UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM DE SISTEMAS COMPLEXOS

FÁBIO TOSETTO REALE

Métodos de Monte Carlo para amostragem de permutações com restrições e aplicações

São Paulo
2018

FÁBIO TOSETTO REALE

Métodos de Monte Carlo para amostragem de permutações com restrições e aplicações

Versão corrigida

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Modelagem de Sistemas Complexos.

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 6 de julho de 2018. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Área de concentração:
Fundamentos de Sistemas Complexos

Orientador:
Prof. Dr. José Ricardo Gonçalves de Mendonça

São Paulo

2018

CATALOGAÇÃO-NA-PUBLICAÇÃO
(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)

Reale, Fábio Tosetto
Métodos de Monte Carlo para amostragem de permutações com restrições e aplicações / Fábio Tosetto Reale ; orientador José Ricardo Gonçalves de Mendonça. – 2018.
57 f. : il.

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Modelagem de Sistemas Complexos, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.
Versão corrigida.

1. Método de Monte Carlo. 2. Grafos aleatórios. 3. Matemática discreta. 4. Sistemas Markovianos de partículas. 5. Processos de exclusão. 6. Matemática da computação. I. Mendonça, José Ricardo Gonçalves de, orient. II. Título.

CDD 22.ed.– 519.282

Dissertação de autoria de Fábio Tosetto Reale sob o título **"Métodos de Monte Carlo para amostragem de permutações com restrições e aplicações"** apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Modelagem de Sistemas Complexos na área de concentração Fundamentos de Sistemas Complexos, aprovada em 6 de julho de 2018 pela comissão julgadora constituída pelos seguintes membros:

Prof. Dr. José Ricardo Gonçalves de Mendonça
Escola de Artes, Ciências e Humanidades – Universidade de São Paulo
Presidente

Prof. Dr. Cristian Favio Coletti
Centro de Matemática, Computação e Cognição – Universidade Federal do ABC
Examinador

Prof. Dr. Masayuki Oka Hase
Escola de Artes, Ciências e Humanidades – Universidade de São Paulo
Examinador

Profa. Dra. Karla Roberta Pereira Sampaio Lima
Escola de Artes, Ciências e Humanidades – Universidade de São Paulo
Examinadora

Resumo

REALE, Fábio Tosetto. **Métodos de Monte Carlo para amostragem de permutações com restrições e aplicações**. 2018. 57 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2018.

Neste trabalho definimos o processo de exclusão simples simétrico em tempo discreto sobre grafos por meio de permutações com restrições sobre os índices dos vértices dos grafos. O processo é uma generalização das permutações dos índices do grafo completo. Apresentamos algoritmos de Monte Carlo e de amostragem sequencial por importância para amostrar permutações com restrições inspirados pelo problema análogo de calcular permanentes. Como aplicação, utilizamos esses algoritmos para estimar os tempos de relaxação do processo de exclusão simples simétrico em tempo discreto sobre grafos aleatórios densos de Erdős-Rényi com laços.

Palavras-chaves: Processo de exclusão. Permutação com restrições. Matrizes aleatórias 0-1. Permanente. Amostragem sequencial por importância.

Abstract

REALE, Fábio Tosetto. **Monte Carlo sampling of restricted permutations and aplications**. 2018. 57 p. Thesis (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2018.

In this work we define the symmetric simple exclusion process in discrete time over graphs by means of suitably restricted permutations over the labels of the vertices of the graphs. The process is a generalization of the shuffling of labels on the complete graph. Straightforward Monte Carlo and sequential importance sampling algorithms to sample restricted permutations inspired by the related problem of computing permanents are discussed. We illustrate the formalism by estimating the relaxation times of the symmetric simple exclusion process in discrete time over dense loop-augmented Erdős-Rényi random graphs.

Keywords: Exclusion process. Restricted permutation. Random 0-1 matrix. Permanent. Sequential importance sampling.

Lista de figuras

Lista de algoritmos

Lista de tabelas

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

1.1 Objetivos e organização

O principal objetivo deste trabalho consiste em definir o processo de exclusão simples simétrico em tempo discreto sobre grafos (DTSEP(G)) como uma generalização das permutações nos índices dos vértices do grafo completo K_n. O estudo rigoroso desse processo envolve a análise de permutações com restrições, um assunto não trivial envolvendo objetos matemáticos exóticos tanto de teoria de grupos quanto combinatórios praticamente incalculáveis. Por outro lado, a simulação estocástica do processo já oferece desafios bastante interessantes. Em função desse objetivo geral, neste trabalho nos detivemos nos seguintes objetivos específicos:

- Implementar dois algoritmos para a geração de permutações com restrições necessárias à simulação estocástica do DTSEP(G);
- Validar e comparar esses algoritmos através de sua aplicação ao cálculo de Monte Carlo de permanentes com valores conhecidos, em um caso envolvendo matrizes densas e no outro caso matrizes esparsas;
- Aplicar os algoritmos para estimar o permanente de matrizes de adjacência de grafos aleatórios que não possuem permanentes conhecidos, comparando os resultados com aproximações assintóticas existentes na literatura;
- Aplicar os algoritmos implementados para estudar tempos de relaxação do DTSEP(G)

Neste trabalho fazemos contato com assuntos oriundos de áreas variadas da matemática aplicada. Embora não tenhamos nos aprofundado muito em nenhum deles, a diversidade das áreas e o fato de que a maioria desses assuntos não é normalmente abordada em nível de graduação nos motivou a expor brevemente, nas seções que se seguem nesse capítulo de introdução, as principais ideias, termos e definições que recorrem neste trabalho. A exposição realizada aqui é breve e objetiva.

Esta dissertação está organizada da seguinte forma. No Capítulo 2, definimos o processo de exclusão simples em tempo discreto sobre grafos e discutimos como representar sua dinâmica através de permutações com restrições de duas maneiras diferentes. No Capítulo 3 apresentamos dois algoritmos para geração das permutações com restrição, um algoritmo simples de aceitação-rejeição e outro de amostragem sequencial por importância. No capítulo 4 discutimos aplicações dos algoritmos para a estimação de permanentes conhecidos, bem como para matrizes de

adjacência de grafos aleatórios. Também discutimos sua utilização para estudar os tempos de relaxação do DTSEP(G) rumo ao estado estacionário em função da conectividade do grafo subjacente e do número de partículas. No capítulo 5 apresentamos e discutimos os resultados obtidos nas aplicações realizadas no Capítulo 4. Por fim, no Capítulo 6 concluímos o trabalho e discutimos nossas perspectivas oriundas deste trabalho para pesquisas futuras.

1.2 Grafos

Esta seção está baseada principalmente nas referências (1, 2).

Grafos são, de maneira geral, estruturas matemáticas utilizadas para estudar conjuntos de objetos e suas relações binárias. Formalmente, definimos um grafo G como uma dupla (V,E) onde V é um conjunto de vértices e $E \subseteq V \times V$ é um conjunto de arestas, e o denotamos por $G = (V,E)$. O conjunto E define as relações entre vértices da seguinte maneira: se $(i,j) \in E$ (denotado por $i \to j$), dizemos que i é antecessor de j ou que j é sucessor de i. No caso em que um vértice é sucessor de si próprio ($i \to i$) a aresta (i,i) é chamada de laço. Frequentemente desejamos que as relações dadas pelas arestas do grafo sejam simétricas, ou seja, que sempre que $i \to j$ também tenhamos $j \to i$. Neste caso denotamos simplesmente $i \sim j$. Se G é um grafo que não possui laços e onde vale essa propriedade dizemos que G é um grafo simples.

Em um grafo simples, o grau de um vértice $i \in V$, denotado por $d(i)$ ou $\deg(i)$, é dado pelo número de arestas incidentes em i. Podemos pensar em $d(i)$ como o número de vértices vizinhos mais próximos de i em G. Formalmente, o conjunto de vizinhos de um vértice $i \in V$ é dado por $N(i) = \{j \in V : j \sim i\}$, e daí $d(i) = |N(i)|$. Um grafo em que $d(i) = k$ para todo vértice $i \in V$ é chamado de grafo k-regular.

Três tipos de grafos muito importantes são (i) as árvores T_n, que não possuem nenhuma trilha fechada do tipo $v_1 \to v_2 \to \cdots \to v_k \to v_1$ com arestas (v_1,v_2), $\ldots$, $(v_k,v_1) \in E$,(ii) os grafos bipartidos, em que o conjunto de vértices é do tipo $V = A \cup B$ com $A \cap B = \varnothing$ e o conjunto de arestas é da forma $E \subseteq A \times B$, isto é, $(i,j) \in E \Leftrightarrow i \in A$, $j \in B$, e (iii) os grafos completos K_n, para os quais $E = V \times V$, isto é, o grafo em que todas as $\binom{n}{2} = \frac{n(n-1)}{2}$ arestas possíveis estão presentes.

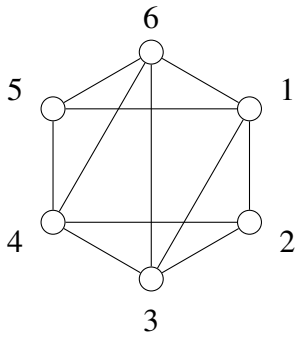
Neste trabalho, além dos grafos simples, estaremos interessados também em grafos que possuem laços em todos os vértices. Podemos construir grafos com laços em todos os vértices a partir de grafos simples simplesmente adicionando os laços ao conjunto das arestas.

Se $G = (V,E)$ é grafo simples e $L = \{(i,i) : i \in V\}$ é o conjunto dos laços associado aos vértices de G, neste trabalho denotamos por $\widetilde{G} = (V, E \cup L)$ o grafo simples G aumentado pelos laços.

Uma maneira conveniente de representar as relações de um grafo $G = (V,E)$ é através de sua matriz de adjacência, denotada por $\text{adj}(G)$. Se $A = (a_{ij})$ é a matriz de adjacência de G, então seus elementos $a_{ij} = 1$ se, e somente se, $i \to j$ e $a_{ij} = 0$ em caso contrário. Observe que a matriz de adjacência A é simétrica, $A = A^T$, se, e somente se, G apresenta simetria nas arestas. Além disso, dada uma matriz qualquer A com entradas no conjunto $\{0,1\}$, podemos associar a ela um grafo G tal que $A = \text{adj}(G)$. Assim sendo, podemos descrever o grafo aumentado $\widetilde{G}$ alternativamente como o grafo cuja matriz de adjacência vale $\text{adj}(\widetilde{G}) = \text{adj}(G) + I$, onde I é a matriz identidade, já que $\text{adj}(L) = I$.

A Figura 1 exemplifica um grafo simples e sua respectiva matriz de adjacências; a Figura 2 exemplifica outros tipos de grafos.

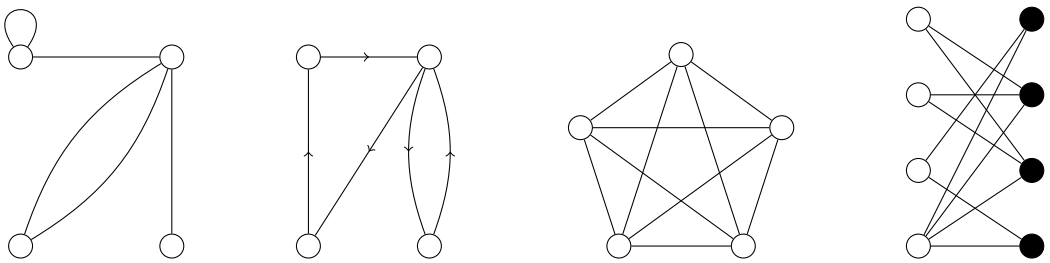
Figura 1 – Grafo simples $G = (V,E)$ com $V = \{1,2,3,4,5,6\}$ e $E = \{(1,2),(1,3),(1,5),(1,6),(2,1),(2,3),\ldots,(6,4),(6,5)\}$ e sua matriz de adjacências $A = \text{adj}(G)$ correspondente. Note que $\text{diag}(A) = (0,\ldots,0)$, já que G não possui laços.



$$\text{adj}(G) = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Fonte: Fábio Tosetto Reale, 2018.

Figura 2 – Exemplos de grafos. Da esquerda para a direita temos um pseudo-grafo (possui um laço e arestas múltiplas), um grafo dirigido (digrafo), o grafo completo K_5 e um grafo bipartido com $V = \text{BRANCOS} \cup \text{PRETOS}$.



Fonte: Fábio Tosetto Reale, 2018.

1.3 Permutações e permanentes

1.3.1 Permutações

Permutações possuem um papel importante neste trabalho, embora não tenhamos explorado sua estrutura geral de grupo ou sua teoria de representação. Ainda assim, convém resumir suas principais propriedades e notação. Nossas principais referências para esta seção são (3, 4).

Na matemática elementar encontramos permutações como arranjos de n objetos distintos que diferem somente em sua ordem. Em outras palavras, uma permutação (ou n-permutação) é uma amostra sem repetições de n objetos que contém todos os n objetos.[1]

Mais formalmente, definimos permutações como sendo os elementos de $\mathscr{S}_V$, o conjunto de todos as funções bijetoras $f : V \to V$ de um conjunto $V \neq \varnothing$. A vantagem de entender permutações como bijeções ao invés de simplesmente como arranjos é que dessa forma elas podem ser compostas e que sua composição também é uma permutação.[2] Quando V é um conjunto finito de n elementos, denotamos o conjunto $\mathscr{S}_V$ por $\mathscr{S}_n$ e o denominamos grupo simétrico de grau n, através da operação de composição de função:

(i) se $f, g \in \mathscr{S}_n$ então sua composição $f \circ g$ (que deve ser entendida como f aplicado sobre o resultado de g) também pertence a $\mathscr{S}_n$;
(ii) existe um único $\varepsilon \in \mathscr{S}_n$ tal que para todo $f \in \mathscr{S}_n$ vale $\varepsilon \circ f = f \circ \varepsilon = f$;
(iii) para todo $f \in \mathscr{S}_n$ existe um único $g \in \mathscr{S}_n$ tal que $f \circ g = g \circ f = \varepsilon$; denotamos tal g por f^{-1}.

Normalmente denota-se os elementos do grupo simétrico pelas letras gregas minúsculas e sua composição $\pi \circ \sigma$ simplesmente pela justaposição $\pi\sigma$. Evidentemente $\mathscr{S}_n$ possui $|\mathscr{S}_n| = n!$ elementos, já que dada uma permutação $\sigma \in \mathscr{S}_n$, $\sigma(1)$ pode assumir qualquer um de n valores, $\sigma(2)$ pode assumir qualquer um de $n-1$ valores distintos de $\sigma(1)$ e assim sucessivamente, até que para $\sigma(n)$ resta um valor possível, de forma que existem $n \cdot (n-1) \cdots 2 \cdot 1 = n!$ possíveis permutações distintas de n elementos.

Existem três maneiras muito comuns de escrever uma permutação: em duas linhas, em uma linha e em ciclos. Dada uma permutação $\sigma \in \mathscr{S}_n$ tal que $i \mapsto \sigma(i)$ como uma lista $\sigma(1), \ldots,$

[1] Os problemas combinatoriais da matemática elementar consistem essencialmente em enumerar listas de elementos (e subconjuntos) com ou sem repetições. A existência, construção e otimização de objetos combinatoriais, assim como seus aspectos algorítmicos, raramente são abordados.

