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

FRANCISCO WALLISON CARLOS ROCHA

SimEDaPE: uma técnica para acelerar simulações de cidades inteligentes baseada na exploração de padrões de mobilidade urbana

São Paulo
2023

FRANCISCO WALLISON CARLOS ROCHA

SimEDaPE: uma técnica para acelerar simulações de cidades inteligentes baseada na exploração de padrões de mobilidade urbana

Texto de Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Orientador: Prof. Dr. Daniel Cordeiro

São Paulo

2023

Ficha catalográfica elaborada pela Biblioteca da Escola de Artes, Ciências e Humanidades,
com os dados inseridos pelo(a) autor(a)
Brenda Fontes Malheiros de Castro CRB 8-7012; Sandra Tokarevicz CRB 8-4936

Carlos Rocha, Francisco Wallison
SimEDaPE: uma técnica para acelerar simulações de cidades inteligentes baseada na exploração de padrões de mobilidade urbana / Francisco Wallison Carlos Rocha; orientador, Daniel de Angelis Cordeiro. -- São Paulo, 2023.
114 p: il.

Dissertacao (Mestrado em Ciencias) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, 2023.
Versão corrigida

1. Cidades Inteligentes. 2. InterSCSimulator. 3. SimPoint. 4. Simulação. I. Cordeiro, Daniel de Angelis, orient. II. Título.

Dedico este trabalho à minha mãe, Francisca Aurineide Rocha, ao meu pai, Francisco Gilson Carlos, ao meu irmão Francisco Álamo Carlos Rocha e a toda a minha família por sempre ter acreditado em mim.

Agradecimentos

Agradeço a Deus, uma força maior e/ou ao universo, por ter-me colocado as pessoas e as situações perfeitas para a chegada deste momento.

Agradeço aos meus pais, Francisca Aurineide Rocha e Francisco Gilson Carlos, especialmente a minha mãe, por está comigo sempre e me ajudando, apoiando e acreditando em mim. Ao meu irmão, Francisco Álamo Carlos Rocha, por sempre acreditar em mim e me apoiar nessa trajetória. Aos meus avós Francisca Eduardo e Agostinho Rocha por serem como pais para mim, sempre me apoiando e me ajudando. A toda minha família, por acreditar e me apoiar durante todo o curso.

Agradeço aos meus dois professores, orientadores e amigos, Prof. Dr. Daniel de Angelis Cordeiro e Prof. Dr. Emilio de Camargo Francesquini, por terem me dado a oportunidade, acreditado no potencial e no meu trabalho. Seus ensinamentos, direcionamentos e contribuições para esta pesquisa foram essenciais para minha formação e para este trabalho. Agradeço também pelos momentos e risadas que proporcionaram momentos alegres durante nossas reuniões e viagens.

Agradeço a todos os meus amigos que estiveram me motivando, acreditando em mim ou que de algum modo contribuíram para a conclusão deste trabalho.

Agradeço aos meus professores e amigos da UFC, Prof. Dr. Markos Oliveira Freitas e o Ms. Marcos Vinícius de Andrade Lima, por terem me incentivado a fazer o mestrado, acreditando no meu potencial.

Aos meus professores, por todo conhecimento compartilhado durante todo o curso, e por todo empenho e dedicação na minha formação técnico-humana. Agradeço ao Programa de Pós-graduação em Sistema de Informação (PPgSI) e a todos que o compõem, pela oportunidade de fazer parte como aluno de mestrado. Agradeço também aos meus colegas de classe, por todas as experiências e conteúdos trocados.

Agradeço ao projeto do InterSCity, projeto de pesquisa colaborativo sediado pelo Instituto Nacional de Ciência e Tecnologia (INCT) ao qual faço parte e recebo apoio e contribuições científicas. Agradeço também às entidades de fomento a pesquisa, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), que financia o projeto do InterSCity e essa pesquisa.

Agradeço aos membros da banca examinadora, por terem aceitado participar da avaliação deste trabalho e pelas sugestões e contribuições para melhoria desta pesquisa.

Finalmente, agradeço a todos os anônimos que me acompanharam e que de alguma forma contribuíram para a conclusão deste trabalho e para me tornar uma pessoa melhor.

"Se fosse fácil achar o caminho das pedras, tantas pedras no caminho não seria ruim."

(Humberto Gessinger)

Resumo

O crescimento populacional irá acarretar no agravamento de diversos problemas já enfrentados nos dias atuais. Megalópoles como São Paulo, Tóquio, Rio de Janeiro, Nova York, Paris, entre outras, já sofrem com problemas de poluição, tráfego, transporte público, serviços de saúde e educação. Tecnologias criadas no contexto de cidades inteligentes podem solucionar ou mitigar tais problemas, mas tais soluções são difíceis (ou caras) de serem avaliadas na escala de uma cidade. Mesmo técnicas computacionais como simulações podem necessitar de muita capacidade de processamento e memória para a simulação de cenários realísticos para megalópoles como São Paulo. Este trabalho propõe um método para a caracterização automática de cenários recorrentes em simulações de cidades inteligentes para acelerar estas simulações.

Palavras-chaves: Cidades Inteligentes; *InterSCSimulator*; *SimPoint*; Simulação.

Abstract

Population growth will worsen several problems already faced today. Megalopolises such as São Paulo, Tokyo, Rio de Janeiro, New York, Paris, among others, already suffer from pollution, traffic, public transport, health and education services. Technologies created in the context of smart cities can solve or mitigate such problems, but such solutions are difficult (or expensive) to assess on a city scale. Even computational techniques such as simulations may require a lot of processing capacity and memory to simulate realistic scenarios for megacities like São Paulo. This work proposes a method for the automatic characterization of recurring scenarios in simulations of smart cities to accelerate these simulations.

Keywords: Smart City; InterSCSimulator; SimPoint; Simulation.

Lista de figuras

Figura 1 – Arquitetura do InterSCSimulator. 24
Figura 2 – Mapa da região da Av. Paulista na cidade de São Paulo-SP com a representação de um viagem Origem-Destino destacada pela cor vermelha. 25
Figura 3 – Componentes do InterSCSimulator. 25
Figura 4 – Uso de recurso durante uma simulação feita pelo *InterSCSimulator*. . . 30
Figura 5 – *Speedup* para execução de 3 cenários em 1, 2 e 3 processos. 30
Figura 6 – Aplicação da técnica *SimPoint* para agrupar e extrair representantes na aplicação *gzip*. 32
Figura 7 – Contagem de execução dos blocos básicos em 3 intervalos de um programa qualquer. 32
Figura 8 – *Basic Block Vectors* gerados a partir da contagem da execução dos blocos básicos A, B, C, D e E nos três intervalos de execução *Interval 1*, *Interval 2* e *Interval 3* da Figura 7. 33
Figura 9 – Cálculo da distância de *Manhattan* dos *Basic Block Vectors* dos três intervalos de execução *Interval 1*, *Interval 2* e *Interval 3* de um programa qualquer. 33
Figura 10 – Alinhamento usando a distância de *Manhattan* (10a) e usando a distância DTW (10b). 36
Figura 11 – Caminho deformado com as menores distâncias e associações entre as duas séries. 37
Figura 12 – Matriz de distâncias calculadas pela DTW entre duas séries. 38
Figura 13 – Cálculo da correlação cruzada de duas séries s e t. 39
Figura 14 – Fluxo de artigos desde a fase de aplicação das *strings* de busca (aplicadas na primeira pesquisa) até a fase de extração com a aplicação dos critérios de inclusão e exclusão. 46
Figura 15 – Fluxo de artigos desde a fase de aplicação das *strings* de busca (aplicadas na segunda pesquisa) até a fase de extração com a aplicação dos critérios de inclusão e exclusão. 46
Figura 16 – Publicações X Ano obtidas com a aplicação das *strings* de busca da primeira pesquisa. 47

Figura 17 – Publicações X Ano obtidas com a aplicação das *strings* de busca da segunda pesquisa. 47
Figura 18 – Porcentagem de trabalhos categorizados por objetivo até o ano de 2020 (18a) e a partir de 2020 (18b). 48
Figura 19 – Porcentagem de trabalhos categorizados por tipo até o ano de 2020 (19a) e a partir de 2020 (19b). 49
Figura 20 – Etapas da execução da SimEDaPE. 59
Figura 21 – Comparação de séries em relação a escala. 62
Figura 22 – Mapa de calor de parte uma cidade cidade, onde as regiões com cores mais quentes mostram ruas com maior quantidade de veículos. 63
Figura 23 – Séries agrupadas no mesmo *cluster* 86 usando o algoritmos de clusterização *K-shape* 64
Figura 24 – Matriz calculada usando DTW (24a) e mapeamento (*Warping Path*) resultante (24b). 66
Figura 25 – Mapeamento usando a distância Euclidiana (25a) e mapeamento (*Warping Path*) usando a DTW (25b). 66
Figura 26 – Tempo gasto na execução das principais etapas da SimEDaPE. 80
Figura 27 – Resultados para estimativas de velocidades médias das simulações parciais usando somente dados de normalização da simulação base. 83
Figura 28 – Resultados para estimativas de velocidades médias das simulações parciais usando somente dados de normalização das simulações parciais. . 84
Figura 29 – Resultados para estimativas de velocidades médias das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais. 85
Figura 31 – Resultados para estimativas de velocidades médias das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais. 87
Figura 32 – Comparação das estimativas da métrica de Velocidade Média realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização. 88
Figura 33 – Resultados para estimativas da Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando somente dados de normalização da simulação base. 90

Figura 34 – Resultados para estimativas da Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando somente dados de normalização da simulação parcial. 91
Figura 35 – Resultados para estimativas de Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais. 92
Figura 36 – Resultados para estimativas de Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais. 94
Figura 37 – Comparação das estimativas da métrica de Porcentagem da Taxa de Ocupação das Vias realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização. 95
Figura 38 – Resultados para estimativas da Média de Veículos das simulações parciais usando somente dados de normalização da simulação base. 96
Figura 39 – Resultados para estimativas da Média de Veículos das simulações parciais usando somente dados de normalização da simulação parcial. 97
Figura 40 – Resultados para estimativas de Média de Veículos das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais. 99
Figura 41 – Resultados para estimativas de Média de Veículos das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais. 100
Figura 42 – Comparação das estimativas da métrica de Porcentagem da Taxa de Ocupação das Vias realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização. 101

Listagens

1 Exemplo de um arquivo config.xml. 25

2 Exemplo de um arquivo map.xml 26

3 Exemplo de um arquivo trips.xml. 27

4 Exemplo de um arquivo de metro.xml. 27

5 Exemplo de um arquivo buses.xml. 28

6 Exemplo de arquivo com a saída do simulador contendo um conjunto de eventos. 28

7 Arquivo com os eventos gerados pelo *InterSCSimulator*. 60

Lista de algoritmos

Algoritmo 1 – *Dynamic Time Warping* 37
Algoritmo 2 – Max 69
Algoritmo 3 – Seleção Aleatória de Viagens Com Base nos *Simulation Points* 71
Algoritmo 4 – Seleção de Viagens Aleatórias Divididas por *Cluster* Com Base nos *Simulation Points* 71

Lista de tabelas

Tabela 1 – Cenários simulados e estimativa 29
Tabela 2 – Tabela com os critérios de inclusão (CI's) e com os critérios de exclusão (CE's). 43
Tabela 3 – Tabela com as *strings* de busca associadas às bibliotecas digitais aplicadas na primeira pesquisa. 44
Tabela 4 – Tabela com as *strings* de busca associadas às bibliotecas digitais as quais foram aplicadas na segunda pesquisa. 45
Tabela 5 – Sumarização dos trabalhos selecionados na etapa de extração até o ano de 2020. 48
Tabela 6 – Sumarização dos trabalhos selecionados na etapa de extração a partir do ano de 2020 com alguns de 2019. 49
Tabela 7 – Erro da estimativa da mesma simulação completa sendo estimada usando a SimEDaPE. 82

Sumário

1 Introdução . . . 17

1.1 Objetivos . . . 20

1.2 Justificativa . . . 20

1.3 Organização . . . 21

2 Fundamentação Teórica . . . 22

2.1 InterSCSimulator . . . 22

2.1.1 Arquitetura . . . 23

2.1.2 Dados de Entrada . . . 23

2.1.3 Desempenho . . . 29

2.2 SimPoint . . . 31

2.2.1 Extração dos *Basic Block Vectors* . . . 31

2.2.2 Agrupamento com *K-Means* . . . 34

2.3 Dynamic Time Warping . . . 35

2.4 K-Shape . . . 38

2.4.1 *Shape Based Distance* . . . 39

2.4.2 *Extração dos Centroides* . . . 41

3 Revisão da literatura . . . 42

3.1 Protocolo Resumido . . . 42

3.2 Análise Geral . . . 44

3.3 Síntese dos Trabalhos Selecionados . . . 49

3.3.1 Simulação de Tráfego . . . 50

3.3.2 Elementos Finitos e Dinâmica dos Fluidos . . . 52

3.3.3 Internet das Coisas . . . 53

3.3.4 Outros Trabalhos . . . 54

3.3.5 Considerações Finais . . . 55

4 Metodologia . . . 57

4.1 SimEDaPE: Simulation Estimation by Data Patterns Exploration . . . 57

4.1.1 Extração do *Dataset* . . . 60

4.1.2 Normalização do *Dataset* 61
4.1.3 Identificação e Agrupamento de Comportamentos 62
4.1.4 Seleção dos Representantes Únicos 64
4.1.5 Cálculo do Mapeamento (*Warping Path*) 65
4.1.6 Execução da Simulação Parcial 67
4.1.7 Estimativa e Extração de Métricas 72
4.2 Outras Propostas e Implementações de Melhoria 76

5 Experimentos e Resultados 79
5.1 SimEDaPE 79
5.1.1 Velocidade Média 82
5.1.2 Porcentagem da Taxa de Ocupação das Vias 89
5.1.3 Média de Veículos 96
5.2 Considerações Finais 102

6 Considerações Finais 104
6.1 Trabalhos Futuros 105
6.2 Publicações Resultantes Deste Trabalho 106
6.3 Financiamento 107

Referências[1] 108

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

1 Introdução

No dia 15 de novembro do ano de 2022, o mundo chegou a marca impressionante de 8 bilhões de pessoas. A organização das Nações Unidas mostra que esse número expressivo de pessoas é devido o aumento da expectativa de vida das pessoas. Com uma estimativa de que até 2037 haverá cerca de 9 bilhões de pessoas[1]. Nessa perspectiva, Boretti e Rosa (2019) mostra uma projeção que até o ano de 2050 o número total de pessoas no planeta atingirá um valor entre 9,4 e 10,2 bilhões. Embora esse crescimento está diretamente ligado ao aumento da expectativa de vida e melhoria de saúde pública. O crescimento populacional irá acarretar no agravamento de diversos problemas já enfrentados nos dias atuais. Megalópoles como São Paulo, Tóquio, Rio de Janeiro, Nova York, Paris, entre outras, já sofrem com problemas de poluição, tráfego, transporte público, serviços de saúde e educação (SANTANA *et al.*, 2018).

A resolução desses problemas apresentados nesse contexto não se dá de maneira trivial, e na maioria das vezes requerem soluções inovadoras. Essas soluções por sua vez acabam se tornando um desafio devido ao um alto custo de implantação, falta de infraestrutura e questões governamentais. Nessa perspectiva, em seu trabalho, Santana *et al.* (2018) afirma que muitas das ideias planejadas nesse contexto ainda não estão implementadas em grandes centros urbanos como o da grande São Paulo com mais de 11 milhões de habitantes devido a sua grande complexidade.

Uma alternativa para a resolução desses problemas seria o uso de aplicações e tecnologias desenvolvidas para Cidades Inteligentes (do inglês *Smart Cities* ou *Digital Cities*) (NAM; PARDO, 2011). Existem várias definições na literatura para o termo *"Smart City"* como é visto no trabalho apresentado por Nam e Pardo (2011). Uma dessas definições é expressa por Yovanof e Hazapis (2009). Yovanof e Hazapis (2009) definem uma Cidade Inteligente como um conjunto de aplicações e tecnologias que melhoram a vida dos habitantes de uma cidade por meio da simplificação de transações públicas, redução de custos de telecomunicações e oferecimento de vários serviços para atender as necessidades dos usuários finais, seja na área da saúde, educação, transportes, lazer ou segurança.

No entanto, nem sempre a implantação das soluções para estes problemas enfrentados se dá de maneira trivial. Por não ser de simples criação e implantação, acaba se tornando um desafio dentro do contexto de Cidades Inteligentes devido ao seu alto custo, falta de

[1] https://www.un.org/en/dayof8billion

infraestrutura e questões governamentais (SANTANA *et al.*, 2018). Nessa perspectiva, em seu trabalho, Santana *et al.* (2018) afirmam que muitas das ideias planejadas nesse contexto ainda não estão implementadas em grandes cidades como São Paulo, com mais de 11 milhões de habitantes, devido a grande dimensão dessas cidades corroboram com o aumento da complexidade dessas ideais. Para auxiliar na experimentação dessas diferentes ideias, Santana *et al.* (2018) propuseram o *InterSCSimulator*, que é um simulador de tráfego urbano. Com a realização de simulações em larga escala nesse simulador é possível validar e testar essas soluções antes de construí-las.

No entanto, simulações em larga escala podem se tornar um desafio devido à grande demanda por poder e tempo computacional. Uma das causas dessa demanda é a quantidade de agentes (pedestres e meios de transportes) envolvidos na simulação, como ocorre na simulação de cenários sobre a cidade de São Paulo. Uma pesquisa feita pelo Instituto Brasileiro de Geografia e Estatística (IBGE)[2] estima que a cidade apresenta mais de 12 milhões de habitantes no ano de 2020 e com mais 8 milhões de veículos em uma pesquisa[3] feita até o ano de 2018. Já a grande e populosa cidade de Tóquio no Japão apresenta uma quantidade de aproximadamente 38 milhões de pessoas[4]. Em contraste com esse número de agentes, o *InteSCSimulator* só é capaz de executar uma simulação com apenas de 4 milhões de agentes simultâneos (SANTANA *et al.*, 2018).

O *InterSCSimulator* é um simulador de tráfego urbano. Ele simula um mapa de uma cidade ou região com viagens de carros, pedestres, ônibus e metrô. Essas viagens partem de um ponto de origem a um ponto de destino no mapa. Ele foi desenvolvido a partir de um simulador de eventos discretos chamado Sim-Diasca, que foi proposto por Song *et al.* (2011). Ambos os simuladores foram desenvolvidos na linguagem de programação funcional Erlang[5]. Ela por sua vez, é uma linguagem implementada baseada em atores, o que propicia a construção de aplicações paralelas e distribuídas com uma maior facilidade.

Ao observar os experimentos realizados por Santana *et al.* (2018) (mostrados em detalhes na Seção 2.1.3), nota-se, entretanto, que o uso do modelo de atores distribuídos ou paralelos de Erlang não é o suficiente para garantir o desempenho necessário para simular cenários complexos como o da cidade de São Paulo. Em tais cenários, o uso do

[2] IBGE. População da Cidade de São Paulo. ⟨https://www.ibge.gov.br/cidades-e-estados/sp/sao-paulo.html⟩

[3] IBGE. Frota de veículos da Cidade de São Paulo. ⟨https://cidades.ibge.gov.br/brasil/sp/sao-paulo/pesquisa/22/28120⟩

[4] United Nations. The world's cities report. ⟨http://wcr.unhabitat.org/⟩

[5] Erlang. ⟨https://www.erlang.org/⟩

simulador apresenta alguns desafios, como o uso excessivo de memória devido ao tamanho e complexidade desses cenários. No experimento realizado com mais de 4 milhões de agentes, o simulador usou cerca de 196 GB de memória RAM. Além disso, estima-se que para executar um cenário como o da cidade de São Paulo seriam necessários mais de 515 GB de memória (SANTANA *et al.*, 2018). Outro problema apresentado pelo *InterSCSimulator* é a má distribuição de carga entre os processos.

Santana *et al.* (2018) realizam um experimento com a finalidade de dividir as tarefas realizadas por 45 mil agentes entre 3 processos em memória distribuída. A partir desse experimento é possível extrair informações a respeito de seu desempenho.

O esperado nesse experimento era que o *speedup* fosse 2 para 2 processos, no entanto, o valor ficou em torno de aproximadamente 1,17. No caso de 3 processos, o valor esperado era 3, porém, ficou em torno de 1,48, o que não é o ideal. Além desses dois problemas, também é identificado um outro problema relacionado ao uso da CPU. Nesse caso, a CPU é usada cerca de somente 50% em todo o tempo de execução da aplicação. Desse modo, o simulador está usando somente metade de todos os recursos de processamento disponíveis, o que não é aceitável (SANTANA *et al.*, 2018).

Diante desses problemas, uma alternativa para melhorar a escalabilidade do *InterSCSimulator* é o uso de uma técnica como a *SimPoint* proposta por Hamerly *et al.* (2005). A aplicação desta técnica tem por finalidade permitir a simulação em larga escala de grandes cenários. No entanto, na literatura não foram encontrados trabalhos com a aplicação desse conceito em simuladores de cidades inteligentes. Esse mesmo problema é enfrentado por pesquisadores de outras áreas que testam suas soluções por meio do uso de simulações. O uso de técnicas de simulação aproximada como a *SimPoint* também são aplicadas em outras áreas, como na pesquisa de *design* de *hardware*.

A pesquisa de *design* de *hardware* moderna requer a compreensão do comportamento do nível de ciclo de um processador durante a execução de um aplicativo. Geralmente pesquisadores utilizam de simuladores ricos em detalhes para modelar esses ciclos. No entanto, as simulações realizadas possuem um tempo de duração muito grande. As mesmas dificuldades com simulações em larga-escala são enfrentadas por pesquisadores de *hardware*.

A *SimPoint* é utilizada para estimar o resultado de grandes simulações. A partir de uma entrada do programa, são detectados comportamentos repetitivos que mudam de acordo com o tempo. A partir desses comportamentos, é possível identificar padrões que permitem reduzir o tempo de simulação ao selecionar um estado da simulação que

seja representativo para cada padrão (HAMERLY *et al.*, 2005). Para identificar esses padrões, é utilizado um algoritmo de agrupamento (ou do inglês *clustering*) para agrupar esses comportamentos. Após realizar o agrupamento, são selecionados representantes únicos que melhor representem os demais de cada grupo, os pontos de simulação. Esses representantes selecionados, são posteriormente utilizados para estimar o resultado dos demais presentes em cada grupo. Assim, tendo em vista que cada representante apresenta o mesmo comportamento de seus semelhantes, é possível estimar o resultado da simulação completa somente executando os pontos de simulação.

