG.debug("issue followees: {}".format(id))
    response = self.get_paginated(self.api.user_follows, id)
    LOG.debug("{} follows {} users".format(id, len(response)))
    return Followees(response)
```

Listing A-5 Fragment implementacji realizujący zapytanie o połączenia wychodzące z węzła. Korzysta ze stronicowania wyników zapytań (`get_paginated`).

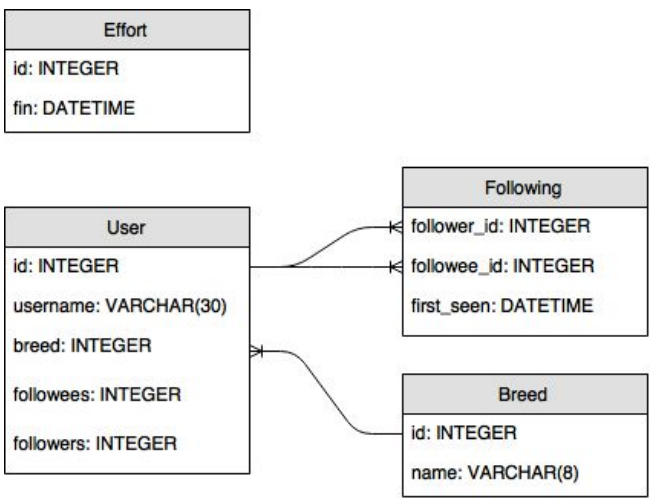
Ograniczenie 5 tysięcy zapytań na godzinę jest faktycznie mocniejszym ograniczeniem, niż mogło by się wydawać. Jedno zapytanie służy do pozyskania podstawowych danych o użytkowniku, w tym liczby śledzonych przez niego profili i profili go śledzących, innymi słowy rzeczywistego stopnia wejściowego i wyjściowego danego węzła. Na podstawie takiego wyniku algorytm jest w stanie zdecydować, czy podany węzeł będzie w przyszłości przetwarzany, czy może nie spełnia wymagań nałożonych przez wartości progowe dla tych parametrów, o czym była mowa we wcześniejszych rozdziałach. Gdy dochodzi do przetwarzania węzła, pobierane są dla niego wszystkie połączenia wychodzące i tutaj przeważnie jedno zapytanie to za mało. Wyniki są stronicowane, więc w zależności od liczby stopnia wyjściowego wierzchołka liczba potrzebnych zapytań waha się. Szczęśliwie biblioteka dla języka Python traktuje te kolejne zapytania jako osobne wywołania funkcji, dzięki czemu jest możliwe zliczanie i takich zapytań. O tym, że zapytanie nie zwróciło jeszcze wszystkich wyników, informuje znacznik w odpowiedzi, stanowiący odnośnik potrzebny do pobrania kolejnej strony wyników w kolejnym zapytaniu.

W ciągu godziny jest więc możliwe przetworzenie znacznie mniej niż 5000 użytkowników, zwłaszcza w fazie wyboru fragmentu sieci do dalszej analizy, kiedy część zapytań zostaje przeznaczona na węzły później odrzucone. Przetworzenie wyznaczonych 50 tysięcy użytkowników w kolejnych fazach, gdy stanowią już zamknięty zbiór i wysyłane są tylko dla nich wszystkich stronicowane zapytania o śledzone profile, zajmuje w przybliżeniu 2 doby.

A.4 Object-relational mapping (ORM)

Użycie mapowania encji na obiekty Pythona (listing A-7) polega na zdefiniowaniu klas odpowiadających tym encjom z polami jako instancjami klasy *Column*. Ponieważ w języku Python nie mamy do czynienia ze statycznym typowaniem zmiennych, pierwszy parametr konstruktora tej klasy określa typ danej kolumny w tabeli. Pozostałe parametry konstruktora określają pozostałe właściwości kolumny, w tym jej obowiązkowość czy przynależność do klucza głównego tabeli.

Relacje między tabelami (rys. A-6) również znajdują odzwierciedlenie w obiektach Pythona. Przy relacjach 1-n, jak na przykład lista profili śledzonych przez lub śledzących danego użytkownika, do wszystkich powiązanych encji zmapowanych na obiekty pojawia się dostęp z poziomu tablicy tych powiązanych obiektów będącej polem danego obiektu.



Rys. A-7 Schemat bazy SQLite użytkowników i połączeń między nimi.

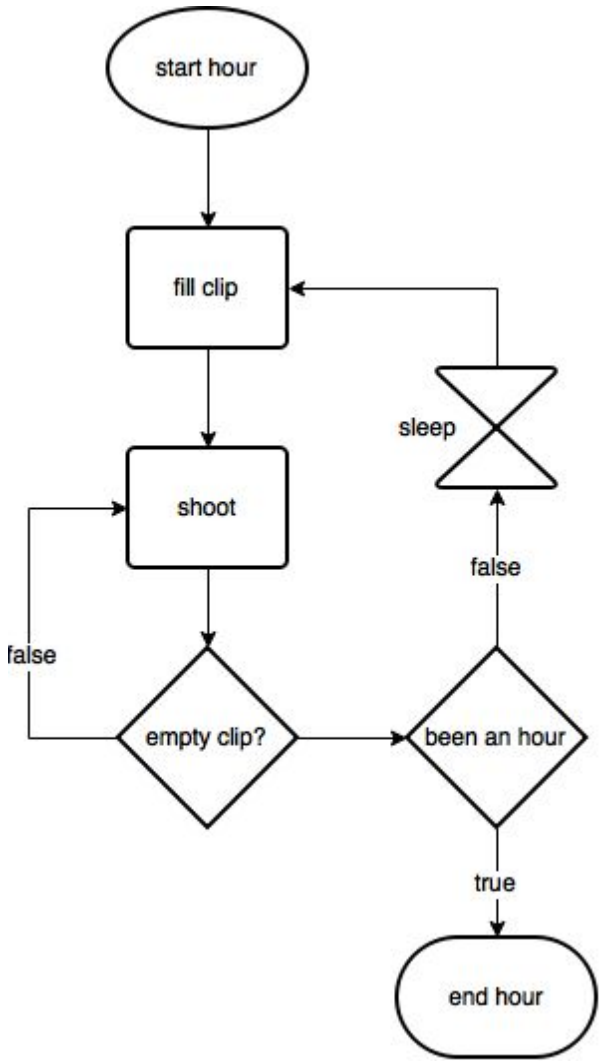
```
class Following(Base):
    __tablename__ = 'following'
    follower_id = Column(Integer, ForeignKey('user.id'), primary_key=True)
    followee_id = Column(Integer, ForeignKey('user.id'), primary_key=True)
    first_seen = Column(DateTime, default=datetime.datetime.now())


