Universidade de São Paulo
Instituto de Matemática e Estatística
Trabalho de Formatura Supervisionado

Gilson Evandro Fortunato Dias

Algoritmos Probabilísticos

Orientador: José Coelho de Pina Júnior

São Paulo, 2010

Resumo

A proposta deste trabalho é produzir um texto sobre algoritmos probabilísticos. O livro utilizado como base foi *Probability and Computing, M. Mitzenmacher e E. Upfal*, e os temas abordados estão relacionados a conceitos de probabilidade e estatística, que dão suporte a teoria aqui apresentada. O papel de eventos aleatórios e em que áreas são utilizados na ciência da computação também merecem ser pesquisados, porém devem ser aprofundados num outro trabalho.

Esta monografia faz parte do meu trabalho de formatura, da disciplina Trabalho de Formatura Supervisionado. Fui orientado pelo professor José Coelho de Pina, e apresento aqui alguns algoritmos aleatórios que resolvem problemas baseados na teoria de Grafos, fizemos a sua análise probabilística, comparamos as suas complexidades com a de algoritmos determinísticos que solucionam os mesmos problemas e estudamos vários conceitos de probabilidade que serviram de ferramentas para o desenvolvimento deste texto.

Conteúdo

1 Introdução

É necessário sabermos o que é um algoritmo e o que aqui consideramos algoritmo aleatório, antes de tentarmos interpretar o texto em si.

Um algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada mecanicamente num período de tempo finito e com uma quantidade de esforço finita. O melhor exemplo que nos aproxima deste conceito seria o de uma receita, embora muitos algoritmos sejam mais complexos. Eles podem fazer iterações ou necessitar de decisões como comparações ou lógicas até que a tarefa seja executada.

Um algoritmo aleatório (ou probabilístico) é um algoritmo que faz escolhas aleatórias durante a sua execução. É como se fosse jogada uma moeda e dependendo do resultado, cara ou coroa, o algoritmo decide entre uma ou outra execução.

Algumas decisões desses algoritmos usam números aleatórios e o tempo de execução dos mesmos depende não só da entrada mas também dos números aleatórios gerados.

Em termos de eficiência, no pior caso acontece o mesmo que nos algoritmos determinísticos (onde não há aleatoriedade) e não existem entradas desfavoráveis ou que levem ao pior caso, apenas números desfavoráveis para entradas particulares.

Antes de apresentar e analisar esses algoritmos vamos mostrar algumas áreas onde eles são bastante utilizados e tentaremos, sempre que possível, apresentar aplicações e/ou exemplos dos algoritmos que aqui estudarmos.

Em muitos dos exemplos que descrevo abaixo não será feita uma abordagem profunda, pois a idéia é apenas mencionar e exemplicar problemas que são resolvidos com algoritmos aleatórios. Até porque foram justamente esse tipo de problemas que me motivaram para fazer este trabalho, portanto pode ser interessante do ponto de vista do leitor, despertando o seu interesse na leitura deste texto mostrando primeiro alguns exemplos e depois a teoria por trás deles.

Vamos para as aplicações. Como primeiro exemplo temos as redes de computadores, mais especificamente o protocolo *Ethernet* que é uma tecnologia de interconexão para redes locais - Local Area Networks (LAN) - baseada no envio de pacotes. Quando um computador deseja enviar alguma informação, precisa obedecer ao seguinte algoritmo:

1. Se o canal está livre, inicia-se a transmissão, senão vai para o passo 4.
2. Se colisão é detectada, a transmissão continua até que o tempo mínimo para o pacote seja alcançado, então segue para o passo 4.

3. Informa-se sucesso da transmissão para as camadas de rede superiores, e sai do modo de transmissão.

4. Espera-se até que o canal esteja livre.

5. Espera-se um **tempo aleatório**, e vai para o passo 1, a menos que o número máximo de tentativa de transmissão tenha sido excedido.

6. Informa falha (número de tentativas de transmissão excedido) para as camadas de rede superiores, sai do modo de transmissão.

Na linha 5 do algoritmo, quando o canal está livre para transmissões, o computador aguarda um tempo aleatório, dado por alguma função que gera números aleatórios, antes de começar a transmissão. E o computador a inicia caso o número de tentativas para a mesma não tenha sido excecido.

Portanto, não é possível determinar quanto tempo um computador vai ter que esperar para começar ou recomeçar a sua transmissão, pois o tempo de espera é aleatório, sendo justamente esse fato que dá a aleatoriedade ao algoritmo.

Este é um exemplo simples de um protocolo para envio de dados, *Ethernet*, que utiliza um algoritmo aleatório para definir como será a lógica para a transmissão de dados.

Em Criptografia, para sabermos se uma chave é segura ou determinar a confiabilidade de um sistema de segurança podemos aplicar alguma heurística baseada em probabilidades para saber a chance da senha ser quebrada; quanto maior o número de testes, melhor o resultado, mas é também maior o custo operacional.

Existem ainda algoritmos aleatórios que testam em tempo polinomial se um número é primo, o chamado Teste de Primalidade Aleatório.

Para recordar, o teorema de Fermat diz-nos que

$$\text{se } p \text{ é primo e } 0 < a < p, \text{ então } a^{p-1} = 1(mod p),$$

e o algoritmo para testar se um número é primo consiste em verificar a condição $2^{n-1} = 1(mod n)$ e dependendo da resposta faz o seguinte:

1. Se é falso, então n não é primo;

2. se é verdadeiro, então n é provavelmente primo.

Podemos fazer uma melhoria no algoritmo evitando os números que enganam esta heurística, os chamados números de Charmichael, que se parecem muito com números primos e algumas vezes são denominados também de

pseudoprimos. Esses números satisfazem $a^{n-1} = 1(modn)$ para **a** e **n** primos entre si.

O algoritmo descrito acima informa a resposta certa quando n realmente não é um número primo, mas falha quanto à veracidade da resposta, já que o mesmo não apresenta os fatores de n, apenas responde que de não é primo. E por outro lado, se ele diz que n é um número primo, então a resposta pode estar certa com uma probabilidade p, geralmente na casa dos 70%.

Se por um lado ele é um algoritmo polinomial, a desvantagem está em ter uma probabilidade de dar a resposta errada. Vamos mostrar mais a frente que os algoritmos aleatórios apresentam essa propriedade, quando são eficazes falham em termos de corretude.

A aleatoriedade no algoritmo, neste caso, está na resposta que tem uma probabilidade associada a si para definir a corretude do algoritmo.

Em Análise de Algoritmos, estudamos vários algoritmos cujas entradas também podem ser vistas como aleatórias. Um vetor de tamanho n, por exemplo, pode ser considerado como uma escolha feita de forma independente e uniformemente aleatória dentre as $n!$ possíveis permutações do vetor. E em casos como esses é interessante analisar o comportamento do algoritmo não apenas do ponto de vista do melhor ou do pior caso, mas sim do caso médio (ou do número esperado de operações).

O algoritmo de ordenação *Quicksort* é um desses casos; para o tornarmos aleatório basta escolhermos o pivô aleatoriamente, com probabilidade $1/n$. E vemos que o comportamento do algoritmo no pior caso é $\Theta(n^2)$, mas no caso médio (relembrando que estamos a falar do algoritmo onde a escolha do pivô é aleatória) consome tempo $\Theta(n \lg(n))$.

Em Algoritmos em Grafos, existem tanto os conceitos de Grafos Aleatórios como o de Passeios Aleatórios que podemos encontrar em alguns problemas clássicos, como verificar se existem um *st*-caminho num grafo por exemplo, ciclos Hamiltonianos, caminho Eulerianos, menor custo num Grafo cujas arestas tenham pesos, e muitos outros.

Como podemos observar pelos exemplos apresentados acima, existe uma quantidade muito grande de problemas que podem ser resolvidos quando aplicamos a aleatoriedade nas suas soluções ou quando trabalhamos com algoritmos que já têm um comportamento aleatório e pretendemos apenas estudá-los em termos de desempenho, complexidade ou alguma outra caraterística.

Foram, então, problemas como os descritos acima que impulsionaram o interesse neste tipo de algoritmos. Vale a pena saber como eles comportam-se? Será que eles são mais eficazes que os algoritmos determinísticos? Compensa trabalhar com um programa que toma decisões aleatórias?

A minha proposta é estudá-los e produzir um texto sobre a ligação entre algoritmos, aleatoriedade e problemas computacionais que conhecemos nas

mais diversas áreas.

Em muitos livros que consultei apercebi-me que é bastante comum utilizarem a palavra randômicos quando referem-se a algoritmos aleatórios. Neste texto utilizarei aleatórios ou aleatorizados para o mesmo significado.

