JAI 2024

43ª Jornada de Atualização em Informática

Organizadores
Claudia Cappelli
Eduardo Adilio Pelinson Alchieri

44º CSBC 2024
21 A 25 DE JULHO - BRASÍLIA-DF
CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

SBC
Sociedade Brasileira de Computação

Organizadores

Claudia Cappelli
Eduardo Alchieri

43ª Jornada de Atualização em Informática
JAI 2024

Porto Alegre
Sociedade Brasileira de Computação – SBC
2024

Dados Internacionais de Catalogação na Publicação (CIP)

C749 Congresso da Sociedade Brasileira de Computação (44.: 21 jul.-25 jul. 2024 : Brasília)
43ª Jornada de Atualização em Informática (JAI 2024) [recurso eletrônico] / organização: Claudia Cappelli; Eduardo Alchieri. Dados eletrônicos. - Porto Alegre: Sociedade Brasileira de Computação, 2024.
199 p.; il.: PDF; 5,30 MB

Inclui bibliografia
ISBN 978-85-7669-597-4 (e-book)

1. Computação - Brasil - Congressos. 2. Informática. I. Cappelli, Claudia. II. Alchieri, Eduardo. III. Sociedade Brasileira de Computação. IV. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali - CRB-10/2339

Biblioteca Digital da SBC - SBC OpenLib

Índice para catálogo sistemático:

1. Ciência e tecnologia informáticas: Computação: Processamento de dados - Publicação de conferências, congressos, simpósios etc... 004(063)

43ª Jornada de Atualização em Informática (JAI 2024)

Editora

Sociedade Brasileira de Computação (SBC)

Coordenação Geral do CSBC 2024

Marcelo Antonio Marotta (UnB)
Lisandro Zambenedetti Granville (UFRGS)

Coordenação da JAI 2024

Claudia Cappelli (UERJ)
Eduardo Alchieri (UnB)

Prefácio

É com prazer que apresentamos a edição de 2024 do livro da série Atualizações em Informática. Os capítulos deste livro constituem um material de apoio aos minicursos selecionados para serem apresentados nas Jornadas de Atualização em Informática (JAI), tradicionalmente realizadas em conjunto com o Congresso da Sociedade Brasileira de Computação (SBC). As JAI são um dos mais relevantes eventos acadêmicos de atualização científica e tecnológica da comunidade brasileira de Computação.

Os minicursos das JAI tratam de temas atuais e relevantes, são ministrados por pesquisadores experientes e constituem uma excelente oportunidade de atualização para acadêmicos e profissionais da área. Nesta edição foram apresentados 4 minicursos durante o Congresso da SBC. No processo de seleção, cada proposta foi avaliada por pelo menos três avaliadores e os textos dos minicursos, que correspondem aos capítulos desse livro, foram revisados por membros da comissão para garantir a qualidade final deles.

O Capítulo 1 traz uma discussão profunda sobre três aspectos essenciais e impulsionadores do ecossistema de finanças descentralizadas (DeFi): (1) aplicações inovadoras, (2) interoperabilidade entre redes blockchain, e (3) segurança para as vulnerabilidades de ambientes descentralizados e competitivos. Nesse sentido, o capítulo também apresenta os fundamentos e exemplos práticos das aplicações DeFi mais populares atualmente, além de discutir as pesquisas no estado da arte sob os três aspectos acima mencionados que suportarão o crescimento de DeFi nos próximos anos.

O Capítulo 2 aborda e contextualiza o aprendizado auto-supervisionado como uma alternativa para aplicações dinâmicas de rede, em que a rotulagem de dados é um desafio crítico devido à discrepância entre a taxa de geração de tráfego e a taxa de rotulagem manual dos dados. São apresentadas técnicas de aprendizado auto-supervisionado generativo e contrastivo por serem eficazes para melhorar o desempenho da rede, expandindo o número de amostras rotuladas e reconhecendo semelhanças e diferenças entre exemplos de amostras. Por fim, o capítulo apresenta os

algoritmos de aprendizado auto-supervisionado e suas características e aplicações em redes, visando capacitar os leitores a compreenderem os princípios, arcabouços e limitações dessa técnica.

O Capítulo 3 apresenta os principais conceitos da tecnologia NoSQL. Além disso, o capítulo mostra como avaliar se essa tecnologia é apropriada para o projeto de banco de dados de um determinado sistema. Adicionalmente, o capítulo também discute algumas estratégias para o ajuste de desempenho de bancos de dados NoSQL.

O Capítulo 4 fornece uma visão geral dos métodos e técnicas diferencialmente privadas para proteger informações sensíveis e, simultaneamente, permitir análises relevantes de redes sociais. São exploramos os princípios da privacidade diferencial, destacando seus mecanismos para adicionar ruído aos dados para evitar a reidentificação dos indivíduos. Além disso, são investigadas as estratégias para aplicar privacidade diferencial na análise de dados em redes sociais, abrangendo a publicação de dados, a análise de grafos e tarefas de aprendizado de máquina de maneira privada.

Gostaríamos de agradecer aos autores, pela submissão das propostas e geração dos textos finais, e à Comissão de Avaliação, pela dedicação e eficiência em todo o processo de seleção dos minicursos. Além disso, esperamos também que todo o material gerado nesta edição contribua para a formação de alunos e profissionais da área.

Claudia Cappelli
Eduardo Alchieri

Coordenadores das JAI 2024

Sumário

Capítulo 1

Finanças Descentralizadas em Redes Blockchain: Perspectivas sobre Pesquisa e Inovação em Aplicações, Interoperabilidade e Segurança

Josué N. Campos, Ronan D. Mendonça, Alexandre Fontinele, Luís H. S. de Carvalho, Isdael R. Oliveira, Ítallo W. F. Cardoso, Rafael Coelho, Allan E. S. Freitas, Glauber D. Gonçalves, José A. M. Nacif, Alex B. Vieira

Abstract

The support for smart contracts in blockchain networks has led to the emergence of a decentralized and automated finance ecosystem, DeFi (Decentralized Finance). The DeFi ecosystem aims to enhance traditional financial services through token trading, to reduce intermediaries and barriers to credit, and to broaden access to financial services in the context of the decentralized web. In this chapter, we provide an in-depth discussion of three essential and driving aspects of the DeFi ecosystem: (1) innovative applications, (2) interoperability between blockchain networks, and (3) security of the decentralized protocols that rule blockchain. In this sense, we currently present the fundamentals and practical examples of the most popular DeFi applications. We also discuss state-of-the-art research focusing on the three aspects above that will support the growth of DeFi in the coming years.

Resumo

O suporte a contratos inteligentes em redes blockchain propiciou a emergência de um novo ecossistema de finanças descentralizado e automatizado, denominado DeFi, do inglês, decentralized finance. O ecossistema DeFi visa aprimorar os serviços financeiros tradicionais por meio da negociação de tokens e da redução de intermediários e de barreiras para o crédito, assim como do acesso mais amplo a serviços financeiros no contexto da web descentralizada. Este capítulo traz uma discussão profunda sobre três aspectos essenciais e impulsionadores do ecossistema DeFi: (1) aplicações inovadoras, (2) interoperabilidade entre redes blockchain, e (3) segurança para as vulnerabilidades de ambientes descentralizados e competitivos. Nesse sentido, apresentamos os fundamentos

e exemplos práticos das aplicações DeFi mais populares atualmente. Por conseguinte, discutimos as pesquisas no estado da arte sob os três aspectos acima mencionados que suportarão o crescimento de DeFi nos próximos anos.

1.1. Introdução

Blockchain é uma tecnologia disruptiva, em especial para o setor produtivo, i.e., organizações nas áreas da agricultura, indústria e serviços, pois fornece recursos para o registro público, seguro e descentralizado de dados [Xu et al. 2019]. Essa tecnologia permite processar e armazenar dados de forma distribuída sem delegar o controle a uma autoridade central, e mesmo na existência de alguma parte não-confiável – o que é uma abordagem prática para o Problema dos Generais Bizantinos, já conhecido em ciência da computação desde a década de 80 [Lamport et al. 2019].

A blockchain foi concebida originalmente para garantir segurança às transações da plataforma de pagamentos baseada na criptomoeda Bitcoin [Nakamoto 2008]. Porém, o potencial dessa tecnologia é amplo e existe um crescente interesse por novas aplicações [Casino et al. 2019]. Em particular, as aplicações descentralizadas ou DApps, que atualmente funcionam no topo das redes blockchain, já ocupam uma posição relevante nas atividades econômicas e financeiras internacionais, e despertam o interesse de governos e empresas [Schär 2021].

DApps são programas de computador autônomos baseados em contratos inteligentes [Szabo 1997] e desenvolvidos em linguagens de alto nível, suportados desde a segunda geração de redes blockchain como a Ethereum [Wood 2014]. Um DApp pode ser constituído por um ou vários contratos inteligentes. Esses contratos, uma vez iniciados, executam automaticamente e de acordo com seu código registrado na blockchain.

Existem uma variedade de DApps que oferecem um intricado ecossistema de serviços, desde a área de finanças ao entretenimento digital. Tal ecossistema vem propiciando a emergência de uma nova geração da Internet baseada na infraestrutura de blockchain, popularmente intitulada web descentralizada [Murray et al. 2023]. Essa nova geração busca unificar as vantagens das raízes descentralizadas da primeira geração da Internet, suportadas por conteúdos públicos e protocolos abertos (e.g., TCP/IP e HTTP), com as funcionalidades da geração vigente (dita Web2), baseadas em plataformas centralizadas, a exemplo, serviços em nuvem e redes sociais providas por Amazon, Google, Facebook etc. A promessa da web descentralizada é que esses serviços tenham versões alternativas em DApps, favorecendo não apenas "*Big Techs*" mas todo um ecossistema de vários pequenos provedores de serviços tecnológicos na web.

A essência dos negócios em DApps se baseia no conceito de *token*, que é um objeto digital registrado na blockchain. Um *token* é único e está associado a um usuário, que é o seu proprietário, e somente este pode transferir a propriedade do *token* a outro usuário. Isso garante a possibilidade de valor ao *token*, i.e., a sua escassez ou impossibilidade de posses duplicadas (gasto duplo), diferentemente de objetos digitais tradicionais da web facilmente. Blockchain atua como a tecnologia base que permite a propriedade de *tokens* em redes de acesso público e confiável com segurança garantida por criptografia.

DeFi, do inglês *Decentralized Finance*, é um ecossistema de DApps que visa repli-

car, aprimorar ou substituir os serviços financeiros tradicionais por meio de negociação e transferência de *tokens*. A ideia de DeFi é eliminar intermediários e barreiras para o crédito no sistema bancário vigente, fornecendo acesso mais amplo a serviços financeiros no contexto da web descentralizada. Nesse sentido, DeFi oferece operações financeiras fundamentais (câmbio, saque e empréstimo com ou sem garantias) na forma de protocolos codificados em DApps de redes blockchain populares como Ethereum, Binance, Avalanche, dentre outras. Os protocolos permitem que usuários interajam diretamente com a blockchain, e permitem também que desenvolvedores e usuários combinem diferentes protocolos para criar soluções financeiras personalizadas e inovadoras.

Inicialmente, concebemos os *tokens* como criptomoedas, ou associados e lastreados a elementos de commodities de mercado (*e.g.* um *token* cuja unidade esteja associado a uma tonelada de minério de ferro). Mas, como no mundo real, um *token* pode ser único per si, ou seja, estar associado a algum objeto único, personalíssimo. Assim, o conceito de *token* foi estendido para a versão não fungível ou NFT, do inglês *Non-Fungible Token*, algo único.

Um *token* não fungível pode ser tipicamente associado a objetos multimídia, cujo conteúdo de texto ou imagem o confere características únicas e o torna também um objeto colecionável. NFT oferece recursos adicionais como a definição de um autor (criador) e recebimento de royalties do autor em transferências do *token*. Devido a esses recursos, NFTs vêm sendo adotados por artistas para criação e distribuição de conteúdo digital, visando a proteção do direito autoral e do ganho com royalties na revenda de itens. Por exemplo, em 11 de março de 2021, o artista Beeple realizou a venda de sua obra de arte digital em formato de NFT na blockchain Ethereum pelo valor de US$ 69 milhões. É importante observar que essas obras podem ser acessadas gratuitamente na Internet por se tratarem de um objeto digital. Contudo, quanto mais popular é o NFT, mais benefícios ele pode trazer ao seu proprietário, que em tese possui direitos exclusivos sobre a sua comercialização e imagem [Okonkwo 2021].

DApps no ecossistema DeFi com seus respectivos *tokens* tradicionais ou NFT são abertos, transparentes e acessíveis a qualquer pessoa com uma conexão à Internet, permitindo que indivíduos cadastrados em uma blockchain via um par de chaves pública e privadas vendam, emprestem ou troquem seus *tokens* de maneira descentralizada. Esse ecossistema cresce gradativamente desde o seu apogeu em 2014 com o surgimento da blockchain Ethereum, e abre novas oportunidades de inovações tecnológicas em aplicações, mas também enfrenta desafios, como a interoperabilidade entre as redes blockchain e a segurança dos contratos inteligentes.

Esse capítulo versa sobre os principais tópicos tecnológicos e de pesquisa envolvendo DeFi. Nesse sentido, iniciamos descrevendo as principais aplicações DeFi e seus fundamentos técnicos na Seção 1.2. A seguir, apresentamos o tópico interoperabilidade entre redes blockchain como uma consequência da evolução do ecossistema DeFi na Seção 1.3. As questões de segurança, vitais para a existência e credibilidade de DeFi são discutidas na Seção 1.4. Finalmente, apresentamos nossas considerações finais com um resumo do capítulo na Seção 1.5. Importante mencionar que materiais adicionais, atualizações desse capítulo, códigos fonte, artigos e resultados de pesquisa desenvolvidos pelos autores podem ser obtidos no repositório `https://github.com/LABPAAD/blockchain_defi`.

1.2. Aplicações

Nesta seção, chamamos a atenção para as aplicações DeFi mais proeminentes recentemente. Contudo, antes de discuti-las aprofundamos nos fundamentos que abrangem todo o ecossistema DeFi.

A base das aplicações de finanças descentralizada é a tomada de decisão sem a necessidade de uma terceira parte confiável. Ou seja, aplicações que possibilitem diferentes tipos de soluções financeiras (desde empréstimos, a câmbio, a hipotecas, a investimentos etc.), sem existir uma entidade financeira como terceira parte envolvida para prover o cumprimento das cláusulas contratuais esperadas. A ausência desta entidade requer que as partes envolvidas estabeleçam um protocolo adequado para a tomada de decisão quanto à execução de uma transação, e um substrato adequado para o registro do resultado desta execução, ambos de forma descentralizada.

O mecanismo de tomada de decisão é o do consenso distribuído, um dos problemas fundamentais da computação distribuída e bloco de construção de diversas soluções descentralizadas. E uma vez atingido o resultado deste consenso, este é persistido de forma distribuída em uma estrutura baseada em uma cadeia de blocos com uso de técnicas criptográficas como chaves pública/privada e sumários criptográficos (*hash* de modo a prover mecanismos de audaitabilidade, autenticidade, não-repúdio e integridade dos dados [Greve et al. 2018]).

Com tais características, esta estrutura provê um livro-razão que garante a integridade das transações neste armazenado, mas mantido de forma distribuída, sem exigência da terceira parte confiável.

Um amplo conjunto de possibilidades de aplicações financeiras descentralizadas, ditas aplicações DeFi, podem ser construídas por meio deste livro-razão distribuído subjacente provido pela blockchain para o registro de transações replicadas em uma rede de pares (p2p). Tomamos a blockchain como uma base para DeFi e encaminhamos o leitor para outros trabalhos existentes (notavelmente [Greve et al. 2018]) para uma exposição mais completa sobre a tecnologia blockchain. Assim, assumimos no presente texto que blockchain é a base para aplicações DeFi, e por conseguinte, estas herdam todas as suas propriedades de segurança, consistência, integridade e disponibilidade de registros. Sem essas propriedades de segurança, aplicações DeFi se tornariam inerentemente inseguras.

1.2.1. Contratos Inteligentes e Transações

Os contratos inteligentes [Szabo 1997] desempenham um papel fundamental no ecossistema DeFi: permitem a automatização e execução de acordos financeiros entre as partes, e.g., vendedor e comprador ou credor e devedor, sem a necessidade de intermediários. Em nosso contexto, os contratos inteligentes são programas de computador autônomos, e são registrados em uma blockchain, estabelecendo regras e condições para orientar as interações entre as partes envolvidas. Logo, a execução de contratos é determinada por lógica programada, o que pode resultar em redução de custos e maior previsibilidade no cumprimento dos acordos financeiros.

A rede blockchain Ethereum foi a primeira a permitir contratos inteligentes [Wood 2014]. O Bitcoin, considerado a primeira blockchain [Nakamoto 2008], permite apenas

operações de transferência de valores entre usuários, sem a possibilidade de desenvolver programas que sejam registrados na blockchain. A maioria das redes blockchain que surgiram posteriormente ao Ethereum já incluem contratos inteligentes, dado a inovação que esse recurso possibilitou na construção de diferentes aplicações, incluindo as de DeFi baseadas em operações financeiras decentralizadas.

```
1  // SPDX-License-Identifier: GPL-3.0
2  pragma solidity >=0.7.0 <0.9.0;
3
4  contract C{
5      bytes32 nome;
6      function get(bytes32 _nome) public{
7       nome = _nome;
8      }
9      function set() public view returns (bytes32) {
10      return nome;
11     }
12 }
```

Algoritmo 1.1. Exemplo de código fonte de contrato inteligente tomando com referência a linguagem Solidity Ethereum, adaptado de [Palma et al. 2022].

Mais especificamente, um contrato inteligente é um programa de computador escrito em uma linguagem de alto nível, como ilustrado no Algoritmo 1.1, adaptado de [Palma et al. 2022]. Note que, após ser desenvolvido e testado, um contrato necessita ser implantado, ou seja, registrado de forma imutável, na blockchain.

Nesta seção, tomamos como referência o desenvolvimento e uso de contratos inteligentes em uma blockchain baseada em Ethereum. O registro do contrato é feito pela compilação deste em bytecode e implantado na blockchain por meio da operação denominada *contract create* Após a execução bem-sucedida dessa operação, o contrato recebe um endereço exclusivo na blockchain, permitindo que qualquer usuário interaja com suas funcionalidades.

O contrato implantado na blockchain com suas respectivas aplicações clientes, que transmitem as requisições dos usuários, constituem o que é conhecido como aplicação descentralizada ou DApp. Para interagir com o DApp, é necessário submeter uma transação à blockchain, tendo como destino o endereço do contrato e o nome da função desejada, como ilustra a Figura 1.1 proposta por [Palma et al. 2022] para fins didáticos. Contudo, essa transação deve ser realizada por meio de uma aplicação cliente conectada à rede blockchain e equipada com as interfaces de comunicação adequadas para o contrato em questão.

Outra forma de interagir com um contrato implantado na blockchain é por meio de outro contrato. Nesse caso, várias requisições podem ser previamente programadas seguindo uma estratégia automatizada para realizar um objetivo específico, o que tipicamente é conhecido como um *bot*. É importante observar ainda que um contrato na blockchain possui um estado definido pelas variáveis programadas no contrato. Consequentemente, a execução de funções podem alterar o estado deste contrato.

Para que um contrato inteligente seja executado, um usuário deve interagir com ele

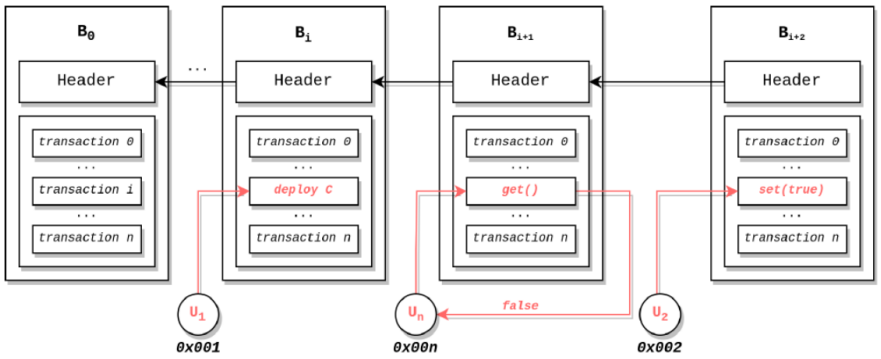


Figura 1.1. Exemplo de execução de uma transação que modifica um estado definido pelas variáveis programadas no contrato [Palma et al. 2022].

por meio de uma transação confirmada na blockchain. Após a confirmação da transação, o código do contrato é executado por um nó da rede e o estado é atualizado de acordo com as instruções do contrato. A tarifação das transações é determinada pelo consumo de "gás", uma unidade específica de custo computacional na blockchain. A Máquina Virtual Ethereum (EVM) usa um conjunto específico de instruções para execução de tarefas, cada uma consumindo uma quantidade específica de gás. O remetente de uma transação deve pagar o total de gás consumido por essa transação, calculado com base nas instruções executadas e no preço do gás naquele momento [Werner et al. 2022].

Execução de uma transação. Quando um participante da rede blockchain deseja fazer uma transação, os detalhes da transação não confirmada são primeiro transmitidos para uma rede de pares, validados e, em seguida, armazenados em uma área de espera (o *mempool* de um nó). Este grupo de transações é então propagado entre os nós da rede. Participantes do livro-razão subjacente responsável por garantir o consenso, os nós proponentes (que podem ser ditos mineradores ou forjadores, caso a rede seja, respectivamente, baseada em proof-of-work, PoW, ou proof-of stake, PoS) escolhem quais transações incluir em um determinado bloco, com base em parte na taxa de transação associada a cada transação. Transações em um bloco são executadas sequencialmente na ordem em que o minerador do respectivo bloco os incluía. Para um tratamento detalhado como esse processo funciona, encaminhamos o leitor para [Greve et al. 2018, Xu et al. 2019]. Nós proponentes têm a capacidade de controlar a sequência em que as transações são executadas. Consequentemente, os proponentes podem solicitar transações de formas que lhes renderão receitas e até mesmo inserir suas próprias transações para extrair mais receitas. Os mesmos podem ainda ser subornados para faça tal reordenação de transação, questões essas que serão discutidas na Seção 1.4.

1.2.2. Arquitetura DeFi

Aplicações DeFi, em blockchain, vem sendo foco de diversos estudos que abordam desde a taxonomia, à estruturação de conceitos e à arquitetura. A proposta mais proeminente e popularmente adotada até então é a arquitetura em camadas de [Schär 2021], que será a adotada neste capítulo. Essa arquitetura utiliza um modelo hierárquico em cinco camadas que têm propósitos distintos e complementares como mostra a Figura 1.2. Como usual nesse tipo de arquitetura, as camadas inferiores suportam as superiores em termos de

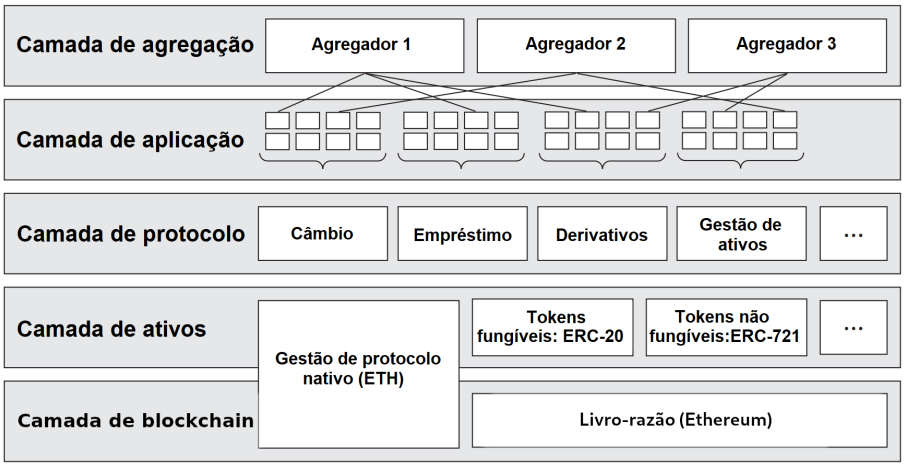


Figura 1.2. Arquitetura hierárquica de cinco camadas usualmente adotada para aplicações DeFi. Figura adaptada de [Schär 2021].

funcionalidades, o que permite às camadas mais altas ter objetivos específicos utilizando todos os recursos disponíveis na pilha de camadas ligeiramente inferior. É importante, contudo, observar que as camadas que aplicações DeFi são hierárquicas e seguras quanto às suas camadas inferiores. Ou seja, por exemplo, se a blockchain na camada mais inferior for comprometida, todas as camadas subsequentes não seriam seguras, e quaisquer esforços de descentralização nas camadas subsequentes seriam ineficazes.

1. Camada blockchain (Camada 1) consiste de um livro-razão para o registro de transações de modo imutável (*append only*) replicado entre vários computadores via redes de pares (e.g., Bitcoin ou Ethereum). Ela permite que a rede armazene informações com segurança e garante que quaisquer novos registros sejam anexados cumprindo um conjunto de regras acordadas entre os pares participantes da rede.

2. Camada de ativos (Camada 2) consiste de todos os ativos, i.e., *tokens*, que são emitidos sobre a camada de Blockchain. Isso inclui o ativo nativo da rede blockchain – e.g., Ether na rede Ethereum ou Matic na rede Polygon, bem como quaisquer outros ativos derivados destes que são emitidos sobre uma blockchain.

3. Camada de protocolos (Camada 3) provê padrões para casos de usos específicos, e.g., serviços para suportar trocas, empréstimos ou gerenciamento de ativos, que são registradas na blockchain. Esses padrões são implementados como um conjunto de contratos inteligentes que podem ser acessados por qualquer usuário (ou aplicativo DeFi).

4. Camada de aplicações (Camada 4) consiste em aplicativos direcionados aos usuários para se conectarem aos protocolos. Nessa camada, os contratos inteligentes geralmente são abstraídos por uma aplicação *front-end* baseado em navegador da Web ou até mesmo um software cliente de sistema operacional, tornando os protocolos fáceis de serem utilizados por usuários. É na camada de aplicação que se encontram as aplicações DeFi.

5. Camada de agregação (Camada 5) é uma extensão da camada de aplicação. Nessa camada os agregadores criam plataformas centradas no usuário que se conectam a vários aplicativos e protocolos. Os agregadores, geralmente, fornecem ferramentas para comparar e classificar protocolos permitindo a execução de tarefas complexas, conectando vários protocolos simultaneamente e combinando informações relevantes de maneira interativa. Geralmente, as aplicações na camada de agregação e aplicação funcionam da mesma maneira, através de uma interface Web ou aplicativo.

Visto o modelo conceitual ora apresentado para aplicações DeFi, vamos focar nas camadas de ativos e de protocolos. Logo, na seção seguinte serão discutidos os principais protocolos para gestão e operação com ativos na blockchain. Isso nos permite estabelecer a base necessária para os próximos conteúdos deste capítulo que tratam de interoperabilidade e segurança em DeFi.

1.2.3. *Tokens*

Negócios em DeFi se baseiam no conceito de *token*, que, de forma simplificada no contexto de redes blockchain, pode ser definido como um objeto digital registrado na blockchain. Um *token* está associado a um usuário, que é o seu proprietário, e somente este pode transferir a propriedade do *token* a outro usuário. Isso garante a possibilidade de valor ao *token*, i.e., a sua escassez ou impossibilidade de posses duplicadas (gasto duplo). Diferentemente de objetos digitais tradicionais da web, o *token* sempre tem um proprietário.

A ideia geral de *tokens* é tornar bens (i.e., ativos) tangíveis ou intangíveis mais acessíveis, assim como as transferências ou negociações desses ativos mais eficientes, assim um ativo so mundo real (dito *tokenizado*) pode ser representado na blockchain por um *token*, e uma transação que altere a propriedade deste *token* no livro-razão corresponderia a própria mudança de propriedade do ativo em si.

Dessa forma, ativos *tokenizados* podem ser transferidos facilmente em fração de segundos para qualquer pessoa ou organização no mundo. Por conseguinte, esses *tokens* podem ser usados por diferentes aplicações descentralizadas, desde que essas sejam constituídas pelos contratos ao qual os *tokens* estão vinculados. Importante mencionar que *tokens* estão associados a contratos inteligentes, por onde podem ser emitidos e transferidos aos respectivos proprietários.

Atualmente existem diferentes tipos de *tokens* onde cada um possui as suas características e a suas utilidades que serão discutidas a seguir.

1.2.3.1. *Token* Fungível

Token fungível foi o primeiro a ser desenvolvido na plataforma Ethereum, através de um padrão denominado ERC-20. Esse padrão tem como objetivo definir e controlar a emissão e distribuição de *tokens*. Assim, o padrão ERC-20 desempenha um papel crucial no ecossistema de *tokens* na Ethereum, simplificando a criação de *tokens* para desenvolvedores e possibilitando uma interação fluida entre usuários e uma diversidade de *tokens* em inúmeros aplicativos e serviços descentralizados.

O padrão ERC-20 estabelece um conjunto de funcionalidades características para um objeto digital ser considerado *token* fungível. São elas:

- **Nome e símbolo**: o *token* deve ter um nome descritivo e um símbolo associado a ele.
- **Decimais**: também deve ser especificado a quantidade de casas decimais que podem ser usadas ao exibir o saldo do *token*.
- ***Total supply***: registro do total de *tokens* em circulação emitidos e associados a um contrato padrão ERC-20.
- **Eventos**: existem dois eventos definidos por esse padrão. Um deles é acionado sempre que ocorre uma transferência de *tokens*, o que possibilita o registro e rastreamento das transferências (*Transfer*). O outro evento possibilita que o proprietário de *tokens* autorize outra conta a gastar uma quantidade específica de seus *tokens* em seu nome (*Approval*)
- ***Transfer***: permite que um proprietário possa enviar *tokens* para outros usuários da blockchain.
- **Emissão e queima**: permissão para emissão (*minting*) de novos *tokens* controlado pelos proprietários do contrato, ao passo que a queima (*burning*), i.e., destruição de *tokens* em circulação é controlada pelo proprietário do *token*.

Considerando esses conceitos e funcionalidades, *tokens* fungíveis são geralmente categorizados nos seguintes grupos:

- ***Tokens* de utilidade**: são criados com a finalidade de proporcionar acesso a serviços, produtos ou funcionalidades específicas dentro de um ecossistema blockchain. Eles permitem aos usuários pagarem por transações, votar em decisões de governança ou acessar serviços exclusivos oferecidos pela organização que emitiu os *tokens*.
- ***Tokens* de segurança**: representa ativos do mundo real, como ações, títulos, imóveis, obras de arte, fundos de investimento e outros ativos financeiros ou tangíveis. Jurisdições de alguns países como USA, Malásia, esses *tokens* podem ser considerados como título financeiro e estão sujeitos a regulamentações rigorosas, assim como títulos tradicionais emitidos por empresas. A principal característica de *token* de segurança é oferecer aos detentores direitos legais e econômicos sobre o ativo subjacente. Consequentemente, os seus possuidores adquirem privilégios tais como o recebimento de dividendos, participações em lucro, direito de voto em deliberações corporativas, entre outros, condicionados ao tipo de ativo que o *token* representa.
- ***Tokens* de governança**: representação digital que concede a seus detentores o privilégio de engajar-se nas deliberações de governança de uma organização autônoma descentralizada ou recursos e serviços especiais de uma plataforma blockchain. O propósito primordial desse *token* é capacitar a comunidade de usuários, assegurando que as determinações associadas ao desenvolvimento e operação de uma organização sejam efetuadas por meio de um processo descentralizado e democrático.

1.2.3.2. *Token* não Fungível

Os *tokens* não fungíveis, popularmente conhecidos como NFTs (*Non-Fungible Token*) é um tipo específico de objeto digital que representa a propriedade de um ativo único ou colecionável na blockchain. O que torna os NFTs únicos é o fato de serem não fungíveis, ou seja cada NFT é distinto e não pode ser trocado diretamente por outro de maneira idêntica em termos de valor, ao contrário das criptomoedas tradicionais, como o Bitcoin, que são fungíveis e podem ser trocadas umas pelas outras em proporções equivalentes.

O padrão de NFTs define um conjunto de regras e interfaces para a criação e interação com *tokens* digitais únicos e indivisíveis. O padrão mais comum para NFTs na blockchain Ethereum é o ERC-721 [Entriken et al. 2018], embora existam outros padrões e variações. As principais características definidas pelo padrão ERC-721 são:

- **Unicidade e Indivisibilidade**: Os *tokens* ERC-721 são únicos e indivisíveis, o que significa que cada *token* tem atributos e características exclusivas que o distinguem de outros *tokens*. Por exemplo, um NFT pode representar uma obra de arte digital específica, um item de jogo ou um bilhete de evento único. Em contraste, os *tokens* ERC-20 são fungíveis, o que significa que cada *token* é igual e pode ser trocado por outro *token* da mesma classe.
- **Transferências Individuais**: No padrão ERC-721, os *tokens* são transferidos individualmente, um de cada vez. Isso permite que cada NFT tenha sua própria história de propriedade e rastreabilidade na blockchain. Em contrapartida, os *tokens* ERC-20 podem ser transferidos em lotes, facilitando transações em massa de *tokens* fungíveis.
- **Métodos de Interface**: O padrão ERC-721 define uma série de métodos de interface padrão, incluindo *balanceOf* (para verificar o saldo de *tokens* de um proprietário), *ownerOf* (para verificar o proprietário de um *token*), *transferFrom* (para transferir a propriedade de um *token*) e outros. Esses métodos são adaptados para lidar com a singularidade e indivisibilidade dos *tokens* NFT.
- **Metadados e Informações Extras**: Os NFTs geralmente contêm metadados adicionais que descrevem o ativo representado pelo *token*. Esses metadados podem incluir informações sobre o autor da obra de arte, a data de criação, a descrição do item de jogo, entre outros detalhes relevantes. Os *tokens* ERC-721 permitem o armazenamento e a recuperação desses metadados de forma eficiente. Os *tokens* ERC-20 também podem armazenar metadados, mas geralmente se concentram em aspectos mais básicos, como nome, símbolo e número de casas decimais.

Cada NFT é único e possui suas próprias características. Um NFT pode ser, por exemplo, uma imagem, uma música ou até mesmo um item colecionável de um jogo (como uma roupa ou armamento). Um *token* não fungível também pode estar associado a um único objeto do mundo real, como um *token* de propriedade de uma obra de arte, ou mesmo um *token* correspondente a um ingresso em um dado lugar em uma apresentação única de uma peça de teatro. Os NFTs não podem ser divididos em partes menores pois são comercializados, por sua própria natureza, como unidades

únicas e indivisíveis. A titularidade de um NFT é registrada e verificada na blockchain, proporcionando autenticidade e singularidade ao ativo digital que ele simboliza. Esses *tokens* são comercializados a partir de plataformas de compra e venda ou até mesmo dentro do ecossistema o qual ele faz parte (dentro de um dApp como o Axie Infinity).

1.2.3.3. Stablecoin

Um ponto negativo nos *tokens* fungíveis é a sua alta volatilidade, o que cria obstáculos para usuários que buscam explorar os benefícios dos aplicativos DeFi, mas têm aversão ao risco associado a ativos voláteis, como o ETH. Como resposta a essa questão, surgiu uma categoria específica de criptomoedas conhecida como stablecoins. As stablecoins são projetadas para manter uma paridade de preço com um ativo específico, como o dólar americano ou o ouro.

Dentro de uma blockchain como a Ethereum, as stablecoins são definidas por padrões como os de *tokens* ERC-20. Esses *tokens* proporcionam a estabilidade necessária que os investidores procuram para participar de diversas aplicações DeFi, permitindo que uma solução nativa em *tokens* propicie posições em ativos digitais com menor volatilidade. Além disso, elas podem ser utilizadas para oferecer exposição *off-chain* aos retornos de ativos externos à blockchain subjacente, como, por exemplo, ouro, ações ou ETFs.

Os mecanismos pelos quais as stablecoins mantêm seu lastro podem variar dependendo da implementação [Goetze 2023] [Binance 2023]. No entanto, os quatro métodos principais são:

- **Stablecoins com garantia ou colateralizadas**: a ideia principal consiste em manter um fundo fiduciário na moeda de lastro. Dessa forma, se o valor da stablecoin sobe acima de uma unidade por moeda fiduciária então os arbitradores passaram a vender a stablecoin para trocar pela moeda fiduciária. Caso o valor da stablecoin caia abaixo de uma unidade então os arbitradores (usuários) passaram a comprar a stablecoin causando uma escassez dessa stablecoin, o que ocasiona uma alta no preço voltando novamente a relação de um para um. O mesmo vale para metais preciosos, petróleo e imóveis. Ainda que exista flutuações no preço essa ação faz com que a flutuação seja frações de centavos, mantendo o preço com uma flutuação de apenas algumas casas centesimais.
- **Stablecoins com garantia em criptomoeda ou cripto-colateralizadas**: pode ser considerado a segunda maior classe de stablecoins. Estas stablecoins são respaldadas por um fundo de outra criptomoeda. Seu valor pode ser ancorado de forma rígida ou flexível ao ativo de lastro, isso depende das diretrizes do protocolo. A stablecoin cripto-colateralizada mais popular é o DAI, criado pelo MakerDAO e é respaldado principalmente por ETH, com suporte colateral para alguns outros criptoativos. Ele está ancorado de forma flexível por meio de mecanismos econômicos que incentivam oferta e demanda para levar o preço a US$ 1. A capitalização de mercado do DAI é de pouco mais de US$ 5 bilhões.
- **Stablecoins sem garantia ou não colateralizadas**: este tipo de stablecoin não são apoiadas por um fundo de reserva. Este tipo de stablecoin utiliza um conjunto de

regras para controlar a oferta e demanda do *token* com o objetivo de manter seu valor próximo ao da moeda fiduciária desejada. Esse algoritmo tem a responsabilidade de ajustar automaticamente a oferta ou a demanda do *token* para trazê-lo de volta à paridade, caso o valor da stablecoin se desvia do valor desejado.

- **Stablecoins híbridas**: As stablecoins híbridas representam uma categoria de criptomoedas que incorpora características tanto das stablecoins colateralizadas quanto das não colateralizadas. Seu objetivo primordial é assegurar estabilidade de preço por meio de uma sinergia entre reservas de ativos tangíveis e algoritmos avançados. Esse modelo híbrido visa atingir um equilíbrio entre a estabilidade associada às stablecoins colateralizadas e a flexibilidade e descentralização características das não colateralizadas. À semelhança das stablecoins colateralizadas, as stablecoins híbridas podem manter reservas compostas por moedas fiduciárias, criptomoedas ou outros ativos digitais, que funcionam como garantia para sustentar o valor do *token*. Um exemplo de stablecoin híbrida é a Frax que mantém parte do seu funcionamento em garantia e parte do fornecimento algoritmo, onde a proporção em garantida e de algorítmicos depende do preço do atual do *token* em mercado.

De longe, a maior parte das stablecoins tem garantia fiduciária e normalmente estes são custodiados por uma entidade externa ou grupo de entidades que são submetidos a auditorias de rotina para verificar a existência das garantias. A maior stablecoin com garantia fiduciária é o Tether (USDT), com uma capitalização de mercado de mais de US$ 100 bilhões de dólares, tornando-se a terceira maior criptomoeda atrás do Bitcoin e do Ethereum. Tether também tem o maior volume de negociação que qualquer criptomoeda. O segundo maior é o USDC, apoiado pela Coinbase e pela Circle. USDT e USDC são muito populares para integração em protocolos DeFi, pois estão disponíveis para troca nas principais corretoras da rede Ethereum, como Uniswap e Jupiter, e a demanda por oportunidades de investimento em stablecoin é alta.

Um dos pontos negativos das stablecoins colateralizadas é a sua característica de que o fundo do ativo de lastro estar armazenado em uma entidade centralizada fora da blockchain. Neste ponto em questão, as stablecoins cripto-colateralizadas têm as vantagens da descentralização e garantia de colateral seguro. A desvantagem é que sua escalabilidade é limitada. Pois, para emitir mais unidades da stablecoin, um usuário deve necessariamente respaldar a emissão por meio de uma posição de dívida super colateralizada. Em alguns casos, como o DAI, há até mesmo um teto de dívida que limita ainda mais o crescimento do suprimento.

1.2.3.4. Central Bank Digital Currency

Uma moeda digital emitida por banco central, do inglês Central Bank Digital Currency (CBDC), é uma forma de moeda digital emitida pelo banco central de um país. É semelhante às criptomoedas, exceto que seu valor é fixado pelo banco central e, em regra, equivale à moeda fiduciária do país, tendo curso forçado, ou seja, é aceita pela economia por força de lei. A aceitação de stablecoins incentivou os bancos centrais a explorar os possíveis benefícios e custos da emissão de moedas digitais de banco central.

No campo das políticas públicas, há discussões intensas sobre os diferentes arranjos possíveis para as CBDCs [Allen et al. 2022]. Primeiramente, discute-se se o CBDC deve ser um instrumento de liquidação entre instituições financeiras (atacado) ou um sistema acessível a todos os consumidores (varejo), onde o CBDC seria um passivo do banco central. Outra possibilidade é, se adotada a CBDC no sistema de varejo, qual o papel do banco central: interagir diretamente com o público ou, delegar ao sistema financeiro a gestão de todas as atividades de atendimento ao cliente. Ainda, há intensos debates sobre as permissões na criação da CBDC, uma vez que diferentes características podem impactar a eficácia da política monetária e a estabilidade financeira.

Muitos países estão a desenvolver CBDCs, e alguns até já as implementaram. Por exemplo, a adoção do pagamento móvel e digital na China tem sido significativamente mais rápida do que na maioria dos outros países. Em 2019, os pagamentos via Alipay e WeChat ultrapassaram 500 milhões e 900 milhões de usuários ativos mensais, representando 36% e 65%, respectivamente, da população total da China [Frost et al. 2019]. Este ambiente de negócios foi favorável à criação de uma CBDC pelo banco central chinês, o Banco Popular da China (do inglês The People's Bank of China – POBC). No final de 2017, após aprovação do Conselho de Estado da China, o PBOC iniciou a colaboração com instituições comerciais para desenvolver e testar a moeda fiduciária digital, o e-CNY. O objetivo era estabelecer um sistema monetário respaldado pelo Estado, além de um simples sistema de pagamentos. Após alguns anos de trabalho, em abril de 2020, o PBOC anunciou testes em quatro cidades (Shenzhen, Suzhou, Xiong'an e Chengdu). Desde janeiro de 2022, a Tencent lançou serviços de e-CNY no WeChat, e diversas outras gigantes da Internet, como JD.com e Didi Taxi, também começaram a aceitar pagamentos em e-CNY nas cidades-piloto. Durante os Jogos Olímpicos de Inverno de 2022, a China testou com sucesso a aceitação do e-CNY, disponibilizando o aplicativo móvel e cartões de pagamento ou pulseiras do e-CNY para visitantes estrangeiros [Allen et al. 2022].

Um outro caso de estudo, ainda em desenvolvimento, é o DREX, CBDC brasileiro. O Brasil possui um dos sistemas bancários digitais mais avançados do mundo em seu ápice com o advento do PIX – mecanismo de pagamentos e transferência de valores que propiciou uma virtual universalização de movimentação de moeda de forma digital pela população brasileira. O Banco Central do Brasil tem sido transparente sobre motivações e processos no desenvolvimento do DREX, o que pode proporcionar uma oportunidade única de explorar o desenvolvimento e a implementação de um CBDC de uma perspectiva holística, considerando não apenas o design técnico, mas também as implicações políticas, econômicas e sociais [Sanches and Diniz 2024]. O seu projeto inclui preocupações quanto ao uso de tecnologia de contabilidade descentralizada, transações transfronteiriças eficientes e ênfase na inclusão financeira. A dinâmica da implementação de um CBDC em uma economia digital emergente como a do Brasil, deve ser observada com atenção, e pode potencializar o cenário de aplicações DeFi no contexto brasileiro.

1.2.4. Corretoras Descentralizadas

Entre janeiro e dezembro de 2023, somente na rede Ethereum, foram lançados mais de 370 mil *tokens* ERC-20. Com toda essa quantidade de *tokens* e um alto volume de capitalização de mercado torna-se indispensável o uso de plataformas onde os usuários interessados possam comprar, vender e trocar seus *tokens*. Essas plataformas permitem

que os proprietários desses ativos possam reequilibrar suas exposições de acordo com suas preferências e perfis de risco, ajustando as alocações de seus portfólios [Schär 2021].

Geralmente, as negociações de ativos descentralizados ocorrem através de corretoras centralizadas, pois em um primeiro momento é necessário trocar uma moeda fiduciária pelo ativo desejado e essa tarefa só é possível através de corretoras centralizadas ou de trocas diretas entre usuários. O grande problema das corretoras centralizadas é que os seus usuários estão expostos a: vazamento de dados pessoais, falência da corretora, ataques de terceiros e fraca regulação. É importante ressaltar que mesmo que os ativos sejam descentralizados e estejam de fato dentro da blockchain, na maioria dos casos, a posse desses ativos não está em uma conta que o usuário tem na blockchain e sim na posse da corretora. Ou seja, o único vínculo que o usuário tem com seus ativos é a sua conta padrão.

Com o objetivo de descentralizar e resolver os problemas associados às corretoras centralizadas, surgiram as corretoras descentralizadas, ou DEXes (*Decentralized Exchanges*). Os protocolos das DEXes são programados na blockchain por meio de contratos inteligentes, o que aumenta a confiabilidade entre as partes envolvidas na transação e mitiga os riscos de perda dos ativos. Dessa forma, os usuários mantêm controle total e exclusivo sobre seus ativos. Além disso, eles podem trocar seus ativos através de DApps fornecidos pela própria corretora ou utilizando contratos desenvolvidos por terceiros, já que as funções dos contratos da descentralizas podem ser acessadas por outros contratos dentro da rede.

As primeiras corretoras descentralizadas foram construídas de forma isolada e sem interação umas com as outras e entre seus protocolos. Inicialmente essas corretoras não possuíam uma liquidez compartilhada o que levava a um baixo volume de transações, grandes *spreads* entre compra e venda, altas taxas cobradas pela rede, processos complicados e lentos para mover fundos de uma uma corretoras para outra. Tudo isso tornava praticamente impossível as oportunidades de arbitragem entre os protocolos da rede.

Mais recentemente, houve uma movimentação em direção aos protocolos de corretoras abertas. Esses projetos visam simplificar a arquitetura das corretoras descentralizadas, estabelecendo padrões sobre como a troca de ativos pode ser conduzida. Eles permitem que qualquer corretoras, construída sobre esses protocolos, utilize *pools* de liquidez compartilhados e outros recursos integrados. O mais importante é que esses protocolos possibilitam que outros projetos DeFi utilizem esses *marketplaces* para trocar ou liquidar *tokens* conforme necessário. Nas próximas subseções apresentamos dois padrões populares de implementações para corretoras descentralizadas, e direcionamos os leitores interessados em outros padrões existentes para [Schär 2021].

1.2.4.1. Livro de Ordens Descentralizado

As corretoras descentralizadas com livro de ordens podem ser implementadas de várias maneiras. Todas utilizam contratos inteligentes para a liquidação das transações, mas diferem significativamente na forma como os livros de ordens são hospedados.

O livro de ordens é um registro digital que lista todas as ordens de compra e venda de um determinado ativo. Nas corretoras descentralizas que implementam esse método, o

livro de ordens é um registro onde os usuários podem cadastrar intenções de compra ou venda apresentando a quantidade de *tokens* e o valor da cotação que se deseja comprar ou vender. Quando duas ordens, de compra e venda, tem valores correspondentes, então a transação é realizada. Um comprador ou vendedor pode definir um valor de cotação, máximo ou mínimo, que deseja pagar, o que pode aumentar a chance de encontrar uma transação correspondente.

Nos protocolos das DEXes esse livro de ordens pode ser implementado dentro do protocolo, ou seja, dentro do contrato inteligente (*on-chain*), ou fora da blockchain (*off-chain*), através de uma aplicação de usuário ou de uma aplicação de terceiros.

Os livros de ordens *on-chain* têm a vantagem de serem totalmente descentralizados, pois as ordens são armazenadas dentro dos contratos inteligentes, o que elimina a necessidade de infraestrutura centralizada. No entanto, essa abordagem apresenta a desvantagem de que cada ação requer uma transação na blockchain, tornando o processo caro e lento. Mesmo a simples declaração de intenção de negociar resulta em taxas de rede. Em mercados voláteis, onde é comum o cancelamento frequente de ordens, essa desvantagem se torna ainda mais significativa.

Por essa razão, muitos protocolos de Descentralizado descentralizadas dependem de livros de ordens *off-chain*, utilizando a blockchain apenas como camada de liquidação. Os livros de ordens *off-chain* são hospedados e atualizados por terceiros centralizados. Eles fornecem aos compradores a informação necessária para selecionar uma ordem que desejam corresponder.

Um dos maiores protocolos que implementam essa abordagem é a dYdX [Juliano 2018]. A dYdX é uma DEX de negociação de derivativos que também oferece negociação de margem e empréstimos. Atualmente ela está entre as maiores corretoras descentralizadas da atualidade. Inicialmente a dYdX utilizava um sistema de livro de ordens *off-chain*, o qual era executado por eles. Porém, em 2023 a dYdX fez uma grande atualização descentralizando totalmente os seus serviços e passando a utilizar um livro de ordem *on-chain* na rede Cosmos [Product 2024].

1.2.4.2. Criadores de Mercado Automatizados

Criadores de Mercado Automatizados (AMM – *Automated Market Makers*) é um modelo de protocolo utilizado por DEXes que tem por objetivo definir a precificação de ativos digitais. O protocolo AMM se baseia em uma fórmula matemática para determinar os preços dos ativos. Alguns AMMs, inclusive, usam fórmulas simples, como é o caso do Uniswap. Provedores de liquidez são usuários que depositam ativos em *pools* de liquidez, que são espaços controlados por contratos inteligentes, e em troca recebem recompensas [Qin et al. 2021].

Qualquer ordem única de compra ou venda pode ser executada independentemente de outras negociações em AMM DEXes. Por exemplo, quando os comerciantes desejam trocar a criptomoeda A por B, eles podem invocar a função do contrato inteligente que transfere A da conta dos comerciantes para o *pool* de liquidez e envia B do *pool* de liquidez para a conta dos comerciantes. O processo de troca não envolve a participação de quaisquer

outros comerciantes. A taxa de câmbio entre A e B é determinada por funções pré definidas de forma transparente codificadas no contrato inteligente da AMM DEX [Wang et al. 2022].

Como as operações de mercado em AMM DEXes são invocadas por transações em blockchain, os usuários são obrigados a pagar uma taxa de transação aos mineradores. Especificamente, no Ethereum cada transação custa uma quantidade predeterminada de "gás", taxa de efetivação de uma transação na rede. O emissor da transação específica quanto está disposto a pagar por unidade de gás (ou seja, o preço do gás). A taxa de transação paga aos mineradores corresponde ao produto do consumo total de gás e do preço do gás [Wang et al. 2022].

A Figura 1.3 apresenta um modelo de produto constante que é aplicado em *pools* de liquidez. Este modelo pode ser expresso como $xy = k$, onde x e y correspondem as reservas dos *tokens* armazenados dentro da *pool*, e k uma constante. Porém, quando um usuário da rede realiza um troca de *tokens* utilizando esta *pool* os valores de x e y serão modificados, então obteremos $(x+\Delta x).(y+\Delta y) = k$. Dessa forma, podemos assumir que $\Delta y = (k/(x+\Delta x)) - y$. Ou seja, se um usuário realiza um *swap* de *tokens* x para y nesta *pool* serão depositados no contrato os *tokens* x e os *tokens* y equivalentes serão enviados para o usuário. Isso faz com que Δy assuma valores negativos pois alguns *tokens* y foram retirados do *pool*, e Δx assume valores positivos [Schär 2021].

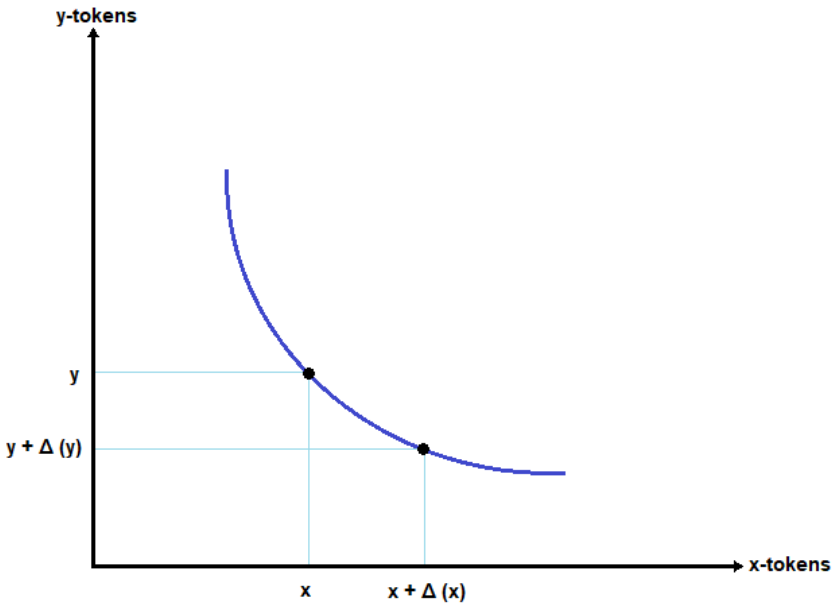


Figura 1.3. *Pool* de liquidez em um modelo de produto constante. Figura adaptada de [Schär 2021].

Toda troca realizada pelo *pool* resultará em um movimento em uma curva convexa, como mostra a Figura 1.3. Um *pool* de liquidez de produto constante modifica os valores dos *tokens* com base no movimento da curva fazendo com que o valor do *token* aumente infinitamente conforme a quantidade desse token se aproxima de zero, e, desta forma, que o *pool* jamais seja esgotado.

É importante ressaltar que os valores dos *tokens* são dados, somente, com base na função aplicada aquela *pool*. Isso significa que o valor de um *token* dentro de uma *pool* pode ser diferente do seu valor de mercado. Apesar dos valores dos *tokens* serem calculados de forma exponencial, dentro e fora do *pool* eles estarão sempre próximos. Quando o valor de mercado do *token* muda, os arbitradores poderão aproveitar a oportunidade para trocar os *tokens* com o contrato inteligente do *pool* até que o valor praticado dentro do *pool* de liquidez esteja igualado com o valor de mercado. Este processo garante que o valor do *token* dentro do *pool* esteja sempre próximo ou igualado com o valor de mercado.

1.2.5. Perspectivas em Aplicações

Nesta seção discutimos brevemente tópicos de pesquisa e inovação promissores em aplicações para finanças descentralizadas, reservando a discussão sobre interoperabilidade e segurança para seções seguintes.

Tokens não fungíveis (NFTs) tem sido alvo de pesquisas científicas, em especial, após o pico de comercialização superior a USD 2 bilhões desses tokens em 2021 e sua adoção pela indústria de mídia e artes digitais desde então. Um trabalho seminal conduzido por [Nadini et al. 2021] coletou dados de 4,7 milhões de NFTs e construiu a rede de interações entre usuários de NFTs visando identificar comunidades. Esse trabalho abriu caminho para pesquisas sobre aquisição de conhecimento aplicado ao mercado de NFTs via análises de grafos e aprendizagem de máquina. Exemplos proeminentes recentes são métricas de redes complexas para extrair comunidades de compradores e vendedores por categorias de NFT [White et al. 2022], e modelos de predição de preços baseado em processamento de imagem e texto com redes neurais profundas em [Costa et al. 2023].

Outro tópico de pesquisa que demanda contribuições da comunidade atualmente concerne a governança de DApps em especial DeFi. Em [Messias et al. 2023], a governança descentralizada é analisada via medições nos DApps Compound e Uniswap, que são duas DeFis populares em propor protocolos de votação para adoção de atualizações em seus códigos. Os autores revelam uma elevada concentração do poder de voto nesses DApps. Exemplos de esforços de pesquisa que fundamentam trabalhos nesse tópico incluindo contratos sociais e organizações autônomas descentralizadas (DAOs) são as análises sobre compromissos entre descentralização e desempenho proposto em [Chen et al. 2021], o estudo abrangente da teoria de governança e a reconceitualização do termo governança adaptada aos DAOs [Zwitter and Hazenberg 2020], e a taxonomia sobre o que constituem os DAOs e suas principais características proposta em [Hassan and De Filippi 2021]

Observando o cenário brasileiro, o desenvolvimento e implementação do DREX representam uma oportunidade significativa para o Brasil modernizar seu sistema financeiro, aumentar a inclusão financeira e digital, e fortalecer sua soberania econômica. No entanto, é fundamental abordar cuidadosamente os desafios técnicos para garantir que os benefícios do DREX sejam plenamente realizados e que os riscos sejam mitigados. A sua implementação pode impulsionar o cenário de finanças descentralizadas (DeFi) no Brasil, oferecendo novas oportunidades para inovação financeira, como empréstimos peer-to-peer, investimentos descentralizados e seguros automatizados. As competências desenvolvidas neste ecossistema podem ser transpostas ao cenário mundial, considerando a experiência digital avançada do sistema financeiro brasileiro, e adaptando-se às peculiaridades que

serão observadas em cada caso de implementação de novas CBDCs de acordo com as diferentes estratégias de cada Banco Central [Teixeira 2023].

1.3. Interoperabilidade

As redes blockchain permitem a execução de diferentes conjuntos de transações por meio de implementações distintas e normalmente isoladas entre si. Esta configuração acaba gerando sistemas heterogêneos, onde há grande dificuldade para troca de informações entre as redes, em especial aplicações DeFi e seus respectivos *tokens* que não podem funcionar em redes blockchain diferentes das quais foram inicialmente desenvolvidas e implantadas. Essas questões trouxeram em evidência a necessidade de interoperabilidade entre redes. Assim, vem se observando recentemente um esforço de várias organizações mantenedoras dessas redes em desenvolver soluções que ampliem as suas capacidades de cooperação, ainda que essas redes utilizem diferentes tecnologias, o que consiste na definição ampla do termo interoperabilidade [Wegner 1996].

Os tipos de interoperabilidade em blockchain mais comuns são: interoperabilidade entre redes blockchain (homogêneas), interoperabilidade entre dApps usando diferentes redes blockchain e interoperabilidade de redes blockchain com outras tecnologias de blockchain (heterogêneas) [Besançon et al. 2019]. Assim, a Figura 1.4 demonstra um diagrama de fragmentação desses tipos e apresenta como são definidas as transações entre os tipos. As transações de redes blockchain (homogêneas), são nomeadas como uma transação *cross-chain* (CC-Tx), onde "CC" significa *cross-chain* e "Tx" transação. Uma transação *cross-blockchain* (CB-Tx) é uma transação entre diferentes redes blockchain (heterogêneas) e, por fim, uma aplicação descentralizada *cross-chain* (CC-dApp) é um dApp que utiliza as transações *cross-blockchain* para implementar seus requisitos [Belchior et al. 2021].

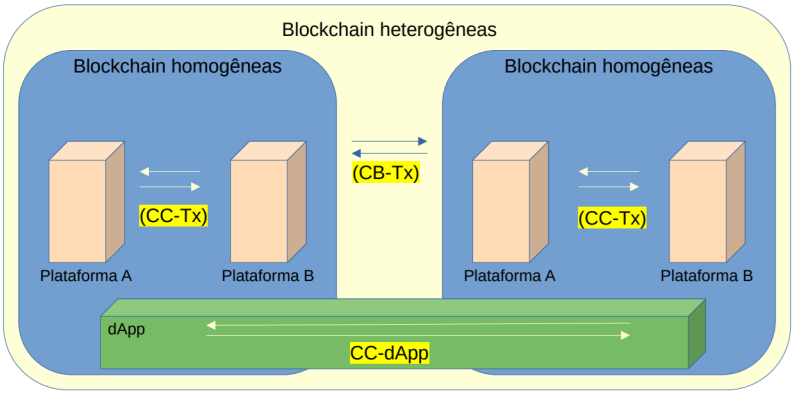


Figura 1.4. Tipos de interoperabilidade e transações. [Mendonça et al. 2024]

A transação *cross-chain* é o tipo de interoperabilidade mais utilizado atualmente. Ela envolve um par de redes blockchain em que um tipo de aplicação descentralizada facilita a transferência de ativos de uma blockchain para outra, provendo a interoperabilidade. Isso significa que é possível mover ativos, como criptomoedas e *tokens*, de uma blockchain para outra sem a necessidade de intermediários centralizados. Uma das maneiras de implementar *cross-chain* envolve a criação de pontos de conexão entre as redes blockchain,

conhecidos como “pontes". As pontes permitem que as transações sejam validadas em ambas as redes blockchain, garantindo a segurança e a integridade dos ativos transferidos. De maneira geral, as transferências de ativos entre cadeias seguem um procedimento atômico, baseado no protocolo *Cross-chain communication protocol (CCCP)* em que há o bloqueio de um ativo na origem, responsabilidade de transferência e a criação do ativo no destino [Belchior et al. 2021].

No contexto dos *tokens*, a interoperabilidade entre plataformas pode contribuir para a usabilidade, ao implicar a capacidade de transferir um ativo entre cadeias distintas, mantendo o estado e histórico consistentes. A interoperabilidade de cadeias deve atingir a eficiência de dois tipos, cada um dos quais trazendo considerações distintas porém contribuindo para a usabilidade. A troca de ativos digitais entre cadeias é um dos tipos de interoperabilidade. Ele deveria conter a capacidade de transferir e trocar ativos originários de diferentes cadeias sem intermediários confiáveis, como trocas centralizadas. Um exemplo disso seria tornar um *token*, originário de uma cadeia, válido em qualquer outra cadeia disponível. Outro tipo de interoperabilidade desejada se diz respeito à troca de informações que mantém a capacidade de fazer algo em uma cadeia que reflete em outra cadeia. Esta troca deve permitir o rastreamento não só de ativos ou itens negociáveis, mas também as operações executadas. Como exemplo, o compartilhamento do histórico de transações de um determinado item contendo negociações e proprietários.

1.3.1. Mecanismos de Interoperabilidade

Com tantas oportunidades de aplicativos de negócios com requisitos em interoperar redes blockchain, prover soluções com mecanismos genéricos de *cross-chain* para conectar redes blockchain homogêneas e heterogêneas, amplia o espaço de desenvolvimento desta tecnologia [Buterin 2016]. Algumas soluções foram propostas para interoperar acesso a dados e transações entre redes blockchain, como as *sidechains*, o mecanismo notarial (*Notary mechanism*) e bloqueio de hash (*Hash-locking* ou *Hash time lock*) [Belchior et al. 2021]. Estes mecanismos propõem soluções que podem abranger um número maior de variedades de aplicações e distintas soluções de blockchain.

1.3.1.1. Sidechain

Uma *sidechain* é uma blockchain independente que opera em paralelo à blockchain principal (ou *mainchain*) e seu conceito foi baseado para permitir a transferência segura de ativos e informações entre ambas. Esse mecanismo de funcionamento é essencial para a interoperabilidade de diferentes redes blockchain, pois amplia a funcionalidade da *mainchain* sem sobrecarregar as limitações de escalabilidade da rede principal, transferindo certas operações e transações para a cadeia secundária.

O conceito de *sidechain* [Back et al. 2014] foi introduzido como uma solução para diversos problemas de escalabilidade, flexibilidade e experimentação de uma *sidechain*, é possível implementar novas funcionalidades e realizar atualizações sem a necessidade de divisões na blockchain principal, o que contribui para sua estabilidade e segurança.

A figura 1.5 ilustra o conceito de *sidechain* com um exemplo de uma blockchain hipotética. A *mainchain* é a cadeia de blocos original onde as transações são registradas

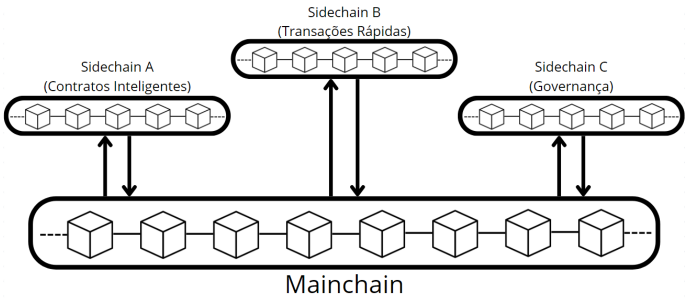


Figura 1.5. Exemplo de sidechains

e a integridade e segurança dos dados são asseguradas. As setas indicam que há uma relação entre a blockchain principal e *sidechains*, demonstrando que os ativos podem ser transferidos entre elas. *Sidechains* são cadeias de blocos adicionais que estão conectadas à blockchain principal. Cada *sidechain* pode ter características e funcionalidades específicas que não estão presentes na cadeia principal. No exemplo ilustrado, a *sidechain* A é projetada para suportar contratos inteligentes. A *sidechain* B é otimizada para transações rápidas, ou seja, podendo lidar com um volume maior de transações por segundo do que a cadeia principal. Por fim, a *sidechain* C foca na governança, ou seja, em como são tomadas decisões sobre a evolução e as regras do sistema blockchain.

Uma das principais características das *sidechains* é a capacidade de transferir ativos de forma segura entre a *mainchain* e a *sidechain*. Isso geralmente é feito através de um mecanismo conhecido como *two-way peg* (ponte bidirecional), que permite a transferência de *tokens* entre as duas cadeias [Singh et al. 2020]. Inicialmente, um usuário envia uma quantidade de *tokens* para um endereço específico (chamado de *lock-box*) na *mainchain*, bloqueando os *tokens* nessa cadeia. Em seguida, a *sidechain* recebe a transação de bloqueio que foi confirmada na *mainchain* e emite réplicas dos *tokens* originais. O usuário pode então utilizar esses *tokens* representados para pagar por serviços ou realizar transferências na *sidechain*. Eventualmente, o usuário pode retirar seus *tokens* da *sidechain* para a *mainchain*, exigindo que os *tokens* representados sejam bloqueados ou queimados na *sidechain*, dependendo da implementação.

Atualmente, existem três principais mecanismos para implementar um *two-way peg*: *two-way peg* centralizado, *two-way peg* federado e verificação simplificada de pagamento (SPV) [Ren et al. 2023]. No *two-way peg* centralizado [Singh et al. 2020], a implementação é realizada por uma terceira parte confiável que fica responsável por garantir o bloqueio e desbloqueio de *tokens* tanto na *mainchain* quanto na *sidechain*. Embora seja fácil de implementar, este esquema centralizado contraria a característica de descentralização de redes blockchain públicas e pode tornar um ponto único de falha.

O *two-way peg* federado [Singh et al. 2020] melhora o método anterior ao adicionar um grupo de partes ditas notários (do inglês *notaries*) para processar as operações de bloqueio e desbloqueio de *tokens*, utilizando esquemas de multi assinaturas. Desse modo, a operação de bloquear ou desbloquear *tokens* da *mainchain* só ocorre se a maioria das partes do grupo concordar com a transação. Embora essa abordagem possa diminuir a centralização, ainda pode apresentar riscos se a maioria dos participantes tiver intenções

maliciosas ou o grupo de assinantes for pequeno.

A verificação simplificada de pagamento (SPV) é outro mecanismo utilizado para implementar a interoperabilidade entre *sidechains* e a *mainchain*. Inicialmente descrito por Satoshi Nakamoto em [Nakamoto and Bitcoin 2008], esse mecanismo permite que nós verifiquem transações sem precisar baixar toda a blockchain. Em um contexto de *sidechains*, a SPV permite que a *sidechain* verifique transações na *mainchain* de forma eficiente, sem a necessidade de sincronizar todos os dados da *mainchain*.

Sidechains operam com seus próprios mecanismos de consenso, regras e sistemas de governança, ou seja, são responsáveis pela sua segurança e não herdam tais propriedades da *mainchain*. Isso possibilita o desenvolvimento e a experimentação de novos protocolos e aplicações descentralizadas sem interferir na operação da blockchain principal. Uma *sidechain* pode ser configurada para suportar contratos inteligentes mais complexos, transações mais rápidas ou diferentes algoritmos de consenso. Por exemplo, a RSK [Lerner et al. 2022] provê uma *sidechain* com suporte a contratos inteligentes para interoperar com a *mainchain* da Bitcoin.

Em resumo, as *sidechains* proporcionam uma maneira de escalar redes, experimentar novas funcionalidades e melhorar a interoperabilidade entre diferentes redes blockchain. Ao permitir a coexistência de múltiplas cadeias especializadas, as *sidechains* ampliam o potencial das aplicações em blockchain, tornando-as mais versáteis e adaptáveis às necessidades específicas dos usuários e desenvolvedores.

1.3.1.2. Mecanismo de Bloqueio de *Hash*

O mecanismo de bloqueio de *hash* ou *hash time-lock contract* (HTLC) representa um marco significativo na evolução dos mecanismos de troca entre redes blockchain, proporcionando uma solução inovadora para realizar transações entre redes sem depender de intermediários confiáveis [Ou et al. 2022]. Ao implementar contratos HTLC nas redes blockchain envolvidas na negociação, o processo de troca de ativos é seguro e confiável. Esse contrato atua como uma garantia, bloqueando os ativos envolvidos até que as condições acordadas sejam atendidas. A utilização do conceito de *hash* adiciona uma camada adicional de segurança, criando uma trava com uma palavra secreta que deve ser correspondente em ambas as extremidades da transação. Além disso, a imposição de um limite de tempo para a conclusão da troca aumenta a eficiência e a segurança do processo. Em caso de não cumprimento dentro do prazo estipulado, o contrato automaticamente cancela a transação, revertendo os *tokens* para suas respectivas carteiras de origem. Esse mecanismo desempenha um papel fundamental na facilitação de trocas descentralizadas, promovendo a confiança e a segurança nas transações entre redes blockchain [Belchior et al. 2021].

A arquitetura do mecanismo de bloqueio de *hash* exige a implementação de contratos inteligentes. Neste caso, o contrato é responsável pela troca segura dos ativos, ou seja, ele é implantado nas duas redes e possui a tarefa de conectá-las. Contudo, não há um terceiro confiável. O contrato inteligente atua sincronizando as redes no que diz respeito à verificação das transações, da palavra secreta e a devolução dos valores, caso necessário. Sendo assim, o contrato HTLC possui as funcionalidades de bloquear os fundos que serão transferidos, registrar o horário da transação e exigir uma palavra secreta no momento

da retirada dos fundos para o destinatário da segunda rede. Caso o tempo tenha ultrapassado o período de bloqueio, o contrato é capaz de reverter todo o processo, conforme a definição do mecanismo. Apesar disso, questões referentes à segurança do mecanismo também devem ser consideradas. O comprometimento do contrato inteligente que realiza a sincronização pode levar ao comportamento desordenado das chamadas de retirada e bloqueio dos fundos, causando possíveis perdas ou resultados inesperados.

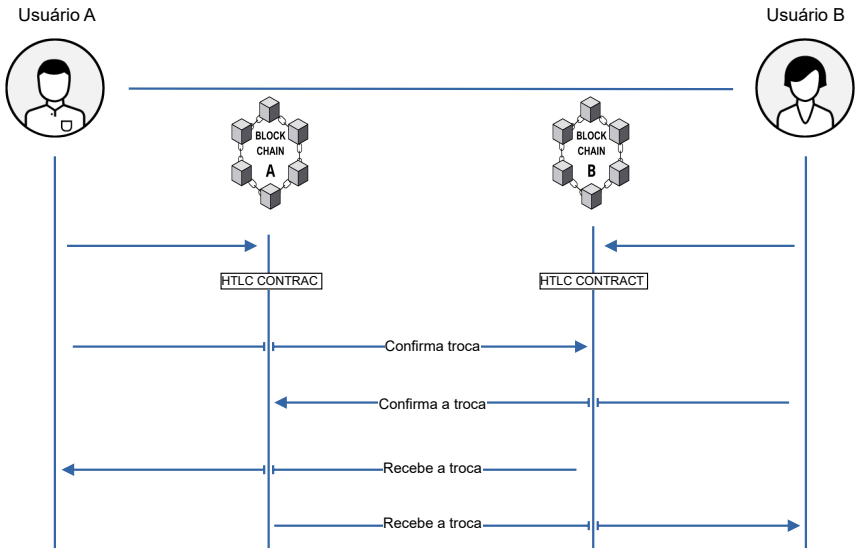


Figura 1.6. Arquitetura implementada do Mecanismo *Hash-Time Lock*.

No contexto do mecanismo de bloqueio de *hash*, conforme a Figura 1.6, o *Usuário A* deseja fazer a transferir seu *token* para uma conta do *Usuário B* em outra rede. Para fazer essa transferência, ele escolhe uma "palavra secreta", e utiliza o *Hash* juntamente com o endereço de B para criar o contrato HTLC. Por meio do contrato criado na *Blockchain A*, ele bloqueia o *Token A* para a transferência ser realizada. De maneira similar, o *Usuário B* implanta o HTLC na *Blockchain B* com o endereço de A e a palavra secreta em *Hash*. Sendo assim, o *Usuário A* faz a retirada (*withdraw*) do *Token B* na *Blockchain B* com sua palavra secreta. Ao fazer isso, a palavra secreta pode ser usada pelo *Usuário B* para retirar o *Token A* da *Blockchain A*.

1.3.1.3. Mecanismo Notarial

O mecanismo notarial é uma forma de implementar a interoperabilidade entre cadeias. Ele consiste em verificar e encaminhar mensagens entre cadeias por meio de uma entidade confiável intermediária chamada de notário. Quando há troca e transferência de ativos entre diferentes redes blockchain, uma ou mais organizações são designadas como notários para monitorar eventos entre as cadeias, e alcançar um consenso sobre a ocorrência do evento por meio de um algoritmo de consenso específico, e, por fim, responder de forma tempestiva [Belchior et al. 2021]. O mecanismo notarial se divide em mecanismo notarial de assinatura única e de múltiplas assinaturas [Ou et al. 2022].

