Projeto e Análise de Algoritmos
Trabalho Prático 2:
Identificando Ataques Sybil

Vagner Clementino[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

vagnercs@dcc.ufmg.br

1. Introdução

As *redes sociais* vêm se mostrando um local propício para que os criminosos virtuais apliquem os seus golpes. Diversos tipos de atividades maliciosas, tais como *spam*, *phishing*, *malware* e *falsificação de perfis* são constantes ameaças aos usuários das redes. Um tipo de ameaça que vêm crescendo dentro das redes sociais é o ataque *Sybil*. Nesse tipo de ameaça, usuários criam múltiplas identidades com o objetivo de criar relacionamentos sociais com usuários reais com finalidades maliciosas [Viswanath et al. 2011].

Em alto nível, todos mecanismos de defesa existentes tentam isolar vértices Sybil que estão incorporados dentro da topologia de uma rede social. Cada mecanismo determina se um vértice na rede ou é Sybil ou não-Sybil (honesto) da perspectiva de um vértice honesto efetivamente particionando os vértices da rede social em duas regiões distintas, a saber: região honesta e a região Sybil. Uma região honesta no grafo G é um subgrafo que consiste de todos os vértices honestos e arestas entre eles. Por sua vez, uma região Sybil, é composta por vértices maliciosos e arestas entre eles. As arestas em G que conectam a região honesta com a região Sybil são chamadas de arestas de ataque, conforme é apresentado na Figura 1.

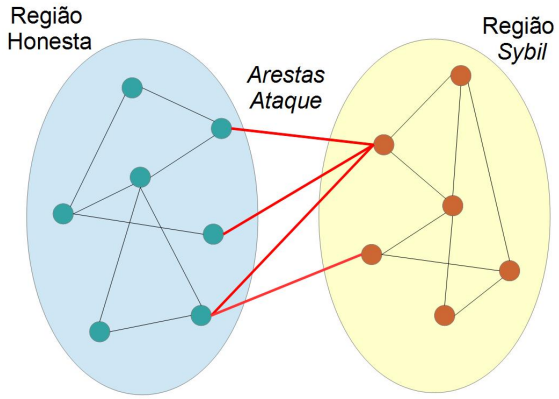

Figura 1. Região honesta e região Sybil.

Este documento está organizado da seguinte forma: a Seção 2 descreve a solução proposta, detalhamento a modelagem do problema bem como o algoritmo

para particionamento do grafo na região Sybil e Honesta; na Seção 3 é apresentado a análise de complexidade de tempo e espaço da algoritmo proposto; a Seção 4 é devotada a análise experimental do algoritmo, tendo um foco especial no comportamento do mesmo para grafo de diferentes tipos utilizando para tal o conjunto de métricas descritos na Subseção 2.3; o documento é finalizado na Seção 5 onde as conclusões obtidas são discutidas bem como as possíveis implicações dos resultados encontrados.

2. Descrição da Solução

Esta Seção é devotada à explicação do algoritmo proposto para detecção de usuários *Sybil*. Em suma, o algoritmo proposto recebe um grafo representado uma rede social e realiza o particionamento do mesmo em duas regiões, a região *honesta (A)* e a região *Sybil (B)*. Iniciemos detalhando a modelagem de grafo utilizada.

2.1. Modelagem do Problema

Uma Rede Social pode ser modelada como um grafo $G = (V, E)$ *não direcionado*. O conjunto V é o conjunto de *vértices*, em cada vértice representará um usuário da rede. De forma análoga, o conjunto E é o conjunto de *arestas*, e tem como objetivo representar um relacionamento social entre dois usuários da rede.

Tradicionalmente os grafos são modelados utilizando estruturas de dados tais como matriz ou lista de adjacência. Neste trabalho, contudo, utilizou-se uma representação um pouco diferente para um grafo. A estrutura de dados grafo foi representada como um conjunto de *Vértices*. Estes, por sua vez, armazenam um conjunto de *Arestas*, as quais armazenam os vértices inicial e final em que a aresta é incidente. A Figura 2 exibe um diagrama de classe UML[Rumbaugh et al. 2004]. A principal vantagem desta modelagem está em sua flexibilidade tendo em vista que é possível construir facilmente tanto grafos direcionados quanto não direcionados. Ademais, este tipo de representação facilidade bastante o processo de remoção ou adição de arestas ou vértices ao grafo durante a execução do algoritmo.