class Breed(Base):
    UNKNOWN = 'unknown'
    REGULAR = 'regular'
    PRIVATE = 'private'
    CELEB = 'celeb'
    MANIAC = 'maniac'
    INACTIVE = 'inactive'

    __tablename__ = 'breed'
    id = Column(Integer, nullable=False, primary_key=True)
    name = Column(String(8), nullable=False)


class User(Base):
    __tablename__ = 'user'
    id = Column(Integer, nullable=False, primary_key=True)
    username = Column(String(30), nullable=False)
    breed = Column(Integer, ForeignKey('breed.id'), default=1)
    followees = Column(Integer, nullable=True)
    follows = relationship(
        'User', secondary='following',
        primaryjoin=(Following.follower_id == id),
        secondaryjoin=(Following.followee_id == id)
    )
    followers = Column(Integer, nullable=True)
    followed_by = relationship(
        'User', secondary='following',
        primaryjoin=(Following.followee_id == id),
        secondaryjoin=(Following.follower_id == id)
    )
```

Listing A-7 Sposób wykorzystania mechanizmu ORM do mapowania encji bazodanowych na obiekty języka Python. Pola klas są inicjowane jako obiekty klasy `Column` o określonym typie i wartościach domyślnych, ewentualnie oznaczone jako klucze obce do pozostałych kolumn.

A.5 Kontrola liczby zapytań

Przy automatycznym uzupełnianiu pewnych kolumn tabeli przy wstawianiu nowego wiersza nic nie stoi również na przeszkodzie, żeby działo się to na mocy funkcji samego języka, a nie bazy danych. Jak wcześniej wspomniano i uzasadniono, encje reprezentujące pierwsze zaobserwowanie danej relacja między użytkownikami posiadają stemple czasowe. Do automatycznego uzupełniania kolumny tego stempla zostaje użyta funkcja określająca aktualny czas z odpowiedniej standardowej biblioteki języka.

Po wyczerpaniu każdej serii zapytań do API, o rozmiarze określonym jako parametr w konfiguracji, i odczekaniu czasu wynikającego z równomiernego rozkładu serii zapytań w skali godziny, do puli zapytań ładowana jest od nowa, identyczna liczba zapytań, a kolejka węzłów do przetworzenia jest aktualizowana według zdefiniowanych przez algorytm kroków. W tym celu do repozytorium węzłów i połączeń w bazie SQLite, za pośrednictwem systemu mapowania encji na obiekty, wysłane zostaje zapytanie o określoną również parametrem konfiguracyjnym liczbę węzłów o najwyższym jak dotąd stopniu wejściowym, liczonym względem już przetworzonych węzłów (rys. A-8).

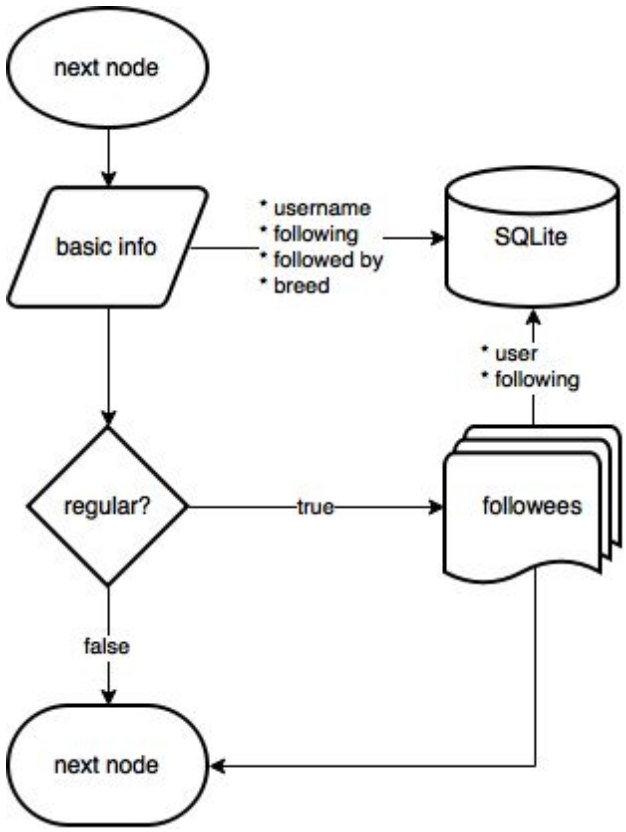


Rys. A-8 Schemat blokowy procesu kontroli liczby wysyłanych zapytań do API Instagramu. Do puli zapytań (*clip*) ładuje się określoną część godzinnego limitu. Przy kolejnych zapytania rozmiar puli jest zmniejszany. Po wyczerpaniu puli i odliczeniu określonego czasu pula jest na powrót ładowana. Po drodze sprawdzane jest nieprzekroczenie limitu godzinnego i czy nie należy rozpocząć zliczania zapytań w ramach kolejnej godziny.

A.6 Przebieg przetwarzania węzłów

W początkowych przebiegach algorytmu kolejka węzłów do przetworzenia jest dodatkowo ograniczana o jeden stopień wielkości. Służy to przyspieszeniu zbiegania algorytmu do większej różnicy w stopniach wejściowych liczonych względem już przetworzonych węzłów i związanego z nim lepszego rozróżnienia przy wyborze kolejnych węzłów wartych odwiedzenia.

Przetworzenie węzła pobranego z kolejki (rys. A-9) zaczyna się na odrzuceniu węzłów niespełniających wymogów na wartości progowe. Są trzy możliwe powody zakwalifikowania węzła jako nieinteresującego dla dalszej analizy dynamiki sieci: liczba obserwujących ponad wartością progową, liczba obserwowanych powyżej wartości progowej, liczba obserwowanych nieprzekraczająca wartości progowej (listing A-10). W konfiguracji użytej podczas dalszych eksperymentów dla tych trzech parametrów przyjęto wartości kolejno: 500, 500, 0.



Rys. A-9 Schemat blokowy algorytmu przetwarzania węzła. W krokach *basic info* oraz *followees* wysyłane są zapytania do API Instagramu. Dalsze przetwarzanie węzła uzależnione jest od rozpoznanego typu (*breed*) węzła, czyli czy jest to na przykład profil prowadzony przez osobę publiczną, czy przez przeciętnego (*regular*) użytkownika. Decyzja o dalszym przetwarzaniu powoduje pobranie listy połączeń wychodzących i ich zapis do bazy.

