Utilização de Grafos de *Nilcatenation* na Identificação de Transações para *Pruning* em *Blockchains*

Igor da Silva Bonomo[1], **Eduardo Alchieri**[1]

[1]Departamento de Ciência da Computação
Universidade de Brasília
Brasília, DF – Brasil.

Abstract. *blockchain technology was consolidated with its use in various areas of knowledge, mainly in the context of cryptocurrencies. However, there are still limitations to its wide adoption in various applications. A relevant limitation is the increasing size of blockchains, which causes both storage problems and leads to an increasing need for node synchronization time. In this context, this paper studies a solution for pruning in blockchains. This technique consists of removing transactions from the network without compromising the consistency of the blockchain. The proposed solution is based on nilcatenation graphs, whose objective is to identify a subgraph that can be removed without impiring the consistency of the network. Experimental results show that the proposed solution can more accurately identify transactions that can be removed (reduced up to 20% of the total transactions), when compared to current techniques (reduced up to 5% of the total transactions) that seek to find transaction cycles in these graphs to be removed.*

Resumo. *A tecnologia blockchain se consolidou com sua utilização em diversas áreas do conhecimento, tendo seu uso maior nas criptomoedas. Porém, ainda existem limitações para a sua ampla utilização em diversas aplicações. Uma limitação relevante é o tamanho crescente das blockchains, o que causa tanto problemas de armazenamento quanto leva a uma necessidade crescente de tempo para sincronização de nós. Neste contexto, este artigo estuda uma solução para pruning em blockchains. Essa técnica consiste em remover transações da rede sem prejudicar a consistência da blockchain. A solução proposta é baseada em grafos de nilcatenation, cujo objetivo é identificar um subgrafo que pode ser removido sem comprometer a consistência da rede. Resultados experimentais mostram que a solução proposta consegue identificar de forma mais precisa as transações que podem ser removidas (chegando a uma redução de até 20% das transações), quanto comparado com técnicas atuais (conseguiram fornecer até 5% de redução) que buscam encontrar ciclos de transações nestes grafos que podem ser removidas.*

1. Introdução

O *blockchain* surgiu como um paradigma para construção de aplicações seguras e descentralizadas com potencial para revolucionar todos os aspectos de nossa vida digital [Pech 2017, Tapscott and Tapscott 2016]. A tecnologia *blockchain* é conhecida por suas inúmeras características, dentre elas a segurança, a transparência

e a descentralização. Além das criptomoedas, *blockchains* podem impactar em vários outros domínios de aplicações. Desde mercados descentralizados (similares ao Uber) [Subramanian 2018] a organizações autônomas regradas por *smart contracts* [Buterin 2014], esta tecnologia é requerida pela maioria dos serviços financeiros, serviços de governos, serviços de cartórios e serviços de cadeias de suprimentos [Tapscott and Tapscott 2016]. De fato, é difícil encontrar um serviço *online* que não poderia se beneficiar de um livro de transações seguro (e programável) sem um ponto central de confiança.

Como podemos perceber, de maneira geral o uso de *blockchains* já é bastante difundido em diversas áreas do conhecimento. No entanto, com a maior utilização desta tecnologia e o surgimento de cada vez mais aplicações, também é possível observar alguns problemas que potencialmente podem limitar sua ampla adoção no futuro. Dentro desse contexto e estudos relacionados, um problema relevante está relacionado com o crescente tamanho das *blockchains*, o que impacta tanto na necessidade de armazenamento quanto no tempo de processamento para sincronização de um novo membro (nó) que entra na rede.

No bitcoin [Nakamoto 2009], a aplicação mais conhecida da tecnologia *blockchain*, tem-se que o tamanho da rede em dezembro de 2022 era de 440GB e sua taxa de crescimento aproximada é de 50GB anual [Blockchain.com 2023]. Além da necessidade grande de armazenamento, o tempo estimado para sincronizar os dados nesta rede pode ser de aproximadamente duas semanas. Dessa maneira, o problema do armazenamento torna-se uma das maiores questões em relação a eficiência e alcance da *blockchain*, visto que os nós da rede precisam fazer o download de uma cópia de todas as transações, i.e., toda a *blockchain*, para sincronização de forma a poder validar novas transações. Então, à medida que o tamanho da *blockchain* aumenta, questões relacionadas com o armazenamento e a sincronização de dados torna-se um desafio cada vez maior [Quan et al. 2019].

Com base nos estudos e pesquisas anteriores, foram surgindo técnicas para resolver ou diminuir o problema de armazenamento da *blockchain*, dentre essas a que será explorada nesse artigo é a técnica de *pruning* [Palm et al. 2018]. Essa técnica consiste, de algum modo, remover transações da rede sem prejudicar a consistência da *blockchain*. Para isso é necessário identificar transações que podem ser removidas da rede. Este processamento pode variar de acordo com algumas característica pré determinadas pelos algoritmos. Vale destacar que neste artigo estamos especificamente interessados em formas de identificar transações a serem removidas, e não nos protocolos para a remoção das transações em si que podem ser encontrados em trabalhos relacionados [Palm et al. 2018].

