Problema APSP via Dimensão-VC e Médias de Rademacher

Alane M. de Lima[1], **André L. Vignatti**[1], **Murilo V. G. da Silva**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – Curitiba – PR – Brazil

{amlima,vignatti,murilo}@inf.ufpr.br

Abstract. *We consider the version of the All-pairs Shortest Paths (APSP) problem, where we are only required to compute paths with high centrality, such that the centrality metric reflects the "importance" of a path in the graph. We propose an algorithm for this problem that uses a sampling approach based on VC-Dimension and Rademacher averages. In the case of sparse graphs with logarithmic diameter, which is a common model for several real-world scenarios, the sample size is exponentially smaller than those obtained by classical techniques (e.g. Hoeffding and union bound).*

Resumo. *Considere o problema de computar o caminho mínimo entre cada par de vértices (APSP) de um grafo, desde que o caminho tenha alta centralidade, sendo que a métrica de centralidade em questão reflete a "importância" do caminho no grafo. Propomos um algoritmo para este problema que usa uma estratégia de amostragem baseada em Dimensão-VC e médias de Rademacher. No caso de grafos esparsos de diâmetro logarítmico, que comumente modelam situações reais, o tamanho da amostra é exponencialmente menor do que aquelas obtidas por técnicas clássicas (e.g.: Hoeffding e limitante da união).*

1. Introdução

O caminho mínimo entre todos os pares de vértices de um grafo (*All-pairs Shortest Paths* – APSP) é um problema amplamente estudado. Na prática, entretanto, computá-lo de maneira exata torna-se inviável em grafos de larga escala. Neste artigo propomos um algoritmo de amostragem de tempo esperado $\mathcal{O}(\lg \mathrm{Diam}_V(G) \max(m + n \log n, \mathrm{Diam}_V(G)^2))$, onde $\mathrm{Diam}_V(G)$ é a maior quantidade de vértices em um caminho mínimo de um grafo G (diâmetro em vértices de G) de n vértices e m arestas. O algoritmo computa um caminho mínimo exato P entre todo par de vértices (u, v) se a *centralidade* de P (uma medida de "importância" de P) for maior do que uma constante fixa. Intuitivamente, quanto mais caminhos mínimos tiverem P como subcaminho, maior sua centralidade. Usando técnicas desenvolvidas por [Riondato e Upfal 2018], aplicamos a teoria de Dimensão Vapnik-Chervonenkis (Dimensão-VC) e médias de Rademacher em um algoritmo de amostragem progressiva para provar que amostrar no máximo $\lceil \frac{c}{\epsilon} \lfloor 2 \lg \mathrm{Diam}_V(G) + 1 \rfloor \ln(\frac{1}{\epsilon}) + \ln \frac{1}{\delta} \rceil$ caminhos mínimos (e inspecionar seus subcaminhos) é suficiente para encontrar com probabilidade $1 - \delta$ todos os caminhos mínimos de centralidade mínima ϵ (c é aproximadamente $\frac{1}{2}$). Observamos que o limitante para o tamanho da amostra obtido neste artigo é exponencialmente menor do que se fosse alcançado por meio de técnicas tradicionais como os limitantes da união e de Hoeffding, por exemplo.

Os algoritmos exatos mais eficientes para o problema APSP são propostos por [Williams 2014] (grafos com pesos em $\mathbb{N}$), de tempo $\mathcal{O}\left(\frac{n^3}{2^{c\sqrt{\log n}}}\right)$ para alguma constante $c > 0$, e por [Pettie e Ramachandran 2002] (grafos com pesos em $\mathbb{R}^+$), de tempo $\mathcal{O}(mn \log \alpha(m,n))$, onde $\alpha(m,n)$ é a função de Ackermann inversa. Observe que em comparação a tais trabalhos nosso algoritmo é mais eficiente em grafos densos, e no pior caso, atinge o tempo do algoritmo de [Pettie e Ramachandran 2002] no caso de grafos esparsos. Contudo, em grafos esparsos com diâmetro logarítmico (cenário presente em diversas aplicações reais [Easley e Kleinberg 2010]), nosso algoritmo é melhor do que o de tais autores para o nosso problema, alcançando o tempo esperado de $O(n \log n \log \log n)$.