1.1 Objetivos

Para simular grandes cenários como o da cidade de São Paulo usando o *InterSCSimulator*, é preciso mitigar as limitações apresentadas pelo simulador. Então, este trabalho tem como objetivo melhorar o desempenho atual apresentado pelo *InterSCSimulator*.

Para alcançar o objetivo principal desse trabalho são definidos como objetivos específicos:

- avaliar o uso da téncnica *SimPoint* no contexto de cidades inteligentes;
- aplicar melhorias e implementações para mitigar ou resolver gargalos e eventuais problemas atrelados ao desempenho do simulador.

1.2 Justificativa

Como visto, os desafios já enfrentados pela atual população de grande cidades como São Paulo poderão ser agravados devido ao crescimento populacional. Para auxiliar profissionais e pesquisadores da área de cidades inteligentes a testarem suas soluções não-triviais, o uso de um simulador é de grande valia.

No entanto, alguns simuladores apresentados encontrados na literatura apresentam limitações e gargalos para simular grandes cenários. Como é o caso do simulador proposto por Santana *et al.* (2018). Assim, para viabilizar a execução desses grandes cenários, esse trabalho propõe uma nova técnica baseada na *SimPoint* para melhorar o uso de memória e o tempo de processamento do *InterSCSimulator*.

1.3 Organização

Este trabalho está organizado nos seguintes capítulos. No Capítulo 2, apresenta-se as terminologias, conceitos, algoritmos e técnicas usadas no restante desse trabalho ou se fazem necessárias para entendimento do mesmo. No Capítulo 3, discute-se sobre os trabalhos relacionados que de algum modo se relacionam com esta pesquisa. No Capítulo 4, discute-se os estudos realizados. No Capítulo 5, são mostrados os resultados dos experimentos realizados para medir o desempenho da metogologia aplicada neste trabalho para resolução do problema abordado. Por fim, no Capítulo 6, são apresentadas as principais contribuições e considerações a respeito deste trabalho até o momento.

2 Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos que de alguma forma estão relacionados com esta pesquisa. Inicialmente, são relatados os termos relacionados ao *InterSCSimulator* e a cidades inteligentes. O *InterSCSimulator* é o simulador de cidades inteligentes que é objeto de estudo deste trabalho. Esses termos são apresentados na Seção 2.1.

Em seguida são abordados os termos referentes às técnicas propostas no Capítulo 4 que são aplicadas com a finalidade de melhorar o *InterSCSimulator*. Na Seção 2.2 são apresentados os termos referentes a principal técnica proposta neste trabalho que é apresentada em detalhes na Seção 4.1. Dentre os termos relacionados a técnica propostas estão também os termos que são referentes a *SimPoint*.

2.1 InterSCSimulator

Proposto por Santana *et al.* (2018), o *InterSCSimulator* é um simulador de tráfego urbano. Ele é uma ferramenta utilizada para testar e validar soluções para resolução de problemas que podem ser solucionados no contexto de Cidades Inteligentes antes de pô-las em prática. Esse simulador foi desenvolvido a partir do simulador de eventos discretos Sim-Diasca (SONG *et al.*, 2011).

O Sim-Diasca é um motor (*engine*) de simulação proposto por Song *et al.* (2011). Implementado na linguagem de programação *Erlang*, o motor provê abstrações básicas para simulação usando o modelo de atores, o que facilita a execução dos atores de modo concorrente e distribuído. Além disso, ele facilita a implementação de novos simuladores feitos em Erlang, como o *InterSCSimulator*. Esse motor também provê facilidades para construção de relatórios e coleta de resultados da simulação.

Proposta e criada nos laboratórios da Ericsson por volta do ano de 1986 (ARMSTRONG, 2007), Erlang é uma linguagem funcional, de programação concorrente e baseada em atores. Por ser baseada em atores, Erlang propicia a criação de aplicações paralelas e distribuídas com uma maior facilidade. Além disso, ela também apresenta características como tolerância a falhas, concorrência com troca de mensagens assíncronas e fácil integração

com outras linguagens. Por isso, é uma linguagem propícia para criação de aplicações com resposta em tempo real, entre outras características (ARMSTRONG *et al.*, 1993).

A seguir o *InterSCSimulator* será mostrado em detalhes. Na Seção 2.1.1 é apresentada em detalhas a arquitetura do simulador. Na Seção 2.1.2 são mostrados os dados de entrada utilizados pelo *InterSCSimulator* para executar uma simulação de tráfego urbano. Por fim, na Seção 2.1.3 são mostrados os resultados dos experimentos realizados com o *InterSCSimulator* feitos por Santana (2019).

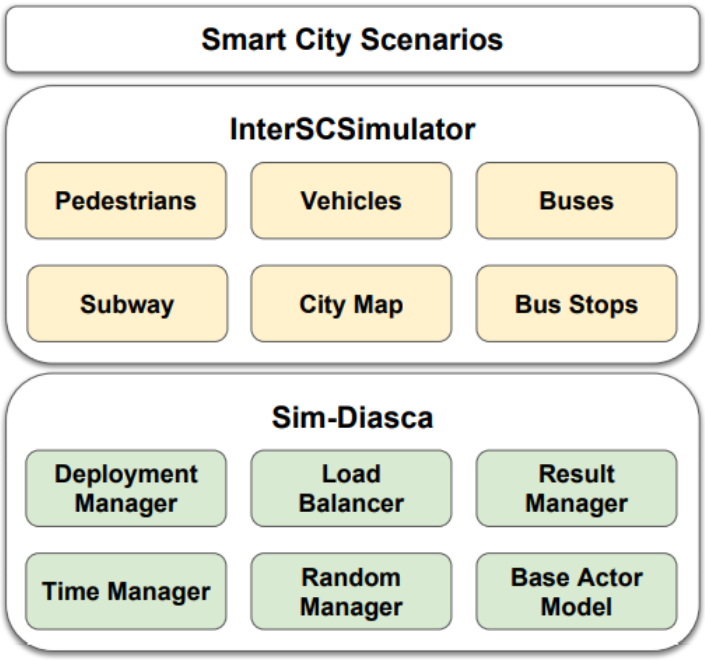
2.1.1 Arquitetura

Em seu trabalho, Santana (2019) apresenta a arquitetura do *InterSCSimulator* com o Sim-Diasca. Ilustrada na Figura 1, a arquitetura é dividida em três camadas: *Smart City Scenarios*, InterSCSimulator e Sim-Diasca. Na camada *Smart City Scenarios* estão representadas as descrições do cenário da simulação. Na camada InterSCSimulator estão descritas as entidades (*Subway*, *Car*, *Pedestrians*, *City Map*, *Buses* e *Bus Stops*) que compõem o cenário e o tipo de simulação que o simulador executa (simulação de tráfego urbano). Por fim, a camada do Sim-Diasca contém os componentes base para a execução das simulações como balanceador de carga, gerenciador de tempo, gerenciador de implantação, entre outros.

2.1.2 Dados de Entrada

Além da arquitetura, Santana (2019) descreve em seu trabalho a simulação realizada pelo *InterSCSimulator*. A simulação pode ser executada tanto em memória compartilhada quanto em memória distribuída, e utiliza como parte da entrada um mapa representado por um grafo $G(V, E)$ com um conjunto V de vértices que representam as conexões entre as ruas e avenidas e um conjunto E de arestas que representam as ruas e as avenidas em G. Além do mapa, o simulador também recebe o conjunto de viagens de pedestres, sistemas de ônibus, metrô e carros. Essas viagens partem de um vértice de origem VO (um ponto de origem no mapa) a um vértice de destino VD (um ponto de destino no mapa) no grafo G. A Figura 2 ilustra um mapa da região próxima da Avenida Paulista da Cidade de São Paulo com as ruas sendo as arestas e os vértices destacados pelos que as conecta

Figura 1 – Arquitetura do InterSCSimulator.



Fonte: (SANTANA, 2019).

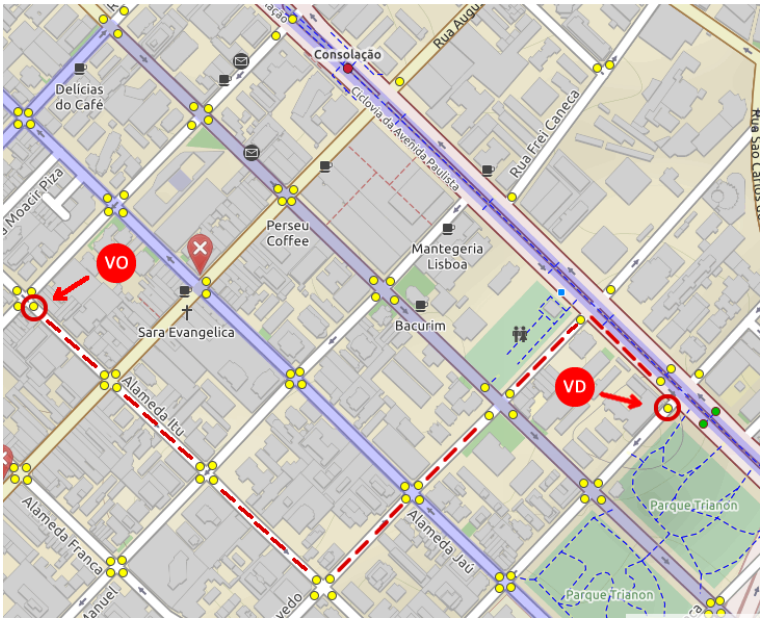
no mapa. Além disso, a figura também ilustra uma viagem partindo do VO seguindo o percurso tracejado (em vermelho) até o VD.

Os dados são representados por cinco arquivos de entrada no formato *Extensible Markup Language* (XML) descrevendo o cenário da simulação e um arquivo de saída XML ou *Comma Separated Values* (CSV) com os eventos gerados no decorrer da simulação seguindo o fluxo da Figura 3.

Os cinco arquivos de entrada descritos abaixo são: config.xml, map.xml, trips.xml, buses.xml e subway.xml.

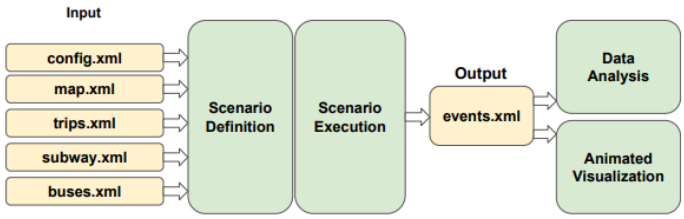
- config.xml: arquivo de configuração contendo informações para localização dos outros arquivos de entrada e o tempo de duração da simulação expresso em segundos. O conteúdo desse arquivo é mostrado na Listagem 1.
- map.xml: arquivo que contém as informações para construir o mapa da cidade ou região. Nele estão representados um conjunto de *nodes* (que representam o vértices) com suas respectivas coordenadas cartesianas (x, y). Além dos vértices representados pelos *nodes*, estão descritas as arestas representadas pelo conjunto de *links* com informações de qual vértice (*node*) de origem a aresta conecta a qual vértice de

Figura 2 – Mapa da região da Av. Paulista na cidade de São Paulo-SP com a representação de um viagem Origem-Destino destacada pela cor vermelha.



Fonte: Mapa retirado de ⟨https://www.openstreetmap.org/⟩ e adaptado por Francisco Wallison Carlos Rocha (2022).

Figura 3 – Componentes do InterSCSimulator.



Fonte: (SANTANA, 2019).

```
<scsimulator_config>
    <config
        trip_file="../base_scenario/trips.xml"
        map_file="../base_scenario/map.xml"
        bus_file="../base_scenario/buses.xml"
        subway_file="../base_scenario/metro.xml"
        output_file="../output/events.csv"
        simulation_time="86400"/>
</scsimulator_config>
```

Fonte: (SANTANA, 2019).
Listagem 1 – Exemplo de um arquivo config.xml.

destino e o tamanho da aresta em metros (m). Na Listagem 2, é mostrado o conteúdo do arquivo map.xml.

```
<network>
    <nodes>
        <node id="1" x="-7347433.28816257" y="-2852981.6323715686" >
        </node>
        ...
        <node id="3" x="-7330256.36170953" y="-2853959.906814551" >
        </node>
    </nodes>
    <links>
        <link id="12" from="1" to="2" length="83.11687370503387"
        ↪  capacity="3000.0" oneway="1"
        modes="car" ></link>
        ...
        <link id="23" from="2" to="3" length="83.11687370503387"
        ↪  capacity="3000.0" oneway="1"
        modes="car" ></link>
    </links>
</network>
```

Fonte: (SANTANA, 2019).
Listagem 2 – Exemplo de um arquivo map.xml

- trips.xml: lista de viagens que partem em um determinado instante de tempo (de 0 ao tempo máximo definido no arquivo config.xml) de um vértice de origem a um vértice de destino ambos representados no map.xml da Listagem 2. Além dos atributos de origem (*origin*) e destino (*destination*), as viagens também apresentam o identificador do agente (*name*) e quantidade (*count*) desse mesmo agente que será gerada em tempos inicio (*start*) diferentes. O conteúdo do arquivo de viagens trips.xml é mostrado na Listagem 3.

```
<scsimulator_matrix>
    <trip name="paraiso_regular_0" origin="1" destination="4"
    link_origin="12" count="1" start="1" mode="car"
    digital_rails_capable="false"/>
    <trip name="paraiso_regular_1" origin="1" destination="6"
    link_origin="12" count="1" start="1" mode="car"
    digital_rails_capable="false"/>
    <trip name="paraiso_bike_1" origin="1" destination="6"
    link_origin="12" count="1" start="1" mode="bike"
    digital_rails_capable="false"/>
</scsimulator_matrix>
```

Fonte: (SANTANA, 2019).
Listagem 3 – Exemplo de um arquivo trips.xml.

- subway.xml: listas de estações de metrô ou trem com os seus respectivos nomes associadas a um vértice no grafo que representa o mapa da cidade com suas respectivas conexões (*links*) com outras estações. Um exemplo do conteúdo do arquivo de subway.xml é mostrado na Listagem 4.

```
<metro>
    <stations>
        <station name="Vila Prudente" idNode="252018920" />
        <station name="Oratorio" idNode="2378322735" />
        ...
        <station name="Conceição" idNode="660766755" />
        <station name="São Judas" idNode="431359341" />
    </stations>
    <links>
        <link nameOrigin="Oratorio" nameDestination="Vila Prudente"
        ↪  idOrigin="2378322735" idDestination="252018920" />
        ...
        <link nameOrigin="Conceição" nameDestination="Jabaquara"
        idOrigin="660766755" idDestination="247909491" />
    </links>
</metro>
```

Fonte: (SANTANA, 2019).
Listagem 4 – Exemplo de um arquivo de metro.xml.

- buses.xml: este arquivo especifica as linhas de ônibus com seus respectivos pontos de paradas (vértices no grafo) e intervalos de desembarque. O arquivo buses.xml é mostrado na Listagem 5.

```
<scsimulator_buses>
    <bus id="1016-10-0" interval="1800" start_time="18000"
    stops="507969889, 2390204059, ..., 507969889" />
    ...
    <bus id="1016-10-1" interval="1800" start_time="18000"
    stops="2396517544, 163220296, ..., 2390192810" />
</scsimulator_buses>
```

Fonte: (SANTANA, 2019).
Listagem 5 – Exemplo de um arquivo buses.xml.

O arquivo de saída é chamado de `events`, que pode ser tanto no formato XML quanto no formato CSV. Ele contém informações de em que instante de tempo o evento aconteceu, qual o tipo de evento, o identificador da entidade (carro, ônibus, pedestre ou metrô) envolvido e em qual rua ou avenida o vento aconteceu. Veja a Listagem 6 que retrata um exemplo desse arquivo.

```
<events>
    <event time="7" type="actend" person="paraiso_regular_0_1"
    link="2105" actType="h" action="ok" />
    <event time="7" type="departure" person="paraiso_regular_0_1"
    link="2105" legMode="car" action="ok" />
    <event time="7" type="PersonEntersVehicle"
    ↪  person="paraiso_regular_0_1"
    vehicle="paraiso_regular_0_1" action="ok" />
    <event time="7" type="wait2link" person="paraiso_regular_0_1"
    link="2105" vehicle="paraiso_regular_0_1" action="ok" />
    <event time="7" type="entered link" person="paraiso_regular_0_1"
    link="2105" vehicle="paraiso_regular_0_1" action="ok" />
    <event time="7" type="actend" person="paraiso_regular_1_1"
    link="2105" actType="h" action="ok" />
    <event time="7" type="departure" person="paraiso_regular_1_1"
    link="2105" legMode="car" action="ok" />
</events>
```

Fonte: (SANTANA, 2019).
Listagem 6 – Exemplo de arquivo com a saída do simulador contendo um conjunto de eventos.

2.1.3 Desempenho

Através da observação dos experimentos realizados com o *InterSCSimulator* por Santana *et al.* (2018), é possível notar que o uso do modelo de atores distribuídos ou paralelos de Erlang não é o suficiente para garantir o desempenho necessário para simular cenários complexos como o da cidade de São Paulo. O *InterSCSimulator* apresentado por Santana *et al.* (2018) apresenta problemas e gargalos em sua execução. Esses problemas estão relacionados ao uso excessivo de memória, a má distribuição de tarefas entre processos em memória distribuída e uso de aproximadamente 50% da CPU.

Na Tabela 1 é mostrado o resultado do experimento de quatro simulações feito por Santana *et al.* (2018). Nesse experimento é possível notar que o simulador necessita de uma grande quantidade memória RAM. Essa quantidade de recurso de memória necessário pode se tornar impraticável para execução de cenários maiores como de megalópoles como a grande São Paulo. Como mostrado no trabalho de Santana *et al.* (2018), para executar o menor cenário (1) do experimento foram necessários 51 GB de RAM. Já para executar o maior cenário (4) foram necessários 196 GB de RAM. Além disso, em uma estimativa feita por Santana *et al.* (2018), para executar um cenário (5) como o da cidade São Paulo com mais 11 milhões de habitantes são necessários aproximadamente 515 GB de memória RAM.

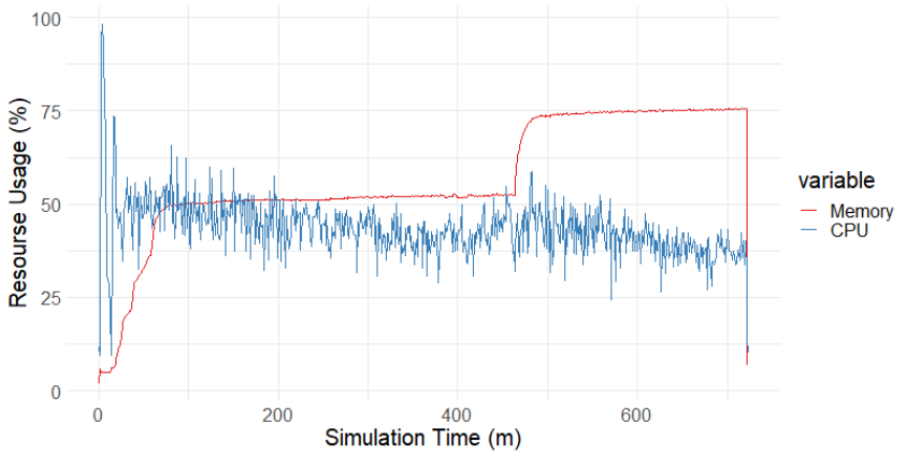
Tabela 1 – Cenários simulados e estimativa

	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Estimativa
Agentes	1 milhão	2 milhões	3 milhões	4 milhões	11 milhões
Tamanho do Mapa	50.000	50.000	50.000	50.000	140.000
Memória	51 GB	98 GB	142 GB	196 GB	515 GB
Tempo (Minutos)	22m	45m	70m	95m	460m
Eventos	70 milhões	140 milhões	210 milhões	280 milhões	910 milhões
Arquivo de Saída	2 GB	4.1 GB	6 GB	8.3 GB	23 GB

Fonte: Adaptado de Santana *et al.* (2018).

Outro problema encontrado no *InterSCSimulator* é o problema do uso de 50% da CPU. Nesse problema, o simulador apenas usa em torno da metade dos recursos de processamento destinado a ele, como é possível ver no gráfico mostrado na Figura 4.

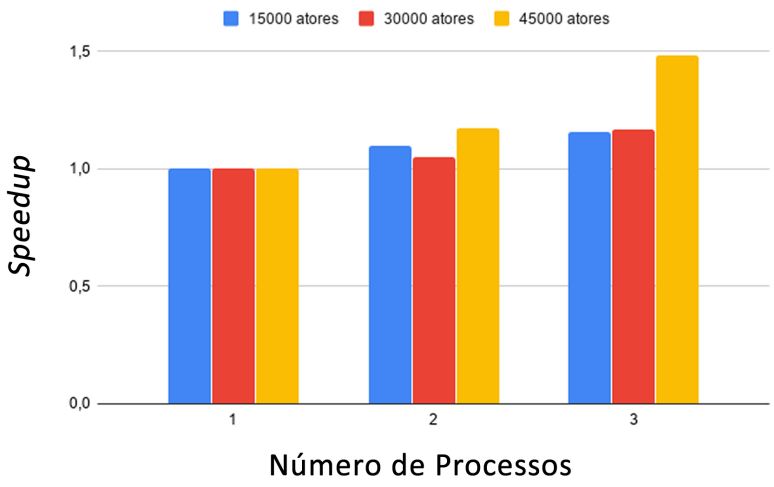
Figura 4 – Uso de recurso durante uma simulação feita pelo *InterSCSimulator*.



Fonte: (SANTANA, 2019).

Além desses problemas, foi identificado um outro problema no simulador, a má distribuição de tarefas entre os processos em memória distribuída. Na Figura 5, é mostrado o gráfico do *speedup* gerado a partir de informações do experimento realizado por Santana *et al.* (2018) em seu trabalho. No gráfico é possível notar que para o caso com um processo o *speedup* é 1, como esperado. No entanto, para os casos com dois e três processos o *speedup* obtido é bem abaixo do esperado. Para dois processos o esperado seria 2, porém, o obtido foi aproximadamente 1,17. No caso de três processos, o resultado esperado seria de 3, no entanto, o valor obtido foi de aproximadamente 1,48.

Figura 5 – *Speedup* para execução de 3 cenários em 1, 2 e 3 processos.



Fonte: Adaptado de Santana *et al.* (2018).