No entanto, no contexto de criptomoedas, as soluções atuais [Palm et al. 2018] consistem em criar um grafo de transações, onde cada nó representa uma carteira e as arestas representam as transações, i.e., transferências entre as carteiras. Após isso, utiliza-se um algoritmo para encontrar um ciclo neste grafo com transações que se cancelam, i.e., estas transferências, em conjunto, não modificam o estado das carteiras pois fazem com que o estado volte para o mesmo de antes das suas execuções.

Apesar de promissora, esta técnica ainda não consegue identificar com precisão transações que podem ser removidas. Sendo assim, neste trabalho propomos a utilização

de grafos *nilcatenation*, onde a ideia é identificar um subgrafo com transações que se cancelam, e não mais ficar restrito a um ciclo. Este tipo de grafo já foi utilizado no contexto de *blockchain*s para auxiliar na identificação de fraudes e lavagem de dinheiro [Amaral et al. 2020].

De forma resumida, este artigo apresenta as seguintes contribuições:

- Discussão de *pruning* em *blockchain*s e proposta de utilização de grafos de *nilcatenation* para identificação de transações que podem ser removidas (subgrafo) de forma segura sem impactar na consistência da *blockchain*.
- Analise experimental comparando a técnica proposta com as técnicas baseadas em detecção de ciclos.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta a fundamentação teórica sobre *blockchain*, grafos e o problema de *nilcatenation*. A Seção 3 apresenta os trabalhos relacionados com o problema de armazenamento, *pruning* e grafos *nilcatenation*. A Seção 4 apresenta a proposta do artigo. A Seção 5 apresenta os experimentos realizado. Por fim, a Seção 6 apresenta as conclusões e discussões sobre trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta a fundamentação teórica necessária para entendimento do artigo. Desta forma, será abordado conceitos gerais de grafos e como funciona o problema de *nilcatenation*, além de uma breve discussão sobre a técnica de *pruning*.

2.1. *Blockchain*

A tecnologia do *blockchain* foi introduzida por *Satoshi Nakamoto*, sendo popularmente reconhecida pelo seu emprego no *Bitcoin* [Nakamoto 2009]. Um *blockchain* pode ser definido com uma estrutura de dados, que funciona como uma sequência de blocos conectados, cada um armazenando um conjunto finito de transações que podem ser operações ou registro de dados. Cada bloco apresenta uma assinatura digital, que é referenciada criptograficamente por um valor de *hash*, promovendo autenticidade e identidade para a criação e verificação das transações. A estrutura que define a organização dos blocos em cadeia é alcançada em virtude das seguintes características: o valor de *hash* do bloco anterior, ou seja, do bloco pai, e o valor de *hash* do próprio bloco, exceto pelo primeiro bloco, conhecido por bloco gênesis. A Figura 1 mostra um exemplo dos blocos em cadeia.

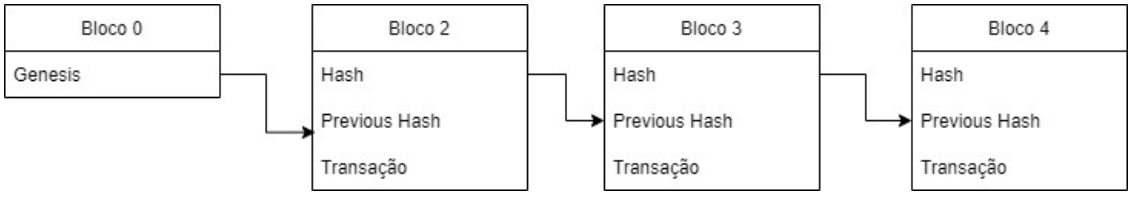

Figura 1. Representação de uma *blockchain*.

Um *blockchain* também adota um modelo de sistema distribuído, P2P, onde os nós cooperam entre si seguindo um protocolo definido para evoluir de forma sincronizada, i.e., para todos os nós adotarem o mesmo bloco seguinte para aumentar a *blockchain*.

2.2. Grafos

Um grafo pode ser definido por G = (V,E) onde V é um conjunto finito de vértices (ou nós) e E é uma coleção finita de dois elementos do conjunto de vértices, conhecidos como arestas. Em um grafo ponderado, tem-se que o peso é o valor associado a cada aresta do grafo. Sendo assim, um valor positivo pode ser associado a um vértice ao qual essa aresta está chegando e um valor negativo a um vértice ao qual a aresta está saindo [Biggs et al. 1986].

Dessa maneira, pode-se associar um grafo em um matriz, cujo os valores associados podem ser positivo se a aresta tiver entrando no vértice, negativo se tiver saindo e 0 se não tiver conexão entre os vértices.

$$Cij = \begin{cases} Wj \\ -Wj \\ 0 \end{cases} \tag{1}$$

A Equação 1 demonstra um grafo i por j no qual o peso Wj é positivo se o arco j está entrando no vértice i. O peso negativo $-Wj$ representa que o arco j está saindo do vértice i. Pro fim, o peso 0 indica que o arco não é incidente.

