Representações Implícitas Probabilísticas de Grafos

Juan P. A. Lopes[1], **Fabiano de S. Oliveira**[1]*, **Paulo E. D. Pinto**[1]*

[1]IME/DICC, Universidade do Estado do Rio de Janeiro (UERJ)
Av. São Francisco Xavier, 524, Maracanã – 20550-013 – Rio de Janeiro – RJ

me@juanlopes.net, fabiano.oliveira@ime.uerj.br, pauloedp@ime.uerj.br

Abstract. *This paper introduces the concept of probabilistic implicit graph representations, extending the definition from [Spinrad 2003] by allowing the adjacency test to have a constant probability of false positives or false negatives. It also discusses two novel representations based on well-known probabilistic data structures: one that uses Bloom filters and can represent general graphs with the same space complexity as the adjacency matrix (but outperforms it for sparse graphs), and other that uses MinHash and can represent trees with lower space complexity than any deterministic implicit representation.*

1. Introdução

Uma *representação implícita de grafos* é um esquema de rotulação dos vértices que resulta numa representação eficiente em espaço e permite o teste de adjacência eficiente de um par de vértices através da comparação de seus rótulos [Muller 1988, Kannan et al. 1992, Spinrad 2003]. Mais formalmente, dada uma classe de grafos C com $2^{f(n)}$ membros de n vértices, uma representação é dita *implícita* se

1. é *assintoticamente ótima*, requerendo $O(f(n))$ bits para representar grafos de C;
2. *distribui informação uniformemente* entre os vértices, ou seja, cada vértice é representado por um *rótulo* usando $O(f(n)/n)$ bits;
3. o *teste de adjacência é local*, isto é, para testar a adjacência entre qualquer par de vértices, somente seus rótulos são usados no processo.

Desta definição, segue-se que a *matriz de adjacências* é uma representação implícita para a classe que contém todos os grafos, pois há $2^{\Theta(n^2)}$ grafos com n vértices e a matriz de adjacência pode representá-los utilizando $\Theta(n^2)$ bits. Por outro lado, a *lista de adjacências* não é uma representação implícita, pois requer $\Theta(m \log n)$ bits para representar a mesma classe de grafos, que pode ser $\Theta(n^2 \log n)$ bits no pior caso (grafos completos).

Neste artigo introduzimos o conceito de *representação implícita probabilistica de grafos*, que estende o conceito de representação implícita por relaxar uma das propriedades: o teste de adjacência admite uma probabilidade constante de falsos negativos ou falsos positivos (0% de probabilitade de falsos positivos/negativos implica uma representação implícita regular). Representações probabilísticas são mais eficientes em termos de espaço por permitir alguns resultados incorretos. Esta característica permite representar muitos grafos que não caberiam na memória principal com representações tradicionais. Em especial, beneficiam-se aplicações de grafos onde não é tão importante obter respostas exatas. Redes sociais, por exemplo, possuem grafos grandes e dinâmicos, onde mesmo operações simples como contagem de componentes não terminariam antes

*Projeto parcialmente financiado por FAPERJ.

do grafo provavelmente já ter sido modificado. Apresentaremos, neste artigo, duas novas representações implícitas probabilísticas, cada uma baseada em uma *estrutura de dados probabilística* distinta.

Estruturas de dados probabilísticas permitem representação eficiente de dados, com menores requisitos de espaço que as estruturas determinísticas. Neste artigo, usamos duas estruturas baseadas em *hash* para criar representações probabilísticas: *Filtro de Bloom* [Bloom 1970], que permite o teste de pertinência a um conjunto com 1% de falsos positivos, usando menos que 10 bits por elemento; e *MinHash* [Broder 1997, Li and König 2010], que permite estimar, para os conjuntos A e B, o coeficiente de *Jaccard* $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$, representando cada conjunto com um número constante de bits.

2. Representação baseada em Filtro de Bloom

O Filtro de Bloom é uma estrutura de dados que representa um conjunto S e permite o teste de pertinência de elementos com alguma probabilidade de falsos positivos, sem admitir, entretanto, falsos negativos [Bloom 1970]. É implementada como um array M de m bits e k funções hash, independentes duas a duas $h_i : S \to [1; m]$ para $1 \leq i \leq k$. A inserção de um elemento x é feita computando k valores hash e atribuindo 1 aos índices correspondentes do array, ou seja, $M[h_i(x)] \leftarrow 1$ para $1 \leq i \leq k$. A pesquisa de pertinência para algum elemento x é feita verificando-se se todos os bits das posições referenciadas pelos valores hash estão ligados, isto é, $M[h_i(x)] = 1$ para todo $1 \leq i \leq k$. Se pelo menos um bit for 0, significa que, com 100% de certeza, o elemento não está no conjunto. Se todos os bits são 1, então o elemento pertence ao conjunto com alta probabilidade, A probabilidade de falsos positivos pode ser determinada pela probabilidade de colisões em todos os k valores hash, que é

$$\Pr[\textsc{FalseP}] = \Pr\left[\bigwedge_{1 \leq i \leq k} M[h_i(x)] = 1\right] = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - \mathrm{e}^{-kn/m}\right)^k$$