2.2 SimPoint

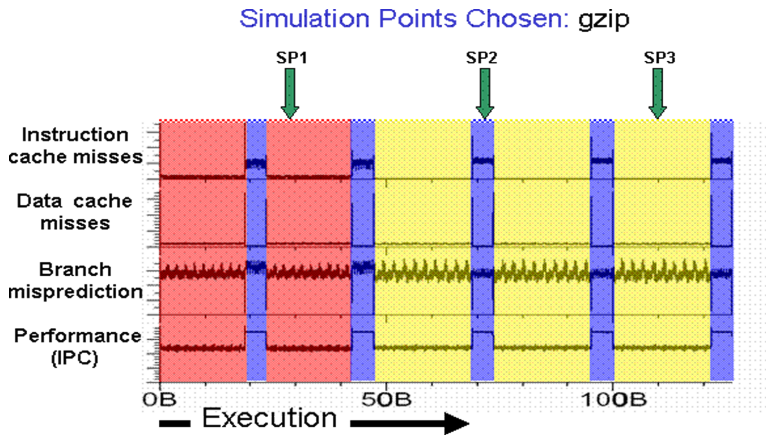
Assim como o *InterSCSimulator*, outros simuladores apresentam problemas com uso excessivo de memória e desempenho. Com o objetivo de acelerar a execução de grandes simulações, Hamerly *et al.* (2005) propuseram a técnica chamada *SimPoint*. Essa técnica tem por finalidade estimar os resultados de grandes simulações através da identificação e agrupamento de padrões de comportamento semelhantes. A partir desse agrupamento é possível selecionar representantes únicos (demonimados de Pontos de Simulação, do inglês "*Simulation Points*") de cada grupo para que se possa computar o resultado dos demais e, consequentemente, estimar o resultado da simulação por completo.

Na Figura 6 é mostrado o resultado da aplicação da técnica de Pontos de Simulação no programa *gzip*. Ao executar aplicações como *gzip*, pode-se notar comportamentos semelhantes em diferentes intervalos de tempo da execução, como é destacado pelas cores amarelo, azul e vermelho. Nesses intervalos estão representados um conjunto de blocos de códigos executados em um determinado intervalo de tempo. Além disso, neles são expressas algumas características como a quantidade de *Instruction cache misses*, *Data cache misses*, *Branch misprediction* e *Performance* (IPC). Além dessas informações, na Figura 6 estão destacados os 3 pontos de simulação SP1, SP2 e SP3 encontrados pela técnica e utilizados para estimar os demais de cada grupo (amarelo, azul e vermelho). Esses pontos de simulação são representantes únicos dos demais intervalos semelhantes a eles. Diante disso, a técnica de Pontos de Simulação pode ser dividida em duas grandes etapas: Extração dos *Basic Block Vectors* (BBV's) mostrada na Seção 2.2.1 e Agrupamento explicada na Seção 2.2.2.

2.2.1 Extração dos *Basic Block Vectors*

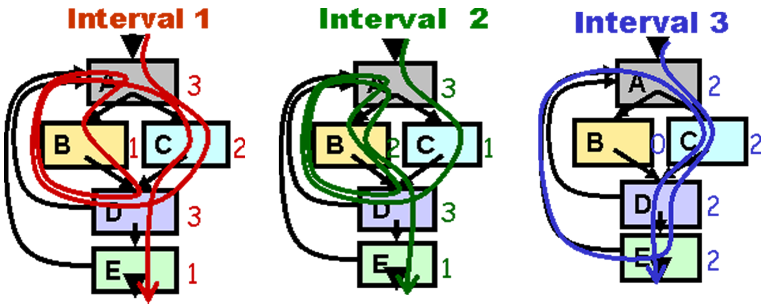
No trabalho apresentado por Hamerly *et al.* (2005), os programas de computadores podem ser divididos em vários intervalos de diversos tamanhos. Cada intervalo possui uma determinada quantidade de blocos básicos de código. Um bloco básico é uma seção de código executada do início ao fim com uma entrada e uma saída. A forma e a quantidade de vezes que esses blocos são executados define o comportamento do intervalo, como ilustrado pela Figura 7.

Figura 6 – Aplicação da técnica *SimPoint* para agrupar e extrair representantes na aplicação *gzip*.



Fonte: ⟨https://cseweb.ucsd.edu/~calder/simpoint/simpoint_overview.htm⟩ (2005).

Figura 7 – Contagem de execução dos blocos básicos em 3 intervalos de um programa qualquer.

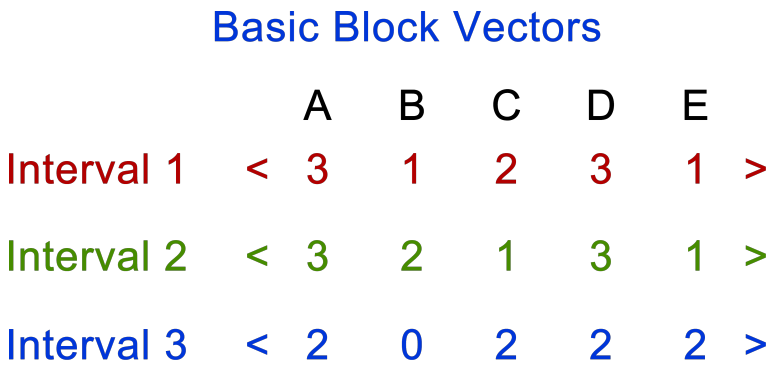


Fonte: ⟨https://cseweb.ucsd.edu/~calder/simpoint/phase_analysis.htm⟩ (2005).

Para comparar os intervalos e agrupar os que apresentam comportamentos semelhantes, Hamerly *et al.* (2005) definiram uma estrutura chamada *Basic Block Vectors* (BBV). Um BBV é um vetor do tamanho da quantidade de blocos que determinado intervalo contém. Cada célula do vetor representa um bloco básico de código e o seu valor é definido pela quantidade de vezes que o seu bloco foi executado naquele intervalo, como ilustrado pela Figura 8.

Com os *Basic Block Vectors* gerados para cada intervalo, é possível compará-los entre si. Para comparar dois BBVs X e X' diferentes, Hamerly *et al.* (2005) utilizam-se da distância de *Manhattan* definida pela Fórmula 1 comumente usada para duas dimensões. No entanto, no caso do BBVs, a distância de *Manhattan* é utilizada apenas com uma dimensão como é mostrado na Fórmula 2, onde x_i representa um elemento na posição i do vetor X e x'_i representa um elemento na posição i do vetor X'.

Figura 8 – *Basic Block Vectors* gerados a partir da contagem da execução dos blocos básicos A, B, C, D e E nos três intervalos de execução *Interval 1*, *Interval 2* e *Interval 3* da Figura 7.



Fonte: ⟨https://cseweb.ucsd.edu/~calder/simpoint/phase_analysis.htm⟩ (2005).

$$D = \sum (x_i - x'_i) + (y_i - y'_i) \tag{1}$$

$$D = \sum (x_i - x'_i) \tag{2}$$

Ao calcular a distância de *Manhattan* entre os BBVs é possível perceber que alguns vetores tem uma distância bem próxima de zero em relação a outros (o que indica uma similaridade entre os vetores) e outros com uma distância maior (o que indica que esses vetores são pouco similares). Um exemplo da aplicação da distância de *Manhattan* é dado na Figura 9.

Figura 9 – Cálculo da distância de *Manhattan* dos *Basic Block Vectors* dos três intervalos de execução *Interval 1*, *Interval 2* e *Interval 3* de um programa qualquer.

Fonte: ⟨https://cseweb.ucsd.edu/~calder/simpoint/phase_analysis.htm⟩ (2005).

2.2.2 Agrupamento com *K-Means*

Após extrair as informações tranformando-as em BBVs, é realizado o processo de agrupamento. Nesse processo, é utilizado algum algoritmo de agrupamento, nesse caso, foi utilizado o algoritmo *K-means* com a distância de *Manhattan* para comparar e agrupar os BBVs com padrões semelhantes. No entanto, é necessário normalizar as informações presentes nos BBVs para que se possa comparar uns com os outros na mesma dimensionalidade antes de realizar o agrupamento.

O *K-means* é um algoritmo de agrupamento (do inglês *clustering*) (MACQUEEN *et al.*, 1967). Ele divide um conjunto de dados D de n-dimensões em k grupos (*clusters*), agrupando-os por sua similaridade (JAIN, 2010). Cada *cluster* C_i possui um representante c_i denominado centroide. Assim, para agrupar os dados de D em k *clusters* é utilizado uma medida de distância, como a distância de *Manhattan* para saber de qual centroide c_i um dado d_i de D está mais próximo (WAGSTAFF *et al.*, 2001).

O *K-means* tem como o principal objetivo encontrar a partição na qual o erro quadrático E do centroide c_i entre os d_j dados de um dado *cluster* C_i seja minimizado (JAIN, 2010). Na Fórmula 3 é mostrado o cálculo do erro $E(C_i)$.

$$E(C_i) = \sum_{d_k \in C_i} (d_k - \overline{x_i})^2 \tag{3}$$

O algoritmo *K-means* pode ser descrito da seguinte maneira:

1. Defina o número máximo de iterações do algoritmo.
2. Para uma divisão em k *clusters*, gere k centroides.
3. Calcule a distância dos d_j dados presentes em D e agrupe cada um de acordo com o centroide que tem a menor distância calculada.
4. Calcule o novo centroide a parir das médias dos $d_k \in C_i$.
5. Repita o processo até que o número máximo de interações ou a tolerância do erro E quadrático da média empírica $\overline{x_i}$ tenha sido atingida.

Após realizar todo o processo de agrupamento, é feita a coleta dos pontos de simulação que melhor representam cada grupo de *Basic Block Vectors*. Como os centroides de cada *cluster* são os que melhor representam cada grupo, eles são os melhores candidatos

para assumir o papel de pontos de simulação. Após todo esse processo é possível estimar os demais intervalos a partir dos representantes únicos encontrados.

2.3 Dynamic Time Warping

No trabalho apresentado por Hamerly *et al.* (2005), foi utilizado *Basic Block Vectors* como assinatura dos intervalos para que se possa compará-los. No entanto, nesse trabalho, no lugar de intervalos são utilizados os comportamentos das vias simuladas pelo *InerSCSimulator*. Além disso, no lugar dos BBVs, são utilizadas séries temporais como assinaturas dos comportamentos dessas vias.

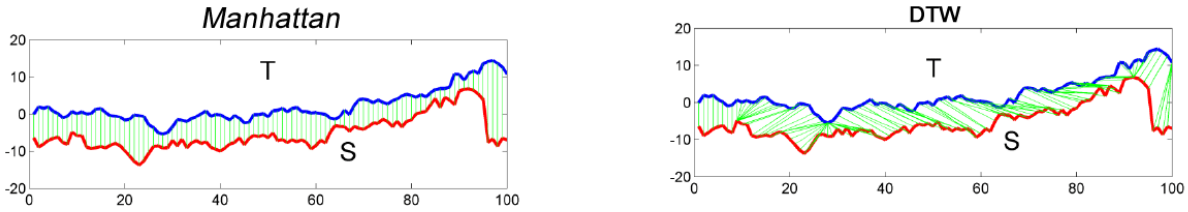
Amplamente usadas nas áreas de estatística, matemática, econometria e processamento de sinais, as séries temporais são definidas como uma sequência de eventos ordenados cronologicamente. Pode-se definir uma série como um conjunto S de $T(t, e)$ tuplas ordenadas pelo tempo, onde t representa o tempo em que o evento ocorreu e e um numeral que representa o valor do evento naquele dado instante de tempo. Geralmente, a dimensão do tempo é subentendida em um vetor unidimensional com os valores numéricos dos eventos como é mostrado na série S apresentada na Equação 4, onde os n valores s_i são os numerais representativos dos eventos.

$$S = s_1, s_2, ..., s_{n-1}, s_n \tag{4}$$

Como feito por Hamerly *et al.* (2005) com os BBVs, nesse trabalho, o objetivo é comparar as séries e saber o quão semelhantes elas são. Nessas comparações geralmente são usadas medidas de distância. Um exemplo de medida de distância é a distância de *Manhattan*, mostrada na Fórmula 2. No entanto, o uso de uma medida de distância simples como a de *Manhattan* pode não ser o suficiente para comparar duas séries. Isso porque as séries podem está deslocadas no tempo em relação a outra o que dificulta a comparação por distâncias simples, como é mostrado na Figura 10. Além disso, séries temporais podem ter tamanhos diferente das demais. Assim, o uso de uma outra métrica de distância faz-se necessário, como a *Dynamic Time Warping* (DTW).

A *Dynamic Time Warping* é uma medida utilizada para comparar séries temporais. Inicialmente ela foi usada para reconhecimento de palavras e padrões em falas (BERNDT; CLIFFORD, 1994). Porém, adequando ao contexto deste trabalho, será exemplificado o

Figura 10 – Alinhamento usando a distância de *Manhattan* (10a) e usando a distância DTW (10b).



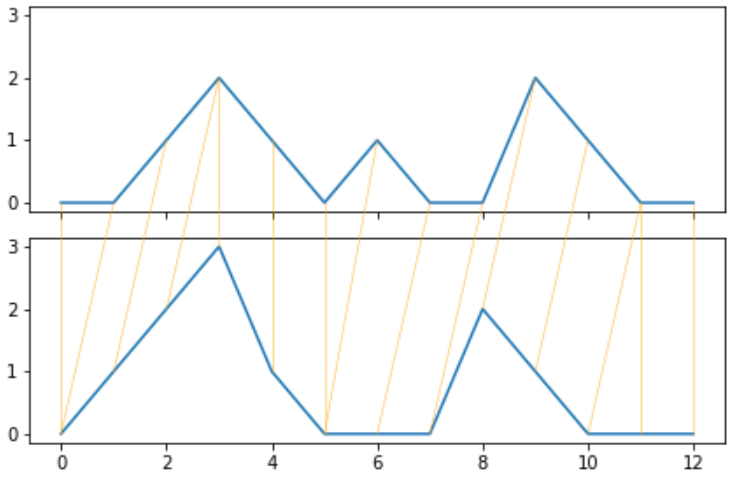
(a) (b)

Fonte: Adaptado de Cassisi *et al.* (2012).

seu funcionamento para comparar ruas e avenidas de acordo com o seu tráfego ao longo de um dia. Nesse exemplo, o valor numérico do evento em cada instante de tempo simboliza a quantidade de veículos presentes na via.

As ruas de uma cidade podem ter um fluxo de veículos baixo ou alto em períodos de tempo específicos. No entanto, nem sempre os picos (os fluxos mais altos que podem representar um congestionamento) acontecem no mesmo instante em todas as ruas. Por exemplo, uma rua no período da manhã pode ter um grande fluxo de veículos. Porém, uma outra rua pode apresentar o mesmo fluxo no período da noite. Assim, é preciso deslocar uma série para que esses comportamentos se aproximem dos seus semelhantes na outra. O DTW é um algoritmo que busca minimizar as distâncias entre os eventos em uma série S (de tamanho n) dos eventos em uma série R (de tamanho m) com o uso de programação dinâmica. Para calcular a distância miníma entre as duas séries pode-se utilizar uma Matriz M de tamanho n-por-m preenchida com as distâncias dos pontos s_i da série temporal S para os pontos r_j da série temporal R. As associações entre os pontos s_i e r_j com menores distâncias é chamada de caminho deformado (do inglês *Warping Path*). A soma de todas essas distâncias do caminho deformado é a distância mínima entre as duas séries temporais. O *caminho deformado* representa o mapeamento elemento a elemento de R e S, como é mostrado na Figura 11.

Figura 11 – Caminho deformado com as menores distâncias e associações entre as duas séries.



Fonte: ⟨https://pypi.org/project/dtaidistance/⟩ (2023).

Para calcular o caminho deformado W é preenchida a matriz M usando programação dinâmica seguindo a recorrência $M[i,j] = \text{distância}(s_i, r_j) + \min(M[i-1,j], M[i,j-1], M[i-1,j-1])$. Assim sendo, como podem haver vários caminhos deformados presentes na matriz M, $DTW(S,R)$ retorna o valor do caminho que tem a soma de todos os seus K elementos com o menor valor (KEOGH; RATANAMAHATANA, 2005). No Algoritmo 1 é mostrado o algoritmo DTW para calcular a distância entre duas séries S e R. Nesse exemplo é mostrado o cálculo da matriz M e o retorno $M[N,M]$ do algoritmo que corresponde ao valor da distância entre as séries temporais.

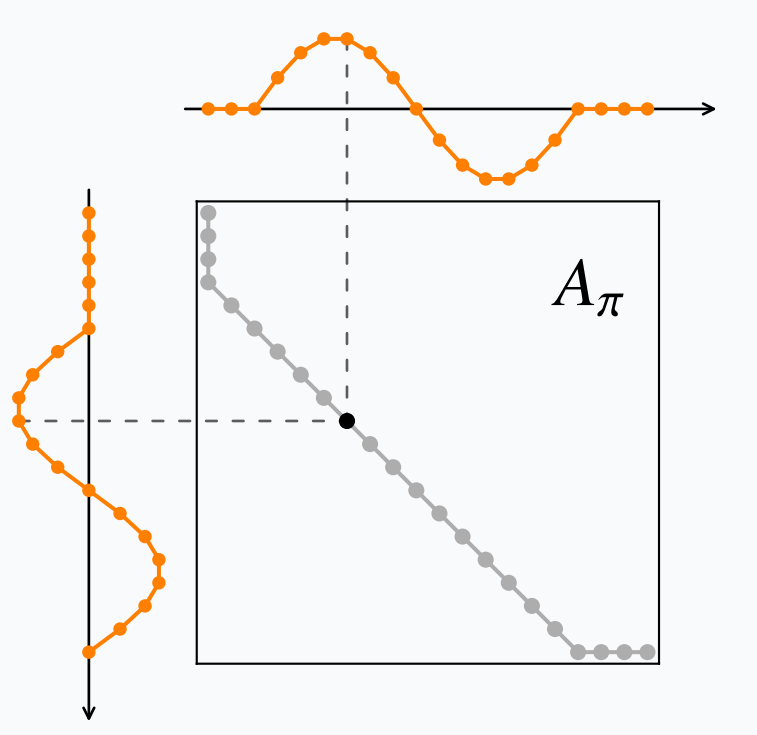
Algoritmo 1 *Dynamic Time Warping*

```
    Entrada: S[0..N], R[0..M]
    Saída: M[N, M]
  M ← VECTOR[0..N, 0..M]
  M[0, 0] ← 0
  for i ← 1..N do
      M[0, i] ← ∞
  for i ← 1..N do
      for j ← 1..M do
          custo ← distancia(S[i], R[j])
          M[i, j] ← custo + MIN(M[i − 1], M[i, j − 1], M[i − 1, j − 1])
  return M[N, M]
```

Na Figura 12, é mostrada a matriz M preenchida com o cálculo das distâncias de R e S usando o algoritmo mostrado no Algoritmo 1. Nessa matriz as menores distâncias estão

representadas com a tonalidade mais escura e as maiores com a mais clara. Além disso, é destacado em vermelho está o caminho deformado que é mostrado por outro ângulo na Figura 11.

Figura 12 – Matriz de distâncias calculadas pela DTW entre duas séries.



Fonte: ⟨https://rtavenar.github.io/blog/dtw.html⟩ (2023).

2.4 K-Shape

A DTW, usada no lugar da distância de *Manhattan* no *K-means*, apresenta uma complexidade computacional quadrática na ordem de $O(m^2)$ para computar a similaridade entre duas séries temporais de tamanho m (PAPARRIZOS; GRAVANO, 2015). Diante desse fator, este trabalho optou pela utilização do algoritmo *K-shape* que apresenta uma complexidade computacional de $O(m \log(m))$, para duas séries temporais de m. Vale ressaltar que, para facilitar o entendimento do algoritmo, será usado nesta explicação, séries do mesmo tamanho.

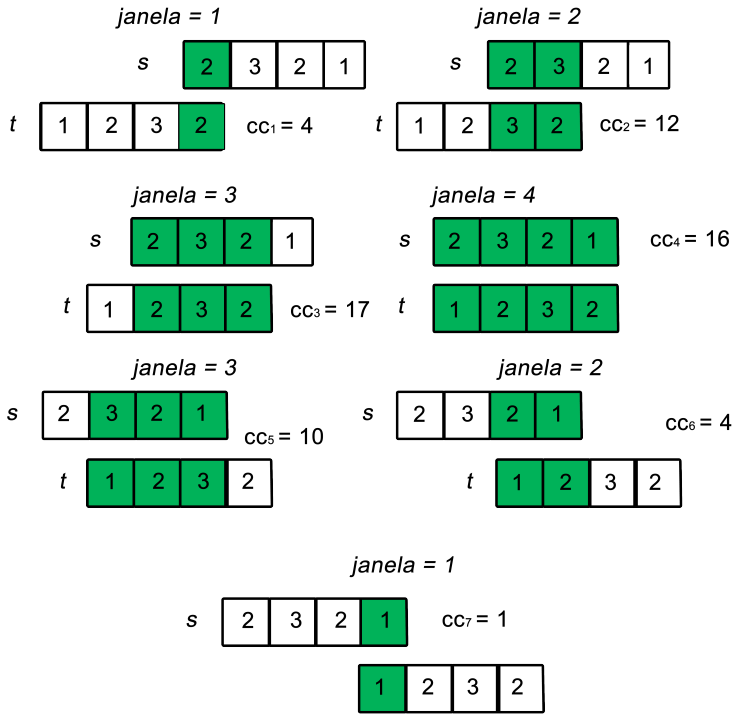
Paparrizos e Gravano (2015) propuseram um algoritmo chamado *K-shape* para agrupar séries temporais. O algoritmo é baseado no algoritmo de agrupamento *K-means*, porém com algumas diferenças significativas. As principais diferenças entre os dois são a forma de se calcular a distância e a forma de extrair o centroide do *cluster*. Ao invés de

usar uma distância de *Manhattan* ou a *Dynamic Time Warping* como é usada no *K-means*, o *K-shape* usa de uma distância baseada na forma. Essa distância é chamada pelo autor de *Shape Based Distance* (SBD), mostrada em mais detalhes na próxima seção. Além disso, nesse algoritmo, com o uso da distância SBD, é possível extrair um centroide que melhor represente as demais séries no *cluster*, como é mostrado na Seção 2.4.2.

2.4.1 *Shape Based Distance*

Para calcular a *Shape Based Distance*, Paparrizos e Gravano (2015) utilizam a medida de similaridade entre duas séries temporais s e t chamada de Correlação Cruzada (CC, do inglês *Cross-Correlation*). Ela calcula o quão uma série s está deslocada em relação a uma outra série t em termos de d (deslocamento). Para isto, é mantida a série s fixa e é deslizada a série t sobre ela calculando o produto interno entre as duas séries a cada ponto do deslizamento, como é mostrado na Figura 13.

Figura 13 – Cálculo da correlação cruzada de duas séries s e t.