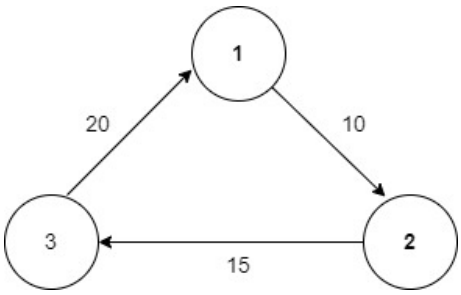

Figura 2. Exemplo de grafo

A Figura 2 demonstra um exemplo de um grafo direcionado com peso associado a suas arestas. A Equação 2 demonstra a matriz gerada por este grafo.

$$Cij = \begin{Bmatrix} 0 & -10 & 20 \\ 10 & 0 & -15 \\ 0 & 15 & -20 \end{Bmatrix} \tag{2}$$

2.3. Componentes fortemente conexas

Defini-se uma componente fortemente conexa de um grafo direcionado G, como um conjunto de vértices no qual para cada par de vértices existem caminhos que conectam os mesmos em ambas direções. Dessa maneira, pode-se dizer os vértices de uma componente fortemente conexa são alcançáveis por qualquer outro vértice dentro da mesma componente [Karp 2011].

2.4. Problema de *Nilcatenation* (NCP)

Dado um multigrafo ponderado, encontrar um grafo de *nilcatenation* consiste em identificar arestas que podem ser removidas sem impactar em nenhum balanço, ou seja, que preserve a semântica do grafo.

Como exemplo de *nilcatenation*, considere o grafo original representado pela Figura 3. Neste grafo, temos um subgrafo (ou grafo) de *nilcatenation* extraído na Figura 4. Para qualquer vértice deste subgrafo, o somatório das arestas de entrada é igual ao somatório das arestas de saída. Por fim, a Figura 5 representa o original sem o *nilcatenation*, que é um subgrafo gerado do grafo original retirando o grafo gerado pelo algoritmo de *nilcatenation*.

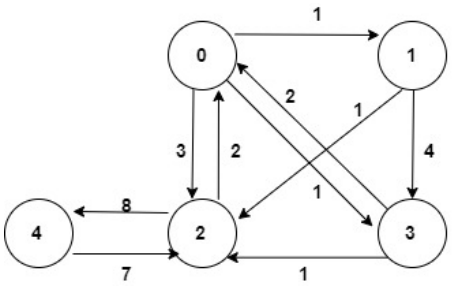

Figura 3. Grafo Original

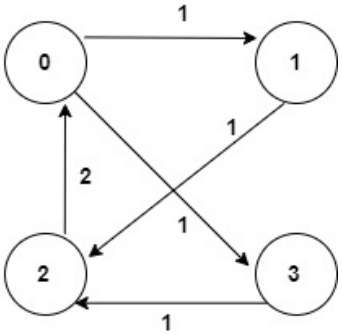

Figura 4. Grafo *Nilcatenation*

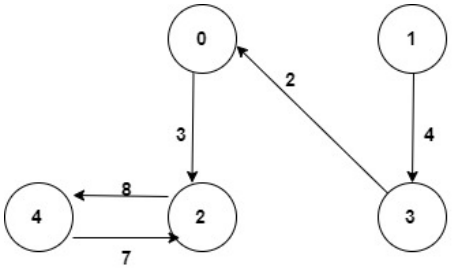

Figura 5. Grafo Original sem o *Nilcatenation*.

2.5. Subset-Sum Problem (SSP)

Dado um conjunto finito de número inteiros A e um valor alvo t, este problema consiste em encontrar um subconjunto S contido em A tal qual o somatório dos valores do subconjunto S seja igual ao valor t [Wang 2004].

2.6. *Pruning*

São técnicas que consistem em remover informação da *blockchain*, podendo ser um bloco inteiro ou transações individuais. Isto significa a perda de graus de segurança ou traceabilidade em troca de tornar a *blockchain* mais compacta.

Os algoritmos de *pruning* podem ser executados de acordo com regras pre-estabelecidas, como por exemplo: transações que se cancelam, mudanças consecutivas em um mesmo dado ou transações aplicadas em dados deletados [Gordon 2020].

3. Trabalhos Relacionados

Nesta seção, serão abordados os trabalhos que possuem uma relação com os temas de armazenamento da *blockchain* e de *pruning*. Desta maneira, serão tratados os aspectos de projetos que envolvem as abordagens para resolver o problema de armazenamento da rede e a utilização da técnica de *pruning*.