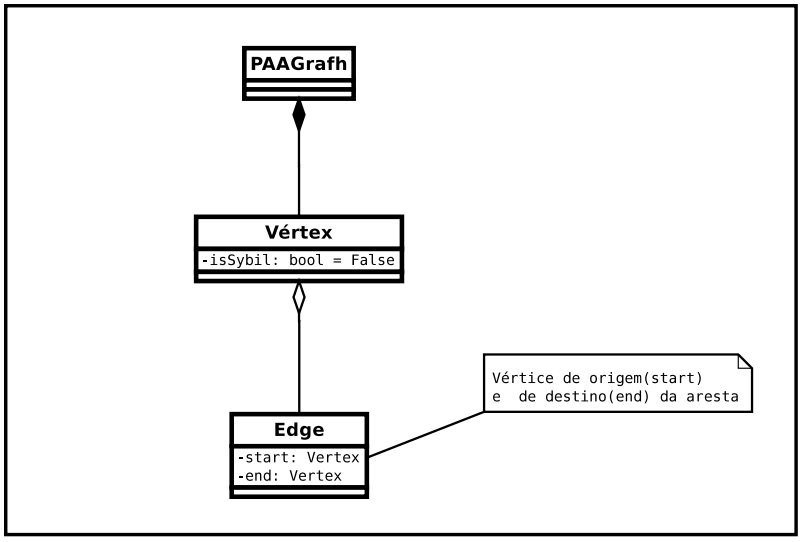

Figura 2. Diagrama de Classe de um Grafo

Para cada vértice pertencente ao conjunto V existe um atributo denominado *isSybil* que determina se aquela vértice pertence à região *honesta (A)* ou à região *Sybil (B)*. Inicialmente todo os vértices são considerados como *Sybil*, ou seja, $V = B$. Posteriormente, com base em informações previamente fornecidas, um total de k vértice serão definidos como *honestos*. Este conjunto $S \in V$ para o qual se sabe previamente serem honesto será denominado *sementes*, e a partir destas sementes os demais vértices serão classificados. O processo de seleção das "sementes"bem como da classificação dos vértices é detalhado na Subseção 2.2.

2.2. Particionamento do Grafo

Estudos vêm demonstrando que um possível mecanismo contra ataques é por meio da implementação de *Algoritmo de Detecção de Comunidades (ADC)* [Viswanath et al. 2011]. Este tipo de algoritmo consiste basicamente em identificar vértices formando uma comunidade local em torno de determinado vértice garantidamente honesto. Os demais vértices, que estão fora da comunidade, são considerados uma ameaça, e portanto classificados como *Sybil*. Uma possível maneira de classifica uma comunidade é através de sua *Condutância Normalizada*.

2.2.1. Condutância Normalizada

Neste trabalho, como técnica de detecção de comunidade utilizou-se o conceito de *Condutância* de um Grafo, que é uma métrica que define a qualidade de uma única comunidade [Kannan et al. 2004]. A condutância de um grafo $G = (V, E)$, onde $A \in V$ e $B = V - A$ formam duas comunidades distintas é definido conforme a Equação 1, onde e_{AB} é o número de aresta entre os vértice da comunidade A e B e e_{AA} representa o número de aresta entre os vértices da comunidade A.

$$C = \frac{e_{AB}}{e_{AA}} \tag{1}$$

No entanto, a definição de condutância conforme descrito na 1 não é uma boa medida para a qualidade de uma comunidade, quando se está analisando comunidades de tamanhos maiores [Mislove et al. 2010], como é caso em redes sociais. Por exemplo, caso todo os vértices fossem classificados em uma única comunidade, como por exemplo a região honesto, a condutância resultante seria igual a *zero*, fornecendo poucas informações sobre a comunidade formada. Visando definir uma métrica aplicável em comunidades maiores foi proposto no trabalho de [Mislove et al. 2010] a *Condutância Normalizada* de um grafo.Para derivar a condutância normalizada, deve-se definir o valor K de uma comunidade A qualquer conforme Equação 2. O valor k é similar à condutância, exceto por estar no intervalo $[0, 1]$.

$$K = \frac{e_{AA}}{e_{AA} + e_{AB}} \tag{2}$$

A Condutância Normalizada CN de uma comunidade A é como K menos o valor esperado de K para um grafo particionado em comunidades de tamanho $|A|$ e $|B|$. Para calcular o valor esperado de K, é necessário calcular o valor esperado

de e_{AA} e e_{AB} mas sem levar em conta as arestas das comunidades. Desse modo, a Condutância Normalizada CN é definida conforme a Equação 3, onde $e_A = e_{AA}+e_{AB}$ que determina o número de aresta que alcançam os vértices na comunidade A e $e_B = e_{BB} + e_{AB}$ representa a mesma quantidade para a comunidade B.

$$CN = \frac{e_{AA}}{e_{AA} + e_{AB}} - \frac{e_A \times e_B}{e_A \times e_A + e_A \times e_B} \tag{3}$$

Os valores de CN estão definidos no intervalo $[-1, 1]$ onde *valores positivos* indicam uma estrutura de comunidade significativa; um valor *0 (zero)* indica que não há uma estrutura de comunidade, e um *valor negativo* indica que a estrutura de comunidade é menor do que a de um grafo aleatório. Uma propriedade particularmente útil dessa definição é que ela é comparável entre grafos de tamanhos e densidades diferentes.