2. Preliminares

Seja $G = (V, E)$ um grafo ponderado não-direcionado e $n = |V|$ e $m = |E|$. S.p.d.g. assumimos G conexo. Uma *árvore de caminhos mínimos* de $u \in V$ é uma árvore geradora T_u de G onde todo caminho em T_u de u para qualquer $v \in V$ é um caminho mínimo p_{uv} em G. Fixamos uma ordenação arbitrária de V e consideramos T_u como uma árvore retornada pelo algoritmo de Dijkstra, denominando-a *árvore Dijkstra*. Um *ramo* $\mathcal{B}_u(v)$ de T_u é um caminho p_{uv} com início na raiz de T_u e destino em v ($u \neq v$). Cada subcaminho de $\mathcal{B}_u(v)$ é também um caminho mínimo em G, e denotamos tal conjunto de subcaminhos (incluindo p_{uv}) por $S(p_{uv})$ ou $S(\mathcal{B}_u(v))$ (por conveniência, ambas as notações são usadas). Como G é não-direcionado, então p_{vu} também é um caminho mínimo em G e $S(p_{vu})$ denota todos os seus subcaminhos (incluindo p_{vu}).

Definição 1 (Centralidade de Caminho Mínimo)**.** *Dados $G = (V, E)$, $(u, v) \in V^2$ e T_a $\forall a \in V$, a* centralidade de caminho mínimo *de (u, v) é $c(u, v) = \frac{t_{uv}}{n(n-1)}$ onde $t_{uv} = \sum_{(a,b) \in V^2 : a \neq b} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$, $\tau_{uv} = \{\mathcal{B}_c(d) \in \bigcup_{a \in V} \bigcup_{b \in V : b \neq a} \mathcal{B}_a(b) : p_{uv} \in S(\mathcal{B}_c(d))\}$.*

A *Dimensão-VC* fornece limitantes superiores à complexidade de amostras, isto é, o tamanho mínimo que uma amostra deve ter para que parâmetros de erro e confiança sejam atingidos, baseada na estrutura do espaço de intervalos que modela um determinado problema. Contudo, encontrar limitantes justos para o tamanho fixo de uma amostra pode ser uma tarefa complexa. Neste caso, uma alternativa é usar um algoritmo baseado em amostragem progressiva, isto é, que aumenta a amostra iterativamente enquanto o parâmetro do limitante máximo para o erro não é atingido. A ideia central consiste no fato que a condição de parada (quando a amostra é grande o suficiente) é baseada em Médias de Rademacher, um conceito que reside no núcleo de teoria de aprendizado estatístico.

Um *espaço de intervalos* é um par $\mathcal{R} = (U, \mathcal{I})$ onde U é um domínio (finito ou infinito) e $\mathcal{I}$ é uma coleção de subconjuntos de U (*intervalos*). Dado $S \subseteq U$, a *projeção* de $\mathcal{I}$ em S é o conjunto $\mathcal{I}_S = \{S \cap I \mid I \in \mathcal{I}\}$. Dizemos que S é *despedaçado* por $\mathcal{I}$ se $|\mathcal{I}_S| = 2^{|S|}$. A Dimensão-VC de $\mathcal{R} = (U, \mathcal{I})$ é $\text{VCDim}(\mathcal{R}) = \max\{k : \exists S \subseteq U$ tal que $|S| = k$ e $|\mathcal{I}_S| = 2^k\}$. De modo geral, para um conjunto de valores de interesse $\mathcal{H}$, existe uma família de funções $\mathcal{F}$ de U a $\mathbb{R}^*$ tal que há uma $f_h \in \mathcal{F}$, $\forall h \in \mathcal{H}$. Seja $S = (z_1, \ldots, z_r)$ uma coleção de r elementos de U amostrados com respeito a π. Então $\forall f_h \in \mathcal{F}$, $L_S(f_h) = (1/r) \sum_{s \in S} f_h(s)$ e $L_U(f_h) = \mathbb{E}_{u \in U}[f_h(u)]$.