Géraud et al [Géraud et al. 2017] propuseram a utilização de técnicas de grafos para redução das transações da *blockchain*. Apesar de também chamarem de grafos de *nilcatenation*, a abordagem empregada busca identificar as componentes fortemente conexas e então, para cada componente, encontrar um conjunto de nós que possuam uma aresta de entrada e uma aresta de saída com o mesmo valor, i.e., esta solução busca por um ciclo cujas arestas possuam o mesmo peso. De fato, é realizada a comparação entre os graus de entrada e de saída de cada vértice. Se o mínimo entre o grau de entrada e o grau de saída for igual a zero, ou seja, se o vértice escolhido somente possuir arestas de entrada ou arestas de saída, as arestas são removidas da solução pois não poderão ser removidas da *blockchain*. Por outro lado, se existir apenas uma aresta de entrada e uma aresta de saída com o mesmo peso, então estas arestas são mantidas no grafo, caso contrário também são removidas. Desta forma, o artigo apresenta uma proposta mais simplificada e somente identifica ciclos.

Tomacheski et al [Amaral et al. 2020] propuseram a utilização de técnicas de grafos para detectar lavagem de dinheiro em criptomoedas. A metodologia aplicada consiste na aplicação do problema de *nilcatenation* para identificar padrões de lavagem de dinheiro. Os autores propõem modelar o histórico de transações de uma criptomoeda em um grafo e aplicar as técnicas de *nilcatenation* para verificar padrões na rede. A dissertação apresenta uma proposta que utiliza o problema de *nilcatenation*, em conjunto com programação linear em uma base de instâncias de teste. O texto apresenta o resultado para identificação de grafos de *nilcatenation* em uma base de dados de testes criado pelos autores, sendo possível identificar padrões de possível lavagem de dinheiro, porém ainda falta um estudo mais a fundo com a utilização em dados reais.

Quan et al [Quan et al. 2019] propuseram a utilização da técnica de *pruning* para reduzir o tamanho das transações de uma rede *blockchain*. A metodologia aplicada consiste em utilização de *downsampling* no corpo dos blocos para reduzir os nós armazenados. Os autores propõem a redução dos nós através de uma técnica de *downsampling* com base na informação de entropia dos dados armazenados. O artigo apresenta uma simulação com os resultados do algoritmo de *downsample* mostrando que a acurácia melhora com uma escolha apropriada para o *threshold* da entropia, o que nem sempre é fácil

de ser configurado. O texto apresenta resultados que reduzem o armazenamento dos nós, porém com dados de simulação e carece de dados reais.

Palm et al [Palm et al. 2018] propuseram um algoritmo de *pruning* para reduzir o tamanho da *blockchain*. A metodologia aplicada consiste em utilizar uma função de predicado para selecionar os blocos que podem ser retirados da rede. Os autores propõem a ideia de um estado de derivabilidade que serve para decidir se a transação pode ou não ser considerada para o algoritmo de *pruning*. O artigo apresenta três estados para a decisão: transações significantes, transações universais insignificantes e transações insignificantes retroativas. O texto apresenta a ideia de transações que se cancelam baseada na detecção de ciclos de transações no grafo, o que faz com que nem todas as transações que podem ser removidas sejam identificadas. Nossa proposta visa evoluir nestes aspectos através da identificação de subgrafos ao invés de apenas ciclos.

4. Proposta

Nesta seção será descrito a proposta apresentada para melhorar o problema de armazenamento da rede *blockchain*, com a aplicação do algoritmo de *nilcatenation* para execução das técnicas de *pruning*, através da identificação de transações que podem ser removidas da rede sem comprometer a sua consistência.

A Figura 6 exemplifica como que a proposta foi pensada para esse problema. Dessa maneira, tem-se dividido em 4 etapas:

- Transações da *blockchain*
- Grafo
- Algoritmo de *nilcatenation*
- *Pruning*

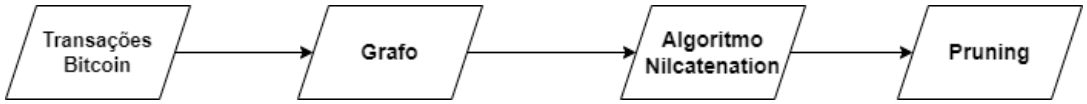

Figura 6. Proposta.

De forma resumida, a proposta desse trabalho é, a partir das transações da *blockchain*, criar um grafo e executar o algoritmo de *nilcatenation* para obter o subgrafo de *nilcatenation*, que será for fim utilizado pelo algoritmo de *pruning*.

4.1. Transações da *Blockchain*

A primeira etapa consiste em obter transações de uma *blockchain*. Neste artigo restringimos nossa discussão para criptomoedas e transações de transferências entre contas, mas os mesmos conceitos podem ser empregados de forma mais ampla. Sendo assim, usa-se um conjunto de transações dessa *blockchain* escolhida, que podem variar de acordo com alguns aspectos como: data, tamanho do bloco e tamanho das transações.

Esse conjunto de transações deve conter algumas informações importantes para esse projeto, sendo elas os endereços de origem, de destino e o valor da transação. A partir dessas informações é possível ter como base os dados que vão servir como entrada para o algoritmo e assim poder encontra quais transações podem ser retiradas da *blockchain*.

Note que não é necessário criar um grafo com toda a *blockchain*, mas pode-se selecionar intervalos de transações de forma a diminuir o tamanho do grafo, juntamente com a quantidade de processamento necessária pelo algoritmo de *nilcatenation*. Porém, com isso também diminui a possibilidade de se encontrar transações que se cancelam, pois os grafos serão analisados separadamente. Encontrar este equilíbrio é um desafio em aberto para trabalhos futuros.

