UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

ANTONIA DÁLIA CHAGAS GOMES

ANÁLISE E IMPLEMENTAÇÃO DE TÉCNICAS DE REDUÇÃO DE TABELAS DE ROTEAMENTO

QUIXADÁ

2024

ANTONIA DÁLIA CHAGAS GOMES

ANÁLISE E IMPLEMENTAÇÃO DE TÉCNICAS DE REDUÇÃO DE TABELAS DE ROTEAMENTO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Arthur de Castro Callado

QUIXADÁ

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G612a Gomes, Antonia Dália Chagas.
Análise e implementação de técnicas de redução de tabelas de roteamento / Antonia Dália Chagas Gomes. – 2024.
178 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2024.
Orientação: Prof. Dr. Arthur de Castro Callado.

1. Tabelas de Roteamento. 2. Técnicas de redução. 3. Minimização de recursos. I. Título.
CDD 005

ANTONIA DÁLIA CHAGAS GOMES

ANÁLISE E IMPLEMENTAÇÃO DE TÉCNICAS DE REDUÇÃO DE TABELAS DE ROTEAMENTO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em: 28/11/2024.

BANCA EXAMINADORA

Prof. Dr. Arthur de Castro Callado (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

Prof. Dr. Jeandro de Mesquita Bezerra
Universidade Federal do Ceará (UFC)

Prof. Dr. Reinaldo Bezerra Braga
Instituto Federal do Ceará (IFCE)

Dedico este trabalho primeiramente a Deus, por me guiar e fortalecer em cada passo desta jornada. Dedico também à minha família, pelo amor, apoio incondicional e por sempre acreditarem em mim.

AGRADECIMENTOS

Todos os que se envolvem em uma pesquisa sabem que não o fazem sozinhos, mesmo que o ato de escrever seja uma tarefa bastante solitária. Agradecer a todos os que contribuíram e fizeram parte dessa longa e densa jornada, não é uma tarefa simples.

Agradeço primeiramente a Deus, não por ser um clichê, mas por acreditar na sua existência e no seu poder de transformar nossas vidas.

Agradeço e dedico este trabalho aos meus pais. À memória do meu pai, José Maria Gomes (in memoriam) que, mesmo não estando mais presente fisicamente , continua sendo uma inspiração constante em minha vida. À minha mãe, Vera Lúcia, por seu amor incondicional, por sempre acreditar em mim e por estar ao meu lado em todos os momentos, oferecendo força, apoio e carinho.

Aos meus irmãos, Adailson Chagas e Daiane Gomes que estiveram por perto, me apoiando e me incentivando. Eu sei que se orgulham por eu ter conseguido chegar até aqui, mas eu me orgulho muito mais por tê-los como irmãos. A força, o amor e a união que compartilhamos foram muito importantes no trilhar dessa jornada.

Aos grandes amores da minha vida: Princesa, Luna, Catarina e Kiara, minhas fiés companheiras de quatro patas que estiveram ao meu lado durante todo o processo de escrita e que, por vezes, com sua presença carinhosa, doce e alegre, aliviaram a tensão dos momentos de maior ansiedade. “Eu as amo tanto que aqui, não cabe tamanho amor incondicional.”

Aos meus amigos que o mestrado me presenteou, Iraneide Lima e Luciano Quirino, que me acompanharam, me fortaleceram e me apoiaram desde o primeiro dia de aula no mestrado até os últimos segundos dessa reta final. Muito obrigada!

Ao meu orientador, Prof. Dr. Arthur de Castro Callado pela paciência e pelo valioso conhecimento compartilhado. Muito obrigada!

À autora Chatterjee, por gentilmente revisar e corrigir o algoritmo adaptado de sua estratégia, assim como pela disponibilidade nas trocas de e-mails, que enriqueceram o desenvolvimento deste trabalho.

A Universidade Federal do Ceará - *Campus* Quixadá, e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro nos meses iniciais do mestrado.

Aos membros da banca, Prof. Dr. Regis Pires Magalhães,Prof. Dr. Jeandro de Mesquita Bezerra e Prof. Dr. Reinaldo Bezerra Braga, obrigada pelas contribuições, sugestões e

pelo tempo dedicado à avaliação deste trabalho.

“Quando as coisas ficarem difíceis, olhe para as pessoas que amam você! Você encontrará energia nelas. ”

(BTS)

RESUMO

Com o aumento expressivo dos sistemas de comunicação, novas estratégias são investigadas com o intuito de minimizar os recursos de memória para simulações em larga escala. Especificamente, a presente pesquisa considera as tabelas de roteamento, tendo em vista que o seu crescimento configura-se como um dos principais problemas de escalabilidade e que tem originado pesquisas que busquem soluções eficientes. Dessa forma, este trabalho identificou, a partir de uma Revisão Sistemática da Literatura (RSL) um conjunto de técnicas de redução de tabelas de roteamento propostas na literatura e analisou sua viabilidade de implementação para o contexto do roteamento IP. Dessa forma, os achados da pesquisa apresentaram possibilidades para o avanço na integração entre redes orientadas ao conteúdo/IP, abordando estratégias inicialmente voltadas para o contexto de *Named Data Networking* (NDN), uma arquitetura inicialmente projetada para atender às necessidades emergentes da comunicação (Zhang; Burke, 2010). Como resultados as estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) foram selecionadas, implementadas e comparadas em termos de reagregação dos prefixos, uso de memória e redução da tabela. A partir da analise, foi possível identificar uma redução da tabela original de 83,94% na estratégia de Chatterjee *et al.* (2021) e 80,66% na de Saxena e Raychoudhury (2016). Adicionalmente, foi realizado um teste estatístico de Mann-Whitney U para verificar as diferenças significativas entre as técnicas em relação ao tempo de execução, uso de memória e tempo de reagregação dos prefixos. Os resultados confirmaram diferenças estatisticamente significativas, com a estratégia de Chatterjee *et al.* (2021) destacando-se por apresentar menores tempos de execução e menor consumo de memória, evidenciando sua superioridade em termos de eficiência. Dessa forma, os resultados indicam que a escolha da técnica impacta diretamente no desempenho da rede, com a estratégia de Chatterjee *et al.* (2021) apresentando reduções significativas no armazenamento. Conclui-se então que, métodos como a *Patricia Trie* em consonância com outros métodos, como é o caso do *Connected Dominating Set* (CDS) oferecem potenciais soluções para a minimização dos recursos das tabelas de roteamento.

Palavras-chave: Tabelas de Roteamento; Técnicas de redução; Revisão Sistemática da Literatura (RSL); Minimização de recursos.

ABSTRACT

With the significant growth of communication systems, new strategies are being investigated in order to minimize memory resources for large-scale simulations. Specifically, this research considers routing tables, since their growth is one of the main scalability problems and has given rise to research seeking efficient solutions. Thus, this work identified, from a Systematic Literature Review (SLR), a set of routing table reduction techniques proposed in the literature and analyzed their feasibility of implementation for the context of IP routing. Thus, the research findings presented possibilities for advancing the integration between content-oriented/IP networks, addressing strategies initially aimed at the context of *Named Data Networking* (NDN), an architecture initially designed to meet the emerging needs of communication (Zhang; Burke, 2010). As a result, the Chatterjee *et al.* (2021) and Saxena e Raychoudhury (2016) strategies were selected, implemented and compared in terms of prefix reaggregation, memory usage and table reduction. From the analysis, it was possible to identify a reduction of the original table of 83.94% in the Chatterjee *et al.* (2021) strategy and 80.66% in the Saxena e Raychoudhury (2016) strategy. Additionally, a Mann-Whitney U statistical test was performed to verify the significant differences between the techniques in relation to execution time, memory usage and prefix reaggregation time. The results confirmed statistically significant differences, with the Chatterjee *et al.* (2021) strategy standing out for presenting shorter execution times and lower memory consumption, evidencing its superiority in terms of efficiency. Thus, the results indicate that the choice of technique directly impacts network performance, with the Chatterjee *et al.* (2021) strategy presenting significant reductions in storage. It is concluded that methods such as the *Patricia Trie* in conjunction with other methods, such as the *Connected Dominating Set* (CDS), offer potential solutions for minimizing routing table resources.

Keywords: Routing Tables; Reduction Techniques; Systematic Literature Review (SLR); Resource Minimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

IP	*Internet Protocol*
BGP	*Border Gateway Protocol*
OSPF	*Open Shortest Path First*
RIB	*Routing Information Base*
FIB	*Forwarding Information Base*
MPLS	*Multiprotocol Label Switching*
SDN	*Software-Defined Networks*
FIFA	*Fast incremental FIB aggregation*
FPGA	*Field Programmable Gate Arrays*
PATRICIA	*Practical Algorithm to Retrieve Information Coded in Alphanumeric*
AIDR	*Aggregation-aware Inter-Domain Routing*
CIDR	*Classless Inter Domain Routing*
FRT	*Flexible Routing Tables*
NDN	*Named Data Networking*
BIRD	*Internet Routing Daemon*
ORTC	*Optimal Routing Table Constructor*
CDS	*Connected Dominating Set*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentadas a justificativa e a contextualização desta investigação, que tem por finalidade investigar o comportamento de técnicas de redução de tabelas de roteamento de redes. Nas seções a seguir, serão apresentadas: (i) Contextualização, (ii) Motivação, (iii) Objetivos e Questões de Pesquisa, e por fim a seção (iv) apresentará a Condução das Atividades e os detalhes a respeito dessa dissertação, com elucidações sobre os capítulos restantes.

1.1 Contextualização

O *Internet Protocol* (IP) é um dos principais responsáveis pela comunicação na Internet, sendo encarregado do endereçamento e roteamento dos pacotes de dados entre os dispositivos interconectados. O IP opera na camada de rede do modelo, onde cada pacote é enviado de um ponto a outro por meio de endereços de Internet (endereços IP). O protocolo IP define como os pacotes devem ser encaminhados entre redes diferentes e garante que os dados cheguem ao destino correto. O processo de encaminhamento envolve várias decisões de roteamento, que são realizadas por roteadores com base nas informações contidas nas tabelas de roteamento.

O *Border Gateway Protocol* (BGP) e *Open Shortest Path First* (OSPF) são protocolos de roteamento que costumam ser usados pelos roteadores para descobrir o caminho percorrido pelos pacotes da origem até o seu destino. Esses resultados de "entradas" e "saídas" são registrados em uma tabela denominada de *Routing Information Base* (RIB), onde são armazenados os possíveis caminhos até os prefixos de rede aprendidos pelos protocolos de roteamento em operação. Uma consequência direta do aumento da RIB é o rápido crescimento da tabela de encaminhamentos *Forwarding Information Base* (FIB).

A FIB consiste em uma estrutura de dados calculada com base na tabela geral RIB. Nela são armazenados os prefixos da rede de destino e o endereço do próximo roteador, evitando assim quaisquer conflitos no caso de rotas para o mesmo destino. A principal diferença entre a FIB e a RIB é que a RIB consiste em uma tabela de roteamento global, com informações abrangentes, enquanto a FIB é uma tabela refinada para permitir um acesso rápido e preciso durante o encaminhamento de pacotes (Liu *et al.*, 2015).

1.2 Motivação

Na última década, a Internet passou por um rápido crescimento (Xian *et al.*, 2008), e as simulações tornaram-se cada vez mais relevantes para avaliar protocolos de rede, ao passo que novas estratégias também foram sendo desenvolvidas (Najiyya *et al.*, 2023). No entanto, o uso de simulações para redes em grande escala enfrenta uma limitação: o tamanho da rede que pode ser avaliada depende diretamente da capacidade dos computadores onde os simuladores estão instalados, o que restringe a análise de desempenho em cenários maiores (Huawei, 2023).

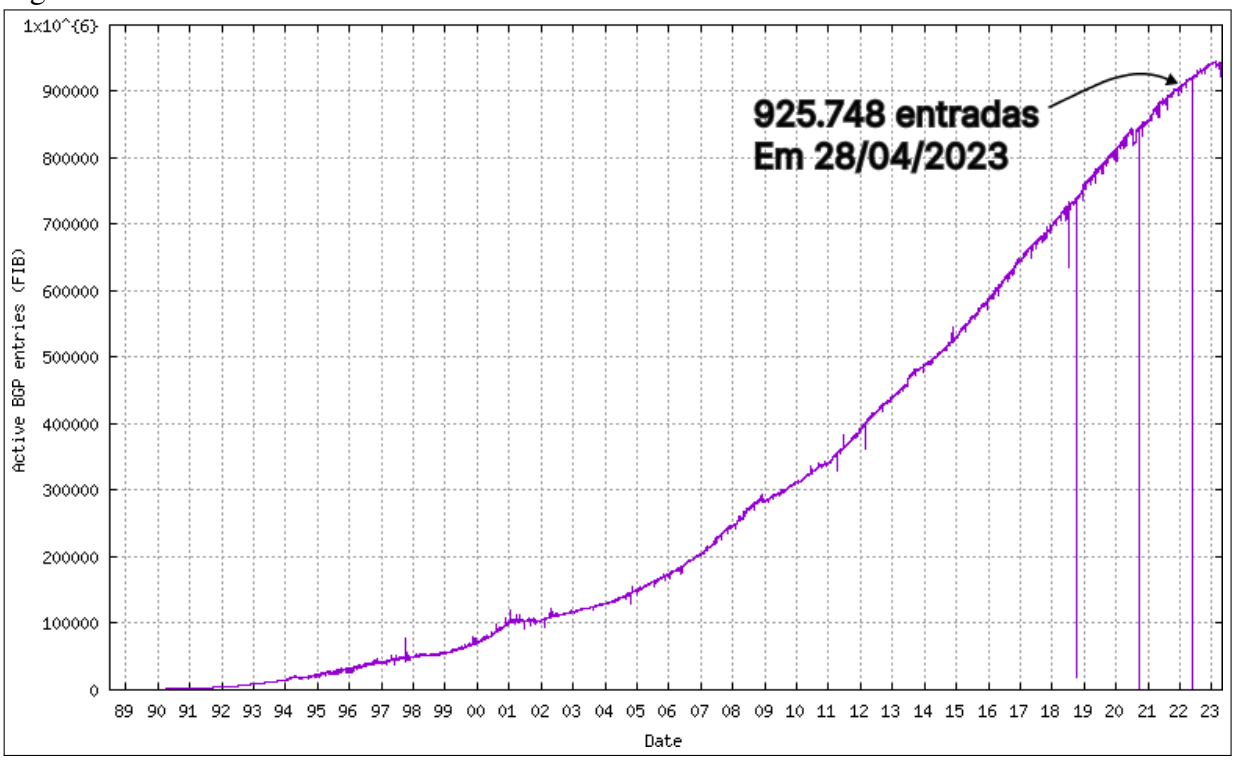
Segundo Hiromori *et al.* (2003) na simulação de redes em larga escala, devido à limitação de recursos disponíveis nos computadores, o tamanho das redes e a escala dos cenários de simulação são frequentemente restritos. Quanto maior a rede a ser avaliada, mais recursos de alto custo são necessários. Neste caso, nos últimos anos pesquisas têm se concentrado em apresentar propostas com o intuito de minimizar esses custos e "permitir que os simuladores possam modelar diferentes tamanhos de redes e protocolos, de ponta a ponta, minimizando a necessidade de máquinas cada vez mais robustas" (Sabai, 2017).

A comunicação da Internet ocorre baseada em um conjunto de informações de roteamento, a que chamamos de rotas. Esse conjunto de rotas possuem dados dos destinos de rede, ou seja, o seu funcionamente correto depende necessariamente de todas as informações da rede e dos seus possíveis destinos (Huawei, 2023). Essas rotas ficam armazenadas em uma tabela de roteamento que tende a um crescimento exponencial, por isso o seu tamanho impacta diretamente nos requisitos de memória e de processamento. Ainda segundo Sabai (2017), "nos últimos anos ela (tabela) se tornou tão grande que muitos roteadores já não suportam armazená-la completamente em sua memória, causando falhas de roteamento."

Para ilustrar o que Sabai (2017) apontava anos atrás, apresentamos o gráfico abaixo com o número de prefixos anunciados na Internet usando o BGP. Esse crescimento resulta em equipamentos cada vez mais robustos e com custos altíssimos, já que demanda a utilização de hardware que seja capaz de armazenar e processar uma quantidade considerável de dados.

Segundo Fei *et al.* (2015) os problemas no sistema de endereçamento e roteamento adotado é evidenciado com o crescente número de usuários, e isso impacta no crescimento das tabelas de roteamento já que os fatores relacionados à administração dos endereços IP contribuem para esse aumento. Mesmo em cenários de rede com diferentes estratégias de encaminhamento de pacotes, como redes baseadas em *Multiprotocol Label Switching* (MPLS) e *Software-Defined Networks* (SDN), no nó de entrada da rede é necessário usar uma tabela de roteamento, portanto

Figura 1 – Crescimento da Tabela de Roteamento Global



Fonte: Adaptado de <https://bgp.potaroo.net/as2.0/bgp-active.html>, (2023)

esses esquemas também se beneficiam de uma tabela de roteamento reduzida.

Quando lançamos olhares para as simulações de redes, estudos para reduzir o tamanho das tabelas de roteamento têm sido apresentados desde o ano de 1979. Estudos como os de Kamoun e Kleinrock (1979) avaliaram a relação entre a redução do comprimento da tabela de roteamento e o aumento no comprimento do caminho da rede. Pahlevani *et al.* (2016) apresentou um protocolo que aumenta a taxa de transferência, entre outros como autores que também abordaram a redução da tabela (Huc *et al.*, 2012), (Westcott; Lauer, 1984) e (Amer *et al.*, 1988).

Outro caso, é a abordagem de Huang e Heidemann (2001), que apresenta uma redução do espaço ocupado pelas tabelas de roteamento. No entanto, seus estudos, que foram realizados há mais de 20 anos não seriam ainda capazes de calcular todas as rotas mais curtas, porque nem todos os caminhos eram cobertos pela árvore de abragência. Posteriormente, Hiromori *et al.* (2003) estendeu esse método para permitir que, a partir da junção de duas tabelas, roteamentos genéricos possam ser permitidos e assim, todas as rotas sejam cobertas.

Outros estudos também foram se configurando ao longo dos anos, alguns voltados a aperfeiçoar os protocolos e arquiteturas já existentes, e outros sugerindo estratégias inovadoras. Daremos ênfase a esses novos métodos nos próximos capítulos.

Dessa maneira, destaca-se a relevância da presente investigação, tendo em vista que as discussões elecandas por diversos autores levaram a hipóteses diferentes que serão apresentadas posteriormente. Assim, torna-se importante a avaliação de técnicas de redução de tabelas com vistas a verificar essas hipóteses em ambientes de linguagem de programação onde os cenários podem ser controlados e a limitação de recursos computacionais é menor.

Adicionalmente, o presente trabalho apresenta contribuições relevantes em se tratando do conhecimento na área de redes de computadores, fornecendo informações importantes que auxiliem na tomada de decisões em projetos e implantações de redes em ambientes de linguagem de programação.

1.3 Objetivos e Questões de Pesquisa

Tendo justificado a problemática na qual o presente trabalho se concentra e comprovado a importância do crescente de estudos que abordem as técnicas de roteamento, este trabalho tem como objetivo geral avaliar o comportamento de técnicas de redução de tabelas de roteamento. Para isso, utilizamos um ambiente de programação para modelar esse conjunto de estatégias e analisar qual técnica apresenta um melhor desempenho. Ao final, esses resultados serão combinados e comparados com o intuito de apresentar qual técnica se comportou mais eficientemente. Com isso, apresentamos os objetivos específicos, que consistem em:

• Analisar a capacidade de implementação dessas técnicas de redução de tabelas de roteamento selecionadas em um ambiente de linguagem de programação.

• Comparar as técnicas de redução de tabelas de roteamento em termos tempo de reagregação dos prefixos, uso de memória e redução da tabela.

As principais questões de pesquisa dessa investigação são:

QP1. Quais técnicas de redução de tabelas de roteamento estão propostas na literatura e quais delas são viáveis de implementação em um ambiente de linguagem de programação?

QP2. Como o tamanho da tabela de roteamento afeta o desempenho de memória e processamento da estação de trabalho?

QP3. Quais das técnicas de redução de tabelas de roteamento avaliadas são mais adequada em termos de desempenho e consumo de recursos?

QP4.Como a escolha da técnica de redução de tabelas de roteamento impacta no consumo de memória e tempo de execução?

QP5. Como as técnicas de redução de tabelas se comportam individualmente, em

termos de desempenho e consumo de recursos de memória?

QP6. Quais técnicas apresentaram um melhor desempenho em termos de tempo para reagregação de prefixos, consumo de memória e redução da tabela?

1.4 Organização da dissertação

Além do capítulo de introdução, com contextualização e objetivos da pesquisa, está dissertação está organizada da seguinte maneira:

No Capítulo 2, consta a fundamentação teórica com conceitos importantes para o desenvolvimento da proposta e um conjunto de estudos que auxiliam no entendimento das técnicas de redução de tabelas. No Capítulo 3, concentram-se os trabalhos relacionados que foram como base de comparação com este trabalho, divididos nas seguintes categorias.

O Capítulo 4 apresenta a constituição da metodologia e as etapas de realização da investigação. No Capítulo 5 é apresentada a Revisão Sistemática da Literatura para identificação das técnicas de redução. O Capítulo 6 explica como foi realizada a escolha do método de análise das técnicas de redução.

No Capítulo 7 são apresentadas as descrições das estratégias selecionadas pela RSL quanto à Descoberta e Manutenção de Rotas, os algoritmos etc. Além disso, o capítulo se concentra em apresentar a viabilidade de implementação das técnicas de redução de tabelas de roteamento em um ambiente de linguagem de programação. Ainda nesse capítulo, apresentamos a comparação entre as estratégias com relação à Estrutura de Dados, a Redução de sobrecarga de Atualização e a Eficiência em Memória e Tempo de Busca. Ainda são apresentados também as configurações do Experimento, o Conjunto de Dados selecionados e a análise a partir da geração de gráficos.

Ainda no Capítulo 7 é apresentado uma análise estatística, baseada no teste de Mann-Whitney U, com o objetivo de verificar se existem diferenças significativas entre as técnicas em termos de tempo de execução, uso de memória e taxa de redução das tabelas.

Por fim, no Capítulo 8 finalizamos apresentando as considerações finais do nosso estudo, sumarizando as contribuições e realizando apontamentos sobre as perspectivas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção apresentaremos o recorte de um conjunto de pesquisas que abordaram os problemas relacionados as técnicas de roteamento em redes de computadores e as suas aplicações. Os primeiros achados de pesquisa foram agrupados de acordo com os seguintes tópicos: (i) Recorte de pesquisas e apresentação de conceitos sobre roteamento; (ii) Tabelas de Roteamento, (iii) Técnicas de Redução de Tabelas e suas Aplicações.

Na sequência, apresentamos um conjunto de técnicas de agregação. Sendo organizadas no seguinte formato: (iv) Agregação usando busca em árvore, e suas subseções: (v) Técnica de redução usando *Spanning Tree*, (vi) FIFA - *Fast incremental FIB aggregation*, (vii) PATRICIA - *Pratical Algorithm to Retrieve Information Coded in Alphanumeric*, (viii) AIDR - *Aggregation-aware Inter-Domain Routing*.

Em seguida, apresentamos as técnicas de sumarização. Elas estão organizadas subseções da seguinte forma: (ix) Sumarização de rotas, (x) Reduzindo as Tabelas de Roteamento usando a Sumarização de Rotas, (xi) CIDR - *Classless Inter Domain Routing*, (xii) FRT - *Flexible Routing Tables*. cada subseção consiste em apresentar a técnicas, descrevendo brevemente suas vantagens e desvantagens.

2.1 Recorte de pesquisas e apresentação de conceitos iniciais sobre roteamento

Os primeiros achados na literatura tratam do trabalho de kleinrock e Kamoun, publicado no ano de 1976. Nele, os autores propuseram pela primeira vez a ideia do roteamento hierárquico como alternativa para solucionar os problemas de roteamento em larga escala (Kleinrock; Kamoun, 1977), (Kamoun; Kleinrock, 1979).

Posteriormente, em 1984 a pesquisa de Westcott e Lauer (1984) apresentava como os protocolos de redes poderiam ser estendidos para redes maiores e quais os principais problemas que se configuravam durante a implementação. Suas principais contribuições estão relacionadas ao agrupamento dinâmico e ao roteamento hierárquico. Passados 4 anos, temos a publicação de dois estudos dos autores Amer *et al.* (1988), que propõem o *Hierarchical Hybrid Adaptive Routing Algorithm* (HHARA). A principal contribuição dos autores é realizar combinações entre o algoritmo adaptativo centralizado e o adaptativo distribuído.

Santoro e Khatib. (1985) e Peleg e Upfal (1989) foram alguns dos primeiros autores a apresentarem a noção de esquemas de roteamento compactos. Nele, existe um pré processamento

em que um algoritmo centralizado calcula as tabelas de roteamento. Além disso, noções como o *Graph spanners, distance oracles* e *compact routing schemes*, presentes na teoria dos grafos, também são explorados com o objetivo de lidar com o espaço e precisão (Roditty; Tov, 2015).

Outros conceitos também são explorados para avaliar os caminhos em grafos, como é o caso de alongamento. Mais especificamente, um caminho entre os vértices u e v é considerado de alongamento (α, β) se o seu comprimento for no máximo $\alpha \cdot d(u,v) + \beta$, onde $d(u,v)$ é a distância real entre u e v (Peleg; Ullman, 1987). O alongamento é um parâmetro que mede o quão próximo o comprimento de um caminho está da distância real entre os vértices conectados por esse caminho. Nesse caso, o alongamento permite uma margem de erro β em relação à distância real. Quanto menor o valor de α e β , mais fiel o caminho é em relação à distância real entre os vértices.

Em *Graph spanners*, os autores Peleg e Ullman (1987) e Peleg e Schaffer (1988 apud Roditty; Tov, 2015) definem *spanner* sendo um subgrafo de um grafo que preserva as distâncias entre os vértices. Mais especificamente

> Um grafo $H = (V, E')$, onde $E' \subseteq E$, é um *spanner* de alongamento (α, β) de G se e somente se, para cada $u, v \in V$, houver um caminho de alongamento (α, β) entre u e v em H.

Nele, um grafo é um subgrafo que preserva as distâncias aproximadas entre os vértices do grafo original. No contexto de um *spanner* de alongamento, o objetivo é encontrar os caminhos mais curtos entre os vértices, de forma que o comprimento de um caminho no *spanner* pode ser no máximo α vezes o comprimento do caminho mais curto no grafo original, acrescido de um valor de desvio β.

Outro conceito importante é o de *distance oracles*, descrito por (Thorup; Zwick, 2001). Nele, é possível pré-processar um grafo ponderado não direcionado em um tempo de $O(mn^{\frac{1}{k}})$ e criar uma distânica de $O(n^{\frac{1+1}{k}})$ capaz de responder a qualquer consulta de distância entre quaisquer dois vértices com alongamento de $(2k-1)$ em um tempo de $O(k)$ (Roditty; Tov, 2015).

Por fim, chegamos ao conceito de *compact routing schemes* definido por Peleg e Upfal (1989). Nessa abordagem, durante a fase de pré-processamento, um algoritmo centralizado calcula a tabela de roteamento para cada nó do grafo. Os esquemas de roteamento compactos são projetados para fornecer tabelas de roteamento mais eficientes em termos de espaço. Essa afirmação foi comprovada por Thorup e Zwick (2001), que apresentaram um esquema de

roteamento compacto para um inteiro $k > 1$, onde o alongamento corresponde a $(4k-5)$. Esse esquema apresentado pelos autores utiliza tabelas de roteamento de tamanho $O(n^{1/k})$ em cada vértice, rótulos de tamanho $O(k\log^2 n)$ bits e cabeçalhos de tamanho $O(\log^2 n)$ bits.

Roditty e Tov (2015) aponta que um problema em se tratando de técnicas e esquemas de roteamento ainda permanece em aberto: "dado um oráculo de distância de alongamento (α,β)com espaço S, podemos também obter um esquema de roteamento de alongamento (α,β) com tabelas de roteamento de espaço $O(S/n)$?". Esse problema visa encontrar correspondências entre *distance oracles* e *compact routing schemes* que preservem o alongamento desejado e otimize o espaço de armazenamento das tabelas de roteamento.

Ghose K. e Desai (1995) desenvolveram uma alternativa para interconexão de grandes sistemas chamada de *hierarchical cubic network* (HCN). Mas, apesar de se mostrar eficiente, os autores destacam que o número de links por nó era restrito.

2.2 Tabelas de roteamento

A funcionalidade de um roteador é baseada em tabelas de roteamento. Nelas estão disponíveis as informações necessárias para encaminhamento de um pacote ao longo do caminho, da origem até o seu destino. Segundo Hiromori *et al.* (2003), um dos principais fatores que faz com que os simuladores de rede consumam recursos de memória é justamente por conta do tamanho dessas tabelas em redes simuladas.

Ainda segundo Hiromori *et al.* (2003), um simulador de rede, antes de iniciar a simulação, configura e mantém tabelas de roteamento com uma configuração simples, mas no entanto com um tamanho consideravelmente grande. Apresentamos um exemplo de tabela de roteamento generalizada a partir da Tabela 1. Essa tabela representa uma visão global da rede, onde é possível acessar as informações que são armazenadas nas tabelas.

A tabela apresentada é uma visão generalizada da rede, onde constam informações sobre o nó atual (representando o roteador de origem), o nó de destino do pacote e o nó vizinho para o qual o pacote será encaminhado. Diferentemente de uma tabela de roteamento individual, que pertence a um único roteador, essa tabela generalizada considera a visão global de todas as tabelas de roteamento da rede.

Para encontrarmos um nó vizinho nessa tabela, o espaço resultante será de O(1), ou seja do nó atual para o nó vizinho teremos um espaço de O(1) sendo ocupado. Agora se desejarmos saber o tamanho do espaço ocupado por essa tabela com N nós, devemos multiplicar

Tabela 1 – Exemplo de Tabela de Roteamento Generalizada

Nó atual	Nó de destino	Nó vizinho
1	2	2
...	...	...
1	n	2
2	1	1
...	...	...
n	n - 1	n-1

Fonte: Elaborado pela autora, (2023).

o tamanho da coluna formada pelo nó atual, pela coluna que constitui o nó de destino. Utilizando essas informações temos como resultado que o tamanho dessa tabela corresponderá a (N^2) - N. Essa tabela, com N nós, ocupará um espaço correspondente a O(N^2).

Embora não tenha se configurado como um dos campos de pesquisas onde os estudos têm se concentrado nos últimos 5 anos segundo (Jian-Peng *et al.*, 2012), os autores (Vorst *et al.*, 2011) nos alertam sobre a importância do desenvolvimento de estratégias que visem minimizar os requisitos de memória para simulações em grande escala, tendo em vista que esses modelos de simulação armazenam uma quantidade significativa de informações de roteamento, e por isso exigem maiores requisitos de memória.

A seguir apresentaremos uma estratégia que tem como objetivo minimizar os requisitos de memória, especificamente no que se refere a tabelas de roteamento.

2.3 Técnicas de Redução de Tabelas e suas Aplicações

Thorup e Zwick (2001) e Roditty e Tov (2015) propuseram duas novas técnicas de roteamento que visam reduzir o tamanho das tabelas de roteamento. Essas técnicas são baseadas em uma extensão de uma ideia fundamental de roteamento entre vértices próximos, em que cada vértice armazena informações sobre a primeira aresta de um caminho mais curto para cada vértice em um conjunto específico. Através dessa extensão, os autores mostram que é possível rotear uma mensagem de um vértice para outro que está arbitrariamente distante em um caminho dado por $(1 + \varepsilon)$, armazenando um número limitado de informações de roteamento dedicadas a cada vértice.

A seguir apresentaremos uma tabela com uma revisão das referências relevantes relacionados ao tamanho e estrutura das tabelas de roteamento em grafos ponderados e não ponderados. Nela são resumidas as informações-chave, incluindo as referências, se o grafo é

ponderado ou não ponderado, o alongamento (*stretch*) e o tamanho da tabela de roteamento.

Tabela 2 – Abordagens para determinar o tamanho das tabelas de roteamento

Referência	Ponderado/Não Ponderado	Alongamento	Tamanho da Tabela
Peleg e Upfal (1989)	Não Ponderado	-	$\tilde{O}(n^2 \log N)$
Thorup e Zwick (2001)	Ponderado	7	$\tilde{O}(n^{1/3})$
Abraham *et al.* (2008)	Ponderado	3	$\tilde{O}(n^{1/2})$
Abraham e Gavoille (2011)	Não Ponderado	(2, 1)	$\tilde{O}(n^{3/4})$
Chechik (2013)	Ponderado	10.52	$\tilde{O}(n^{1/4} \log D)$
Roditty e Tov (2015)	Ponderado	$(5+\varepsilon)$	$\tilde{O}(\frac{1}{\varepsilon}\, n^{1/3} \log D)$

Fonte: Elaborado pela autora, (2023).

A análise e comparação das referências destacam a diversidade de abordagens para determinar o tamanho das tabelas de roteamento em grafos ponderados e não ponderados. Observamos que as referências de Peleg e Upfal (1989) e Abraham e Gavoille (2011) fornecem em seus estudos o tamanho da tabela em grafos não ponderados, com alongamentos de X e (2, 1), respectivamente. Por outro lado, Thorup e Zwick (2001) e Chechik (2013) e Roditty e Tov (2015) exploram os grafos ponderados, revelando alongamentos de 3, 10.52 e $(5+\varepsilon)$ respectivamente. Esses resultados destacam a influência desses parâmetros no tamanho da tabela e apontam para direções promissoras para pesquisas futuras na área. No próximo capítulo, aprofundaremos a discussão sobre esses resultados e forneceremos uma análise mais detalhada de suas implicações para a construção e otimização de tabelas de roteamento em redes de comunicação.

2.3.1 Agregação usando a busca em árvore

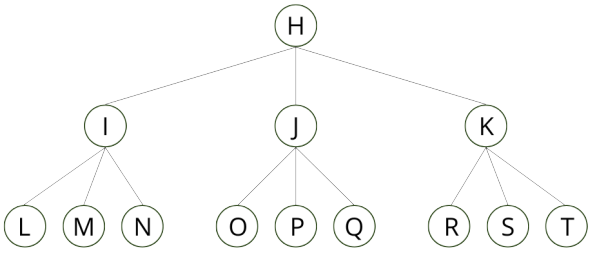
Apresenta-se, nesta subseção, um conjunto de conceitos e autores que debatem acerca do desempenho e funcionamento do algoritmo de busca em árvore.

2.3.2 Técnica de redução usando Spanning Tree

A abordagem de roteamento algorítmico proposto por (Huang; Heidemann, 2001) reduz o tamanho das tabelas de roteamento dando números especiais para todos os nós. Utilizando esses nós especiais, um nó vizinho para o qual o pacote é encaminhado pode ser determinado a partir do nó atual onde o pacote chega. Essa estratégia é baseada em *spanning tree*, que consiste no princípio de que cada nó é capaz de alcançar todos os outros. A definição matemática para *spanning tree* ou árvore geradora é apresentada a seguir, segundo (Jaiswal; Davidrajuh, 2021)

"Uma árvore T é chamada de árvore geradora de um grafo G se T é um subgrafo de G que possui todos os vértices de G.".

Figura 2 – Exemplo de *Spanning Tree*



Fonte: Elaborado pela autora.

Note que na Figura 2 temos uma árvore enraizada, onde o vértice raiz ou raiz da árvore é, indiscutivelmente o **H**. A partir dele temos um conjunto disjuntos e não vazios, **I, J** e **K**, os quais comportam os demais vértices da árvore de abrangência. Os conjuntos **I, J** e **K** representam as subárvores de **H**, também sendo denominados de árvores. De maneira geral, podemos afirmar que a raiz da árvore aparece com a configuração e a aplicação semelhante ao que o grafo da Figura 2 representa, ou seja, em uma árvore de abrangência sempre vai ter caminhos que conectam a árvore raiz (**H**) até um certo vértice, como por exemplo o vértice **N**.

O vértice que vier nesse caminho a partir de **H**, antecedendo **N**, é chamado de vértice pai de **N** e os que vierem exatamente após **N**, são chamados filhos de **N**. Nessa configuração, temos **I** como vértice pai e **N** como vértice filho. Se o vértice não possui descendentes, receberá a denominação de "folha". Como mencionamos anteriormente, tudo inicia no vértice raiz, ou seja ele não tem pai, mas recebe a denominação de pai de todos, pois tudo inicia nele. Como é o caso da nossa representação, onde **H** é o nó raiz, e que dá origem aos demais nós.

O conceito de árvore é um dos conceitos importantes na Teoria dos Grafos. Além dessa aplicação, ela também pode ser usada na estrutura que representa o processo de busca binária (Nicoletti, 2017).

2.3.3 FIFA - Fast incremental FIB aggregation

Segundo Mottaghi *et al.* (2022) o *Fast incremental FIB aggregation* (FIFA) propõe modificar a estrutura dos blocos de construção principais do *Field Programmable Gate Arrays* (FPGA), como os Blocos Lógicos (LBs) e os recursos de roteamento, compostos por Blocos de Comutação (SBs) e Blocos de Conexão (CBs). (Liu *et al.*, 2013) o FIFA possui três algoritmos: FIFA-S, FIFA-T e FIFA-H.

FIFA-S: Prioriza manter o tamanho da Tabela de Roteamento de Fluxos (FIB) o menor possível. Segundo Farias *et al.* (2014) possui um alto custo computacional, pois todas as mudanças são tratadas no momento da agregação.

FIFA-T: Ele minimiza o número de alterações na FIB, ou seja, as atualizações na tabela de roteamento são feitas com menor frequência. Nele ocorre a reagregação da raiz quando as alterações chegam ao limite determinado (Farias *et al.*, 2014).

FIFA-H: é uma abordagem híbrida que combina as vantagens dos algoritmos FIFA-S e FIFA-T. Ele é mais rápido do que o FIFA-T pois a quantidade de prefixos a serem alteradas é menor (Farias *et al.*, 2014).

2.3.4 PATRICIA - Pratical Algorithm to Retrieve Information Coded in Alphanumeric

O algoritmo *Practical Algorithm to Retrieve Information Coded in Alphanumeric* (PATRICIA) foi criado por (Morrison, 1968a) e tem como objetivo reduzir a árvore binária. Por isso, alguns autores referem-se a ele como *PATRICIA Tree*, (Hammer; Hellwagner, 2022),(Saxena; Raychoudhury, 2016). O princípio por trás da árvore PATRICIA é usar o prefixo como chave, permitindo assim que a busca seja realizada em partes dessa chave. Medeiros (2021) exemplifica que se uma árvore possuir apenas dois prefixos (192.168.1.0/24 e 192.168.2.0/24) o primeiro encontro seria 128.0.0.0 (primeiro bit 1), sendo que o segundo resultaria no témino.

A árvore PATRICIA reduz o consumo de memória, o tempo de busca e pesquisa, ao mesmo tempo em que mantém uma baixa complexidade computacional (Saxena; Raychoudhury, 2016). Isso a torna uma escolha eficiente para lidar com estruturas de dados baseadas em prefixos (Medeiros, 2021).

2.3.5 AIDR - Aggregation-aware Inter-Domain Routing

O trabalho de Wang *et al.* (2012) apresenta um esquema de roteamento consciente de agregação chamado *Aggregation-aware Inter-Domain Routing* (AIDR), que leva em consideração a diversidade de caminhos e a agregação de prefixos na seleção da melhor rota BGP. O objetivo principal do AIDR é aproveitar a diversidade de caminhos disponíveis e selecionar as melhores rotas, considerando a eficiência da agregação de prefixos (Wang *et al.*, 2011a).

AIDR propõe uma combinação de nomes planos e endereços de rede para alcançar um desempenho eficiente e reduzir o tamanho e a sobrecarga das tabelas de roteamento. Essa abordagem híbrida permite um roteamento escalável e eficaz, garantindo a entrega confiável de

dados (McMahon, 2013).

2.4 Sumarização de rotas

A Internet é uma extensa rede interconectada globalmente. Se todos os roteadores da Internet incluírem todos os segmentos de rede do mundo em suas tabelas de roteamento, ela se tornará uma tabela de roteamento massiva (Ashraf; Yousaf, 2017). Segundo Huawei (2023) cada vez que um roteador encaminha um pacote, ele precisa verificar sua tabela de roteamento para determinar a melhor saída, e essa verificação constante, inevitavelmente, resulta em atrasos no processamento.

Em contrapartida, se os segmentos de rede com endereços consecutivos forem atribuídos a redes em locais fisicamente contínuas, é possível realizar combinações de segmentos de rede remotos em uma única rota, o que é chamado de sumarização de rota (Huawei, 2023). Dessa forma, não importa quantas rotas existem na tabela, elas vão poder ser anunciadas em uma rota "resumo" (Jian; Fang, 2011).

2.4.1 Reduzindo as Tabelas de Roteamento usando a Sumarização de Rotas

Em Huawei (2023) o autor apresenta como implementar a sumarização de rotas. Nela, há redes em Pequim e Shijiazhuang que podem ser consideradas como localizadas fisicamente próximas. Através da sumarização de rotas, é possível atribuir segmentos de rede consecutivos para essas regiões, consolidando várias rotas individuais em um único resumo. Por exemplo, ao invés de adicionar uma rota para cada segmento de rede em Shijiazhuang, é possível adicionar apenas uma rota para o resumo que engloba todos os segmentos. O mesmo princípio é aplicado às redes em Pequim. Essa abordagem reduz significativamente o número de entradas na tabela de roteamento, simplificando seu gerenciamento.

Após a sumarização, tanto para os roteadores em Pequim quanto para os roteadores em Shijiazhuang, é necessário adicionar apenas uma rota para representar todas as redes da região oposta (Huawei, 2023). Isso simplifica consideravelmente a tabela de roteamento, uma vez que não é mais necessário adicionar uma rota individual para cada segmento de rede (Ashraf; Yousaf, 2018). Por exemplo, ao adicionar uma rota para o resumo das redes em Shijiazhuang, todos os roteadores em Pequim têm acesso a todas as redes daquela região. O mesmo ocorre para os roteadores em Shijiazhuang, que precisam apenas de uma rota para acessar todas as redes em

Pequim. Essa técnica de sumarização reduz a complexidade da tabela de roteamento e otimiza o desempenho dos roteadores (Farias *et al.*, 2014).

2.4.2 CIDR - Classless Inter Domain Routing

O *Classless Inter Domain Routing* (CIDR) é usado para realização de "agregação" de rotas, onde uma única rota abrange o espaço de endereçamento de outros ou de vários números de rede (Luoma, 2019). Essa abordagem reduz a carga administrativa local ao atualizar o roteamento externo, economiza espaço nas tabelas de roteamento de roteadores principais e minimiza a instabilidade nas rotas, evitando mudanças frequentes (Fuller; Li, 2006) e (Luoma, 2019). Como resultado, há uma redução na carga de processamento dos roteadores principais. Além disso, segundo (Fuller *et al.*, 1993) o CIDR permite a atribuição de porções do espaço de endereçamento que anteriormente eram denominadas "números de rede" aos clientes, o que otimiza a utilização do espaço de endereçamento disponível de forma mais eficiente.

2.4.3 FRT - Flexible Routing Tables

Flexible Routing Tables (FRT) consiste em um esquema que projeta algoritmos de roteamento para redes de sobreposição (Ando *et al.*, 2014). Nele é possível construir tabelas de roteamento com base no número de nós no sistema, na vida útil dos nós, na disponibilidade dos nós e nos requisitos de desempenho.

O FRT é um conceito que oferece vantagens, tais como: permitir que o tamanho da tabela de roteamento seja ajustado dinamicamente, possibilidade de projetar algoritmos que lidam de maneira consistente tanto com o roteamento de salto único (encaminhamento direto) quanto com o roteamento de vários saltos (encaminhamento por múltiplos nós intermediários) e permitir a extensibilidade dos algoritmos para considerar diferentes métricas ou índices de desempenho de roteamento (Hojo *et al.*, 2016).

2.5 Fundamentos de *Named Data Networking (NDN)*

Named Data Networking (NDN) é uma arquitetura de rede de computadores projetada para atender às necessidades emergentes da comunicação (Zhang; Burke, 2010). À medida que as investigações sobre sua aplicação avançam para a fase de teste, novas discussões surgem a respeito da sua implantação no mundo real, especialmente em relação à compatibilidade entre

redes TCP/IP e NDN (Detti *et al.*, 2012; Zhang *et al.*, 2014; Luo *et al.*, 2018).

NDN possui três estruturas de dados principais: *content store* (CS), *Pending interest table* (PIT) e *Forwarding Information Base* (FIB) (Najiyya *et al.*, 2023). Pesquisas envolvendo a migração IP/NDN foram realizadas propondo modificação de pilha (Refaei *et al.*, 2017), encapsulamento e tradução (Fahrianto; Kamiyama, 2020) e a que segundo Najiyya *et al.* (2023) é a mais viável de implementação é de mecanismos de tradução (Refaei *et al.*, 2017).

Segundo Zhang *et al.* (2014), a compatibilidade entre TCP/IP e NDN representa um desafio significativo, pois exige que a nova arquitetura NDN seja não apenas integrável às redes IP existentes, mas também capaz de oferecer suporte para aplicações que foram projetadas com base no modelo TCP/IP. Essa necessidade de interoperabilidade é comparável à transição de redes IPv4 para IPv6, que requer estratégias de adaptação e integração (Bi *et al.*, 2007).

Contribuindo para os estudos focados nessa transição, o presente trabalho explora a adaptação das técnicas de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021), inicialmente desenvolvidas para o contexto de NDN, para redes IP. As autoras fazem uso de métodos como *Patricia Trie* e *Connected Dominating Set* (CDS), conforme apresentado na Tabela 5, que demonstram potencial para otimizar o roteamento em redes IP. Esses métodos, ao serem aplicados no contexto de IP, visam reduzir o tamanho das tabelas de roteamento e melhorar a eficiência de busca e uso de memória, abordando desafios específicos da arquitetura TCP/IP enquanto aproveitam os avanços proporcionados pelo NDN.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados que têm como base o mesmo contexto deste trabalho: avaliação de técnicas de redução de tabelas de roteamento de redes em larga escala. Alguns destes trabalhos concentram-se em realizar as avaliações com base em simuladores que atendam suficientemente as características estabelecidas. Ao final, uma comparação entre estes trabalhos relacionados e este trabalho é realizada com base em características específicas.

Farias *et al.* (2014) apresenta em seu trabalho que as estratégias de redução de tabelas de encaminhamento têm se concentrado em duas técnicas: agregação e filtragem. A filtragem seleciona um grupo de prefixos com base em caminhos semelhantes pela topologia da rede e busca uma forma reduzida de representá-los. Já a agregação consiste em agrupar rotas que possuem o mesmo destino em apenas uma entrada, com isso limita-se o número de entradas da tabela de roteamento global Pacheco (2014). Segundo Li *et al.* (2011) a agregação reduz as tabelas de encaminhamento (FIB) apenas com atualização local no roteador, não necessitando de mudança de hardware ou alterações de protocolos.

Farias *et al.* (2014) discutem em seu trabalho as seguintes técnicas de agregação e filtragem: sumarização de rotas, agregação com algoritmos em árvore, *Simple Virtual Aggregation*, *Protocol based selective FIB download*. Os autores também apresentam que as Redes Definidas por Software (SDN) têm surgido para solucionar os problemas associados às demandas de processamento.

A nossa proposição se difere da pesquisa de Farias *et al.* (2014) no que se refere ao uso do software *Internet Routing Daemon* (BIRD) para realização dos experimentos. O BIRD é uma suíte de roteamento usada em pontos de troca de trafégo (IXP, *Internet Exchange Point*) na Europa como roteador centralizador e distribuidor de rotas Ondrej *et al.* (2012). Nele, foram usadas máquinas virtuais do tipo contêiner, onde cada contêiner gerado agiu como um roteador, por intermédio do BIRD.

Nele, é possível estabelecer caminhos redundantes obtendo tolerâncias em situações de falhas, além de uma interface com um ambiente controlado (Almuhtadi *et al.*, 2022). Vale ressaltar que Farias *et al.* (2014) também propuseram um algoritmo para redução da tabela de encaminhamento a partir do cálculo da tabela de roteamento (RIB).

Uma avaliação de algoritmos para minimização de tabelas IP foi realizada por Fanelli *et al.* (2011). Nesse estudo, foram introduzidas duas heurísticas de compressão, o BFM e o

BFM-Cluster, que exploram a reatribuição de endereços para reduzir o tamanho das tabelas de roteamento. Os autores também avaliaram o desempenho dessas heurísticas em conjunto com outras técnicas existentes, com o objetivo de medir e comparar as taxas de compressão dos algoritmos e avaliar os efeitos dessa compressão no tempo de busca.

Nossa pesquisa também se difere nos aspectos de execução, pois em seu trabalho Fanelli *et al.* (2011) utilizou 15 tabelas de roteamento reais que foram baixadas a partir de projetos disponibilizados no *Internet Performance and Analysis* (Project, IPMA) e listas de servidores de rotas fornecidas por Kernen. (2005). Adicionalmente os autores também tinham como objetivo construir estruturas de dados auxiliares tanto para as tabelas iniciais, quanto para as tabelas comprimidas.

Os objetivos se concentram em apresentar as técnicas, avaliá-las e apresentar quais delas possuem um melhor desempenho em relação a diferentes métricas. Não foram realizadas proposições de novas técnicas, bem como também não serão apresentadas novas construções de uma estrutura de dados.