[2] É fácil mostrar que se $f : X \to Y$ e $g : Y \to Z$ são bijetoras, então $g \circ f : X \to Z$ também é bijetora.

$\sigma(n)$, podemos representar σ em duas linhas como $\sigma = \begin{pmatrix} 1 & 2 & \cdots & n \\ \sigma(1) & \sigma(2) & \cdots & \sigma(n) \end{pmatrix}$. Como não há perda de generalidade em apresentar os elementos da permutação na ordem natural $\sigma(1)$, $\sigma(2)$, $\ldots$, $\sigma(n)$, podemos também representar σ em uma linha simplesmente por $\sigma = \sigma(1)\sigma(2)\cdots\sigma(n)$, com o lado direito envolto ou não por parênteses e, caso haja parênteses, com os elementos separados ou não por vírgulas ou espaços. Finalmente, podemos representar qualquer permutação em termos de seus ciclos. Um ciclo de comprimento k em uma permutação σ é uma sequência de índices $i_1, \ldots, i_k$ tais que $\sigma(i_1) = i_2$, $\sigma(i_2) = \sigma(\sigma(i_1)) = i_3$, $\ldots$, $\sigma(i_{k-1}) = \sigma^{(k-2)}(i_1) = i_k$ e, finalmente, $\sigma(i_k) = \sigma^{(k-1)}(i_1) = i_1$, fechando o ciclo. Em notação de duas linhas um ciclo possui a forma $\begin{pmatrix} i_1 & i_2 & \cdots & i_k \\ i_2 & i_3 & \cdots & i_1 \end{pmatrix}$. Denotamos um ciclo de comprimento k por $(i_1 \cdots i_k)$ ou, alternativamente, por $(i_1\sigma(i_1)\cdots\sigma^{(k-1)}(i_1))$. Uma permutação $\sigma \in \mathscr{S}_n$ com a_1 ciclos de comprimento 1, a_2 ciclos de comprimento 2, $\ldots$, a_k ciclos de comprimento k é dita do tipo $(a_1, a_2, \ldots, a_n)$. Obviamente devemos ter $\sum_k k a_k = n$. Os pontos fixos de uma permutação, que são elementos para os quais $\sigma(i) = i$, são 1-ciclos denotados por (i), enquanto transposições, que são pares de índices para os quais $\sigma(i) = j$ e $\sigma(j) = i$, são 2-ciclos denotados por (ij). Por exemplo, a permutação $\sigma = 174326985 = (8)(6)(34)(2795)(1)$ possui 5 ciclos e é do tipo $(3,1,0,1)$, onde, como de costume, omitimos os elementos nulos $a_5 = a_6 = \cdots = a_9 = 0$ ao final da especificação do tipo. Neste exemplo usamos uma convenção útil para escrever a permutação em ciclos: escrevemos os ciclos começando pelo seu menor elemento e ordenamos os ciclos em ordem decrescente em relação a esse elemento. Essa representação estabelece uma bijeção entre duas propriedades importantes das permutações: seu número de ciclos e seu número de descidas, que é o número de pares $\sigma(i)\sigma(i+1)$ com $\sigma(i) > \sigma(i+1)$.

Quando escrevemos uma permutação em ciclos, não há ambiguidade em omitir seus pontos fixos. No nosso exemplo anterior, poderíamos ter escrito $\sigma = 174326985 = (34)(2795)$ sem perda alguma de informação, inclusive de seu número de descidas. No entanto, sob essa mesma convenção, poderíamos interpretar $(34)(2795)$ como o produto das permutações (34) e (2795), agindo de forma a levar primeiro $123456789 \to 173426985$ pela aplicação da permutação (2795) e em seguida levando $173426985 \to 174326985$ pela ação da permutação (34). Pode-se no entanto imediatamente verificar que o resultado é o mesmo que se obteria pela aplicação da permutação $(34)(2795)$. Assim, em notação de ciclos a composição de permutações e sua mera justaposição são a mesma coisa; sequer a ordem em que os ciclos aparecem importa, já que eles são necessariamente disjuntos. Isso significa que toda permutação pode ser escrita como um produto de ciclos disjuntos.

Consideremos agora um ciclo qualquer $(i_1 \cdots i_k)$ de comprimento k. Podemos facilmente verificar que $(i_1 \cdots i_k)$ pode ser escrito como um produto de transposições na forma $(i_1 \cdots i_k) = (i_1 i_2)(i_1 i_3) \cdots (i_1 i_k)$. Por exemplo, a permutação 53124 corresponde ao ciclo (15423), que por sua vez pode ser escrito como $(15)(14)(12)(13)$, como se pode verificar: $(15)(14)(12)(13)\,12345 = (15)(14)(12)\,32145 = (15)(14)\,23145 = (15)\,43125 = 53124$. Reparamos, de passagem, que a permutação 53124 é um desarranjo, isto é, uma permutação sem nenhum ponto fixo. Todo ciclo de comprimento $k > 1$ é um desarranjo. A decomposição de qualquer permutação em ciclos e, portanto, em um produto de transposições, além de demonstrar que $\mathscr{S}_n$ pode ser gerado por transposições[3] permite definir o sinal de uma permutação σ qualquer escrita como um produto de transposições $\tau_1 \cdots \tau_k$ como $\operatorname{sgn}(\sigma) = (-1)^k$. Pode-se mostrar que o sinal de uma permutação é independente da decomposição particular de σ em um produto de transposições e que dadas duas permutações σ e π vale $\operatorname{sgn}(\pi\sigma) = \operatorname{sgn}(\pi)\operatorname{sgn}(\sigma)$.

Existe uma importante conexão entre o grupo simétrico $\mathscr{S}_n$ e o conjunto das partições inteiras do número n. Podemos ver isso a partir da estrutura em ciclos $(a_1, \ldots, a_n)$ das permutações. Como $\sum_k k\,a_k = n$, existe uma relação biunívoca entre o tipo de uma permutação e as partições de n, pois

$$\lambda = (\underbrace{n}_{a_n}, \ldots, \underbrace{2, \ldots, 2}_{a_2}, \underbrace{1, \ldots, 1}_{a_1}) \vdash n, \tag{1}$$

onde $\lambda \vdash n$ denota "λ é uma partição (inteira) de n". Por exemplo, $\lambda = (4,2,1,1,1) \vdash 9$, pois $4+2+1+1+1 = 9$. Essa partição identifica permutações do tipo $(3,1,0,1)$. Assim, podemos tanto dizer que uma permutação é do tipo $(a_1, \ldots, a_n)$ quanto que ela é do tipo $\lambda = (n^{a_n}, \ldots, 1^{a_1})$. A correspondência entre estrutura em ciclos e partições inteiras está na base da teoria clássica da representação do grupo simétrico $\mathscr{S}_n$ em termos de diagramas (*tableaux*) de Young (5).

Finalmente, mencionamos que permutações podem ser representadas através de matrizes com elementos em $\{0,1\}$. Dada uma permutação $\sigma \in \mathscr{S}_n$, podemos associar a ela uma matriz $S_\sigma = (s_{ij})$ de ordem n tal que $s_{ij} = 1$ se $j = \sigma(i)$ e $s_{ij} = 0$ de outra forma. Ou seja, a matriz S_σ possui n elementos $s_{i\sigma(i)} = 1$ e todos os outros elementos nulos. Com esta definição a permutação σ age sobre uma sequência de índices como uma multiplicação de matriz por vetor. Por exemplo, se $\sigma = 2314$ temos

$$2314 \cdot abcd \equiv \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} b \\ c \\ a \\ d \end{pmatrix} \equiv bcad. \tag{2}$$

[3] Na verdade, podemos gerar $\mathscr{S}_n$ a partir do conjunto de transposições adjacentes (12), (23), $\ldots$, $(n-1,n)$.

Da mesma forma, o produto de permutações $\pi\sigma$ se traduz no produto das matrizes $S_\pi S_\sigma$. Alternativamente, podemos definir $S'_\sigma = (s'_{ij})$ com $s'_{ij} = 1$ se $i = \sigma(j)$ e $s'_{ij} = 0$ de outra forma. Naturalmente $S'_\sigma = S_\sigma^T$, já que S'_σ contém o elemento $s_{\sigma(i)i} = s_{ji}$ como seu elemento s'_{ij}. Nesta representação as multiplicações de matriz por vetor e entre matrizes devem ser realizadas da direita pra a esquerda, com $\pi\sigma \equiv S'_\sigma S'_\pi$.

1.3.2 Permanentes

Mesmo entre pós-graduandos e pesquisadores, é comum encontrar pessoas que desconhecem completamente as definições, aplicações e propriedades dos permanentes. Apesar de ser muito menos conhecido que o determinante, o permanente de uma matriz possui inúmeras aplicações em análise combinatória, teoria dos grafos, teoria da computação, probabilidade e estatística e, portanto, também nas áreas de aplicação dessas disciplinas. Por exemplo, a partir dos anos 1950, aproximadamente, permanentes se tornaram ferramentas matemáticas importantes em mecânica estatística (no estudo de modelos bidimensionais sobre grafos planares, por exemplo, do recobrimento de uma superfície por monômeros, dímeros ou poliominos mais complicados) e teoria quântica de campos (por exemplo, na representação de funções de correlação para sistemas de muitos bósons).[4] Nossas principais referências para este tópico são (6, 7); uma referência enciclopédica é (8).

O permanente de uma matriz A de ordem n é definido como

$$\mathrm{per}(A) = \sum_{\sigma\in\mathscr{S}_n}\prod_{i=1}^{n} a_{i\sigma(i)}. \tag{3}$$

Note que essa definição do permanente é muito semelhante à definição do determinante de A,

$$\det(A) = \sum_{\sigma\in\mathscr{S}_n}\mathrm{sgn}(\sigma)\prod_{i=1}^{n} a_{i\sigma(i)}, \tag{4}$$

onde $\mathrm{sgn}(\sigma)$ é o sinal da permutação $\sigma \in \mathscr{S}_n$. A semelhança entre as expressões (3) e (4) faz com que o permanente compartilhe de algumas propriedades do determinante, mas não muitas. Assim, enquanto $\mathrm{per}(A) = \mathrm{per}(A^T)$, em geral $\mathrm{per}(AB) \neq \mathrm{per}(A)\,\mathrm{per}(B)$. Por outro, lado o permanente não se altera pela transposição de linhas ou colunas de A, enquanto o determinante segue mudando de sinal a cada transposição. Transposições de linhas ou colunas preservam o valor do permanente devido à ausência do fator $\mathrm{sgn}(\sigma)$ em sua definição.

[4] Devemos essas observações ao orientador deste trabalho!

Em comum, tanto determinantes quanto permanentes podem ser calculados através do desenvolvimento de Laplace, que permite expressar o determinante (ou o permanente) de uma matriz A em termo dos determinantes (ou permanentes) de submatrizes de A. O desenvolvimento de Laplace para o determinante de uma matriz A de ordem n é dado por

$$\det(A) = \sum_{i=1}^{n} a_{ij}(-1)^{i+j} \det(A_{ij}) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} \det(A_{ij}), \tag{5}$$

onde A_{ij} é a submatriz de ordem $n-1$ que se obtém excluindo a i-ésima linha e a j-ésima coluna de A. O determinante de A_{ij} é chamado de menor do elemento a_{ij} e o produto de $(-1)^{i+j}$ por $\det(A_{ij})$ é chamado de cofator de a_{ij}. Repare que no primeiro somatório da equação (5) fixamos uma coluna j qualquer para desenvolver o determinante, enquanto no segundo somatório fixamos uma linha i qualquer. Analogamente, o desenvolvimento de Laplace para o permanente de uma matriz A assume a forma (mais simples)

$$\operatorname{per}(A) = \sum_{i=1}^{n} a_{ij} \operatorname{per}(A_{ij}) = \sum_{j=1}^{n} a_{ij} \operatorname{per}(A_{ij}). \tag{6}$$

Além de oferecer um procedimento recursivo para o cálculo de determinantes e permanentes, essa formulação permite explorar a estrutura de zeros e as simetrias das matrizes para reduzir a complexidade dos cálculos. No Apêndice A calculamos em detalhes os permanentes de dois casos particulares de interesse.

No caso geral, entretanto, o cálculo de permanentes é um problema de enumeração difícil (9, 10, 11). Enquanto o procedimento de eliminação gaussiana oferece um algoritmo para o cálculo de determinantes em $O(n^3)$, não se conhece nenhum método determinístico eficiente para o cálculo de permanentes. Um cálculo ingênuo para $\operatorname{per}(A)$ baseado nas equações (3) ou (6) requer $n \cdot n!$ operações, enquanto o melhor algoritmo genérico conhecido para computar permanentes, criado por Ryser (6) e aperfeiçoado por Nijenhuis e Wilf (12), requer $O(n \cdot 2^{n-1})$ operações. Assim, para o caso de matrizes arbitrárias, torna-se necessário utilizar técnicas de aproximação para calcular o permanente em tempo polinomial. Encontramos na literatura diversas estratégias para gerar tais estimativas, como randomização de elementos da matriz (13), algoritmos de Monte Carlo em cadeias de Markov (14, 15, 16) ou estratégias de amostragem por importância (17, 18, 19, 20). A ideia em comum por trás de todas essas abordagens aproximadas é escapar da complexidade não-polinomial do problema ao custo da exatidão do cálculo, mantendo, porém, tanto quanto possível, controle sobre o erro da estimativa.

Existem diversos problemas que se reduzem ao cálculo de permanentes de matrizes 0-1, isto é, matrizes com elementos 0 ou 1. Um dos mais interessantes e diretamente relacionados

com o assunto desta dissertação é o do número de emparelhamentos perfeitos (*perfect matchings*) de um grafo bipartido. Seja um grafo bipartido $G = (A \cup B, E)$ com $|A| = |B| = n$; veja a Figura 1. Um emparelhamento (*matching*) em G é um conjunto de arestas sem vértices em comum, isto é, se uma aresta de um emparelhamento incide no vértice i, nenhuma outra aresta do emparelhamento pode incidir no vértice i (2). Por exemplo, se G é um grafo completo bipartido ($E = A \times B$), então as arestas (a_1, b_3) e (a_2, b_1) constituem um emparelhamento em G. Se o emparelhamento é tal que as arestas incidem em todos os vértices, ou seja, para cada vértice i do grafo, existe uma aresta do emparelhamento que incide em i, então o emparelhamento é dito perfeito (*perfect matching*). No nosso exemplo existem 6 possíveis emparelhamentos perfeitos. Quando o grafo bipartido não é completo, a enumeração dos emparelhamentos perfeitos se complica consideravelmente. É possível mostrar, no entanto, que o número de emparelhamentos perfeitos em um grafo bipartido qualquer de ordem par é dado por $\mathrm{per}(A)$, onde A é a matriz de adjacência de G. No problema dos emparelhamentos perfeitos em grafos bipartidos surge uma relação interessante entre o determinante e o permanente: se $\det(A) \neq 0$ então existem $\mathrm{per}(A)$ emparelhamentos perfeitos em G.