```
def check_breed(self, follower):
    try:
        info = self.api.info(follower.id)
    except UserPrivateException:
        LOG.info("<private>")
        self.db.set_private(follower)
        return Breed.PRIVATE

    followed_by = info.followers_count()
    self.db.set_followers(follower, followed_by)
    LOG.info("followed by: {}".format(followed_by))

    follows = info.followees_count()
    self.db.set_followees(follower, follows)
    LOG.info("follows: {}".format(follows))

    if followed_by >= CELEB_THRESHOLD:
        LOG.info("<celeb>")
        self.db.set_celeb(follower)
        return Breed.CELEB

    if follows >= MANIAC_THRESHOLD:
        LOG.info("<maniac>")
        self.db.set_maniac(follower)
        return Breed.MANIAC

    if follows <= INACTIVE_THRESHOLD:
        LOG.info("<inactive>")
        self.db.set_inactive(follower)
        return Breed.INACTIVE

    self.db.set_regular(follower)
    return Breed.REGULAR
```

Listing A-10 Fragment implementacji kategoryzujący użytkownika na podstawie publicznej dostępności profilu i liczby połączeń wchodzących/wychodzących.

A.7 Parametry konfiguracyjne

Konfigurację parametrów sterujących implementacją mechanizmów pobierania fragmentu sieci, jak i późniejszej analizy jej dynamiki ułatwia zebranie ich w osobnych pliku konfiguracyjnym. Dostrajanie całości implementacji na rzecz lepszych wyników eksperymentu jest dzięki temu możliwe nie z poziomu kodu, gdzie czasem trudno doszukać się definicji danego parametru, a w jednym miejscu, w formie czytelnej nawet bez znajomości użytego języka programowania.

A.8 Logowanie

Przy monitorowaniu sieci o rozmiarze rzędu dziesiątek tysięcy użytkowników wykonanie programu przeciąga się do wielu dziesiątek godzin. W takiej sytuacji niezbędne jest narzędzie monitorowania przebiegu programu, umożliwiające aktualny stan jego wykonania. Najprostszym i najbardziej skutecznym sposobem na zrealizowanie takiego wymagania jest odwołanie się do dobrych praktyk logowania. Każda dedykowana biblioteka pozwala na definiowanie wpisów do logu na różnych poziomach szczegółowości. W ten sposób przy normalnym, długim wykonaniu aplikacji wygodnie jest ukryć nadmiarowe informacje diagnostyczne, by śledzić wyłącznie aktualne wysokopoziomowe informacje.

W produkowanym logu (listing A-7, A-8). , dostępnym do stałego wglądu w trakcie działania aplikacji, widać informację o aktualnie przetwarzanym węźle wypisanym w postaci łącza do profilu danego użytkownika, co samo w sobie może stanowić pewną rozrywkę. W momentach przeładowywania dostępnych zapytań do API i uzupełniania kolejki o najbliższe węzły warte odwiedzenia wyświetlana jest również informacja o postępach przetwarzania sieci.

Przy uruchomieniu trwającym dziesiątki godzin konieczne jest chronienie się przed sytuacjami wyjątkowymi - jak okresowe błędy zgłaszane przez API lub problemy z siecią. Długie testowanie implementacji pozwoliło na wychwycenie wszystkich sytuacji wyjątkowych i implementację zachować koniecznych do przywrócenia programu do prawidłowego dalszego działania w przypadku ich wystąpienia.