Definindo $q = m/n$ como o número de bits por elemento de S no array M, é possível mostrar que a probabilidade de falsos positivos é minimizada quando $k = q \ln 2$. Então, temos: $\Pr[\textsc{FalseP}] \approx (1 - e^{-\ln(2)})^{q \ln(2)} \approx (0.6185)^q$. Com 10 bits por elemento e 7 funções hash, estima-se a pertinência com menos de 1% de falsos positivos portanto.

Filtros de Bloom podem ser diretamente aplicados para representar as adjacências de cada vértice. Para tanto, cria-se um filtro de Bloom para cada vértice, representando as adjacências do mesmo. O conjunto dos filtros de Bloom para todos os vértices constitui uma representação implícita probabilística. Esta representação requer $\Theta(m)$ bits para representar qualquer grafo, o que a torna equivalente à matriz de adjacências no pior caso (grafos completos, por exemplo). Entretanto, esta representação tem menor complexidade de espaço para grafos esparsos que a representação determinística. De fato, ela é melhor para qualquer grafo com $m = o(n^2)$. Note que ela tem a propriedade de não permitir falsos negativos no teste de adjacência e, portanto, nunca deixará de reportar uma aresta existente, embora possa reportar uma não-aresta com pequena probabilidade.

3. Representação baseada em MinHash

MinHash (também conhecida como *min-wise hashing*) é uma estrutura de dados probabilística que representa conjuntos A e B e permite estimar seu coeficiente de Jaccard

$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ [Broder 1997]. Para tanto, computa-se a assinatura para cada conjunto S, usando k funções hash $h_1, \ldots, h_k$, independentes duas a duas. Cada elemento na assinatura é o valor hash mínimo $h_i^{\min}(S) = \min\{h_i(x) : x \in S\}$ para $1 \leq i \leq k$. A probabilidade de dois conjuntos A e B terem um elemento comum às assinaturas é igual ao seu coeficiente de Jaccard, ou seja, $\Pr[h_i^{\min}(A) = h_i^{\min}(B)] = J(A,B)$. Então, a comparação de elementos correspondentes nas duas assinaturas forma um conjunto independente de estimadores não-enviesados para o coeficiente de Jaccard dos dois conjuntos. Assim, o aumento do número k de elementos da assinatura decresce a variância do estimador. Os limites de erro na estimativa de similaridade podem ser provados usando a desigualdade de Chernoff. Em especial, para atingir um fator de erro θ com probabilidade maior que $1 - \delta$, k deve ser definido de tal forma que $k \geq \frac{2+\theta}{\theta^2} \times \ln(2/\delta)$.

A idéia principal da representação baseada em MinHash é encontrar, para qualquer grafo $G = (V, E)$ de uma classe C, um método de construir conjuntos de representantes $S_v \neq \emptyset$ para $v \in V$, tal que $J(S_u, S_v) \geq \delta_B$ se e somente se $(u, v) \in E$ e $J(S_u, S_v) \leq \delta_A$ se e somente se $(u, v) \notin E$. Nenhum coeficiente de Jaccard entre conjuntos de representantes do grafo pode estar no intervalo $(\delta_A; \delta_B)$. Desta forma, a aresta (u, v) pode ser testada estimando-se $J(S_u, S_v) > \delta$ para algum $\delta_A \leq \delta \leq \delta_B$. A estimativa pode ser feita usando MinHash. Uma vez que as assinaturas requerem um número constante de elementos (que podem ser representados com um número constante de bits [Li and König 2010]), uma representação baseada em MinHash requer $O(n)$ bits para representar qualquer classe onde tal construção seja possível.

Apresentamos, a seguir, uma construção dos conjuntos para árvores com $\delta_A = 1/3$ e $\delta_B = 1/2$. Dada uma árvore T, a construção é feita recursivamente, enraizando a árvore em um vértice arbitrário v, definindo um conjunto para S_v com $2\Delta(T)$ inteiros arbitrários. Então, para cada nível na árvore enraizada, o procedimento alterna entre escolher um subconjunto ou estender o conjunto de cada vértice pai (exemplificado na Figura 1).