A partir dos valores calculados de CN é possível particionar um grafo em uma região honesta (comunidade A) e região Sybil (comunidade B). A ideia básica é definir uma comunidade inicial A em que inicialmente todos os vértices são garantidamente honesto e permitir a inclusão de um vértice v na comunidade A apenas se a inclusão de v aumentar a *Condutância Normalizada* de A. O algoritmo guloso que particiona um grafo com base na *Condutância Normalizada* é descrito na Subseção 2.2.2.

2.2.2. Algoritmo Guloso para Particionamento de um Grafo

Suponha que nos seja dado um grafo $G(V, E)$ para o qual se deseja particioná-lo em duas regiões A e B. A região A representa os vértices dito *honestos* e a região B conterá os vértices classificados como *Sybil*. Suponha ainda que conheçamos previamente que exista um conjunto $H \in V$ de vértices *garantidamente* honestos. A partir do conjunto H é definido um subconjunto $S \in H$ escolhidos aleatoriamente de forma a produzir inicialmente o conjunto A.

Procedendo conforme descrito no parágrafo anterior, o G é dividido em duas partições A e B, inicialmente com $A = S$ e $B = (V - S)$. A cada passo, é selecionado um vértice $v \in B$ ao *apenas será adicionado ao conjunto A caso v produza um aumento no valor CN de A*. Esse processo se repete adicionando vértices em A até que não reste nenhum vértice que aumente o valor de CN . Então, o algoritmo pára e retorna como saída a comunidade A definida como a região honesta. De forma análoga, $B = (V - A)$ será considerada como a região Sybil. O Algoritmo 1 demonstra todos os passos deste processo.

Conforme pode ser observado o algoritmo escolhe de forma aleatória um conjunto de sementes S (*linha 7*. Este conjunto se torna inicialmente o conjunto honesto A (*linha 8*). Posteriormente os demais vértices $V - S$ são adicionados a uma fila Q e serão inseridos nos conjuntos A ou B (*linhas 11 a 19*). Em suma, um vértice é inicialmente definido como honesto (*linha 13*), caso esta suposição aumento o valor CN do grafo (*linha 15*) o vértice é mantido em A. Do contrário ele é removido e adicionado em B (*linhas 18 e 19*). O algoritmo termina quando fila Q estiver vazia.

O tempo de execução do Algoritmo 1 dependerá basicamente do tamanho do conjunto S e da função CALCULE-CN. A seção 3 discutirá com mais detalhes a complexidade de tempo e de espaço do algoritmo. Contudo, podemos ressaltar neste ponto que $|H| = 100$ e $|S| = 20$, ou seja, para cada grafo analisado foi definido inicialmente um total de 100 vértices honestos, deste total 20 são escolhidos randomicamente para ser as sementes do conjunto A.

Algorithm 1: Algoritmo Guloso para definição de Sybil's em um grafo.

Input: Um grafo $G(V, A)$ e um conjunto de vértices $H \in V$ definidos como honestos

Output: Os subconjuntos $A, B \in V$ tal que $A \cap B = \emptyset$, onde A contêm os vértices classificados como honesto e B os vértices classificados como Sybil

```
1  A ← ∅
2  B ← ∅
3  Q ← ∅
4  S ← ∅
5  CN_current ← 0
6  CN_new ← 0
7  S ← CHOOSE-RANDOM-SEED(H)
8  A ← S
9  Q ← (V − S)
10 CN_current ← CALCULE-CN(G(V, E))
11 while Q ≠ ∅ do
12     v ← DEQUEUE(Q)
13     A ← A ∪ v
14     CN_new ← CALCULE-CN(G(V, E))
15     if CN_new ≥ CN_current then
16         CN_current ← CN_new
17     else
18         A ← A − v
19         B ← B ∪ v
20 return A, B
```

2.3. Cálculo das Métricas

Com o objetivo de mensurar a qualidade dos resultados obtidos pelo algoritmo proposto neste trabalho, foi utilizado um conjunto dez métricas. A partir destas métricas é possível comparar a qualidade da resposta obtida bem como verificar o comportamento do algoritmo para grafos de diferentes estruturas. As métricas utilizadas são:

(a) grau médio dos vértices:
(b) modularidade[1]

[1]Conforme proposto em [Mislove et al. 2010]

(c) condutância da região Sybil
(d) condutância da região honesta
(e) coeficiente de agrupamento da região Sybil
(f) coeficiente de agrupamento da região honesta
(g) fração de vértices Sybil corretamente classificados;
(h) fração de vértices honestos corretamente classificados
(i) fração de falsos positivos
(j) fração de falsos negativos de forma incorreta como honestos).