As grandes motivações para estudarmos este tipo de algoritmos estão relacionadas ao fato deles geralmente serem mais simples e/ou mais eficazes que algoritmos determinísticos. Mas estas caraterísticas acabam por ter as suas desvantagens porque estão diretamente relacionadas à corretude e o seu tempo de execução.

Quanto à corretude, estes algoritmos podem apresentar uma certa probabilidade de dar a resposta errada, como veremos em exemplos mais a frente. E o tempo de execução dos mesmos agora já não depende exclusivamente da entrada do problema, mas principalmente das escolhas aleatórias que forem feitas durante a sua execução, o que pode alterar significativamente o desempenho do algoritmo.

Vamos estudar neste texto alguns métodos e ferramentas para analisar algoritmos aleatórios e aprender a lidar com a aleatoriedade dos mesmos, e sempre que possível vamos comparar os seus resultados com os de algoritmos determinísticos.

Portanto, na segunda seção do texto apresentamos conceitos de probabilidade, de variáveis aleatórias e de seus momentos, abordamos também o que representa o número esperado ou esperança de uma variável aleatória, variância e algumas ferramentas para limitar variáveis aleatórias, nomeadamente as desigualdades de Markov e Chebyshev.

Na terceira seção, apresentamos o estudo feito dos capítulos 5 e 7 do livro utilizado como base onde definimos e abordamos o modelo de Bolas e Sacos, a distribuição de Poisson também merece uma atenção espepcial, falamos ainda de Grafos Aleatórios e o conceito de Cadeias de Markov, que têm uma importância fundamental para os resultados apresentados. Para finalizar esta seção é apresentado um problema sobre Passeios Aleatórios em que utilizamos todas as ferramentas vistas para resolvê-lo.

Finalmente é apresentada a subjetiva deste trabalho, onde é feita uma descrição sobre as atividades realizadas, métodos de estudo, reuniões com o meu orientador e conclusões sobre o trabalho. Menciono ainda nesta seção as experiências adquiridas durante a realização do trabalho, as maiores dificuldades, os desafios e agradecimentos à pessoas que contribuiram para que eu pudesse desenvolvê-lo.

2 Conceitos de Probabilidade

Já é sabido que a probabilidade tem uma importância muito grande nos algoritmos, pois alguns deles fazem escolhas aleatórias durante a sua execução e outros têm sua eficiência e complexidade determinadas probabilisticamente. Uma função de probabilidade é a que mapeia eventos de um espaço amostral para um conjunto de números. E um espaço amostral é definido pelos axiomas que se seguem.

2.1 Axiomas

O espaço amostral tem uma importância fundamental quando estamos a falar de probabilidades, ele é composto pelas seguintes caraterísticas:

1. Um espaço amostral Ω é o conjunto de todos os resultados de um experimento aleatório;
2. quaisquer subconjuntos de Ω são chamados de eventos e os subjconjuntos unitários de Ω definem os eventos um por um do experimento;
3. Pr: $F \to \Re$, é a função de probabilidade, onde F é o conjunto de eventos de Ω.

Por sua vez, uma função de probabilidade Pr precisa respeitar as seguintes condições:

1. Para todo evento $E \subseteq \Omega$, $0 \leq \Pr[E] \leq 1$;
2. $\Pr[\Omega] = 1$;
3. para eventos $E_1, E_2, \ldots$ disjuntos dois a dois, então $\Pr[\cup_i E_i] = \sum_i \Pr[E_i]$.

Por uma questão de simplicidade, e sem perda de generalidade, vamos trabalhar apenas com espaços de probabilidade *discreta*, onde o espaço amostral é finito ou enumerável, e, portanto, os eventos correspondem a todos os subconjuntos de Ω.

Vamos considerar como exemplo o lançamento de um dado não viciado de seis faces, com o espaço de probabilidade definido por $\Omega = \{1, 2, 3, 4, 5, 6\}$ conjunto de todos os resultados possíveis e a função de probabilidade Pr que associa cada evento i como:
X = resultado obtido foi i, onde $i = 1, \ldots, 6$ e temos então,

$$\Pr[X] = \Pr(X = i) = \frac{1}{6}.$$

Temos, ainda, como segundo exemplo o caso em que queremos saber qual é a probabilidade de sair um número ímpar ao lançarmos o dado uma única vez. Temos então,

$$\begin{aligned} \Pr(X = 1) + \Pr(X = 3) + \Pr(X = 5) &= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} \\ &= \frac{1}{2}. \end{aligned}$$

2.1.1 Independência e Probabilidades Condicionais

Dois eventos A e B de um espaço amostral Ω podem ser independentes ou não, propriedade esta que muda consideravelmente a probabilidade de um terceiro evento acontecer, estando ele ligado aos dois primeiros eventos A e B. Vamos, então as definições:

Definição 1: Dois eventos A e B são independentes se e somente se

$$\Pr(A \cap B) = \Pr(A) \times \Pr(B).$$

Definição 2: A probabilidade condicional de o evento A ocorrer, dado que o evento B aconteceu é

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(F)}.$$

Outra forma de verificar se os eventos A e B são independentes é se $\Pr(A|B) = \Pr(A)$, pois neste caso teríamos que:

$$\begin{aligned} \Pr(A|B) &= \frac{\Pr(A \cap B)}{\Pr(B)} \\ &= \frac{\Pr(A) \times \Pr(B)}{\Pr(B)} \\ &= \Pr(A). \end{aligned}$$

No exemplo do lançamento do dado de seis faces não viciado, suponha que exista um outro dado semelhante e queremos então saber qual é a probabilidade de sar um número maior do que 3, dado que saiu 1 quando lançamos o primeiro dado. Como o lançamento de dois dados são independentes, ou seja, o lançamento de um não interfere no resultado do segundo dado, então a resposta continua a ser $\frac{1}{2}$.

Agora, para um exemplo com probabilidade condicional, podemos supor que existe um saco com 2 bolas vermelhas e 3 bolas brancas. Sabemos a priori que foi retirada uma bola do saco e que ela é vermelha, a probabilidade de tirarmos uma bola vermelha do saco, agora é de $\frac{1}{4}$, ao contrário dos $\frac{2}{5}$ se não tivessemos retirado nenhuma bola do saco.

2.2 Variáveis Aleatórias e Esperança

Definição 3: Uma variável aleatória é uma função. Mais concretamente, uma variável aleatória X em Ω é uma função tal que $X : \Omega \to \Re$.

Uma variável aleatória *discreta* trabalha somente um número finito ou enumerável de valores.

Vamos denotar a $Pr(X = a)$, onde X é uma variável aleatória e a um valor real, como a probabilidade do evento acontecer, onde o evento é definido pelo conjunto $s \in \Omega | X(s) = a$.

A *esperança* ou o *valor esperado* de uma variável aleatória X, denotada por $E[X]$ é definida por

$$E[X] = \sum_i i \Pr(X = i).$$

Por exemplo, seja X uma variável aleatória que representa os possíveis valores de um dado não viciado de 6 lados, então a $\Pr(X = i) = \frac{1}{6}$ para $1 \leq i \leq 6$. Podemos então calcular a esperança de X do dado e temos que:

$$E[X] = (1 \times \frac{1}{6}) + (2 \times \frac{1}{6}) + \ldots + (6 \times \frac{1}{6}) = \frac{21}{6}.$$

2.2.1 Linearidade da Esperança

Uma propriedade muito importante da esperança que simplifica muito a nossa tarefa na hora de calcular a esperança da soma de variáveis aleatórias ou a soma das esperanças de variáveis aleatórias é definida pelo teorema abaixo.

Teorema 1 [Linearidade da Esperança]: Para qualquer conjunto de variáveis aleatórias discretas $X_1, X_2, \ldots, X_n$, com esperanças finitas, temos que

$$E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i].$$

Um resultado diretamente relacionado à linearidade da esperança é dado pelo seguinte lema.

Lema 1: Para qualquer constante c e uma variável aleatória X temos que,

$$\begin{aligned} E[cX] &= \sum_i ci\Pr(X = i) \\ &= c\sum_i i\Pr(X = i) \\ &= cE[X]. \end{aligned}$$

2.3 Algumas Variáveis Aleatórias

Depois de definirmos o que são variáveis aleatórias, vamos conhecer algumas variáveis aleatórias que usaremos ao longo do texto, e no final desta seção apresentamos um exemplo para algumas delas.

Bernoulli

É uma variável aleatória que pode assumir dois valores 0 (normalmente chamada de fracasso, quando não ocorre determinado evento) ou 1 (sucesso, quando ocorre determinado evento).