O mecanismo notarial de assinatura única, também denominado mecanismo notarial centralizado, consiste em designar um único nó ou instituição independente para atuar como notário, e o notário assume as tarefas de coleta de dados, verificação e confirmação de transações no processo de interação entre cadeias. O notário é composto por, pelo menos, uma conta nas cadeias de origem e de destino. Este mecanismo consegue ter um processamento rápido de transações e é bastante adaptável, apesar do escopo restrito, limitando-se a troca de ativos.

No mecanismo notarial de múltiplas assinaturas o notário é geralmente composto por vários nós, onde cada nó possui uma chave e somente quando uma determinada porcentagem destes nós assinam em conjunto é que há um consenso e as transações entre cadeias podem ser confirmadas. Durante a verificação da transação, uma parte dos notários é selecionada aleatoriamente do grupo notarial, diminuindo o grau de dependência da confiabilidade dos notários.

Na arquitetura deste mecanismo, os usuários envolvidos na transferência de *tokens* devem interagir com o notário. Essa interação pode ocorrer por meio de *dApps* (aplicativos descentralizados executados em blockchain) ou contratos inteligentes, com o domínio de um terceiro confiável. O notário desempenha o papel de receptor do *token* do usuário A (remetente) na *blockchain* A, transferindo-o para o usuário B (destinatário) na *blockchain* B e registrando informações sobre as transações realizadas. O notário deve garantir a entrega segura dos recursos ao destinatário designado.

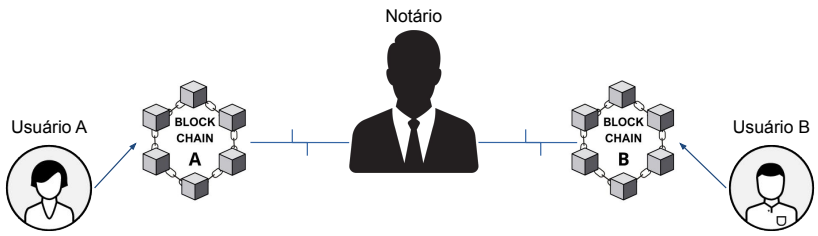


Figura 1.7. Arquitetura implementada para o Mecanismo Notarial.

Vale ressaltar que, embora o mecanismo seja eficaz, ele ainda possui limitações. Em particular a segurança, que é de suma importância em qualquer aplicação blockchain. No caso desta arquitetura em específico, o mecanismo notarial atua de maneira centralizada, sendo assim a segurança da transação depende da integridade do notário, visto que ele é o responsável por receber os fundos na blockchain de origem e transferi-los para a blockchain de destino. Se o notário for comprometido de alguma forma, isso pode acarretar em perdas financeiras para os usuários das redes. Porém, uma vez que o notário seja exaustivamente testado e reconhecido como confiável o mecanismo se torna extremamente eficiente.

Atualmente, algumas organizações utilizam mecanismos notariais, como o proto-

colo CCIP da Chainlink, para garantir a interoperabilidade entre redes blockchain. No entanto, diversos projetos que inicialmente adotaram essa abordagem agora estão explorando alternativas. Por exemplo, o Interledger, criado pela Ripple, foi projetado para facilitar pagamentos entre diferentes redes blockchain utilizando o Protocolo Interledger (ILP), um conjunto aberto de protocolos que permite transações de forma atômica e universal [Thomas and Schwartz 2015].

Originalmente, o ILPv1 visava proporcionar uma maneira flexível de realizar pagamentos entre diversas redes blockchain. Ele implementava transações atômicas através de dois modos: *hash-locks* e quórum de notários, conhecido como Protocolo de Transporte Atômico. Os *hash-locks*, baseados em contratos HTLC, são bloqueios criptográficos que podem ser desbloqueados ao revelar um segredo *s* cujo resultado da função de *hash H(s)* corresponde ao valor configurado no bloqueio [Siris et al. 2019]. No modo atômico, além dos *hash-locks*, as transações são coordenadas por meio de um grupo AD-HOC de notários selecionados pelos participantes para verificá-las e validá-las.

Por outro lado, o modo universal do ILP permitia transações entre conectores não confiáveis, dispensando notários. Enquanto o modo atômico utiliza notários para garantir a execução adequada de um pagamento, o modo universal depende dos incentivos de participantes racionais para eliminar a necessidade de coordenação externa. Este modo fornece segurança para todos os participantes não defeituosos conectados, sob a suposição de sincronia limitada com um limite conhecido. Em vez de notários, utilizava o XRP, a moeda nativa do Ripple, para facilitar essas transações [Thomas and Schwartz 2015].

Com a evolução para o ILPv4, houve uma mudança significativa na abordagem do protocolo devido à preocupações de segurança associadas aos longos tempos de espera nas transações de garantia. O protocolo ILPv1 sofria com deficiências, como o problema da opção gratuita, onde remetentes e destinatários poderiam manipular as taxas de câmbio, e ataques de negação de serviço que vinculavam fundos dos conectores intermediários. Para mitigar esses riscos, o ILPv4 foi redesenhado para usar transferências rápidas de pacotes de baixo valor [Siris et al. 2019]. Esse novo modelo elimina a necessidade de notários e compromissos de alto valor, permitindo que os conectores estabeleçam relações de confiança bilaterais, de forma que as as *sidechains* podem ser usadas como um sistema de liquidação entre duas entidades em interação.

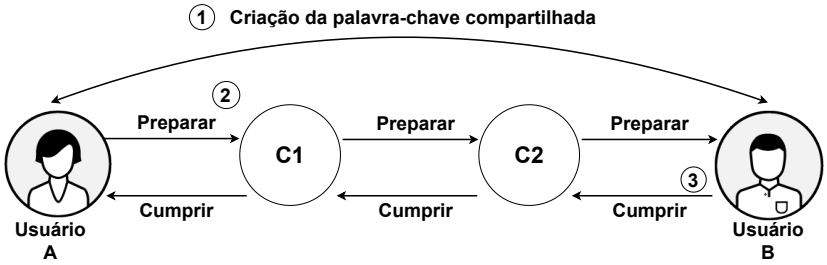


Figura 1.8. Representação de pagamento utilizando o ILPv4.

Um pagamento ILPv4 entre o usuário A (remetente) e o usuário B (destinatário) usando dois conectores prossegue conforme visualizado na Figura 1.8, essa transferência de valor ocorre em duas fases: preparar e cumprir. Inicialmente, o remetente gera um pacote de preparação contendo o valor e o *hash* de uma palavra-chave conhecida apenas pelo remetente e pelo destinatário. Este pacote é encaminhado ao receptor através de conectores. Cada conector, ao encaminhar o pacote, compromete-se a pagar ao próximo conector somente se este fornecer prova de pagamento ao conector subsequente. Quando o receptor recebe o pacote de preparação, ele gera um pacote de cumprimento revelando a palavra-chave e o envia de volta pelo mesmo caminho, começando pelo último conector. Cada conector valida a palavra-chave, efetua o pagamento e encaminha o pacote de cumprimento ao conector anterior. Esse processo permite que os pacotes de preparação e cumprimento se propaguem rapidamente, evitando os atrasos significativos associados aos métodos de pagamento baseados em garantia do ILPv1 [Siris et al. 2019].

1.3.2. Soluções para Interoperabilidade

Diversas soluções têm sido implementadas para superar as barreiras da fragmentação, promovendo um ecossistema mais integrado e funcional no que diz respeito à interoperabilidade de redes blockchain. Estas soluções variam desde protocolos de comunicação padronizados até plataformas de troca e *bridges* que facilitam a interoperabilidade entre diferentes redes blockchain. Nesta subseção, examinaremos duas das principais abordagens implementadas para alcançar a interoperabilidade em blockchain: Cosmos e Chainlink.

1.3.2.1. Cosmos

O Cosmos pode ser definido como a "internet das redes blockchain". Essa estrutura descentralizada concentra-se principalmente na interoperabilidade e escalabilidade, oferecendo maior flexibilidade tanto para desenvolvedores quanto para usuários. O Cosmos Hub é a principal blockchain da rede, ele atua como ponto central de coordenação e segurança. No Cosmos Hub, todas as atividades e transações são registradas, e a criptomoeda nativa, ATOM, é hospedada. O *Inter-Blockchain Communication Protocol* (IBC) permite que várias redes blockchain se conectem ao Cosmos Hub, possibilitando a troca segura de informações entre estas redes blockchain. Estas redes blockchain conectadas são chamadas de Zonas, cada uma delas sendo uma blockchain individual com suas próprias funcionalidades. Com a conexão estabelecida, as Zonas podem interoperar entre si. Isso significa que dados podem ser trocados entre redes blockchain com diferentes mecanismos de consenso e verificação. A representação na Figura 1.9 ilustra essa interconexão.

A rede Cosmos, faz uso do mecanismo de interoperabilidade *sidechain*, proposto para dimensionar redes blockchain alternativas que são "atreladas bidirecionalmente". A indexação bidirecional é equivalente a uma ponte. Assim, o Cosmos Hub atua como clientes leves um do outro, utilizando provas SPV para determinar quando as moedas devem ser transferidas para a *sidechain* e vice-versa [Kwon and Buchman 2019]. Além de sua função como rede, o Cosmos é um projeto de código aberto. Ele oferece aos desenvolvedores ferramentas e estruturas para construir suas próprias redes blockchain específicas, as chamadas Zonas. Essas Zonas podem ser profundamente customizadas para atender às necessidades de diferentes usuários. Nesse sentido, para o funcionamento do

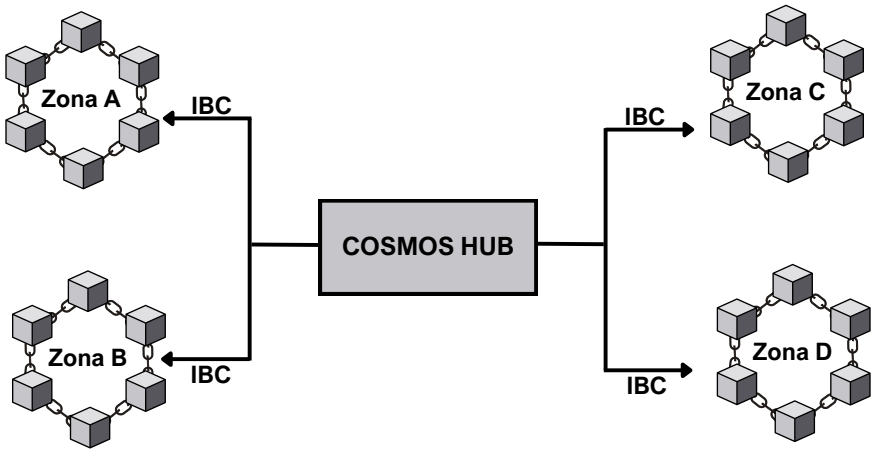


Figura 1.9. Representação do cosmos.

Cosmos há três componentes fundamentais: *Tendermint*, Cosmos SDK e o *Inter-Blockchain Communication Protocol* (IBC).

O *Tendermint* é um conjunto de ferramentas de desenvolvimento que tem como objetivo simplificar o desenvolvimento de camadas de rede e consenso através do *Tendermint Core* (motor de consenso) e a *Application Blockchain Interface* (ABCI) (interface de aplicação genérica).

- **Tendermint Core:** Garante que as mesmas transações sejam registradas em todas as máquinas na mesma ordem e tolera até um terço de suas máquinas falharem arbitrariamente. Isto inclui comportamento explicitamente malicioso.
- **Application BlockChain Interface (ABCI):** Permite que as transações sejam processadas em qualquer linguagem de programação.

O Cosmos SDK é um kit de desenvolvimento de software de código-aberto disponibilizado que permite a criação de redes blockchain e com compatibilidade na rede cosmos. O protocolo de consenso padrão é o *Tendermint Core*, mas há grande variedade de módulos integrados disponíveis. Ele simplifica consideravelmente o processo e oferece todos os padrões para criação de uma blockchain. Atualmente, há alguns projetos conhecidos que foram desenvolvidos com o Cosmos SDK, entre eles a Binance Smart Chain (BSC), KAVA, Celestia, Band Protocol e Terra.

O *Inter-Blockchain Communication Protocol* (IBC) é o protocolo responsável por estabelecer a comunicação *interblockchain*. Ele é responsável por lidar com a autenticação e permite que diferentes redes blockchain transfiram dados e valores entre si diretamente, sem depender de intermediários. O IBC, por meio de um nó de processamento de transações, executa a hospedagem de fundos, envia dados de bloqueio de ativos para outra cadeia, inicia a proposta e desbloqueia o ativo especificado para a cadeia de destino. Além disso, o modo de funcionamento do IBC é semelhante à comunicação TCP/IP utilizada para troca de dados entre a Internet e outras redes. É composto por duas camadas: a camada de transporte (TAO) e a camada de aplicação (APP) [Goes 2020].

A camada de transporte fornece a infraestrutura chave para conectar-se com segurança a outras cadeias e é a base para a construção de outras aplicações. Ela é responsável pelas etapas de autenticação, transporte e ordenação de pacotes de dados. A camada de transporte consiste em canais, clientes leves, conexões e retransmissores. Por outro lado, a camada de aplicação é construída na camada de transporte e determina como os pacotes de dados são empacotados, interpretados e utilizados pelas cadeias de envio e recebimento. Através da interface da camada de aplicação, os usuários finais podem interagir com a *Interchain*, interagindo com NFTs, *tokens* ou outras aplicações que utilizam a camada de transporte.

1.3.2.2. Chainlink

O *Cross-Chain Interoperability Protocol* (CCIP) da Chainlink surge como um protocolo inovador que revoluciona a interoperabilidade entre redes blockchain. Através de uma rede descentralizada de oráculos e contratos inteligentes, o CCIP facilita a transferência de *tokens* e dados entre diferentes redes blockchain, permitindo um ecossistema blockchain interconectado [Breidenbach et al. 2022]. Diferente de *bridges* tradicionais centralizadas, o CCIP interopera de dados e *tokens* descentralizados usando o mecanismo notarial. O protocolo da Chainlink utiliza de três redes descentralizadas e roteadores para fazer a ponte entre redes blockchain, como mostra a Figura 1.10. Para que a transação via CCIP seja efetivada, é necessário confiar nas redes descentralizadas e roteadores, visto que elas serão o notário da transação.

Para que uma transação de envio de dados ou *tokens* seja transportada de uma blockchain para a outra, é necessário que o contrato inteligente da blockchain de origem envie uma transação para o roteador. O contrato que atua como roteador da rede origem recebe como transação uma mensagem codificada com as instruções sobre a blockchain de destino, qual é o destinatário, alguns *tokens* fundíveis como taxa pelo serviço (*feeToken*) e os dados ou *tokens* que serão transferidos. Dessa maneira, o roteador de origem envia a transação para a rede de *commiting* que, por conseguinte, envia para a blockchain de destino, na qual a transação fica armazenada em uma *commit store*. Assim, todo o mecanismo aguarda pela aprovação da *risk management network*, que verifica nas redes de origem e destino se as informações entre elas são compatíveis e não houve alteração. Após a verificação, uma terceira rede, a *executing* efetiva a transação para que o roteador de destino receba a mensagem e execute a instrução no contrato destino.

Apesar de ser um protocolo descentralizado, o CCIP possui a necessidade de confiar em redes blockchain, que estão passíveis de falhas. Além disso, a compatibilidade de redes é um fator importante para o funcionamento do protocolo, ou seja, é necessário que exista um contrato roteador na rede de origem e na rede de destino. Além disso, os *tokens* interoperáveis entre as duas redes podem ser incompatíveis com o protocolo, necessitando a utilização de outras *bridges*.

1.3.3. Perspectivas em Interoperabilidade

A interoperabilidade de redes blockchain, embora tenha avançado significativamente, ainda enfrenta diversos desafios devido à infraestrutura heterogênea dessas redes. Um dos

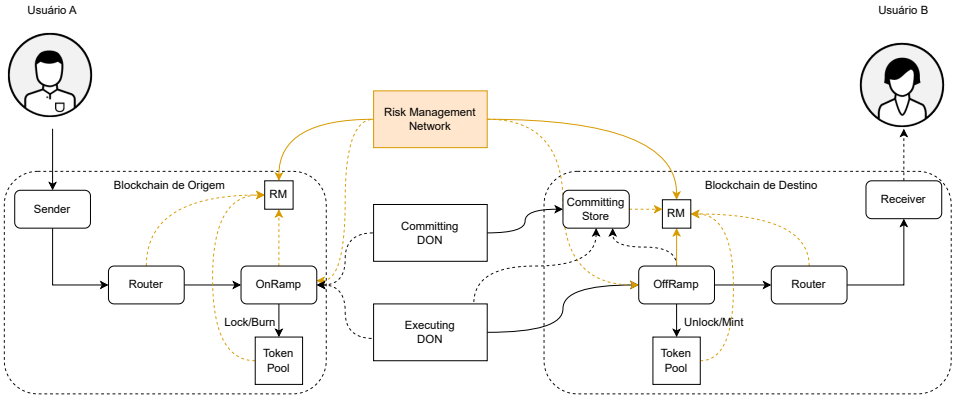


Figura 1.10. Representação do protocolo CCIP.

principais problemas é a integração entre uma blockchain permissionada (privada) e uma não permissionada (pública) [Helliar et al. 2020]. Uma blockchain permissionada exige autenticação e permissões de acesso, dificultando a confiança e a verificação de transações por clientes de uma blockchain não permissionada.

Além disso, uma blockchain permissionada geralmente utiliza protocolos de consensos diferentes, como o RAFT [Ongaro and Ousterhout 2014] (um protocolo CFT – *Crash Fault Tolerance*), ou o BFT-SMART [Bessani et al. 2014] (um protocolo BFT – *Byzantine Fault Tolerance* baseado em quórums), ao invés de PoW e PoS (protocolos BFT com abordagens probabilísticas), comuns nas redes blockchain públicas. Por exemplo, o uso de CFT assume que todos os nós são confiáveis e apenas falham por colapso. Abordagens baseadas em BFT são mais custosas por tolerar nós que podem agir de maneira maliciosa ou arbitrária, e abordagens baseadas em quórum tendem a favorecer consistência e disponibilidade, um par diverso de abordagens probabilísticas quando consideramos a tríade do teorema CAP [Gilbert and Lynch 2002]. Esta diferença de consensos envolve um *trade-off* entre eficiência e robustez.

Outro desafio é a interoperabilidade com redes blockchain que utilizam linguagens de *script* limitadas, como o Bitcoin, que não suporta contratos inteligentes Turing-completos. Isso contrasta com redes blockchain como a Ethereum [Tikhomirov 2018], que permite cálculos complexos necessários para protocolos de cadeia cruzada, dificultando a conexão entre essas redes. Além disso, esquemas de assinatura digital e algoritmos de *hashing* entre diferentes redes blockchain tornam a verificação e implementação de transações cruzadas complexas, já que cada blockchain pode usar métodos distintos para essas operações fundamentais.

1.4. Segurança

Nesta seção serão apresentados os aspectos principais relacionados à segurança no contexto de contratos inteligentes e consequentemente aplicações DeFi. De fato, nos últimos anos, diferentes estudos tratam das pesquisas relacionadas à segurança nos ambientes voltados aos contratos inteligentes [Rouhani and Deters 2019, Harz and Knottenbelt 2018], discutindo a importância e a constante evolução desses. O conceito de segurança em

uma rede blockchain está diretamente relacionado aos métodos de ataque e também aos mecanismos de defesa a esses ataques [Sayeed et al. 2020].

Nesse sentido, tratamos nas seções seguintes, primeiramente, do tópico auditoria de contratos inteligentes, que aborda mecanismos de defesa no aspecto estático e dinâmico do código fonte desses programas. A seguir, focamos no comportamento dos usuários do sistema e as ameaças ocasionadas a partir desse, em especial, ataques de *front-running*, manipulações de oráculos, *flash-loans* e ataque de liquidez, que são algumas das ameaças mais recentes em redes blockchain públicas. Por conseguinte, discutimos na seção final as perspectivas em segurança explorando as pesquisas estado-da-arte que abordam ambos os aspectos estáticos e dinâmicos em segurança de contratos inteligentes e do ecossistema DeFi.

1.4.1. Auditoria de Contratos Inteligentes

Conforme discutido na Seção 1.2, contratos inteligentes são programas escritos em uma linguagem de programação, que utilizam redes blockchain como camada de execução [Zou et al. 2019]. Atualmente, os contratos inteligentes são amplamente utilizados em aplicações de finanças descentralizadas (DeFi) para a emissão de *tokens*, delegação de posse, transferência de ativos digitais e para pagamentos P2P automáticos. Eles ganharam popularidade nos últimos anos por incorporar propriedades de redes blockchain. O código intermediário de um contrato inteligente (*bytecode*) e as transações dele derivadas são gravados de maneira imutável, consistente e descentralizada em vários nós da rede blockchain. Logo, contratos inteligentes possibilitam a implantação e a operação de aplicações descentralizadas (DApps) com segurança, o que é um atrativo para diferentes áreas de negócios, desde finanças ao entretenimento digital [Harvey et al. 2021].

Semelhantemente, as redes blockchain também ganharam visibilidade nos últimos anos e, com isso, diversas redes tais como Ethereum [Buterin et al. 2014] e HyperLedger Fabric [Androulaki et al. 2018] surgiram. Enquanto algumas redes são públicas, isto é, o acesso é aberto e para executar uma transação basta apenas pagar as taxas da rede (mineração e complexidade das operações no contrato), outras redes são permissionadas, restringindo seu acesso a membros ou grupos confiáveis. Pelo fato das redes públicas exigirem taxas ao executar uma operação do contrato inteligente, seu código deve ser simples e otimizado. Por outro lado, as redes permissionadas permitem às organizações e/ou indivíduos executarem transações em contratos com códigos contendo operações mais complexas que nas redes públicas.

Independentemente da complexidade do contrato inteligente implementado, faz-se necessário verificar a sua corretude, ou seja, se a implementação não possui vulnerabilidades de segurança e segue boas práticas de codificação. Uma vulnerabilidade de segurança pode ser definida como uma falha em um sistema que pode ser explorada por um agente malicioso [Almakhour et al. 2020]. Se tratando de contratos inteligentes e blockchain, uma vulnerabilidade pode ser uma falha devido à má codificação, como o uso inadequado de funcionalidades da linguagem de programação do contrato, ou uma falha explorada a partir da arquitetura blockchain utilizada. Em ambos os casos, uma vulnerabilidade pode acarretar em comportamentos inesperados ou a perda do controle parcial ou total do contrato pelo agente malicioso [Ivanov et al. 2023].

Adicionalmente, para corretude e boas práticas de programação de contratos inteligentes é importante verificar as regras de negócio embutidas no contrato, isto é, se o mesmo atende às regras esperadas e as desempenham de forma segura. Esta verificação independe da rede Blockchain utilizada, seja permissionada ou pública, diferindo apenas na forma e nos métodos de verificação. Uma vez que o contrato inteligente esteja instalado e em funcionamento na rede blockchain, ainda assim ele pode estar suscetível a vulnerabilidades de segurança [Yamashita et al. 2019]. Nesse caso, as transações do contrato também devem ser monitoradas visando identificar uma vulnerabilidade antes que essa seja explorada.

Nesse contexto, diversas pesquisas estão sendo desenvolvidas para garantir a segurança de contratos inteligentes em redes Blockchain [Kushwaha et al. 2022]. Por exemplo, [Kalra et al. 2018] propõe a ferramenta Zeus, criada com o objetivo de analisar vulnerabilidades em contratos inteligentes voltados para as redes Ethereum e HyperLedger Fabric, utilizando a técnica de verificação formal *Model Checking*. Já em [Ding et al. 2021], é apresentada a ferramenta HFContractFuzzer, que mapeia as vulnerabilidades de contratos inteligentes de redes permissionadas codificados na linguagem de alto-nível Go. Com o auxílio da técnica de Fuzzing, os autores conseguiram identificar falhas de segurança em contratos de diversas redes Blockchain. Além das vulnerabilidades de segurança associadas aos contratos inteligentes, as regras de negócio embutidas também devem ser consideradas como pontos importantes de revisão e, dessa forma, em [Liao et al. 2022] é discutido a plataforma ModCon, utilizada principalmente para a geração e execução de testes funcionais voltados para contratos inteligentes escritos na linguagem Solidity.

Trabalho	Análise	Entrada	Técnica	Rede
[Xu et al. 2021]	Estática	Código	Aprendizado de máquina	Ethereum
[Yan et al. 2022]	Estática	Código	Aprendizado de máquina	Ethereum
[Li et al. 2022]	Estática/Dinâmica	Código	Execução simbólica	Hyperledger
[Ghaleb et al. 2023]	Estática	Bytecode	Execução simbólica	Ethereum
[Beillahi et al. 2022]	Estática	Opcodes	Grafo de fluxo de controle	Ethereum
[Zhang et al. 2023a]	Estática	Código	Grafo de fluxo de controle	Ethereum
[Xu et al. 2023]	Estática	Código	Árvore Sintática Abstrata	Hyperledger
[Yadav and Naval 2023]	Estática	Bytecode	Execução simbólica	Ethereum
[Ye et al. 2020]	Estática	Código	Grafo de fluxo de controle	Ethereum
[Wang et al. 2020]	Dinâmica	Opcodes	Aprendizado de máquina	Ethereum
[Rodler et al. 2023]	Dinâmica	Bytecode	Fuzzing	Ethereum
[Liu et al. 2023]	Dinâmica	Código	Fuzzing	Ethereum
[Liao et al. 2022]	Estática	Bytecode	Rede Neural	Ethereum

Tabela 1.1. Trabalhos que propõem ferramentas para auditoria de contratos inteligentes.

A auditoria de contratos inteligentes é uma das etapas mais importantes no ciclo de desenvolvimento de uma aplicação descentralizada envolvendo contratos inteligentes [David et al. 2023]. Geralmente, a auditoria ocorre após a etapa de desenvolvimento do código dos contratos, sendo realizada por empresas especializadas em segurança de software. O objetivo da auditoria é detectar falhas e comportamentos inesperados, *i.e.*, vulnerabilidades, antes da instalação e utilização do contrato na rede blockchain principal. O processo de auditoria consiste principalmente em inspeção manual do código pelo auditor, baseado em históricos de vulnerabilidades catalogados por comunidades de especialistas em segurança.

Nesse caso, o catálogo *Common Weakness Enumeration* é um dos mais conhecidos, sendo mantido por comunidades de desenvolvedores da indústria, academia e governos[1]. Importante também mencionar consórcios de corporações com negócios baseados em redes blockchain que formam alianças para definir especificações de segurança em contratos inteligentes nessas redes. Por exemplo, a *Enterprise Ethereum Alliance* (EEA) propuseram e vem mantendo um conjunto de especificações para a rede Ethereum denominado *EEA EthTrust Security Levels Specification*[2].

Adicionalmente à inspeção manual, existem métodos de análise automáticas de códigos fonte que auxiliam o trabalho de auditores, e os mais conhecidos são os métodos de análise estática e dinâmica. Uma forma simples de conceituar esses métodos concerne o ambiente onde os contratos são executados, *i.e.*, a rede blockchain. A análise estática lida com a identificação de vulnerabilidades no código do contrato sem propriamente implantá-lo no ambiente de execução [Ivanov et al. 2023]. Por sua vez, a análise dinâmica atua identificando vulnerabilidades no que diz respeito ao ambiente de execução em que o código foi implantado, i.e., vulnerabilidades relacionadas ao comportamento das funcionalidades do contrato e das transações gravadas na rede blockchain [Almakhour et al. 2020].

Há um grande esforço no desenvolvimento de ferramentas automáticas que auxiliam o processo de auditoria de contratos inteligentes. Contudo, existem questões práticas de adoção dessas ferramentas que precisam ser cuidadosamente consideradas [Chaliasos et al. 2024]. Primeiro, uma parte das ferramentas de código-aberto já difundidas na literatura, que atuam como *benchmarks* em diversos trabalhos, podem ter sido descontinuadas, *e.g.* Oyente [Luu et al. 2016] e Vandal [Brent et al. 2018]. Segundo, é comum que ferramentas de auditoria deixem de ser projetos de código-aberto e passam a ser incorporadas à uma organização privada, *e.g.* Mythril [ConsenSys 2024] e Smartcheck [Tikhomirov et al. 2018].

A Tabela 1.1 apresenta 13 projetos de ferramentas automáticas de análise de contratos inteligentes que abordam os desafios e mecanismos de verificação automática de código para o contexto do presente trabalho, indicando se a ferramenta utiliza análise análise estática ou dinâmica e se a entrada para a análise é o código fonte, *opcodes* ou *bytecode* do código fonte. Ainda, descrevemos a técnica base para a análise e a rede blockchain utilizada, sendo Ethereum e Hyperledger Fabric as redes que essas ferramentas suportam até o momento.

Em cada ferramenta são utilizadas uma ou mais técnicas de verificação de código já existentes na literatura para verificação de software no geral. Normalmente, as técnicas exigem a conversão do código do contrato inteligente para uma Representação Intermediária (IR), semelhante aos compiladores de linguagens de programação convencionais. Na literatura, as ferramentas variam ou combinam técnicas de acordo com as vulnerabilidades que desejam identificar. Por exemplo, a ferramenta Manticore [Mossberg et al. 2019] utiliza a técnica Execução Simbólica para explorar o espaço de estados do programa. Conforme a Figura 1.11, a partir de uma entrada simbólica, o código de um contrato inteligente pode ser analisado pelos estados gerados de acordo com a entrada, com o objetivo de identificar

[1] `https://cwe.mitre.org/`

[2] `https://entethalliance.org/specs/ethtrust-sl/`

se estados e/ou propriedades que não podem ser atingidos (*i.e.* vulnerabilidades) realmente não são.

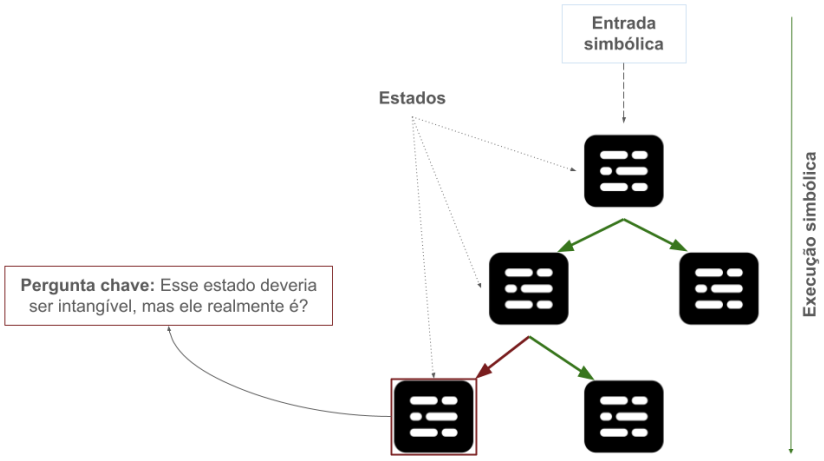


Figura 1.11. Fluxo da execução simbólica sobre os estados de um contrato inteligente.

Geralmente, a técnica de execução simbólica pode ser combinada com as técnicas de verificação formal, como o *Model Checking*. Em [Song et al. 2022], é apresentada a ferramenta ESBMC-Solidity, que utiliza a verificação formal em conjunto com a execução simbólica para checar as propriedades de vulnerabilidades em contratos inteligentes escritos na linguagem de programação Solidity. Dessa maneira, ameaças intrínsecas ao código e da linguagem do contrato podem ser identificadas. De contrapartida, brechas de segurança referentes ao fluxo de execução do contrato inteligente exigem técnicas capazes de analisar o comportamento das funcionalidades do código que são invocadas durante sua utilização. Sendo assim, técnicas de análise dinâmica como *Fuzzing* mostram-se como uma solução promissora na literatura.

Inicialmente, a técnica de *Fuzzing* foi concebida como um mecanismo de testagem de software [Zeller et al. 2019]. Os recentes trabalhos expandem este conceito para o contexto de blockchain e contratos inteligentes, em que a técnica pode ser utilizada como um mecanismo de verificação do comportamento de utilização do código de um contrato, como mostra a Figura 1.12. A partir de entradas geradas aleatoriamente e em grande quantidade, o contrato inteligente pode ser monitorado de acordo com os resultados das execuções de suas funcionalidades, com o objetivo de serem identificadas vulnerabilidades que dizem respeito não somente ao código do contrato, mas também da arquitetura da rede blockchain utilizada e as regras de negócio embutidas.

Nesse sentido, recentes trabalhos integram o *fuzzing* de diversas maneiras com o ecossistema de contratos e blockchain. Em [Wüstholz and Christakis 2020] é proposto a ferramenta Harvey, que estende o conceito de geração de entradas da técnica convencional para gerar resultados satisfatórios no contexto de contratos inteligentes. Além disso, a ferramenta executa os testes com as entradas priorizando a exploração de estados possíveis do contrato de maneira inteligente, levando em consideração as transações que ocorrem na rede devido às funcionalidades do contrato. Semelhantemente em [Jiang et al. 2018], a ferramenta propõe a utilização da técnica *fuzzing* combinada com a criação de oráculos de

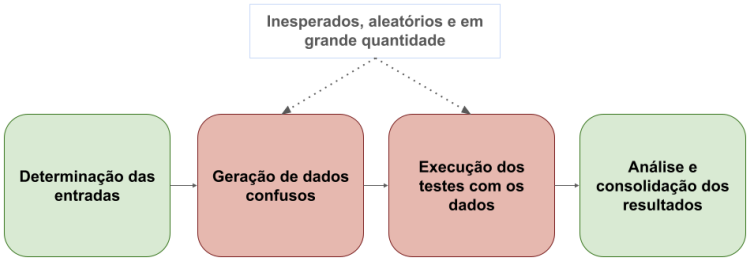


Figura 1.12. Fluxo da técnica de *fuzzing* para identificação de falhas.

teste, em que cada oráculo é responsável por identificar uma ameaça específica.

Em suma, o processo de auditoria dos contratos inteligentes auxilia na mitigação de perdas de ativos financeiros no ecossistema DeFi. A junção das técnicas de verificação de software convencionais com o contexto de contratos inteligentes podem acarretar em análises assertivas, desde que haja especificações detalhadas de possíveis falsos positivos e constante suporte às ferramentas difundidas na literatura.

1.4.2. *Front-running* em Redes Blockchain

Qualquer ação realizada pelo usuário que modifique o estado da blockchain é registrada como uma transação. Uma vez efetivada, a transação não pode ser revertida e é esse recurso que torna a blockchain um livro razão imutável. Nesse sentido, os contratos inteligentes, assim como os usuários externos, são capazes de armazenar ativos em contas endereçáveis e realizar transações que alteram o estado da rede [Varun et al. 2022].

Contratos inteligentes são programas criados usando uma linguagem de programação de alto nível como Solidity. Eles são projetados para serem executados quando um conjunto predeterminado de condições forem atendidas. Geralmente, eles automatizam a execução de um ativo ou acordo que garante que todos os participantes conheçam instantaneamente o resultado, sem o envolvimento de qualquer parte intermediária. Os contratos inteligentes também podem automatizar processos, ou seja, desencadear outra ação assim que o contrato atual for executado. Uma vez que são implantados na blockchain, quando as condições exigidas são atendidas, eles são executados por uma rede de nós que chegam a um consenso antes que o estado executado seja armazenado na blockchain.

Uma blockchain é um registro de transações anexado. Por sua vez, as transações são armazenadas em uma *pool* e tratadas igualmente, independentemente do horário específico em que elas foram adicionadas, para posteriormente serem combinadas em um bloco por nós mineradores. Dessa maneira, mais de 95% dos mineradores optam por ordenar as transações em relação ao preço do gás, que é a taxa de transação que os mineradores recebem por anexar uma transação em um bloco [Zhou et al. 2021].

Uma das principais ameaças ao ecossistema DeFi na blockchain Ethereum atualmente é a manipulação na ordem de transações que serão efetivadas na rede por usuários estratégicos que monitoram constantemente a fila pública de transações pendentes (i.e., *mempool*), ataque esse conhecido como *front-running*.

Como mostra a Figura 1.13, nesse tipo de ataque o valor da tarifa da transação é

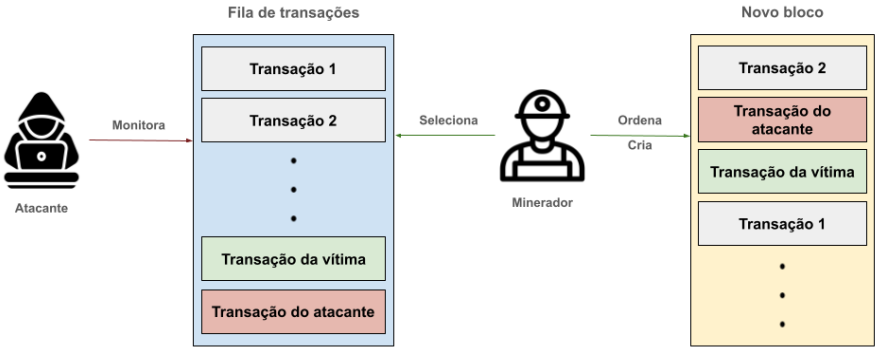


Figura 1.13. Demonstração do cenário de ataque *front-running*.

utilizado pelo atacante para manipular a ordem das transações que aguardam na *mempool* para constituírem o novo bloco. Por essência, o consenso distribuído em blockchain elimina uma autoridade central para gerenciar a *mempool* e evitar ataques *front-running*. Assim, vem ocorrendo um aumento no número de atacantes, bem como várias formas de ataques *front-unning* [Varun et al. 2022, Torres et al. 2021, Zhang et al. 2023c, Zhang et al. 2023b].

Existem três tipos principais de ataques *front-running*: deslocamento, inserção e supressão. O ataque de inserção é conhecido como ataque sanduíche. No ataque de deslocamento, a transação de ataque desloca a transação da vítima por ter um preço de gás mais alto. Como a transação de ataque oferece melhor incentivo, ela é extraída antes da transação da vítima. Este ataque é usado em jogos de quebra-cabeça onde é preciso enviar a chave para vencer o jogo. Assim que o atacante vê a solução da vítima, ele cria a mesma transação com um preço de gás mais alto, inutilizando a transação da vítima. No ataque de supressão, um atacante faz múltiplas transações com um preço de gás muito mais alto do que a transação da vítima para evitar que a transação da vítima seja explorada no mesmo bloco. Esse ataque também é conhecido como *cluster* de supressão. É frequentemente usado em jogos de loteria que têm como regra que a última pessoa a entrar na loteria ganha.

O ataque sanduíche, que é um tipo *front-running*, é uma estratégia de negociação já conhecida nos sistemas financeiros tradicionais, onde um usuário com visão privilegiada do sistema identifica um negócio promissor de outro usuário e o executa antecipadamente obtendo os benefícios desse negócio. Esse tipo de ataque vem chamando a atenção de pesquisadores recentemente no contexto da blockchain Ethereum e DeFi [Torres et al. 2021, Zhang et al. 2023b]. Nesse caso, um atacante inicia monitorando a *mempool* em busca de transações pendentes que estão prestes a negociar grandes somas de um determinado ativo. Uma grande transação resultará em uma flutuação no preço do ativo. Na sequência, o atacante cria o chamado "sanduíche", cercando esta grande transação com duas de suas próprias transações. Na primeira transação, o atacante executa uma grande transação para comprar ou vender alguma quantidade de ativo antes que o preço do ativo flutue. Na segunda transação, o atacante retrocede a grande transação para recomprar o ativo original por um preço mais baixo ou vender o ativo recém-adquirido por um preço mais alto. O atacante obtém lucro devido à diferença de preço e a vítima pode sofrer prejuízo [Weintraub et al. 2022].

Os atacantes podem ser mineradores ou não mineradores. Os mineradores não são obrigados a pagar um preço mais alto de gás para manipular a ordem das transações, pois têm controle total sobre as transações que são incluídas em um bloco. Os não mineradores, por outro lado, são obrigados a pagar um preço mais elevado de gás para antecipar as transações de outros não mineradores. Em [Torres et al. 2021] é assumido que o atacante é um não minerador financeiramente racional com a capacidade de monitorar a *mempool* de transações. O atacante precisa processar as transações na *mempool*, encontrar uma vítima e criar transações de ataque antes que a transação da vítima seja minerada.

O atacante não seria capaz de reagir rápido o suficiente para realizar todas as tarefas necessárias para efetuar o ataque manualmente. Portanto, seguindo [Torres et al. 2021], o atacante possui pelo menos um programa de computador (*Bot*) que executa automaticamente as tarefas necessárias para o ataque. O *Bot* precisa de pelo menos uma ou mais contas de propriedade externa (EOA - *Externally Owned Accounts*) para atuar como remetente de qualquer transação de ataque. O uso de várias contas de propriedade externa auxilia os atacantes a ocultar suas atividades, semelhante aos esquemas de lavagem de dinheiro. Assumimos que o atacante possui um saldo suficientemente grande em todas as suas contas, a partir do qual pode enviar transações de ataque com gás suficientemente maior que o gás da transação da vítima. No entanto, o atacante também pode empregar contratos inteligentes para manter parte da lógica do ataque. Esses contratos inteligentes são referidos como contratos de *Bot*, que são invocados pelas contas do atacante.

A Figura 1.14 apresenta um cenário com ataque sanduíche. No ataque sanduíche, o atacante envia duas transações, uma com um preço de gás mais alto do que a transação da vítima e outra com um preço de gás mais baixo para intercalar a transação da vítima. É usado em plataformas descentralizadas de câmbio (DEXes - *Descentralized Exchanges*) para fazer sanduíches de transações prestes a negociar grandes somas de um determinado ativo, também conhecido na literatura como transações baleia [Varun et al. 2022].

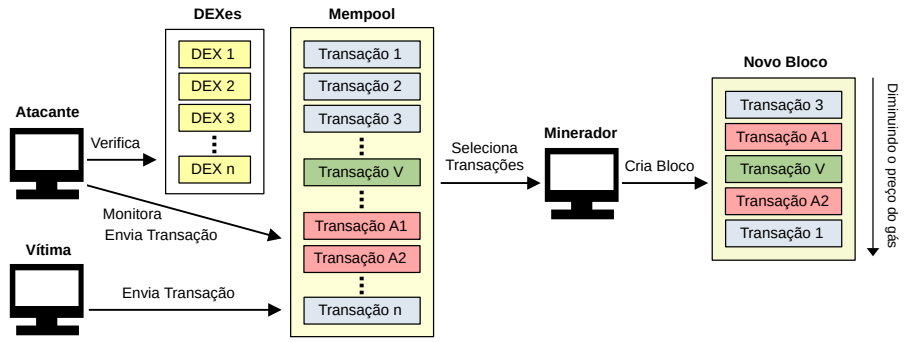


Figura 1.14. Demonstração de um cenário com ataque sanduíche.

Observa-se na Figura 1.14 que o atacante monitora a *mempool* na espera de uma possível transação vítima (Transação V). O atacante verifica nas DEXes se o ativo da transação da vítima pode gerar lucro. Em caso positivo, o atacante gera duas transações (Transação A1 e Transação A2). Elas, por sua vez, possuem valores de gás que fazem com que realizem um "sanduíche" com a transação da vítima quando o minerador selecionar transações para criar um bloco seguindo a ordenação convencional pelo preço do gás.

Os ataques sanduíche exploram as AMM DEXes. Uma ordem de compra aumentará o preço de um ativo, enquanto uma ordem de venda diminuirá o preço do ativo. Portanto, os especuladores podem monitorar continuamente a rede (*i.e. mempool* e AMMs) para encontrar transações pendentes para AMM DEXes (*i.e.* transação da vítima) que implicarão diferenças de preços. Dessas DEXes, os especuladores podem comprar o ativo por um preço baixo antes que a transação da vítima seja executada (transação T_{A1}) e venderem o ativo após a transação da vítima aumentar de preço (transação T_{A2}), gerando lucro para si. Ao final do ataque, a cotação da transação vítima é pior do que seria sem a transação T_{A1}, resultando em perda financeira para a vítima [Wang et al. 2022].

Para identificar uma arbitragem de ataque sanduíche, podem ser verificadas todas as transações de transferência em um bloco. Uma transação de transferência é definida como $T = (s, r, a, c, h, i)$, em que s é remetente dos *tokens*, r é o receptor dos *tokens*, a é o número de *tokens* transferidos, c é o endereço do contrato do *token*, h é o *hash* da transação, e i é o índice da transação. Ao verificar todas as transações em um bloco, busca-se encontrar três transações de transferência: T_{A1}, T_V e T_{A2}. As transações T_{A1} e T_{A2} estão relacionadas com o atacante, e a transação T_V está relacionada com a vítima.

Um possível algoritmo para detecção de arbitragem de ataque sanduíche segue as suposições a seguir baseadas em [Torres et al. 2021]. As transações T_{A1}, T_V e T_{A2} devem estar nessa ordem, ou seja, o índice de T_{A1} deve ser menor do que o índice de T_V e o índice de T_V deve ser menor que o índice de T_{A2} ($i_{A1} < i_V < i_{A2}$). O atacante e a vítima compram *tokens* em T_{A1} e em T_V, respectivamente. Em seguida, o atacante vende os *tokens* em T_{A2} que comprou anteriormente em T_{A1}. O número de *tokens* comprados por T_{A1} deve ser semelhante ao número de *tokens* vendidos por T_{A2} (ou seja, $a_{A1} \approx a_{A2}$). O atacante e a vítima realizam transações sobre os mesmos *tokens*, ou seja, os endereços de contrato de *token* de T_{A1}, T_V e T_{A2} devem ser idênticos ($c_{A1} = c_V = c_{A2}$). O remetente de T_{A1} deve ser idêntico ao remetente de T_V, bem como o receptor de T_{A2}, e o receptor de T_{A1} deve ser idêntico ao remetente de T_{A2} (ou seja, $s_{A1} = s_V = r_{A2} \wedge r_{A1} = s_{A2}$). Os *hashes* de transação de T_{A1}, T_V e T_{A2} devem ser diferentes (ou seja, $h_{A1} \neq h_V \neq h_{A2}$). O preço do gás de T_{A1} deve ser maior que o preço de gás de T_V. E o preço do gás de T_{A2} deve ser menor ou igual ao preço do gás de T_V (ou seja, $g_{A1} > g_V \geq g_{A2}$).

O algoritmo para detecção de arbitragem de ataque sanduíche assume que os ataques sanduíche sempre ocorrem dentro do mesmo bloco. Essa suposição permite verificar os blocos em paralelo, uma vez que só é preciso comparar as transações dentro de um bloco. No entanto, esta suposição nem sempre se aplica à realidade, uma vez que as transações podem ser dispersas por diferentes blocos durante o processo de mineração. Sendo assim, podem existir ataques sanduíches realizados em vários blocos e que o algoritmo não é capaz de detectar. Portanto, essa abordagem representa um limite inferior para análises de ataques sanduíches na blockchain Ethereum. Contudo, existem abordagens que expandem a análise para mais de um bloco, como no trabalho de [Zhang et al. 2023b] que analisa os ataques em uma janela de 3 blocos consecutivos.

1.4.3. Manipulação de Oráculos

Um oráculo conecta o mundo blockchain on-chain com o mundo off-chain, por meio de interface com provedores de dados externos.

Oráculos provêm informações fundamentais para ativar gatilhos e tomadas de decisão em contratos inteligentes. Por serem entidades de confiança, têm o "privilégio"de fornecer dados aceitos incondicionalmente. Este detalhe é crucial, pois todo o ecossistema da blockchain gira em torno do conceito de imutabilidade e interação sem confiança por meio da descentralização. Conectar a blockchain a um ponto de falha centralizado, como um oráculo, resulta fundamentalmente em uma perda de descentralização. Esse dilema é conhecido como "o problema do oráculo"e afeta todas as aplicações blockchain do mundo real. Dependendo do setor e do tipo de oráculo, diferentes consequências podem surgir [Caldarelli and Ellul 2021].

Existem várias circunstâncias em que um oráculo pode deixar de fornecer dados confiáveis, assumindo um modelo de falhas bizantino: Os oráculos podem ser mal programados, apresentar *bugs*, sofrer sabotagem ou mau funcionamento. O problema do oráculo pode envolver tanto questões técnicas, mas também aspectos sociais.

Como um ponto único de falha, a chance de um oráculo ser comprometido ou dos administradores conspirarem para alterar os dados fornecidos pode estar correlacionada ao valor dos contratos inteligentes: quanto maior o valor dos ativos manipulados pelo contrato inteligente, maior a probabilidade dele ser alvo de ataques. Um oráculo pode assim ser comprometido por meio de conluio ou suborno dos administradores da entidade gestora do oráculo que pode alterar a transferência de dados para fins egoístas.

Um oráculo que provê informações sobre quantidade e preço de ativos financeiros negociados provê um conjunto de informações pública que pode ser facilmente verificável. O uso de um mecanismo de quórum com diferentes provedores desta informação pode prover um oráculo decentralizado resiliente a provedores bizantinos. O provimento de canais seguros para evitar a adulteração de dados providos pelo oráculo também pode ser utilizado. Por fim, aspectos temporais podem ser observados, em face a situações que um atacante tente atrasar a atualização de informações providas por um oráculo (*e.g.*, a alteração de preço de um dado ativo) para tomar vantagem da posição desatualizada.

1.4.4. Ataques de *Flash Loan*

Um ataque de *flash loan* é um mecanismo de exploração sofisticado no ecossistema DeFi por meio do uso de empréstimos rápidos (ditos *flash loans*). Estes empréstimos rápidos são uma proposta sem precedentes do ecossistema de Finanças Descentralizadas (DeFi), em 2018 o projeto *Marble 1* idealizou o mecanismo para permitir que qualquer pessoa emprestasse ativos sem garantia para aproveitar oportunidades de arbitragem, desde que os fundos fossem devolvidos no âmbito da mesma transação [Cao et al. 2021].

O desafio é que o atacante utiliza desta possibilidade para manipular valores de um dado ativo e obter vantagens indevidas. Em um passo a passo exemplificamos como um ataque de *flash loan* pode acontecer:

1. Tomada do Empréstimo: O atacante adquire um empréstimo rápido de uma grande quantidade de um dado *token* sem fornecer garantias. Isso é possível porque o reembolso do empréstimo é garantido pelo próprio protocolo de empréstimo dentro da mesma transação;

2. Manipulação do Mercado: Usando os fundos do empréstimo, o atacante pode

manipular o preço de um ativo em uma ou várias corretoras descentralizadas. Por exemplo, ele pode comprar uma grande quantidade de um dado *token* para inflar seu preço ou vender em massa para diminuir o preço;

3. Realização do Lucro: Após manipular o mercado e explorar as vulnerabilidades, o atacante troca os ativos de volta para a criptomoeda original, mas agora com lucro; e

4. Reembolso do Empréstimo: Finalmente, dentro da mesma transação, o atacante reembolsa o empréstimo rápido junto com a taxa de empréstimo. Como toda a operação ocorre dentro de um único bloco, se em algum ponto o atacante não puder reembolsar o empréstimo, a transação é revertida, e nada acontece.

Este mecanismo é sofisticado e requer uma análise do padrão comportamental das transações para inferir possíveis intenções dos remetentes [Wang et al. 2021].

1.4.5. Ataques de Liquidez

Os mecanismos descentralizados inerentes a DeFi provêm uma assimetria de informações que pode ser utilizada por atacantes. Por exemplo, ataques de arbitragem de liquidez exploram as diferenças de preço entre diferentes *pools* de liquidez ou corretoras. Um atacante pode usar *bots* de arbitragem para detectar e explorar essas discrepâncias de preços, comprando *tokens* em uma plataforma onde o preço é baixo e vendendo-os em outra onde o preço é mais alto. Embora a arbitragem em si não seja maliciosa, quando feita de forma agressiva ou com informações privilegiadas, pode levar à drenagem de liquidez de certos *pools*, prejudicando os participantes honestos.

Outra possibilidade é manipular o preço de um ativo dentro de um pool de liquidez injetando uma grande quantidade de um *token* em um pool, o que inflaciona artificialmente o preço do *token*. Uma vez que o preço é manipulado, o atacante pode executar negociações subsequentes em outras plataformas ou explorar contratos inteligentes que dependem de oráculos de preço que agora refletem o preço manipulado.

Os ataques de manipulação de preços originam-se de vulnerabilidades lógicas de aplicativos DeFi, o que torna a detecção não trivial. Ou seja, a detecção de tais ataques exige que analisemos as transações entre vários contratos inteligentes e compreendamos a semântica de alto nível dos aplicativos DeFi [Wu et al. 2021].

1.4.6. Perspectivas em Segurança de DApps

A segurança e correção de contratos inteligentes representam um desafio importante para o ecossistema de aplicações DeFi e blockchain. Esses contratos, que gerenciam grandes quantidades de criptomoedas e ativos digitais são alvos frequentes de ataques [Ivanov et al. 2023]. Devido à imutabilidade das redes blockchain, contratos inteligentes precisam ser projetados e testados rigorosamente antes da implantação. Apesar dos avanços na segurança dos contratos inteligentes, ataques recentes a plataformas, destacam vulnerabilidades persistentes. *Bugs* podem surgir tanto das limitações das linguagens de programação de contratos inteligentes quanto de erros na lógica de negócios, sendo o último mais difícil de detectar e corrigir, exigindo uma modelagem precisa e análise cuidadosa por especialistas.

Nesse sentido, a combinação de técnicas de verificação de contratos inteligentes

pode ser um caminho promissor. Alguns trabalhos propõem a utilização de análise estática e dinâmica para identificação de brechas em contratos [Linoy et al. 2021], enriquecendo a etapa de análise de brechas e riscos para uma aplicação DeFi no geral. Embora a combinação de técnicas desempenhe um papel inovador para o campo de verificação de contratos, estratégias que melhoram o desempenho das soluções atuais também são importantes. Os contratos inteligentes implantados em redes blockchain, muitas das vezes, possuem diversas chamadas de contratos externos, contratos estes que podem possuir brechas significativas para um prejuízo financeiro [Liao et al. 2022]. Logo, a verificação *cross-contract* mostra-se como um desafio para as soluções atuais difundidas na literatura [Ye et al. 2020]. Por fim, a utilização de técnicas *out-of-the-box* traçam um caminho inovador para a verificação de contratos, isto é, a utilização de mecanismos e técnicas que não são do campo de verificação de software no geral, como o aprendizado de máquina. Esta prática, como mostra [Xu et al. 2021], apesar de ser um meio de verificação que deve ser extensivamente analisado em busca de falsos positivos e negativos, pode gerar *insights* e oportunidades de pesquisa capazes de propor estratégias de mitigação de certas brechas que são comuns para redes blockchain.

No que diz respeito à interoperabilidade entre redes blockchain, a segurança torna-se um desafio crítico, com ataques a pontes de cadeia cruzada causando perdas financeiras significativas. Um dos principais problemas de segurança é o ataque de duplo gasto [Chohan 2021] entre redes blockchain. Esse ataque ocorre quando uma criptomoeda é usada mais de uma vez, explorando fraquezas em redes blockchain PoW, especialmente quando um grupo de mineradores controla mais de 50% do poder de mineração [Conti et al. 2018]. No contexto de cadeias cruzadas, esses ataques podem acontecer antes ou depois das transações, impactando negativamente ambas as redes blockchain envolvidas. Para mitigar esses riscos, protocolos de cadeia cruzada geralmente implementam tempos de espera elevados para confirmar transações, pois isso permite uma verificação mais robusta e garante que a transação seja irreversível e aceita por todos os nós da rede.

Adicionalmente, a privacidade é um desafio significativo na interoperabilidade devido ao potencial de violação do anonimato e rastreabilidade. Em abordagens tradicionais de interoperabilidade, partes confiáveis atuam como intermediárias, comprometendo o anonimato dos usuários ao verificar transações e bloquear ativos. Para preservar a privacidade, é necessário desenvolver uma infraestrutura de interoperabilidade que elimine a necessidade de intermediários e manter o mesmo nível de anonimato que os sistemas de blockchain originais. Uma solução para isso que vem sendo avaliada por pesquisadores é a utilização do mecanismo de *Zero Knowledge Proof* (ZKP). Neste mecanismo, uma informação sensível como dados de transações de interoperabilidade podem ser propagadas para a rede sem necessariamente revelá-las [Sun et al. 2021]. Nesse sentido, um dos principais problemas nesse contexto é a rastreabilidade das transações entre redes blockchain. Por exemplo, trocas HTLC (*Hashed Time-Lock Contract*) são utilizadas em transações de criptomoedas para garantir que duas partes só possam concluir uma troca se certas condições forem atendidas. No entanto, se o mesmo valor *hash* for usado em várias transações, pode haver um risco de segurança, pois o valor secreto revelado em uma transação pode ser usado para reivindicar outras transações que dependem do mesmo *hash*. Para resolver isso, são necessários mecanismos adicionais de preservação da privacidade, como assinaturas adaptadoras [Deshpande and Herlihy 2020], semelhante ao ZKP, em que

permitem que uma transação seja assinada de uma maneira que não revele informações sobre a transação em si. Ainda, ambientes de execução confiáveis [Sabt et al. 2015] [Li et al. 2021] podem ser avaliados, garantindo que os dados e códigos sejam mantidos em segredo, mesmo quando são processados para impedir que essas conexões sejam detectadas.

Além disso, a privacidade dos dados também é um desafio crítico no que diz respeito às aplicações descentralizadas, especialmente em setores sensíveis a esse tipo de problema. O compartilhamento e transferência de dados sensíveis que circulam nas redes podem gerar riscos às plataformas financeiras. Por exemplo, alguns trabalhos discutem possibilidades estratégicas de mitigação de ataques criando filas de transações pendentes privadas [Eskandari et al. 2020], para que aplicações não exponham transações importantes livremente na rede. Apesar de ser uma solução promissora, o seu uso impacta diretamente no pilar de descentralização de redes blockchain, sendo portanto um desafio de constantes pesquisas e inovações. Já em [Ma et al. 2019], são discutidos os mecanismos de proteção de privacidade implementados no *Hyperledger Fabric* e sua aplicação no contexto da cadeia de suprimentos financeira. De maneira semelhante, essa perspectiva beneficia diretamente quando se trata de interoperabilidade em redes blockchain garantindo a privacidade de todas as partes envolvidas. A interoperabilidade deve garantir que a privacidade seja mantida, mesmo quando interconectando diferentes redes blockchain protegendo informações sensíveis de acesso não autorizado.

1.5. Considerações Finais

Este capítulo abordou os conceitos, aplicações e perspectivas sobre o ecossistema de finanças descentralizadas. Com o advento das redes blockchain e o uso de contratos inteligentes com a rede Ethereum, as aplicações de finanças descentralizadas (DeFi - *Decentralized Finance*) emergiram como uma solução flexível para movimentação de ativos financeiros. Este novo ecossistema visa aprimorar os serviços financeiros tradicionais por meio da negociação de *tokens*, da redução de intermediários e de barreiras para o crédito, bem como do acesso mais amplo a serviços financeiros no contexto da *web* descentralizada. Este novo ecossistema propiciou um conjunto de aplicações inovadoras, desafios a interoperabilidade entre diferentes redes blockchain, bem como novas demandas de segurança para tratar as possíveis vulnerabilidades e ameaças neste ambiente descentralizado. Neste sentido, apresentamos os fundamentos e exploramos, com exemplos práticos, algumas das mais utilizadas aplicações DeFi. Observando o ecossistema DeFi sob a ótica destes três aspectos, aplicações, interoperabilidade e segurança, discutimos as perspectivas, os desafios e o estado-da-arte que suportará o crescimento e uso de DeFi nos próximos anos.

Em aplicações, discutimos a arquitetura de DeFi no contexto de redes blockchain públicas. A partir da introdução dos *tokens* em conjunto com os contratos inteligentes, as finanças descentralizadas representam majoritariamente as transações que ocorrem em redes blockchain como a Ethereum. É possível notar a introdução de instituições financeiras e governamentais neste ecossistema por meio das *stablecoins* e CDBCs. Além disso, as corretoras descentralizadas representam o meio mais utilizado atualmente para os usuários usufruírem dos recursos que o ecossistema DeFi tem a oferecer com a utilização de redes blockchain.

Em interoperabilidade apontamos os mecanismos de transferência de ativos entre

redes mais utilizados atualmente. A interoperabilidade, se tratando de redes blockchain e DeFi, representa a comunicação entre duas redes distintas, ou seja, o usuário possui a capacidade de trafegar entre redes diferentes, movimentar ativos e utilizar plataformas de câmbio de maneira segura e escalável. Além disso, discutimos as soluções implementadas em redes como Ethereum, levando em consideração aspectos de segurança e centralização de ativos. Por exemplo, o mecanismo de *sidechains* permite que as redes blockchain principais (*mainchains*) possuam diferentes implementações com objetivos distintos, *e.g sidechains* com foco em governança, latência de transações por segundo, etc. Por outro lado, enquanto mecanismos como o HTLC abordam transações atômicas sem a necessidade de um terceiro confiável, mas geram riscos de privacidade, mecanismos como o notarial introduzem um terceiro confiável, com o objetivo de proporcionar maior privacidade, porém centraliza toda a troca de ativos em um notário.

Em segurança abordamos os desafios práticos de implantação de uma aplicação descentralizada por meio de contratos inteligentes. O código destes contratos podem estar suscetíveis à vulnerabilidades de segurança e, portanto, passam por um rigoroso processo de auditoria. Por meio de técnicas convencionais e *out-of-the-box* ("fora da caixa") de verificação de softwares, pesquisadores e iniciativas privadas fomentam um campo de pesquisa de criação de técnicas e *frameworks* capazes de identificar potenciais ameaças em ambientes redes blockchain. Nesse sentido, apresentamos algumas das principais vulnerabilidades que ocorrem em redes blockchain públicas como a Ethereum e ecossistemas DeFi, *i.e. front-running*, manipulações de oráculos, *flash-loans* e ataques de liquidez. Estas práticas representam um perigo para redes públicas, devido à influência negativa que estas transações podem ocasionar em preços de *tokens* ao tratarmos de finanças descentralizadas.

Finalmente, condensamos as perspectivas de segurança ao abordamos os aspectos de auditoria, interoperabilidade e aplicações. Evidenciamos oportunidades de pesquisa nas três subáreas que constituem o ecossistema DeFi, de forma que este capítulo proporcione uma base de conhecimento e perspectivas do estado-da-arte para interessados na área de finanças descentralizadas.

Agradecimentos

Os autores agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) processo no. 88887.918434/2023-00, Fundação de Amparo à Pesquisa do Piauí (FAPEPI) processo no. 00110.000235/2022-78 e o Comitê Técnico Blockchain da Rede Nacional de Pesquisas CT-Blockchain RNP. Agradecem também o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

Referências

[Allen et al. 2022] Allen, F., Gu, X., and Jagtiani, J. (2022). Fintech, cryptocurrencies, and cbdc: Financial structural transformation in china. *Journal of International Money and Finance*, 124:102625.

[Almakhour et al. 2020] Almakhour, M., Sliman, L., Samhat, A. E., and Mellouk, A. (2020). Verification of smart contracts: A survey. *Pervasive and Mobile Computing*,

67:101227.

[Androulaki et al. 2018] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15.

[Back et al. 2014] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., and Wuille, P. (2014). Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72:201–224.

[Beillahi et al. 2022] Beillahi, S. M., Keilty, E., Nelaturu, K., Veneris, A., and Long, F. (2022). Automated auditing of price gouging tod vulnerabilities in smart contracts. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–6. IEEE.

[Belchior et al. 2021] Belchior, R., Vasconcelos, A., Guerreiro, S., and Correia, M. (2021). A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41.

[Besançon et al. 2019] Besançon, L., Silva, C. F. D., and Ghodous, P. (2019). Towards blockchain interoperability: Improving video games data exchange. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 81–85.

[Bessani et al. 2014] Bessani, A., Sousa, J., and Alchieri, E. E. (2014). State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE.

[Binance 2023] Binance (2023). O que é uma stablecoin? `https://academy.binance.com/pt/articles/what-is-a-stablecoin`. (Accessed on 05/23/2024).

[Breidenbach et al. 2022] Breidenbach, L., Cachin, C., Chan, B., Coventry, A., Ellis, S., Juels, A., Koushanfar, F., Miller, A., Magauran, B., Moroz, D., et al. (2022). Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. 2021. *Available also from: https://research. chain. link/whitepaper-v2. pdf*.

[Brent et al. 2018] Brent, L., Jurisevic, A., Kong, M., Liu, E., Gauthier, F., Gramoli, V., Holz, R., and Scholz, B. (2018). Vandal: A scalable security analysis framework for smart contracts. *arXiv preprint arXiv:1809.03981*.

[Buterin 2016] Buterin, V. (2016). Chain interoperability. *R3 research paper*, 9:1–25.

[Buterin et al. 2014] Buterin, V. et al. (2014). A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1.

[Caldarelli and Ellul 2021] Caldarelli, G. and Ellul, J. (2021). The blockchain oracle problem in decentralized finance—a multivocal approach. *Applied Sciences*, 11(16):7572.

[Cao et al. 2021] Cao, Y., Zou, C., and Cheng, X. (2021). Flashot: a snapshot of flash loan attack on defi ecosystem. *arXiv preprint arXiv:2102.00626*.

[Casino et al. 2019] Casino, F., Dasaklis, T. K., and Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and informatics*, 36:55–81.

[Chaliasos et al. 2024] Chaliasos, S., Charalambous, M. A., Zhou, L., Galanopoulou, R., Gervais, A., Mitropoulos, D., and Livshits, B. (2024). Smart contract and defi security tools: Do they meet the needs of practitioners? In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13.

[Chen et al. 2021] Chen, Y., Richter, J. I., and Patel, P. C. (2021). Decentralized governance of digital platforms. *Journal of Management*, 47(5):1305–1337.

[Chohan 2021] Chohan, U. W. (2021). The double spending problem and cryptocurrencies. *Available at SSRN 3090174*.

[ConsenSys 2024] ConsenSys (2024). Mythril: A security analysis tool for evm bytecode.

[Conti et al. 2018] Conti, M., Kumar, E. S., Lal, C., and Ruj, S. (2018). A survey on security and privacy issues of bitcoin. *IEEE communications surveys & tutorials*, 20(4):3416–3452.

[Costa et al. 2023] Costa, D., La Cava, L., and Tagarelli, A. (2023). Show me your nft and i tell you how it will perform: Multimodal representation learning for nft selling price prediction. In *Proceedings of the ACM Web Conference 2023*, pages 1875–1885.

[David et al. 2023] David, I., Zhou, L., Qin, K., Song, D., Cavallaro, L., and Gervais, A. (2023). Do you still need a manual smart contract audit? *arXiv preprint arXiv:2306.12338*.

[Deshpande and Herlihy 2020] Deshpande, A. and Herlihy, M. (2020). Privacy-preserving cross-chain atomic swaps. In *International Conference on Financial Cryptography and Data Security*, pages 540–549. Springer.

[Ding et al. 2021] Ding, M., Li, P., Li, S., and Zhang, H. (2021). Hfcontractfuzzer: Fuzzing hyperledger fabric smart contracts for vulnerability detection. In *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering*, pages 321–328.

[Entriken et al. 2018] Entriken, W., Shirley, D., Evans, J., and Sachs, N. (2018). Ethereum improvement proposal 721.

[Eskandari et al. 2020] Eskandari, S., Moosavi, S., and Clark, J. (2020). Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 170–189. Springer.

[Frost et al. 2019] Frost, J., Gambacorta, L., Huang, Y., Shin, H. S., and Zbinden, P. (2019). Bigtech and the changing structure of financial intermediation. *Economic policy*, 34(100):761–799.

[Ghaleb et al. 2023] Ghaleb, A., Rubin, J., and Pattabiraman, K. (2023). Achecker: Statically detecting smart contract access control vulnerabilities. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 945–956. IEEE.

[Gilbert and Lynch 2002] Gilbert, S. and Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59.

[Goes 2020] Goes, C. (2020). The interblockchain communication protocol: An overview. *arXiv preprint arXiv:2006.15918*.

[Goetze 2023] Goetze, C. (2023). Stablecoins: o que são e quais os tipos? `https://hubdoinvestidor.com.br/stablecoins-o-que-sao-e-quais-os-tipos/`. (Accessed on 05/23/2024).

[Greve et al. 2018] Greve, F., Sampaio, L., Abijaude, J., Coutinho, A. A., Brito, I., and Queiroz, S. (2018). Blockchain e a Revolução do Consenso sob Demanda. In *Proc. of SBRC Minicursos*.

[Harvey et al. 2021] Harvey, C. R., Ramachandran, A., and Santoro, J. (2021). *DeFi and the Future of Finance*. John Wiley & Sons.

[Harz and Knottenbelt 2018] Harz, D. and Knottenbelt, W. (2018). Towards safer smart contracts: A survey of languages and verification methods. *arXiv preprint arXiv:1809.09805*.

[Hassan and De Filippi 2021] Hassan, S. and De Filippi, P. (2021). Decentralized autonomous organization. *Internet Policy Review*, 10(2):1–10.

[Helliar et al. 2020] Helliar, C. V., Crawford, L., Rocca, L., Teodori, C., and Veneziani, M. (2020). Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54:102136.

[Ivanov et al. 2023] Ivanov, N., Li, C., Yan, Q., Sun, Z., Cao, Z., and Luo, X. (2023). Security threat mitigation for smart contracts: A comprehensive survey. *ACM Computing Surveys*, 55(14s):1–37.

[Jiang et al. 2018] Jiang, B., Liu, Y., and Chan, W. K. (2018). Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, pages 259–269.

[Juliano 2018] Juliano, A. (2018). dydx: A standard for decentralized margin trading and derivatives. *URl: https://whitepaper. dydx. exchange*.

[Kalra et al. 2018] Kalra, S., Goel, S., Dhawan, M., and Sharma, S. (2018). Zeus: analyzing safety of smart contracts. In *Ndss*, pages 1–12.

[Kushwaha et al. 2022] Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., and Lee, H.-N. (2022). Ethereum smart contract analysis tools: A systematic review. *Ieee Access*, 10:57037–57062.

[Kwon and Buchman 2019] Kwon, J. and Buchman, E. (2019). Cosmos whitepaper. *A Netw. Distrib. Ledgers*, 27:1–32.

[Lamport et al. 2019] Lamport, L., Shostak, R., and Pease, M. (2019). *The Byzantine Generals Problem*, page 203–226. Association for Computing Machinery, New York, NY, USA.

[Lerner et al. 2022] Lerner, S. D., Cid-Fuentes, J. Á., Len, J., Fernàndez-València, R., Gallardo, P., Vescovo, N., Laprida, R., Mishra, S., Jinich, F., and Masini, D. (2022). Rsk: A bitcoin sidechain with stateful smart-contracts. *Cryptology ePrint Archive*.

[Li et al. 2021] Li, M., Weng, J., Li, Y., Wu, Y., Weng, J., Li, D., Xu, G., and Deng, R. (2021). Ivycross: A privacy-preserving and concurrency control framework for blockchain interoperability. *Cryptology ePrint Archive*.

[Li et al. 2022] Li, P., Li, S., Ding, M., Yu, J., Zhang, H., Zhou, X., and Li, J. (2022). A vulnerability detection framework for hyperledger fabric smart contracts based on dynamic and static analysis. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, pages 366–374.

[Liao et al. 2022] Liao, Z., Zheng, Z., Chen, X., and Nan, Y. (2022). Smartdagger: a bytecode-based static analysis approach for detecting cross-contract vulnerability. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 752–764.

[Linoy et al. 2021] Linoy, S., Ray, S., and Stakhanova, N. (2021). Etherprov: Provenance-aware detection, analysis, and mitigation of ethereum smart contract security issues. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 1–10. IEEE.

[Liu et al. 2023] Liu, Z., Qian, P., Yang, J., Liu, L., Xu, X., He, Q., and Zhang, X. (2023). Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting. *IEEE Transactions on Information Forensics and Security*, 18:1237–1251.

[Luu et al. 2016] Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269.

[Ma et al. 2019] Ma, C., Kong, X., Lan, Q., and Zhou, Z. (2019). The privacy protection mechanism of hyperledger fabric and its application in supply chain finance. *Cybersecurity*, 2(1):1–9.

[Mendonça et al. 2024] Mendonça, R. D., Cardoso, I. W. F., Coelho, R., Campos, J. N., Gonçalves, G. D., Vieira, A. B., and Nacif, J. A. (2024). Mecanismos de interoperabilidade em blockchains: Um comparativo de custo de transações cross-chain para tokens

erc-20. In *Anais do VII Workshop em Blockchain: Teoria, Tecnologias e Aplicações*. SBC.

[Messias et al. 2023] Messias, J., Pahari, V., Chandrasekaran, B., Gummadi, K. P., and Loiseau, P. (2023). Understanding blockchain governance: Analyzing decentralized voting to amend defi smart contracts.

[Mossberg et al. 2019] Mossberg, M., Manzano, F., Hennenfent, E., Groce, A., Grieco, G., Feist, J., Brunson, T., and Dinaburg, A. (2019). Manticore: A user-friendly symbolic execution framework for binaries and smart contracts. In *2019 34th IEEE/ACM (ASE)*, pages 1186–1189. IEEE.

[Murray et al. 2023] Murray, A., Kim, D., and Combs, J. (2023). The promise of a decentralized internet: What is web3 and how can firms prepare? *Business Horizons*, 66(2):191–202.

[Nadini et al. 2021] Nadini, M., Alessandretti, L., Di Giacinto, F., Martino, M., Aiello, L. M., and Baronchelli, A. (2021). Mapping the nft revolution: market trends, trade networks, and visual features. *Scientific reports*, 11(1):1–11.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[Nakamoto and Bitcoin 2008] Nakamoto, S. and Bitcoin, A. (2008). A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4.

[Okonkwo 2021] Okonkwo, I. E. (2021). Nft, copyright; and intellectual property commercialisation. *SSRN*. https://ssrn.com/abstract=3856154.