3. Análise de Complexidade

3.1. Complexidade Temporal

Conforme discutido anteriormente o principal algoritmo proposto neste trabalho é aquele descrito no pseudocódigo do Algoritmo 1. Aquele algoritmo dependerá basicamente do tamanho do conjunto de sementes S e do tempo de execução dos métodos CHOOSE-RANDOM-SEED e CALCULE-CN.

A função CHOOSE-RANDOM-SEED têm como objetivo escolher randomicamente $|S|$ vértices em um conjunto de tamanho $|H|$. Esta escolha pode ser feita no pior caso em tempo linear ao tamanho do conjunto H. Desta forma a complexidade de tempo do método CHOOSE-RANDOM-SEED pode ser definida como $O(H)$;

A *Condutância Normalizada* obtida através do método CALCULE-CN é calculada avaliando cada uma das arestas dos vértices em V. Neste sentido, no pior caso, a complexidade de tempo da função CALCULE-CN será dada por $O(V \times A)$. Contudo, conforme pode ser observado na *linha 14* do Algoritmo 1, o método CALCULE-CN será executado $|V - S|$ vezes. Neste sentido, a participação de CALCULE-CN na complexidade do algoritmo 1 será de $(V - S) \times V \times A$. Contudo, se consideramos que $S \ll V$ a complexidade de tempo do Algoritmo 1 será dada por $O(V^2 \times A)$.

3.2. Complexidade Espacial

A Subseção 2.1 descreveu que neste trabalho foi utilizado uma representação de grafo onde cada vértice armazena um conjunto de suas sua arestas (vide Figura 2). Partindo desta premissa a *Complexidade de Espaço* do algoritmo será de $O(V + A)$, ao qual se refere-se no espaço necessário para armazenar os $|V|$ vértice e as $|A|$ arestas. Haverá a necessidade de armazenar os conjuntos A e B, contudo, como $A \cup B = V$, assintoticamente não mudaria o valor da complexidade de espaço do algoritmo.

4. Análise Experimental

Com o objetivo de verificar a qualidade das respostas dadas pelo algoritmo foi recuperado o conjunto de métricas descritos na Subseção 2.3. Os testes consistem em executar o algoritmo por *100* vezes para cada um dos grafos de entrada. Esse procedimento é necessário tendo em vista a escolha aleatória das sementes. Os testes foram executados em computador com o sistema operacional Ubuntu versão 12.04

GRAFO	MÉTRICAS			
	grau médio dos vértices	modularidade	Condutância Sybil	Condutância Honesta
A	15.7708	0,825	14765.7	16276.9
B	15.9552	0.899	16135	12504.3

Tabela 1. Métricas Grafos A e B - Parte 01/02

kernel 3.13.0-37-generic 64 bits e com 4GB de memória RAM. O valor médio para cada uma das execuções estão exibidos nas Tabelas 1 e 2

É possível verificar que mesmo para grafos de estrutura diferentes (Grafo A e B) o algoritmo apresentou resultando semelhantes mostrando que o uso da *Condutância Normalizada* conseguiu melhorar a qualidade dos resultados independente do tamanho e estrutura das comunidades existentes nos grafos.

GRAFO	MÉTRICAS					
	Coef. Agrup Sybil	Coef. Agrup Honesto	Sybil Corretos	Honestos Corretos	falsos positivos	falsos negativos
A	0.0912714	0.0959462	1	0,09766	0.474333	0
B	0.0891779	0.0937305	1	0.86	0.122807	0

Tabela 2. Métricas Grafos A e B - Parte 02/02

5. Conclusão

Este trabalho resolver o problema de particionar um grafo em duas em duas regiões, a região *honesta (A)* e a região *Sybil (B)*. Utilizando um algoritmo guloso que faz uso da métrica *Condutância Normalizada* foi possível detectar de forma satisfatória vértices como Sybil. O fato de utilizar a Condutância Normalizada possibilitou a aplicação do algoritmo em grafos de diferente estrutura e tamanhos, obtendo resultados semelhantes para cada um deles.

Referências

Kannan, R., Vempala, S., and Vetta, A. (2004). On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515.

Mislove, A., Viswanath, B., Gummadi, K. P., and Druschel, P. (2010). You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM.

Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.

Viswanath, B., Post, A., Gummadi, K. P., and Mislove, A. (2011). An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 41(4):363–374.