4.2. *Blockchain* em Grafo

Com o resultado da primeira etapa que obteve o conjunto de transações que serão utilizadas, a segunda etapa consiste em transformar esse conjunto de transações em um grafo. O intuito dessa etapa é transformar os endereços de destino e de origem em vértices do grafo, sendo que as transações serão as arestas.

Com base nas informações de endereços de origem, de destino e valor das transações, consegue-se gerar um grafo e viabilizar a execução do algoritmo de *nilcatenation*. A Figura 7 exemplifica como que ocorre essa transformação de uma transação da *blockchain* em um grafo. Dessa maneira, tem-se um endereço de origem e um endereço de destino sendo os vértices e representados por um *hash*, além de um valor $(0,092)$ da transação da origem para o destino sendo a aresta desse grafo.

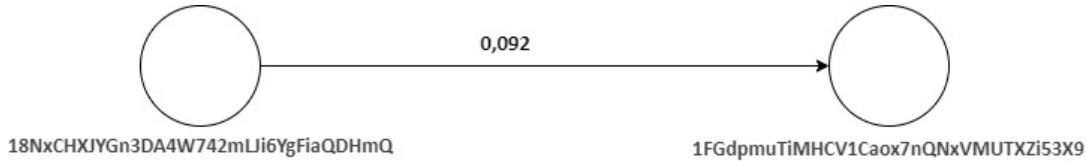

Figura 7. Grafo da Transação.

4.3. Algoritmo de *Nilcatenation*

A terceira etapa consiste na execução do algoritmo de *nilcatenation*, ou seja, através do grafo que foi gerado das transações da *blockchain*, obter o grafo de *nilcatenation*. Dessa maneira, usando como entrada de dados o grafo das transações, o algoritmo de *nilcatenation* vai gerar um grafo com as possíveis transações da *blockchain* que podem ser retiradas da rede. Para a execução do algoritmo é necessário a realização de algumas etapas: a etapa de análise dos vértices e a de análise de arestas. Para a realização dessas etapas, é preciso primeiro identificar as componentes fortemente conexas do grafo de transações, pois um subgrafo de *nilcatenation* somente estará presente nestas componentes e com isso diminui-se a quantidade de processamento necessária.

4.3.1. Análise dos Vértices

Após identificação das componentes fortemente conexas, deve-se percorrer cada vértice desse subgrafo gerado. Sendo assim, a partir do vértice analisado, deve-se buscar uma solução para os pesos das arestas de entrada e saída do mesmo, utilizando uma solução para o problema SSP discutido na fundamentação teórica. A primeira parte é criar uma lista com pesos positivos, ou seja, aqueles que entram no vértice, a segunda é criar uma

lista com os pesos negativos, ou seja, aqueles que saem do vértice. Dessa maneira, faz-se um união entra as listas positivas e negativas, buscando uma solução para essa união. Nessa etapa, é necessário verificar se o peso das arestas que entram é igual ao peso das arestas que saem. Sendo assim, deve-se buscar uma solução para soma dessa lista que seja igual a zero. Se não existir uma solução para esse problema da soma, o vértice pode ser removido do conjunto de vértices e as arestas envolvidas também. Isto quer dizer que esses vértices e arestas não fazem parte da solução final. Se existir uma solução, aquele vértice não pode ser removido da solução pois possivelmente fará parte do subgrafo de *nilcatenation*.

for *i = 1; i* $< Vertice$ *; i++* **do**
    Apos = Wj | ej $\in$ E+(vi);
    Aneg = -wj | ej $\in$ E-(vi);
    possuiSol = Sol(Apos $\bigcup$ Aneg, 0);
    if $\neg$ *possuiSol* **then**
        V = V $\setminus$ vi
        E = E $\setminus$ vi
end

Algorithm 1: Análise dos vértices

O Algoritmo 1 demonstra o funcionamento da etapa de análise dos vértices para execução da primeira parte do algoritmo de *nilcatenation*.

4.3.2. Análise das Arestas

Após a etapa de análise dos vértices, deve-se proceder para a análise das arestas, percorrendo todas as arestas presentes dentro do subgrafo analisado. A partir de cada aresta analisada, tem que se buscar duas soluções: uma de entrada e uma de saída.

Para a solução de entrada, primeiro assume-se o vértice de saída de onde a aresta analisada sai. Dessa maneira, cria-se duas listas: a primeira com os pesos positivos, aqueles que entram no vértice; e a segunda lista com os pesos negativos, aqueles que saem do vértice com exceção do peso da aresta analisada. Dessa maneira, faz-se uma união entre as listas positivas e negativas, buscando uma solução para essa união. Nessa etapa, é necessário verificar se existe uma solução na qual o resultado dessa união é igual ao peso negativo da aresta analisada.

