ALGORITMOS PARA AMOSTRAR ÁRVORES GERADORAS

NATHAN BENEDETTO PROENÇA

1. Motivação

Em projeto e análise de algoritmos há grande valor em assumir que as entradas ocorrem de acordo com uma distribuição de probabilidade específica. Analisar o tempo de execução esperado, por exemplo, só é possível com esse tipo de hipótese. Uma distribuição torna possível raciocinar com mais cuidado sobre a relevância de casos nos quais o algoritmo irá demorar muito, por exemplo. Apenas esta distinção já permite que muitas análises de tempo esperado tragam resutados mais desejáveis do que análises de pior caso.

O algoritmo *Quicksort* é, discutivelmente, o exemplo mais conhecido deste fenômeno. É possível exibir casos nos quais ele leva tempo quadrático para efetuar a ordenação. Contudo, quando comparados com o conjunto de todas as permutações possíveis do vetor, esses eventos são raríssimos. Na maioria das vezes, o *Quicksort* é extremamente eficiente. Uma forma de formalizar essa ideia é assumir que a entrada é uma permutação do vetor dado, escolhida de maneira uniforme dentre todas as permutações possíveis, e afirmar que o tempo esperado de execução é $O(n \log n)$.

Contudo, para fins práticos, não se pode simplesmente assumir que as entradas de fato respeitam a distribuição que o projetista do algoritmo escolheu durante a análise. O que se faz é agir para que as coisas sejam como se deseja. No caso do *Quicksort*, pode-se processar o vetor que será ordenado, a fim de garantir que ele esteja de acordo com a distribuição de probabilidade que for mais conveniente. É comum nas implementações do *Quicksort* que a amostragem seja feita durante o algoritmo, na própria recursão, por razões de otimização. No entanto, isso não muda o fato de que se está impondo uma distribuição na entrada que não estava lá.

Surge então a necessidade de criar algoritmos eficientes para amostrar. Em outras palavras, de escolher em tempo polinomial um objeto dentre vários, de forma que essas escolhas respeitem uma distribuição de probabilidade específica.

O próprio caso de amostrar uniformemente uma permutação de um vetor, necessário para o *Quicksort*, já é um problema interessante. Demonstrar a corretude de um algoritmo deste tipo, por exemplo, exige bons argumentos probabilísticos. Objetos aleatórios não devem ser gerados de forma aleatória.

Note também que há algoritmos que dependem da capacidade de amostrar em conjuntos muito mais complexos do que o de permutações. Tome por exemplo o algoritmo de aproximação de Goemans e Williamson, em [5], para o problema do corte máximo: ele utiliza um ponto amostrado uniformemente numa esfera $(n-1)$-dimensional. Naturalmente, a subrotina que amostra esse ponto se torna crucial para a execução do algoritmo.

Este trabalho irá abordar o problema de amostrar árvores geradoras de um grafo. Parte da motivação se encontra nos algoritmos que dependem da capacidade de fazer isso eficientemente. Outra parte se encontra nas ferramentas que possibilitam abordar tal questão.

As ferramentas são discutidas mais profundamente na metodologia. Sobre as aplicações, há dois exemplos bem notáveis. O primeiro é um algoritmo desenvolvido em [4], que utiliza árvores geradoras aleatórias para construir grafos expansores. Grafos expansores são grafos esparsos com propriedades fortes de conectividade, que podem ser utilizados para construção de códigos corretores de erros, geradores de números pseudo-aleatórios, e ainda outras aplicações, discutidas em [6].

Uma distribuição um pouco mais interessante é necessária para o algoritmo descrito em [2]. Controlando a amostragem através de pesos nas arestas, pode-se implementar um algoritmo que aproxima uma das versões do problema do caixeiro viajante. É a aplicação que motiva a definição mais geral do problema que será tratado, definida a seguir.

2. Objetivos e metodologia

O principal objetivo deste trabalho é desenvolver, de maneira rigorosa, as ferramentas teóricas necessárias para entender e projetar algoritmos eficientes para amostrar árvores geradoras.

Deseja-se desenvolver um algoritmo polinomial $\mathcal{A}$ que, para cada grafo conexo $G = (V, E)$ e função $c \in \mathbb{R}_+^E$, devolva uma árvore geradora T de G de forma que a distribuição das árvores seja proporcional ao produtório dos pesos das suas arestas, ou seja, tal que

$$\mathbb{P}\left(\mathcal{A}(G, c) = T\right) = \alpha \prod_{e \in T} c(e),$$

para algum valor de $\alpha \in \mathbb{R}$ constante.