3.1 Comparação entre os Estudos

A Tabela 3 apresenta os trabalhos relacionados com o presente estudo, demonstrando as relações comparativas com semelhanças e diferenças. Para tanto, foram criados os seguintes critérios para se obter uma visão geral, a saber: (i) Tabela: Identifica se o estudo trabalha com uma FIB, RIB, ou com outras estruturas de roteamento. No caso de estruturas diferentes, a tabela marca como "Outra", distinguindo esses estudos dos que lidam diretamente com FIB ou RIB; (ii) Ambiente de Programação: Indica se o desenvolvimento foi realizado em um ambiente de programação ou não; (iii) Cenário: Descreve o contexto em que os experimentos foram realizados e o tamanho dos dados ou tabelas usadas nos testes; e por fim, temos o (iv) Eficiência da Redução: Apresenta os ganhos obtidos com a estratégia de redução de tabelas.

No critério de "Tabela", é possível verificar que a maioria dos estudos selecionados concentra-se na otimização da Forwarding Information Base (FIB), demonstrando a importância de discussões em torno dessa estrutura e de sua otimização. Entretanto, os estudos de Gawrychowski *et al.* (2021) e Hammer e Hellwagner (2022), aplicam suas técnicas a estruturas alternativas, como árvores com rotulagem. Essas abordagens apresentam escopos mais específicos que podem não atender à mesma escala de redes contemplada por este estudo, que foca especificamente na FIB.

Tabela 3 – Comparação entre os Estudos Relacionados

Estudos	Tabela	Uso de Linguagem de Programação	Cenário	Eficiência da Redução
(Li *et al.*, 2011)	FIB	Outra	*Dataset* (328K entradas)	3% do tamanho original
(Farias *et al.*, 2014)	FIB	Outra	Máquinas Virtuais do tipo Contêiner	72,13%
(Fanelli *et al.*, 2011)	FIB	Sim	15 tabelas reais e 56 geradas	Não
(Chatterjee *et al.*, 2021)	FIB	Sim	*Dataset* (até 1000 nós)	70–80%
(Saxena; Raychoudhury, 2016)	FIB	Sim	*Datasets* (4M, 4.5M, 29M)	68,18%
(Gawrychowski *et al.*, 2021)	Não	Não	Árvores com otimização de rótulos	Redução para log n + O((log log n)^2)
(Hammer; Hellwagner, 2022)	Não	Sim	Testes em Raspberry Pi (até 229 chaves de 57 bits)	Uso de 8 a 16 bytes por chave

Fonte: Elaborado pela autora, (2024).

Em relação ao “Ambiente de Programação”, observa-se que este trabalho compartilha a prática de implementar e testar suas estratégias em ambiente programável, assim como os estudos de Fanelli *et al.* (2011), Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016), diferenciando-se pelo fato de que não estaremos realizando a proposição de um novo algoritmo de redução, e sim avaliando os disponíveis na literatura.

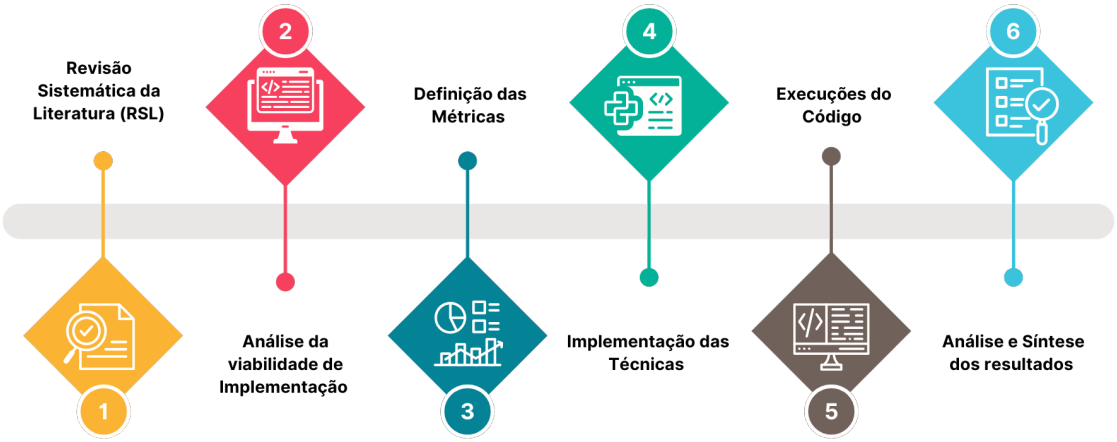
O “Cenário” constitui um diferencial entre os estudos. A proposta deste trabalho se distingue por processar um Conjunto de Dados de 6.054.768 entradas, excedendo significativamente os volumes de dados analisados em Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016), que, embora robustos, envolvem quantidades menores de entradas. Outros estudos, como Farias *et al.* (2014), realizam testes em ambientes de máquinas virtuais e utilizam máquinas virtuais do tipo Contêiner.

Por fim, no critério de “Eficiência da Redução”, os resultados de compressão relatados em Saxena e Raychoudhury (2016) (68,18%) e Chatterjee *et al.* (2021) (até 80%) indicam uma alta eficiência das técnicas aplicadas nos respectivos contextos de teste. Embora este trabalho não tenha como foco inicial o desenvolvimento de uma nova técnica de redução da FIB, a proposta deste estudo se distingue ao identificar, por meio de uma Revisão Sistemática da Literatura (RSL), as técnicas de redução de tabelas de roteamento propostas na literatura e avaliar o comportamento dessas técnicas em um cenário de grande volume de dados.

4 METODOLOGIA

A metodologia da presente pesquisa é constituída das seguintes etapas: Identificação das técnicas de redução de tabelas de roteamento por meio da Revisão Sistemática da Literatura, Definição das métricas, Implementação das Técnicas no ambiente de linguagem de programação, Execução dos códigos, Análise e Comparação dos resultados. A seguir são apresentadas as etapas metodológicas da proposta dessa dissertação.

Figura 3 – Fluxo metodológico do trabalho



Fonte: Elaborado pela autora, (2023).

Apresentamos nas seções seguir cada uma das etapas mais detalhadamente, definindo os passos realizados até o presente o momento e delineando os próximos a serem executados.

4.0.1 Passos metodológicos

Passo 1: Revisão Sistemática da Literatura - A Revisão Sistemática da Literatura integra a primeira etapa de elaboração e responde diretamente a uma das questões de pesquisa, a saber ***QP1***. *Quais técnicas de redução de tabelas de roteamento estão propostas na literatura e quais delas são viáveis de implementação em um ambiente de linguagem de programação?* A Revisão tem o objetivo de identificar com base na literatura um conjunto de técnicas de redução da tabela de roteamento e analisar a sua viabilidade de implementação em ambientes de linguagem de programação. Essa fase impacta diretamente nas demais etapas, tendo em vista

que a definição e viabilidade de implementação para comparação das técnicas em ambientes de linguagem de programação são analisadas previamente de acordo com a abordagem utilizada pelos autores em seus estudos.

Passo 2: Definição das Métricas - A próxima etapa refere-se a escolha e definição das métricas. Esse passo é um ponto crucial neste trabalho, pois ele tem o objetivo de comparar as técnicas de redução de tabelas de roteamento com base nas métricas definididas e responde diretamente a questão de pesquisa central do estudo, a saber, ***QP6.*** *Quais técnicas de redução de tabelas apresentam um melhor desempenho?*. Para a escolha de métricas satisfatórias foram pesquisados artigos, além dos trabalhos relacionados do Capítulo 3 desta proposta.

Passo 3: Implementação das técnicas - Esse passo consiste na implementação das técnicas identificadas pela revisão no ambiente de linguagem de programação e faz parte do objetivo que versa sobre a comparação das técnicas de redução de tabelas de roteamento em termos tempo de reagregação dos prefixos, uso de memória e redução da tabela.

Passo 4: Executar os códigos - Após a implementação das técnicas uma sequência de execuções dos códigos serão realizadas com o objetivo de apresentar o desempenho das técnicas avaliadas. Esse passo responde a questão de pesquisa que busca investigar quais técnicas apresentaram um melhor desempenho em termos de tempo, memória e redução da tabela.

Passo 5: Análise e sintese dos resultados - O último passo consiste na validação dos resultados usando abordagem estatística.

4.0.2 Análise das Técnicas de Redução de Tabelas de Roteamento

Nesse passo, foram identificados e descritos os passos metodológicos utilizados na presente pesquisa cujo objetivo é investigar as técnicas de redução de tabelas de roteamento. Para tanto, adotamos uma abordagem qualitativa e quantitativa.

4.0.2.1 Análise qualitativa

A pesquisa apresenta características de uma abordagem qualitativa pois busca identificar as técnicas de redução de tabelas de roteamento propostas na literatura e suas possibilidades de adaptação ao contexto de roteamento IP. Esta etapa de análise qualitativa incluiu a Revisão Sistemática da Literatura, a categorização das técnicas de acordo com o ambiente em que estava sendo proposto e a sua classificação de acordo com o impacto do periódico em que foi publicado.

O estudo qualitativo permitiu uma avaliação da viabilidade das técnicas de acordo

com suas características serem implementadas em um ambiente de programação. Dessa forma, essa etapa possibilitou orientar a seleção das técnicas quanto à sua implementação, destacando aquelas que se adequadavam ao contexto de roteamento IP.

4.0.2.2 Análise quantitativa

A análise quantitativa deste estudo visa avaliar o desempenho das técnicas de redução de tabelas de roteamento propostas na literatura. Para isso, foram definidos parâmetros como tempo de reagregação dos prefixos, consumo de memória, e percentual de redução da tabela, aplicados para quantificar a eficiência das estratégias e comparar seus resultados em cenários com diferentes volumes de dados.

O tempo de reagregação refere-se ao segundos necessário para reagrupar prefixos em cada estratégia, uma vez que essa etapa influencia diretamente o desempenho e o custo computacional das tabelas de roteamento. O consumo de memória foi calculado para cada técnica de forma a verificar o impacto no armazenamento e por fim, o percentual de redução foi calculado para identificar a eficiência da compressão alcançada em cada técnica em relação ao tamanho original da tabela.

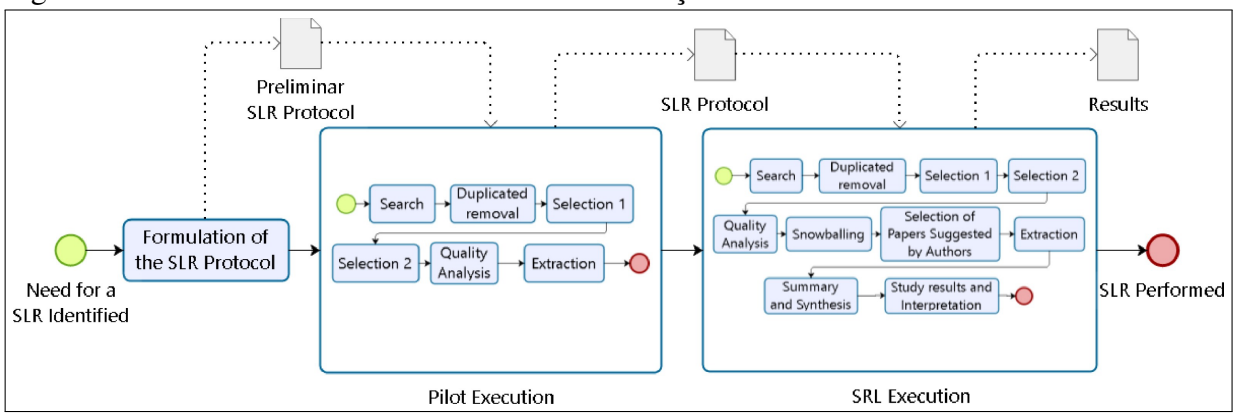
Em colaboração com os dados dessa análise, foi realizado testes estatisticos. Porém, antes de realizarmos a comparação entre as técnicas selecionadas, investigamos a normalidade dos dados para determinar o teste de hipótese mais apropriado para cada variável. Como as amostras não foram consideradas normais, optamos por pela aplicação do teste de Mann-Whitney Wilcoxon para comparar as amostras e identificar se existem diferenças estatisticamente significativas entre as duas estratégias.

5 REVISÃO SISTEMÁTICA DA LITERATURA

Para respondermos diretamente a uma das nossas questões de pesquisa, optou-se pela realização de uma Revisão Sistemática da Literatura (RSL), na qual foram identificados estudos relevantes que abordam um conjunto de técnicas de redução de tabelas de roteamento. A RSL, segundo Kitchenham e Brereton (2013), consiste em identificar, avaliar e interpretar os resultados de uma pesquisa que trate diretamente de questões, área temática ou fenômeno com o objetivo de detectar as melhores evidências e fundamentar conclusões.

Para a realização desta revisão, utilizaremos as diretrizes propostas por Kitchenham e Brereton (2013). Os autores indicam que a realização da RSL deve seguir, de forma resumida, as seguintes etapas: (1) identificação da necessidade de uma revisão sistemática; (2) formulação do Protocolo SLR; (3) execução do piloto e (4) execução do RSL. A figura abaixo, extraída de Gonçalves *et al.* (2018) , apresenta o processo seguido para a realização desta revisão sistemática.

Figura 4 – Revisão Sistemática da Literatura: condução



Fonte: Extraído de Gonçalves *et al.* (2018).

Um dos objetivos do presente estudo é identificar um conjunto de técnicas de redução de tabelas de roteamento presentes na literatura. Desse objetivo, surge a necessidade de aplicação da Revisão Sistemática da Literatura (RSL).

A etapa inicial da RSL consiste na identificação da necessidade da revisão e na formulação do protocolo, nele foram definidos as fontes de busca, as questões de pesquisa, as *strings* que seriam utilizadas em cada base de dados, os critérios de inclusão e exclusão dos estudos, e quais dados seriam extraídos das pesquisas. Conforme seguimos os passos de Gonçalves *et al.* (2018), nessa fase também realizamos a remoção dos estudos duplicados, a seleção, a análise da qualidade e a extração.

Na segunda etapa foram realizados os testes preliminares do protocolo com o ob-

jetivo de delimitar as questões de pesquisa, refinar as *strings* e garantir a qualidade do estudo aprimorando os critérios de seleção dos estudos. Após essa etapa, iniciamos a aplicação no protocolo com os devidos ajustes. Para validar a RSL, foi selecionado um conjunto de artigos cuja incidência foi previamente notada nas buscas manuais e na aplicação do protocolo piloto, são eles: (Fanelli *et al.*, 2011), (Jian-Peng *et al.*, 2012), (Farias *et al.*, 2014), (Roditty; Tov, 2015), (Sabai, 2017), (Hsiao *et al.*, 2018), (Anand *et al.*, 2019).

Em seguida, por meio da ferramenta *parsifal*, foi realizada a identificação dos estudos duplicados e iniciamos a segunda etapa de seleção. Nela, foi realizada a leitura do título, resumo, palavras-chave e seleção de estudos para realização da leitura completa. Posteriormente, a análise da qualidade e a extração de dados foram realizadas.

A execução da RSL consiste nas mesmas etapas que foram realizadas na execução piloto, com a diferença de que os estudos selecionados foram sintetizados e interpretados, com vistas a identificar os melhores resultados que respondem as questões de pesquisa.

5.1 Identificação da necessidade de uma Revisão Sistemática

A primeira etapa consiste na identificação da necessidade da RSL (Kitchenham *et al.*, 2007). Após a leitura de vários artigos relacionados as técnicas de minimização de tabelas, redução do uso de memória em simulações e simulações em larga escala, observou-se a necessidade de uma revisão. Apesar de muitos trabalhos apresentarem a proposição de um conjunto de técnicas (Kamoun; Kleinrock, 1979), (Wang *et al.*, 2006), (Farias *et al.*, 2014), (Anand *et al.*, 2019), não foi possível localizar um estudo recente que apresenta-se um estado da arte sobre as técnicas de redução de tabelas.

Outros trabalhos também foram encontrados na literatura que mencionam superficialmente as técnicas são os de (Thorup; Zwick, 2001) e (Roditty; Tov, 2015), no entanto ambos não apresentam um método sistemático de identificação e avaliação das técnicas de redução de tabelas de roteamento.

5.1.1 Questões de pesquisa

Detalhamos a seguir as questões de pesquisa, os critérios de inclusão e exclusão e uma síntese com os resultados obtidos com a aplicação da revisão.

As informações apresentadas pelas **QP.1** e **QP.2** podem ajudar os pesquisadores da

Quadro 1 – Lista com as Questões de Pesquisa e a Motivação

Questão de Pesquisa	**Motivação**
QP.1 Quais técnicas de redução de tabelas de roteamento estão propostas na literatura e quais delas são viáveis de implementação em um ambiente de linguagem de programação?	Identificar um conjunto de técnicas de redução de tabelas de roteamento propostas na literatura e analisar quais são passíveis de implementações em ambientes de linguagem de programação.
QP.2 Como o tamanho da tabela de roteamento afeta o desempenho de memória e processamento da estação de trabalho?	Analisar, a partir da literatura e com a implementação das técnicas identificadas, como o tamanho da tabela de roteamento afeta o desempenho de memória e o tempo de reagregação de prefixos.
QP.3 Quais das técnicas de redução de tabelas de roteamento avaliadas são mais adequadas em termos de desempenho e consumo de recursos?	Apresentar as técnicas de redução de tabelas que apresentaram melhor desempenho quanto ao consumo de memória.
QP.4 Como a escolha da técnica de redução de tabelas de roteamento impacta no consumo de memória, tempo de execução e processamento?	Identificar as técnicas que apresentam melhor desempenho quanto ao consumo de memória, tempo de execução do algoritmo de reagregação.
QP.5 Como as técnicas de redução de tabelas se comportam individualmente, em termos de desempenho e de consumo de recursos?	Comparar as técnicas de redução de tabela e apresentar quais delas apresentam melhor desempenho quanto ao tempo de execução e consumo de memória.

Fonte: Elaborado pela autora, (2023).

área de redes de computadores a conhecerem o conjunto de técnicas desenvolvidas e difundidas na literatura, além de, a partir desse conhecimento, conseguir identificar como o crescimento da tabela de roteamento pode influenciar no consumo de recursos da estação de trabalho. As questões subsequentes, concentram-se na implementação e avaliação de desempenho das técnicas de redução identificadas pela RSL. As **QP.3** e **QP.4** visam apresentar o desempenho relativo ao consumo dos recursos, tais como: tempo de execução, consumo de memória e processamento.

A **QP.5** se concentra em apresentar individualmente e posteriormente comparar as técnicas, com o objetivo de apresentar aquelas cujo desempenho se mostrou mais satisfatório, contribuindo para que os pesquisadores da área possam realizar a escolha da técnica que se adequa ao seu ambiente e objetivo, com base nos desempenho de memória, tempo de reagregação e compressão de entradas.

Nesse capítulo, são apresentadas as respostas para a primeira questão de pesquisa **QP.1**, cujo objetivo inicial consiste em localizar as técnicas de redução e analisar quais delas são possíveis de implementação.

5.2 Seleção e busca nas Bases de Dados

A identificação dos estudos envolveu as seguintes fontes de pesquisa: SCOPUS, IEEE, Science Direct e Biblioteca Digital da ACM, englobando o período de 12 anos, ou seja, de 2011 até o ano de 2022. A seleção das bases de dados foi devido à alta qualidade de estudos

envolvendo a área da computação, e por estarem inclusas no seu repositório anais de conferências conceituadas na área (Gonçalves *et al.*, 2018).

Para identificação e seleção de artigos, optamos por utilizar a ferramenta Parsifal *(https://parsifal.al)*, uma ferramenta simples e de fácil manipulação. Para realização da busca automática nas bases de dados, foram elaboradas *strings* de buscas, que estão diretamente relacionadas às questões de pesquisa e às palavras-chave que foram definidas para localizar os trabalhos científicos. No quadro abaixo são mostradas as bases de dados e as respectivas *strings* de busca utilizadas. Para a presente investigação, optamos por utilizar A Biblioteca Digital da ACM, Science Direct, SCOPUS e o IEEE. As *strings* foram criadas com base nas questões de pesquisa e adaptadas conforme cada uma das bases.

Quadro 2 – Lista com as bases de dados e as strings de busca

Quant.	**Bases de dados**	**Strings**
1	Biblioteca Digital da ACM	(“Reducing size table” OR “routing table” OR “Minimizing routing table”)
2	Science Direct	((“Routing table size” OR “Reducing Table Size” OR “Routing Techniques”) AND (“Space Routing tables” OR “Minimizing routing table”))
3	SCOPUS	(reducing AND size AND routing AND table)
4	IEEE	((“Table Size” OR “Routing table” OR “new routing schemes” OR “Minimizing routing table”) AND (“New Routing Techniques” OR “Reducing size table” OR “Routing” OR “Techniques” OR “table” OR “Space Routing tables”))

Fonte: Elaborado pela autora, (2023).

Com o objetivo de selecionar os principais trabalhos relevantes que relacionados às questões de pesquisa foram utilizados os seguintes critérios de inclusão (CI) e de exclusão (CE). O Quadro 3 a seguir contém a apresentação dos critérios de inclusão e exclusão dos artigos para os presente estudo.

A seleção dos artigos foi conduzida em três fases distintas. Inicialmente, realizamos uma verificação automática dos trabalhos em busca de duplicatas e redundantes, utilizando a ferramenta *parsifal*. Essa ferramenta possui um recurso automático que avalia os trabalhos duplicados. Em seguida, procedemos à leitura dos títulos, resumos e palavras-chave de todos os artigos restantes, que não apresentavam duplicação. Durante essa etapa, aplicamos critérios de inclusão e exclusão presentes na Tabela 3 e mantivemos os artigos em que não pudemos tomar uma decisão definitiva. Por fim, realizamos uma leitura completa dos artigos remanescentes

Quadro 3 – Lista com os Critérios de Inclusão (CI) e Critérios de Exclusão (CE)

Critérios de Inclusão (CI)	**Critérios de Exclusão (CE)**
1. Estudos primários que apresentam a proposição de técnica de redução de tabelas	1. Estudos secundários, outras SLRs ou mapeamento sistemático, não foram selecionados
2. Estudos em inglês publicados no período de (2011-2022) foram incluídos e filtrados no processo de busca	2. Estudos não escritos em inglês ou publicados fora do período de (2011-2022) foram excluídos processo de busca
3. Estudos que respondam a alguma questão de pesquisa	3. Trabalhos duplicados foram excluídos
4. Estudos que definam a técnica, apresentem os conceitos e princípios usados, avaliação e proposta de implementação	4. Estudos que não definam a técnica, não apresentem os conceitos, não façam avaliação nem proposta de implementação
5. Publicados de forma integral	5. Trabalhos com menos de 5 páginas são excluídos

Fonte: Elaborado pela autora, (2023).

e aplicamos novamente os critérios. A primeira etapa de seleção ocorreu por título, conforme sugerem Kitchenham *et al.* (2007)

A condução da seleção dos estudos está na figura 5. Nela, temos a fases da revisão, as bases de dados e a quantidade de estudos em cada base e em cada fase.

A revisão sistemática identificou 2332 artigos presentes nas bases de dados. Após a primeira análise, os resultados apontaram 182 artigos a serem mantidos, 1809 excluídos e 341 duplicados. Após a realização da etapa de selação por leitura completa, foram selecionados 18 artigos, excluindo assim os 164 artigos que não se encaixavam nos critérios.

Foram selecionadas as seguintes quantidades de artigos para os respectivos anos: 3 em 2011, 1 em 2012, 2 em 2013, 1 em 2015, 3 em 2016, 1 em 2017, 1 em 2018, 1 em 2019, 1 em 2020, 2 em 2021, 2 em 2022.

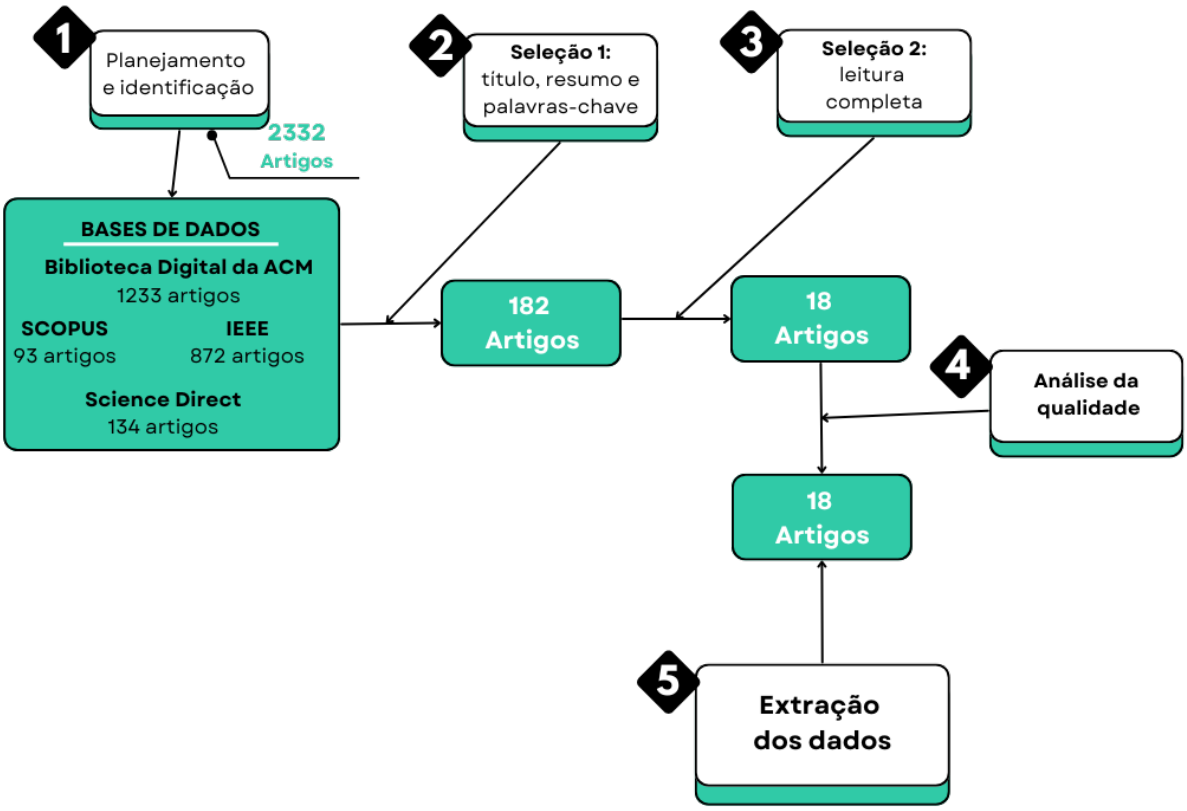
Os resultados e agrupamentos de técnicas localizadas pela Revisão Sistemática são apresentados a seguir.

5.3 Extração dos Dados

Nessa etapa, os dados foram extraídos de cada um dos 18 estudos de acordo com os citérios de extração detalhados a seguir. Os critérios de análise dos estudos visam responder ao nosso objetivo referente a possibilidade de implementação das técnicas de redução de tabelas de roteamento em ambientes de programação.

Técnica de redução utilizada - Esse critério consiste em extrair dos estudo a técnica

Figura 5 – Fluxo de Seleção dos artigos



Fonte: Elaborado pela autora, (2023).

usada, apresentando o princípio de utilização, vantagens, desvantagens e o quanto da tabela de roteamento foi possível ser reduzida.

Abordagem usada - Apresentar a abordagem usada pelos autores para avaliação de desempenho da técnica de redução de tabela proposta. Avaliação de viabilidade de uso em linguagem de programação.

Redução da tabela em comparação com outros métodos - Análise do método proposto em comparação com as estratégias de redução existentes.

Esses critérios permitiram registrar todos os detalhes dos artigos sob revisão e especificar como cada um deles abordou a questão de pesquisa. A saber, *Quais técnicas de redução de tabelas de roteamento estão sendo propostas na literatura?*. Essa questão de pesquisa tem o objetivo de identificar com base na literatura um conjunto de técnicas de redução de tabelas de roteamento e analisar quais delas podem ser implementadas em um ambiente de programação.

O conjunto de técnicas encontradas na literatura foram agrupadas no Quadro 4 e no Quadro 7. Após a seleção e leitura completa, as técnicas de redução foram agrupadas da seguinte forma: no Quadro 4 estão dispostas as técnicas de agregação, enquanto no Quadro 7

estão agrupadas as técnicas de sumarização.

Quadro 4 – Agrupamento de Técnicas de Agregação

Agregação	**Autores**
Spanning Tree	(Castaneda *et al.*, 2020) (Gawrychowski *et al.*, 2021), (Gopalan; Ramasubramanian, 2016)
PATRICIA - *Pratical Algorithm to Retrieve Information Coded in Alphanumeric*	(Hammer; Hellwagner, 2022), (Saxena; Raychoudhury, 2016) (Chatterjee *et al.*, 2021)
FIFA - *Fast incremental FIB aggregation*	(Liu *et al.*, 2013), (Mottaghi *et al.*, 2022)
AIDR - *Aggregation-aware Inter-Domain Routing*	(Wang *et al.*, 2012), (Wang *et al.*, 2011a)

Fonte: Elaborado pela autora, (2023).

O quadro a seguir apresenta a abordagem usada pelos autores em seus estudos usando as técnicas de redução de tabelas. Nele é possível verificar a utilização de ambientes de simulação desenvolvidos pelos autores para implementação e validação da proposta de redução da tabela.

Quadro 5 – Abordagem usada

Técnica de agregação	**Ambiente e Linguagem de Programação**
Spanning Tree (Castaneda *et al.*, 2020), (Gawrychowski *et al.*, 2021), (Gopalan; Ramasubramanian, 2016)	Simulação, C++
PATRICIA (Hammer; Hellwagner, 2022), (Saxena; Raychoudhury, 2016), (Chatterjee *et al.*, 2021)	Python, C++ e JavaTM
FIFA (Liu *et al.*, 2013), (Mottaghi *et al.*, 2022)	Blocos de Comutação, VTR 8
AIDR (Wang *et al.*, 2012), (Wang *et al.*, 2011a)	RouteViews, RIPE

Fonte: Elaborado pela autora, (2023).

– ***Avaliação das abordagens e viabilidade de implementação:***

São apresentadas a seguir a descrição dos resultados da extração da Revisão Sistemática da Literatura organizados segundo a abordagem utilizadas pelos autores em seus estudos. Esse tópico tem o objetivo de analisar a viabilidade de implementação em ambientes de simulação usando programação em geral.

– **Abordagens usadas no *Spanning Tree:***

No trabalho de Castaneda *et al.* (2020) é proposta uma estratégia semelhante à utilizada em outros trabalhos como o de Elkin e Neiman (2019, 2018). A implementação ocorre com base no algoritmo de Peleg e Upfal (1989), apresentado no Capítulo 3. Castaneda *et al.* (2020) não especifica em que ambiente foram construídas as simulações, no entanto de acordo com Peleg e Upfal (1989), trabalho usado como base para a proposta de Castaneda *et al.* (2020), e os pseudocódigos fornecidos ao longo do texto nos relevam que foi desenvolvido um ambiente de simulação.

Gawrychowski *et al.* (2021) apresentam uma análise teórica detalhada da estratégia proposta e mostram que ela é mais eficiente do que os esquemas existentes em termos de tamanho do rótulo e tempo de consulta. Os autores usaram o framework baseado nos estudos de Dahlgaard *et al.* (2015). Neles são descritos os algoritmos, o que nos sugere possibilidades de aplicabilidade usando linguagem C++, apesar dos autores não terem aplicado nesse contexto.

Gopalan e Ramasubramanian (2016) avaliam o desempenho dos esquemas propostos usando ambiente de simulação desenvolvido pelos próprios autores usando linguagem C++. Neste trabalho, os autores desenvolveram um algoritmo que constrói três árvores independentes de borda que podem ser usadas simultaneamente para realizar o roteamento. A estratégia usada pelos autores reduz o número de entradas na tabela de roteamento, e se mostrou eficiente em comparação com outras estratégias, como a de Kini *et al.* (2009).

– **Abordagens usadas no *PATRICIA:***

Hammer e Hellwagner (2022) apresentam uma estratégia chamada de *TinyTricia*, nela o foco principal é o consumo mínimo de memória. Adicionalmente os autores projetaram o algoritmo para obter alto desempenho de pesquisa. Para implementação, foi utilizado Python.

Na abordagem usada por Chatterjee *et al.* (2021) os autores usaram um ambiente com os seguintes parâmetros: Memoria RAM de 8 GB; Processador: Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz; Sistema operacional: Ubuntu e linguagens de programação: Python e C. Também foi utilizado o *software NetworkX* para criar grafos grandes e do mundo real.

Saxena e Raychoudhury (2016) implementam a sua proposta de redução usando a linguagem de programação Java. O programa foi executado em um servidor com processador Intel(R) Xenon(R) CPU E5-2695 v2 a 2,40 GHz e 128 GB de RAM DDR3. O ambiente operacional usado foi o Windows 8.1 de 64 bits.

– **Abordagens usadas no *FIFA:***

Mottaghi *et al.* (2022) implementaram a sua proposta usando o conjunto de ferramentas de código aberto VTR 8 que permite definir e avaliar blocos de comutação personalizados.

Em Liu *et al.* (2013) os autores propõem três algoritmos: FIFA-S para menor tamanho de tabela, FIFA-T para menor tempo de execução e FIFA-H para ambas, menor tabela e menor tempo de execução. Para avaliar o desempenho da proposta, eles realizaram uma comparação com os métodos *BasicOptSize*, *BasicMinTime* e SMALTA, que são outros esquemas de agregação de FIB baseados em *Optimal Routing Table Constructor* (ORTC). Os resultados dos autores mostram que uma redução da sobrecarga de computação dos roteadores e demonstram

que todos os algoritmos FIFA são mais eficientes em termos de velocidade e redução da tabela FIB.

Os autores utilizaram quatro métricas de desempenho: (1) Tamanho da FIB: número total de entradas na FIB; (2) Tempo de Execução: tempo necessário para aplicar as alterações de roteamento na FIB, incluindo o tempo de reagregação, se houver; (3) Mudanças na FIB: número total de atualizações na FIB causadas por todas as atualizações de roteamento; e (4) Burst na FIB: número de alterações na FIB causadas por uma mudança de rota. A avaliação foi realizada em uma máquina com um processador Intel Core 2 Quad de 2.83GHz (Liu *et al.*, 2013).

– **Abordagens usadas no *AIDR:***

Em Wang *et al.* (2011a) os autores propõem inovação baseada no BGP, chamada de AIDR. Ela aproveita os caminhos redundantes para o mesmo destino na Internet e leva em consideração a agregação de rotas na seleção para obter mais agregação para a tabela de encaminhamento (FIB). s resultados mostram que o AIDR pode produzir FIBs agregadas de tamanho aproximadamente 20~36% do tamanho original da tabela. Wang *et al.* (2012) realizaram experimentos usando as tabelas de roteamento BGP coletadas dos coletores do RouteViews (project., 2023) e do RIPE. Essas tabelas continham informações sobre os prefixos e os caminhos dos sistemas autônomos (AS) associados a eles. Em seu outro trabalho, os autores Wang *et al.* (2012) realizam uma avaliação da proposta em comparação com outras estratégias.

A seguir são apresentados os agrupamentos de Técnicas de Sumarização. De acordo com a Revisão foi possível localizar duas técnicas: CIRD e o FRT.

Quadro 6 – Agrupamento de Técnicas de Sumarização

Sumarização	**Autores**
CIDR - *Classless Inter-Domain Routing*	(Malgosa *et al.*, 2017), (Yu; Pao, 2019), (Du *et al.*, 2011)
FRT - *Flexible Routing Tables*	(Nagao; Shudo, 2011), (Hojo *et al.*, 2016)

Fonte: Elaborado pela autora, (2023).

Detalhamos as técnicas no quadro abaixo, e em seguida apresentamos as abordagens utilizadas pelos autores para validar suas propostas.

– **Abordagens usadas no *CIDR:***

No trabalho de Malgosa *et al.* (2017) é apresentado o Maskless Inter-Domain Routing (MIDR), um novo esquema de roteamento que visa solucionar o problema de esgotamento de endereços IP no contexto do protocolo IP e comparado com o CIDR (Classless Inter-Domain

Quadro 7 – Abordagem usada

Sumarização	Abordagem
CIDR - *Classless Inter Domain Routing* (Malgosa *et al.*, 2017), (Du *et al.*, 2011)	MATLAB
FRT- *Flexible Routing Tables* (Nagao; Shudo, 2011), (Hojo *et al.*, 2016)	Overlay Weaver

Fonte: Elaborado pela autora, (2023).

Routing). Segundo os autores, a partir do estudo foi notado que, enquanto o esquema de roteamento CIDR usa uma máscara para definir a divisão entre o ID do host e a parte de rede, o MIDR vai além, permitindo a definição de redes de qualquer tamanho, sem a necessidade de máscaras limitadas por potências de dois.

Du *et al.* (2011) aborda o problema de busca eficiente em listas CIDR em roteadores de alto desempenho. Inicialmente, é feita uma análise aprofundada dos algoritmos de árvore balanceada B-Tree com base no comprimento do endereço de prefixo, revelando que eles não são os melhores para esse propósito. Em seguida, é proposto um novo algoritmo de árvore B+-Tree com índice particionado, otimizando a estrutura de armazenamento e a busca de endereços IP na lista CIDR.

Utilizando o MATLAB, Du *et al.* (2011) realizaram experimentos de simulação para três algoritmos. O primeiro algoritmo foi proposto na literatura por Lim *et al.* (2003) e o segundo por Hui-jun (2009). O terceiro algoritmo foi o novo algoritmo proposto pelos autores, citado anteriormente.

Os experimentos realizados pelos autores de Du *et al.* (2011) demonstraram que o novo algoritmo alcança o objetivo de busca rápida de endereços IP na lista CIDR, atendendo às demandas dos roteadores de alto desempenho em redes de backbone.

– **Abordagens usadas no *FRT:***

Nagao e Shudo (2011) apresentam como proposta o FRT-Chord, um DHT *(Distributed Hash Table)* baseado em FRT, e demonstra sua eficiência na redução da tabela de roteamento. O FRT-Chord proposto pelos autores é implementado e experimentado utilizando o Overlay Weaver Shudo *et al.* (2008). Os experimentos mostram dos autores mostram que a tabela de roteamento é refinada de forma eficiente, alcançando um desempenho de pesquisa de $O(N)$ saltos em uma rede de N nós.

Hojo *et al.* (2016) propõem uma abordagem onde cada entrada da tabela de roteamento contém um nó diferente dos demais. Isso significa que um nó nunca ocupa mais de uma entrada na tabela de roteamento. Essa abordagem resulta em uma tabela de roteamento mais compacta e eficiente, eliminando a redundância de entradas para o mesmo nó. Os autores

realizaram implementação usando o Overlay Weaver[1] Shudo *et al.* (2008) .

[1] http://overlayweaver.sourceforge.net.

6 ESCOLHA DO MÉTODO DE ANÁLISE DAS TÉCNICAS DE REDUÇÃO DE TABELAS DE ROTEAMENTO

Para determinação da qualidade dos trabalhos encontrados e a definição dos métodos de redução de tabelas a serem implementados na dissertação, foram consideradas as seguintes características, em ordem de relevância:

1. Possui Código Fonte disponível;

2. Tipo de Ambiente;

3. Publicado em uma revista ou periódico de alto impacto.

Foi levada em consideração para a seleção das técnicas a serem comparadas a disponibilidade de código fonte para que houvesse a possibilidade de implementação do método proposto pelos autores de maneira mais fiel possível (Tabela 4). Além disso, consideramos o Tipo de Ambiente em que foi testada a proposta, analisando se foi proposta em Linguagem de Programação ou em uma ambiente de Simulação (Tabela 5). Por fim, também foi levado em consideração o periódico em que o artigo foi publicado, considerando o impacto segundo a (Tabela 6).

A Tabela 4 apresenta as informações referentes às bases de dados utilizadas, os autores dos estudos, os periódicos em que os artigos foram publicados e a indicação sobre a disponibilidade ou não do código-fonte da técnica de redução proposta pelos autores. A partir dessa seleção, foi possível filtrar as técnicas que se encaixem na proposta e no objetivo do estudo, que consiste em avaliar o comportamento de técnicas de redução de tabelas de roteamento, utilizando um ambiente de programação para modelar esse conjunto de estatégias e analisar qual estratégia apresenta um melhor desempenho.

Tabela 4 – Informações de base, periódico e código-fonte dos artigos selecionados na revisão sistemática

Base	**Artigo**	**Periódico Publicado**	**Código Fonte**
ACM	Castaneda *et al.* (2020)	*Association for Computing Machinery*	Não
	Chatterjee *et al.* (2021)	*Autom. Control Comput. Sci*	Sim
	Gawrychowski *et al.* (2021)	*Society for Industrial and Applied Mathematics*	Análise Teórica
	Gopalan e Ramasubramanian (2016)	*IEEE/ACM Trans. Netw*	Não

Tabela 4 – *Continuação da tabela*

Base	**Artigo**	**Periódico Publicado**	**Código Fonte**
	Duquennoy *et al.* (2013)	*Association for Computing Machinery*	Não
	Hammer e Hellwagner (2022)	*International Conference on Utility and Cloud Computing*	Sim
	Mottaghi *et al.* (2022)	*IEEE Transactions on Computers*	Não
	Yu e Pao (2019)	*Microprocessors and Microsystems*	Não
	Sultana e Asaduzzaman (2018)	*Joint 7th International Conference on Informatics, Electronics Vision (ICIEV)*	Não
	Malgosa *et al.* (2017)	*Sixth International Conference on Future Generation Communication Technologies*	Não
IEEE	Kawaguchi *et al.* (2016)	*IEEE Computer Society*	Não
	Liu *et al.* (2013)	*Proceedings IEEE INFOCOM*	Não
	Wang *et al.* (2012)	*International Conference on Computer Communications and Networks (ICCCN)*	Não
	Du *et al.* (2011)	*International Conference of Information Technology, Computer Engineering and Management Sciences*	Não
	Wang *et al.* (2011a)	*19th IEEE International Conference on Network Protocols*	Não
	Nagao e Shudo (2011)	*IEEE International Conference on Peer-to-Peer Computing*	Não
Science Direct	Li *et al.* (2011)	*Computer Communications*	Não
	Saxena e Raychoudhury (2016)	*Journal of Network and Computer Applications*	Sim

Fonte: Elaborado pela autora, (2024).

Um outro fator relevante para o estudo foi o Tipo de Ambiente utilizado nos estudos e os Métodos Desenvolvidos. Nessa etapa, foram avaliadas as estratégias que cada artigo aplicou para desenvolver e testar suas propostas. A análise também se preocupou em identificar o uso de simulações, linguagens de programação específicas e ambientes teóricos, considerando como cada abordagem influenciou a aplicabilidade e a eficiência das propostas. Na Tabela 5 é

possível identificar as informações após a análise e assim obter uma visão mais objetiva sobre as abordagens e seus contextos.

Tabela 5: Informações do método e tipo de ambiente utilizado nos artigos selecionados

Artigo	Método Desenvolvido	Tipo de ambiente
Castaneda *et al.* (2020)	*Spanning Trees*	Simulação
Chatterjee *et al.* (2021)	*Connected Dominating Set* (CDS) e Patricia Trie	Python, C
Gawrychowski *et al.* (2021)	*Spanning Trees*	Análise Teórica
Gopalan e Ramasubramanian (2016)	*Spanning Trees*	Simulação
Duquennoy *et al.* (2013)	ORPL	Simulação
Hammer e Hellwagner (2022)	*Patricia Trie*	Python
Mottaghi *et al.* (2022)	FPGAs COTS	Simulação
Yu e Pao (2019)	FPGAs	Simulação
Sultana e Asaduzzaman (2018)	*Minimum Spanning Tree* (MST)	Simulador Personalizado
Malgosa *et al.* (2017)	MIDR	Simulação
Kawaguchi *et al.* (2016)	Skip Graph	Simulação
Liu *et al.* (2013)	FIFA-S, FIFA-T, FIFA-H	Simulação
Wang *et al.* (2012)	AIDR	Simulação
Du *et al.* (2011)	B-Tree particionado	Simulação
Wang *et al.* (2011a)	AIDR	Simulação
Nagao e Shudo (2011)	FRT	Python
Li *et al.* (2011)	Topologias reais	Simulação
Saxena e Raychoudhury (2016)	*Patricia Trie*	Java

Fonte: Elaborado pela autora, (2024).

A Tabela 6 expressa os resultados da consulta realizada no *Journal Citation Reports* sobre o fator de impacto dos periódicos nos quais os artigos selecionados foram publicados. O fator de impacto é um indicador utilizado para determinar a relevância de um periódico acadêmico, baseando-se na frequência com que seus artigos são citados em outras pesquisas. Dessa forma, ao selecionar artigos de periódicos com maior fator de impacto, busca-se garantir que as técnicas comparadas neste estudo estejam alinhadas com as pesquisas desenvolvidas na área de redes e roteamento, aumentando a credibilidade e relevância dos resultados obtidos.

Tabela 6: Periódicos de maior impacto na área

Periódico	**Fator de Impacto**	**Quantidade de artigos**
IEEE Transactions on Computers	3,5892	2
Microprocessors and Microsystems	1,8657	1
Computer Communications	4,4652	1
Journal of Network and Computer Applications	4,6636	1
Society for Industrial and Applied Mathematics	0,325	1
Automatic Control and Computer Sciences	6,429	1

Fonte: Elaborado pela autora, (2024).

As técnicas selecionadas pela análise passaram pelos critérios, **1.** Publicado em uma revista ou periódico de alto impacto; **2.** Possui Código Fonte disponível e **3.** Publicado em

uma revista ou periódico de alto impacto. Segundo os dados coletados, apenas as propostas de Chatterjee *et al.* (2021), Hammer e Hellwagner (2022) e Saxena e Raychoudhury (2016) apresentaram o código fonte e também foram publicados em periódicos de alto impacto, sendo que o trabalho de Hammer e Hellwagner (2022) foi publicado em uma conferência internacional, cujo o fator de impacto não foi localizado.

Dessa forma, as técnicas de redução de tabelas de roteamento propostas na literatura viáveis de implementação em um ambiente de programação, que foram publicadas em um periódico de alto impacto e que se adequam aos critérios estabelecidos nesta dissertação foram os de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021).

7 RESULTADOS

A seguir, descrevemos a problemática em que o presente trabalho se concentra, com objetivo de avaliar o comportamento de técnicas de redução de tabelas de roteamento. Para isso, primeiramente, relembramos nossos objetivos, que consistem em: Analisar a capacidade de implementação dessas técnicas de redução de tabelas de roteamento selecionadas em um ambiente de simulação e Comparar as técnicas de redução de tabelas de roteamento em termos tempo de reagregação dos prefixos, uso de memória e redução da tabela. Para isso, utilizamos linguagem de programação para modelar esse conjunto de estatégias e analisar para qual técnica a redução da tabela se comporta de maneira mais eficiente.

Dessa forma, este estudo considera as técnicas empregadas em NDN, abordadas na Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021), como inspiração para novas abordagens no contexto de roteamento IP. As técnicas de estruturação e redução de tabelas de roteamento, tais como a *Patricia Trie* e o *Connected Dominating Set* (CDS), foram adaptadas e implementadas para avaliar o seu potencial de otimização em redes baseadas em IP.

7.1 Abordagem em Named Data Networking (NDN) usando Patricia Trie/N-FIB: Saxena e Raychoudhury (2016)

Saxena e Raychoudhury (2016) propõem o N-FIB, um esquema escalável e com eficiência de memória que suporta agregação FIB sem a necessidade de alteração dos protocolos de roteamento. Nessa estratégia é utilizada a árvore Patricia para armazenar nomes de roteamento e encaminhar pacotes de interesse para as interfaces de saída. O uso da árvore “reduz o consumo de memória, o tempo de busca e pesquisa, enquanto mantém baixa complexidade computacional” (Saxena; Raychoudhury, 2016) .

Na estratégia do N-FIB proposta por Saxena e Raychoudhury (2016), uma atualização na rota não implica em múltiplas modificações na FIB agregada, o que significa uma otimização do processo de atualização. O N-FIB também assegura uma forte correção de encaminhamento, evitando loops de pacotes, problema comumente associado a esquemas de correção de encaminhamento mais fracos. Com isso, as operações de estrutura de dados e de busca são simplificadas, e o N-FIB mantém os mesmos comportamentos de encaminhamento antes e depois da agregação (Saxena; Raychoudhury, 2016).

7.1.1 Descoberta das Rotas

A descoberta de rotas no N-FIB ocorre utilizando uma estrutura de dados baseada na árvore Patricia, que armazena os prefixos de nomes de conteúdo de forma hierárquica e compacta. Quando um roteador recebe um Interest Packet, ele realiza a busca pelo prefixo mais longo (Longest Prefix Match - LPM) na tabela de roteamento (Forwarding Information Base - FIB) para identificar o próximo salto adequado. Esse processo de busca permite encaminhar os pacotes de maneira eficiente para a interface de saída correta.

No N-FIB, a árvore Patricia reduz o consumo de memória e o tempo de busca, pois elimina redundâncias ao consolidar prefixos semelhantes, tornando o processo de descoberta mais ágil. Além disso, a correção de encaminhamento é mantida, o que evita problemas de looping de pacotes e garante que as rotas corretas sejam seguidas.

As principais contribuições da estratégia do N-FIB são: “otimização do consumo de memória e do tempo de busca (uso da árvore Patricia), suporte a agregação de FIB, o que reduz o número de alterações necessárias durante as atualizações de rota, e manutenção de uma forte correção de encaminhamento, essencial para evitar loops e inconsistências” (SAXENA e RAYCHOUDHURY, 2016).

7.1.2 Manutenção de Rotas

A manutenção de rotas no N-FIB é realizada de forma eficiente com atualizações incrementais. O N-FIB permite que novos prefixos de rota sejam inseridos na árvore Patricia sem a necessidade de recalcular toda a estrutura do FIB, o que minimiza a sobrecarga de processamento. Quando uma nova atualização de rota é recebida, o N-FIB insere o prefixo diretamente no FIB, ajustando apenas os nós necessários, o que otimiza o desempenho e reduz o tempo de atualização.