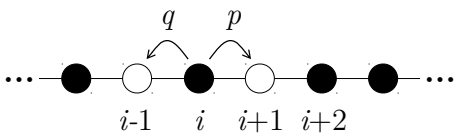
1.4 O processo de exclusão simples

O processo de exclusão simples é um processo de Markov que descreve o movimento de partículas sobre um reticulado que podemos identificar com um subconjunto $\Lambda \subseteq \mathbb{Z}^d$, $d \geq 1$. A palavra "simples" no nome do processo significa que as partículas somente podem se movimentar entre vértices vizinhos entre si em Λ, enquanto a palavra "exclusão" significa que cada vértice do reticulado pode conter no máximo uma partícula de cada vez. Em uma dimensão ($d = 1$), podemos pensar num conjunto de partículas que independentemente tentam se deslocar para o vértice à sua direita com uma probabilidade $p \in [0, 1]$ e para o vértice à sua esquerda com probabilidade $q = 1 - p$. Quando $p = \dfrac{1}{2}$ o processo é dito simétrico, caso contrário ele é dito assimétrico. O caso simétrico é recorrente, enquanto no caso assimétrico as partículas tendem a viajar com uma velocidade média constante em determinada direção e escapar da origem para sempre (com probabilidade 1). O fluxo de tempo no processo pode ser discreto ou contínuo, porém o caso de tempo discreto é muito pouco estudado na literatura; veja a discussão abaixo.[5]

[5] Na verdade, um dos principais pontos deste trabalho é definir adequadamente o processo de exclusão simples em tempo discreto.

Veja a Figura 3. A construção precisa do processo de exclusão simples em tempo contínuo pode ser encontrada em (21, 22); uma versão um pouco mais acessível é dada em (23).

Figura 3 – Ilustração do processo de exclusão simples assimétrico em uma dimensão. A partícula no vértice i pode se mover no próximo instante tanto para a direita quanto para a esquerda; já a partícula no vértice $i+2$ só pode se mover para a esquerda, isso se a partícula em i não decidir saltar à direita no mesmo instante de tempo, o que no processo em tempo contínuo tem probabilidade desprezível $O(dt^2)$.



Fonte: Fábio Tosetto Reale, 2018.

Existem muitos motivos para o estudo de processos de exclusão, bem como, em particular, para o estudo de processos de exclusão sobre grafos. A generalidade da definição do processo deixa claro que ele pode ser usado para modelar muitos processos dinâmicos envolvendo todo tipo de objetos, bastando para isso substituir a palavra "partícula" por ribossomo, molécula, automóvel, pacotes de dados etc. Em física, processos de exclusão são modelos simples que fornecem resultados não triviais sobre questões básicas como a dinâmica de relaxação de fluidos rumo ao equilíbrio termodinâmico (21, 22, 23). Processos de exclusão também são relevantes para a modelagem de sistemas de filas (24, 25), tráfego de veículos e pedestres (26, 27) e sinalização em redes de telecomunicações ou de computadores (28), entre outras. Além disso, processos de exclusão são generalizações naturais do problema do passeio aleatório simples. A matemática de passeios aleatórios sobre grafos e grupos tem sido uma área de pesquisa muito ativa nas últimas décadas, tendo levado a muitos avanços em probabilidade pura e aplicada, estatística, ciência da computação, combinatória, teoria de grupos e análise harmônica, entre outros (29, 30, 31).

Processos de exclusão são normalmente modelados como processos estocásticos em tempo contínuo, onde o tempo de espera para os saltos é dado por uma distribuição de probabilidade exponencial de parâmetro 1. No estudo de tráfego e dinâmica de pedestres utilizando autômatos celulares, diversas técnicas mistas para atualização de processos de exclusão foram propostas, tais como o *shuffle update*, no qual cada partícula tem sua posição atualizada apenas uma vez por unidade de tempo, com uma ordem predeterminada ou aleatória para essas atualizações (32, 33). Esses protocolos de atualização evitam o problema de lidar com as exclusões de atualizações simultâneas, que é exatamente o problema com o qual pretendemos lidar,

mas não são inteiramente processos em tempo discreto, uma vez que as atualizações acontecem em momentos distintos para partículas distintas.

2 Processo de exclusão simples simétrico em tempo discreto sobre grafos

2.1 Definição do DTSEP(G)

Seja $G = (V,E)$ um grafo onde $V = \{1,\ldots,n\}$ e $A = \text{adj}(G)$. Para cada vértice $i \in V$ definimos uma variável η_i, assumindo valores em $\{0,1\}$. Se $\eta_i = 1$ dizemos que o vértice i está ocupado por uma partícula, caso contrário temos $\eta_i = 0$ e dizemos que o vértice i está vazio. Podemos agregar todos esses estados de ocupação em um único vetor coluna

$$\eta = (\eta_1,\ldots,\eta_n)^T \in \{0,1\}^n, \tag{7}$$

que denominamos o estado de ocupação de G ou, mais simplesmente, o estado de G.

O processo de exclusão simples simétrico em tempo discreto sobre grafos, daqui em diante denominado DTSEP(G) (do inglês *discrete time symmetric exclusion process*) é o processo estocástico segundo o qual a cada instante de tempo discreto $t = 0,1,\ldots$, cada partícula em G seleciona um dentre os vértices sucessores daquele ocupado por ela e tenta se deslocar para ele no instante $t+1$. As partículas escolhem os vértices para os quais desejam se deslocar independente e uniformemente dentre os possíveis sucessores dos vértices inicialmente ocupados por elas. Se nenhum vértice tiver sido escolhido como destino por mais de uma partícula (exclusão) o movimento sucede e todas as partículas se movem aos seus novos vértices simultaneamente, caso contrário dizemos que houve uma colisão e o movimento falha. Neste caso o sistema não evolui para $t+1$, permanecendo no mesmo estado que estava no tempo t.

Em modelagem matemática, normalmente queremos que as partículas possam permanecer no mesmo vértice em que se encontram, seja porque há um custo para que elas deixem o vértice, seja porque o vértice corresponde a um "sorvedouro" de algum tipo,[1] seja porque nada na dinâmica do modelado subjacente implica que a partícula deva se mover a cada instante de tempo. Nesses casos basta investigar o DTSEP(G) sobre grafos aumentados com laços em todos os vértices, pois daí a partícula pode eventualmente vir a escolher o próprio vértice como destino no próximo instante (nada impedindo que outras partículas vizinhas também o façam). Daí nosso interesse em descrever grafos com laços na Seção 1.2.

Dadas essas definições e observações, podemos definir o processo estocástico em tempo discreto $\{\eta^t, t \in \mathbb{N}\}$ como uma sequência de vetores de ocupação $\eta^t = (\eta_1^t,\ldots,\eta_n^t)^T \in \{0,1\}^n$

[1] Neste caso o grafo subjacente seria um grafo dirigido $\vec{G}$.

de G que descreve a evolução do DTSEP(G). Nas próximas seções veremos como podemos realizar a evolução de η^t segundo as regras do DTSEP(G).

2.2 Representação direta para a dinâmica do DTSEP(G)

A dinâmica do DTSEP(G) pode ser modelada por meio de permutações $\sigma = \sigma(1)\cdots\sigma(n)$ de $\mathscr{S}_n$, o conjunto das permutações de n objetos. A ideia é evoluir o estado de G por aplicações sucessivas de permutações permitidas pelo grafo G. Essa representação é conveniente pois a natureza bijetiva das permutações implica na conservação do número de partículas e consequente manutenção da propriedade de exclusão. No entanto, a restrição de conectividade exigida pelo grafo G faz com que o conjunto de permutações permitidas contenha apenas permutações que levam o índice i à imagem $\sigma(i)$ se i é vizinho de $\sigma(i)$, isto é, se $i \sim \sigma(i)$. Essa restrição nas possíveis permutações de $\mathscr{S}_n$ pode ser caracterizada em termos da matriz de adjacência A de G como

$$\mathscr{S}_n(A) = \left\{ \sigma \in \mathscr{S}_n : \prod_{i=1}^{n} a_{i\sigma(i)} = 1 \right\}. \tag{8}$$

O número de permutações em $\mathscr{S}_n(A)$ é dado pelo permanente de A

$$|\mathscr{S}_n(A)| = \sum_{\sigma\in\mathscr{S}_n} \prod_{i=1}^{n} a_{i\sigma(i)} = \mathrm{per}(A). \tag{9}$$

Podemos, então, definir DTSEP(G) como um processo estocástico $\{\eta^t, t \geq 0\}$ que, dado uma ocupação inicial η^0 nos vértices de G, evolui em tempo discreto de acordo com

$$\eta^{t+1}_{\sigma(i)} = \eta^t_i, \tag{10}$$

onde σ é selecionado uniformemente de $\mathscr{S}_n(A)$.

A Figura 4 apresenta dois exemplos para evolução formalizada na equação (10)

2.3 Representação dual para a dinâmica do DTSEP(G)

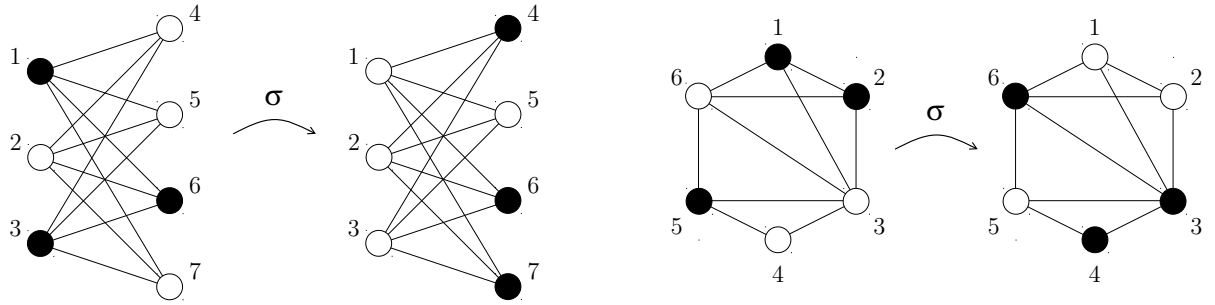
É possível descrever o DTSEP(G) na seguinte representação dual alternativa. Seja

$$\zeta^t = (\zeta^t_1, \ldots, \zeta^t_k) \in \{1, \ldots, n\}^k \tag{11}$$

o vetor das posições das $k \leq n$ partículas no instante t. Nesta representação a dinâmica do DTSEP(G) é dada por

$$\zeta^{t+1}_i = \sigma(\zeta^t_i), \tag{12}$$

Figura 4 – Evolução do DTSEP(G) sobre os grafos completo bipartido aumentado $\widetilde{K}_{3,4}$ (à esquerda) e "genérico" G (à direita), ambos com $k = 3$ partículas (círculos pretos). Os laços nos vértices dos grafos não são mostrados por clareza. No primeiro caso $\eta^t = (1,0,1,0,0,1,0) \to \eta^{t+1} = (0,0,0,1,0,1,1)$ pela ação da permutação $\sigma = 7542163$, enquanto no segundo caso $\eta^t = (1,1,0,0,1,0) \to \eta^{t+1} = (0,0,1,1,0,1)$ pela ação da permutação $\sigma = 362541$.



Fonte: Fábio Tosetto Reale, 2018.

com $\sigma \in \mathscr{S}_n(A)$, como antes. Na verdade, σ pertence ao conjunto menor $\mathscr{S}_n(A^t)$ com A^t uma matriz $k \times n$ dada por

$$A^t = \begin{pmatrix} A_{\zeta_1^t} \\ \vdots \\ A_{\zeta_k^t} \end{pmatrix}, \tag{13}$$

onde A_j denota a j-ésima linha da matriz de adjacência de G. Por exemplo, para a configuração de partículas sobre o grafo $\widetilde{K}_{3,5}$ da Figura 4, $\zeta^t = (\zeta_1^t, \zeta_2^t, \zeta_3^t) = (1,3,6)$ e

$$A^t = \begin{pmatrix} A_1 \\ A_3 \\ A_6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{14}$$

ao passo que para $\zeta^{t+1} = (7,4,6)$ temos

$$A^{t+1} = \begin{pmatrix} A_7 \\ A_4 \\ A_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{15}$$

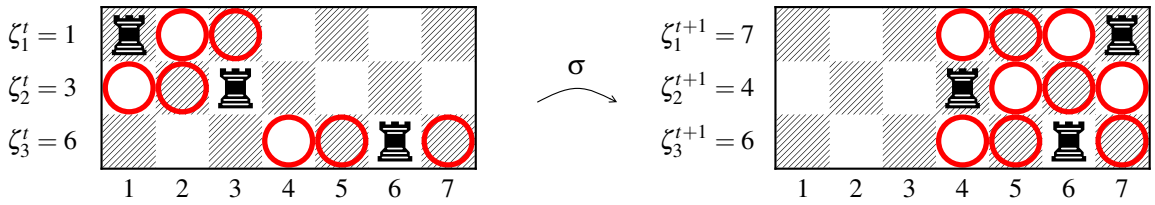
A matriz retangular A^t pode ser vista como um tabuleiro de $k \times n$ posições permitidas para as partículas no instante t e também no instante seguinte $t+1$, uma vez que, por definição,

$$a_{ij}^t = a_{\zeta_i^t, \zeta_i^{t+1}} = 1, \tag{16}$$

já que $\zeta_i^{t+1} = \sigma(\zeta_i^t)$ com $\sigma \in \mathscr{S}_n(A)$ ou $\mathscr{S}_n(A^t)$, o conjunto de movimentos permitidos. Vemos que $|\mathscr{S}_n(A^t)| = \mathrm{per}(A^t)$ nada mais é que o k-ésimo *rook number* de A^t, definido como o número

de maneiras de posicionar k torres (peças de xadrez) indistinguíveis nas casas de um tabuleiro $k \times n$ de tal forma que elas não se ataquem mutuamente (*non-attacking* ou *non-taking rooks*) com as casas (ij) removidas se $a_{ij}^t = 0$.[2] Note que o permanente de uma matriz retangular é bem definido matematicamente (6, 7). A Figura 5 exemplifica a representação do DTSEP(G) em termos de torres e tabuleiros.

Figura 5 – Colocação de $k = 3$ torres que não se atacam mutuamente nos tabuleiros correspondentes às configurações das partículas sobre o grafo $\widetilde{K}_{3,5}$ da Figura 4. A configuração inicial $\zeta^t = (\zeta_1^t, \zeta_2^t, \zeta_3^t) = (1,3,6)$ (à esquerda) evolui pela ação da permutação $\sigma = 7542163$ para a configuração $\zeta^{t+1} = (\sigma(\zeta_1^t), \sigma(\zeta_2^t), \sigma(\zeta_3^t)) = (7,4,6)$ (à direita). As casas marcadas correspondem aos destinos proibidos no próximo instante de tempo para as torres nas respectivas linhas.



Fonte: Fábio Tosetto Reale, 2018.