Teorema 1 ([Har-Peled e Sharir 2011])**.** *Dados $\mathcal{R} = (U, \mathcal{I})$ com VCDim$(\mathcal{R}) \leq k$, uma distribuição de probabilidades π em U, $0 < \epsilon, \delta < 1$ e $c > 0$:*

1. $\Pr(|L_S(f_h) - L_U(f_h)| \leq \epsilon) \geq 1 - \delta$, $\forall f_h \in \mathcal{F}$, *para uma coleção de elementos* $S \subseteq U$ *amostrados segundo* π *tal que* $|S| = (c/\epsilon^2)\,(k + \ln(1/\delta))$.
2. $\Pr(|I \cap S| \geq 1$ *se* $\Pr_\pi(I) \geq \epsilon) \geq 1 - \delta$, $\forall I \in \mathcal{I}$, *para uma coleção de elementos* $S \subseteq U$ *amostrados segundo* π *tal que* $|S| = (c/\epsilon)\,(k \ln 1/\epsilon + \ln(1/\delta))$.

Definição 2. *A média empírica de Rademacher de uma família de funções* $\mathcal{F}$ *com respeito a* S *e uma dist.* σ *de* r *variáveis aleatórias de Rademacher com* $\Pr(\sigma_i = 1) = \Pr(\sigma_i = -1) = 1/2$ $(1 \leq i \leq r)$ *é definida como* $\tilde{R}_r(\mathcal{F}, S) = \mathbb{E}_\sigma\left[\sup_{f_h \in \mathcal{F}} (1/r) \sum_{i=1}^{r} \sigma_i f_h(z_i)\right]$.

Teorema 2 ([Oneto et al. 2013])**.** $\sup_{f_h \in \mathcal{F}} |L_S(f_h) - L_U(f_h)| \leq 2\tilde{R}_r(\mathcal{F}, S) + \ln\frac{3}{\delta} + \left(\sqrt{(\ln\frac{3}{\delta} + 4r\tilde{R}_r(\mathcal{F}, S))\ln\frac{3}{\delta}}\right)/r + \sqrt{\ln(\frac{3}{\delta})/(2r)}$ *com probabilidade pelo menos* $1 - \delta$.

Teorema 3 ([Riondato e Upfal 2018])**.** *Considere* $v_{f_h} = (f_h(z_1), \ldots, f_h(z_r))$ *em uma amostra* $S = \{z_1, \ldots, z_r\}$, $|S| = r$, *e* $\mathcal{V}_S = \{v_{f_h}, f_h \in \mathcal{F}\}$. *Então* $\tilde{R}_r(\mathcal{F}, S) \leq \min_{s \in \mathbb{R}^+} w(s)$, *onde* $w(s) = \frac{1}{s}\ln\sum_{v_{f_h} \in \mathcal{V}_S} \exp\left(s^2 ||v_{f_h}||_2^2/(2r^2)\right)$.