Fonte: Francisco Wallison Carlos Rocha (2020).

Esse processo gera um vetor cc de tamanho $2m - 1$ que representa a correlação entre as duas séries. Para preencher cc, Paparrizos e Gravano (2015) usam $CC_w(\vec{s}, \vec{t})$ que

é definida na Fórmula 5. Onde $w \in \{1, 2, 3, ..., 2m-1\}$ é cada posição do vetor cc a ser preenchida, m é o tamanho da maior série comparada e a função R é responsável por deslizar uma série sobre a outra para calcular o produto interno para cada cc_w.

$$CC_w(\vec{s}, \vec{t}) = R_{w-m}(\vec{s}, \vec{t}) \tag{5}$$

onde R é calculada da seguinte forma:

$$R_k(\vec{s}, \vec{t}) = \begin{cases} \sum_{l=1}^{m-k} s_{l+k} \cdot t_l, k \geq 0 \\ R_{-k}(\vec{t}, \vec{s}), k < 0 \end{cases} \tag{6}$$

Onde $k = w - m$ representa o deslocamento de t em s e $m - k$ representa o tamanho da *janela* na qual será computado o produto interno do deslocamento para cada $w \in \{1, 2, 3, ..., 2m-1\}$. Assim, ao deslizar uma série sobre a outra preenchendo cc, é possível medir o tamanho do deslocamento necessário para que t se aproxime de s. Então, seja w_i a posição do elemento de maior valor em cc, o deslocamento necessário é dado por $d = w_i - m$. Para um d positivo ou igual a 0, as d posições iniciais de t são preenchidas com 0 e as demais com os $t(1, 2, ...,t_{m-d})$ elementos. Se d é negativo, as $m - d$ posições iniciais são preenchidas com os $t(t_{1-d}, ..., t_{m-1}, t_m)$ elementos e as demais d posições com o valor 0. Os dois casos estão exemplificados na Fórmula 7.

$$t_{(d)} = \begin{cases} (\overbrace{0, ..., 0}^{d}, t_1, t_2, ..., t_{m-d}), d \geq 0 \\ (t_{1-d}, ..., t_{m-1}, t_m, \overbrace{0, ..., 0}^{d}), d < 0 \end{cases} \tag{7}$$

No entanto, Paparrizos e Gravano (2015) afirmam que dependendo do domínio da aplicação é necessário aplicar uma normalização em cc antes de selecionar o elemento na posição w_i. No trabalho de Paparrizos e Gravano (2015), e também neste trabalho, a normalização aplicada é definida pela Fórmula 8 que coloca os valores de cc em uma escala entre -1 e 1.

$$NCC(\vec{s}, \vec{t}) = \frac{cc}{\sqrt{R_0(\vec{s}, \vec{s}) \cdot R_0(\vec{t}, \vec{t})}} \tag{8}$$

O maior valor normalizado obtido na posição w_i do vetor cc é usado para calcular a distância $SBD = 1 - cc_{w_i}$ que tem uma faixa de valores entre 0 e 2. O valor 0 determina uma similaridade perfeita entre as duas séries e o valor 2 o contrário (PAPARRIZOS; GRAVANO, 2015).

Entretanto, a complexidade de $CC_w(\vec{s},\vec{t})$ é $O(m^2)$, assim como na DTW. Para reduzir a complexidade de $O(m^2)$ para $O(m\log(m))$, Paparrizos e Gravano (2015) propõem usar a Transformada Discreta Inversa de Fourier (F^{-1}) do produto das Transformadas Discretas de Fourier (F) (BRACEWELL; BRACEWELL, 1986) individuais das séries temporais para calcular a correlação cruzada de duas séries temporais. Assim, a nova forma de calcular $CC_w(\vec{s},\vec{t})$ é definida na Fórmula 9.

$$CC_w(\vec{s},\vec{t}) = F^{-1}\{F(\vec{s}) \cdot F(\vec{t})\} \tag{9}$$

No entanto, o cálculo de $CC_w(\vec{s},\vec{t})$ usando a Transformada Discreta de Fourier ainda é $O(m^2)$. Porém, é possível usar uma implementação rápida dessa transformada que a deixa com uma complexidade de $O(mlog(m))$.

2.4.2 *Extração dos Centroides*

Após realizar o cálculo das distâncias de todas as séries para o centroide c_i do *cluster* C_i, Paparrizos e Gravano (2015) agrupam todas as séries deslocadas calculadas pela Fórmula 7 em uma matriz X. Após realizar esse agrupamento, aplicam-se algumas transformações matriciais em X para obter uma matriz simétrica M, como é mostrada na Fórmula 10.

$$M = Q^T \cdot U \cdot Q \tag{10}$$

onde,

$$U = X^T \cdot X \tag{11}$$

e,

$$Q = I - \frac{1}{m}O \tag{12}$$

Onde X^T é a transposta de X, I é a matriz identidade, O é uma matriz completamente preenchida por $1's$, m é o tamanho das séries e Q^T é a transposta de Q.

Além de obter uma matriz M simétrica, essas transformações aplicadas por Paparrizos e Gravano (2015) nos permitem chegar a um problema bem conhecido que é a maximização do quociente de Rayleigh (PARLETT, 1974). O quociente de Rayleigh permite extrair o *autovetor* associado ao maior *autovalor* de M. Esse *autovetor* extraído corresponde ao centroide c_i do C_i. Esse centroide pela definição da Análise dos Componentes Principais é o que melhor representa o conjunto de valores presentes em M (WOLD; ESBENSEN; GELADI, 1987).

3 Revisão da literatura

O crescimento populacional irá acarretar no agravamento de diversos problemas já enfrentados nos dias atuais. Megalópoles como São Paulo, Tóquio, Rio de Janeiro, Nova York, Paris, entre outras, já sofrem com problemas de poluição, tráfego, transporte público, serviços de saúde e educação (SANTANA *et al.*, 2018). Para a resolução desses problemas, uma alternativa seria o uso de aplicações e tecnologias desenvolvidas para Cidades Inteligentes (do inglês *Smart Cities* ou *Digital Cities*) (NAM; PARDO, 2011). Contudo, no contexto atual, a literatura já contempla diversas abordagens para solucionar ou mitigar alguns dos problemas remetentes a cidades.

Alguns dos trabalhos de revisões sistemáticas (RS) presentes na literatura agrupam essas abordagens a fim de apresentar o estado da arte a respeito de métodos, técnicas, abordagens e algoritmos relacionados ao contexto de cidades inteligentes. Em Kyriazopoulou (2015) é apresentada uma revisão sistemática para discutir as tecnologias e arquiteturas-chave propostas para o contexto de cidades inteligentes. Nosratabadi *et al.* (2019) mostram o estado da arte de algoritmos de *machine learning* e *deep learning* no contexto de cidades inteligentes obtido após a aplicação de uma RS. Já em (CHAMOSO *et al.*, 2018) é exposto um apanhado das tendências e tecnologias. Esses trabalhos apresentam uma RS das tecnologias existentes de modo geral.

Por outro lado, este trabalho apresenta especificamente o estado da arte de técnicas, métodos e abordagens para acelerar simuladores de cidades inteligentes presentes atualmente na literatura.

Nesta seção, são analisadas publicações científicas através de um estudo baseado em uma revisão sistemática, com o propósito de identificar e analisar simuladores, métodos, técnicas e algoritmos no contexto de simulações em cidades inteligentes em larga escala.

3.1 Protocolo Resumido

A revisão sistemática realizada, foi conduzida em três fases: (i) planejamento, na qual é direcionada pelo protocolo da revisão, (ii) condução, na qual é realizada a seleção dos trabalhos relevantes para este estudo usando os critérios de inclusão e exclusão definidos

no protocolo descrito na Tabela 2 e, por fim, (iii) extração, que é realizada a extração das informações ditas como mais relevantes definidas no formulário de extração desse trabalho.

Tabela 2 – Tabela com os critérios de inclusão (CI's) e com os critérios de exclusão (CE's).

Código	Critério
CI1	Serão considerados artigos a partir do ano de 2014.
CI2	Devem ser trabalhos publicados e disponíveis integralmente em bases de dados científicas.
CI3	Os trabalhos devem tratar especificamente de técnicas, abordagens, métodos ou algoritmos para melhorar o desempenho de aplicações ou simuladores em larga escala no contexto de cidades inteligentes.
CI4	Os trabalhos escritos na língua inglesa.
CE1	Serão desconsiderados os trabalhos que não apresentem resultados do estudo dessas técnicas.
CE2	Serão desconsiderados trabalhos cuja a versão completa não esteja disponível entre as fontes selecionadas.
CE3	Serão desconsiderados trabalhos que não estão escritos na língua inglesa.
CE4	Serão desconsiderados trabalhos que não apresentem alguma característica dos critérios de inclusão.
CE5	Não apresentem simuladores, métodos, técnicas, abordagens ou algoritmos em larga escala no contexto de cidades inteligentes.

Fonte: Francisco Wallison Carlos Rocha (2020).

A RS foi conduzida para responder a seguinte questão: "Quais são os simuladores, abordagens, métodos, técnicas e algoritmos existentes no contexto de cidades inteligentes para execução de simulações em larga escala?".

Para responder essa questão foram definidas *strings* de busca em três grandes bibliotecas digitais ACM, IEEE e *Springer Link*. Foram aplicadas duas *strings* de buscas em cada uma das bibliotecas digitais, uma para cada período. Devido ao tempo da primeira aplicação das primeiras *strings* mostradas na Tabela 3 e a conclusão deste trabalho, fez-se necessário aplicar mais uma nova pesquisa usando as *strings* da Tabela 4. A primeira pesquisa foi aplicada a partir do ano de 2014 até o ano de 2019. A segunda pesquisa foi a partir do ano de 2020 até o ano de 2022. Um outro detalhe importante é que devido a quantidade de artigos retornada pela *string* na biblioteca digital da ACM nos anos de 2020 até 2022, essa *string* foi aplicada a partir do ano de 2019. Por esse motivo, foram encontrados artigos do ano de 2019 na pesquisa.

A etapa de seleção das fontes após aplicação da *string* está sendo conduzida do da seguinte forma: serão considerados trabalhos que apresentarem os termos relevantes

Tabela 3 – Tabela com as *strings* de busca associadas às bibliotecas digitais aplicadas na primeira pesquisa.

Biblioteca	*String* de Busca
ACM	*[[Publication Title: simulation] OR [Publication Title: simulator] OR [Publication Title: simulate]] AND [[All: approach] OR [All: method] OR [All: algorithm] OR [All:technique] OR [All: simulator]] AND [[All: fast] OR [All: efficient] OR [All: scalable] OR [All: scalability] OR [All: large-scale] OR [All: parallel] OR [All: distributed]] AND [All: smart] AND [[All: city] OR [All: smart]] AND [[All: cities] OR [All: digital]] AND [[All: city] OR [All: smart]] AND [[All: mobility] OR [All: digital]] AND [All:cities] AND [Publication Date: Past 5 years]*
IEEE	*(Performance OR Scalability OR scalable) AND ("Smart City Simulation" OR Simulator OR simulate) AND (Algorithm OR Approach OR Technique OR Method) AND ("Smart City" OR "Smart Cities" OR "Smart Mobility")*
Sptring Link	*NOT "simulated annealing" OR "signal processing" OR "speech processing") AND (Scalability) AND ("Smart City Simulation" OR Simulator) AND (approach OR algorithm OR method OR technique) AND ("Smart City" OR "Smart Cities" OR "Smart Mobility")*

Fonte: Francisco Wallison Carlos Rocha (2022).

no título, resumo e/ou palavras-chave de cada documento presente nas base de dados. O pesquisador deverá realizar a remoção de entradas duplicadas, caso estas existam. Após a seleção inicial, foi realizada a leitura do título e do resumo com a aplicação dos critérios de inclusão e exclusão. Quando necessário, foi realizada a leitura das demais partes do documento (introdução, conclusão e metodologia, nessa ordem) aplicado novamente os mesmos critérios para incluir ou excluir os trabalhos. Os resultados dessa etapa para a primeira pesquisa são mostrados na Figura 14. Já os resultados para a segunda pesquisa são mostrados na Figura 15.

Após a seleção, serão colhidos os dados preenchendo o formulário de extração e conduzida a análise quantitativa e qualitativas dos trabalhos selecionados após a aplicação dos critérios de inclusão e exclusão.

3.2 Análise Geral

Este trabalho mostra que nos últimos anos houve uma crescente evolução de artigos relacionados aos termos de busca aplicados. A Figura 16 mostra o número de artigos do ano de 2014 ao ano de 2019. Nessa mesma figura também é possível notar que há um grande número de artigos no ano de 2018. A Figura 17 mostra o número de artigos por

Tabela 4 – Tabela com as *strings* de busca associadas às bibliotecas digitais as quais foram aplicadas na segunda pesquisa.

Biblioteca	String de Busca
ACM	*[[Publication Title: simulation] OR [Publication Title: simulator] OR [Publication Title: simulate]] AND [[All: approach] OR [All: method] OR [All: algorithm] OR [All: technique] OR [All: simulator]] AND [[Abstract: fast] OR [Abstract: large-scale] OR [Abstract: parallel] OR [Abstract: distributed]] AND [[Abstract: smart city] OR [Abstract: smart cities] OR [Abstract: digital city] OR [Abstract: smart mobility] OR [Abstract: digital cities]] AND [E-Publication Date: (01/01/2019 TO 12/31/2022)]*
IEEE	*(“All Metadata”:Performance OR “All Metadata”:Scalability OR “All Metadata”:scalable) AND (“All Metadata”:Smart City Simulation OR “All Metadata”:Simulator OR “All Metadata”:simulate) AND (“All Metadata”:algorithm OR “All Metadata”:Approach OR “All Metadata”: Technique OR “All Metadata”:Method) AND (“Abstract”:Smart City OR “Abstract”:Smart Cities OR “Abstract”: Smart Mobility)*
Sptring Link	*NOT (“simulated annealing” OR “signal processing” OR “speech processing”) AND (Scalability) AND (“Smart City Simulation” OR Simulator) AND (approach OR algorithm OR method OR technique) AND (“Smart City” OR “Smart Cities” OR “Smart Mobility”)*

Fonte: Francisco Wallison Carlos Rocha (2023).

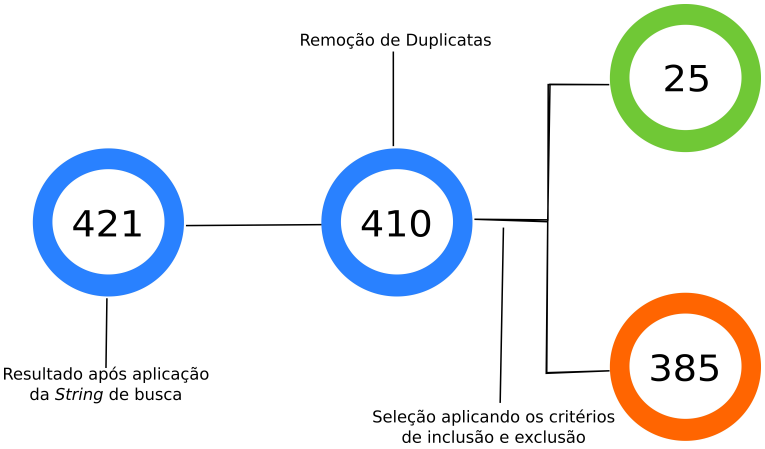
ano a partir do ano de 2019. Na Figura 17 o destaque vai para o ano de 2021 que apresenta a maior quantidade de artigos retornados.

Na Tabela 5 estão sumarizados os trabalhos obtidos na etapa de extração da primeira busca. Esta tabela apresenta os campos Trabalho, Ano, Fonte, Objetivo e Abordagem. Todos os estes presentes foram preenchidos a partir do formulário de extração definido previamente na etapa de planejamento.

Na Tabela 6 estão sumarizados os trabalhos obtidos na etapa de extração da segunda busca. Esta tabela apresenta os campos Trabalho, Ano, Fonte, Objetivo e Abordagem. Todos os estes presentes foram preenchidos a partir dos critérios de aceite e exclusão da Tabela 2.

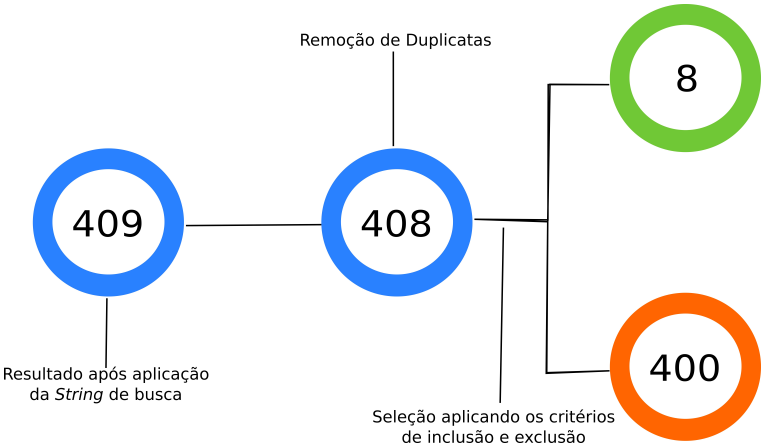
No processo de extração aplicado na primeira busca foi possível constatar que cerca de 40% dos artigos aceitos tem por finalidade a simulação de tráfego urbano. 12% são voltados para Internet das Coisas (do inglês *Internet of Things*) (IoT) e redes de

Figura 14 – Fluxo de artigos desde a fase de aplicação das *strings* de busca (aplicadas na primeira pesquisa) até a fase de extração com a aplicação dos critérios de inclusão e exclusão.



Fonte: Francisco Wallison Carlos Rocha (2020).

Figura 15 – Fluxo de artigos desde a fase de aplicação das *strings* de busca (aplicadas na segunda pesquisa) até a fase de extração com a aplicação dos critérios de inclusão e exclusão.

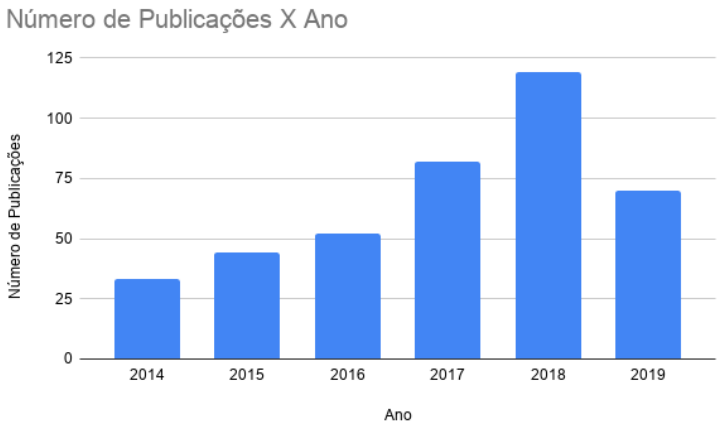


Fonte: Francisco Wallison Carlos Rocha (2023).

computadores. Outros 12% correspondem a trabalhos que estão relacionados a Elementos Finitos e Dinâmica dos Fluidos (EF e DF). Uma outra categoria denominada aqui como outros, que corresponde a 36% dos que se dividem entre métodos para melhoria da escalabilidade, simuladores de diferentes tipos, entre outras finalidades como mostra a Figura 18a. Já no processo de extração aplicado na segunda busca foi constado que 50% dos artigos estão relacionados com IoT. Os relacionados ao tráfego urbano são 37,5% e 12,5% correspondem a outros tipos, como é mostrado na Figura 18b.

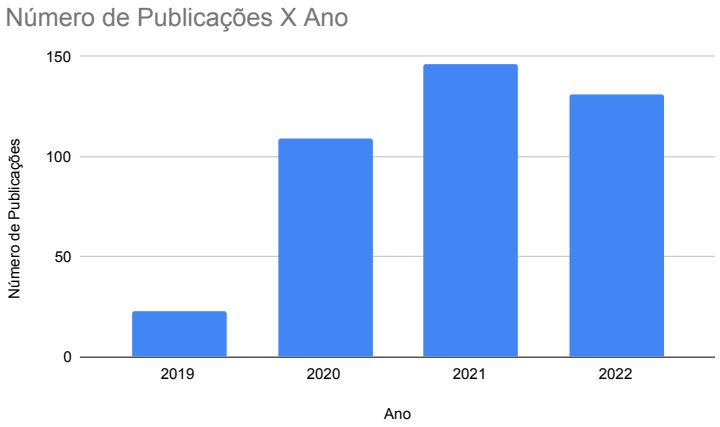
Além disso, os trabalhos foram divididos em quatro tipos: algoritmo, método, abordagem e simulador. A Figura 19a mostra que na primeira busca a maioria dos

Figura 16 – Publicações X Ano obtidas com a aplicação das *strings* de busca da primeira pesquisa.



Fonte: Francisco Wallison Carlos Rocha (2020).

Figura 17 – Publicações X Ano obtidas com a aplicação das *strings* de busca da segunda pesquisa.



Fonte: Francisco Wallison Carlos Rocha (2023).

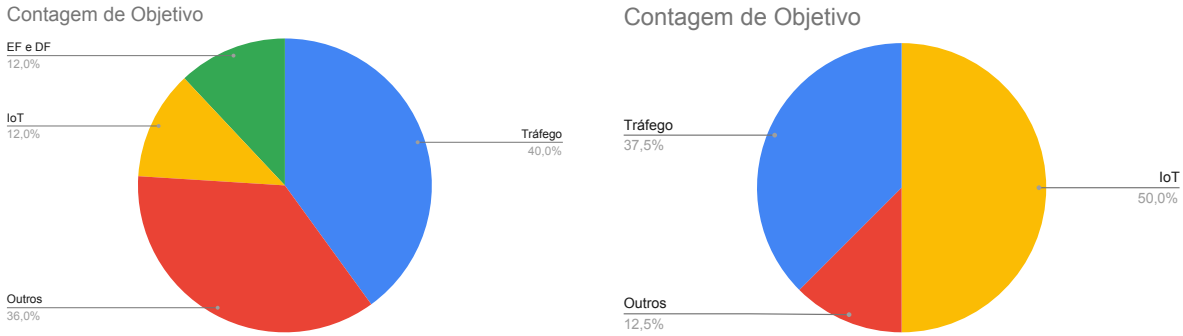
trabalhos (52%) são simuladores, em segundo lugar vem os abordagens, representando 24% e algoritmos e métodos com 12% dos trabalhos relacionados a simulações em cidades inteligentes em larga escala. Na segunda busca os trabalhos também estão divididos em quatro tipos. Os trabalhos relacionados com o tipo simuladores correspondem a 37,5%. As abordagens e métodos correspondem a 25% cada. Os restantes 12,5% correspondem aos algoritmos. Como é mostrado na Figura 19b.