```
18:26:43,683 (user_mine) [INFO]: next up (1) -> instagram.com/mura_boutique
18:26:44,330 (user_mine) [INFO]: followed by: 601017
18:26:44,330 (user_mine) [INFO]: follows: 1208
18:26:44,331 (user_mine) [INFO]: <celeb>
18:26:44,331 (user_mine) [INFO]: next up (1) -> instagram.com/lookbook
18:26:44,929 (user_mine) [INFO]: followed by: 1023696
18:26:44,930 (user_mine) [INFO]: follows: 1195
18:26:44,930 (user_mine) [INFO]: <celeb>
18:26:44,958 (user_mine) [INFO]: limit queue size to 30
18:26:44,961 (user_mine) [INFO]: next up (4) -> instagram.com/naumoba
18:26:45,568 (user_mine) [INFO]: followed by: 231
18:26:45,568 (user_mine) [INFO]: follows: 393
18:26:51,920 (user_mine) [INFO]: <done>
18:26:51,920 (user_mine) [INFO]: next up (4) -> instagram.com/weronikadraus
18:26:52,560 (user_mine) [INFO]: followed by: 101
18:26:52,560 (user_mine) [INFO]: follows: 96
18:26:53,995 (user_mine) [INFO]: <done>
18:26:53,995 (user_mine) [INFO]: next up (3) -> instagram.com/putyourhoodon
18:26:54,596 (user_mine) [INFO]: followed by: 245
18:26:54,596 (user_mine) [INFO]: follows: 219
18:26:58,234 (user_mine) [INFO]: <done>
18:26:58,235 (user_mine) [INFO]: next up (3) -> instagram.com/lucyszulinska
18:26:58,837 (user_mine) [INFO]: followed by: 254
18:26:58,837 (user_mine) [INFO]: follows: 211
18:27:02,365 (user_mine) [INFO]: <done>
18:27:02,366 (user_mine) [INFO]: next up (3) -> instagram.com/awanturaobasie
18:27:02,708 (user_mine) [INFO]: <private>
```

Listing A-11 Fragment logu z uruchomienia implementacji wybory fragmentu sieci. Widoczne są w nim kolejne przetwarzane profile i uzyskiwane dla nich wartości podstawowych parametrów. Niektóre na podstawie stopnia wejściowego od razu klasyfikuje się jako *celeb*. Inne, z dostępem prywatnym zostają oznaczone jako *private*.