2.4 Simulação estocástica do DTSEP(G)

A dificuldade em simular a dinâmica sugerida pela equação (10) consiste em amostrar permutações σ uniformemente de $\mathscr{S}_n(A)$. Um possível método de realizar essa tarefa é amostrar uniformemente permutações σ de $\mathscr{S}_n$ e excluir do conjunto de amostras as permutações que violam as restrições impostas pela estrutura de G. Em outras palavras, a ideia é amostrar uniformemente de $\mathscr{S}_n$ e aceitar apenas as amostras que também pertencem a $\mathscr{S}_n(A)$, rejeitando as outras. Para verificar se uma permutação σ pertence a $\mathscr{S}_n(A)$ precisamos apenas verificar se

$$\prod_{i=1}^{n} a_{i\sigma(i)} = 1, \tag{17}$$

que é a condição que define $\mathscr{S}_n(A)$ na equação (8). Como o armazenamento de σ necessita somente de $O(n)$ em espaço de memória e a condição (17) pode ser verificada em tempo $O(n)$, dados A e $\sigma \in \mathscr{S}_n$ podemos verificar se $\sigma \in \mathscr{S}_n(A)$ em $O(n)$ tanto em tempo quanto em espaço de memória. Além disso, como a geração de permutações aleatórias também pode ser realizada em tempo e espaço $O(n)$, um algoritmo de aceitação-rejeição simples para geração de permutações

[2] Devemos essa observação ao Prof. Dr. Cândido F. Xavier de Mendonça (EACH/USP).

aleatórias com as restrições dadas pela matriz de adjacência A (ou seja, pelo grafo G) pode ser implementado em $O(n)$ em espaço. A razão de aceitação do algoritmo, no entanto, depende fortemente da estrutura de G: quanto maior a conectividade de G, maior a razão de aceitação

$$f(A) = \frac{|\mathscr{S}_n(A)|}{|\mathscr{S}_n|} = \frac{\text{per}(A)}{n!}. \tag{18}$$

A equação (18) imediatamente sugere um método de Monte Carlo para estimar o permanente de A uma vez que tenhamos um algoritmo para amostrar permutações com restrições em $\mathscr{S}_n(A)$. A taxa de aceitação $f(A)$, no entanto, é exponencialmente pequena a não ser para grafos bastante densos, ou seja, grafos onde é bastante alta a probabilidade de um par de vértices qualquer pertencer ao conjunto de arestas.

No Capítulo 3 discutimos estratégias para amostrar permutações de $\mathscr{S}_n(A)$ e suas consequências.

3 Geração de permutações aleatórias com restrições

3.1 Método da aceitação-rejeição

O algoritmo normalmente utilizado para geração de permutações aleatórias sem restrições é o de Fisher-Yates-Durstenfeld (34). Esse algoritmo retorna uma permutação aleatória distribuída uniformemente sobre $\mathscr{S}_n$ em tempo $O(n)$. No entanto, esse algoritmo possui uma séria limitação relacionada com os números pseudo-aleatórios necessários para selecionar aleatoriamente as transposições que formam a base do algoritmo. Resumidamente, queremos poder gerar qualquer uma das $n!$ permutações distintas, o que significa que precisamos que o gerador dos números pseudo-aleatórios utilizados tenha pelo menos essa quantidade de estados internos. A maioria dos geradores pseudo-aleatórios possui 2^p estados internos. Dessa forma, é necessário garantir que o gerador utilizado possua $2^p > n!$. Tomando esse cuidado, utilizaremos o Algoritmo 1 para gerar as permutações necessárias à simulação do DTSEP(G), onde *rnd* é uma realização do gerador pseudo-aleatório de um valor uniformemente distribuído no intervalo $(0,1)$ e $\lceil x \rceil$ denota o topo de x (menor inteiro maior ou igual a x). Nossa implementação do algoritmo de Fisher-Yates-Durstenfeld propriamente dito, especificada nas linhas 2–6 do Algoritmo 1, não necessita nem da inicialização da permutação como sendo a identidade, $\sigma = 1\,2\cdots n$, nem da transposição $\sigma(i) \leftrightarrow \sigma(j)$ de valores.

O Algoritmo 1 implementa um método simples de aceitação-rejeição: geramos uniformemente permutações em $\mathscr{S}_n$ (linhas 2–6) e aceitamos somente aquelas que pertencem a $\mathscr{S}_n(A)$ (linha 7). Dessa forma, cada permutação efetivamente produzida pelo Algoritmo 1 pode ser usada para evoluir η^t segundo a equação (10) ou (12).

Algoritmo 1 Geração de permutações com restrições pelo algoritmo de Fisher-Yates-Durstenfeld

Entrada: Matriz 0-1 $A = (a_{ij})$ de ordem $n \geq 1$

1: **repita**
2: $\quad\sigma(1) \leftarrow 1$
3: $\quad$**para** $i = 2$ **até** n **faça**
4: $\quad\quad j \leftarrow \lceil rnd \cdot i \rceil$
5: $\quad\quad\sigma(i) \leftarrow \sigma(j)$
6: $\quad\quad\sigma(j) \leftarrow i$
7: **até que** $\prod_{i=1}^{n} a_{i\sigma(i)} = 1$

Saída: $\sigma(1)\cdots\sigma(n)$ é uma permutação aleatória de $1\cdots n$ uniformemente distribuída em $\mathscr{S}_n(A)$

Observação 1. Como assumimos que para o DTSEP(G) o grafo G possui laços em seus vértices, precisamos que o Algoritmo 1 seja capaz de produzir permutações com pontos fixos $\sigma(i) = i$. Caso G não possua laços, ou seja, seja de fato um grafo simples, é possível utilizar um algoritmo que gera apenas desarranjos, que são permutações sem pontos fixos. Uma alternativa para gerar somente desarranjos, evitando um passo de aceitação-rejeição, seria começar com um desarranjo qualquer, por exemplo, a permutação $23\cdots n1$, e realizar um certo número de transposições aleatórias $\sigma(i) \leftrightarrow \sigma(j)$ certificando que transposições envolvendo pares onde $\sigma(i) = j$ ou $\sigma(j) = i$ sejam evitadas (rejeitadas). Não se sabe, nesse caso, quantas transposições aleatórias seriam necessárias para que os desarranjos gerados estejam uniformemente distribuídos pelo espaço dos desarranjos.

3.2 Amostragem sequencial por importância

A ideia por trás de algoritmos de amostragem sequencial por importância (SIS, do inglês *sequential importance sampling*) é a de amostrar sequencialmente os elementos que compõe a distribuição conjunta de interesse, nesse caso, para $\sigma(1)\cdots\sigma(n)$. Esse processo é realizado passo a passo, condicionando as probabilidades dos passos futuros ao que já foi gerado anteriormente segundo a identidade

$$\mathbb{P}(\sigma(1)\cdots\sigma(n)) = \mathbb{P}(\sigma(1))\mathbb{P}(\sigma(2)\cdots\sigma(n)\,|\,\sigma(1))\,. \tag{19}$$

Em seguida, notamos que $\mathbb{P}(\sigma(2)\cdots\sigma(n)\,|\,\sigma(1))$ é também uma distribuição conjunta e, portanto, podemos continuar separando sequencialmente as distribuições até obtermos

$$\mathbb{P}(\sigma(1)\cdots\sigma(n)) = \mathbb{P}(\sigma(1))\,\mathbb{P}(\sigma(2)\,|\,\sigma(1))\cdots\mathbb{P}(\sigma(n)\,|\,\sigma(1)\cdots\sigma(n-1)). \tag{20}$$

Em simulações de Monte Carlo, o lado direito da equação (20) se torna $\mathbb{P}_1(\sigma(1))\,\mathbb{P}_2(\sigma(2)\,|\,\sigma(1))\cdots$ $\mathbb{P}_n(\sigma(n)\,|\,\sigma(1)\cdots\sigma(n-1))$, com as probabilidades (ou densidades de probabilidade) $\mathbb{P}_i(\,\cdot\,)$ estimadas ou inferidas incrementalmente baseadas em funções-peso para os objetos parciais $\sigma(1)\cdots\sigma(i)$. Note que a distribuição conjunta calculada sequencialmente na equação (20) pode levar em conta as restrições, ou seja, podemos calcular $\mathbb{P}(\sigma(1)\cdots\sigma(n)\,|\,\sigma\in\mathscr{S}_n(A))$. Os fundamentos teóricos do SIS foram dados em (35) e são revisados em (36, 37, 38).

O Algoritmo 2 descreve um SIS para a geração de permutações aleatórias com restrições inspirado pelo problema análogo de calcular o permanente da matriz de adjacência A (17, 18, 19, 20). Note que o primeiro passo consiste em reordenar as linhas e colunas de A em ordem

Algoritmo 2 Geração de permutações com restrições por amostragem sequencial por importância

Entrada: Matriz $A = (a_{ij})$ de ordem $n \geq 1$, com entradas binárias 0-1

1: Reordene as linhas de A em ordem crescente com respeito à $\sum_j a_{ij}$
2: $J \leftarrow \{1, \cdots, n\}$
3: **para** $i = 1$ **até** n **faça**
4: Calcule $R_i = \sum_{j \in J} a_{ij}$
5: **se** $R_i \neq 0$ **então**
6: Escolha $j \in J$ com probabilidade a_{ij}/R_i
7: $\sigma(i) \leftarrow j$
8: $J \leftarrow J \setminus \{j\}$
9: **senão**
10: Pare

Saída: $\sigma(1) \cdots \sigma(n)$ é permutação aleatória de $1 \cdots n$ em $\mathscr{S}_n(A)$

crescente das somas de cada linha. Em outras palavras, queremos iniciar o procedimento SIS com os menores índices da matriz A tendo sido atribuídos aos vértices com menor número de vizinhos. Isso é feito para que seja minimizada a probabilidade de colisão entre índices escolhidos posteriormente no procedimento. Essa estratégia já foi implementada anteriormente na literatura para amostragem de tabelas de contingência com restrições (36, 37, 38). Uma vez que A é uma matriz 0-1, o passo 4 do Algoritmo 2 equivale a $R_i = |J_i \cap J|$ onde $J_i = \{j : a_{ij} = 1\}$, o número de elementos na imagem da permutação disponíveis para serem escolhidos para o índice i (caso haja algum), e a probabilidade no passo 6 se torna a distribuição uniforme sobre os índices disponíveis nesse passo. Mais detalhadamente, R_i conta o número de escolhas possíveis para $\sigma(i)$ dadas as escolhas para os $\sigma(j)$ onde $j < i$, de forma que $\dfrac{1}{R_i}$ é um estimador natural para $\mathbb{P}_i$ e, pela identidade (20), $\prod R_i$ é um estimador para $|\mathscr{S}_n(A)|$. No caso em que as permutações não tem restrições temos $R_i = n - i + 1$ uma vez que só temos exclusão de possibilidades no passo 8 e são excluídos os $i - 1$ valores selecionados anteriormente. Observe que nesse caso temos $\prod R_i = n(n-1)\cdots 1 = n! = |\mathscr{S}_n|$. A partir desas observações, vemos que o Algoritmo 2 oferece, para cada permutação gerada, a possibilidade de produzir um estimador para $f(A)$ na forma

$$\hat{f}(A) = \frac{1}{n!} R_1 \cdots R_n = \prod_{i=1}^{n} \frac{R_i}{(n-i+1)}. \tag{21}$$

Esse estimador é não viesado. Para demonstrar esse fato é suficiente mostrar que $\mathbb{E}(R_1 \cdots R_n) = \text{per}(A)$.

Lema. *O produto dos fatores R_i, $1 \leq i \leq n$ produzidos pelo Algoritmo 2 fornecem um estimador não-viesado para* $\text{per}(A)$.

Demonstração. Dos passos 4, 6 e 8 do Algoritmo 2 vemos que $\mathbb{E}(R_1 \cdots R_n) = \mathbb{E}(R_1)\,\mathbb{E}(R_2 \,|\, R_1) \cdots \mathbb{E}(R_n \,|\, R_1 \cdots R_{n-1})$, de tal forma que seu produto vale

$$\begin{aligned}\mathbb{E}(R_1 \cdots R_n) = \sum_j a_{1j} \left(\sum_{i_1} \frac{a_{1i_1}}{\sum_j a_{1j}} \right) \cdot \sum_{j \neq i_1} a_{2j} \left(\sum_{i_2 \neq i_1} \frac{a_{2i_2}}{\sum_{j \neq i_1} a_{2j}} \right) \cdots \\ \cdots \sum_{j \neq i_1, i_2, \cdots, i_{n-1}} a_{nj} \left(\sum_{i_n \neq i_1, i_2, \cdots, i_{n-1}} \frac{a_{ni_n}}{\sum_{j \neq i_1, i_2, \cdots, i_{n-1}} a_{nj}} \right),\end{aligned} \tag{22}$$

e após os cancelamentos triviais obtemos

$$\mathbb{E}(R_1 \cdots R_n) = \sum_{i_1} a_{1i_1} \cdot \sum_{i_2 \neq i_1} a_{2i_2} \cdots \sum_{i_n \neq i_1, i_2, \cdots, i_{n-1}} a_{ni_n} = \operatorname{per}(A). \tag{23}$$

□

Assim, além de produzir permutações com restrições em $\mathscr{S}_n(A)$, o Algoritmo 2 efetivamente implementa uma maneira de se estimar o permanente da matriz A, desde que A possua todos os elementos $a_{ij} \geq 0$.

3.3 Detalhes de implementação

Implementamos os algoritmos na linguagem Julia, versão 0.6.2.[1] Utilizamos diversas funções da biblioteca padrão oferecida pela linguagem, em especial, o gerador de números pseudo-aleatórios nativo, que implementa o bem conhecido Mersenne Twister, também utilizado na linguagem R. Esse gerador possui um número de estados internos igual a $2^{623 \times 32} = 2^{19936} \simeq 10{,}880 \times 2080!$, ou seja, muito além do necessário para as aplicações utilizadas neste trabalho, veja Seção 3.1.

A biblioteca padrão do Julia oferece uma implementação do algoritmo de Fysher-Yates-Durstenfeld para geração de permutações aleatórias que corresponde aos passos 2–6 do Algoritmo 1, mas decidimos não nos aproveitar desse código pronto. Como comparamos nossos algoritmos entre si, julgamos apropriado realizar implementação própria. O código-fonte é apresentado no Apêndice C. Nosso código-fonte implementa uma versão bastante direta do Algoritmo 1; a única diferença digna de nota é no passo 4, já que o gerador de números pseudo-aleatórios nativo gera números no intervalo $[0, 1)$, ou seja, é capaz de produzir o valor 0, o que poderia levar a $j \leftarrow 0$, fora de seu intervalo de validade, que é $1 \leq j \leq i$. O código apresentado

[1] A linguagem Julia é voltada ao cálculo numérico e possui semelhanças com a linguagem Matlab, porém seus scripts geram executáveis que são compilados e não interpretados, o que permite que ela ofereça desempenho semelhante ao de linguagens como o C e Fortran. O leitor interessado pode visitar o site ⟨https://julialang.org/⟩.

retorna o número total de permutações geradas, além da permutação atendendo às restrições, uma vez que essa importância é necessária à estimação dos permanentes. Uma característica interessante do código, proporcionada pela linguagem, é que o teste de aceitação possui dois comportamentos distintos, um para tratar o caso em que a matriz de adjacência A é dada por valores inteiros 0-1, outro para tratar o caso em que a matriz de adjacência A é dada por valores booleanos **true**-**false**. No primeiro caso o código executa exatamente o passo 7 do Algoritmo 1, retornando o produto das entradas $a_{i\sigma(i)}$ a partir da permutação gerada. No segundo caso ele realiza uma operação lógica **and** em todos as entradas da matriz pertencentes à permutação. A vantagem, nesse caso, é que só são testados os elementos da permutação até que o primeiro valor **false** seja encontrado.

O código para o Algoritmo 2 é apresentado no Apêndice D. A implementação para esse caso também é bem direta. A observação mais importante a fazer é que o código apresentado não implementa o passo 1 do Algoritmo 2. Em implementações práticas, esse sempre será o caso, pois uma vez que produziremos múltiplas permutações por matriz, é mais eficiente realizar a reordenação anteriormente à chamada dessa função. Outra observação é sobre a utilização de um vetor auxiliar v para realizar o cálculo de R_i e o sorteio aleatório para j. O vetor v recebe, na i-ésima iteração do laço principal do algoritmo, a lista de possíveis valores a serem selecionados para $\sigma(i)$, o que é feito ao buscar elementos unitários no produto, elemento a elemento, do vetor indicador J pela i-ésima linha de A. Caso não exista nenhum elemento 1, v é um vetor vazio de 0 elementos.