A estratégia de agregação do N-FIB é essencial para a manutenção eficiente das rotas. Prefixos semelhantes são combinados sempre que possível, reduzindo a quantidade de entradas e o consumo de memória na FIB. Por exemplo, prefixos que compartilham uma parte inicial comum podem ser agregados, evitando a duplicação de informações. Com isso, o N-FIB consegue garantir que as atualizações ocorram de forma contínua e com baixo impacto no desempenho de roteamento.

As principais contribuições da manutenção de rotas no N-FIB são: “agregação de

prefixos para reduzir o consumo de memória, suporte a atualizações incrementais no FIB para evitar recalcular a tabela inteira, e uma abordagem eficiente de inserção e remoção de prefixos que preserva a integridade da estrutura da árvore Patricia" (Saxena; Raychoudhury, 2016)

7.1.3 N-FIB: Algoritmo disponibilizado por Saxena e Raychoudhury (2016)

A Tabela 7 é apresentada as descrições dos diferentes símbolos utilizados no algoritmo descrito por Saxena e Raychoudhury (2016).

Tabela 7: Descrição dos diferentes símbolos usados no algoritmo apresentado por Saxena e Raychoudhury (2016).

Símbolos	**Descrição**
k	CN a ser inserida na FIB
v	Próximos saltos (next-hop(s)) associados à chave de entrada k
$\text{node}^{p}_{\text{get}}$	Retorna o prefixo p no nó da trie
C^{p}_{len}	Comprimento do prefixo comum entre k e node_{get}
$\text{node}^{\text{has\_v}}$	Retorna verdadeiro se o nó contém o valor de próximo salto(s)
$\text{node}^{v}_{\text{get}}$	Retorna o valor v associado ao nó da trie
$\text{node}^{v}_{\text{set}}$	Define o v para o nó durante a inserção do nome
$\text{Left}^{k}_{\text{over}}$	Parte restante da chave k (excluindo o prefixo comum) a ser inserida na trie
$\text{cnode}^{p}_{\text{get}}$	Retorna o prefixo p associado ao filho (que tem o prefixo comum com o restante de k) de um nó
$\text{node}^{children}_{\text{get}}$	Retorna o endereço dos filhos de um nó
$\text{Left}^{p}_{\text{over}}$	Parte restante de p do nome restante a ser inserido na trie
$\text{node}^{p}_{\text{set}}$	Define o prefixo p para o nó
$\text{node}^{has\_v}_{\text{set}}$	Ativa o sinalizador para ter o valor de próximo salto(s)

Fonte: Extraido de Saxena e Raychoudhury (2016)

7.2 Abordagem em Named Data Networking (NDN) usando Connected Dominating Set (CDS) e Patricia Trie: Chatterjee *et al.* (2021)

Chatterjee *et al.* (2021) apresenta uma proposta de abordagem de otimização para a arquitetura Named Data Networking (NDN), visando a eficiência no armazenamento e na busca de dados através do uso de *Connected Dominating Set* (CDS) e da árvore Patricia. A técnica de CDS permite que apenas alguns nós específicos mantenham uma cópia completa da tabela de roteamento, enquanto os nós comuns mantêm apenas informações de roteamento

Algoritmo 1: Algoritmo disponibilizado por Saxena e Raychoudhury (2016)

Input: Chave k a ser inserida com próximo salto(s) valor v

begin
$C^{p}_{\text{len}} = \text{commonPrefix}(k, \text{node}^{p}_{\text{get}})$
if $C^{p}_{\text{len}} == \text{node}^{p}_{\text{get}}.\text{length}()$ **&&** $C^{p}_{\text{len}} == k.\text{length}()$ **then**
  Found a node with an exact match
  if $\text{node}^{v}_{\text{get}} == \text{null}$ || $!\text{node}^{has\_v}$ **then**
    $\text{node}^{v}_{\text{get}} = v$
    $\text{node}^{has\_v}_{\text{set}}$ = true
  end if
  if $\text{cnode}^{p}_{\text{get}}$ é um componente possível de qualquer k **then**
    Mesclar o(s) valor(es) dos filhos no nó e remover filhos para o prefixo do nome
  end if
end if
if $C^{p}_{\text{len}} < k.\text{length}()$ **&&** $C^{p}_{\text{len}} >= \text{node}^{p}_{\text{get}}.\text{length}()$ **then**
  for each (node: child) **do**
    if $\text{cnode}^{p}_{\text{get}}.\text{charAt}(0) == \text{Left}^{k}_{\text{over}}.\text{charAt}(0)$ **then**
      if $\text{cnode}^{v}_{\text{get}} >= 0$ || cnode^{has_v} **then**
        Mesclar o(s) valor(es) do prefixo do nome com o prefixo do nome em
        return FIB_aggregation(cnode, $\text{Left}^{k}_{\text{over}}, v$)
        break
      end if
    end if
  end for
  $\text{node}^{children}_{\text{get}}.\text{add}(\text{Left}^{k}_{\text{over}}, v)$ {Se nenhum filho existe com qualquer prefixo da chave dada, adicione um novo}
else
  if $C^{p}_{\text{len}} < \text{node}^{p}_{\text{get}}.\text{length}()$ **then**
    Dividir nó
    if $C^{p}_{\text{len}} == k.\text{length}()$ **then**
      $\text{node}^{\text{value}}_{\text{set}} = v$
      $\text{node}^{\text{has\_v}}_{\text{set}} = \text{true}$
    else
      Criar nó n' com $n'^{p}_{\text{set}} = \text{Left}^{k}_{\text{over}}$ e $n'^{v}_{\text{get}} = v$
      $\text{node}^{children}_{\text{get}}.\text{add}(n')$
    end if
  end if
end if

parciais, reduzindo a sobrecarga de memória. A árvore Patricia, por sua vez, organiza os dados hierarquicamente, facilitando a busca rápida por prefixos de nome, otimizando o desempenho e o tempo de resposta para recuperação de dados.

Saxena e Raychoudhury (2016) escrevem que NDN representa uma arquitetura inovadora para a Internet, alterando a lógica de comunicação de um modelo centrado em

hosts para um modelo centrado no conteúdo. Isso significa que, ao invés de buscar conteúdo diretamente de hosts específicos, o NDN permite que o conteúdo seja recuperado de qualquer nó intermediário que possua o dado válido. Dessa forma, o NDN promove um paradigma centrado em conteúdo, em oposição ao paradigma centrado em hosts utilizado na Internet atual (Hu *et al.* (2011), Saxena *et al.* (2016) e Wang *et al.* (2011b).)

A estratégia de Chatterjee *et al.* (2021) propõe o uso da árvore Patricia junto ao CDS. Isso permite que apenas alguns nós estratégicos armazenem uma cópia completa da tabela de roteamento. Esses nós mantêm uma visão global da topologia, enquanto os demais armazenam apenas dados dos vizinhos, reduzindo assim a sobrecarga de memória.

7.2.1 Descoberta das Rotas

A descoberta de rotas ocorre através do CDS, onde cada nó "especial" possui acesso a uma base de dados completa, chamada FDB (Full Data Base), enquanto os nós "comuns" possuem uma base parcial de dados. Segundo a proposta de Chatterjee *et al.* (2021), ao receber um pacote de requisição de dados (*Interest Packet*), o nó verifica primeiramente sua base local. Caso a informação não seja encontrada, a solicitação é encaminhada ao nó "especial" mais próximo, onde a árvore Patricia realiza a busca no FDB.

As principais contribuições da estratégia proposta por Chatterjee *et al.* (2021) são, segundo os autores: "reduzir a sobrecarga de armazenamento usando o CDS e a árvore Patricia, reduzir o tempo de busca (árvore Patricia), desenvolvimento de um esquema de encaminhamento que mantêm o banco de dados em alguns dos nós estrategicamente localizados dentro da rede, em vez de em todos os nós."

7.2.2 Manutenção de Rotas

A manutenção de rotas na estratégia de Chatterjee *et al.* (2021) foca na atualização eficiente do roteamento. Para garantir que as atualizações de roteamento sejam propagadas sem sobrecarregar a rede, os autores propõem um modelo de disseminação utilizando o BFS (*Breadth First Search*) aplicado no CDS. Esse método assegura que as atualizações sejam enviadas diretamente para os nós "especiais" e seus vizinhos mais próximos, sem a necessidade de inundação, prática comum em métodos convencionais e que aumenta a sobrecarga de comunicação (Chatterjee *et al.*, 2021).

7.3 Análise e comparação das estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016)

As nossas questões de pesquisa se concentravam inicialmente localizar quais técnicas de redução de tabelas de roteamento estão propostas na literatura. De acordo com a Tabela 4 foi possivel localizar um conjunto com 18 técnicas que se concentram em reduzir o tamanho relação da tabela de roteamento. As propostas usam métodos como *Spanning Trees*, *Connected Dominating Set*, MIDR, AIDR e Patricia Trie e são desenvolvidos em sua maioria em ambientes de simulação e em linguagem de programação.

A RSL capturou pesquisas envolvendo a redução de tabelas FIB na arquitetura de rede NDN e técnicas que se concentram no roteamento convencional IP. As propostas foram então, categorizadas segundo a sua abordagem e se elas se concentram em roteamento IP ou em NDN, como mostrado na Quadro 8.

Quadro 8: Resumo das propostas para redução de tabelas FIB, categorizadas por abordagem e arquitetura de rede

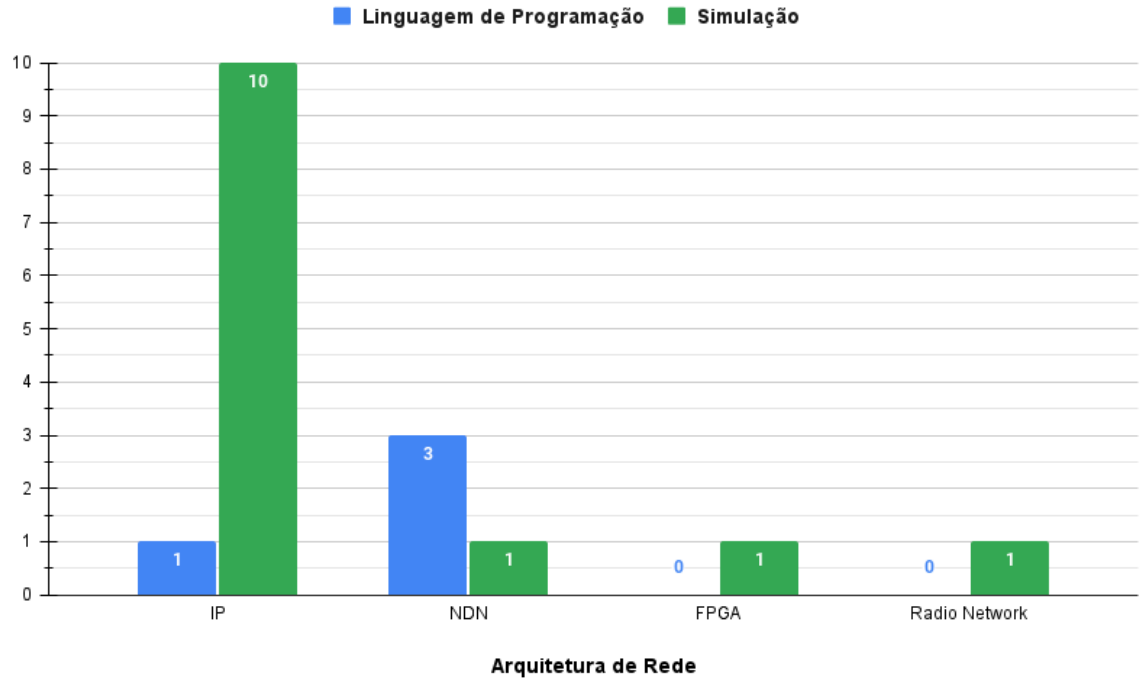
Proposta	**Abordagem**	**Arquitetura de Rede**
Castaneda *et al.* (2020)	CompactFTZ	IP
Chatterjee *et al.* (2021)	Connected Dominating Set (CDS) / Patricia Trie	NDN
Gawrychowski *et al.* (2021)	Análise Teórica	IP
Gopalan e Ramasubramanian (2016)	*IP Fast Rerouting com Three Edge-Independent Spanning Trees*	IP
Duquennoy *et al.* (2013)	Trie Compactada (PATRICIA)	IP
Hammer e Hellwagner (2022)	TinyTricia	IP
Mottaghi *et al.* (2022)	FIFA/BTI-Induced Aging Effects	FPGA
Yu e Pao (2019)	Hardware Accelerator for FIB Lookup	NDN
Sultana e Asaduzzaman (2018)	*Spanning Tree*	Radio Network
Malgosa *et al.* (2017)	Maskless Inter-Domain (MIDR)	IP
Kawaguchi *et al.* (2016)	*Self-Refining Skip Graph*	NDN
Liu *et al.* (2013)	FIB Aggregation	IP
Wang *et al.* (2012)	AIDR	IP
Du *et al.* (2011)	Algoritmo AB-Tree	IP
Wang *et al.* (2011a)	AIDR	IP
Nagao e Shudo (2011)	Self-Refining Skip Graph	NDN
Li *et al.* (2011)	Nexthop-Selectable FIB Aggregation	IP
Saxena e Raychoudhury (2016)	N-FIB	NDN

Fonte: Elaborado pela autora, (2023).

Para uma análise mais precisa, foi desenvolvido um gráfico (ver Figura 6) que evidencia a quantidade de estudos que apresentam as estratégias em ambientes de simulação redes de roteamento IP, enquanto há poucas estratégias focadas no uso de linguagens de programação. Observa-se que, na arquitetura de rede IP, 10 estudos utilizaram simulações, enquanto apenas 1

fez uso de linguagem de programação. Para a arquitetura NDN, foram identificados 3 estudos que empregaram linguagens de programação, em contraste com apenas 1 baseado em simulação. As arquiteturas FPGA e Radio Network apresentam 1 estudo cada, ambos utilizando simulação, e nenhum estudo registrado com uso de linguagem de programação.

Figura 6: Comparação entre Arquitetura e Tipo de Ambiente



Fonte: Elaborado pela autora, (2024).

Relembramos a Tabela 4 que apresenta as informações referentes às bases de dados utilizadas, os autores dos estudos, os periódicos em que os artigos foram publicados e a indicação sobre a disponibilidade do código-fonte da técnica pelos autores e ressaltamos os critérios de seleção do estudo, que de acordo a relevância para a seleção da técnica se configura um fato determinante na escolha das estratégias.

Para tanto, de acordo com os dados e análises coletadas, os estudos que se alinhavam a proposta desta dissertação e que atendem aos critérios estabelecidos foram as estratégias: (Chatterjee *et al.*, 2021) e (Saxena; Raychoudhury, 2016). Ambas apresentando propostas para NDN, com disponibilidade do código fonte e uso de linguagem de programação para modular suas estratégias.

A Tabela 8 descreve as principais características entre as duas propostas e apresenta uma comparação entre os seguintes aspectos: Estrutura de Dados, Redução de Sobrecarga e atualização e Eficiência de Memória e Tempo de Busca.

Tabela 8: Comparação entre os Trabalhos de Chatterjee et al. e Saxena et al.

Aspectos	**Chatterjee *et al.* (2021)**	**Saxena e Raychoudhury (2016)**
Estrutura de Dados	Uso do Conjunto Dominante Conectado (CDS) para selecionar nós "especiais" para armazenarem uma cópia completa da tabela de encaminhamento por meio da FDB, enquanto os nós "normais" mantêm apenas dados locais.	Baseado em *Patricia Trie* com agregação de FIB, combinando múltiplas entradas.
Redução de Sobrecarga de Atualização	Atualizações ocorrem principalmente entre nós "especiais" e vizinhos próximos.	Atualizações inseridas na árvore Patricia sem a necessidade de recálculo de toda a estrutura, não afetando as outras entradas na FIB agregada.
Eficiência em Memória e Tempo de Busca	A *Patricia Trie* é usada para otimizar o tempo de busca em nós "especiais".	Segundo o artigo das autoras, alcança economia de até 96,94% de memória e reduz o número de nós, proporcionando buscas mais rápidas e eficiente uso de memória.

Fonte: Elaborado pela autora, (2023).

A estratégia de Chatterjee *et al.* (2021) e de Saxena e Raychoudhury (2016) envolvem o uso de Named Data Networking (NDN) para a construção de estruturas de dados com o objetivo de melhorar o desempenho na busca e armazenamento de nomes em redes de grande escala. Ambas as abordagens utilizam a árvore Patricia para a compressão de prefixos e economia de memória ao armazenar dados de roteamento (Hammer; Hellwagner, 2022).

No trabalho de Saxena e Raychoudhury (2016), a estratégia N-FIB (Next-hop Forwarding Information Base) é apresentada como uma técnica que utiliza a *Patricia Trie* para reduzir significativamente o consumo de memória em comparação com abordagens tradicionais.

Em Chatterjee *et al.* (2021) uma abordagem similar à do N-FIB é adotada, onde a Patricia Trie é usada para reduzir os custos de armazenamento, tempo de busca e consequentemente tempo de transmissão de dados. Enquanto que em Saxena e Raychoudhury (2016) o objetivo é o equilíbrio entre economia de espaço e eficiência de busca. Uma comparação mais detalhada foi realizada na Tabela 8.

7.4 Especificações do Experimento e propostas de Implementação

As subseções a seguir discorrem a respeito da análise, viabilidade e implementação das estratégias dos estudos selecionadas pelos critérios estabelecidos nesta dissertação. Dessa forma, apresentamos as configurações e as especificações do Dataset que será utilizado para a

análise e comparação das estratégias.

7.4.1 Configurações do Experimento e Conjunto de dados

Nessa seção, avaliaremos o comportamento de técnicas de redução de tabelas de roteamento (FIB) definidas pelas autoras Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016). Para isso, utilizamos um ambiente de programação em C para modelar esse conjunto de estratégias e analisar qual técnica apresenta um melhor desempenho quanto a redução de custo de armazenamento.

Neste experimento foram utilizados um conjunto de dados coletados no RouteViews[1], com as especificações indicadas na tabela. O RouteViews é um projeto da Universidade de Oregon (n.d.), ele funciona para o monitoramento de roteamento da Internet que coleta e disponibiliza tabelas de roteamento ((Saxena; Raychoudhury, 2016)).

Tabela 9: Informações do *Dataset*

Dataset	**Número de Entradas**	**Total de Componentes**
Dataset	6.054.768	27.447.502

Fonte: Elaborado pela autora, (2023).

Na tabela 9 são apresentadas as informações do conjunto de dados utilizado para os experimentos. O *Dataset* foi obtido através do projeto RouteViews, especificamente da base route-views.fortaleza, do diretório de janeiro de 2024.

O campo "Número de Entradas" indica a quantidade total de prefixos de rede presentes no *Dataset*, que representam diferentes rotas e destinos de rede. O "Total de Componentes" representa o número de Sistemas Autônomos envolvidos nos caminhos de roteamento para esses prefixos.

O conjunto de dados extraído do RouteViews apresenta informações relacionadas à tabela de roteamento da Internet, especificamente às rotas BGP registradas. Cada linha do arquivo representa uma entrada na tabela de roteamento, detalhado os seguintes campos:

- **TABLE:** Todas as linhas começam com `TABLE_DUMP2`, indicando que se trata de uma exportação de tabela de roteamento.
- **Timestamp:** O segundo campo (`1704067200`) representa o carimbo de data e hora em formato *Unix*, correspondente ao momento em que a entrada foi registrada.

[1] Conjunto de dados disponível em: https://www.routeviews.org/routeviews/index.php/archive/.

- **Tipo de Rota:** O campo seguinte (`B`) indica o tipo de rota.
- **Endereço IP do *Peer* BGP:** O endereço IP (`45.184.144.39`) corresponde ao peer BGP que anunciou a rota.
- **Sistema Autônomo (AS) do *Peer*:** O número AS (`263945`) associado ao *peer* que anunciou a rota.
- **Prefixo IP:** Representa a rede roteada (`1.0.178.0/24`), indicando o bloco de endereços IP anunciado.
- **Caminho de AS (AS_PATH):** Lista sequencial dos sistemas autônomos pelos quais o prefixo passa (`263945 6939 4651 23969`), começando pelo AS do *Peer* até o destino.
- **Tipo de Origem:** O tipo de origem (`IGP`) indica como a rota foi aprendida.
- **IP do Anunciador:** Repete o IP do *Peer* que anunciou a rota (`45.184.144.39`).
- **Medidas de Tempo e Métricas:** Valores adicionais relacionados a métricas internas ou políticas de roteamento (`0` e `1440`).
- **BGP:** O campo (`1031:300 1031:303`) contém comunidades BGP que codificam informações adicionais, como políticas de roteamento aplicadas ou preferências.
- **Status do Anúncio:** Indicado por `NAG`, que pode sugerir ausência de alteração ou estado específico da rota.

Foi realizado inicialmente um estudo com uma parte das entradas de maneira isolada, para uma melhor compreensão do comportado do conjunto de dados. Para isso, isolamos as entradas iniciais e selecionamos as rotas percorridas. Uma parte desses dados é apresentada a seguir:

O conjunto de dados contendo os registros é apresentado na Figura 8, destacando a estrutura e o seu formato. Cada linha no arquivo representa uma entrada da tabela de roteamento, com campos detalhados sobre os prefixos anunciados, os sistemas autônomos envolvidos e as métricas associadas.

As especificações do sistema utilizadas em nossos experimentos estão descritas na Tabela 10. O ambiente usado na experimentação tinha a seguinte configuração: 8 GB de RAM e um processador Intel Core i5, executando o sistema operacional Windows 11. O experimento foi desenvolvido e implementado em Python, escolhida pela sua eficiência no processamento de dados em larga escala (Weiwei, 2011).

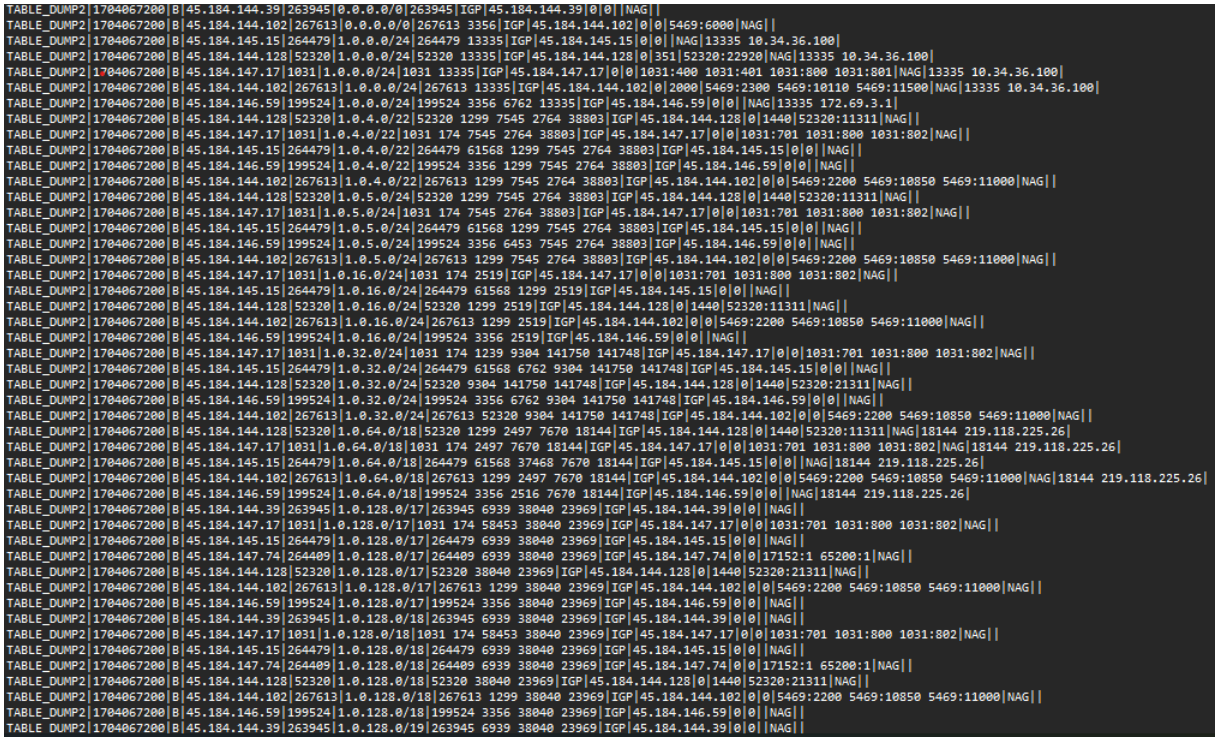
Incialmente foram realizados experimentos com limitações no tamanho da tabela, com o número de entrada variando entre os intervalos de 50.000 até 400.000.000. Essa variação

Figura 7: Análise de uma amostra das entradas iniciais

```
TABLE_DUMP2|1704067200|B|45.184.144.39|263945|1.0.178.0/24|263945 6939 4651 23969|IGP|45.184.144.39|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.147.17|1031|1.0.178.0/24|1031 4651 23969|IGP|45.184.147.17|0|0|1031:300 1031:303 1031:800
1031:801|NAG||
TABLE_DUMP2|1704067200|B|45.184.145.15|264479|1.0.178.0/24|264479 6939 4651 23969|IGP|45.184.145.15|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.147.74|264409|1.0.178.0/24|264409 6939 4651 23969|IGP|45.184.147.74|0|0|17152:1 65200:1|NAG||
TABLE_DUMP2|1704067200|B|45.184.144.128|52320|1.0.178.0/24|52320 38040 23969|IGP|45.184.144.128|0|1440|52320:21311|NAG||
TABLE_DUMP2|1704067200|B|45.184.144.102|267613|1.0.178.0/24|267613 3356 38040 23969|IGP|45.184.144.102|0|0|5469:2200 5469:10850
5469:11000|NAG||
TABLE_DUMP2|1704067200|B|45.184.146.59|199524|1.0.178.0/24|199524 3356 38040 23969|IGP|45.184.146.59|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.144.39|263945|1.0.179.0/24|263945 6939 4651 23969|IGP|45.184.144.39|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.147.17|1031|1.0.179.0/24|1031 4651 23969|IGP|45.184.147.17|0|0|1031:300 1031:303 1031:800
1031:801|NAG||
TABLE_DUMP2|1704067200|B|45.184.145.15|264479|1.0.179.0/24|264479 6939 4651 23969|IGP|45.184.145.15|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.147.74|264409|1.0.179.0/24|264409 6939 4651 23969|IGP|45.184.147.74|0|0|17152:1 65200:1|NAG||


TABLE_DUMP2|1704067200|B|45.184.144.128|52320|1.0.179.0/24|52320 38040 23969|IGP|45.184.144.128|0|1440|52320:21311|NAG||
TABLE_DUMP2|1704067200|B|45.184.144.102|267613|1.0.179.0/24|267613 3356 38040 23969|IGP|45.184.144.102|0|0|5469:2200 5469:10850
5469:11000|NAG||
TABLE_DUMP2|1704067200|B|45.184.146.59|199524|1.0.179.0/24|199524 3356 38040 23969|IGP|45.184.146.59|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.144.39|263945|1.0.182.0/24|263945 6939 4651 23969|IGP|45.184.144.39|0|0||NAG||
TABLE_DUMP2|1704067200|B|45.184.147.17|1031|1.0.182.0/24|1031 4651 23969|IGP|45.184.147.17|0|0|1031:300 1031:303 1031:800
1031:801|NAG||
```

Fonte: Elaborado pela autora, (2024).

Figura 8: Amostra do Arquivo extraído do RouteViews



Fonte: Elaborado pela autora, (2024).

foi escolhida por considerarmos que a partir de diferentes escalas é possível analisar de forma mais assertiva o comportamento das técnicas, desde tabelas menores até tabelas com número maior de entradas, observando como a implementação de cada estratégia se comportava.

Esses dados foram analisados de forma isolada, observando-se o comportamento da técnica à medida que o número de entradas aumentava. O resultados são discutidos na seção 7.9

Tabela 10: Especificações do Experimento e Configurações

Parâmetros	**Especificações**
RAM	8,00 GB
Processador	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Sistema Operacional	Windows 11
Linguagem de Programação	Python 3.10
Dataset	https://archive.routeviews.org/route-views.fortaleza/bgpdata

Fonte: Elaborado pela autora, (2024).

7.5 Viabilidade de Implementação da proposta descrita por Chatterjee *et al.* (2021)

Como mencionado na Tabela 8 a estratégia utilizada por Chatterjee *et al.* (2021) utiliza o CDS para selecionar os nós "especiais" e a *Patricia Trie* para otimizar o tempo de busca. As atualizações da tabela ocorrem primordialmente nos nós "especiais" e em seus vizinhos que possuem todas as informações da rede.

Pesquisas realizadas utilizando CSD em outros contextos de arquitetura foram realizadas, como é o caso de Yadav *et al.* (2009) onde é proposto um algoritmo modificado que pode calcular o conjunto dominante mínimo conectado para redes *ad hoc*. Outra proposta apresenta CDS sendo usado para fazer o backbone virtual, considerando a rede como o gráfico e os nós sensores como os vértices do gráfico (Praba *et al.*, 2019) e (Sun *et al.*, 2019).

Segundo Yadav *et al.* (2009), (CDS) pode ser usado para reduzir a redundância causada por transmissões. Em um grafo simples $G(N,E)$, N é o conjunto de nós e E é o conjunto de arestas. Suponha um conjunto de nós T, que é um subconjunto de N, tal que, para todo u em $N-T$, existe v em T, tal que a aresta (u,v) pertence a E. Essa é a *propriedade de cobertura* para o CDS. O conjunto T é chamado de *Conjunto Dominante*. O conjunto T também é chamado de *Conjunto Dominante Conectado* (Yadav *et al.*, 2009).

7.5.1 Proposta de adaptação ao Contexto de Roteamento IP

Em nossa análise também utilizamos CDS, assim como proposto por Chatterjee *et al.* (2021). Dessa forma, selecionamos inicialmente os nós que armazenam a maior parte das informações de roteamento e são encarregados de realizar as atualizações na rede. A tabela abaixo foi construida com a identificação dos nós especiais e suas respectivas quantidades de conexões.

Para seleção dos nós especiais podem ser considerados os seguintes aspectos: análise

da rede e análise do sistema (Li *et al.*, 2020), (Xue *et al.*, 2019). A análise da rede identifica os nós dominantes com base em dois critérios: Centralidade e Proximidade (Chang *et al.*, 2016).

Para a seleção dos nós dominantes/especiais foram considerados os aspectos definidos pelos autores Chang *et al.* (2016), que são selecionados com base nos seguintes critérios:

Centralidade - a centralidade de grau avalia o número de outros nós que estão diretamente conectados a um nó, refletindo a centralidade local do nó. Assim, os nós com maior centralidade de grau estão conectados a mais nós, possuem mais direitos formais e influência e, como consequência, podem exercer influência sobre toda a rede através dos nós conectados a eles, ganhando mais oportunidades de acesso a recursos (Chang *et al.*, 2016).

Proximidade - Esse aspecto mede a importância de um nó com base em sua "proximidade" com todos os outros nós da rede. Ele se baseia em calcular as menores distâncias entre um nó específico e todos os outros, dando uma ideia de quão rapidamente um nó pode alcançar ou ser alcançado por outros (Chang *et al.*, 2016).

A escolha dos endereços IP mais frequentes para compor os nós "especiais" no CDS baseia-se na centralidade e Proximidade conforme descrito por Chang *et al.* (2016). Esses IPs representam roteadores com maior número de conexões, sugerindo que são pontos estratégicos de comunicação.

Essa seleção foi feita em código Python que mapeou a rede e identificou a frequência de ocorrência dos IP dos roteadores no *Dataset*. O código completo foi descrito e apresentado no Apêndice F, nele são realizados o mapeamento, extração e ordenação decrescente do IP e suas respectivas frequências.

A Tabela 11 foi construída para identificar a frequência de anúncios de rotas pelos IPs de roteadores do *Dataset*, em ordem decrescente. O total somado das frequências de anúncios é 6.053.407, sendo o total de entradas na tabela de roteamento completa correspondente a 6.054.768, resultando em uma diferença de 1.361 anúncios que não estão cobertos pelos IPs listados. Isso ocorreu devido a anúncios feitos por roteadores menos frequentes. Uma tabela completa com esses anúncios foi acrescida nos apêndices.

O próximo passo após a implementação do uso dos endereços IP dos roteadores para identificar nós especiais é a estruturação da *Patricia Trie* como descrito na proposta de Chatterjee *et al.* (2021).

PATRICIA é um sistema de indexação projetado para armazenar e acessar dados em uma biblioteca binária de forma eficiente. Ele cria um índice que permite recuperar informações

Tabela 11: Frequência de ocorrência dos IPs de roteadores na tabela de roteamento

IP Roteador	**Frequência dos Anúncios**
45.184.145.15	972.764
45.184.144.102	940.863
45.184.146.59	926.034
45.184.144.128	924.337
45.184.147.17	910.269
2001:12f8:0:9::187	200.044
2001:12f8:0:9::39	190.839
2001:12f8:0:9::102	188.664
45.184.147.74	188.479
2001:12f8:0:9::128	183.237
2001:12f8:0:9::146:59	181.176
45.184.144.39	172.738
2001:12f8:0:9::147:17	58.024
45.184.145.162	10.811
2001:12f8:0:9::145:162	4.884
2001:12f8:0:9::145:122	187
45.184.145.122	57
Total	**6.053.407**

Fonte: Elaborado pela autora, (2023).

(ou “alvos”) a partir de chaves específicas e também facilita a atualização desse índice quando a biblioteca é modificada (Morrison, 1968b).

O sistema utiliza uma estrutura simplificada, onde o índice contém apenas números que representam localizações no texto ou índice, sem armazenar as próprias chaves ou textos. Esse método, segundo Chatterjee *et al.* (2021), reduz significativamente o uso de memória e a quantidade de processamento.

A implementação da *Patricia Trie* inicia com a criação de uma estrutura de dados que representa cada nó da Trie como um ponto de decisão na busca binária. Nesse processo, os bits dos endereços IP são analisados de forma sequencial para determinar o caminho adequado. Cada nó contém referências para os nós “filhos” e um indicador de nó terminal, que marca o fim de um prefixo IP específico.

Para inserir os prefixos IP na estrutura, percorrem-se os bits de cada endereço IP e segue-se o caminho correspondente na Trie. Caso o caminho não exista, são criados novos nós até que o endereço completo esteja armazenado.A busca de um endereço IP na *Patricia Trie*

segue uma lógica similar à da inserção, analisando-se cada bit do endereço e navegando-se pela estrutura até que um nó terminal correspondente seja encontrado. Essa abordagem permite uma recuperação rápida e eficiente dos dados associados aos nós especiais, conforme a proposta de Chatterjee *et al.* (2021).

Na aplicação ao contexto de roteamento IP, a *Patricia Trie* é configurada para tratar os endereços IP como chaves binárias, dividindo o prefixo conforme necessário para atender à estrutura hierárquica dos IPs. Dessa forma, a Trie oferece suporte eficiente tanto para os nós especiais (endereços IP de roteadores) quanto para consultas rápidas aos prefixos. Essa implementação da *Patricia Trie*, portanto, facilita a administração dos nós especiais e contribui para uma resposta rápida a consultas de roteamento.

Para a adaptação do Patricia Trie no contexto de roteamento IP, foi desenvolvida uma implementação em Python, conforme descrito a seguir. Essa estrutura permite armazenar prefixos de endereços IP de forma compacta e eficiente, aproveitando o princípio de compressão para reduzir o espaço de armazenamento e otimizar a busca.

O código descrito no Apêndice G apresenta as classes e funções principais utilizadas para construir e operar a *Patricia Trie*. A classe `PatriciaTrieNode` define cada nó da árvore, enquanto a classe `PatriciaTrie` gerencia a inserção e busca de prefixos. Além disso, foram implementadas funções auxiliares para carregar dados dos nós especiais a partir de um arquivo e para propagar atualizações na rede a partir dos nós especiais.

7.5.2 O tamanho da Tabela de Roteamento e o Desempenho de Memória

A eficiência no roteamento de pacotes em redes IP está diretamente ligada ao tamanho da tabela de roteamento (Sabai, 2017). Com o crescimento exponencial da internet, a quantidade de prefixos e rotas nas tabelas de roteamento também aumenta. Segundo Huawei (2023) esse crescimento implica em maiores exigências de memória e processamento dos dispositivos de rede, especialmente em roteadores que precisam manter essas tabelas e realizar buscas eficientes para encaminhamento de pacotes.

Em uma estrutura de roteamento, cada entrada na tabela de roteamento representa um caminho específico para alcançar um destino (Kamoun; Kleinrock, 1979). No entanto, à medida que a tabela aumenta, o custo de armazenamento e o tempo necessário para buscar cada entrada também aumentam (Huc *et al.*, 2012). Segundo Fei *et al.* (2015) esse aumento não afeta apenas o desempenho de memória, mas também impacta no tempo de processamento, pois o

roteador deve buscar a melhor rota em uma tabela muito maior, gastando mais tempo para achar a rota buscada.

A técnica de redução de tabelas, como a implementação da *Patricia Trie* e a identificação dos nós dominantes (CDS) proposta por Chatterjee *et al.* (2021), busca otimizar essa estrutura, reduzindo o número de entradas e, consequentemente, o uso de memória. Além disso, a compactação da tabela de roteamento identificada pelas autoras permite diminuir o custo computacional associado à busca e à atualização das rotas, fatores que são críticos para a eficiência de redes de grande escala.

Dessa maneira, nesta seção apresentamos uma análise quantitativa do impacto do tamanho da tabela de roteamento. A partir de uma tabela original e uma tabela reduzida, calculamos o percentual de redução obtido, comparando o número de entradas antes e depois do uso das estratégias definidas por Chatterjee *et al.* (2021).

Para avaliar a eficiência da técnica de redução de entradas aplicada à tabela de roteamento, utilizamos o cálculo do percentual de redução. Esse percentual permite entender o quanto a tabela foi compactada usando *Patricia Trie* e CDS, em relação ao seu tamanho original, oferecendo assim um resultado direto a respeito do impacto da técnica em termos de economia de espaço e potencial ganho de desempenho (Chatterjee *et al.*, 2021).

Para o cálculo do percentual de redução foi utilizada a seguinte fórmula:

$$\text{Percentual de Redução} = \left(1 - \frac{\text{Tamanho da Tabela Reduzida}}{\text{Tamanho da Tabela Original}}\right) \times 100$$

Nesta fórmula, o Tamanho da Tabela Original representa o número de entradas na tabela de roteamento antes da aplicação da técnica de redução. Por outro lado, o Tamanho da Tabela Reduzida indica o número de entradas restantes na tabela após a aplicação da técnica de Chatterjee *et al.* (2021).

Tabela 12: Quantitativo de Entradas e Percentual de Redução na Tabela de Roteamento - Chatterjee *et al.* (2021)

Descrição	**Quantidade**
Tamanho da Tabela Original	6.053.407
Tamanho da Tabela Reduzida	972.764
Percentual de Redução	83,94%

Fonte: Elaborado pela autora, (2023).

A Tabela 12 destaca o quantitativo de entradas antes e depois da redução da Tabela. Após a aplicação das estratégias, foi observado uma redução de 83,94%, valor próximo ao que

foi encontrado pelas autoras Chatterjee *et al.* (2021), cuja redução identificada em seus estudos foram o equivalente 70–80%.

7.6 Viabilidade de Implementação da proposta descrita por Saxena e Raychoudhury (2016)

A proposta N-FIB, apresentada por Saxena e Raychoudhury (2016), foi desenvolvida para otimizar o encaminhamento de pacotes em redes NDN. A proposta das autoras recebe o nome de N-FIB, ou Forwarding Information Base baseado em Nomes. Ele consiste em uma estratégia de encaminhamento baseada no uso da *Patricia Trie*, uma estrutura de dados que permite a compressão de prefixos (Hammer; Hellwagner, 2022) reduzindo o espaço para armazenar informações e melhorar o desempenho das buscas.

A estratégia de N-FIB visa consolidar múltiplas entradas de FIB em uma única estrutura, mantendo a precisão de encaminhamento e, ao mesmo tempo, suportando atualizações incrementais sem a necessidade de reconfiguração total (Saxena; Raychoudhury, 2016). Segundo as autoras, esse processo de agregação, combinado com a *Patricia Trie*, permite ao N-FIB manter uma estrutura compacta e eficiente em termos de tempo e memória. Saxena e Raychoudhury (2016), "a *Patricia Trie* reduz significativamente a profundidade da árvore de prefixos, ao mesmo tempo que mantém a integridade e eficiência do encaminhamento."

Dessa forma, o objetivo desta seção é avaliar a viabilidade da implementação do N-FIB no contexto de roteamento IP. Embora a técnica tenha sido desenvolvida para NDN, a adaptação para roteamento IP pode trazer benefícios semelhantes, especialmente em redes com uma grande quantidade de entradas.

7.6.1 Proposta de adaptação ao Contexto de Roteamento IP

A implementação do N-FIB no roteamento IP que propomos segue uma estrutura semelhante tomando como base a proposta das autoras Saxena e Raychoudhury (2016), onde a *Patricia Trie* é usada para permitir que múltiplos prefixos IP sejam agregados de forma compactada. Essa técnica pode ser particularmente vantajosa em redes que lidam com grande quantidade de prefixos IP, como é o caso do *Dataset* selecionado. Segundo Saxena e Raychoudhury (2016) o N-FIB não apenas otimiza o espaço em memória, mas também facilita atualizações incrementais de rota, reduzindo a necessidade de reprocessamento em cada alteração de rota.

Com relação aos processos de agregação dos prefixos no N-FIB, a árvore calcula o prefixo comum (PC) entre o novo prefixo a ser inserido (NP) e o prefixo já armazenado (PA). Dependendo da extensão desse prefixo comum, quatro cenários diferentes podem ocorrer, determinando como o prefixo será agregado na *Patricia Trie* (Saxena; Raychoudhury, 2016):

- Caso o prefixo comum seja igual ao prefixo existente e ao novo prefixo, ambos os prefixos são unidos com os próximos saltos.
- Caso o prefixo comum seja menor, mas ainda corresponda ao prefixo armazenado, ele é agregado à árvore existente.
- Se o prefixo comum é menor que o prefixo existente, o próximo salto é ajustado conforme necessário, mantendo a compactação da árvore.

Com a *Patricia Trie*, a complexidade de memória do N-FIB é reduzida para $O(N)$ uma vez que múltiplos nós filhos são combinados em uma única estrutura (Saxena; Raychoudhury, 2016). Em termos de tempo, a busca por um prefixo, é realizada em $O\left(\frac{M}{k}\right)$, onde M é o comprimento do prefixo e k representa a profundidade reduzida da trie, como especificado anteriormente.

A implementação do N-FIB no roteamento IP que propomos segue uma estrutura semelhante, onde a *Patricia Trie* permite que múltiplos prefixos IP sejam agregados de forma compacta. Em Saxena e Raychoudhury (2016) foi realizado especificamente para NDN, em vez de roteamento IP. Nele, a *Patricia Trie* funciona da seguinte forma: inicialmente, ela agrega múltiplas entradas de FIB em uma única estrutura compacta e posteriormente realiza uma busca pelo prefixo mais longo correspondente ao nome de conteúdo *Longest Prefix Match*/ (LPM) para determinar o próximo salto.

Iniciamos esta proposta, o N-FIB adaptado armazena prefixos IP em uma *Patricia Trie*, onde cada prefixo é convertido em uma *string* binária. Assim como no NDN, a *Patricia Trie* ajuda a reduzir o consumo de memória, armazenando apenas as diferenças nos bits dos prefixos (Saxena; Raychoudhury, 2016). Dessa forma, cada prefixo IP foi convertido em um formado binário e inserido na *Patricia Trie*. Isso permite que prefixos que compartilham sequências iniciais comuns sejam armazenados de maneira otimizada.

A Tabela 13 apresenta os prefixos IP com os seus respectivos próximos saltos. Nela, é possível observar a conversão em binário que foi realizada. Essa conversão permitiu que prefixos que compartilham sequências comuns fossem otimizados.

Imprimimos as primeiras entradas com seus respectivos prefixos, forma binária

Tabela 13: Tabela de prefixos IP com seus próximos saltos e formas binárias

Prefixo IP	Forma Binária do Prefixo	Próximo Salto
1.0.0.0/24	00000001000000000000000	45.184.145.15
1.0.0.0/24	00000001000000000000000	45.184.144.128
1.0.0.0/24	00000001000000000000000	45.184.147.17

Fonte: Elaborado pela autora, (2023).

e próximo salto da tabela, para exemplificar como ficou a estrutura antes da otimização. A Tabela 14 representa somente as primeiras 13 entradas.

Tabela 14: Tabela de Prefixos IP com Próximos Saltos e Formas Binárias Antes da Agregação

Prefixo IP	Forma Binária do Prefixo	Próximo Salto
1.0.0.0/24	00000001000000000000000	45.184.145.15
1.0.0.0/24	00000001000000000000000	45.184.144.128
1.0.0.0/24	00000001000000000000000	45.184.147.17
1.0.0.0/24	00000001000000000000000	45.184.144.102
1.0.0.0/24	00000001000000000000000	45.184.146.59
1.0.4.0/22	00000001000000000001	45.184.144.128
1.0.4.0/22	00000001000000000001	45.184.147.17
1.0.4.0/22	00000001000000000001	45.184.145.15
1.0.4.0/22	00000001000000000001	45.184.146.59
1.0.4.0/22	00000001000000000001	45.184.144.102
1.0.5.0/24	00000001000000000000101	45.184.144.128
1.0.5.0/24	00000001000000000000101	45.184.147.17
1.0.5.0/24	00000001000000000000101	45.184.145.15

Fonte: Elaborado pela autora, (2023).

Com a *Patricia Trie*, os prefixos que compartilham uma sequência inicial comum foram agrupados, otimizando o armazenamento da tabela. O princípio por trás da árvore PATRICIA é usar o prefixo como chave, permitindo assim que a busca seja realizada em partes dessa chave. Como podemos observar, a estratégia adaptada de Saxena e Raychoudhury (2016) também é possível de implementação no contexto de roteamento IP.

7.6.2 O tamanho da Tabela de Roteamento e o Desempenho de Memória

A complexidade de memória é calculada com base no total de armazenamento necessário para guardar os N prefixos ((Peleg; Upfal, 1989), (Saxena; Raychoudhury, 2016), (Thorup; Zwick, 2001)). Segundo a proposta de Saxena e Raychoudhury (2016) com a *Patricia Trie*, a complexidade de memória do N-FIB é reduzida para $O(N)$, uma vez que múltiplos nós

prefixos de nome e reduzindo a profundidade da trie para k (Saxena; Raychoudhury, 2016). Isso quer dizer que uma busca nesse caso teria complexidade equivalente a $O\left(\frac{M}{k}\right)$, onde M é o comprimento do prefixo e k representa a profundidade reduzida da trie.

Dessa forma, apresentamos na Tabela 15 com a redução completa da tabela de roteamento, utilizando a uma adaptação da estratégia N-FIB para agregação de prefixos. A redução de 80,66% no número de entradas evidencia potencial do N-FIB. No entanto, na proposta das autoras Saxena e Raychoudhury (2016) para NDN com suas respectivas especificações de máquina e ambiente, o desempenho foi inferior, apresentando uma redução de 68,18%.

Tabela 15: Quantitativo de Entradas e Percentual de Redução na Tabela de Roteamento - Saxena e Raychoudhury (2016)

Descrição	**Quantidade**
Tamanho da Tabela Original	6.053.407
Tamanho da Tabela Reduzida	1.170.964
Percentual de Redução	80,66%

Fonte: Elaborado pela autora, (2023).

Após verificarmos a viabilidade de implementação das estratégias propostas por Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021), nas próximas seções, nos concentraremos na análise comparativa da redução da tabela de roteamento alcançada por cada método, com o objetivo de commparar as técnicas de redução de tabelas de roteamento em termos tempo de reagregação dos prefixos, uso de memória e redução da tabela. Para isso, descreveremos inicialmente as configurações do nosso esperimento e o conjunto de dados selecionados.

7.7 Análise e Comparação das Técnicas em termos de Quantidade de Prefixos

Inicialmente, vamos analisar o desempenho das duas estratégias de redução de tabelas de roteamento propostas por Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016). A Tabela 16 apresenta o quantitativo da tabela de roteamento original e os resultados obtidos após a aplicação de cada uma das estratégias de redução.

A Tabela Original, conforme descrito em Tabela 16 apresenta o percentual de redução de 99,97% devido ao número de entradas computadas com informações inválidas. Após a leitura do arquivo, as linhas incompletas ou com informações inválidas do arquivo foram rejeitadas ou ignoradas.

Conforme observado, a estratégia de Chatterjee *et al.* (2021) reduziu a tabela original

Tabela 16: Desempenho das Estratégias de Redução de Tabelas de Roteamento de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021)

Descrição	**Quantidade de Prefixos**	**Percentual de Redução**
Tamanho da Tabela Original	6.053.407	99,97%
Chatterjee *et al.* (2021) Tamanho da Tabela Reduzida	972.764	83,94%
Saxena e Raychoudhury (2016) Tamanho da Tabela Reduzida	1.170.964	80,66%

Fonte: Elaborado pela autora, (2023).

de 6.053.407 entradas para 972.764, representando uma redução significativa de 83,94%. A estratégia de Saxena e Raychoudhury (2016), embora também tenha alcançado uma redução expressiva, resultou em um número final de 1.170.964 entradas, o que corresponde a um percentual de redução de 80,66%.