3. Algoritmo Proposto

Nesta seção definimos $c(u, v)$ em termos de um espaço de intervalos e dados $0 < \delta, \epsilon < 1$, apresentamos o Algoritmo 1 que retorna as distâncias entre caminhos com $c(u, v) \geq \epsilon$ (tabela d) de G, com probabilidade $1 - \delta$. A tabela d contém as distâncias que foram computadas pelo algoritmo. Além disso, são retornadas as estimativas da centralidade entre cada par de vértices (tabela $\tilde{c}$), se assim for desejado. A atualização da tabela de árvores canônicas $\tilde{t}$ e do conjunto $\mathcal{V}$ (que contém os valores de $\tilde{t}$ sem repetição) é feita de modo análogo ao Algoritmo 2 em [Riondato e Upfal 2018]. Dado um cronograma de crescimento de amostra $(|S_i|)_{i \geq 1}$, seja $\mathcal{T}$ o conjunto de n árvores Dijkstra de G; temos que $\mathcal{H} = V^2$ e $U = \bigcup_{a \in V} \bigcup_{b \in V : b \neq a} \mathcal{B}_a(b)$, e $\forall (u, v) \in V^2$, $\mathcal{I} = \{\tau_{uv} : (u, v) \in V^2\}$. Para cada $\mathcal{B}_a(b) \in U$, seja a função $f_{uv} : U \to \{0, 1\}$ definida como $f_{uv}(\mathcal{B}_a(b)) = \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$. Temos que $\mathcal{F} = \{f_{uv} : (u, v) \in V^2\}$. Cada $\mathcal{B}_a(b)$ é amostrado com probabilidade $\frac{1}{n(n-1)}$ em U e $\mathbb{E}[f_{uv}(\mathcal{B}_a(b))] = L_U(f_{uv}) = c(u, v)$, $\forall (u, v) \in V^2$. Seja $S = \{\mathcal{B}_{a_i}(b_i), 1 \leq i \leq r\}$ um conjunto de r ramos amostrados independentemente em U. A estimativa da centralidade é $\tilde{c}(u, v) = L_S(f_{uv}) = \frac{1}{r}\sum_{\mathcal{B}_{a_i}(b_i) \in S} f_{uv}(\mathcal{B}_{a_i}(b_i))$. Por argumentos semelhantes ao Teorema 4.1 em [de Lima et al. 2020], provamos um limitante superior para $\mathrm{VCDim}(\mathcal{R})$. A corretude e o limite superior para o tempo de execução do algoritmo são demonstrados nos Teoremas 5 e 6. Por limitações de espaço, as provas dos lemas enunciados nesta seção serão omitidas neste trabalho e apresentadas em sua respectiva versão estendida.

Teorema 4. *VCDim*$(\mathcal{R}) \leq \lfloor 2 \lg \mathit{Diam}_V(G) + 1 \rfloor$.

Na amostragem progressiva, sejam S_1 e δ_1 os valores iniciais do tamanho da amostra e de δ, respectivamente. Então $S_1 \geq (\ln(6/\delta)(1 + \sqrt{1 + 8^2\epsilon^2}))/(4\epsilon^2)$ (aplicação do Teorema 2 considerando $\delta_1 = \delta/2$, $\tilde{\mathcal{R}}_r(\mathcal{F}, S_1) \geq 0$ e um limite superior de ϵ). Em cada iteração i, $S_i = g^i S_1$, $\forall i \geq 1$, para uma constante $g > 1$.

Teorema 5. *Considere* $S_r \subseteq U$ *de tamanho* r, *e seja* η_i *o valor de* η *na* i*-ésima iteração do laço 2–11. Então* $\eta_r = 2w_s + \ln\frac{3}{\delta_r} + \sqrt{(\ln\frac{3}{\delta_r} + 4|S_r|w_s)\ln\frac{3}{\delta_r}}/|S_r| + \sqrt{\ln\frac{3}{\delta_r}/(2|S_r|)}$, *onde* $\delta_r = \delta/2^r$, *é o valor em que* $r = \min\{i \geq 1\}$ *tal que* $\eta_r \leq \epsilon$ *para o grafo* $G = (V, E)$ *e constantes* $0 < \epsilon, \delta < 1$. *O Algoritmo 1 retorna com probabilidade* $1 - \delta$ *a distância exata* $d(u, v)$, $\forall (u, v) \in V^2$, *tal que* $d(u, v) > 0$ *se* $c(u, v) \geq \epsilon$, *e garante* $\Pr(|\tilde{c}(u, v) - c(u, v)| \leq \epsilon) \geq 1 - \delta$.