Tabela 5 – Sumarização dos trabalhos selecionados na etapa de extração até o ano de 2020.

Trabalho	Ano	Fonte	Objetivo	Tipo
Andelfinger *et al.* (2018)	2018	ACM	Tráfego	Método
Khunayn *et al.* (2017)	2017	ACM	Tráfego	Algoritmo
Birdsey, Szabo e Falkner (2016)	2016	ACM	Outros	Método
Brambilla *et al.* (2014)	2014	ACM	IoT	Simulador
Brito *et al.* (2015)	2015	ACM	Outros	Simulador
Molitor *et al.* (2014)	2014	IEEE	Outros	Simulador
Elbery *et al.* (2019)	2019	*Springer*	Tráfego	Simulador
Fu, Yu e Sarwat (2019)	2019	ACM	Tráfego	Simulador
D'Angelo, Ferretti e Ghini (2017)	2017	IEEE	IoT	Abordagem
Gütlein, German e Djanatliev (2018)	2018	ACM	Tráfego	Simulador
Garzon *et al.* (2016)	2016	ACM	Outros	Simulador
Ianni *et al.* (2018)	2018	ACM	Outros	Algoritmo
Ichimura *et al.* (2015)	2015	IEEE	EF e DF	Simulador
Ichimura *et al.* (2014)	2014	IEEE	EF e DF	Simulador
Jang *et al.* (2019)	2019	ACM	Tráfego	Algoritmo
Kaminka e Fridman (2018)	2018	ACM	Tráfego	Método
Lounis *et al.* (2017)	2017	*Springer*	IoT	Abordagem
Motlagh e Li (2019)	2019	ACM	Outros	Abordagem
Ramamohanarao *et al.* (2017)	2016	ACM	Tráfego	Simulador
Santana *et al.* (2018)	2018	*Springer*	Tráfego	Simulador
Tang *et al.* (2017)	2017	*Springer*	Outros	Abordagem
Xu e Tan (2014)	2014	ACM	Tráfego	Simulador
He *et al.* (2019)	2019	IEEE	EF e DF	Simulador
Lin e Yao (2015b)	2015	IEEE	Outros	Abordagem
Lin e Yao (2015a)	2015	IEEE	Outros	Abordagem

Fonte: Francisco Wallison Carlos Rocha (2020).

Figura 18 – Porcentagem de trabalhos categorizados por objetivo até o ano de 2020 (18a) e a partir de 2020 (18b).



(a) (b)

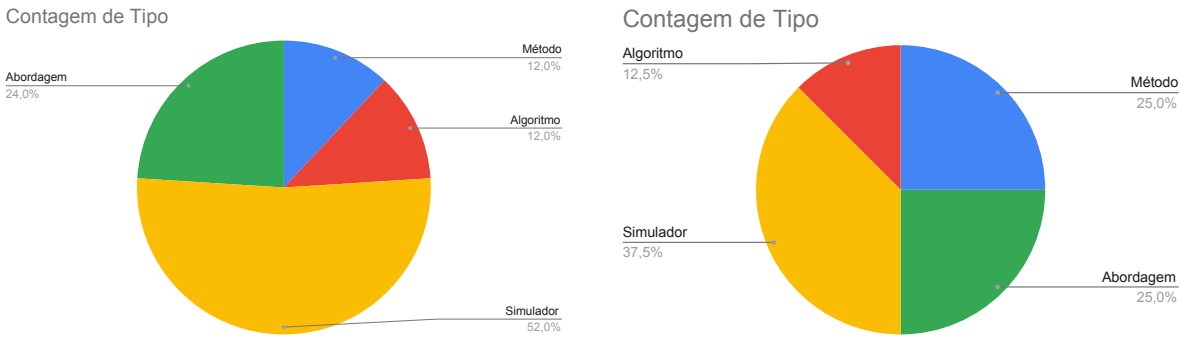
Fonte: Francisco Wallison Carlos Rocha (2023).

Tabela 6 – Sumarização dos trabalhos selecionados na etapa de extração a partir do ano de 2020 com alguns de 2019.

Trabalho	Ano	Fonte	Objetivo	Tipo
Francisco, Pinho e Luís (2021)	2021	IEEE	IoT	Método
Petrović *et al.* (2020)	2020	IEEE	IoT	Abordagem
Schiera *et al.* (2020)	2020	IEEE	Outros	Simulador
Montori *et al.* (2019)	2019	ACM	IoT	Abordagem
Rakow (2019)	2019	ACM	Tráfego	Simulador
Weyl *et al.* (2019)	2019	IEEE	Tráfego	Simulador
Manzanilla-Salazar *et al.* (2021)	2021	IEEE	IoT	Algoritmo
Hanai *et al.* (2019)	2019	ACM	Tráfego	Método

Fonte: Francisco Wallison Carlos Rocha (2023).

Figura 19 – Porcentagem de trabalhos categorizados por tipo até o ano de 2020 (19a) e a partir de 2020 (19b).



(a) (b)

Fonte: Francisco Wallison Carlos Rocha (2023).

3.3 Síntese dos Trabalhos Selecionados

Os trabalhos apresentados nesta seção estão agrupados de acordo com a finalidade apresentada. Pode-se observar que a maioria dos trabalhos selecionados que apresentam técnicas, abordagens ou simuladores estão direcionados à simulação do tráfego urbano. Assim sendo, esses trabalhos podem ser divididos em três níveis: microscópio, mesoscópico e macroscópico. Além desses trabalhos, foram encontrados outros trabalhos relacionados aos outros temas no contextos de cidades inteligentes. Esses trabalhos estão organizados em duas seções. Na Seção 3.3.1 dividida pelos tópicos dos níveis estão representados o artigos referentes ao tráfego urbano. Já na Seção 3.3.4 estão descritos os trabalhos que remetem a outros temas.

3.3.1 Simulação de Tráfego

Nessa seção estão representados os trabalhos relacionados a simulação de tráfego urbano divididos nos modelos de abstração *microscópico*, *mesoscópico* e *macroscópico*. Esses três modelos são comumente utilizados na literatura (SANTANA, 2019). (BARCELÓ *et al.*, 2010) diferenciam esses modelos principalmente em níveis de detalhamento. O *microscópico* apresenta um maior nível de detalhamento das entidades e de suas interações com as demais envolvidas na simulação. Já o *mesoscópico* apresenta um grande nível de detalhamento das entidades, mas descreve sua interações e atividades com menos detalhes que o *microscópico*. Por fim, o *macroscópico* representa sua entidades e interações entre elas com um baixo nível de detalhe.

Microscópico

Andelfinger *et al.* (2018) apresentam um método para melhorar a tomada de decisão de agentes (que simulam pedestres) em situações mais complexas como uma situação de rotas em um emergência. Fu, Yu e Sarwat (2019) apresentam um simulador de tráfego urbano microscópico, escalável e distribuído usando uma *quadtree* e *Apache Spark*. Ramamohanarao *et al.* (2017) mostram um simulador distribuído, o que o possibilita alcançar uma maior escalabilidade diferente dos demais encontrados na literatura pelo o autor. Já Khunayn *et al.* (2017) apresentam algoritmos para melhoria da escalabilidade de simuladores de cidades inteligentes. Jang *et al.* (2019) utilizaram aprendizado profundo por reforço para treinar veículos autônomos para liderar uma frota de veículos em uma rotatória e depois transferir essa política para um cenário para um de uma cidade com uma escala maior. Ma, Preum e Stankovic (2017) apresentam um detector de conflito de agentes heterogêneos no contexto de tráfego urbano.

Com o objetivo de fornecer um simulador em larga escala, Weyl *et al.* (2019) propuseram em seu trabalho o MARS V3. Um simulador de tráfego urbano microscópico, onde usou como teste um volume de aproximadamente 1 milhão de agentes na cidade Hamburgo na Alemanha.

Xu e Tan (2014) apresentam um simulador de tráfego microscópico em larga escala chamado Sim-Tree. Devido a natureza das simulações microscópicas utilizarem de uma árvore espacial para mapear os veículos na simulação e a necessidade constante

de reequilibrar essa árvore torna-se um processo custoso. Os autores perceberam que não há necessidade de verificar ou reequilibrar sua estrutura de árvore quando veículos individuais atualizam frequentemente suas localizações, sendo essa a principal característica do Sim-Tree.

Semelhante a este trabalho, Hanai *et al.* (2019) propuseram um trabalho para acelerar simulações de tráfego urbano no contexto de cidades inteligentes executando apenas partes de uma simulação. Como em alguns casos de testes de simulações realizados para propor novas soluções para cidade podem ocorrer mudanças em pequenas partes da simulação. Uma grande parte da simulação não se altera. Diante disso, em seu trabalho Hanai *et al.* (2019) executam uma simulação inteira. Após a execução dessa simulação por completo é obtido um arquivo contendo todos os eventos dessa simulação. Esses eventos são todos mapeados, para que apenas o eventos que sofrem alteração com a modificação da simulação e os eventos que estão ligados a eles sejam executados. Os demais eventos não selecionados, são reciclados devido não terem relação com a mudança da simulação. A grande diferença deste trabalho, é que identificamos todas as partes principais da simulação independente da quantidade de eventos afetados. Depois disso, somente o necessário para formar essas partes é executado e o restante da simulação é estimada. Então, a nossa abordagem mantém a mesma quantidade de partes principais independente da quantidade de modificações da simulação e dos eventos afetados, o que difere da abordagem de Hanai *et al.* (2019).

Mesoscópico

Santana *et al.* (2018) desenvolveram um simulador escrito em Erlang e escalável de tráfego urbano para suprir os desafios de escalabilidade existentes em trabalhos anteriores presentes na literatura. Já Gütlein, German e Djanatliev (2018) construíram um *framework* de simuladores multinível de tráfico de transportes urbanos em larga escala, o que oferece a vantagem de alto desempenho, um nível de detalhe mais flexível e adaptação decente ao caso de uso ao mesmo tempo.

Elbery *et al.* (2019) apresentam um *framework* chamado de INTEGRATION Ver. 3.0 para a simulação multimodal em larga escala baseado em agentes. Ele apresenta uma abordagem híbrida, simulando tanto nos níveis microscópico quanto no mesoscópico. Além

disso, ele conta com uma distribuição das tarefas através do particionamento geográfico assim como foi feito no estudo inicial desse trabalho. Além do particionamento espacial, ele também apresenta o particionamento vertical separando a rede por meio de transporte para modos que interagem minimamente.

Macroscópico

Uma abordagem promissora para a modelagem de multidão depende de simulações baseadas em agentes de nível micro, onde as interações de agentes individuais simulados na multidão resultam em dinâmicas de multidão de nível macro que são o objeto de estudo. Este (KAMINKA; FRIDMAN, 2018) relata um modelo de multidões de pedestres urbanos baseado em agentes, em que a cultura é explicitamente modelada. Estendemos um modelo de agente social baseado em agente estabelecido, inspirado pela psicologia social, para dar conta dos atributos culturais individuais discutidos na literatura das ciências sociais. Ele permite lidar com vários atributos culturais individuais dos agentes.

Rakow (2019) apresentam um framework de simulações de tráfego urbano chamado JIACVI-ITS. Esse simulador usa uma abordagem baseada em atores com a troca de mensagens como forma de comunicação entre os agentes. Essa abordagem, assim como em Santana *et al.* (2018), tem como finalidade executar grandes cenários em larga escala.

3.3.2 Elementos Finitos e Dinâmica dos Fluidos

Diante do desafio que é realizar uma simulação em tempo real online pela rede de internet para simular os gases presentes na atmosfera. He *et al.* (2019) propuseram um modelo para simular a propagação de gases de poluição através da dinâmica dos fluidos em um ambiente computacional em nuvem. Ichimura *et al.* (2014) apresentam Gamera uma aplicação hibrida baseada no uso de MPI-OpenMP. Ela é utilizada para simular terremotos em áreas urbanas com a utilização de elementos finitos analisando a estrutura de edifícios. Nessa aplicação foi usado mais 290 mil núcleos de CPU. Em seu trabalho mais recente, Ichimura *et al.* (2015) realizam uma simulação com uma área bem maior do que é apresentada no estado da arte com o uso de mais 600 mil núcleos de CPU. Ele

simula terremotos apresentando uma análise de propagação e uma análise da evacuação de civis na área.

3.3.3 Internet das Coisas

Em seu trabalho, D'Angelo, Ferretti e Ghini (2017) discutem um conjunto de abordagens para prover cenários de simulações de componentes de Internet das Coisas (do inglês *Internet of Things*) (IoT) escaláveis. Ele também discute como a combinações de técnicas pode melhorar a escalabilidade para que se torne possível a execução massiva e em tempo real desses ambientes de IoT. Brambilla *et al.* (2014) criaram um simulador para testar a cooperatividade e o comportamento de dispositivos em rede, equipados com sensores, atuadores, instalações computacionais e de armazenamento em larga escala. Já Lounis *et al.* (2017) apresentam um modelo para calcular e estimar o consumo de energia em sensores de uma rede *wireless*. O trabalho usa um modelo discreto de simulação de eventos que pode ser facilmente implantado no simulador do CupCarbon.

Francisco, Pinho e Luís (2021) abordam os grandes desafios da grande demanda crescente por informação. Esses desafios são enfrentados no contexto de cidades inteligentes. As Low Power Wide Area Networks (LPWANs) são vistas como tecnologias promissoras e facilitadoras para a coleta de dados. No entanto, propor soluções novas nesse contexto é um desafio. Então, para testar e propor novas soluções inovadoras. Francisco, Pinho e Luís (2021) propõem o uso do simulador de redes LorRa, o LoRaSim. Além disso, no trabalho foi implementado uma forma de ajustar a granularidade dos dados e a riqueza das informações. Ainda na perspectiva do desafio de redes móveis, Petrović *et al.* (2020) apresentam o uso de simulações para testar novas soluções. Para executar simulações de forma mais rápida, Petrović *et al.* (2020) propuseram o uso de GPUs para acelerar estas simulações.

Diante o desafio da coleta de dados em grandes ambientes urbanos, Montori *et al.* (2019) apresentam uma nova versão do CrowdSenSim. Montori *et al.* (2019) apresentam uma abordagem para permitir que algoritmos de coletas de dados que serão testados por pesquisadores. Além disso, as implementações melhoram o desempenho do simulador em relação à memória e ao tempo de execução.

Manzanilla-Salazar *et al.* (2021) propuseram um algoritmo para amenizar o número de sincronizações de *threads* realizadas no decorrer de simulações de redes sem fios em

larga escala. A sincronização ocorre devido a troca de uma estação base que atende um equipamento de usuário. Essas trocas são constantes e o tempo de sincronização das *threads* acaba afetando os ganhos de desempenho usando uma abordagem *multi-thread*.

3.3.4 Outros Trabalhos

Birdsey, Szabo e Falkner (2016) apresentam uma linguagem declarativa com a capacidade de capturar as principais características de um sistema complexo e também facilitar a criação de modelos para esse sistema. Birdsey, Szabo e Falkner (2016) em seu trabalho definem um sistema complexo como um sistema onde as entidade e ambiente são incentivados a interagir entre si para alcançar as suas propriedades desejadas, como no domínio de cidades inteligentes. A solução se dá através do uso de uma *Complex Adaptive System Language* (CASL). Brito *et al.* (2015) desenvolveram uma plataforma de simuladores heterogêneos baseada em *High Level Architecture* (HLA), com um *middleware* para simulação de eventos discretos distribuídos. O objetivo era criar um ambiente com execução de alto desempenho de sistemas embarcados em larga escala, heterogêneos e complexos. Garzon *et al.* (2016) construíram um simulador com a finalidade de suprir as carências da simulação de ambientes externos no laboratório, essas abordagens são baseadas em modelos de propagação de rádio para a simulação de redes sem fio ou são limitadas a um espaço virtual específico. Ianni *et al.* (2018) buscam oferecer uma eficiência maior em determinadas circunstâncias em simulações executadas em sistemas de memória compartilhada. Tang *et al.* (2017) apresentam uma abordagem utilizando o Apache Spark para acelerar simulações de veículos autônomos em um ambiente distribuído. Ele aplica essa abordagem para melhorar o desempenho da aplicação Robot Operating System (ROS). Molitor *et al.* (2014) apresentam uma plataforma de de simulação de múltiplos domínios para realizar uma análise holística de energia de uma cidade. A plataforma permite a simulação de uma grande quantidade de edifícios com várias perspectivas do fornecimento de energia para o prédio.

Devido ao grande consumo de memória por parte de simulações de eventos discretos, Lin e Yao (2015b) propõem uma abordagem de computação reversa em seu simulador de eventos discretos *multithreaded*. A abordagem proposta é usada para simulação de difusão de reação estocástica em larga escala para resolver o problema do consumo excessivo de

memória. Diferentemente desse trabalho, que visa utilizar a técnica do SimPoint para estimar o resultado de grandes simulações e consequentemente diminuir o consumo de memória com essas estimativas.

Outro trabalho de Lin e Yao (2015a) apresentam uma técnica para o balanceamento de carga em simuladores de eventos discretos paralelos e *multithread*. Ele utiliza uma abordagem de Q-learning para migrar as cargas das *threads* entres os processos. A aplicação da técnica permitiu uma melhoria de 31% no desempenho em *threads* dentro do mesmo processo, 21% em *threads* pertencentes a outros processos e 16% em *threads* distribuídas. Já o trabalho aqui apresentado, apresenta uma abordagem de distribuição espacial para balancear as tarefas entre os processos.

Schiera *et al.* (2020) apresentam uma plataforma de co-simulação flexível e distribuída para lidar com simulações de energia dos grandes responsáveis pelo grande consumo de energia das cidades, os edifícios.

Motlagh e Li (2019) examinam um método de modelagem de séries temporais em medidores inteligentes no contexto de cidades inteligentes. As séries temporais geradas por esses medidores apresentam uma grande variabilidade e um grande volume de dados, daí a necessidade da criação de um método de modelagem. As curvas de carga altamente granulares fornecem pistas valiosas para a compreensão dos padrões de consumo de eletricidade de uma casa, porém, os dados apresentados pelas séries apresentam uma granularidade temporal fina segundo Motlagh e Li (2019). Diante disto, eles propõem a compressão de séries para melhorar na transmissão, armazenamento, computação e outras despesas atreladas a esses dados. Em seu trabalho são apresentados alguns modelos de compressão. Um desses modelos foi apresentado por Takens (1981), o qual é o utilizado por Motlagh e Li (2019) e por este trabalho. O modelo utilizado consiste em dividir a série temporal original em m séries que rastreiam os estados do sistema, criando uma tração mais alta em comparação com a série original única.

3.3.5 Considerações Finais

Neste capítulo, foi apresentado uma revisão da literatura dos trabalhos relacionados a simulações em larga escala no contexto de cidades inteligentes. O trabalho de Motlagh e Li (2019) com uso de series temporais e o trabalho de Elbery *et al.* (2019) com a

distribuição geográfica. Embora o trabalho proposto por Motlagh e Li (2019) não apresente nenhuma técnica para simulações em larga escala, esse trabalho, assim como o trabalho aqui proposto, utiliza de séries temporais. Além dessa semelhança, o trabalho apresentado por Motlagh e Li (2019) também utilizam do mesmo modelo de compressão que este trabalho para comprimir as séries temporais. Elbery *et al.* (2019) contam com uma distribuição das tarefas através do particionamento geográfico, assim como foi feito no estudo inicial desse trabalho.

Essa revisão também mostrou que a maioria dos trabalhos selecionados estão relacionados a simuladores. Esse grupo corresponde a 52% do total. Além disso, uma grande parte dos trabalhos estão relacionados ao tráfego urbano com 40% do total.

Um trabalho que semelhante a este em sua abordagem também foi encontrado, trabalho de Hanai *et al.* (2019). Em seu trabalho os autores abordam o conceito de execução parcial da simulação para poder acelerar simulações de tráfego urbano. No entanto, o método e algoritmo utilizado difere do deste trabalho.

Além disso, um dos trabalho encontrados nessa revisão foi o trabalho apresentado por Santana *et al.* (2018), o *InterSCSimulator*. Embora sua finalidade seja ser um simulador de alta escalabilidade, ao analisar o *InterSCSimulator*, foi possível detectar alguns problemas. Nesse trabalho, serão investigadas técnicas, algoritmos e abordagens para mitigar ou resolver os problemas apresentados por esse simulador.

4 Metodologia

O principal objetivo deste trabalho é melhorar o desempenho apresentado pelo *InterSCSimulator* tal como proposto por Santana (2019), detectando e removendo os gargalos que prejudicam o desempenho da aplicação. Problemas como uso excessivo de memória ou má distribuição de tarefas entre processos em um ambiente distribuído são as causas típicas para problemas de desempenho de aplicações desse tipo, como foi apresentado na Seção 2.1. Com a melhoria do desempenho do simulador será possível a execução de simulações de cenários bem maiores como o de megalópoles como a cidade de São Paulo. Geralmente no contexto de cidades inteligentes, para experimentar suas soluções pesquisadores e profissionais da área, necessitam executar várias vezes suas simulações com diferentes parâmetros. O processo repetitivo de testar várias simulações pode se tornar algo custoso devido ao tamanho, ao tempo de simulação e as limitações apresentadas por simuladores como *InterSCSimulator*. Para isto, este trabalho propõe uma nova técnica baseada na SimPoint, denominada ***Simulation Estimation by Data Patterns Exploration*** **(SimEDaPE)**. A técnica SimEDaPE visa melhorar o desempenho do InterSCSimulator economizando no consumo de memória e reduzindo o tempo de execução de novas simulações semelhantes a simulação inicial. Além da aplicação da SimEDaPE, outras abordagens foram propostas e implementadas para melhorar o desempenho apresentado pelo simulador.

4.1 SimEDaPE: Simulation Estimation by Data Patterns Exploration

Dentre as limitações enfrentadas por aplicações, uma delas é a grande quantidade de memória necessária para executar as simulações. No caso do *InterSCSimulator*, esse problema o impossibilita de executar grandes cenários. Santana *et al.* (2018) realizaram uma estimativa e mostram que seriam necessários 515 GB para execução desse cenário, como é mostrado na Seção 2.1.