```
18:29:42,383 (user_mine) [INFO]: next up (6) -> instagram.com/oficyna_peryferia
18:29:43,038 (user_mine) [INFO]: followed by: 207
18:29:43,039 (user_mine) [INFO]: follows: 21
18:29:43,695 (user_mine) [INFO]: <done>
18:29:43,873 (user_mine) [INFO]: ==============reload===============
18:29:43,917 (user_mine) [INFO]: limit queue size to 30
18:29:43,917 (user_mine) [INFO]: mode: open
18:29:43,917 (user_mine) [INFO]: cycle = 240
18:29:43,951 (mimesis) [INFO]: ----------------------
18:29:43,952 (mimesis) [INFO]:         stats
18:29:43,952 (mimesis) [INFO]: ----------------------
18:29:43,952 (mimesis) [INFO]: users     = 6570
18:29:43,952 (mimesis) [INFO]: processed = 108
18:29:43,952 (mimesis) [INFO]: queued    = 6462
18:29:43,952 (mimesis) [INFO]: regular   = 47
18:29:43,952 (mimesis) [INFO]: privates        = 9
18:29:43,952 (mimesis) [INFO]: celebs    = 46
18:29:43,952 (mimesis) [INFO]: maniacs   = 6
18:29:43,952 (mimesis) [INFO]: inactive        = 0
18:29:43,952 (mimesis) [INFO]: ----------------------
18:29:43,952 (user_mine) [INFO]: under 1 min left
```

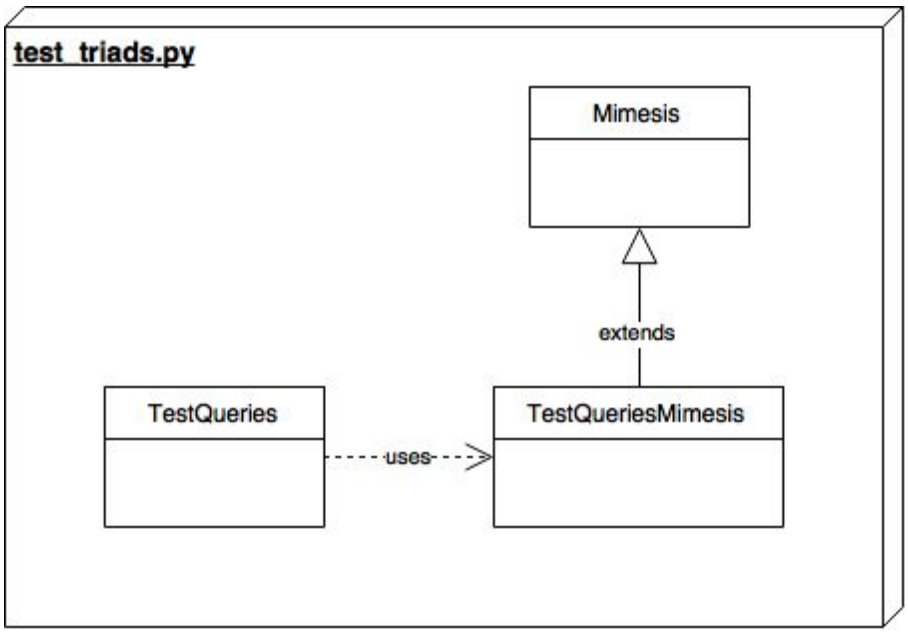
Listing A-12 Fragment logu z operacji przeładowywania kolejki. Drukowane są statystyki na temat przetworzonych dotychczas węzłów. Następnie implementacja odczekuje wyliczony czas przed wznowieniem działania, tak aby nie zostały przekroczone godzinne limity na liczbę zapytań.

Dodatek B: Testowanie

Wobec złożoności implementacji algorytmów identyfikacji triad poszczególnych typów wśród połączeń między węzłami absolutnie konieczne okazało się duże pokrycie testami. Początkowe traktowanie tej kwestii jako drugorzędnej i pobieżne podejście do niej skutkowało poważnymi błędami w działaniu implementacji i w produkowanych przez nią wynikach. Skala danych, na jakich działa eksperyment, ujawnia wszystkie najdrobniejsze błędy implementacji i powoduje ich katastrofalne skutki. Specyfika powstałego oprogramowania polega na jego zupełnej nieodporności na błędy podczas przetwarzania danych. Źle zidentyfikowana lub niezidentyfikowana na poziomie poszukiwania triad triada na pewno ujawni się w kroku poszukiwania poprzedników w kolejnym obiegu algorytmu.