Seja X uma variável aleatória tal que, $\Pr(X = 1) = p$ e $\Pr(X = 0) = 1-p$. Então dizemos que X tem distribuição de *Bernoulli* ou simplesmente que X é uma variável aleatória de *Bernoulli*.

Calculando a esperança de uma variável aleatória de *Bernoulli*, temos que

$$\begin{aligned} E[X] &= 0.\Pr(X = 0) + 1.\Pr(X = 1) \\ &= 0 + \Pr(X = 1) \\ &= p. \end{aligned}$$

Binomial

Se realizarmos um experimento de Bernoulli n vezes, obtemos uma variável aleatória *Binomial*, $B(n, p)$, onde n é o número de experimentos e p a probabilidade de sucesso para cada experimento, e tem distribuição para $j = 1, \ldots, n$:

$$\Pr(X = j) = p^j(1 - p)^{n-j}.$$

Sabendo que $X = \sum_{i=1}^{n} X_i$, e lembrando que cada X_i é uma Bernoulli com probabilidade de sucesso igual a p. Então, claramente $E[X_i] = p$ e pela linearidade da esperança temos que

$$\begin{aligned} E[X] &= E[\sum_{i=1}^{n} X_i] \\ &= \sum_{i=1}^{n} E[X_i] \\ &= np. \end{aligned}$$

Geométrica

Aqui, estamos preocupados com o número de tentativas necessárias até obtermos o primeiro sucesso de um experimento aleatório.

Podemos considerar o caso em que lançamos uma moeda e queremos saber qual é o número de lançamentos que precisamos fazer até sair cara. Este número é dado pela variável aleatória geométrica.

Então, a probabilidade

$$\Pr(X = n) = p(1-p)^{n-1}, \text{ para } n = 1, 2, 3, \ldots$$

é a probabilidade de precisarmos realizar n tentivas até obtermos sucesso. No exemplo da moeda, seria o mesmo que lançarmos n vezes a moeda até sair *cara*.

A distribuição geométrica tem a propriedade de não ter memória, isto é, a probabilidade de que o número de provas até o primeiro sucesso seja $s + t$, sabendo-se que as s primeiras tentativas foram fracassos é igual a probabilidade de o número de provas até o primeiro sucesso ser igual às t tentativas restantes, ou seja,

$$\Pr(X = s + t | X > s) = \Pr(X = t).$$

E justamente por esse motivo é que a $E[X] = 1/p$. Mais para frente veremos ainda outras distribuições como a de *Poisson* e a *Estacionária*.

2.4 Variância e Momentos de uma Variável Aleatória

Definição 4: O *k-ésimo momento* de uma variável aleatória X é $E[X^k]$. E a esperança de x, $E[X]$, é chamada de primeiro momento de X.

Para introduzirmos um técnica que vai ajudar-nos a criar limitações para variáveis aleatórias, chamada *desigualdade de Chebyshev*, precisamos definir o que é a variância de uma variável aleatória. E, para isso, vamos usar o primeiro e o segundo momento de X.

Definição 5: A *variância* de uma variável aleatória X é defina como

$$\begin{aligned} Var[X] &= E[(X - E[X])^2] \\ &= E[X^2 - 2XE[X] + E[X]^2] \end{aligned}$$

$$
\begin{aligned}
&= E[X^2] - 2E[X]E[X] + E[X]^2 \\
&= E[X^2] - E[X]^2.
\end{aligned}
$$

Novamente, utilizando o exemplo em que X representa um dado não viciado com 6 faces vamos calcular então a $V[X]$. Primeiramente temos que calcular

$$
\begin{aligned}
E[X^2] &= (1^2 \times \frac{1}{6}) + (2^2 \times \frac{1}{6}) + \ldots + (6^6 \times \frac{1}{6}) \\
&= \frac{91}{6}.
\end{aligned}
$$

e como $E[X] = 21/6$ então $(E[X])^2 = 441/36$. Portanto,

$$
\begin{aligned}
Var[X] &= E[X^2] - E[X]^2 \\
&= 91/6 - 441/36 \\
&= 91/6.
\end{aligned}
$$

O *desvio padrão* de uma variável aleatória X é dado por

$$\sigma^2[X] = Var[X].$$

Calculando o desvio padrão de X temos então que,

$$
\begin{aligned}
\sigma[X] &= \sqrt{Var[X]} \\
&= \sqrt{\frac{91}{6}}.
\end{aligned}
$$

2.5 Desigualdades de Markov e Chebyshev

Nesta subseção apresentaremos duas ferramentas que utilizaremos para limitar variáveis aleatórias. Começamos, então, pela Desigualdade de Markov, definida pelo teorema a seguir.

Teorema 2 [Desigualdade de Markov]: Seja X uma variável aleatória que assume apenas valores positivos. Então, para todo $a > 0$,

$$\Pr(X \geq a) \leq \frac{E[X]}{a}.$$

A Desigualdade de Markov fornece um limite universal, válido, independentemente da distribuição de X.

Teorema 3 [Desigualdade de Chebyshev]: Para qualquer $a > 0$,

$$\Pr(|X - E[X]| \geq a) \leq \frac{Var[X]}{a^2}.$$

E a desigualdade explica como a variância mede dispersão. De uma maneira geral, ambas desigualdades não geram resultados muito precisos, porque não fizemos nenhuma a respeito da distribuição de X, exceto que possui esperança e variância conhecidas.

Mesmo no caso em que temos mais informação, a Desigualdade de Chebyshev continua verdadeira, mas podemos obter um resultado mais preciso. O fato é que essas desigualdades são úteis em inferir valores-limite para certas probabilidades, levando-se em conta a pouca informação que tomam por hipótese.

Exemplo 1: Vamos supor que X representa a demanda diária por um certo item e que em média $E[X] = 28$ unidades são requisitadas, com variância $V[X] = 16$. Quantos itens devem ser disponibilizados diariamente para atender à demanda diária em pelo menos 90% dos casos?

Queremos encontrar um k tal que $\Pr(X \leq k) \geq 0.9$, que equivale a $\Pr(X \geq k) \leq 0.1$.

Usando a Desigualdade de Markov temos $\Pr(X \geq k) \leq \frac{E[X]}{k} = \frac{28}{k} = 0.1$
$k = 280$.

Portanto, se disponibilizarmos 280 itens diariamente, a demanda será atendida em pelo menos 90% dos casos.

Utilizando a Desigualdade de Chebyshev temos

$$\Pr(|X - 28| \geq k) \leq \frac{Var[X]}{k^2} = \frac{16}{k^2} = 0.1 \; k = 4\sqrt{10}.$$

e, portanto,

$$\begin{aligned} \Pr(|X - 28| \geq 4\sqrt{10}) \leq 0.1 \quad &\Rightarrow \quad \Pr(X - 28 \geq 4\sqrt{10}) \leq 0.1 \\ &\Rightarrow \quad \Pr(X \geq 28 + 4\sqrt{10}) \leq 0.1 \\ &\Rightarrow \quad \Pr(X \geq 41) \leq 0.1. \end{aligned} \tag{2}$$

Pela desigualdade de Chebyshev temos que disponibilizar 41 itens diariamente para que a demanda seja atendida em pelo menos 90% dos casos. Um resultado mais limitado, e portanto, melhor.

3 Grafos e Passeios Aleatórios

Nesta seção o foco vai para um dos processos aleatórios mais simples que conhecemos por modelos de Bolas e Sacos, onde existem m bolas que são colocadas em n sacos, sendo que cada bola de um saco é escolhida de forma indepedente e uniformemente aleatória.

Serão abordadas também algumas caraterísticas da distribuição de Poisson e apresentaremos uma aplicação da mesma para modelar uma sequência de *bits* gerada por um algoritmo de dispersão, um *hash*.

E finalmente, vamos estudar um pouco sobre as cadeias de $Markov$, apresentar o conceito de distribuição Estacionária e aplicar estes mecanismos para resolver um problema que envolve Grafos e Passeios Aleatórios.

Para quem tiver interesse, alguns destes conceitos, demonstrações e problemas podem ser vistos também nos capítulos 5 e 7 do livro.

A seção termina com o exemplo do problema do st-caminho, em Grafos Aleatórios. Problema este que foi abordado também durante a apresentação do meu trabalho e está disponível no pôster.

3.1 Modelo Bolas e Sacos

Este modelo, além de ser bastante usado, é muito conhecido, principalmente na área de probabilidades.