Dois princípios permeiam este trabalho, influenciando desde a decisão dos assuntos abordados, passando pela forma com as quais são tratados, e atingindo até mesmo escolhas de estilo e nomenclatura. Estes princípios são de *formalismo como técnica* e da *exposição como finalidade*.

Pensar em *formalismo como técnica* não se trata apenas de ressaltar a necessidade de rigor na matemática. Trata-se de entender esse rigor como uma das ferramentas utilizadas na busca do conhecimento, e não apenas uma forma de expressar conceitos abstratos. O uso cuidadoso de uma linguagem precisa é benéfico por cobrar um entendimento profundo, e por ressaltar relações que nem sempre ficam óbvias nas discussões que ocorrem em "alto nível".

O segundo princípio é da *exposição como finalidade*. O que está se produzindo, afinal, é um texto. E a escrita só tem valor quando lida. Assim, bastante tempo será investido em dar significado para os resultados encontrados, contextualizá-los, motivá-los. A pretensão é de que este texto seja consultado por outros, e não apenas uma versão mais velha do autor.

Dito tudo isto, o foco dos estudos será nos resultados teóricos, principalmente de álgebra linear e teoria da medida, que fundamentam as ferramentas necessárias para o projeto dos algoritmos de amostragem. A meta é desenvolver os resultados com rigor, e expô-los com elegância, para que os algoritmos em si surjam como agradáveis consequências de teorias maduras e bem montadas.

Com essa imagem geral do trabalho em mente, eis os objetivos parciais que se pretende atingir:

2.1. **Matrix Tree Theorem.** O *Matrix Tree Theorem* de Kirchhoff é um dos alicerces deste trabalho. Ele torna possível calcular o número de árvores geradoras de um grafo em tempo polinomial, através de um determinante. Contudo, há mais valor na prova do teorema do que na validade do seu enunciado.

Pode-se pensar que se tratam de duas etapas neste objetivo: primeiro, há a necessidade de construir a fundação algébrica sobre a qual irá se trabalhar; depois, de definir os objetos que carregam as informações do grafo, e utilizar as ferramentas da álgebra linear para chegar ao teorema.

Ambas etapas são interessantes. A fundamentação envolve definir, de maneira precisa, conceitos que serão utilizados posteriormente. Trata-se de ideias bem conhecidas: matrizes, submatrizes, determinantes, e outros. Há aqui uma quantidade considerável de trabalho, reflexo do fato de que essas definições raramente são feitas de maneira satisfatória.

A segunda parte gira em torno de raciocinar, em paralelo, sobre as propriedades algébricas e combinatórias de certas estruturas construídas, e utilizar ferramentas de ambas as áreas para se demonstrar o teorema. Este caminho leva à demonstração do *Matrix Tree Theorem* de Tutte, da qual segue o Teorema de Kirchhoff.

2.2. **Resistências efetivas como probabilidades.** A partir de alguns preceitos simples sobre circuitos elétricos, e com uma linha de raciocínio direcionada, o Laplaciano de um grafo surge de maneira natural. Sendo essa a matriz envolvida no Teorema de Kirchhoff, esse paralelo dá mais significado para o ferramental desenvolvido para se demonstrar 2.1.

Em particular, um dos valores de interesse no estudo de circuitos elétricos é a resistência efetiva. Essa incursão em um problema físico se paga pois, após relacionar propriamente grafos e circuitos elétricos, a resistência efetiva entre dois vértices nos permite calcular a probabilidade marginal de uma aresta estar numa árvore geradora amostrada uniformemente.

Um dos objetivos deste trabalho é fazer o paralelo necessário e demonstrar essa curiosa ocorrência de interdisciplinariedade. Para tal, é necessário mais um mergulho em álgebra linear. Para se calcular a resistência efetiva deve-se antes calcular a *pseudo-inversa* do Laplaciano do grafo. Por essa razão, este trabalho irá definir esse conceito e demonstrar alguns resultados sobre eles que serão utilizados.

2.3. **Algoritmo ingênuo.** A estrutura da amostragem pode ser quebrada de forma recursiva. Utilizando então o fato de que as probabilidades marginais de cada aresta podem ser calculadas eficientemente, pode-se decidir aresta por aresta se ela irá ou não pertencer à árvore geradora. Essas observações gerais, junto dos detalhes que elas ocultam, descrevem um primeiro algoritmo polinomial de amostragem de arestas.

O objetivo aqui é deduzir as minúcias envolvidas no algoritmo, e analisar seu tempo de execução. Esse estudo cuidadoso irá contextualizar melhor o próximo objetivo.