B.1 Istotność testów

W przeważającej ilości przypadków najbardziej wrażliwe na błędy fragmenty implementacji operują na bazie danych, czy to zapisując do niej wyniki kolejnych kroków algorytmów, czy to pobierając z niej dane do dalszego przetwarzania. Wobec tego do najlepszego pokrycia mogło prowadzić wykonywanie testów na faktycznych bazach danych SQLite i MySQL (rys. B-1). Powstałe testy można więc uznać nie tylko za jednostkowe, ale i integracyjne.



Rys. B-1 Zawartość modułu testów zapytań SQL o typy triad. Klasa testowa *TestQueries* korzysta z rozszerzenia modułu persystencji na cele testowe.

B.2 Testowanie integracyjne z udziałem baz danych

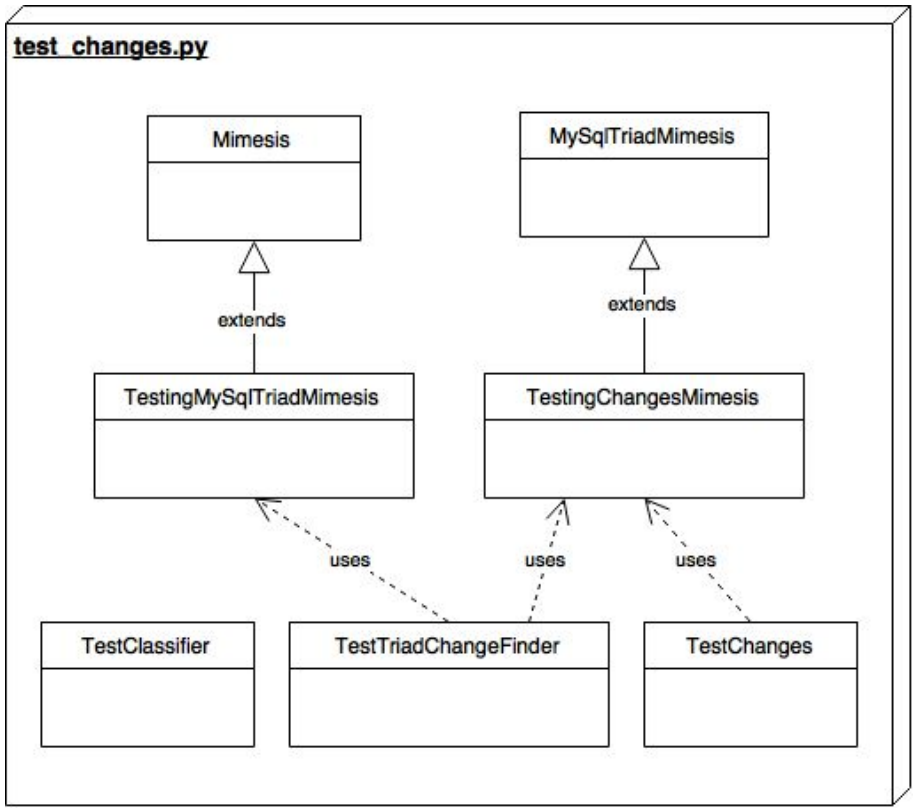
Na rzecz testów przygotowane zostały schematy baz danych identyczne z używanymi podczas zwykłego uruchomienia implementacji, ale jednak oddzielne, aby uniknąć wystąpień konfliktów między danymi testowymi a autentycznymi czy chociażby wygodnie automatycznie czyścić w całości bazy testowe przed ponownym uruchomieniem implementacji. Przełączanie między bazami testowymi i produkcyjnymi umożliwia przygotowanie oddzielnych plików konfiguracyjnych używanych przy faktycznym i testowym uruchomieniu. Ścieżka do używanej konfiguracji jest stała, z kolei pod tę ścieżkę przed rozpoczęciem testów kopiowana jest konfiguracja testowa, z uprzednim zachowaniem pod inną ścieżką konfiguracji produkcyjnej, by po zakończeniu testów ponownie zamienić te pliki miejscami. Definicja operacji wykonywanych przed i po zakończeniu wszystkich testów jest najwygodniejsza w umieszczeniu w dedykowanych do tego metodach klas testowych, zdefiniowanych w pakiecie przeznaczonym do testowania jednostkowego języka Python.

B.3 Rodzaje testów