Como já descrevemos na introdução, neste modelo existem m bolas que são introduzidas de forma aleatória (ou não) nos n sacos e são retiradas dos mesmos, com ou sem reposição (depende de cada aplicação), de forma uniforme e independentemente aleatória.

Para melhor fixarmos este modelo, apresentamos abaixo um exemplo simples que visa definir este conceito de maneira prática.

Exemplo 2: Suponha que existe um saco que contém 3 bolas brancas e 4 bolas pretas. Se extrairmos simultaneamente 3 bolas do saco , qual é a probabilidade de que pelo menos duas sejam brancas?

Para resolvermos este problema vamos primeiro definir os eventos E_1 e E_2, que são mutuamente exclusivos.

$E_1 =$ saírem 3 bolas brancas.

$E_2 =$ saírem 2 bolas brancas e 1 bola preta.

Para respondermos a pergunta, precisamos calcular

$$P(E_1 \bigcup E_2) = \Pr(E_1) + \Pr(E_2).$$

Para calcularmos a probabilidade do evento E_1, podemos considerar que ocorre uma ordem de aparecimento para as 3 bolas extraídas, o que não altera o problema (bolas são extraídas simultaneamente).

E, neste caso, calculamos diretamente a probabilidade de a primeira bola ser branca, a da segunda ser branca e a última também ser uma bola branca; temos então:

$$\Pr(E_1) = \frac{3}{7} \times \frac{2}{6} \times \frac{1}{5} = \frac{1}{35}.$$

Agora, para calcularmos a probabilidade do evento E_2, alguma destas sequências teria que acontecer, quando as 3 bolas forem extraídas:

1. Branca/Branca/**Preta**
2. Branca/**Preta**/Branca
3. ou **Preta**/Branca/Branca.

Assim, o evento E_2 será justamente a reunião dessas três maneiras mutuamente exclusivas. Calculando a probabilidade, temos que:

$$\Pr(E_2) = 3 \times \frac{3}{7} \times \frac{2}{6} \times \frac{4}{5} = \frac{12}{35}.$$

Portanto, a probabilidade de saírem pelo menos duas bolas brancas das três que foram extraídas é de

$$\Pr(E_2) = \frac{1}{35} + \frac{12}{35} = \frac{13}{35}.$$

Um outro exemplo deste modelo, apenas a título de curiosidade, seria o de querermos calcular a probabilidade de dois alunos fazerem aniversário na mesma data, numa sala com trinta alunos.

A primeira vez que ouvi falar sobre este problema foi numa aula de Probabilidade e Estatística I, em que a professora pretendia mostrar que na verdade tal probabilidade não era tão baixa, como muitos de nós alunos achávamos.

Mas, por ironia, não havia pelo menos dois alunos que fizessem aniversário no mesmo dia, para desespero da professora.

Para este problema, basta assumirmos que a data do aniversário de cada aluno foi escolhida de forma aleatória dentre os 365 dias do ano, de forma uniforme e independente. Assim, os sacos seriam cada um dos 365 dias do ano e as bolas representariam os dias dos aniversários de cada aluno.

Temos, assim, um modelo de Bolas e Sacos em que queremos saber se existem duas bolas num único saco, ou seja, dois alunos que fazem aniversário no mesmo dia.

Podemos calcular a probabilidade por pessoa, onde a primeira pessoa da sala tem uma data de aniversário. A segunda pessoa tem a probabilidade de fazer aniversário numa data diferente da primeira que é $(1 - 1/365)$. A probabilidade da terceira ter uma data diferente sabendo que as duas primeiras fazem aniversário em datas distintas é $(1 - 2/365)$.

E, então, calculamos a probabilidade final da seguinte forma:

$$(1 - \frac{1}{365}) \times (1 - \frac{2}{365}) \times \ldots \times (1 - \frac{29}{365}).$$

E se fizermos essas contas à mão veremos que esse produto é algo em torno de 0.2937; portanto, em uma sala com trinta alunos existe mais de 70% de chance de dois alunos fazerem aniversário no mesmo dia.

Vistos os exemplos acima, a questão agora é saber qual é cara da distribuição de probabilidade que o modelo de Bolas e Sacos tem. Daria para aproximarmos a alguma das distruibuições vistas na seção anterior?

Além de querermos saber que tipo de distribuição aleatória tem este modelo, podemos fazer várias perguntas interessantes, como, por exemplo, quantos sacos estão vazios? Quantas bolas tem o saco mais cheio? Quantos sacos têm pelo menos n bolas?

Estas questões têm aplicações quando vamos definir um algoritmo ou mesmo analisar o seu comportamento, pois podemos ter alguns problemas que necessitem saber as respostas de algumas perguntas feitas acima com uma determinada eficiência e confiabilidade.

3.2 Distribuição de Poisson

Nesta seção vamos apresentar uma nova distribuição de probabilidade, chamada de Distribuição de Poisson. Geralmente utilizamos esta distribuição quando se deseja contar o número de eventos de certo tipo que ocorrem num intervalo de tempo, superfície ou volume.

Como, por exemplo,

1. Calcular o número de falhas de um computador num dia de operação;

2. número de acidentes de carro numa semana;

3. número de usuários que utilizaram um algoritmo ou programa;

4. quantidade de água que passa numa torneira por minuto.

Nesta distribuição temos os seguintes parâmetros: n representa o tamanho da amostra, p que representa a probabilidade de ocorrer o evento e k, onde $k = 1, 2, \ldots, n$.

De um modo geral, dizemos que uma variável aleatória X tem uma Distribuição de Poisson com parâmetro $\lambda > 0$ se

$$\Pr(N = k) = \frac{e^{-\lambda}\lambda^k}{k!}, k = 1, 2, \ldots$$

Mas, como vimos no capítulo anterior qualquer distribuição de probabilidades é caraterizada pela soma das suas probabilidades ser 1. Vamos, então, verificar tal fato:

$$\begin{aligned}
\sum_{k=0}^{\infty} \Pr(X = k) &= \sum_{k=0}^{\infty} \frac{e^{-\lambda}\lambda^k}{k!} \\
&= e^{-\lambda} \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \\
&= e^{-\lambda} e^{\lambda} \\
&= 1.
\end{aligned}$$

onde usamos na segunda igualdade a expansão de Taylor

$$e^x = \sum_{j=0}^{\infty} (x^j/j!).$$

Portanto, esta é uma distribuição de probabilidade.

Vamos agora calcular a esperança da variável aleatória X, que como vimos tem Distribuição de Poisson:

$$
\begin{aligned}
E[X] &= \sum_{k=0}^{\infty} k\Pr(X=k) \\
&= \sum_{k=1}^{\infty} k\frac{e^{-\lambda}\lambda^{k}}{k!} \\
&= \lambda\sum_{k=1}^{\infty}\frac{e^{-\lambda}\lambda^{k-1}}{(k-1)!} \\
&= \lambda\sum_{k=0}^{\infty}\frac{e^{-\lambda}\lambda^{k}}{k!} \\
&= \lambda.
\end{aligned}
$$

Portanto, para um modelo em que existem m bolas e n sacos teremos aproximadamente uma distribuição de *Poisson* com $\lambda = m/n$, que representa exatamente o número de esperado de bolas por saco.

Uma propriedade interessante na Distribuição de Poisson é dada pelo seguinte lema.

Lema 2: A soma de um número finito de variáveis aleatórias de *Poisson* independentes é uma variável aleatória de Poisson.

Definida a Distribuição de Poisson, vamos apresentar agora uma possível aplicação para os conceitos que vimos, tentando modelar um *hash* a partir disso.

3.2.1 Hash

No plano teórico um *hash* é a transformação de uma grande em uma pequena quantidade de informações. Mais concretamente, é uma sequência de *bits* geradas por um algoritmo de dispersão, onde essa sequência busca unicamente um arquivo ou informação.

Por exemplo, senhas, arquivos, chaves criptografadas e outros. Além disso, funções usadas em criptografia garantem que não é possível a partir de um valor de *hash* retornar à informação original.

Uma função de *hash* recebe um valor de um determinado tipo e retorna um código para ele.

Suponha que pretendamos implementar um verificador de senhas, que ajuda os usuários a não escolherem senhas muito comuns ou facilmente decifráveis, guardando um conjunto de senhas inaceitáveis (baseado em algum aspeto como tamanho, símbolos, dificuldade).

A ideia é que quando o usuário tentasse cadastrar uma senha o sistema deveria ser capaz de analisar se essa senha faz parte do conjunto de senhas inaceitáveis, rejeitando se sim e aceitando, caso contrário.