Para a solução de saída, assume-se o vértice de entrada de onde a aresta analisada entra. Dessa maneira, também cria-se duas listas: a primeira com os pesos positivos, aqueles que entram no vértice com exceção do peso da aresta analisada, e a segunda lista com os pesos negativos, aqueles que saem do vértice. Após a busca pela solução de entrada e a de saída, verifica-se a existência de uma solução para entrada e para a saída. Se não existir uma solução para os dois, remove-se a aresta do conjunto de arestas do grafo. Se existir uma solução para entrada ou saída, essa aresta ainda continua no conjunto de arestas para o grafo de *nilcatenation*. Dessa maneira, faz uma união entre as listas, buscando uma solução para essa união. Nessa etapa, é necessário verificar se existe uma solução na qual o resultado dessa união é igual ao peso positivo da aresta analisada.

for *i = 1; i* $< Aresta$ *; i++* **do**
  vi = verticeEntrada(ej)
  Apos = Wj | ej $\in$ E+(vi);
  Aneg = -wj | ej $\in$ E-(vi) - ej;
  possuiSolEntrada = Sol(Apos $\bigcup$ Aneg, wj);
  vk = verticeEntrada(ej)
  Akpos = wk | ek $\in$ E+(vk) - ej;
  Akneg = -wk | ek $\in$ E-(vk);
  possuiSolSaida = Sol(Akpos $\bigcup$ Akneg, -wj)
  if $\neg$ *(possuiSolEntrada* $\wedge$ *possuiSolSaida)* **then**
    E = E $\setminus$ vi
end

Algorithm 2: Algoritmo de *nilcatenation*

O Algoritmo 2 demonstra o funcionamento da etapa de análise das arestas para execução do algoritmo de *nilcatenation*.

As etapas de análise de vértice e aresta são repetidas até nenhum vértice ou aresta poder ser removida do grafo ou até não ter mais nenhuma componente fortemente conexa. Dessa forma, após toda a execução do algoritmo, o resultado é o grafo de *nilcatenation*, caso existir.

4.4. *Pruning*

A etapa anterior gera um subgrafo de *nilcatenation*, com as transações que podem ser retiradas da rede sem prejuízo para a semântica da rede. Dessa maneira, a ideia é executar o algoritmo de *pruning* sobre essa transações obtidas da etapa anterior.

A análise pensada para esta etapa consiste, nesse primeiro momento, em retirar as transações que se cancelam. Isto é, um conjunto de transações ao qual o saldo final delas seja igual a zero. Por exemplo, um usuário A envia um valor X para o usuário B, o mesmo envia o valor X para o usuário C que envia o valor X para o usuário A. Dessa maneira, o saldo final das transações se equivale ao saldo inicial, podendo assim ser considerado que essas transações podem ser canceladas. Os grafos de *nilcatenation* representam essa parte das transações que se cancelam, visto que esse grafo representa um subgrafo no qual não impacta na semântica do grafo original, permitindo que essas transações possam servir como entrada para o algoritmo de *pruning*. A Figura 8 exemplifica um caso onde as transações podem se cancelar, ou seja, como saldo final após a análise das transações é igual a zero.

Para realização dessa etapa, é necessário garantir algum modo de poder recuperar essas transações se caso for necessário. Sendo assim, a ideia é salvar os *hashs* dessas transações que foram retiradas da rede em uma lista auxiliar. Com isso, a ideia é marcar cada uma das transações que foram geradas pelo grafo de *nilcatenation*, como uma etapa de preparação, salvar o *hashs* dessas transações em uma lista auxiliar e assim executar o algoritmo de *pruning* para pode removê-las da *blockchain*, reduzindo o espaço de armazenamento da rede. Conforme já comentado, os protocolos para esta etapa vão além do escopo deste trabalho e podem ser encontrados na literatura [Palm et al. 2018].

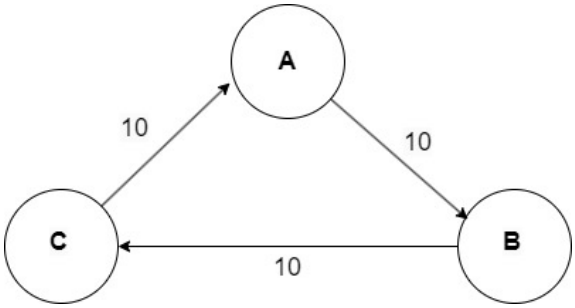

Figura 8. Transações *blockchain*.

5. Experimentos

Esta seção descreve os experimentos realizados com a proposta da seção anterior, bem como sua comparação com uma solução encontrada na literatura baseada na detecção de ciclos.

5.1. Dados de Entrada

Para a realização do experimento desse artigo, foram criados grafos aleatórios de testes que simulassem transações de uma rede *blockchain*. Os arquivos de dados usados para realizar o experimento consistem em arquivos csv contendo os dados de 'Endereço de Origem", "Endereço de Destino" e o "Valor".

Para a realização dos experimentos foram criados quatro grafos aleatórios de diferentes tamanhos:

- Grafo 1 - Pequeno com 30 vértices e 391 arestas;
- Grafo 2 - Pequeno com 60 vértices e 383 arestas;
- Grafo 3 - Médio com 75 vértices e 2080 arestas;
- Grafo 4 - Grande com 400 vértices e 3394 arestas.