Esses resultados indicam que ambas as estratégias são eficazes na compactação da tabela de roteamento, com a estratégia de Chatterjee *et al.* (2021) mostrando uma redução mais vantajosa em termos de percentual.

7.8 Técnicas de redução mais adequadas em termos de Tempo de Reagregação dos Prefixos e Uso de Memória

Esta seção se concentra em responder as sequintes questões de pesquisa: ***QP5***. *Como as técnicas de redução de tabelas se comportam individualmente, em termos de desempenho e consumo de recursos de memória?* e ***QP6.*** *Quais técnicas apresentaram um melhor desempenho em termos de tempo para reagregação de prefixos, consumo de memória e redução da tabela?*

Dessa forma, discutimos o desempenho das técnicas de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) com base nas métricas de **tempo de reagregação dos prefixos** e **Consumo de memória**. Essa análise contribui para avaliarmos quais estratégias apresentam os menores custos computacionais com relação a um número elevado de entradas a serem processadas.

As estratégias propostas por Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) adotam abordagens distintas para a reagregação, embora ambas utilizem a estrutura de *Patricia Trie*. A técnica de Chatterjee *et al.* (2021) se destaca pelo uso do Conjunto Dominante Conectado (CDS) para selecionar nós “especiais”, que armazenam uma cópia completa da tabela de roteamento. Com isso, somente esses nós dominantes mantêm toda a tabela, enquanto os

demais, os nós “normais”, armazenam apenas dados locais.

Após a seleção dos nós dominantes, a *Patricia Trie* é usada para otimizar o tempo de busca em nós especiais. Dessa forma, para a análise do tempo de reagregação dos prefixos em Chatterjee *et al.* (2021) utilizamos a tabela contida nos nós dominantes, e observamos o tempo de reagregação dos prefixos.

A estratégia de Saxena e Raychoudhury (2016) utiliza a árvore Patricia Trie para realizar sua agregação, combinando múltiplas entradas que possuem os mesmos prefixos iniciais. Para tanto, também foi considerado o tempo de agregação após a estrutura Trie ter sido construída.

A Tabela 17 apresenta os resultados obtidos para o número de entradas e o tempo de processamento em cada estratégia. Observa-se que a técnica de Saxena e Raychoudhury (2016) apresenta um maior número de entradas e consequentemente um tempo de processamento superior em comparação à técnica de Chatterjee *et al.* (2021). Isso demonstra que a estrutura elaborada utilizando a combinação de CDS e da *Patricia Trie* são efetivamente mais eficientes, tanto com relação ao número de entradas que foi reduzido quanto ao tempo de reagregação dos prefixos.

Para a estratégia de Saxena e Raychoudhury (2016), observamos tem tempo de 6.131,103 segundos, enquanto que para a de Chatterjee *et al.* (2021) temos o tempo equivalente a 1.024,822 segundos. O tempo de processamento de Saxena e Raychoudhury (2016) levou 5.106 segundos a mais do que a de Chatterjee *et al.* (2021).

Tabela 17: Comparação de Entradas e Tempo(s) entre os Métodos de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016)

Método	**Entradas**	**Tempo *(s)***
Saxena e Raychoudhury (2016)	1.170.964	6.131,103
Chatterjee *et al.* (2021)	972.764	1.024,822

Fonte: Elaborado pela autora, (2023).

A técnica de Chatterjee *et al.* (2021) se destaca pelo menor tempo de processamento, o que a torna mais eficiente quando ambientes possuir um número de entradas maior, exigindo assim que reagregações sejam frequentes.

Com relação aos requisitos de memória, a estratégia de Chatterjee *et al.* (2021) apresentou o tamanho total em bytes de 2781480, enquanto que Saxena e Raychoudhury (2016) exige um total de 3839378 bytes.

Após a análise dos dados apresentados na Tabela 18, é possível observar uma

Tabela 18: Comparação de Uso de Memória entre Estratégias e Tabela Original

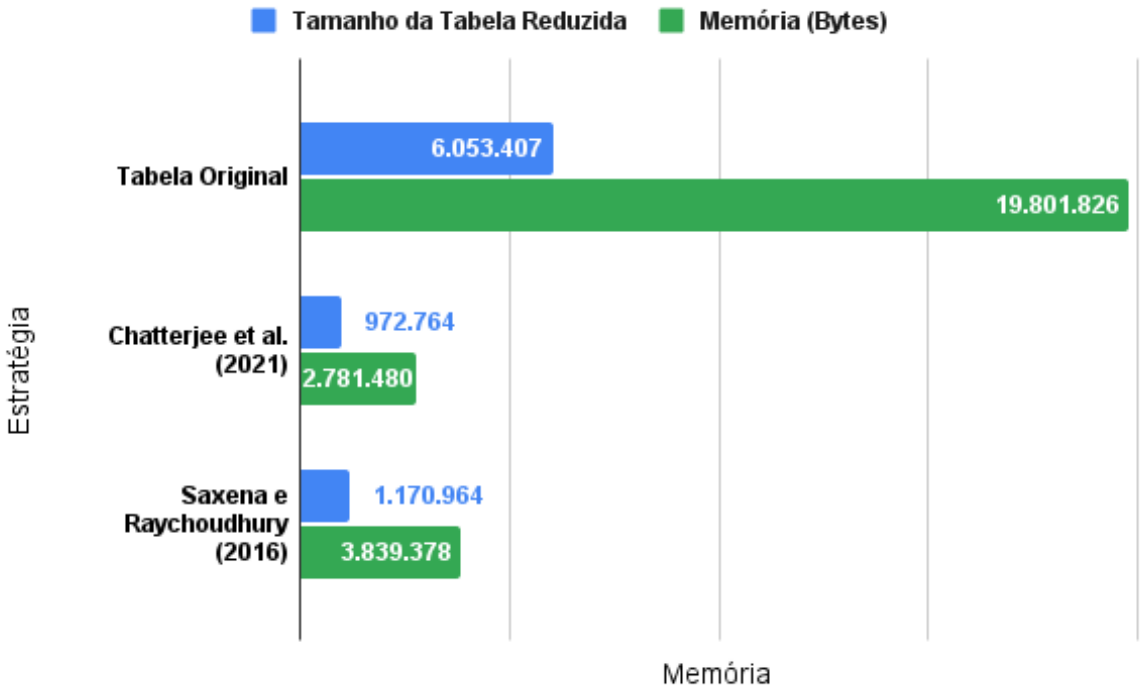
Estratégia	**Tamanho da Tabela Reduzida**	**Memória (Bytes)**
Tabela Original	6.053.407	19.801.826
Chatterjee *et al.* (2021)	972.764	2.781.480
Saxena e Raychoudhury (2016)	1.170.964	3.839.378

Fonte: Elaborado pela autora, (2023).

diferença significativa no uso de memória entre a tabela original e as estratégias de redução. A tabela original, sem nenhuma técnica de agregação, ocupa o equivalente a 19.801.826 bytes, enquanto as estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) conseguem reduzir o uso de memória para 2.781.480 bytes e 3.839.378 bytes, respectivamente.

Para uma visualização dessas diferenças de memória, o Gráfico 9 ilustra o uso de memória para cada uma das abordagens. Nele é possível observar efetivamente a eficiência das estratégias de redução em comparação com a tabela original, evidenciando assim os ganhos em economia de espaço proporcionados pelas técnicas de agregação.

Figura 9: Comparação de Uso de Memória entre a Tabela Original e as Técnicas de Redução



Fonte: Elaborado pela autora, (2024).

Como é possível observar a partir dos dados apresentados nesta seção, a escolha da técnica de redução de tabelas de roteamento impacta significativamente tanto o tempo de

reagregação dos prefixos quanto no uso de memória. A técnica de Chatterjee *et al.* (2021) se mostrou mais eficiente em termos de uso de memória e reagregação de prefixos, o que a torna mais vantajosa para volumes maiores de entradas e que consequetemente apresentam frequentes necessidades de reagregação. Embora a técnica de Saxena e Raychoudhury (2016) ofereça uma redução significativa, ela exigiu mais tempo de processamento e um uso de memória maior em comparação com a abordagem de Chatterjee *et al.* (2021).

7.9 Escolha da Técnica e o impacto no Consumo de Memória e o tempo de Execução

.

Para respondermos as nossas questões de pesquisa: ***QP3***. *Quais das técnicas de redução de tabelas de roteamento avaliadas são mais adequadas em termos de desempenho e consumo de recursos?* e ***QP4***. *Como a escolha da técnica de redução de tabelas de roteamento impacta no consumo de memória e tempo de execução?*, realizamos uma analise comparativa entre as estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) com relação ao consumo de memória e a taxa de compressão.

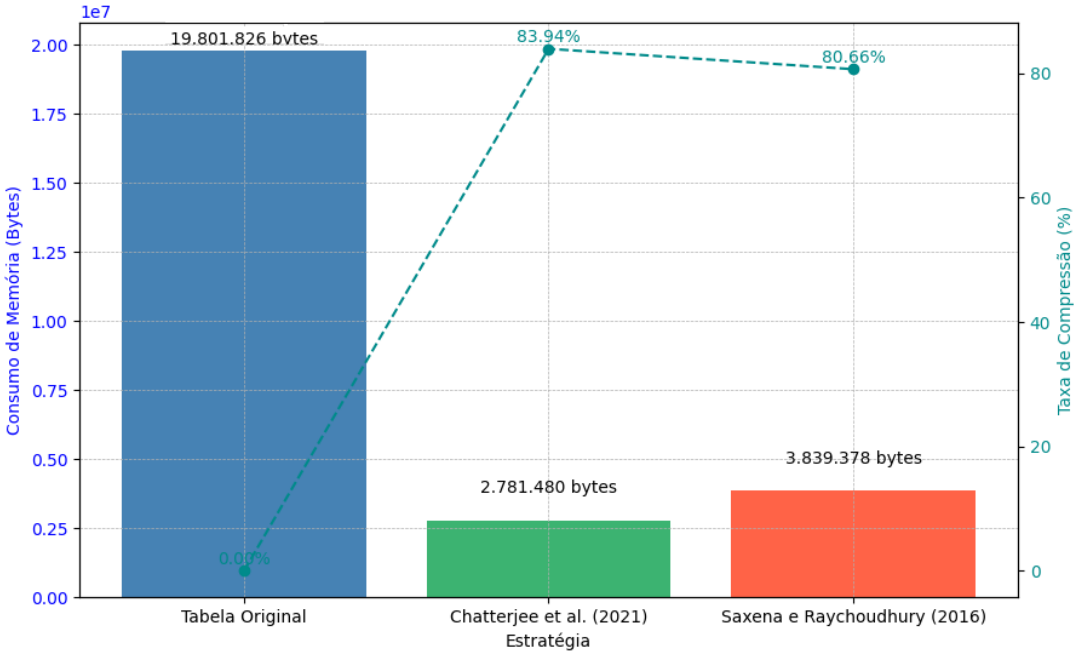
A análise das estratégias de roteamento selecionadas foram realizadas a partir da geração dos gráficos utilizando o software R, Python e um conjunto de tabelas geradas no *Google Sheets*. Para isso, utilizamos como métricas o tempo de execução do código, tempo de conversão da *Patricia Trie*, taxa de compressão e o uso de memória em *bytes*.

O Gráfico 10 apresenta a comparação do consumo de memória e a taxa de compressão entre as estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016), além de apresentar também os valores referentes a tabela original, sem aplicação de técnicas de redução. A partir do gráfico, é possível evidenciar uma diminuição significativa no consumo de memória com relação à tabela inicial. Sem a utilização das estratégias, o espaço ocupado pela tabela original é de aproximadamente 19.801.826 bytes. Com a técnica de Chatterjee *et al.* (2021), observa-se uma redução para 2.781.480 bytes, o que corresponde a uma taxa de compressão de 83,94%.

Em contrapartida, Saxena e Raychoudhury (2016) apresentam um consumo de 3.839.378 bytes, com uma taxa de compressão de 80,66%, apresentando um desempenho menor do que a de Chatterjee *et al.* (2021). Embora ambas as técnicas demonstrem uma redução substancial no uso de memória em comparação com a tabela original, a técnica de Chatterjee *et al.* (2021) que utiliza como recurso o (CDS) com a combinação da *Patricia Trie* apresenta uma

taxa de compressão superior, evidenciando assim a sua eficiência em cenários com um conjunto maior de entradas a serem agregadas.

Figura 10: Comparação do Consumo de Memória e Taxa de Compressão entre as Estratégias de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021)



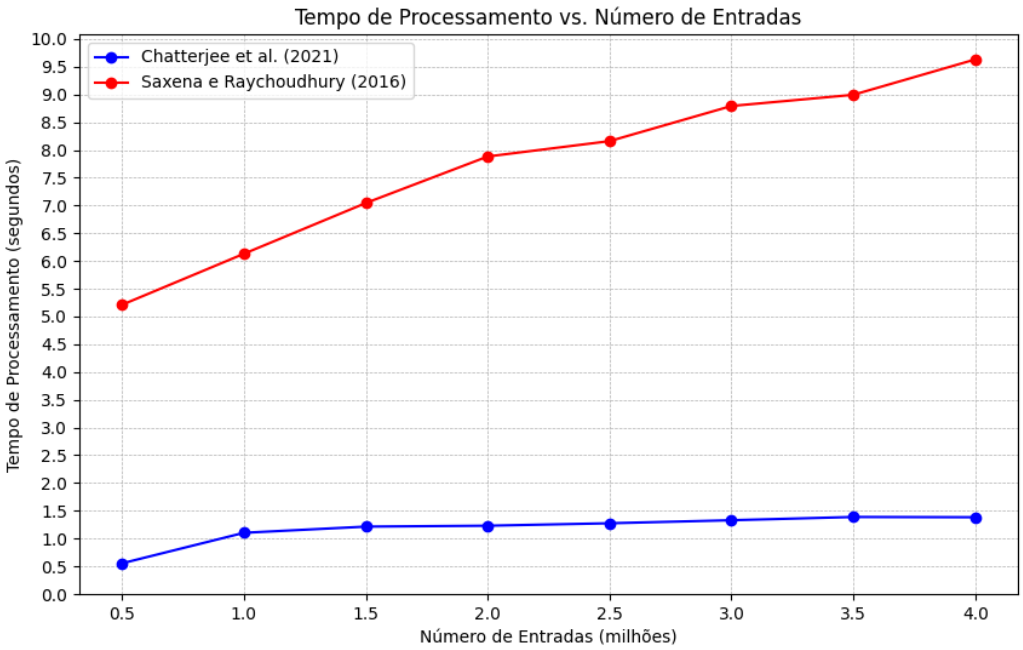
Fonte: Elaborado pela autora, (2024).

Segundo Chatterjee *et al.* (2021) com a utilização da *Patricia Tree* é possível reduzir tanto a memória quanto o tempo de execução, como foi possível observar no Gráfico 10. Ainda segundo as autoras, a combinação do CDS em conjunto com a árvore *Tree* apresentam "considerável melhoria na sobrecarga de armazenamento, atualização e tempo de busca".

Em contrapartida, em Saxena e Raychoudhury (2016), que apresentam a proposta do N-FIB, também é possível considerar uma eficiência significativa na reagregação dos prefixos e no consumo de memória. Segundo as autoras "a agregação direta dos prefixos reduz a sobrecarga de memória".

Para respondermos a questão de pesquisa, a saber: ***QP4***. *Como a escolha da técnica de redução de tabelas de roteamento impacta no consumo de memória e tempo de execução?*, o Gráfico 11 foi construido. Nele apresenta a relação entre o número de entradas (em milhões) e o tempo de processamento (em segundos) para as técnicas de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016). É possível observar que, conforme o número de entradas aumenta, o tempo de processamento também aumenta para ambas as estratégias.

Figura 11: Tempo e Número de Entradas.



Fonte: Elaborado pela autora, (2024).

A técnica de Saxena e Raychoudhury (2016), apresenta um aumento mais acentuado no tempo de processamento das entradas à medida que o volume de dados cresce, contabilizando aproximadamente 4 segundos ao processar 4 milhões de entradas. Esses dados configuram um comportamento que sugere que a técnica apresenta um desempenho menor quando ocorre o aumento no número de entradas, indicando a existência de uma sobrecarga de processamento.

A Tabela 19 a seguir apresenta os desempenhos detalhadamente para cada uma das estratégias com seus respectivos números de entradas. Para os volumes de dados iniciais, até 2 milhões, o tempo se mantém em intervalos razoáveis. Porém, à medida que o número de entradas aumenta, o tempo de processamento também cresce, com uma elevação significativa para valores maiores, como é o caso de 4 milhões de entradas.

Em contrapartida, a técnica de Chatterjee *et al.* (2021), apresenta um crescimento mais gradual no tempo de processamento, mantendo-se abaixo de 1.5 segundos mesmo com o máximo de 4 milhões de entradas. Representando assim um tempo linear, o que indica uma eficiência da técnica em lidar com grandes volumes de dados.

A partir dessa análise, é possível reforçar que a escolha da técnica de redução de tabelas de roteamento impacta diretamente o tempo de processamento, especialmente em contextos onde o número de entradas é elevado. A técnica de Chatterjee *et al.* (2021) demonstra ser mais eficiente em termos de tempo de processamento e agregação dos prefixos, enquanto a

Tabela 19: Comparação entre as técnicas de Chatterjee et al. (2021) e Saxena e Raychoudhury (2016).

Chatterjee et al. (2021)		**Saxena e Raychoudhury (2016)**
Entrada	**Tempo (Média)**	**Tempo (Média)**
0,5	0,553257	5.211,726
1	1.024,822	6.131,103
1,5	1.214,768	7.045,202
2	1.231,067	7.883,943
2,5	1.275,106	8.160,904
3	1.329,781	8.794,192
3,5	1.388,389 3,5	8.995,542
4	1.386,778	9.632,748

Fonte: Elaborado pela autora, (2023).

técnica de Saxena e Raychoudhury (2016) apresenta uma maior sensibilidade ao aumento do volume de dados.

7.9.1 Análise Estatística

Nesta seção, são apresentados os métodos estatísticos utilizados para analisar os dados obtidos a partir da implementação das técnicas de redução de tabelas de roteamento de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016). O objetivo é verificar se há diferenças significativas entre as técnicas em termos de tempo de execução, uso de memória e taxa de redução das tabelas.

7.9.1.1 Teste de Normalidade

Antes de realizarmos a comparação estatística entre as técnicas selecionadas, fez-se necessário verificar os dados obtidos a partir da realização das execuções do código. Para isso, utilizamos a ferramenta SPSS. Inicialmente investigamos a normalidade dos dados para determinar o teste de hipótese mais apropriado para cada variável. Para essa análise, foram aplicados os testes de normalidade de Kolmogorov-Smirnov e Shapiro-Wilk para as variáveis "Tempo de Execução", "Uso de Memória" e "Tempo de Agregação" em ambas as técnicas.

Na Tabela 20 estão apresentados os resultados obtidos por meio do teste de normalidade. Os valores de significância (Sig.) obtidos no relatório por meio da ferramenta SPDSS

Tabela 20: Resultados dos testes Kolmogorov-Smirnov e Shapiro-Wilk para as estratégias analisadas

Estratégia (Autores)	**Métricas**	**Kolmogorov-Smirnov**	**Shapiro-Wilk**
Chatterjee *et al.* (2021)	Tempo de Execução	0,036	0,008
	Uso de Memória	<0,001	<0,001
	Tempo de Agregação	0,036	0,008
Saxena e Raychoudhury (2016)	Tempo de Execução	0,154	0,005
	Uso de Memória	<0,001	<0,001
	Tempo de Agregação	0,154	0,005

Fonte: Elaborado pela autora, (2023).

indicam se as variáveis seguem uma distribuição normal. Para as variáveis de “Tempo de Execução”, “Uso de Memória” e “Tempo de Agregação” a técnica de Chatterjee *et al.* (2021) apresenta valores de uma distribuição não normal, tanto no teste de Kolmogorov-Smirnov quanto no teste de Shapiro-Wilk, ambos indicando valores inferiores ao nível de significância de 0,05.

No entanto, como é possível observar na Tabela 20, a técnica de Saxena e Raychoudhury (2016) apresentou resultados diferentes, sendo considerada como uma distribuição normal pelo teste de Kolmogorov-Smirnov (Sig. = 0,154) e não normal pelo teste de Shapiro-Wilk (Sig. = 0,005) no que se refere a “Tempo de Execução” e “Tempo de Agregação”. Já para a variável “Uso de Memória”, os valores de significância para essa técnica foi inferior a 0,001, indicando que essa variável não segue uma distribuição normal.

Os resultados indicaram que os dados de "Tempo de Execução", "Uso de Memória"e "Tempo de Agregação"da técnica de Chatterjee *et al.* (2021) seguem uma distribuição não normal, enquanto os dados de “Tempo de Execução” e “Tempo de Agregação” para Saxena e Raychoudhury (2016), apresentaram normalidade e apenas o “Uso de Memória” apresenta resultados de não normalidade.

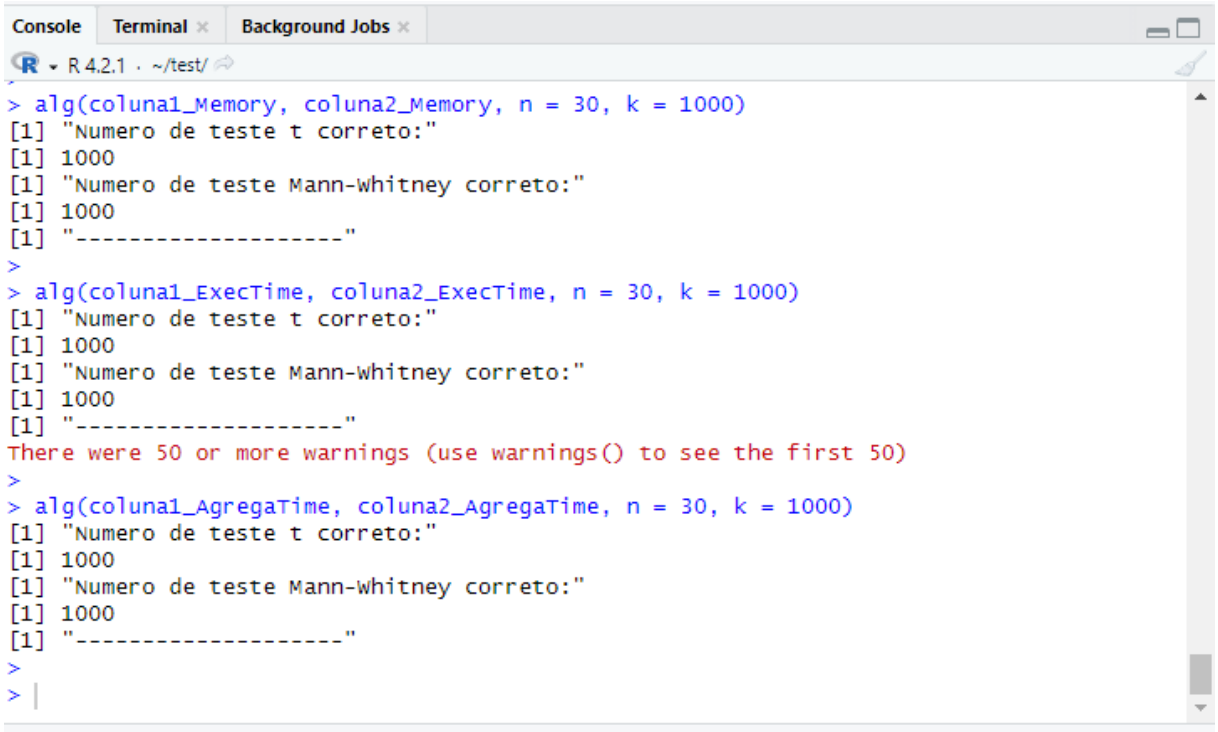
Komatsu (2017) sugere que a escolha do tipo de teste seja feita com base em características da população, estimadas pela amostra. O autor realiza uma estimativa para determinar o poder dos testes t de *Student* e Mann-Whitney Wilcoxon por meio do método de Monte Carlo, comparando duas amostras vindas de populações com 10 milhões de elementos. O autor disponibilizou um algoritmo que permite projetar o tamanho amostral e definir o melhor teste para a comparação no site https://github.com/andrevk/SBBHQK/.

Dessa forma, implementamos o algoritmo proposto por Komatsu (2017) com o objetivo de identificar o teste estatístico mais apropriado para amostras. O procedimento foi

aplicado às variáveis **“tempo de execução”, “uso de memória” e “tempo de agregação”**, considerando amostras separadas por grupo, sendo 1 representando a estratégia de Saxena e Raychoudhury (2016) e 2 a de Chatterjee *et al.* (2021).

Os resultados apresentados na imagem Figura 12 indicaram que ambos os testes, t de *Student* e Mann-Whitney Wilcoxon, apresentaram desempenho consistente nas 1000 iterações realizadas. Essa consistência sugere que ambos os testes podem ser utilizados na análise comparativa. O aviso *"There were 50 or more warnings"* indica que o teste Mann-Whitney encontrou dificuldades para calcular valores exatos de p devido a empates nos dados. Esse é um comportamento esperado quando há muitos valores repetidos ou amostras homogêneas.

Figura 12: Resultados da Aplicação do Algoritmo de Comparação de Testes Estatísticos



Fonte: Elaborado pela autora.

Dessa forma, Komatsu (2017) nos sugere que o teste não paramétrico deve ser preferido para os casos em que: “a distribuição é cortada em um dos lados, como por exemplo não se pode obter valores menores que zero e não há limite (ou o limite é grande) para valores positivos” ou quando a “mediana é a medida que melhor representa a característica da população.” Por fim, o autor conclui que a escolha do teste de hipótese a ser utilizado deve ser feita com base nas características da população (Komatsu, 2017), (Sprent; Smeeton, 2016).

Ao verificarmos as características das amostras observamos que os testes de normalidade (Kolmogorov-Smirnov e Shapiro-Wilk) indicaram que as distribuições não seguem a

normalidade (p-valor < 0,05). Como as amostras não foram consideradas normais, optamos por pela aplicação do teste de Mann-Whitney Wilcoxon para comparar as amostras. Essa decisão está alinhada com as recomendações de Komatsu (2017), Guerra *et al.* (2016) e Sprent e Smeeton (2016), que destacam a adequação dos testes não paramétricos em cenários onde as suposições dos testes paramétricos não são atendidas.

7.9.1.2 Teste de Hipótese

Os testes de hipótese são ferramentas estatísticas utilizadas em pesquisas de diversas áreas do conhecimento. Segundo Siegel e Jr (2006), para decidir se uma determinada hipótese é confirmada por um conjunto de dados, é necessário dispor de um procedimento objetivo que permita rejeitar ou não rejeitar a hipótese. Ao utilizarmos um teste de hipótese, existe o risco de cometer erros que podem distorcer a interpretação dos resultados. Por essa razão, busca-se minimizar, tanto quanto possível, a probabilidade de ocorrência desses erros.

Para avaliarmos os valores comparativos das técnicas de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021) optamos pela aplicação do teste não paramétrico de Mann-Whitney Wilcoxon. (Guerra *et al.*, 2016) os testes não paramétricas são utilizados para os casos em que os dados não satisfazem os testes paramétricos. Ainda segundo Guerra *et al.* (2016) uma das limitações desse tipo de teste é que eles são menos robustos que os testes paramétricos. No entanto, de acordo com a comparação realizada pelo autor Komatsu (2017) as limitações podem ser mitigada pois a escolha do teste depende exclusivamente das características da população da amostra.

7.9.1.3 Teste Wilcoxon-Mann-Whitney

O Teste Wilcoxon-Mann-Whitney foi desenvolvido por Wilcox *et al.* (1998) para duas amostras independentes de tamanhos iguais, e posteriormente foi generalizado para amostras de tamanhos desiguais por Mann e Whitney (1947). Esse tipo de teste é indicado quando se deseja comparar duas amostras não pareadas que pertencem ou não à mesma população uma vez que as suposições do teste t de *Student* não forem possíveis de serem utilizadas (Guerra *et al.*, 2016).

De acordo com Siegel e Jr (2006), o cálculo do teste Mann-Whitney U começa combinando as observações de ambos os grupos em ordem crescente ou decrescente. Os postos mais baixos são atribuídos aos menores valores (ou maiores negativos, se presentes). Em seguida,

considerando apenas um dos grupos, o valor de U é calculado com base no número de vezes que um escore em um grupo precede os escores do outro grupo na classificação ordenada.

Para amostras grandes de n_1 e n_2, os postos são atribuídos de forma ordenada, começando com o valor mais baixo, considerando o grupo total com $(n_1 + n_2)$ elementos. A estatística U é calculada para um dos grupos a partir da fórmula:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1, \tag{7.1}$$

onde R_1 representa a soma dos postos atribuídos ao grupo de tamanho n_1. De maneira similar, pode-se calcular para o outro grupo:

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2, \tag{7.2}$$

onde R_2 é a soma dos postos atribuídos ao grupo de tamanho n_2.

O valor final da estatística U é definido como o menor valor entre U_1 e U_2:

$$U = \min(U_1, U_2). \tag{7.3}$$

O critério para rejeição da hipótese nula é estabelecido quando o valor de U excede $U_{1-\alpha}$, definido pela relação:

$$P(U > U_{1-\alpha}) = \alpha, \tag{7.4}$$

onde α representa o nível de significância escolhido para o teste. Em situações de empate, onde algumas observações compartilham o mesmo valor, estas recebem a média dos postos atribuídos. Embora empates entre escores dentro de um mesmo grupo não causem grandes impactos, quando ocorrem entre grupos distintos, podem influenciar o cálculo de U.

Para corrigir este efeito, uma adaptação é realizada, atribuindo aos empates os postos médios. Isso permite uma aproximação da estatística à distribuição normal, e a estatística z é definida como:

$$z = \frac{R - \frac{n(N+1)}{2}}{\sqrt{\frac{n_1+n_2}{N(N+1)}\left[\left(\sum_{i=1}^{n} R_1^*\right)^2 + \left(\sum_{i=1}^{n} R_2^*\right)^2\right] - \frac{n_1 n_2 (N+1)^2}{4(N+1)}}}. \tag{7.5}$$

em que:

- $N = n_1 + n_2$ é o total de observações nas duas amostras;
- R_j^* representa a soma dos postos empatados na amostra j;
- $Z_{1-\alpha}$ corresponde ao quantil $(1-\alpha)$ da distribuição normal.

Para o presente estudo, utilizamos a ferramenta IBM SPSS para a execução do teste Wilcoxon-Mann-Whitney. Embora o SPSS não exiba diretamente o processo de atribuição de postos e combinação das observações, ele realiza esses cálculos internamente e fornece como saída as estatísticas de postos médios, soma dos postos e o valor de U.

7.9.1.4 Resultados do Teste Wilcoxon-Mann-Whitney

Esta seção apresenta os resultados do teste de Mann-Whitney U, realizado para comparar as amostras em relação às métricas Tempo de Execução, Uso de Memória e Tempo de Agregação, para as estratégias de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021).

A Tabela 21 apresenta a sumarização dos resultados para as três variáveis consideradas: Tempo de Execução (em microssegundos), Uso de Memória (em MB) e Tempo de Agregação (em segundos). Em todos os casos, a hipótese nula foi rejeitada com um nível de significância menor que 0,001, indicando que as distribuições não são iguais entre as categorias. Essa contribui com a decisão de que há diferenças indicando diferenças estatisticamente significativas.

Tabela 21: Tabela de Sumarização de Teste de Hipótese

Hipótese nula	**Teste**	**Sig.**	**Decisão**
A distribuição de Tempo de Execução (em microssegundos) é igual nas categorias de Definição da técnica.	Amostras Independentes de Teste *U* de Mann-Whitney	< 0,001	Rejeitar a hipótese nula.
A distribuição de Uso de Memória (em MB) é igual nas categorias de Definição da técnica.	Amostras Independentes de Teste U de Mann-Whitney	< 0,001	Rejeitar a hipótese nula.
A distribuição de Tempo de Agregação (em segundos) é igual nas categorias de Definição da técnica.	Amostras Independentes de Teste U de Mann-Whitney	< 0,001	Rejeitar a hipótese nula.

Fonte: Elaborado pela autora, (2023).

7.9.1.5 Tempo de Execução

Com relação ao tempo de execução os resultados indicam diferenças estatisticamente significativas entre as duas estratégias ($p < 0,001$), evidenciado pelos postos médios de 45,50 para Saxena e Raychoudhury (2016) e 15,50 para Chatterjee *et al.* (2021).

Essa diferença reflete o comportamento distinto das técnicas em termos de desempenho computacional, sendo a técnica Saxena e Raychoudhury (2016) associada a maiores tempos de execução quando comparada à técnica Chatterjee *et al.* (2021).

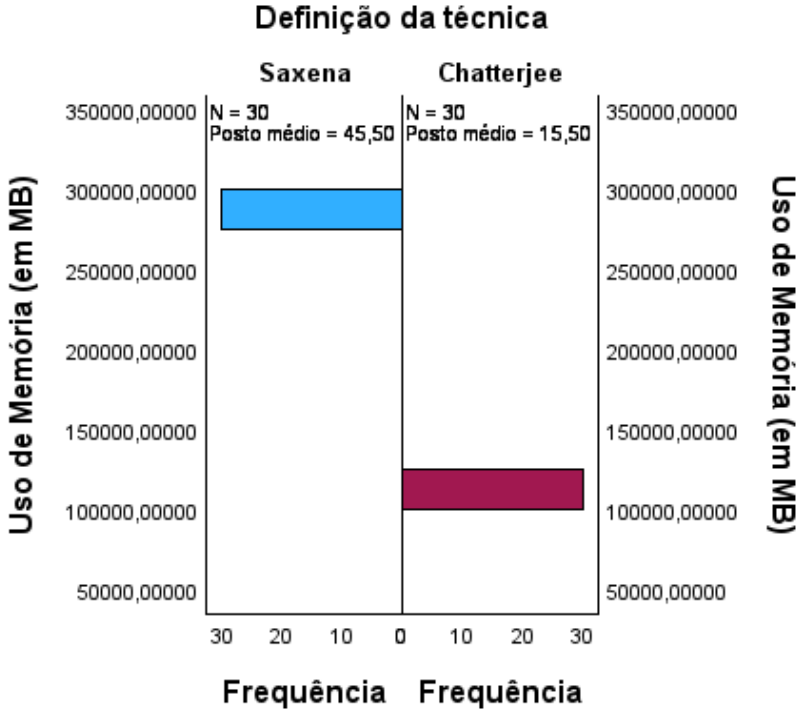
Essa métrica ilustra a superioridade da estratégia adaptada de Chatterjee *et al.* (2021), que realiza a combinação de CDS e *Patricia Trie*, demonstrando que ela se mantém consistente e eficiente mesmo em cenários de alta complexidade. Esses resultados reforçam que a adequação para aplicações onde o tempo de reagregação é um fator crítico. Conforme também concluído por Chatterjee *et al.* (2021): *"O esquema proposto alcança uma distribuição eficiente de dados, resultando em buscas e atualizações mais rápidas com menores requisitos de armazenamento"*.

7.9.1.6 Memória

Quanto ao uso de memória, o gráfico da Figura 13 apresenta os postos médios e as distribuições para as técnicas de Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021). Observa-se uma diferença significativa entre os postos médios das duas técnicas, com valores de 45,50 para Saxena e Raychoudhury (2016) e 15,50 para Chatterjee *et al.* (2021), indicando que a técnica de Chatterjee *et al.* (2021) apresenta um consumo de memória menor e, consequentemente, maior eficiência no uso de memória em comparação à técnica de Saxena e Raychoudhury (2016).

Além disso, podemos concluir que a técnica de Chatterjee *et al.* (2021) apresenta vantagens em relação Saxena e Raychoudhury (2016) devido à sua menor demanda de armazenamento e maior eficiência com um volume grandes de dados. Como destacado em Chatterjee *et al.* (2021), *"O grande volume de dados é armazenado em cada nó utilizando uma estrutura de dados compacta como a Patricia Trie, o que reduz significativamente o tempo de busca"*. Isso explica os valores inferiores de memória observados na técnica em comparação à de Saxena e Raychoudhury (2016).

Figura 13: Amostras Independentes de Teste *U* de Mann-Whitney: Memória



Fonte: Elaborado pela autora.

7.9.2 Discussões

Os resultados obtidos indicam que as estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) adotam abordagens distintas para a reagregação. A técnica de Chatterjee *et al.* (2021) se destaca pelo uso da combinação do Conjunto Dominante Conectado (CDS) aliado a *Patricia Trie*. Esses métodos em conjunto apresentaram uma maior eficiência na redução da tabela e também um maior equilíbrio entre tempo de execução e uso de memória. Isso nos sugere que a escolha da técnica de redução de tabelas de roteamento impacta no consumo de memória e tempo de execução.

Além disso, foi possível constatar que em termos de desempenho, a técnica de Chatterjee *et al.* (2021) também se destacou pelo menor tempo de reagregação dos prefixos. Os testes estatísticos realizados, incluindo o teste de Mann-Whitney U, confirmaram diferenças estatisticamente significativas entre as duas estratégias nas métricas analisadas, demonstrando a eficiência da técnica de Chatterjee *et al.* (2021).

A análise estatística permitiu verificar diferenças significativas entre as duas técnicas, evidenciando postos médios de 45,50 para Saxena e Raychoudhury (2016) e 15,50 para Chatterjee *et al.* (2021) para o tempo de execução, memória e tempo de agregação. Os dados refletem que Saxena e Raychoudhury (2016) associada a maiores tempos de execução, demanda um maior

armazenamento e superioridade de tempo com relação à agregação dos prefixos.

Apesar de Saxena e Raychoudhury (2016) terem apresentado valores inferiores a de Chatterjee *et al.* (2021), ambas as estratégias analisadas são adequadas para redução de tabelas de roteamento. Em cenários onde o consumo de memória e o tempo de reagregação são fatores críticos, a técnica de Chatterjee *et al.* (2021) se apresenta como a melhor opção.

Quando comparados com estudos anteriores, como os de Detti *et al.* (2012) e Zhang *et al.* (2014), os resultados desta pesquisa contribuem com a aplicabilidade de técnicas baseadas em NDN em redes IP, apesar de apontarem possibilidades de melhorias futuras. Esses estudos destacam os desafios na integração de NDN/IP e apontam estratégias de migração entre as duas arquiteturas, o que se alinha aos achados deste trabalho ao considerar a complexidade na adaptação dessas estratégias. Dessa forma, os achados desta pesquisa contribuem para a discussão sobre a compatibilidade entre as arquiteturas NDN e TCP/IP e evidencia a possibilidade de integração de estratégias entre essas duas arquiteturas.

7.9.3 Conclusão

A presente dissertação investigou e comparou duas técnicas de redução de tabelas de roteamento baseadas em conceitos de *Named Data Networking* (NDN), adaptadas para o contexto de roteamento IP. As estratégias de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) foram avaliadas segundo as métricas de consumo de memória, tempo de execução, tempo de reagregação de prefixos e taxa de compressão da tabela. A análise resultou que ambas as técnicas são eficazes, porém, apresentam vantagens e limitações dependendo do contexto de aplicação.

A técnica de Chatterjee *et al.* (2021) destacou-se como a mais eficiente em termos de consumo de memória e tempo de execução, alcançando uma taxa de compressão superior (83,94%) e menor tempo de reagregação, devido à combinação do *Connected Dominating Set* (CDS) com a *Patricia Trie*. Essa estratégia se mostrou particularmente adequada para cenários que envolvem grandes volumes de dados e exigem alta eficiência em busca e armazenamento.

Por outro lado, a técnica de Saxena e Raychoudhury (2016) demonstrou ser robusta e eficiente em termos de simplicidade na implementação, mas apresentou maior consumo de memória e tempo de reagregação. Apesar dessas limitações, a técnica N-FIB é uma alternativa promissora para redes que requerem atualizações frequentes, uma vez que a árvore *Trie* otimiza a manutenção da estrutura sem a necessidade de reconstrução total da tabela.

Os resultados também indicam que a escolha da técnica de redução impacta significativamente no desempenho e nos recursos utilizados. A análise estatística, baseada no teste de Mann-Whitney U, confirmou que existem diferenças significativas entre as estratégias, evidenciando a superioridade da abordagem de Chatterjee *et al.* (2021) nas métricas avaliadas.

Além disso, a pesquisa contribuiu para a discussão sobre a adaptabilidade de técnicas desenvolvidas para NDN no contexto de roteamento IP, apresentando evidências de que estratégias originalmente projetadas para arquiteturas de redes de dados podem ser aplicadas a redes IP.

7.9.4 Ameaças à validade

Nesta seção abordaremos as ameaças à validade do estudo, destacado a partir das subseções de Validade Interna e Validade Externa.

Validade interna: Uma ameaça interna do presente estudo é o número de técnicas comparadas, restringindo-se a apenas duas selecionadas para a análise a partir dos critérios que foram estabelecidos. Embora a escolha seja limitada, as técnicas selecionadas representam o grupo de estratégias de otimização de recursos aplicadas no contexto de NDN e adaptadas para o protocolo IP. A escolha por essas estratégias se justifica por meio do estabelecimento dos critérios da RSL, que identifiou um número restrito de pesquisas que realizam estudos nessa perspectiva de adaptabilidade.

Outra questão identificada é que é a ausência de simulações como parte do método de validação. A opção por não utilizarmos simulação foi tomada para mantermos o foco da análise das técnicas de redução em um ambiente controlado de programação, usando dados extraídos de um *Dataset*. No entanto, notamos que a ausência de simulação limita a capacidade de análise do comportamento das técnicas em contextos reais e com redes mais complexas. Para mitigar essa limitação foram realizadas inicialmente capturas com volumes de dados isolados para análise das estratégias, e em seguida foi-se aumentando o número de entradas e analisando os respectivos comportamentos das técnicas.

A captura dos dados foi então realizada para volumes de entradas diferentes, e posteriromente foi verificado o seu valor completo de entradas.

Validade externa: As estratégias foram implementadas em Python, a partir dos algoritmos fornecidos pelos autores. Mesmo assim, as técnicas implementadas podem representar uma potencial ameaça a validade. Por terem sido adaptadas de um contexto que se diferencia da

opção escolhida pelos autores, isso pode representar uma variação entre os valores apresentados e os identificados pelo presente estudo.

Outra ameaça identificada está relacionada ao *Dataset* utilizado e às especificações da estação de trabalho, fatores que podem influenciar os resultados e o número de entradas selecionadas para análise. Para contornar esse problema, foi utilizado um conjunto de dados de referência, extraído do RouteViews.

Inicialmente, foram levantados dados para a realização de comparações com os valores de referência disponíveis na literatura das técnicas propostas. Aliado a isso, foi aplicado um teste de hipótese para avaliar se as diferenças observadas entre as técnicas são estatisticamente significativas.

8 CONSIDERAÇÕES FINAIS

O presente trabalho se concentrou inicialmente em identificar, a partir de uma Revisão Sistemática da Literatura (RSL) as técnicas de redução de tabelas de roteamento propostas na literatura. Por meio dela, foi possível localizar as estratégias de redução de tabela e categorizar os ambientes em que se concentram as analises dessas soluções.

Após a seleção dos estudos, foi realizado a escolha do método de análise e escolha das estratégias. Para isso, foram estabelecidos critérios de qualidade dos trabalhos. Foram levados em consideração para a seleção das técnicas a serem comparadas a disponibilidade de código fonte, para que houvesse a possibilidade de implementação do método, o tipo de ambiente em que foi testada a proposta, analisando se foi proposta em linguagem de programação ou em uma ambiente de simulação. Por fim, também foi selecionado como critério o impacto do período em que o artigo foi publicado.

Identificamos que a RSL capturou estudos envolvendo a redução de tabelas de roteamento aplicadas à arquitetura NDN e técnicas para o roteamento convencional IP. Foi observado então, a partir da triangulação entre as tabelas que continham os critérios de qualidade: 4-Informações de base, periódico e código-fonte, 5- Informações do método e tipo de ambiente utilizado, 6- Periódicos de maior impacto na área, que as estratégias aprovadas posterior ao refinamento foram as das autoras Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016).

A abordagem das técnicas de Chatterjee *et al.* (2021) e Saxena e Raychoudhury (2016) são configuradas para NDN, utilizando a combinação de CDS e *Patricia Tree*, e apenas *Patricia Tree*, respectivamente. Dessa maneira, prosseguiu-se para o estudo da viabilidade de implementação adaptando as propostas ao contexto de roteamento IP.

Assim, as técnicas aplicadas ao contexto de NDN foram utilizadas como inspiração para novas abordagens do roteamento IP. Para tanto, foi realizada uma análise individual das técnicas, abordando dois aspectos: como elas realizam a “Descoberta” e a “Manutenção” das “Rotas”. Em seguida, foram elaboradas as propostas e implementadas as técnicas com base em seus conceitos fundamentais. No que diz respeito a Chatterjee *et al.* (2021), foram selecionados os nós dominantes com base na centralidade e proximidade, definido por Chang *et al.* (2016). Posteriormente, foi implementada a *Patricia Tree* para facilitar a atualização com base em chaves binárias (Morrison, 1968b).

Para Saxena e Raychoudhury (2016), a *Patricia Trie* é usada para permitir que múltiplos prefixos IP sejam agregados de forma compactada. O processo de agregação dos

prefixos ocorre da seguinte forma: a árvore calcula o prefixo comum (PC) entre o novo prefixo a ser inserido (NP) e o prefixo já armazenado (PA). Dependendo da extensão desse prefixo comum, as autoras apresentam os seguintes cenários possíveis: quando o PC é idêntico ao NP e ao PA, os prefixos são unidos com seus respectivos próximos saltos; quando o PC é menor, mas ainda corresponde ao PA, ele é agregado à árvore existente; e, por fim, quando o PC é menor que o PA, o próximo salto é ajustado conforme necessário, preservando a compactação da árvore.

Dessa forma, as técnicas foram adaptadas e implementadas para o contexto de roteamento IP, utilizando estruturas definidas pelas autoras, como a *Patricia Trie* e o CDS. A análise revelou que a partir da utilização dessas estratégias, é possível alcançar uma redução no uso de memória e tempo de reagregação dos prefixos, o que contribui diretamente para o aumento da eficiência no processamento de volumes de dados com número considerável de entradas.

A técnica de Chatterjee *et al.* (2021), que utiliza a combinação de *Patricia Trie* e o CDS para otimizar o armazenamento nós especiais, demonstrou uma redução de até 83,94% na tabela de roteamento original, enquanto a estratégia de Saxena e Raychoudhury (2016) resultou em uma economia de 80,66%. Os dados apresentados revelam o impacto positivo da utilização de estruturas otimizadas no desempenho da compressão da tabela de roteamento, tornando-se viável suas aplicações para o processamento de volumes maiores de dados.

Para determinar se existiam diferenças significativas entre as técnicas selecionadas, foram realizados testes estatísticos. Por meio dos testes de normalidade de Kolmogorov-Smirnov e Shapiro-Wilk, constatou-se que as variáveis “Tempo de Execução”, “Uso de Memória” e “Tempo de Agregação” não apresentaram uma distribuição normal para a técnica de Chatterjee *et al.* (2021), enquanto os resultados para Saxena e Raychoudhury (2016) foram parcialmente normais pelo teste de Kolmogorov-Smirnov (Sig. = 0,154) e não normal pelo teste de Shapiro-Wilk (Sig. = 0,005) no que se refere a “Tempo de Execução” e “Tempo de Agregação”.

Com base nessas observações, e utilizando o algoritmo proposto por Komatsu (2017) identificamos que o teste estatístico mais apropriado para amostras com dados não paramétrico é o de Mann-Whitney Wilcoxon, que revelou diferenças estatisticamente significativas entre as duas estratégias em todas as métricas analisadas ($p < 0,001$). Em termos de desempenho, a técnica de Chatterjee *et al.* (2021) demonstrou maior eficiência, com menor tempo de execução e agregação, além de menor consumo de memória em comparação à técnica de Saxena e Raychoudhury (2016).

Os resultados estatísticos reforçam que a utilização da combinação de CDS e a *Patricia Trie* proposta por Chatterjee *et al.* (2021), destaca-se como a melhor abordagem em cenários onde o tempo de reagregação e o consumo de memória são fatores críticos. Apesar disso, é possível afirmar que ambas as técnicas se mostraram adequadas para a redução de tabelas de roteamento, atendendo aos requisitos de desempenho destacados nesta dissertação.

Os resultados refletem as possibilidades de adaptação de estratégias NDN em contextos IP, nos revelando que a escolha de implementação da técnica de redução influencia diretamente o tempo de reagregação dos prefixos e o consumo de recursos de memória. Por fim, concluimos que a implementação de estratégias de compressão e agregação, como é o caso da *Patricia Trie* associada ao CDS, apresenta um grande potencial para melhorar o desempenho de processamento das tabelas de roteamento no contexto das redes IP.

8.1 Contribuições

A presente dissertação tem como principal objetivo avaliar o comportamento de técnicas de redução de tabelas de roteamento, estruturando-se metodologicamente em uma Revisão Sistemática da Literatura. Inserindo-se em uma abordagem mista de pesquisa, combinado métodos qualitativos e quantitativos, o estudo utiliza ferramentas estatísticas para analisar os dados obtidos a partir da implementação prática das técnicas selecionadas. Essa estratégia visa garantir a precisão dos resultados e minimizar a probabilidade de erros, assegurando a validade dos resultados.