Uma maneira de resolvermos esse problema poderia ser armazenar em ordem alfabética o conjunto de senhas inaceitáveis e sempre que o usuário tentar cadastrar alguma senha, o sistema faz uma busca binária para verificar se aceita ou rejeita a senha.

Por exemplo, para buscarmos n palavras o consumo de tempo seria praticamente $\Theta(\lg(n))$.

Mas podemos também colocar cada palavra inaceitável num conjunto de sacos e quando for necessário buscar uma palavra basta verificar o saco que a contém. Ou seja, se o conjunto de palavras inaceitáveis tiver m palavras e existirem n sacos, podemos então dizer que temos uma representação do modelo Bolas e Sacos(m, n).

Mas, para tal, é necessário assumirmos que a distribuição de m palavras em n sacos é aleatória, preservando, assim, que as palavras são colocadas nos sacos de forma independente e uniforme. Portanto, o conjunto de sacos representa neste momento a nossa tabela de *hash*. Vamos então assumir que a probabilidade de uma palavra x ser colocada num saco i é 1 para $0 \leq i \leq n$.

Analisaremos o custo para buscarmos uma palavra num saco. Para encontrarmos uma palavra, precisamos primeiro saber em qual saco ela está e depois percorrer a lista de palavras desse saco até encontrarmos a palavra desejada.

Se procuramos uma palavra que faz parte do alfabeto do sistema, então o número esperado de palavras no saco onde está a palavra que buscamos é $(m-1)/n$. Portanto, o número de palavras no saco é $1 + (m-1)/n)$.

E, se por acaso tivermos que $m = n$, então o número de palavras em cada saco é constante, pois $m/n = 1$. Se a função de *hash* tem tempo constante, então o tempo total esperado para a busca também o é.

Neste modelo é facilmente verificável que quanto menor o número de sacos, maior será o tempo de busca de uma palavra, porque se tivermos, por exemplo, cinquenta palavras e apenas dois sacos, o número esperado de palavras em cada saco é $50/2 = 25$ e, portanto, teríamos uma lista em cada saco de comprimento 25.

No pior caso, bastaria que a palavra que buscamos fosse uma das últimas da lista para que o tempo de busca seja em torno de $m/2 = 25$, muito maior que o $\Theta(\lg(n))$, a primeira solução apresentanda.

Outra desvantagem é que pode ocorrer um desperdício de espaço, já que nada garante que não vai existir pelo menos um saco vazio, esta situação ocorre quando $m \simeq n$.

Portanto, não há muito o que escolher; ou escolhemos $n \simeq m$ e lidamos com sacos vazio, ou um n relativamente pequeno e temos um tempo de busca maior que o normal.

3.3 Grafos Aleatórios

Não é possível falarmos de grafos ou de problemas que apresentam soluções baseadas na teoria de grafos sem nos questionarmos acerca do comportamento de tais algoritmos.

Geralmente, para problemas que pertencem à classe NP-difícil é sempre interessante saber se determinados problemas são difíceis para grande parte da entrada ou para uma quantidade realtivamente pequena entre todos os possíveis grafos. Os Grafos Aleatórios ajudam-nos a analisar esse tipo de perguntas.

Existem dois modelos a partir dos quais podemos definir o conceito de Grafos Aleatórios. O primeiro é o $G_{n,p}$ em que consideramos todos os grafos não dirigidos com n vértices distintos $v_1, v_2, \ldots, v_n$. Um grafo deste modelo com m arestas tem probabilidade

$$p^m(1-p)^{\binom{n}{2}-m}.$$

Podemos gerar um grafo em $G_{n,p}$ considerando cada uma das $\binom{n}{2}$ possíveis arestas em alguma ordem e depois adicionar independentemente cada arestas ao grafo com probabilidade p. Seja X o número esperado de arestas, então

$$E[X] = \Pr(x = 1) = \binom{n}{2}p$$

e cada vértice tem um número esperado de grau $(n-1)p$.

No modelo $G_{n,N}$, consideramos todos os grafos não dirigidos com n vértices que tenham exatamente N arestas. Existem $\binom{\binom{n}{2}}{N}$ grafos possíveis, cada um escolhido com a mesma probabilidade que os demais.

Uma maneira de gerarmos um grafo com tais caraterísticas seria começarmos com um grafo sem nenhuma aresta. Escolhemos então uma das $\binom{n}{2}$ possíveis arestas de forma aleatória e adicionamos no grafo. Depois, voltamos a escolher uma aresta aleatoriamente dentre as $\binom{(n-1)}{2}$ arestas restantes e voltamos a adicionar ao grafo. Continuamos a fazer o mesmo procedimento até que o grafo tenha extamente N arestas, e paramos.

3.3.1 Ciclos Hamiltonianos em Grafos Aleatórios

Um caminho e um ciclo Hamiltoniano num grafo são respetivamente um caminho e um ciclo que passa por todos os vértices exatamente uma única vez.

Vamos então ver um algoritmo aleatório cuja análise probabilística depende da distribuição da entrada do problema e também das escolhas aleatórias que faz durante a sua execução.

O problema em encontrar um ciclo Hamiltoniano é NP-difícil. Mas a análise do nosso algoritmo aleatório vai mostrar-nos que não é tão difícil, achar ciclos Hamiltonianos para alguns grafos aleatórios selecionados.

O nosso algoritmo vai utilizar uma simples operação que chamaremos de *rotação*. Seja G um grafo não dirigido. Suponha que

$$P = v_1, v_2, v_3, \ldots, v_k$$

é um caminho simples no grafo G e que (v_k, v_i) é uma aresta de G. Então

$$P' = v_1, v_2, \ldots, v_i, v_{k-1}, \ldots, v_{i+2}, v_{i+1}$$

é também um caminho, que nós chamaremos de *rotação* de P com a aresta de rotação (v_k, v_i).

Como exemplo, vamos considerar um caminho

$$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6$$

e seja (v_6, v_4) a aresta de rotação. Então, esta aresta define um novo caminho dado por

$$v_1 \to v_2 \to v_3 \to v_4 \to v_6 \to v_5.$$

Portanto, do vértice 4 passa a ter uma nova aresta para o vértice 6 e o vértice 5 passa a ser o último vértice do caminho.

Se (v_i, v_k) é uma aresta de rotação, então a mudança é feita a partir do vértice v_k onde uma nova aresta vai ligar o subcaminho $v_{k+1}, v_{k+2}, \ldots, v_n$ rotacionado, sendo que v_{k+1} passa a ser o último vértice do novo caminho, v_{k+2} o penúltimo, e assim por diante, até v_n ser o primeiro vértice do subcaminho que vai ligar ao vértice v_k.

Um primeiro algoritmo seria escolher arbitrariamente um vértice para começar o caminho; este é a *cabeça* inicial do caminho. A cabeça é sempre um dos vértices finais do caminho.

Se $v_1, v_2, \ldots, v_n$ é um caminho, então v_n é o vértice final desse caminho. Daqui em diante o algoritmo aumenta o caminho deterministicamente a partir da cabeça ou rotaciona o mesmo desde que existam arestas adjacentes na lista de cabeças.

Descrevendo o algoritmo, teríamos:

AlgoritmoCicloHamiltoniano (V, E):

1. Comece com um vértice aleatório e o defina como a cabeça do caminho.

2. Repita os passos abaixo até que o algoritmo encontre um ciclo Hamiltoniano ou as arestas não usadas da lista de cabeças esteja vazia.
   (a) Defina $P = v_1, \ldots, v_k$ como sendo o caminho, onde v_k é a cabeça e defina (v_k, u) a primeira aresta na lista de cabeças;
   (b) remova (v_k, u) da lista de cabeças e u da lista;
   (c) se $u \neq v_[i$ para $1 \leq i \leq k$, adicione $u = v_{k+1}$ no fim do caminho e defina u como sendo a cabeça;
   (d) caso contrário, se $u = v_i$, rotacione o atual caminho com a aresta (v_k, v_i) e defina v_{i+1} como sendo a cabeça.

3. Devolva ciclo Hamiltoniano se achar, caso contrário devolva não.

No passo 2, alínea **(d)** o algoritmo encontra um caminho Hamiltoniano se $k = n$ e a aresta escolhida é (v_n, v_1).

Mas existe uma dificuldade em analisarmos este algoritmo, porque quando o algoritmo encontra algumas arestas na lista destas, a distruibuição das arestas restantes é condicionada naquelas que o algoritmo já visitou.

É como se tivéssemos um problema com distribuição do modelo Bolas e Sacos, onde não existe reposição, ou seja, se tirarmos uma bola do saco altera a probabilidade de tiramos qualquer outra bola do saco. Não existe reposição, neste caso fica mais difícil fazer a análise probabilística.