4 Aplicações

4.1 Estimação de permanentes conhecidos

O Algoritmo 1 da Seção 3.1 fornece estimativas de Monte Carlo do volume de $\mathscr{S}_n(A)$ como

$$|\mathscr{S}_n(A)| = f(A) \cdot n!, \tag{24}$$

onde $f(A)$ é a fração das permutações geradas pelo algoritmo que são aceitas, enquanto o Algoritmo 2 da Seção 3.2 estimam $\mathrm{per}(A)$ como $\mathbb{E}(R_1 \cdots R_n)$, em que R_i são as somas das linhas de A geradas pelo algoritmo.

Aplicamos os algoritmos 1 (aceitação-rejeição) e 2 (SIS) para estimar os permanentes de algumas matrizes com permanentes conhecidos: matrizes de desarranjo D_n, cujos elementos são dados por $D_{ij} = 1 - \delta_{ij}$ (onde δ_{ij} é o delta de Kronecker), que são matrizes densas, correspondendo às matrizes de adjacência de grafos simples completos K_n, além das matrizes 0-1 tridiagonais T_n cujos elementos são dados por $T_{ij} = \delta_{i,j+1} + \delta_{ij} + \delta_{i,j-1}$, que são matrizes esparsas correspondendo às matrizes de adjacência dos grafos caminho P_n onde foram adicionados os laços cada vértice. Em forma matricial temos:

$$D_n = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 1 & 0 & 1 \\ 1 & \cdots & 1 & 1 & 1 & 0 \end{pmatrix}, \quad T_n = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{25}$$

É um resultado clássico que $\mathrm{per}(D_n) = n!(1 - 1/1! + 1/2! - \cdots (-1)^n/n!)$, os números de *rencontres* (OEIS[1] A000166). Por outro lado, $\mathrm{per}(T_n) = F_{n+1}$, o $(n+1)$-ésimo número de Fibonacci (OEIS A000045), sendo que esse fato pode ser facilmente obtido ao realizar a expansão por Laplace de $\mathrm{per}(T_n)$ da primeira coluna da matriz. Cálculos para $\mathrm{per}(D_n)$ e $\mathrm{per}(T_n)$ são apresentados no apêndice A. Claramente, $f(D_n) = \mathrm{per}(D_n)/n!$ é a probabilidade de que uma permutação aleatória não possua ponto fixo, já que $D_{ii} = 0$. Não é difícil ver que $f(D_n)$ difere de $e^{-1} = 0{,}367\,879\,441\cdots$ por menos do que $1/(n+1)!$, que rapidamente se torna bem pequeno conforme n aumenta.

[1] The On-Line Encyclopedia of Integer Sequences® (OEIS) é uma enciclopédia de sequências numéricas e pode ser acessada através do endereço https://oeis.org

Chen e Liu (38), baseando-se em um resultado de Bender (39), obtiveram o seguinte resultado assintótico para o permanente de "matrizes de restrição" A binárias (0-1) de ordem n,

$$\operatorname{per}(A) \approx n! \cdot \exp\left[-\frac{1}{n}\sum_{j=1}^{n}(n-C_j)\right]. \tag{26}$$

Nesta fórmula, os C_j são as somas dos elementos da j-ésima coluna de A. Note que $\sum_j (n-C_j)$ é o número de elementos nulos de A. A equação (26) fornece boas aproximações para valores grandes de n e quando $C_j \sim O(n)$, isto é, para matrizes grandes e com poucas entradas nulas. Como ilustração, observe que (26) fornece valores exatos para $\operatorname{per}(J_n) = n!$, onde J_n é a matriz $n \times n$ cujas entradas são todas iguais a 1 (nesse caso todos os $C_j = n$). Além disso, a aproximação fornecida por (26) para as matrizes de desarranjos D_n (nesse caso todos os $C_j = n-1$) é bastante próxima do valor exato, $\operatorname{per}(D_n) \approx n!/e$. Já para as matrizes tridiagonais T_n, a aproximação de $\operatorname{per}(T_n)$ dadas por (26) é bem ruim, pois nesse caso

$$\operatorname{per}(T_n) \approx n! \cdot \exp\left[-\frac{(n-1)(n-2)}{n}\right]; \tag{27}$$

para $n = 32$, por exemplo, a aproximação acima resulta em $\operatorname{per}(T_{32}) \approx 6{,}288 \cdot 10^{22}$, valor bem distante do valor exato $\operatorname{per}(T_{32}) = F_{33} = 3\,524\,578$.

Os resultados das simulações são apresentados nas Tabelas 1 e 2 e discutidos na Seção 5.1

4.2 Grafos de Erdős-Rényi com laços

Nesta seção investigamos o permanente das matrizes de adjacência de grafos de Erdős-Rényi aumentados com laços, que denotaremos por $\widetilde{G}_{n,p}$, $0 < p \leq 1$. Grafos de Erdős-Rényi possuem muitas propriedades matemáticas simples e interessantes, além de serem bastante utilizados em simulações em geral, de forma que existe interesse em estudar o DTSEP(G) sobre esse tipo de grafo, com ou sem a adição de laços. Como aplicação inicial, no entanto, decidimos estimar os permanentes das matrizes de adjacência de grafos $\widetilde{G}_{n,p}$ densos, com $p \geq 0{,}8$.[2]

Grafos do tipo $\widetilde{G}_{n,p}$ podem ser obtidos a partir do grafo completo K_n, onde cada uma das $\binom{n}{2} = \dfrac{n(n-1)}{2}$ arestas é removida com probabilidade $1-p$ e onde são adicionados laços em todos os n vértices. Sabe-se que para qualquer $\varepsilon > 0$, $G_{n,p}$ (e portanto $\widetilde{G}_{n,p}$) é provavelmente conexo se $p \geq \dfrac{(1+\varepsilon)}{n}\ln n$ (2, 40). O grau médio de $G_{n,p}$ é $\overline{d} = 2\dfrac{|\overline{E}|}{|V|} = (n-1)p$, sendo assim,

[2] Na literatura matemática, normalmente a denominação de grafo de Erdős-Rényi remete a grafos $G_{n,p}$ com $n \nearrow \infty$ e $p \to 0$ tais que $np = \text{const}$. Nos outros casos fala-se em grafo aleatório do modelo binomial (1, 2).

para garantirmos, digamos, que $\overline{d} > \dfrac{n}{2}$ (pois nos interessa que o grafo seja denso), precisamos garantir que $p > \dfrac{1}{2}$. Uma vez que, para todo $n \geq 1$ temos $p = \dfrac{1}{2} \geq \dfrac{(1+\varepsilon)}{n} \ln n$, com $\varepsilon = \dfrac{e}{2} - 1 = 0{,}359\,141\cdots$, sempre tomaremos $p > \dfrac{1}{2}$, ou seja, sempre obteremos $G_{n,p}$ de K_n removendo, em média, menos da metade das arestas. Também sempre tomaremos $n \leq 64$ para evitar tempos de simulação proibitivamente longos.

Para grafos aleatórios de Erdős-Rényi com laços $\widetilde{G}_{n,p}$, a média da soma das colunas é dada por $\overline{C}_j = 1 + (n-1)p$, tal que $\sum_j (n - \overline{C}_j) = (1-p)(n-1)n$ e a estimativa dada por (26) resulta em

$$\operatorname{per}(\widetilde{G}_{n,p}) \approx n! \cdot \exp\left[-(1-p)(n-1)\right], \tag{28}$$

onde, por abuso de notação, nos referimos ao permanente de $A(\widetilde{G}_{n,p})$ como permanente de $\widetilde{G}_{n,p}$, apenas. Esperamos que as estimativas dadas pela equação (28) sejam melhores conforme $p \to 1$, ou seja, conforme $\widetilde{G}_{n,p} \to \widetilde{K}_n$.

Os resultados das simulações são apresentados na tabela 3 e discutidos na Seção 5.2

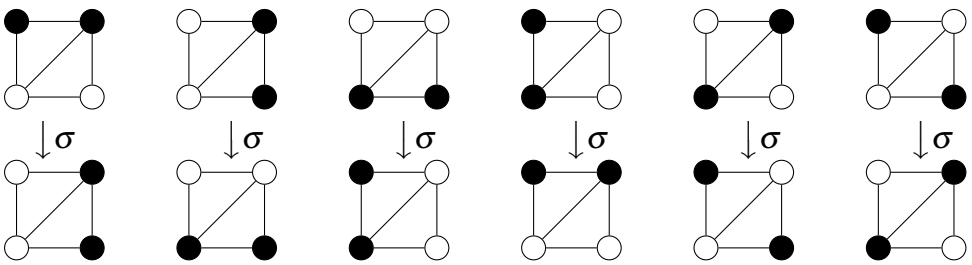
4.3 Tempos de relaxação do DTSEP(G)

O estado estacionário do DTSEP(G) com k partículas sobre um grafo simplesmente conexo G de n vértices é dado por

$$\eta^{\infty} = \binom{n}{k}^{-1} \sum_{1 \leq i_1 < \cdots < i_k \leq n} (1_{i_1}, \ldots, 1_{i_k})^T, \tag{29}$$

onde $(1_{i_1}, \cdots, 1_{i_k})$ denota a configuração com as k partículas ocupando os vértices $i_1, \cdots, i_k$ de G. Qualquer permutação de i_1, i_2, $\cdots$, i_k mantém η^{∞} invariante. A figura 6 exemplifica esse fato.

Figura 6 – Possíveis ocupações para um grafo de 4 vértices (a numeração dos vértices é dada em sentido horário) com 2 partículas, antes e após a aplicação da permutação $\sigma = 2341$. Podemos observar que a permutação preserva o conjunto de estados.



Fonte: Fábio Tosetto Reale, 2018

Da equação (29) podemos ver que a densidade de ocupação de cada vértice de G no estado estacionário é dada por $\eta_i^\infty = k/n$. Por outro lado, a distribuição empírica para as ocupações dos vértices até o tempo $t \geq 1$ é dada por

$$\xi^t = \frac{1}{t}\sum_{s=1}^{t} \eta^s = \frac{t-1}{t}\xi^{t-1} + \frac{1}{t}\eta^t, \tag{30}$$

onde descartamos arbitrariamente a configuração inicial η^0 da média. A equação (30) nos mostra que ξ^t de fato converge a uma distribuição limite, uma vez que $\|\eta^t\| \leq n$ em qualquer norma ℓ_p. Uma possível medida de distância entre uma realização de ξ^t do DTSEP(G) e a distribuição estacionária η^∞ é dada pela distância

$$\chi^2(\xi^t, \eta^\infty) = \sum_{i=1}^{n} \frac{(\xi_i^t - \eta_i^\infty)^2}{\eta_i^\infty} = \frac{n}{k}\sum_{i=1}^{n}(\xi_i^t - \frac{k}{n})^2. \tag{31}$$

Medimos $\chi^2(\xi^t, \eta^\infty)$ para os grafos $\widetilde{K}_n$ (grafo completo aumentado) e $\widetilde{G}_{n,p}$ (grafos Erdős-Rényi aumentados). Em ambos os casos casos, fixamos $n = 64$, variando k entre os valores 16 e 32. Calculamos a média de $\chi^2(\xi^t, \eta^\infty)$ sobre 100 realizações independentes de ξ^t, cada uma a partir de uma ocupação inicial aleatória. Repetimos esse cálculo para 100 grafos aleatórios independentes, calculando a média das médias. Para os grafos $\widetilde{G}_{n,p}$ usamos $p = 0{,}8$ e $0{,}9$. As permutações com restrições necessárias para evoluir o DTSEP(G) foram geradas a partir do Algoritmo 2, muito embora para grafos densos como os $\widetilde{K}_n$ e os $\widetilde{G}_{n,p}$ com $p = 0,8$ e $p = 0,9$ a diferença de desempenho entre os Algoritmos 1 e 2 não seja significativa.

Adicionalmente, realizamos algumas simulações para os grafos esparsos dados pelas matrizes de adjacência T_n (grafos caminho aumentados). Nesse caso, o Algoritmo 1 tem taxa de aceitação extremamente pequena já para valores pequenos de n (como discutido na Seção 5.1). Até mesmo o Algoritmo 2 tem alguma dificuldade com um grafo tão esparso. Para T_3 a taxa de sucesso é $0,75$. Para T_{16} essa taxa cai para menos do que $0,049$. Para T_{64} essa grandeza é menor do que $1,861 \cdot 10^{-6}$, ou seja, esperamos ter de chamar o Algoritmo 2 algo em torno de 500 000 vezes para que este produza 1 permutação. O cálculo das taxas de sucesso do Algoritmo 2 no caso específico onde as entradas são matrizes do tipo T_n são apresentados no apêndice B. Os resultados das simulações são apresentados e discutidos na Seção 5.3.

5 Resultados e Discussão

5.1 Comparação dos resultados para estimação de permanentes

As tabelas 1 e 2 contém os dados de simulação produzidos para as matrizes de desarranjos (D_n) e tridiagonais (T_n), respectivamente. Os valores exatos são apresentados para comparação uma vez que isso é possível nesses casos.

Tabela 1 – Estimativas para $\mathrm{per}(D_n)$ pelos Algoritmos 1 e 2 e os respectivos tempos de execução. Para o Algoritmo 1 as estimativas foram obtidas a partir de 10 000 permutações em $\mathscr{S}_n(D_n)$, enquanto para o Algoritmo 2 elas foram obtidas pela média de 10 000 amostras.

n	$\mathrm{per}(D_n)$	$\hat{\mathrm{per}}_1(D_n)$	$\hat{\mathrm{per}}_2(D_n)$	t_1	t_2
8	14 833	14 836	14 760	0,014 s	0,194 s
10	$1{,}3350 \cdot 10^6$	$1{,}3368 \cdot 10^6$	$1{,}3328 \cdot 10^6$	0,015 s	0,245 s
12	$1{,}7621 \cdot 10^8$	$1{,}7688 \cdot 10^8$	$1{,}7635 \cdot 10^8$	0,022 s	0,297 s
14	$3{,}2071 \cdot 10^{10}$	$3{,}2345 \cdot 10^{10}$	$3{,}2019 \cdot 10^{10}$	0,020 s	0,343 s
16	$7{,}6971 \cdot 10^{12}$	$7{,}6174 \cdot 10^{12}$	$7{,}6949 \cdot 10^{12}$	0,032 s	0,410 s

Fonte: Fábio Tosetto Reale, 2018.

Tabela 2 – Estimativas para $\mathrm{per}(T_n)$ pelos Algoritmos 1 e 2 e os respectivos tempos de execução. Para o Algoritmo 1 as estimativas foram obtidas a partir de 10 000 permutações em $\mathscr{S}_n(T_n)$, enquanto para o Algoritmo 2 elas foram obtidas pela média de 10 000 amostras.

n	$\mathrm{per}(T_n)$	$\hat{\mathrm{per}}_1(T_n)$	$\hat{\mathrm{per}}_2(T_n)$	t_1	t_2
8	34	33,73	32,88	3,82 s	0,199 s
10	89	88,93	88,83	148 s	0,247 s
12	233	234,31	233,47	12 432 s	0,290 s
14	610	—	593,92	—	0,340 s
16	1597	—	1608,91	—	0,384 s

Fonte: Fábio Tosetto Reale, 2018.