Dessa forma, as contribuições dessa dissertação incluem os seguintes aspectos:

- **Metodologia baseada em uma Revisão Sistemática da Literatura (RSL):** A identificação e seleção das técnicas foram feitas com base em uma RSL, garantindo a relevância e atualidade das estratégias selecionadas para a análise do estudo. A partir dos estudos iniciais, foi possível identificar que apesar de muitos trabalhos apresentarem a proposição de um conjunto de técnicas, como é o caso de Kamoun e Kleinrock (1979), Wang *et al.* (2006), Farias *et al.* (2014) e Anand *et al.* (2019), não foi possível localizar um estudo recente que apresenta-se um estado da arte comparando as técnicas de redução de tabelas propostas na literatura atualmente.
- **Proposta de Ambiente Programável:** Diferentemente de muitos trabalhos que utilizam simulações, como foi possível observar nos artigos de Castaneda *et al.* (2020), Gopalan e Ramasubramanian (2016), Duquennoy *et al.* (2013) e Mottaghi *et al.* (2022), a presente dis-

sertação implementa as técnicas de redução diretamente em um ambiente de programação, permitindo um controle e análise detalhados dos cenários e das amostras de dados.

- **Adaptação ao Contexto de Roteamento IP:** A pesquisa demonstra a viabilidade de aplicar técnicas originalmente projetadas para NDN (Named Data Networking) no contexto de roteamento IP. À medida que as investigações sobre NDN avançam, estudos como os de Detti *et al.* (2012) e Zhang *et al.* (2014) destacam que novas discussões surgem a respeito da sua implantação no mundo real, especialmente em relação à compatibilidade entre redes TCP/IP.
- **Comparação de Técnicas de Redução:** O trabalho apresenta uma análise detalhada entre as técnicas de redução de tabelas de roteamento propostas por Saxena e Raychoudhury (2016) e Chatterjee *et al.* (2021). A análise permitiu observar ambas as estratégias são eficazes na compactação da tabela de roteamento, com a estratégia de Chatterjee *et al.* (2021) mostrando uma redução mais vantajosa em termos de percentual.
- **Resultados Práticos e Percentuais de Redução:** Os resultados identificaram que a técnica baseada em CDS e *Patricia Trie* (Chatterjee *et al.* (2021)) proporcionou uma redução significativa de 83,94% na tabela de roteamento, enquanto a estratégia de Saxena e Raychoudhury (2016) alcançou 80,66%. Resultados percentuais se aproximam consideravelmente dos valores de referência quando aplicados ao contexto NDN pelas autoras, o que permite concluirmos que as duas soluções fornecem possibilidades para mitigar os desafios de memória e processamento enfrentados por redes com números maiores de entradas de dados.

8.2 Trabalhos Futuros

Apresenta-se por fim como propostas para trabalhos futuros:

- **Explorar outras estratégias de otimização de tabelas de roteamento:** Comparar outras estratégias desenvolvidas especificamente para *Named Data Networking* (NDN) para o contexto do roteamento IP, investigando como as adaptações realizadas impactam no desempenho, eficiência e agregação dos prefixos.
- **Análise de Desempenho baseada em Simulações:** Realizar uma análise de desempenho das estratégias utilizando simuladores, buscando minimizar os recursos computacionais exigidos na estação de trabalho e permitir experimentações em ambientes com maior escala e complexidade.

- **Investigação em cenários de grande escala:** Avaliar o impacto das estratégias de redução utilizando *datasets* com volumes de entradas significativamente maiores, explorando como o aumento do tamanho da tabela influencia métricas como tempo de execução, uso de memória e taxa de agregação dos prefixos.
- **Aprimoramento da *Patricia Trie*:** Investigar possíveis melhorias na estrutura *Patricia Trie* com base na literatura e na aplicabilidade desse método em diferentes contextos, com o objetivo de otimizar o consumo de memória e reduzir o tempo de reagregação.

REFERÊNCIAS

ABRAHAM, I.; GAVOILLE, C. On approximate distance labels and routing schemes with affine stretch. In: PELEG, D. (Ed.). **Distributed Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 404–415. ISBN 978-3-642-24100-0.

ABRAHAM, I.; GAVOILLE, C.; MALKHI, D.; NISAN, N.; THORUP, M. Compact name-independent routing with minimum stretch. **ACM Trans. Algorithms**, Association for Computing Machinery, New York, NY, USA, v. 4, n. 3, jul 2008. ISSN 1549-6325.

ALMUHTADI, W.; FENWICK, W.; HENLEY-VACHON, L.; MITCHELL, P. Assessing network infrastructure-as-code security using open source software analysis techniques applied to bgp/bird. In: **2022 IEEE International Conference on Consumer Electronics (ICCE)**. [*S. l.*: *s. n.*], 2022. p. 1–6.

AMER, F.; LIEN, Y.-N.; GHIETH, A. Performance evaluation of a hierarchical hybrid adaptive routing algorithm for large scale computer communication networks. In: **[1988] Proceedings. Computer Networking Symposium**. [*S. l.*: *s. n.*], 1988. p. 266–275.

ANAND, S.; MATHIKSHARA, P. M.; JAYAVIGNESH, T. An efficient mask reduction strategy to optimize storage and computational complexity in routing table lookups. In: **2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)**. [*S. l.*: *s. n.*], 2019. p. 1–5.

ANDO, Y.; NAGAO, H.; MIYAO, T.; SHUDO, K. Routing table construction method solely based on query flows for structured overlays. In: IEEE. **14-th IEEE International Conference on Peer-to-Peer Computing**. [*S. l.*], 2014. p. 1–5.

ASHRAF, Z.; YOUSAF, M. Optimized routing information exchange in hybrid ipv4-ipv6 network using ospfv3 & eigrpv6. **International Journal Of Advanced Computer Science And Applications**, Science and Information (SAI) Organization Limited, v. 8, n. 4, 2017.

ASHRAF, Z.; YOUSAF, M. Optimized convergence of ospfv3 in large scale hybrid ipv4-ipv6 network. In: **2018 14th International Conference on Emerging Technologies (ICET)**. [*S. l.*: *s. n.*], 2018. p. 1–6.

BI, J.; WU, J.; LENG, X. Ipv4/ipv6 transition technologies and univer6 architecture. **International Journal of Computer Science and Network Security**, Citeseer, v. 7, n. 1, p. 232–243, 2007.

CASTANEDA, A.; LEFéVRE, J.; TREHAN, A. Fully compact routing in low memory self-healing trees. In: **Proceedings of the 21st International Conference on Distributed Computing and Networking**. New York, NY, USA: Association for Computing Machinery, 2020. (ICDCN '20). ISBN 9781450377515.

CHANG, K.-C.; CHEN, C.; KIANG, Y.-J.; ZHOU, W. A study of influencing factors of patent value based on social network analysis. In: IEEE. **2016 Portland International Conference on Management of Engineering and Technology (PICMET)**. [*S. l.*], 2016. p. 1470–1477.

CHATTERJEE, T.; RUJ, S.; BIT, S. D. Efficient data storage and name look-up in named data networking using connected dominating set and patricia trie. **Automatic Control and Computer Sciences**, Pleiades journals, v. 55, p. 319–333, 7 2021. ISSN 1558108X.

CHECHIK, S. Compact routing schemes with improved stretch. In: **Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing**. New York, NY, USA: Association for Computing Machinery, 2013. (PODC '13), p. 33–41. ISBN 9781450320658. Disponível em: https://doi.org/10.1145/2484239.2484268. Acesso em: 26 dez. 2024.

DAHLGAARD, S.; KNUDSEN, M. B. T.; ROTBART, N. A simple and optimal ancestry labeling scheme for trees. In: HALLDÓRSSON, M. M.; IWAMA, K.; KOBAYASHI, N.; SPECKMANN, B. (Ed.). **Automata, Languages, and Programming**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. p. 564–574. ISBN 978-3-662-47666-6.

DETTI, A.; POMPOSINI, M.; BLEFARI-MELAZZI, N.; SALSANO, S.; BRAGAGNINI, A. Offloading cellular networks with information-centric networking: The case of video streaming. In: **2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)**. [*S. l.*: *s. n.*], 2012. p. 1–3.

DU, H.; WEI, W.; YANG, J. A b-tree algorithm for partitioned index based on cidr list in high-end router. In: **2011 International Conference of Information Technology, Computer Engineering and Management Sciences**. [*S. l.*: *s. n.*], 2011. v. 2, p. 124–128.

DUQUENNOY, S.; LANDSIEDEL, O.; VOIGT, T. Let the tree bloom: Scalable opportunistic routing with orpl. In: **Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems**. New York, NY, USA: Association for Computing Machinery, 2013. (SenSys '13). ISBN 9781450320276.

ELKIN, M.; NEIMAN, O. Near-optimal distributed routing with low memory. In: **Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing**. New York, NY, USA: Association for Computing Machinery, 2018. (PODC '18), p. 207–216. ISBN 9781450357951.

ELKIN, M.; NEIMAN, O. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in rnc. In: **The 31st ACM Symposium on Parallelism in Algorithms and Architectures**. New York, NY, USA: Association for Computing Machinery, 2019. (SPAA '19), p. 333–341. ISBN 9781450361842.

FAHRIANTO, F.; KAMIYAMA, N. Comparison of migration approaches of icn/ndn on ip networks. In: **2020 Fifth International Conference on Informatics and Computing (ICIC)**. [*S. l.*: *s. n.*], 2020. p. 1–7.

FANELLI, A.; FLAMMINI, M.; MANGO, D.; MELIDEO, G.; MOSCARDELLI, L. Experimental evaluations of algorithms for ip table minimization. **Journal of Interconnection Networks**, World Scientific Publishing Co. Pte Ltd, v. 12, p. 299–318, 2011. ISSN 17936713.

FARIAS, L. F.; DINIZ, M. C.; LUCENA, S. C. de. Algoritmo de seleção e filtragem de rotas para redução das tabelas de encaminhamento. 2014. Disponível em: http://www.repositorio-bc.unirio.br:8080/xmlui/bitstream/handle/unirio/12020/MI%2019-2014.pdf?sequence=1&isAllowed=y. Acesso em: 26 dez. 2024.

FEI, Z.; YANG, J.; LU, H. Improving routing efficiency through intermediate target based geographic routing. **Digital Communications and Networks**, v. 1, n. 3, p. 204–212, 2015. ISSN 2352-8648. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352864815000413. Acesso em: 26 dez. 2024.

FULLER, V.; LI, T. **Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan**. [*S. l.*], 2006.

FULLER, V.; LI, T.; VARADHAN, K.; YU, J. **Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy**. RFC Editor, 1993. RFC 1519. (Request for Comments, 1519). Disponível em: https://www.rfc-editor.org/info/rfc1519. Acesso em: 26 dez. 2024.

GAWRYCHOWSKI, P.; JANCZEWSKI, W.; ŁOPUSZAńSKI, J. **Shorter Labels for Routing in Trees**. 2021.

GHOSE K. E DESAI, K. Hierarchical cubic networks. **IEEE Transactions on Parallel and Distributed Systems**, v. 6, 1995.

GONÇALVES, E.; CASTRO, J.; ARAúJO, J.; HEINECK, T. A systematic literature review of istar extensions. **Journal of Systems and Software**, v. 137, p. 1–33, 2018. ISSN 0164-1212. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121217302741. Acesso em: 26 dez. 2024.

GOPALAN, A.; RAMASUBRAMANIAN, S. Ip fast rerouting and disjoint multipath routing with three edge-independent spanning trees. **IEEE/ACM Transactions on Networking**, v. 24, n. 3, p. 1336–1349, 2016.

GUERRA, G. d. S. *et al.* Avaliação do desempenho e adequação de diferentes metodologias aplicadas para modelagem e inferência de emissão de gases do efeito estufa. Universidade Federal Rural de Pernambuco, 2016.

HAMMER, J.; HELLWAGNER, H. Tinytricia – a space-optimized patricia trie for transparent access to edge computing services. In: **2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)**. [*S. l.*: *s. n.*], 2022. p. 340–345.

HIROMORI, A.; YAMAGUCHI, H.; YASUMOTO, K.; HIGASHINO, T.; TANIGUCHI, K. Reducing the size of routing tables for large-scale network simulation. In: **Seventeenth Workshop on Parallel and Distributed Simulation, 2003. (PADS 2003). Proceedings.** [*S. l.*: *s. n.*], 2003. p. 115–122.

HOJO, M.; BANNO, R.; SHUDO, K. Frt-skip graph: A skip graph-style structured overlay based on flexible routing tables. In: **2016 IEEE Symposium on Computers and Communication (ISCC)**. [*S. l.*: *s. n.*], 2016. p. 657–662.

HSIAO, S.-F.; CHEN, K.-C.; CHEN, Y.-H. Optimization of lookup table size in table-bound design of function computation. In: **2018 IEEE International Symposium on Circuits and Systems (ISCAS)**. [*S. l.*: *s. n.*], 2018. p. 1–4.

HU, H.; BI, J.; FENG, T.; WANG, S.; LIN, P.; WANG, Y. A survey on new architecture design of internet. In: **2011 International Conference on Computational and Information Sciences**. [*S. l.*: *s. n.*], 2011. p. 729–732.

HUANG, P.; HEIDEMANN, J. Minimizing routing state for light-weight network simulation. In: **MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems**. [*S. l.*: *s. n.*], 2001. p. 108–116.

HUAWEI, T. C. Ip routing fundamentals. In: ____. **Data Communications and Network Technologies**. Singapore: Springer Nature Singapore, 2023. p. 155–192. ISBN 978-981-19-3029-4. Disponível em: https://doi.org/10.1007/978-981-19-3029-4_5. Acesso em: 26 dez. 2024.

HUC, F.; JARRY, A.; LEONE, P.; ROLIM, J. On the efficiency of routing in sensor networks. **Journal of Parallel and Distributed Computing**, v. 72, n. 7, p. 889–901, 2012. ISSN 0743-7315.

HUI-JUN, Y. N. D. **The BIRD Internet Routing Daemon Project**. 2009. Disponível em: https://bird.network.cz/. Acesso em: 26 dez. 2024.

JAISWAL, R.; DAVIDRAJUH, R. A simple algorithm for finding steiner spanning trees. In: **2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)**. [*S. l.*: *s. n.*], 2021. p. 1–5.

JIAN-PENG, Z.; SHI-ZE, G.; KANG-FENG, Z.; YI-XUN, H.; FANG-FANG, D. A routing computation strategy based on improved clustering algorithm. In: **2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control**. [*S. l.*: *s. n.*], 2012. p. 486–489.

JIAN, S.; FANG, Y. Y. Research and implement of ospfv3 in ipv6 network. In: **Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference**. [*S. l.*: *s. n.*], 2011. v. 1, p. 743–746.

KAMOUN, F.; KLEINROCK, L. Stochastic performance evaluation of hierarchical routing for large networks. **Computer Networks (1976)**, v. 3, n. 5, p. 337–353, 1979. ISSN 0376-5075.

KAWAGUCHI, T.; BANNO, R.; HOJO, M.; SHUDO, K. Self-refining skip graph: A structured overlay approaching to ideal skip graph. In: **2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)**. Los Alamitos, CA, USA: IEEE Computer Society, 2016. p. 377–378. ISSN 0730-3157. Disponível em: https://doi.ieeecomputersociety.org/10.1109/COMPSAC.2016.188. Acesso em: 26 dez. 2024.

KERNEN., T. 2005. Disponível em: http://www.traceroute.org/#RouteServers. Acesso em: 26 dez. 2024.

KINI, S.; RAMASUBRAMANIAN, S.; KVALBEIN, A.; HANSEN, A. F. Fast recovery from dual link failures in ip networks. In: **IEEE INFOCOM 2009**. [*S. l.*: *s. n.*], 2009. p. 1368–1376.

KITCHENHAM, B.; BRERETON, P. A systematic review of systematic review process research in software engineering. **Information and Software Technology**, v. 55, n. 12, p. 2049–2075, 2013. ISSN 0950-5849. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950584913001560. Acesso em: 26 dez. 2024.

KITCHENHAM, B.; CHARTERS, S. *et al.* **Guidelines for performing systematic literature reviews in software engineering**. [*S. l.*]: UK, 2007.

KLEINROCK, L.; KAMOUN, F. Hierarchical routing for large networks performance evaluation and optimization. **Computer Networks (1976)**, v. 1, n. 3, p. 155–174, 1977. ISSN 0376-5075.

KOMATSU, A. V. Comparação dos poderes dos testes t de student e man-whitney wilcoxon pelo método de monte carlo. **Revista da Estatística da Universidade Federal de Ouro Preto**, v. 6, 2017.

LI, J.; WEN, X.; WU, M.; LIU, F.; LI, S. Identification of key nodes and vital edges in aviation network based on minimum connected dominating set. **Physica A: Statistical Mechanics and its Applications**, v. 541, p. 123340, 2020. ISSN 0378-4371.

LI, Q.; WANG, D.; XU, M.; YANG, J. On the scalability of router forwarding tables: Nexthop-selectable fib aggregation. In: **2011 Proceedings IEEE INFOCOM**. [*S. l.*: *s. n.*], 2011. p. 321–325.

LIM, H.; SEO, J.-H.; JUNG, Y.-J. High speed ip address lookup architecture using hashing. **IEEE Communications Letters**, v. 7, n. 10, p. 502–504, 2003.

LIU, Y.; LEHMAN, V.; WANG, L. Efficient fib caching using minimal non-overlapping prefixes. **Computer Networks**, v. 83, p. 85–99, 2015. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128615000894. Acesso em: 26 dez. 2024.

LIU, Y.; ZHANG, B.; WANG, L. Fifa: Fast incremental fib aggregation. In: **2013 Proceedings IEEE INFOCOM**. [*S. l.*: *s. n.*], 2013. p. 1–9.

LUO, S.; ZHONG, S.; LEI, K. Ip/ndn: A multi-level translation and migration mechanism. In: **NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium**. [*S. l.*: *s. n.*], 2018. p. 1–5.

LUOMA, M. **Classless Inter Domain Routing-CIDR**. [*S. l.*]: Citeseer, 2019.

MALGOSA, J.; FLORES, A.; MUNOZ, J. P.; MANZANARES, P. Solving ip exhaustion with maskless inter-domain routing (midr) scheme. In: **2017 Sixth International Conference on Future Generation Communication Technologies (FGCT)**. [*S. l.*: *s. n.*], 2017. p. 1–5.

MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, JSTOR, p. 50–60, 1947.

MCMAHON, A. **Discovery of delay-tolerant networking endpoint elements**. Tese (Doutorado) – Trinity College (Dublin, Ireland). School of Computer Science & Statistics, 2013.

MEDEIROS, D. de F. Implementação e análise de protocolos de roteamento para redes mesh sem fio lora. p. 123f, 2021.

MORRISON, D. R. Patricia—practical algorithm to retrieve information coded in alphanumeric. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 4, p. 514–534, oct 1968. ISSN 0004-5411.

MORRISON, D. R. Patricia—practical algorithm to retrieve information coded in alphanumeric. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 4, p. 514–534, out. 1968. ISSN 0004-5411.

MOTTAGHI, M. H.; SEDIGHI, M.; ZAMANI, M. S. Fifa: A fully invertible fpga architecture to reduce bti-induced aging effects. **IEEE Transactions on Computers**, v. 71, n. 9, p. 2277–2286, 2022.

NAGAO, H.; SHUDO, K. Flexible routing tables: Designing routing algorithms for overlays based on a total order on a routing table set. In: **2011 IEEE International Conference on Peer-to-Peer Computing**. [*S. l.*: *s. n.*], 2011. p. 72–81.

NAJIYYA, H.; SHOFI, M. S.; ANGGRAENY, A.; LEANNA, V. Y.; ISTIKMAL; TODY, A. W. Ip-to-ndn integration with socket translation gateway. In: **2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE)**. [*S. l.*: *s. n.*], 2023. p. 155–158.

NICOLETTI, M. d. C. **Fundamentos da Teoria dos Grafos para Computação.** [*S. l.*]: Grupo GEN, 2017, 2017.

ONDREJ, F.; MARTIN, M.; ONDREJ, Z. **The BIRD Internet Routing Daemon Project**. 2012. Disponível em: https://bird.network.cz/. Acesso em: 26 dez. 2024.

OREGON, U. of. **Route Views Project**. n.d. Disponível em: https://www.routeviews.org/routeviews/index.php/archive/. Acesso em: 21 nov. 2024.

PACHECO, C. **UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ DEPARTAMENTO ACADÊMICO DE ELETRÔNICA ESPECIALIZAÇÃO EM TELEINFORMATICA E REDES DE COMPUTADORES A IMPORTÂNCIA DO IPV6 E UMA PROPOSTA DE ABORDAGEM PARA MIGRAÇÃO MONOGRAFIA DE ESPECIALIZAÇÃO**. 2014.

PAHLEVANI, P.; KHAMFROUSH, H.; LUCANI, D. E.; PEDERSEN, M. V.; FITZEK, F. H. Network coding for hop-by-hop communication enhancement in multi-hop networks. **Computer Networks**, v. 105, p. 138–149, 2016. ISSN 1389-1286.

PELEG, D.; SCHAFFER, A. **Graph Spanners**. IBM Thomas J. Watson Research Division, 1988. Disponível em: https://books.google.com.br/books?id=pk4SrgEACAAJ. Acesso em: 26 dez. 2024.

PELEG, D.; ULLMAN, J. D. **Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing.** [*S. n.*], 1987. 740–747 p. ISBN 089791239X. Disponível em: https://dl.acm.org/doi/pdf/10.1145/41840.41847. Acesso em: 26 dez. 2024.

PELEG, D.; UPFAL, E. **A Trade-Off between Space and Efficiency for Routing Tables**. 1989.

PRABA, T. S.; SETHUKARASI, T.; SARAVANAN, S. Energy measure semigraph-based connected edge domination routing algorithm in wireless sensor networks. **Mobile Information Systems**, Wiley Online Library, v. 2019, n. 1, p. 4761304, 2019.

PROJECT., R. **Route Views – University of Oregon Route Views Project**. 2023. Disponível em: https://www.routeviews.org/routeviews/. Acesso em: 26 dez. 2024.

PROJECT, S. I. IPMA. Disponível em: https://www.merit.edu/. Acesso em: 26 dez. 2024.

REFAEI, T.; MA, J.; HA, S.; LIU, S. Integrating ip and ndn through an extensible ip-ndn gateway. In: **Proceedings of the 4th ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2017. (ICN '17), p. 224–225. ISBN 9781450351225.

RODITTY, L.; TOV, R. New routing techniques and their applications. In: **Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing**. New York, NY, USA: Association for Computing Machinery, 2015. (PODC '15), p. 23–32. ISBN 9781450336178. Disponível em: https://doi.org/10.1145/2767386.2767409. Acesso em: 26 dez. 2024.

SABAI, F. J. **Uma Abordagem Intra-AS para Diminuir o Tamanho da Tabela de Encaminhamento de Roteadores da Internet**. 2017.

SANTORO, N.; KHATIB., R. Labelling and implicit routing in networks. p. 5–8, 1985. Disponível em: https://academic.oup.com/comjnl/article/28/1/5/467936. Acesso em: 26 dez. 2024.

SAXENA, D.; RAYCHOUDHURY, V. N-fib: Scalable, memory efficient name-based forwarding. **Journal of Network and Computer Applications**, v. 76, p. 101–109, 2016. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804516302132. Acesso em: 21 jun. 2024.

SAXENA, D.; RAYCHOUDHURY, V.; SURI, N.; BECKER, C.; CAO, J. Named data networking: A survey. **Computer Science Review**, v. 19, p. 15–55, 2016. ISSN 1574-0137.

SHUDO, K.; TANAKA, Y.; SEKIGUCHI, S. Overlay weaver: An overlay construction toolkit. **Computer Communications**, v. 31, n. 2, p. 402–412, 2008. ISSN 0140-3664. Special Issue: Foundation of Peer-to-Peer Computing.

SIEGEL, S.; JR, N. J. C. **Estatística não-paramétrica para ciências do comportamento**. [*S. l.*]: Artmed Editora, 2006.

SPRENT, P.; SMEETON, N. C. **Applied nonparametric statistical methods**. [*S. l.*]: CRC press, 2016.

SULTANA, S.; ASADUZZAMAN, A. A minimum spanning tree based routing protocol for multi-hop and multi-channel cognitive radio. In: **2018 Joint 7th International Conference on Informatics, Electronics Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision Pattern Recognition (icIVPR)**. [*S. l.*: *s. n.*], 2018. p. 206–211.

SUN, X.; YANG, Y.; MA, M. Minimum connected dominating set algorithms for ad hoc sensor networks. **Sensors (Basel, Switzerland)**, v. 19, 2019. Disponível em: https://api.semanticscholar.org/CorpusID:131777141. Acesso em: 26 dez. 2024.

THORUP, M.; ZWICK, U. **Compact routing schemes**. 2001.

VORST, N. V.; LI, T.; LIU, J. How low can you go? spherical routing for scalable network simulations. In: **2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems**. [*S. l.*: *s. n.*], 2011. p. 259–268.

WANG, X.; ZHANG, T.; HUANG, Y.; XIAO, J. A new algorithm for relative attribute reduction in decision table. In: **2006 6th World Congress on Intelligent Control and Automation**. [*S. l.*: *s. n.*], 2006. v. 1, p. 4051–4054.

WANG, Y.; BI, J.; WANG, J. Towards an aggregation-aware internet routing. In: **2012 21st International Conference on Computer Communications and Networks (ICCCN)**. [*S. l.*: *s. n.*], 2012. p. 1–7.

WANG, Y.; BI, J.; WU, J. Aidr: Aggregation of bgp routing table with as path stretch. In: **2011 19th IEEE International Conference on Network Protocols**. [*S. l.*: *s. n.*], 2011. p. 137–138.

WANG, Y.; DAI, H.; JIANG, J.; HE, K.; MENG, W.; LIU, B. Parallel name lookup for named data networking. In: **2011 IEEE Global Telecommunications Conference - GLOBECOM 2011**. [*S. l.*: *s. n.*], 2011. p. 1–5.

WEIWEI, J. Research and practice of bilingual education of c language programming. In: **2011 IEEE 3rd International Conference on Communication Software and Networks**. [*S. l.*: *s. n.*], 2011. p. 129–131.

WESTCOTT, J.; LAUER, G. Hierarchical routing for very large networks. In: **MILCOM 1984 - IEEE Military Communications Conference**. [*S. l.*: *s. n.*], 1984. v. 2, p. 214–218.

WILCOX, R. R.; KESELMAN, H.; KOWALCHUK, R. K. Can tests for treatment group equality be improved?: The bootstrap and trimmed means conjecture. **British Journal of Mathematical and Statistical Psychology**, Wiley Online Library, v. 51, n. 1, p. 123–134, 1998.

XIAN, X.; SHI, W.; HUANG, H. Comparison of omnet++ and other simulator for wsn simulation. In: **2008 3rd IEEE Conference on Industrial Electronics and Applications**. [*S. l.*: *s. n.*], 2008. p. 1439–1443.

XUE, F.; HE, C.-L.; SUN, Z.-S.; YU, X. Key node identification method of chengdu metro network based on comprehensive assessment. In: SPRINGER. **Advances in Smart Vehicular Technology, Transportation, Communication and Applications: Proceeding of the Second International Conference on Smart Vehicular Technology, Transportation, Communication and Applications, October 25-28, 2018 Mount Emei, China, Part 1 2**. [*S. l.*], 2019. p. 48–58.

YADAV, M.; RISHIWAL, V.; ARORA, G.; MAKKA, S. Modified minimum connected dominating set formation for wireless adhoc networks. **arXiv preprint arXiv:0912.4323**, 2009.

YU, W.; PAO, D. Hardware accelerator for fib lookup in named data networking. **Microprocessors and Microsystems**, v. 71, p. 102877, 2019. ISSN 0141-9331.

ZHANG, D. E. L.; BURKE, J. "named data networking (ndn) project". In: **Named Data Networking Technical Report NDN-0001**. [*S. l.*: *s. n.*], 2010.

ZHANG, L.; AFANASYEV, A.; BURKE, J.; JACOBSON, V.; CLAFFY, k.; CROWLEY, P.; PAPADOPOULOS, C.; WANG, L.; ZHANG, B. Named data networking. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 66–73, jul. 2014. ISSN 0146-4833.

APÊNDICE A – RELATÓRIO DE PROCESSAMENTO DE DADOS DE SAXENA E RAYCHOUDHURY (2016) E CHATTERJEE ET AL. (2021) - IBM SPSS STATISTICS

Explorar

Observações

Saída criada		16-NOV-2024 14:12:46
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60
Tratamento de valores omissos	Definição de omisso	Os valores omissos definidos pelo usuário para variáveis dependentes são tratados como omissos.
	Casos utilizados	As estatísticas são baseadas em casos sem valores omissos para qualquer variável dependente ou fator usado.
Sintaxe		EXAMINE VARIABLES=Memory_usage Exec_time Aggregation_time BY Grupo /PLOT BOXPLOT STEMLEAF HISTOGRAM NPPLOT /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Recursos	Tempo do processador	00:00:01,51
	Tempo decorrido	00:00:02,45

Definição da técnica

Resumo de processamento de casos

		Casos			
		Válido		Omisso	
	Definição da técnica	N	Porcentagem	N	
Tempo de Execução (em microssegundos)	Saxena	30	100,0%	0	
	Chatterjee	30	100,0%	0	
Uso de Memória (em MB)	Saxena	30	100,0%	0	
	Chatterjee	30	100,0%	0	
Tempo de Agregação (em segundos)	Saxena	30	100,0%	0	
	Chatterjee	30	100,0%	0	

Resumo de processamento de casos

		Casos			
		Omisso		Total	
	Definição da técnica	Porcentagem		N	Porcentagem
Tempo de Execução (em microssegundos)	Saxena	0,0%		30	100,0%
	Chatterjee	0,0%		30	100,0%
Uso de Memória (em MB)	Saxena	0,0%		30	100,0%
	Chatterjee	0,0%		30	100,0%
Tempo de Agregação (em segundos)	Saxena	0,0%		30	100,0%
	Chatterjee	0,0%		30	100,0%

Descritivas

	Definição da técnica			Estatística
Tempo de Execução (em microssegundos)	Saxena	Média		24,809525193
		95% de Intervalo de Confiança para Média	Limite inferior	20,859328444
			Limite superior	28,759721942
		5% da média aparada		24,375047147
		Mediana		23,744050000
		Variância		111,911
		Erro Padrão		10,578821182
		Mínimo		11,7641470
		Máximo		45,5648785
		Amplitude		33,8007315
		Amplitude interquartil		13,8077339
		Assimetria		,735
		Curtose		-,500
	Chatterjee	Média		1,668953733
		95% de Intervalo de Confiança para Média	Limite inferior	1,594804214
			Limite superior	1,743103253
		5% da média aparada		1,655690519
		Mediana		1,614705000
		Variância		,039
		Erro Padrão		,1985760605

Descritivas

	Definição da técnica			Estatística do teste Padrão
Tempo de Execução (em microssegundos)	Saxena	Média		1,9314196644
		95% de Intervalo de Confiança para Média	Limite inferior	
			Limite superior	
		5% da média aparada		
		Mediana		
		Variância		
		Erro Padrão		
		Mínimo		
		Máximo		
		Amplitude		
		Amplitude interquartil		
		Assimetria		,427
		Curtose		,833
	Chatterjee	Média		,0362548626
		95% de Intervalo de Confiança para Média	Limite inferior	
			Limite superior	
		5% da média aparada		
		Mediana		
		Variância		
		Erro Padrão		

Descritivas

	Definição da técnica			Estatística
		Mínimo		1,4152550
		Máximo		2,2020210
		Amplitude		,7867660
		Amplitude interquartil		,1983645
		Assimetria		1,094
		Curtose		,828
Uso de Memória (em MB)	Saxena	Média		278317,55165
		95% de Intervalo de Confiança para Média	Limite inferior	278309,38256
			Limite superior	278325,72073
		5% da média aparada		278318,07572
		Mediana		278336,13570
		Variância		478,613
		Erro Padrão		21,87721517
		Mínimo		278289,38570
		Máximo		278336,28420
		Amplitude		46,89850
		Amplitude interquartil		43,64850
		Assimetria		-,329
		Curtose		-1,985
	Chatterjee	Média		124553,20233
		95% de Intervalo de Confiança para Média	Limite inferior	124552,81089
			Limite superior	124553,59378
		5% da média aparada		124553,01037
		Mediana		124552,87000
		Variância		1,099
		Erro Padrão		1,04830794
		Mínimo		124552,87000
		Máximo		124557,15000
		Amplitude		4,28000
		Amplitude interquartil		,00000
		Assimetria		3,332
		Curtose		10,351
Tempo de Agregação (em segundos)	Saxena	Média		24,808639026
		95% de Intervalo de Confiança para Média	Limite inferior	20,858509044
			Limite superior	28,758769008
		5% da média aparada		24,374139332
		Mediana		23,744050000
		Variância		111,908
		Erro Padrão		10,578642378
		Mínimo		11,7641470
		Máximo		45,5628045
		Amplitude		33,7986575

Descritivas

	Definição da técnica			Estatística do teste Padrão
		Mínimo		
		Máximo		
		Amplitude		
		Amplitude interquartil		
		Assimetria		,427
		Curtose		,833
Uso de Memória (em MB)	Saxena	Média		3,99421475
		95% de Intervalo de Confiança para Média	Limite inferior	
			Limite superior	
		5% da média aparada		
		Mediana		
		Variância		
		Erro Padrão		
		Mínimo		
		Máximo		
		Amplitude		
		Amplitude interquartil		
		Assimetria		,427
		Curtose		,833
	Chatterjee	Média		,19139397
		95% de Intervalo de Confiança para Média	Limite inferior	
			Limite superior	
		5% da média aparada		
		Mediana		
		Variância		
		Erro Padrão		
		Mínimo		
		Máximo		
		Amplitude		
		Amplitude interquartil		
		Assimetria		,427
		Curtose		,833
Tempo de Agregação (em segundos)	Saxena	Média		1,9313870194
		95% de Intervalo de Confiança para Média	Limite inferior	
			Limite superior	
		5% da média aparada		
		Mediana		
		Variância		
		Erro Padrão		
		Mínimo		
		Máximo		
		Amplitude		

Descritivas

Definição da técnica			Estatística
	Amplitude interquartil		13,8072207
	Assimetria		,735
	Curtose		-,500
Chatterjee	Média		1,668953733
	95% de Intervalo de Confiança para Média	Limite inferior	1,594804214
		Limite superior	1,743103253
	5% da média aparada		1,655690519
	Mediana		1,614705000
	Variância		,039
	Erro Padrão		,1985760605
	Mínimo		1,4152550
	Máximo		2,2020210
	Amplitude		,7867660
	Amplitude interquartil		,1983645
	Assimetria		1,094
	Curtose		,828

Descritivas

Definição da técnica			Estatística do teste Padrão
	Amplitude interquartil		
	Assimetria		,427
	Curtose		,833
Chatterjee	Média		,0362548626
	95% de Intervalo de Confiança para Média	Limite inferior	
		Limite superior	
	5% da média aparada		
	Mediana		
	Variância		
	Erro Padrão		
	Mínimo		
	Máximo		
	Amplitude		
	Amplitude interquartil		
	Assimetria		,427
	Curtose		,833

Testes de Normalidade

		Kolmogorov-Smirnov[a]			Shapiro-...
	Definição da técnica	Estatística	gl	Sig.	Estatística
Tempo de Execução (em microssegundos)	Saxena	,137	30	,154	,891
	Chatterjee	,165	30	,036	,899
Uso de Memória (em MB)	Saxena	,357	30	<,001	,673
	Chatterjee	,491	30	<,001	,360
Tempo de Agregação (em segundos)	Saxena	,137	30	,154	,891
	Chatterjee	,165	30	,036	,899

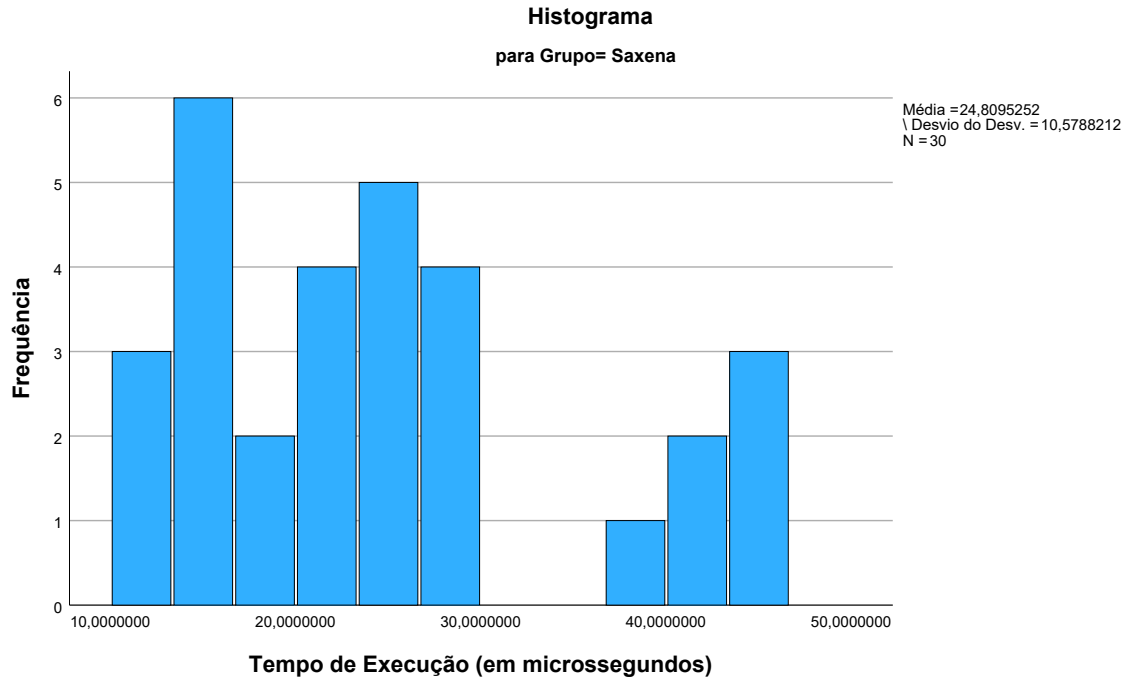
Testes de Normalidade

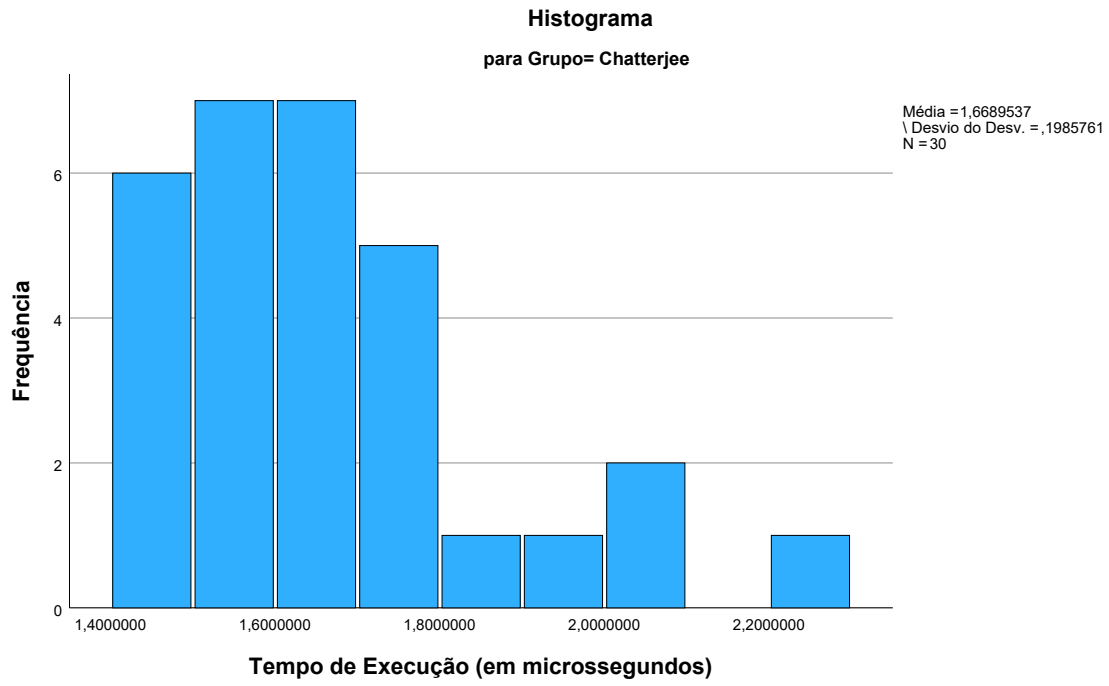
		Shapiro-Wilk	
	Definição da técnica	gl	Sig.
Tempo de Execução (em microssegundos)	Saxena	30	,005
	Chatterjee	30	,008
Uso de Memória (em MB)	Saxena	30	<,001
	Chatterjee	30	<,001
Tempo de Agregação (em segundos)	Saxena	30	,005
	Chatterjee	30	,008

a. Correlação de Significância de Lilliefors

Tempo de Execução (em microssegundos)

Histogramas





Gráficos de Ramos e Folhas

Tempo de Execução (em microssegundos) Gráfico de ramos e folhas para Grupo= Saxena

Frequência	Raiz &	Folha
8,00	1 .	12334444
3,00	1 .	577
7,00	2 .	1223444
6,00	2 .	567889
,00	3 .	
1,00	3 .	8
3,00	4 .	024
2,00	4 .	55

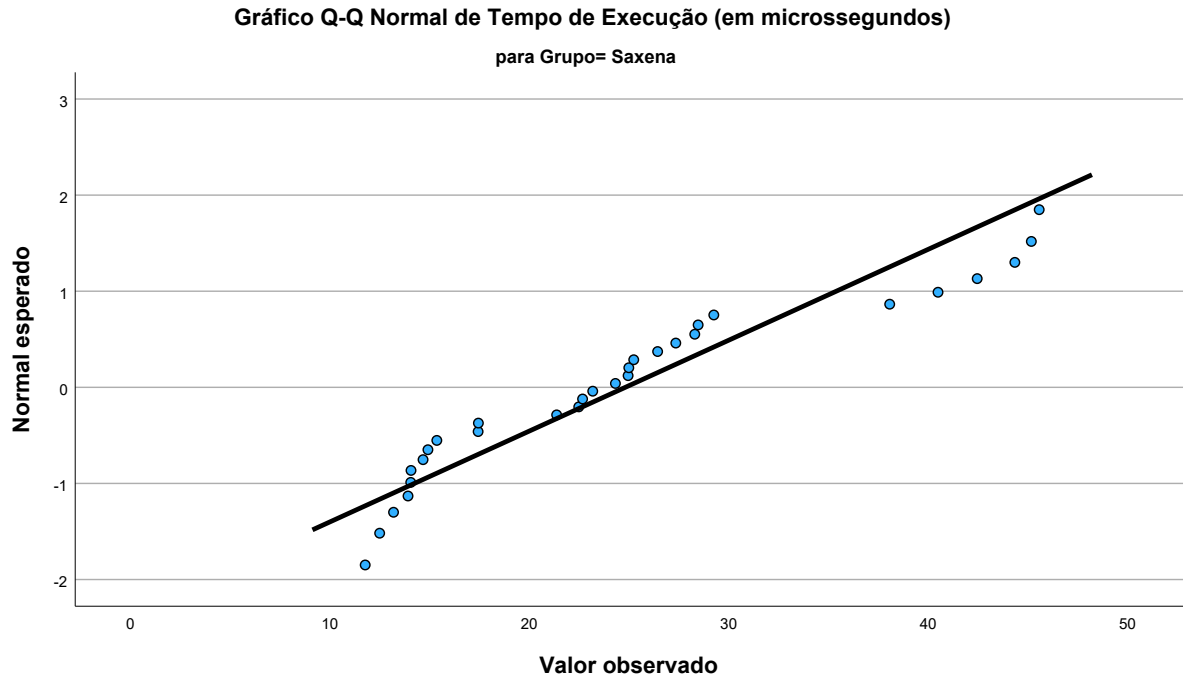
Largura do ramo: 10,00000
Cada folha: 1 caso(s)

Tempo de Execução (em microssegundos) Gráfico de ramos e folhas para Grupo= Chatterjee

Frequência	Raiz &	Folha
6,00	14 .	134466
7,00	15 .	5577778
7,00	16 .	0111569
5,00	17 .	35566
1,00	18 .	8
1,00	19 .	7

```
    3,00 Extremos    (>=2,05)

Largura do ramo:  ,1000000
Cada folha:        1 caso(s)
```

Gráfico Q-Q normais



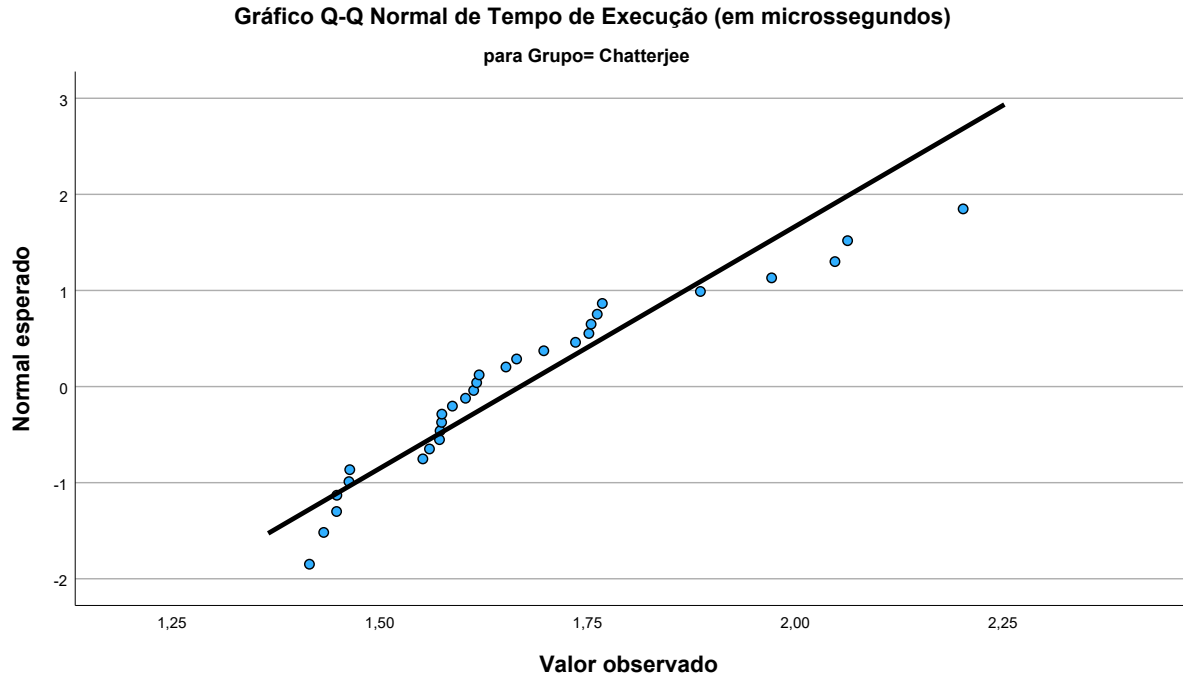


Gráfico Q-Q normais sem tendência

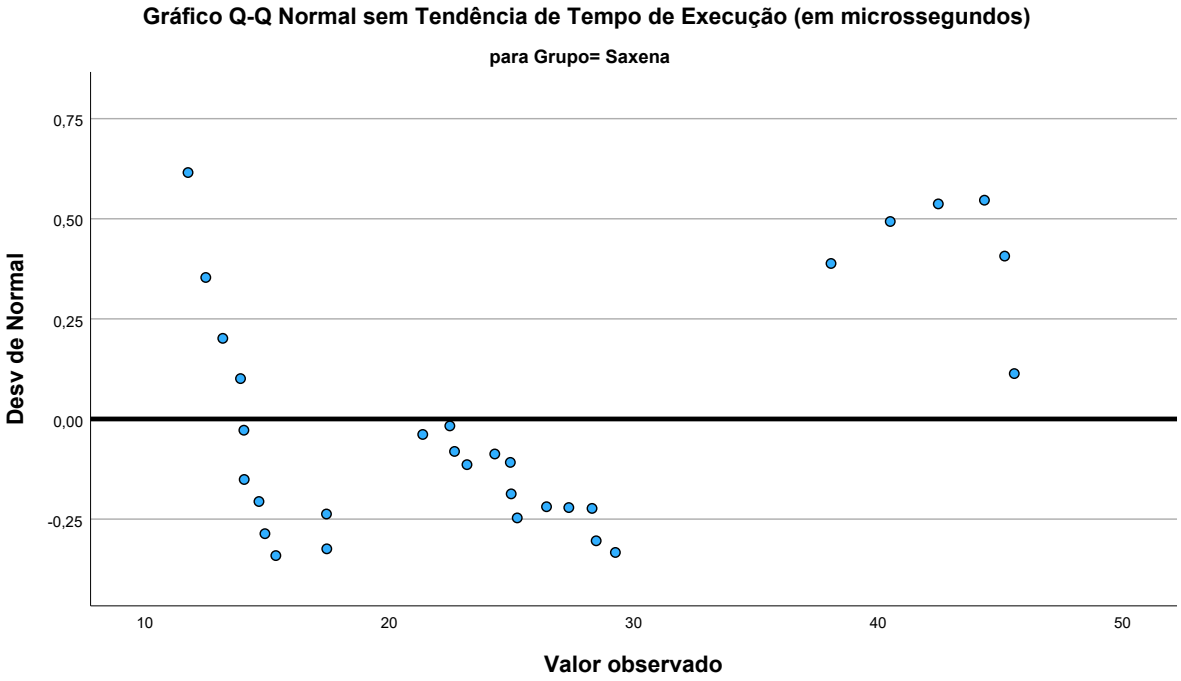


Gráfico Q-Q Normal sem Tendência de Tempo de Execução (em microssegundos)
para Grupo= Saxena
Desv de Normal
0,75
0,50
0,25
0,00
-0,25
10
20
30
40
50
Valor observado