Pierwszą liczną grupą testów są te odpowiedzialne za sprawdzenie poprawności wszystkich zapytań SQL przygotowanych do rozpoznawania triad poszczególnych typów w najbardziej aktualnej sieci połączeń między śledzonymi węzłami (rys. B-2). Przez rozpoczęciem testów baza testowa zapełniona zostaje połączeniami odpowiadającymi wszystkim możliwym triadom we wszystkich ich wariantach. Następnie poszczególne metody testowe o nazwach odpowiadających typom triad sprawdzają prawidłowe rozpoznanie tych i tylko tych triad wśród wszystkich połączeń. W wielu zapytaniach unikalne identyfikatory węzłów służą do wyeliminowania duplikatów triad, zwłaszcza tych o cyklicznej konstrukcji, stąd konieczność powielenia wielu przypadków testowych w tym względzie i wprowadzenia wszystkich możliwych kombinacji triad co do uporządkowania ich węzłów według wartości unikalnych identyfikatorów węzłów.

Równie istotne jak zapytania rozpoznające triady jest prawidłowe działanie zaprojektowanego klasyfikatora dla triad w ogólności mniej interesujących, a co za tym idzie nie zapisywanych w bazie przy regularnym rozpoznawaniu nowych triad, potrzebnych jednak podczas kroku identyfikacji poprzedników dla wszystkich nowych triad. W tym przypadku implementacja jest niezwiązana z zapytaniami języka SQL, operuje w zamian na samym listach połączeń w ramach sprawdzanej triady sprzed ostatniej zakończonego obiegu całości algorytmu poszukiwania domknięć w triadach. Dzięki temu te testy mogą obyć się w ogóle bez połączenia z bazą danych.

Fragment kodu testów złożonych ze scenariuszów o największej liczbie kroków to część odpowiedzialna za badanie prawidłowego działania implementacji pod względem rozpoznawania przejść między triadami. Testy jak na listinu B-4 symulują upływ czasu między kolejnymi nowo dodawanymi i opatrzonymi stemplami czasowymi połączeniami pomiędzy węzłami. Umożliwiają to rozszerzenia do standardowych mechanizmów persystencji (listing B-3). Ponadto wprowadzają do testowej bazy danych kilka kroków domykania się triady, by następnie upewnić się o prawidłowym rozpoznaniu i zapisie do dedykowanej tabeli informacji o zaobserwowanych domknięciach.



Rys. B-2 Zawartość modułu testów zapytań SQL o typy triad. Klasy testowe korzystają z rozszerzenia modułów persystencji użytkowników, połączeń, triad i przejść między nimi na cele testowe.