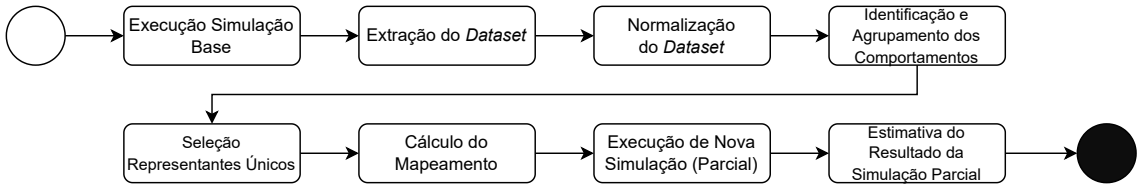
Para resolver esse problema, neste trabalho foi proposta SimEDaPE. A SimEDaPE é uma abordagem inspirada na SimPoint que é mostrada na Seção 2.2. A SimPoint é uma técnica que identifica as partes principais de uma simulação para que apenas estas partes sejam executadas e o restante da simulação seja estimada. Essa técnica foi desenvolvida

com objetivo de acelerar simulações de *hardware*.Essas simulações são usadas para testar novas arquiteturas e necessitam de semanas para serem executadas.

Com o surgimento no contexto de arquitetura de hardware. Na literatura são encontradas várias aplicações da SimPoint para acelerar essas simulações. A SimPoint tem como principal objetivo identificar as principais partes de uma simulação. Em novas simulações somente as partes principais são executadas e as demais são estimadas. Para isto, a partir dos padrões identificados na execução completa de uma simulação base é possível agrupar esses padrões em *clusters* usando algum algoritmo de clusterização. Após o agrupamento, o próximo passo é selecionar um representante único para cada um dos grupos (chamados de *simulation points*). A forma que SimPoint usa para estimar o resultado é executando somente os *simulation points* em uma nova simulação. Ao fim da execução são aplicados pesos a cada um dos *simulation points*. Os pesos são atribuídos de acordo com os *clusters* obtidos com aplicação da SimPoint na simulação base. Assim a SimPoint permite acelerar essas simulações. Assim, para ser usada é necessário a definição de métricas de interesse dos designers de arquiteturas de computadores que podem resumir o comportamento da simulação.

Este trabalho propõe uma nova abordagem inspirada na *SimPoint* de (HAMERLY *et al.*, 2005), a *Simulation Estimation By Data Patterns Exploration* (SimEDaPE). Essa abordagem é uma nova forma de aplicar a SimPoint em um novo contexto e com uso de séries temporais. Ela tem como proposta melhorar o desempenho do apresentado pelo *InterSCSimulator*. Assim como a SimPoint, a SimEDaPE tem como objetivo identificar e agrupar padrões recorrentes no decorrer da simulação para selecionar representantes únicos de cada grupo e computar o resultado dos demais. A SimEDaPE pode estimar o resultado de grandes simulações por completo a partir desses representantes e extrair métricas de interesse no contexto de cidades inteligentes. Essas métricas trazem informações importantes da simulação e ajuda os interessados a tomarem decisões a respeito de suas soluções propostas. Assim como em simulações de hardware, os pesquisadores e profissionais da área de cidades inteligentes também estão interessados em métricas que podem ser obtidas usando simulações de cidades inteligentes. Diferentemente de simulações de arquiteturas, as métricas de interesse por parte dos pesquisadores e profissionais de cidades inteligentes em simulações de tráfego urbano são velocidade média, taxa de ocupação das vias, porcentagem de veículos nas vias e entre outras métricas. A Figura 20 mostra o fluxo das etapas de execução e aplicação da técnica SimEDaPE.

Figura 20 – Etapas da execução da SimEDaPE.



Fonte: Francisco Wallison Carlos Rocha (2023).

O primeiro passo para a execução da SimEDaPE é executar uma simulação completa. Esta simulação servirá como entrada para treinar o modelo da SimEDaPE. Neste trabalho, ela será referida como simulação base. Após a execução da simulação base realizada pelo *InterSCSimulator*, a SimEDaPE extrai o *dataset* de um arquivo de eventos gerado na saída do simulador.

Após obter o *dataset*, temos os eventos que aconteceram no decorrer de toda simulação computados em séries temporais. Devido a diferença entre escala das séries temporais obtidas é necessário realizar uma normalização do conjunto de dados. Essa normalização neste trabalho é o processo de padronizar as séries temporais na mesma escala. Uma escala onde todos os eventos das séries temporais estarão presentes. Para assim, poder realizar o agrupamento das séries temporais de acordo com os seus padrões de comportamentos usando um algoritmo de clusterização.

Com os dados agrupados, uma etapa muito importante para o processo de estimação é o cálculo do mapeamento. Nesse passo, é realizado o mapeamento entre os *simulation points* e os demais elementos do *cluster* e isso permite estimar os elementos faltantes na nova simulação a ser estimada. Feito todas as etapas de pré-processamento (treinamento do modelo) da SimEDaPE, o próximo passo é executar uma simulação parcial (a qual será estimada) semelhante à primeira simulação pré-processada (a simulação base). Essa também é uma etapa crucial que determina a taxa de erro da estimativa da nossa técnica, pois, ela é nela onde serão selecionadas as viagens que serão executadas e isso implica nos dados a serem utilizados na estimativa. Por fim, com o pré-processamento e a simulação parcial executada podemos realizar a etapa de estimativa e extração de métricas usando os dados obtidos nessas etapas. Assim a SimEDaPE nos permite estimar o resultado de grandes simulações sem a necessidade de executá-las por completo, consequentemente acelerando-as.

A seguir mostraremos em detalhes cada uma das etapas executadas para que a aplicação da SimEDaPE seja possível: **Extração do *Dataset***, **Normalização do *Dataset***, **Indentificação e Agrupamento de Comportamentos**, **Seleção dos Representantes Únicos**, **Cálculo do *Warping Path***, **Execução da Simulação Parcial** e **Estimativa e extração de métricas**.

4.1.1 Extração do *Dataset*

Como visto, o primeiro passo após a execução da simulação base é a extração do *dataset*. O *dataset* é obtido a partir do processamento do arquivo de eventos gerado pelo *InterSCSimulator*. Esse arquivo contém os seguintes campos: tempo que o evento ocorreu na simulação, nome do evento, agente (pessoa ou veículo) e o *link* (rua ou parte de uma rua) onde o evento ocorreu. Um exemplo desse arquivo é mostrado no Código-fonte 7.

```
1  time,event,vehicle,link
2  3,actend,Trip-0-3600-286_1,4091
3  3,departure,Trip-0-3600-286_1,4091
4  3,PersonEntersVehicle,Trip-0-3600-286_1,4091
5  ...
6  3,wait2link,Trip-0-3600-286_1,4091
7  3,entered link,Trip-0-3600-286_1,4091
8  27,left link,Trip-0-3600-457_1,3639
9  27,entered link,Trip-0-3600-457_1,534
10 29,actend,Trip-0-3600-1490_1;2797,
```

Fonte: Francisco Wallison Carlos Rocha (2023).
Listagem 7 – Arquivo com os eventos gerados pelo *InterSCSimulator*.

Para obter as informações que compõem o nosso *dataset* estamos interessados nos eventos de entrada (*entered link*) e saída (*left link*) de cada *link* e no tempo que esses eventos ocorrem. Então, percorremos do início ao fim esse arquivo da Listagem 7 procurando os eventos de entrada e saída. Neste trabalho, esses eventos são separados e agrupados pelos *links*, ou seja, ao final do processamento do arquivo temos os eventos de entrada e saída de cada *link* computados. Esse processamento nos dá o fluxo de veículos de cada *link*.

Esses fluxos são representados pelo uso de séries temporais. As séries temporais são formadas de acordo com cada veículo que entra em um *link*, onde é computado um valor

positivo na série temporal de acordo com a sua quantidade. Do mesmo modo que sai um veículo ou mais de um *link* é computado negativamente esse valor. Ao final do processo temos o fluxo de veículos representados por séries temporais.

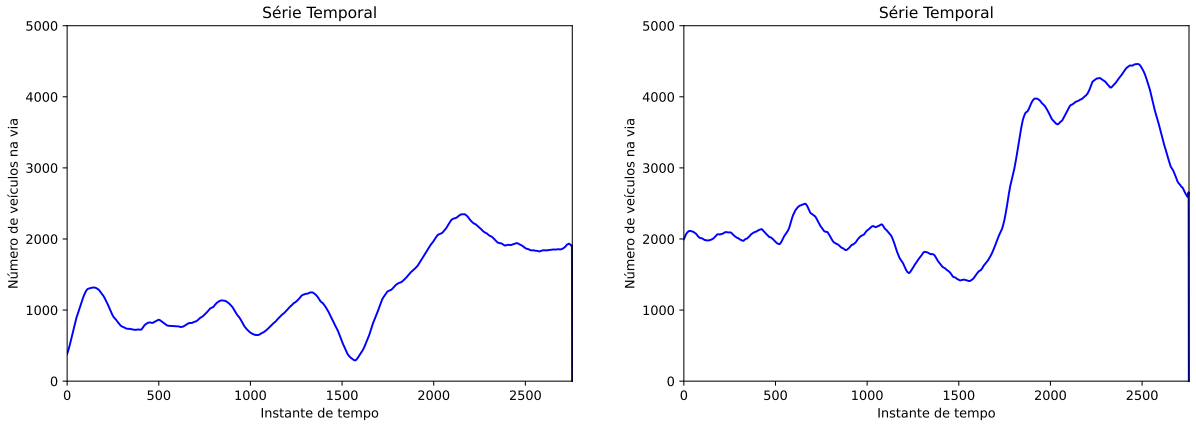
Então, após processar todo o arquivo de eventos, obtemos o fluxo de veículos de cada um dos *links* para a simulação completa. Dependendo da simulação e dos objetivos de pesquisadores e profissionais ou do processamento, depois de processados podemos dividir esse fluxo de veículos em intervalos de tempo. Por exemplo, para um simulação de 24 horas podemos dividir em intervalos de 3 horas que correspondem em algumas cidades como São Paulo aproximadamente a duração dos horários de pico (horário onde a quantidade de veículos circulando é bem maior durante o dia).

Ao dividir o fluxo de veículos de uma via para um dia inteiro em várias partes, obtemos várias séries temporais. Essas séries não possuem o mesmo tamanho. Porém, para aplicar o algoritmo de clusterização usado neste trabalho, o *K-shape* (seu uso é mostrado na Seção 4.1.3), é necessário que todas as séries temporais tenham o mesmo tamanho. Para tornar o uso do *K-shape* possível, precisamos interpolar as séries temporais para que elas tenham o mesmo tamanho da maior série temporal presente no *dataset*. Ao final de todo o processo é obtido o *dataset* para prosseguir com a aplicação da SimEDaPE.

4.1.2 Normalização do *Dataset*

A etapa de normalização do *dataset* é realizada após a extração. Esta etapa é realizada para que as séries temporais possam ser comparadas e agrupadas usando um algoritmo de clusterização. Essas séries temporais representam o número de veículos em uma via em função do tempo. Muitas séries temporais podem apresentar um formato de curva semelhante, ou seja, um comportamento semelhante. Porém, devido a quantidade de veículos presentes no intervalo de tempo do *link* que ela representa, as séries podem ser dadas como diferentes ao aplicar uma das medidas de distâncias usadas na clusterização. Ao aplicar a normalização é possível manter propriedades como o formato da série temporal. Na Figura 21 são mostradas duas séries que demonstram formatos de curvas semelhantes. No entanto, a volumetria de veículos presentes em cada uma no instante de tempo são diferentes.

Figura 21 – Comparação de séries em relação a escala.



(a) (b)

Fonte: Francisco Wallison Carlos Rocha (2023).

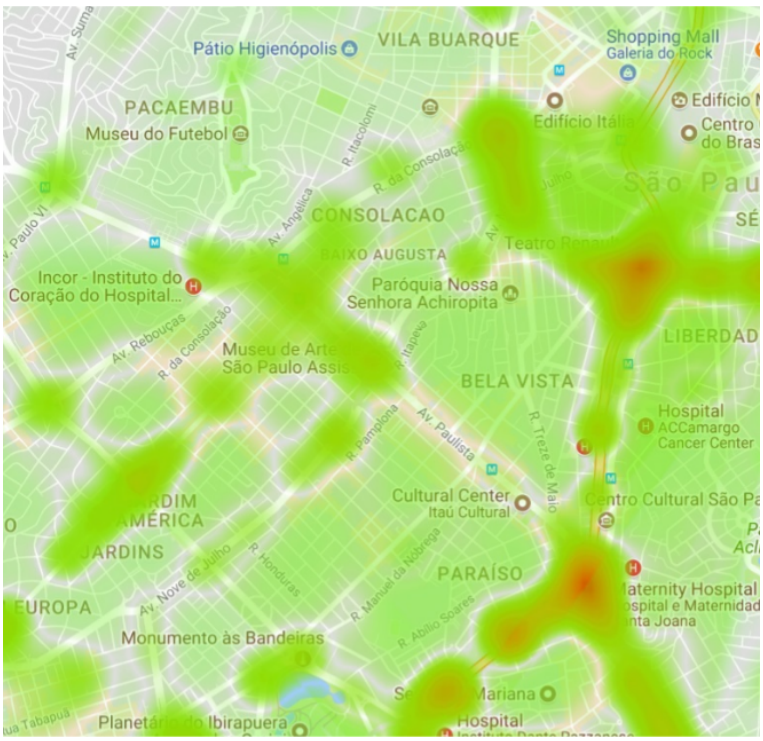
Então, para comparar séries com comportamentos semelhantes como estas, é aplicada uma normalização usando a *z-score*. A *z-score* converte os valores dos eventos das séries para um faixa de valor entre -4 e 4 mantendo o formato delas.

$$w' = \frac{w - \overline{x}}{\sigma} \tag{13}$$

4.1.3 Identificação e Agrupamento de Comportamentos

No decorrer de toda a simulação é possível identificar padrões comportamentais relacionados ao fluxo de veículos nas vias da cidade simulada. O mapa de calor em uma região simulada da cidade de São Paulo retrata bem esses padrões de comportamento. Na Figura 22, as ruas com uma cor mais quente demonstram uma quantidade de veículos maior naquele instante de tempo Várias outras ruas com cores mais frias mostram um número menor de veículos trafegando naquele determinado intervalo de tempo. Esse tipo de comportamento é visto no decorrer de toda a simulação. Um outro ponto que se nota, é que esse mapa de calor é formado justamente pelo fluxo de veículos nas vias. Ao notar esse aspecto, é possível extrair esses fluxos de veículos em forma de séries temporais para representar esses comportamentos. Além disso, também é possível segmentar esses comportamentos das vias por intervalo de tempo. Como se pode ver é exatamente o conteúdo e formato do nosso *dataset*.

Figura 22 – Mapa de calor de parte uma cidade cidade, onde as regiões com cores mais quentes mostram ruas com maior quantidade de veículos.

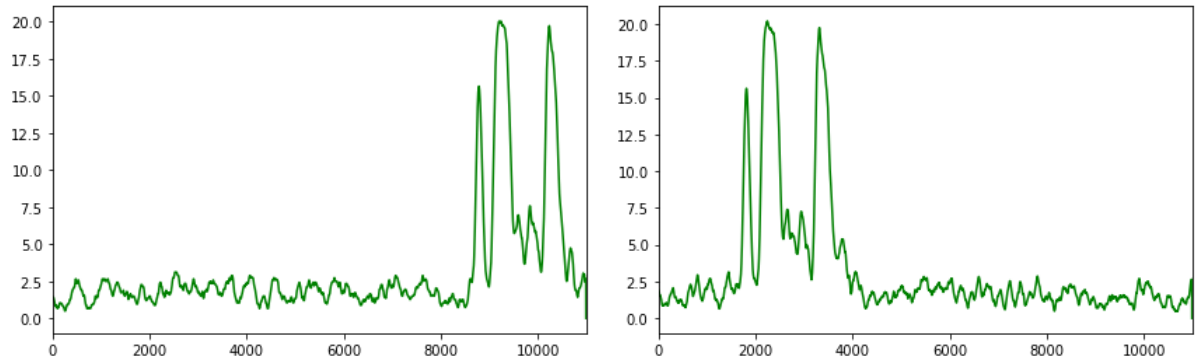


Fonte: (SANTANA, 2019).

Após a extração do *dataset* e a normalização das suas séries temporais, podemos realizar o agrupamento dos comportamentos pela sua semelhança. Um exemplo desses comportamentos semelhantes está expresso na Figura 23. A Figura 23a representa uma série com 3 grandes picos de veículos na via no final do intervalo e o restante do tempo mais suave em relação a esses instantes de tempo. Já na Figura 23b que representa uma outra série temporal, onde acontece justamente o contrário, o tempo mais suave acontece depois da ocorrência de 3 picos. Ainda que distintas, essas séries possuem comportamentos bastante semelhantes: três momentos de pico seguidos (ou precedidos) por momentos de estabilidade no fluxo.

Para realizar essa etapa serão utilizados algoritmos de machine learning, no caso, os de clusterização. No trabalho da SimPoint foi utilizado o algoritmo *K-means* com a Distância de Manhattan. Neste trabalho, será utilizado um outro algoritmo de clusterização semelhante ao *K-means*, o *K-shape* (PAPARRIZOS; GRAVANO, 2015). O *K-shape* recebe esse nome devido a medida de distância utilizada por ele, a *Shape Based Distance* (SBD). A SBD é uma medida própria para calcular a similaridade entre séries temporais. Assim, dado o contexto e o formato dos dados do nosso *dataset*, optou-se pelo seu uso. O uso de

Figura 23 – Séries agrupadas no mesmo *cluster* 86 usando o algoritmos de clusterização *K-shape*



(a) (b)

Fonte: Francisco Wallison Carlos Rocha (2020).

medidas distâncias comuns como a Distância Euclidiana ou a Manhattan, não conseguem identificar semelhanças tão bem entre séries como as das Figuras 23a e 23b. Além disso, em seu trabalho, Paparrizos e Gravano (2015) mostra a acurácia e tempo de processamento do uso do *K-shape* em relação ao uso de outras medidas de distância com o *K-means* tanto para comparar dados mais gerais quanto para séries temporais como a *Dynamic Time Warping* (DTW).

4.1.4 Seleção dos Representantes Únicos

Após agrupar os componentes de acordo com sua semelhança e padrões de comportamentos, a próxima etapa é selecionar um representante único para cada um desses grupos. Assim como trabalho original sobre SimPoint, neste trabalho também denominamos esse representante único de *Simulation Point*. Uma outra semelhança com a SimPoint é que também são usados os centroides dos *clusters* para determinar o representante único. O centroide é o elemento central da clusterização usado para medir e determinar o quão próximo o elemento é daquele grupo. Ele é usado para determinar se a série temporal deve ou não está em seu *cluster*.

No entanto, no nosso contexto, o representante a ser selecionado é uma rua, avenida ou parte delas. O centroide em alguns casos é formado pela média dos elementos do seu grupo, ou seja, não é a série temporal do nosso *dataset*. Assim, precisamos selecionar uma outra série temporal. Embora o centroide não seja uma série presente no nosso *dataset*,

podemos usar ele para selecionar a série temporal ideal do grupo, o nosso representante único.

Para determinar qual série temporal será o representante único procuramos a série que mais se assemelha com o centroide. Para isso, é possível usar uma medida de distância para comparar séries temporais como a *Dynamic Time Warping* (DTW) ou a *Shape Based Distance*.

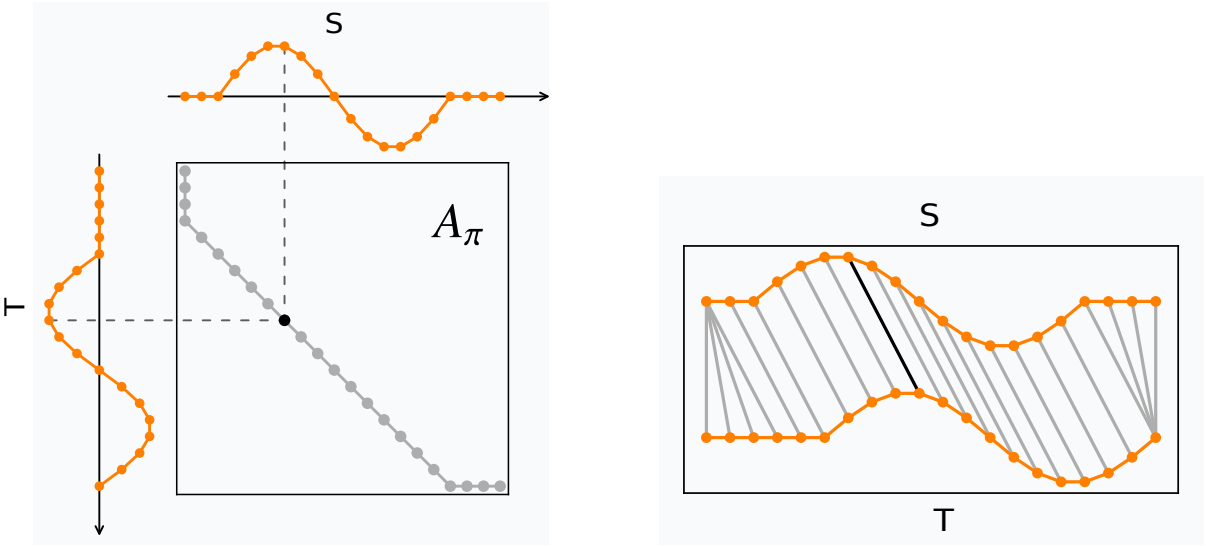
4.1.5 Cálculo do Mapeamento (*Warping Path*)

Após a realização da seleção dos *simulation points*, agora é possível realizar o cálculo do mapeamento, o *Warping Path* (WP). Dado duas séries temporais S e T arranjadas em um plano ou *grid* n-por-n, onde cada ponto do *grid* (i, j) corresponde ao alinhamento dos elementos s_i e t_j. O WP é o mapeamento ou alinhamento desses elementos de S e T, de modo que a distância entre elas seja minimizada. O WP é obtido a partir da aplicação da *Dynamic Time Warping* (DTW) (BERNDT; CLIFFORD, 1994).

A DTW também pode ser usada como medida para comparar a similaridade entre séries temporais. A DTW usa de programação dinâmica (Algoritmo 1) para calcular todas as distâncias entre os todos os pontos s_i da série temporal S e todos os pontos t_j da série temporal T. O calculo de todas as distâncias resulta em uma matriz com os todos os valores calculados. A partir dessa matriz é possível selecionar a melhor combinação (pontos com a menor distância entre S e T) de todos os pontos. Então, através da aplicação da DTW é possível obter o WP.

A Figura 24 mostra como é calculada a matriz das distâncias entre todos os pontos de duas séries temporais S e T. No eixo x está a série S e no eixo y está a série T. Destacado em cinza as coordenadas para as associações do que seriam as menores distâncias entre os pontos das séries S e T, ou seja, o WP. Já na Figura 24b é mostrado o WP por um outro ângulo, mostrando como ficou o mapeamento final das duas séries. Além disso, em destaque o ponto preto na Figura 24a que corresponde ao mapeamento em preto da Figura 24b, onde mostra a distância entre os pontos e o seu deslocamento no tempo.