[Ongaro and Ousterhout 2014] Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*, pages 305–319.

[Ou et al. 2022] Ou, W., Huang, S., Zheng, J., Zhang, Q., Zeng, G., and Han, W. (2022). An overview on cross-chain: Mechanism, platforms, challenges and advances. *Computer Networks*.

[Palma et al. 2022] Palma, L., Martina, J., and Vigil, M. (2022). On and off: Extracting the transaction history of permissioned blockchain networks. Universidade Federal de Santa Catarina. Computing Science PhD Candidate Dissertation.

[Product 2024] Product (2024). dYdX v4 - full decentralization. `https://dydx.exchange/blog/v4-full-decentralization`. (Accessed on 05/20/2024).

[Qin et al. 2021] Qin, K., Zhou, L., and Gervais, A. (2021). Quantifying blockchain extractable value: How dark is the forest? *CoRR*, abs/2101.05511.

[Ren et al. 2023] Ren, K., Ho, N.-M., Loghin, D., Nguyen, T.-T., Ooi, B. C., Ta, Q.-T., and Zhu, F. (2023). Interoperability in blockchain: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

[Rodler et al. 2023] Rodler, M., Paaßen, D., Li, W., Bernhard, L., Holz, T., Karame, G., and Davi, L. (2023). Ef cf: High performance smart contract fuzzing for exploit generation. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 449–471. IEEE.

[Rouhani and Deters 2019] Rouhani, S. and Deters, R. (2019). Security, performance, and applications of smart contracts: A systematic survey. *IEEE Access*, 7:50759–50779.

[Sabt et al. 2015] Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, volume 1, pages 57–64. IEEE.

[Sanches and Diniz 2024] Sanches, B. H. and Diniz, E. (2024). From crypto-libertarian utopia to central bank digital currencies: The transfigurative convergence of bitcoin's prefiguration. *Available at SSRN 4818428*.

[Sayeed et al. 2020] Sayeed, S., Marco-Gisbert, H., and Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8:24416–24427.

[Schär 2021] Schär, F. (2021). Decentralized finance: On blockchain-and smart contract-based financial markets. *FRB of St. Louis Review*.

[Singh et al. 2020] Singh, A., Click, K., Parizi, R. M., Zhang, Q., Dehghantanha, A., and Choo, K.-K. R. (2020). Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471.

[Siris et al. 2019] Siris, V. A., Nikander, P., Voulgaris, S., Fotiou, N., Lagutin, D., and Polyzos, G. C. (2019). Interledger approaches. *Ieee Access*, 7:89948–89966.

[Song et al. 2022] Song, K., Matulevicius, N., de Lima Filho, E. B., and Cordeiro, L. C. (2022). Esbmc-solidity: An smt-based model checker for solidity smart contracts. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pages 65–69.

[Sun et al. 2021] Sun, X., Yu, F. R., Zhang, P., Sun, Z., Xie, W., and Peng, X. (2021). A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205.

[Szabo 1997] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First monday*.

[Teixeira 2023] Teixeira, D. (2023). O caminho para o real digital. *Revista LIFT papers*, 5(5).

[Thomas and Schwartz 2015] Thomas, S. and Schwartz, E. (2015). A protocol for interledger payments. *URL https://interledger. org/interledger. pdf*.

[Tikhomirov 2018] Tikhomirov, S. (2018). Ethereum: state of knowledge and research perspectives. In *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10*, pages 206–221. Springer.

[Tikhomirov et al. 2018] Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., and Alexandrov, Y. (2018). Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, pages 9–16.

[Torres et al. 2021] Torres, C. F., Camino, R., and State, R. (2021). Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1343–1359. USENIX Association.

[Varun et al. 2022] Varun, M., Palanisamy, B., and Sural, S. (2022). Mitigating frontrunning attacks in ethereum. In *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure*, BSCI '22, page 115–124. Association for Computing Machinery.

[Wang et al. 2021] Wang, D., Wu, S., Lin, Z., Wu, L., Yuan, X., Zhou, Y., Wang, H., and Ren, K. (2021). Towards a first step to understand flash loan and its applications in defi ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, pages 23–28.

[Wang et al. 2020] Wang, W., Song, J., Xu, G., Li, Y., Wang, H., and Su, C. (2020). Contractward: Automated vulnerability detection models for ethereum smart contracts. *IEEE Transactions on Network Science and Engineering*.

[Wang et al. 2022] Wang, Y., Zuest, P., Yao, Y., Lu, Z., and Wattenhofer, R. (2022). Impact and user perception of sandwich attacks in the defi ecosystem. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. Association for Computing Machinery.

[Wegner 1996] Wegner, P. (1996). Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287.

[Weintraub et al. 2022] Weintraub, B., Torres, C. F., Nita-Rotaru, C., and State, R. (2022). A flash(bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, page 458–471. Association for Computing Machinery.

[Werner et al. 2022] Werner, S., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., and Knottenbelt, W. (2022). Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46.

[White et al. 2022] White, B., Mahanti, A., and Passi, K. (2022). Characterizing the opensea nft marketplace. In *Companion Proceedings of the Web Conference 2022*, pages 488–496.

[Wood 2014] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32.

[Wu et al. 2021] Wu, S., Wang, D., He, J., Zhou, Y., Wu, L., Yuan, X., He, Q., and Ren, K. (2021). Defiranger: Detecting price manipulation attacks on defi applications. *arXiv preprint arXiv:2104.15068*.

[Wüstholz and Christakis 2020] Wüstholz, V. and Christakis, M. (2020). Harvey: A greybox fuzzer for smart contracts. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1398–1409.

[Xu et al. 2023] Xu, X., Hu, T., Li, B., and Liao, L. (2023). Ccdetector: Detect chaincode vulnerabilities based on knowledge graph. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 699–704. IEEE.

[Xu et al. 2019] Xu, X., Weber, I., and Staples, M. (2019). *Architecture for blockchain applications*. Springer.

[Xu et al. 2021] Xu, Y., Hu, G., You, L., and Cao, C. (2021). A novel machine learning-based analysis model for smart contract vulnerability. *Security and Communication Networks*, 2021:1–12.

[Yadav and Naval 2023] Yadav, K. and Naval, S. (2023). Cfg analysis for detecting vulnerabilities in smart contracts. In *International Conference on Smart Trends for Information Technology and Computer Communications*, pages 753–763. Springer.

[Yamashita et al. 2019] Yamashita, K., Nomura, Y., Zhou, E., Pi, B., and Jun, S. (2019). Potential risks of hyperledger fabric smart contracts. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 1–10. IEEE.

[Yan et al. 2022] Yan, X., Wang, S., and Gai, K. (2022). A semantic analysis-based method for smart contract vulnerability. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity)*, pages 23–28. IEEE.

[Ye et al. 2020] Ye, J., Ma, M., Lin, Y., Sui, Y., and Xue, Y. (2020). Clairvoyance: Cross-contract static analysis for detecting practical reentrancy vulnerabilities in smart contracts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, pages 274–275.

[Zeller et al. 2019] Zeller, A., Gopinath, R., Böhme, M., Fraser, G., and Holler, C. (2019). The fuzzing book.

[Zhang et al. 2023a] Zhang, P., Yu, Q., Xiao, Y., Dong, H., Luo, X., Wang, X., and Zhang, M. (2023a). Bian: Smart contract source code obfuscation. *IEEE Transactions on Software Engineering*.

[Zhang et al. 2023b] Zhang, W., Wei, L., Cheung, S.-C., Liu, Y., Li, S., Liu, L., and Lyu, M. R. (2023b). Combatting front-running in smart contracts: Attack mining, benchmark construction and vulnerability detector evaluation. *IEEE Transactions on Software Engineering*, 49(6):3630–3646.

[Zhang et al. 2023c] Zhang, Y., Liu, P., Wang, G., Li, P., Gu, W., Chen, H., Liu, X., and Zhu, J. (2023c). Frad: Front-running attacks detection on ethereum using ternary classification model. *arXiv preprint arXiv:2311.14514*.

[Zhou et al. 2021] Zhou, L., Qin, K., Torres, C. F., Le, D. V., and Gervais, A. (2021). High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE.

[Zou et al. 2019] Zou, W., Lo, D., Kochhar, P. S., Le, X.-B. D., Xia, X., Feng, Y., Chen, Z., and Xu, B. (2019). Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*.

[Zwitter and Hazenberg 2020] Zwitter, A. and Hazenberg, J. (2020). Decentralized network governance: blockchain technology and the future of regulation. *Frontiers in Blockchain*, 3:12.

Capítulo 2

Aprendizado Auto-Supervisionado Generativo e Contrastivo: Tendências e Desafios para Aplicações Dinâmicas em Redes

João Vitor Valle Silva, Guilherme Nunes Nasseh Barbosa,
Willian Tessaro Lunardi, Martin Andreoni,
Diogo Menezes Ferrazani Mattos

Abstract

This chapter addresses and contextualizes self-supervised learning as an alternative for dynamic network applications, where data labeling is a critical challenge due to the discrepancy between the traffic generation rate and the manual data labeling rate. Generative and contrastive self-supervised learning techniques come to the fore because they effectively improve network performance, expand the number of labeled samples, and enable recognition of similarities and differences between sample examples. Finally, self-supervised learning algorithms and their characteristics and network applications are presented, aiming to enable readers to understand this technique's principles, frameworks, and limitations.

Resumo

Este capítulo aborda e contextualiza o aprendizado auto-supervisionado como uma alternativa para aplicações dinâmicas de rede, em que a rotulagem de dados é um desafio crítico devido à discrepância entre a taxa de geração de tráfego e a taxa de rotulagem manual dos dados. São apresentadas técnicas de aprendizado auto-supervisionado generativo e contrastivo por serem eficazes para melhorar o desempenho da rede, expandindo o número de amostras rotuladas e reconhecendo semelhanças e diferenças entre exemplos de amostras. Por fim, são apresentados os algoritmos de aprendizado auto-supervisionado e suas características e aplicações em redes, visando capacitar os leitores a compreender os princípios, arcabouços e limitações dessa técnica.

2.1. Introdução

A computação ubíqua é parte fundamental do cotidiano das pessoas, principalmente na utilização de dispositivos inteligentes, como sensores, *wearables* e telefones inteligentes (*smartphones*), que compõem a Internet das Coisas. Para que a Internet das Coisas possa integrar cada vez mais dispositivos heterogêneos, é necessário desenvolver sistemas capazes de obter informações por meio da detecção e coleta de dados para controle e gerenciamento de múltiplas redes [Zafar et al., 2022]. Com bilhões de dispositivos conectados, a integração e interconexão entre eles representam um desafio para o gerenciamento de tráfego e otimização da rede. As redes móveis, tais como as redes de quinta geração (5G) e as previsões das redes de sexta geração (6G), são fundamentais para conectar dispositivos da Internet das Coisas. No entanto, um dos principais desafios para as redes 5G e além é suportar simultaneamente a implementação de Qualidade de Serviço (QoS) em um ambiente altamente heterogêneo, composto por diversos tipos de tráfego e aplicações adaptadas a diferentes requisitos, sob recursos de rede limitados e condições de rede dinâmicas [Zhang e Zhu, 2023]. O dinamismo existente nessas redes ainda é um desafio para aprimorar a implementação de uma arquitetura baseada em virtualização das funções de redes e redes definidas por software. Outro ponto importante é a adoção de uma arquitetura *Zero Trust*, que requer mecanismos para gerenciar eventos e informações de segurança (*Security Information and Event Management* - SIEM), além do armazenamento de registros de atividades (*logs*) de sistemas e redes [Stafford, 2020]. Para a análise dessas informações, é essencial que sejam utilizados algoritmos de aprendizado de máquina.

O aprendizado de máquina abrange vários paradigmas, cada um com características e aplicações distintas. Os quatro tipos principais de aprendizado de máquina são: (i) supervisionado, (ii) não-supervisionado, (iii) semi-supervisionado e (iv) auto-supervisionado. O aprendizado supervisionado envolve modelos de treinamento em conjuntos de dados rotulados, em que cada exemplo de treinamento está associado a um rótulo de saída. Em contraste, as tarefas de aprendizado não-supervisionado visam identificar padrões ou estruturas subjacentes em conjuntos de dados não rotulados. O aprendizado semi-supervisionado combina elementos de aprendizado supervisionado e não-supervisionado, utilizando uma pequena quantidade de dados rotulados juntamente com um conjunto maior de dados não rotulados. Por fim, o aprendizado auto-supervisionado é uma forma de aprendizado não-supervisionado em que o modelo gera rótulos a partir dos próprios dados, normalmente através de tarefas de pretexto, para aprender representações úteis. Cada tipo de aprendizado tem suas vantagens e desvantagens, tornando-os adequados para diferentes cenários e aplicações na área de aprendizado de máquina. A Tabela 2.1 apresenta as principais características de cada tipo de aprendizado de máquina.

Dentre as diferentes abordagens de aprendizado de máquina, as Redes Neurais Artificiais demonstram seu potencial diariamente em diversas aplicações, especialmente em tarefas de aprendizado supervisionado. O uso de redes neurais oferece a capacidade de processar grandes quantidades de dados, tornando tarefas como classificação de imagens, segmentação semântica, processamento de linguagem natural e aprendizado de grafos mais acessíveis e eficientes [Thisanke et al., 2023]. Embora os modelos de aprendizado supervisionado tenham se tornado ferramentas valiosas em diversas áreas, sua aplicação em problemas que envolvem dados não relacionais, como fluxos de rede, ainda apresenta

Tabela 2.1. Comparação entre aprendizado supervisionado, não-supervisionado, semi-supervisionado e auto-supervisionado

	Aprendizado Supervisionado	**Aprendizado Não Supervisionado**	**Aprendizado Semi-Supervisionado**	**Aprendizado Auto-Supervisionado**
Definição	Treina em dados rotulados	Treina em dados não rotulados	Treina com uma pequena quantidade de dados rotulados e uma grande quantidade de dados não rotulados	Gera rótulos a partir dos próprios dados e treina com esses rótulos gerados automaticamente
Requisitos de Dados	Grande quantidade de dados rotulados	Apenas dados não rotulados	Combinação de dados rotulados e não rotulados	Dados não rotulados com sinais de supervisão gerados
Treinamento	Aprende a mapear entradas para saídas com base nos rótulos fornecidos	Encontra padrões ou estruturas nos dados	Usa dados rotulados para guiar o aprendizado e captura padrões dos dados não rotulados	Resolve tarefas pretextuais criadas a partir dos dados para aprender representações úteis
Aplicações	Classificação, regressão, etc.	Agrupamento, redução de dimensionalidade, detecção de anomalias	Situações com escassez de dados rotulados (por exemplo, imagens médicas)	Pré-treinamento de modelos para tarefas de PLN e visão computacional
Exemplos	Detecção de spam em emails, previsão de preços de casas	Segmentação de clientes, PCA	Classificação de imagens com poucos exemplos rotulados	BERT para PLN, SimCLR para visão computacional
Vantagens	Previsões precisas com dados rotulados suficientes	Descobre padrões ocultos sem dados rotulados	Combina as vantagens do aprendizado supervisionado e não supervisionado	Aprende com dados não rotulados em grande escala, útil para pré-treinamento
Desvantagens	Requer grandes conjuntos de dados rotulados, que podem ser caros	Nem sempre encontra padrões úteis	Ainda requer alguns dados rotulados	Tarefas pretextuais complexas podem não estar sempre alinhadas com as tarefas posteriores

desafios. Esses modelos são altamente dependentes de conjuntos de dados rotulados, o que os torna suscetíveis a erros de generalização. A coleta de dados rotulados ou a rotulagem manual é difícil em diversos os aspectos. Treinar uma rede do zero é uma tarefa com alto custo computacional [Thisanke et al., 2023]. Por outro lado, os modelos auto-supervisionados têm se destacado como tendências de pesquisas recentes devido à sua eficiência em lidar com uma vasta quantidade de dados não rotulados e à sua alta capacidade de generalização. Ressalta-se que parte dos dados utilizados para o treinamento auto-supervisionado pode estar incompleta, ter sofrido transformações, como distorções ou traduções, ou estar parcialmente corrompida, como ocorre em algumas técnicas de aumento de dados (*data augmentation*). Nesses casos, o modelo aprende a recuperar a parte que sofreu o dano, todo o conjunto de dados ou simplesmente algumas características de interesse.

O aprendizado auto-supervisionado é visto como um ramo do aprendizado não-supervisionado, pois não há rótulo prévio associado às amostras. No entanto, o aprendizado não-supervisionado concentra-se em detectar padrões específicos nos dados, como agrupamentos, descoberta de comunidades ou detecção de anomalias, enquanto o aprendizado auto-supervisionado visa recuperar informações e, portanto, está alinhado ao paradigma de aprendizado supervisionado. O aprendizado auto-supervisionado pode ser categorizado de acordo com o treinamento realizado, sendo dividido em quatro categorias [Wu et al., 2023]:

- **Contrastivo**. A ideia central dos modelos contrastivos reside em tratar cada instância como uma classe distinta. As variantes da mesma instância são aproximadas no espaço de incorporação, enquanto as variantes de instâncias diferentes são separadas. Essas variantes são criadas através da aplicação de diferentes transformações nos dados originais;

- **Generativo**. Modelos generativos utilizam uma tarefa auto-supervisionada, em que o perfil original da instância é reconstruído a partir de versões corrompidas. O modelo é treinado para prever uma parte dos dados disponíveis, sendo as tarefas mais comuns a reconstrução da estrutura e das características;

- **Preditivo**. Embora os modelos preditivos e generativos possam parecer semelhantes, devido ao envolvimento de previsões em ambos, seus objetivos subjacentes divergem significativamente. Os métodos generativos direcionam seus esforços para a previsão de partes ausentes nos dados originais, o que pode ser interpretado como uma forma de autoprevisão. Em contrapartida, os métodos preditivos geram novas amostras ou rótulos a partir dos dados originais, visando auxiliar nas tarefas de pretexto;

- **Híbrido**. Combinar diversas tarefas auto-supervisionadas e integrá-las em um único modelo configura-se como uma estratégia viável. Essa abordagem híbrida geralmente demanda a utilização de múltiplos codificadores. Diferentes tarefas auto-supervisionadas podem ser executadas em paralelo ou colaborar entre si.

Os modelos generativos tiveram uma grande influência a partir do uso das Redes Adversariais Generativas (*Generative Adversarial Networks* - GANs). As GANs fornecem uma maneira de aprender representações profundas sem dados de treinamento extensivamente rotulados. Isso é possível derivando sinais de retropropagação por meio de um processo competitivo envolvendo um par de redes neurais: um gerador e um discriminador. O gerador é responsável por gerar dados, enquanto o discriminador é utilizado para distinguir entre os dados reais e gerados. Esse processo competitivo leva as duas redes a melhorar continuamente seu desempenho, resultando em representações profundas que capturam as características essenciais dos dados [Creswell et al., 2018]. No entanto, as Redes Adversárias Generativas (GANs) apresentam desafios particulares durante o treinamento dos modelos. Isso se deve à não convergência dos parâmetros, que oscilam de forma significativa. Além disso, o discriminador pode ser tão eficiente que impede a rede geradora de criar dados próximos da realidade, interrompendo o treinamento [Jaiswal et al., 2021].

O modelo contrastivo, por sua vez, configura-se como uma abordagem discriminativa. Seu objetivo é agrupar amostras semelhantes, aproximando-as entre si, enquanto afasta amostras distintas umas das outras [Jaiswal et al., 2021]. Nas abordagens discriminativas, as representações são aprendidas por meio da modelagem da distribuição condicional $p(y|x)$. Esse processo envolve duas etapas principais: a inferência, em que os valores das variáveis latentes $p(v|x)$ são inferidos a partir da entrada x, e a tomada de decisão, em que, com base nas variáveis latentes inferidas v, a decisão final sobre o rótulo y é tomada, representada por $p(y|v)$ [Le-Khac et al., 2020].

Este capítulo aborda os paradigmas de aprendizado auto-supervisionado, tanto generativo quanto contrastivo, e discute a aplicação desse tipo de aprendizado em atividades complexas nas redes de computadores, como Sistemas de Detecção de Intrusão e Sistemas Automatizados de Provisão de Qualidade de Serviço. Um dos principais desafios na aplicação do aprendizado auto-supervisionado ao tráfego de rede para detecção de anomalias reside na representação significativa dos dados. Mesmo com dados previamente rotulados, a detecção de anomalias continua sendo um grande desafio devido à constante criação de novos ataques, que podem se camuflar facilmente em grandes fluxos de rede. Por outro lado, a provisão de qualidade de serviço (QoS) enfrenta ambientes cada vez mais dinâmicos e heterogêneos, exigindo que o aprendizado auto-supervisionado seja parametrizado da maneira mais eficiente possível. O capítulo discute, então, as principais propostas de implementação de aprendizado auto-supervisionado, detalhando tarefas de pretexto e ressaltando a aplicabilidade das técnicas abordadas.

O restante do capítulo está organizado da seguinte forma. A Seção 2.2 apresenta tarefas de pretexto para a transformação dos dados e técnicas para a maximização da informação. O aprendizado auto-supervisionado generativo é discutido na Seção 2.3, enquanto o aprendizado auto-supervisionado contrastivo é abordado na Seção 2.4. A Seção 2.5 explora propostas de aprendizado auto-supervisionado baseadas em agrupamentos. Casos de uso de aprendizado auto-supervisionado em aplicações dinâmicas de redes de computadores são detalhados na Seção 2.6. As tendências de pesquisa, oportunidades e desafios são discutidos na Seção 2.7. A atividade prática proposta é descrita na Seção 2.8. Por fim, a Seção 2.9 conclui este capítulo.

2.2. Modelos de transformação e maximização da informação

Os modelos de transformação e a maximização da informação são importantes para compreensão do aprendizado auto-supervisionado. Os modelos de transformação referem-se a arquiteturas que podem gerar diversas transformações de dados, como rotações, translações ou mudanças de cores, aplicadas aos dados de entrada. Essas transformações são utilizadas como tarefas de pretexto no aprendizado auto-supervisionado, em que o modelo é treinado para prever a transformação aplicada aos dados de entrada. Tarefas de pretexto no aprendizado auto-supervisionado são tarefas artificialmente criadas que geram sinais de supervisão a partir de dados não rotulados, permitindo que modelos aprendam representações úteis dos dados. Exemplos comuns incluem a predição de rotação, na qual o modelo prevê a rotação aplicada a uma imagem; o preenchimento de partes faltantes, em que o modelo completa imagens ou textos com lacunas inseridas artificialmente; e a ordenação de blocos menores (*patches*) da imagem, em que o modelo reordena blocos embaralhados para a estrutura original. Essas tarefas ajudam o modelo a entender características significativas dos dados, que podem ser transferidas para outras tarefas supervisionadas, como classificação ou detecção de anomalias, facilitando o treinamento em grandes volumes de dados não rotulados.

A maximização da informação, por outro lado, é um princípio que orienta o processo de aprendizagem, incentivando o modelo a extrair o máximo possível de informações úteis dos dados de entrada. No aprendizado auto-supervisionado, a maximização da informação é alcançada através da concepção de tarefas de pretexto que exigem que o modelo capture características ou representações significativas dos dados de entrada. Ao

combinar modelos de transformação com maximização de informações, as abordagens de aprendizado auto-supervisionado podem efetivamente aproveitar dados não rotulados para aprender representações úteis sem exigir anotação manual.

Muitos métodos de aprendizado de representação auto-supervisionados fazem uso de transformações de imagem. Redes de Quebra-Cabeça e de Rotação aplicam transformações selecionadas a exemplos de imagem com o objetivo de prever a parametrização da transformação. Em contraste, outros métodos se concentram em aprender representações que são invariantes a certas transformações, o que pode levar a um fenômeno conhecido como colapso de representação. Este colapso descreve soluções triviais, como representações constantes, que atendem ao objetivo de invariância, mas oferecem pouco ou nenhum valor informativo para tarefas reais.

Para evitar esse colapso de representação, foram desenvolvidos os chamados métodos de maximização de informação, que formam uma classe de técnicas de representação focadas no conteúdo informativo das incorporações. Por exemplo, alguns desses métodos descorrelacionam explicitamente todos os elementos dos vetores de incorporação, evitando efetivamente o colapso e resultando em uma maximização indireta do conteúdo de informação. Esses métodos utilizam técnicas como a matriz de correlação cruzada normalizada de incorporações entre visualizações [Zbontar et al., 2021], a matriz de covariância para visualizações únicas [Bardes et al., 2022] e operações de embranquecimento para implementar essa abordagem [Ermolov et al., 2021]. A seguir, esses métodos são elencados e detalhados. Ressalta-se que grande parte dos trabalhos consideram imagens como dados de entrada. Sendo assim, neste capítulo os dados de entrada dos modelos são considerados também como imagens, a menos que seja explicitamente definido a natureza dos dados de entrada dos modelos considerados.

2.2.1. *Barlow Twins*

A ideia central por trás desse arcabouço é o princípio da redução de redundância. Esse princípio, introduzido por Barlow em 1961, afirma que a redução de redundância é crucial para a organização de mensagens sensoriais no cérebro. A estrutura por trás dessa técnica é ilustrada na Figura 2.1.

Para implementar esse princípio de redução de redundância, a abordagem *Barlow Twins* utiliza um conjunto de imagens X e cria duas visualizações $X(1) = t(X)$ e $X(2) = t(X)$ dessas imagens, em que $t \sim T$ é uma transformação que é amostrada aleatoriamente de T para cada imagem e cada visualização. Um codificador f_θ computa representações $Y(1) = f_\theta(X(1))$ e $Y(2) = f_\theta(X(2))$, que são alimentadas em um projetor g_ϕ para calcular projeções $Z(1) = [z(1)_1, ..., z(1)_n] = g_\phi(Y(1))$ e $Z(2) = [z(2)_1, ..., z(2)_n] = g_\phi(Y(2))$ para ambas as visualizações.

A ideia dos Barlow Twins é regularizar a matriz de correlação cruzada entre as projeções de ambas as visualizações. A matriz é calculada como

$$C = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{z(1)_i - \mu(1)}{\sigma(1)} \right) \left(\frac{z(2)_i - \mu(2)}{\sigma(2)} \right)^\top,$$

em que $\mu(j)$ e $\sigma(j)$ são a média e o desvio padrão sobre o conjunto de projeções da

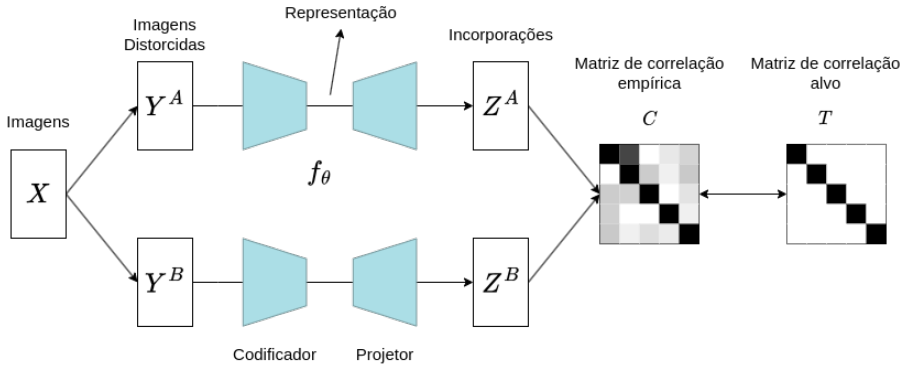


Figura 2.1. A técnica *Barlow Twins* busca tornar as representações obtidas a partir das redes neurais similares ao alimentá-las com versões distorcidas de um conjunto de dados. Isso reduz a redundância entre os componentes dos vetores de incorporação (*embedding*). É competitivo com outros métodos de autoaprendizagem, é simples conceitualmente, evita incorporações constantes triviais e é robusto ao tamanho do lote de treinamento. Adaptado de [Zbontar et al., 2021].

j-ésima visualização, calculadas como

$$\mu(j) = \frac{1}{n}\sum_{i=1}^{n} z(j)_i,$$

$$\sigma(j) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(z(j)_i - \mu(j))^2}.$$

A função de perda é então definida como

$$L_{BT}(\theta,\phi) = \sum_{k=1}^{d}(1-C[k,k])^2 + \lambda\sum_{k=1}^{d}\sum_{k'\neq k} C[k,k']^2,$$

em que d é o número de dimensões da projeção e $\lambda > 0$ é um hiperparâmetro. O parâmetro d promove invariância em relação às transformações aplicadas enquanto $\lambda > 0$ descorrelaciona as representações aprendidos, ou seja, reduz a redundância. Ao utilizar esta perda, o codificador f_θ é incentivado a prever representações que são descorrelacionadas, logo, que não são redundantes. Os *Barlow Twins* são treinados utilizando o otimizador LARS [You et al., 2017].

2.2.2. VICReg

VICReg (*Variance-Invariance-Covariance Regularization*) [Bardes et al., 2022] é um modelo auto-supervisionado de incorporação conjunta que se enquadra na categoria de métodos de maximização da informação. A proposta visa maximizar o acordo entre representações de diferentes visualizações de uma entrada, enquanto previne o colapso informacional usando dois termos de regularização adicionais. A Figura 2.2 representa a arquitetura básica da técnica VICReg, apresentando os seus principais elementos e operações.

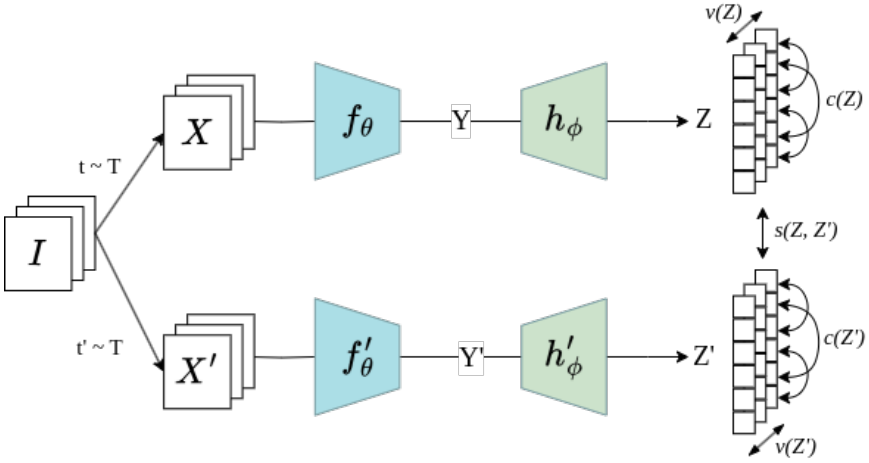


Figura 2.2. A técnica VICReg produz dois conjuntos de visualizações a partir de um lote de imagens, codifica essas visualizações em representações e, então, expande essas representações em incorporações. O objetivo é minimizar a distância entre incorporações da mesma imagem, manter a variância de cada variável de incorporação acima de um limite e atrair a covariância entre pares de variáveis de incorporação para zero. Adaptado de [Bardes et al., 2022].

Especificamente, a VICReg define termos de regularização para variância, invariância e covariância. Dado um lote de imagens X, duas visualizações $X(1) = t(X)$ e $X(2) = t(X)$ são definidas, onde $t \sim T$ é, novamente, amostrado aleatoriamente de T para cada imagem e cada visualização. Um codificador Siamese f_θ computa representações $Y(1) = f_\theta(X(1))$ e $Y(2) = f_\theta(X(2))$, que são alimentadas em um projetor Siamese g_ϕ para calcular projeções $Z(1) = [z(1)_1, ..., z(1)_n] = g_\phi(Y(1))$ e $Z(2) = [z(2)_1, ..., z(2)_n] = g_\phi(Y(2))$. Cada projeção possui d dimensões. Para cada visualização, a matriz de covariância das projeções é computada.

O termo de variância visa manter o desvio padrão de cada elemento do incorporação acima de uma margem b. Praticamente, isso impede que os vetores de incorporação sejam os mesmos em todo o lote e é um dos dois mecanismos que visam prevenir o colapso. Pode ser implementado usando uma perda de bisel. O termo de covariância descorrelaciona os elementos dos vetores de incorporação para visualizações únicas a fim de reduzir a redundância e evitar o colapso. Isso é alcançado minimizando os elementos fora da diagonal ao quadrado da matriz de covariância em direção a 0. Por fim, o termo de invariância é usado para maximizar o acordo entre duas projeções da mesma imagem, induzindo invariância às transformações aplicadas a x_i. Para isso, é calculado o erro quadrático médio entre as projeções. Em geral, a perda da VICReg pode ser definida como a soma ponderada de todas as três regularizações para as visualizações fornecidas, onde os parâmetros λ_V, λ_I e λ_C balanceiam as perdas individuais.

2.2.3. Técnica de Embranquecimento para Representações

Embranquecimento é uma transformação linear aplicada a um conjunto de dados, tornando-os descorrelacionados e com variância unitária, ou seja, a matriz de covariância torna-se a matriz identidade. O método de Embranquecimento com Perda Por Mínimos Quadrados (*Whitening Mean Squared Error* - W-MSE) [Ermolov et al., 2021] aplica essa ideia às incorporações de imagens para prevenir o colapso da representação.

Dado um lote de imagens X, transformações aleatórias são aplicadas para obter m visualizações $X(j)$ para todo $j \in \{1,...,m\}$. Um codificador Siamese f_θ mapeia as visualizações para representações $Y(j) = f_\theta(X(j))$, que são então alimentadas em um projetor Siamese g_ϕ para calcular projeções $Z(j) = [z(j)_1,...,z(j)_n] = g_\phi(Y(j))$. Todas as projeções são então concatenadas em uma única matriz $Z = [z(1)_1,...,z(1)_n,...,z(m)_1,...,z(m)_n]$. Esta matriz é branqueada para obter $Z^\sim$ removendo a média e descorrelacionando-a usando a decomposição de Cholesky da matriz de covariância inversa. Para treinar os modelos, o erro quadrático normalizado entre todos os pares de projeções branqueadas é minimizado. A função de perda é definida como

$$L_{WMSE}(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}\frac{2}{m(m-1)}\sum_{j=1}^{m}\sum_{k=j+1}^{m} d_{nse}(\tilde{z}_i^{(j)},\tilde{z}_i^{(k)}).$$

O passo de branqueamento é essencial para evitar o colapso das representações. O objetivo é maximizar a similaridade entre todos os pares aumentados, enquanto também previne o colapso da representação ao impor covariância unitária nas projeções.

2.2.4. Modelo *Transformer* de redes neurais artificiais

O modelo *Transformer* é uma arquitetura de rede neural avançado no campo do processamento de linguagem natural (*Natural Language Processing* - NLP) [Vaswani et al., 2017]. Embora, o modelo *Transformer* não seja considerado um modelo de aprendizado auto-supervisionado, esse modelo apresenta algumas características que são importantes para a definição do aprendizado auto-supervisionado. Diferentemente dos modelos tradicionais, que empregam camadas recorrentes, o *Transformer* utiliza um mecanismo chamado **atenção** para capturar dependências de longo alcance entre os elementos de uma sequência [Torbarina et al., 2024]. Essa abordagem permite uma maior paralelização durante o treinamento e resulta em uma melhor qualidade de tradução em tarefas de NLP, como a tradução de idiomas. O modelo de atenção é crucial para o *Transformer*, permitindo que cada elemento em uma sequência seja comparado com todos os outros para calcular sua importância relativa, possibilitando a aprendizagem de dependências de longo alcance de forma eficaz.

Além da autoatenção dentro do codificador e do decodificador, o *Transformer* também utiliza atenção entre o codificador e o decodificador [Vaswani et al., 2017]. Essa atenção cruzada permite que o modelo se concentre em partes relevantes da entrada durante a geração da saída, facilitando a captura de dependências entre a entrada e a saída. A arquitetura do *Transformer* é composta por várias camadas empilhadas de blocos de atenção e redes *feedforward*, tanto no codificador quanto no decodificador [Vaswani et al., 2017].

O *EsVit* (*Efficient Self-Supervised Vision Transformers*) propõe uma abordagem inovadora para a aplicação de *Transformers* em tarefas visuais [Li et al., 2021]. Essa arquitetura professor-aluno substitui os *Transformers* Visuais por transformadores multi-estágio, mesclando blocos menores (*patches*) de imagem em cada camada para reduzir o processamento [Caron et al., 2021]. No entanto, para mitigar a perda de correspondências locais importantes, o *EsVit* introduz uma perda de correspondência de região adicional [Li et al., 2021]. Isso é feito através de uma extensão da função de perda para identificar

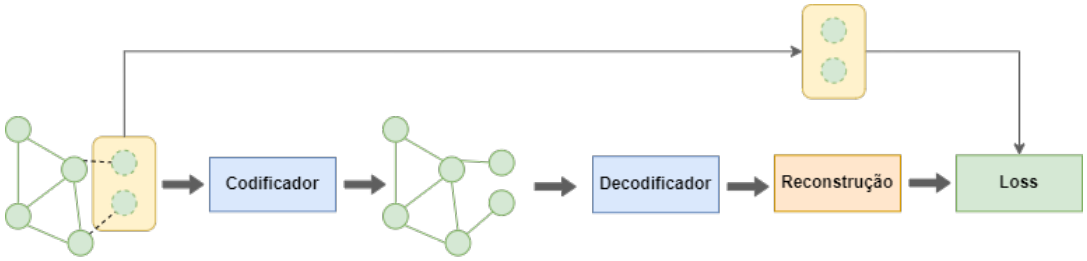


Figura 2.3. Estrutura básica de um modelo auto-supervisionado generativo. O método auto-supervisionado generativo concentra-se nas informações codificadas no grafo, frequentemente baseadas em tarefas de reconstrução de subgrafos ou de elementos do grafo. Desse modo, os atributos e as estruturas dos dados do grafo são utilizados como sinais de auto-supervisão, guiando o aprendizado do modelo sem a necessidade de rótulos externos. Adaptado de [Wu et al., 2023].

características em nível de região, combinando uma perda de nível de visualização e uma perda de nível de região [Li et al., 2021]. Essa abordagem mostra melhorias significativas na captura de correspondências, especialmente em arquiteturas multiestágio, superando limitações anteriores.

2.3. Aprendizado auto-supervisionado generativo

O aprendizado auto-supervisionado generativo se caracteriza pela geração de informações a partir de dados brutos, sem a necessidade de rótulos externos. Através de um decodificador, o modelo reconstrói os dados de entrada, extraindo conhecimento da própria estrutura da informação. Em outras palavras, o modelo busca prever e recriar os dados originais de forma autônoma, sem depender de tarefas supervisionadas externas, conforme ilustrado na Figura 2.3. Existem duas abordagens principais no aprendizado auto-supervisionado generativo: **codificadores automáticos** e **modelos auto-regressivos**. Os codificadores automáticos tem por objetivo reconstruir os dados de entrada de uma só vez, enquanto os modelos auto-regressivos fazem isso de forma iterativa, prevendo um elemento dos dados de cada vez com base nos elementos anteriores.

2.3.1. Modelos auto-regressivos

Os **Modelos auto-regressivos (AR)** são ferramentas estatísticas que descrevem a relação entre variáveis sequenciais, sendo amplamente utilizados na análise de séries temporais. São frequentemente utilizados em cenários estacionários, cuja a média é constante ao longo do tempo [Barbosa et al., 2021a]. A distribuição conjunta pode ser fatorada como um produto de condicionais, em que a probabilidade de cada variável depende das variáveis anteriores [Liu et al., 2023], de acordo com:

$$\max_{\theta} p_{\theta}(x) = \sum_{t=1}^{T} \log p_{\theta}(x_t | x_{1:t-1}).$$

em que $p_{\theta}(x)$ representa a distribuição de probabilidade conjunta de todas as variáveis na série temporal x, onde θ denota os parâmetros do modelo.

Na modelagem de linguagem auto-regressiva em processamento de linguagem natural (*Natural Language Processing* - NLP), como em GPT-4 [Lee et al., 2023], o ob-

jetivo é maximizar a verossimilhança sob a fatoração auto-regressiva para representações unificadas em diferentes áreas, desde processamento de linguagem natural até visão computacional e geração de grafos.

Os modelos autorregressivos também têm sido empregados em visão computacional, como no PixelRNN [Oord et al., 2016b] e PixelCNN [Van den Oord et al., 2016]. Com base no PixelCNN, foi proposto o WaveNet [Oord et al., 2016a], um modelo generativo para áudio bruto. Para lidar com dependências temporais de longo alcance, os autores desenvolveram convoluções causais dilatadas para melhorar o campo receptivo. Além disso, blocos residuais com portas e conexões de salto são empregados para aumentar a expressividade.

A adaptação de domínio é uma técnica de aprendizado de máquina que visa aprimorar o desempenho de um modelo em um domínio de destino, utilizando dados de um domínio de origem relacionado, porém diferente. O objetivo é permitir que um modelo treinado em um conjunto de dados (domínio de origem) generalize seu aprendizado para um conjunto de dados diferente (domínio de destino). A adaptação de domínio pode ser supervisionada, semi-supervisionada ou não supervisionada, dependendo da disponibilidade de rótulos nos conjuntos de dados de origem e destino. Em problemas que envolvem séries temporais, para resolver a Adaptação de Domínio Não Supervisionada (*Unsupervised Domain Adaptation* - UDA), surgem desafios adicionais. Por exemplo, a maioria das soluções existentes é desenvolvida especificamente para dados visuais, e muitas abordagens de adaptação de domínio dependem do pré-treinamento em um grande banco de dados visuais, tal como o ImageNet, como inicialização do modelo. Nesse contexto, [Ragab et al., 2022] propõe uma nova estrutura de Adaptação de Domínio AutoRegressiva Auto-supervisionada (*Self-Supervised AutoRegressive Domain Adaptation* - SLARDA) para aprimorar o desempenho da UDA em séries temporais.

O SLARDA emprega aprendizado autoregressivo por meio de um discriminador autoregressivo que considera a dependência temporal entre as características de séries temporais de origem e destino durante o alinhamento de domínio. O discriminador autoregressivo consiste em uma rede autoregressiva que codifica as dependências temporais entre as características de ambos os domínios em representações vetoriais. Essa abordagem permite capturar a dinâmica temporal dos dados de séries temporais durante o processo de alinhamento de domínio, o que é crucial para obter uma melhor adaptação entre os domínios.

Ao considerar a dependência temporal, o discriminador autoregressivo pode discernir melhor entre as características de séries temporais de origem e destino, evitando ser enganado e alcançando um estado de alinhamento mais satisfatório. Isso contrasta com abordagens anteriores que ignoram a dimensão temporal ao discriminar entre as características de séries temporais de diferentes domínios, resultando em um desempenho limitado para o alinhamento de domínio. Portanto, ao incorporar o aprendizado autoregressivo no SLARDA, a abordagem consegue capturar a dependência temporal das características de séries temporais, melhorando significativamente a capacidade de alinhamento de domínio e, consequentemente, o desempenho geral da adaptação de domínio.

Os modelos autorregressivos também podem ser aplicados a problemas de domínio de grafos, como a geração de grafos. O trabalho de [You et al., 2018b] propõe o

GraphRNN para gerar grafos realistas com modelos autorregressivos profundos. Eles decompõem o processo de geração de grafos em uma sequência de geração de nós e arestas condicionadas ao grafo gerado até o momento. O objetivo do GraphRNN é definido como a verossimilhança das sequências de geração de grafos observadas. O GraphRNN pode ser visto como um modelo hierárquico, em que uma rede neural recorrente (*Recurrent Neural Network* - RNN) de nível de grafo mantém o estado do grafo e gera novos nós, enquanto uma RNN de nível de aresta gera novas arestas com base no estado atual do grafo. Como consequência, outras propostas também se baseiam em abordagens autorregressivas, tais como MRNN [Popova et al., 2019] e GCPN [You et al., 2018a].

O GPT-GNN [Li et al., 2024b] propõe uma estrutura autoregressiva para realizar a reconstrução de nós e arestas em um grafo dado de forma iterativa. Dado um grafo $g_t = (A_t, X_t)$ com seus nós e arestas mascarados aleatoriamente na iteração t, o GPT-GNN gera um nó mascarado X_i e suas arestas conectadas E_i para obter um grafo atualizado $g_{t+1} = (A_{t+1}, X_{t+1})$ e otimiza a probabilidade de geração de nós e arestas na próxima iteração $t+1$, com o objetivo de aprendizado definido como

$$
\begin{aligned}
& p_\theta(X_{t+1}, A_{t+1} | X_t, A_t) \\
& = \sum_o p_\theta(X_i, E_{\neg o_i} | E_{oi}, X_t, A_t) \cdot p_\theta(E_{oi} | X_t, A_t) \\
& = E_o p_\theta(X_{t+1} | E_{oi}, X_t, A_t) \cdot p_\theta(E_{\neg o_i} | E_{oi}, X_{t+1}, A_t),
\end{aligned}
$$

em que o é uma variável que denota o vetor de índice de todas as arestas dentro de E_t na iteração t. Assim, E_{oi} denota as arestas observadas na iteração t, e $E_{\neg o_i}$ denota as arestas mascaradas na iteração $t+1$. Finalmente, o processo de geração de grafo é fatorado em uma etapa de geração de atributo de nó $p_\theta(X_{t+1} | E_{oi}, X_t, A_t)$ e uma etapa de geração de aresta $p_\theta(E_{\neg o_i} | E_{oi}, X_{t+1}, A_t)$.

2.3.2. Modelos baseados em auto-codificadores

Modelos baseados em codificador automático (*autoencoders* - AE) são uma classe popular de modelos utilizados em aprendizado de máquina, especialmente em tarefas de reconstrução e geração de dados. Em diversas situações, atua como uma etapa preparatória para outras redes neurais, diminuindo a complexidade dos dados e eliminando possíveis ruídos [Barbosa et al., 2021b]. Essa otimização beneficia o aprendizado de algoritmos supervisionados. A estrutura do codificador automático é composta por três camadas distintas: entrada, oculta e saída. A **camada oculta** atua como um codificador, compactando as informações, enquanto a **camada de saída** funciona como o decodificador, reconstruindo os dados originais com base na representação latente [Bochie et al., 2020]. Durante a fase de codificação, o modelo reduz a dimensionalidade dos dados de entrada, capturando suas principais características em uma representação latente de menor dimensão. Em seguida, na fase de decodificação, o modelo tenta reconstruir os dados originais a partir da representação latente. Esses modelos têm sido amplamente explorados em diferentes variações, como o *autoencoder denoising*, que é treinado para reconstruir dados a partir de versões corrompidas ou ruidosas, e o *autoencoder* variacional (*Variational Autoecoder* - VAE), que introduz uma abordagem probabilística na geração de dados, permitindo a aprendizagem de representações latentes mais ricas e estruturadas.

Modelo de previsão de contexto

Cada variação dos codificadores automáticos tem suas próprias características e aplicações específicas. Por exemplo, o modelo de previsão de contexto (*Context Prediction Model* - CPM) é útil em tarefas em que a contextualização das informações é importante, enquanto o VAE é valioso quando se deseja aprender representações mais significativas dos dados, especialmente em contextos em que a incerteza é relevante. A escolha da arquitetura, da função de perda e da técnica de treinamento é fundamental para o desempenho desses modelos em diferentes contextos. Esses modelos são frequentemente aplicados em uma variedade de domínios, incluindo processamento de linguagem natural, visão computacional e geração de dados, demonstrando sua versatilidade e eficácia em várias aplicações de aprendizado de máquina.

A ideia do Modelo de Predição de Contexto (CPM) é prever informações contextuais com base nas entradas. Em Processamento de Linguagem Natural (PLN), quando se trata de aprendizado auto-supervisionado *word embedding*, *CBOW* e *Skip-Gram* [Mikolov et al., 2013, de Oliveira et al., 2021] são trabalhos pioneiros. O objetivo do CBOW é prever os *tokens* de entrada com base nos *tokens* de contexto. Em contraste, o objetivo do Skip-Gram é prever os *tokens* de contexto com base nos *tokens* de entrada. Normalmente, a amostragem negativa é empregada para garantir eficiência computacional e escalabilidade.

Inspirados pelo progresso dos modelos de *word embedding* em PLN, muitos modelos de incorporação de rede são propostos com base em um objetivo semelhante de predição de contexto. O Deepwalk [Perozzi et al., 2014] amostra caminhadas aleatórias truncadas para aprender a incorporação latente de nós com base no modelo Skip-Gram. O modelo trata caminhadas aleatórias como o equivalente a frases. No entanto, outra abordagem de incorporação de rede, o LINE [Tang et al., 2015], visa gerar vizinhos em vez de nós em um caminho com base nos nós atuais:

$$O = -\sum_{(i,j)\in E} w_{ij} \log p(v_j|v_i),$$

em que E denota o conjunto de arestas, v denota o nó, w_{ij} representa o peso da aresta (v_i, v_j). O LINE também usa amostragem negativa para amostrar múltiplas arestas negativas a fim de aproximar o objetivo.

Modelo de redução de ruído

Uma das variações de modelos que utilizam codificadores automáticos é o *Modelo de Redução de Ruído (Denoising Autoencoder)*. Tradicionalmente, o codificador automático não é capaz de obter características relevantes em ambientes heterogêneos. Para contornar esse problema, o modelo de redução de ruído é treinado para reconstruir os dados de entrada após aplicação de ruídos [Abusitta et al., 2023].

A intuição é que a representação deve ser robusta à introdução de ruído. O modelo de linguagem mascarada (*Masked Language Model* - MLM), uma das arquiteturas mais bem-sucedidas no processamento de linguagem natural, pode ser considerado como um modelo de codificador automático de remoção de ruído. Para modelar sequências de texto, o modelo de linguagem mascarada (MLM) oculta aleatoriamente alguns dos

tokens da entrada e então os prevê com base em suas informações de contexto. BERT [Devlin et al., 2018] é o trabalho mais representativo nesse campo. Especificamente, no BERT, um *token* único $[MASK]$ é introduzido no processo de treinamento para mascarar alguns *tokens*. No entanto, uma limitação desse método é que não há *tokens* de entrada $[MASK]$ para tarefas posteriores. Para mitigar isso, os autores nem sempre substituem os *tokens* previstos por $[MASK]$ durante o treinamento. Em vez disso, eles os substituem por palavras originais ou palavras aleatórias com uma pequena probabilidade.

Outra abordagem promissora na aprendizagem de representação envolve a combinação de modelos de condrificadores automáticos e baseados em fluxo. Modelos baseados em fluxo mapeiam uma distribuição de base para a distribuição de interesse, gerando amostras de alta qualidade e calculando a densidade de probabilidade exata. O modelo GraphAF [Liu et al., 2019a] é um exemplo dessa combinação usado no contexto de geração de moléculas. Além disso, incorpora conhecimento detalhado do domínio na definição da recompensa, como a verificação de valência.

A técnica de “dequantização"também é aplicada para converter dados discretos em contínuos, sendo útil em tarefas de processamento de imagem. Essa técnica é particularmente útil em tarefas de processamento de imagem, em que os dados são frequentemente discretos. Essas abordagens oferecem vantagens significativas na geração de amostras de alta qualidade e no cálculo preciso de densidades de probabilidade em diversos domínios de aplicação. A combinação de modelos *autoencoders* e baseados em fluxo é uma área ativa de pesquisa com potencial aplicação em diversas áreas, como o processamento de tráfego de redes.

Modelos híbridos

Modelos híbridos combinam modelos auto-regressivos (AR) e *autoencoders* (AE) e são uma abordagem promissora para obter representações latentes mais ricas e estruturadas dos dados. Essa abordagem é útil em tarefas de processamento de linguagem natural. Além disso, a combinação de modelos AR e AE pode ser estendida para outras áreas de pesquisa, como processamento de imagem, processamento de sinais e processamento de tráfego de redes.

Alguns pesquisadores propõem combinar as vantagens dos modelos autoregressivos (AR) e dos *autoencoders* (AE). O modelo MADE (*Masked Autoencoder for Distribution Estimation*) [Germain et al., 2015] faz uma modificação simples no *autoencoder*, mascarando os parâmetros para respeitar as restrições autoregressivas. Especificamente, enquanto os neurônios entre camadas adjacentes são totalmente conectados no *autoencoder* original, no MADE, algumas conexões são mascaradas para garantir que cada dimensão de entrada seja reconstruída apenas a partir de suas próprias dimensões. MADE pode ser facilmente paralelizado em computações condicionais, permitindo estimativas diretas e econômicas de probabilidades conjuntas em alta dimensão.

No processamento de linguagem natural (PLN), o Modelo de Linguagem por Permutação (*Permutation Language Model* - PLM) é um modelo representativo que combina as vantagens dos modelos autoregressivos e dos *autoencoders*. O XLNet, que introduz o PLM, é um método de pré-treinamento autoregressivo generalizado [Yang et al., 2019]. O XLNet permite aprender contextos bidirecionais maximizando a probabilidade esperada

sobre todas as permutações da ordem de fatoração.

Para formalizar a ideia, seja Z_T o conjunto de todas as permutações possíveis da sequência de índices de comprimento T $[1,2,...,T]$, o objetivo do PLM pode ser expresso como:

$$\max_{\theta} \mathbb{E}_{z \sim Z_T} \left[\sum_{t=1}^{T} \log p_\theta (x_{z_t} | x_{z<t}) \right].$$

Para cada sequência de texto, diferentes ordens de fatoração são amostradas, permitindo que cada *token* veja sua informação contextual de ambos os lados. Com base na ordem permutada, o XLNet também realiza a reparametrização com posições para que o modelo saiba qual posição precisa prever. Então, uma atenção especial em dois fluxos é introduzida para a previsão consciente do alvo.

Além disso, diferentemente do BERT, inspirado pelos avanços recentes no modelo AR, o XLNet integra o mecanismo de recorrência de segmentos e o esquema de codificação relativa do Transformer-XL [Dai et al., 2019] no pré-treinamento, o que permite modelar melhor a dependência de longo alcance em comparação com o modelo Transformer [Torbarina et al., 2024].

DINO - *Self-Distillation With No Labels*

O DINO [Caron et al., 2018] define uma rede aluno e uma rede professor. O aluno consiste em um codificador f_θ e um projetor g_ϕ com parâmetros θ e ϕ. O codificador é implementado como um *transformer* visual e o projetor como uma rede perceptron de múltiplas camadas (*Multi-Layer Perceptron* - MLP). O professor consiste em um codificador $f_{\bar{\theta}}$ e um projetor $g_{\bar{\phi}}$ com a mesma arquitetura do aluno, mas um conjunto separado de parâmetros $\bar{\theta}$ e $\bar{\phi}$. A arquitetura do Modelo é apresentada na Figura 2.4.

DINO usa uma estratégia *multi-crop* primeiro para criar um lote de m visualizações $X_i = [x_{(1)i}, \ldots, x_{(m)i}]$ de uma imagem x_i. Cada visualização é uma recorte aleatório de x_i seguido por mais transformações. A maioria dos recortes cobre uma pequena região da imagem, mas alguns recortes são de alta resolução, referidos como visualizações locais e globais, respectivamente. Seja M_i o conjunto de índices das visualizações globais. A ideia é que a rede aluno tenha acesso a todas as visualizações, enquanto a rede professor tenha acesso apenas às visualizações globais, criando correspondências “do local para o global" [Caron et al., 2021].

O aluno calcula representações $y_{(j)i} = f_\theta(x_{(j)i})$ e projeções $z_{(j)i} = g_\phi(y_{(j)i})$ para cada visualização. O professor calcula projeções alvo $\bar{z}_{(j)i} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_{(j)i}))$ para as visualizações globais $j \in M_i$.

Para prevenir o colapso, os autores identificam experimentalmente duas formas distintas em que ele pode ocorrer, a distribuição de probabilidade calculada pode ser uniforme ou uma única dimensão pode dominar, independentemente da entrada. Isso motiva a adoção de duas contramedidas específicas:

- Para evitar o colapso para uma distribuição uniforme, a distribuição alvo do professor é ajustada definindo o parâmetro de temperatura τ para um valor pequeno.

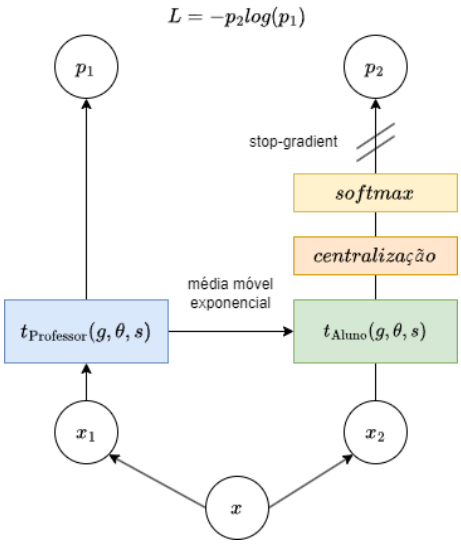


Figura 2.4. Arquitetura do Modelo DINO utiliza um par de imagens transformadas aleatoriamente para treinar redes de aluno e professor com a mesma arquitetura, mas diferentes parâmetros. As saídas das redes são normalizadas e comparadas usando uma perda de entropia cruzada. Um operador *stop-gradient* é aplicado ao professor para propagar gradientes apenas para o aluno. Os parâmetros do professor são atualizados usando uma média móvel exponencial dos parâmetros do aluno. Adaptado de [Caron et al., 2018].

- Para evitar que uma dimensão domine, a saída do professor é centralizada para torná-la mais uniforme.

Isso é realizado adicionando um vetor de centralização c como um viés ao professor, que é calculado com uma média móvel exponencial:

$$c \leftarrow \beta c + (1-\beta)\bar{z}, \quad \text{onde } \beta \in [0,1]$$

é um hiperparâmetro de decaimento que determina em que medida o vetor de centralização é atualizado, e

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|M_i|}\sum_{j\in M_i} z_{(j)i}$$

é a média de todas as projeções do professor no lote atual.

Para realizar o aprendizado de características invariantes por meio de rótulos suaves, o método DINO formula a tarefa de prever as projeções alvo como uma tarefa de destilação de conhecimento. As projeções do professor e do aluno são convertidas em distribuições de probabilidade, aplicando a função *softmax* sobre todos os componentes. Assim, a perda de entropia cruzada pode ser aplicada, em que o professor gera rótulos suaves para o aluno. A função de perda total associa cada visualização do aluno com cada visualização global do professor, dado por

$$L_{\text{DINO}}^{\theta,\phi} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j\in M_i}\sum_{k\neq j}\text{dce}\left(\text{softmax}_\tau(\bar{z}_{(j)i} - c), \text{softmax}_\rho(z_{(k)i})\right),$$

em que $\tau, \rho > 0$ são hiperparâmetros que controlam a temperatura das distribuições para o professor e o aluno, respectivamente. No geral, as atualizações de parâmetros são muito semelhantes às do BYOL, uma vez que a rede aluno é atualizada minimizando a perda $L_{\text{DINO}}^{\theta,\phi}$ usando o otimizador AdamW, e a rede professor é atualizada por uma média móvel exponencial do aluno, ou seja,

$$\bar{\theta} \leftarrow \alpha\bar{\theta} + (1-\alpha)\theta, \quad \bar{\phi} \leftarrow \alpha\bar{\phi} + (1-\alpha)\phi,$$

em que $\alpha \in [0,1]$ controla a taxa na qual os pesos da rede professor são atualizados com os pesos da rede aluno.

2.4. Aprendizado auto-supervisionado contrastivo

Do ponto de vista estatístico, os modelos de aprendizado de máquina são categorizados em modelos generativos e discriminativos. Dada a distribuição conjunta $P(X,Y)$ da entrada X e do alvo Y, o modelo generativo calcula $P(X|Y=y)$ enquanto o modelo discriminativo visa modelar $P(Y|X=x)$.

2.4.1. Aprendizado contrastivo de representação

Embora não seja possível avaliar $P(X)$ ou $P(X|Y)$ diretamente, pode-se usar amostras dessas distribuições, permitindo usar técnicas como Estimação por Contraste de Ruído [Oord et al., 2018, Gutmann e Hyvärinen, 2010], que se baseia na comparação do valor alvo com valores negativos amostrados aleatoriamente. A técnica de Estimação por Contraste de Ruído (*Noise-Contrastive Estimation* - NCE) é um método de estimação de parâmetros em modelos estatísticos parametrizados, que foi proposto como uma abordagem eficiente para lidar com modelos estatísticos não normalizados. NCE introduz um ruído contrastivo no processo de estimação, permitindo que a normalização do modelo seja tratada como um problema de otimização. A ideia geral do modelo de aprendizado contrastivo é apresentada na Figura 2.5.

Estimação de Ruído Contrastivo - *Noise Contrastive Estimation* - NCE

A ideia da Estimação de Ruído Contrastivo é formular a tarefa de aprendizado de representação contrastiva como um problema de classificação supervisionada. Uma suposição do NCE é que os negativos são independentes do âncora, ou seja, $p_{\text{neg}}(x^-|x^*) = p_{\text{neg}}(x^-)$. Nesse contexto, os negativos são frequentemente chamados de ruído. Existem duas abordagens amplamente utilizadas, o NCE original e o InfoNCE [Oord et al., 2018]. De forma geral, o NCE realiza classificação binária para decidir se uma amostra individual é positiva ou negativa, enquanto o InfoNCE realiza classificação multi-classe em um conjunto de amostras para decidir qual é a positiva.

A função objetivo da Estimação de Ruído Contrastivo (NCE) é dada por:

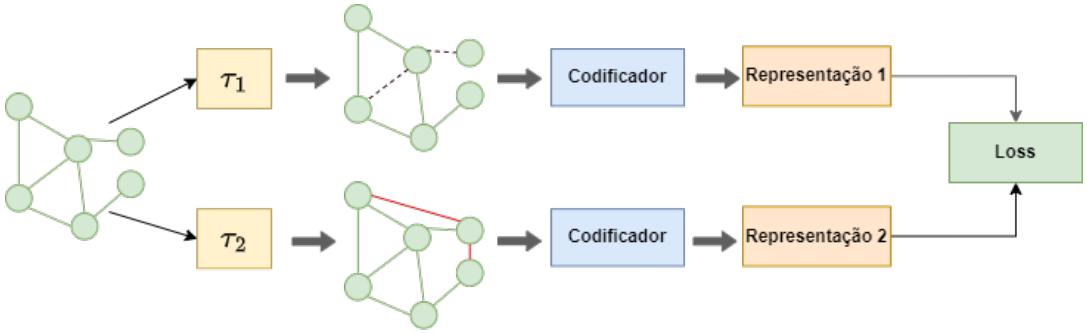


Figura 2.5. Estrutura básica de um modelo auto-supervisionado contrastivo. O método contrastivo compara as informações geradas por diferentes funções T1 e T2. A informação sobre as diferenças e semelhanças entre pares de dados (inter-dados) é usada como sinais de auto-supervisão. Adaptado de [Wu et al., 2023].

$$L = \mathbb{E}_{x,x^+,x^-}\left[-\log\left(\frac{e^{f(x)/T}}{e^{f(x)^T f(x^+)}} + e^{f(x)^T f(x^-)}\right)\right],$$

em que x^+ é similar a x, x^- é dissimilar a x e f é um codificador (função de representação).

InfoNCE

Para cada âncora x^*, o InfoNCE gera uma amostra positiva a partir de $p_{\text{pos}}(x^+|x^*)$ e $n-1$ amostras negativas de $p_{\text{neg}}(x^-)$. Seja $X = [x_1,\ldots,x_n]$ o conjunto dessas amostras, em que x_c é a positiva com índice $c \in \{1,\ldots,n\}$. No contexto do aprendizado de representação, calculam-se representações adicionais usando um codificador f_θ e se obtém o conjunto $Y = [y_1,\ldots,y_n]$. O InfoNCE agora define uma tarefa de classificação supervisionada, em que a entrada é (y^*,Y) e o rótulo de classe é o índice do positivo c. Um classificador $p_\psi(c|Y,y^*)$ com parâmetros ψ é treinado para combinar a verdadeira distribuição de dados dos rótulos $p_{\text{data}}(c|Y,y^*)$. Um objetivo comum de aprendizado supervisionado é minimizar a entropia cruzada entre a distribuição de dados e a distribuição do modelo, ou seja,

$$\min_{\psi,\theta} \mathbb{E}_{Y,y^*}[H(p_{\text{data}}(c|Y,y^*), p_\psi(c|Y,y^*))] = \min_{\psi,\theta} \mathbb{E}_{Y,y^*}[H(p_{\text{data}}(c|Y,y^*), p_\psi(c|Y,y^*))].$$

Este é um problema de previsão anti-causal, em que a causa subjacente (rótulo) é prevista a partir de seu efeito. No InfoNCE, é conhecido o mecanismo subjacente, já que os rótulos são gerados artificialmente, então pode-se derivar o classificador ideal usando o teorema de Bayes.

Primeiro, estima-se a distribuição de dados de um conjunto Y dado um rótulo e uma âncora, ou seja,

$$p_{\text{data}}(Y|c,y^*) = \prod_{i=1}^{n} p_{\text{data}}(y_i|c,y^*) = \prod_{i=1}^{n} \begin{cases} p_{\text{pos}}(y_i|y^*), & \text{se } i = c, \\ p_{\text{neg}}(y_i), & \text{se } i \neq c, \end{cases}$$

$$= p_{\text{pos}}(y_c|y^*) \prod_{i \neq c} p_{\text{neg}}(y_i) = p_{\text{pos}}(y_c|y^*) \cdot \left(\prod_{i=1}^{n} p_{\text{neg}}(y_i) \right),$$

em que se assume independência condicional entre as amostras em Y. O InfoNCE ainda assume que os rótulos são amostrados uniformemente, ou seja, $p_{\text{data}}(c) = \frac{1}{n}$. Na sequência, aplica-se o teorema de Bayes:

$$p_{\text{data}}(c|Y, y^*) = \frac{p_{\text{data}}(Y|c, y^*) p_{\text{data}}(c)}{\sum_{c'=1}^{n} p_{\text{data}}(Y|c', y^*) p_{\text{data}}(c')},$$

$$= \frac{p_{\text{pos}}(y_c|y^*) p_{\text{neg}}(y_c) \prod_{i=1}^{n} p_{\text{neg}}(y_i)}{\sum_{c'=1}^{n} p_{\text{pos}}(y_{c'}|y^*) p_{\text{neg}}(y_{c'}) \prod_{i=1}^{n} p_{\text{neg}}(y_i)},$$

$$= \frac{p_{\text{pos}}(y_c|y^*) p_{\text{neg}}(y_c)}{\sum_{c'=1}^{n} p_{\text{pos}}(y_{c'}|y^*) p_{\text{neg}}(y_{c'})}.$$

Um classificador ideal com entropia cruzada zero coincidiria com essa distribuição. A probabilidade ótima de uma classe é a razão de densidade $\frac{p_{\text{pos}}(y_c|y^*)}{p_{\text{neg}}(y_c)}$, normalizada em todas as classes. Isso descreve a probabilidade de y_c ser uma amostra positiva para y^* versus ser uma amostra negativa. Isso motiva a escolha do classificador do InfoNCE, que é definido de forma semelhante:

$$p_{\psi}(c|Y, y^*) = \frac{s_{\psi}(y^*, y_c)}{\sum_{c'=1}^{n} s_{\psi}(y^*, y_{c'})},$$

em que $s_{\psi}(y^*, y)$ é um preditor que calcula uma pontuação positivo real. Minimizar a entropia cruzada aproxima a distribuição do modelo $p_{\psi}(c|Y, y^*)$ à distribuição de dados $p_{\text{data}}(c|Y, y^*)$, o que garante que s_{ψ} se aproxime da razão de densidade dos dados, ou seja, $s_{\psi}(y^*, y) \approx \frac{p_{\text{pos}}(y|y^*)}{p_{\text{neg}}(y)}$, mas só precisa ser proporcional à razão de densidade.

A razão de densidade é alta para amostras positivas e próxima de zero para amostras negativas, o que significa que $s_{\psi}(y^*, y)$ aprende alguma medida de similaridade entre as representações. Como ψ e θ ou seja, preditor e codificador, são otimizados em conjunto, o codificador é incentivado a aprender incorporações semelhantes para uma âncora e seu positivo, e a aprender incorporações diferentes para uma âncora e suas amostras negativas. Em outras palavras, o codificador é incentivado a extrair informações que são "únicas"para a âncora e a amostra positiva. Esse objetivo maximiza a informação mútua entre y^* e y^+, que é um limite inferior para a informação mútua entre x^* e x^+.

A perda geral do InfoNCE para (y^*, Y, c) é definida como:

$$\text{InfoNCE}_{s_{\psi}}(y^*, Y, c) = -\log \left(\frac{s_{\psi}(y^*, y^+)}{\sum_{c'=1}^{n} s_{\psi}(y^*, y_{c'})} \right).$$

A perda do InfoNCE calcula a entropia cruzada softmax comumente usada. Em vez de especificar o rótulo de classe, denota-se o positivo por y^+ e o conjunto de negativos

por Y. Assim, a definição final da perda do InfoNCE para uma função de pontuação $s_\psi(y^*, y)$ é:

$$\text{InfoNCE}_{s_\psi}(y^*, y^+, Y) = -\log\left(\frac{\exp(s_\psi(y^*, y^+))}{\exp(s_\psi(y^*, y^+)) + \sum_{y \in Y} \exp(s_\psi(y^*, y))}\right).$$

O aprendizado auto-supervisionado contrastivo é uma técnica bastante usada para aprender representações de alta qualidade, focando a maximização da similaridade entre exemplos positivos e a minimização da similaridade entre exemplos negativos. A ideia central é que exemplos semelhantes devem ser representados próximos no espaço de representação, enquanto exemplos diferentes devem ser representados distantes. Essa abordagem tem sido aplicada com sucesso em diversas tarefas, incluindo classificação de imagens, detecção de objetos e reconhecimento de fala.

2.4.2. Contraste por instância de contexto

O **contraste por instância de contexto**, também conhecido como contraste global-local, é uma abordagem que visa modelar a relação entre a característica local de uma amostra e sua representação de contexto global. Quando o modelo aprende a representação para uma característica local, essa característica é associada à representação do conteúdo global, como nós para seus vizinhos. Existem dois tipos principais de Contraste de Contexto-Instância: *Previsão de Posição Relativa* e *Maximização de Informação Mútua*.

Previsão de posição relativa

A Previsão de Posição Relativa se concentra em aprender posições relativas entre componentes locais, em que o contexto global atua como um requisito implícito para prever essas relações. Muitos dados contêm ricas relações espaciais ou sequenciais entre suas partes. Vários modelos consideram o reconhecimento de posições relativas entre partes como a tarefa de pretexto [Jing e Tian, 2020]. Pode ser para prever as posições relativas de dois blocos menores (*patches*) a partir de uma amostra [Doersch et al., 2015], ou para recuperar as posições de segmentos embaralhados de uma imagem, resolver quebra-cabeças [Kim et al., 2018, Noroozi e Favaro, 2016, Wei et al., 2019], ou para inferir o grau de rotação de uma imagem [Gidaris et al., 2018]. A previsão de posição relativa também pode servir como ferramentas para criar amostras positivas difíceis. Por exemplo, a técnica do quebra-cabeça é aplicada no PIRL [Misra e Maaten, 2020] para aumentar a amostra positiva, mas o PIRL não considera resolver o quebra-cabeça e recuperar a relação espacial como seu objetivo.

Nos modelos de linguagem pré-treinados, ideias semelhantes, como a Previsão da Próxima Sentença (*Next Sentence Prediction* - NSP), também são adotadas. A perda de NSP foi inicialmente introduzida pelo BERT [Devlin et al., 2018], em que, para uma frase, o modelo deve distinguir a seguinte de uma amostrada aleatoriamente. No entanto, alguns trabalhos posteriores provam empiricamente que NSP ajuda pouco, ou até prejudica o desempenho. Portanto, no RoBERTa [Liu et al., 2019b], a perda de NSP é removida.

Para substituir a NSP, o ALBERT [Lan et al., 2019] propõe a tarefa de Previsão da Ordem das Sentenças (*Sentence Order Prediction* - SOP). No ALBERT, a tarefa de Previsão da Ordem das Sentenças (SOP) é uma estratégia de treinamento que visa melhorar a capacidade do modelo de capturar a coerência inter-sentença em um texto. Nessa tarefa, o modelo é apresentado com pares de segmentos de texto consecutivos de um documento e é treinado para prever a ordem correta desses segmentos.

Para realizar o treinamento, são utilizados exemplos positivos e negativos. Os exemplos positivos consistem em dois segmentos consecutivos do mesmo documento, mantendo a ordem original. Por outro lado, os exemplos negativos são compostos pelos mesmos dois segmentos, mas com a ordem trocada. Essa abordagem desafia o modelo a entender e capturar as relações de coerência entre as sentenças, em vez de simplesmente prever tópicos ou palavras isoladas.

Ao focara coerência inter-sentença, o ALBERT busca melhorar a capacidade do modelo de compreender o contexto global de um texto e a relação entre diferentes partes do mesmo. Isso é fundamental para tarefas de processamento de linguagem natural que exigem uma compreensão mais profunda do texto, como tradução automática, resumo de texto, análise de sentimentos e questionamento e resposta. A tarefa SOP no ALBERT atua como um mecanismo de treinamento eficaz para melhorar a representação de texto e a capacidade de modelagem de coerência textual.

Maximização de informação mútua

Este tipo de método deriva da informação mútua (*Mutual Information* - MI). A informação mútua visa modelar a associação entre duas variáveis. Geralmente, esses modelos otimizam $\max_{g_1 \in G_1, g_2 \in G_2} I(g_1(x_1), g_2(x_2))$, em que g_i é o codificador de representação, G_i é uma classe de codificadores com algumas restrições, e $I(\cdot,\cdot)$ é um estimador baseado em amostras para a informação mútua precisa. Na prática, MI é notória por sua computação complexa. Uma prática comum é maximizar alternativamente o limite inferior de I com um objetivo NCE.