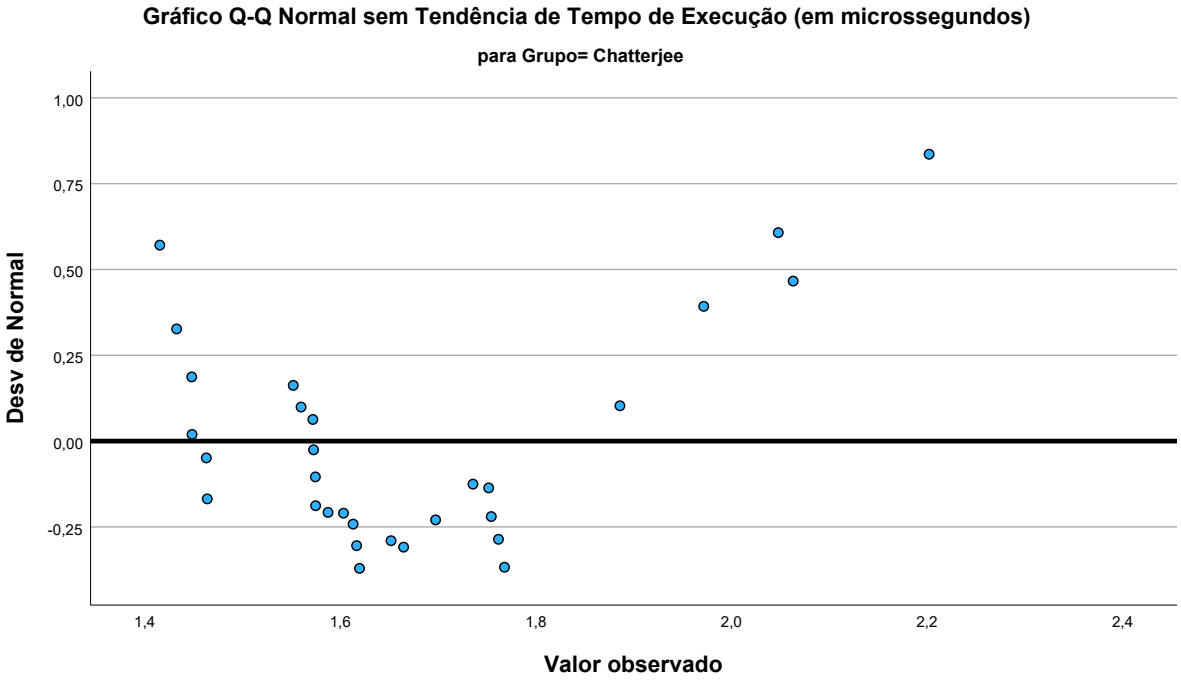
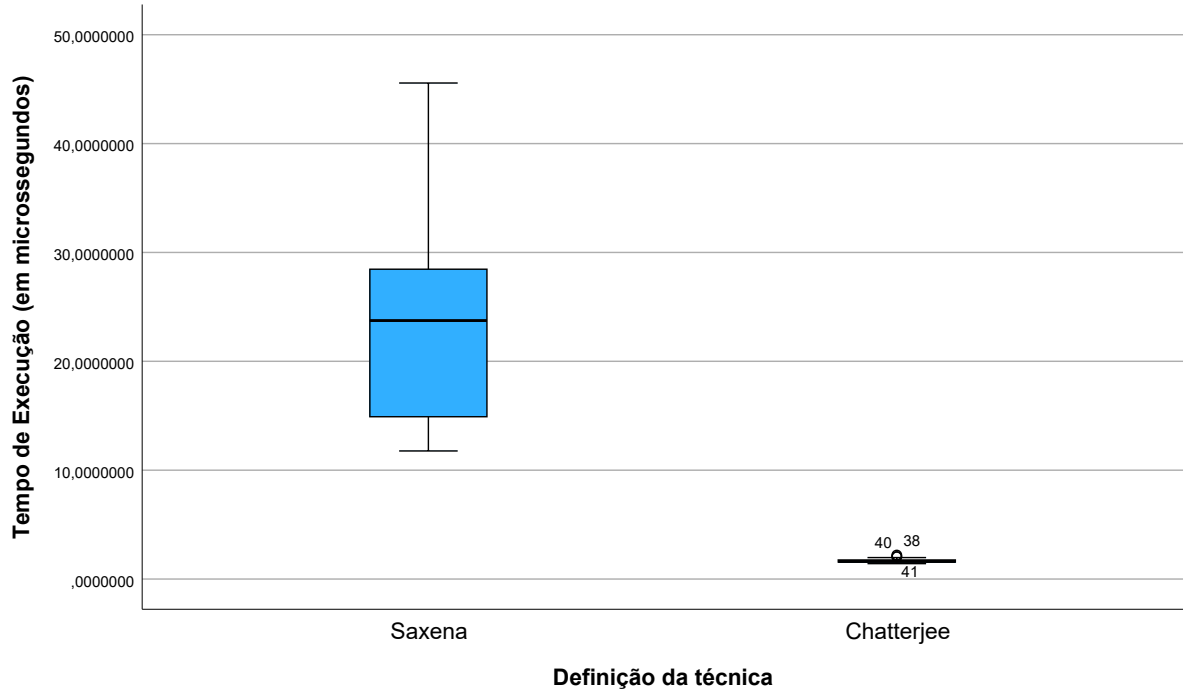
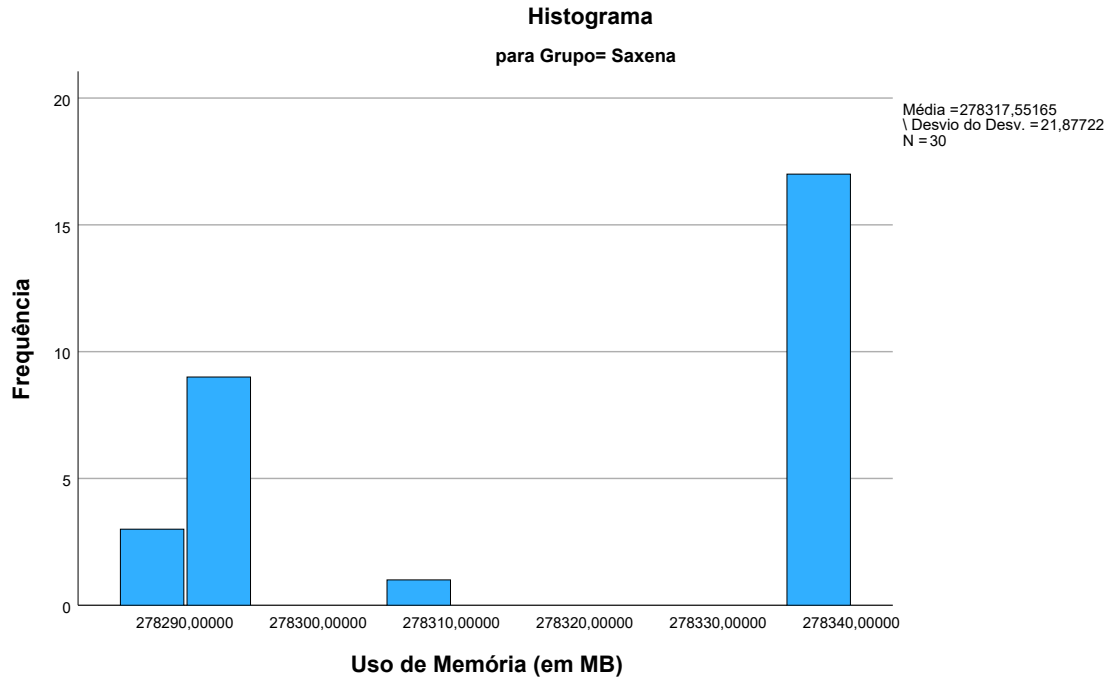


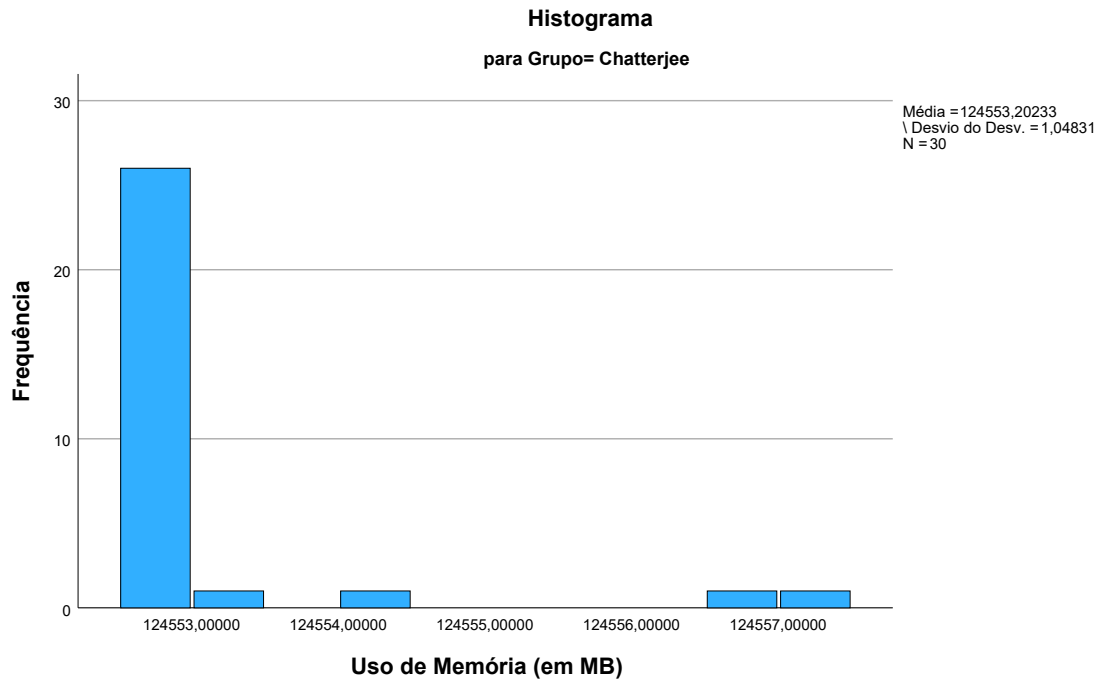
Gráfico Q-Q Normal sem Tendência de Tempo de Execução (em microssegundos)
para Grupo= Chatterjee
Desv de Normal
1,00
0,75
0,50
0,25
0,00
-0,25
1,4
1,6
1,8
2,0
2,2
2,4
Valor observado



Uso de Memória (em MB)

Histogramas

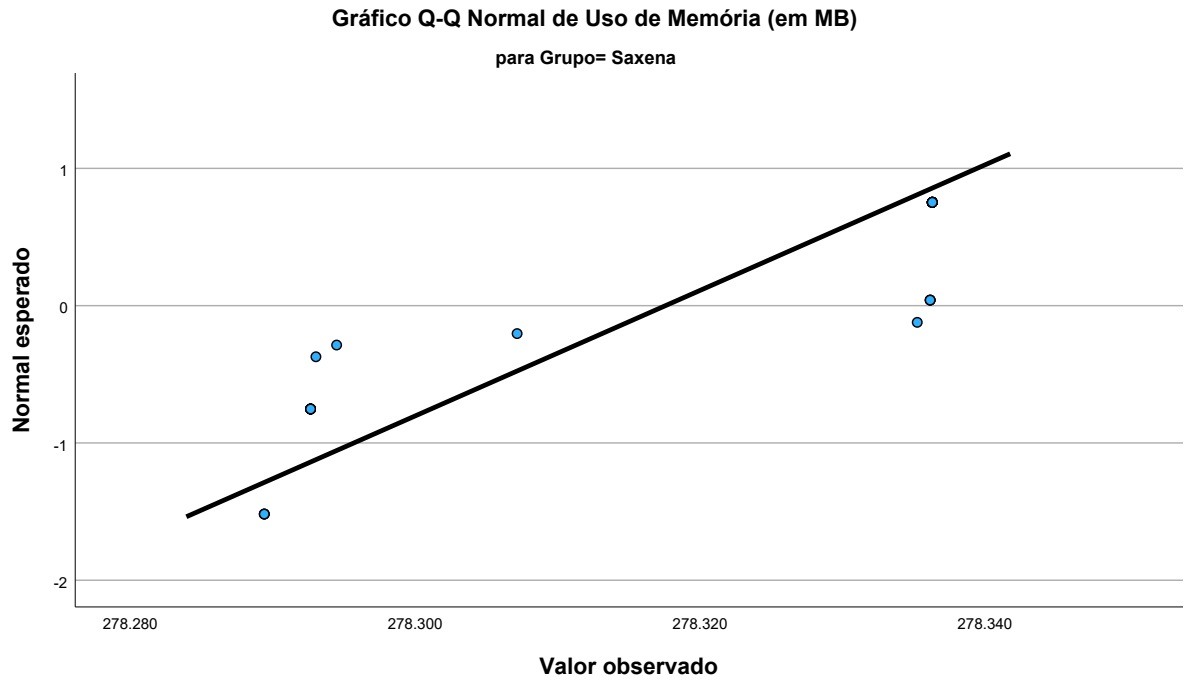




Gráficos de Ramos e Folhas

```
Uso de Memória (em MB) Gráfico de ramos e folhas para
Grupo= Saxena

 Frequência  Raiz &  Folha

     3,00    27828 .  999
     9,00    27829 .  222222234
     1,00    27830 .  7
      ,00    27831 .
      ,00    27832 .
    17,00    27833 .  56666666666666666

 Largura do ramo:  10,00000
 Cada folha:        1 caso(s)


Uso de Memória (em MB) Gráfico de ramos e folhas para
Grupo= Chatterjee

 Frequência  Raiz &  Folha

    26,00        1 .  22222222222222222222222222
     4,00 Extremos    (>=124553)

 Largura do ramo:  100000,0
 Cada folha:        1 caso(s)
```

Gráfico Q-Q normais



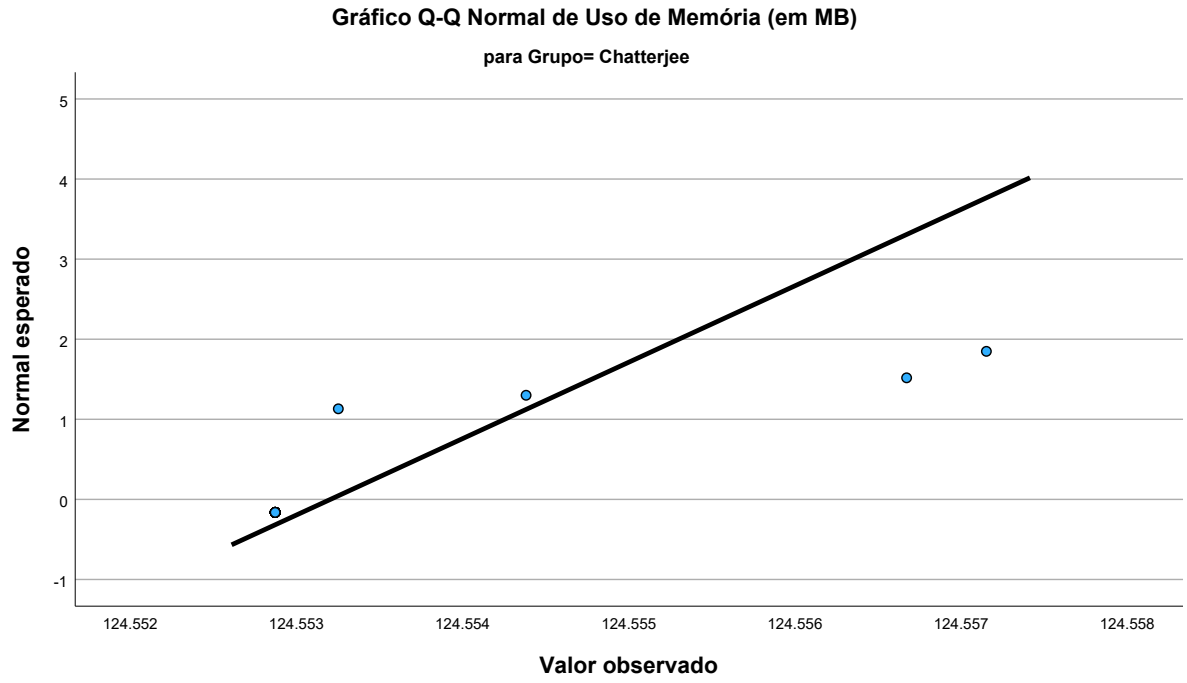


Gráfico Q-Q normais sem tendência

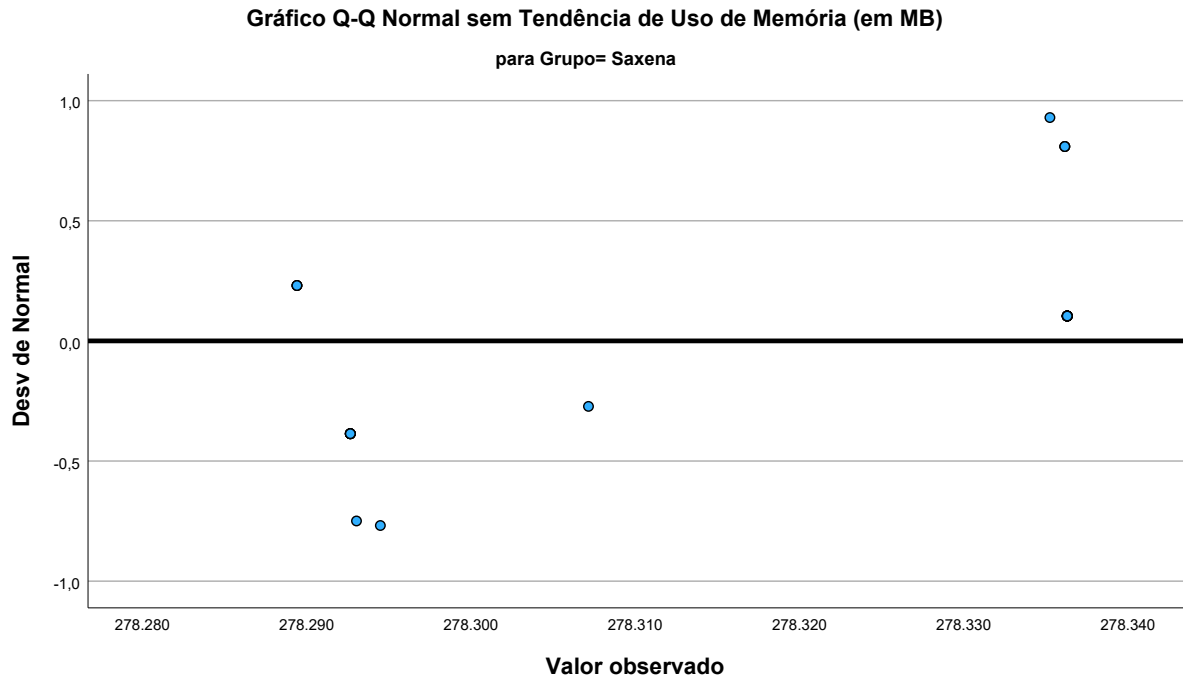


Gráfico Q-Q Normal sem Tendência de Uso de Memória (em MB)
para Grupo= Saxena
Desv de Normal
1,0
0,5
0,0
-0,5
-1,0
278.280
278.290
278.300
278.310
278.320
278.330
278.340
Valor observado

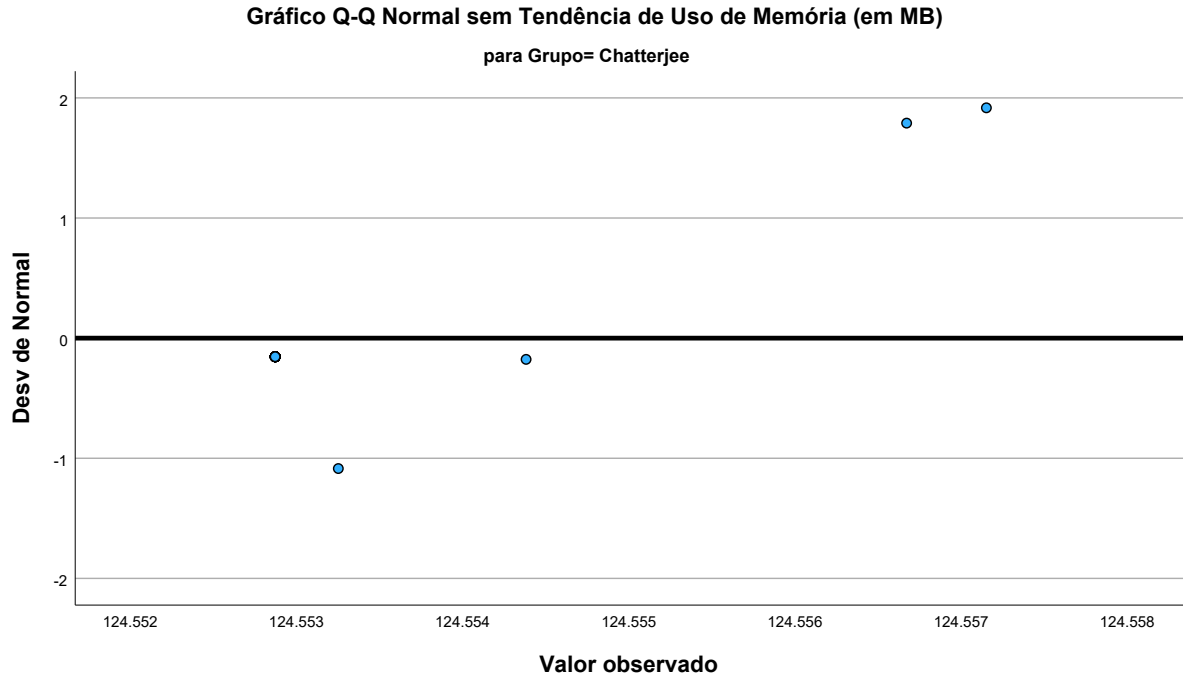
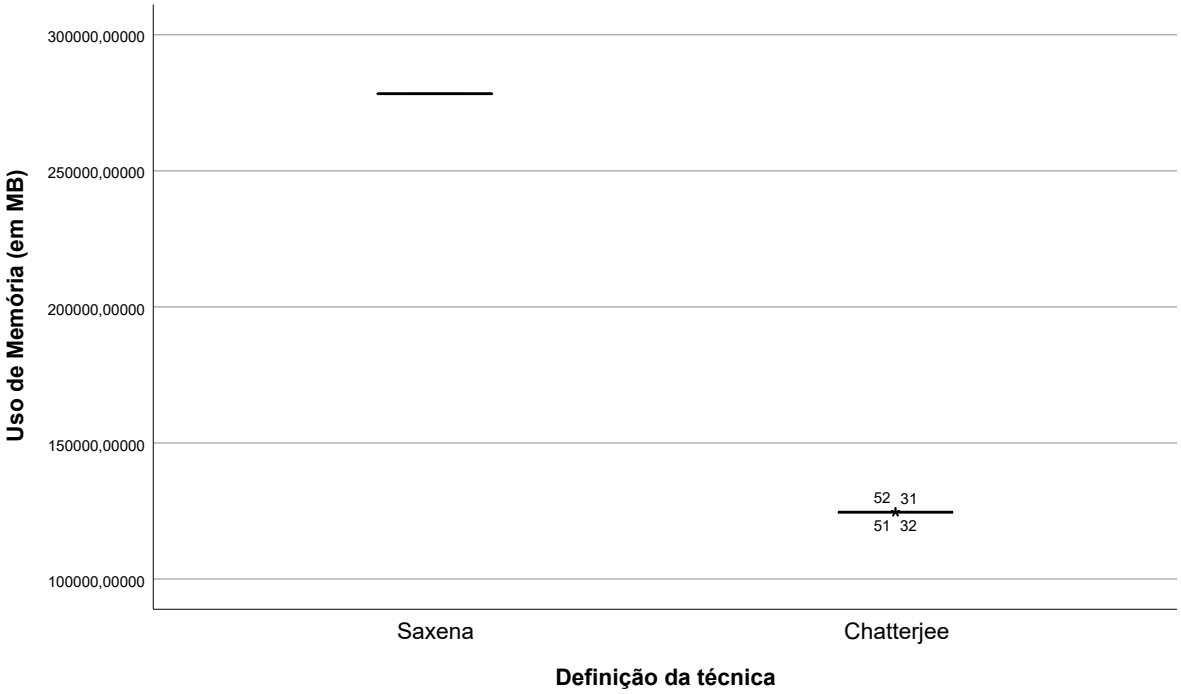
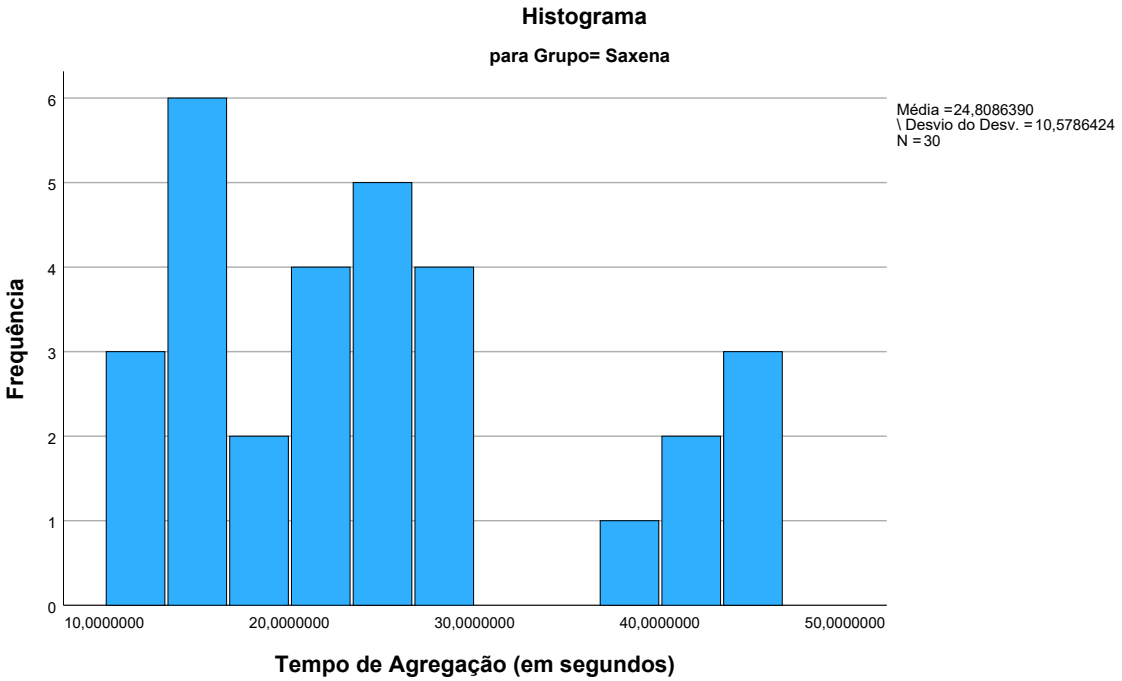


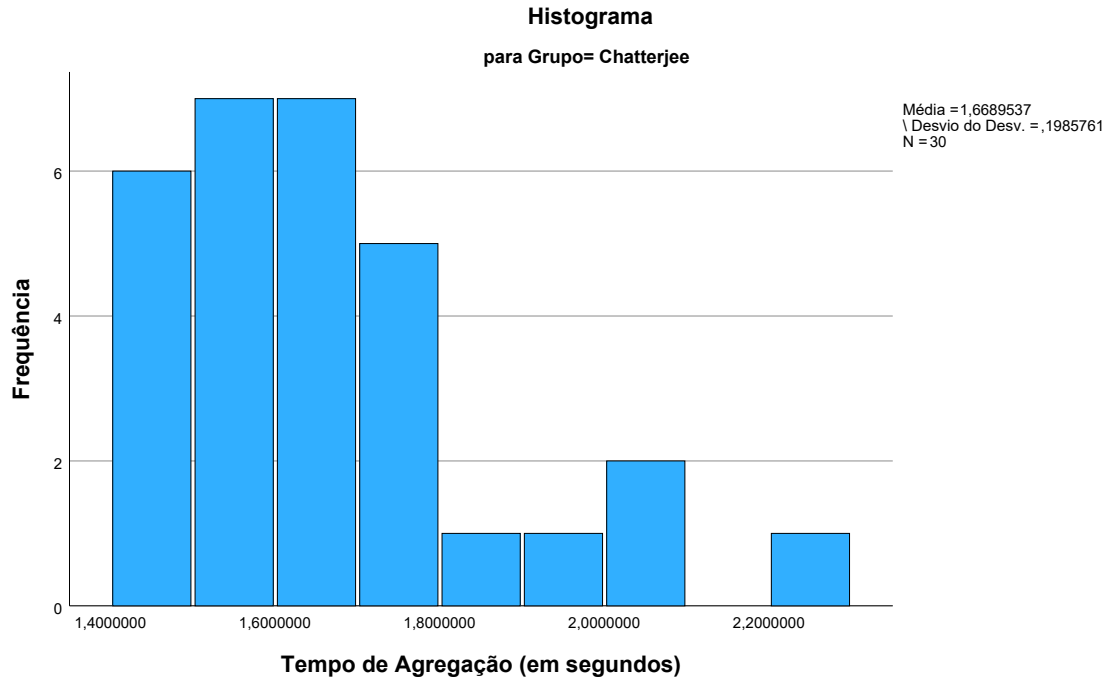
Gráfico Q-Q Normal sem Tendência de Uso de Memória (em MB)
para Grupo= Chatterjee
Desv de Normal
2
1
0
-1
-2
124.552
124.553
124.554
124.555
124.556
124.557
124.558
Valor observado



Tempo de Agregação (em segundos)

Histogramas





Gráficos de Ramos e Folhas

Tempo de Agregação (em segundos) Gráfico de ramos e folhas para Grupo= Saxena

Frequência	Raiz &	Folha
8,00	1 .	12334444
3,00	1 .	577
7,00	2 .	1223444
6,00	2 .	567889
,00	3 .	
1,00	3 .	8
3,00	4 .	024
2,00	4 .	55

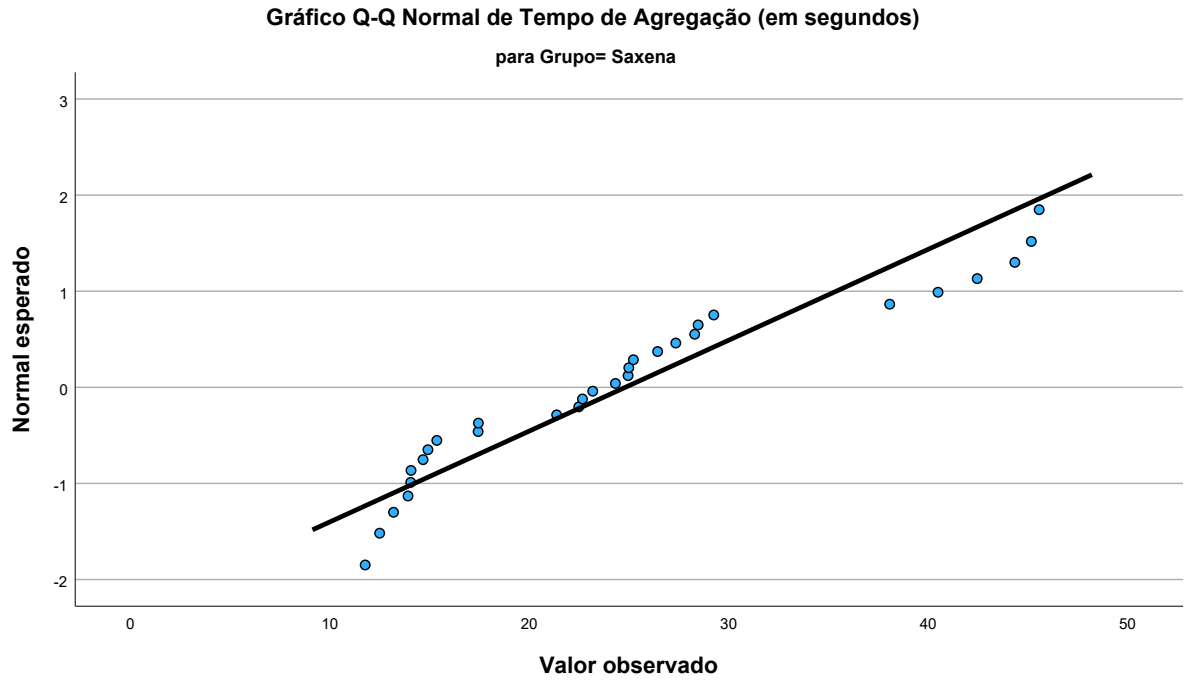
Largura do ramo: 10,00000
Cada folha: 1 caso(s)

Tempo de Agregação (em segundos) Gráfico de ramos e folhas para Grupo= Chatterjee

Frequência	Raiz &	Folha
6,00	14 .	134466
7,00	15 .	5577778
7,00	16 .	0111569
5,00	17 .	35566
1,00	18 .	8
1,00	19 .	7

```
    3,00 Extremos    (>=2,05)

Largura do ramo:  ,1000000
Cada folha:        1 caso(s)
```

Gráfico Q-Q normais



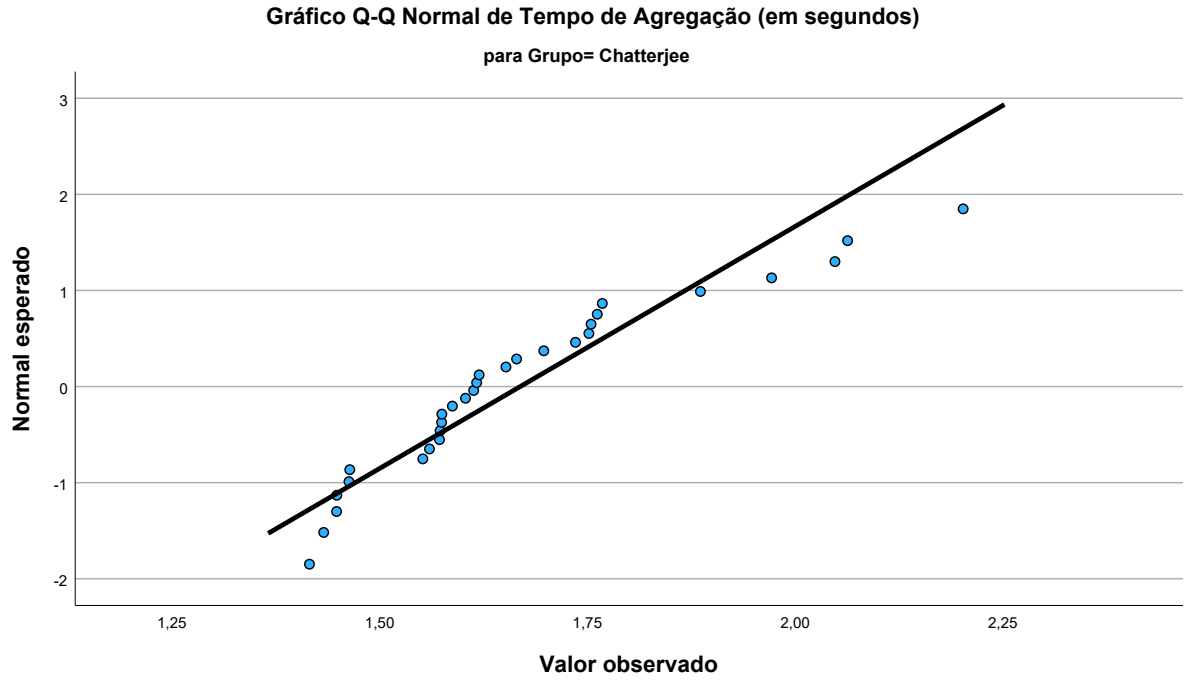


Gráfico Q-Q normais sem tendência

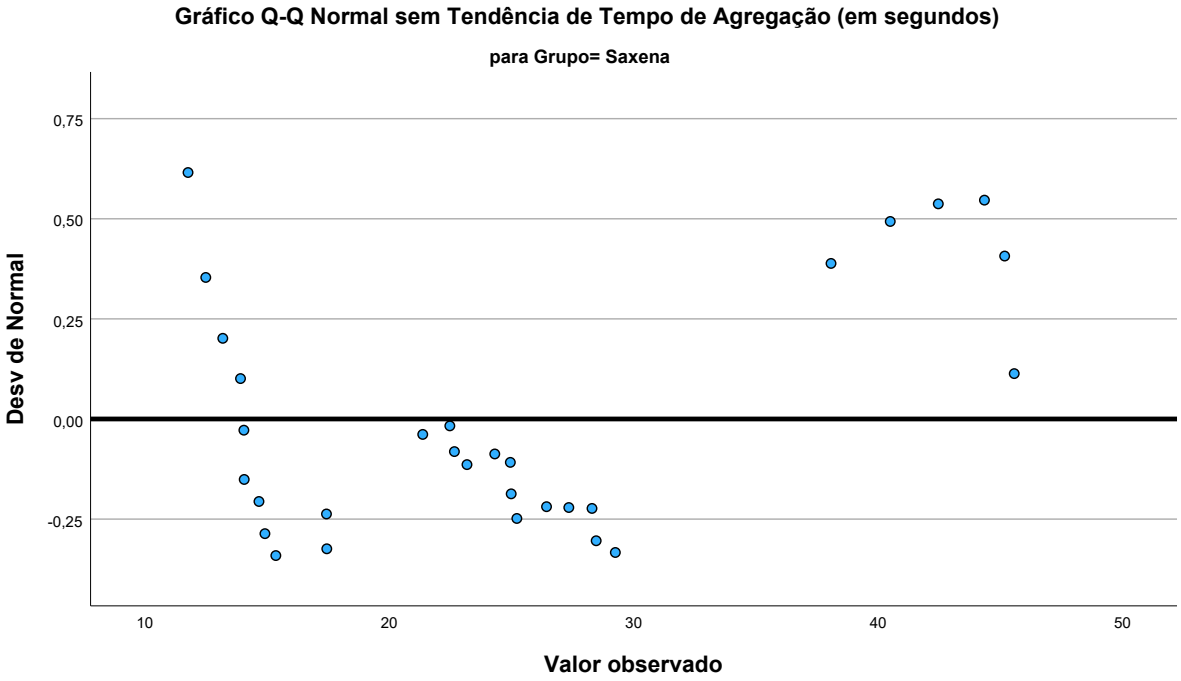


Gráfico Q-Q Normal sem Tendência de Tempo de Agregação (em segundos)
para Grupo= Saxena
Desv de Normal
0,75
0,50
0,25
0,00
-0,25
10
20
30
40
50
Valor observado

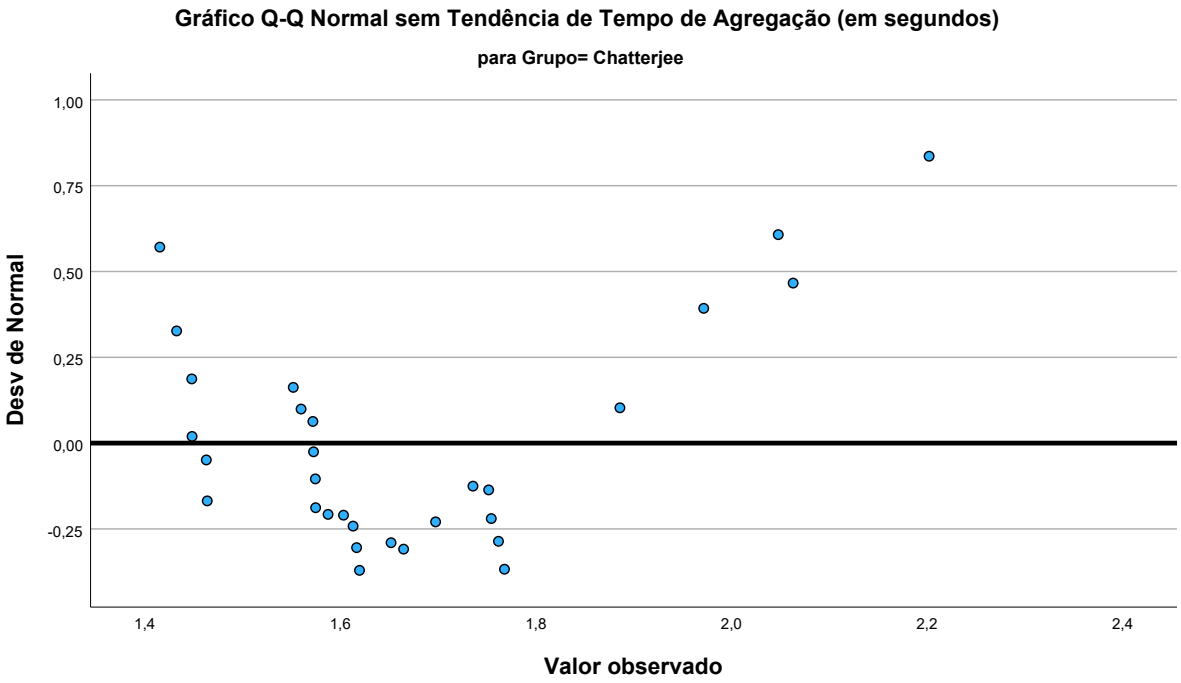
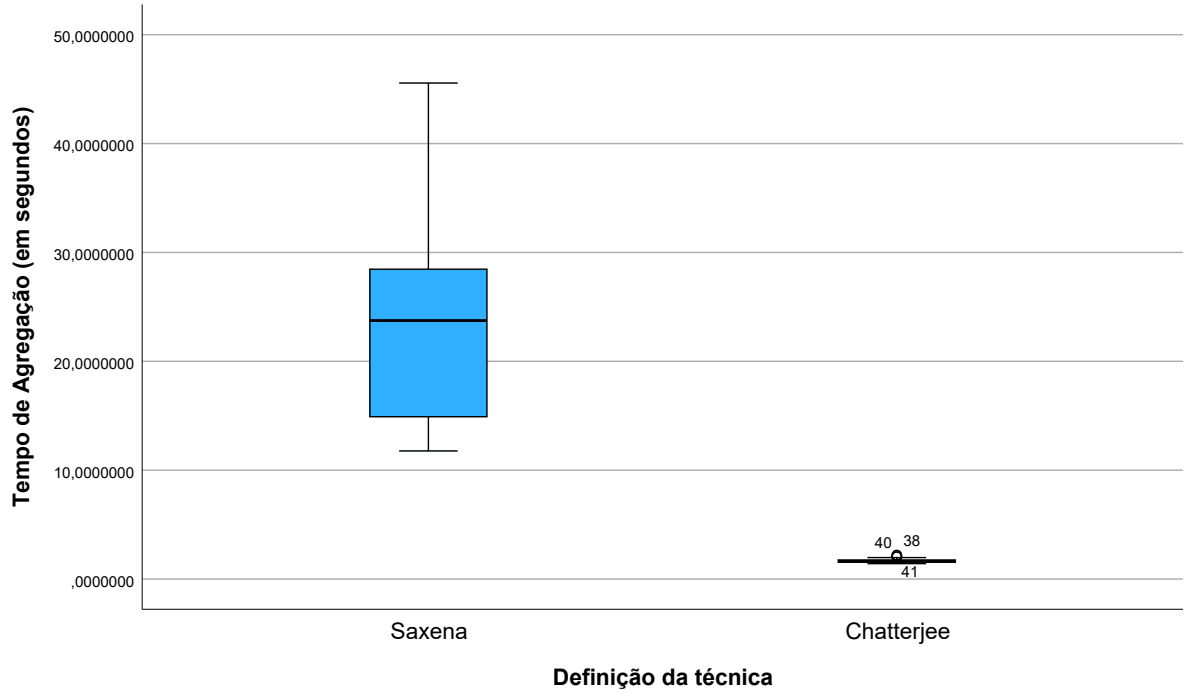


Gráfico Q-Q Normal sem Tendência de Tempo de Agregação (em segundos)
para Grupo= Chatterjee
Desv de Normal
1,00
0,75
0,50
0,25
0,00
-0,25
1,4
1,6
1,8
2,0
2,2
2,4
Valor observado



Testes não paramétricos

Observações

Saída criada		16-NOV-2024 19:55:08
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60
Sintaxe		NPTESTS /INDEPENDENT TEST (Memory_usage Exec_time Aggregation_time) GROUP (Grupo) MANN_WHITNEY /MISSING SCOPE=ANALYSIS USERMISSING=EXCLUDE /CRITERIA ALPHA=0.05 CILEVEL=95.
Recursos	Tempo do processador	00:00:00,56
	Tempo decorrido	00:00:01,50

[ConjuntodeDados1] C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav

Sumarização de Teste de Hipótese

	Hipótese nula	Teste	Sig.[a,b]
1	A distribuição de Tempo de Execução (em microssegundos) é igual nas categorias de Definição da técnica .	Amostras Independentes de Teste U de Mann-Whitney	<,001
2	A distribuição de Uso de Memória (em MB) é igual nas categorias de Definição da técnica .	Amostras Independentes de Teste U de Mann-Whitney	<,001
3	A distribuição de Tempo de Agregação (em segundos) é igual nas categorias de Definição da técnica .	Amostras Independentes de Teste U de Mann-Whitney	<,001

Sumarização de Teste de Hipótese

	Decisão
1	Rejeitar a hipótese nula.
2	Rejeitar a hipótese nula.
3	Rejeitar a hipótese nula.

a. O nível de significância é ,050.

b. A significância assintótica é exibida.