Teorema 6. *Dado $G = (V, E)$, o Algoritmo 1 tem tempo esperado $\mathcal{O}(\lg \textit{Diam}_V(G) \cdot \max(m + n \log n, \textit{Diam}_V(G)^2))$ para computar d e $\tilde{c}$ em uma amostra de tamanho $r = \lceil \frac{c}{\epsilon}(\lfloor 2 \lg \textit{Diam}_V(G) + 1 \rfloor \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta}) \rceil$ e $r = \lceil \frac{c}{\epsilon^2}(\lfloor 2 \lg \textit{Diam}_V(G) + 1 \rfloor + \ln \frac{1}{\delta}) \rceil$, respectivamente.*

Algoritmo 1: APSP_PROBABILÍSTICO(G,ϵ,δ,$(S_i)_{i \geq 1}$)

$|S_0| \leftarrow 0,\ i \leftarrow 0,\ \mathcal{V} \leftarrow \emptyset,\ \tilde{t} \leftarrow 0, d \leftarrow 0$
faça
  $i \leftarrow i + 1$
  para $l \leftarrow 1$ **até** $|S_i| - |S_{i-1}|$ **faça**
    amostre $a \in V$ com probabilidade $\frac{1}{n}$
    execute Dijkstra, obtendo T_a e atualizando as distâncias d
    amostre $b \in \{T_a \setminus \{a\}\}$ com probabilidade $\frac{1}{n-1}$ para obter $\mathcal{B}_a(b)$
    atualize $\tilde{t}$ e $\mathcal{V}$ por meio do Algoritmo 2 em [Riondato e Upfal 2018]
  $w_s \leftarrow \min_{s \in \mathbb{R}^+} \frac{1}{s} \ln \sum_{t \in \mathcal{V}} \exp(s^2 t/(2|S_i|^2))$ e $\delta_i \leftarrow \delta/2^i$
  $\eta \leftarrow 2w_s + \ln \frac{3}{\delta_i} + \sqrt{(\ln \frac{3}{\delta_i} + 4|S_i| w_s) \ln \frac{3}{\delta_i}}/|S_i| + \sqrt{\ln \frac{3}{\delta_i}/(2|S_i|)}$
enquanto $\eta > \epsilon$
retorne d

4. Considerações Finais

Apresentamos um algoritmo baseado em amostragem que executa em tempo esperado $\mathcal{O}(\lg \text{Diam}_V(G) \cdot \max(m + n \log n, \text{Diam}_V(G)^2))$ que retorna a distância exata entre (u, v) com probabilidade $1 - \delta$ se a centralidade de caminho mínimo de (u, v) é pelo menos ϵ. No caso de grafos esparsos de diâmetro logarítmico (presentes em diversas aplicações reais), nosso algoritmo executa em tempo esperado $O(n \log n \log \log n)$.

Referências

de Lima, A. M., da Silva, M. V. G., e Vignatti, A. L. (2020). Estimating the percolation centrality of large networks through pseudo-dimension theory. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Easley, D. e Kleinberg, J. (2010). *Networks, crowds, and markets*. Cambridge Un. Press.

Har-Peled, S. e Sharir, M. (2011). Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496.

Oneto, L., Ghio, A., Anguita, D., e Ridella, S. (2013). An improved analysis of the rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44:107 – 111.

Pettie, S. e Ramachandran, V. (2002). Computing shortest paths with comparisons and additions. In *Proc. of ACM-SIAM Symp. on Discrete Algorithms*, pages 267–276.

Riondato, M. e Upfal, E. (2018). Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *ACM Transactions on Knowledge Discovery from Data*, 12(5):61:1–61:38.

Williams, R. (2014). Faster all-pairs shortest paths via circuit complexity. In *Proc. of the Forty-sixth Annual ACM Symp. on Theory of Computing*, STOC '14, pages 664–673.