Deep InfoMax (DIM) [Hassani e Khasahmadi, 2020] foi o primeiro a modelar explicitamente a informação mútua por meio de uma tarefa de aprendizado contrastivo, maximizando a MI entre um *patch* local e seu contexto global. Em classificação de imagens, por exemplo, pode-se codificar uma imagem de gato x em $f(x) \in \mathbb{R}^{M \times M \times d}$ e extrair um vetor de característica local $v \in \mathbb{R}^d$. Para conduzir o contraste entre instância e contexto, precisa-se de uma função resumo $g : \mathbb{R}^{M \times M \times d} \to \mathbb{R}^d$ para gerar o vetor de contexto $s = g(f(x)) \in \mathbb{R}^d$ e outra imagem de gato x^- e seu vetor de contexto $s^- = g(f(x^-))$.

Deep InfoMax proporciona um novo paradigma e impulsiona o desenvolvimento do aprendizado auto-supervisionado. Um modelo derivado do Deep InfoMax é o Contrastive Predictive Coding (CPC) [Oord et al., 2018] para reconhecimento de fala, que maximiza a associação entre um segmento de áudio e seu contexto. CPC também foi aplicado na classificação de imagens. DeepInfoMax de Escala Aumentada *Augmented Multiscale DIM - AMDIM* [Bachman et al., 2019] aprimora a associação positiva entre uma característica local e seu contexto, amostrando aleatoriamente duas visões diferentes de uma imagem para gerar o vetor de característica local e o vetor de contexto, respectivamente. CMC estende essa ideia para várias visões de uma imagem, amostrando outra

imagem irrelevante como negativa.

O Deep Graph Structural Infomax (DGSI) [Zhao et al., 2023] é uma extensão do Deep Graph InfoMax (DGI) que incorpora informações estruturais e topológicas para melhorar a representação dos nós de maneira auto-supervisionada. O DGSI consegue capturar informações semânticas mais detalhadas e informações estruturais benéficas utilizando várias abordagens. Uma dessas abordagens é a maximização da informação mútua estrutural. O DGSI aplica o princípio do *Information Bottleneck* para estabelecer restrições de informação mútua estrutural, equilibrando a suficiência e a minimalidade da representação ao preservar a estrutura topológica. Dessa forma, o modelo maximiza a informação mútua estrutural em relação às arestas e aos vizinhos locais, capturando detalhes finos da estrutura do grafo.

Além disso, o DGSI integra informações locais e globais ao maximizar a informação mútua entre a representação do nó e o resumo global do grafo, considerando também a conexão detalhada entre o nó e sua região receptiva local. Combinando essas restrições de informação mútua estrutural e representacional em uma única estrutura, o DGSI é capaz de capturar tanto a informação semântica dos nós quanto a informação estrutural do grafo, resultando em representações de nós mais ricas e abrangentes.

2.4.3. Contraste de instância para instância

O **aprendizado contrastivo de instância para instância** é uma abordagem que se concentra na modelagem da relação entre pares de instâncias, contrastando-as para aprender representações que capturem a estrutura subjacente dos dados. Ao contrário do contraste contexto-instância, que se concentra na relação entre uma amostra e seu contexto, essa técnica visa aprender representações discriminativas por meio da comparação direta de pares de instâncias. Essa abordagem motiva a rede a aprender representações discriminativas maximizando a similaridade entre pares de instâncias positivas e minimizando a similaridade entre instâncias negativas. Em outras palavras, o modelo é incentivado a aprender a distinguir entre diferentes instâncias, o que é fundamental para tarefas de classificação.

O aprendizado contrastivo de instância para instância é uma técnica poderosa para aprender representações discriminativas a partir de dados não rotulados. Ela tem demonstrado bons resultados em várias tarefas de aprendizado de máquina, especialmente em problemas de classificação linear, em que a capacidade de distinguir entre classes é essencial. Além disso, é uma técnica simples e eficaz que pode ser facilmente aplicada a uma ampla gama de problemas de aprendizado de máquina, proporcionando uma abordagem flexível e robusta para a aprendizagem de representações de dados.

2.4.4. Pré-treinamento contrastivo

Enquanto o aprendizado autos-supervisionado baseado em aprendizado contrastivo continua a ultrapassar limites em vários *benchmarks*, os rótulos ainda são importantes porque há uma lacuna entre os objetivos de treinamento do aprendizado auto-supervisionado e do aprendizado supervisionado. Em outras palavras, não importa o quanto os modelos de aprendizado auto-supervisionado melhorem, eles ainda são apenas extratores de características poderosos e, para transferir para a tarefa subsequente,

ainda há a necessidade de rótulos de alguma forma. Como resultado, para preencher a lacuna entre o pré-treinamento auto-supervisionado e as tarefas subsequentes, o aprendizado semi-supervisionado é um tipo de recurso empregado.

Aprendizado semi-supervisionado é uma abordagem de aprendizado de máquina que combina uma pequena quantidade de dados rotulados com muitos dados não rotulados durante o treinamento. Vários métodos derivam de diferentes suposições feitas sobre a distribuição dos dados, sendo o auto-treinamento o mais antigo. No auto-treinamento, um modelo é treinado com a pequena quantidade de dados rotulados e, em seguida, gera rótulos nos dados não rotulados. Apenas aqueles dados com rótulos altamente confiáveis são combinados com os dados rotulados originais para treinar um novo modelo. O processo ocorre de maneira iterativa até encontrar o melhor modelo.

O pré-treinamento contrastivo visa aprender representações úteis dos dados, enfatizando a similaridade entre instâncias positivas e reduzindo a similaridade entre instâncias negativas. Essa abordagem busca criar um espaço de representação em que instâncias semelhantes estejam próximas umas das outras e instâncias diferentes estejam distantes. Esse método é formulado através de uma função de perda, na qual as instâncias de entrada são representadas por x_i, suas representações latentes por $f(x_i)$, e $sim(x_i, x_j)$ é uma medida de similaridade entre elas. A função de perda é definida considerando o número total de instâncias N e uma temperatura τ que controla a suavização da distribuição de similaridade.

O método inclui a etapa de ajuste do modelo usando dados rotulados no processo de auto-treinamento semi-supervisionado. Esse processo envolve gerar pseudo-rótulos para dados não rotulados, usando o modelo treinado, e repetir o treinamento várias vezes até que o desempenho do modelo se estabilize. O pré-treinamento contrastivo tem sido aplicado com sucesso em tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação semântica, com exemplos notáveis como SimCLR [Chen et al., 2020b], MoCo [He et al., 2020a] e BYOL [Grill et al., 2020].

À luz do sucesso do auto-treinamento semi-supervisionado, é natural repensar sua relação com os métodos auto-supervisionados, especialmente com os métodos de pré-treinamento contrastivo bem-sucedidos. Para tarefas de visão computacional, soluções como as apresentadas por [Zoph et al., 2020] estudam o pré-treinamento do MoCo e um método de auto-treinamento em que um professor é primeiro treinado em um conjunto de dados subsequente e depois gera pseudo-rótulos em dados não rotulados e finalmente um modelo aluno aprende conjuntamente sobre rótulos reais no conjunto de dados subsequente e pseudo-rótulos em dados não rotulados. O estudo aponta que o desempenho do pré-treinamento é prejudicado enquanto o auto-treinamento ainda se beneficia de um forte aumento de dados. Além disso, a adição de mais dados rotulados diminui o valor do pré-treinamento, enquanto o auto-treinamento semi-supervisionado sempre apresenta melhorias. Também foi identificado que as melhorias provenientes do pré-treinamento e do auto-treinamento são ortogonais entre si, ou seja, contribuem para o desempenho a partir de diferentes perspectivas. O modelo que combina pré-treinamento e auto-treinamento é o que apresenta o melhor desempenho.

O SimCLR de [Chen et al., 2020b] mostra que com apenas 10% dos rótulos originais do ImageNet [Deng et al., 2009], o ResNet-50 [Wen et al., 2020] pode superar o

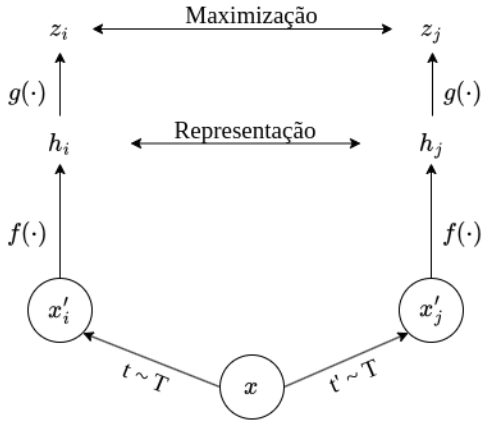


Figura 2.6. O arcabouço SimCLR. Duas operações de aumento de dados separadas são amostradas da mesma família de aumentações ($t \sim T$ e $t_0 \sim T$) e aplicadas a cada exemplo de dados para obter duas visualizações correlacionadas. Uma rede codificadora base $f(\cdot)$ e uma cabeça de projeção $g(\cdot)$ são treinadas para maximizar o acordo usando uma perda contrastiva. Após o treinamento ser concluído, descarta-se a cabeça de projeção $g(\cdot)$ e usa-se a codificadora $f(\cdot)$ e a representação h para tarefas posteriores. Adaptado de [Chen et al., 2020b].

supervisionado com pré-treinamento e auto-treinamento conjuntos. Eles propõem um arcabouço de 3 etapas:

1. Realizar pré-treinamento auto-supervisionado como o SimCLR v1, com algumas pequenas modificações na arquitetura e um ResNet mais profundo;
2. Ajustar as últimas camadas com apenas 1% ou 10% dos rótulos originais do ImageNet;
3. Usar a rede ajustada como professor para gerar rótulos em dados não rotulados para treinar um ResNet-50 aluno menor.

O sucesso na combinação de pré-treinamento auto-supervisionado contrastivo e auto-treinamento semi-supervisionado é uma tendência para o futuro paradigma de aprendizado profundo eficiente em dados. Mais trabalhos são esperados para investigar seus mecanismos latentes.

SimCLR - ***Simple Framework for Contrastive Learning of Visual Representations***

O arcabouço proposto por [Chen et al., 2020b] e apresentado na Figura 2.6 é similar a métodos anteriores como VICReg ou Barlow Twins. Dado um lote de imagens X, duas visualizações $X^{(1)} = t(X)$ e $X^{(2)} = t(X)$ são criadas usando transformações aleatórias $t \sim T$. Um codificador Siamese f_θ calcula representações $Y^{(1)} = f_\theta(X^{(1)})$ e $Y^{(2)} = f_\theta(X^{(2)})$, que são então alimentadas em um projetor Siamese g_ϕ para obter projeções $Z^{(1)} = [z_1^{(1)}, \ldots, z_n^{(1)}] = g_\phi(Y^{(1)})$ e $Z^{(2)} = [z_1^{(2)}, \ldots, z_n^{(2)}] = g_\phi(Y^{(2)})$. A Figura 2.6 mostra uma visão geral do processo.

O SimCLR usa uma perda contrastiva para maximizar a similaridade entre as duas projeções da mesma imagem enquanto minimiza a similaridade com as projeções de outras imagens. Especificamente, para uma imagem x_i, são aplicadas duas perdas InfoNCE.

A primeira usa o âncora $z_i^{(1)}$, o positivo $z_i^{(2)}$ e os negativos $\bar{Z}_i = [z_1^{(1)}, z_1^{(2)}, \ldots, z_n^{(1)}, z_n^{(2)}] \setminus \{z_i^{(1)}, z_i^{(2)}\}$, que são todas projeções de outras imagens no lote. A segunda perda InfoNCE troca os papéis de âncora e positivo, mas usa o mesmo conjunto de negativos. Portanto, a função de perda é definida como

$$L_{\text{SimCLR}}^{\theta,\phi} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\text{InfoNCE}_{s_\tau}(z_i^{(1)}, z_i^{(2)}, \bar{Z}_i) + \text{InfoNCE}_{s_\tau}(z_i^{(2)}, z_i^{(1)}, \bar{Z}_i)\right),$$

em que as similaridades são calculadas como $s_\tau(z, z') = \text{scos}(z, z')/\tau$, ou seja, a similaridade cosseno dividida por um hiperparâmetro de temperatura $\tau > 0$.

As transformações consistem em um recorte aleatório seguido por um redimensionamento de volta ao tamanho original, uma distorção de cor aleatória e um desfoque gaussiano aleatório. Um ResNet é usado como codificador f_θ e o projetor g_ϕ é implementado como uma MLP com uma camada oculta. Para treinar o SimCLR, são utilizados tamanhos de lote grandes em combinação com o otimizador LARS. Os autores observam que seu método não precisa de bancos de memória, como é o caso de outros métodos contrastivos, e é portanto mais fácil de implementar.

MOCO - *Momentum Contrast*

O Momentum Contrast [He et al., 2020a] é uma abordagem de aprendizado contrastivo que utiliza um codificador de *momentum* com uma fila de codificação. Em essência, ele permite a otimização de um objetivo contrastivo com custos computacionais significativamente reduzidos, tanto em termos de tempo quanto de memória da GPU [Chen et al., 2020a]. Conforme mostrado na Figura 2.7, MoCo define uma rede de aluno consistindo de um codificador f_θ e um projetor g_ϕ com parâmetros θ e ϕ, e uma rede de professor consistindo de um codificador $f_{\bar{\theta}}$ e um projetor $g_{\bar{\phi}}$ com parâmetros $\bar{\theta}$ e $\bar{\phi}$. Dada uma imagem x_i, duas visualizações $x_i^* = t(x_i)$ e $x_i^+ = t(x_i)$ são criadas usando transformações aleatórias $t \sim T$. O aluno computa a representação $y_i^* = f_\theta(x_i^*)$ e a projeção $z_i^* = g_\phi(y_i^*)$, enquanto o professor computa a representação $y_i^+ = f_{\bar{\theta}}(x_i^+)$ e a projeção $z_i^+ = g_\phi(y_i^+)$. O MoCo minimiza a perda do InfoNCE para aprender projeções que sejam semelhantes para duas visualizações da mesma imagem e diferentes das projeções de visualizações de outras imagens. A perda do MoCo é então definida como:

$$L_{MoCo}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}\text{InfoNCE}_\tau(z_i^*, z_i^+, Z_i^-),$$

em que as similaridades são calculadas usando o produto escalar $s_\tau(z^*, z) = z^\top z^*/\tau$ dividido por um hiperparâmetro de temperatura $\tau > 0$. O professor é atualizado por uma média móvel exponencial do estudante, ou seja,

$$\bar{\theta} \leftarrow \alpha\bar{\theta} + (1-\alpha)\theta$$
$$\bar{\phi} \leftarrow \alpha\bar{\phi} + (1-\alpha)\phi$$

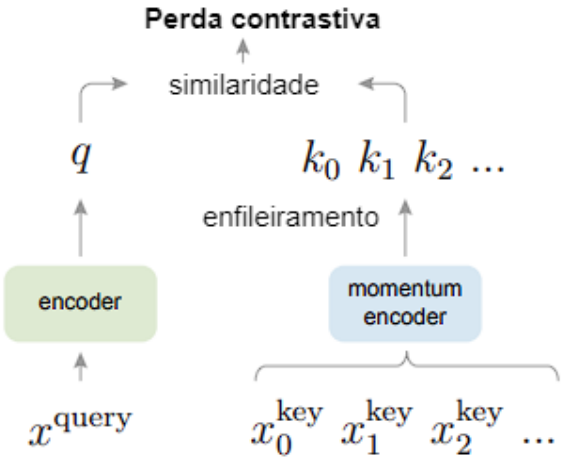


Figura 2.7. O MoCo treina um codificador de representações visuais combinando uma consulta codificada q a um dicionário de chaves codificadas usando uma perda contrastiva. O dicionário é dinamicamente definido pelas amostras de dados e construído como uma fila, permitindo um dicionário grande e consistente para aprender representações visuais. As chaves são codificadas por um codificador de progressão lenta, impulsionado por uma atualização de *momentum* com o codificador de consulta. Adaptado de [He et al., 2020a].

em que $\alpha \in [0,1]$ controla a taxa na qual os pesos da rede do professor são atualizados com os pesos da rede do estudante. O MoCo v2.0 [Chen et al., 2020a] introduz várias mudanças menores para melhorar ainda mais o desempenho *downstream* e superar o SimCLR. As mudanças mais notáveis incluem a substituição da camada de projeção linear do MoCo por um MLP, bem como a aplicação de um agendador de taxa de aprendizado cosseno e modificações adicionais. O novo cabeçalho MLP de 2 camadas foi adotado seguindo o SimCLR. Note-se que o MLP é usado apenas durante o treinamento não supervisionado e não é destinado a tarefas *downstream*. Em termos de aumentos de dados adicionais, o MoCo v2.0 também adota a operação de desfoque usada no SimCLR.

PIRL - *Pretext-Invariant Representation Learning*

Nas tarefas de pretexto, calculam-se representações de imagens transformadas para prever propriedades específicas, como ângulos de rotação ou permutações de *patches*. Embora isso incentive a covariância às transformações, o foco é representações semanticamente significativas e invariantes à transformação. Para isso, [Misra e Maaten, 2020] desenvolveram o PIRL, uma abordagem que refinou a formulação da perda da tarefa de pretexto e utiliza bancos de memória. A representação do modelo PIRL está expressa na Figura 2.8.

O objetivo do PIRL é treinar uma rede codificadora f_θ que mapeia imagens $x_i^{(1)} = x_i$ e imagens transformadas $x_i^{(2)} = t_\pi(x_i)$ para representações $y_i^{(1)}$ e $y_i^{(2)}$, respectivamente, que são invariantes às transformações utilizadas. Nesse caso, t_π denota uma transformação de quebra-cabeça, consistindo em uma permutação aleatória de *patches* de imagem, em que π é a permutação correspondente. A formulação da perda das tarefas de pretexto, enfatiza que o codificador aprende representações que contêm informações sobre a transformação, e não sobre a semântica. Sejam $z_i^{(1)} = g_\phi(f_\theta(x_i^{(1)}))$ e

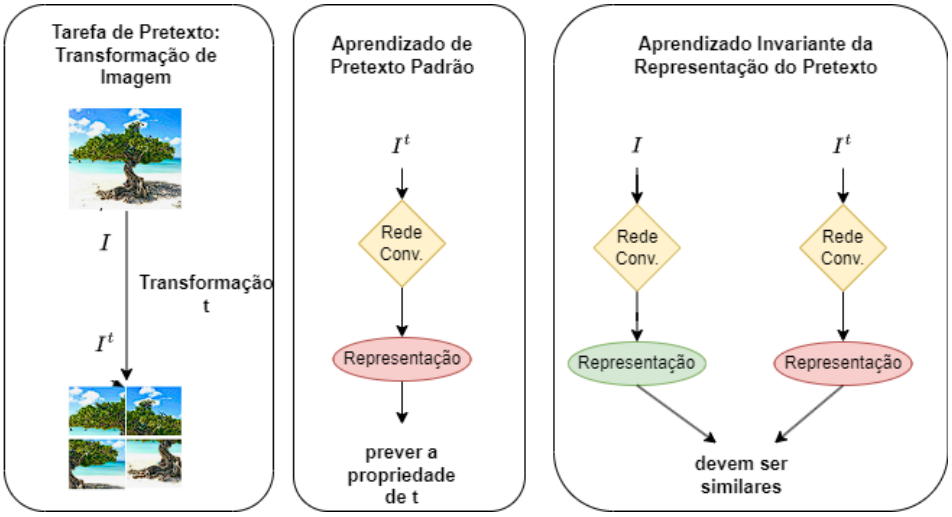


Figura 2.8. Muitas tarefas de pretexto no aprendizado auto-supervisionado envolvem transformar uma imagem, calcular sua representação transformada e prever propriedades da transformação. Enquanto essas representações geralmente variam com a transformação e podem ter pouca informação semântica, o PIRL aprende representações que são invariantes à transformação e mantêm informações semânticas. Adaptado de [Misra e Maaten, 2020].

$z_i^{(2)} = g_\psi(f_\theta(x_i^{(2)}))$ as projeções obtidas pelo codificador f_θ e dois projetores separados g_ϕ e g_ψ. A rede é treinada minimizando uma combinação convexa de dois estimadores contrastivos de ruído (NCE) [Gutmann e Hyvärinen, 2010]:

$$\mathscr{L}_{PIRL}(\theta,\phi,\psi) = \frac{1}{n}\sum_{i=1}^{n} \lambda \mathscr{L}_{NCE}(m_i, z_i^{(2)}, \bar{M}_i) + (1-\lambda)\mathscr{L}_{NCE}(m_i, z_i^{(1)}, \bar{M}_i)$$

em que m_i é uma projeção de um banco de memória correspondente à imagem original x_i, cada amostra positiva é atribuída a um conjunto aleatório de projeções negativas $\bar{M}_i$ de imagens diferentes de x_i obtidas do banco de memória, e $\lambda \in [0,1]$ é um hiperparâmetro. Em contraste com as tarefas de pretexto introduzidas anteriormente, a formulação da perda do PIRL não visa explicitamente prever propriedades particulares das transformações aplicadas, como rotação ou índices de *patches*. Em vez disso, é definida apenas em imagens e suas contrapartes transformadas correspondentes. O NCE aplica classificação binária a cada ponto de dados para distinguir amostras positivas e negativas. Nesse caso, a perda NCE é formulada como:

$$\mathscr{L}_{NCE}(m,z,\bar{M}) = -\log[h(m,z,\bar{M})] - \sum_{\bar{m}\in\bar{M}} \log[1-h(z,m,\bar{M})]$$

em que h modela a probabilidade de que (x_i, x_i') seja derivado de X como:

$$h(u,v,\bar{M}) = \frac{\exp(\text{scos}(u,v)/\tau)}{\exp(\text{scos}(u,v)/\tau) + \sum_{\bar{m}\in\bar{M}} \exp(\text{scos}(\bar{m},v)/\tau)}.$$

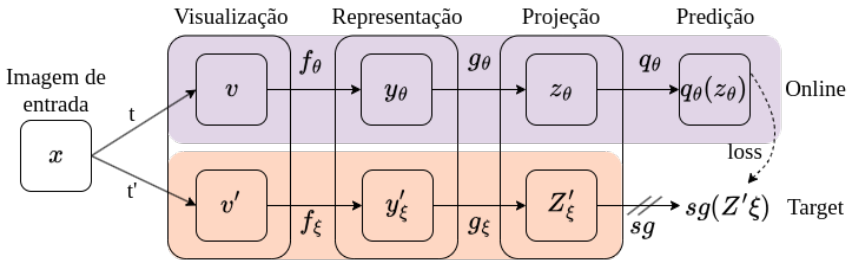


Figura 2.9. Arquitetura do Modelo BYOL. O BYOL minimiza uma perda de similaridade entre $q_\theta(z_\theta)$ e $sg(z_{\xi_0})$, em que θ são os pesos treinados, ξ é uma média móvel exponencial de θ e sg significa *stop-gradient*. Adaptado de [Grill et al., 2020].

BYOL - *Bootstrap Your Own Latent*

O BYOL [Grill et al., 2020] é inspirado na observação de que aprender representações ao prever representações fixas de uma rede alvo inicializada aleatoriamente evita o colapso da representação, apesar de apresentar desempenho inferior. Isso naturalmente implica uma arquitetura de professor-aluno, em que o professor fornece representações estáveis para o aluno aprender.

BYOL define duas redes diferentes: uma rede aluno e uma rede professor. A arquitetura é mostrada na Figura 2.9, a rede aluno e a rede professor consistem das seguintes partes:

- Rede aluno: codificador f_θ, projetor g_ϕ, preditor q_ψ
- Rede professor: codificador $f_{\bar{\theta}}$, projetor $g_{\bar{\phi}}$

O codificador f e o projetor g estão presentes tanto nas redes aluno quanto nas professor, enquanto o preditor q faz parte apenas da rede aluno.

Como os métodos de maximização de informação, os métodos professor-aluno aprendem representações aplicando diferentes transformações às imagens. Dada uma imagem x_i, o BYOL aplica transformações amostradas aleatoriamente $t \sim T$ para obter duas visualizações diferentes $x_{(1)i} = t(x_i)$ e $x_{(2)i} = t(x_i)$. A rede aluno calcula representações $y_{(j)i} = f_\theta(x_{(j)i})$, projeções $z_{(j)i} = g_\phi(y_{(j)i})$ e previsões $\hat{z}_{(j)i} = q_\psi(z_{(j)i})$ para ambas as visualizações $j \in \{1,2\}$. As visualizações também são alimentadas na rede professor para obter projeções alvo $\bar{z}_{(1)i} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_{(1)i}))$ e $\bar{z}_{(2)i} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_{(2)i}))$.

BYOL minimiza dois erros quadrados normalizados:

- entre a previsão da primeira visualização e a projeção alvo da segunda visualização
- entre a previsão da segunda visualização e a projeção alvo da primeira visualização

A função de perda final é:

$$L_{\text{BYOL}}^{\theta,\phi,\psi} = \frac{1}{n}\sum_{i=1}^{n}\left[\text{dnse}(\hat{z}_{(1)i}, \bar{z}_{(2)i}) + \text{dnse}(\hat{z}_{(2)i}, \bar{z}_{(1)i})\right].$$

A perda é mínima quando a similaridade de cosseno entre os vetores é 1. Assim, são aprendidas representações que são semelhantes para duas transformações diferentes.

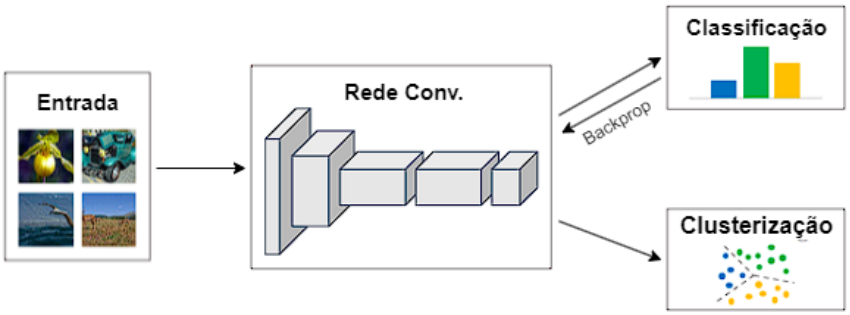


Figura 2.10. Modelo SimSiam em que características profundas são agrupadas iterativamente e as atribuições de cluster são usadas como pseudo-rótulos para aprender os parâmetros da rede convolucional. Adaptado de [Chen e He, 2021].

Em outras palavras, o conteúdo de informação nas representações aprendidas é maximizado.

Em cada etapa de treinamento, a perda é minimizada em relação a θ, ϕ e ψ. Ou seja, apenas os pesos do aluno são atualizados pelo gradiente da função de perda usando o otimizador LARS. Os pesos do professor são atualizados pela média móvel exponencial, ou seja,

$$\bar{\theta} \leftarrow \tau\bar{\theta} + (1-\tau)\theta, \quad \bar{\phi} \leftarrow \tau\bar{\phi} + (1-\tau)\phi,$$

em que $\tau \in [0,1]$ controla a taxa na qual os pesos da rede professor são atualizados com os pesos da rede aluno.

SimSiam - *Simple Siamese Representation Learning*

O SimSiam utiliza uma arquitetura e função de perda similares ao BYOL. No entanto, o professor e o aluno compartilham os mesmos parâmetros e, portanto, um codificador de *momentum* não é usado como nos métodos professor-aluno apresentados anteriormente.

Dado um lote de imagens X, para cada imagem x_i, duas visualizações $x_{(1)i} = t(x_i)$ e $x_{(2)i} = t(x_i)$ são criadas usando transformações aleatórias $t \sim T$ que são amostradas para cada imagem e cada visualização. Para cada uma dessas visualizações, um codificador Siamese f_θ calcula uma representação $y_{(j)i} = f(x_{(j)i})$ e um projetor Siamese g_ϕ calcula uma projeção $z_{(j)i} = g_\phi(y_{(j)i})$. Finalmente, a projeção é alimentada através de um preditor q_ψ para obter uma previsão $\hat{z}_{(j)i} = q_\psi(z_{(j)i})$.

O objetivo do preditor é prever a projeção da outra visualização. Portanto, a perda calcula a similaridade cosseno negativa entre a previsão da primeira visualização e a projeção da segunda visualização, e vice-versa, ou seja,

$$L_{\text{SimSiam}}^{\theta,\phi,\psi} = -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\text{scos}(\hat{z}_{(1)i}, \text{sg}(z_{(2)i})) + \text{scos}(\hat{z}_{(2)i}, \text{sg}(z_{(1)i}))\right),$$

em que $\text{sg}(\cdot)$ é o operador de *stop-gradient* que impede que os gradientes sejam retropropagados por esse ramo do grafo computacional.

O codificador f_θ é implementado como uma ResNet. O projetor g_ϕ e o preditor q_ψ são MLPs. Os autores mostram empiricamente que um preditor é crucial para evitar o colapso. Os autores [Chen e He, 2021] argumentam que o gradiente da perda simetrizada com um preditor que é a identidade está na mesma direção que o gradiente da perda simetrizada entre as duas projeções, de modo que a operação de *stop-gradient* é cancelada, levando assim ao colapso da representação. Usar um preditor aleatório também não funciona e [Chen e He, 2021] argumentam que o preditor deve sempre aprender as representações mais recentes.

Outro ingrediente importante para o método é a normalização em lote, que é usada tanto para f_θ quanto para g_ϕ. Além disso, os autores experimentam com o objetivo de treinamento substituindo-o pela perda de entropia cruzada. Suas experiências mostram que isso também funciona, no entanto, o desempenho é pior. A principal vantagem do SimSiam é que o treinamento não requer lotes grandes, permitindo o uso de *Stochastic Gradient Descent*.

2.5. Aprendizado baseado em *clusters*

Outra estratégia, denominada *Cluster Discrimination*, parte do princípio de que amostras do mesmo agrupamento (*cluster*) devem ser representadas próximas umas das outras, enquanto amostras de agrupamentos diferentes devem ter representações distantes. Essa abordagem tem demonstrado alto desempenho em tarefas subsequentes, especialmente em problemas de classificação linear. No entanto, a eficácia dessas técnicas pode depender de fatores como a escolha apropriada de hiperparâmetros e a qualidade dos dados de treinamento. Portanto, ao implementar essas abordagens, é crucial considerar esses aspectos para garantir resultados satisfatórios em diversas aplicações de aprendizado de máquina.

Classificação de imagens pede que o modelo categorize imagens corretamente e a representação de imagens na mesma categoria deve ser semelhante. Portanto, a motivação é aproximar imagens similares no espaço de incorporação (*embedding*). No aprendizado supervisionado, esse processo de aproximação é realizado por meio da supervisão de rótulos; no entanto, no aprendizado auto-supervisionado, não há rótulos. Para resolver o problema dos rótulos, DeepCluster [Caron et al., 2018] propõe utilizar clusterização para gerar pseudo-rótulos e pede a um discriminador para prever os rótulos das imagens. O treinamento pode ser formulado em duas etapas. Na primeira etapa, o DeepCluster usa K-means para agrupar a representação codificada e produz pseudorrótulos para cada amostra. Então, na segunda etapa, o discriminador prevê se duas amostras são do mesmo cluster e realiza o backpropagation para o codificador. Essas duas etapas são realizadas iterativamente.

O Local Aggregation (LA) [Zhuang et al., 2019] avançou os limites do método baseado em clusterização. Ele aponta várias desvantagens do *DeepCluster* e faz as correspondentes otimizações. Primeiro, no *DeepCluster*, as amostras são atribuídas a clusters mutuamente exclusivos, mas o LA identifica vizinhos separadamente para cada exemplo. Segundo, o *DeepCluster* otimiza uma perda discriminativa de entropia cruzada, enquanto o LA emprega uma função objetivo que otimiza diretamente uma métrica de soft-clustering local. Essas duas mudanças aumentam substancialmente o desempenho

da representação do LA em tarefas subsequentes.

Um trabalho semelhante ao LA é o VQ-VAE [Razavi et al., 2019] para superar a deficiência tradicional do VAE de gerar imagens de alta fidelidade, o VQ-VAE propõe quantizar vetores. Para a matriz de características codificada a partir de uma imagem, o VQ-VAE substitui cada vetor unidimensional na matriz pelo mais próximo em um dicionário de incorporação. Esse processo é de certa forma semelhante ao que o LA está fazendo. Modelos alternativos ao VQ-VAE, como proposto por [Peng et al., 2021], usa um módulo de atenção estrutural dentro da rede de geração de textura, em que o módulo utiliza a informação estrutural para capturar correlações distantes. Desta forma, reutlizando o VQ-VAE para calcular duas perdas de características, que ajudam a melhorar a coerência da estrutura e o realismo da textura, respectivamente.

Apesar do sucesso anterior do aprendizado contrastivo baseado em discriminação de clusters, o paradigma de treinamento em duas etapas é demorado e de baixo desempenho comparado aos métodos posteriores baseados em discriminação de instâncias, incluindo CMC [Tian et al., 2020], MoCo [He et al., 2020b] e SimCLR [Chen et al., 2020b]. Esses métodos baseados em discriminação de instâncias eliminaram a etapa de agrupamento lento e introduziram estratégias eficientes de *data augmentation* para aumentar o desempenho. Em virtude desses problemas, os autores do SwAV [Caron et al., 2020] trazem ideias de realizar clusterização *online* e estratégias de *data augmentation* para a abordagem de discriminação de clusters. O SwAV propõe objetivos contrastivos de predição trocada para lidar com o aumento de dados *multiview*. A intuição é que, dados alguns protótipos (agrupados), diferentes visualizações das mesmas imagens devem ser atribuídas aos mesmos protótipos. O SwAV chama essa “atribuição"de “códigos". Para acelerar o cálculo dos códigos, os autores do SwAV desenvolvem uma estratégia de cálculo *online*. Com base no SwAV, um modelo auto-supervisionado [Goyal et al., 2021] com 1,3 bilhão de parâmetros foi treinado em 1 bilhão de imagens da web coletadas do Instagram.

2.5.1. *DeepCluster*

Um dos primeiros métodos a implementar a ideia de agrupamento para aprendizado de representação é o DeepCluster [Caron et al., 2018]. Ele alterna entre a criação de pseudo-rótulos via atribuições de clusters e o ajuste da representação para classificar imagens de acordo com seus rótulos inventados. A motivação por trás disso é aumentar o desempenho de arquiteturas convolucionais que já exibem um forte viés indutivo, já que essas tendem a se sair razoavelmente bem com pesos inicializados aleatoriamente. No geral, os autores propõem alternar repetidamente entre as seguintes duas etapas para melhorar ainda mais a rede codificadora:

1. Agrupar as representações $y_i = f_\theta(x_i)$ produzidas pelo estado atual do codificador f_θ em k clusters (por exemplo, usando agrupamento k-means);

2. Usar as atribuições de clusters da etapa 1 como pseudo-rótulos β_i para supervisão e atualizar os pesos, ou seja,

$$L_{\text{DeepCluster}}(\theta, \psi) = \frac{1}{n} \sum_{i=1}^{n} d_{\text{classification}}(q_\psi(y_i), \beta_i),$$

em que uma rede preditora q_ψ tenta prever as atribuições de cluster das representações $y_i = f_\theta(x_i)$.

Em seus experimentos, os autores utilizam uma rede AlexNet padrão [Krizhevsky et al., 2012] com K-means.

2.5.2. SwAV - *Swapping Assignments Between Multiple Views of the Same Image*

[Caron et al., 2020] propõem um algoritmo alternativo, chamado SwAV, que promove ao mesmo tempo consistência entre as atribuições de clusters em diferentes visualizações. Ao contrário do DeepCluster, o SwAV é uma abordagem de agrupamento *online*, ou seja, não alterna entre uma atribuição de cluster e uma etapa de treinamento. Uma rede codificadora f_θ é usada para calcular as representações de imagem $y^{(1)}$ e $y^{(2)}$ de duas visualizações da mesma imagem x. Essas representações são então mapeadas para um conjunto de protótipos parametrizados $C_\psi = [c_1, \ldots, c_k]$, resultando em códigos correspondentes $q^{(1)}$ e $q^{(2)}$. Em seguida, é abordado um problema de previsão trocada, em que os códigos derivados de uma visualização são previstos usando a codificação da segunda visualização. Para alcançar isso, minimiza-se L_{SwAV}, em que $\ell(q,y) = dce(q, \text{softmax}_\tau(C^\top y))$ quantifica a correspondência entre a representação y e o código q para uma temperatura $\tau > 0$. Embora o SwAV se beneficie do aprendizado contrastivo, não requer o uso de um grande banco de memória ou uma rede de *momentum*.

Além deste método, os autores também propõem a técnica de aumento chamada multi-crop, que também foi usada para DINO. Em vez de usar duas visualizações com resolução completa, é usada uma mistura de visualizações com diferentes resoluções. Nesta abordagem, múltiplas transformações são comparadas usando transformações consideravelmente menores, o que leva a uma melhoria adicional de métodos anteriores como SimCLR e DeepCluster.

2.6. Casos de uso para detecção de intrusões e provisão de QoS

Esta seção explora o uso do aprendizado auto-supervisionado em aplicações de redes de computadores dinâmicas, focando a detecção de intrusão em dispositivos IoT e na provisão de Qualidade de Serviço (QoS). Com o aumento de dispositivos conectados, as redes enfrentam desafios contínuos em segurança e eficiência. A detecção de intrusão é essencial para identificar e mitigar ameaças cibernéticas em tempo real, enquanto a provisão de QoS assegura que serviços críticos mantenham desempenho e confiabilidade em ambientes dinâmicos. O aprendizado auto-supervisionado utiliza grandes volumes de dados não rotulados para aprender representações úteis e tomar decisões inteligentes, sendo uma abordagem promissora para enfrentar esses desafios. A seção discute métodos inovadores e resultados que demonstram o uso dessas técnicas.

2.6.1. Detecção de intrusões em Internet das Coisas

A detecção de intrusões em Internet das Coisas (IoT) beneficia-se significativamente do uso de aprendizado auto-supervisionado, aproveitando-se da habilidade dessa abordagem de manipular grandes volumes de dados não rotulados [Barbosa et al., 2024]. Neste contexto, aprendizagem auto-supervisionado permite a captura de características essenciais e padrões comportamentais sem a necessidade de supervisão explícita, o que é ideal em cenários de IoT, onde etiquetar dados pode ser impraticável. Com modelos pré-treinados neste vasto conjunto de informações, o ajuste fino requer apenas um volume reduzido de dados rotulados, tornando o processo não apenas econômico, mas também mais ágil na resposta a possíveis ameaças. Treinar os modelos nesta configuração pode não apenas melhorar a eficácia do modelo na detecção de intrusões em ambientes IoT, mas também aprimorar a segurança em outros tipos de redes, evidenciando a versatilidade e abrangência dessa abordagem.

FeCo - ***Federated Contrastive Learning Framework***

O trabalho de [Wang et al., 2022a] apresenta uma solução para melhorar a capacidade de detecção de intrusão em redes IoT por meio de aprendizado contrastivo federado. A solução proposta é o FeCo, um arcabouço de aprendizado contrastivo federado que coordena dispositivos IoT na rede para aprender modelos de detecção de intrusão de forma conjunta. O FeCo emprega técnicas de aprendizado profundo e contrastivo para extrair e comparar representações de dados de tráfego de rede, identificando padrões anômalos que possam indicar atividades suspeitas. A implementação do FeCo envolve a colaboração entre os dispositivos de IoT, que compartilham localmente informações sobre o tráfego de rede com um agregador de modelo central.

Conforme mostrado na Figura 2.11, o diagrama de fluxo do FeCo inicia com a extração de características dos dados de tráfego de rede dos dispositivos de IoT, seguido pelo aprendizado contrastivo para comparar e distinguir entre padrões benignos e maliciosos. Posteriormente, o modelo treinado é utilizado para a detecção em tempo real de atividades suspeitas. A implementação do aprendizado federado permite a melhoria contínua do modelo, sem comprometer a privacidade dos dados dos dispositivos individuais, resultando em um sistema eficaz de detecção de intrusões para redes de IoT.

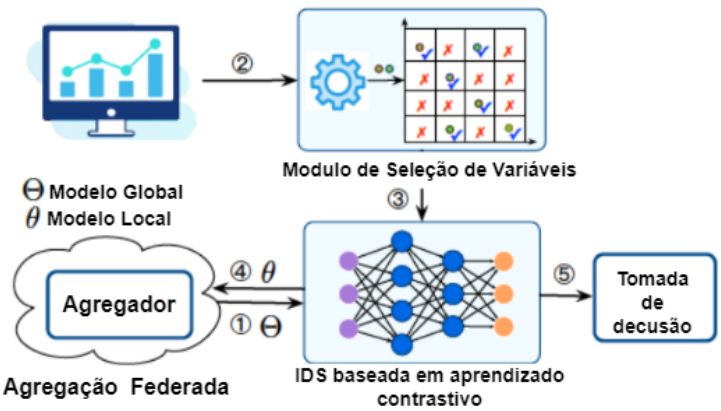


Figura 2.11. Diagrama de fluxo da proposta FeCo. A proposta implementa o aprendizado contrastivo federado. Inspirado em [Wang et al., 2022a].

O IDS baseado em aprendizado contrastivo é o bloco de construção do FeCo. O

objetivo ao implantar o aprendizado contrastivo é treinar um modelo que produza representações semelhantes para todas as instâncias normais de tráfego e torne as representações de intrusão distantes das representações normais. Especificamente, o aprendizado contrastivo treina um modelo de rede neural artificial (*Artificial Neural Network* - ANN) que recebe $x_i \in \mathbb{R}^d$ como entrada e gera uma nova representação $z_i \in \mathbb{R}^o$. O modelo de ANN pode ser representado por uma função $f_\theta : \mathbb{R}^d \to \mathbb{R}^o$, em que θ denota os parâmetros do modelo. O modelo de ANN do FeCo consiste em quatro camadas: a camada de entrada, duas camadas ocultas e, finalmente, a camada de saída.

Como o objetivo do aprendizado contrastivo é maximizar a similaridade entre v_i e v_j e minimizar a similaridade entre v_i e u_m para $m \in [M]$ (define-se $[M] := \{1, 2, ..., M\}$). Para atingir esse objetivo, usa-se uma função de perda L_{ij}, dada por:

$$L_{ij} = -\log \frac{\exp(v_i^T v_j/\tau)}{\exp(v_i^T v_j/\tau) + \sum_{m=1}^{M} \exp(v_i^T u_m/\tau)}, \quad ,$$

em que $\tau \in [0,1]$ representa um parâmetro de temperatura, $v_i = f_\theta(x_i)$ representa a saída de uma entrada benigna x_i e $u_i = f_\theta(x_i)$ a saída de uma entrada de intrusão x_i. Após o treinamento, para medir a similaridade $S(x_{\text{test}_j})$ entre um fluxo de tráfego iminente x_{test_j} e o *template* normalizado utiliza-se o estimador de similaridade cosseno:

$$S(x_{\text{test}_j}) = \frac{\bar{z}^T f_\theta(x_{\text{test}_j})}{\|\bar{z}\| \times \|f_\theta(x_{\text{test}_j})\|}.$$

O escore de similaridade $S(x_{\text{test}_j})$ varia de 0 a 1. É necessário haver um limiar $0 \leq \rho \leq 1$ para determinar se x_{test_j} é uma anomalia ou não. O FeCo foi construído incorporando o IDS baseado em aprendizado por contraste no arcabouço de aprendizado federado. Nesse caso, cada cliente participa no processo de Aprendizado Federado (FL) fornecendo sua atualização de parâmetro de modelo. Para isso, utilizaram o algoritmo FedAVG [McMahan et al., 2017] para agregar as atualizações de múltiplos clientes. No passo de tempo t, o agregador de modelo A computa o modelo global Θ_t por:

$$\Theta_t = \Theta_{t-1} + \sum_i c_i \cdot (\theta_i - \Theta_{t-1}),$$

em que θ_i são os parâmetros de modelo locais no cliente i e c_i é um coeficiente de peso. No caso da solução proposta, c_i é baseado no tamanho do conjunto de dados de treinamento local no cliente i. Especificamente, c_i é definida como a razão entre o tamanho do conjunto de dados de treinamento local no cliente i e o número total de amostras de treinamento em todos os clientes selecionados.

O desempenho do FeCo foi avaliado em comparação com outros modelos de detecção de intrusões em redes de IoT, utilizando o conjunto de dados NSL-KDD. Entre os modelos de *benchmark* utilizados estão o *Support Vector Machine* (SVM), o *Random Forest* (RF), o *Multi-Layer Perceptron* (MLP), o *Deep Neural Network* (DNN), o *Autoencoder* (AE), o *Variational Autoencoder* (VAE) e o *Generative Adversarial Network* (GAN). Os resultados mostraram que o FeCo superou outros modelos em termos de detecção de

intrusões, com uma taxa de detecção de 99,2% e uma taxa de falsos positivos de 0,8%. Além disso, o FeCo apresentou uma redução significativa na sobrecarga de comunicação em comparação com outros modelos, com uma redução de 99,9% na quantidade de dados transmitidos durante o treinamento. O FeCo também demonstrou sua escalabilidade, sendo capaz de lidar com um grande número de dispositivos de IoT em uma rede. Esses resultados indicam que o FeCo é uma solução eficaz e viável para aprimorar a segurança em redes de IoT.

Aprendizado Contrastivo sobre Características de Fourier Aleatórias

[Lopez-Martin et al., 2023] utilizam aprendizado contrastivo e características de Fourier aleatórias (RFFs). Conforme mostrado na Figura 2.12, a técnica mapeia as características da rede e os rótulos para um espaço comum, em que é possível medir a similaridade para realizar a classificação. O modelo é especialmente otimizado para identificar ataques desconhecidos, utilizando técnicas de regularização L2 e contrastiva para evitar o sobreajuste. Testes com os conjuntos de dados públicos demonstram que o modelo proposto supera as alternativas existentes na detecção de novos ataques, podendo ser executado em dispositivos de baixos recursos. A diversidade intra-classe e similaridade inter-classe em tráfego de rede são abordados em [Yue et al., 2022]. O método utiliza mascaramento aleatório de sequências de pacotes de rede para criar tarefas contrastivas, calculando a perda contrastiva para medir distâncias intra-classe e inter-classe. Experimentos com conjuntos de dados reais e de *benchmark* mostram melhorias significativas na precisão e na taxa de detecção de intrusões em ambientes complexos de rede. De forma semelhante, [Li et al., 2024a] utiliza o aprendizado contrastivo para melhorar a distinção entre tráfego benigno e malicioso no espaço de representação, resultando em maior precisão na detecção de ataques desconhecidos.

Detecção de Anomalias em Redes com Aprendizado Auto-Supervisionado

A proposta AOC-IDS visa a detecção de intrusões em tempo real em ambientes em que os comportamentos dos sistemas e as estratégias de ataque evoluem constantemente[Zhang et al., 2024]. O AOC-IDS integra um módulo de detecção de anomalias (ADM) e um arcabouço em tempo real que permite adaptação contínua. O ADM utiliza um *autoencoder* (AE) com uma função de perda contrastiva personalizada, chamada *Cluster Repelling Contrastive (CRC) loss*, que melhora a capacidade de representação dos dados. A estrutura online do AOC-IDS gera pseudo-rótulos automaticamente para atualizar periodicamente o ADM, eliminando a necessidade de intervenção humana para rotulagem e facilitando a adaptação autônoma do sistema a novos dados sem rótulos. De forma semelhante, o ContraMTD é um método não supervisionado para detecção de tráfego malicioso, também baseado em aprendizado contrastivo[Han et al., 2024]. O ContraMTD extrai características de comportamento local e interação global do tráfego normal, utilizando aprendizado contrastivo para aprender a relação entre elas e detectar anomalias. O ContraMTD é composto por cinco módulos principais: agregação de tráfego de rede, extração de características de comportamento local, extração de características de interação global, aprendizado contrastivo e detecção de anomalias. Utiliza técnicas de agrupamento para formar pares de amostras positivas e negativas, e um gráfico de interação de *hosts* com uma rede neural de atenção de borda dupla (DE-GAT) para capturar características da topologia e atributos do gráfico, realizando a detecção de anomalias em

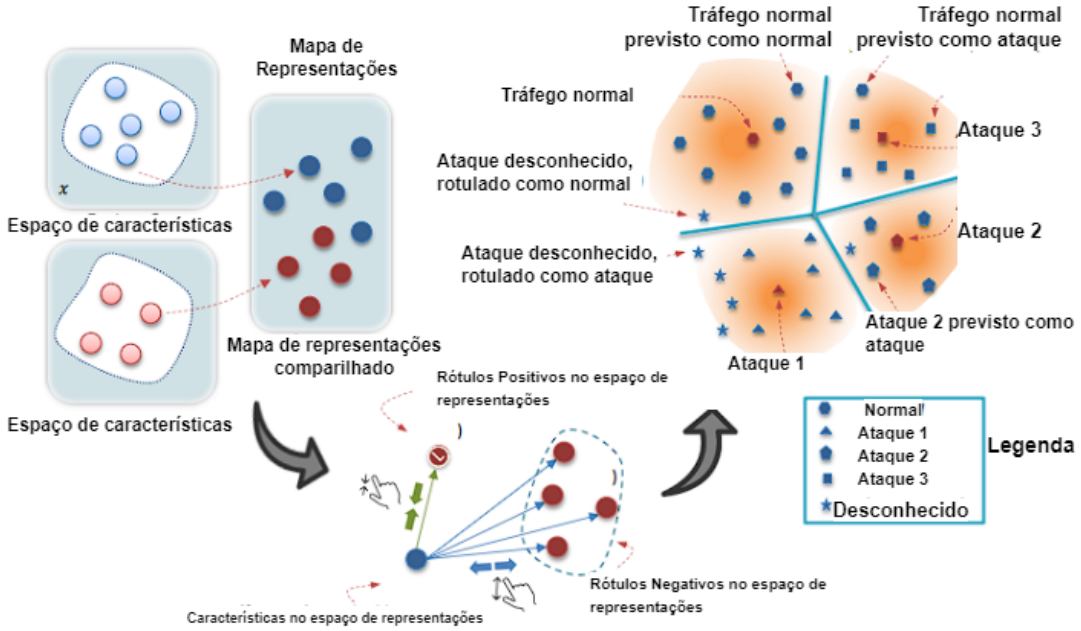


Figura 2.12. As características de amostra e os rótulos são mapeados para o mesmo espaço de incorporação. Cada rótulo cria um hipercone de separação usado para classificar as amostras. Por exemplo, ataques conhecidos e tráfego normal são representados por hipercones separados. Um ataque desconhecido pode ser classificado corretamente como um ataque ou incorretamente como tráfego normal, dependendo de onde cai no espaço de incorporação. Adaptado de [Lopez-Martin et al., 2023].

múltiplas rodadas para melhorar a precisão.

A **detecção de malware** também pode se beneficiar do aprendizado contrastivo. o EVOLIoT [Dib et al., 2022], apresenta uma estrutura de aprendizado contrastivo auto-supervisionado para detectar e caracterizar variantes evolutivas de *malware* em IoT. O método combate o "desvio de conceito" (*concept drift)* e limitações na classificação de *malware* entre famílias. Utilizando representações semânticas de binários de *malware* de IoT, o sistema diferencia amostras evoluídas sem a necessidade de rótulos caros. Avaliações mostram que o sistema melhora a precisão na identificação de variantes e na preservação de informações semânticas em um cenário de *malware* de IoT em rápida evolução. [Yang et al., 2022] apresenta um método para detecção e classificação de *malware* em dispositivos Android baseado em aprendizado contrastivo. Utilizando codificação de características sem *token* e um modelo TextCNN, o sistema extrai características variáveis dos dados de entrada. O treinamento do modelo é realizado com a técnica *Bootstrap Your Own Latent* (BYOL) que não depende de amostras negativas, melhorando a precisão e a robustez do detector.

A solução como a de [Kye et al., 2022] busca detectar anomalias em redes de computadores utilizando aprendizado auto-supervisionado motivado pela necessidade de identificar anomalias extremas que podem causar danos significativos. A solução propõe uma abordagem hierárquica com múltiplos estágios de detecção baseados no nível de anormalidade. Utilizando o espaço oculto do *autoencoder*, a solução é treinada com sinais de auto-supervisão, eliminando a dependência de dados anormais escassos. Avaliada

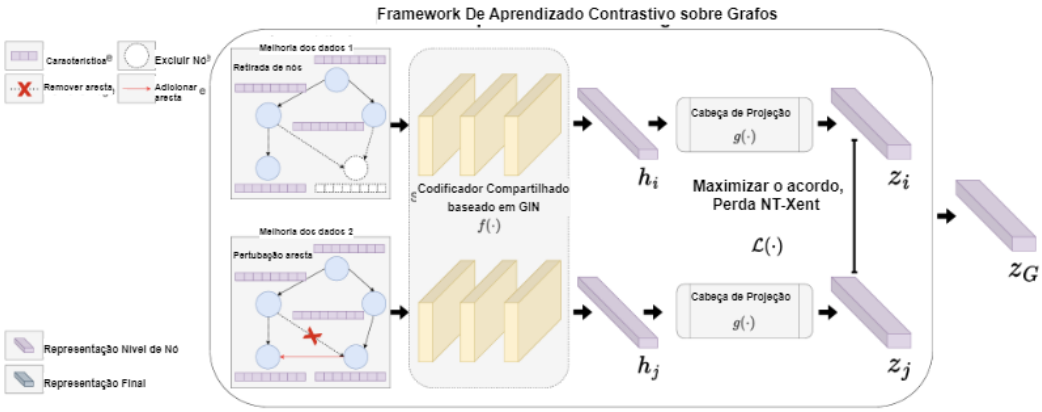


Figura 2.13. Pré-treinamento no aprendizado contrastivo de grafos maximizando a concordância entre visualizações aumentadas do mesmo grafo, utilizando um codificador rede de isomorfismo de grafo (GIN), uma cabeça de projeção não linear, e a função de perda contrastiva NT-Xent para obter representações finais robustas. Adaptado de [Gao et al., 2022].

em conjuntos de dados populares, o método se destaca na detecção eficiente de anomalias extremas em redes de computadores.

Outra aplicação de aprendizado auto-supervisionado trata da **detecção de anomalias em dados de grafo**, uma área crucial para aplicações como segurança de redes e detecção de fraudes. Os métodos tradicionais, inadequados para estruturas complexas não euclidianas, demandam o desenvolvimento de abordagens eficazes. O SL-GAD (Self-Supervised Learning for Graph Anomaly Detection) [Zheng et al., 2021] é proposto para superar essas limitações, adotando aprendizado auto-supervisionado. Gerando duas visualizações de subgrafo do nó alvo, o SL-GAD utiliza reconstrução de atributos generativos e aprendizado contrastivo multi-visão para identificar anomalias.

O SL-GAD se destaca ao construir diferentes subgrafos contextuais (visões) com base em um nó alvo e ao empregar estratégias de aprendizado auto-supervisionado para fazer comparações e obter a pontuação de anomalia para cada nó. Ele utiliza um codificador de rede neural de grafo (GNN) para aprender a representação latente de cada nó a partir das diferentes visões amostradas. Em seguida, os módulos de regressão de atributos generativos e aprendizado contrastivo multi-visão são empregados para explorar as informações disponíveis de forma auto-supervisionada.

Essa abordagem inovadora permite que o SL-GAD capture anomalias em dados de grafo de forma mais eficaz do que os métodos existentes. Ele supera as limitações dos métodos rasos que não conseguem capturar a complexa interdependência dos dados de grafo e dos métodos de autoencoder de grafo que não conseguem explorar totalmente as informações contextuais como sinais de supervisão para detecção eficaz de anomalias.

Uma abordagem para classificação de *malware* em arquivos executáveis usando aprendizado contrastivo em grafos, não supervisão é proposto por [Gao et al., 2022]. O pré-treinamento é realizado maximizando a concordância entre duas visualizações aumentadas do mesmo grafo usando a perda contrastiva no espaço latente. A estrutura consiste em quatro componentes principais: (1) *Aumento de Dados de Grafos*, em que os dados de grafos G são aumentados para gerar dois grafos relacionados $\hat{G}_i$ e $\hat{G}_j$ como pa-

res positivos; (2) *Codificador baseado em rede de isomorfismo de grafo (GIN)*, utilizado para gerar representações vetoriais dos grafos, com três camadas e uma camada oculta de 64 dimensões, em que a função de leitura soma as incorporações de todos os nós para obter as representações iniciais h_i e h_j; (3) *Cabeça de Projeção*, uma transformação não linear que mapeia as representações aumentadas para outro espaço latente, em que a perda contrastiva é calculada, e z_i e z_j são obtidos aplicando um perceptron de duas camadas (MLP); (4) *Função de Perda Contrastiva*, que maximiza a consistência entre pares positivos z_i e z_j e minimiza entre pares negativos, utilizando a perda de entropia cruzada normalizada pela temperatura (NT-Xent) para obter uma representação final do grafo z_G.

2.6.2. Qualidade de Serviço (QoS)

A **recomendação de serviços com base na Qualidade de Serviço (QoS)** tem despertado grande interesse, especialmente no que diz respeito à criação de matrizes de fatoração e à definição de conjuntos de dados para avaliar algoritmos. No entanto, a literatura existente tem se concentrado principalmente na precisão da previsão (MAE/RMSE), negligenciando os tempos de treinamento e invocação, cruciais para a implantação desses algoritmos em dispositivos de borda com recursos limitados. Além disso, ao contrário de cenários estáticos, os fatores de QoS, como tempo de resposta e vazão, são dinâmicos, requerendo um retreinamento frequente dos modelos. Para enfrentar esses desafios, White et al. propuseram uma abordagem que reduz o tempo de treinamento da previsão de QoS utilizando um empilhamento de autocodificadores, adequado para dispositivos de computação em borda, o que facilita a análise de ambientes dinâmicos e influencia a composição efetiva de serviços [White et al., 2019].

O empilhamento de autocodificadores funciona comprimindo os dados de entrada para a camada oculta e, em seguida, decodificando-os. Com essa abordagem, hierarquias úteis são capturadas, levando a um melhor desempenho. O modelo utiliza múltiplas camadas ocultas, treinadas de forma gulosa para obter parâmetros. A primeira camada destaca características de primeira ordem, como bordas, enquanto camadas subsequentes aprendem características de ordem superior. Assim, o autocodificador empilhado estende o modelo, gerando representações mais ricas e adaptadas às necessidades dinâmicas dos sistemas de recomendação de serviços baseados em QoS.

Yin *et al.* abordam a previsão da Qualidade de Serviço (QoS) em serviços de cidades inteligentes [Yin et al., 2023]. O desafio é prever os valores de QoS ausentes, considerando a variabilidade das condições de rede e o estado dos servidores, que aumentam a dimensão e a complexidade dos dados, além de intensificar o problema de esparsidade dos dados. Os autores propõem o CLpred, um método baseado em aprendizado contrastivo que utiliza uma sequência temporal de dados de QoS. O CLpred emprega um codificador *transformer*, que consiste em várias camadas de atenção própria multi-cabeças e redes *feed-forward* posicionais, permitindo modelar a sequência temporal dos dados de QoS. O codificador *transformer* do CLpred é capaz de capturar interações complexas entre usuários e serviços ao longo do tempo, proporcionando representações mais eficientes dos dados. O aprendizado contrastivo no CLpred envolve a criação de amostras positivas e negativas por meio de técnicas de aumento de dados, como recorte, mascaramento e reordenação de sequências de QoS. Essas técnicas ajudam a amplificar os dados e reduzir a esparsidade. O modelo é treinado usando uma função de perda contrastiva, que ajusta as

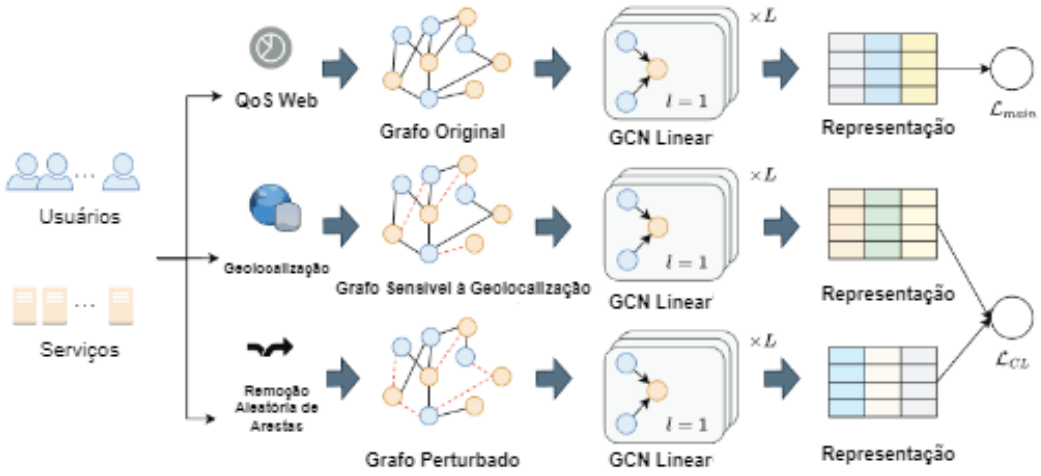


Figura 2.14. O arcabouço *QoS-aware Graph Contrastive Learning* (QAGCL) para recomendação de serviços web. O processo começa com a pré-processamento dos dados de invocação de usuários e serviços para formar um grafo inicial baseado em valores de QoS. Vistas adicionais do grafo são então construídas usando informações de geolocalização e exclusão aleatória de arestas. As incorporações iniciais são processadas por diferentes Redes Convolucionais de Grafos (GCNs) para os grafos aumentados, com uma incorporação usada para recomendação e as outras duas para aprendizado contrastivo. Adaptado de [Choi e Ryu, 2023].

representações dos dados para que amostras positivas, originadas da mesma sequência de usuário, fiquem mais próximas, enquanto amostras negativas, originadas de sequências diferentes, fiquem mais distantes.

Técnicas de aprendizado contrastivo para melhorar a recomendação e a previsão de QoS em serviços web, abordando problemas de esparsidade de dados e cold-start são explorados em [Choi e Ryu, 2023, Zhu et al., 2023]. Choi *et al.* propõem o *QoS-aware graph contrastive learning* (QAGCL), um modelo que utiliza redes convolucionais de grafos e aprendizado contrastivo para integrar informações contextuais, como geolocalização, em grafos aumentados, mostrado na Figura 2.14. Este modelo constrói vistas contextualmente aumentadas para aprender incorporações de usuários e serviços, melhorando a precisão das recomendações de serviços mesmo com interações limitadas. Os experimentos demonstram que o QAGCL supera modelos existentes em termos de precisão de recomendação, especialmente em condições de alta esparsidade de dados. O segundo artigo introduz o BGCL, uma rede bi-subgrafo baseada em aprendizado contrastivo para prever QoS em situações de *cold-start*. O BGCL gera diferentes perspectivas de subgrafos de vizinhança de usuários e serviços a partir de grafos bipartidos esparsos. Em seguida, utiliza aprendizado contrastivo de grafos e mecanismos de agregação de atenção para aprender incorporações de usuários e serviços. Essas incorporações são então alimentados em uma rede perceptora multicamadas para prever valores de QoS. Resultados experimentais mostram que o BGCL supera vários modelos existentes em termos de precisão de previsão, demonstrando eficácia em ambientes de baixa densidade de dados.

A classificação de tráfego de rede criptografado é um desafio crítico para a gestão de redes, qualidade de serviço (QoS) e segurança de redes. Com o crescimento das aplica-

Tabela 2.2. Comparação de trabalhos com foco em detecção de intrusões e provisão de QoS.

Referência	Descrição
[Wang et al., 2022a]	**FeCo - Federated Contrastive Learning Framework:** Solução para melhorar a detecção de intrusões em redes IoT por meio de aprendizado contrastivo federado, coordenando dispositivos IoT para aprender modelos de detecção de intrusão de forma conjunta.
[Lopez-Martin et al., 2023]	**Aprendizado Contrastivo sobre Características de Fourier Aleatórias:** Técnica que mapeia as características da rede e os rótulos para um espaço comum para medir a similaridade e realizar a classificação, especialmente otimizada para identificar ataques desconhecidos.
[Yue et al., 2022]	Técnica de aprendizado contrastivo utilizando mascaramento aleatório de sequências de pacotes de rede para criar tarefas contrastivas, melhorando a precisão e a taxa de detecção de intrusões em ambientes complexos de rede.
[Li et al., 2024a]	Aprendizado contrastivo para melhorar a distinção entre tráfego benigno e malicioso no espaço de representação, resultando em maior precisão na detecção de ataques desconhecidos.
[Zhang et al., 2024]	**AOC-IDS:** Sistema de detecção de intrusões em tempo real com um módulo de detecção de anomalias (ADM) e um framework em tempo real que permite adaptação contínua usando um autoencoder com uma função de perda contrastiva personalizada.
[Han et al., 2024]	**ContraMTD:** Método não supervisionado para detecção de tráfego malicioso, extraindo características de comportamento local e interação global do tráfego normal usando aprendizado contrastivo.
[Dib et al., 2022]	**EVOLIoT:** Estrutura de aprendizado contrastivo auto-supervisionado para detectar e caracterizar variantes evolutivas de malware em IoT, utilizando representações semânticas de binários de malware.
[Yang et al., 2022]	Método para detecção e classificação de malware em dispositivos Android baseado em aprendizado contrastivo, utilizando codificação de características sem token e um modelo TextCNN.
[Kye et al., 2022]	Abordagem hierárquica para detecção de anomalias em redes de computadores utilizando aprendizado auto-supervisionado com múltiplos estágios de detecção baseados no nível de anormalidade.
[Zheng et al., 2021]	**SL-GAD:** Aprendizado auto-supervisionado para detecção de anomalias em dados de grafo, utilizando reconstrução de atributos generativos e aprendizado contrastivo multi-visão para identificar anomalias.
[Gao et al., 2022]	Abordagem para classificação de malware em arquivos executáveis usando aprendizado contrastivo em grafos não supervisionado, maximizando a concordância entre visualizações aumentadas do mesmo grafo.
[White et al., 2019]	Redução do tempo de treinamento da previsão de QoS usando um empilhamento de autocodificadores, adequado para dispositivos de computação em borda.
[Yin et al., 2023]	**CLpred:** Método baseado em aprendizado contrastivo para previsão de QoS em serviços de cidades inteligentes, utilizando um codificador transformer para modelar a sequência temporal dos dados de QoS.
[Choi e Ryu, 2023]	**QAGCL:** Modelo de aprendizado contrastivo de grafos para recomendação de serviços web, integrando informações contextuais como geolocalização.
[Zhu et al., 2023]	**BGCL:** Rede bi-subgrafo baseada em aprendizado contrastivo para prever QoS em situações de cold-start, utilizando mecanismos de agregação de atenção para aprender embeddings de usuários e serviços.
[Tian et al., 2022]	Método de identificação de tráfego criptografado baseado em aprendizado contrastivo, utilizando um modelo pré-treinado e técnicas de clusterização para adicionar pseudo-rótulos.
[Wang et al., 2022b]	Abordagem utilizando Redes Neurais de Grafo (GNN) e Redes de Ponteiro (PN) para a composição de serviços cientes da Qualidade de Serviço (QoS), utilizando um algoritmo de otimização baseado no comportamento de baleias para ajustar a solução inicial.

ções e protocolos criptografados, métodos tradicionais de detecção tornaram-se ineficazes devido à perda de informações semânticas e à dificuldade na extração de características. [Tian et al., 2022] propõe um método de identificação de tráfego criptografado baseado em aprendizado contrastivo. Este método utiliza um modelo pré-treinado com aprendizado contrastivo supervisionado e expande o conjunto de dados rotulados por meio de técnicas de clusterização para adicionar pseudo-rótulos. O algoritmo é baseado em três componentes principais: i) Seleção de Granularidade: utiliza sessões, que são coleções de fluxos definidos pelas cinco-tupla, IP de origem, IP de destino, porta de origem, porta de destino e protocolos de nível de transporte; ii) Processamento de Dados Não Rotulados: usa Análise de Componentes Principais (PCA) para reduzir a dimensionalidade dos dados de tráfego e técnicas de agrupamento para expandir o conjunto de dados rotulados com pseudo-rótulos; iii) Identificação de Tráfego Criptografado: implementa aprendizado contrastivo supervisionado com uma rede neural ResNet para extração e projeção de características.

Wang *et al.* propõem uma abordagem utilizando Redes Neurais de Grafo (GNN) e Redes de Ponteiro (PN), baseado em aprendizado por reforço, para a composição de serviços cientes da Qualidade de Serviço (QoS) [Wang et al., 2022b]. Primeiramente, os dados de tarefas e serviços são estruturados como grafos, permitindo que a GNN extraia correlações subjacentes e preveja a probabilidade de uso de cada serviço. Com base nesses serviços de alta probabilidade, a rede de ponteiro, frequentemente utilizada para problemas de otimização combinatória, é empregada para construir a solução inicial de serviços. Adicionalmente, para melhorar a capacidade de generalização da rede, uma camada extra é adicionada à PN. Finalmente, um algoritmo de otimização baseado no comportamento de baleias é utilizado para ajustar a solução inicial, incorporando serviços raramente usados.

Detecção de anomalias em dados de séries temporais multivariadas (MTS), com foco especial em sua aplicação em cibersegurança para detecção de ataques desconhecidos. [González et al., 2023] apresentam o DC-VAE, uma abordagem recente que utiliza Variational Auto Encoders (VAEs) e Dilated Convolutional Neural Networks (DCNNs) para modelar dados MTS complexos e de alta dimensão. No entanto, os autores reconhecem que a detecção de anomalias usando VAEs pode resultar em degradação de desempenho e até esquecimento catastrófico quando treinados em medidas de rede dinâmicas e em evolução, especialmente em casos de mudanças no conceito dos dados. Portanto, eles propõem uma extensão do DC-VAE para um cenário de aprendizado contínuo, aproveitando as propriedades da inteligência artificial generativa dos modelos subjacentes para lidar com dados em constante evolução.

2.7. Tendências e desafios de pesquisa

A aplicação de técnicas de aprendizado auto-supervisionado (*Self-Supervised Learning* - SSL) em redes de computadores visa resolver desafios complexos, como a classificação de tráfego, a detecção de intrusões e a otimização do desempenho. Para maximizar a eficácia dessas técnicas, é crucial abordar questões específicas como a rotulagem dos dados, a evolução dos métodos de aprendizado e a adaptação dos modelos à natureza dinâmica dos dados de rede. Além disso, alguns desafios estão relacionados a outros paradigmas de aprendizado de máquina, como a falta de explicabilidade e a ausência de

fundamentação teórica. Esta seção explora diversas abordagens e estratégias emergentes no campo, destacando suas contribuições e desafios em redes de computadores.

Rotulagem dos dados: O desafio crucial na implantação de mecanismos de aprendizado de máquina em aplicações de redes de computadores é a rotulagem dos dados. O tráfego de rede, frequentemente gerado a altas taxas, contrasta drasticamente com a taxa de rotulagem realizada por especialistas humanos. A discrepância se manifesta em uma escala de poucos fluxos de rede por evento de rotulagem frente à velocidade de linha de transmissão de dados em enlaces monitorados. Diante desse desafio, o domínio de aplicações de redes busca soluções para tornar a rotulagem de dados mais eficiente, especialmente considerando o aumento exponencial do volume de dados. O aprendizado auto-supervisionado na área de redes está se desenvolvendo para criar representações mais eficientes e significativas dos dados, permitindo uma compreensão mais profunda dos padrões de tráfego.

Evolução do aprendizado contrastivo: Uma tendência notável nesse campo é o crescimento do paradigma contrastivo. Nesse método, são utilizados pares de dados, em que cada par consiste em uma instância de dados e uma versão modificada ou distorcida dessa instância. O objetivo é ensinar ao modelo a distinguir entre instâncias semelhantes e diferentes. Para fazer isso, durante o treinamento, o modelo é incentivado a aproximar instâncias semelhantes no espaço latente, enquanto afasta instâncias diferentes. Isso é alcançado por meio de técnicas que minimizam a distância entre instâncias similares e maximizam a distância entre instâncias diferentes no espaço de representação.

Essa abordagem contrastiva tem se mostrado muito eficaz na aprendizagem de representações de alta qualidade, especialmente em conjuntos de dados grandes e complexos. Ao aprender a identificar padrões significativos nos dados através da comparação de instâncias, os modelos construídos com base nesse paradigma conseguem capturar nuances sutis nos dados, resultando em representações mais ricas e informativas. Isso, por sua vez, leva a melhorias significativas no desempenho de tarefas relacionadas à análise de tráfego de rede, como classificação de aplicativos, detecção de intrusões e previsão de falhas.

Transferência de conhecimento: A transferência de conhecimento é uma estratégia fundamental no campo do aprendizado de máquina, que visa aproveitar o conhecimento adquirido em um domínio específico para melhorar o desempenho em tarefas relacionadas ou diferentes. Essa abordagem é particularmente útil quando há uma escassez de dados rotulados em um domínio-alvo, mas abundância de dados em um domínio-fonte relacionado. Uma das formas mais comuns de transferência de conhecimento é a técnica de pré-treinamento de modelos em grandes conjuntos de dados genéricos, seguida pelo ajuste fino (*fine-tuning*) em conjuntos de dados específicos da tarefa. Essa abordagem permite que modelos pré-treinados capturem padrões gerais nos dados durante a fase de pré-treinamento e, em seguida, ajustem-se aos padrões específicos do novo domínio durante o ajuste fino. Além disso, a aplicação multidomínio da transferência de conhecimento destaca-se como uma tendência importante. Nesse contexto, modelos pré-treinados são utilizados em uma variedade de domínios, explorando o aprendizado auto-supervisionado em diversas áreas. Isso significa que os modelos podem ser treinados em conjuntos de dados que abrangem diferentes aspectos, como segurança cibernética, otimização de de-

sempenho de redes e detecção de anomalias. Essa abordagem multidomínio permite que os modelos adquiram uma compreensão mais abrangente dos padrões e características dos dados, o que pode levar a um melhor desempenho em uma ampla gama de tarefas e cenários. A transferência de conhecimento também pode ocorrer através do treinamento de modelos auto-supervisionados para aprender representações significativas de dados, que são então utilizados para outras tarefas distintas, como a classificação de tráfego com base nas representações aprendidas.