Então, embora percamos um pouco em termos de eficiência, vamos estudar um algoritmo modificado que corrige esse problema, para depois podermos analisá-lo probabilisticamente.

Precisamos, então, modificar o processo de rotação de tal forma que a próxima cabeça da lista seja escolhida de forma uniforme e independente. Para isso, vamos trabalhar com duas listas de arestas onde uma representa as arestas que ainda não foram visitadas (não utilizadas) pelo algoritmo e a outra guarda as já visitadas (utilizadas).

Ao escolher a aresta de rotação, o algoritmo pega uma aresta da lista dass visitadas ou uma que ainda não foram usadas com as devidas probabilidades.

Após isso invertemos o caminho com uma pequena probabilidade em cada passo.

O algoritmo modificado rotaciona ou inverte o caminho, mas estas duas operações não aumentam o tamanho do caminho o que nos leva a acreditar que é um desperdício, que deveríamos levar em conta um novo vértice que aumente o caminho e possa fechar o ciclo Hamiltoniano.

A questão neste algoritmo modificado não tem a ver com eficiência, mas sim com a facilidade em analisá-lo em termos de probabilidade.

No pôster, do trabalho falamos sobre o problema do álbum de figurinhas e vamos utilizá-lo aqui, mas não faremos uma demonstração completa sobre o problema.

Resumidamente vamos assumir que existem n diferentes figurinhas (jogadores de futebol neste exemplo), distribuídas independente e uniformemente, e que cada pacote (que pode ser comprado em algum supermercado ou loja) contém apenas uma figurinha.

Queremos saber qual é o número esperado de pacotes que precisamos comprar para completar o álbum de figurinhas do nosso time?

Seja X o número de pacotes comprados até obtermos todas as figurinhas. Se X_i é o número de pacotes comprados enquanto temos $i-1$ figurinhas diferentes, e cada X_i é uma variável aleatória com distribuição geométrica, então $X = \sum_{i=1}^{n} X_i$.

E quando temos $i-1$ figurinhas, a probabilidade de obtermos uma nova figurinha é

$$p_i = 1 - \frac{i-1}{n}$$

e, portanto,

$$E[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$$

O número esperado de pacotes que precisamos comprar para obter todas as figurinhas, é, portanto

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{n} X_i\right] \\ &= \sum_{i=1}^{n} E[X_i] \\ &= \sum_{i=1}^{n} \frac{n}{n-i+1} \\ &= n\sum_{i=1}^{n} \frac{1}{i}. \end{aligned}$$

Onde a soma $\sum_{i=1}^{n} \frac{1}{i}$ é conhecida como o número harmônico $H(n)$, e $H(n) = \ln n + \Theta(1)$.

E, portanto,

$$\begin{aligned} E[X] &= n \ln n + \Theta(n) \\ &= \Theta(n \ln n). \end{aligned}$$

Para o algoritmo modificado, o problema de achar um ciclo Hamiltoniano se parece exatamente com o problema do álbum de figurinhas, onde a probabilidade de encontrar um novo vértice para adicionar no caminho quando sobraram k vértices a serem adicionados é k/n.

Uma vez que todos os vértices estão no caminho, a probabilidade de um ciclo ser fechado a cada rotação é $1/n$.

Se a lista de arestas não visitadas não for muito grande esperamos encontrar um caminho Hamiltoniano em $O(n \lg(n))$ rotações e com mais outras $O(n \lg(n))$ rotações para fechar o caminho que forma o ciclo Hamiltoniano.

O algoritmo falha quando:

1. O algoritmo roda $3n \lg(n)$ passos sem que a lista de arestas não utilizadas fique vazia, mas fracassa ao construir um ciclo Hamiltoniano;

2. pelo menos uma das listas de arestas não utilizadas fique vazia durante os primeiros $3n \lg(n)$ iterações do *loop*.

Logo, a probabilidade de o algoritmo falhar é $\Pr(E_1) + \Pr(E_2)$.

A probabilidade do evento E_1 é o mesmo que calcular a probabilidade de o caminho encontrado não se tornar um ciclo em $n \ln(n)$ iterações, que é dada por

$$(1 - \frac{1}{n})^{n \ln(n)} \leq e^{-\ln(n)} = \frac{1}{n}.$$

Esta probabilidade é muito parecida ao cálculo da probabilidade de não termos conseguido uma específica figurinha (problema do álbum de figurinhas), depois de já termos comprado $2n \ln(n)$ pacotes, que é dada por

$$(1 - \frac{1}{n})^{2n \ln(n)} \leq e^{-2 \ln(n)} = \frac{1}{n^2}.$$

Para calcular a probabilidade do E_2 teriam que acontecer um dos seguintes casos: pelo menos $9n\ln(n)$ arestas foram removidas da lista de arestas não utilizadas de pelo menos um vértice nas primeiras $3n\ln(n)$ iterações do *loop* ou pelo menos um vértice tinha menos de $10n\ln(n)$ arestas inicialmente na sua lista de arestas não utilizadas.

Tais probabilidades representam o cálculo de $\Pr(X \geq 9n\ln(n))$ e $\Pr(Y \leq 10\ln(n))$. Ambas são menores ou iguais a 1, prova que deixaremos em aberto.

Portanto, $\Pr(E_2) \leq \frac{2}{n}$.

No total, a probabilidade de o algoritmo não encontrar um ciclo Hamiltoniano em $3n\ln(n)$ iterações é limitada por

$$\Pr(E_1) + \Pr(E_2) \leq \frac{4}{n}.$$

3.4 Cadeias de Markov

Outra ferramenta interessante que nos ajuda a modelar processos aleatórios são as Cadeias de Markov. Elas apresentam a propriedade Markoviana, chamada assim em homenagem ao matemático Andrei Andreyevich Markov.

A definição desta propriedade, também chamada de memória markoviana, é que os estados anteriores são irrelevantes para a predição dos estados seguintes, desde que o estado atual seja conhecido.

Para definirmos uma Cadeia de Markov precisamos apresentar primeiro o conceito de processo estocástico. Um processo estocástico

$$X = \{X(t) : t \in T\}$$

é uma coleção de variavéis aleatórias, onde t representa o tempo. Se T é um conjunto enumerável, então T é um processo de tempo discreto.

Um processo estocástico de tempo discreto X_0, X_1, X_2,... é uma Cadeia de Markov se

$$\Pr(X_t = a_t | X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, ..., X_0 = a_0) = \Pr(X_t = a_t | X_{t-1} = a_{t-1}).$$

Portanto, o valor de X_t depende do valor de X_{t-1}, mas não da sequência de estados que levaram o sistema a esse valor.

A probabilidade do processo sair do estado i para o estado j é um passo é dada pela probabilidade de transição

$$P_{i,j} = \Pr(X_t = j | X_{t-1}) = i.$$

Podemos, então definir uma matriz de transição onde na linha i e coluna j, e temos a probabilidade de transição $P_{i,j}$. Como estamos a falar de uma função de probabilidade, é necessário que para todo i, $\sum_{j\geq 0} P_{i,j} = 1$.

Para o caso em que o processo dá $m \geq 0$ passos até chegar a algum estado, podendo ser o estado de origem, definimos o $m - passo$ da probabilidade de transição

$$P_{i,j}{}^{m} = \Pr(X_{t+m} = j | X_t = i)$$

como a probabilidade do processo sair do estado i e chegar ao estado j em exatamente m passos.

3.4.1 Classificação de Estados

O primeiro passo para analisarmos o comportamento de uma Cadeia de Markov é classificar os seus estados. Vamos definir, então, o que é um estado acessível.

Definição 6: Um estado i é acessível a partir de um estado j, se para algum inteiro $k \geq 0$, temos que $P_{i,j}{}^{k} \geq 0$. Se dois estados i e j são acessíveis entre si, nós dizemos que eles se comunicam e escrevemos $i \leftrightarrow j$.

A relação de comunicação entre dois estados define sempre uma relação de equivalência, ou seja, é:

1. **reflexiva:** para qualquer estado i, $i \leftrightarrow i$;
2. **simétrica:** se $i \leftrightarrow j$, então $j \leftrightarrow i$;
3. **transitiva:** se $i \leftrightarrow j$ e $j \leftrightarrow k$ então $i \leftrightarrow k$.

Definição 7: Uma Cadeia de Markov é irredutível se todos os estados pertecem a uma classe de comunicação.

Se para todos os pares de estados existir uma probabilidade maior do que 0 de o processo sair do primeiro estado para o segundo, então a Cadeia de Markov é dita irredutível. Podemos dizer ainda que uma Cadeia de Markov é irredutível se e somente se o grafo que a representa é fortemente conexo.