Amostras Independentes de Teste U de Mann-Whitney

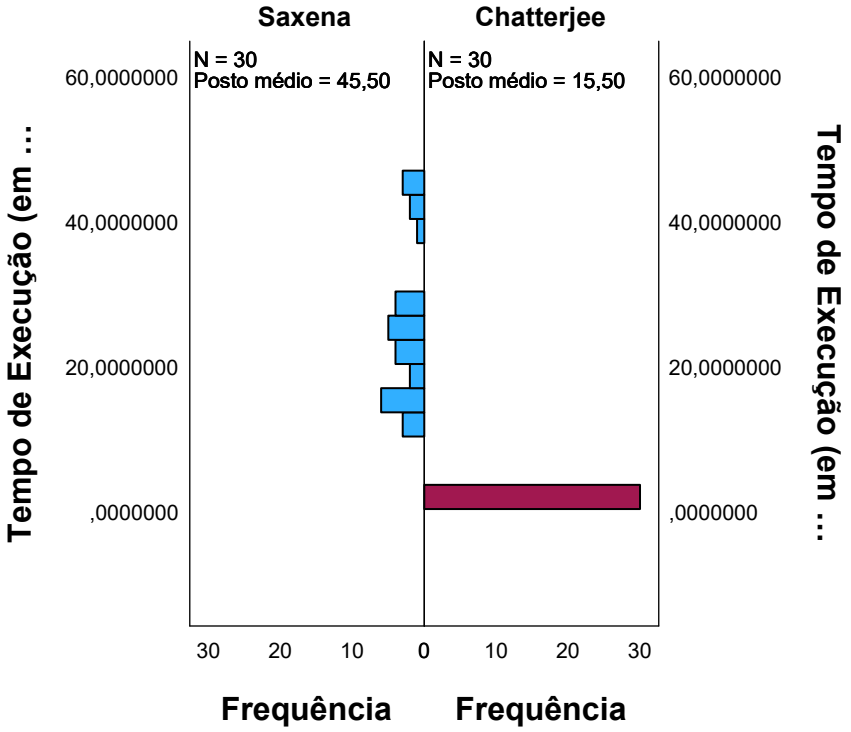
Tempo de Execução (em microssegundos) entre Definição da técnica

Amostras Independentes de Resumo de Teste U de Mann-Whitney

N total	60
U de Mann-Whitney	,000
Wilcoxon W	465,000
Estatística de teste	,000
Erro padrão	67,639
Estatística de Teste Padronizado	-6,653
Sinal assintótico (teste de dois lados)	<,001

Amostras Independentes de Teste U de Mann-Whitney

Definição da técnica



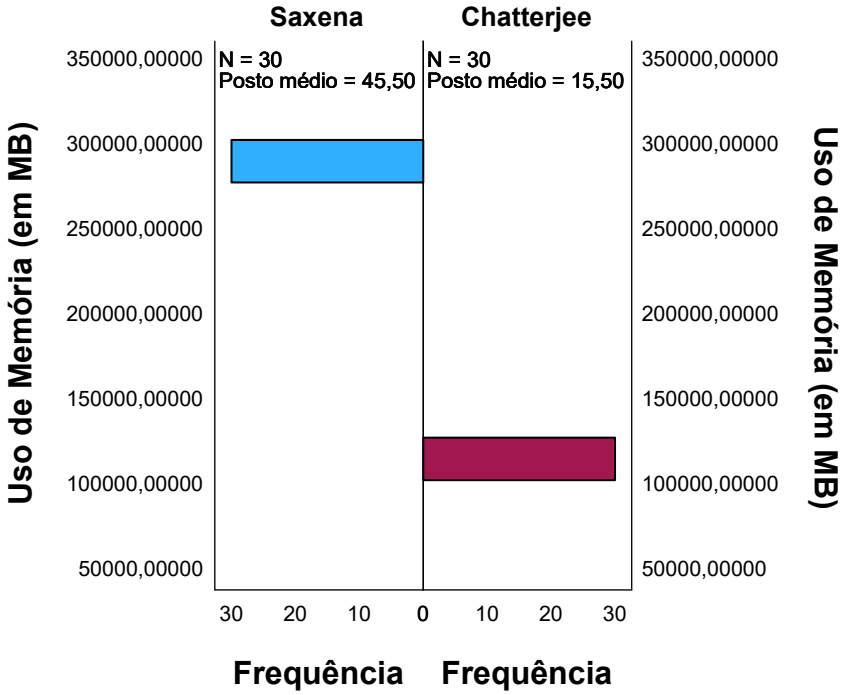
Uso de Memória (em MB) entre Definição da técnica

Amostras Independentes de Resumo de Teste U de Mann-Whitney

N total	60
U de Mann-Whitney	,000
Wilcoxon W	465,000
Estatística de teste	,000
Erro padrão	64,411
Estatística de Teste Padronizado	-6,986
Sinal assintótico (teste de dois lados)	<,001

Amostras Independentes de Teste U de Mann-Whitney

Definição da técnica

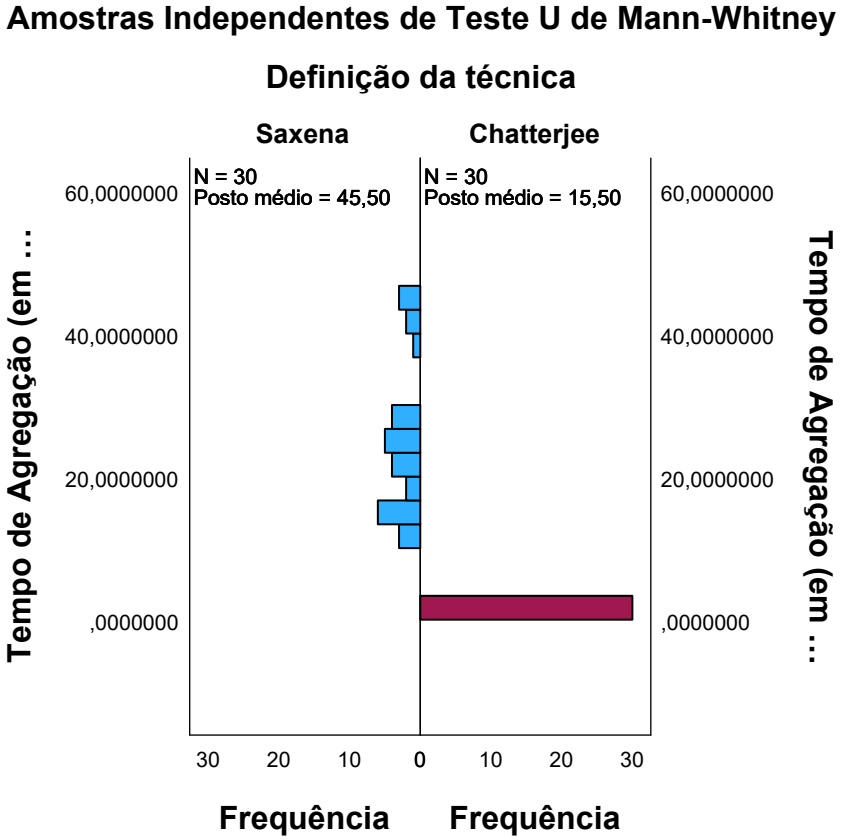


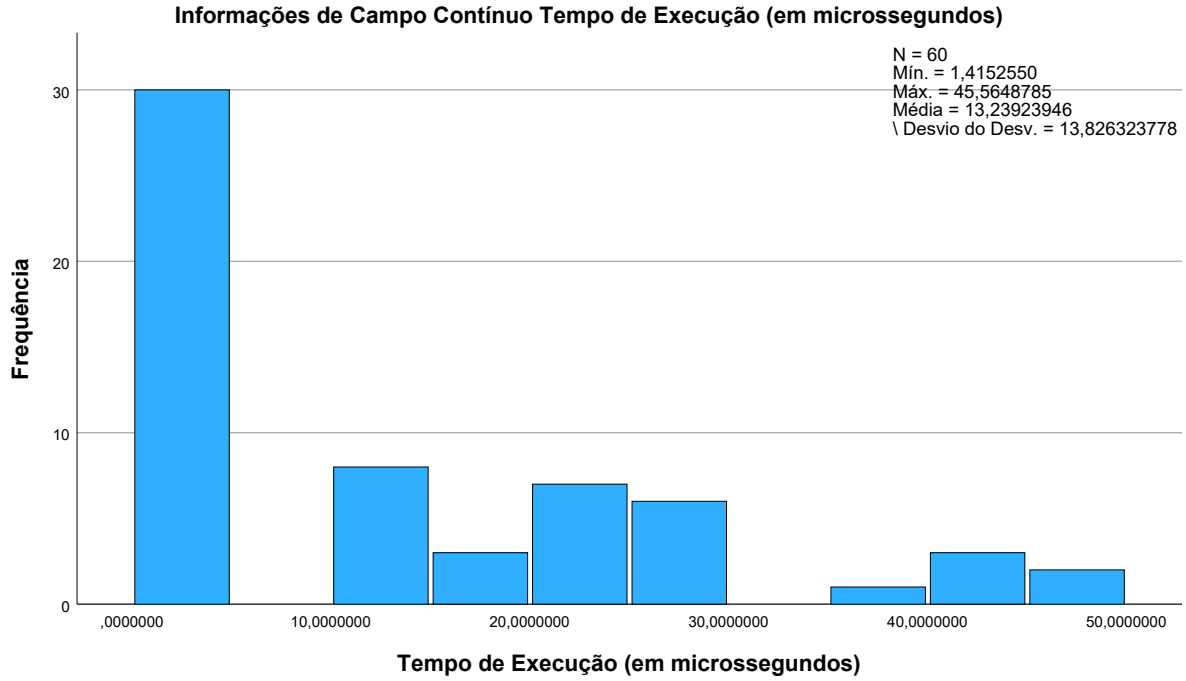
Tempo de Agregação (em segundos) entre Definição da técnica

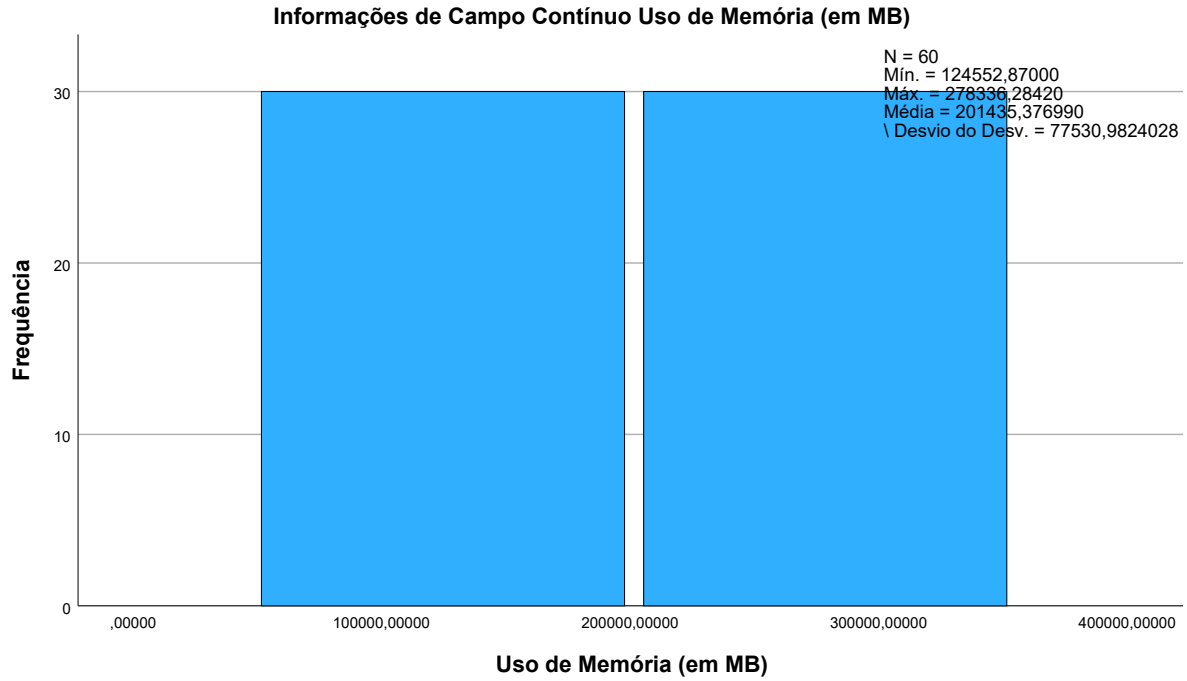
Amostras Independentes de Resumo de Teste U de Mann-Whitney

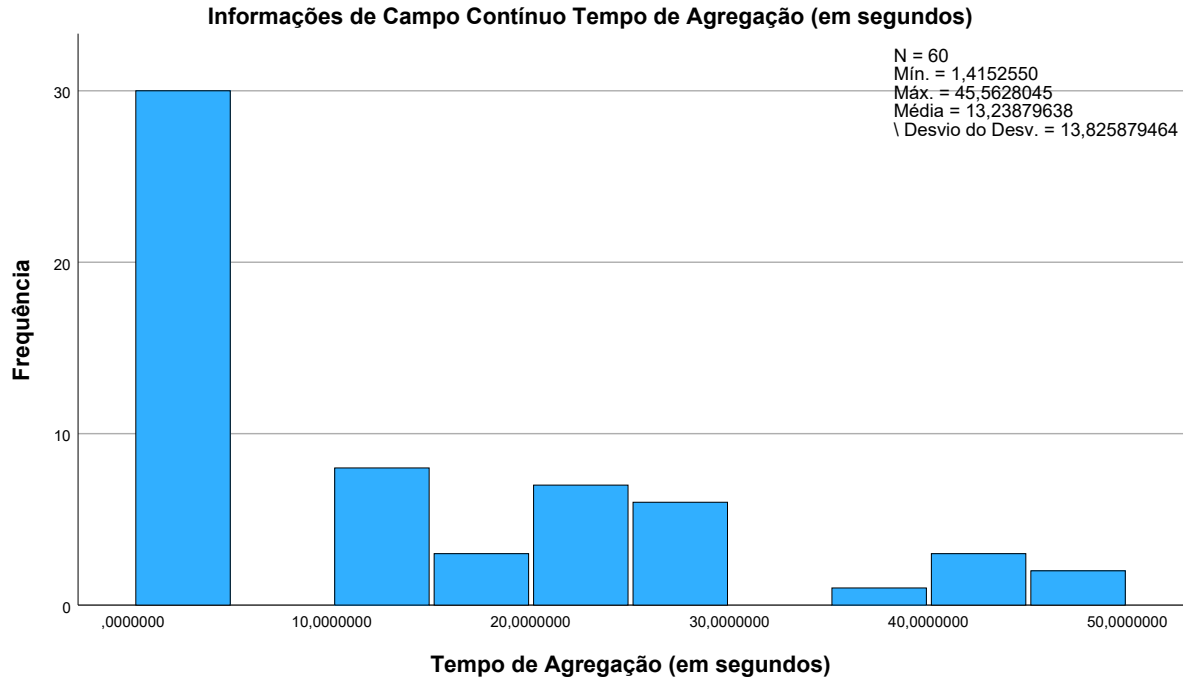
N total	60
U de Mann-Whitney	,000
Wilcoxon W	465,000
Estatística de teste	,000
Erro padrão	67,639
Estatística de Teste Padronizado	-6,653
Sinal assintótico (teste de dois lados)	<,001

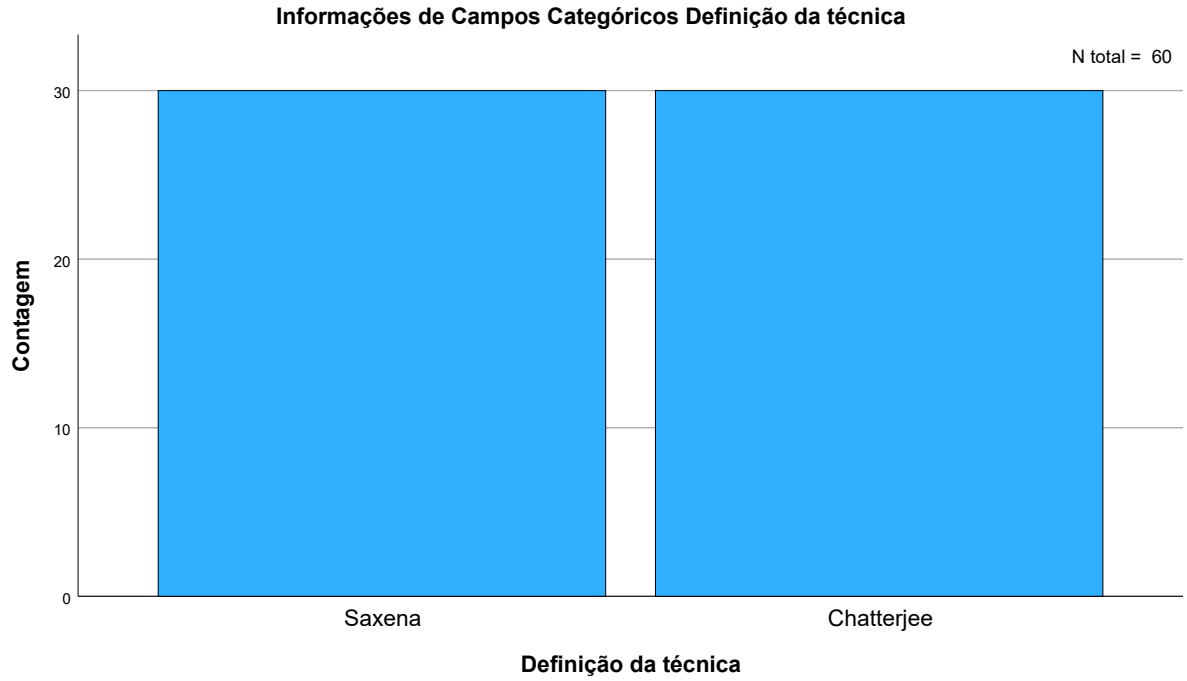
Amostras Independentes de Teste U de Mann-Whitney






Informações de Campo Contínuo Uso de Memória (em MB)
N = 60
Mín. = 124552,87000
Máx. = 278336,28420
Média = 201435,376990
\ Desvio do Desv. = 77530,9824028
Frequência
30
20
10
0
,00000
100000,00000
200000,00000
300000,00000
400000,00000
Uso de Memória (em MB)


Informações de Campo Contínuo Tempo de Agregação (em segundos)
N = 60
Mín. = 1,4152550
Máx. = 45,5628045
Média = 13,23879638
\ Desvio do Desv. = 13,825879464
Frequência
30
20
10
0
,0000000
10,0000000
20,0000000
30,0000000
40,0000000
50,0000000
Tempo de Agregação (em segundos)



Explorar

Observações

Saída criada		16-NOV-2024 21:47:36
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60
Tratamento de valores omissos	Definição de omisso	Os valores omissos definidos pelo usuário para variáveis dependentes são tratados como omissos.
	Casos utilizados	As estatísticas são baseadas em casos sem valores omissos para qualquer variável dependente ou fator usado.

Observações

Sintaxe		EXAMINE VARIABLES=Memory_usage Exec_time Aggregation_time /ID=Grupo /PLOT BOXPLOT STEMLEAF HISTOGRAM NPPLOT /COMPARE GROUPS /STATISTICS DESCRIPTIVES /CINTERVAL 95 /MISSING LISTWISE /NOTOTAL.
Recursos	Tempo do processador	00:00:02,78
	Tempo decorrido	00:00:02,34

Resumo de processamento de casos

	Casos				
	Válido		Omisso		Total
	N	Porcentagem	N	Porcentagem	N
Tempo de Execução (em microssegundos)	60	100,0%	0	0,0%	60
Uso de Memória (em MB)	60	100,0%	0	0,0%	60
Tempo de Agregação (em segundos)	60	100,0%	0	0,0%	60

Resumo de processamento de casos

	Casos
	Total
	Porcentagem
Tempo de Execução (em microssegundos)	100,0%
Uso de Memória (em MB)	100,0%
Tempo de Agregação (em segundos)	100,0%

Descritivas

			Estatística
Tempo de Execução (em microssegundos)	Média		13,239239463
	95% de Intervalo de Confiança para Média	Limite inferior	9,667521294
		Limite superior	16,810957632
	5% da média aparada		12,129244881
	Mediana		6,983084000
	Variância		191,167
	Erro Padrão		13,826323778
	Mínimo		1,4152550
	Máximo		45,5648785
	Amplitude		44,1496235
	Amplitude interquartil		22,4145695
	Assimetria		,903
	Curtose		-,278
Uso de Memória (em MB)	Média		201435,37699
	95% de Intervalo de Confiança para Média	Limite inferior	181407,00047
		Limite superior	221463,75351
	5% da média aparada		201434,35476
	Mediana		201423,26785
	Variância		6011053232,3
	Erro Padrão		77530,982403
	Mínimo		124552,87000
	Máximo		278336,28420
	Amplitude		153783,41420
	Amplitude interquartil		153783,26570
	Assimetria		,000
	Curtose		-2,070
Tempo de Agregação (em segundos)	Média		13,238796380
	95% de Intervalo de Confiança para Média	Limite inferior	9,667192989
		Limite superior	16,810399771
	5% da média aparada		12,128790974
	Mediana		6,983084000
	Variância		191,155
	Erro Padrão		13,825879464
	Mínimo		1,4152550
	Máximo		45,5628045
	Amplitude		44,1475495
	Amplitude interquartil		22,4145695
	Assimetria		,903
	Curtose		-,278

Descritivas

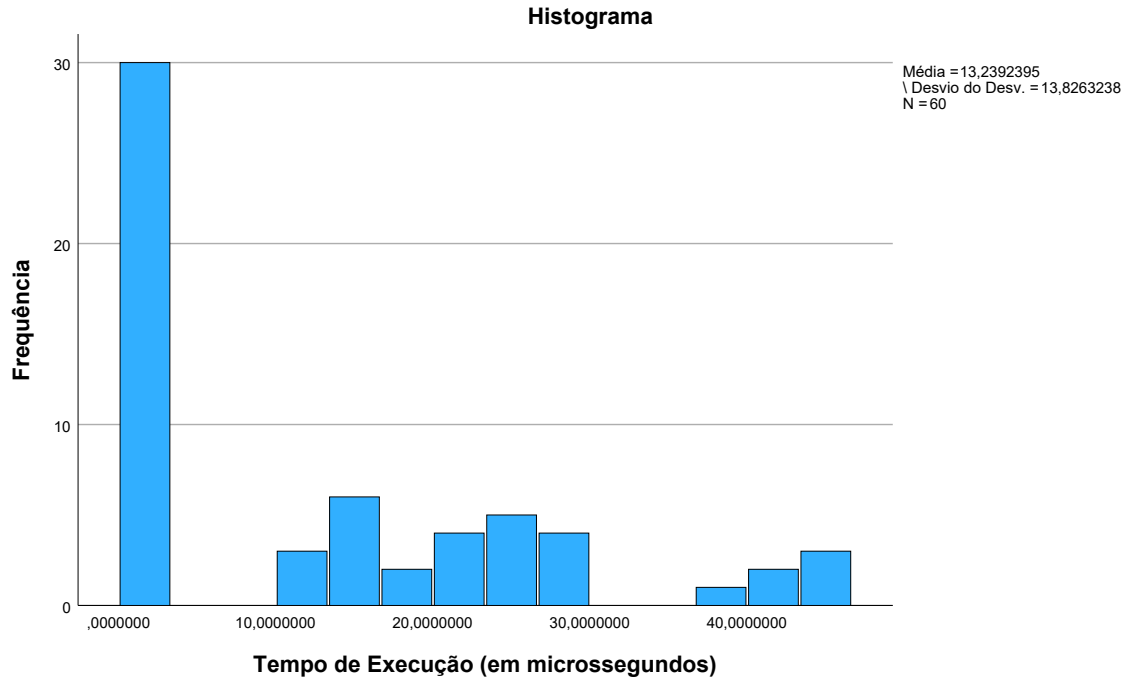
			Estatística do teste Padrão
Tempo de Execução (em microssegundos)	Média		1,7849707244
	95% de Intervalo de Confiança para Média	Limite inferior	
		Limite superior	
	5% da média aparada		
	Mediana		
	Variância		
	Erro Padrão		
	Mínimo		
	Máximo		
	Amplitude		
	Amplitude interquartil		
	Assimetria		,309
	Curtose		,608
Uso de Memória (em MB)	Média		10009,206789
	95% de Intervalo de Confiança para Média	Limite inferior	
		Limite superior	
	5% da média aparada		
	Mediana		
	Variância		
	Erro Padrão		
	Mínimo		
	Máximo		
	Amplitude		
	Amplitude interquartil		
	Assimetria		,309
	Curtose		,608
Tempo de Agregação (em segundos)	Média		1,7849133638
	95% de Intervalo de Confiança para Média	Limite inferior	
		Limite superior	
	5% da média aparada		
	Mediana		
	Variância		
	Erro Padrão		
	Mínimo		
	Máximo		
	Amplitude		
	Amplitude interquartil		
	Assimetria		,309
	Curtose		,608

Testes de Normalidade

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estatística	gl	Sig.	Estatística	gl	Sig.
Tempo de Execução (em microssegundos)	,288	60	<,001	,807	60	<,001
Uso de Memória (em MB)	,339	60	<,001	,637	60	<,001
Tempo de Agregação (em segundos)	,288	60	<,001	,807	60	<,001

a. Correlação de Significância de Lilliefors

Tempo de Execução (em microssegundos)



Tempo de Execução (em microssegundos) Gráfico de Ramos e Folhas

```
 Frequência  Raiz &  Folha

   30,00        0 .  111111111111111111111111111222
     ,00        0 .
    8,00        1 .  12334444
    3,00        1 .  577
    7,00        2 .  1223444
    6,00        2 .  567889
     ,00        3 .
    1,00        3 .  8
    3,00        4 .  024
    2,00        4 .  55

 Largura do ramo:  10,00000
 Cada folha:        1 caso(s)
```

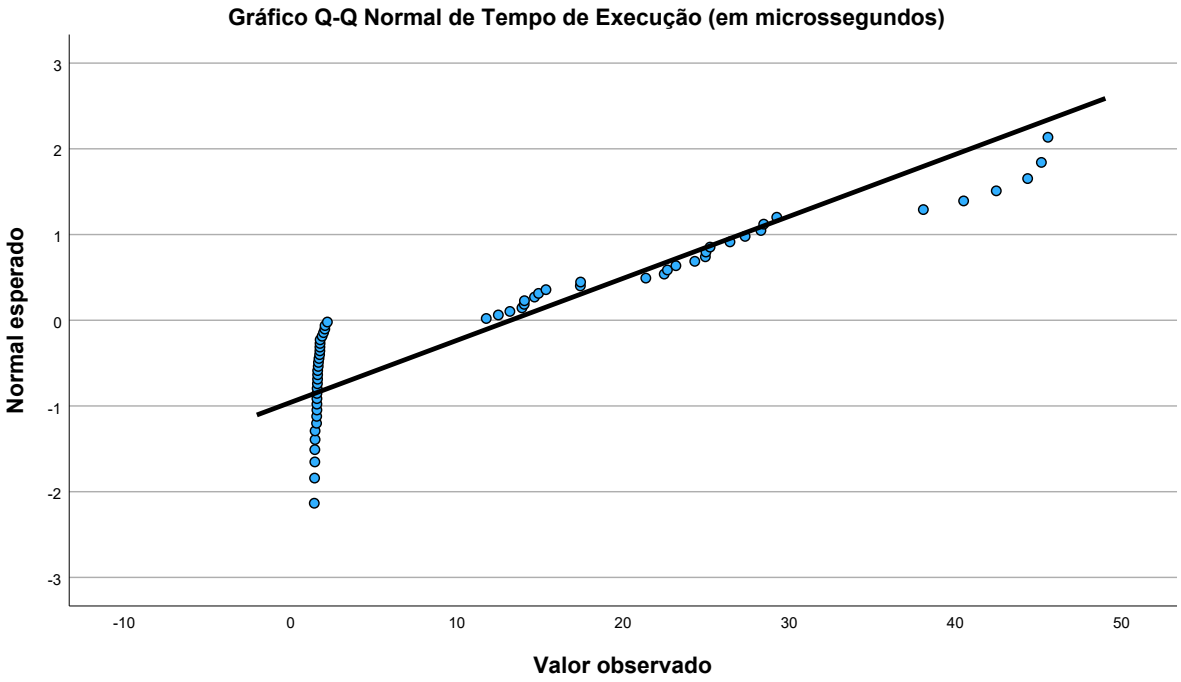


Gráfico Q-Q Normal de Tempo de Execução (em microssegundos)
Normal esperado
3
2
1
0
-1
-2
-3
-10
0
10
20
30
40
50
Valor observado

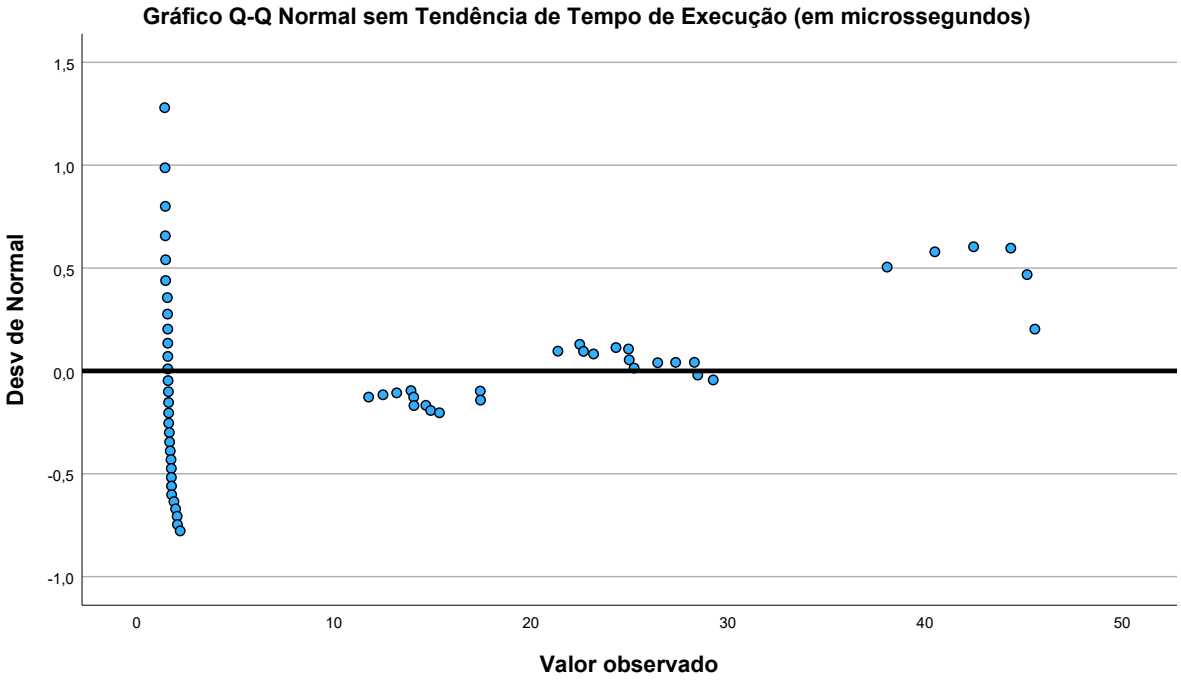
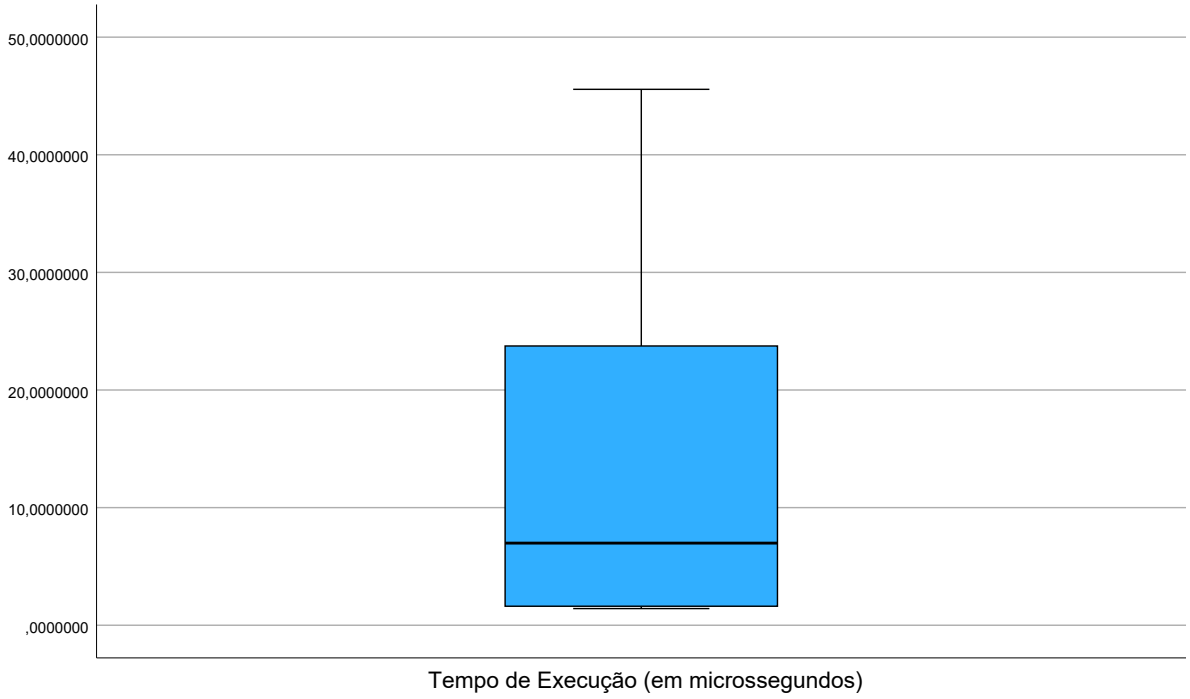
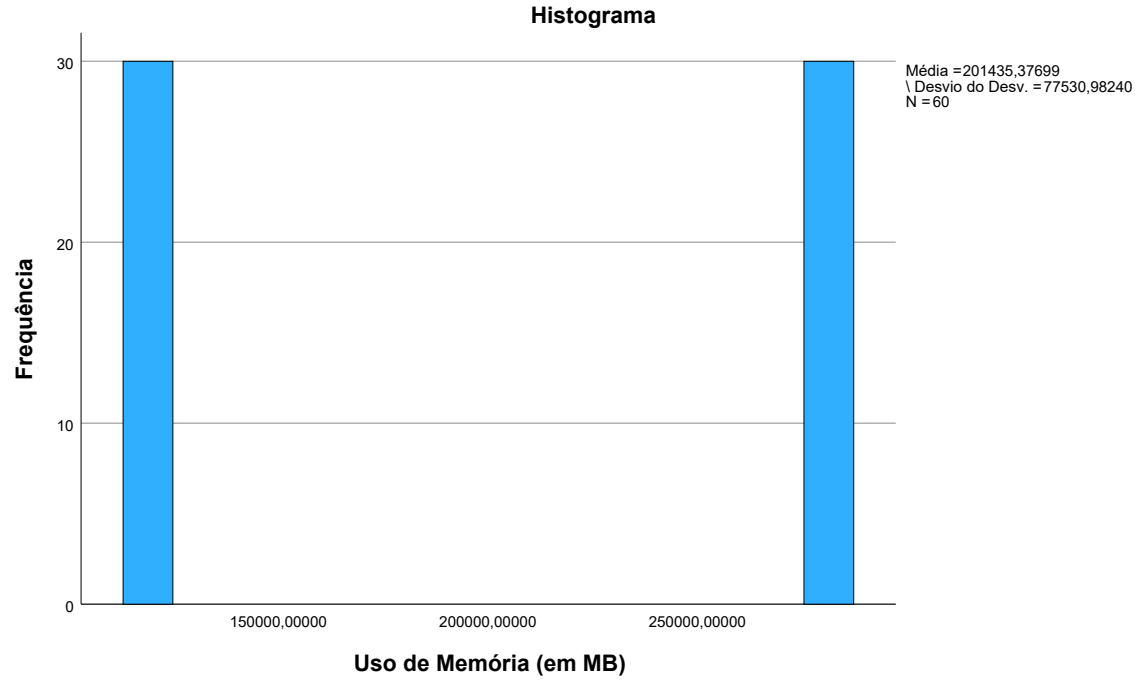


Gráfico Q-Q Normal sem Tendência de Tempo de Execução (em microssegundos)
Desv de Normal
1,5
1,0
0,5
0,0
-0,5
-1,0
0
10
20
30
40
50
Valor observado



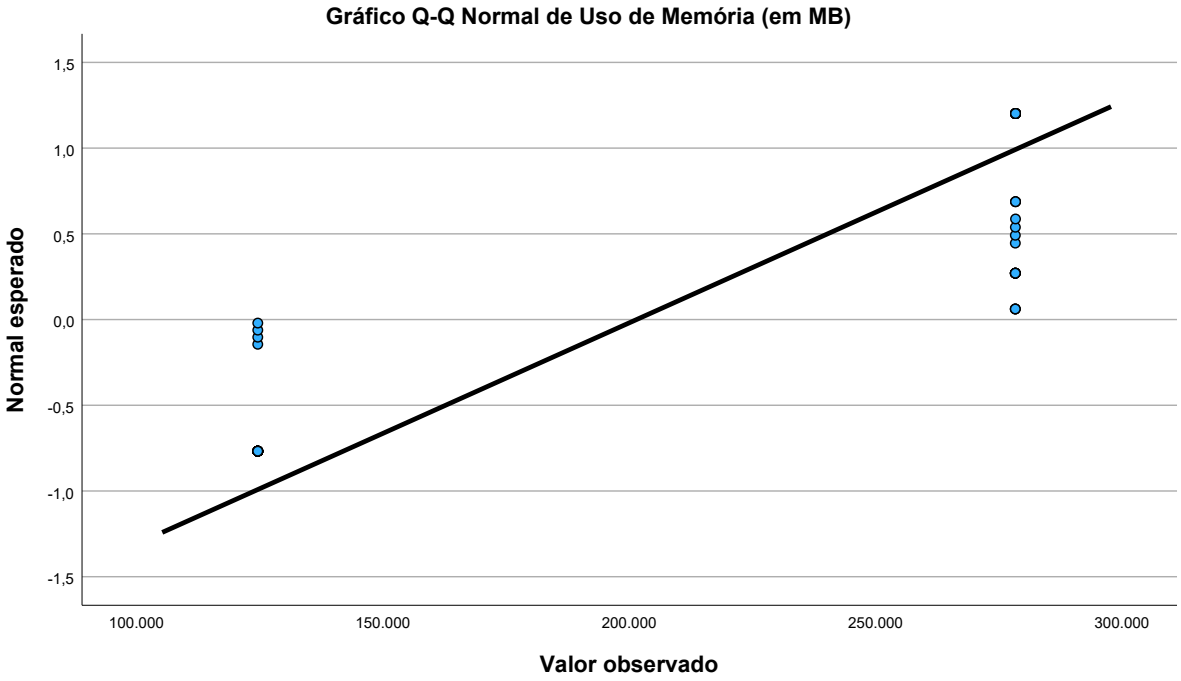
Uso de Memória (em MB)

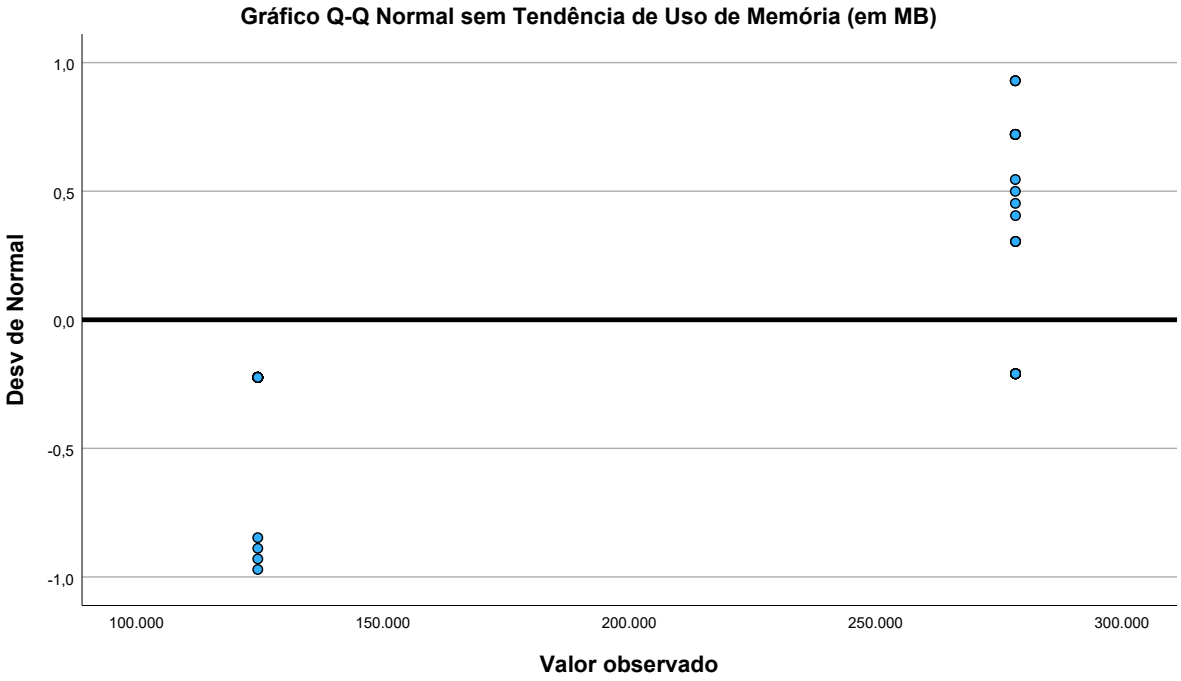


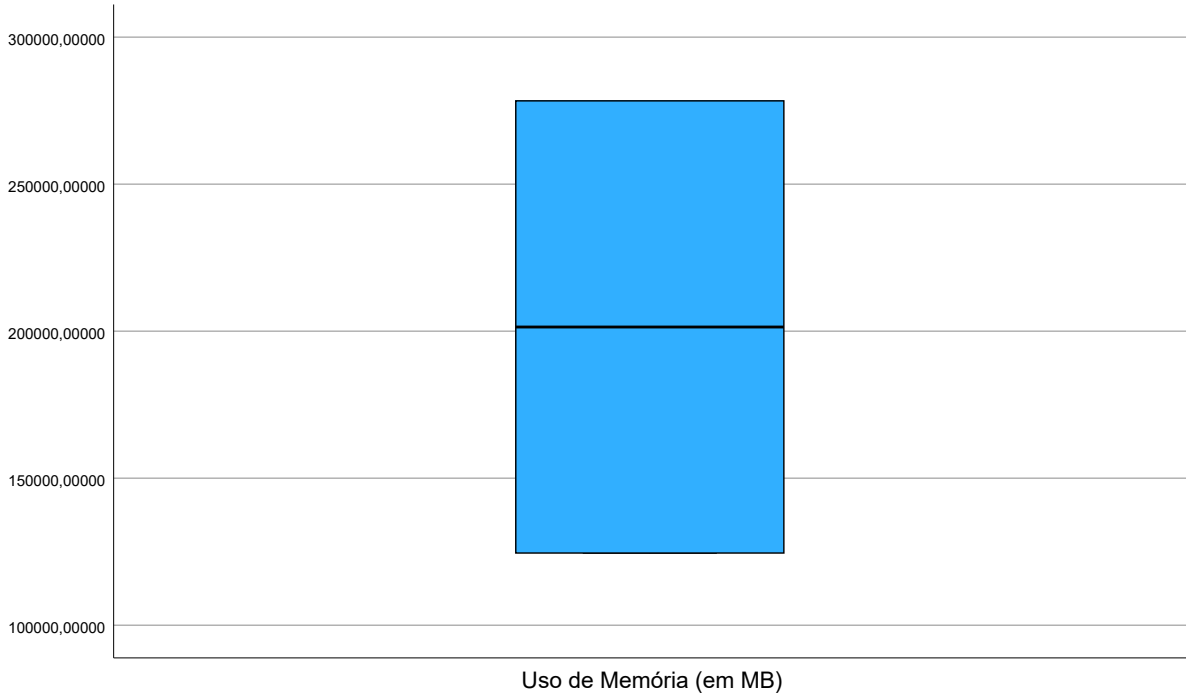
Uso de Memória (em MB) Gráfico de Ramos e Folhas

```
 Frequência  Raiz &  Folha

    30,00        1 .  222222222222222222222222222222
     ,00        1 .
     ,00        1 .
     ,00        1 .
```

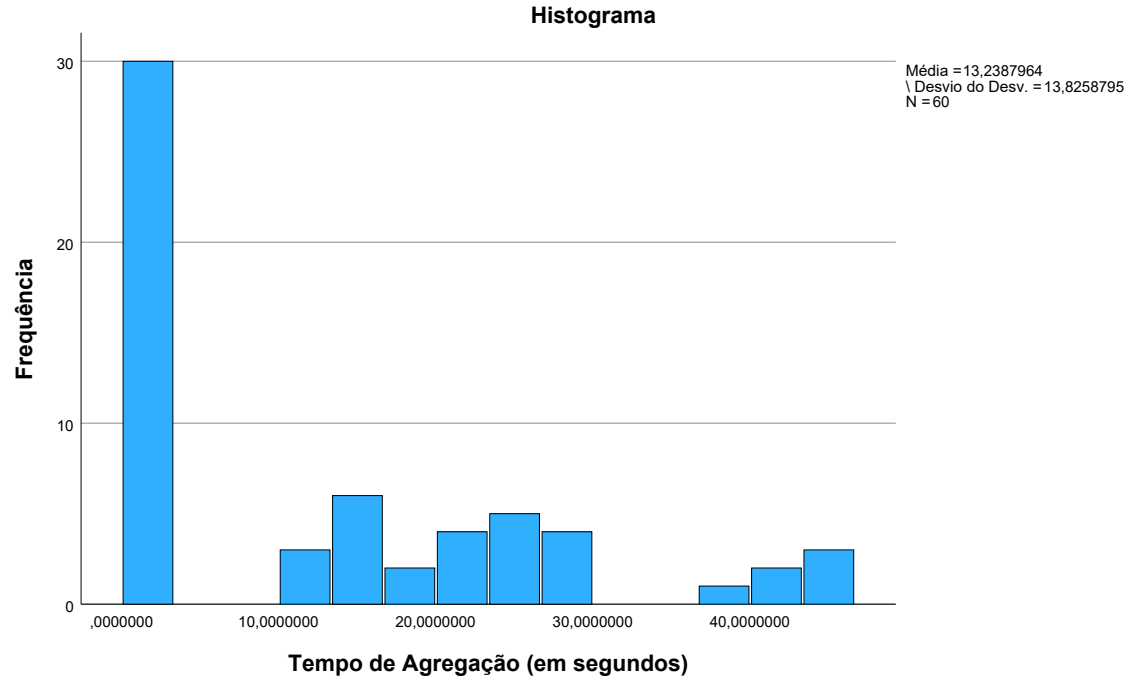
```
    ,00        2 .
    ,00        2 .
    ,00        2 .
  30,00        2 .  77777777777777777777777777777

Largura do ramo:  100000,0
Cada folha:        1 caso(s)
```







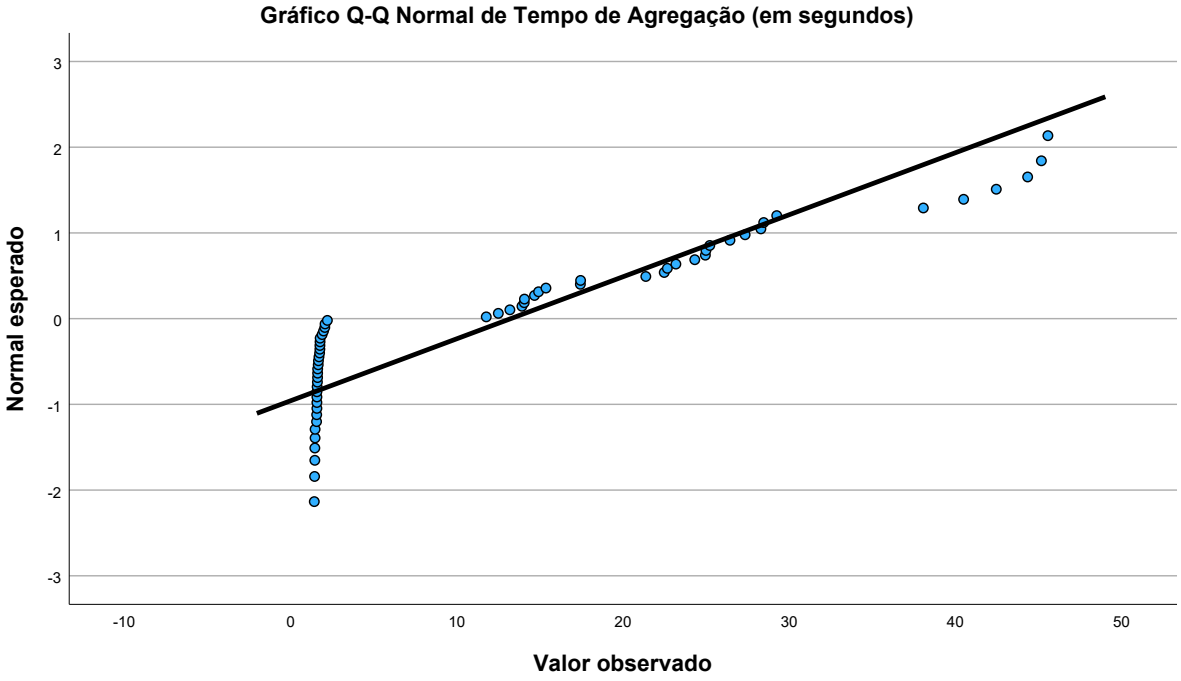
Tempo de Agregação (em segundos)



Tempo de Agregação (em segundos) Gráfico de Ramos e Folhas

```
 Frequência  Raiz &  Folha

    30,00        0 .  111111111111111111111111111222
      ,00        0 .
     8,00        1 .  12334444
     3,00        1 .  577
```

```
    7,00        2 .  1223444
    6,00        2 .  567889
     ,00        3 .
    1,00        3 .  8
    3,00        4 .  024
    2,00        4 .  55

Largura do ramo:  10,00000
Cada folha:        1 caso(s)
```



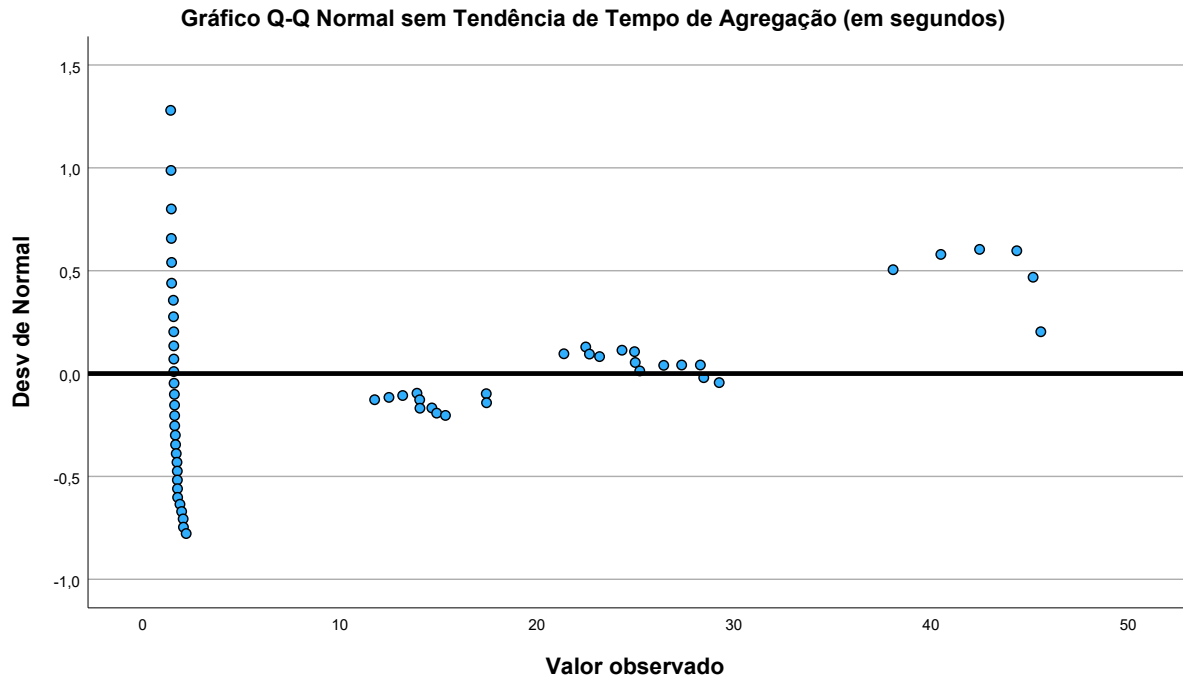
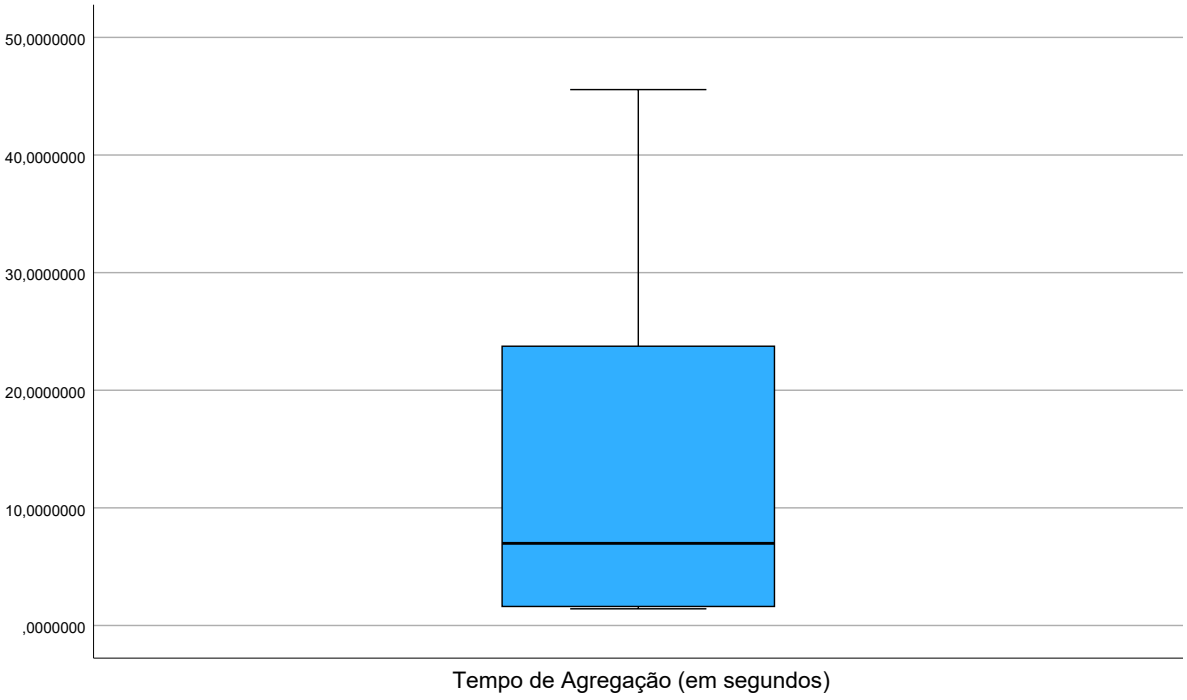