Coleta, armazenamento de dados pré-processados e adaptação de métodos de aprendizado de máquina à natureza dos dados: O armazenamento de quantidades infinitas de dados é inviável, e a obtenção de dados na natureza muitas vezes implica custos de tempo devido a limitações de largura de banda ou velocidade do sensor. Isso torna o treinamento baseado em épocas impraticável e uma implementação ingênua de abordagens SSL convencionais, utilizando cada amostra apenas uma vez, resultaria em aprendizes ineficientes. Uma solução é utilizar buffers de replay para separar a aquisição de dados do pipeline de treinamento. Uma questão importante é avaliar a eficácia desses mecanismos de replay em permitir que as representações continuem a melhorar enquanto os dados estão sendo coletados.

Aprendizado continuado e não-estacionariedade dos dados: Os dados do mundo real são não-estacionários, apresentando variações temporais significativas. Por exemplo, durante a Copa do Mundo, há um aumento no número de imagens relacionadas ao futebol. Além disso, robôs explorando ambientes internos encontram distribuições semânticas temporalmente agrupadas. Um sistema inteligente de aprendizado contínuo deve ser capaz de assimilar novos conceitos sem esquecer os antigos em meio a essas mudanças. No entanto, abordagens convencionais de aprendizado contrastivo podem se ajustar excessivamente à distribuição atual, levando ao esquecimento de informações anteriores. Portanto, a questão central é como projetar métodos SSL capazes de aprender efetivamente em ambientes não-estacionários.

Necessidade de novas técnicas de *Data Augmentation*: Avanços recentes na aprendizagem de representações visuais são atribuídos a estratégias de *data augmentation*, como redimensionamento, rotação e coloração. No entanto, aplicar essas técnicas diretamente a dados de grafos é desafiador devido à sua natureza não euclidiana. As estratégias de aumento de dados em grafos geralmente envolvem adicionar ou remover nós e arestas. Para melhorar o aprendizado auto-supervisionado em grafos, é importante projetar estratégias de aumento mais eficientes, seguindo diretrizes específicas e garantindo que sejam aplicáveis, adaptáveis, eficientes e dinâmicas.

Falta de explicabilidade: Embora os métodos de SSL em grafos tenham alcançado bons resultados em diversas tarefas, ainda não compreendemos totalmente o que eles aprendem nas tarefas de pretexto auto-supervisionadas. É importante entender se esses métodos capturam padrões de características, estruturas significativas ou relações entre características e estruturas. Além disso, é necessário determinar se esse aprendizado é explícito ou implícito e se é possível encontrar interpretações claras nos dados de entrada. Essas questões são cruciais para entender o comportamento do modelo, mas estão ausentes na maioria dos trabalhos atuais de SSL em redes. Portanto, é necessário explorar a interpretabilidade desses métodos e analisar profundamente o comportamento do mo-

delo para melhorar sua generalização e robustez em tarefas relacionadas à segurança ou privacidade.

Falta de fundamentação teórica: Apesar do sucesso do SSL em grafos em várias tarefas, a maioria dos métodos existentes baseia-se na intuição, carecendo de fundamentação teórica sólida. Isso resulta em limitações de desempenho e explicabilidade. Construir uma base teórica sólida para o SSL em redes de computadores, minimizando a lacuna entre teoria e prática, é uma direção promissora. Por exemplo, é importante investigar se a maximização da informação mútua é o único método para o aprendizado contrastivo em grafos. Além disso, embora tenhamos introduzido objetivos contrastivos alternativos, como margem de triplo e perda de quadruplo, a conexão teórica entre essas abordagens e a informação mútua ainda precisa ser explorada mais profundamente.

Margem de pré-treinamento: A estratégia comum de treinamento em aprendizado auto-supervisionado (SSL) de grafos envolve o pré-treinamento com tarefas auto-supervisionadas e, em seguida, o uso do modelo pré-treinado para tarefas específicas, seja ajustando os pesos ou mantendo-os congelados. No entanto, a transferência do conhecimento pré-treinado para as tarefas secundárias continua sendo um desafio latente. Embora inúmeras estratégias tenham sido propostas para resolver esse problema nos domínios de visão computacional e processamento de linguagem natural, aplicá-las diretamente a grafos e redes é desafiador devido à sua estrutura não euclidiana inerente. Portanto, é essencial projetar técnicas específicas para grafos que minimizem a diferença entre o pré-treinamento e as tarefas secundárias.

2.8. Descrição da prática da aplicação de aprendizado auto-supervisionado

Esse capítulo se complementa de uma prática da utilização de algoritmos de aprendizado auto-supervisionado para a criação de um sistema de detecção de intrusão. Os participantes do curso são convidados a programar algoritmos de aprendizado de máquina auto-supervisionado, focando a análise de tráfego de redes utilizando conjuntos de dados como NSL-KDD[1] e CICIDS 2017[2]. O roteiro[3] se inicia com a configuração do ambiente de desenvolvimento, em que os participantes instalarão e configurarão todas as ferramentas e bibliotecas necessárias. Em seguida, realizarão uma análise exploratória de dados (*Exploratory Data Analysis* - EDA), em que explorarão as características dos conjuntos de dados, identificarão padrões e anomalias, e realizarão a limpeza dos dados. Posteriormente, serão aplicadas técnicas de *feature engineering* para transformar e preparar os dados de modo adequado para alimentar os modelos de aprendizado auto-supervisionado. Na etapa seguinte, o apresentador explicará detalhadamente o processo de construção do modelo auto-supervisionado, começando pelo pré-treinamento contrastivo, em que os participantes aprenderão a criar representações eficazes dos dados sem a necessidade de rótulos. Após essa etapa, será realizado o refinamento (*fine-tuning*), em que os modelos pré-treinados serão ajustados usando um subconjunto de dados rotulados para melhorar a precisão da detecção de intrusões. Após o treinamento, os participantes interpretarão os resultados, analisando métricas de desempenho como acurácia, precisão, revocação e a

[1]Disponível em `https://www.unb.ca/cic/datasets/nsl.html`.

[2]Disponível em `https://www.unb.ca/cic/datasets/ids-2017.html`.

[3]O roteiro da atividade prática pode ser acessado em `https://github.com/joaovitor-valle/Minicurso-SSL---JAI.git`.

curva ROC. Eles explorarão o ciclo completo, desde a preparação dos dados até a análise de resultados, promovendo uma compreensão sólida dos desafios práticos enfrentados ao lidar com dados reais de tráfego de redes e suas representações. Ao final, espera-se que os participantes tenham adquirido habilidades práticas para a aplicação de algoritmos auto-supervisionados em problemas de redes de computadores, especialmente relacionados à representação de dados de tráfego de redes e à aumentação de dados rotulados.

2.9. Considerações finais

A evolução contínua das ameaças cibernéticas exige que os (*Intrusion Detection Systems* - IDS) se adaptem constantemente para enfrentar novas e sofisticadas formas de ataques. A integração de técnicas avançadas, como inteligência artificial, é cada vez mais comum para melhorar a precisão na detecção e reduzir falsos positivos. A evolução dos algoritmos de aprendizado por grafos e o surgimento de modelos mais complexos que envolvem aprendizado auto-supervisionado têm o potencial de transformar a forma como as máquinas aprendem e representam informações. Esse capítulo abordou o uso do aprendizado auto-supervisionado em aplicações dinâmicas de redes de computadores. O uso de técnicas avançadas de aprendizado de máquina, como o aprendizado auto-supervisionado, oferece uma solução promissora ao enfrentar a escassez de dados rotulados. O capítulo apresentou uma revisão detalhada e atualizada do estado da arte da aplicação do aprendizado auto-supervisionado à criação de sistemas de detecção de intrusões, incluindo os principais modelos e arcabouços, bem como as vantagens e desvantagens de cada abordagem. O capítulo disponibiliza ainda uma atividade prática disponível no repositório `https://github.com/joaovitor-valle/Minicurso-SSL---JAI.git`.

O potencial do aprendizado auto-supervisionado para transformar a segurança e a eficiência das redes de computadores é imenso. A contínua exploração dessas técnicas promete melhorias significativas em termos de segurança, além de expandir os horizontes do aprendizado de máquina em cenários complexos e dinâmicos. A pesquisa nessa área é fortemente incentivada, dada a sua capacidade de enfrentar desafios críticos e abrir novas fronteiras tecnológicas. Contudo, é crucial abordar os problemas e superar as limitações existentes para avançar no uso de soluções de aprendizado auto-supervisionado na área de redes de computadores. Algumas das direções futuras incluem a exploração de novas abordagens, como a análise temporal do tráfego de rede, e a aplicação de outras técnicas como transferência de conhecimento para tornar a predição e o treinamento de modelos mais eficientes de forma distribuída.

O capítulo discutiu ainda diversos tópicos fundamentais e tendência de pesquisas para o aprendizado auto-supervisionado, incluindo a rotulagem de dados, a evolução do aprendizado contrastivo, a transferência de conhecimento, a coleta e armazenamento de dados pré-processados, e a adaptação dos métodos de aprendizado de máquina à natureza dos dados de rede. Adicionalmente, abordaram-se os desafios do aprendizado contínuo em cenários de dados não-estacionários, a necessidade de novas técnicas de *data augmentation* e as lacunas na explicabilidade e fundamentação teórica dos modelos. Também foi destacada a importância da margem de pré-treinamento como uma estratégia crítica para o sucesso das aplicações de aprendizado auto-supervisionado em redes e grafos.

Os principais desafios atuais para área de pesquisa incluem a discrepância entre

a alta taxa de geração de tráfego de rede e a taxa de rotulagem manual, a complexidade de adaptar métodos de aprendizado a dados dinâmicos e não-euclidianos, e a necessidade de melhorar a explicabilidade e fundamentação teórica dos modelos. No entanto, essas dificuldades também representam oportunidades significativas de pesquisa. A pesquisa contínua em métodos auto-supervisionados, o desenvolvimento de técnicas de *feature engineering* e a exploração de novos paradigmas de transferência de conhecimento são áreas promissoras que podem levar a avanços substanciais em aplicações dinâmicas de redes com a detecção de intrusões e otimização de redes com amparo de técnicas de aprendizado auto-supervisionado generativo e contrastivo.

Agradecimentos

Este capítulo foi realizado com recursos do CNPq, FAPERJ, CAPES, RNP e INCT ICONIOT.

Referências

[Abusitta et al., 2023] Abusitta, A., de Carvalho, G. H., Wahab, O. A., Halabi, T., Fung, B. C. e Mamoori, S. A. (2023). Deep learning-enabled anomaly detection for iot systems. *Internet of Things*, 21:100656.

[Bachman et al., 2019] Bachman, P., Hjelm, R. D. e Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32.

[Barbosa et al., 2024] Barbosa, G. N. N., Andreoni, M. e Mattos, D. M. F. (2024). Optimizing feature selection in intrusion detection systems: Pareto dominance set approaches with mutual information and linear correlation. *Ad Hoc Networks*, 159:103485.

[Barbosa et al., 2021a] Barbosa, G. N. N., Andreoni Lopez, M., Medeiros, D. S. V. e Mattos, D. M. F. (2021a). An entropy-based hybrid mechanism for large-scale wireless network traffic prediction. Em *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, p. 1–6.

[Barbosa et al., 2021b] Barbosa, G. N. N., Bezerra, G. M. G., de Medeiros, D. S. V., Andreoni Lopez, M. e Mattos, D. M. F. (2021b). Segurança em Redes 5G: Oportunidades e Desafios em Detecção de Anomalias e Predição de Tráfego baseadas em Aprendizado de Máquina. Em *Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, p. 145–189.

[Bardes et al., 2022] Bardes, A., Ponce, J. e LeCun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *ICLR, Vicreg*, 1:2.

[Bochie et al., 2020] Bochie, K., da Silva Gilbert, M., Gantert, L., Barbosa, M. d. S. M., de Medeiros, D. S. V. e Campista, M. E. M. (2020). Aprendizado profundo em redes desafiadoras: Conceitos e aplicações. *Sociedade Brasileira de Computação*.

[Caron et al., 2020] Caron, M., Misra, I., Mairal, J., Goyal, P. e Bojanowski, P. (2020). Unsupervised learning of visual features by contrasting cluster assignments. Em *European Conference on Computer Vision*, p. 3–19.

[Caron et al., 2018] Caron, M., Sun, R. e Schölkopf, B. (2018). Counterfactuals uncover the modular structure of deep generative models. *arXiv preprint arXiv:1812.03253*.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P. e Joulin, A. (2021). Emerging properties in self-supervised vision transformers. Em *Proceedings of the IEEE/CVF international conference on computer vision*, p. 9650–9660.

[Chen et al., 2020a] Chen, T., He, X., Fan, Y., Zhang, Y. e Xie, J. (2020a). Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

[Chen et al., 2020b] Chen, T., Kornblith, S., Norouzi, M. e Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. Em *International conference on machine learning*, p. 1597–1607. PMLR.

[Chen e He, 2021] Chen, X. e He, K. (2021). Exploring simple siamese representation learning. Em *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, p. 15750–15758.

[Choi e Ryu, 2023] Choi, J. e Ryu, D. (2023). Qos-aware graph contrastive learning for web service recommendation. Em *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, p. 171–180. IEEE.

[Creswell et al., 2018] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. e Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65.

[Dai et al., 2019] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V. e Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

[de Oliveira et al., 2021] de Oliveira, N. R., Pisa, P. S., Andreoni Lopez, M., de Medeiros, D. S. V. e Mattos, D. M. F. (2021). Identifying fake news on social networks based on natural language processing: Trends and challenges. *Information*, 12(1).

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. e Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Em *CVPR09*.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K. e Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dib et al., 2022] Dib, M., Torabi, S., Bou-Harb, E., Bouguila, N. e Assi, C. (2022). Evoliot: A self-supervised contrastive learning framework for detecting and characterizing evolving iot malware variants. Em *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, p. 452–466.

[Doersch et al., 2015] Doersch, C., Gupta, A. e Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. Em *Proceedings of the IEEE international conference on computer vision*, p. 1422–1430.

[Ermolov et al., 2021] Ermolov, A., Siarohin, A., Sangineto, E. e Sebe, N. (2021). Whitening for self-supervised representation learning. Em *International conference on machine learning*, p. 3015–3024. PMLR.

[Gao et al., 2022] Gao, Y., Hasegawa, H., Yamaguchi, Y. e Shimada, H. (2022). Unsupervised graph contrastive learning with data augmentation for malware classification. Em *Proc. 16th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2022), IARIA*, p. 41–47.

[Germain et al., 2015] Germain, M., Gregor, K., Murray, I. e Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. *International Conference on Machine Learning*, p. 881–889.

[Gidaris et al., 2018] Gidaris, S., Singh, P. e Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*.

[González et al., 2023] González, G. G., Casas, P. e Fernández, A. (2023). Fake it till you detect it: Continual anomaly detection in multivariate time-series using generative ai. Em *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, p. 558–566. IEEE.

[Goyal et al., 2021] Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A. et al. (2021). Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altche, F., Tallec, C. L. e Richemond, P. H. (2020). Bootstrap your own latent: A new approach to self-supervised learning. Em *Advances in Neural Information Processing Systems*, p. 33–44.

[Gutmann e Hyvärinen, 2010] Gutmann, M. e Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. Em *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, p. 297–304. JMLR Workshop and Conference Proceedings.

[Han et al., 2024] Han, X., Cui, S., Qin, J., Liu, S., Jiang, B., Dong, C., Lu, Z. e Liu, B. (2024). Contramtd: An unsupervised malicious network traffic detection method based on contrastive learning. Em *Proceedings of the ACM on Web Conference 2024*, p. 1680–1689.

[Hassani e Khasahmadi, 2020] Hassani, K. e Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. Em *International conference on machine learning*, p. 4116–4126. PMLR.

[He et al., 2020a] He, K., Fan, H., Wu, Y., Xie, S. e Girshick, R. (2020a). Momentum contrast for unsupervised visual representation learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 9729–9738.

[He et al., 2020b] He, K., Fan, H., Wu, Y., Xie, S. e Girshick, R. (2020b). Momentum contrast for unsupervised visual representation learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 9729–973.

[Jaiswal et al., 2021] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D. e Makedon, F. (2021). A survey on contrastive self-supervised learning. *Technologies*, 9(1).

[Jing e Tian, 2020] Jing, L. e Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058.

[Kim et al., 2018] Kim, D., Cho, D., Yoo, D. e Kweon, I. S. (2018). Learning image representations by completing damaged jigsaw puzzles. Em *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, p. 793–802. IEEE.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I. e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kye et al., 2022] Kye, H., Kim, M. e Kwon, M. (2022). Hierarchical detection of network anomalies: A self-supervised learning approach. *IEEE Signal Processing Letters*, 29:1908–1912.

[Lan et al., 2019] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. e Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

[Le-Khac et al., 2020] Le-Khac, P. H., Healy, G. e Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934.

[Lee et al., 2023] Lee, P., Bubeck, S. e Petro, J. (2023). Benefits, limits, and risks of gpt-4 as an ai chatbot for medicine.

[Li et al., 2021] Li, C., Yang, J., Zhang, P., Gao, M., Xiao, B., Dai, X., Yuan, L. e Gao, J. (2021). Efficient self-supervised vision transformers for representation learning. *arXiv preprint arXiv:2106.09785*.

[Li et al., 2024a] Li, L., Lu, Y., Yang, G. e Yan, X. (2024a). End-to-end network intrusion detection based on contrastive learning. *Sensors*, 24(7).

[Li et al., 2024b] Li, Z., Xia, L., Xu, Y. e Huang, C. (2024b). Gpt-st: Generative pre-training of spatio-temporal graph neural networks. *Advances in Neural Information Processing Systems*, 36.

[Liu et al., 2019a] Liu, H., Li, S., Li, H., Li, X., Gao, J. e Ji, S. (2019a). Graphaf: A flow-based autoregressive model for molecular graph generation. Em *Proceedings of the 36th International Conference on Machine Learning*, p. 3872–3881.

[Liu et al., 2023] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J. e Tang, J. (2023). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876.

[Liu et al., 2019b] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. e Stoyanov, V. (2019b). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Lopez-Martin et al., 2023] Lopez-Martin, M., Sanchez-Esguevillas, A., Arribas, J. I. e Carro, B. (2023). Contrastive learning over random fourier features for iot network intrusion detection. *IEEE Internet of Things Journal*, 10(10):8505–8513.

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S. e y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. Em *Artificial intelligence and statistics*, p. 1273–1282. PMLR.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. e Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[Misra e Maaten, 2020] Misra, I. e Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. Em *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, p. 6707–6717.

[Noroozi e Favaro, 2016] Noroozi, M. e Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. Em *European conference on computer vision*, p. 69–84. Springer.

[Oord et al., 2016a] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. e Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[Oord et al., 2016b] Oord, A. v. d., Kalchbrenner, N. e Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

[Oord et al., 2018] Oord, A. v. d., Li, Y. e Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[Peng et al., 2021] Peng, J., Liu, D., Xu, S. e Li, H. (2021). Generating diverse structure for image inpainting with hierarchical vq-vae. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10775–10784.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R. e Skiena, S. (2014). Deepwalk: Online learning of social representations. Em *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 701–710. ACM.

[Popova et al., 2019] Popova, M., Shvets, M., Oliva, J. e Isayev, O. (2019). Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*.

[Ragab et al., 2022] Ragab, M., Eldele, E., Chen, Z., Wu, M., Kwoh, C.-K. e Li, X. (2022). Self-supervised autoregressive domain adaptation for time series data. *IEEE Transactions on Neural Networks and Learning Systems*.

[Razavi et al., 2019] Razavi, A., Van den Oord, A. e Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

[Stafford, 2020] Stafford, V. (2020). Zero trust architecture. *NIST special publication*, 800:207.

[Tang et al., 2015] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. e Mei, Q. (2015). Line: Large-scale information network embedding. Em *Proceedings of the 24th international conference on world wide web*, p. 1067–1077.

[Thisanke et al., 2023] Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R. e Herath, D. (2023). Semantic segmentation using vision transformers: A survey. *Engineering Applications of Artificial Intelligence*, 126:106669.

[Tian et al., 2022] Tian, S., Gao, Y., Yuan, G., Zhang, R., Zhao, J. e Zhang, S. (2022). An encrypted traffic classification method based on contrastive learning. Em *Proceedings of the 8th International Conference on Communication and Information Processing*, p. 101–105.

[Tian et al., 2020] Tian, Y., Krishnan, D. e Isola, P. (2020). Contrastive multiview coding. Em *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, p. 776–794. Springer.

[Torbarina et al., 2024] Torbarina, L., Ferkovic, T., Roguski, L., Mihelcic, V., Sarlija, B. e Kraljevic, Z. (2024). Challenges and opportunities of using transformer-based multi-task learning in nlp through ml lifecycle: A position paper. *Natural Language Processing Journal*, 7:100076.

[Van den Oord et al., 2016] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A. et al. (2016). Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. e Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang et al., 2022a] Wang, N., Chen, Y., Hu, Y., Lou, W. e Hou, Y. T. (2022a). Feco: Boosting intrusion detection capability in iot networks via contrastive learning. Em *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, p. 1409–1418. IEEE.

[Wang et al., 2022b] Wang, X., Xu, H., Wang, X., Xu, X. e Wang, Z. (2022b). A graph neural network and pointer network-based approach for qos-aware service composition. *IEEE Transactions on Services Computing*.

[Wei et al., 2019] Wei, C., Xie, L., Ren, X., Xia, Y., Su, C., Liu, J., Tian, Q. e Yuille, A. L. (2019). Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 1910–1919.

[Wen et al., 2020] Wen, L., Li, X. e Gao, L. (2020). A transfer convolutional neural network for fault diagnosis based on resnet-50. *Neural Computing and Applications*, 32(10):6111–6124.

[White et al., 2019] White, G., Palade, A., Cabrera, C. e Clarke, S. (2019). Autoencoders for qos prediction at the edge. Em *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, p. 1–9.

[Wu et al., 2023] Wu, L., Lin, H., Tan, C., Gao, Z. e Li, S. Z. (2023). Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4216–4235.

[Yang et al., 2022] Yang, S., Wang, Y., Xu, H., Xu, F. e Chen, M. (2022). An android malware detection and classification approach based on contrastive lerning. *Computers & Security*, 123:102915.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. e Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

[Yin et al., 2023] Yin, Y., Di, Q., Wan, J. e Liang, T. (2023). Time-aware smart city services based on qos prediction: A contrastive learning approach. *IEEE Internet of Things Journal*.

[You et al., 2018a] You, J., Liu, B., Ying, Z., Pande, V. e Leskovec, J. (2018a). Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31.

[You et al., 2018b] You, J., Ying, R., Ren, X., Hamilton, W. e Leskovec, J. (2018b). Graphrnn: Generating realistic graphs with deep auto-regressive models. Em *International Conference on Machine Learning*, p. 5708–5717.

[You et al., 2017] You, Y., Gitman, I. e Ginsburg, B. (2017). Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

[Yue et al., 2022] Yue, Y., Chen, X., Han, Z., Zeng, X. e Zhu, Y. (2022). Contrastive learning enhanced intrusion detection. *IEEE Transactions on Network and Service Management*, 19(4):4232–4247.

[Zafar et al., 2022] Zafar, S., Lv, Z., Zaydi, N. H., Ibrar, M. e Hu, X. (2022). Dsmlb: Dynamic switch-migration based load balancing for software-defined iot network. *Computer Networks*, 214:109145.

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y. e Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. Em *International conference on machine learning*, p. 12310–12320. PMLR.

[Zhang et al., 2024] Zhang, X., Zhao, R., Jiang, Z., Sun, Z., Ding, Y., Ngai, E. C. e Yang, S.-H. (2024). Aoc-ids: Autonomous online framework with contrastive learning for intrusion detection. *arXiv preprint arXiv:2402.01807*.

[Zhang e Zhu, 2023] Zhang, X. e Zhu, Q. (2023). Ai-enabled network-functions virtualization and software-defined architectures for customized statistical qos over 6g massive mimo mobile wireless networks. *IEEE Network*, 37(2):30–37.

[Zhao et al., 2023] Zhao, W., Xu, G., Cui, Z., Luo, S., Long, C. e Zhang, T. (2023). Deep graph structural infomax. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, p. 4920–4928.

[Zheng et al., 2021] Zheng, Y., Jin, M., Liu, Y., Chi, L., Phan, K. T. e Chen, Y.-P. P. (2021). Generative and contrastive self-supervised learning for graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*.

[Zhu et al., 2023] Zhu, J., Li, B., Wang, J., Li, D., Liu, Y. e Zhang, Z. (2023). Bgcl: Bi-subgraph network based on graph contrastive learning for cold-start qos prediction. *Knowledge-Based Systems*, 263:110296.

[Zhuang et al., 2019] Zhuang, C., Zhai, A. L. e Yamins, D. (2019). Local aggregation for unsupervised learning of visual embeddings. Em *Proceedings of the IEEE/CVF international conference on computer vision*, p. 6002–6012.

[Zoph et al., 2020] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J. e Le, Q. V. (2020). Rethinking pre-training and self-training. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10286–10295.

Capítulo 3

Projeto e ajuste de desempenho de bancos de dados NoSQL

Arlino Magalhães, Francisco Imperes, Manoel Melo

Abstract

Many contemporary applications need to store and process large amounts of data, often in real-time, to make information retrieval feasible. However, traditional storage systems have yet to keep pace with this demand. An alternative to this problem has been using NoSQL databases due to their high availability, scalability, and flexibility for managing large amounts of data. This mini-course aims to present the main concepts of NoSQL technology. Additionally, it shows how to evaluate whether this technology is appropriate for the database design of a particular system. Furthermore, the mini-course also discusses some strategies for tuning the performance of NoSQL databases.

Resumo

Muitas sistemas contemporâneas tem tido a necessidade de armazenar e processar grandes quantidades de dados, muitas vezes em tempo real, a fim de tornar a recuperação da informação viável. Entretanto, os sistemas de armazenamento tradicionais não tem acompanhado essa demanda. Uma alternativa para esse problema tem sido a utilização de bancos de dados NoSQL devido às suas características como alta disponibilidade, escalabilidade e flexibilidade para gerenciar grandes quantidades de dados. Esse minicurso visa apresentar os principais conceitos da tecnologia NoSQL. Além disso, ele mostra como avaliar se essa tecnologia é apropriada para o projeto de banco de dados de um determinado sistema. Adicionalmente, o minicurso também discute algumas estratégias para o ajuste de desempenho de bancos de dados NoSQL.

3.1. Introdução

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) relacionais têm sido a principal opção de armazenamento de dados em sistemas de informação, tanto

grandes como pequenos. Contudo, muitas sistemas contemporâneos têm necessitado de um fluxo massivo de dados tão grande que os sistemas relacionais não conseguem gerenciar. Os SGBDs NoSQL têm se mostrado como uma alternativa para sistemas que gerenciam grande volume de dados e que necessitam de desempenho escalável, alta disponibilidade e resiliência [Davoudian et al. 2018, Magalhães et al. 2021].

NoSQL é uma abreviação do termo em inglês *"não apenas SQL"*. Contudo, esse termo não define esses SGBDs de forma clara. O termo NoSQL refere-se a alguns bancos de dados não relacionais que possuem algumas características, como: dados sem esquema, execução em *clusters* e tolerância à inconsistência. Alguns exemplos desses sistemas são o Redis [Redis Database 2024], MongoDB [MongoDB Database 2024], Cassandra [Cassandra Database 2024] e Neo4J [Neo4J Database 2024].

É importante ressaltar que os SGBDs relacionais são uma ferramenta poderosa e uma opção confiável que ainda continuará sendo utilizada no desenvolvimento de sistemas por muito tempo, talvez décadas. Entretanto, NoSQL tem se mostrado como uma solução mais eficiente para o gerenciamento de grandes quantidades de dados. Nesse sentido, muitos sistemas contemporâneos têm se utilizado de uma persistência poligloto que faz o uso de várias tecnologias de gerenciamento de dados. Consequentemente, os arquitetos devem se familiarizar com essas tecnologias para avaliar quais delas podem ser utilizadas para diferentes necessidades [Davoudian et al. 2018, Magalhães et al. 2023].

Cada projeto tem suas peculiaridades e não há um guia preciso de como escolher o armazenamento de dados ótimo. Embora a tecnologia NoSQL exista há bastante tempo, o seu uso adequado em projetos de bancos de dados não é amplamente conhecido. Além disso, a maioria dos livros e cursos de bancos de dados foca no projeto relacional, que é significativamente diferente do projeto NoSQL. Enquanto o projeto relacional se baseie em tabelas, relacionamentos e normalização, o projeto NoSQL leva em consideração agregados, distribuição dos dados e até desnormalização dos dados. Além disso, em muitos casos, sistemas NoSQL permitem uma consistência mais relaxada, ou seja, dados inconsistentes podem ser tolerados em algum nível [Davoudian et al. 2018, Magalhães et al. 2018].

Esse minicurso discute ferramentas relacionados a tecnologia NoSQL, como produtos, linguagens de acesso, manipulação e processamento dos dados. Porém, o principal foco do curso é fornecer uma base teórica e prática suficiente para julgar se a tecnologia NoSQL é adequada a um projeto. O minicurso utiliza uma amostra representativa dos SGBDs NoSQL, um representante de cada uma das quatro categorias existentes: chave-valor, documentos, colunar e grafos. Embora sejam utilizados exemplos específicos, a maioria da discussão pode ser aplicada e reproduzida facilmente em outros SGBDs. Além disso, todos os códigos das atividades práticas abordadas no curso e uma imagem no VirtualBox do sistema operacional Linux com todos os *softwares* necessário instalados para o curso estão disponíveis para *download*[1].

[1] https://github.com/ArlinoMagalhaes/curso-nosql

Após a implantação de um sistema, o gerenciamento de seu desempenho é necessário a fim de fornecer a continuidade do serviço e, se possível, aprimorar o seu desempenho [Mohammad 2016]. Assim, esse minicurso também provê noções básicas de ajuste de desempenho de bancos de dados NoSQL, tais como: análise de fatores que influenciam o desempenho, análise de tempo de resposta, avaliação de operações e ajuste de consultas.

O restante minicurso está organizado como segue. A Seção 3.2 apresenta brevemente uma revisão de bancos de dados, focando na diferença entre SGBDs relacionais e NoSQL. A Seção 3.3 cobre aspectos relacionados a tecnologia NoSQL, incluindo modelos de dados chave-valor, orientados a documentos, colunar e orientados a grafos. A Seção 3.4 descreve como projetar um sistema de banco de dados NoSQL. O ajuste de desempenho em bancos de dados NoSQL é tratado na Seção 3.5. Por fim, as conclusões são apresentadas na Seção 3.6.

3.2. Revisão de bancos de dados

Desde o surgimento dos primeiros computadores, houve a necessidade de armazenar e manipular dados. As primeiras estruturas de bancos de dados surgiram na década de 60, tais como o banco de dados hierárquico e o em rede. Porém, esses primeiros sistemas eram caros e difíceis de utilizar. Assim, houve bastante investimento na área de pesquisa de banco de dados. Na década de 80, o barateamento de *hardware/-software* fez o SGBD (Sistema de Gerenciamento de Banco de Dados) relacional se tornar a opção padrão para bancos de dados. A popularidade do banco de dados relacional se deve também a sua facilidade em projetar, armazenar e recuperar dados. Atualmente o SGBD relacional é um dos principais componentes nos sistemas de informação [Ramakrishnan and Gehrke 2003, Elmasri and Navathe 2000, Magalhães et al. 2021].

Essa seção faz uma breve revisão dos principais conceitos dos SGBDs relacionais, focando nas características que fizeram esses sistemas se tornarem a escolha padrão para armazenamento de grandes volumes de dados, especialmente no mundo de aplicativos corporativos. Além disso, essa seção discute as limitações desses sistemas que levaram a procura por alternativas de armazenamento de dados, como os bancos de dados NoSQL. Leitores conhecedores desses conceitos podem pular essa seção.

3.2.1. Bancos de dados relacionais

Os SGBDs relacionais oferecem aos usuários/sistemas processos de validação, verificação, segurança e garantias de integridade dos dados. Para isso, esses sistemas são projetados em componentes, como armazenamento, controle de concorrência, otimização de consultas, indexação e recuperação após falhas. Esses componentes facilitam a construção de sistemas, visto que os desenvolvedores não precisam se preocupar com o processo de armazenamento de dados, possibilitando que eles possam focar exclusivamente no sistema [Ramakrishnan and Gehrke 2003, Elmasri and Navathe 2000].

Os SGBDs relacionais utilizam o modelo relacional. Esse modelo organiza

os dados em estruturas de relações (tabelas) e tuplas (linhas). Uma tabela é um conjunto de linhas e uma linha é um conjunto de valores. Cada linha (registro) representa um objeto da entidade representada pela tabela. Cada registro possui um ID (chave) exclusivo que o diferencia entre todos os outros registros da entidade. A tabela possui colunas que são os atributos dos valores de cada linha. A Figura 3.1 ilustra uma tabela que representa uma entidade de empregados de uma empresa [Heuser 2009].

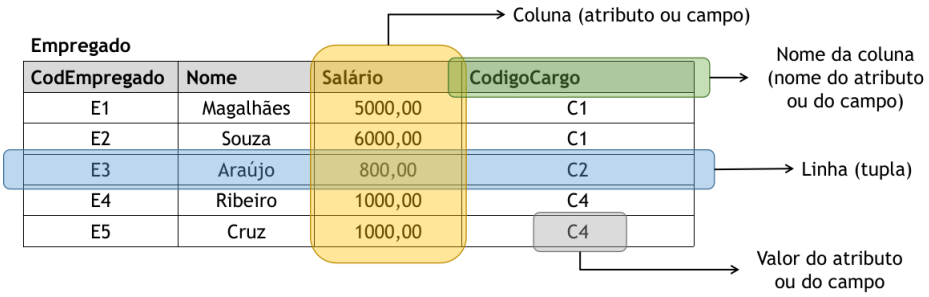


Figura 3.1. Tabela de banco de dados relacional [Heuser 2009].

O modelo relacional utiliza comumente o processo de Normalização cujo objetivo é, principalmente, organizar o projeto de banco de dados para reduzir a redundância de dados e aumentar a integridade de dados e o desempenho do sistema. Para isso, a normalização aplica algumas regras sobre as tabelas do banco de dados. Uma abordagem básica da normalização consiste na separação dos dados referentes a elementos distintos em tabelas distintas associadas através da utilização das chaves [Heuser 2009].

O modelo relacional utiliza uma linguagem padrão para definição, manipulação e consulta de dados, a SQL (*Structured Query Language*). SQL é uma linguagem declarativa, ou seja, ela descreve o que uma instrução deve fazer e não a maneira como a instrução deve ser executada. O SGBD fica responsável por escolher a estratégia mais eficiente para execução das instruções SQL. Assim, o desenvolvedor fica livre do trabalho complexo de acesso ao armazenamento, podendo se concentrar mais nas aplicações. A SQL é inspirada em álgebra relacional que é uma linguagem de consulta formal que possui uma coleção de operações de alto nível sobre relações ou conjuntos. Sua simplicidade e alto poder de expressão fizeram de SQL a linguagem de consulta mais utilizada em bancos de dados [Imielinski and Jr. 1984].

Os SGBDs relacionais utilizam o modelo ACID para garantir consistência dos dados durante o processamento das transações. Uma transação é um conjunto de operações submetidas ao banco de dados, por exemplo, inserir, modificar e excluir linhas de tabelas. ACID é um acrônimo derivado da primeira letra das suas quatro principais propriedades: atomicidade, consistência, isolamento e durabilidade. A atomicidade faz uma transação indivisível, ou seja, todas as operações de uma transação devem ser executadas apropriadamente ou nenhuma delas. A consistência exige que a execução isolada de uma transação não viole a consistência do sistema. O isolamento permite que cada transação não perceba a execução concorrente de outras transações. Por fim, a durabilidade exige que transações finalizadas tenham

seus dados persistidos no sistema [Härder and Reuter 1983, Magalhães et al. 2021].

Os SGBDs comumente utilizam algum método de controle de concorrência para garantir a consistência dos dados em face a múltiplos acessos aos dados. O controle de concorrência em um banco de dados assegura que transações distintas acessem os dados concorrentemente sem levar o sistema a uma inconsistência. O gerenciamento do controle de concorrência é um tópico de desempenho estudado desde as primeiras pesquisas em bancos de dados relacionais (por exemplo, [Gray et al. 1975], [Bernstein et al. 1987], [Gray and Reuter 1993], [Bernstein and Goodman 1981]).

A maioria dos bancos de dados relacionais utiliza algum protocolo de controle de concorrência baseado em bloqueio, como o Bloqueio em Duas Fazes (*Two-Phase Locking* - 2PL). Nesse protocolo, por exemplo, uma transação T_1 deve adquirir bloqueios nos dados que necessita antes de começar a modificá-los. Uma transação T_2 que necessite alterar dados bloqueados por T_1 deve esperar até que T_1 não precise mais desses dados e desbloqueie-os. Esse tipo de consistência, que assegura que diferentes tipos de dados façam sentido juntos, é chamado de consistência lógica [Magalhães et al. 2021, Magalhães et al. 2023].

3.2.2. Por que utilizar bancos de dados NoSQL?

3.2.2.1. Escalabilidade

Nas últimas décadas, houve um crescimento exponencial na quantidade de dados a serem armazenados. Assim, muitos sistemas tem tido a necessidade de armazenar e processar uma quantidade massiva de dados, muitas vezes em tempo real, a fim de tornar a recuperação da informação viável. Entretanto, os sistemas de armazenamento tradicionais não têm acompanhado essa demanda. Esse cenário foi inicialmente percebido por empresas de *internet*, como Amazon, Google, Facebook e Twitter. Contudo, atualmente outras empresas/organizações têm encontrado esse mesmo obstáculo para fornecer serviços com tempos de resposta aceitáveis [Magalhães et al. 2018].

Juntamente com o aumento da quantidade de dados, também houve o aumento de usuários. Lidar com aumento de dados e tráfego exige a adoção de mais recursos computacionais. Para suportar esse crescimento, há duas opções de escalabilidade: vertical e horizontal. Escalonar verticalmente (*scale up/down*) significa adicionar recursos ao servidor, como processador, memória e disco. Porém, máquinas mais robustas tornam-se cada vez mais caras. Além disso, há limites físicos quanto ao aumento de recursos em uma única máquina. Escalonar horizontalmente (*scale out/in*) significa utilizar um ou mais *clusters*. Um *cluster* é um conjunto de servidores (nós ou nodos) interconectados, que atuam como se fossem um único sistema trabalhando juntos. Um *cluster* pode utilizar máquinas de *hardware* mais acessível e barato. Essa estratégia também pode ser mais tolerante a falhas, visto que, embora nodos possam falhar, o *cluster* pode continuar funcionando na totalidade [Hill 1990, Bondi 2000, Michael et al. 2007]. A Seção 3.4.4 detalha bancos de dados distribuídos.

Os SGBDs relacionais foram concebidos para funcionar em um servidor centralizado. Assim, embora existam bancos de dados relacionais distribuídos (por exemplo: MySQL NDB Cluster, Oracle RAC e Microsoft SQL Server), a escalabilidade horizontal ainda pode ser um desafio. Existem vários problemas a serem resolvidos na implementação de um banco de dados distribuído. Manter a consistência dos dados é um dos desafios mais importantes e complexos. Por exemplo, uma única transação pode manipular dados em máquinas diferentes e, por esse motivo, sincronizar atualizações torna-se um processo complexo. Geralmente, esses sistemas utilizam um sistema de arquivos para gerenciar os dados no *cluster*, o que representa um ponto único sujeito a falhas [Sadalage and Fowler 2019, Davoudian et al. 2018].

Os SGBDs NoSQL têm sido uma alternativa para cenários em que os sistemas tradicionais não têm se mostrado tão eficientes. Esses SGBDs possuem arquitetura diferenciada e seguem conceitos diferentes para facilitar o tratamento com alta escalabilidade de dados. Por exemplo, alguns SGBDs NoSQL comumente utilizam uma consistência relaxada (ver Seção 3.4.5 para mais detalhes). Eles também podem utilizar o conceito de agregado para facilitar a distribuição de seus dados em *clusters*. A Seção 3.4.1 explica agregados em detalhes e a Seção 3.4.4 discute a distribuição de dados [Sadalage and Fowler 2019, Davoudian et al. 2018].

É importante ressaltar que os SGBDs relacionais ainda são uma opção confiável e bem consolidada para desenvolvimento de muitos sistemas contemporâneos. Eles são muito utilizados em vários setores, como finanças, saúde, varejo e muitos outros, devido à sua capacidade de garantir a integridade dos dados e fornecer recursos flexíveis a consultas nos dados. Portanto, ao considerar a implementação de um banco de dados, é essencial avaliar as necessidades específicas do seu projeto e considerar as vantagens e desvantagens dos bancos relacionais em relação a outras opções, como bancos de dados NoSQL [Sadalage and Fowler 2019, Davoudian et al. 2018]. A Seção 3.4 discute tópicos que podem ajudar na escolha do SGBD a ser implementado no projeto de banco de dados.

3.2.2.2. Incompatibilidade de impedância

As operações em um banco de dados relacional consomem e retornam relações. Essas operações são simples e práticas, mas também possuem limitações. Os valores de uma tupla devem ser simples. Contudo, as estruturas de dados em memória utilizadas nas linguagens de programação podem ser muito mais complexas. Elas podem conter estruturas como registros ou listas, muitas vezes aninhadas. Como consequência, antes de armazenar um dado, é necessário traduzi-lo para o modelo relacional. Esse processo traz um trabalho adicional para os desenvolvedores. Essa diferença de representação do dado entre as estruturas em memória e o modelo relacional é chamada de incompatibilidade de impedância [Sadalage and Fowler 2019, Davoudian et al. 2018].

A Figura 3.2 ilustra a incompatibilidade de impedância de dados entre uma estrutura de dados em memória (a) e tabelas de um banco de dados relacional (b). Os dados representam informações de um pedido de compras de um cliente cujos

objetos estão aninhados. Antes de salvar esses objetos no banco de dados, eles devem ser desaninhados e distribuídos em tabelas diferentes [Sadalage and Fowler 2019].

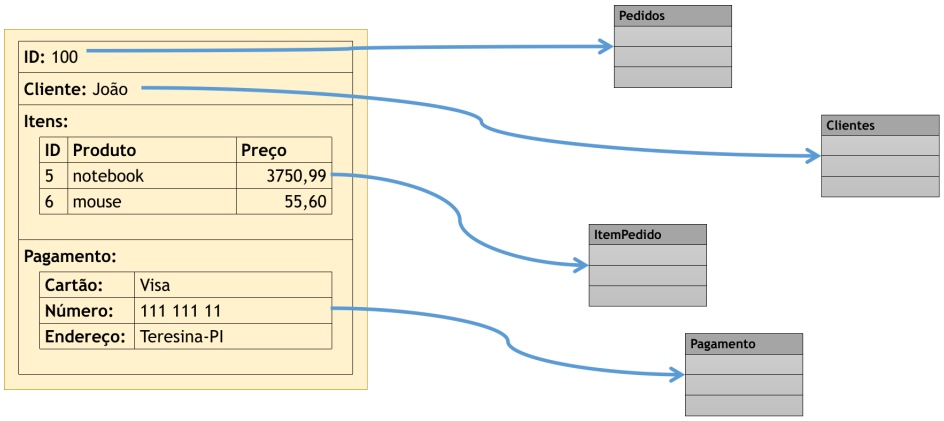


Figura 3.2. Incompatibilidade de impedância entre uma estrutura de dados em memória (a) e tabelas de banco de dados (b) [Sadalage and Fowler 2019].

Alguns *frameworks* de Mapeamento Objeto-Relacional (ORM), como Hibernate, JPA, Sequelize e Django, são amplamente utilizados para lidar com a incompatibilidade de impedância. Esses *frameworks* implementam padrões para tornar essa incompatibilidade mais transparente, poupando muito trabalho dos desenvolvedores. Porém, esses sistemas podem comprometer o desempenho do SGBD em alguns casos. O ORM adiciona uma camada extra de abstração e processamento que pode ser mais lenta do que usar instruções SQL diretas ao SGBD em sistemas que lidam com grandes volumes de dados ou que precisam de tempos de resposta muito rápidos. Além disso, o ORM pode não conseguir mapear tão bem objetos em tabelas do banco de dados, o que pode resultar em inconsistências ou comportamentos inesperados. Adicionalmente, o ORM pode não permitir que os desenvolvedores otimizem consultas para um melhor desempenho. O ORM ainda pode adicionar complexidade ao sistema. Por exemplo, Hibernate possui mais de mil opções de configuração [Ambler 2000].

Os bancos de dados orientados a objetos (BDOOs) surgiram na década de 80 como uma promessa de solução para a incompatibilidade de impedância. Esse SGBD armazenam as informações na forma de objetos, ou seja, utiliza a estrutura de dados denominada orientação a objetos, a qual permeia as linguagens mais modernas. Contudo, os BDOOs não conseguiram se consolidar no mercado devido a algumas desvantagens, como falta de padronização, dificuldade em expressar consultas mais complexas, os produtos existentes não estão tão maduros como os SGBDs relacionais, dentre outros [Boscarioli et al. 2006].

3.3. Bancos de dados NoSQL

Google e Amazon estiveram à frente no desenvolvimento de sistemas de bancos de dados alternativos para gerenciar quantidades massivas de dados em *clusters*.

Na primeira década de 2000, elas apresentaram os bancos de dados Big Table e Dynamo, respectivamente. Os bancos de dados NoSQL podem ser vistos como alternativa para trabalhar com esses casos, por possuírem arquitetura diferenciada e seguirem conceitos diferentes, para facilitar o tratamento com alta escalabilidade de dados. Grandes empresas, por exemplo, Google, oFacebook e Twitter, adotaram a utilização de bancos de dados NoSQL, mais especificamente de modelo colunar, devido à grande demanda por consultas, vinculado ao elevado número de informações que manipulam [Paniz 2016, Vieira et al. 2012].

Modelos de dados referem-se às formas como os dados são organizados e estruturados em um banco de dados NoSQL. Ao contrário dos bancos de dados relacionais, que seguem um modelo de dados tabular com linhas e colunas, os bancos de dados NoSQL permitem uma variedade de modelos de dados para armazenar informações [Marquesone 2016, Vieira et al. 2012].

Cada modelo de dados tem suas particularidades, vantagens e deve ser projetado conforme os tipos de aplicativos, problema e casos de uso específicos. A escolha do modelo de dados adequado depende dos requisitos específicos do aplicativo, incluindo escalabilidade, flexibilidade e complexidade das consultas. Nessa seção são discutidos os principais tipos de bancos de dados NoSQL: chave-valor, orientado a documentos, colunar e orientado a grafos. O principal objetivo dessa seção é tratar das ferramentas relacionados a tecnologia NoSQL, como produtos, linguagens de acesso, manipulação e processamento dos dados [Marquesone 2016, Vieira et al. 2012].

3.3.1. Modelo chave-valor

O modelo chave-valor armazena dados em pares de chave e valor simples. Exemplos de bancos de dados chave-valor incluem Redis, Amazon DynamoDB, Microsoft Azure Cosmos DB, Memcached, etcd, Hazelcast, Aerospik, Ehcache, Riak KV e InterSystems IRIS. De acordo com [Silva et al. 2021], essa estrutura de armazenamento se baseia em uma função *hash* (Figura 3.3), que contém uma chave única e um apontador para um item. A organização baseada em *hashing* fornece um acesso muito rápido às informações.

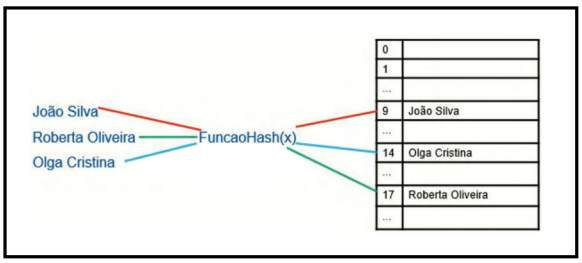


Figura 3.3. Representação da tabela *Hash*. [Silva et al. 2021].

Redis possui vários comandos simples para manipular valores em chaves. Para executá-los, algum programa de *interface* pode ser utilizado, como o Redis-Cli (*Command Line Interface*). O Redis-Cli é uma *interface* de linha de comandos

instalada juntamente com o Redis. Para utilizá-lo, basta apenas digitar o nome do programa (redis-cli) em algum terminal de comandos do sistema operacional [Redis Documentation 2024]. Exite também a opção de utilizar uma *interface* gráfica através do programa *Redis Insight* [Redis Insight 2024]. Seguem alguns dos comandos mais comuns e exemplos de uso deles.

- **SET**: O comando SET é usado para definir um valor para uma chave. Caso a chave já exista, o seu valor é atualizado. O comando SET também permite definir opções, como o tempo de expiração da chave [Redis Documentation 2024].

```
1       SET minha_chave "valor da chave"
2
```

Listagem 3.1. Comando SET.

- **APPEND**: O comando APPEND é usado para anexar um valor a uma chave existente. Se a chave não existir, o comando criará uma nova chave com o valor especificado [Redis Documentation 2024].

```
1       APPEND minha_chave " valor adicional"
2
```

Listagem 3.2. Comando APPEND.

- **GET**: O comando GET retorna o valor associado uma chave [Redis Documentation 2024].

```
1       GET minha_chave
2
```

Listagem 3.3. Comando GET.

- **HSET**: O comando HSET define campos especificados com seus respectivos valores em uma *hash* armazenada em uma chave [Redis Documentation 2024].

```
1       HSET minha_hash campo1 "valor1" campo2 "valor2"
2
```

Listagem 3.4. Comando HGET.

- **HGET**: O comando HGET retorna o valor de um campo associado a uma *hash* armazenada em uma chave [Redis Documentation 2024].

```
1       HGET minha_hash campo1
2
```

Listagem 3.5. Comando HSET.

- **INCR**: O comando INCR incrementa o valor associado a uma chave em 1 [Redis Documentation 2024].

```
1       INCR meu_numero
2
```

Listagem 3.6. Comando INCR.

- **DECR**: O comando DECR decrementa o valor associado a uma chave em 1 [Redis Documentation 2024].

```
DECR meu_numero

```

Listagem 3.7. Comando DECR.

- **DEL**: O comando DEL exclui uma ou mais chaves especificadas [Redis Documentation 2024].

```
DEL meu_numero

```

Listagem 3.8. Comando DEL.

Redis armazena valores simples. Caso seja necessário manipular valores complexos, eles devem ser gerenciados pela aplicação. A Listagem 3.9 exemplifica o armazenamento de uma estrutura de dados complexa ilustrada na Figura 3.4. Essa figura representa um exemplo de um sistema de carrinho de compras *online* em que são armazenadas informações sobre itens de produtos selecionados por um cliente [Paniz 2016, Lazoti 2016, Redis Documentation 2024].

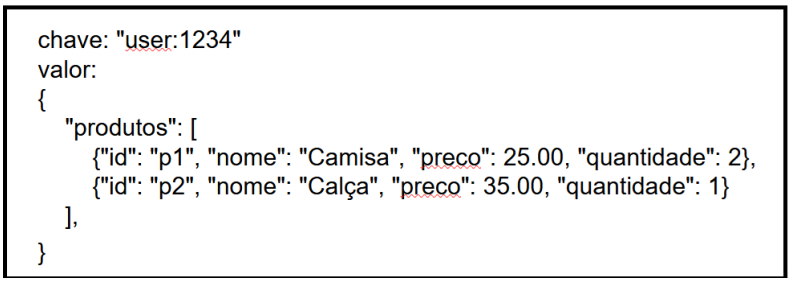


Figura 3.4. Ilustração do modelo Chave-valor para exemplo de carrinho de compras *online*.

```
# Adiciona os produtos ao carrinho do utilizador com ID 1234
HSET user:1234 produto:p1 '{"id": "p1",
                            "nome": "Camisa",
                            "preço": "25.00",
                            "quantidade": "2"}'
HSET user:1234 produto:p2 '{"id": "p2",
                            "nome": "Calça",
                            "preço": "35.00",
                            "quantidade": "1"}'
# Retorna os produtos do carrinho do utilizador com ID 1234
HGETALL user:1234

```

Listagem 3.9. Comandos de exemplo em Redis CLI.

O código da Figura 3.4 utiliza uma *hash* para armazenar os itens de produtos no carrinho de um cliente. A chave da *hash* (user:1234) possui um identificador único do usuário, por exemplo, o ID do usuário. Um item de produto é um campo da *hash*

que possui um identificador do produto, como o ID do produto. No exemplo, a *hash* possui os campos *produto:p1* e *produto:p2*. O valores dos campos são estruturas de dados complexas, em um formato semelhante a JSON, serializados em *strings*. Para a informação contida nos valores fazerem sentido, a aplicação deve gerenciá-los. Algumas linguagens de programação possuem bibliotecas para serializar JSONs [Paniz 2016, Lazoti 2016, Redis Documentation 2024].

3.3.2. Modelo orientado a documentos

Modelos de dados orientados a documentos trabalham com dados não relacionais e os armazena como documentos estruturados, geralmente nos formatos *XML* (*eXtensible Markup Language*; ou Linguagem de Marcação Extensível, em português) ou *JSON* (*JavaScript Object Notation*; ou Notação de Objetos JavaScript, em português) [Harrison 2015].

Os dados são estruturados de forma encadeada, podendo conter atributos das coleções, *tags* e metadados, e seguem uma hierarquia de informações. Além disso, é permitido que tenham redundância e inconsistência, conforme o ambiente NoSQL em que estão inseridos [Silva et al. 2021]. Exemplos de bancos orientados a documentos são: MongoDB, CouchDB, BigCouch, RavenDB, Clusterpoint Server, ThruDB, TerraStore, RaptorDB, JasDB, SisoDB, SDB, SchemaFreeDB e djondb.

Este trabalho dará ênfase ao MongoDB, um projeto *open source* com distribuição gratuita para Linux, Mac e Windows. Embora seja escrito em C++, seu ambiente iterativo e suas buscas são escritas em *JavaScript* [MongoDB Documentation 2024]. MongoDB possui uma *interface* de linha de comandos chamada de Mongo. Para utiliza-lo, basta apenas digitar o nome do programa (mongo) em algum terminal de comandos do sistema operacional. Exite também a opção de *interface* gráfica através do programa Studio 3T [Studio 3T 2024].

Os dados de um documento em uma coleção (equivalente à linha de uma tabela) são armazenados no formato BSON (*Binary* JSON). O BSON é um formato de serialização de documentos em binário, projetado para armazenar e transmitir dados de forma eficiente e estruturada. BSON é um superconjunto de JSON com mais alguns tipos de dados, principalmente a matriz de *bytes* binários. Um BSON pode ser definido de forma semelhante a um JSON entre os caracteres de chaves "{}". Dentro das chaves, são definidos grupos de chaves e seus respectivos valores. Os valores também podem ser listas de chave-valor. A Listagem 3.10 ilustra um registro no formato BSON [MongoDB Documentation 2024, Paniz 2016].

```
1    {
2      firstname: 'Arlino'
3      lastename: 'Magalhães'
4      email: ['home': 'arlinoh@gmail.com',
5              'work': 'arlino@ufpi.edu.br'],
6      phone: ['home': '+810001000'],
7      adress: []
8    }
9
```

Listagem 3.10. Um documento no formato BSON de uma base de dados em MongoDB.

A Listagem 3.11 mostra como inserir documentos utilizando o comando *insertOne* do MongoDB. Esse documento representa um álbum de músicas inserido na coleção *albuns* cujos campos são *nome_album* e *duracao*. Adicionalmente, podem ser inseridos outros álbuns com diferentes informações, tais como: data de lançamento, estúdio de gravação, nome do artista que produziu a capa, número de *singles*, quantidade de semanas que ficou em primeiro lugar na *billboard*, etc. Os álbuns não precisam ter todas essas informações, ou seja, eles não precisam seguir o mesmo esquema de dados. Essa abordagem é bastante prática, pois, por exemplo, a grande maioria dos álbuns não possui nenhum *single* e, consequentemente, esse dado seria inexistente em muitos documentos. Em contraste, em um banco de dados relacional, todos os registros de uma tabela devem seguir um mesmo esquema. Os bancos NoSQL baseados em documentos permitem que os documentos, de uma mesma coleção, tenham esquemas diferentes [MongoDB Documentation 2024, Paniz 2016].

```
1    db.albuns.insertOne ({
2      "nome_album": "Legião Urbana",
3      "duracao": 3286
4    )}
5
```

Listagem 3.11. Inserção de documento no MongoDB (coleção albuns).

O comando *find* pode ser utilizado para buscar documentos no MongoDB. A Listagem 3.12 mostra como fazer uma busca na coleção *albuns* utilizando um filtro no formato BSON que busca pelo álbum de nome Legião Urbana. Sem o filtro (BSON vazio) o comando *find* busca por todos os documentos da coleção. Para fins ilustrativos, a instrução SQL da Listagem 3.13 em um SGBD relacional é equivalente ao comando MongoDB da Listagem 3.12. Assim como nos SGBDs relacionais, nos quais uma tabela deve possuir uma chave primária, no MongoDB todas as coleções possuem um campo chamado *_id*, que também funciona como chave primária, mas é gerenciado pelo própria banco. Assim, o resultado obtido pelo comando *find* da Listagem 3.12 deve retornar o _id de cada documento, além dos demais campos do documento [MongoDB Documentation 2024, Paniz 2016].

```
1    db.albuns.find ({'nome_album': 'Legião Urbana'})
2
```

Listagem 3.12. Busca de um documento no MongoDB.

```
1    SELECT * FROM albuns WHERE nome_album = 'Legião Urbana'
2
```

Listagem 3.13. Busca em um SGBD relacional.

Os relacionamentos no MongoDB são realizados referenciando o ID de um objeto de outra coleção (semelhante a uma chave estrangeira em SGBDs relacionais). Para exemplificar relacionamentos, considere o gênero musical *MPB* inserido através do código da Listagem 3.14. O documento desse gênero musical está representado na Listagem 3.15 cujo ID é igual a *ObjectId(54d1562cf7bb967d7b976d07)*. O ID do

gênero musical é utilizado para referenciá-lo em um álbum. O comando da Listagem 3.16 insere o álbum *Tom Jobim* com referência (campo *id_genero*) ao ID do gênero musical *MPB*. O documento gerado por esse comando é ilustrado na Listagem 3.17.

```
db.generos.insertOne({
  nome_genero: "MPB"
});
```

Listagem 3.14. Inserção de documento MongoDB (coleção gêneros).

```
{
  "_id": ObjectId("54d1562cf7bb967d7b976d07"),
  "nome_genero": "MPB"
}
```

Listagem 3.15. Documento da coleção de gêneros musicais.

```
db.generos.insertOne({
  "nome_album": "Tom Jobim"
  "duracao": 3200
  "id_genero": ObjectId("54d1562cf7bb967d7b976d07")
});
```

Listagem 3.16. Inserção de documento com referência a outro documento.

```
{
  "_id": ObjectId("63c828b5342dda43e93bee1e"),
  "nome_album": "Tom Jobim"
  "duracao": 3200
  "id_genero": ObjectId("54d1562cf7bb967d7b976d07")
}
```

Listagem 3.17. Documento de uma coleção com referência a outro documento.

Para exibir todos os álbuns e seus respectivos gêneros musicais, basta apenas listar todos os documentos da coleção de álbuns e buscar o documento da coleção de gêneros referenciado através do campo *id_genero*. O código da Listagem 3.18 mostra como fazer essa busca. Nesse código, a primeira linha busca todos os álbuns armazenando-os na variável *albuns*. Na segunda linha há a função *forEach* que recebe como argumento outra função (*album*), na qual é feita uma nova busca na coleção *generos* pelo documento correspondente ao _id *genero* de cada álbum [MongoDB Documentation 2024, Paniz 2016]

```
var albuns = db.albuns.find({});
albuns.forEach(function(album) {
  var genero = db.generos.findOne({"_id": album["id_genero"]});
  print(album["nome_album"], genero["nome_genero"]);
})
```

Listagem 3.18. Código para buscar todos os álbuns e seus respectivos gêneros musicais.

3.3.3. Modelo Colunar

O modelo colunar, ou famílias de colunas, armazenam dados em famílias de colunas como linhas. Essas linhas possuem inúmeras colunas relacionadas com a chave dessa linha. São coleções de dados que estão relacionados e são acessados juntos. Cada família de colunas pode ser comparada a um contêiner de linhas em uma tabela de um SGBD relacional. Nos SGBDs relacionais, a chave é usada para reconhecer cada linha. No modelo colunar, cada linha é composta por inúmeras colunas [Benymol and Sajimon 2017].

O modelo colunar utiliza-se de tabelas para representação de entidades e os dados são gravados em disco agrupados por coluna, o que reduz o tempo de leitura e escrita em disco. Os bancos colunares são os que mais se assemelham aos bancos relacionais por terem uma tabela, mesmo que, na verdade, eles sejam muito diferentes [Benymol and Sajimon 2017].

Cassandra, foco desta seção, é um dos bancos de dados de famílias de colunas mais conhecidos [Paniz 2016, Cassandra Documentation 2024]. Outros bancos de dados importantes de famílias de colunas são HBase, Microsoft Azure Cosmos DB, Datastax Enterprise, ScyllaDB, Microsoft Azure Table Storage, Accumulo, Google Cloud Bigtable e Amazon Keyspaces.

Cassandra utiliza o conceito de *keyspace*, que é similar a um *database*, onde tabelas são agrupadas para uma finalidade específica, geralmente para separar dados de aplicações diferentes. Outro conceito importante é o de família de colunas (*column family*) que é semelhante a uma entidade. Nas versões mais recentes de Cassandra o termo família de colunas foi substituído por tabela [Paniz 2016, Cassandra Documentation 2024]. A Figura 3.5 ilustra como os dados são organizados em Cassandra.

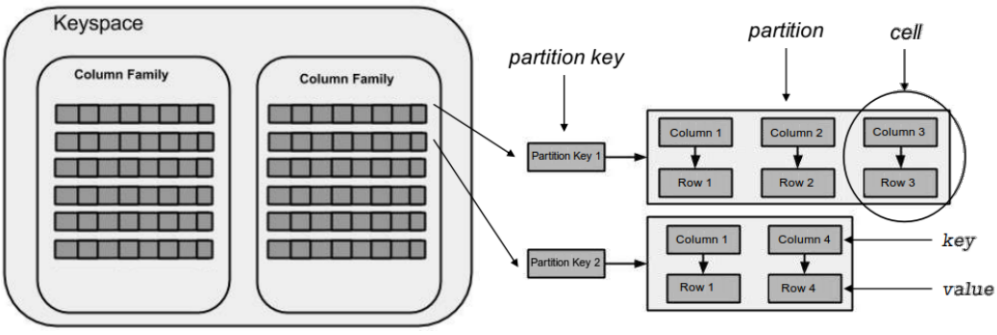


Figura 3.5. Colunas dinâmicas no banco de dados Cassandra.

Cassandra utiliza a linguagem CQL (*Cassandra Query Language*) que é semelhante a SQL utilizada por SGBDs relacionais. A *interface* de linhas de comandos cqlsh pode ser usada para interagir com Cassandra. Para usá-la, basta apenas digitar o seu nome (cqlsh) em um terminal de comandos do sistema operacional [Paniz 2016, Cassandra Documentation 2024]. A Listagem 3.19 mostra como criar uma tabela em Cassandra através do comando *CREAT TABLE*, no qual é criada uma tabela de filmes. Os comandos *SELECT, INSERT, UPDATE e DELETE* também são utilizados de forma idêntica a SQL [Paniz 2016,Cassandra Documentation 2024].

```
CREATE TABLE filmes (
  videoId uuid,
  nome_filme varchar,
  descricao varchar,
  localizacao_imagem_capa text,
  data timestamp,
  PRIMARY KEY (videoId)
)

```

Listagem 3.19. Craindo tabela no Cassandra.

Apesar da sintaxe da busca em Cassandra ser idêntica a da busca nos SGBD relacionais, diferentemente deles, Cassandra não faz buscas em campos que não possuem índices. Nos SGBDs relacionais, apesar de não ser performático, esse tipo de busca funciona. Porém, devido à maneira como o banco de dados Cassandra organiza os dados, uma busca por campos sem índices seria tão ineficiente que ele prefere não suportar. Isso ocorre porque Cassandra é otimizado para operações de leitura rápidas e eficientes, que são possíveis quando os dados são acessados diretamente por meio de chaves primárias ou índices secundários. Buscas em colunas sem índices podem exigir varreduras completas de tabelas, o que é ineficiente em ambientes distribuídos e escaláveis. Para ilustrar melhor essa situação, na Listagem 3.20 há uma consulta em CQL que não será executada se não for criado um índice na coluna filtrada, conforme ilustrado na Listagem 3.21 [Paniz 2016, Cassandra Documentation 2024].

```
SELECT * FROM filmes WHERE nome_filme = 'Matrix Revolutions'

```

Listagem 3.20. Busca utilizando um filtro em Cassandra.

```
CREATE INDEX ON filmes (nome_filme)

```

Listagem 3.21. Criação de índice em Cassandra.

A tabela criada com a sintaxe definida na Listagem 3.19 é muito semelhante a uma tabela no modelo relacional. Entretanto, o modelo de dados utilizado comumente em bancos colunares é o baseado em colunas dinâmicas (*dynamic columns*). A Figura 3.5 ilustra o modelo de colunas dinâmicas implementado em Cassandra. Nesse modelo, as tabelas são *schameless*, nas quais uma linha é chamada de partição (*partition*). Cada partição possui sua chave única (*partition key*) que, a partir desta chave, podem ser adicionados vários pares chave/valor chamados de pares coluna/-linha (*colunm/row*). E cada par coluna/linha em uma partição é chamado de célula (*cell*). As células de uma mesma partição são gravadas agrupadas para reduzir o tempo de leitura e escrita em disco [Paniz 2016, Cassandra Documentation 2024].

Para criar uma tabela com colunas dinâmicas é necessário utilizar o parâmetro *WITH COMPACT STORAGE* na criação da tabela. Além disso, a tabela deve possuir, pelo menos, três colunas: uma que será a chave da partição; uma que será o nome da coluna dinâmica e a última será o valor da linha. Adicionalmente,

a chave primária deve ser uma chave composta entre a chave da partição e o nome da coluna [Paniz 2016, Cassandra Documentation 2024].

A Listagem 3.22 exemplifica como criar uma tabela com colunas dinâmicas em Cassandra usando o parâmetro *WITH COMPACT STORAGE*. Nesse exemplo, a tabela *filmes_assistidos* terá o campo *userId* como chave de partição, os campos *watch_date* e *videoId* como nome da coluna dinâmica e, finalmente, os campos *nome_filme* e *localizacao_imagem_capa* como valores da linha. Nessa tabela, os campos *nome_filme* e *localizacao_imagem_capa* são campos provenientes da tabela filmes (Listagem 3.19) [Paniz 2016, Cassandra Documentation 2024].

```
1    CREATE TABLE filmes_assistidos (
2      userId uuid,
3      watch_date timestamp,
4      videoId uuid,
5      nome_filme varchar,
6      localizacao_imagem_capa text,
7      PRIMARY KEY (userId, watch_date, videoId)
8    ) WITH COMPACT STORAGE
9
```

Listagem 3.22. Criação de tabela com colunas dinâmicas em Cassandra por meio do parâmetro WITH COMPACT STORAGE.

Cassandra não tem nenhum conceito de integridade referencial e, portanto, não possui a operação *join*. Por esse motivo, os dados foram desnormalizados, evitando acessos à tabela *filmes*, para que uma busca na tabela *filmes_assistidos* fosse mais rápida. Como já dito anteriormente, a linguagem CQL de Cassandra é muito semelhante a SQL dos bancos relacionais, mesmo que a organização dos dados nos dois tipos de bancos sejam bem diferentes, como mostrado na Figura 3.5. Isso foi feito para que os programadores acostumados com SQL utilizem CQL mais facilmente. Em versões mais antigas do Cassandra, era permitido acessar as colunas dinâmicas de forma semelhante aos dados nos bancos chave-valor [Paniz 2016, Cassandra Documentation 2024].

Em versões mais recentes de Cassandra (a partir da versão 3.0) o uso de *WITH COMPACT STORAGE* foi desaconselhado e pode ser removido em versões futuras. A criação de tabelas com colunas dinâmicas pode ser feita sem esse parâmetro usando outras abordagens, como tabelas estáticas e coleções. A Listagem 3.23 apresenta uma código sem o uso do parâmetro *WITH COMPACT STORAGE* [Cassandra Documentation 2024].

```
1    CREATE TABLE filmes_assistidos (
2      userId uuid,
3      watch_date timestamp,
4      videoId uuid,
5      nome_filme varchar,
6      localizacao_imagem_capa text,
7      PRIMARY KEY (userId, watch_date, videoId)
8    )
9
```

Listagem 3.23. Criação de tabela com colunas dinâmicas em Cassandra sem o parâmetro WITH COMPACT STORAGE.

3.3.4. Modelo orientado a grafos

A matemática e a computação caminham juntas, e suas teorias foram aplicadas nos sistemas computacionais. Uma delas é a teoria dos grafos, que pode ser aplicada na representação de diversos problemas e implementada em *software* [Silva et al. 2021]. A teoria dos grafos oferece inúmeras ferramentas que podem ser aplicadas para quantificar e analisar os padrões de conectividade de sistemas complexos [Phillips et al. 2015].

A teoria dos grafos pode ser aplicada no contexto NoSQL, em especial ao modelo orientado a grafos. Esse modelo possui estruturas de dados modeladas na forma de grafos. A manipulação de dados é expressa mediante operações orientadas a grafos. Com isso, eles abstraem a complexidade dessas estruturas provendo ferramentas para explorar seu potencial.

Os dados nos bancos de grafos são representados por nodos, arestas e propriedades. Eles não implementam tabelas. Os nodos representam as entidades, as arestas expressam as relações entre os nodos e as propriedades apresentam características das entidades e relacionamentos [Phillips et al. 2015]. As bases de dados orientadas a grafos processam com eficiência densos conjuntos de dados e o seu *design* permite a construção de modelos preditivos e análise de correlações e de padrões de dados. Seguem as principais vantagens de se utilizar o modelo orientado a grafos:

- alto desempenho independentemente do tamanho total do conjunto de dados (garantido pelas travessias nos grafos);
- o modelo de grafos aproxima os domínios técnicos e de negócios facilitando a modelagem dos dados e;
- facilidade de se alterar o esquema de dados incluindo novas entidades e relacionamentos sem a necessidade de reestruturar o esquema de dados.

Exemplos de bancos orientados a grafos são: Neo4J, Microsoft Azure Cosmos DB, Aerospike, Virtuoso, ArangoDB, GraphDB, OrientDB, Memgraph, Amazon Neptune, NebulaGraph, Stardog, JanusGraph, TigerGraph, Fauna, Dgraph, Giraph e AllegroGraph. Nessa seção serão mostrados exemplos práticos de como modelar, consultar e recuperar dados no banco de dados Neo4j. Ele pode ser executado na máquina virtual Java e possui distribuições para Windows, Mac e Linux. Semelhantemente a outros bancos orientados a grafos, Neo4j pode ser aplicado a problemas que envolvem gerenciamento de rede, *softwares* analíticos, pesquisa cientifica, roteamento, gestão organizacional e de projeto, recomendações, redes sociais, dentre outros [Paniz 2016, Neo4J Documentation 2024].

Neo4J utiliza a linguagem de consulta *cypher* como uma ferramenta intuitiva para interagir com bancos de dados orientados a grafos. Cada nodo criado no Neo4j pode ter um tipo e propriedades associadas a ele. Na Listagem 3.24 são criados dois nodos que representam pessoas, utilizando o comando CREATE. Os dois nodos são do tipo Pessoa e possuem as propriedades *nome* e *idade*. Eles estão associados às variáveis *neo* e *smith* [Paniz 2016, Neo4J Documentation 2024].

```
CREATE (neo:Pessoa {nome:'Neo', idade: 29})
CREATE (smith:Pessoa {nome:'Agente Smith', idade: 36})
```

Listagem 3.24. Comando para criar nodos no Neo4J.

Para exibir os nodos inseridos, deve ser utilizado o comando *MATCH*. Além do *MATCH*, a outra parte obrigatória em um comando de busca é o *RETURN*, usado para especificar o que será retornado. O comando da Listagem 3.25 busca por todos os nodos do tipo Pessoa retornando seus nomes. Para fins ilustrativos, comando *cypher* da Listagem 3.25 é equivalente ao comando SQL da Listagem 3.26 [Paniz 2016, Neo4J Documentation 2024].

```
MATCH (m:Pessoa)
RETURN m.nome
```

Listagem 3.25. Comando para buscar todos os nodos de um determinado tipo no Neo4J.

```
SELECT m.nome FROM Pessoa m
```

Listagem 3.26. Comando para buscar todos os registros de uma tabela em SGBD relacional.

Uma alternativa ao método expresso na Listagem 3.24 é a utilização do comando *MERGE*. Porém, este comando verifica se o nodo já existe no banco de dados antes se o inserir. Se ele existir, ele não cria um novo nodo, evitando duplicação. A Listagem 3.27 representa a sintaxe para uso do comando *MERGE* no Neo4J [Paniz 2016, Neo4J Documentation 2024].

```
MERGE (neo:Pessoa {nome: 'Neo', idade: 29});
MERGE (smith:Pessoa {nome: 'Agente Smith', idade: 36});
```

Listagem 3.27. Comando para criar nodos no Neo4J sem duplicação.