Figura 24 – Matriz calculada usando DTW (24a) e mapeamento (*Warping Path*) resultante (24b).

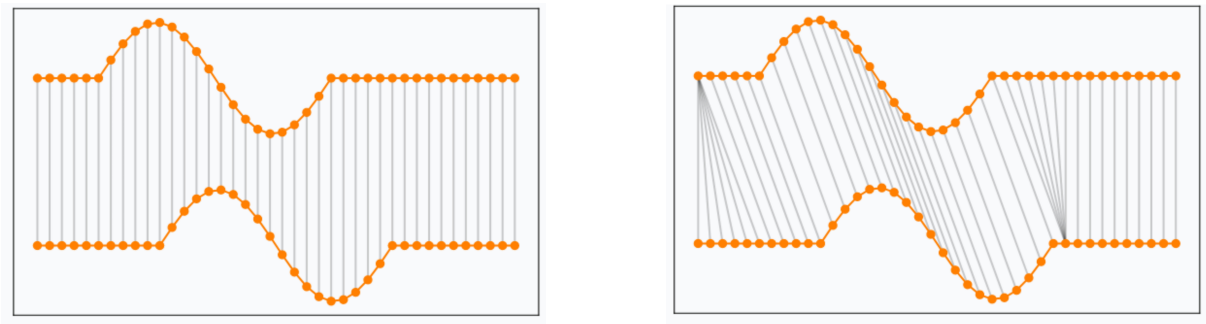


(a) (b)

Fonte: ⟨https://rtavenar.github.io/blog/dtw.html⟩ (2023).

A opção pelo uso da *Dynamic Time Warping* em detrimento de outras medidas de distância é mostrada na Figura 25. A Figura 25 mostra visualmente a diferença entre o uso de uma distância simples como a distância Euclidiana (Figura 25a) e uma distância como a DTW que nos dá o *Warping Path* (Figura 25b). Nesse caso, pode-se ver que a distância Euclidiana não considera que os eventos podem estar deslocados no tempo, diferentemente da DTW.

Figura 25 – Mapeamento usando a distância Euclidiana (25a) e mapeamento (*Warping Path*) usando a DTW (25b).



(a) (b)

Fonte: ⟨https://rtavenar.github.io/blog/dtw.html⟩ (2023).

O *Warping Path* oferece um mapeamento com a distorção temporal entre duas séries temporais. Esse mapeamento é uma lista de 4-tuplas no seguinte formato (i, j, d, b),

onde i corresponde ao instante de tempo i em S, j ao instante de tempo j em T, d corresponde a distância entre os elementos s_i e t_j, e b é um bit de controle que nos diz qual equação usar na estimativa de acordo com os valores dos elementos das posições i e j. Se o elemento s_i for maior que t_j então o valor é 1, caso contrário é 0. O *Warping Path* é utilizado para mapear os *simulation points* para todas as séries temporais dos seus respectivos *clusters*. A Seção 4.1.7 mostra o uso dos *Warping Paths* calculados no processo de estimativa.

4.1.6 Execução da Simulação Parcial

Uma simulação parcial neste trabalho é uma simulação na qual nem todas as viagens da simulação completa são executadas, somente uma amostra. O desempenho da acurácia da estimativa da simulação realizada pela SimEDaPE depende diretamente das viagens selecionadas para execução da simulação parcial. O motivo disso é que uma quantidade menor de viagens selecionadas pode fazer com que a SimEDaPE venha a apresentar um erro maior na estimativa. Além disso, dependendo das viagens selecionadas pode-se obter uma construção dos *simulation points* melhor ou pior, ou até mesmo a sua não construção, impossibilitando a estimativa das séries que o *simulation point* não gerado representava.

Diante disso, este trabalho também apresenta algumas alternativas de algoritmos para a geração de uma nova simulação parcial: Seleção Aleatória de Viagens, Seleção Aleatória de Viagens Divididas por *Cluster*, Max, Min e Seleção Aleatória de Viagens Baseada nos *Simulation Point*.

Seleção Aleatória de Viagens

Este algoritmo de geração de cenário, como o próprio nome já diz, é um algoritmo que seleciona viagens de maneira aleatória. Dado um conjunto V de viagens, esse algoritmo seleciona q viagens de forma aleatória. Onde q representa o total de viagens da amostra definido inicialmente.

Seleção Aleatória de Viagens Divididas por *Clusters*

Com a finalidade de amenizar as deficiências apresentadas pelo Algoritmo Seleção Alearória de Viagens, o Seleção Aleatória de Viagens Divididas por *Clusters* (SADC) usa uma abordagem um pouco diferente para selecionar as viagens de maneira aleatória. Aqui as viagens são divididas em n subconjuntos de V viagens de acordo com as séries presentes em cada *cluster*, onde n é o número total de *clusters*. Por exemplo, todas as viagens presentes no subconjunto V_i, são viagens que participaram na formação das séries temporais presentes no *cluster* c_i. A seleção de viagens funciona da seguinte forma: a cada iteração é selecionada uma viagem de maneira aleatória de cada um dos conjuntos em sequência e adicionar ao conjunto P de viagens da simulação parcial. Esse processo é repetido até que a quantidade q de viagens da simulação pré-definida seja satisfeita.

A intenção por trás deste algoritmo é aumentar a probabilidade de selecionar viagens que contribuam para a formação dos *simulation points*, uma melhoria do Algoritmo Seleção Aleatória de Viagens. No entanto, ainda é aleatório e não se tem o total controle sobre a seleção das viagens, o que implica no mesmo risco do Algoritmo Seleção Aleatória de Viagens, mas menor.

Max e Min

Com a intenção de dar prioridade a formação do *simulation points* que representavam uma maior quantidade de séries temporais na simulação, o Algoritmo Max foi criado. Assim como o Algoritmo Aleatório por *Cluster* , esse algoritmo também baseia-se nas séries presentes em cada *cluster* para compor os conjuntos de viagens. Além dessa informação, o Max também necessita da quantidade de elementos presentes em cada um dos *clusters*. Desse modo, o Max não irá selecionar as viagens de cada conjunto seguindo a ordem predefinida dos *clusters*, mas sim, pela representatividade de cada um dos conjuntos na simulação em relação a quantidade de séries que estão presentes em cada *cluster*.

No Algoritmo 2, uma das grandes diferenças entre o Max e o Seleção Aleatória de Viagens Divididas por *Clusters* é o uso da função `ordena_por_tamanho_desc`. Essa função permite ordenar os conjuntos em ordem decrescente de acordo com as séries de cada *cluster*. Para isso ela precisa receber a lista S dos tamanhos de cada um dos *clusters*. Onde s_i é o

tamanho correspondente ao *cluster* c_i. A outra grande diferença entre este algoritmo e o Seleção Aleatória de Viagens Divididas por *Clusters* é que são adicionadas todas as viagens de um conjunto antes de passar para o conjunto subsequente. Desse modo, caso não seja possível atingir a quantidade de viagens pré-definida com as viagens presentes em O_i, O_{i+1} será o próximo a ser processado. Assim, são adicionadas (usando a função `adicionar`) ao conjunto P de viagens da simulação parcial as viagens dos conjuntos em ordem decrescente que são selecionadas de maneira aleatória usando a função `seleciona_viagem_aleatoria`. Esse processo é repetido até que a quantidade q de viagens da simulação pré-definida seja satisfeita ou não haja mais viagens nos subconjuntos de O. Um ponto importante, é que não necessariamente as viagens são selecionadas aleatoriamente, ou seja, podem ser selecionadas em sequência.

Algoritmo 2 Max

```
   Entrada: V ← {V_0, V_1, ..., V_{n-1}, V_n}, q, S ← {s_0, s_1, ..., s_{n-1}, s_n}
   Saída: P
O ← ordena_por_tamanho_desc(V, S)
while tamanho(P) < q do
    for i = 0..n do
        while O_i ≠ ∅ do
            v ← seleciona_viagem_aleatoria(O_i)
            P ← adicionar(P, v)
            if tamanho(P) ≥ q then
                return P
return P
```

Embora o Max vise compor na simulação parcial a maior quantidade de viagens significativas da simulação, esse algoritmo pode deixar de gerar uma grande quantidade de *simulation points* e demais séries temporais, justamente por não incluir os demais clusters dependendo do valor de q pré-definido.

A fim de estudo, também foi proposto o Algoritmo Min. O Min é semelhante ao Max, a diferença é que ele seleciona o *clusters* em ordem crescente. Para implementar o algoritmo Min basta adaptar o Algoritmo 2 substituindo a função `ordena_por_tamanho_desc` por uma outra que ordene os conjuntos em ordem crescente. Adicionando as viagens dos conjuntos correspondentes aos clusters de menor volumetria. A ideia por trás do Min é incluir o maior número de *clusters* possível, ou seja, formar o maior número de *simulation points* possível. No entanto, como esperado, o Algoritmo Min apresenta um comportamento semelhante ao Max, assim, apresentando as mesmas limitações.

Baseado em *Simulation Point*

Uma outra aboragem para selecionar as viagens é dando prioridade para as viagens que compõem os *simulation points*. Nessa abordagem, as viagens que são responsáveis por formar os *simulation points* têm uma prioridade maior em relação às demais viagens. Então, essas viagens são inseridas primeiro no conjunto que será executado.

Essa abordagem tem como objetivo selecionar viagens que estejam diretamente ligadas a formação dos *simulation points* na simulação parcial. Ela modifica os algoritmos Seleção Aleatória de Viagens e Seleção Aleatória de Viagens Divididas por *Clusters* para criar outras possibilidades de algoritmos: Seleção Aleatória de Viagens Com Base nos *Simulation Points* e o Seleção de Viagens Aleatórias Dividas por *Cluster* Com Base nos *Simulation Points*.

Seleção Aleatória de Viagens Com Base nos *Simulation Points*

O algoritmo Seleção Aleatória de Viagens Com Base nos *Simulation Points* (SABSP) consiste em dividir as viagens em dois conjuntos. O conjunto R das viagens que formam os *simulation points* e o conjunto Q das demais viagens. Então são selecionadas viagens aleatoriamente do conjunto R até que a quantidade q de viagens pré-definidas seja suprida, ou até que todas as viagens do conjunto R sejam selecionadas. Quando as viagens do R se esgotarem e a quantidade pré-requisita para a simulação parcial ainda não tiver sido atingida, serão selecionadas de maneira aleatória as viagens do conjunto Q para atingir a quantidade desejada. Para selecionar as viagens de modo aleatório do conjunto Q usamos o Algoritmo de Seleção Aleatória de Viagens que é implementado pela função `algoritmo_aleatorio`. Ao final do processo, temos o conjunto de viagens parciais selecionadas de maneira aleatória com as viagens dos *simulation points* priorizadas, como é mostrado no Algoritmo 3.

Desse modo é possível garantir que os *simulation points* sejam gerados eliminando os problemas com o Aleatório Seleção Aleatória de Viagens, mas ainda não todos dependendo do tamanho da simulação parcial.

Algoritmo 3 Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Entrada: R, Q, q
Saída: P
while `tamanho`$(P) < q$ **E** $R \neq \emptyset$ **do**
  $v \leftarrow$ `seleciona_viagem_aleatoria`(R)
  $P \leftarrow$ `adicionar`(P, v)
$P \leftarrow$ `adicionar`$(P,$ `algoritmo_aleatorio`$(Q, q))$
return P

Seleção de Viagens Aleatórias Divididas por *Cluster* Com Base nos *Simulation Points*

O algoritmo apresentado nesta seção, assim como o Seleção Aleatória de Viagens Com Base nos *Simulation Points*, é uma melhoria do Seleção Aleatória de Viagens Divididas por *Clusters*. No caso do Seleção de Viagens Aleatórias Divididas por *Cluster* Com Base nos *Simulation Points* (SADCBSP), as viagens estão divididas em conjuntos baseados nos *clusters*.

No Algoritmo 4, é usado a mesma lógica do Aleatório por *Cluster*. No entanto, as primeiras viagens a serem incluídas no conjunto de P de viagens devem ser as dos subconjuntos dos *simulation points*. Como as viagens dos *simulation points* estão divididas por *clusters*, é possível usar o Algoritmo Seleção Aleatório de Viagens Divididas por *Cluster* implementado pela função `algoritmo_aleatorio_por_cluster`. Então, o Algoritmo 4 recebe conjunto R com os i subconjuntos R_i das viagens das séries temporais que compõem os *simulation points* de cada *cluster*. O Aleatório por *Cluster* Baseado em *Simulation Points* por *Cluster* também recebe o conjunto Q dos subconjuntos Q_i das viagens que compõem as demais séries temporais.

Algoritmo 4 Seleção de Viagens Aleatórias Divididas por *Cluster* Com Base nos *Simulation Points*

Entrada: $R \leftarrow \{R_0, R_1, ..., R_{n-1}, R_n\}, Q \leftarrow \{Q_0, Q_1, ..., Q_{n-1}, Q_n\}, q$
Saída: P
$P \leftarrow$ `adicionar`$(P,$ `algoritmo_aleatorio_por_cluster`$(R, q))$
$P \leftarrow$ `adicionar`$(P,$ `algoritmo_aleatorio_por_cluster`$(Q, q))$
return P

4.1.7 Estimativa e Extração de Métricas

A última etapa de execução da SimEDaPE é a realização da Estimativa e Extração de Métricas. Essa etapa pode ser divida em duas etapas menores, Estimativa das Séries Temporais e Extração de Métricas. A estimativa consiste em estimar as demais séries temporais não geradas na simulação parcial. Já a extração de métricas consiste em obter as métricas como velocidade média, a porcentagem da taxa de ocupação das vias, entre outras, a partir das séries temporais estimadas. A seguir as duas etapas menores são mostradas em detalhes.

Estimativa das Séries Temporais

Após executar a simulação parcial gerada usando um dos algoritmos descritos na Seção 4.1.6, o próximo passo é estimar o restante da simulação parcial. Até o momento, temos séries temporais com rótulos correspondentes a séries temporais da simulação original, é importante ressaltar que nem todas as séries foram geradas como foi visto anteriormente na seção de execução da simulação.

Para realizar o processo de estimativa das séries temporais iremos precisar de alguns componentes: as séries geradas na simulação parcial, os *Warping Paths* calculados e as médias aritméticas e os desvios padrão.

As séries geradas nessa nova simulação parcial são utilizadas como componentes para gerar uma estimativa do restante da simulação. Principalmente as séries temporais geradas nessa nova simulação que correspondem aos *simulation points* da simulação base. Como nem todas as séries temporais podem ser geradas na simulação, podemos não ter todos os *simulation points*, impossibilitando assim, de estimar as séries temporais as quais ele representa. Além disso, dependendo da abordagem de utilização de média e desvios padrão podemos não conseguir estimar todas as séries como explicamos mais a frente.

Os *Warping Paths* que foram calculados dos *simulation points* para as demais séries do seu cluster, agora são usados para transformar os *simulation points* gerados nesta nova simulação parcial. Para realizar a distorção e transformar os *simulation points* correspondentes nas demais séries usamos a Equação 14. A Equação 14 mostra como o ponto de simulação é deformado para estimar cada série, onde u_j é o elemento na posição

j na série a ser estimada, d_{ij}^2 é a distância entre o elemento c_i no *cluster* e o elemento u_j na série.

$$u'_j = \begin{cases} c_i - \sqrt{d_{ij}^2 - d_{ij-1}^2}, \text{ if } u_j \geq c_i \\ \sqrt{d_{ij}^2 - d_{ij-1}^2} + c_i, \text{ if } u_j < c_i \end{cases} \tag{14}$$

Além disso, é necessário aplicar a desnormalização para retornar a série à sua amplitude original. Isto porque antes de realizarmos a clusterização no treinamento do modelo precisamos aplicar uma normalização em todas as séries temporais, aplicando a *z-score*. Essa normalização tem por finalidade permitir que comparemos todas as séries na mesma dimensionalidade. Aplicando essa normalização obtemos séries temporais com valores fora do nosso contexto, por exemplo, com valores negativos. Nesse caso, não existe uma quantidade de veículos negativa em uma via. Então, assim realizamos uma desnormalização para termos os valores originais das séries temporais. Para isso é usado a Equação 15. Onde w é a série temporal desnormalizada, w' é a série temporal normalizada, σ é a o desvio padrão da série, $\overline{x}$ é a média. O mesmo desvio padrão e média usada para normalizar as séries são armazenados para serem usados nesse momento de estimativa.

$$w = w' \cdot \sigma + \overline{x} \tag{15}$$

Além do uso das médias e desvios padrão da simulação base (SB) pode se usar as médias e os desvios padrão da simulação parcial (SP) com a finalidade de substituir as da SB para ter mais dados próximos da SP e melhorar a sua estimativa. No caso de uso das médias e desvios padrão da simulação parcial, corremos o risco de não conseguir estimar todas as séries temporais, pois nem todas as séries temporais são geradas na SP, logo consequentemente não teremos as médias e desvios padrões. Assim sendo, temos algumas possibilidades de usar as médias e desvios padrão para realizar as estimativas: simulação base, simulação parcial, média aritmética, média ponderada e híbrido.

No uso da simulação base, se tivermos com todos os *simulation points* gerados na SP, é possível estimar todas as séries temporais da nova simulação semelhante. Ao usar os dados da simulação parcial corremos o risco de além de não estimar por falta de *simulation points*, não estimar por conta de algumas séries que não foram geradas na SP, porém, o intuito é melhorar a estimativa da nova simulação. No entanto, nem sempre a série temporal estará completa, pode ser uma aproximação, dado que a simulação é parcial. Assim, a média e o desvio padrão não será o mesmo de executar a nova simulação por

completo. Desse modo, tentando amenizar esse problema com a parcialidade, foi proposto o uso da média aritmética entre as médias aritméticas e desvios padrões da simulação base e parcial. Desse modo, realizamos uma modificação na Equação 15 como mostra a Equação 16. Onde σ_b é o desvio padrão da simulação base e σ_p da simulação parcial e $\overline{x}_b$ a média da simulação base e $\overline{x}_p$ a média da simulação parcial.

$$w = w' \cdot \frac{\sigma_b + \sigma_p}{2} + \frac{\overline{x}_b + \overline{x}_p}{2} \tag{16}$$

Com o intuito de melhorar a estimativa feita pela média aritmética foi proposto o uso da média ponderada. Nesse caso são atribuídos pesos de acordo com a porcentagem da simulação executada. Para os dados da simulação base é atribuído um peso de 1 (100%) e para os dados da simulação parcial para uma porcentagem de 45% é atribuído o mesmo peso 0.45 (45%), como mostra a Equação 17. Onde p_1 é o peso da simulação completa e p_2 da parcial.

$$w = w' \cdot \frac{\sigma_b \times p_1 + \sigma_p \times p_2}{p_1 + p_2} + \frac{\overline{x}_b \times p_1 + \overline{x}_p \times p_2}{p_1 + p_2} \tag{17}$$

No entanto, embora use dados da SB essas abordagens de média aritmética e ponderada apresentam a mesma deficiência da simulação parcial, nem todas as séries temporais são geradas, assim não podemos estimar essas séries não geradas. Para contornar esse problema de séries temporais não geradas e mesmo assim usar os dados da simulação parcial, podemos usar a abordagem híbrida. Nessa abordagem, usamos os dois casos, tanto dados da base quanto da parcial. Quando a série temporal não é gerada na SP, usamos os dados da SB. Também, podemos implementar uma das médias quando temos os dados em ambas.

Com o modelo treinado, a simulação parcial gerada usando um dos algoritmos e executada e estratégia de desnormalização selecionada, é possível estimar o resultado da nossa simulação, estimando as séries temporais. Para isso aplicamos a Equação 14 para estimar as demais séries a partir dos novos *simulation points* gerados. Depois da deformação usando o *Warping Path* aplicamos uma das equações de desnormalização. Com toda nova simulação estimada, agora podemos extrair as métricas.

Extração de Métricas

Além de permitir realizar estimativas de simulações, a SimEDaPE também permite extrair métricas dessa simulação estimada. Afinal as métricas são o grande interesse dos pesquisadores e profissionais da área quando se executam simulações. Neste trabalho, algumas métricas foram estudadas no contexto da simulação realizada pelo *InterSCSimulator* que é de nível mesoscópico. As métricas de interesse de estudo neste trabalho foram velocidade média, porcentagem da taxa de ocupação da vias e ocupação média das vias. Para calcular cada uma dessas métricas usamos as séries temporais onde em cada um dos seus respectivos eventos temos a quantidade de veículos presente na via, ou seja o fluxo de veículos. Esse fluxo de veículos (série temporal) nos permite extrair essas métricas.

Para calcular a velocidade média usamos uma fórmula própria para calcular velocidade média em simulações de nível mesoscópico mostrada na equação 18. Essa equação é a mesma usada no simulador *InterSCSimulator* do trabalho de Santana (2019). Onde V_m é a valeciodade média do *link*. Onde V_m é a média gemométrica de todas as velocidades médias calculadas para cada instante de tempo da série temporal. N é o total de instantes de tempo, vl é a velocidade livre (quando os veículos trafegam livremente na via), n_i é o número de carros presentes na via no instante de tempo de tempo i e c a capacidade de veículos da via. Além disso, α e β recebem um como valor padrão nesse caso 1.

$$V_m = \sqrt[N]{\prod_{i=0}^{N} vl \times (1 - (\frac{n_i}{c})^{\beta})^{\alpha}} \tag{18}$$

O cálculo da porcentagem da taxa de ocupação da via pode ser calculada usando a Equação 19. Onde P_o porcentagem da taxa de ocupação da via é a média geométrica das porcentagens de ocupação de cada instante de tempo. N é a quantidade de instantes de tempo i, n_i quantidade de veículos na via no instante de tempo i e c é a capacidade total da via. Tirando a média de todos os instantes de tempo temos a porcentagem da taxa de ocupação média daquela via em um determinado intervalo de tempo.

$$P_o = \sqrt[N]{\prod_{i=0}^{N} \frac{n_i}{c} \times 100} \tag{19}$$

A média de veículos em uma via em um determinado intervalo de tempo é um pouco mais simples de se calcular. Para fazer esse cálculo basta realizar o somatório da

quantidade de veículos n_i de todos os e instantes de tempo i do intervalo de tempo e dividir pelo total de instantes de tempo N como mostra a Equação 20.

$$M_v = \frac{\sum_{i=0}^{N} n_i}{N} \tag{20}$$

4.2 Outras Propostas e Implementações de Melhoria

Além da principal técnica apresentada neste trabalho, a SimEDaPE. Aqui também são apresentadas outras propostas e outras implementações de melhorias diretamente no simulador. Essas melhorias visam melhorar o desempenho do *InterSCSimulator* quanto permitir que ele consiga executar e lidar com grandes volumes de dados.