```
def test_003_030C_120C_210(self):
    db = TestingChangesMimesis()
    triad_conn = mysql.connector.connect(user='instamine',
                                         password='instamine',
                                         host='localhost',
                                         database='test')
    triad_mimesis = TestingMySqlTriadMimesis(triad_conn)
    finder = TriadFinder(triad_mimesis)
    changes = TriadChangeFinder(triad_mimesis)

    self.add_followings(db, [(40, 41), (41, 42), (42, 40)])
    self.mine_cycle(changes, db, finder, triad_mimesis)
    self.assertChangesList(triad_conn, 1, {40, 41, 42, '003', '030C'})

    self.add_followings(db, [(41, 40)])
    self.mine_cycle(changes, db, finder, triad_mimesis)
    self.assertChangesList(triad_conn, 2, {40, 41, 42, '030C', '120C'})

    self.add_followings(db, [(42, 41)])
    self.mine_cycle(changes, db, finder, triad_mimesis)
    self.assertChangesList(triad_conn, 3, {40, 41, 42, '120C', '210'})
```

Listing B-4 Fragment implementacji testujący prawidłową identyfikację przejść między triadami. Test integracyjny korzysta z testowej bazy danych.

```
class TestingChangesMimesis(Mimesis):
    def effort_minutes_ago(self, m):
                                record    =    Effort(fin=datetime.datetime.now()    -
datetime.timedelta(minutes=m))
        self.session.add(record)

    def new_fin(self):
        fin = self.last_fin() + datetime.timedelta(minutes=2)
        self.session.add(Effort(fin=fin))
        self.session.commit()

    def they_go_way_back(self, follower_id, followee_id, to):
                                                                  following         =
self.session.query(Following).filter_by(follower_id=follower_id,
followee_id=followee_id).first()
        following.first_seen = to

    def add_following(self, follower_id, followee_id):
        first_seen = self.last_fin() + datetime.timedelta(minutes=1)
                                self.session.add(Following(follower_id=follower_id,
followee_id=followee_id, first_seen=first_seen))
```

Listing B-3 Fragment implementacji mechanizmu rozszerzenia klasy odpowiedzialnej za persystencję użytkowników i połączeń do celów testowych.