Após nodos terem sido criados, arestas (relacionamentos) são necessárias para dar forma ao grafo. A Listagem 3.28 mostra como criar um relacionamento no Neo4j. Primeiro, dois nodos já existentes são buscados e armazenados nas variáveis *p1* e *p2* através do comando MATCH. Em seguida, um relacionamento é criado entre *p1* e *p2* através do comando CREATE, no qual é atribuído a propriedade *CONHECE* [Paniz 2016, Neo4J Documentation 2024].

```
MATCH (p1:Pessoa {nome: 'Neo'}),
  (p2:Pessoa {nome: 'Agente Smith'})
CREATE (p1)-[:CONHECE]->(p2)
```

Listagem 3.28. Comando para criar relacionamento no Neo4J.

A Listagem 3.29 apresenta uma alternativa para criar arestas em relação ao expresso na Figura 3.28. O diferencial para esta abordagem é que ela verifica a existência do relacionamento antes de criá-lo. A cláusula *WHERE NOT* garante que o relacionamento *CONHECE* será criado somente se ainda não existir entre *p1* e *p2* [Paniz 2016, Neo4J Documentation 2024].

```
MATCH (p1:Pessoa {nome: 'Neo'}),
  (p2:Pessoa {nome: 'Agente Smith'})
  WHERE NOT (p1)-[:CONHECE]->(p2)
CREATE (p1)-[:CONHECE]->(p2)

```

Listagem 3.29. Comando para criar de relacionamento no Neo4J sem duplicação.

Enfim, existem várias alternativas para criar relacionamentos no Neo4j, dependendo do contexto e das necessidades específicas da aplicação. A escolha da alternativa mais apropriada depende dos requisitos, como evitar duplicações, melhorar a flexibilidade, ou verificar condições adicionais antes da criação do relacionamento. Todas essas alternativas são válidas e podem ser usadas conforme a situação específica do banco de dados e da aplicação em questão [Paniz 2016, Neo4J Documentation 2024].

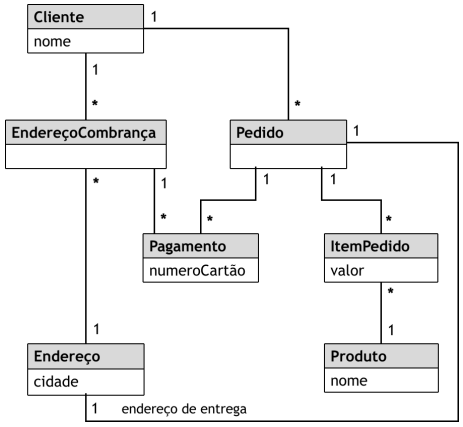
3.4. Projeto de bancos de dados NoSQL

3.4.1. Agregados

Os bancos de dados NoSQL chave-valor (Seção 3.3.1), documento (Seção 3.3.2) e colunar (Seção 3.3.3) podem fazer uso de uma estrutura de dados complexa chamada de agregado. Essa estrutura é capaz de conter listas e outras estruturas de dados aninhadas dentro dela. O agregado é a representação de um conjunto de objetos relacionados tratados como uma unidade que pode ser atualizada por meio de operações atômicas. Ele facilita a execução de bancos de dados em um *cluster*, visto que um agregado representa uma unidade natural para replicação e fragmentação (ver Seção 3.4.4). Além disso, os agregados são mais simples de ser manipulados pelos desenvolvedores, pois eles se assemelham às estruturas de dados utilizadas nas linguagens de programação. Os bancos de dados NoSQL orientados a grafos (Seção 3.3.4) não implementam agregados [Strauch et al. 2011].

Embora os SGBD orientados a agregados sejam projetados para trabalhar com unidades de objetos agreados, esses sistemas não implementam essas unidades de forma automatizada. A fim de exemplificar a implementação de agregados, considere a modelagem relacional de um sistema de pedidos de produtos representada na Figura 3.6(a). A Figura 3.6(b) exemplifica dados típicos para o banco de dados da modelagem da Figura 3.6(a). A modelagem está normalizada, de modo a não existirem valores repetidos em tabelas diferentes, igual como se espera no modelo relacional. Contudo, essa modelagem não está detalhada devido a limitações de espaço e por questões didáticas [Sadalage and Fowler 2019].

O mesmo sistema representado na Figura 3.6 está representado na Figura 3.7. Porém, enquanto a Figura 3.6 implementa uma modelagem relacional, a modelagem da Figura 3.7 considera a orientação agregada, que é bastante diferente. Na modelagem da Figura 3.7(a), os dados agregados estão representados através do marcador de composição UML, o losango preto. Assim, existem os agregados Cliente, Pedido e Pagamento. A Figura 3.7(b) representa os objetos modelados na Figura 3.7(a) [Sadalage and Fowler 2019].



(a) Modelagem relacional.

Cliente

id	nome
1	João

Pedidos

id	idCliente	idEndereçoEntrega
31	1	21

Endereço

id	cidade
21	Teresina

EndereçoCobrança

id	idCliente	idEndereço
11	1	21

ItemPedido

id	idPedido	idProduto	preço
41	31	5	3750.99
43	31	6	55.60

Produto

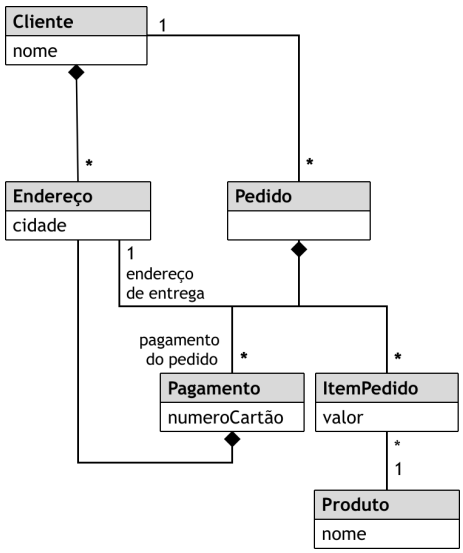
id	nome
5	notebook
6	mouse

Pagamento

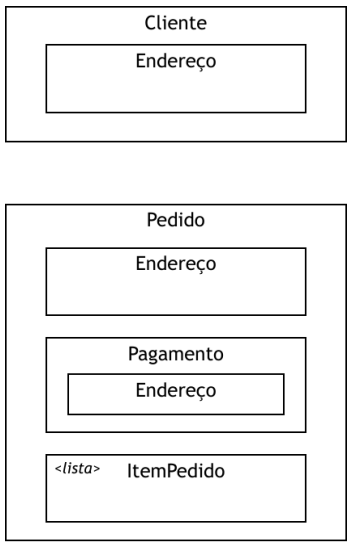
id	numeroCartão	idPedido	idEndereçoCobrança
11	111 111 11	31	21

(b) Dados típicos de tabelas de um banco de dados.

Figura 3.6. Modelagem de sistema de pedidos de produtos em SGBD relacional [Sadalage and Fowler 2019].



(a) Agregados de clientes e pedidos separados.

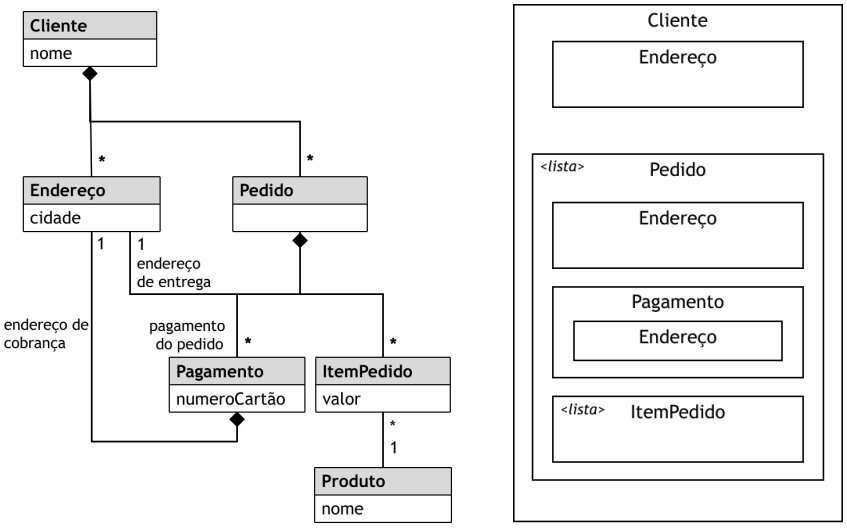


(b) Objetos de clientes e pedidos separados.

Figura 3.7. Esquema de pedidos de produtos em modelagem orientada a agregados [Sadalage and Fowler 2019].

O agregado Cliente possui o objeto Endereço embutido dentro dele. O agregado Pedido, por sua vez, possui os objetos Endereço e Pagamento e, ainda, uma lista de objetos ItemPedido aninhada. Por fim, o objeto Pagamento é um agregado que possui um objeto Endereço embutido. Observe que um mesmo registro de endereço pode se repedir três vezes para um pedido de um cliente na forma de endereço do cliente, endereço de cobrança e endereço de envio. A conexão entre o cliente e seus pedidos é feita por relacionamento entre agregados através do ID do cliente. A modelagem aplicou uma desnormalização, onde os produtos passaram a fazer parte de itens de pedido, ou seja, cada objeto da lista ItemPedido também possui informações dos produtos [Sadalage and Fowler 2019, Frozza et al. 2022].

Não existe uma maneira específica e bem definida de como modelar agregados. Esse processo deve ser feito levando em consideração como os dados devem ser acessados pelo aplicativo. A orientação agregada espera que um agregado seja um conjunto de objetos relacionados e acessados (recuperados/gravados) juntos. A modelagem da Figura 3.7(a) permite que todas as informações de um pedido e de todos os seus produtos sejam acessadas como uma unidade. Contudo, os limites dos agregados poderiam ser definidos de forma diferente, como no exemplo da Figura 3.8. Nesse exemplo, todos os pedidos (e produtos) de um cliente estão no agregado Cliente. Dessa maneira, é possível acessar todos os pedidos de um determinado cliente por vez. A Figura 3.8(b) ilustra os objetos da modelagem da Figura 3.8(a).



(a) Agregado de cliente com pedidos embutidos.

(b) Agregado de cliente com pedidos embutidos.

Figura 3.8. Dois esquemas de pedidos de produtos em modelagem orientada a agregados [Sadalage and Fowler 2019].

A Listagem 3.30 exemplifica dados típicos considerando a modelagem da Figuras 3.7. Nessa listagem os objetos *cliente* e *pedido* estão separados. Em contraste, a Listagem 3.31 exemplifica dados da modelagem 3.8 cujos objetos estão aninhados [Sadalage and Fowler 2019].

```
// Cliente
{
  "id": 1,
  "nome": "João",
  "enderecoCobrança": [ {"cidade": "Teresina"} ]
}
// Pedido
{
  "id": 31,
  "idCliente": 1,
  "enderecoEntrega": {"cidade": "Teresina"},
  "itemPedido": [
                  {
                    "idProduto": 6,
                    "nome": "mouse",
                    "preço": 55.60
                  },
                  {
                    "idProduto": 5,
                    "nome": "notebook",
                    "preço": 3750.99
                  }
                ],
  "pagamento": [
                  {
                    "numeroCartão": "111 111 11",
                    "EndereçoCobrança":
                    {
                      "cidade": "Teresina"
                    }
                  }
                ]
}

```

Listagem 3.30. Dados de clientes e pedidos separados.

```
// Cliente e pedidos embutidos
{
  "id": 1,
  "nome": "João",
  "enderecoCobrança": [ {"cidade": "Teresina"}],
  "pedidos":[
              {
                "id": 31,
                "enderecoEntrega": {"cidade": "Teresina"},
                "itemPedido": [
                                {
                                  "idProduto": 6,
                                  "nome": "mouse",
                                  "preço": 55.60
                                },
                                {
                                  "idProduto": 5,
                                  "nome": "notebook",
                                  "preço": 3750.99
```

```
                                        }
                                    ],
                        "pagamento": [
                        {
                          "numeroCartão": "111 111 11",
                          "EndereçoCobrança": {"cidade": "Teresina"}
                        }]
                      }
    ]
}

```

Listagem 3.31. Dados de clientes com pedidos embutidos.

Os exemplos mostrados nessa seção evidenciam que não há uma maneira universal para modelar agregados. Esse processo depende de como os dados serão manipulados, ou seja, depende da forma pela qual os dados são usados pelo aplicativo. Cada modelagem possui seus *trade-offs*. O segundo agregado (Figura 3.8) permite um acesso rápido a todos os pedidos de um cliente. Porém, ele torna custoso o acesso a determinados pedidos como, por exemplo, o acesso aos pedidos do último trimestre. Em contraste, o primeiro agregado (Figura 3.7) facilita a busca por um determinado pedido, mas é potencialmente mais lento para encontrar todos os pedidos de um cliente [Sadalage and Fowler 2019, Rodrigues et al. 2017].

Como desvantagem, os bancos de dados orientados a agregados não suportam transações ACID. Esses sistemas suportam apenas manipulações atômicas em um único agregado por vez. Assim, a manipulação atômica de múltiplos agregados, como em uma transação ACID, deve ser gerenciada pelo código do aplicativo. Caso contrário, pode haver inconsistências nos dados em caso de uma falha. Na modelagem orientada a agregados, é esperado que as necessidades de atomicidade dos dados sejam agrupadas em um único agregado. Em contraste, os bancos de dados NoSQL orientados a grafos implementam ACID ao invés de agregados [Sadalage and Fowler 2019, Rodrigues et al. 2017].

3.4.2. Relacionamentos

Os relacionamentos entre entidades representam as associações e interações entre diferentes entidades no banco de dados. Esses relacionamentos são estabelecidos para refletir as conexões lógicas entre as entidades do mundo real. Os relacionamentos em SGBDs NoSQL são semelhantes aos relacionamentos em SBDS relacionais. Porém, existem algumas observações importantes a serem ressaltadas.

3.4.2.1. Relacionamentos em bancos de dados orientados a agregados

Os agregados são a parte central na modelagem de bancos de dados NoSQL orientados a agregados. Eles são modelados para capturar a maioria das necessidades de acesso aos dados de um sistema. Porém, às vezes as aplicações precisam acessar dados relacionados em agregados diferentes [Sadalage and Fowler 2019, Frozza et al. 2022].

No exemplo da Figura 3.7, os agregados de clientes e pedidos estão separados. Contudo, eles devem estar relacionados de alguma maneira para garantir que os clientes encontrem seus pedidos (e vice-versa). Uma maneira simples de conectar agregados separados é o uso de IDs. O agregado Pedido da Figura 3.7 possui o campo idCliente que contem o ID do cliente do pedido. Nesse caso, por exemplo, para obter os dados do cliente através de um pedido, é necessário ler o pedido, descobrir o ID do cliente e acessar o banco de dados novamente para ler os dados do cliente através de seu ID [Sadalage and Fowler 2019, Rodrigues et al. 2017].

Usar IDs para relacionar agregados é uma abordagem simples e eficiente, mas o SGBD não possui ciência dos relacionamentos. Esses SGBDs não tratam restrições de integridade de relacionamentos. Nesse caso, a consistência dos dados deve ser gerenciada pelas aplicações. Por exemplo, a aplicação que precisar atualizar múltiplos agregados de uma vez deve lidar com uma eventual falha durante o processo. Sistemas com muitos relacionamentos são mais adequados em SGBDs relacionais do que em SGBDs NoSQL. SGBDs relacionais permitem que múltiplos registros possam ser modificados em uma transação com as garantias ACID [Sadalage and Fowler 2019, Rodrigues et al. 2017].

3.4.3. Relacionamentos em bancos de dados orientados a grafos

A maioria dos bancos de dados NoSQL (de agregados) foi concebida pela necessidade de escalar grandes quantidades de dados em *clusters* (ver Seção 3.2.2.1 para mais detalhes). Por outro lado, os bancos de dados de grafos foram criados com uma intenção diferente: relacionar muitos registros pequenos como muitas conexões complexas. Embora os SGBDs relacionais possam implementar relacionamentos, as junções utilizadas em consultas com relacionamentos podem ser muito custosas, diminuindo o desempenho de consultas. Em contrate, os SGBDs NoSQL de grafos são mais simples. Eles utilizam os conceitos de grafos, ou seja, nodos ligados por arestas. Além disso, eles possuem a implementação dos algoritmos clássicos de grafos, como caminhos de custo mínimo, travessia em largura, travessia em profundidade, etc [Gueidi et al. 2021, Boudaoud et al. 2022, Passos et al. 2023].

Em muitos casos, os SGBDs orientados a grafos pode ser utilizados para resolver problemas em relacionamentos que são muito custosos em SGBDs relacionais. Nesse caso, uma alternativa para o problema é migrar os dados de um SGBD para o outro. Em muito casos, não é necessário migrar o banco inteiro, apenas a parte dos dados indispensáveis para resolver o problema. Diversas propostas na literatura lidam com a problemática de mapeamento entre SGBDs relacionais e SGBDs de grafos [Gueidi et al. 2021, Boudaoud et al. 2022, Passos et al. 2023]. No entanto, existe um consenso quanto às regras de mapeamento do modelo relacional para o modelo de grafos na grande maioria dos trabalhos, como ilustrado na Tabela 3.1. Cada uma das colunas da tabela relaciona os objetos que podem ser equivalentes nos dois tipos de SGBD.

O exemplo da Figura 3.9 considera as regras clássicas de mapeamento relacional-grafo mostradas na Tabela 3.1. Nesse exemplo, as tabelas de um SGBD relacional (Figura 3.9(a)) são convertidas em nodos e arestas (Figura 3.9(b)) de um SGBD

em grafos. Cada uma das tuplas das tabelas concebeu um nodo. Cada um dos nodos possui uma propriedade cujo valor é único (não mostrado na figura) gerado a partir das chaves primárias. Este controle é realizado por uma restrição de integridade implementada no SGBD. Cada uma das arestas foi gerada a partir das chaves estrangeiras. Por exemplo, o nodo do personagem *Homer Jay Simpson* possui um relacionamento *Familiar*, por meio da aresta *idFamilia*, com o nodo *Simpsons* [Passos et al. 2023].

Tabela 3.1. Regras de mapeamento relacional-grafo [Passos et al. 2023].

Relacional	tabela	tupla	atributo	chave primária	chave estrangeira
Grafo	rótulo	nodo	propriedade	propriedade "ID"	aresta

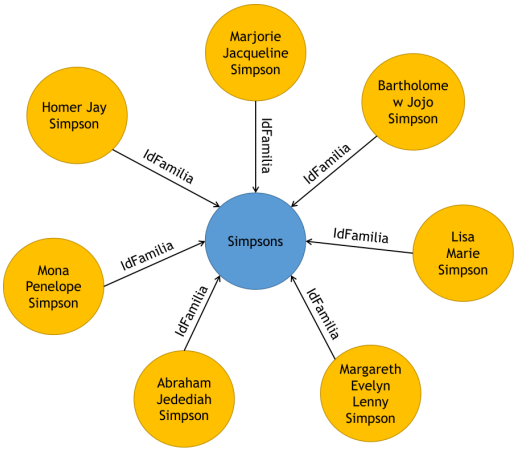
Personagens

id	Nome	IdFamilia
1	Homer Jay Simpson	1
2	Marjorie Jacqueline Simpson	1
3	Bartholomew Jojo Simpson	1
4	Lisa Marie Simpson	1
5	Margareth Evelyn Lenny Simpson	1
6	Abraham Jedediah Simpson	1
7	Mona Penelope Simpson	1

Familias

id	Nome
1	Simpsons

(a) Tabelas de banco de dados relacional.



(b) Nodos e arestas de banco de dados em grafo.

Figura 3.9. Exemplo de mapeamento relacional-grafo [Passos et al. 2023].

3.4.4. Distribuição de dados

Uma das principais motivações para o uso da tecnologia NoSQL é sua facilidade de escalar em Bancos de Dados Distribuídos (BDD). Em um BDD, o banco de dados é dividido em vários servidores conectados por meio de uma rede de computadores de forma transparente aos usuários. Assim, um sistema de BDD armazena fisicamente os dados em várias máquinas (*sites*) e os gerencia de forma independente. Esses sistemas computacionais são também chamados de *clusters* e possuem alta disponibilidade, balanceamento de carga e processamento paralelo. Existem duas maneiras de implementar um *cluster*: fragmentação e replicação [VolDB Documentation 2020, Faerber et al. 2017, Özsu and Valduriez 1996].

3.4.4.1. Fragmentação

Na fragmentação, uma tabela é particionada em fragmentos e cada fragmento pode ser armazenado em diferentes *sites* do sistema. Dessa maneira, um banco de dados

pode possuir muito mais recursos (armazenamento, memória e processamento) do que teria em apenas um *site*. As tabelas podem ser dividas verticalmente (em colunas) ou horizontalmente (em linhas). A Figura 3.10 ilustra um particionamento de tabelas de banco de dados. Por exemplo, a tabela A é divida nas partes A', A" e A"' armazenadas nas partições X, Y e Z, respectivamente. Cada uma das partições é armazenada em um máquina diferente. Uma transação que precise acessar A' deve acessar apenas o *Site* 1. Para uma transação percorrer a tabela A inteira, ela deve acessar os três *sites* [VolDB Documentation 2020, Taft et al. 2014, Özsu and Valduriez 1996].

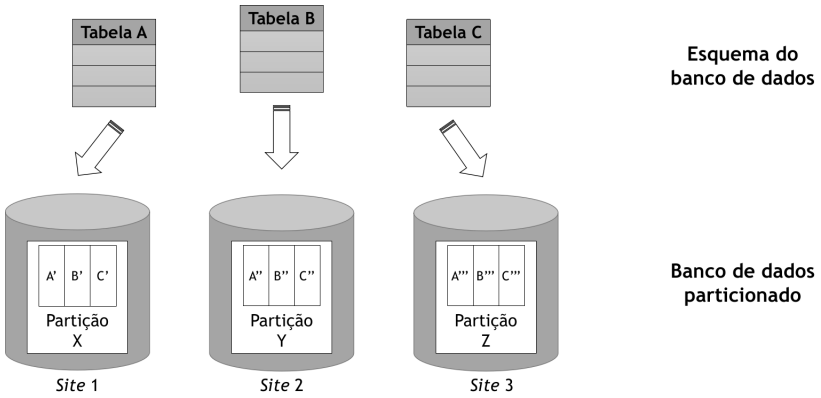


Figura 3.10. Tabelas de banco de dados particionadas [VolDB Documentation 2020].

Em um projeto ótimo de BDD, cada transação deve acessar apenas um *site* para obter uma resposta mais rápida. Além disso, a carga de trabalho deve ser balanceada entre os *sites* para não sobrecarregar uma determinada máquina. Por motivos óbvios, esse cenário é raro. Para se aproximar dele, é necessário que os dados acessados em conjunto estejam em apenas uma partição. Nesse contexto, o agregado pode funcionar como uma unidade óbvia de distribuição, uma vez que ele é projetado para combinar dados que são, normalmente, acessados em conjunto [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

Existem diversos fatores que podem influenciar no desempenho do sistema quanto a maneira como os dados devem ser organizados em cada nodo. Se o acesso a um conjunto de agregados é baseado em uma localização física, os dados podem ser colocados próximo ao lugar onde são acessados. Por exemplo, o pedido de um cliente da cidade de São Paulo pode ser armazenado no centro de dados da região sudeste do país. Uma alternativa é organizar os agregados de modo que sejam distribuídos em paralelo entre os nodos para que eles recebam cargas de trabalho iguais. Também pode ser "performático"juntar dados que comumente são acessados em sequência [Faerber et al. 2017, Özsu and Valduriez 1996].

Em muitos casos, a fragmentação segue a lógica do aplicativo. Por exemplo, a primeira letra do sobrenome do usuário pode ser usada como critério de organização dos agregados entre os nodos diferentes. Contudo, isso dificulta a programação dos aplicativos, visto que o código deve gerenciar o acesso aos diversos fragmentos. Além disso, um rebalanceamento no sistema implica em alterar o código do aplicativo e

migrar os dados. Alguns bancos de dados possuem auto fragmentação, deixando o banco de dados responsável pelo gerenciamento do acesso ao fragmento de cada dado [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

3.4.4.2. Replicação

A replicação provê a manutenção de várias cópias idênticas do banco de dados em vários *sites* diferentes. Entre os principais benefícios da replicação de dados estão: (i) a redundância, o que torna o sistema tolerante a falhas; (ii) a possibilidade de um balanceamento de carga do sistema, uma vez que o acesso pode ser distribuído entre as réplicas, e finalmente, (iii) o *backup online* dos dados, já que todas as réplicas estariam sincronizadas [Özsu and Valduriez 1996]. As técnicas de replicação mais conhecidas são a Mestre-Escravo (*Master-Slave*) e a Ponto a Ponto (*Peer-to-Peer* – P2P) .

A replicação Mestre-Escravo (Figura 3.11) possui um nodo primário, denominado mestre, responsável por processar e gerenciar todas as operações de gravação. Ela ainda possui um ou mais nodos secundários, chamados de escravos, que replicam passivamente os dados do mestre e atendem consultas de leitura. Caso aconteça um *crash* no mestre, uma das réplicas pode assumir o seu lugar. Alguns sistemas fazem esse processo de forma automática. Por exemplo, Redis possui um serviço, chamado de Sentinel, que monitora a integridade de seus nodos, detecta falhas e promove um novo nodo mestre se o mestre atual falhar [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

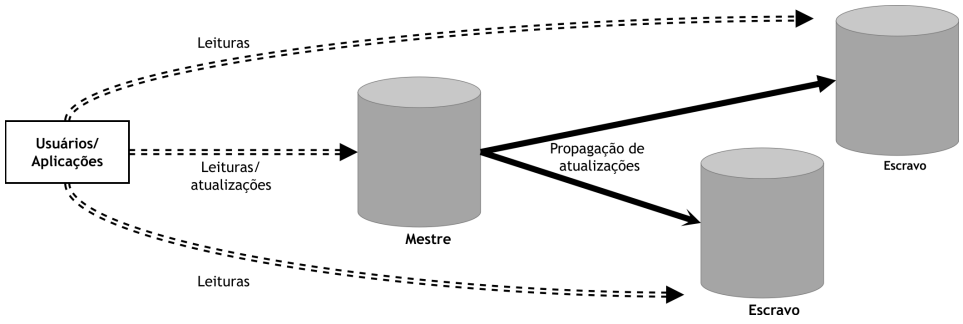


Figura 3.11. Replicação mestre/escravo.

A replicação Mestre-Escravo possui as vantagens de distribuir efetivamente a carga de trabalho entre vários nodos, permitindo desempenho otimizado de consulta, maior confiabilidade dos dados e minimização do tempo de inatividade do sistema. Existem ainda variantes do paradigma Mestre-Escravo que oferecem diferentes níveis de desempenho, tolerância a falhas e garantias de consistência [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

Uma das grandes vantagens da replicação é a resiliência de leitura. Nesse caso, se o mestre falhar, as réplicas ainda podem continuar atendendo requisições de leitura. Contudo, as requisições de gravação ficam comprometidas até o mestre ser restaurado ou um novo mestre seja eleito. Para garantir a resiliência de leitura,

é necessário que os caminhos de leitura e gravação sejam diferentes. Por exemplo, as conexões de leitura e gravação devem ser diferentes. Assim, caso a conexão de gravação falhe, a conexão de leitura pode continuar operando. Porém, muitas bibliotecas de interação com bancos de dados não suportam essa funcionalidade [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

Uma desvantagem da replicação é a possibilidade de inconsistência. Clientes diferentes podem ler valores diferentes de escravos diferentes, caso as atualizações não sejam propagadas de forma uniforme em cada escravo. Assim, um cliente pode ler um valor desatualizado. Contudo, essa inconsistência é temporária. Em algum momento as atualizações serão propagados e os clientes poderão ler dados consistentes. Esse intervalo de tempo em que as réplicas estão desatualizadas é chamado de janela de inconsistência. Uma alternativa seria ter apenas uma réplica para evitar leituras inconsistentes nos escravos. Porém, mesmo assim, caso o mestre falhe antes de propagar suas atualizações, o escravo ainda continuará inconsistente. Uma outra hipótese é um cliente não conseguir ver sua própria atualização. Por exemplo, um cliente atualiza seus dados no nodo mestre e os lê em um nodo escravo. Caso a sua atualização ainda não tenha sido propagada para o escravo, ele lerá dados desatualizados [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

A replicação Ponto a Ponto (Figura 3.12) não possui um mestre. Todos os nodos podem receber gravações ou leituras e a perda de um nodo não impede o acesso ao armazenamento de dados. Além disso, novos nodos podem ser inseridos mais facilmente para melhorar o desempenho. Uma rede Ponto a Ponto é mais adequada em armazenamento de dados imutáveis. Inconsistências de leitura podem acontecer, mas elas são temporárias [Sadalage and Fowler 2019, Özsu and Valduriez 1996].

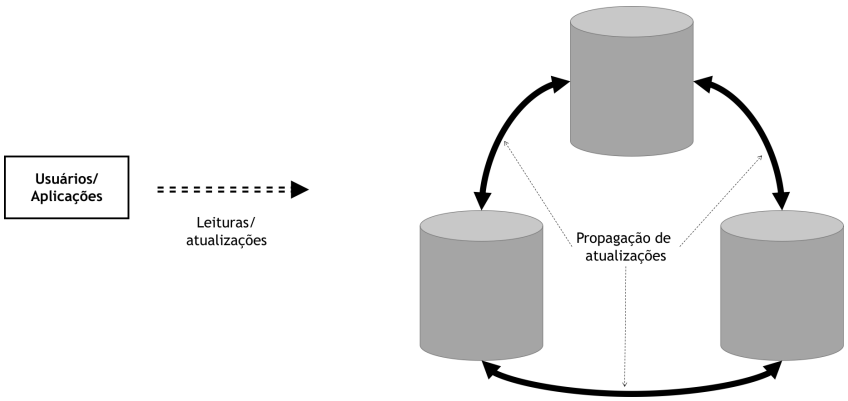


Figura 3.12. Replicação ponto a ponto.

A utilização de replicação Ponto a Ponto em sistemas muito atualizados é mais complexa. Caso duas réplicas atualizem o mesmo dado ao mesmo tempo, pode acontecer um conflito de gravação. Por exemplo, dois nodos podem ter o mesmo dado alterado simultaneamente por clientes diferentes. Nesse caso, haverá um impasse de qual atualização deve ser a correta. Para resolver esse problema, pode ser utilizado um servidor (ou servidores) para gerenciar versões de atualização de

dados e identificar a versão corrente. Contudo, o servidor pode se tornar um ponto de falha. Uma alternativa são os protocolos de propagação de atualização, tais como Phase King [Berman and Garay 1993], Chubby [Burrows 2006] e Paxos [Chandra et al. 2007]. Contudo, esse protocolos implicam em aumento do tráfego na rede para confirmar as atualizações. A Seção 3.4.5 discute outras maneiras de como lidar com conflitos em sistemas NoSQL.

3.4.5. Consistência e durabilidade

A consistência é o aspecto mais diferente entre os projetos de bancos de dados relacional e NoSQL. Os sistemas relacionais tentam implementar uma consistência forte que evita inconsistência a todo momento. Por exemplo, o SGBD deve garantir que o comprimento máximo do tipo de dados, restrições de chaves primária e estrangeira, etc., sejam absolutamente mantidos. Em contraste, os sistemas NoSQL podem tolerar algum nível de inconsistência, como uma consistência eventual [Davoudian et al. 2018, Rodrigues et al. 2017].

No contexto de sistemas de banco de dados, a consistência garante que a execução das transações não violem nenhuma restrição de integridade do sistema. Para isso, comumente, os SGBDs utilizam algum método de controle de concorrência (ver Seção 3.2.1 para mais detalhes). Por outro lado, a consistência em sistemas distribuídos garante que cada réplica tenha a mesma visão de dados em um determinado momento, independentemente de qualquer atualização de dados realizada pelos clientes. Por exemplo, uma solicitação para qualquer nodo deve retornar exatamente a mesma resposta, dando a impressão de que o sistema possui apenas um nodo [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Os bancos de dados NoSQL orientados a grafos implementam uma consistência baseada no modelo ACID, semelhante aos bancos de dados relacionais. Por outro lado, os SGBDs orientados a agregados (ver Seção 3.4.1) não suportam transações ACID. Ao invés disso, eles suportam atualizações atômicas dentro de um único agregado. Assim, atualizações em um agregado não geram inconsistência lógica. Porém, se o banco de dados for distribuído, pode haver janelas de inconsistência. Sistemas NoSQL podem ter uma janela de inconsistência bem curta, com menos de um segundo [Rodrigues et al. 2017, Rodrigues et al. 2017].

A modelagem da Figura 3.7 permite atualizações consistentes nos agregados cliente e pedido. Assim, atualizações em um cliente ou em itens de um pedido são feitas de forma atômica. Porém, não é possível uma atualização atômica em um cliente e em seus pedidos, visto que cliente e pedido são agregados diferentes. Uma atualização desse tipo deve ser gerenciada pela aplicação, sob pena de gerar alguma inconsistência. A modelagem da Figura 3.8 evita esse problema fazendo do cliente e seus pedidos um único agregado [Rodrigues et al. 2017, Sadalage and Fowler 2019].

3.4.5.1. Consistência de replicação

A replicação de dados pode introduzir a consistência de replicação no sistema, que é significativamente diferente da consistência lógica. A inconsistência de replicação

pode acontecer devido a janela de inconsistência. Caso o sistema falhe durante a propagação de dados, alguma réplica pode não ser atualizada devidamente. Assim, as réplicas vão retornar resultados diferentes a uma mesma solicitação [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Para ilustrar essa consistência de replicação, considere o exemplo de sistema de reservas de quartos de hotel ilustrado na Figura 3.13. Esse sistema possui um banco de dados distribuído entre os nodos 1, 2 e 3, que são réplicas idênticas. Suponha que os usuários 1, 2 e 3 estejam acessando o sistema no mesmo momento e visualizando os dados através dos nodos 1, 2 e 3, respectivamente. Suponha também que existe apenas um quarto disponível no hotel. Os usuários 2 e 3 se conhecem e estão conversando por telefone sobre o último quarto. Enquanto isso, o usuário 1 faz a reserva do quarto. Consequentemente, o sistema atualiza a disponibilidade replicada do quarto. Porém, a propagação dos dados chega no nodo 2 antes do nodo 3. Isso pode ocorrer devido a atrasos na rede, por exemplo. Assim, o usuário 2 vê o quarto reservado enquanto o usuário 3 vê o quarto disponível, ou seja, eles veem respostas diferentes para a mesma solicitação. Contudo, essa leitura inconsistente é temporária. Em algum momento a atualização será totalmente propagada e todos os usuários verão a mesma informação. Essa situação é chamada de consistência eventual, ou seja, em algum momento os nodos podem estar inconsistentes, mas eles acabarão sendo atualizados [Rodrigues et al. 2017, Sadalage and Fowler 2019].

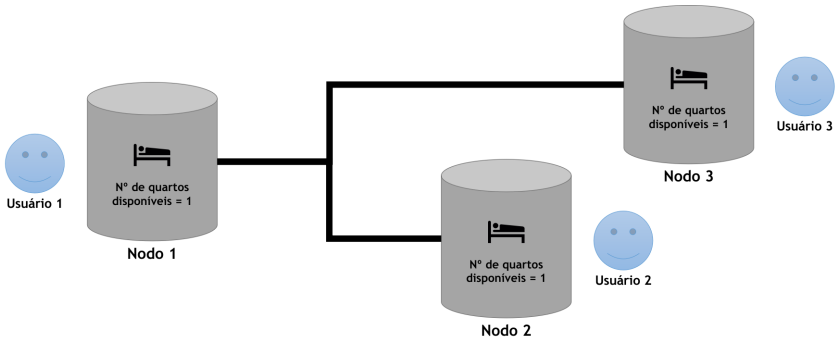


Figura 3.13. Sistema distribuído de reservas de quartos de hotel.

Janelas de inconsistência também podem ser um problema para usuários quando a inconsistência é consigo mesmo. Considere um sistema de *blog* implantado em um *cluster* cujos usuários podem fazer postagens e comentários instantâneos. Suponha que esse sistema usa o modelo mestre-escravo. Além disso, o balanceamento de carga do sistema permite escritas apenas no mestre e leitura apenas nos escravos. Nesse cenário, pode acontecer uma situação em que um usuário faz uma postagem, mas não consegue lê-la, dando a impressão de que ela foi pedida. Isso pode acontecer se a propagação de dados do mestre para o escravo possui um atraso. Ou seja, o usuário escreve no mestre, mas não consegue ver sua postagem no escravo devido à janela de inconsistência. Obviamente, a inconsistência é temporária e o usuário verá sua postagem em um futuro próximo, mesmo que talvez demore apenas um segundo. A Seção 3.4.5.2 discute alguns exemplos práticos de inconsistência de replicação e manerias de lidar com ela [Rodrigues et al. 2017, Sadalage and Fowler 2019].

3.4.5.2. Teorema CAP

O balanceamento da inconsistência é algo inevitável em projetos de sistemas distribuídos NoSQL. Além disso, diferentes domínios de projeto possuem diferentes tolerâncias à inconsistência. Para considerar a tolerância a inconsistência, os projetos NoSQL usam frequentemente o teorema CAP com o objetivo relaxar a consistência. CAP é um acrônimo derivado de suas três propriedades principais: *Consistency, Availability, and Partition Tolerance* (Consistência, Disponibilidade e Tolerância a Partições) [Rodrigues et al. 2017, Sadalage and Fowler 2019].

A definição básica do teorema CAP afirma que um sistema distribuído pode garantir apenas duas das suas três características desejadas. Consistência significa que todos os clientes veem os mesmos dados ao mesmo tempo, não importa em qual nodo eles se conectem. Disponibilidade significa que qualquer cliente que fizer uma solicitação de dados obterá uma resposta, mesmo que um ou mais nodos estejam desativados. Tolerância de partição significa que o *cluster* deve continuar a funcionar mesmo se ocorrer uma ou mais falhas de comunicação entre os nodos no sistema [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Os SGBDs relacionais centralizados utilizam um sistema CA (*Consistency and Availability*), ou seja, garantem consistência e disponibilidade, mas não tolerância a partição. Uma vez que existe apenas um nodo no sistema, não é necessário se preocupar com particionamentos. Ainda é possível implementar o *cluster* CA. Porém, se houver um particionamento, todos os nodos ficam incomunicáveis. A implementação de um *cluster* CA é possível, mas muito custosa, visto que deve ser assegurado que ele raramente se particionará [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Considere o cenário do exemplo de reservas de quartos de hotel ilustrado na Figura 3.13. Suponha que o sistema está replicado em um modelo ponto-a-ponto. Para assegurar a consistência, antes do sistema confirmar a reserva do usuário 1 no nodo1, ele deve se comunicar com os outros nodos para que eles concordem com a solicitação do usuário. A atualização é confirmada para o usuário apenas após todos os nodos confirmarem a atualização. Esse protocolo assegura a consistência. Porém, caso a conexão de rede falhe em algum dos nodos, nenhum deles poderá fazer reservas, ou seja, o sistema não proverá disponibilidade [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Manter um *cluster* CA é proibitivamente caro. Assim, os *clusters* devem ser tolerantes a partições. O teorema CAP afirma que só se pode ter duas de suas três propriedades. Porém, um sistema não pode descartar totalmente a consistência ou a disponibilidade. Dessa maneira, um sistema distribuído deve balancear a consistência com a disponibilidade, ou seja, o sistema não deve ser totalmente consistente e nem totalmente disponível [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Para melhorar a disponibilidade, o *cluster* descrito nessa seção poderia utilizar o protocolo mestre-escravo. Suponha que o nodo 1 foi o escolhido como mestre e o usuário 1 fez a reserva do último quarto de hotel. Adicionalmente, suponha que a propagação de atualização do mestre pra os escravos teve algum atraso. Como

consequência, os usuários 2 e 3 verão informações inconsistentes, mas não conseguirão fazer reservas. Porém, ainda haverá um problema caso a conexão entre um dos escravos e o mestre falhe. Por exemplo, caso o nodo 2 (escravo) não consiga se conectar ao nodo 1 (mestre), o usuário 2 não conseguirá fazer reservas mesmo que ainda tenham quartos disponíveis, ou seja, haverá uma falha de disponibilidade [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Uma alternativa para aumentar a disponibilidade é permitir que todos os nodos possam continuar fazendo reservas mesmo que algum deles perca a conexão com os outros nodos. Dessa maneira, usuários em nodos diferentes desconectados poderão fazer a reserva do último quarto de hotel, podendo gerar alguma inconsistência de gravação. Contudo, dependendo da maneira como o hotel trabalha, esse cenário não é um problema. Muitos hotéis trabalham com *overbooking* para lidar com cancelamentos de reservas. Em contraste, alguns hotéis possuem quartos extras para lidar com imprevistos. Existe também a possibilidade de hotéis possuírem parcerias para transferência de clientes entre si. Alguns hotéis podem simplesmente cancelar a reserva caso achem que os custos de cancelamentos é menor do que perder reservas por falha de rede [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Outro cenário em que a janela de inconsistência não é um problema é o sistema de *blog*. Um usuário pode suportar não ver o comentário que acabou de fazer por alguns segundos ou minutos devido a uma falha de rede, por exemplo. Ou esse mesmo usuário não vai se aborrecer se o seu comentário aparecer mais de uma vez devido a uma inconsistência de gravação. Nesse caso a solução é simples: apagar o comentário repetido [Rodrigues et al. 2017, Sadalage and Fowler 2019].

O carrinho de compras de *sites* de *e-comerce* é um exemplo em que se exige alta disponibilidade. Os usuários esperam poder comprar sempre e rapidamente, ou seja, eles podem não tolerar indisponibilidade em seus carrinhos. Caso contrário, eles podem procurar outro *site*. Porém, a implementação de disponibilidade pode resultar em inconsistências de gravação. Por exemplo, a inconsistência pode gerar múltiplos carrinhos de compras. Nesse caso, para resolver o problema, o sistema pode identificar os vários carrinhos existentes e fazer uma operação de união entre seus itens, resultando em um único carrinho. Uma alternativa é permitir que o usuário analise os itens do carrinho antes de finalizar a compra [Rodrigues et al. 2017, Sadalage and Fowler 2019].

O balanceamento da inconsistência está ligada diretamente ao domínio do sistema. As decisões de como lidar com inconsistência pode ser mais associada ao conhecimento dos especialistas na área do sistema do que ao conhecimento dos desenvolvedores do sistema. Os exemplos dos sistemas de reserva de quartos de hotel, comentários em *blog* e carrinho de compras apresentados nessa seção podem permitir alguma inconsistência. Por outro lado, um sistema de transações financeiras pode não tolerar informações inconsistentes ou desatualizadas [Rodrigues et al. 2017, Sadalage and Fowler 2019].

3.4.5.3. Durabilidade

Os bancos de dados em disco utilizam um mecanismo de *buffer* para acessar registros na memória secundária. Quando é necessário acessar um registro no banco de dados, o gerenciador de *buffer* verifica se a página do banco de dados que contém esse registro está na memória. Caso contrário, ele deve copiar o bloco (ou blocos) do disco para a memória principal. Caso seja necessário, atualizações são copiadas da memória para o disco. Para evitar que dados sejam perdidos em caso de uma de falha de sistema (por exemplo, falta de energia), os SGBDs comumente utilizam um mecanismo de recuperação após falhas. Por limitações de espaço, a recuperação de bancos de dados após falhas de sistema não pode ser descrita nesse trabalho. Porém, esse tópico pode ser visto com detalhes no *survey* [Magalhães et al. 2021], no minicurso [Magalhães et al. 2023] e nos tutoriais [Magalhães et al. 2023b] e [Magalhães et al. 2023a].

Parece ser algo impensável desativar o mecanismo de recuperação após falhas de um SGBD, visto que ele garante a durabilidade das alterações no sistema. Contudo, esse mecanismo pode diminuir o desempenho do sistema, uma vez que ele deve fazer acessos ao disco, que é muito mais lento do que a memória. Assim, em alguns casos, relaxar a durabilidade (ou seja, desativar o mecanismo de recuperação) pode ser aceitável em troca de um melhor desempenho. Além disso, para melhorar ainda mais o desempenho, atualizações no banco de dados podem ser mantidas em memória e descarregadas apenas periodicamente para o disco. Obviamente, o preço dessa abordagem é a perda de atualizações em caso de falhas de sistema [Rodrigues et al. 2017, Sadalage and Fowler 2019].

Considere como exemplo um *website* que monitora e armazena informações de sessão de seus usuários. Suponha que o *website* possui um tráfego muito grande. Como consequência, o armazenamento de muitas informações de usuário pode prejudicar a responsividade do *site*. Para evitar esse problema, as informações do que os usuários estão fazendo no *site* são armazenadas temporariamente na memória e descarregadas periodicamente para um banco de dados. Essa abordagem melhora o desempenho do sistema, uma vez que diminui a quantidade de dados enviados para o disco. Porém, ela pode fazer o *site* perder alguns dados de sessão de usuário em caso de falha de sistema. Contudo, isso não seria um problema tão grande quanto uma lentidão no sistema [Rodrigues et al. 2017, Sadalage and Fowler 2019].

3.4.5.4. Map-reduce

Além da mudança na forma de armazenamento, os sistemas distribuídos também mudam a maneira de como processar os dados. Um vez que os dados estão espalhados em máquinas diferentes, pode ser necessário acessar várias máquinas para atender a uma requisição. Além disso, se existirem muitos dados armazenados em um *cluster*, é necessário processá-los de uma maneira eficiente para recuperá-los em um tempo viável. Um cenário ideal em um *cluster* é dividir o processamento entre as várias maquinas de forma paralela. Além disso, deve-se reduzir a quantidade de dados transferidos pela rede, processando tudo que for possível em cada

nodo [Davoudian et al. 2018, Magalhães et al. 2018].

MapReduce é um modelo de execução distribuída projetado para processar grandes volumes de dados em paralelo. A grande vantagem desse modelo é remover a complexidade da programação distribuída. Para isso, ele utiliza duas etapas de processamento: (i) *Map* e (ii) *Reduce*. Na etapa de *Mapping* (mapeamento), os dados são divididos em blocos e processados em tarefas menores em paralelo. Uma lógica de transformação pode ser aplicada a cada bloco de dados. Após a execução das tarefas, a fase *Reduce* (redução) agrega os dados [Davoudian et al. 2018, Magalhães et al. 2018].

Para exemplificar a ideia básica de MapReduce, considere o exemplo de agregado da Figura 3.7. Esse exemplo possui o agregado *Pedido*, em que cada pedido contem itens de produto. Esse agregado atende a requisições que precisem ler todos as informações de um pedido, incluindo seus itens. Porém, o agregado pode não ser "performático"em requisições que necessitem acessar mais de um pedido. Por exemplo, uma consulta que busque a receita total da venda de um produto no último trimestre. Essa busca irá precisar acessar vários agregados em vários nodos, possivelmente tornado a recuperação da informação muito lenta. Esse tipo de cenário demanda a utilização de MapReduce [Sadalage and Fowler 2019].

A etapa *map* é uma função que recebe como entrada um único agregado e retorna alguns pares chave-valor. O exemplo da Figura 3.14 atende a uma requisição que deseja saber a receita total de *notebooks* vendidos. O *mapping* recebe um agregado de pedido e retorna os itens correspondentes ao produto *notebook*. Essa função de *mapping* é aplicada a todos os agregados de pedido. Cada uma das aplicações da função *map* pode ser realizada de forma paralela, uma vez que elas são independentes [Sadalage and Fowler 2019].

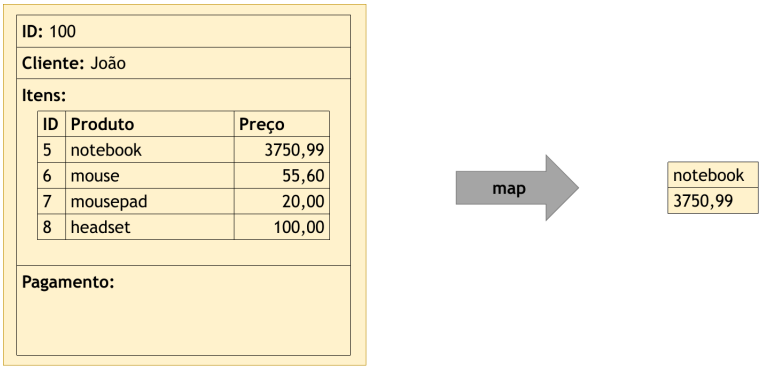


Figura 3.14. A função *map* lê agregados de pedido do banco de dados e produz pares chave-valor [Sadalage and Fowler 2019].

Após a etapa *map*, a função *reduce* recebe múltiplos resultado do mapeamento e combina seus valores. A Figura 3.15 representa a etapa *reduce* após a etapa *map* da Figura 3.14. A redução faz o somatório dos valores dos *notebooks* mapeados para calcular o valor da receita total [Sadalage and Fowler 2019]. Por questões de limitação de espaço e para fins didáticos, os exemplos apresentados possuem poucos dados, mas os agregados pode ser milhares ou milhões.

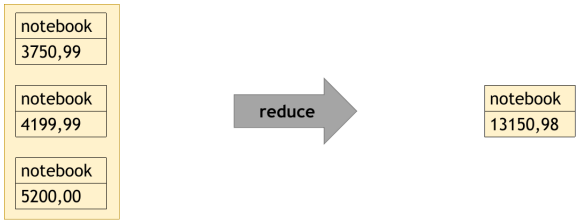


Figura 3.15. A função *reduce* agrega pares chave-valor [Sadalage and Fowler 2019].

3.5. Ajuste de desempenho em bancos de dados NoSQL

O projeto de banco de dados é essencial para o desenvolvimento de sistemas funcionais, confiáveis e seguros que possibilitem acessar e armazenar dados de forma eficiente. Contudo, a quantidade de dados e informações em uma empresa aumenta a cada dia. Além disso, suas necessidades podem mudar com o tempo. Assim, embora os SGBDs sejam projetados para suportar altas demandas de seus sistemas, o desempenho do sistema pode se tornar não aceitável com o passar do tempo [Mohammad 2016].

A fim de fornecer a continuidade do bom funcionamento do sistema e a qualidade dos serviços, é necessário o ajuste (*tuning*) do sistema de banco de dados. (*Tuning*) é um processo que envolve a configuração e ajuste de vários parâmetros e estruturas para otimizar o desempenho do SGBD. A falta de ajustes periódicos no SGBD podem trazer tempos de resposta lentos, processamento emperrado, dificuldades para conduzir operações corporativas que antes levavam menos tempo, etc [Mohammad 2016].

Essa seção discute as noções básicas de ajuste de desempenho apenas no banco de dados NoSQL MongoDB, por questões de limitação de espaço. A seção apresenta ferramentas, comandos e boas práticas que podem ajudar no desempenho do SGBD. Embora os comandos e ferramentas apresentados sejam específicos, a abordagem utilizada é genérica e pode ser facilmente reproduzida em outros SGBDs, inclusive em SGBDs relacionais.

3.5.1. Fatores que influenciam no desempenho

Para um bom desempenho das aplicações é necessário analisar o desempenho do sistema. A queda de desempenho de bancos de dados ocorre em função de vários fatores, tais como: falta de estratégias de acesso eficientes, pouca disponibilidade de *hardware*, número excessivo de conexões abertas e/ou projetos inadequados. Essa seção discute os principais tópicos a serem analisados em caso de queda do desempenho de um SGBD [MongoDB Documentation 2024].

3.5.1.1. Bloqueios

Bloqueios são utilizados para manter a consistência dos dados durante o processamento concorrente de transações. Por exemplo, se algum usuário desejar modificar determinado dado no banco, ele deve bloquear esse dado antes da modificação para

impedir que outros usuários sobrescrevam as alterações dele sobre o dado. É certo que operações longas de bloqueio podem gerar longas filas de espera por acesso a dados, podendo degradar drasticamente o desempenho de um sistema [MongoDB Documentation 2024].

Os bloqueios realizados no MongoDB podem ser verificados através do comando *serverStatus*. Adicionalmente, esse comando provê uma visão geral do estado do sistema com informações sobre o servidor, configurações de segurança, armazenamento, memória, conexões, rede, dentre outras. A Listagem 3.32 ilustra como utilizar o comando *serverStatus* [MongoDB Documentation 2024].

```
1  db.serverStatus()
2
```

Listagem 3.32. Comando para retornar informações gerais do estado do sistema.

Por meio do comando *serverStatus*, podem ser obtidas informações gerais e específicas de bloqueios através dos atributos *globalLock* e *locks*, respectivamente. A Figura 3.16 mostra um exemplo de informações obtidas por meio do comando *serverStatus* (Listagem 3.32) [MongoDB Documentation 2024].

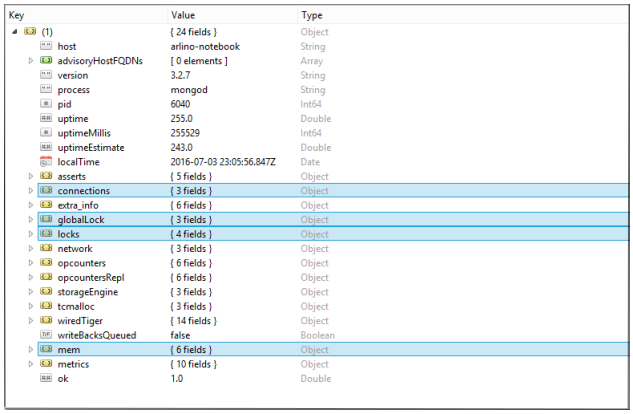


Figura 3.16. Informações gerais sobre o sistema obtidas através do comando *serverStatus*.

3.5.1.2. Memória

MongoDB possui um módulo de gerenciamento de memória que dado um conjunto de dados, o módulo irá alocar toda a memória necessária a esses dados no sistema. O comando *serverStatus* (Listagem 3.32) exibe em seu atributo *men* (Figura 3.16) a quantidade de memória mapeada e em uso no sistema, dentre outras informações [MongoDB Documentation 2024].

Se a quantidade de memória mapeada for maior que a quantidade de memória física disponível, poderão ocorrer *page faults*. Uma *page fault* acontece quando um dado que deve ser acessado não está localizado correntemente na memória física e, por esse motivo, é necessário fazer acesso ao disco. Muitas *page faults* podem prejudicar o desempenho do sistema devido ao aumento de acesso ao disco. O atributo *extra_info* da saída de *serverStatus* (Figura 3.16) possui informações sobre

as *page faults* ocorridas no MongoDB. Aumentar a quantidade de memória RAM pode ajudar a reduzir a frequência de *page faults*. Porém, se isso não for possível, a implementação de um *cluster* (ou adição de um novo nodo em um *cluster* já implantado) pode resolver o problema mediante o compartilhamento de dados e de recursos disponíveis [MongoDB Documentation 2024].

3.5.1.3. Número de conexões abertas

Em alguns casos, o número de conexões entre as aplicações e o SGBD pode prejudicar a habilidade do servidor em responder às requisições. Se, em dado momento, o número de requisições for muito grande, o sistema poderá ter problema em atendê-las. Nesse caso, é necessário aumentar os recursos do servidor. Um *cluster* de banco de dados com problemas em atender a muitas conexões pode adicionar novos nodos parar aumentar os recursos disponíveis. Se há muitas requisições de leitura, é necessário aumentar o número nodos réplicas para distribuir o processamento entre os membros do *cluster*. Por outro lado, se há muitas requisições de escrita, é necessário aumentar o número de nodos de compartilhamento para dividir a carga de trabalho entre as instâncias do banco [MongoDB Documentation 2024].

3.5.2. Avaliação de operações

Essa seção descreve estratégias de como monitorar e avaliar o desempenho de operações no MongoDB com o objetivo de identificar gargalos de desempenho no sistema [MongoDB Documentation 2024].

3.5.2.1. Coleta de operações lentas

A ferramenta *Database Profiler* pode ajudar a monitorar operações lentas no MongoDB. Essa ferramenta é capaz de identificar e armazenar operações lentas para poderem ser analisadas posteriormente. Ela possui níveis de configuração mostrados na Tabela 3.2. O nível de armazenamento de operações por meio da ferramenta *Database Profiler* pode ser ajustado através do comando *setProfilingLevel*, como ilustrado na Listagem 3.33. Nesse exemplo, o número 1 (um) indica que foi escolhido por armazenar apenas as operações lentas do sistema [MongoDB Documentation 2024].

Nível	Configuração
0	Não armazena nada.
1	Armazena apenas as operações lentas.
2	Armazena todas as operações.

Tabela 3.2. Níveis de configuração da ferramenta *Database Profiler*.

```
1    db.setProfilingLevel(1)
2
```

Listagem 3.33. Comando de configuração da ferramenta *Database Profiler*

Opcionalmente, o comando *setProfilingLevel* pode possuir um segundo parâmetro numérico para informar qual o tempo mínimo (em milissegundos) que *Database Profiler* deve considerar como necessário para uma operação ser lenta. Por padrão, uma operação é considerada lenta se executar em, pelo menos, 100ms. Após as operações lentas serem identificadas e armazenadas, o comando *system.profile.find* (Listagem 3.34) deve ser utilizado para recuperá-las. A Figura 3.17 exibe uma lista com operações armazenadas por *Database Profiler*. Nessa figura, o atributo das informações da segunda operação está expandido e é possível observar que ela se trata de uma consulta feita sobre a coleção chamada de *foo* do banco *test* [MongoDB Documentation 2024].

```
1    db.system.profile.find()
2
```

Listagem 3.34. Comando para recuperar as operações armazenadas por *Database Profiler*.

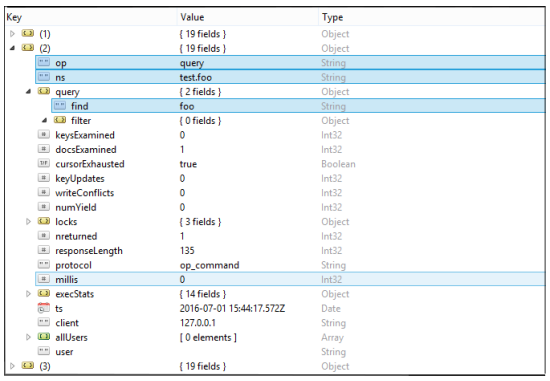


Figura 3.17. Lista de operações armazenadas por meio de *Database Profiler*.

A ferramenta *Database Profiler* deve ser utilizada com cuidado, pois pode afetar negativamente no desempenho do sistema. Por esse motivo, ela deve ser ativada apenas em intervalos de tempo estratégicos que não prejudiquem (ou prejudiquem minimamente) o sistema em produção [MongoDB Documentation 2024].

3.5.2.2. Operações em processo de execução

Em alguns casos, pode ser necessário monitorar as consultas enquanto estão sendo executadas. Para essa abordagem, MongoDB possui a função *currentOp*. Esse comando retorna as operações que estão em processo de execução no sistema. Por padrão, *currentOp* retorna apenas as operações importantes correntemente em execução. Opcionalmente, esse comando pode ter os parâmetros exibidos na Tabela 3.3 que o fazem retornar conjuntos de operações diferentes [MongoDB Documentation 2024].

A Figura 3.35 exibe o comando *currentOp* com o parâmetro *true* que permite mostrar todas as operações em execução no MongoDB. A Figura 3.18 ilustra um resultado obtido por *currentOp*, no qual a operação 6 está expandida exibindo informações sobre uma operação de conexão chamada de *conn3* cujo atributo *active* está

com o valor *false*, indicando que a conexão está ociosa [MongoDB Documentation 2024].

Parâmetro	Retorno
true	Todas as operações, incluindo as de sistema e conexões ociosas.
registro BSON	Conjunto de operações especificadas no registro.

Tabela 3.3. Parâmetros do comando *currentOp*.

```
db.currentOp(true)

```

Listagem 3.35. Comando para recuperar todas as operações em execução no MongoDB.

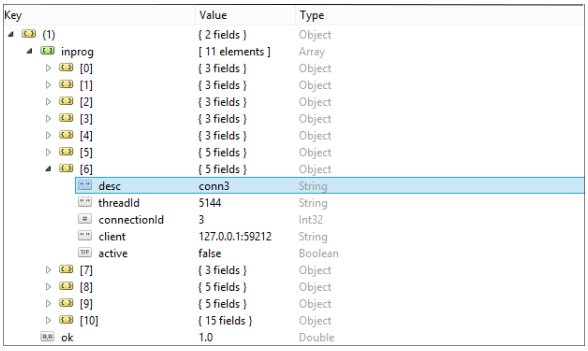


Figura 3.18. Lista de todas as operações em execução no MongoDB.

O comando *currentOP* ainda permite utilizar um BSON como parâmetro para restringir sua busca por operações específicas, tais como operações esperando por um bloqueio, ativas sem rendimento, ativas em um banco de dados específico, que utilizam índices, dentre outras. O exemplo do comando da Listagem 3.36 exibirá apenas as operações de escrita em execução que estão esperando por um bloqueio [MongoDB Documentation 2024].

```
db.currentOp({
  "waitingForLock": true,
  $or: [
  {"op": {"$in": ["insert", "update", "remove"] }},
  {"query.findandmodify": {$exists: true}}
  ]
})

```

Listagem 3.36. Comando que retorna as operações em execução no MongoDB que estão esperando por um bloqueio.

3.5.2.3. Plano de execução de consulta

Uma operação de banco de dados é formada por um conjunto de etapas para alcançar os resultados esperados. Esse conjunto de etapas é chamado de plano de

execução. Para uma determinada operação, podem haver vários planos de execução. O otimizador de consultas do SGBD avalia diferentes planos de execução e escolhe o que considera mais eficiente [MongoDB Documentation 2024].

O comando *explain* é capaz de retornar informações sobre o plano de execução de uma consulta selecionada pelo otimizador do MongoDB. Além disso, *explain* pode retornar informações estatísticas sobre o plano de execução da consulta. As informações retornadas por esse comando dependem dos parâmetros passados, descritos na Tabela 3.4. O parâmetro *queryPlanner* é o padrão de execução de *explain* [MongoDB Documentation 2024].

Tabela 3.4. Modos de execução do comando *explain*.

Parâmetro	Retorno
queryPlanner	Informações sobre o plano de execução escolhido.
executionStats	Informações sobre o plano de execução escolhido, incluindo suas informações estatísticas.
allPlansExecution	Informações sobre o plano de execução escolhido e informações estatísticas desse plano e de outros planos de execução encontrados pelo otimizador no processo de seleção.

Para exemplificar o uso do comando explain, considere a consulta C1 da Listagem 3.37. Essa consulta retorna todas a tuplas da coleção *foo*. A Listagem 3.38 mostra como utilizar *explain* para retornar o plano de execução escolhido para a consulta C1. A Figura 3.19 exibe o plano de execução da consulta C1 gerado por *explain*. Nesse plano, a estratégia utilizada para retornar as tuplas de C1 foi *CollScan* (percorrer a coleção inteira) [MongoDB Documentation 2024].

```
1    db.foo.find()
2
```

Listagem 3.37. Consulta C1.

```
1    db.foo.find().explain()
2
```

Listagem 3.38. Comando para retornar o plano de execução da consulta C1.

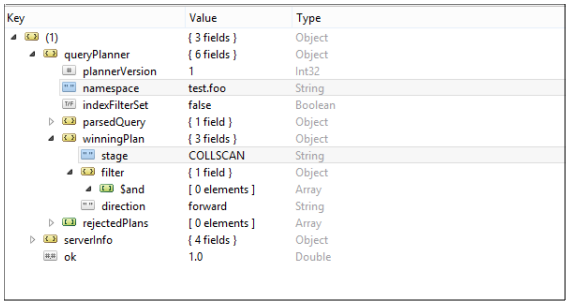


Figura 3.19. Plano de execução da consulta C1.

3.5.3. Otimização de desempenho de consultas

3.5.3.1. Criação de índices

Índices são estruturas que utilizam estratégias de acesso rápido aos dados. Consultas lentas podem ter um melhor desempenho se índices forem criados para seus atributos, visto que percorrer índices pode ser mais rápido do que percorrer uma coleção de dados. A criação de índices deve ser muito bem planejada. Apesar de um índice melhorar o desempenho de uma determinada consulta, ele pode prejudicar o desempenho da execução de outras operações. Como consequência, o desempenho geral do sistema pode degradar [MongoDB Documentation 2024].

A Listagem 3.39 exemplifica como criar um índice para o campo *author_name* da coleção *posts*. Nesse caso, buscas pelo nome de autores na coleção *posts* podem melhorar o desempenho. Entretanto, operações de atualização no atributo *author_name* devem ficar mais lentas devido ao trabalho adicional de atualização na estrutura de índice. Índices são mais adequados a um sistema que faça muito mais consultas do que atualizações [MongoDB Documentation 2024].

```
1     db.posts.createIndex({author\_name: 1})
2
```

Listagem 3.39. Criação de índice em uma coleção no MongoDB.

3.5.3.2. Limite do número de resultados de uma consulta

Limitar o número de resultados de uma consulta é uma boa estratégia para reduzir a quantidade de dados trafegados pela rede, pois retornar todos os dados de uma consulta pode ser desnecessário. Às vezes, o usuário pode não precisar utilizar todas as linhas retornadas por uma consulta. Além disso, carregar todas as linhas de uma consulta de uma só vez pode prejudicar a visualização dos resultados. No MongoDB, o número de resultados de uma consulta pode ser limitado utilizando a função *limit* cujo funcionamento é similar ao do comando *Limit* no SQL. A função *limit* recebe como parâmetro o número de linhas a serem retornadas. Na Figura 3.40 há um exemplo de como utilizar *limit* para uma consulta retornar apenas os 10 primeiros elementos [MongoDB Documentation 2024].

```
1     db.posts.find().limit(10)
2
```

Listagem 3.40. Comando *limit* aplicado a uma consulta no MongoDB.

Uma alternativa para reduzir o tráfego de dados na rede é o uso de projeções. Uma projeção permite que apenas um subconjunto dos campos de uma coleção seja retornado por uma consulta. O uso de projeções pode trazer ganhos de desempenho a uma consulta, visto que apenas os campos necessários são retornados, diminuindo a quantidade de dados recuperados. Por exemplo, na consulta da Listagem 3.41, ao invés de retornar a coleção *posts* inteira, apenas os campos *timestamp*, *title*, *author*, e *abstract* são recuperados da coleção [MongoDB Documentation 2024].

```
db.posts.find(
{
  timestamp: 1 ,
  title: 1 ,
  author: 1 ,
  abstract: 1
}
).sort({timestamp: -1})

```

Listagem 3.41. Consulta com uso de projeção.

3.5.3.3. Uso de *hints* para selecionar um determinado índice

O otimizador do SGBD tenta selecionar o melhor plano de execução para uma consulta. Por exemplo, o otimizador pode escolher um plano de execução que usa os índices mais adequados à execução da consulta. Porém, nem sempre o otimizador consegue fazer isso. Nesses casos, uma alternativa é forçar o otimizador a selecionar um determinado plano de execução por meio de *hints* (dicas). *Hints* pode ser fornecidos pelos desenvolvedores nas consultas, a fim de melhorar o desempenho delas [MongoDB Documentation 2024].

Como exemplo, a consulta da Listagem 3.42 possui uma *hint* que faz a consulta utilizar o índice do campo *age* da coleção *users*. Alternativamente, uma *hint* pode especificar o nome do índice a ser utilizado na execução de uma consulta. A Listagem 3.43 ilustra uma *hint* que força a consulta sobre a coleção *users* utilizar o índice *age_1* [MongoDB Documentation 2024].

```
db.users.find().hint({ age: 1 })

```

Listagem 3.42. *Hint* que força a consulta a utilizar o índice de um determinado campo de uma coleção.

```
db.users.find().hint("age\_1")

```

Listagem 3.43. *Hint* que força a consulta a utilizar um determinado índice.

3.6. Conclusões

Esse minicurso elucidou as principais questões relacionadas a sistemas de bancos de dados NoSQL, principalmente as relacionadas ao projeto e ajuste de desempenho. Para isso, o trabalho provê uma visão geral da tecnologia NoSQL, como produtos, linguagens de acesso, manipulação e processamento dos dados. Em seguida foram discutidos os principais tópicos a se pensar em um projeto de sistema NoSQL: agregados, relacionamentos, distribuição de dados, consistência e durabilidade e, por fim, execução distribuída de grandes quantidades de dados. O minicurso ainda discutiu noções básicas de ajuste de desempenho em bancos de dados NoSQL, mostrando ferramentas, comandos e boas práticas que podem ajudar no desempenho.

Referências

[Ambler 2000] Ambler, S. W. (2000). Mapping objects to relational databases. *White Paper, Ronin International*.

[Benymol and Sajimon 2017] Benymol, J. and Sajimon, A. (2017). Exploring the merits of nosql: A study based on mongodb. *International Conference on Networks & Advances in Computational Technologies (NetACT)*, pages 20–22.

[Berman and Garay 1993] Berman, P. and Garay, J. A. (1993). Cloture votes: n/4-resilient distributed consensus in t+ 1 rounds. *Mathematical systems theory*, 26:3–19.

[Bernstein and Goodman 1981] Bernstein, P. A. and Goodman, N. (1981). Concurrency control in distributed database systems. *ACM Comput. Surv.*, 13(2):185–221.

[Bernstein et al. 1987] Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

[Bondi 2000] Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Second International Workshop on Software and Performance, WOSP 2000, Ottawa, Canada, September 17-20, 2000*, pages 195–203. ACM.

[Boscarioli et al. 2006] Boscarioli, C., Bezerra, A., BENEDICTO, M. d., and Delmiro, G. (2006). Uma reflexão sobre banco de dados orientados a objetos. In *Congresso de Tecnologias para Gestão de Dados e Metadados do Cone Sul, Paraná, Brasil*. sn.

[Boudaoud et al. 2022] Boudaoud, A., Mahfoud, H., and Chikh, A. (2022). Towards a complete direct mapping from relational databases to property graphs. In Fournier-Viger, P., Yousef, A. H., and Bellatreche, L., editors, *Model and Data Engineering: 11th International Conference, MEDI 2022, Cairo, Egypt, November 21-24, 2022, Proceedings*, volume 13761 of *Lecture Notes in Computer Science*, pages 222–235. Springer.

[Burrows 2006] Burrows, M. (2006). The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350.

[Cassandra Database 2024] Cassandra Database (2024). Open Source Nosql Database. Disponível em: "https://cassandra.apache.org/". Acessado em: 01/05/2024.

[Cassandra Documentation 2024] Cassandra Documentation (2024). Welcome to apache cassandras documentation! Disponível em: "https://cassandra.apache.org/doc/latest/". Acessado em: 01/05/2024.

[Chandra et al. 2007] Chandra, T. D., Griesemer, R., and Redstone, J. (2007). Paxos made live: an engineering perspective. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 398–407.

[Davoudian et al. 2018] Davoudian, A., Chen, L., and Liu, M. (2018). A survey on nosql stores. *ACM Comput. Surv.*, 51(2):40:1–40:43.

[Elmasri and Navathe 2000] Elmasri, R. and Navathe, S. B. (2000). *Fundamentals of Database Systems, 3rd Edition*. Addison-Wesley-Longman.

[Faerber et al. 2017] Faerber, F., Kemper, A., Larson, P., Levandoski, J. J., Neumann, T., and Pavlo, A. (2017). Main memory database systems. *Foundations and Trends in Databases*, 8(1-2):1–130.

[Frozza et al. 2022] Frozza, A. A., Schreiner, G. A., and dos Santos Mello, R. (2022). Projeto de bancos de dados nosql. *37th Brazilian Symposium on Data Bases*.

[Gray et al. 1975] Gray, J., Lorie, R. A., Putzolu, G. R., and Traiger, I. L. (1975). Granularity of locks in a large shared data base. In Kerr, D. S., editor, *Proceedings of the International Conference on Very Large Data Bases, September 22-24, 1975, Framingham, Massachusetts, USA*, pages 428–451. ACM.

[Gray and Reuter 1993] Gray, J. and Reuter, A. (1993). *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann.

[Gueidi et al. 2021] Gueidi, A., Gharsellaoui, H., and Ahmed, S. B. (2021). Towards unified modeling for nosql solution based on mapping approach. In Watróbski, J., Salabun, W., Toro, C., Zanni-Merk, C., Howlett, R. J., and Jain, L. C., editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES-2021, Virtual Event / Szczecin, Poland, 8-10 September 2021*, volume 192 of *Procedia Computer Science*, pages 3637–3646. Elsevier.

[Härder and Reuter 1983] Härder, T. and Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317.

[Harrison 2015] Harrison, G. (2015). *Next Generation Databases: NoSQLand Big Data*. Apress Berkeley, CA.

[Heuser 2009] Heuser, C. A. (2009). *Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS*. Bookman Editora.

[Hill 1990] Hill, M. D. (1990). What is scalability? *SIGARCH Comput. Archit. News*, 18(4):18–21.

[Imielinski and Jr. 1984] Imielinski, T. and Jr., W. L. (1984). The relational model of data and cylindric algebras. *J. Comput. Syst. Sci.*, 28(1):80–102.

[Lazoti 2016] Lazoti, R. (2016). *Armazenando dados com Redis*. Casa do Código.

[Magalhães et al. 2023] Magalhães, A., Brayner, Â., and Monteiro, J. M. (2023). Bancos de dados em memória e suas estratégias de recuperação após falhas. *Sociedade Brasileira de Computação*.

[Magalhães et al. 2023a] Magalhães, A., Brayner, A., and Monteiro, J. M. (2023a). Main memory database instant recovery. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 255–269. SBC.

[Magalhães et al. 2023b] Magalhães, A., Brayner, A., and Monteiro, J. M. (2023b). Main memory database recovery strategies. In Das, S., Pandis, I., Candan, K. S., and Amer-Yahia, S., editors, *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 31–35. ACM.

[Magalhães et al. 2018] Magalhães, A., Monteiro, J. M., and Brayner, A. (2018). Gerenciamento e processamento de big data com bancos de dados em memória. In *I Jornada latino-americana de atualização em informática , JOLAI 2017, São Paulo, SP, Brazil, 2018*.