Podemos observar que ambos os algoritmos produzem estimativas próximas do valor exato tanto para as matrizes de desarranjos quanto para as tridiagonais. Os tempos de execução, por outro lado, nos mostram grande diferença de desempenho entre os dois algoritmos. Notamos que o algoritmo sequencial (Algoritmo 2) apresenta virtualmente o mesmo tempo de execução para estimar o permanente de matrizes de mesmo tamanho, independentemente de serem as matrizes densas D_n ou as matrizes esparsas T_n. Já o algoritmo de aceitação-rejeição (Algoritmo 1) apresenta tempos de execução muito altos para T_n e variações muito pequenas para os tempos de execução para D_n. Isso ocorre pois o desempenho desse algoritmo depende muito da taxa

de aceitação e temos que $f(T_n)$ cai muito rapidamente com o aumento de n enquanto $f(D_n)$ é sempre bem próximo de $1/e$.

5.2 Comparação entre os algoritmos para os grafos

A tabela 3 apresenta os resultados obtidos para grafos aleatórios de Erdős-Rényi com laços ($\widetilde{G}_{n,p}$). Nesse caso não existem valores exatos para realização de comparação, no entanto, calculamos as aproximações dadas pela equação (28) nos diferentes parâmetros.

Tabela 3 – Aproximação para $f(\widetilde{G}_{n,p})$ dada pela equação (28), bem como as estimativas dadas pelos Algoritmos 1 e 2 e os respectivos tempos de execução. Os valores foram calculados a partir de 1 000 grafos aleatórios para cada par n, p. No caso do Algoritmo 1 foram geradas permutações até 1 000 aceitações enquanto para o Algoritmo 2 foi calculada a média de 1 000 estimadores

n	p	$f(\widetilde{G}_{n,p}) \approx$	$\hat{f}_1(\widetilde{G}_{n,p})$	$\hat{f}_2(\widetilde{G}_{n,p})$	t_1	t_2
16	0,9	0,2231	0,2192	0,2198	3,07 s	20,7 s
32	0,9	0,0450	0,0410	0,04128	23,0 s	46,4 s
48	0,9	0,0091	0.0075	0,0076	220 s	77,5 s
64	0,9	0,0018	—	0,0014	—	111 s
16	0,8	$4,98 \cdot 10^{-2}$	$4,01 \cdot 10^{-2}$	$3,96 \cdot 10^{-2}$	21,4 s	23,3 s
32	0,8	$2,03 \cdot 10^{-3}$	$1,20 \cdot 10^{-3}$	$1,18 \cdot 10^{-3}$	991 s	47,3 s
48	0,8	$8,3 \cdot 10^{-5}$	—	$3,3 \cdot 10^{-5}$	—	78,2 s
64	0,8	$3,4 \cdot 10^{-6}$	—	$0,9 \cdot 10^{-6}$	—	112 s

Fonte: Fábio Tosetto Reale, 2018.

Os valores apresentados nas Tabelas 1 e 2 evidenciam que os algoritmos fazem boas aproximações. Os argumentos dados no final da Seção 4.1 mostram que aproximações feitas a partir da equação (26), da qual a equação (28) é derivada, são melhores conforme $p \to 1$. Sendo assim, faz sentido interpretamos os valores para os estimadores apresentados na tabela 3 como uma medida do quão boa são as aproximações dadas pela equação (28) e, como argumentado anteriormente, essas aproximações funcionam melhor conforme $p \to 1$.

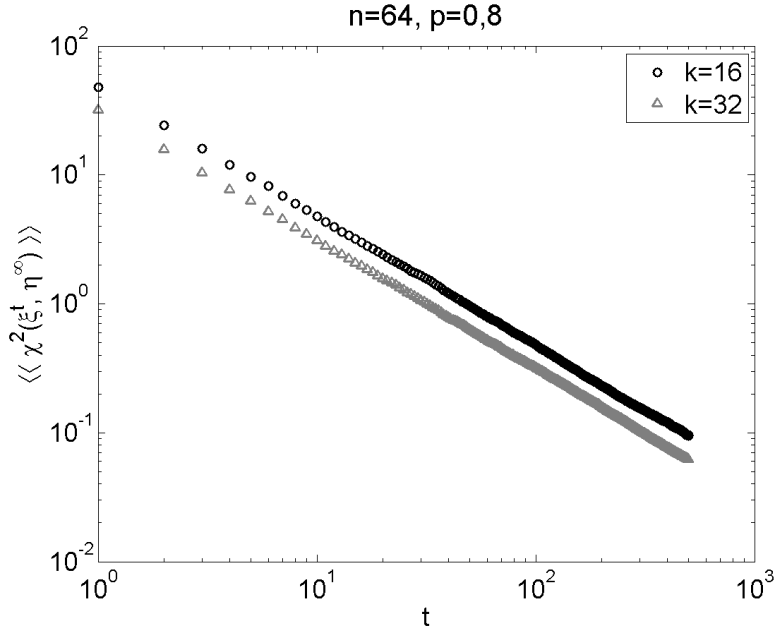
Observamos que os tempos de execução para o Algoritmo 1 aumentam quando a probabilidade de incluir cada aresta no grafo cai de 0,9 para 0,8. Esse aumento era esperado, uma vez que quando p diminui o permanente da matriz de adjacência também diminui, reduzindo a taxa de aceitação. Já os tempos de execução para o Algoritmo 2 se mostram insensíveis à variação da probabilidade p. No entanto, notamos que os tempos de execução parecem bastante altos, quando comparados com os das tabelas 1 e 2. Esse tempo elevado se deve ao fato de que,

para esses casos, foi realizada a reordenação das linhas da matriz. Por conta da regularidade das matrizes D_n e T_n julgamos desnecessário realizar o reordenamento nesses casos.

5.3 Avaliação de tempos de relaxação

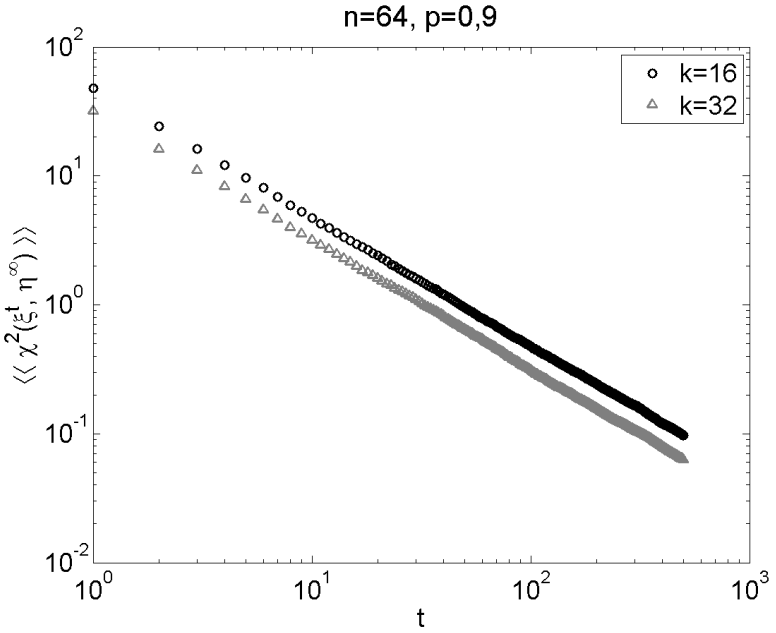
Realizamos simulações para o DTSEP(G) para os grafos completo aumentado ($\widetilde{K}_n$) e de Erdős-Rényi ($\widetilde{G}_{n,p}$), medindo a distância entre a distribuição empírica e a estacionária teórica segundo $\chi^2(\xi^t, \eta^\infty)$. Em ambos os casos fixamos $n = 64$, variando k entre os valores 16 e 32. Os valores para $\chi^2(\xi^t, \eta^\infty)$ foram ponderados sobre 10000 realizações independentes de ξ^t: a partir de 100 distribuições iniciais independentes η^0 e sobre 100 realizações independentes de $\widetilde{G}_{n,p}$. Esses valores ponderados serão denotados por $\langle\langle \chi^2(\xi^t, \eta^\infty)\rangle\rangle$. Para os grafos $\widetilde{G}_{n,p}$ usamos $p = 0{,}8$ e $0{,}9$. Na prática, os grafos $\widetilde{K}_n$ foram gerados como $\widetilde{G}_{n,1}$, ou seja, a probabilidade de incluir cada aresta foi fixada em 1. As permutações com restrições necessárias para evoluir o DTSEP(G) foram geradas a partir do Algoritmo 2, muito embora para grafos densos como os $\widetilde{G}_{n,p}$ com $p = 0,8$, $p = 0,9$ e $p = 1,0$, a diferença de desempenho entre os Algoritmos 1 e 2 não seja significativa. As figuras 7, 8 e 9 apresentam, respectivamente, gráficos das simulações para $\widetilde{G}_{64,p}$ com $p = 0,8$, $p = 0,9$ e $p = 1,0$.

Figura 7 – Distâncias para a distribuição estacionária em um DTSEP(G), onde G é um grafo de Erdős-Rényi com 64 vértices e $p = 0,8$ e para os casos com 16 e 32 partículas.



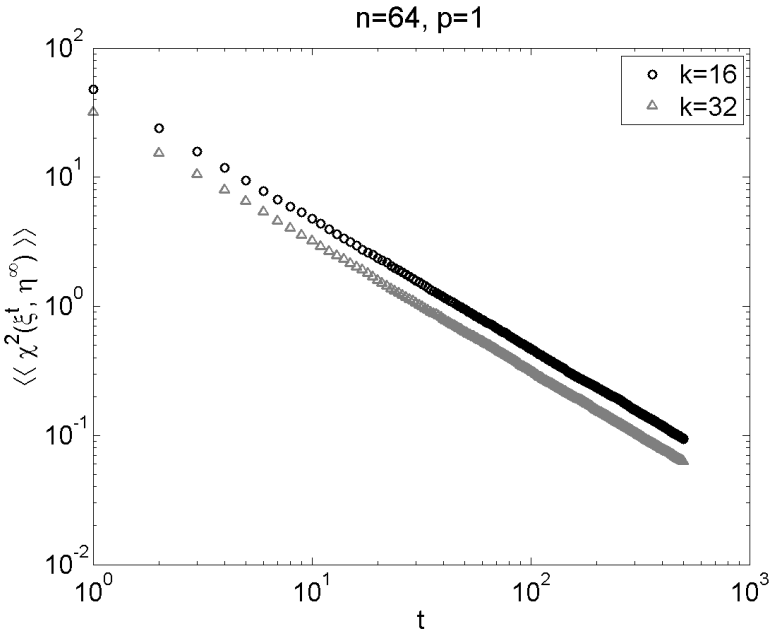
Fonte: Fábio Tosetto Reale, 2018

Figura 8 – Distâncias para a distribuição estacionária em um DTSEP(G), onde G é um grafo de Erdős-Rényi com 64 vértices e $p = 0,9$ e para os casos com 16 e 32 partículas.



Fonte: Fábio Tosetto Reale, 2018

Figura 9 – Distâncias para a distribuição estacionária em um DTSEP($\widetilde{K}_{64}$), onde $\widetilde{K}_n$ foi gerado como um grafo de Erdős-Rényi com 64 vértices e $p = 1$ e para os casos com 16 e 32 partículas.



Fonte: Fábio Tosetto Reale, 2018

Podemos observar que os gráficos são virtualmente idênticos. Além disso, as distâncias aparentam decrescer conforme alguma potência de t, uma vez que aparentam seguir uma reta na escala log-log. Por esse motivo, realizamos ajuste linear nos dados transformados pela expressão $\ln\left[\ \langle\langle \chi^2(\xi^t, \eta^\infty)\rangle\rangle\ \right] = a + b\ln(t)$. Observando os resultados dos ajustes, apresentados na tabela 4, vemos que $\chi^2(\xi^t, \eta^\infty) \approx \dfrac{n-k}{t}$.

Tabela 4 – Ajuste de curvas na forma $\langle\langle \chi^2(\xi^t, \eta^\infty)\rangle\rangle = e^a \cdot t^b$ para o DTSEP($\widetilde{G}_{64,p}$) com k partículas.

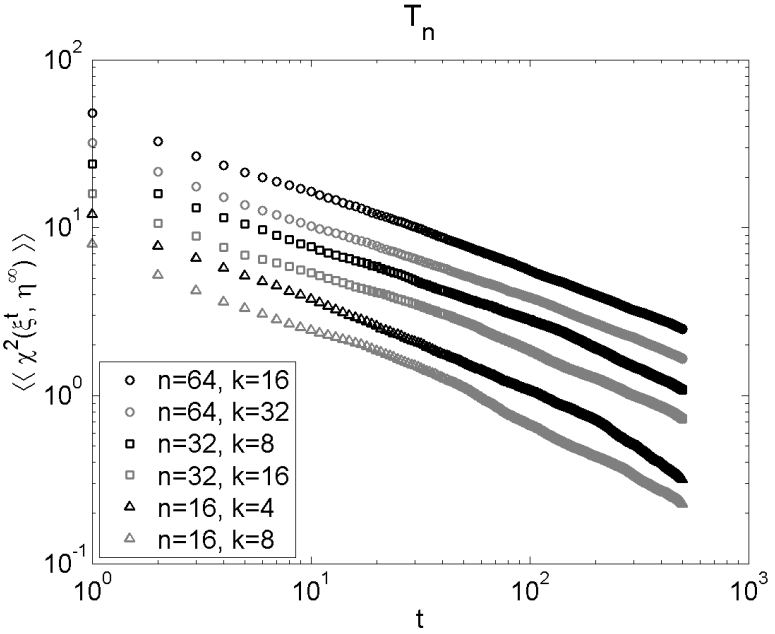
k	p	a	b
16	80	$3,884$	$-1,005$
16	90	$3,838$	$-0,992$
16	100	$3,852$	$-1,000$
32	80	$3,489$	$-1,008$
32	90	$3,483$	$-1,004$
32	100	$3,482$	$-1,005$

Fonte: Fábio Tosetto Reale, 2018.

Também realizamos simulações para grafos esparsos. Porém, como é essencial que o grafo analisado seja conexo, ao invés de utilizar grafos aleatórios de Erdős-Rényi com baixa probabilidade de inclusão de aresta, decidimos utilizar os grafos cujas matrizes de adjacência são dadas pelas T_n, apresentadas na Seção 4.1. Assim como para os grafos densos, medimos a distância entre a distribuição empírica e a estacionária teórica segundo $\chi^2(\xi^t, \eta^\infty)$, ponderando através de 100 trajetórias distintas. Assim como anteriormente, as permutações foram geradas pelo Algoritmo 2. Para o grafos densos, essa escolha fazia pouca diferença. Para os grafos esparsos, no entanto, esse não é o caso. Para $n = 16$, por exemplo, a taxa de aceitação para o Algoritmo 1 é dada por $\dfrac{\text{per}(T_{16})}{16!} \approx 7,6 \cdot 10^{-11}$. Já para o Algoritmo 2, a taxa de aceitação é aproximadamente $0,05$. Observe que, mesmo a taxa de aceitação do Algoritmo 2 sendo consideravelmente maior, para valores maiores de n, essa taxa também se torna bem baixa. Por essa razão, começamos com valores pequenos de n. A figura 10 apresenta os dados agregados para $n = 16, 32, 64$.