5.2. Implementação

Para a realização da implementação desse projeto, incluindo a parte de obtenção dos dados, geração dos grafos e aplicação do algoritmo de *nilcatenation*, foi utilizado um notebook com Intel Core i7 2.20GHz com 16GB de memória, 256SSD com sistema operacional Windows de 64bits. Além disso, para implementação dos algoritmos foi utilizado a linguagem de programação Python em conjunto com as bibliotecas Pandas e Numpy.

As seguintes etapas foram necessárias para a realização destes experimentos.

Entrada de dados. Como abordado na proposta, é necessário a utilização de dados que contenham algumas informações específicas para poder gerar um grafo das transações. Dessa maneira, através de arquivos csv gerados aleatoriamente, foi possível obter tais informações. A partir destes dados, consegue-se fazer a adaptação necessária para seu uso, dividindo-os em tipos: Endereço de Origem; Endereço de Destino; e Valor de Transferência.

Manipulação dos Dados. Com os dados divididos em tipos, foi necessária uma etapa intermediária de manipulação de dados. A partir dos tipos de Endereço de Origem e Endereço de Destino, que são representado por *hashes*, foi feito um dicionário chave/valor para utilização de um inteiro como identificador dos vértices do grafo. Assim, cada vez que um endereço novo aparecesse era criado um par com uma chave representada por um inteiro começando em 1 e que sempre vai incrementando em 1.

Ao final desta etapa foi possível então obter o grafo de transações com os endereços de origem e destino sendo os vértices e os valores de transferência sendo as arestas.

Manipulação do Grafo. Com o grafo estruturado é necessário encontrar as componentes fortemente conexas. Para isso foi utilizado o algoritmo Tarjan [Tarjan 1971], que verifica se existe ou não componentes fortemente conexas e retorna todas que existirem dentro do grafo.

A partir das componentes fortemente conexas, executou-se o algoritmo de *nilcatenation* discutido na seção anterior e a proposta baseada em ciclos encontrada na literatura [Géraud et al. 2017].

Depois da execução dos algoritmos, a saída de dados consiste em um grafo de *nilcatenation* ou nos ciclos a serem removidos. A partir destes dados, calculou-se a porcentagem de dados contidos na *blockchain* que foram identificados para remoção.

5.3. Resultados

Com a execução dos algoritmos sobre os grafos usados como entrada de dados, passando por todas as etapas, foi obtido os seguintes resultados em termos de possibilidades de diminuição do tamanho da *blockchain*.

Primeiramente, tanto para o grafo 1 quanto para o grafo 3, nenhuma das abordagens encontrou transações que pudessem ser removidas da *blockchain*. Como os grafos foram gerados de forma aleatória, não fica refletido nesta análise as questões relacionadas com a localidade de transações, onde conjuntos de indivíduos acabam realizando mais transferências entre si. Assim, a Figura 9 apresenta os resultados para os grafos 2 e 4.

Nesta figura, podemos perceber que a solução proposta neste artigo reduziu aproximadamente 20% das transações contidas no grafo 2, enquanto que a solução baseada em ciclos identificou apenas aproximadamente 5% de transações para serem removidas. Considerando o grafo 4, a solução proposta novamente obteve uma redução maior, de aproximadamente 6%, enquanto que a solução baseada em ciclos reduziu apenas 3,6% da quantidade de transações.

Através destes experimentos fica evidente que a solução baseada na identificação de um subgrafo de *nilcatenation* apresenta um desempenho superior, pois consegue identificar tanto os ciclos quanto outras transações que poderiam ser removidas. Outra vantagem é não considerar apenas os valeres oriundos de uma única aresta, mas conseguir identificar casos em que mais de uma transação são equivalentes a uma única outra transação. Por exemplo, considere uma transação $t1$ que transfere 2 *bitcoins* de A para B e duas

outras transações $t2$ e $t3$ transferem 1 *bitcoin* cada uma de B para A. Neste caso, as três transações poderiam ser removidas, mas a solução baseada em ciclos falha nesta identificação.