Vamos denotar $r_{i,j}{}^{t}$ como sendo a probabilidade de a primeira transição do estado i para o estado j ocorrer no tempo t, ou seja,

$$r_{i,j}^t = \Pr(X_t = j \text{ e, para } 1 \leq s \leq t-1, X_t \neq j | X_0 = i).$$

Definição 8: Um estado é *Recorrente* se $\sum_{t \geq 1} r_{i,i}^t = 1$. Um estado é dito *Transitório* se $\sum_{t \geq 1} r_{i,i}^t < 1$. Uma Cadeia de Markov é recorrente se todo o estado da cadeia é recorrente.

Vamos denotar também o tempo esperado para retornar ao estado i quando começamos pelo estado i por

$$h_{i,i} = \sum_{t \geq 1} t.r_{i,i}^t.$$

Similarmente, $h_{i,j}$ representa o tempo esperado para chegar ao estado j saindo do estado i.

Definição 9: Um estado recorrente é positivo recorrente se $h_{i,i} < \infty$. Caso contrário, dizemos que o estado é nulo recorrente.

Como consequência da definição acima, em uma Cadeia de Markov:

1. pelo menos um estado é recorrente; e
2. todos os estados recorrentes são positivos recorrentes.

Definição 10: um estado i de uma Cadeia de Markov de tempo discreto é *periódico* se existir um inteiro $k > 1$ tal que $\Pr(X_{t+s} = j | X_t = j) = 0$, a menos que s seja divisível por k. Uma Cadeia de Markov é periódica se qualquer estado da cadeia é periódico. Um estado ou uma cadeia que não é periódico é dito aperiódico.

Definição 11: Um estado recorrente positivo aperiódico é um estado *ergódico*. Uma Cadeia de Markov é ergódica se todos os seus estados são ergódicos.

Corolário 1: Qualquer Cadeia de Markov aperiódica, irredutível e finita é uma cadeia ergódica.

Como exemplo, vamos imaginar que um jogador ganhe um real com probabilidade 1/2 ou perde um real com probabilidade 1/2.

Tal jogador pode perder no máximo p_1 ou pode ganhar no máximo g_1, sendo que o jogo acaba quando o jogador atinge $-p_1$ ou g_1.

Qual seria a probabilidade de o $jogador_1$ ganhar g_2 antes que o $jogador_2$ perca p_1?

Se $g_1 = p_1$, então por simetria essa probabilidade é $1/2$ e claramente $-p_1$ e g_1 são estados recorrentes e todos os outros são transientes.

3.5 Distribuição Estacionária

Como já definimos acima, as probabilidades de transição podem ser armazenadas numa matriz de tal forma que na linha i e coluna j esteja a probabilidade Pi, j. Se P a matriz de probabilidade de transição em 1 passo de uma Cadeia de Markov.

Se $\tilde{p}$ representa a distribuição de probabilidade do estado da cadeia no tempo t, então

$$\tilde{p}(t+1) = \tilde{p}(t)\mathbf{P}.$$

Definição 12: Uma distribuição estacionária de uma Cadeia de Markov é uma distribuição de probabilidade $\tilde{\pi}$ tal que:

$$\tilde{\pi} = \tilde{\pi}\mathbf{P}.$$

Teorema 4: Qualquer Cadeia de Markov ergódica, irredutível e finita tem as seguintes propriedades:

1. a cadeia tem uma única distribuição estacionária $\tilde{\pi} = (\pi_0, \pi_1, \ldots, \pi_n)$;
2. para todo j e i, o $\lim_{t\to\infty} P^t_{j,i}$ existe e é independente de j;
3. $\pi_j = \lim_{t\to\infty} P^t_{j,i} = \dfrac{1}{h_{i,j}}$.

É possível encontrarmos $\tilde{\pi}$ resolvendo o sistema linear

$$\begin{array}{c} \tilde{\pi}\mathbf{P} = \tilde{\pi} \\ \sum_{i=0}^{n} \pi_i = 1 \end{array}$$

3.6 Passeios Aleatórios

Seja $G = (V, E)$ um grafo finito e conexo, onde $n = |V|$ e $m = |E|$.

Um *caminho* num grafo é uma sequência de vértices com a seguinte propriedade: se v e w são vértices consecutivos na sequência, então vw é um arco.

Dados s e t em $V(G)$, consideramos o problema de decidir se existe ou não um st-caminho.

Um *passeio aleatório* em G é uma Cadeia de Markov definida pela sequência de movimentos de uma partícula P entre os vértices de G.

O estado deste processo é definido pelo lugar onde a partícula se encontra num certo momento.

No estado i, a probabilidade da partícula P seguir pela aresta $i - j$ é $\Pr(P = j) = \dfrac{1}{d(i)}$, onde $d(i)$ é número de arestas que saem do vértice i.

Lema 2: Um passeio aleatório em um grafo não dirigido G é aperiódico se e somente se G não for bipartido.

Prova:

($\Rightarrow$) Se o grafo for bipartido, então ele possui período $d = 2$.

($\Leftarrow$) Se o grafo não for bipartido, então ele possui pelo menos um ciclo ímpar, garantindo que todos os vértices possuem um caminho com tamanho ímpar para si.

Teorema 12: Um passeio aleatório em G converge para uma distribuição de probabilidades estacionária dada por

$$\pi_{v_i} = \frac{d(v_i)}{2|E|},$$

para $1 \leq i \leq n$.

Relembrando que $h_{v,u}$ denota o número de passos para sair de v para u, temos então a seguinte propriedade,

$$h_{u,u} = \frac{2|E|}{d(u)}.$$

Lema 5: Para quaisquer dois vértices u e v, vale:

$$(u, v) \in E \Rightarrow h_{v,u} < 2|E|.$$

O *tempo de cobertura* de um grafo $G = (V, E)$ é a esperança do tempo necessário para visitar todos os vértices $v \in V$ a partir de um passeio aleatório que começa no vértice v.

Mostraremos que o tempo de cobertura de um grafo $G = (V, E)$ é limitado superiormente por $4|V| \times |E|$.

Seja T uma árvore geradora de G. Existe um *Ciclo Euleriano* que visita exatamente duas vezes todas as arestas de T (basta fazer uma busca em profundidade e guardar os vértices do ciclo).

Seja $v_0, v_1, \ldots, v_{2|V|-2} = v_0$ a sequência de vértices visitados pelo Ciclo Euleriano, começando pelo vértice v_0. O tempo esperado para atravessar a sequência de vértices do ciclo é um limitante superior para o tempo de cobertura.

Verificaremos, agora, se existe um caminho entre dois vértices de G. Basta usar uma busca em largura ou em profundidade, e tais algoritmos consomem espaço $\Theta(n)$.

O algoritmo aleatório abaixo para verificar se existe um st-caminho trabalha apenas com $O(\log n)$ *bits* da memória retornando sim se existir o caminho em até $4n^3$ passos, e não caso contrário.

Algoritmo st-Caminho:

1. Começa o passeio aleatório em s;
2. se o passeio atingir t em até $4n^3$ passos;
3. então devolva *sim*;
4. senão, devolva *não*.

O algoritmo do st-caminho com passeios aleatórios em G retorna a resposta certa quando não existir o st-caminho, e pode errar a resposta se ele não achar um st-caminho em até $4n^3$ passos, no passeio aleatório que ele faz em G.

Seja Y uma variável aleatória que assume valores positivos. Utilizando a Desigualdade de Markov que diz que para todo $a > 0$,

$$\Pr(Y \geq a) \leq \frac{E[Y]}{a}.$$

temos então que

$$\Pr(X \geq 4n^3) \leq \frac{2n^3}{4n^3} = \frac{1}{2}.$$

Portanto, a probabilidade de o algoritmo retornar a resposta correta é $\frac{1}{2}$.

4 Parte Subjetiva

Nesta última seção da monografia falaremos sobre o que significou fazer este trabalho, quais foram as maiores dificuldades, em como este trabalho contribuiu para aumentar conhecimento do autor, que ideias e projetos o autor pretende realizar após esta etapa, falar sobre os professores e disciplinas que deram a ele as ferramentas necessárias para que chegasse ao sucesso e agradecimentos às pessoas que contribuíram, de forma direta ou indireta, para concluir mais uma fase deste studante-autor.