Nesse caso as curvas não são tão próximas a retas na escala log-log. Observamos um comportamento mais instável nos casos onde temos um número baixo de partículas. Além disso, observamos que o decaimento das distâncias no começo do processo é diferente do decaimento no final dele. Por essa razão, decidimos realizar ajuste nos dados a partir da metade dos instantes t simulados. Os valores ajustados se encontram na tabela 5 e indicam uma tendência de aumento no valor do expoente de t com o aumento do número de vértices.

Figura 10 – Distâncias para a distribuição estacionária em DTSEP(T_n), com $n = 16, 32, 64$ e onde o número de partículas é $n/2$ e $n/4$.



Fonte: Fábio Tosetto Reale, 2018

Tabela 5 – Ajuste de curvas na forma $\langle\langle \chi^2(\xi^t, \eta^\infty)\rangle\rangle = e^a \cdot t^b$ para o DTSEP(T_n) com k partículas. Para esse ajuste, utilizamos a segunda metade dos valores t, ou seja, $250 \leq t \leq 500$.

n	k	a	b
16	4	4,431	−0,895
16	8	3,129	−0,739
32	8	3,830	−0,602
32	16	3,361	−0,588
64	16	3,984	−0,492
64	32	3,852	−0,537

Fonte: Fábio Tosetto Reale, 2018.

6 Conclusões e perspectivas

6.1 Conclusões

Neste trabalho definimos o processo de exclusão simples simétrico em tempo discreto sobre grafos, que denominamos DTSEP(G), por meio de permutações com restrições sobre os índices dos vértices dos grafos e apresentamos dois algoritmos para amostrar as permutações com restrições necessárias à simulação desse processo: um algoritmo simples de aceitação-rejeição (Algoritmo 1) e um algoritmo de amostragem sequencial por importância (Algoritmo 2). Esses algoritmos, apresentados no Capítulo 3, foram implementados, testados e comparados através de sua utilização na estimação de permanentes, como descrito nas Seções 4.1 e 5.1. A título de aplicação em modelagem, os algoritmos foram também aplicados na estimação dos tempos de relaxação do DTSEP(G) para grafos aleatórios densos de Erdős-Rényi com laços, como apresentado nas Seções 4.2 e 4.3.

O DTSEP(G) fornece uma plataforma interessante para o desenvolvimento de vários assuntos diferentes, da mesma forma que o embaralhamento de um baralho de cartas (que corresponde a $G = K_n$ com $k = n$ partículas distinguíveis) tem sido uma plataforma para o desenvolvimento de uma grande variedade de ideias matemáticas nas últimas décadas. Neste trabalho buscamos um objetivo modesto: definir o processo e investigar os rudimentos de sua simulação estocástica.

Poderíamos ter considerado o DTSEP (G) sobre grafos simples sem laços; a única diferença seria que o volume de $\mathscr{S}_n(A)$ nestes casos seria necessariamente menor. O modelo com laços, no entanto, nos parece mais natural do ponto de vista da modelagem de fenômenos naturais. Outra vantagem da configuração com laços é que ela recupera o processo simétrico sobre G usual (a tempo contínuo) se limitarmos a dinâmica a uma única transposição por passo de tempo.

O formalismo desenvolvido neste trabalho também se aplica a processos de exclusão assimétricos, com G um digrafo e a matriz de adjacência $A(G)$ assimétrica. O caso assimétrico pode levar a situações interessantes, com componentes fortemente conexos e os gargalos resultantes entre eles afetando significativamente a dinâmica. Os estados estacionários do processo de exclusão totalmente assimétrico sobre G são um problema potencialmente muito complicado. Por exemplo, se G é acíclico, a dinâmica pode eventualmente "congelar". Laços nos vértices de G impediriam que $\mathscr{S}_n(A)$ viesse eventualmente a ser o conjunto vazio. Processos de exclusão

parcialmente assimétricos podem ser mais tratáveis e mais interessantes ainda. Neste caso as arestas possuem pesos diferentes para as transições de $u \to v$ e de $v \to u$, com u e v vértices de G.

6.2 Perspectivas

A representação dual para o DTSEP(G) descrita na Seção 2.3 abre algumas possibilidades interessantes de modelagem matemática. Por exemplo, poderíamos marcar uma ou mais partículas e deixá-las saltar para qualquer vértice do grafo, como se fossem "partículas voadoras", enquanto as outras partículas se movem de acordo com as conexões locais (arestas) disponíveis. Isto pode ser útil para modelar processos de reação e difusão de partículas com diferentes difusividades, modelos do tipo predador-presa e demais modelos semelhantes. Pretendemos explorar algumas dessas possibilidades em trabalhos futuros.

Da mesma forma, talvez seja possível otimizar o Algoritmo 2 através de uma estratégia de *backtracking*, evitando que permutações que incorrem em problemas durante sua construção sejam descartadas por completo. Isso deve fazer diferença nos casos em que seja necessário gerar permutações relativamente grandes (digamos, $n \gtrsim 1000$) para várias matrizes diferentes (por exemplo, quando cada matriz de restrições A representa a realização de um grafo aleatório, como fizemos na Seção 4.2), com somas de linhas R_i bastante diferentes entre si. Esse caso teria ocorrido caso estivéssemos usando grafos segundo o modelo de Barabási–Albert ao invés de grafos de Erdős-Rényi na Seção 4.2. Nestes casos a etapa de ordenação das linhas das matrizes dominaria o tempo de execução e uma implementação com *backtracking* poderia melhorar o desempenho do algoritmo.

Referências[1]

1 VAN STEEN, M. *Graph Theory and Complex Networks: An Introduction*. 2a. ed. Amsterdam: Maarten van Steen, 2010. ISBN 978-90-815406-1-2. Disponível em: ⟨https://www.distributed-systems.net/index.php/books/gtcn/⟩. Citado nas páginas 13 e 34.

2 BONDY, J. A.; MURTY, U. S. R. *Graph Theory*. 2a. ed. New York: Springer, 2010. (Graduate Texts in Mathematics 244). ISBN 978-1-84628-969-9. Citado nas páginas 13, 20 e 34.

3 FRALEIGH, J. B. *A First Course in Abstract Algebra*. 6a. ed. Reading, MA: Addison-Wesley, 2000. ISBN 0-201-33596-4. Citado na página 15.

4 CHARALAMBIDES, C. A. *Enumerative Combinatorics*. Boca Raton, CA: Chapman & Hall/CRC Press, 2002. ISBN 1-58488-290-5. Citado na página 15.

5 ANDREWS, G. E.; ERIKSSON, K. *Integer Partitions*. Cambridge, UK: Cambridge University Press, 2004. ISBN 0-521-84118-6. Citado na página 17.

6 RYSER, H. J. *Combinatorial Mathematics*. Rahway, NJ: Mathematical Association of America – Printed by Quinn & Boden Co., 1963. (Carus Mathematical Monographs 14). Citado nas páginas 18, 19 e 26.

7 BRUALDI, R. A.; RYSER, H. J. *Combinatorial Matrix Theory*. Cambridge, UK: Cambridge University Press, 1991. (Encyclopedia of Mathematics and Its Applications 39). ISBN 0-521-32265-0. Citado nas páginas 18 e 26.

8 MINC, H. *Permanents*. Reading, MA: Addison-Wesley, 1978. (Encyclopedia of Mathematics and its Applications 6). ISBN 0-201-13505-1. Citado na página 18.

9 VALIANT, L. G. The complexity of computing the permanent. *Theoretical Computer Science*, v. 8, n. 2, p. 189–201, 1979. ISSN 0304-3975. Citado na página 19.

10 VALIANT, L. G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, v. 8, n. 3, p. 410–421, 1979. ISSN 0097-5397. Citado na página 19.

11 BRODER, A. Z. How hard is it to marry at random? (On the approximation of the permanent). In: *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*. New York: ACM, 1986. (STOC '86), p. 50–58. ISBN 0-89791-193-8. Citado na página 19.

12 NIJENHUIS, A.; WILF, H. S. *Combinatorial Algorithms: For Computers and Calculators*. 2a. ed. New York: Academic Press, 1978. (Computer Science and Applied Mathematics Series). ISBN 0-12-519260-6. Citado na página 19.

13 KARMARKAR, N. et al. A Monte-Carlo algorithm for estimating the permanent. *SIAM Journal on Computing*, v. 22, n. 2, p. 284–293, 1993. ISSN 0097-5397. Citado na página 19.

14 JERRUM, M.; SINCLAIR, A. Approximating the permanent. *SIAM Journal on Computing*, v. 18, n. 6, p. 1149–1178, 1989. ISSN 0097-5397. Citado na página 19.

15 JERRUM, M.; SINCLAIR, A.; VIGODA, E. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, v. 51, n. 4, p. 671–697, 2004. ISSN 0004-5411. Citado na página 19.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

16 SINCLAIR, A. J. *Algorithms for Random Generation & Counting: A Markov Chain Approach*. Boston: Birkhäuser, 1993. ISBN 0-8176-3658-7. Citado na página 19.

17 KUZNETSOV, N. Y. Computing the permanent by importance sampling method. *Cybernetics and Systems Analysis*, v. 32, n. 6, p. 749–755, 1996. ISSN 1060-0396. Citado nas páginas 19 e 29.

18 RASMUSSEN, L. E. Approximating the permanent: A simple approach. *Random Structures & Algorithms*, v. 5, n. 2, p. 349–361, 1994. ISSN 1042-9832. Citado nas páginas 19 e 29.

19 BEICHL, I.; SULLIVAN, F. Approximating the permanent via importance sampling with application to the dimer covering problem. *Journal of Computational Physics*, v. 149, n. 1, p. 128–147, 1999. ISSN 0021-9991. Citado nas páginas 19 e 29.

20 SMITH, P.; DAWKINS, B. Estimating the permanent by importance sampling from a finite population. *Journal of Statistical Computation and Simulation*, v. 70, n. 3, p. 197–214, 2001. ISSN 0094-9655. Citado nas páginas 19 e 29.

21 LIGGETT, T. M. *Interacting Particle Systems*. Berlin: Springer-Verlag, 1985. (Grundlehren der Mathematischen Wissenschaften 276). ISBN 3-540-22617-6. Citado na página 21.

22 SPOHN, H. *Large Scale Dynamics of Interacting Particles*. Berlin: Springer-Verlag, 1991. (Texts and Monographs in Physics). ISBN 3-540-53491-1. Citado na página 21.

23 SCHÜTZ, G. Exactly solvable models for many-body systems far from equilibrium. In: DOMB, C.; LEBOWITZ, J. L. (Ed.). *Phase Transitions and Critical Phenomena*. London: Academic Press, 2001, (Phase Transitions and Critical Phenomena, v. 19). p. 1–251. Citado na página 21.

24 ARITA, C. Queueing process with excluded-volume effect. *Physical Review E*, v. 80, n. 5, p. 051119, 2009. ISSN 2470-0045. Citado na página 21.

25 ARITA, C.; YANAGISAWA, D. Exclusive queueing process with discrete time. *Journal of Statistical Physics*, v. 141, n. 5, p. 829–847, 2010. ISSN 0022-4715. Citado na página 21.

26 CHOWDHURY, D.; SANTEN, L.; SCHADSCHNEIDER, A. Statistical physics of vehicular traffic and some related systems. *Physics Reports*, v. 329, n. 4, p. 199–329, 2000. ISSN 0370-1573. Citado na página 21.

27 SCHADSCHNEIDER, A.; CHOWDHURY, D.; NISHINARI, K. *Stochastic Transport in Complex Systems: From Molecules to Vehicles*. Amsterdam: Elsevier, 2011. ISSN 978-0-444-52853-7. Citado na página 21.

28 GKANTSIDIS, C.; MIHAIL, M.; SABERI, A. Random walks in peer-to-peer networks: Algorithms and evaluation. *Performance Evaluation*, v. 63, n. 3, p. 241–263, 2006. ISSN 0166-5316. Citado na página 21.

29 ALDOUS, D.; FILL, J. A. Reversible Markov Chains and Random Walks on Graphs. Monografia inacabada; versão recompilada (2014). 2002. Disponível em: ⟨http://www.stat.berkeley.edu/~aldous/RWG/book.html⟩. Citado na página 21.

30 DIACONIS, P. *Group Representations in Probability and Statistics*. Hayward, CA: Institute of Mathematical Statistics, 1988. v. 11. i–192 p. ISSN 0749-2170. Citado na página 21.

31 LOVÁSZ, L. Random walks on graphs: A survey. In: MIKLÓS, D.; SÓS, V. T.; SZŐNYI, T. (Ed.). *Combinatorics, Paul Erdős is Eighty*. Budapest: János Bolyai Mathematical Society, 1996. v. 2, p. 353–398. ISBN 9-638-02275-2. Citado na página 21.

32 KESSEL, A. et al. Microscopic simulation of pedestrian crowd motion. In: SCHRECKENBERG, M.; SHARMA, S. D. (Ed.). *Pedestrian and Evacuation Dynamics*. Berlin: Springer-Verlag, 2002. p. 193–200. ISBN 978-3-540-42690-5. Citado na página 21.

33 APPERT-ROLLAND, C.; CIVIDINI, J.; HILHORST, H. J. Frozen shuffle update for an asymmetric exclusion process on a ring. *Journal of Statistical Mechanics: Theory and Experiment*, v. 2011, n. 07, p. P07009, 2011. ISSN 1742-5468. Citado na página 21.

34 KNUTH, D. E. *The Art of Computer Programming: Seminumerical Algorithms*. 3a. ed. Reading, MA: Addison-Wesley, 1998. ISBN 0-201-89684-2. Citado na página 28.

35 LIU, J. S.; CHEN, R. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, v. 93, n. 443, p. 1032–1044, 1998. ISSN 0162-1459. Citado na página 29.

36 DIACONIS, P.; GRAHAM, R.; HOLMES, S. P. Statistical problems involving permutations with restricted positions. *Lecture Notes-Monograph Series*, Institute of Mathematical Statistics, Hayward, CA, v. 36, p. 195–222, 2001. ISSN 0749-2170. Citado nas páginas 29 e 30.

37 CHEN, Y. et al. Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, v. 100, n. 469, p. 109–120, 2005. ISSN 0162-1459. Citado nas páginas 29 e 30.

38 CHEN, Y.; LIU, J. S. Sequential Monte Carlo methods for permutation tests on truncated data. *Statistica Sinica*, v. 17, n. 3, p. 857–872, 2007. ISSN 1017-0405. Citado nas páginas 29, 30 e 34.

39 BENDER, E. A. The asymptotic number of non-negative integer matrices with given row and column sums. *Discrete Mathematics*, v. 10, n. 2, p. 217–223, 1974. ISSN 0012-365X. Citado na página 34.

40 ERDŐS, P.; RÉNYI, A. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, v. 6, p. 290–297, 1959. ISSN 0033-3883. Citado na página 34.