[Magalhães et al. 2021] Magalhães, A., Monteiro, J. M., and Brayner, A. (2021). Main memory database recovery: A survey. *ACM Comput. Surv.*, 54(2):46:1–46:36.

[Marquesone 2016] Marquesone, R. (2016). *Big Data: Técnicas e tecnologias para extração de valor dos dados*. Editora Casa do Código.

[Michael et al. 2007] Michael, M. M., Moreira, J. E., Shiloach, D., and Wisniewski, R. W. (2007). Scale-up x scale-out: A case study using nutch/lucene. In *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–8. IEEE.

[Mohammad 2016] Mohammad, S. (2016). *Self-tuning for cloud database clusters*. PhD thesis, University of Magdeburg, Germany.

[MongoDB Database 2024] MongoDB Database (2024). MongoDB: The Developer Data Platform | MongoDB. Disponível em: "https://www.mongodb.com". Acessado em: 01/05/2024.

[MongoDB Documentation 2024] MongoDB Documentation (2024). Mongodb documentation. Disponível em: "https://www.mongodb.com/docs/". Acessado em: 01/05/2024.

[Neo4J Database 2024] Neo4J Database (2024). Neo4j Graph Database & Analytics. Disponível em: "https://neo4j.com". Acessado em: 01/05/2024.

[Neo4J Documentation 2024] Neo4J Documentation (2024). Neo4j documentation. Disponível em: "https://neo4j.com/docs/". Acessado em: 01/05/2024.

[Özsu and Valduriez 1996] Özsu, M. T. and Valduriez, P. (1996). Distributed and parallel database systems. *ACM Comput. Surv.*, 28(1):125–128.

[Paniz 2016] Paniz, D. (2016). *NoSQL: Como armazenar os dados de uma aplicação moderna*. Casa do Código.

[Passos et al. 2023] Passos, P. V., de Oliveira, L. S., Schreiner, G. A., Machado, V. L., and dos Santos Mello, R. (2023). Sql2neo: Uma interface de acesso SQL para o neo4j. In *Proceedings of the 38th Brazilian Symposium on Databases, SBBD 2023, Belo Horizonte, MG, Brazil, September 25-29, 2023*, pages 179–191. SBC.

[Phillips et al. 2015] Phillips, D. J., McGlaughlin, A., Ruth, D., Jager, L. R., and Soldan, A. (2015). Graph theoretic analysis of structural connectivity across the spectrum of Alzheimer's disease: The importance of graph creation methods. *NeuroImage: Clinical*, 7:377–390.

[Ramakrishnan and Gehrke 2003] Ramakrishnan, R. and Gehrke, J. (2003). *Database management systems (3. ed.)*. McGraw-Hill.

[Redis Database 2024] Redis Database (2024). Redis - the real-time data platform. Disponível em: "https://redis.io". Acessado em: 01/05/2024.

[Redis Documentation 2024] Redis Documentation (2024). Redis documentation! Disponível em: "https://redis.io/docs/latest/". Acessado em: 01/05/2024.

[Redis Insight 2024] Redis Insight (2024). Redis insight the best redis gui. Disponível em: "https://studio3t.com". Acessado em: 01/05/2024.

[Rodrigues et al. 2017] Rodrigues, W. B. et al. (2017). Nosql: a análise da modelagem e consistência dos dados na era do big data.

[Sadalage and Fowler 2019] Sadalage, P. J. and Fowler, M. (2019). *NoSQL essencial: um guia conciso para o mundo emergente da persistência poliglota*. Novatec Editora.

[Silva et al. 2021] Silva, L. F. C., Riva, A. D., and Rosa, G. A. (2021). *Banco de Dados Não Relacional*. Minha Biblioteca.

[Strauch et al. 2011] Strauch, C., Sites, U.-L. S., and Kriha, W. (2011). Nosql databases. *Lecture Notes, Stuttgart Media University*, 20(24):79.

[Studio 3T 2024] Studio 3T (2024). Studio 3T, the Ultimate GUI for MongoDB. Disponível em: "https://studio3t.com". Acessado em: 01/05/2024.

[Taft et al. 2014] Taft, R., Mansour, E., Serafini, M., Duggan, J., Elmore, A. J., Aboulnaga, A., Pavlo, A., and Stonebraker, M. (2014). E-store: Fine-grained elastic partitioning for distributed transaction processing. *Proceedings of the VLDB Endowment*, 8(3):245–256.

[Vieira et al. 2012] Vieira, M. R., FIGUEIREDO, J. M. d., Liberatti, G., and Viebrantz, A. F. M. (2012). Bancos de dados nosql: conceitos, ferramentas, linguagens e estudos de casos no contexto de big data. *Simpósio Brasileiro de Bancos de Dados*, 27:1–30.

[VolDB Documentation 2020] VolDB Documentation (2020). Voldb documentation. Disponível em: "https://docs.voltdb.com". Acessado em: 01/05/2024.

Capítulo 4

Análise de Dados Privada em Redes Sociais

André L. C. Mendonça, Felipe T. Brito, Javam C. Machado

Abstract

With the increasing use of social networks, analyzing user interactions has become crucial for understanding social dynamics and discovering valuable insights under a variety of domains. However, in the context of social network analytics, concerns regarding individuals' privacy persist, requiring measures to protect sensitive information. In recent years, differential privacy has become the de facto standard for privacy-preserving data analysis under strong mathematical guarantees based on the concept of indistinguishability. This chapter provides a comprehensive overview of existing differentially private methods and techniques to protect sensitive information while enabling meaningful social network analysis. We explore the principles of differential privacy, highlighting its mechanisms for adding noise to data to prevent individual re-identification. Additionally, we investigate the strategies for applying differential privacy in social network analytics, encompassing data publishing, graph analysis, and machine learning tasks privately.

Resumo

Com o uso crescente das redes sociais, a análise das interações dos usuários tornou-se crucial para compreender as dinâmicas sociais e descobrir informações valiosas em diversos domínios. No entanto, no contexto de análise de dados em redes sociais, existem preocupações inerentes à privacidade dos indivíduos, as quais exigem medidas para proteger informações sensíveis. Nos últimos anos, a privacidade diferencial tornou-se o padrão para prover privacidade na análise de dados sob fortes garantias matemáticas baseada no princípio da indistinguibilidade. Este capítulo fornece uma visão geral dos métodos e técnicas diferencialmente privadas para proteger informações sensíveis e, simultaneamente, permitir análises relevantes de redes sociais. Exploramos os princípios da privacidade diferencial, destacando seus mecanismos para adicionar ruído aos dados para evitar a reidentificação dos indivíduos. Além disso, investigamos as estratégias para aplicar privacidade diferencial na análise de dados em redes sociais, abrangendo a publicação de dados, a análise de grafos e tarefas de aprendizado de máquina de maneira privada.

4.1. Introdução

Redes sociais são plataformas *online* que possibilitam a interação entre pessoas por meio do compartilhamento de informações, ideias, interesses e outros tipos de conteúdo. Essas plataformas, que incluem exemplos amplamente conhecidos como Facebook, Twitter, Instagram e LinkedIn, funcionam como sistemas que conectam usuários com base em suas relações pessoais, profissionais ou interesses comuns. A evolução das redes sociais começou no final dos anos 90 e início dos anos 2000 com sites como Six Degrees e Friendster [Boyd and Ellison 2007], que permitiam aos usuários criar perfis e conectar-se com amigos. Com o lançamento do MySpace e, posteriormente, do Facebook, as redes sociais começaram a se expandir rapidamente, oferecendo recursos mais avançados como compartilhamento de fotos, vídeos e eventos, além de mecanismos de comunicação direta, como mensagens instantâneas e comentários.

Hoje, as redes sociais desempenham um papel central na vida moderna, influenciando a maneira como as pessoas se comunicam, consomem informações e realizam negócios. Elas oferecem um espaço virtual onde indivíduos podem expressar suas opiniões, manter contato com amigos e familiares, seguir notícias e tendências, e até mesmo buscar oportunidades de emprego. Empresas e organizações também utilizam essas plataformas para marketing, atendimento ao cliente e construção de marca. Além disso, as redes sociais têm um impacto significativo na formação de comunidades e movimentos sociais. Elas facilitam a organização e mobilização de grupos em torno de causas comuns, permitindo que indivíduos compartilhem suas experiências e apoiem uns aos outros de maneira rápida e eficaz [Prell 2011].

Como reflexo da vida social real, as redes sociais fornecem um meio para compartilhar muitas informações privadas e sensíveis. Por exemplo, em muitas redes sociais online, é comum que os usuários sejam solicitados a fornecer informações pessoais, como nome, gênero, data de nascimento, nível de educação, estado civil, foto pessoal e até mesmo número de telefone celular. Além disso, conteúdos gerados pelos usuários, como textos, fotos, vídeos e localizações geográficas, também são armazenados em bancos de dados [Baden et al. 2009]. Esses dados podem ser compartilhados com terceiros para serviços comerciais adicionais, como análise de dados, publicidade direcionada, recomendações e avaliações de aplicativos. Contudo, uma vez que dados de redes sociais contêm informações sensíveis, compartilhar esse tipo de dado sem garantias suficientes de privacidade pode comprometer seriamente a privacidade dos indivíduos [Brito et al. 2024]. As leis e regulamentações atuais sobre privacidade de dados, como o *General Data Protection Regulation (GDPR)* [European Commission 2018] e a *Lei Geral de Proteção de Dados Pessoais (LGPD)* [Brazil 2018], exigem que os indivíduos não possam ser reidentificados a partir das informações divulgadas.

Se informações pessoais privadas forem acessadas por terceiros que, teoricamente, não deveriam ter acesso a elas, os indivíduos afetados podem enfrentar várias consequências negativas. Isso inclui a exposição a ataques de intrusão, como *spam*, mensagens indesejadas e, em casos mais graves, a divulgação não autorizada de dados pode levar a danos à reputação pessoal. Além disso, a violação de privacidade pode também resultar em roubo de identidade, onde os dados pessoais são utilizados de forma fraudulenta para obter benefícios financeiros ou cometer crimes [Abdulhamid et al. 2014].

O problema da proteção da privacidade dos dados foi inicialmente proposto na década de 1970 [Dalenius 1977], que apontou que o objetivo de proteger informações privadas em uma base de dados é evitar que qualquer usuário, incluindo usuários legítimos e possíveis atacantes, obtenha informações precisas sobre indivíduos específicos ao acessar a base de dados. Com o passar dos anos, essa preocupação evoluiu significativamente, levando ao desenvolvimento de várias técnicas e metodologias para proteger a privacidade dos dados, como k-anonimato [Sweeney 2002], l-diversidade [Machanavajjhala et al. 2007], t-proximidade [Li et al. 2006], δ-presença [Nergiz et al. 2007], dentre outros. Contudo, cada um desses modelos oferece proteção apenas contra um tipo específico de ataque e não consegue se defender contra novos tipos de ataques que venham a ser desenvolvidos. A causa fundamental dessa deficiência reside no fato de que a efetividade de um modelo de preservação da privacidade depende da suposição de um conhecimento prévio específico por parte de um atacante, também denominado adversário. No entanto, é praticamente impossível enumerar todos os tipos possíveis de conhecimento prévio que um atacante pode obter. Portanto, é altamente desejável um modelo de preservação da privacidade que seja independente do conhecimento prévio do atacante.

A privacidade diferencial (PD) [Dwork 2006] surgiu como a noção padrão de privacidade, em vez de uma única ferramenta, para o compartilhamento de dados de maneira privada. Ela tem sido utilizada na indústria [Kenthapadi et al. 2019] por empresas como Apple, Google e Uber [Cormode et al. 2018], e também no setor público por agências dos EUA, como o U.S. Census Bureau [Garfinkel et al. 2018]. A privacidade diferencial assume que um atacante pode obter todas as informações de um conjunto de dados, exceto o registro alvo, o que pode ser considerado o conhecimento máximo que um atacante pode ter. Sob o conceito de privacidade diferencial, os resultados de análises realizadas em um conjunto de dados são insensíveis à alteração de um único registro, ou seja, a presença, ou ausência, de um único registro no conjunto de dados tem pouco efeito sobre a distribuição de saída das análises (consultas) realizadas. Em outras palavras, um atacante não pode obter informações precisas sobre um indivíduo ao observar os resultados das análises realizadas sobre um conjunto de dados diferencialmente privado, pois o risco de divulgação de privacidade causado pela adição, ou exclusão, de um único registro é mantido dentro de um intervalo aceitável especificado pelo usuário.

Existem diversos métodos e implementações para realizar análises de dados com privacidade diferencial. Embora tais abordagens tenham sido inicialmente projetadas para dados tabulares, a privacidade diferencial também pode ser aplicada à análise de dados em redes sociais [Silva et al. 2017, Jiang et al. 2021]. Em termos gerais, as redes sociais podem ser modeladas como grafos e se tornam extremamente complexas em larga escala. Dessa forma, existem desafios fundamentais que precisam ser enfrentados para aplicar a privacidade diferencial na análise de dados de redes sociais. Primeiro, é necessário estudar a privacidade diferencial de dados tabulares no contexto de dados de rede. Em seguida, é preciso abordar a questão da alta sensibilidade em dados de redes sociais complexas, uma vez que a presença de relações interdependentes entre elementos da rede pode amplificar o impacto das alterações de dados individuais, dificultando a proteção da privacidade. Por fim, é fundamental explorar o equilíbrio entre a utilidade dos dados para a análise e a garantia de privacidade, pois adicionar muito ruído para garantir a privacidade diferencial pode tornar os resultados das análises inúteis.

Este capítulo apresenta uma visão geral dos métodos e técnicas de privacidade diferencial destinados a proteger informações sensíveis, permitindo, simultaneamente, análises de dados relevantes em redes sociais. A Seção 4.2 apresenta os conceitos fundamentais sobre a análise de dados em redes sociais, enquanto a Seção 4.3 introduz problemas inerentes à privacidade dos indivíduos, decorrentes de análises indevidas sobre os dados. Em seguida, o modelo de privacidade diferencial é detalhado na Seção 4.4, com destaque para suas principais propriedades e configurações. Já a Seção 4.5 destaca as principais variantes do modelo de privacidade diferencial para o contexto de redes sociais, tais como *node*-PD, *edge*-PD, *edge-weight* PD e *attributed*-PD. Identificamos os principais tipos de análises realizadas sobre redes sociais, juntamente com as diversas técnicas de privacidade diferencial existentes para a realização dessas análises de maneira privada na Seção 4.6. Por fim, a Seção 4.7 explora as perspectivas futuras dos tópicos abordados neste capítulo, destacando as principais dificuldades a serem superadas.

4.2. Fundamentos de Análises de Dados em Redes Sociais

Na era digital, dados têm se transformado em ativos importantes para as organizações, principalmente devido ao seu elevado valor e importância. Atualmente, grandes volumes de dados, de diversas naturezas, encontram-se disponíveis e tornam-se grandes aliados estratégicos de empresas em seus processos de tomada de decisão a partir da análises realizadas. De maneira sucinta, a análise de dados consiste no processo de inspecionar, tratar, transformar e modelar os dados com o objetivo de descobrir informações úteis, *insights* e conclusões que auxiliem na tomada de decisão de empresas e organizações. A análise de dados dispõe de múltiplas técnicas e abordagens, abrangendo diferentes domínios [Mendonça et al. 2023]. A mineração de dados é uma das técnicas mais conhecidas utilizada para a análise de dados. Ela faz parte do grupo de análise preditiva, tendo como foco principal a modelagem estatística e a descoberta de conhecimento para fins preditivos. Juntamente com a análise preditiva, as análises prescritiva, descritiva e diagnóstica compõem os tipos de análise de dados existentes [Cook and Holder 2006].

Comumente, as análises de dados ocorrem sobre dados tabulares, representados por meio de registros, o que limita as análises sobre estruturas de dados mais complexas, como as redes sociais. Diferentemente das análises sobre dados tabulares, as análises sobre redes sociais priorizam os relacionamentos entre os indivíduos que compõem as redes e seus respectivos relacionamentos, ou conexões. Redes sociais são estruturas mais complexas que, intuitivamente, são modeladas a partir de estruturas em grafos. Em resumo, um grafo consiste em uma estrutura de dados formada, originalmente, por nós e arestas, onde os nós representam as entidades (indivíduos) e as arestas os relacionamentos entre nós. Vale ressaltar que neste capítulo, as análises de dados em redes sociais são conduzidas por meio de estruturas de grafos. Dessa forma, uma rede social é definida como um grafo $G = (V, E)$, onde V é o conjunto de vértices (nós) e E é o conjunto de arestas. Devido às características inerentes às redes sociais, diversas estatísticas podem ser extraídas de análises sobre grafos. As estatísticas mais comuns incluem os graus dos nós, juntamente com suas respectivas distribuição de graus, métricas de centralidade e outras medidas pertinentes. Adicionalmente, contagens de subgrafos, bem como diversas métricas de distância, também são exemplos de métricas frequentemente examinadas em análise de grafos.

4.2.1. Contagem de subgrafos

A análise de subgrafos desempenha um papel crucial na compreensão das redes sociais, proporcionando uma visão da estrutura e dos padrões subjacentes às interações entre os nós [Ribeiro et al. 2021]. Dentre os diferentes tipos de subgrafos, destacam-se os triângulos, estrelas e cliques, cada um fornecendo informações relevantes sobre a conectividade e a estrutura da rede. A Figura 4.1 exemplifica quatro tipos diferentes de subgrafos que podem estar presentes em uma rede social: triângulos, *k*-estrelas, *k*-cliques e *k*-triângulos.

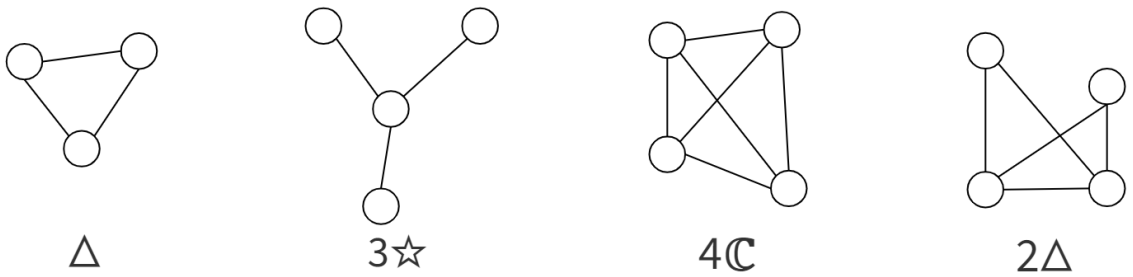


Figura 4.1: Exemplos de subgrafos que podem estar presentes em uma rede social: triângulos, estrelas de tamanho 3, cliques de tamanho 4 e triângulos de tamanho 2.

Os triângulos são subgrafos compostos por três nós interconectados, formando uma estrutura triangular. A contagem de triângulos em uma rede social é importante para identificar a presença de agrupamentos densamente conectados de três indivíduos. A ocorrência de triângulos indica relações de proximidade local, o que pode sugerir a existência de comunidades dentro da rede.

Por outro lado, uma *k*-estrela consiste em um nó central conectado a *k* nós periféricos, formando uma estrutura semelhante a uma estrela. Este conceito é particularmente valioso para identificar e caracterizar os nós mais influentes e os padrões de interação dentro de uma rede. Além disso, a distribuição de *k*-estrelas ajuda a compreender a dinâmica das redes sociais. Em redes de comunicação, por exemplo, um nó central com muitas conexões pode atuar como um *hub* de comunicação, onde a informação é agregada e distribuída.

Um *k*-clique é um subgrafo completo composto por *k* nós, onde cada par de nós está diretamente conectado por uma aresta. A contagem de *k*-cliques é essencial para identificar grupos coesos e altamente conectados na rede social, como grupos de amigos próximos ou equipes de trabalho colaborativo. A existência de *k*-cliques sugere que os membros desses subgrafos compartilham interesses comuns ou têm uma alta frequência de comunicação.

Os *k*-triângulos são uma extensão do conceito de triângulos em grafos. Em particular, esse subgrafo é formado por um conjunto de *k* triângulos que compartilham um vértice em comum. Esta estrutura é utilizada para identificar áreas de alta interconectividade em uma rede, onde múltiplas relações convergem em um único nó. A análise de *k*-triângulos é particularmente útil para identificar nós centrais ou altamente influentes na rede. Em redes de colaboração científica, por exemplo, um nó central com muitos *k*-triângulos pode representar um pesquisador que colabora com diversos grupos distintos, servindo como um ponto de interseção entre diferentes comunidades de pesquisa.

4.2.2. Histogramas

Os histogramas são representações gráficas que ilustram a distribuição de um conjunto de dados. Compostos por barras retangulares, cada barra representa a frequência de dados dentro de intervalos específicos, chamados *bins*. A altura das barras corresponde ao número de ocorrências dos dados em cada intervalo. Ao representar visualmente a distribuição de certas características de um grafo, os histogramas permitem uma compreensão clara de como a conectividade é estruturada dentro da rede. Eles ajudam a identificar padrões recorrentes e tendências que podem ser difíceis de perceber de outra maneira [Cook and Holder 2006]. Por exemplo, um histograma da distribuição de grau de um grafo revela como as conexões estão distribuídas entre os nós, destacando aqueles altamente conectados (*hubs*) e indicando se a rede segue uma distribuição de lei de potência ou se é mais homogênea. Um outro exemplo seria um histograma de pesos de arestas em grafos ponderados, no qual é possível identificar quais arestas carregam maior relevância ou influência na dinâmica da rede. Nesse cenário, arestas com pesos consistentemente altos, evidenciadas por picos no histograma, sugerem conexões críticas ou relacionamentos robustos entre os nós. A Figura 4.2 ilustra dois histogramas, um de grau dos nós e outro de peso das arestas, a partir de um grafo ponderado.

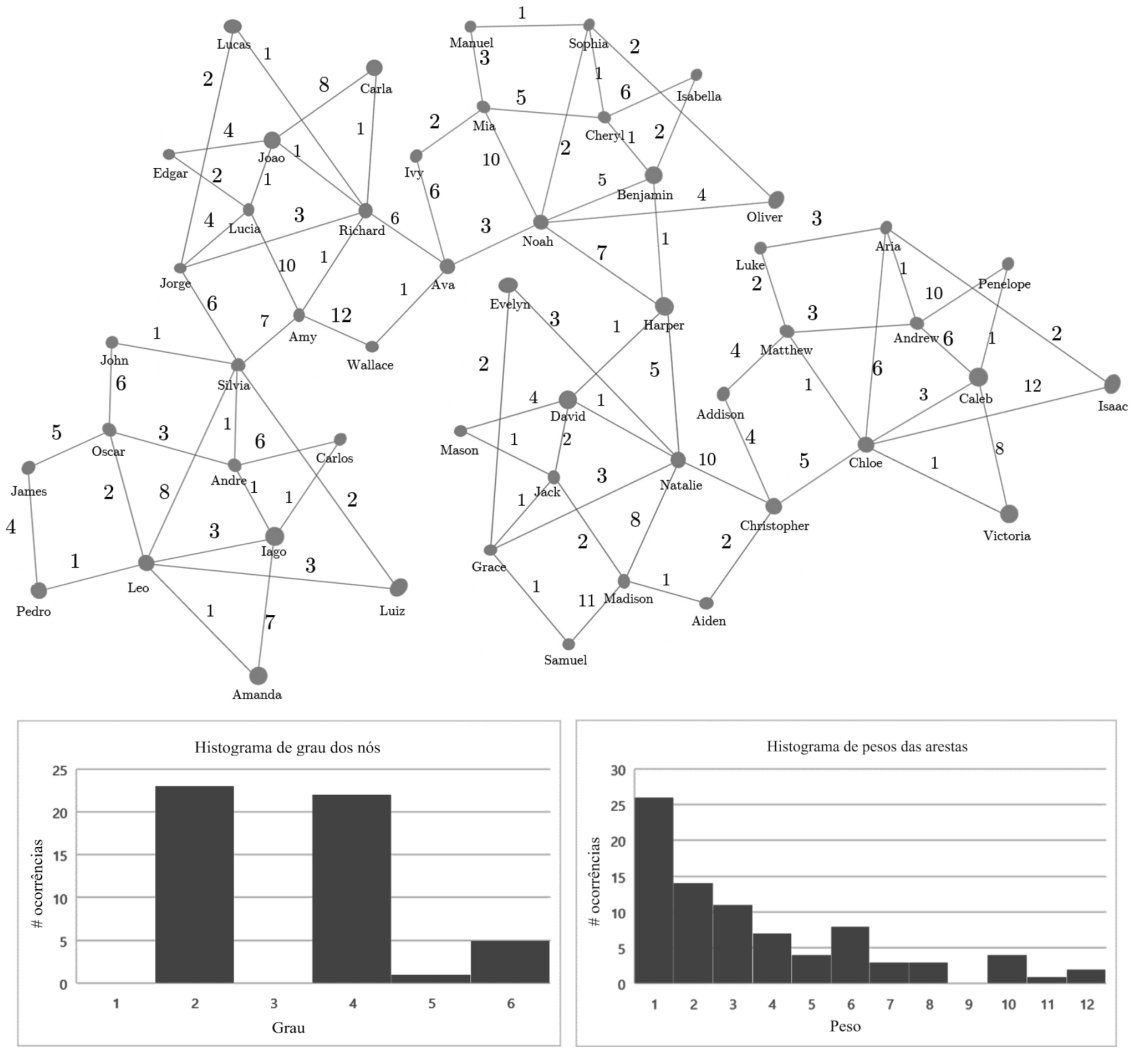


Figura 4.2: Histogramas de grau dos nós e de peso das arestas a partir de um dado grafo.

Adicionalmente, histogramas são úteis para examinar propriedades estruturais dos grafos, como a distribuição de coeficientes de agrupamento, que medem a tendência dos nós a formar *clusters*. Um histograma de coeficientes pode mostrar se a rede possui comunidades bem definidas ou se é mais dispersa. Outro aspecto crítico dos histogramas é sua capacidade de detectar anomalias. Ao comparar os histogramas de diferentes períodos ou subconjuntos da rede, é possível identificar mudanças súbitas ou padrões incomuns que possam indicar comportamentos anômalos, como ataques cibernéticos, falhas sistêmicas ou até mesmo a formação de novos grupos sociais inesperados. Por exemplo, uma alteração significativa na distribuição de grau pode sugerir a adição, ou remoção, massiva de nós ou arestas, apontando para eventos extraordinários na rede.

4.2.3. Centralidade

A centralidade é um conceito essencial na teoria de redes e análise de grafos, utilizado para medir a importância, ou influência, relativa de um nó dentro de uma rede [Bloch et al. 2023]. Essa métrica ajuda a identificar os nós mais centrais ou relevantes, que desempenham papéis cruciais na conectividade e dinâmica da rede. Existem várias medidas de centralidade, cada uma capturando diferentes aspectos da importância dos nós.

Centralidade de Grau (*Degree Centrality*): A centralidade de grau é a medida mais simples e é definida pelo número de conexões diretas (arestas) que um nó possui. Um nó com uma alta centralidade de grau é considerado popular ou altamente visível na rede, já que possui muitas conexões diretas com outros nós. Esta métrica é útil em redes sociais para identificar indivíduos com muitos contatos.

Centralidade de Proximidade (*Closeness Centrality*): A centralidade de proximidade mede a distância média de um nó a todos os outros nós na rede. Um nó com alta centralidade de proximidade pode alcançar todos os outros nós rapidamente, tornando-se central em termos de proximidade. Essa medida é particularmente útil para identificar nós que são eficientes na disseminação de informações pela rede.

Centralidade de Intermediação (*Betweenness Centrality*): A centralidade de intermediação quantifica o número de vezes que um nó atua como ponte ao longo dos caminhos mais curtos entre outros pares de nós. Nós com alta centralidade de intermediação são críticos para a comunicação dentro da rede, pois eles facilitam o fluxo de informações entre diferentes partes da rede.

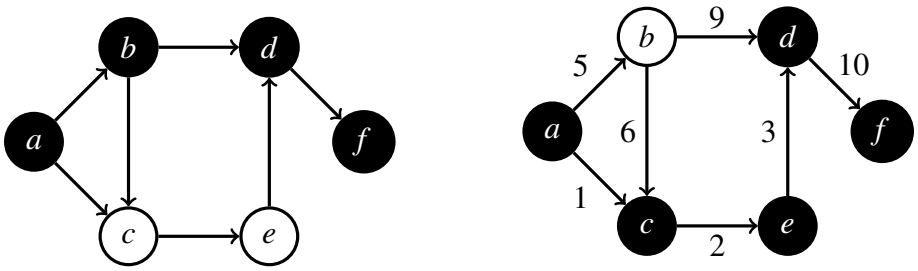
Centralidade de Autovetor (*Eigenvector Centrality*): A centralidade de autovetor mede a influência de um nó com base na importância dos seus vizinhos. Um nó com alta centralidade de autovetor está conectado a outros nós que também são altamente conectados. Essa métrica é útil para identificar não apenas nós centrais, mas também aqueles que estão ligados a outros nós influentes, proporcionando uma visão mais holística da influência dentro da rede.

Centralidade de PageRank (*PageRank Centrality*): Popularizada pelo algoritmo de ranqueamento (*ranking*) do Google, a centralidade de PageRank é semelhante à centralidade de autovetor, mas ajusta a influência de um nó pelo número de conexões que os seus vizinhos têm. É amplamente utilizada para determinar a importância de páginas *web* e pode ser aplicada a outras redes para identificar nós altamente influentes.

Cada uma das medidas de centralidade mencionadas proporciona uma perspectiva distinta sobre a estrutura e a dinâmica de uma rede, permitindo uma análise variada da importância dos nós. Ao combinar essas diferentes medidas, é possível obter uma compreensão ainda mais abrangente dos nós influentes e do papel que desempenham na rede. Dessa forma, a análise de centralidade é uma ferramenta poderosa para otimizar o desempenho, a robustez e a resiliência das redes em diversas aplicações, incluindo não somente redes sociais, mas também redes de comunicação, de transporte e financeiras.

4.2.4. Caminhos mínimos e distâncias

Um caminho mínimo entre dois nós em um grafo é definido como a sequência de arestas que conecta esses dois nós com a menor soma de arestas ou de pesos. Se o grafo é não ponderado, o caminho mínimo é simplesmente o caminho com o menor número de arestas. Já em grafos ponderados, o caminho mínimo é o que minimiza a soma dos pesos das arestas ao longo do caminho [Mitchell 2017]. A Figura 4.3 ilustra uma rede com caminho mínimo em um grafo não ponderado (Figura 4.3a) e em um grafo ponderado (Figura 4.3b).



(a) Grafo não ponderado. (b) Grafo ponderado.

Figura 4.3: Exemplo de caminhos mínimos entre os nós a e f.

Outra métrica importante envolvendo distâncias em grafos é a média de caminhos mínimos. Ela é calculada da seguinte forma: para cada caminho mínimo entre todos os pares de nós, são somadas todas as distâncias e esse valor é dividido pelo número total de pares de nós. Em redes sociais, uma média baixa de caminhos mínimos sugere uma rede mais coesa, onde os indivíduos estão mais próximos uns dos outros, facilitando a disseminação de informações, influências e interações. Por outro lado, uma média alta de caminhos mínimos indica uma rede mais dispersa, onde as distâncias entre os indivíduos são maiores. Tal fato pode dificultar a propagação rápida de informações.

Por fim, a métrica de diâmetro em um grafo é outra medida importante utilizada para caracterizar a distância máxima entre os vértices. Essa métrica mede a maior distância encontrada ao percorrer todos os pares de vértices no grafo, representando a distância mais longa que deve ser percorrida para conectar qualquer par de nós na rede. Diâmetros em redes sociais estão relacionados ao conceito de “seis graus de separação” [Samoylenko et al. 2023], o qual sugere que qualquer pessoa no mundo pode ser conectada a qualquer outra pessoa através de no máximo seis intermediários.

4.3. Ameaças de Privacidade em Redes Sociais

Ameaças de privacidade abrangem uma ampla gama de atividades que resultam no acesso a informações sensíveis por partes não autorizadas. Essas partes, que não deveriam ter acesso a tais informações, podem utilizá-las para fins mal-intencionados. Existem duas principais categorias de ataques de inferência observadas em redes sociais: a inferência de atributos privados [Mislove et al. 2010, Dey et al. 2012], que envolve a dedução de informações pessoais como gostos, interesses ou características demográficas; e a desanonimização (*de-anonymization)* [Narayanan and Shmatikov 2009, Brito et al. 2015, Qian et al. 2016], que visa identificar a identidade real de usuários anônimos, conectando dados anônimos a perfis pessoais conhecidos.

A inferência de atributos privados visa descobrir valores de atributos ocultos que são protegidos pelo usuário ou pelo curador dos dados. Os ataques de inferência baseados em vizinhança [Mislove et al. 2010, Dey et al. 2012, Gong et al. 2014] aproveitam o fato de que usuários próximos podem ter atributos semelhantes, ou idênticos, com alta probabilidade. Esses ataques inferem atributos privados de um usuário explorando os valores de atributos conhecidos de outros usuários com interesses similares [Chaabane et al. 2012]. Por exemplo, se a maioria dos amigos de um usuário trabalha como desenvolvedores de *software*, há uma grande chance de que o próprio usuário também trabalhe nessa área. Já a inferência baseada em comportamento tenta identificar semelhanças entre determinados valores de atributos através de dados comportamentais, como interesses, características pessoais e comportamentos culturais. Por exemplo, se a maioria dos filmes, séries e músicas que um usuário gosta são do gênero "suspense", é provável que o usuário tenha uma forte preferência por esse gênero.

A desanonimização [Narayanan and Shmatikov 2008, Narayanan and Shmatikov 2009, Qian et al. 2016] envolve o uso de uma rede social anonimizada e uma rede social de referência que contém as identidades verdadeiras dos indivíduos, mapeando os nós nesses dois grafos para que as identidades dos usuários no grafo anonimizado possam ser reidentificadas. Uma rede social anonimizada é geralmente disponibilizada por um curador dos dados a vários solicitantes, como pesquisadores, anunciantes, desenvolvedores de aplicativos e agências governamentais, após ocultar informações privadas identificáveis por meio de várias técnicas de anonimização. Uma rede social de referência pode ser obtida através de informações coletadas de outras fontes, como uma rede social diferente que compartilha usuários com a rede publicada. Geralmente, uma rede social de referência pode conter menos atributos sobre os nós do que uma rede social anonimizada, mas ainda assim pode ser usada para correlacionar e identificar indivíduos.

Essas duas categorias de ataques à privacidade resultam na exposição de diferentes tipos de informações sensíveis. A fim de proteger os dados privados em redes sociais e formalizar o conceito de privacidade nesse contexto, foram identificadas as principais ameaças à privacidade, conforme ilustrado na Figura 4.4. Essas ameaças são detalhadas a seguir.

Descoberta de identidade: Em redes sociais, a identidade de um indivíduo pode ser considerada privada, enquanto os atacantes podem explorar diversas informações para reidentificar um usuário da rede ou determinar se um indivíduo-alvo está presente, ou não, nela. Por exemplo, uma empresa disponibiliza para análise uma rede social profissional

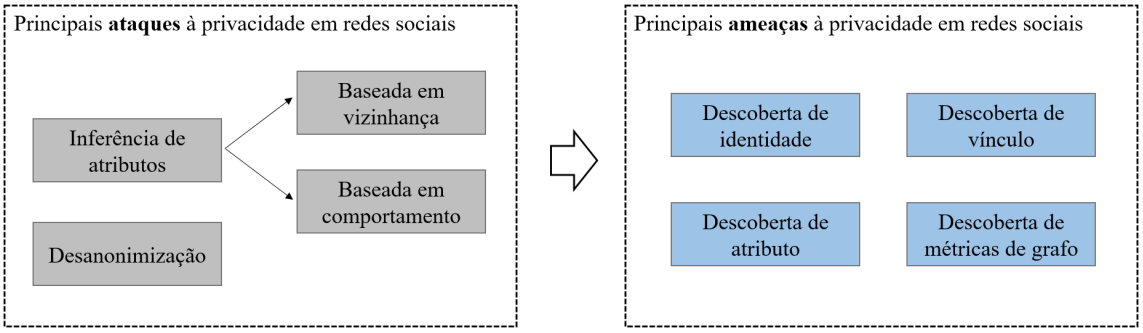


Figura 4.4: Principais ataques e ameaças à privacidade em redes sociais.

anonimizada, a qual contém Maria, com sua identidade oculta. Utilizando informações de outra rede social pública, um adversário observa que um colega de trabalho publica uma foto de uma festa de escritório em que Maria está presente, marcando-a pelo nome verdadeiro. Como resultado, a identidade de Maria pode ser inferida na rede social profissional anonimizada, comprometendo sua privacidade.

Descoberta de vínculo: As relações sociais entre indivíduos podem ser modeladas como arestas em um grafo. As informações de conexão podem ser consideradas sensíveis em alguns casos. Por exemplo, Kossinets e Watts [Kossinets and Watts 2006] analisaram um grafo derivado das comunicações por *e-mail* entre estudantes e membros do corpo docente em uma universidade, onde os relacionamentos de *e-mail* de "quem enviou *e-mails* para quem" foram considerados sensíveis. Um outro exemplo de descoberta de vínculo se dá no seguinte cenário: Ana é uma usuária ativa em uma rede social de *hobbies*, onde ela mantém um perfil anônimo para compartilhar seu interesse em pintura. Ela prefere manter sua associação com um grupo de discussão sobre arte em segredo de seu círculo social pessoal. No entanto, um algoritmo de recomendação de amigos da plataforma sugere conexões com seus colegas de trabalho, revelando indiretamente sua participação no grupo de discussão de arte e expondo seus interesses pessoais. Nesse caso, o algoritmo de recomendação compromete a privacidade de Ana ao expor sua participação no grupo de discussão de arte e seus interesses pessoais.

Descoberta de características do indivíduo: O perfil de um usuário em uma rede social comumente inclui uma variedade de atributos, como idade, gênero, área de estudo e ocupação. Alguns desses atributos, como salário, estado de saúde e informações sobre doenças, são considerados sensíveis e privados. Por exemplo, imagine uma plataforma de rede social voltada para saúde e bem-estar, onde os usuários compartilham informações sobre seus hábitos alimentares, rotinas de exercícios e condições médicas. João é um usuário ativo nessa plataforma, que compartilha regularmente detalhes sobre sua dieta, exercícios e algumas preocupações de saúde. Ele prefere manter sua condição médica de diabetes em segredo de seus colegas de trabalho e familiares, já que prefere tratar essa questão de forma privada. No entanto, João compartilha informações gerais sobre sua dieta saudável e rotina de exercícios na plataforma. A plataforma então decide lançar um recurso de análise de dados para os usuários, que fornece *insights* sobre hábitos saudáveis com base nas informações compartilhadas. Esse recurso utiliza algoritmos avançados para analisar os padrões de comportamento dos usuários na rede social e sugerir mudanças positivas

de estilo de vida. No entanto, durante a análise, o algoritmo da plataforma identifica João como um usuário com diabetes, com base em seus hábitos alimentares e padrões de atividade física. Consequentemente, a plataforma começa a fornecer sugestões específicas para gerenciar o diabetes, como dietas especiais e rotinas de exercícios, para todos os seus contatos na plataforma. Como resultado, João é confrontado com mensagens inesperadas de apoio e conselhos de seus colegas de trabalho e familiares, que ficaram sabendo de sua condição médica através das sugestões automáticas da plataforma.

Descoberta de métricas de grafo: Uma vez que as redes sociais podem ser representadas como grafos, métricas como grau, intermediação, centralidade, comprimento do caminho mais curto, contagem de subgrafos e peso de aresta podem ser utilizadas para realizar análises de redes sociais. No entanto, a divulgação dessas informações pode, inadvertidamente, resultar em vazamentos de dados pessoais. Por exemplo, considere uma rede social de uma comunidade universitária, onde os estudantes interagem entre si e com os professores por meio de postagens, comentários e mensagens privadas. A universidade está interessada em entender os padrões de interação entre os estudantes para melhorar a experiência educacional. A equipe da universidade decide usar métricas de grafo, como centralidade de grau, para identificar os alunos mais influentes na rede. Eles planejam usar essas informações para melhorar a distribuição de recursos, como atividades extracurriculares e oportunidades de liderança. Ao analisar as métricas de grafo, a equipe identifica Pedro como um dos alunos mais influentes na rede, com um alto grau de centralidade. Eles decidem então convidar Pedro para participar de um programa de mentoria para novos alunos, com base em sua influência percebida na comunidade. No entanto, Pedro não queria que sua posição na rede social fosse revelada dessa forma. Ele preferia manter sua participação discreta e demonstrou desagrado diante da atenção indesejada que recebeu como resultado da divulgação de suas métricas de grafo.

4.4. Introdução à Privacidade Diferencial

Nos últimos anos, a Privacidade Diferencial (PD) tornou-se o padrão para a preservação de privacidade em análises de dados sob fortes garantias matemáticas. Muito se deve às limitações existentes nos modelos de privacidade mais tradicionais, ou sintéticos, como o k-anonimato, l-diversidade, t-proximidade, δ-presença, dentre outros, os quais ainda deixavam os indivíduos vulneráveis quanto à sua privacidade [Brito and Machado 2017].

4.4.1. Intuição e Definição

Na definição original de privacidade diferencial, os dados privados são vistos como uma coleção de registros, onde cada registro corresponde a um indivíduo. Em essência, a privacidade diferencial assegura a proteção à privacidade do indivíduo ao injetar ruído no resultado de consultas aplicadas sobre os dados dos indivíduos, ou seja, modificando os dados originais através da introdução de aleatoriedade [Dwork et al. 2006]. A premissa base da privacidade diferencial é de que o resultado de qualquer consulta é, praticamente, igualmente possível de ocorrer, independente da presença, ou ausência, de qualquer indivíduo no conjunto de dados. É importante mencionar que a privacidade diferencial não é uma simples ferramenta, mas um paradigma capaz de quantificar e gerenciar os riscos de violação de privacidade. Portanto, a privacidade diferencial pode ser aplicada em desde simples estimativas estatísticas até mesmo em aprendizado de máquina.

Seja Q uma consulta a ser realizada sobre um conjunto de dados D, o qual contém informações sensíveis sobre um conjunto de indivíduos. A PD é definida através de um algoritmo aleatório M, também chamado de mecanismo, o qual é executado sobre D. A PD assegura que a saída de $M(D)$ deve ser semelhante à saída de $Q(D)$, ou seja, o objetivo da PD é fazer com que a saída de $M(D)$ seja o mais próximo possível à saída de $Q(D)$ para garantir a utilidade dos dados e, ao mesmo tempo, preservar a privacidade de todos os indivíduos presentes no conjunto de dados.

Para preservar a privacidade de todos os indivíduos através do mecanismo M, a PD estabelece a noção de conjuntos de dados vizinhos. Dois conjuntos de dados D e D' são ditos vizinhos se diferirem em, no máximo, um registro, denotado como $D \sim D'$. D' pode ser obtido a partir de D adicionando, ou removendo, um único registro. A PD também garante que, independentemente de quando a entrada seja D ou D', a probabilidade de uma determinada saída ocorrer a partir de $M(D)$ ou $M(D')$ é quase a mesma. Essa propriedade é denotada como indistinguibilidade de conjuntos de dados vizinhos. Em outras palavras, a PD afirma que qualquer resposta de uma consulta ocorre com probabilidade semelhante, independentemente da presença, ou ausência, de qualquer indivíduo no conjunto de dados.

Antes de apresentar a definição de PD, considere que $Range(M)$ consiste em todas as saídas possíveis de M, ou seja, seu domínio de saídas. Por exemplo, se M calcula o número de registros em um conjunto de dados, então $Range(M)$ é igual a um conjunto de números inteiros não negativos. Por fim, a definição de PD é apresentada abaixo:

Definição 1 (ε-Privacidade Diferencial [Dwork 2006])**.** *Um mecanismo M satisfaz ε-privacidade diferencial se, para quaisquer dois conjuntos de dados vizinhos D e D', e para qualquer saída possível $O \subseteq Range(M)$,*

$$\Pr[M(D) = O] \leq \exp(\varepsilon) \times \Pr[M(D') = O], \tag{1}$$

onde $\Pr[\cdot]$ *representa a probabilidade do mecanismo produzir a saída O.*

4.4.2. Orçamento de Privacidade

O parâmetro ε que aparece na Definição 1 é denominado orçamento de privacidade. Este parâmetro é responsável por controlar as diferenças entre as probabilidade das saídas de um mecanismo que é executado em dois conjuntos de dados vizinhos, ou seja, o orçamento de privacidade garante que essas diferenças sejam limitadas a no máximo ε.

O orçamento de privacidade consiste em um número real positivo que controla o nível de privacidade que um mecanismo M fornece. Um ε menor fornece garantias de privacidade mais fortes, com distribuições de probabilidade mais indistinguíveis, mas uma menor utilidade de dados, uma vez que mais ruído deve ser adicionado ao resultado. De maneira análoga, um ε maior fornece garantias de privacidade mais fracas, mas uma maior utilidade de dados.

Definir o valor adequado de ε para uma aplicação é uma tarefa bastante desafiadora. Tal tarefa exige o esforço de diversas partes, como especialistas em privacidade, *stakeholders* e proprietários de dados, ou seja, indivíduos que compartilham seus dados, a fim de fornecer *feedback* contínuo para garantir a privacidade dos indivíduos e, ao mesmo tempo, divulgar informações significativas. No entanto, o orçamento de privacidade normalmente assume valores pequenos, tornando as probabilidades de saída do mecanismo

quase as mesmas, independente da entrada do mecanismo ser D ou D'. Vários estudos já foram abordados com o objetivo de determinar um valor desejável para ε [Hsu et al. 2014, Li et al. 2016]. No entanto, tem sido bastante defendido que $0{,}1 \leq \varepsilon \leq 1$ fornece garantias de privacidade fortes e níveis de utilidade aceitáveis, enquanto $\varepsilon \geq 5$ é aceitável apenas em algumas aplicações específicas.

4.4.3. Sensibilidade

Conforme mencionado anteriormente, a PD pode ser alcançada ao adicionar uma quantidade apropriada de ruído aos resultados das consultas. Entretanto, adicionar ruído excessivo pode prejudicar drasticamente a utilidade dos dados, diminuindo a precisão das análises, enquanto uma quantidade insuficiente de ruído pode não fornecer as garantias de privacidade adequadas. Para isso, o ruído adicionado a uma consulta Q depende da sensibilidade global de Q [Dwork et al. 2006]. A definição de sensibilidade global é dada abaixo:

Definição 2 (Sensibilidade Global [Dwork et al. 2006])**.** *A sensibilidade global de uma consulta Q consiste na máxima distância l_1 entre as saídas de Q em quaisquer dois conjuntos de dados vizinhos D e D', dada por:*

$$\Delta Q = \max_{D,D'} ||Q(D) - Q(D')||_1. \tag{2}$$

A sensibilidade global, também chamada apenas de sensibilidade, mede o impacto máximo nos resultados da consulta a partir da adição, ou remoção, de qualquer registro no conjunto de dados. A sensibilidade serve como um parâmetro essencial para determinar a quantidade adequada de ruído adicionado à consulta. É importante mencionar que a sensibilidade está relacionada apenas à função de consulta e é independente do conjunto de dados. Por exemplo, uma consulta com baixa sensibilidade exige que apenas uma pequena quantidade de ruído seja adicionada aos resultados da consulta para mascarar o impacto da adição, ou remoção, de um registro. Por outro lado, quando a sensibilidade é alta, uma quantidade significativa de ruído deve ser adicionada aos resultados da consulta, a fim de garantir a privacidade dos indivíduos, comprometendo a utilidade dos dados.

Para algumas consultas, a sensibilidade é simples de ser calculada. Por exemplo, a sensibilidade de consultas de contagem é 1, uma vez que adicionar, ou remover, um registro no conjunto de dados afetará o resultado das consultas em no máximo 1. Por outro lado, a sensibilidade de consultas mais complexas, como consultas de máximo e soma, não é tão simples de calcular como as consultas de contagem.

Por exemplo, considere uma consulta que calcula a soma dos pesos das pessoas em um determinado conjunto de dados. A inclusão de um novo registro no conjunto de dados aumentará o resultado da consulta em um valor equivalente ao peso do indivíduo adicionado. Portanto, a sensibilidade dependerá do valor do peso do indivíduo adicionado ao conjunto de dados. Note que o mesmo raciocínio é valido para a remoção de um registro do conjunto de dados. Deseja-se, então, atribuir um valor específico para representar a sensibilidade dessa consulta, uma vez que a consulta deve ser independente do conjunto de dados. Para o domínio específico de pesos, existe um limite superior racional conhecido para o peso máximo que um indivíduo pode ter. De acordo com [Allardyce

2012], Jon Brower Minnoch foi a pessoa mais pesada conhecida no mundo já documentada, pesando impressionantes 635 quilos. Dessa forma, é plausível atribuir um valor de 635 à sensibilidade dessa consulta. No entanto, isto não serve como prova definitiva, uma vez que é impossível garantir que outra pessoa volte a pesar 635 quilos, ou mais. Então, em alguns domínios, determinar uma sensibilidade razoável pode ser uma tarefa desafiadora [Near and Abuah 2021].

4.4.4. Mecanismos

Mecanismos são algoritmos capazes de garantir as propriedades da PD. Para consultas numéricas, a PD pode ser alcançada através de vários mecanismos, como o mecanismo de Laplace [Dwork 2006] e o mecanismo geométrico [Ghosh et al. 2009]. Embora ambos os mecanismos tenham sido projetados para consultas numéricas, eles divergem no tipo de ruído que é adicionado ao resultado da consulta. O mecanismo de Laplace é recomendado para consultas que geram valores reais, pois esse mecanismo produz valores de ruído $\in \mathbb{R}$. Por sua vez, o mecanismo geométrico é recomendado para consultas que geram valores inteiros, pois esse mecanismo produz valores de ruído $\in \mathbb{Z}$. No entanto, nem todas as consultas geram valores numéricos. Para esse tipo de consulta, também chamada de consulta categórica, o mecanismo exponencial [McSherry and Talwar 2007] é mais adequado.

4.4.4.1. Mecanismo de Laplace

Conforme brevemente mencionado anteriormente, o mecanismo de Laplace adiciona ruído de valor real aos resultados da consulta. Como o nome sugere, o mecanismo depende da distribuição de Laplace para gerar valores aleatórios, os quais serão adicionados ao resultado da consulta. Seja x o ruído adicionado ao resultado de uma de consulta Q, a distribuição de Laplace é definida conforme:

Definição 3 (Distribuição de Laplace)**.** *A distribuição de Laplace com média 0 e escala b é a distribuição com função densidade de probabilidade*

$$Lap(x|b) = \frac{1}{2b} \cdot \exp(-\frac{|x|}{b}). \tag{3}$$

Considere $Lap(b)$ a distribuição de Laplace com escala b, Q uma consulta e D um conjunto de dados. O funcionamento do mecanismo de Laplace é dado por calcular o resultado de $Q(D)$ e perturbar esse resultado com a adição de um ruído gerado a partir da distribuição de Laplace. A escala b do ruído gerado é calibrada através da relação entra a sensibilidade da consulta e o orçamento de privacidade, de maneira que $b = \frac{\Delta Q}{\varepsilon}$.

Teorema 1 (Mecanismo de Laplace [Dwork et al. 2006])**.** *O mecanismo de Laplace que adiciona o ruído gerado a partir de* $Lap(\frac{\Delta Q}{\varepsilon})$ *satisfaz* ε*-PD.*

4.4.4.2. Mecanismo Geométrico

O mecanismo geométrico [Ghosh et al. 2009] consiste na versão discreta do mecanismo de Laplace, ou seja, adiciona ruído inteiro aos resultados das consultas seguindo a dis-

tribuição geométrica. Dessa forma, garante-se que o resultado final da consulta seja um número inteiro. Portanto, o mecanismo geométrico é especializado em melhorar o desempenho de consultas de contagem [Ghosh et al. 2009]. A distribuição geométrica é definida conforme:

Definição 4 (Distribuição Geométrica)**.** *Uma variável aleatória X gerada a partir da distribuição geométrica tem uma função massa de probabilidade*

$$P(X = x) = \frac{1-\alpha}{1+\alpha}\alpha^{|x|}, \tag{4}$$

onde $0 \leq \alpha \leq 1$.

Em particular, quando $\alpha = e^{-\frac{\varepsilon}{\Delta Q}}$, o mecanismo geométrico é ε-PD.

Teorema 2 (Mecanismo Geométrico [Ghosh et al. 2009])**.** *O mecanismo geométrico que adiciona o ruído gerado a partir da distribuição geométrica, com* $\alpha = e^{-\frac{\varepsilon}{\Delta Q}}$*, satisfaz* ε*-PD.*

4.4.4.3. Mecanismo Exponencial

Conforme mencionado anteriormente, o mecanismo exponencial surge como uma solução para consultas que não retornam valores numéricos. Para tanto, [McSherry and Talwar 2007] propuseram o mecanismo exponencial, o qual garante PD para consultas categóricas. Sua ideia principal consiste em escolher uma saída O do espaço de saída $\mathscr{O}$, de acordo com uma função de utilidade u. Essa função de utilidade atribui probabilidades de escolha exponencialmente maiores às saídas com utilidades mais elevadas. Além disso, a escolha de u depende da aplicação. Dessa forma, aplicações diferentes levam a funções de utilidade distintas. Diferente dos demais mecanismos anteriores, Laplace e geométrico, os quais geram seus ruídos proporcionalmente à sensibilidade da consulta Q, no mecanismo exponencial é utilizado o conceito de sensibilidade da função de utilidade para prover a saída do mecanismo. A sensibilidade da função de utilidade é definida abaixo:

Definição 5 (Sensibilidade Global da Função de Utilidade [McSherry and Talwar 2007])**.** *A sensibilidade global de uma função de utilidade u é dada por*

$$\Delta u = \max_{O \in \mathscr{O}} \max_{D,D' vizinhos} |u(D,O) - u(D',O)|. \tag{5}$$

Teorema 3 (Mecanismo Esponencial [McSherry and Talwar 2007])**.** *Dada uma função de utilidade* $u : (D \times \mathscr{O} \rightarrow \mathbb{Z})$ *para um conjunto de dados D, o mecanismo M que gera uma saída* $O \in \mathscr{O}$*, com probabilidade proporcional a* $\exp(\frac{\varepsilon \times u(D,O)}{2\Delta u})$*, satisfaz* ε*-PD.*

4.4.5. Propriedades

Várias propriedades úteis integram os mecanismos de PD, como o pós-processamento, a composição sequencial e a composição paralela. A propriedade de pós-processamento assume que qualquer função aplicada à saída de um mecanismo diferencialmente privado também satisfaz PD. Por outro lado, a propriedade de composição sequencial assume que

qualquer sequência de mecanismos diferencialmente privados, aplicados sobre o mesmo conjunto de dados, e que satisfaça PD isoladamente, também fornece PD. Por sua vez, a propriedade de composição paralela assume que um mesmo mecanismo diferencialmente privado, aplicado sobre conjuntos de dados disjuntos, e que satisfaça PD isoladamente, também fornece PD.

Teorema 4 (Pós-processamento [Dwork et al. 2014])**.** *Seja M qualquer mecanismo tal que $M(D)$ seja ε-diferencialmente privado, e seja f qualquer função. Então, $f(M(D))$ também satisfaz ε-PD.*

Teorema 5 (Composição Sequencial [Dwork et al. 2014])**.** *Considere que cada mecanismo M_i satisfaz ε_i-PD. A sequência de mecanismos diferencialmente privados $M_i(D)$ satisfaz $\sum \varepsilon_i$-PD.*

Teorema 6 (Composição Paralela [Dwork et al. 2014])**.** *Considere que cada conjunto de dados D_i é disjunto e que um mecanismo M satisfaz ε_i-PD para o conjunto de dados D_i. A sequência de mecanismos diferencialmente privados $M(D_i)$ satisfaz* $\max(\varepsilon_i)$*-PD.*

Quando combinadas, essas propriedades fornecem flexibilidade para desenvolver uma maneira de agregar várias etapas diferencialmente privadas em um único mecanismo que satisfaça a PD.

4.4.6. Privacidade Diferencial Local

A privacidade diferencial, conforme apresentada anteriormente, considera a existência de um curador confiável (*third-party*), o qual é responsável por coletar os dados, perturbar os resultados das consultas através de um mecanismo que satisfaça a PD e disponibilizar os resultados ruidosos. Essa configuração de PD é geralmente denominada de PD global, PD centralizada, ou simplesmente PD. No entanto, encontrar um curador confiável para coletar e processar os dados pode ser uma tarefa bastante desafiadora em cenários práticos. Portanto, a falta de curadores confiáveis restringe a aplicabilidade da PD global. Por conta disso, a Privacidade Diferencial Local (PDL) [Duchi et al. 2013] foi proposta como uma abordagem diferencialmente privada que desconsidera a necessidade de um curador de dados confiável. Dessa forma, em vez de centralizar o fluxo de dados em uma única entidade externa supostamente confiável, cada indivíduo é responsável por proteger os seus próprios dados, perturbando-os localmente, por meio de um mecanismo diferencialmente privado, antes de enviá-los ao curador de dados. Na PDL, o curador de dados também é comumente chamado de *agregador*.

Quando comparada à PD global, a PDL é uma noção mais forte de privacidade, a qual mantém os dados sensíveis dos indivíduos privados até mesmo de curadores de dados não confiáveis. No entanto, por se tratar de uma noção de privacidade mais forte, espera-se que a PDL introduza mais ruído aos resultados sob as mesmas circunstâncias, ou seja, utilizando-se o mesmo orçamento de privacidade, em comparação com o PD global. A definição formal da PDL é formalmente apresentada abaixo:

Definição 6 (ε-Privacidade Diferencial Local)**.** *Um mecanismo M satisfaz ε-privacidade diferencial local se, para qualquer par de valores de entrada v e v', e para qualquer saída possível $O \subseteq Range(M)$:*

$$\Pr[M(v) = O] \leq \exp(\varepsilon) \times \Pr[M(v') = O]. \tag{6}$$

A principal diferença entre a PD e a PDL está nos dados de entrada que os mecanismos recebem. Um mecanismo de PD global recebe um conjunto de dados D como entrada, ou seja, os dados de todos os indivíduos, e garante que a saída seja indistinguível, enquanto que a PDL recebe apenas os dados de um único indivíduo v como entrada e gera respostas ruidosas por indivíduo, de maneira independente.

4.4.7. Protocolos de Privacidade Diferencial Local

Como mencionado anteriormente, os mecanismos são formas de garantir as propriedades de PD. Na PDL, essas propriedades são alcançadas através do uso de protocolos. Portanto, na PDL, os mecanismos são chamados de protocolos. Em resumo, protocolos são técnicas que modificam os dados do indivíduo para garantir as propriedades da PDL. O fluxo padrão de um protocolo de PDL consiste em: (I) codificar os dados do indivíduo; (II) perturbar os dados do indivíduo; e (III) enviar os dados ruidosos do indivíduo ao curador de dados.

(I) Codificação: Nessa etapa, os dados v do indivíduo são codificados em um vetor de bits B de tamanho d formado por 0's e 1's, de maneira que o valor 1 é atribuído às posições do vetor B que correspondem a v e 0 para as demais. Portanto, define-se a função `Codificar`(v) = B, tal que $B[v] = 1$ e $B[i] = 0$ para todo $i \neq v$.

(II) Perturbação: Nessa etapa, o vetor de bits codificado B é perturbado de acordo com dois parâmetros principais: p e q, formando um novo vetor de bits B', conforme mostrado na Equação 7. O parâmetro p consiste na probabilidade de que um bit i de B atribuído com o valor 1 permaneça sendo 1 mesmo após ser perturbado, ou seja, $B[i] = 1 \rightarrow B'[i] = 1$. Por sua vez, q é a probabilidade de que um bit de B atribuído com 0 se torne 1 após ser perturbado, ou seja, $B[i] = 0 \rightarrow B'[i] = 1$. Intuitivamente, $(1-p)$ e $(1-q)$ representam as probabilidade de $B[i] = 1 \rightarrow B'[i] = 0$ e $B[i] = 0 \rightarrow B'[i] = 0$, respectivamente. Essa etapa é executada de maneira diferencialmente privada, através do uso do parâmetro de orçamento de privacidade ε para determinar os valores de probabilidade p e q. Além disso, os valores de p e q dependem não apenas do valor de ε, mas também do protocolo escolhido. Uma vez perturbado, o vetor B' é reportado ao agregador.

$$\Pr[B'(i) = 1] = \begin{cases} p, \text{ se } B[i] = 1 \\ q, \text{ se } B[i] = 0 \end{cases} \tag{7}$$

(III) Agregação: Nessa etapa, o agregador coleta todos os vetores de bits perturbados B' reportados pelos indivíduos, e realiza a análise desses dados a partir de informações agregadas. A base para realizar as análises consiste em identificar o número de ocorrências de cada possível valor de entrada v a partir dos vetores B', através de uma função de `Suporte`. Por exemplo, um vetor B' é dito que suporta um valor de entrada v se $B'[v] = 1$, ou seja, `Suporte`$(B') = \{v \mid B'[v] = 1\}$ é conjunto de valores presentes em B'. De maneira semelhante, `Suporte`(v) é definido como o número de ocorrências do valor v nos vetores B' reportados.

É importante mencionar que as funções de `Codificar` e `Suporte` são diretamente dependentes do protocolo de PDL utilizado. Dessa forma, protocolos diferentes podem implementar essas funções de maneira diferente. Além disso, algumas informa-

ções relevantes são de domínio público, ou seja, conhecidas pelo agregador. Em resumo, o número de respostas n, o tamanho do vetor codificado d, o orçamento de privacidade ε e o protocolo de PDL utilizado, todas essas informações são de conhecimento do agregador. Dessa forma, o agregador é capaz de compreender a partir de qual protocolo de PDL que os dados foram sanitizados e, então, calcular os respectivos valores de probabilidade p e q. Finalmente, o agregador pode realizar uma estimativa imparcial da frequência dos valores reportados de acordo com o Teorema 7. Quando cada indivíduo envia os seus dados apenas uma única vez, o número de respostas n pode ser tratado como o número de indivíduos de maneira equivalente.

Teorema 7 (Estimativa Imparcial [Wang et al. 2017])**.** *Dado um protocolo de PDL, o número de ocorrências (contagem) de um valor v, dado por* $\tilde{c}(v) = \frac{\texttt{Suporte}(v) - n \times q}{p - q}$*, é imparcial, onde* $\texttt{Suporte}(v)$ *é o número de respostas que contém o valor v e n é o número de respostas.*

O problema de estimativa de frequências, onde o agregador busca estimar as frequências dos valores em um domínio previamente estabelecido, é um dos problemas mais fundamentais que a PDL se propõe a resolver. Problemas dessa natureza são comumente conhecidos como Oráculos de Frequência (OF). Vários estudos já foram realizados para desenvolver protocolos de OF [da Costa Filho and Machado 2023, Acharya et al. 2019, Bassily and Smith 2015, Ye and Barg 2018], onde o Protocolo de Resposta Aleatória (PRA) [Dwork et al. 2006] e o Protocolo de Codificação Unária (PCU) [Erlingsson et al. 2014] estão entre os mais disseminados na literatura.

4.4.7.1. Protocolo de Resposta Aleatória (PRA)

O protocolo de resposta aleatória foi um dos primeiros protocolos de OF propostos na literatura. Dentre as suas principais características, destaca-se a de permitir que um valor v seja codificado em um vetor de bits B, de maneira que B possua mais de um bit representativo, ou seja, marcado com 1. Tal representação pode ser bastante útil em diversos domínios. Imagine, dentro do cenário de redes sociais, que um indivíduo u deseja reportar as suas conexões existentes com outros indivíduos. Nesse caso, o valor v a ser reportado consiste em uma lista contendo os demais indivíduos que se conectam com u. Portanto, uma maneira de codificar v seria transformá-lo em em um vetor de bits B, de modo que $B[i] = 1$ indica que existe uma conexão entre os indivíduos u e i, enquanto $B[i] = 0$ indica a inexistência dessa conexão.

Visando garantir as propriedades da PDL, foi provado em [Erlingsson et al. 2014] que, para que o PRA satisfaça ε-PDL, a etapa de perturbação precisa ser realizada com valores específicos de p e q, tal que $p = \frac{1}{1+e^{\varepsilon}}$ e $q = 1 - p$.

4.4.7.2. Protocolo de Codificação Unária (PCU)

O protocolo de codificação unária difere-se do protocolo de resposta aleatória, principalmente, na forma em como é realizada a codificação dos dados. Como o nome sugere, no PCU, a codificação de qualquer valor é realizada através de um único bit representativo.

Assim, para um dado valor v, este será codificado em um vetor de bits B, de maneira que B possua um único bit marcado com 1, enquanto todos os demais bits assumem um valor igual a 0. Tal representação também é bastante útil em diversos domínios, principalmente em domínios mais simples e de menor dimensão d.

Assim como no PRA, também é necessário estabelecer quais são os valores de p e q que permitem que o PCU satisfaça ε-PDL, afim de garantir as propriedades da PDL. Diante disso, surgiram alguns protocolos baseados no protocolo de codificação unária com características particulares. Dentre eles, destacam-se o Protocolo de Codificação Unária Simétrica (PCUS) [Erlingsson et al. 2014] e o Protocolo de Codificação Unária Otimizada (PCUO) [Wang et al. 2017]. Ambos os protocolos são bastante similares em relação as etapas executadas por cada um, a grande diferença está na escolha dos parâmetros p e q a serem utilizados na etapa de perturbação de cada protocolo.

No protocolo de PCUS, os valores de p e q assumem valores que tratam os bits iguais a 0 e 1 de maneira simétrica. Os valores de p e q são dados por $p = \frac{e^{\frac{\varepsilon}{2}}}{e^{\frac{\varepsilon}{2}}+1}$ e $q = \frac{1}{e^{\frac{\varepsilon}{2}}+1}$, de maneira que $p+q = 1$. Por sua vez, o PCUO consiste em um melhoramento do PCUS, o qual propõe valores ótimos para p e q. Dessa forma, atribuir $p = \frac{1}{2}$ e $q = \frac{1}{e^{\varepsilon}+1}$ potencializa a utilidade das frequências estimadas. Note que, o valor de $p+q$ nunca será igual a 1, no máximo será próximo, visto que o valor de ε é sempre positivo. Em resumo, a ideia do PCUO é de que, com esses valores de p e q, o protocolo tenta intensificar que os bits reportados como 1 sejam, de fato, aqueles que originalmente eram 1, assim como os bits iguais a 0 são aqueles que, originalmente, também eram iguais a 0.

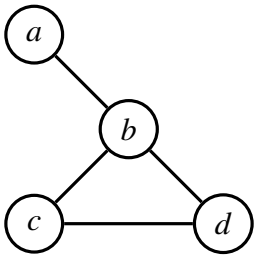
Por fim, é importante destacar que não existe um protocolo que seja o melhor para tudo. Existem diversos protocolos na literatura, com características e finalidades diferentes. Portanto, determinar qual protocolo é mais recomendado para um determinada tarefa se torna uma tarefa bastante desafiadora [Wang et al. 2017].

4.5. Privacidade Diferencial para Redes Sociais

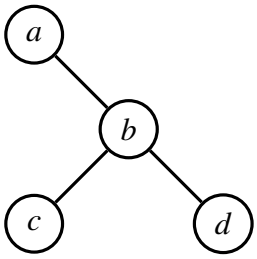
O conceito fundamental de privacidade diferencial depende diretamente da definição de conjuntos de dados vizinhos. Nas definições anteriores, um conjunto de dados vizinho é definido como um conjunto de dados obtido pela adição, ou remoção, de um único registro. Entretanto, no contexto de redes sociais, que concentram-se principalmente nas relações entre indivíduos, a associação entre dados privados e os registros dos conjuntos de dados tornam-se menos aparente. Dessa forma, antes de podermos aplicar PD em redes sociais, é necessário estabelecer uma nova definição de conjuntos de dados vizinhos que considere a estrutura de rede social e a semântica de privacidade associada à rede.

4.5.1. *Edge*-Privacidade Diferencial

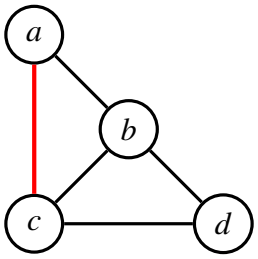
A primeira configuração de PD para redes sociais é chamada de *edge*-privacidade diferencial (*edge*-PD) [Hay et al. 2009]. A noção de *edge*-PD considera que duas redes sociais são vizinhas se uma puder ser obtida a partir da outra ao adicionar, ou remover, uma única aresta, ou adicionar, ou remover, um único usuário isolado, ou seja, um nó sem nenhuma aresta conectada a ele. A definição de *edge*-PD é formalmente definida abaixo:



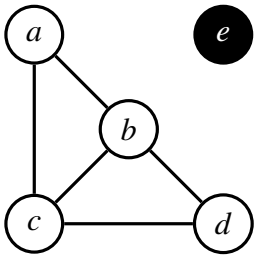
(a) Rede social G.



(b) Rede social vizinha G'_1 obtida a partir de G após a remoção da aresta (c,d).



(c) Rede social vizinha G'_2 obtida a partir de G após a adição da aresta (a,c).



(d) Rede social vizinha G'_3 obtida a partir de G após a adição do usuário e sem nenhuma aresta.

Figura 4.5: Exemplo de uma rede social e suas possíveis redes sociais vizinhas de acordo com a noção *edge*-PD.

Definição 7 (ε-*Edge*-Privacidade Diferencial [Hay et al. 2009])**.** *Um mecanismo M satisfaz ε-edge privacidade diferencial se, para qualquer par de grafos $G = (V,E)$ e $G' = (V',E')$, tal que $|V \oplus V'| + |E \oplus E'| = 1$, e para qualquer saída possível $O \subseteq Range(M)$,*

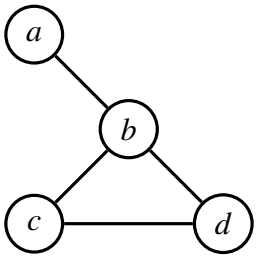
$$\Pr[M(G) = O] \leq \exp(\varepsilon) \times \Pr[M(G') = O]. \tag{8}$$

Dessa forma, um algoritmo que garante *edge*-PD fornece proteção contra a descoberta de arestas dos usuários, ou seja, os relacionamentos. Portanto, é importante destacar que esse nível de privacidade pode ser adequado para algumas aplicações. No entanto, existem alguns cenários em que é desejável estender as garantias de privacidade para além dos relacionamentos dos usuários.

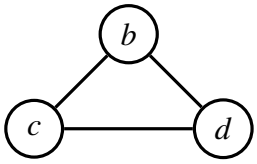
A Figura 4.5 apresenta um exemplo de uma rede social G e três possíveis redes sociais vizinhas G'_1, G'_2 e G'_3, de acordo com a noção de vizinhança provida na *edge*-PD. A rede G é composta, inicialmente, por 4 nós, ou usuários, e 4 arestas representando os relacionamentos entre os usuários (Figura 4.5a). Dentre as possíveis redes sociais vizinhas, a rede G'_1 é gerada a partir da adição de uma aresta (Figura 4.5b), enquanto a rede G'_2 é gerada a partir da remoção de uma aresta (Figura 4.5c) e, por fim, a rede G'_3 é gerada a partir da adição de um nó sem nenhuma uma aresta (Figura 4.5d).

4.5.2. *Node*-Privacidade Diferencial

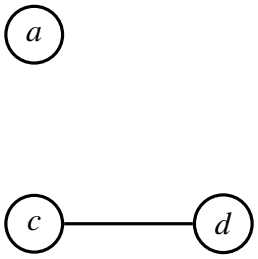
Proposto por [Kasiviswanathan et al. 2013], a *node*-privacidade diferencial (*node*-PD) consiste em uma noção mais estrita de PD quando comparada à *edge*-PD, visto que obje-



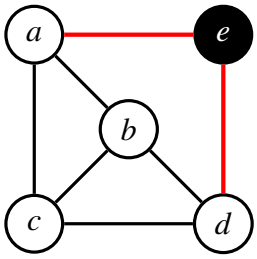
(a) Rede social G.



(b) Rede social vizinha G'_1 obtida a partir de G após a remoção do usuário a, juntamente com todas as suas arestas adjacentes (a,b).



(c) Rede social vizinha G'_2 obtida a partir de G após a remoção do usuário b, juntamente com todas as suas arestas adjacentes (b,a), (b,c) e (b,d).



(d) Rede social vizinha G'_3 obtida a partir de G após a adição do usuário e, juntamente com todas as suas arestas adjacentes (e,a) e (e,d).

Figura 4.6: Exemplo de uma rede social e suas possíveis redes sociais vizinhas de acordo com a noção *node*-PD.

tiva limitar a inferência sobre a existência, ou ausência, de um usuário em uma rede social. Assim, a *node*-PD fornece não somente garantias de privacidade aos usuários, mas também a todos os seus relacionamentos adjacentes. Assim, duas redes sociais G e G' são consideradas vizinhas se diferirem em um único nó e todas as suas arestas adjacentes. A definição de *node*-PD é formalmente apresentada abaixo:

Definição 8 (ε-*Node*-Privacidade Diferencial [Kasiviswanathan et al. 2013])**.** *Um mecanismo M satisfaz ε-node privacidade diferencial se, para qualquer par de grafos $G = (V,E)$ e $G' = (V',E')$, tal que $|V \oplus V'| = 1$ e $E \oplus E' = \{(u,v) | u \in (V \oplus V') \text{ ou } v \in (V \oplus V')\}$, e para qualquer saída possível $O \subseteq Range(M)$,*

$$\Pr[M(G) = O] \leq \exp(\varepsilon) \times \Pr[M(G') = O]. \tag{9}$$

Alcançar a PD no modelo de privacidade *node*-PD é muito mais difícil do que na *edge*-PD, uma vez que a *node*-PD fornece garantias de privacidade mais fortes. Dessa forma, pode ser que seja inviável projetar algoritmos que garantam a *node*-PD e, simultaneamente, forneçam análises precisas em redes sociais. A Figura 4.6 apresenta um exemplo de uma rede social G e três possíveis redes sociais vizinhas G'_1, G'_2 e G'_3, de acordo com a noção de vizinhança provida na *node*-PD.