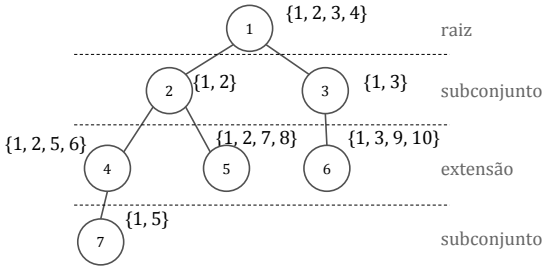

Figura 1. Construção de conjuntos para uma árvore exemplo.

A seleção de subconjuntos é feita tal que, para um conjunto $A = \{a_1, \ldots, a_x\}$, selecionam-se $x/2$ subconjuntos, cada um com $x/2$ elementos, tendo cada par de subconjuntos $x/4$ elementos comuns. Desta forma, os subconjuntos $S_1, \ldots, S_{x/2}$ de A são escolhidos tendo-se $J(S_i, S_j) = 1/3$ para $1 \leq i, j \leq x/2$ e $J(S_i, A) = 1/2$ para $1 \leq i \leq x/2$. O processo da seleção baseia-se na geração de uma cadeia binária s, com $|s| = x/2$ que represente a seleção de um subconjunto $S \subset A$ de modo que se o i-ésimo bit de s tem o valor b, então a_{2i-1+b} pertence a S. A geração das cadeias binárias pode ser feita através de uma construção iterativa, na qual partindo de uma matriz 1×1, a cada passo a matriz é quadruplicada, sendo os bits do quadrante inferior direito invertidos. O processo de

construção, para um conjunto de 8 elementos, é exemplificado na Figura 2. A extensão de um conjunto A se dá pela inclusão de $|A|$ novos elementos únicos a toda a árvore.

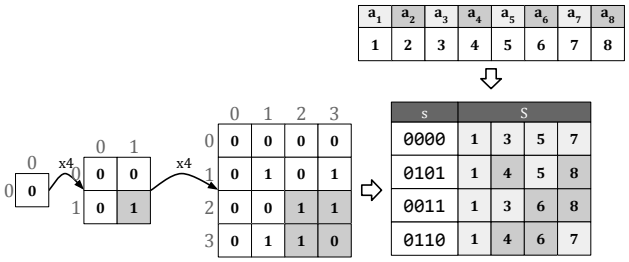

Figura 2. Exemplo de seleção de subconjuntos.

A assinatura MinHash é então computada para cada conjunto construído. As assinaturas são usadas como rótulos para os vértices correspondentes. Como este esquema de rotulação requer apenas $O(n)$ para representar árvores probabilisticamente, que é uma classe com $2^{\Theta(n \log n)}$ grafos com n vértices, podemos dizer que ela tem melhor complexidade de espaço que a representação determinística ótima.

4. Considerações sobre grafos bipartidos

[Spinrad 2003] mostra que qualquer classe hereditária de grafos com $2^{\Theta(n^2)}$ membros com n vértices deve incluir inteiramente uma das classes: bipartidos, co-bipartidos ou split. Além disso, é possível transformar qualquer grafo $G = (V, E)$ para um grafo bipartido $G' = (V', E')$ tal que $V' = \{v_1, v_2 : v \in V\}$ e $E' = \{(v_1, u_2), (u_1, v_2) : (v, u) \in E\}$. Qualquer representação eficiente para G' pode ser usada para representar eficientemente G. Isto torna a pesquisa de representações implícitas para grafos bipartidos especialmente relevante. Entretanto, é possível provar que não existem certas representações probabilísticas. Por exemplo, é impossível construir uma representação baseada em MinHash com $\delta_A = 0.4$ e $\delta_B = 0.6$ para $K_{3,3}$, mostrando a inviabilidade de solução de um problema de programação linear inteira. A decisão da existência ou não de representação implícita probabilística para grafos em geral permanece um problema em aberto.

Referências

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE.

Kannan, S., Naor, M., and Rudich, S. (1992). Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603.

Li, P. and König, A. C. (2010). b-bit minwise hashing. In *Nineteenth International World Wide Web Conference (WWW 2010)*. Association for Computing Machinery, Inc.

Muller, J. H. (1988). *Local structure in graph classes*. PhD thesis, Georgia Institute of Technology.

Spinrad, J. P. (2003). *Efficient graph representations*. American mathematical society.