2.4. **O Algoritmo de Harvey-Xu.** O algoritmo de Harvey-Xu é baseado nos mesmos preceitos da abordagem ingênua. Ao observar que a quantidade de posições da matriz pseudo-inversa acessada para o cálculo das probabilidades marginais é constante, pode-se diminuir o tempo de execução assintótico.

A implementação em si é feita de maneira recursiva, e se baseia em atualizações *lazy* da matriz pseudo-inversa do Laplaciano. Por tudo isso, consegue ter um tempo de execução limitado pela operação de multiplicação de matrizes, e ainda utilizar esta subrotina como caixa preta.

Este trabalho irá discutir o funcionamento, demonstrar a corretude, e analisar o tempo de execução desse algoritmo.

2.5. **O algoritmo de Aldous-Broder.** Assim como no estudo dos circuitos elétricos, alguns preceitos simples e um raciocínio direcionado são o suficiente para que o Laplaciano também apareça no contexto de passeios aleatórios em grafos. Novamente, será necessário entender as relações entre diferentes áreas. Felizmente, boa parte do maquinário algébrico já está desenvolvido; a maior parte do trabalho será em torno dos formalismos por trás de cadeias de Markov e processos estocásticos em geral.

Para respeitar o princípio do *formalismo como ferramenta*, torna-se necessário utilizar teoria da medida para fundamentar a discussão desses resultados probabilísticos. Uma das partes mais ricas deste trabalho deve surgir justamente do esforço para definir precisamente esses conceitos amplamente utilizados, mas raramente tratados com o devido rigor.

Aos poucos, esses conceitos irão dialogar com a teoria estabelecida no resto do trabalho, e construir as ideias necessárias para descrever um novo algoritmo, que amostra uma árvore geradora a partir de um passeio aleatório. Em posse de toda essa fundamentação teórica, este trabalho irá discutir esse algoritmo, projetado por Aldous, em [1], e Broder, em [3].

2.6. **Limitantes no cover time e no mean hitting time.** O algoritmo de Aldous-Broder só termina sua execução quando o passeio aleatório visita todos os vértices do grafo. O valor esperado da quantidade de iterações do passeio aleatório antes que isso aconteça recebe o nome de *cover time*.

Há também um algoritmo de natureza semelhante ao de Aldous-Broder, descrito em [7]. Seu tempo de execução é limitado pelo *mean hitting time*, um parâmetro do grafo também relacionado com passeios aleatórios e sempre menor ou igual que o *cover time*.

Assim, este trabalho irá estudar limitantes para ambos parâmetros, para conseguir fazer afirmações sobre o tempo de execução dos algoritmos baseados em passeio aleatórios de forma precisa.

Como cronograma, pretende-se finalizar os objetivos descritos nas seções 2.1, 2.2 e 2.3 até 15 de julho de 2017. A parte do trabalho que envolve o algoritmo de Aldous-Broder, descrito na seção 2.5, será então o foco do trabalho, durante as férias, até meados de setembro. Os demais objetivos, 2.4 e 2.6, serão abordados entre setembro e começo de novembro.

Referências

[1] D. J. Aldous. "The random walk construction of uniform spanning trees and uniform labelled trees". Em: *SIAM J. Discrete Math.* 3.4 (1990), páginas 450–465. URL: http://dx.doi.org/10.1137/0403039 (ver página 3).

[2] A. Asadpour, M. X. Goemans, A. Mądry, S. Oveis Gharan e A. Saberi. "An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem". Em: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 2010, páginas 379–389 (ver página 1).

[3] A. Broder. "Generating random spanning trees". Em: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS 1989*. 1989, páginas 442–447 (ver página 3).

[4] A. Frieze, N. Goyal, L. Rademacher e S. Vempala. "Expanders via random spanning trees". Em: *SIAM J. Comput.* 43.2 (2014), páginas 497–513. URL: http://dx.doi.org/10.1137/120890971 (ver página 1).

[5] M. X. Goemans e D. P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". Em: *J. Assoc. Comput. Mach.* 42.6 (1995), páginas 1115–1145. URL: http://dx.doi.org/10.1145/227683.227684 (ver página 1).

[6] S. Hoory, N. Linial e A. Wigderson. "Expander graphs and their applications". Em: *Bull. Amer. Math. Soc. (N.S.)* 43.4 (2006), 439–561 (electronic). URL: http://dx.doi.org/10.1090/S0273-0979-06-01126-8 (ver página 1).

[7] D. B. Wilson. "Generating random spanning trees more quickly than the cover time". Em: páginas 296–303 (ver página 3).