A rede G é composta, inicialmente, por 4 nós, ou usuários, e 4 arestas representando os relacionamentos entre os usuários (Figura 4.6a). Dentre as possíveis redes

sociais vizinhas, as redes G_1' e G_2' são geradas a partir da remoção de um nó, juntamente com todas as suas arestas adjacentes (Figuras 4.6b e 4.6c), enquanto a rede G_3' é gerada a partir da adição de um nó, juntamente com todas as suas arestas adjacentes (Figura 4.6d).

4.5.3. *Edge-weight* Privacidade Diferencial

As duas principais alternativas propostas para aplicar privacidade diferencial em grafos, *node*-privacidade diferencial e *edge*-privacidade diferencial, não são adequadas quando os grafos são ponderados. Em geral, não é possível disponibilizar algumas informações como, por exemplo, caminhos mínimos, com um nível de utilidade significativo através das noções de *node*-PD ou *edge*-PD, uma vez que modificar uma única aresta pode alterar as distâncias do grafo significativamente. Dessa forma, surge um modelo de privacidade diferencial adequado para grafos ponderados, o qual é denominado *edge-weight* privacidade diferencial (*edge-weight* PD).

Nesse contexto, existem dois tipos principais de *edge-weight* PD, um que considera que a topologia de um grafo é conhecida [Sealfon 2016] e outro que a considera desconhecida [Brito et al. 2023]. Suponha $G = (V, E, \omega)$ um grafo ponderado não direcionado com uma função de peso $\omega : V^2 \to \mathbb{R}^+$ que mapeia conexões entre um par de vértices (u, v) para pesos em G.

Definição 9 (Funções de peso vizinhas com topologia conhecida [Sealfon 2016])**.** *Duas funções de peso ω, $\omega' : V^2 \to \mathbb{R}^+$ são vizinhas, denotadas por $\omega \sim \omega'$, se:*

$$||\omega - \omega'||1 := \sum u, v \in V |\omega(u, v) - \omega'(u, v)| \leq 1. \tag{10}$$

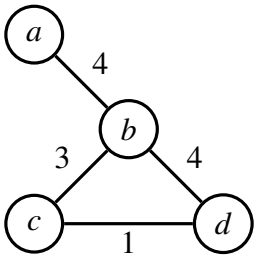
Dois grafos ponderados G e G' são vizinhos se possuem o mesmo conjunto de vértices e arestas, e se suas funções de peso diferem em uma unidade.

Definição 10 (Grafos ponderados vizinhos com topologia conhecida [Sealfon 2016])**.** *Considere $G = (V, E, \omega)$ e $G' = (V', E', \omega')$ dois grafos ponderados. G e G' são vizinhos se $V = V'$, $E = E'$ e $\omega \sim \omega'$.*

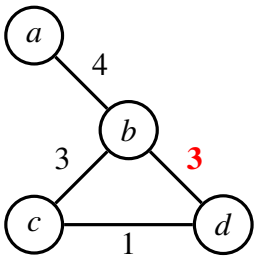
Vários trabalhos na literatura utilizam essas definições em seus estudos [Li et al. 2017, Pinot et al. 2018, Wang and Long 2019, Chen et al. 2022, Fan and Li 2022].

Para algumas aplicações do mundo real, a suposição de que a topologia do grafo é pública pode não ser verdadeira. Por exemplo, ao proteger a presença, ou ausência, de interações em uma rede de contatos entre dispositivos *IoT*, ou a existência, ou ausência, de chamadas telefônicas, mensagens de texto, ou a presença, ou ausência, de coautoria em um artigo. Esses tipos de interações não são cobertos pelas definições de Sealfon em termos de privacidade. Uma vez que uma aresta já é conhecida, qualquer mecanismo diferencialmente privado não mudará a presença, ou ausência, dessa conexão. Dessa forma, considerar apenas o cenário onde a topologia do grafo é publicamente conhecida não é eficaz para fornecer as garantias de privacidade desejadas.

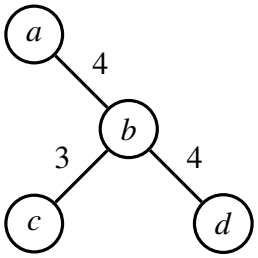
Para abordar essa limitação, o trabalho em [Brito et al. 2023] adapta a noção proposta por Sealfon para funções de peso vizinhas e fornece uma nova definição para grafos ponderados vizinhos com topologia desconhecida. Dessa forma, Seja $G = (V, E, \omega)$ um



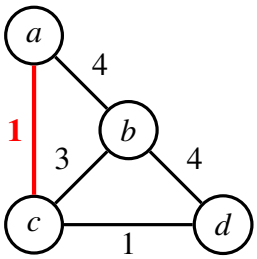
(a) Rede social ponderada G.



(b) Rede social vizinha G'_1 obtida a partir de G após a subtração de uma unidade de peso da aresta (b,d).



(c) Rede social vizinha G'_2 obtida a partir de G após a remoção da aresta (c,d) de peso igual a 1.



(d) Rede social vizinha G'_3 obtida a partir de G após a adição da aresta (a,c) de peso igual a 1.

Figura 4.7: Exemplo de uma rede social e suas possíveis redes sociais vizinhas de acordo com a noção *edge-weight* PD com topologia desconhecida.

grafo não direcionado e ponderado e uma função de peso $\omega : V^2 \to \mathbb{Z}_{\geq 0}$ que mapeia conexões (interações) entre um par de vértices (u,v) para pesos em G. O par $(u,v) \in E$ se os vértices u e v compartilham uma aresta comum, e $(u,v) \notin E$, caso contrário. Se $(u,v) \notin E$, então $\omega(u,v) = 0$. Como G é não direcionado, $\omega(u,v) = \omega(v,u)$.

Definição 11 (Funções de peso vizinhas com topologia desconhecida [Brito 2023])**.** *Duas funções de peso ω, $\omega' : V^2 \to \mathbb{Z}_{\geq 0}$ são vizinhas, denotado por $\omega \sim \omega'$, se:*

$$||\omega - \omega'||1 := \sum u, v \in V |\omega(u,v) - \omega'(u,v)| = 1. \tag{11}$$

A Definição 11 difere da Definição 9 no sentido de que agora os pesos podem assumir valores zero e os pesos também são valores inteiros. Assim, dois grafos G e G' são considerados vizinhos se tiverem o mesmo conjunto de vértices e se as funções de peso diferirem em uma unidade, conforme definição a seguir:

Definição 12 (Grafos ponderados vizinhos com topologia desconhecida [Brito 2023])**.** *Sejam $G = (V,E,\omega)$ e $G' = (V',E',\omega')$ dois grafos ponderados, G e G' são vizinhos se $V = V'$ e $\omega \sim \omega'$.*

A Figura 4.7 mostra um exemplo de três grafos ponderados vizinhos com topologia desconhecida a partir de uma rede G de entrada.

Com base nas definições apresentadas anteriormente, a definição formal de privacidade diferencial para grafos ponderados, considerando arestas e pesos como sendo informações privadas, é descrita como:

Definição 13 (ε-*Edge-weight* Privacidade Diferencial [Brito 2023])**.** *Um mecanismo M satisfaz ε-privacidade diferencial de pesos das arestas, se para qualquer par de grafos $G = (V, E, \omega)$ e $G' = (V', E', \omega')$, tais que G e G' são grafos de pesos vizinhos e para qualquer possível saída $O \subseteq Range(M)$,*

$$Pr[M(G) = O] \leq \exp(\varepsilon) \times Pr[M(G') = O]. \tag{12}$$

4.5.4. *Attributed*-Privacidade Diferencial

Em cenários reais, as redes sociais raramente são representadas apenas por nós e seus relacionamentos diretos. Em sua grande maioria, as redes sociais são representadas por uma complexa estrutura de dados, a qual consiste em uma fonte de informação extremamente rica. É bastante comum observarmos informações adicionais associadas à estrutura do grafo, sejam nos nós, ou nas arestas. A existência dessas informações, seja nas arestas, ou nos nós, consiste em informações valiosas para a compreensão e explicação de diversos comportamentos dos usuários. Assim, denominam-se esses grafos de grafos com atributos.

Grafos com atributos são uma classe particular de grafos nas quais informações adicionais, também chamadas de atributos, são anexadas à estrutura do grafo. Como antecipado anteriormente, os atributos podem estar associados tanto às arestas, quanto aos nós do grafo. Em relação à natureza dos atributos, estes possuem uma grande flexibilidade, podendo ser de qualquer natureza, seja numérica, categórica, lógica, dentre outras. Além disso, é importante destacar que os atributos são se limitam à uma única informação, de maneira que mais de um atributo pode estar relacionado às arestas, ou nós, do grafo.

Nesse contexto, os grafos passam a ser definido por $G = (V, E, X)$, onde V e E continuam sendo os conjuntos de vértices (nós) e arestas, respectivamente. Já o novo conjunto, denotado por X, consiste no conjunto de atributos associados à estrutura do grafo. Em um grafo com atributos nas arestas, X representará o conjunto dos atributos associados a cada aresta de E. Analogamente, em um grafo com atributos nos nós, X representará o conjunto dos atributos associados a cada nó de V.

Os grafos com atributos nas arestas têm sido amplamente adotados em diversos campos para explicar os motivos que levam os usuários a possuírem conexões entre si. Em redes sociais dessa natureza, o tipo de relacionamento entre os usuários da rede é de extrema importância. Alguns exemplos incluem redes de comunicação [Wang et al. 2013], redes de coautoria [Alsmadi and Alhami 2015] e redes de informação heterogênea [Shi et al. 2016]. Assim, o estudo de grafos com atributos nas arestas transformou-se em uma área de pesquisa próspera, tornando-se relevante para diversas aplicações como: detecção de anomalias [Shah et al. 2016], análise de mobilidade [Kaytoue et al. 2017] e pesquisa de comunidade [Li et al. 2023].

A Figura 4.8 apresenta um exemplo de um grafo com atributos nas arestas, onde os nós representam os colaboradores de uma empresa e as arestas afirmam a existência de

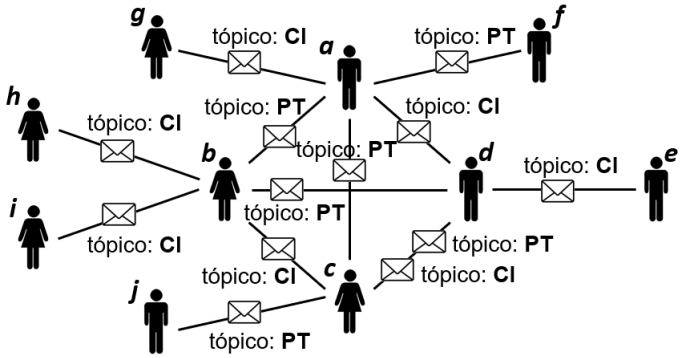


Figura 4.8: Grafo com atributos nas arestas, onde os nós representam os usuários e as arestas os *e-mails* trocados entre os usuários. O tópico dos *e-mails* trocados representa o atributo da aresta, onde "CI" denota assuntos relacionados a Comunicação Interna e "PT" retrata assuntos relacionados a Problemas Técnicos.

e-mails trocados entre dois colaboradores, juntamente com o tópico do *e-mail*. Os tópicos dos *e-mails* limitam-se a Comunicação Interna (CI) e Problemas Técnicos (PT), sendo detalhados a seguir.

- **Comunicação Interna (CI):** Utilizado para a comunicação interna da equipe, incluindo atualizações de projetos, reuniões e informações em geral.
- **Problemas Técnicos (PT):** Utilizado para questões relacionadas a problemas técnicas, onde problemas com plataformas de serviços e utilização de *softwares* são os mais comuns.

Por sua vez, redes sociais com atributos nos nós são bastante estudadas em tarefas de predição, detecção de padrões, descoberta de subgrafos e nós discrepantes, as quais levam em consideração os nós com atributos semelhantes. A existência e descoberta de nós com atributos semelhantes está fortemente relacionada ao conceito de homofilia da rede. A homofilia consiste na tendência de nós com atributos semelhantes se conectarem entre si [McPherson et al. 2001]. Em cenários reais, a homofilia tende a ser bastante presente, visto que há uma preferência entre os usuários por manterem relacionamentos com outros usuários que possuem características semelhantes às suas. A Figura 4.9 apresenta um exemplo de um grafo com atributos nos nós. O grafo em questão é composto por 7 nós, ou usuários, e 3 atributos, os quais são: gênero (*M*: masculino; *F*: feminino), idade e altura, sendo o primeiro atributo do tipo categórico e os demais do tipo numérico. Por fim, as arestas presentes na rede conectam dois usuários que são amigos entre si.

No entanto, devido às características inerentes às redes sociais com atributos, os modelos de privacidade diferencial para redes sociais apresentados anteriormente não são suficientemente robustos para capturar essas propriedades e, simultaneamente, prover as garantias de PD. Por mais que um uma rede social ponderada, apresentada na Seção 4.5.3, possa ser vista como uma rede social com atributos nas arestas, visto que os atributos são de natureza numérica, estes apresentam funções diferentes. Em um grafo ponderado, a informação da aresta, também chamada de peso, tem uma importância diferente de acordo com o seu valor. Geralmente, quanto maior o peso, maior a importância da aresta.

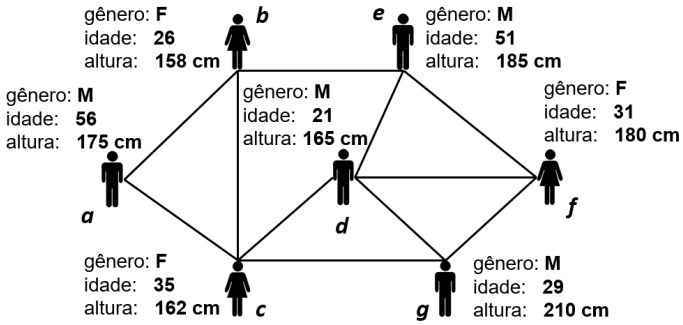


Figura 4.9: Grafo com atributos (gênero, idade e altura) associados aos nós, ou usuários. As arestas representam os relacionamentos entre usuários.

Entretanto, em um grafo com atributos nas arestas, um atributo de natureza numérica desempenhará a função de atributo categórico. Em outras palavras, dadas duas arestas com atributos numéricos de valores diferentes, ambas terão a mesma importância, mesmo que uma aresta tenha um valor muito mais elevado, ou inferior, em relação à outra aresta. Portanto novos modelos de PD para redes sociais com atributos precisam ser propostos a fim de prover as devidas garantias de PD. A seguir, são apresentadas as noções de PD mais utilizadas para redes sociais com atributos nas arestas e nos nós, respectivamente.

Proposta por [Liu et al. 2020], a *attribute-wise* privacidade diferencial (*attribute-wise* PD) consiste na noção de privacidade mais aplicada para grafos com atributos nas arestas. Sua definição é apresentada abaixo:

Definição 14 (ε-*Attribute-wise* Privacidade Diferencial)**.** *Um mecanismo M satisfaz ε-attribute-wise privacidade diferencial se, para qualquer par de grafos com atributos nas arestas vizinhos G e G' diferindo em um atributo e todas as arestas relacionadas ao atributo, e para qualquer saída possível $O \subseteq Range(M)$,*

$$\Pr[M(G) = O] \leq \exp(\varepsilon) \times \Pr[M(G') = O]. \tag{13}$$

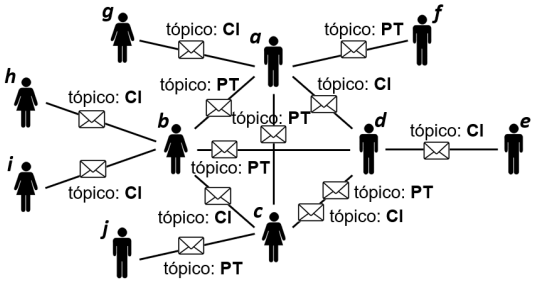
A Figura 4.10 apresenta um exemplo de grafo G com atributos nas arestas e dois possíveis grafos vizinhos G'_1 e G'_2 obtidos a partir de G, conforme a Definição 14. A Figura 4.10a apresenta o grafo original G, enquanto as Figuras 4.10b e 4.10c apresentam os grafos vizinhos G'_1 e G'_2 obtidos, respectivamente, a partir da remoção dos atributos "CI" e "PT" do grafo G.

Por sua vez, [Jorgensen et al. 2016] propôs uma nova noção de privacidade para redes sociais com atributos nos nós, denominada *edge-adjacent attributed* privacidade diferencial (*edge-adjacent attributed* PD). Sua definição é apresentada abaixo:

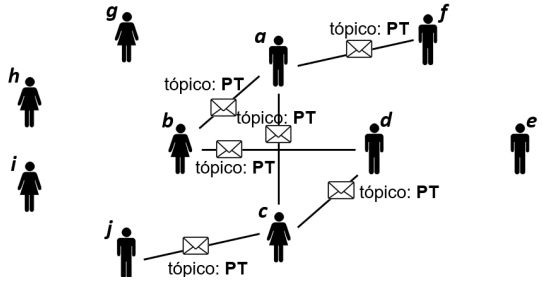
Definição 15 (*Edge-adjacent attributed graphs*)**.** *Dois grafos com atributos nos nós G e G' são ditos edge-adjacent (ou vizinhos) se diferirem em uma única aresta ou nos atributos associados a um único nó.*

Definição 16 (ε-*Edge-adjacent attributed* Privacidade Diferencial)**.** *Um mecanismo M satisfaz ε-edge-adjacent attributed privacidade diferencial se, para qualquer par de grafos edge-adjacent G e G', e para qualquer saída possível $O \subseteq Range(M)$,*

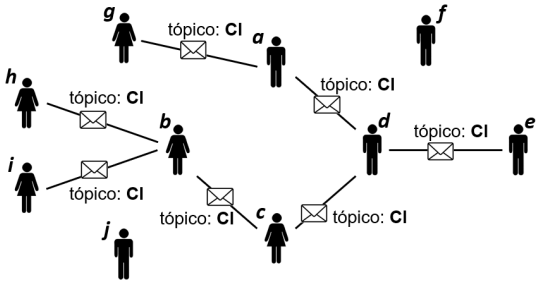
$$\Pr[M(G) = O] \leq \exp(\varepsilon) \times \Pr[M(G') = O]. \tag{14}$$



(a) Rede social G.



(b) Rede social vizinha G'_1 obtida a partir de G após a remoção das arestas com atributo "CI".



(c) Rede social vizinha G'_2 obtida a partir de G após a remoção das arestas com atributo "PT".

Figura 4.10: Exemplo de uma rede social com atributos nas arestas e suas redes sociais vizinhas de acordo a noção *attribute-wise* PD.

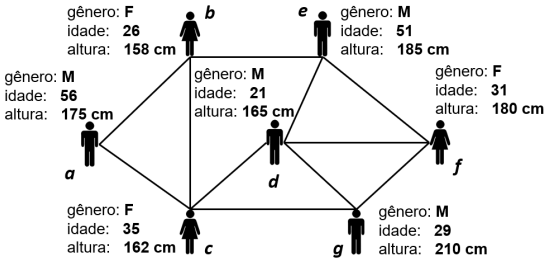
A Figura 4.11 apresenta um exemplo de uma rede social com atributos nos nós G e duas possíveis redes sociais vizinhas G'_1 e G'_2 obtidas a partir de G, conforme a Definição 16. A Figura 4.11a apresenta a rede social G, enquanto as Figuras 4.11b e 4.11c apresentam as redes sociais vizinhas G'_1 e G'_2, respectivamente. Na Figura 4.11b, a rede foi obtida após a remoção da aresta (c, g) da rede G, enquanto que a rede da Figura 4.11c foi obtida após a modificação dos atributos idade e altura do usuário c da rede G.

4.5.5. Definições Complementares de Privacidade Diferencial para Redes Sociais

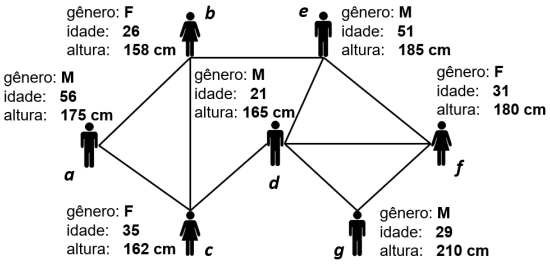
Nas seções anteriores, foram apresentadas várias noções de PD para redes sociais sob diferentes contextos: *edge*-PD, *node*-PD, *edge-weight* PD e *attributed*-PD, como as principais noções no que diz respeito a prover garantias de PD em redes sociais. No entanto, existem diversas noções adicionais de PD que ainda não encontraram uma aplicação muito difundida e são, na sua maioria, derivadas das noções apresentadas anteriormente. Mais detalhes sobre algumas definições e variações adicionais de PD são apresentadas nesta seção.

***Out-link* Privacidade Diferencial:** A primeira dessas novas noções de PD para redes sociais é a *out-link* privacidade diferencial (*out-link* PD) [Task and Clifton 2014]. Para esse contexto, são consideradas redes sociais direcionadas, onde é possível distinguir as arestas que entram e saem dos nós. Sob essa noção, dois grafos são considerados vizinhos se todas as arestas de saída de um nó arbitrário forem adicionadas, ou removidas. A definição formal de *out-link* PD é apresentada abaixo:

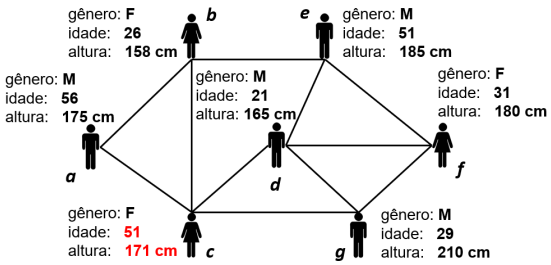
Definição 17 (*Out-link* Privacidade Diferencial)**.** *Dois grafos direcionados são ditos vizinhos se diferirem em todos os out-links (arestas que saem de um nó) de um nó arbitrário.*



(a) Rede social G.



(b) Rede social vizinha G'_1 obtida a partir de G após a remoção da aresta (c, g).



(c) Rede social vizinha G'_2 obtida a partir de G após a modificação dos atributos idade e altura do usuário c.

Figura 4.11: Exemplo de uma rede social com atributos nos nós e suas redes sociais vizinhas de acordo a noção *edge-adjacent attributed* PD.

As garantias de privacidades providas pela *out-link* PD são estritamente mais fracas que as garantias da *node*-PD. Entretanto, em muitos cenários, a *out-link* PD é comparável à *edge*-PD. Sob essa noção de PD, um invasor mal-intencionado não seria capaz de determinar se um usuário contribuiu com seus dados para a construção da rede social com suas respectivas arestas de saída. Em cenários reais, por exemplo, um usuário pode recusar amizades. Por sua vez, outros usuários podem alegar que são amigos de um determinado usuário, mas este último pode negar que essas amizades existem. Dessa forma, os autores argumentam que a *out-link* PD simplifica o cálculo da sensibilidade e reduz a magnitude do ruído adicionado, permitindo consultas que seriam inviáveis sob as definições de *edge*-PD.

***Graph*-Privacidade Diferencial:** A *graph*-privacidade diferencial (*graph*-PD) [Mueller et al. 2022] consiste em uma outra noção de PD para o contexto de redes sociais. Diferente das outras noções apresentadas, a *graph*-PD é uma noção de privacidade mais recente, a qual se aplica tanto à análise de dados em redes sociais, quanto à redes neurais para grafos (GNNs). No entanto, por ser uma noção bem mais recente, a mesma não foi tão explorada ainda. A sua definição formal é apresentada abaixo:

Definição 18 (*Graph*-Privacidade Diferencial)**.** *Dois multigrafos são ditos vizinhos se diferirem em um único grafo, obtido através da adição ou remoção de um gráfico inteiro.*

Apesar de ser uma noção de privacidade recente, a *graph*-PD possui uma grande versatilidade, adaptando-se a diversos contextos diferentes. Dentre os quais a mineração de padrões frequentes e o treinamento de GNNs se destacam em multigrafos. Em resumo, um multigrafo consiste em uma grafo no qual múltiplas arestas entre os mesmos nós são permitidas, de maneira que dois nós podem ser conectados por mais de uma aresta. É

importante destacar que as noções de PD aplicadas à redes sociais não se limitam às que foram apresentadas nesta seção.

4.6. Mecanismos de Privacidade Diferencial para Redes Sociais

Esta seção apresenta uma visão geral dos métodos e abordagens que podem ser aplicados na análise de dados em redes sociais utilizando privacidade diferencial.

4.6.1. Análise Privada para Publicação de Redes Sociais e Redes Sociais Aleatórias

O compartilhamento diferencialmente privado de grafos completos tem sido um segmento de pesquisa estudado extensivamente nos últimos anos. A principal vantagem dessa abordagem é que ela é independente do tipo de análise, de maneira que é possível realizar qualquer tipo de consulta estatística sobre o grafo disponibilizado. Neste cenário, o modelo Pygmalion [Sala et al. 2011] foi proposto com o objetivo de disponibilizar a topologia do grafo sob as garantias de privacidade do modelo *edge*-PD através da extração da estrutura detalhada do grafo em uma versão privada de um dK-grafo [Mahadevan et al. 2006] para, então, gerar um grafo sintético. Posteriormente, melhorias foram propostas na utilidade do dK-grafo por meio de uma melhor calibração do ruído através da sensibilidade *smooth* [Wang and Wu 2013]. Os autores primeiro derivaram do grafo original parâmetros como correlações de grau, e os utilizaram no modelo de dK-grafo, garantindo a privacidade diferencial das arestas nos parâmetros aprendidos para gerar grafos sintéticos. Nos últimos anos, foi desenvolvido um *framework* baseado em microagregação para a anonimização de grafos, denominado *dK-microaggregation*, o qual reduz a magnitude do ruído ao adicionar uma etapa de microagregação ao dK-grafo antes de adicionar o ruído de Laplace [Iftikhar et al. 2020]. Este *framework* é ilustrado na Figura 4.12.

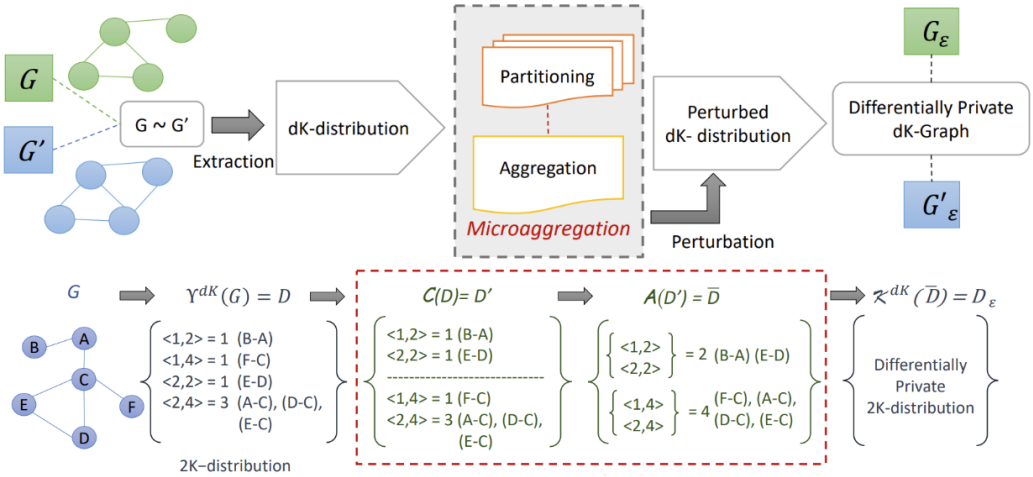


Figura 4.12: *Framework* baseado em microagregação para a anonimização de redes sociais [Iftikhar et al. 2020].

Através de uma abordagem diferente [Xiao et al. 2014], um novo modelo para representação de grafos, denominado *Hierarchical Random Graph* (HRG) [Clauset et al. 2006], foi proposto para a publicação de dados de grafos. Os autores do modelo observaram que, ao estimar as probabilidades de conexão entre nós, a escala de ruído imposta

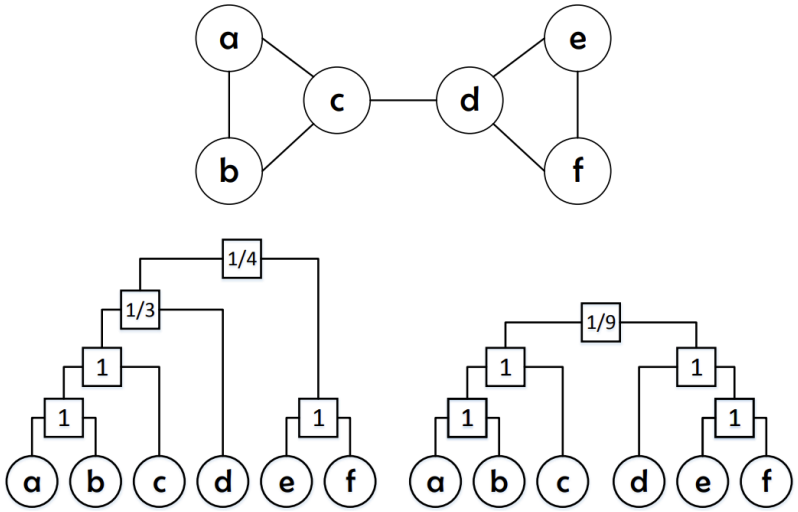


Figura 4.13: Exemplo de um grafo original e dois dendrogramas [Xiao et al. 2014].

pela privacidade diferencial poderia ser reduzida. Modelos clássicos de grafos são construídos considerando as arestas observadas, enquanto o HRG utiliza as probabilidades de conexão entre vértices para formar dendrogramas. Um exemplo de dendrogramas a partir de um grafo de entrada é ilustrado na Figura 4.13.

Adicionalmente, algumas abordagens focam na perturbação da matriz de adjacência original, o que permite representar um grafo e adotar estratégias de perturbação de matriz. O trabalho em [Chen et al. 2014] apresenta um mecanismo de exploração e reconstrução baseado em densidade (DER) para compartilhar a matriz de adjacência de um grafo. No entanto, tanto o HRG quanto o DER possuem complexidade quadrática em relação ao número de nós. Por sua vez, o algoritmo *Top-m Filter* (TmF) [Nguyen et al. 2015] foi proposto com o objetivo de solucionar os problemas de escalabilidade existentes em trabalhos anteriores. O método consiste em adicionar ruído de Laplace a cada célula da matriz de adjacência e utiliza uma ideia semelhante ao algoritmo *High-pass Filter* [Cormode et al. 2012] para evitar a materialização da matriz de adjacência ruidosa.

Os trabalhos mencionados acima utilizam *edge*-PD como modelo de privacidade diferencial para disponibilizar grafos inteiros e aleatórios. No entanto, devido às dificuldades em obter mecanismos privados que proveem alta utilidade de dados após a publicação do grafo completo, há evidências de apenas um único trabalho recente [Jian et al. 2021] relacionado ao modelo de privacidade *node*-PD quando comparado ao modelo de privacidade *edge*-PD. Os autores propõem um algoritmo de perturbação de nós para alcançar o modelo *node*-PD adicionando e removendo nós aleatoriamente. Primeiro, remove-se aleatoriamente cada nó no grafo de entrada de forma independente com probabilidade p. Em seguida, gera-se um número aleatório k, que segue uma distribuição geométrica com probabilidade de sucesso q. Logo, adiciona-se aleatoriamente k nós ao grafo, de modo que cada um desses k nós seja conectado a todos os nós existentes com uma probabilidade de 0,5. Após essa etapa, o número de nós nos grafos resultantes será indistinguível se os grafos de entrada forem grafos vizinhos.

4.6.2. Privacidade em Contagem de Subgrafos

Contagens de subgrafos são outras estatísticas amplamente estudadas no âmbito da análise de dados em grafos. Consultas dessa natureza realizam contagens sobre o número de vezes que uma determinada estrutura de subgrafo aparece em um grafo. Dentre os subgrafos mais comuns, podemos citar os triângulos, as estrelas e os cliques. Nesse contexto, [Karwa et al. 2011] estendeu os resultados da sensibilidade *smooth* [Nissim et al. 2007] para publicar consultas de k-estrelas e k-triângulos de maneira diferencialmente privada. Especificamente, os autores apresentaram um algoritmo para calcular a sensibilidade *smooth* do número de k-estrelas em um grafo de entrada e utilizaram uma abordagem diferente, baseada na sensibilidade local, a fim de fornecer um algoritmo diferencialmente privado para o compartilhamento de contagens de subgrafos. Outra técnica utilizada, a função *ladder* [Zhang et al. 2015] é utilizada para obter uma alta precisão de contagem de subgrafos com complexidades de tempo eficientes. Essa abordagem combina efetivamente o conceito de sensibilidade local à distância t, do *framework* de sensibilidade *smooth* com o mecanismo exponencial, de maneira a permitir, também, a contagem de k-cliques em um grafo.

Nos últimos anos, uma nova noção de privacidade diferencial, denominada *Decentralized Differential Privacy* (DDP) [Sun et al. 2019], foi apresentada juntamente com uma técnica para publicar, de maneira privada, algumas estatísticas de grafos, tais como triângulos, caminhos *three-hop* e contagens de k-cliques através da privacidade diferencial local, ao passo em que garante as propriedades impostas pela DDP. Recentemente, os autores em [Imola et al. 2021] apresentaram novos algoritmos para contagem de subgrafos com privacidade diferencial local que utilizam uma rodada adicional de interação entre os usuários e o curador de dados. Os autores melhoraram os resultados recentes tanto para consultas de contagem de triângulos quanto de k-estrelas.

4.6.3. Privacidade em Histogramas de Graus

Outra estatística de grafo bastante estudada é a sequência de graus de um grafo (*degree sequence*). Em resumo, a sequência de graus consiste em uma lista que contém os graus, ou seja, número de conexões, de cada um dos nós do grafo, geralmente ordenada de maneira decrescente. A sequência de graus torna-se uma estatística bastante robusta quando aplicada, principalmente, em análises envolvendo as distribuições dos graus de um grafo.

Sob o modelo de *edge*-PD, [Hay et al. 2009] propõem uma técnica baseada em inferência de restrições para disponibilizar sequências de graus via mecanismos de privacidade diferencial. Os autores adaptaram a definição de PD para dados estruturados em grafos e foram os primeiros a introduzir a noção de *edge*-PD. O método proposto baseia-se em adicionar ruído às contagens de grau de cada nó. Especificamente, para cada nó, uma pequena quantidade de ruído de Laplace é adicionada à sua contagem de grau verdadeira. Os autores também realizaram uma etapa de pós-processamento nas respostas ruidosas para inferir um resultado mais preciso. Um problema com essa inferência é que a sequência de graus pode não ser realizável, ou seja, não ser possível construir um grafo sem laços ou múltiplas arestas entre o mesmo par de vértices com a sequência de graus retornada pelo mecanismo diferencialmente privado. Para superar isso, [Karwa and Slavković 2012] introduziram uma etapa de otimização após a inferência de restrições. Os

autores propuseram uma abordagem inovadora para gerar grafos sintéticos que satisfazem a privacidade diferencial utilizando um modelo probabilístico baseado na sequência de graus realizável, ou seja, um vetor que descreve a distribuição de graus de um grafo. Eles também incluíram uma etapa adicional de pós-processamento para garantir que a sequência de graus liberada seja realizável.

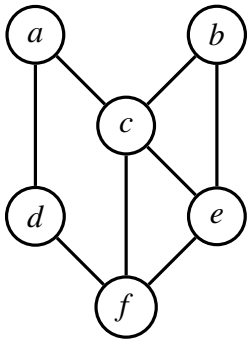
Já no cenário de privacidade diferencial de nós (*node*-PD), os autores em [Kasiviswanathan et al. 2013] discutem vários algoritmos diferencialmente privados voltados para a análise da histogramas de graus. A ideia principal por trás de suas técnicas é projetar o grafo de entrada no conjunto de grafos com um grau máximo abaixo de um determinado limite, denominado θ. No entanto, o desafio dessa abordagem é que a operação de projeção pode ser muito sensível a uma mudança de um único nó no grafo original.

Por outro lado, os autores em [Day et al. 2016] propõem duas abordagens baseadas, respectivamente, em agregação e histograma cumulativo para publicar a distribuição de graus. Ambas as abordagens adotam um novo método de projeção de grafos que é baseado em um processo de adição de arestas. Os autores provaram que publicar um histograma de graus a partir do grafo projetado tem sensibilidade $2\theta + 1$, e publicar um histograma cumulativo de graus tem sensibilidade $\theta + 1$. Esse processo transforma um grafo em um grafo limitado a θ graus. A escolha ideal de θ depende tanto do conjunto de dados quanto do orçamento de privacidade. Em ambas as abordagens, os autores também utilizam uma etapa adicional de pós-processamento para melhorar a precisão dos histogramas. A Figura 4.14 mostra um exemplo de projeção de grafos, para $\theta = 3$ (Figura 4.14b) e para $\theta = 2$ (4.14c), respectivamente.

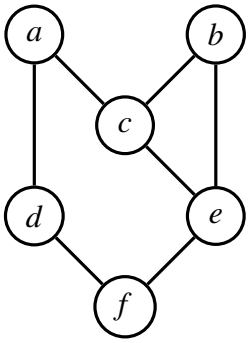
4.6.4. Análise Privada de Medidas de Centralidade

Na análise de dados em redes sociais, estatísticas mais complexas surgem como uma necessidade fundamental, possibilitando a mineração de topologias complexas e, também, a compreensão das interações e relacionamentos entre os indivíduos presentes na rede. Nesse contexto de estatísticas mais complexas, aparecem as medidas de centralidade, as quais são bastante relevantes em mensurar a importância dos nós em um grafo. Tais informações podem ser amplamente aplicadas em análises que envolvem partes de uma rede que necessitam de maiores atenções ou cuidados.

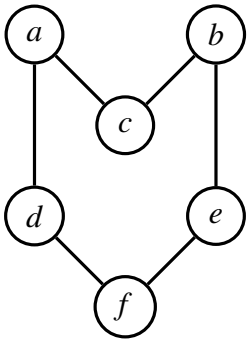
Nesse contexto, o método *PrivateEBC* [Roohi et al. 2019] foi proposto visando resolver o problema relacionado à centralidade de intermediação (*betweenness centrality*) dos nós quando informações relevantes sobre as arestas são disseminadas. Os autores desenvolvem um protocolo que permite calcular a centralidade de intermediação egocêntrica de um nó em um grafo de maneira privada, onde as informações relevantes sobre as arestas estão distribuídas entre duas partes que não se confiam mutuamente, como dois provedores de telecomunicações. Uma outra estratégia que adapta a noção de sensibilidade local para o contexto não numérico [Farias et al. 2020,Farias et al. 2023] é proposta a fim de desenvolver um mecanismo genérico para dados categóricos e disponibilizar informações de centralidade de intermediação utilizando privacidade diferencial. O trabalho busca estender os princípios da privacidade diferencial, tradicionalmente aplicados a dados numéricos, para contextos onde os dados são de natureza não numérica, oferecendo uma abordagem que assegura a privacidade enquanto lida com esse tipo de informação.



(a) Rede social G.



(b) Exemplo de rede social projetada G'_1 com $\theta = 3$.



(c) Exemplo de rede social projetada G'_2 com $\theta = 2$.

Figura 4.14: Exemplo de uma rede social e suas possíveis projeções para diferentes valores de θ.

Por outro lado, o trabalho em [Laeuchli et al. 2022] apresenta limites inferiores e superiores com base na abordagem de sensibilidade *smooth* para centralidade de autovetor, de Laplace e de proximidade. Os autores investigam como a adição de ruído para preservar a privacidade afeta a precisão das medidas de centralidade. No entanto, os autores concluem que a abordagem de sensibilidade *smooth* é inviável ou impraticável.

4.6.5. Detecção de Comunidades de Maneira Privada

Diferentemente das medidas de centralidade, onde o foco de atenção maior era sobre a importância dos indivíduos, uma técnica de detecção de comunidades busca identificar similaridades sobre grupos de indivíduos. Em resumo, a detecção de comunidades consiste em técnicas aplicadas com o objetivo de identificar grupos em redes, geralmente complexas, de acordo com as propriedades estruturais da rede em questão. A expectativa é de que, dado que um grafo pode ser dividido em diversos conjuntos de nós disjuntos, os nós internos de cada um desses conjuntos são densamente conectados entre si.

Nesse contexto, o método LouvainDP [Nguyen et al. 2016], um algoritmo popular para detecção de comunidades, é utilizado como base para a adaptação sob as restrições da privacidade diferencial. Um exemplo do algoritmo de Louvain é demonstrado na Figura 4.15. Para que o algoritmo seja diferencialmente privado, os autores adicionam ruído de Laplace aos pesos das arestas gerados pelo algoritmo de Louvain. A vantagem dessa

estratégia é que ela é linear, em termos de complexidade de algoritmos, em relação ao número de arestas do grafo original. Além disso, os autores incluem uma análise detalhada das dificuldades inerentes à aplicação da privacidade diferencial aos algoritmos de detecção de comunidades.

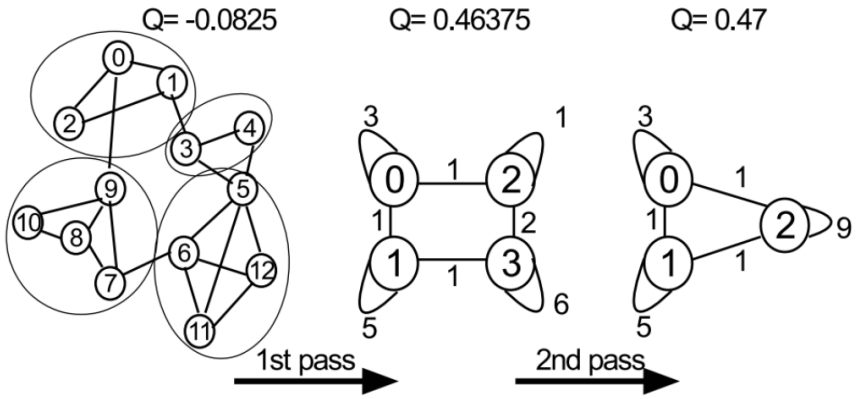


Figura 4.15: Exemplo de detecção de comunidades com o método de Louvain.

Já os autores em [Mohamed et al. 2022] desenvolvem métodos que permitem a detecção de comunidades em redes utilizando o modelo de bloco estocástico (*Stochastic Block Model - SBM*), enquanto garantem a privacidade diferencial. O estudo explora como diferentes configurações do modelo de bloco estocástico afetam o desempenho dos métodos de detecção de comunidades sob privacidade diferencial. Além disso, o mecanismo XOR [Ji et al. 2021] apresenta resultados promissores na detecção de comunidades dada a topologia da rede. Esse novo mecanismo é projetado para fornecer respostas privadas a consultas sobre dados binários e dados em formato de matriz (grafos). O mecanismo perturba os bits dos dados originais, aplicando a operação XOR com um vetor de ruído. Durante uma consulta, os bits perturbados são utilizados para calcular a resposta. A operação XOR garante que a privacidade dos indivíduos é preservada, mesmo que os dados perturbados sejam analisados. Os autores estendem o mecanismo para dados em formato de matriz, onde cada elemento da matriz é perturbado de maneira semelhante aos bits binários, aplicando a operação XOR com um vetor de ruído específico para matrizes.

Por sua vez, o algoritmo DPCD (*Differentially Private Community Detection*) [Ji et al. 2019] consiste em um algoritmo capaz de proteger tanto a topologia da rede quanto os atributos dos nós na detecção de comunidades em redes sociais. A estratégia baseia-se em um modelo probabilístico generativo que realiza a detecção de comunidades resolvendo um problema de máxima verossimilhança, com garantias de privacidade diferencial. Em particular, os autores dividem o problema de máxima verossimilhança em subproblemas convexos, cada um lidando com os relacionamentos e os atributos de um usuário específico. Para proteger os relacionamentos privados de cada usuário, a função objetivo referente às suas relações é perturbada por um ruído injetado com uma distribuição de probabilidade projetada. Para proteger a privacidade dos atributos dos usuários, cada usuário é obrigado a gerar ruído independentemente, enquanto a soma desses ruídos satisfaz a distribuição projetada.

4.6.6. Análise Privada de Caminhos Mínimos e Distâncias

Quando os grafos possuem arestas ponderadas, os modelos de privacidade diferencial para arestas e nós mencionados anteriormente podem não oferecer garantias de privacidade adequadas. Em vez disso, um ajuste mais apropriado é adaptar a definição de privacidade diferencial no contexto de grafos ponderados.

Sealfon [Sealfon 2016] propôs fundamentos teóricos para considerar a privacidade diferencial dos pesos em um grafo. Seu estudo visa compartilhar caminhos mínimos ponderados entre pares de nós e compartilhar distâncias aproximadas entre todos os pares de nós sem revelar informações sensíveis sobre os pesos das arestas. Para o problema de liberar caminhos mais curtos de forma privada, o autor apresentou um limite inferior robusto baseado em reconstrução, mostrando que não é possível disponibilizar um caminho mínimo entre um par de vértices com erro adicionado melhor que $\Omega(|V|)$, sob privacidade diferencial. Esse limite inferior é obtido ao reduzir o problema de reconstruir muitas das linhas de um banco de dados ao problema de encontrar um caminho com baixo erro. O autor também mostrou que um algoritmo que utiliza o mecanismo de Laplace chega perto de atingir esse limite. Já considerando o problema de compartilhar caminhos mais curtos ponderados entre todos os pares de nós com privacidade diferencial, o autor argumentou que técnicas padrão produzem erro de $O(|V|log|V|)/\varepsilon$ para cada consulta diferencialmente privada. Por outro lado, ele obteve algoritmos melhorados para duas classes especiais de grafos: (1) árvores e (2) grafos com pesos de arestas limitados.

[Fan and Li 2022] revisitou o problema de disponibilizar distâncias aproximadas entre todos os pares de nós de maneira privada e melhorou os resultados de Sealfon. Os autores primeiro dividiram uma árvore em caminhos pesados disjuntos. Em um caminho pesado, cada nó não-folha seleciona um ramo, ou seja, a aresta para o filho que tem a maior profundidade. As arestas selecionadas formam um caminho pesado. A Figura 4.16 exemplifica três caminhos pesados na árvore de entrada.

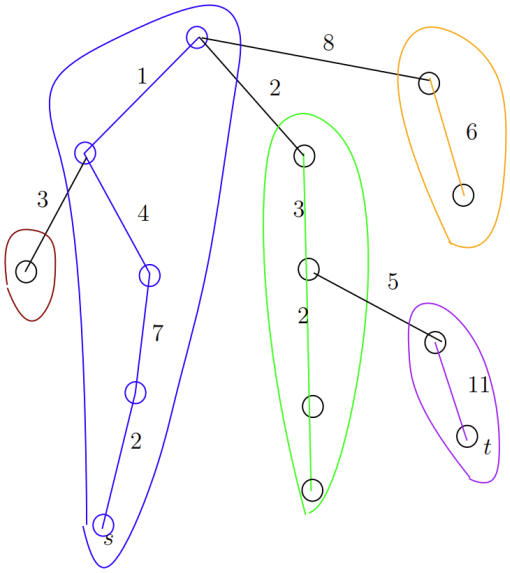


Figura 4.16: Um exemplo de uma árvore particionada em três caminhos pesados. Uma cor única é atribuída a cada caminho pesado [Fan and Li 2022].

Como a árvore é decomposta em um conjunto de caminhos, algumas arestas podem não estar incluídas em nenhum dos caminhos pesados produzidos durante este método. Já o trabalho em [Brito et al. 2023] estabelece uma nova definição de grafos vizinhos considerando tanto a topologia do grafo quanto os pesos das arestas como informações privadas. Os autores fornecem uma solução escalável de perturbação de grafos ponderados e introduzem tanto uma abordagem global quanto uma abordagem utilizando privacidade diferencial local. O modelo de privacidade utilizado é o *edge-weight* privacidade diferencial (*edge-weight* PD). Os autores obtém resultados significativos em várias métricas relacionadas a caminhos mínimos e distâncias, como soma dos pesos totais das arestas, média de caminhos mínimos, soma dos pesos das arestas adjacentes a um nó, entre outros. A Figura 4.17 ilustra um exemplo de como essa abordagem funciona.

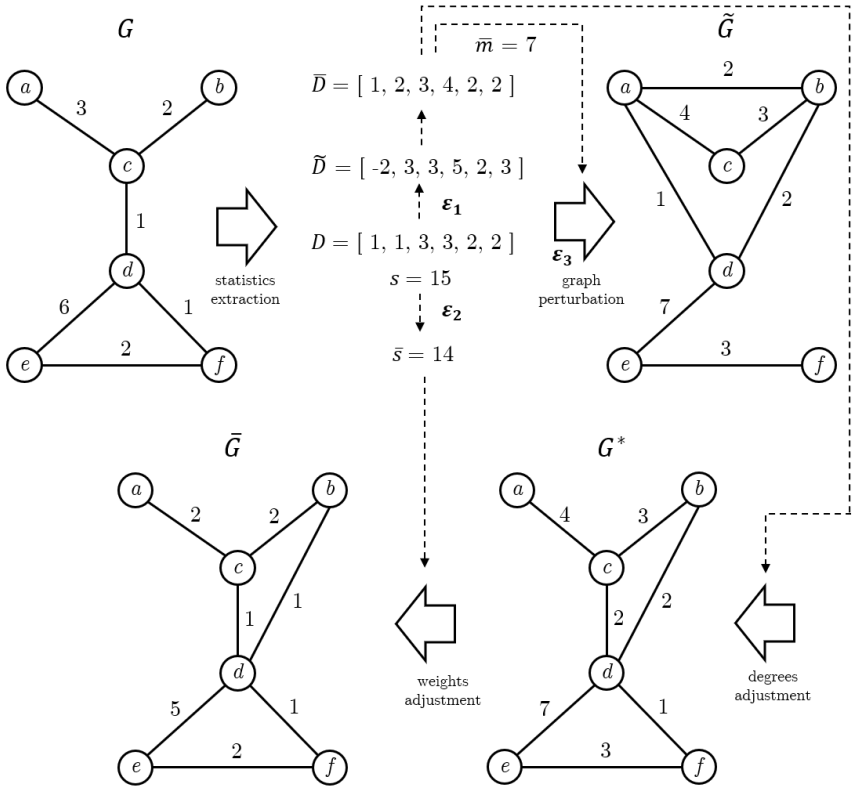


Figura 4.17: Exemplo de compartilhamento de um grafo ponderado utilizando *edge-weight* PD.

4.6.7. Privacidade em Redes Neurais para Redes Sociais

O recente sucesso das redes neurais impulsionaram pesquisas sobre reconhecimento de padrões e mineração de dados. Diversas tarefas de aprendizado de máquina, como detecção de objetos, tradução automática e reconhecimento de fala, ganharam bastante atenção através dos paradigmas de aprendizado profundo, como as redes neurais convolucionais (RNC), redes neurais recorrentes (RNR) e codificadores automáticos. No entanto, os métodos de aprendizado profundo se mostram adequados para identificar padrões sobre dados euclidianos, como imagens, textos e vídeos, inviabilizando sua aplicação sobre dados gerados a partir de domínios não euclidianos, como as estruturas de grafos, as quais

são utilizadas para modelar complexos relacionamentos e interdependências entre entidades. Por conta das limitação dos modelos existentes, recentemente, as redes neurais para grafos (GNN) receberam atenção significativa devido, principalmente, à sua capacidade de capturar relacionamentos complexos e dependências entre os nós de um grafo. As GNN pertencem à uma classe de redes neurais que podem ser aplicadas sobre estruturas de grafos. Dentre os tipos de aplicações mais conhecidas, pode-se mencionar tarefas relacionadas à classificação de nós e grafos, visualização de grafos, predição de conexões, agrupamento de grafos, dentre outras.

No entanto, por se tratarem de um conjunto de aplicações que fazem uso direto dos dados, questões quanto à privacidade dos indivíduos tornam-se novamente aparentes. Nesse contexto, algumas técnicas diferencialmente privadas começaram a surgir. Dentre as primeiras técnicas a surgirem, o *framework* PrivGnn permite a proteção de dados sensíveis mesmo após a publicação de um modelo de redes neurais para grafos. Por sua vez, [Daigavane et al. 2021] propôs um esquema de amostragem de vizinhança de grafos ao passo em que assegura o modelo de privacidade *node*-PD. No entanto, todos os trabalhos recém mencionados reforçam a privacidade somente durante a etapa de treinamento e/ou liberação do modelo. Esse fato pode colocar informações acerca dos indivíduos em sérios riscos, caso a parte interessada seja maliciosa. Diante dessa limitação, recentemente, foi proposto o *framework* RGNN [Bhaila et al. 2023], o qual é baseado na reconstrução e aprendizado de redes neurais para grafos e garante a privacidade dos nós do grafo através de privacidade diferencial local.

4.7. Conclusão

A análise de dados privados em redes sociais é um desafio essencial no contexto atual, onde a preservação da privacidade dos usuários é crucial. Realizar análises de dados de maneira privada é fundamental para manter a privacidade dos usuários e proteger informações sensíveis contra acessos indevidos. A evolução dos modelos de privacidade reflete o progresso nesta área, começando com modelos sintáticos e avançando para abordagens mais robustas como a privacidade diferencial.

Os modelos sintáticos, como a anonimização e a generalização, inicialmente tentaram proteger a privacidade substituindo, ou ocultando, informações que identificavam unicamente os usuários. No entanto, a reidentificação através de ataques complexos demonstrou as limitações desses modelos, revelando a necessidade de técnicas mais avançadas. Nesse contexto, a privacidade diferencial emergiu como como o novo padrão para a proteção de dados, oferecendo uma abordagem matemática robusta para quantificar e limitar os riscos de privacidade.

A privacidade diferencial apresenta duas configurações principais: a global e a local. Na configuração global, o ruído é adicionado aos dados, ou resultados das análises, através de um curador centralizado, o qual detém a posse dos dados. Em contrapartida, na configuração local, o controle dos dados é colocado sob responsabilidade dos próprios usuários, garantindo que os dados sejam protegidos no momento da coleta, antes mesmo de serem enviados ao curador de dados, o qual é considerado não confiável. Cada uma das configurações apresenta suas vantagens e desvantagens, sendo a escolha dependente do contexto e das necessidades específicas de privacidade e precisão das análises.

Além das considerações de privacidade, é importante reconhecer a diversidade das redes sociais. Existem desde as redes sociais mais simples, constituídas apenas por nós e arestas, onde cada nó representa um usuário e cada aresta representa um relacionamento entre os usuários. No entanto, também existem redes sociais bem mais complexas como, por exemplo, as redes sociais ponderadas, onde as conexões têm pesos, os quais representam a intensidade, ou frequência, das interações. Um outro tipo de rede são as redes sociais com atributos, as quais as arestas, ou nós, possuem características adicionais que podem influenciar nas análises. Assim, a variedade e complexidade das redes sociais exigem abordagens flexíveis e adaptativas para a análise de dados privada. Portanto, avançar na implementação de modelos de privacidade eficazes, como a privacidade diferencial, em diferentes tipos de redes sociais é crucial para garantir que a análise de dados possa ser realizada de maneira segura, garantindo a privacidade dos indivíduos enquanto se extraem *insights* valiosos.

No entanto, apesar da vasta literatura existente sobre privacidade diferencial para redes sociais, ainda destaca-se a carência de ferramentas de código aberto acessíveis para a implementação dessas técnicas. Essa lacuna não só limita a aplicação prática das metodologias discutidas, como também representa uma barreira para pesquisadores e desenvolvedores que buscam integrar privacidade diferencial em suas análises em redes sociais de maneira eficiente e replicável. Além disso, diversas estatísticas valiosas, como as medidas de centralidade, ainda carecem de versões diferencialmente privadas. Essa ausência aponta para uma rica área de pesquisa em aberto, onde a adaptação e desenvolvimento de métodos que garantam a privacidade diferencial para essas estatísticas podem trazer significativos avanços, tanto teóricos, quanto práticos.

Outro aspecto essencial é a necessidade de alinhar as expectativas e interpretações dos usuários em relação à privacidade diferencial aplicada em redes sociais. O orçamento de privacidade ε é muitas vezes considerado contraintuitivo, além de que a compreensão das diferentes abordagens de privacidade, seja a nível de aresta, nó ou atributo, são áreas que exigem maior clareza na sua explicação. Além disso, o desempenho computacional apresenta-se como um fator limitante na implementação de técnicas de privacidade diferencial para análises em redes sociais. As técnicas atuais, na grande maioria das vezes, exigem recursos computacionais consideráveis, o que pode limitar sua aplicabilidade em cenários com grandes volumes de dados, ou em tempo real. Por fim, estender a privacidade diferencial para ambientes de redes sociais dinâmicas, com fluxo de dados contínuos (*streaming*), permanece sendo uma área aberta para pesquisa.

Em resumo, a realização de análise privada de dados em redes sociais é um campo dinâmico que demanda inovação contínua para equilibrar a utilidade dos dados com a necessidade de proteger a privacidade dos usuários. No entanto, apesar dos avanços significativos realizados nos modelos de privacidade, diversos desafios e oportunidades de pesquisa permanecem em aberto. O desenvolvimento de novas técnicas, juntamente com a adaptação contínua a diferentes estruturas de redes sociais, é essencial para um futuro onde a privacidade e a análise de dados possam coexistir harmoniosamente.

Agradecimentos

Este trabalho foi parcialmente financiado pela Lenovo, como parte do seu investimento em Pesquisa e Desenvolvimento (P&D) de acordo com a Lei de Informática. Os autores também agradecem ao financiamento fornecido pela CAPES, sob os processos de número 88881.189723/2018-01 e 88882.454571/2019-01, e pelo CNPq, sob o processo de número 316729/2021-3.

Referências

[Abdulhamid et al. 2014] Abdulhamid, S. M., Ahmad, S., Waziri, V. O., and Jibril, F. N. (2014). Privacy and national security issues in social networks: the challenges. *arXiv preprint arXiv:1402.3301*.

[Acharya et al. 2019] Acharya, J., Sun, Z., and Zhang, H. (2019). Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1120–1129. PMLR.

[Allardyce 2012] Allardyce, C. S. (2012). *Fat chemistry: The science behind obesity*. Royal Society of Chemistry.

[Alsmadi and Alhami 2015] Alsmadi, I. and Alhami, I. (2015). Clustering and classification of email contents. *Journal of King Saud University-Computer and Information Sciences*, 27(1):46–57.

[Baden et al. 2009] Baden, R., Bender, A., Spring, N., Bhattacharjee, B., and Starin, D. (2009). Persona: an online social network with user-defined privacy. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 135–146.

[Bassily and Smith 2015] Bassily, R. and Smith, A. (2015). Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135.

[Bhaila et al. 2023] Bhaila, K., Huang, W., Wu, Y., and Wu, X. (2023). Local differential privacy in graph neural networks: a reconstruction approach. *arXiv preprint arXiv:2309.08569*.

[Bloch et al. 2023] Bloch, F., Jackson, M. O., and Tebaldi, P. (2023). Centrality measures in networks. *Social Choice and Welfare*, 61(2):413–453.

[Boyd and Ellison 2007] Boyd, D. M. and Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of computer-mediated Communication*, 13(1):210–230.

[Brazil 2018] Brazil (2018). Lei Geral de Proteção de Dados Pessoais. `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm`. Acesso em: 22 de maio de 2024.

[Brito 2023] Brito, F. T. (2023). *Differentially private release of count-weighted graphs*. PhD thesis, Universidade Federal do Ceará.

[Brito et al. 2023] Brito, F. T., Farias, V. A., Flynn, C., Majumdar, S., Machado, J. C., and Srivastava, D. (2023). Global and local differentially private release of count-weighted graphs. *Proceedings of the ACM on Management of Data*, 1(2):1–25.

[Brito and Machado 2017] Brito, F. T. and Machado, J. C. (2017). Preservação de privacidade de dados: Fundamentos, técnicas e aplicações. *Jornadas de atualização em informática*, pages 91–130.

[Brito et al. 2024] Brito, F. T., Mendonça, A. L. C., and Machado, J. C. (2024). A differentially private guide for graph analytics. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 850–853. OpenProceedings.org.

[Brito et al. 2015] Brito, F. T., Neto, A. C. A., Costa, C. F., Mendonça, A. L., and Machado, J. C. (2015). A distributed approach for privacy preservation in the publication of trajectory data. In *Proceedings of the 2nd Workshop on Privacy in Geographic Information Collection and Analysis*, pages 1–8.

[Chaabane et al. 2012] Chaabane, A., Acs, G., Kaafar, M. A., et al. (2012). You are what you like! information leakage through users' interests. In *Proceedings of the 19th annual network & distributed system security symposium (NDSS)*. Citeseer.

[Chen et al. 2022] Chen, L., Han, K., Xiu, Q., and Gao, D. (2022). Graph clustering under weight-differential privacy. In *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 1457–1464. IEEE.

[Chen et al. 2014] Chen, R., Fung, B., Yu, P. S., and Desai, B. C. (2014). Correlated network data publication via differential privacy. *The VLDB Journal*, 23(4):653–676.

[Clauset et al. 2006] Clauset, A., Moore, C., and Newman, M. E. (2006). Structural inference of hierarchies in networks. In *ICML Workshop on Statistical Network Analysis*, pages 1–13. Springer.

[Cook and Holder 2006] Cook, D. J. and Holder, L. B. (2006). *Mining graph data*. John Wiley & Sons.

[Cormode et al. 2018] Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. (2018). Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658.

[Cormode et al. 2012] Cormode, G., Procopiuc, C., Srivastava, D., and Tran, T. T. (2012). Differentially private summaries for sparse data. In *Proceedings of the 15th International Conference on Database Theory*, pages 299–311.

[da Costa Filho and Machado 2023] da Costa Filho, J. S. and Machado, J. C. (2023). FELIP: A local differentially private approach to frequency estimation on multidimensional datasets. In *Proceedings 26th International Conference on Extending Database*

Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023, pages 671–683. OpenProceedings.org.

[Daigavane et al. 2021] Daigavane, A., Madan, G., Sinha, A., Thakurta, A. G., Aggarwal, G., and Jain, P. (2021). Node-level differentially private graph neural networks. *arXiv:2111.15521*.

[Dalenius 1977] Dalenius, T. (1977). Towards a methodology for statistical disclosure control.

[Day et al. 2016] Day, W.-Y., Li, N., and Lyu, M. (2016). Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138.

[Dey et al. 2012] Dey, R., Tang, C., Ross, K., and Saxena, N. (2012). Estimating age privacy leakage in online social networks. In *2012 proceedings ieee infocom*, pages 2836–2840. IEEE.

[Duchi et al. 2013] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2013). Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE.

[Dwork 2006] Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer.

[Dwork et al. 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

[Dwork et al. 2014] Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

[Erlingsson et al. 2014] Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM.

[European Commission 2018] European Commission (2018). 2018 reform of EU data protection rules. `https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf`. Acesso em: 22 de maio de 2024.

[Fan and Li 2022] Fan, C. and Li, P. (2022). Distances release with differential privacy in tree and grid graph. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2190–2195. IEEE.

[Farias et al. 2020] Farias, V. A., Brito, F. T., Flynn, C., Machado, J. C., Majumdar, S., and Srivastava, D. (2020). Local dampening: differential privacy for non-numeric queries via local sensitivity. *Proceedings of the VLDB Endowment*, 14(4):521–533.

[Farias et al. 2023] Farias, V. A., Brito, F. T., Flynn, C., Machado, J. C., Majumdar, S., and Srivastava, D. (2023). Local dampening: Differential privacy for non-numeric queries via local sensitivity. *The VLDB Journal*, pages 1–24.

[Garfinkel et al. 2018] Garfinkel, S. L., Abowd, J. M., and Powazek, S. (2018). Issues encountered deploying differential privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 133–137. ACM.

[Ghosh et al. 2009] Ghosh, A., Roughgarden, T., and Sundararajan, M. (2009). Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 351–360.

[Gong et al. 2014] Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., Shi, E., and Song, D. (2014). Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2):1–20.

[Hay et al. 2009] Hay, M., Li, C., Miklau, G., and Jensen, D. (2009). Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE.

[Hsu et al. 2014] Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., and Roth, A. (2014). Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE.

[Iftikhar et al. 2020] Iftikhar, M., Wang, Q., and Lin, Y. (2020). dk-microaggregation: Anonymizing graphs with differential privacy guarantees. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 191–203. Springer.

[Imola et al. 2021] Imola, J., Murakami, T., and Chaudhuri, K. (2021). Locally differentially private analysis of graph statistics. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 983–1000.

[Ji et al. 2021] Ji, T., Li, P., Yilmaz, E., Ayday, E., Ye, Y., and Sun, J. (2021). Differentially private binary-and matrix-valued data query: an xor mechanism. *Proceedings of the VLDB Endowment*, 14(5):849–862.

[Ji et al. 2019] Ji, T., Luo, C., Guo, Y., Ji, J., Liao, W., and Li, P. (2019). Differentially private community detection in attributed social networks. In *Asian Conference on Machine Learning*, pages 16–31. PMLR.

[Jian et al. 2021] Jian, X., Wang, Y., and Chen, L. (2021). Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering*.

[Jiang et al. 2021] Jiang, H., Pei, J., Yu, D., Yu, J., Gong, B., and Cheng, X. (2021). Applications of differential privacy in social network analysis: A survey. *IEEE transactions on knowledge and data engineering*, 35(1):108–127.

[Jorgensen et al. 2016] Jorgensen, Z., Yu, T., and Cormode, G. (2016). Publishing attributed social graphs with formal privacy guarantees. In *Proceedings of the 2016 international conference on management of data*, pages 107–122.

[Karwa et al. 2011] Karwa, V., Raskhodnikova, S., Smith, A., and Yaroslavtsev, G. (2011). Private analysis of graph structure. *PVLDB*, 4(11):1146–1157.

[Karwa and Slavković 2012] Karwa, V. and Slavković, A. B. (2012). Differentially private graphical degree sequences and synthetic graphs. In *International Conference on Privacy in Statistical Databases*, pages 273–285. Springer.

[Kasiviswanathan et al. 2013] Kasiviswanathan, S. P., Nissim, K., Raskhodnikova, S., and Smith, A. (2013). Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer.

[Kaytoue et al. 2017] Kaytoue, M., Plantevit, M., Zimmermann, A., Bendimerad, A., and Robardet, C. (2017). Exceptional contextual subgraph mining. *Machine Learning*, 106:1171–1211.

[Kenthapadi et al. 2019] Kenthapadi, K., Mironov, I., and Thakurta, A. (2019). Privacy-preserving data mining in industry. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 1308–1310. ACM.

[Kossinets and Watts 2006] Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *science*, 311(5757):88–90.

[Laeuchli et al. 2022] Laeuchli, J., Ramírez-Cruz, Y., and Trujillo-Rasua, R. (2022). Analysis of centrality measures under differential privacy models. *Applied Mathematics and Computation*, 412:126546.

[Li et al. 2023] Li, L., Zhao, Y., Luo, S., Wang, G., and Wang, Z. (2023). Efficient community search in edge-attributed graphs. *IEEE Transactions on Knowledge and Data Engineering*.

[Li et al. 2006] Li, N., Li, T., and Venkatasubramanian, S. (2006). t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE.

[Li et al. 2016] Li, N., Lyu, M., Su, D., and Yang, W. (2016). Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, & Trust*, 8(4):1–138.

[Li et al. 2017] Li, X., Yang, J., Sun, Z., and Zhang, J. (2017). Differential privacy for edge weights in social networks. *Security and Communication Networks*, 2017.

[Liu et al. 2020] Liu, Z., Huang, L., Xu, H., Yang, W., and Wang, S. (2020). Privag: Analyzing attributed graph data with local differential privacy. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 422–429. IEEE.

[Machanavajjhala et al. 2007] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es.

[Mahadevan et al. 2006] Mahadevan, P., Krioukov, D., Fall, K., and Vahdat, A. (2006). Systematic topology analysis and generation using degree correlations. *ACM SIGCOMM Computer Communication Review*, 36(4):135–146.

[McPherson et al. 2001] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

[McSherry and Talwar 2007] McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE.

[Mendonça et al. 2023] Mendonça, A. L., Brito, F. T., and Machado, J. C. (2023). Privacy-preserving techniques for social network analysis. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 174–178. SBC.

[Mislove et al. 2010] Mislove, A., Viswanath, B., Gummadi, K. P., and Druschel, P. (2010). You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260.

[Mitchell 2017] Mitchell, J. S. (2017). Shortest paths and networks. In *Handbook of discrete and computational geometry*, pages 811–848. Chapman and Hall/CRC.

[Mohamed et al. 2022] Mohamed, M. S., Nguyen, D., Vullikanti, A., and Tandon, R. (2022). Differentially private community detection for stochastic block models. In *International Conference on Machine Learning*, pages 15858–15894. PMLR.

[Mueller et al. 2022] Mueller, T. T., Paetzold, J. C., Prabhakar, C., Usynin, D., Rueckert, D., and Kaissis, G. (2022). Differentially private graph neural networks for whole-graph classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Narayanan and Shmatikov 2008] Narayanan, A. and Shmatikov, V. (2008). Robust deanonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE.

[Narayanan and Shmatikov 2009] Narayanan, A. and Shmatikov, V. (2009). Deanonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE.

[Near and Abuah 2021] Near, J. P. and Abuah, C. (2021). Programming differential privacy. *URL: https://uvm*.

[Nergiz et al. 2007] Nergiz, M. E., Atzori, M., and Clifton, C. (2007). Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 665–676.

[Nguyen et al. 2015] Nguyen, H. H., Imine, A., and Rusinowitch, M. (2015). Differentially private publication of social graphs at linear cost. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 596–599. IEEE.

[Nguyen et al. 2016] Nguyen, H. H., Imine, A., and Rusinowitch, M. (2016). Detecting communities under differential privacy. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 83–93.

[Nissim et al. 2007] Nissim, K., Raskhodnikova, S., and Smith, A. (2007). Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84.

[Pinot et al. 2018] Pinot, R., Morvan, A., Yger, F., Gouy-Pailler, C., and Atif, J. (2018). Graph-based clustering under differential privacy. *arXiv preprint arXiv:1803.03831*.

[Prell 2011] Prell, C. (2011). *Social network analysis: History, theory and methodology*. Sage.

[Qian et al. 2016] Qian, J., Li, X.-Y., Zhang, C., and Chen, L. (2016). De-anonymizing social networks and inferring private attributes using knowledge graphs. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE.

[Ribeiro et al. 2021] Ribeiro, P., Paredes, P., Silva, M. E., Aparicio, D., and Silva, F. (2021). A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)*, 54(2):1–36.

[Roohi et al. 2019] Roohi, L., Rubinstein, B. I., and Teague, V. (2019). Differentially-private two-party egocentric betweenness centrality. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2233–2241. IEEE.

[Sala et al. 2011] Sala, A., Zhao, X., Wilson, C., Zheng, H., and Zhao, B. Y. (2011). Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98.

[Samoylenko et al. 2023] Samoylenko, I., Aleja, D., Primo, E., Alfaro-Bittner, K., Vasilyeva, E., Kovalenko, K., Musatov, D., Raigorodskii, A. M., Criado, R., Romance, M., et al. (2023). Why are there six degrees of separation in a social network? *Physical Review X*, 13(2):021032.

[Sealfon 2016] Sealfon, A. (2016). Shortest paths and distances with differential privacy. In *Proceedings of the 35th Symposium on Principles of Database Systems*, pages 29–41.

[Shah et al. 2016] Shah, N., Beutel, A., Hooi, B., Akoglu, L., Gunnemann, S., Makhija, D., Kumar, M., and Faloutsos, C. (2016). Edgecentric: Anomaly detection in edge-attributed networks. In *2016 IEEE 16Th international conference on data mining workshops (ICDMW)*, pages 327–334. IEEE.

[Shi et al. 2016] Shi, C., Li, Y., Zhang, J., Sun, Y., and Philip, S. Y. (2016). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37.

[Silva et al. 2017] Silva, R. R. C., Leal, B. C., Brito, F. T., Vidal, V. M., and Machado, J. C. (2017). A differentially private approach for querying rdf data of social networks. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, pages 74–81.

[Sun et al. 2019] Sun, H., Xiao, X., Khalil, I., Yang, Y., Qin, Z., Wang, H., and Yu, T. (2019). Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–717.

[Sweeney 2002] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

[Task and Clifton 2014] Task, C. and Clifton, C. (2014). What should we protect? defining differential privacy for social network analysis. In *State of the Art Applications of Social Network Analysis*, pages 139–161. Springer.

[Wang et al. 2013] Wang, C.-D., Lai, J.-H., and Philip, S. Y. (2013). Neiwalk: Community discovery in dynamic content-based networks. *IEEE transactions on knowledge and data engineering*, 26(7):1734–1748.

[Wang and Long 2019] Wang, D. and Long, S. (2019). Boosting the accuracy of differentially private in weighted social networks. *Multimedia Tools and Applications*, 78(24):34801–34817.

[Wang et al. 2017] Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745.

[Wang and Wu 2013] Wang, Y. and Wu, X. (2013). Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy*, 6(2):127.

[Xiao et al. 2014] Xiao, Q., Chen, R., and Tan, K.-L. (2014). Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 911–920.

[Ye and Barg 2018] Ye, M. and Barg, A. (2018). Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, 64(8):5662–5676.

[Zhang et al. 2015] Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., and Xiao, X. (2015). Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 731–745.