Gráfico Q-Q Normal sem Tendência de Tempo de Agregação (em segundos)
Desv de Normal
1,5
1,0
0,5
0,0
-0,5
-1,0
0
10
20
30
40
50
Valor observado


50,0000000
40,0000000
30,0000000
20,0000000
10,0000000
,0000000
Tempo de Agregação (em segundos)

Observações

Saída criada		16-NOV-2024 22:10:26
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60

Observações

Sintaxe	GGRAPH /GRAPHDATASET NAME="graphdataset" VARIABLES=Grupo MEAN (Aggregation_time) MEAN (Exec_time) MEAN(Memory_usage) MISSING=LISTWISE REPORTMISSING=NO TRANSFORM=VARSTOC ASES(SUMMARY=" #SUMMARY" INDEX=" #INDEX") /GRAPHSPEC SOURCE=INLINE /COLORCYCLE COLOR1 (85,150,230), COLOR2 (0,180,160), COLOR3 (33,213,210), COLOR4 (79,33,150), COLOR5(0,158,154), COLOR6(0,114,195), COLOR7(208,176,255), COLOR8(0,97,97), COLOR9(250,117,166), COLOR10(0,60,115), COLOR11(169,112,255), COLOR12(209,39,101), COLOR13(108,202,255), COLOR14(110,50,201), COLOR15(1,186,182), COLOR16(118,11,57), COLOR17(17,147,232), COLOR18(0,125,121), COLOR19(255,160,194), COLOR20(137,63,252) /FRAME OUTER=NO INNER=NO /GRIDLINES XAXIS=NO YAXIS=YES /STYLE GRADIENT=NO. BEGIN GPL SOURCE: s=userSource (id("graphdataset")) DATA: Grupo=col(source (s), name("Grupo"), unit. category()) DATA: SUMMARY=col (source(s), name ("#SUMMARY")) DATA: INDEX=col(source (s), name("#INDEX"), unit. category()) GUIDE: axis(dim(2), label ("Média")) GUIDE: legend(aesthetic (aesthetic.color.interior), label("")) GUIDE: text.title(label ("Comparação de Métricas entre as Estratégias de Chatterjee et al. ", "(2021) e Saxena e Raychoudhury (2016)")) SCALE: cat(dim(1),

Observações

Recursos	Tempo do processador	00:00:00,39
	Tempo decorrido	00:00:00,50

Observações

Saída criada		16-NOV-2024 22:11:01
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60

Observações

Sintaxe	GGRAPH /GRAPHDATASET NAME="graphdataset" VARIABLES=Grupo MEAN (Aggregation_time) MEAN (Exec_time) MEAN(Memory_usage) MISSING=LISTWISE REPORTMISSING=NO TRANSFORM=VARSTOC ASES(SUMMARY=" #SUMMARY" INDEX=" #INDEX") /GRAPHSPEC SOURCE=INLINE /COLORCYCLE COLOR1 (85,150,230), COLOR2 (0,180,160), COLOR3 (33,213,210), COLOR4 (79,33,150), COLOR5(0,158,154), COLOR6(0,114,195), COLOR7(208,176,255), COLOR8(0,97,97), COLOR9(250,117,166), COLOR10(0,60,115), COLOR11(169,112,255), COLOR12(209,39,101), COLOR13(108,202,255), COLOR14(110,50,201), COLOR15(1,186,182), COLOR16(118,11,57), COLOR17(17,147,232), COLOR18(0,125,121), COLOR19(255,160,194), COLOR20(137,63,252) /FRAME OUTER=NO INNER=NO /GRIDLINES XAXIS=NO YAXIS=YES /STYLE GRADIENT=NO. BEGIN GPL SOURCE: s=userSource (id("graphdataset")) DATA: Grupo=col(source (s), name("Grupo"), unit. category()) DATA: SUMMARY=col (source(s), name ("#SUMMARY")) DATA: INDEX=col(source (s), name("#INDEX"), unit. category()) GUIDE: axis(dim(2), label ("Média")) GUIDE: legend(aesthetic (aesthetic.color.interior), label("")) GUIDE: text.title(label ("Comparação de Métricas entre as Estratégias de Chatterjee et al. ", "(2021) e Saxena e Raychoudhury (2016)")) SCALE: cat(dim(1),

Observações

Recursos	Tempo do processador	00:00:00,22
	Tempo decorrido	00:00:00,26

Observações

Saída criada		16-NOV-2024 22:15:21
Comentários		
Entrada	Dados	C:\Users\dalia\OneDrive\Documentos\MESTRADO UFC\teste de hipotese UFC1.sav
	Conjunto de dados ativo	ConjuntodeDados1
	Filtro	<none>
	Ponderação	<none>
	Dividir Arquivo	<none>
	N de linhas em arquivo de dados de trabalho	60

Observações

Sintaxe	GGRAPH /GRAPHDATASET NAME="graphdataset" VARIABLES=Grupo MEAN (Aggregation_time) MEAN (Exec_time) MEAN(Memory_usage) MISSING=LISTWISE REPORTMISSING=NO TRANSFORM=VARSTOCASES(SUMMARY="#SUMMARY" INDEX="#INDEX") /GRAPHSPEC SOURCE=INLINE /COLORCYCLE COLOR1 (85,150,230), COLOR2 (0,180,160), COLOR3 (237,75,75), COLOR4 (79,33,150), COLOR5(0,158,154), COLOR6(0,114,195), COLOR7(208,176,255), COLOR8(0,97,97), COLOR9(250,117,166), COLOR10(0,60,115), COLOR11(169,112,255), COLOR12(209,39,101), COLOR13(108,202,255), COLOR14(110,50,201), COLOR15(1,186,182), COLOR16(118,11,57), COLOR17(17,147,232), COLOR18(0,125,121), COLOR19(255,160,194), COLOR20(137,63,252) /FRAME OUTER=NO INNER=NO /GRIDLINES XAXIS=NO YAXIS=YES /STYLE GRADIENT=NO. BEGIN GPL SOURCE: s=userSource (id("graphdataset")) DATA: Grupo=col(source (s), name("Grupo"), unit. category()) DATA: SUMMARY=col (source(s), name ("#SUMMARY")) DATA: INDEX=col(source (s), name("#INDEX"), unit. category()) GUIDE: axis(dim(2), label ("Média")) GUIDE: legend(aesthetic (aesthetic.color.interior), label("")) SCALE: cat(dim(1), include("1", "2")) SCALE: linear(dim(2), include(0)) SCALE: cat(aesthetic (aesthetic.color.interior), include(

Observações

Recursos	Tempo do processador	00:00:00,41
	Tempo decorrido	00:00:00,35

APÊNDICE B – TEMPO MÉDIO ENTRE AS ESTRATÉGIAS NA FERRAMENTA IBM SPSS - SAXENA E RAYCHOUDHURY (2016) - GRUPO 1 E CHATTERJEE ET AL. (2021) - GRUPO 2

RUN	MEMORY_USAGE	EXEC_TIME	AGGREGATION_TIME	GRUPO
1	278294,46480	22.473,9072	22.473,907	1
2	278336,28420	14.0617,559	14.061,755	1
3	278336,28420	14.910,244	14.910,224	1
4	278336,13570	45.170,897	45.170,897	1
5	278292,63570	44.343,770	44.343,770	1
6	278293,01070	45.564,878	45.562,804	1
7	278336,28420	40.490,742	40.490,742	1
8	278336,28420	42.456,185	42.456,185	1
9	278336,28420	38.067,562	38.0675,620	1
10	278336,28420	21.366,005	21.366,005	1
11	278307,13570	24.312,694	24.312,694	1
12	278292,63570	23.175,406	23.175,406	1
13	278336,28420	28.460,432	28.460,432	1
14	278292,63570	27.340,846	27.340,846	1
15	278336,28420	22.666,017	22.661,105	1
16	278292,63570	28.291,741	28.289,794	1
17	278289,38570	24.949,089	24.949,089	1
18	278336,13570	24.984,468	24.984,468	1
19	278336,13570	29.246,646	29.244,593	1

Continua na próxima página

RUN	MEMORY_USAGE	EXEC TIME	AGGREGATION_TIME	GRUPO
20	278292,63570	26.429,825	26.429,825	1
21	278289,38570	25.231,037	25.215,438	1
22	278335,22950	14.666,333	14.666,333	1
23	278292,63570	17.427,633	17.427,633	1
24	278336,28420	11.764,147	11.764,147	1
25	278292,63570	17.440,796	17.440,796	1
26	278289,38570	15.356,501	15.356,501	1
27	278336,28420	13.185,877	13.185,877	1
28	278336,28420	14.048,661	14.048,661	1
29	278336,28420	13.911,662	13.911,662	1
30	278336,28420	12.490,017	12.490,017	1
1	124557,15000	1.754,235	1.754,235	2
2	124556,67000	1.885,806	1.885,806	2
3	124552,87000	1.587,301	1.587,301	2
4	124552,87000	1.462,797	1.462,797	2
5	124552,87000	1.651,742	1.651,742	2
6	124552,87000	1.697,295	1.697,295	2
7	124552,87000	1.616,510	1.616,510	2
8	124552,87000	2.202,021	2.202,021	2
9	124552,87000	1.971,512	1.971,512	2
10	124552,87000	2.062,958	2.062,958	2
11	124552,87000	2.047,734	2.047,734	2

Continua na próxima página

RUN	MEMORY_USAGE	EXEC TIME	AGGREGATION_TIME	GRUPO
12	124552,87000	1.767,727	1.767,727	2
13	124552,87000	1.612,900	1.612,900	2
14	124552,87000	1.447,917	1.447,917	2
15	124552,87000	1.574,609	1.574,609	2
16	124552,87000	1.572,478	1.572,478	2
17	124552,87000	1.448,227	1.448,227	2
18	124552,87000	1.619,479	1.619,479	2
19	124552,87000	1.751,571	1.751,571	2
20	124552,87000	1.761,616	1.761,616	2
21	124554,38000	1.735,501	1.735,501	2
22	124553,25000	1.463,775	1.463,775	2
23	124552,87000	1.551,727	1.551,727	2
24	124552,87000	1.574,316	1.574,316	2
25	124552,87000	1.559,712	1.559,712	2
26	124552,87000	1.415,255	1.415,255	2
27	124552,87000	1.603,112	1.603,112	2
28	124552,87000	1.432,426	1.432,426	2
29	124552,87000	1.571,773	1.571,773	2
30	124552,87000	1.664,580	1.664,580	2

APÊNDICE C – CÓDIGO UTILIZADO PARA O TESTE DE DESEMPENHO DE SAXENA E RAYCHOUDHURY (2016)

```
import ipaddress
import time
import tracemalloc
import csv
from collections import defaultdict

class PatriciaTrieNode:
    def __init__(self):
        self.children = {}
        self.is_end = False
        self.next_hops = set()

class PatriciaTrie:
    def __init__(self):
        self.root = PatriciaTrieNode()

    def insert(self, binary_prefix, next_hop):
        node = self.root
        for bit in binary_prefix:
            if bit not in node.children:
                node.children[bit] = PatriciaTrieNode()
            node = node.children[bit]
        node.is_end = True
        node.next_hops.add(next_hop)

    def search_common_prefixes(self):
        def dfs(node, prefix, results):
            if node.is_end:
```

```
29                  results.append((prefix, list(node.next_hops
                        )))
30              for bit, child in node.children.items():
31                  dfs(child, prefix + bit, results)
32
33          results = []
34          dfs(self.root, '', results)
35          return results
36
37      def calculate_aggregation_time(self):
38          start_time = time.time()
39          common_prefixes = self.search_common_prefixes()
40          end_time = time.time()
41          aggregation_time = end_time - start_time
42          return common_prefixes, aggregation_time
43
44  def ip_to_binary(ip_prefix):
45      try:
46          network = ipaddress.ip_network(ip_prefix, strict=
                False)
47          binary_prefix = ''.join(f"{bin_part:08b}" if
                isinstance(bin_part, int) else f"{int(bin_part,
                16):016b}" for bin_part in network.
                network_address.exploded.split(':') if bin_part)
48          return binary_prefix[:network.prefixlen]
49      except ValueError as e:
50          print(f"Erro ao converter o prefixo IP '{ip_prefix
                }': {e}")
51          return None
52
53  def process_file(file_path):
54      trie = PatriciaTrie()
```

```
55      with open(file_path, 'r') as file:
56          next(file)  # Pula o cabe alho
57          for line in file:
58              parts = line.strip().split()
59              if len(parts) == 2:
60                  ip_prefix = parts[0]
61                  next_hop = parts[1]
62                  binary_prefix = ip_to_binary(ip_prefix)
63              elif len(parts) == 3:
64                  ip_prefix = parts[0]
65                  binary_prefix = parts[1]
66                  next_hop = parts[2]
67              else:
68
69
70              if binary_prefix:
71                  trie.insert(binary_prefix, next_hop)
72              else:
73                  print(f"Prefixo bin rio inv lido para o
                        IP: {ip_prefix}")
74
75      return trie
76
77  def measure_performance(trie):
78      tracemalloc.start()
79      start_time = time.time()
80      common_prefixes, aggregation_time = trie.
           calculate_aggregation_time()
81      end_time = time.time()
82      current, peak = tracemalloc.get_traced_memory()
83      tracemalloc.stop()
84
```

```
85      exec_time = end_time - start_time
86      memory_usage = peak / 1024
87      return exec_time, memory_usage, aggregation_time
88
89  def run_performance_test(input_file, output_file, num_runs
        =30):
90      data = []
91      for i in range(num_runs):
92          trie = process_file(input_file)
93          exec_time, memory_usage, aggregation_time =
                measure_performance(trie)
94
95          data.append({
96              'run': i + 1,
97              'exec_time': exec_time,
98              'memory_usage': memory_usage,
99              'aggregation_time': aggregation_time
100         })
101
102     with open(output_file, 'w', newline='') as file:
103         writer = csv.DictWriter(file, fieldnames=data[0].
                keys())
104         writer.writeheader()
105         writer.writerows(data)
106     print(f"Dados de desempenho coletados e salvos em {
            output_file}")
107
108 # Caminho dos arquivos de entrada e sa da
109 input_file = r"C:\Diretorio"
110 output_file = r"C:\Diretorio"
111
112 # Executa o teste de desempenho
```

```
113 run_performance_test(input_file, output_file)
```

APÊNDICE D – CÓDIGO UTILIZADO PARA O TESTE DE DESEMPENHO DE CHATTERJEE ET AL. (2021)

```
import time
import tracemalloc  # For memory usage measurement
import csv

# Function to load prefixes from file
def load_prefixes(file_path):
    prefixes = []
    with open(file_path, 'r') as file:
        lines = file.readlines()
        for line in lines[3:]:  # Skip the first three header lines
            parts = line.strip().split()
            if len(parts) >= 4:
                ip_prefix = parts[0]          # IP Address / Prefix
                mask = parts[1]               # Mask
                full_binary = parts[2]        # Full Binary Representation
                compact_binary = parts[3]     # Compact Binary Prefix
                prefixes.append({
                    "Endereco IP / Prefixo": ip_prefix,
                    "Mascara": mask,
                    "Representacao Binaria Completa": full_binary,
                    "Prefixo Binario Compacto": compact_binary
                })
    return prefixes
```

```
24
25 # Function to group common prefixes
26 def group_common_prefixes(prefixes):
27     grouped = {}
28     for prefix in prefixes:
29         compact_binary = prefix["Prefixo Binario Compacto"]
30         if compact_binary not in grouped:
31             grouped[compact_binary] = []
32         grouped[compact_binary].append(prefix)
33     return grouped
34
35 # Function to measure performance metrics
36 def measure_performance(prefixes):
37     tracemalloc.start()  # Start memory tracking
38     start_time = time.time()  # Start execution timer
39     grouped_prefixes = group_common_prefixes(prefixes)
40     end_time = time.time()  # End execution timer
41     current, peak = tracemalloc.get_traced_memory()  # Get
       memory usage
42     tracemalloc.stop()
43     exec_time = end_time - start_time  # Execution time
44     memory_usage = peak / 1024  # Memory usage in KB
45     return grouped_prefixes, exec_time, memory_usage
46
47 # Function to save grouped prefixes to a file
48 def save_grouped_prefixes(grouped_prefixes, output_file,
   grouping_time):
49     with open(output_file, 'w') as file:
50         file.write(f"Tempo de agrupamento dos prefixos: {
              grouping_time:.6f} segundos\n")
51         file.write("=" * 60 + "\n")
```

```
52          file.write(f"{'Prefixo Binario Compacto':<30} {'
               Enderecos IP / Prefixos':<40}\n")
53          file.write("=" * 60 + "\n")
54
55          for compact_binary, prefixes in grouped_prefixes.
               items():
56              ip_prefixes = ', '.join([p["Endereco IP /
                   Prefixo"] for p in prefixes])
57              file.write(f"{compact_binary:<30} {ip_prefixes
                   :<40}\n")
58
59  # Function to save performance metrics to a CSV file
60  def save_metrics_to_csv(output_csv, run, exec_time,
       memory_usage, aggregation_time):
61      with open(output_csv, 'a', newline='') as file:
62          writer = csv.writer(file)
63          if run == 1:  # Add header only on the first run
64              writer.writerow(["run", "exec_time", "
                   memory_usage", "aggregation_time"])
65          writer.writerow([run, exec_time, memory_usage,
               aggregation_time])
66
67  # Paths for input and output files
68  input_file = r"C:\Diretorio"
69  output_csv_file = r"C:\Diretorio"
70  output_grouped_file = r"C:\Diretorio"
71
72  # Number of executions
73  num_runs = 30  # Example: 30 runs
74
75  # Run multiple tests and collect performance metrics
76  for run in range(1, num_runs + 1):
```

```
77      prefixes = load_prefixes(input_file)  # Load prefixes
78      grouped_prefixes, exec_time, memory_usage =
           measure_performance(prefixes)  # Measure performance
79      aggregation_time = exec_time  # Assuming aggregation
           time equals execution time for simplicity
80
81      # Print results to the console
82      print(f"Run: {run}, Exec Time: {exec_time:.6f}s, Memory
            Usage: {memory_usage:.2f}KB, Aggregation Time: {
           aggregation_time:.6f}s")
83
84      # Save metrics to CSV
85      save_metrics_to_csv(output_csv_file, run, exec_time,
           memory_usage, aggregation_time)
86
87      # Save grouped prefixes on the last run (optional)
88      if run == num_runs:
89          save_grouped_prefixes(grouped_prefixes,
              output_grouped_file, aggregation_time)
90
91  print(f"Performance data saved to: {output_csv_file}")
92  print(f"Grouped prefixes saved to: {output_grouped_file}")
```

APÊNDICE E – IPS NÃO COBERTOS PELA *PATRICIA TREE*

Tabela 22: Lista de endereços IP com menor frequência de anúncios

IP	IP	IP
108.156.16.0/21	93.185.82.0/24	185.43.236.0/22
2001:49f0:d122::/48	2404:bf40:f005::/48	2a04:4e40:5830::/44
192.61.72.0/21	194.126.174.0/24	2a0c:14c1::/32
114.111.148.0/23	193.187.92.0/22	72.19.184.0/23
64.189.106.0/24	148.164.49.0/24	2405:f600:17::/48
2a0b:dbc0::/29	217.110.62.0/24	109.86.104.0/24
2001:16a4:23e::/47	81.181.105.0/24	46.227.253.0/24
45.236.92.0/24	149.3.162.0/23	70.163.104.0/21
2a10:500:4f00::/40	61.30.204.0/22	147.92.135.0/24
146.209.222.0/24	97.135.128.0/18	187.194.20.0/24
157.124.19.0/24	200.2.182.0/24	103.38.70.0/24
165.140.156.0/24	2402:800:61f4::/48	86.110.194.0/24
2806:370:6120::/44	65.111.102.0/24	50.114.13.0/24
185.127.113.0/24	91.224.142.0/23	197.210.116.0/24
206.164.195.0/24	2804:2a80:38::/45	161.77.46.0/24
94.187.244.0/24	240a:6b:b700::/40	74.247.16.0/21
151.244.104.0/23	143.165.80.0/24	2a00:4bc0:214a::/48
2804:43f0:3000::/36	2a0b:4340:c8::/48	185.134.136.0/22
185.130.59.0/24	210.72.30.0/23	219.69.0.0/17
2607:fcd0:100:2d06::/64	2800:810:490::/48	103.79.124.0/24
108.158.130.0/23	186.251.88.0/24	122.155.221.0/24

IP	IP	IP
185.86.96.0/22	164.251.240.0/22	2001:559:845d::/48
216.75.214.0/24	103.196.178.0/24	112.214.16.0/24
91.233.15.0/24	176.120.119.0/24	76.73.232.0/22
62.165.62.0/23	201.192.191.0/24	114.135.0.0/16
2001:1248:4640::/42	2a02:cb42::/32	191.250.208.0/20
2409:8b61::/32	185.205.232.0/22	211.161.97.0/24
2406:e002:5800::/37	2a05:d050:6040::/46	115.118.12.0/22
2404:1c40:34::/48	170.0.236.0/24	103.191.146.0/23
185.136.16.0/22	178.170.143.0/24	91.196.93.0/24
202.218.32.0/22	113.213.240.0/23	88.218.206.0/24
171.20.76.0/24	2a0e:800:ff40::/42	62.182.80.0/23
205.176.218.0/24	187.210.247.0/24	190.146.160.0/23
111.68.1.0/24	182.190.240.0/21	203.135.19.0/24
170.82.99.0/24	2607:ffb0:3011::/48	82.161.0.0/16
124.219.84.0/24	24.245.105.0/24	2001:67c:258c::/48
140.99.74.0/24	2800:300:6220::/44	220.196.152.0/24
151.104.192.0/21	38.13.253.0/24	170.246.254.0/24
2402:e280:3e78::/48	222.255.64.0/23	2401:b740::/32
196.128.72.0/22	72.240.203.0/24	148.59.224.0/22
189.233.239.0/24	2001:4528:fff::/48	36.66.11.0/24
190.1.191.0/24	50.114.242.0/24	177.200.54.0/23
193.252.153.0/24	217.156.52.0/24	103.200.0.0/24
113.11.120.0/24	104.192.48.0/22	203.148.86.0/23

IP	IP	IP
109.160.16.0/24	200.198.114.0/24	27.71.109.0/24
2001:16a2:c370::/46	2001:579:e188::/46	64.223.87.0/24
131.62.15.0/24	165.140.123.0/24	2405:3440:a::/48
139.38.18.0/24	149.51.226.0/24	187.164.23.0/24
42.118.27.0/24	66.38.15.0/24	50.50.4.0/22
91.219.85.0/24	103.10.99.0/24	103.82.196.0/24
118.211.24.0/21	206.44.232.0/21	140.182.0.0/19
45.163.156.0/22	223.224.178.0/24	5.83.32.0/24
66.193.38.0/24	177.53.11.0/24	136.244.80.0/20
130.22.104.0/24	46.228.48.0/20	203.193.188.0/24
32.40.224.0/20	50.3.92.0/22	186.219.232.0/22
125.213.160.0/21	144.220.19.0/24	200.35.8.0/21
190.24.96.0/20	5.62.38.0/24	24.123.128.0/20
23.216.78.0/24	2a0f:5fc0:b00b::/48	208.131.234.0/24
35.200.112.0/20	45.197.192.0/19	68.162.192.0/19
205.177.224.0/24	103.70.138.0/24	216.108.227.0/24
213.55.64.0/19	88.223.128.0/22	200.194.64.0/24
208.38.176.0/23	185.252.181.0/24	38.44.49.0/24
45.157.115.0/24	103.20.212.0/22	2a00:f2a:b000::/36
123.200.174.0/24	214.28.40.0/24	103.154.134.0/24
199.146.122.0/24	202.44.105.0/24	213.202.81.0/24
2a02:26f7:cac6::/48	2803:4480:4000::/34	109.65.134.0/24
200.111.4.0/24	2400:3ca0:34::/48	194.135.141.0/24

IP	IP	IP
113.196.132.0/24	194.29.0.0/20	2001:df2:1140::/48
182.74.43.0/24	93.125.11.0/24	2a00:12e8:f111::/48
23.149.16.0/24	2001:16a2:c0a1::/48	120.225.80.0/20
148.227.0.0/17	2607:fb10:7194::/48	2604:1a00::/32
2a02:2e02:670::/44	2804:5100:3000::/40	54.231.112.0/21
177.73.17.0/24	185.5.212.0/24	86.111.152.0/21
192.62.56.0/21	2606:4700:3019::/48	2804:2968::/32
2403:49c0:600::/40	210.97.160.0/19	196.245.152.0/24
187.135.143.0/24	206.71.150.0/24	67.215.104.0/22
75.107.219.0/24	165.251.64.0/22	149.126.88.0/22
164.52.120.0/22	84.95.224.0/20	2a0d:f8c0::/43
62.168.205.0/24	2800:370:e4::/46	60.242.84.0/24
179.189.140.0/22	125.165.104.0/21	2408:8024:7000::/36
15.221.128.0/22	43.252.18.0/24	103.193.69.0/24
70.35.174.0/23	183.81.226.0/24	188.68.7.0/24
122.161.84.0/22	61.29.39.0/24	74.1.248.0/23
117.55.128.0/21	69.68.56.0/24	2a03:a4e0::/40
202.8.216.0/21	198.147.156.0/24	13.32.128.0/23
79.180.202.0/24	163.170.144.0/24	24.76.44.0/22
168.78.129.0/24	2606:1f80:f000::/36	146.104.249.0/24
2a04:4e40:aa30::/44	174.46.132.0/24	65.151.0.0/24
24.96.20.0/24	198.187.215.0/24	203.194.122.0/24
202.152.94.0/24	2001:1248:4b89::/48	179.109.62.0/24

IP	IP	IP
198.206.194.0/24	39.130.195.0/24	179.152.187.0/24
1.44.40.0/21	61.44.64.0/23	141.226.0.0/21
58.208.170.0/24	64.83.167.0/24	189.195.242.0/24
40.211.0.0/17	91.228.99.0/24	2800:530:b000::/36
136.226.68.0/23	103.165.23.0/24	101.55.33.0/24
185.142.93.0/24	77.240.94.0/24	178.133.0.0/17
2402:3a80:4036::/47	63.223.115.0/24	94.96.0.0/14
94.28.96.0/22	103.87.44.0/24	45.62.211.0/24
166.137.125.0/24	181.77.16.0/23	37.10.72.0/21
103.11.100.0/23	45.250.36.0/23	164.164.93.0/24
192.141.105.0/24	103.254.81.0/24	195.46.96.0/19
201.156.26.0/24	2804:6280::/32	185.227.191.0/24
200.80.10.0/24	240e:37e:1800::/39	199.67.161.0/24
102.177.185.0/24	64.46.98.0/24	154.192.191.0/24
202.176.217.0/24	182.180.109.0/24	2a0e:97c0:390::/44
193.27.192.0/23	204.116.92.0/24	211.138.128.0/24
207.244.168.0/22	198.186.174.0/23	24.67.188.0/22
203.135.45.0/24	94.77.255.0/24	103.197.246.0/24
154.210.223.0/24	2804:7704::/32	192.189.129.0/24
204.15.169.0/24	23.106.248.0/24	186.216.30.0/24
192.231.213.0/24	2402:3a80:4267::/48	69.58.150.0/24
79.98.176.0/24	2804:6208::/32	220.149.64.0/23
142.252.106.0/24	194.28.0.0/22	182.73.19.0/24

IP	IP	IP
2409:8750:5200::/40	2400:6420:2b::/48	154.127.32.0/20
2602:ff88::/36	36.255.223.0/24	119.160.181.0/24
52.219.165.0/24	177.73.246.0/24	37.111.140.0/24
119.205.178.0/24	49.205.100.0/22	182.156.77.0/24
89.148.42.0/24	103.4.74.0/24	192.144.32.0/24
120.56.96.0/20	103.242.13.0/24	202.69.112.0/20
196.43.97.0/24	223.187.177.0/24	105.96.13.0/24
90.94.100.0/22	103.10.120.0/22	103.29.141.0/24
65.229.4.0/24	85.185.186.0/24	103.106.137.0/24
110.43.176.0/21	196.29.39.0/24	201.193.172.0/22
206.160.177.0/24	23.208.172.0/22	2602:107:e000::/40
45.45.129.0/24	46.153.16.0/20	74.81.81.0/24
2a02:2e02:ce0::/44	61.82.148.0/24	61.205.128.0/18
206.119.226.0/24	240e:804:680::/43	2a0a:1800:12::/48
106.210.152.0/22	147.237.70.0/24	201.122.17.0/24
91.208.28.0/24	122.254.239.0/24	201.198.236.0/24
204.8.227.0/24	2405:2d40:34::/48	170.79.116.0/22
193.178.229.0/24	164.164.17.0/24	162.82.155.0/24
89.16.200.0/24	204.10.234.0/24	91.116.0.0/19
147.46.128.0/19	61.1.144.0/20	176.123.122.0/24
143.137.220.0/22	185.120.85.0/24	97.111.144.0/22
212.233.168.0/24	84.32.136.0/22	82.80.10.0/24
218.240.160.0/21	45.232.230.0/23	2804:66a8::/32

IP	IP	IP
200.221.0.0/16	78.142.85.0/24	37.77.142.0/24
162.216.232.0/22	2409:8904:7520::/44	2605:dd40:c000::/36
50.204.196.0/24	196.133.112.0/21	141.98.218.0/24
185.124.105.0/24	75.88.144.0/21	115.64.240.0/24
103.114.208.0/22	103.168.119.0/24	177.155.96.0/22
94.31.172.0/24	110.92.160.0/22	38.147.199.0/24
87.206.0.0/15	175.118.171.0/24	172.252.144.0/20
200.16.127.0/24	45.45.228.0/24	177.154.243.0/24
2606:2800:e::/48	186.237.160.0/24	177.66.204.0/22
200.185.128.0/18	2800:e02:6000::/36	202.52.239.0/24
1.78.44.0/22	31.177.56.0/21	2402:3a80:1ae8::/45
46.21.72.0/22	5.188.112.0/21	62.150.168.0/24
23.53.52.0/22	2404:f340:4000::/34	94.182.150.0/24
160.120.29.0/24	72.196.189.0/24	203.238.0.0/17
187.224.96.0/24	210.208.80.0/24	14.201.49.0/24
50.67.12.0/22	118.69.63.0/24	179.6.172.0/24
173.214.108.0/24	69.69.218.0/23	69.70.128.0/18
181.79.23.0/24	103.153.242.0/23	170.61.198.0/24
131.100.246.0/24	23.37.236.0/22	83.143.168.0/21
186.32.33.0/24	208.72.20.0/22	146.255.220.0/24
37.186.32.0/19	200.51.252.0/22	36.113.96.0/20
204.186.20.0/24	154.160.7.0/24	103.205.176.0/24
209.131.228.0/22	103.147.220.0/23	202.141.241.0/24

IP	IP	IP
2803:ecc0::/32	170.245.113.0/24	39.134.19.0/24
200.94.16.0/21	2804:1cf8:8000::/34	2a0e:d606::/48
83.231.120.0/22	72.249.32.0/23	2a00:c281:a::/48
187.250.0.0/17	63.250.96.0/19	2408:856d::/32
2a0e:e6c7:f000::/36	65.175.135.0/24	45.194.58.0/24
2804:33e8::/33	116.48.0.0/19	170.78.38.0/24
195.226.241.0/24	122.128.64.0/22	207.192.163.0/24
196.135.80.0/22	109.87.159.0/24	58.145.54.0/24
2a03:5640:f107::/48	2404:1c40:2d1::/48	89.150.35.0/24
194.121.50.0/24	111.65.32.0/24	103.207.48.0/22
110.167.61.0/24	203.83.127.0/24	38.51.173.0/24
212.225.156.0/22	203.118.240.0/24	41.217.56.0/23
80.71.231.0/24	162.253.15.0/24	179.42.147.0/24
199.87.76.0/24	2a13:c907::/32	206.238.85.0/24
106.207.152.0/22	119.12.104.0/21	139.186.48.0/20
64.29.233.0/24	195.54.166.0/24	72.166.192.0/20
220.216.88.0/21	45.165.18.0/23	185.164.125.0/24
157.145.220.0/24	103.122.220.0/23	197.237.165.0/24
177.67.9.0/24	2a11:e487:cafe::/48	8.24.204.0/22
113.173.16.0/20	65.167.193.0/24	2402:3a80:1a52::/48
46.16.152.0/21	166.88.121.0/24	134.0.49.0/24
118.140.216.0/24	2402:8100:3979::/48	194.35.101.0/24
54.231.199.0/24	213.139.203.0/24	117.27.227.0/24

IP	IP	IP
116.85.20.0/24	147.203.32.0/19	165.84.212.0/22
2a10:e800:1::/48	2a02:88d:8134::/48	110.227.197.0/24
2803:7200:800c::/48	84.114.0.0/15	187.60.143.0/24
190.99.107.0/24	183.217.188.0/23	2602:fed2:730e::/48
109.199.252.0/24	45.224.198.0/23	173.211.80.0/21
2409:8a15:3a00::/39	172.225.98.0/24	180.213.12.0/24
205.151.208.0/23	74.222.145.0/24	213.137.137.0/24
162.211.233.0/24	2804:30d4::/32	124.192.128.0/18
140.106.94.0/24	211.28.128.0/19	109.203.107.0/24
107.186.154.0/24	133.226.96.0/19	8.29.195.0/24
45.182.108.0/22	149.200.212.0/22	190.84.136.0/23
2402:3a80:4058::/47	36.72.136.0/21	221.181.204.0/24
221.181.204.0/24	179.0.6.0/23	2806:370:91d0::/44
2409:8a3c:1a00::/40	45.228.177.0/24	185.176.172.0/22
178.173.13.0/24	108.34.128.0/17	3.3.28.0/22
194.33.45.0/24	2a01:9700:1f40::/44	172.111.206.0/24
240e:183:8102::/48	168.91.22.0/24	82.206.32.0/21
190.66.24.0/24	43.143.64.0/18	189.163.210.0/23
118.213.196.0/22	154.212.140.0/23	45.182.0.0/24
92.38.135.0/24	114.5.34.0/24	221.238.141.0/24
79.143.186.0/23	171.250.128.0/21	62.114.123.0/24
20.36.0.0/14	95.164.38.0/24	45.145.1.0/24
154.20.0.0/16	190.104.169.0/24	209.180.240.0/20

IP	IP	IP
125.62.70.0/24	199.20.7.0/24	201.131.247.0/24
201.197.139.0/24	2804:14d:5cc0:a000::/51	195.91.166.0/24
116.102.216.0/21	98.98.174.0/23	198.178.225.0/24
2a02:26f7:7e::/48	108.161.56.0/21	91.210.171.0/24
209.22.234.0/24	103.172.158.0/24	240a:6b:c200::/40
83.149.24.0/24	24.76.224.0/22	105.186.104.0/24
37.34.80.0/22	167.211.104.0/24	66.22.113.0/24
176.54.188.0/22	81.199.176.0/21	120.50.84.0/22
62.100.202.0/24	43.229.4.0/22	64.32.17.0/24
20.157.243.0/24	155.3.252.0/23	2402:3a80:1ee0::/48
199.204.95.0/24	165.16.160.0/19	46.236.64.0/18
104.198.96.0/19	2405:4803:dc78::/46	2a03:db02::/32
2a0b:6d80::/29	188.16.144.0/20	199.193.50.0/24
71.37.176.0/21	2407:97c0:8::/48	2409:8907:2520::/44
66.15.78.0/24	188.227.32.0/24	58.242.32.0/21
38.43.108.0/22	216.255.190.0/24	185.237.64.0/22
152.166.242.0/23	192.206.19.0/24	78.186.0.0/17
175.45.125.0/24	78.158.206.0/24	2804:87a4:8000::/34
42.106.192.0/22	84.255.164.0/22	154.56.88.0/24
170.150.219.0/24	2806:2f0:51a0::/43	201.33.0.0/24
2402:4000:11d0::/48	47.122.64.0/20	193.111.251.0/24
91.120.250.0/24	156.229.63.0/24	79.175.44.0/22
153.96.204.0/23	217.196.96.0/24	103.241.156.0/24

IP	IP	IP
66.228.0.0/20	95.185.54.0/23	125.208.21.0/24
85.192.50.0/23	2409:8959:40::/46	148.253.242.0/24
2001:16a2:c28f::/48	23.230.77.0/24	193.200.76.0/23
110.172.17.0/24	207.229.0.0/18	211.241.26.0/23
2607:2a00:2::/48	55.22.144.0/24	71.42.176.0/20
2001:dc7:ffd1::/48	200.39.249.0/24	213.188.216.0/23
36.159.54.0/23	126.25.0.0/16	2804:214:82ae::/49
185.141.238.0/23	202.168.6.0/24	1.47.64.0/21
209.14.145.0/24	2806:260:1013::/48	173.231.232.0/21
177.72.160.0/23	2806:2f0:9bc2::/48	206.214.37.0/24
2a02:bf0:4000::/36	128.1.57.0/24	177.55.224.0/20
2a00:1670::/32	138.99.16.0/23	45.84.219.0/24
138.182.0.0/16	240d:c010:ba::/48	2a0c:fc00::/32
185.165.56.0/24	38.35.32.0/19	45.174.169.0/24
51.252.66.0/23	2a01:8640:13::/48	81.16.141.0/24
168.199.41.0/24	2001:1248:9876::/48	2606:5e83:f000::/36
103.111.124.0/24	179.0.41.0/24	208.94.112.0/22
105.196.16.0/22	103.177.89.0/24	202.179.240.0/20
55.91.155.0/24	94.46.8.0/22	83.166.176.0/24
2001:67c:1b8c::/48	2001:dce:3::/48	185.66.48.0/22
123.208.110.0/23	2804:f7c:5000::/36	35.195.16.0/20
2a04:4e40:9c00::/48	62.150.19.0/24	2600:1fa0:8050::/46
203.142.208.0/21	46.31.150.0/24	2a0d:2900:d::/48

IP	IP	IP
185.56.218.0/24	147.28.184.0/23	240a:a2c5::/32
115.69.116.0/22	112.35.116.0/22	2001:ed0::/32
120.238.144.0/24	156.225.15.0/24	177.38.28.0/22
103.173.219.0/24	2600:6c38:1a7::/48	190.207.32.0/19
98.73.232.0/23	185.100.32.0/22	172.120.41.0/24
2a01:c50f:9ac0::/42	3.228.127.0/24	52.219.129.0/24
186.224.80.0/21	43.251.254.0/24	154.22.146.0/24
181.66.158.0/24	2405:84c0:4000::/36	181.208.53.0/24
87.68.192.0/20	159.142.146.0/24	80.240.35.0/24
181.191.194.0/23	2804:3f58:8000::/33	2804:214:86a0::/44
2408:8406:3d70::/44	176.62.44.0/22	198.57.12.0/24
23.245.52.0/23	203.0.241.0/24	188.225.138.0/24
149.240.224.0/20	207.34.33.0/24	99.198.71.0/24
103.230.62.0/24	201.194.208.0/22	186.38.47.0/24
2.18.106.0/23	5.101.185.0/24	119.46.8.0/21
42.114.112.0/24	2401:4900:369d::/48	104.143.67.0/24
160.101.131.0/24	2001:67c:bf4::/48	88.118.172.0/22
116.0.0.0/21	223.93.128.0/18	184.63.2.0/24
116.99.168.0/21	2407:e1c0:33::/48	2407:bc40::/32
45.235.56.0/22	45.41.184.0/23	185.178.219.0/24
95.38.39.0/24	111.118.69.0/24	240e:3a0:5019::/48
2402:f840:42::/48	190.106.7.0/24	195.216.219.0/24
223.187.178.0/24	152.96.0.0/16	186.85.196.0/24

IP	IP	IP
103.144.9.0/24	125.213.206.0/24	186.81.60.0/22
74.217.171.0/24	160.119.2.0/23	120.136.128.0/22
177.129.229.0/24	192.193.176.0/24	223.108.208.0/20
110.234.37.0/24	66.175.204.0/23	38.43.252.0/24
120.138.12.0/24	86.39.64.0/20	185.14.172.0/24
27.7.32.0/21	83.215.0.0/16	185.156.194.0/24
2804:5494:2000::/36	157.51.128.0/17	91.238.156.0/22
24.71.148.0/22	152.199.21.0/24	2600:1419:c400::/48
95.79.64.0/22	195.209.191.0/24	149.157.0.0/16
193.115.32.0/19	192.88.32.0/24	194.69.30.0/24
102.213.74.0/23	103.78.151.0/24	79.180.196.0/24
89.130.20.0/22	120.197.17.0/24	178.18.15.0/24
174.199.96.0/19	45.187.241.0/24	62.84.224.0/20
125.18.105.0/24	117.102.26.0/24	201.174.49.0/24
206.222.64.0/19	27.122.16.0/24	45.64.78.0/24
201.120.74.0/24	2401:4900:5030::/48	2c0f:f528:8::/48
197.132.180.0/22	2a0d:2681:a00::/40	117.240.39.0/24
146.112.208.0/24	70.16.208.0/24	222.255.183.0/24
200.130.16.0/24	89.212.192.0/18	121.198.0.0/16
58.177.176.0/21	206.84.179.0/24	185.47.5.0/24
139.15.200.0/24	2409:8904:1730::/44	2402:8100:2064::/48
45.207.13.0/24	2001:559:82db::/48	200.24.114.0/23
94.102.128.0/20	121.51.20.0/23	45.145.132.0/22

IP	IP	IP
185.245.216.0/24	38.91.49.0/24	103.103.99.0/24
69.73.72.0/24	2604:cb00:16::/48	196.191.128.0/20
44.4.17.0/24	185.35.172.0/22	189.45.245.0/24
180.191.195.0/24	185.211.64.0/24	104.125.15.0/24
50.205.51.0/24	81.219.0.0/16	122.170.9.0/24
160.20.200.0/22	103.243.149.0/24	198.153.154.0/24
208.76.180.0/24	41.69.112.0/20	240e:2c:8000::/34
185.61.184.0/22	80.80.184.0/21	61.69.194.0/24
81.18.216.0/21	190.2.111.0/24	211.255.108.0/24
85.157.0.0/16	185.177.196.0/24	91.226.248.0/23
2001:1248:4307::/48	120.227.220.0/23	193.17.220.0/24
17.57.164.0/23	185.248.234.0/23	2604:6840:ac12::/48
120.198.175.0/24	2a02:26f7:cdc0::/48	45.114.50.0/24
182.156.235.0/24	177.128.132.0/24	212.112.100.0/24
203.158.192.0/24	115.40.221.0/24	104.244.180.0/22
38.50.107.0/24	65.228.170.0/24	103.94.133.0/24
2a02:1140:113::/48	91.217.26.0/24	23.158.16.0/24
103.162.241.0/24	189.124.14.0/24	86.38.2.0/24
66.210.135.0/24	194.88.243.0/24	63.247.107.0/24
20.157.44.0/24	103.121.121.0/24	91.195.200.0/24
217.9.8.0/24	203.226.20.0/24	151.124.251.0/24
186.147.64.0/22	85.29.154.0/24	185.156.254.0/23
94.73.175.0/24	115.238.184.0/24	170.130.148.0/22

IP	IP	IP
61.56.244.0/22	179.50.136.0/22	193.46.237.0/24
176.236.186.0/24	2803:9800:b841::/48	192.93.158.0/24
205.70.231.0/24	240e:980:2700::/40	202.255.237.0/24
125.17.74.0/24	190.88.28.0/23	170.83.146.0/23
177.248.16.0/22	2401:4900:30c8::/45	192.48.8.0/24
67.197.13.0/24	31.130.120.0/21	122.160.110.0/24
190.25.181.0/24	2409:8920:3400::/40	37.111.60.0/24
105.158.112.0/21	167.249.178.0/23	179.0.74.0/24
107.6.246.0/24	2606:ef40:4200::/40	2408:8957:1680::/42
91.203.180.0/24	203.27.118.0/24	45.170.202.0/23
103.11.60.0/24	195.234.130.0/24	173.222.105.0/24
87.255.130.0/23	200.122.164.0/24	2804:8128::/32
143.208.7.0/24	98.164.62.0/24	2409:886c::/32
2402:3a80:1250::/44	2602:fd15:4::/48	211.156.198.0/24
207.38.88.0/24	2a0b:f300:44b::/48	223.178.124.0/22
166.173.192.0/18	2605:b080::/32	45.125.222.0/24
2a05:d050:2084::/46	75.159.0.0/16	27.46.78.0/24
195.55.104.0/22	194.127.88.0/24	216.255.160.0/20
103.12.241.0/24	166.1.219.0/24	45.95.204.0/24

APÊNDICE F – CÓDIGO PYTHON PARA CONTAGEM E ORDENAÇÃO DE IPS DE ROTEADOR

Código-fonte 1: Código Python para contagem e ordenação de IPs de roteador

```
1
2  import pandas as pd
3
4  # Caminho para o arquivo de entrada
5  input_file = 'Diretorio'
6
7  # Lista para armazenar as informacoes
8  data = []
9
10 # Lendo o arquivo linha por linha
11 with open(input_file, 'r') as file:
12     for line in file:
13         # Dividindo a linha pelos delimitadores "|"
14         fields = line.strip().split('|')
15
16         # Verificando se a linha possui o numero esperado
               de campos
17         if len(fields) >= 9:
18             # Extraindo o IP do roteador (campo 4)
19             ip_roteador = fields[3]
20
21             # Adicionando o IP a lista de dados
22             data.append({'IP Roteador': ip_roteador})
23
24 # Criando um DataFrame a partir da lista de dados
25 df = pd.DataFrame(data)
26
27 # Salvando a lista completa de IPs em um CSV
```

```
output_csv_full = 'C:\\Users\\dalia\\Downloads\\
    tabela_ips_completa.csv'
df.to_csv(output_csv_full, index=False)
print(f'Tabela completa de IPs salva em {output_csv_full}')

# Contando a frequencia de cada IP de roteador, ordenando
    por frequencia e salvando em outro CSV
ip_counts = df['IP Roteador'].value_counts().reset_index()
ip_counts.columns = ['IP Roteador', 'Frequencia']
ip_counts = ip_counts.sort_values(by='Frequencia',
    ascending=False)  # Ordenando do maior para o menor

# Salvando o resultado da contagem e ordenacao
output_csv_counts = 'Diretorio'
ip_counts.to_csv(output_csv_counts, index=False)
print(f'Frequencia de IP Roteador salva em {
    output_csv_counts}')
```

APÊNDICE G – CÓDIGO PYTHON PARA CONTAGEM E ORDENAÇÃO DE IPS DE ROTEADOR

Código-fonte 2: Implementação de *Patricia Trie* para roteamento IP

```
from collections import defaultdict, deque

# Estrutura Patricia Trie para armazenamento das rotas
class PatriciaTrieNode:
    def __init__(self):
        self.children = {}
        self.is_end = False
        self.prefix = None

class PatriciaTrie:
    def __init__(self):
        self.root = PatriciaTrieNode()

    def insert(self, prefix):
        node = self.root
        for bit in prefix:
            if bit not in node.children:
                node.children[bit] = PatriciaTrieNode()
            node = node.children[bit]
        node.is_end = True
        node.prefix = prefix

    def search(self, prefix):
        node = self.root
        for bit in prefix:
            if bit in node.children:
                node = node.children[bit]
            else:
                return None
        return node.prefix if node.is_end else None

# Funcao para carregar os nos especiais do arquivo CSV
def load_special_nodes(file_path):
    special_nodes = {}
```

```
35      with open(file_path, 'r') as file:
36          next(file)  # Pula o cabecalho
37          for line in file:
38              # Usa virgula como delimitador para separar IP e
                    frequencia
39              ip, frequency = line.strip().split(',')
40              special_nodes[ip] = int(frequency)
41      return special_nodes
42
43  # Exemplo de aplicacao
44  if __name__ == "__main__":
45      special_nodes_file = 'Diretoirio'
46      non_special_nodes = [
47          "179.0.6.0/23", "2806:370:91d0::/44", "2409:8a3c:1a00::/40",
48          "45.184.145.15","45.184.144.102", "45.184.146.59", "
                45.184.144.128",
49          # Continue a lista conforme necessario...
50      ]
```