4.1 Atividades Realizadas

Como descrevemos um pouco na introdução do texto, a orientação deste trabalho foi feita pelo professor José Coelho de Pina Júnior. O interesse por esta área foi crescendo quando o mesmo cursou disciplinas como Algoritmos em Grafos e Análise de Algoritmos onde teve o prazer de ter justamente o citado orientador como professor de tais disciplinas. Antes de um contato formal, o autor pesquisou um pouco a página pessoal do professor com o propósito de saber se havia o interesse do mesmo para orientar alunos em trabalhos de conclusão de curso.

Foi quando uma proposta informal foi encontrada, pois o professor Coelho interessava-se em estudar um livro, que veio a ser o livro-base para este trabalho, e que abordava justamente a relação entre a probabilidade e os algoritmos.

O autor foi, então, ter com o professor, pois tinha interesse em estudar algo ligado a algoritmos, complexidade, otimização e outros. Nessa conversa decidimos que produziria um texto sobre algoritmos probabilísticos, depois de estudarmos o livro-base.

O estudo foi sendo feito com reuniões semanais, onde o autor deveria estudar alguns assuntos do livro e na reunião discutir com o orientador sobre os problemas, fazer demonstrações de teoremas, alguns testes em exemplos apresentados pelo livro e aproveitar também para tirar dúvidas de alguns conceitos que não tinham ficado tão claros para o autor.

Destas reuniões, participou também o colega e aluno Israel Lacerra que foi orientado também pelo professor Coelho e no seu trabalho trata de assuntos como o método probabilístico, modelos de Monte Carlos, Las Vegas e outros. Para quem tiver interesse em conhecer um pouco mais, acessar: http://www.ime.usp.br/∼ cef/mac499-08/monografias/rec/israel/.

Esse estudo foi até meados de setembro/outubro e depois iniciou-se a escrita da presente monografia.

4.2 Experiência Obtida

Ao realizar este trabalho o autor aprendeu muitos conceitos novos e obteve experiências interessantes. A principal delas deve ter sido em relação a desenvolver técnicas para trabalhar em grupo. As reuniões com o orientador foram muito importantes nesse sentido, pois a relação de respeito e distância entre aluno e professor que havia da parte do autor melhorou. Este passou a olhar para o orientador mais como um colega, amigo e não como um professor na sua essência.

Melhorou bastante algumas técnicas de demonstração de teoremas, pois precisou ser mais rigoroso, muitos dos resultados que apresentava não tinham uma fundamentação que deixasse claro que estava correto. Tudo o que foi apresentado, precisou ser demonstrado nas mencionadas reuniões, mesmo que o assunto não fosse posteriormente inserido no trabalho final.

A disciplina que precisei criar em termos de horários e programação foi também bastante proveitosa. O trabalho é feito ao longo do ano e para que não fique tudo em cima da hora, é sempre bom haver uma programação sobre as atividades a serem desenvolvidas. Particularmente, sempre houve dificuldades da parte do autor nesse aspecto, e com o trabalho de conclusão de curso aprendi a ser um pouco mais organizado nesse sentido.

A cobrança do orientador sempre foi grande e o autor acredita ter dado mais trabalho para ele do que o contrário. Por esse conjunto de experiências adquiridas durante esse tempo em que trabalharam juntos, fica aqui o grande agradecimento do autor a seu orientador.

Outra grande dificuldade, formalizar conceitos ou ideias. Como o presente trabalho tem uma base matemática e algébrica muito grande, foi necessário escrever um texto um pouco mais formal do que os que o autor apresentava em relatórios de exercício-programa, por exemplo. Definir, apresentar algum teorema e demonstrá-los eram as rotinas do autor ao longo das reuniões. O estudante-autor ainda tem muito a aprender nesse aspecto, mas sem dúvida o presente trabalho ajudou imensamente.

4.3 Conclusão

Os algoritmos probabilísticos são muito importantes, não apenas pelos exemplos de aplicação fornecidos na introdução deste texto, mas porque eles são bem mais comuns em computação do que imagina. O fato de eles terem um comportamento aleatório não significa que não temos o controle sobre eles.

Há uma ideia de que não sabemos muito bem o que ele vai fazer, como vai ser o seu comportamento e nem que resultados vamos obter utilizando um algoritmo assim. Mas o interesse nestes algoritmos é que não queremos saber como ele se comporta no melhor ou pior caso, mas sempre estamos interessados em saber o que esperar dele no caso médio. É como se os deixássemos rodar com entradas diferentes e fossemos coletando as suas saídas, para depois fazermos uma aproximação quanto ao seu comportamento.

Nesse aspecto, saber analisar probabilísticamente um algoritmo destes é talvez mais importante do que o próprio algoritmo em si. Pois, se não temos a menor ideia de como ele se comporta, utilizá-lo nessas circunstâncias pode ser um desperdício de tempo, espaço, e em alguns casos, de dinheiro também. Então, as ferramentas apresentadas como as Desigualdades de Markov e Chebyshev, as Distribuições de Poisson e a Estacionária, Variáveis Aleatórias, Esperança e Variância assumem uma importância vital para conseguirmos deduzir como deve ser o comportamento de um algoritmo probabilístico.

Outra conclusão é que geralmente os algoritmos probabilísticos são mais fáceis de se implementar comparando com os determinísticos, pois eles apresentam maior flexibilidade em relação à decisões que tomam. Num grafo, por exemplo, tanto pode ir por uma aresta ou por outra que o resultado final não se altera.

Muitos deles são também mais eficazes consumindo menos espaço ou tendo um tempo de resposta menor. A maior dificuldade em relação a estes algoritmos está justamente na sua análise. Por vezes, a implementação é bastante simples, mas a sua análise é complexa justamente por envolver aleatoriedade.

É muito comum modificar estes algoritmos, eliminar alguma propriedade ou acrescentar outras opções que aumentam o número de operações do algoritmo no total, com a finalidade de simplificar a sua análise. Nesses casos, perde-se eficiência ou o consumo de espaço torna-se maior, mas pouco importa desde que seja possível ou mais fácil analisar o comportamento do algoritmo e definirmos a complexidade e corretude do mesmo.

Falando em corretude, outro fator que costuma diminuir o interesse por algoritmos probabilísticos é justamente o fato de alguns deles terem uma

certa incerteza na resposta. É muito comum estes algoritmos darem uma resposta correta quando a mesma for *não* ou não existir, mas no caso da resposta ser *sim* ou existir o que se procura, esses algoritmos costumam dar a resposta correta com uma determinada probabilidade e mesmo que acertem na resposta, dificilmente conseguem provar que a resposta estava certa, ou seja, mostrar um certificado da mesma.

Por exemplo, um algoritmo que verifica se num grafo existe um caminho entre dois vértices o algoritmo simplesmente responde *sim* ou *não*. E quando a resposta é *sim*, raramente estes algoritmos apresentam o caminho em si, que serviria como certificado para o usuário de que realmente existe tal caminho.

Pretende-se num trabalho posterior implementar algoritmos e testá-los exaustivamente, no sentido de verificar se têm um comportamento parecido à complexidades que foram determinadas na análise probabilística dos mesmos; bem como estudar o restante dos assuntos abordados no livro e produzir um texto sobre os mesmos, embora os conceitos mais a frente apresentem uma dificuldade muito maior.

Agradecimentos ao meu orientador, pela oportunidade dada, pela disponibilidade para ajudar no que fosse necessário e, além disso, por ser também um excelente professor. Foi um prazer ter sido seu aluno e orientando.

A todos os professores do IME, de uma maneira geral, não apenas pelos conhecimentos que adquiri, mas também pelo comprometimento e corretude dos mesmos com a universidade, alunos e funcionários. Mais do que professores, vocês são amigos dos alunos. Obrigado.

Aos meus colegas, que estiveram nessa luta comigo ao longo desses anos, votos de que sejam bem-sucedidos profissional e pessoalmente.

Aos meus familiares, pessoas para as quais os agradecimentos serão sempre eternos, obrigado por tudo.

Referências

[1] M. Mitzenmacher e E. Upfal, Probability and Computing, *Cambridge*, 2005.

[2] L. Breiman, Probability and Stochastic Processes, *Mifflin*, 1969.

[3] W. Bussab e P. Morettin, Estatística Básica, 2004.

[4] www.ime.usp.br/∼ pf.

[5] Rejeev Motwani and Prabhakar Raghavan, Randomized Algorithms, ACM Compunting Surveys, Vol 28, 1996.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Algoritmos: Teoria e Prática, Campus, 2002.

[7] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, 1989.

[8] H.R. Lewis, C.H. Papadimitriou, Elementos de Teoria da Computação, 2nd ed., Bookman, 2000.

[9] J.M. Martinez, S.A. Santos, Métodos Computacionais de otimização, SBMAC, Goiânia 1996.