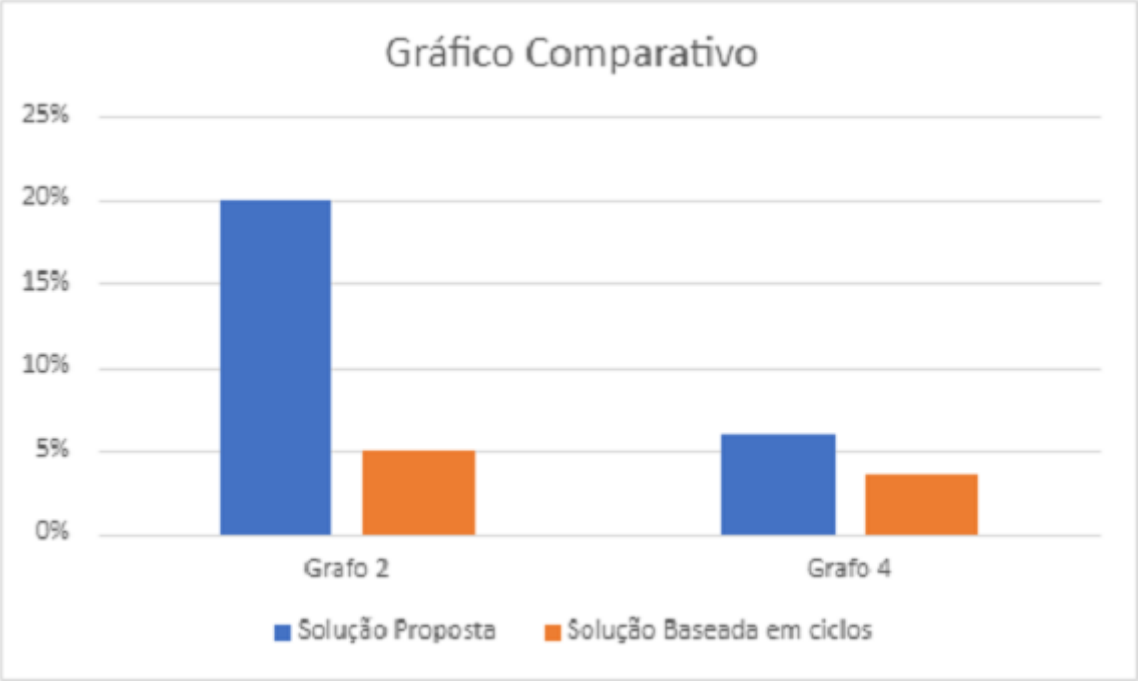

Figura 9. Comparação entre a solução proposta baseada na identificação de subgrafos de *nilcatenation* e a solução existente baseada na identificação de ciclos.

6. Conclusões

O armazenamento da *blockchain* é um dos problemas relevantes que podem afetar sua ampla adoção nas mais variadas aplicações no futuro. Desenvolver soluções para resolver esse desafio é um ponto importante dentro desse contexto. Neste contexto, este artigo apresentou uma proposta baseada em identificação de subgrafos de *nilcatenation* para auxiliar no *pruning* de transações em *blockchains*. A ideia geral é identificar um subconjunto de transações que podem ser removidas sem aferar a consistência da rede. Uma análise experimental preliminar mostra que a solução proposta apresenta ganhos quando comparado com as abordagens existentes na literatura, pois consegue identificar com maior precisão quais transações podem ser removidas.

Como trabalhos futuros, pretendemos executar estes algoritmos em dados obtidos de *blockchains* reais, como a rede *bitcoin*, além de utilizar técnicas de inteligência artificial para identificar subgrafos de *nilcatenation* e com isso diminuir o tempo de processamento destes algoritmos, pois o problema de encontrar o grafo de *nilcatenation* é NP-Completo. Nestes casos, após a identificação de um subgrafo de *nilcatenation*, será necessário executar uma validação rápida para verificar se o mesmo realmente possui as características necessárias, além disso possibilidades de redução da *blockchain* poderão não ser identificadas pelos algoritmos de inteligência artificial. Assim, fica claro que existe um *trade-off* entre quantidade de processamento necessário e quantidade de transações identificadas para serem removidas.

Referências

Amaral, C. A. T. et al. (2020). Algoritmos para o problema de nilcatenation com aplicação na detecção de lavagem de dinheiro em criptomoedas.

Biggs, N., Lloyd, E. K., and Wilson, R. J. (1986). *Graph Theory, 1736-1936*. Oxford University Press.

Blockchain.com, t. . B. S. (2023).

Buterin, V. (2014). AOs, DACs, DAS and more: An incomplete terminology guide. `https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/`.

Géraud, R., Naccache, D., and Roşie, R. (2017). Twisting lattice and graph techniques to compress transactional ledgers. In *International Conference on Security and Privacy in Communication Systems*, pages 108–127. Springer.

Gordon, N. (2020). A survey of blockchain storage requirement mitigation techniques.

Karp, R. M. (2011). Computational complexity of combinatorial and graph-theoretic problems. In *Theoretical Computer Science*, pages 97–184. Springer.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`.

Palm, E., Schelén, O., and Bodin, U. (2018). Selective blockchain transaction pruning and state derivability. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 31–40. IEEE.

Pech, M. (2017). Blockchains: How they work and why they'll change the world. *IEEE Spectrum*.

Quan, L., Huang, Q., Zhang, S., and Wang, Z. (2019). Downsampling blockchain algorithm. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 342–347. IEEE.

Subramanian, H. (2018). Decentralized blockchain-based electronic marketplaces. *Communications of the ACM*, 61(1):78–84.

Tapscott, D. and Tapscott, A. (2016). Blockchain revolution: How the technology behind bitcoin is changing money, business and the world. *Portfolio Penguin*.

Tarjan, R. (1971). Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 114–121.

Wang, R. L. (2004). A genetic algorithm for subset sum problem. *Neurocomputing*, 57:463–468.