Apêndices

Apêndice A – Cálculo dos permanentes das matrizes T_n e D_n

Começaremos pelo caso mais simples das matrizes T_n, que têm a forma

$$T_n = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{32}$$

Observe que T_{n-1} e T_{n-2} são submatrizes de T_n,

$$T_n = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & & & & \\ 0 & & T_{n-1} & & \\ \vdots & & & & \\ 0 & & & & \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ 0 & 1 & & & \\ \vdots & \vdots & & T_{n-2} & \\ 0 & 0 & & & \end{pmatrix}.$$

Empregando o desenvolvimento de Laplace para o permanente de T_n a partir de sua primeira linha (v. equação (6)) temos

$$\operatorname{per}(T_n) = \operatorname{per}(T_{11}) + \operatorname{per}(T_{12})$$

(não confunda T_{11} ou T_{12} aqui com T_n com $n = 11$ ou $n = 12$). Claramente, $T_{11} = T_{n-1}$. Aplicando agora o desenvolvimento de Laplace a T_{12} a partir de sua primeira coluna (que contém um único elemento 1) obtemos $\operatorname{per}(T_{12}) = \operatorname{per}(T_{n-2})$. Assim,

$$\operatorname{per}(T_n) = \operatorname{per}(T_{n-1}) + \operatorname{per}(T_{n-2}),$$

e obtemos para $\operatorname{per}(T_n)$ uma relação de recorência idêntica à da sequência de Fibonacci. Uma vez que $\operatorname{per}(T_1) = 1 = F_2$ e $\operatorname{per}(T_2) = 2 = F_3$ deduzimos que $\operatorname{per}(T_n) = F_{n+1}$.

As matrizes de desarranjos D_n são dadas por

$$D_n = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 1 & 0 & 1 \\ 1 & \cdots & 1 & 1 & 1 & 0 \end{pmatrix},$$

e, assim como no caso anterior, podemos observar as submatrizes D_{n-1} e D_{n-2} em D_n:

$$D_n = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & & & & \\ 1 & & D_{n-1} & & \\ \vdots & & & & \\ 1 & & & & \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & & & \\ \vdots & \vdots & & D_{n-2} & \\ 1 & 1 & & & \end{pmatrix}.$$

Utilizaremos novamente o desenvolvimento de Laplace para encontrar uma relação de recorrência para $\operatorname{per}(D_n)$. Desenvolvendo o permanente de D_n a partir de sua primeira linha temos, de forma geral, que

$$\operatorname{per}(D_n) = \sum_{j=1}^{n} d_{1j} \operatorname{per}(D_{1j}) = \sum_{j=2}^{n} \operatorname{per}(D_{1j}), \tag{33}$$

já que $d_{11} = 0$. Observamos agora que ao trocarmos entre si as linhas j e $j+1$ de uma submatriz $D_{1,j+1}$ qualquer obtemos a matriz D_{1j}. Como o permanente é invariante pela troca de linhas, concluímos que $\operatorname{per}(D_{1j}) = \operatorname{per}(D_{12})$ para todo $2 \leq j \leq n$. Definindo $B_{n-1} = D_{12}$ temos que

$$\operatorname{per}(D_n) = (n-1) \operatorname{per}(B_{n-1}) \tag{34}$$

Reduzimos assim o problema de calcular $\operatorname{per}(D_n)$ ao problema de calcular o permanente de matrizes B_n da forma

$$B_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & & & & \\ 1 & & D_{n-1} & & \\ \vdots & & & & \\ 1 & & & & \end{pmatrix}$$

O desenvolvimento de Laplace de $\operatorname{per}(B_n)$ pela primeira linha fornece

$$\operatorname{per}(B_n) = \operatorname{per}(D_{n-1}) + \sum_{j=2}^{n} \operatorname{per}(D_{1j}) = \operatorname{per}(D_{n-1}) + \operatorname{per}(D_n); \tag{35}$$

a primeira igualdade segue do fato de que B_n difere de D_n apenas pela primeira linha. Tomando a equação (35) para $n-1$ e substituindo na equação (34) chegamos à relação de recorrência

$$\operatorname{per}(D_n) = (n-1)\big[\operatorname{per}(D_{n-1}) + \operatorname{per}(D_{n-2})\big]. \tag{36}$$

Distribuindo o termo $n-1$ encontramos a relação

$$\operatorname{per}(D_n) - n \operatorname{per}(D_{n-1}) = (-1)\big[\operatorname{per}(D_{n-1}) - (n-1) \operatorname{per}(D_{n-2})\big]. \tag{37}$$

De maneira geral, pode-se mostrar que

$$\operatorname{per}(D_n) - n\operatorname{per}(D_{n-1}) = (-1)^k\big[\operatorname{per}(D_{n-k}) - (n-k)\operatorname{per}(D_{n-1-k})\big], \tag{38}$$

com $\operatorname{per}(D_2) = 1$ e $\operatorname{per}(D_1) = 0$. Fazendo $k = n-2$ obtemos

$$\operatorname{per}(D_n) - n\operatorname{per}(D_{n-1}) = (-1)^n, \tag{39}$$

e dividindo os dois lados dessa equação por $n!$ e somando para obtermos cancelamentos sucessivos chegamos finalmente em

$$\operatorname{per}(D_n) = n!\sum_{i=2}^{n}\frac{(-1)^i}{i!}, \tag{40}$$

que é o resultado mencionado na Seção 4.1.

Quando nos referimos ao resultado acima como "clássico", certamente não estávamos nos referindo ao cálculo de $\operatorname{per}(D_n)$, e sim à resolução do famoso *problème des rencontres* (em português, o problema dos encontros). O problema dos encontros foi proposto pela primeira vez pelo matemático francês Pierre Rémond de Montmort (1678–1719) em 1708 e resolvido por ele mesmo e pelo matemático suíço Nicolas Bernoulli (1695–1726) em 1713.[1] Em sua formulação original, o problema envolvia dois jogadores segurando mãos idênticas de 2 a 13 cartas, a do desafiante na ordem usual (por exemplo, A, 2, 3, …, 10, J, Q, K) e a do apostador bem embaralhada. O jogo consistia em irem os jogadores, simultânea e sequencialmente, mostrando suas cartas até que duas cartas coincidissem, caso em que o desafiante ganhava, ou as cartas terminassem, quando então o apostador ganhava. Obviamente, a probabilidade do apostador ganhar é dada pela probabilidade de que a ordem das cartas de seu baralho seja um desarranjo da ordem usual. Essa probabilidade é dada por $d_n/n!$, onde os chamados números de *rencontres* d_n nada mais são que $\operatorname{per}(D_n)$. Uma notação aceita para esses números é $d_n = !n$, que se lê "n subfatorial". Começando em $n = 1$, os primeiros d_n valem 0, 1, 2, 9, 44, 265, 1854, 14833, 133496 e 1334961 (OEIS A000166). Como se pode ver, o jogo de Montmort só era honesto com $n = 2$ cartas! Com $n \geq 4$ cartas o desafiante poderia até apostar \$3 contra \$2 em sua vitória (a fim de persuadir o apostador a entrar no jogo) e ainda assim esperar lucrar com as apostas.

O *problème des rencontres* aparece parafraseado de inúmeras maneiras na literatura matemática elementar, por exemplo, como o problema dos chapéus (cavalheiros *vs*. seus chapéus num restaurante em que acaba a luz), das cartas (envelopes *vs*. destinatários nas mãos de um carteiro atrapalhado) etc. Nos textos elementares o problema é resolvido através do princípio da

[1] Veja o artigo na Wikipedia em francês, ⟨https://fr.wikipedia.org/wiki/probleme_des_rencontres⟩.

inclusão-exclusão: seja $N(i_1,\ldots,i_k)$ o número de permutações de n objetos que fixam as posições $\sigma(i_1)=i_1,\ \ldots,\ \sigma(i_k)=i_k$. Temos $N(i_1,\ldots,i_k)=(n-k)!$, já que ao fixarmos k índices restam ainda $n-k$ índices que podem ser permutados à vontade. Como existem $\binom{n}{k}$ maneiras diferentes de escolher os índices $i_1,\ \ldots,\ i_k$ em $N(i_1,\ldots,i_k)$, o número de permutações que não fixam nenhum índice, isto é, o número de desarranjos, é dado pelo princípio da inclusão-exclusão por

$$\begin{aligned} d_n &= n! - \sum_i N(i) + \sum_{i<j} N(i,j) - \cdots + (-1)^n N(1,2,\ldots,n) = \\ &= n! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! - \cdots + (-1)^n \binom{n}{n} 0! = \\ &= n!\Big(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + \frac{(-1)^n}{n!}\Big), \end{aligned} \tag{41}$$

exatamente como antes, veja a equação (40).

Apêndice B – Taxa de sucesso do Algoritmo 2 para matrizes T_n

O Algoritmo 2, apresentado na Seção 3.2, produz permutações respeitando as restrições dadas por uma matriz de adjacência. Porém, esse algoritmo pode falhar caso a variável R_i venha a ser nula para alguma iteração i no passo 4. Essa falha ocorre pois a variável R_i conta o número de atribuições possíveis para $\sigma(i)$. Não é fácil determinar a probabilidade dessa ocorrência no caso geral, porém, a regularidade encontrada em T_n nos permite calcular essa probabilidade através de uma relação de recorrência.

Definimos por f_n a taxa de sucesso do Algoritmo 2 com T_n como a matriz de entrada. Trivialmente, a taxa de sucesso é 1 para T_1 e T_2, ou seja $f_1 = f_2 = 1$. No primeiro caso só podemos ter $\sigma(1) = 1$. No segundo caso, $\sigma(1) = 1$ ou $\sigma(1) = 2$ de forma que existe exatamente uma opção para $\sigma(2)$, ou seja, $R_2 = 1$ e o algoritmo retorna uma permutação sem falhar.

Para calcular a probabilidade de sucesso para um n qualquer, nos aproveitaremos da mesma observação feita no apêndice A, no caso, que podemos escrever T_n em função de T_{n-1} ou T_{n-2} como

$$T_n = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & & & & \\ 0 & & & & \\ \vdots & & T_{n-1} & & \\ 0 & & & & \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ 0 & 1 & & & \\ \vdots & \vdots & & T_{n-2} & \\ 0 & 0 & & & \end{pmatrix}.$$

O algoritmo começa atribuindo algum valor para $\sigma(1)$. A linha 1 de T_n faz com que somente os valores 1 ou 2 possam ser atribuídos a $\sigma(1)$. Para os casos:

(i) $\sigma(1) = 1$. Em seguida, o algoritmo exclui o valor 1 do conjunto J e segue produzindo uma permutação segundo as restrições de T_{n-1}, cuja probabilidade de sucesso é f_{n-1}.

(ii) $\sigma(1) = 2$. Em seguida, o algoritmo exclui o valor 2 do conjunto J e retorna ao começo do laço principal fazendo $i = 2$. Agora, as possibilidades de escolha para $\sigma(2)$ são 1 ou 3. Observe, pela primeira coluna de T_n, que só podemos ter $\sigma(i) = 1$ para $i = 1$ ou $i = 2$. Dessa forma, se tivermos $\sigma(1) = 2$ e $\sigma(2) = 3$ não podemos ter $\sigma(i) = 1$ para nenhum i e o algoritmo falha eventualmente[1]. Por outro lado, se tivermos $\sigma(1) = 2$ e $\sigma(2) = 1$, o algoritmo exclui os valores 1 e 2 do conjunto J e segue produzindo uma permutação segundo as restrições de T_{n-2}, cuja probabilidade de sucesso é f_{n-2}.

[1] O algoritmo só encerra com falha na última iteração, pois teremos $\sigma(i) = i+1$ até que, finalmente, $R_n = 0$.

Portanto, temos que

$$f_n = \frac{1}{2} \cdot f_{n-1} + \frac{1}{2} \cdot (\frac{1}{2} \cdot f_{n-2} + \frac{1}{2} \cdot 0) = \frac{1}{2} \cdot f_{n-1} + \frac{1}{4} \cdot f_{n-2}. \tag{42}$$

Em seguida, escrevemos o sistema linear

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} f_2 \\ f_1 \end{pmatrix}. \tag{43}$$

Substituindo os valores para f_1 e f_2, obtemos

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{44}$$

Em seguida, calculamos a exponenciação encontrada na equação (44) utilizando a diagonalização da matriz. Os autovalores da matriz de interesse são $\lambda_1 = \frac{1+\sqrt{5}}{4}$ e $\lambda_2 = \frac{1-\sqrt{5}}{4}$. Os autovetores associados são $v_1 = (\lambda_1 \ 1)^T$ e $v_2 = (\lambda_2 \ 1)^T$, respectivamente. Assim, a expressão da equação (44) é equivalente a

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{pmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1^{n-1} & 0 \\ 0 & \lambda_2^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{45}$$

Por fim, realizamos os produtos matriciais para encontrar uma expressão fechada para f_n (e f_{n+1})

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{pmatrix} (1-\lambda_2)\lambda_1^n - (1-\lambda_1)\lambda_2^n \\ (1-\lambda_2)\lambda_1^{n-1} - (1-\lambda_1)\lambda_2^{n-1} \end{pmatrix}. \tag{46}$$

Substituindo os valores de λ_1 e λ_2 e separando a equação relativa a f_n temos

$$f_n = \left(\frac{2}{\sqrt{5}}\right) \cdot \left[\frac{(3+\sqrt{5})(1+\sqrt{5})^{n-1} - (3-\sqrt{5})(1-\sqrt{5})^{n-1}}{4^n}\right]. \tag{47}$$

A tabela 6 apresenta os valores calculados para algumas dessas taxas de sucesso.

Tabela 6 – Taxa de sucesso f_n para o Algoritmo 2, quando a matriz de entrada é T_n.

n	f_n
16	$0,0487$
24	$0,0089$
32	$0,0016$
40	$3,01 \cdot 10^{-4}$
48	$5,53 \cdot 10^{-5}$
56	$1,01 \cdot 10^{-5}$
64	$1,86 \cdot 10^{-6}$

Fonte: Fábio Tosetto Reale, 2018

Apêndice C – Código em Julia para o Algoritmo 1

```
function permutacao_fyd(A::Matrix{T}) where {T<:Union{Int, Bool}}
    n = size(A)[1]
    sigma = Vector{Int}(n)          # vetor para retorno da permutação
    ac = zero(T)                    # variável "booleana" de aceitação
    k = 0                           # contador de permutações produzidas
    while ac == zero(T)
        rnd = rand(Float64, n-1)    # n-1 números pseudo-aleatórios
        sigma[1] = 1
        for i in 2:n
            j = 1 + floor(Int, i*rnd[i-1])
            sigma[i] = sigma[j]
            sigma[j] = i
        end
        ac = accept_perm(A, sigma)
        k += 1
    end
    return sigma, k
end

accept_perm(A::Matrix{Int}, sigma::Vector{Int}) = prod(diag(A[:, sigma]))
accept_perm(A::Matrix{Bool}, sigma::Vector{Int}) = all(diag(A[:, sigma]))
```

Fonte: Fábio Tosetto Reale, 2018.

Apêndice D – Código em Julia para o Algoritmo 2

```
function permutacao_sis(A::Matrix{T}) where {T<:Union{Int,Bool}}
   n = size(A)[1]
   sigma = Vector{Int}(n)  # vetor para retorno da permutação
   R = zeros(Int, n)       # vetor para retorno dos valores R_i
   J = ones(T, n)          # identificador de índices disponíveis
   for i in 1:n
      v = find(el -> el == one(T), J.*A[i,:])
      R[i] = length(v)     # v é vetor de índices disponíveis
      if R[i] != 0
         j = rand(v)       # seleciona uniformemente elemento de v
         sigma[i] = j
         J[j] = 0
      else                 # caso PARE
         return sigma, R
      end
   end
   return sigma, R
end
```

Fonte: Fábio Tosetto Reale, 2018.