A primeira grande melhoria foi na modificação do tamanho do nome identificador das viagens. Uma outra grande melhoria realizada foi na mudança da criação dos atores das ruas. Além dessas melhorias, foi necessário mudar a forma que o arquivo de viagens era carregado e processado na memória. Essas foram algumas mudanças simples que nos permitiu ter grandes ganhos e também executar grandes simulações para aplicar a técnica da SimEDaPE.

Tamanho do Nome do Identificador das Viagens

Uma simples melhoria realizada no conjunto de dados a ser processado foi a mudança do tamanho do nome identificador das viagens do arquivo carregado, esse arquivo é mostrado e descrito na Seção 2.1. Na simulação, as viagens possuem um identificador único para que a viagem possa ser identificada no decorrer de toda a simulação. No caso de um arquivo de um dos cenários executados, esse identificador tinha um formato de um Identificadores Universalmente Únicos (do inglês *Universally Unique Identifiers*, ou UUIDS). Os UUIDS são *strings* que contém 36 caracteres que incluem letras, números e hífens.

A troca desses identificadores UUIDS por identificadores sequenciais ou hexadecimais, embora seja algo muito simples, em grandes simulações com milhões de agentes/atores faz uma grande diferença na economia de memória.

Criação dos Atores das Ruas

Um outro problema encontrado para executar o *InterSCSimulator* foi a criação dos atores das ruas para um grande mapa de uma cidade ou região. Na implementação do InterSCSimulator, é carregado o mapa em um formato de arquivo XML. Neste arquivo, contém uma lista de nós (conexões entre os *links*) e uma lista de *links* (que representam ruas ou partes das ruas), esse arquivo é mostrado e descrito na Seção 2.1.

Esse arquivo com nós e *links* que juntos formam um grafo que representa o mapa da cidade ou região é carregado e processado. O processamento resultante tem como saída uma lista com todas as ruas do mapa. Para gerenciar essa lista de ruas, o *InterSCSimulator* criava um único ator. No entanto, devido ao tamanho do mapa e consequentemente da lista de ruas ser muito grande, o simulador não estava conseguindo criar esse ator. A consequência disso era que o ator principal da simulação, chamado de *Manager*, recebia um sinal de erro do Erlang de `noconnection`. Esse erro acontece por conta que uma conexão entre nós foi interrompida ou no caso não foi estabeleciada[1].

Para contornar esse problema, este trabalho usou a abordagem de dividir o gerenciamento desse conjunto de viagens por vários atores. Uma alternativa foi criar um ator para gerenciar cada uma das vias. Uma outra alternativa a se usar é criar menos atores para gerenciar um conjunto maior de vias, com uma relação de 1-para-n (um ator para n listas, onde n é menor que a quantidade total de vias).

Carregamento de Viagens

Além das melhorias mostradas nas seções anteriores, uma outra melhoria realizada foi no carregamento das do arquivo de viagens e gerenciamento dessas viagens carregadas na memória. A depender do tamanho do arquivo de viagens não é possível carregá-lo por completo na memória. Isto ocorre não por conta da quantidade de memória disponível, mas sim por limitações da biblioteca de carregar arquivos XML utilizada e da linguagem ao carregar grandes *strings*, pois o conteúdo é carregado em uma grande *string* antes de ser processado e transformados em elementos estruturais da linguagem, os objetos. Desse modo o arquivo não é carregado sem erros que impeçam seguir a execução, porém, nenhuma viagem é carregada.

[1] ⟨https://www.erlang.org/doc/reference_manual/errors.html⟩

Desse modo, uma solução para esse problema utilizada neste trabalho foi o carregamento em lotes. Este trabalho divide o grande arquivo de viagens em vários arquivos menores que podem ser carregados na memória usando essa a biblioteca XML atualmente usada pelo simulador. Isto permite carregar todas as viagens e transformá-las em objetos da linguagem, gerando assim uma lista.

Além disso, um outro ponto de melhoria identificado, foi que após a criação dos atores das viagens, a lista de viagens não é mais utilizada no decorrer da simulação. Como deve-se imaginar é que devido ao seu tamanho desse arquivo carregado, a lista também tem um grande tamanho e ocupa um espaço considerável na memória. No entanto, essa lista não é mais utilizada o *Garbage Collection* do Erlang não realiza a limpa dessa lista. Isto pode estar ocorrendo pelo fato que a simulação é executada dentro de uma única função chamada `run` onde se encontra essa lista. Como uma variável uma vez definida em Erlang o seu valor não pode ser alterado com algo vazio, ou algo que indique ao *Garbage Collection* para fazer a coleta. Este trabalho adota como alternativa usar o escopo de funções para carregar as viagens, processar a lista e criar os atores. Desse modo, ao colocar essas ações dentro de várias funções, ao fim de cada e ao retorno para função `run` é possível que essa memória usada seja desalocada. Assim, conseguimos economizar e melhorar o uso de memória na simulação, ganhando espaço para executar simulações maiores.

5 Experimentos e Resultados

Para validar o desempenho computacional proporcionado ao *InterSCSimlator* obtido através aplicação da técnica SimEDaPE e de outras melhorias realizadas no simulador. Além de experimentos realizados para medir os ganhos com o desempenho do simulador, foram realizados experimentos com a finalidade de medir a acurácia da estimativa da técnica SimEDaPE. Na Seção 5.1, são mostrados os resultados referentes aos ganhos de desempenho do *InterSCSImulator* ao se aplicar a SimEDaPE e a acurácia da estimativa da técnica.

5.1 SimEDaPE

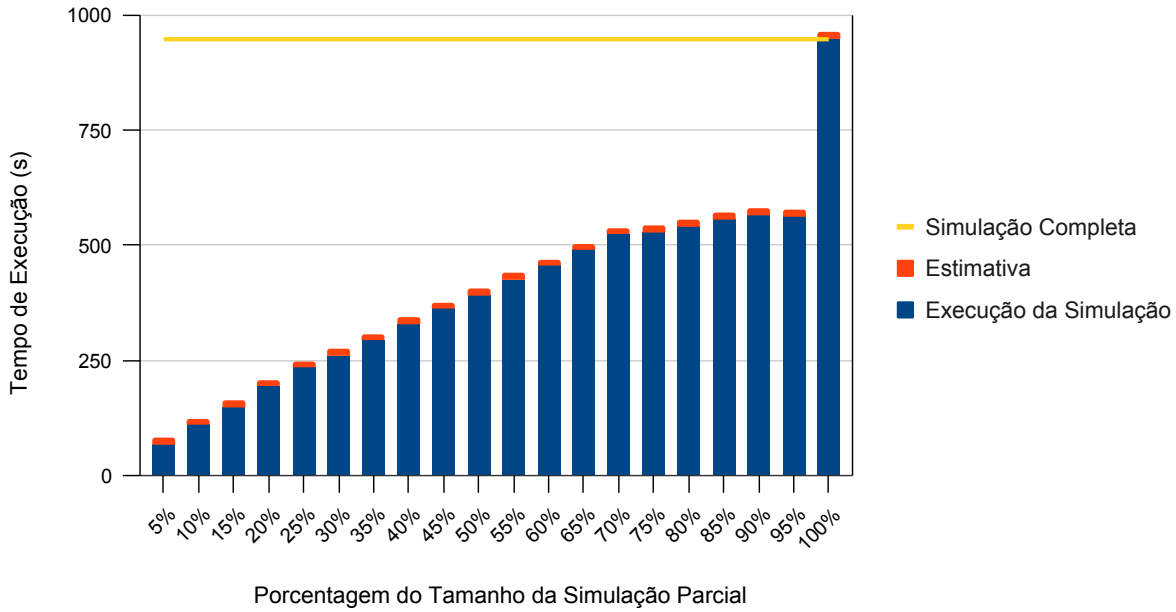
Os resultados experimentais foram obtidos usando um cenário com mais de 1,2 milhões de viagens em uma região de São Paulo com 7.072 *links*. Este cenário foi usado em nos experimentos para testar o desempenho das otimizações de desempenho de baixo nível e para gerar um conjunto de dados com 844 séries temporais de tamanho 2.756. As séries temporais foram distribuídas em 36 *clusters* para serem usados com a SimEDaPE. Para definir esse número de *clusters*, foram realizados experimentos empíricos. Foi inciado com 64 *clusters* e foi diminuindo até chegar ao número 36 que não deixava nenhum *cluster* vazio. A máquina usada para executar o experimento possui 64 GB de RAM e um processador Intel Core i7-7500U CPU 2,70 GHz octa-core.

A motivação para o desenvolvimento do SimEDaPE é permitir que especialistas em mobilidade urbana e em cidades inteligentes possam investigar novos cenários de mobilidade derivados de um cenário previamente simulado (por exemplo, de semáforos, criação de uma nova linha de ônibus ou metrô, construção de novas vias, entre outras coisas). Por isso, nos experimentos apresentados nesta seção, o novo cenário difere do cenário base por possuir 13 novos semáforos na região simulada.

Para medir o desempenho com os ganhos de tempo adquiridos ao aplicar a SimEDaPE foi computado quanto tempo é gasto nas principais etapas para estimar uma simulação parcial. As principais etapas são a Clusterização, o cálculo do *Warping Path*, Execução da Simulação Parcial e Estimativa. Na Figura 26 são mostrados os resultados dos tempos calculados. Com a redução do tamanho da amostra é possível ver os ganhos de tempo em relação ao tempo para executar uma simulação completa (tempo gasto de

948s). Mesmo somando o tempo da simulação parcial com o da execução das etapas da SimEDaPE, ainda é possível ter ganho. As amostras de cada uma das simulações parciais são valores múltiplos de 5, partindo de 5% até 95% do total das viagens da nova simulação completa. Já o tempo gasto para executar a estimativa para esse experimento foi cerca de 15s. O tempo da execução da clusterização foi de 1410s. O tempo para executar o cálculo do *Warping Path* é cerca de 13s. Um ponto importante é que as etapas de clusterização e cálculo do *Wapring Path* são executadas uma única vez para cada simulação base, por isso não estão presentes na imagem.

Figura 26 – Tempo gasto na execução das principais etapas da SimEDaPE.



Fonte: Francisco Wallison Carlos Rocha (2023).

Para validar a acurácia da SimEDaPE, foram realizados experimentos dos 6 algoritmos de geração de cenários parciais descritos na Seção 4.1.6. Para estimar as séries da nova simulação, as execuções desses algoritmos foram combinadas com as 4 formas de reverter a normalização aplicada descritas na Seção 4.1.7. Para cada combinação de algoritmo com uma forma de reverter a normalização, foram realizados 19 experimentos de simulações parciais.

Ao executar as simulações para testar suas abordagens e soluções os pesquisadores estão interessados em algumas métricas. Este trabalho visa mostrar a acurácia das métricas obtidas com a estimativa da SimEDaPE (aplicada nas simulações parciais) em comparação com a execução completa da nova simulação. As métricas presentes aqui nos experimentos

foram Velocidade Média, Porcentagem da Taxa de Ocupação das Vias e Média de Veículos. Todas as quatro métricas e como realizar seus cálculos estão descritas na Seção 4.1.7.

Para medir o erro E (em porcentagem) das métricas das simulações parciais estimadas pela SimEDaPE em relação as da simulação completa, foi usada a Equação 21. O erro E é definido como a média geométrica dos n valores x_i da métrica obtida de cada série temporal i da simulação completa menos a média geométrica dos m valores y_i da métrica obtida de cada série temporal estimada da simulação parcial. O valor da subtração das duas médias geométricas é dividido pelo valor da média geométrica dos valores da métrica obtida de cada série temporal da simulação completa. Tudo isso multiplicado por 100 para obter o erro em porcentagem. Os experimentos consistiram em executar as simulações parciais construídas usando os algoritmos de seleção de viagens. Para cada uma dessas simulações parciais foram usadas as 4 formas de aplicar a Equação 15 para reverter a normalização das séries temporais descritas na Seção 4.1.7.

$$E(x,y) = \frac{\sqrt[n]{\prod_{i=1}^{n} x_i} - \sqrt[m]{\prod_{i=1}^{m} y_i}}{\sqrt[n]{\prod_{i=1}^{n} x_i}} \cdot 100, |x| = n, |y| = m \tag{21}$$

Um detalhe importante em relação ao erro apresentado nas estimativas realizadas pela SimEDaPE, é o erro intrínseco na execução da simulação. Ao executar duas vezes uma simulação com a mesma entrada e comparar as duas execuções, é possível identificar um erro entre as duas simulações. Para a métrica da Velocidade Média o erro é de 0,09% e os erros da Porcentagem da Taxa de Ocupação das Vias e da Média de Veículos são de 0,01%.

Um outro erro, é o erro presente no uso dos dados da simulação base: *Warping Path*, médias e desvios padrão. Esses dados são usados para realizar a estimativa da nova simulação semelhante a base, mas contexto diferente. Por ser um contexto diferente, as séries temporais apresentarão algumas diferenças, consequentemente as médias e deviso padrão também. Na Tabela 7, são mostrados os erros para cada umas da métricas: Velocidade Média (M1), Porcentagem da Taxa de Ocupação da Vias (M2) e Média de Veículos (M3) estimadas com as formas de reverter a normalização aplicando a SimEDaPE em uma simulação completa e comparando com ela sem usar a SimEDaPE.

Além desses dos erros intrínsecos a simulação e aos dados usados para estimar, também é importante citar que a interpolação do *dataset* pode agravar ou amenizar os erros na estimativa. O tamanho das séries temporais do *dataset* da simulação base é de

Tabela 7 – Erro da estimativa da mesma simulação completa sendo estimada usando a SimEDaPE.

Dados da Normalização	**Métricas**		
	M1	**M2**	**M3**
Simulação Base	3,64%	11,92%	12,10%
Simulação Parcial	5,99%	0,49%	0,49%
Média da Simulação Base e Parcial	4,62%	5,55%	5,56%
Média Ponderada da Simulação Base e Parcial	4,62%	5,56%	5,56%

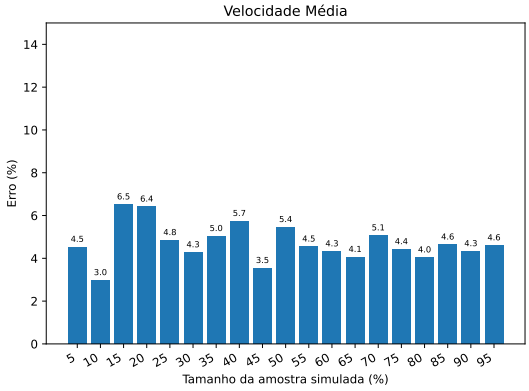
Fonte: Francisco Wallison Carlos Rocha (2023).

2.756. Em consequência disso, o tamanho do *Warping Path* também é de 2.756. As séries do *dataset* da simulação parcial devem ser interpoladas para o tamanho da maior série do *dataset* da simulação base para que possa ser aplicado o *Warping Path* nos novos representantes únicos da simulação parcial. Essa nova interpolação pode também agravar ou amenizar os erros da estimativa. A seguir são mostrados os resultados para cada um dos experimentos de estimativas das métricas. Um detalhe importe é necessário considerar que esses erros apresentados acima em combinação com as simulações parciais podem melhorar ou piorar as estimativas realizadas pela SimEDaPE.

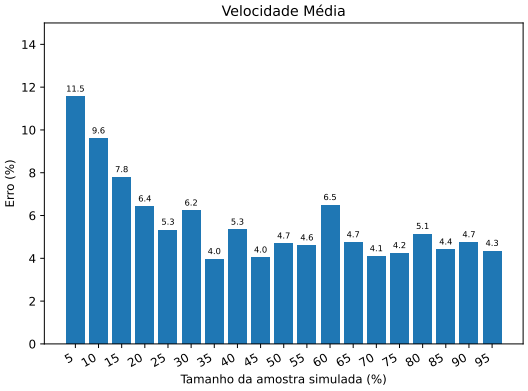
5.1.1 Velocidade Média

Nesta seção, são mostrados todos os resultados dos erros em porcentagem para métrica de Velocidade Média usando os seis algoritmos de seleção de viagens para geração de simulações parciais, combinados com as formas de reverter a normalização das séries temporais. A Figura 27 mostra os resultados dos erros das simulações parciais para cada um dos algoritmos combinados com a aplicação da Equação 15 somente com as médias e os desvios padrão da simulação base.

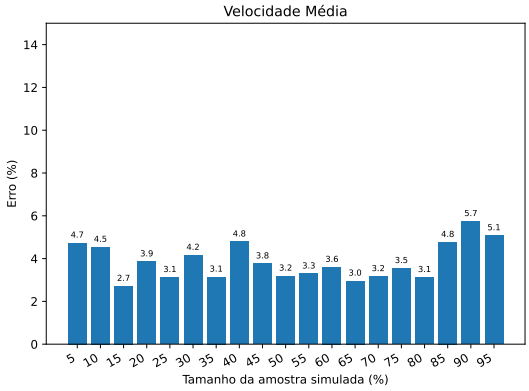
Figura 27 – Resultados para estimativas de velocidades médias das simulações parciais usando somente dados de normalização da simulação base.



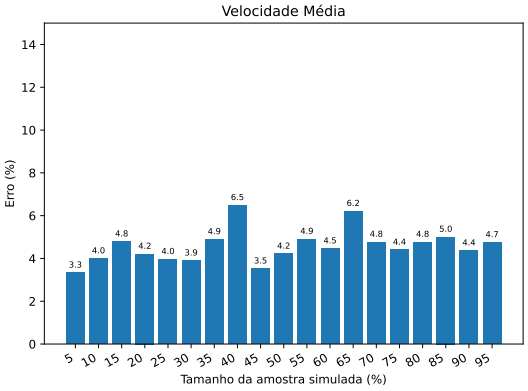
(a) Resultado do algoritmo Max



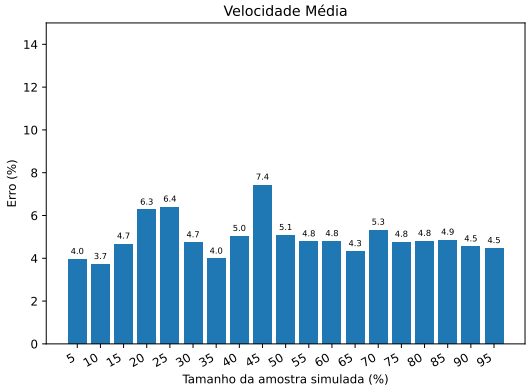
(b) Resultado do algoritmo Min



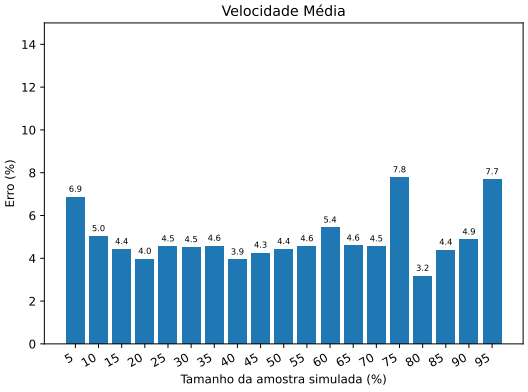
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

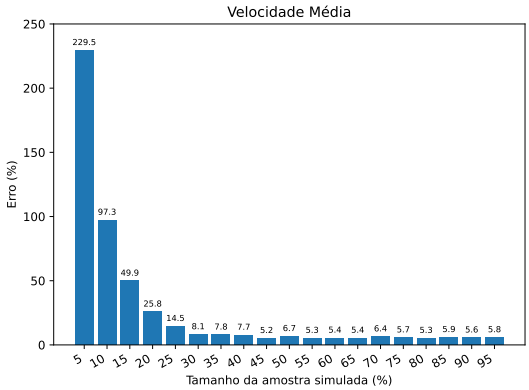
Fonte: Francisco Wallison Carlos Rocha (2023).

Para as estimativas mostradas na Figura 27 pode-se notar que os erros das simulações parciais não apresentam um padrão definido em todos os algoritmos. No entanto, todos

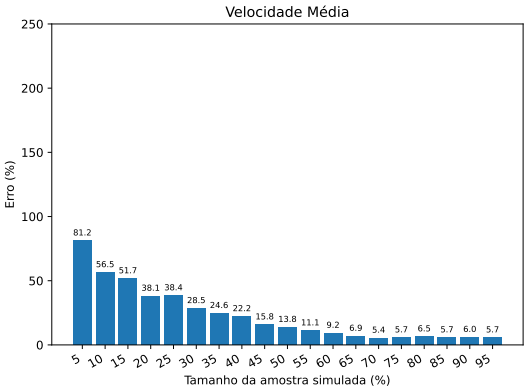
os erros ficaram abaixo dos 25%. No contexto desse experimento, com o uso dos dados da normalização da simulação base, o algoritmo que se saiu melhor foi o de Seleção de Viagens Aleatória.

Na Figura 28 são mostrados os resultados da métrica Velocidade Média para todos os algoritmos. Todas as estimativas desses resultados foram realizadas com uso dos dados de normalização somente da simulação parcial.

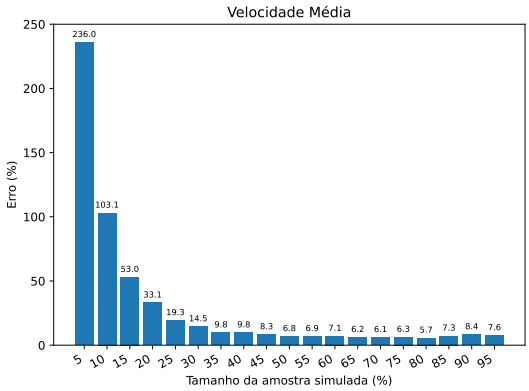
Figura 28 – Resultados para estimativas de velocidades médias das simulações parciais usando somente dados de normalização das simulações parciais.



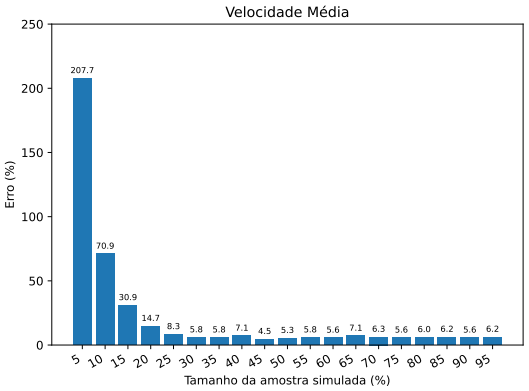
(a) Resultado do algoritmo Max



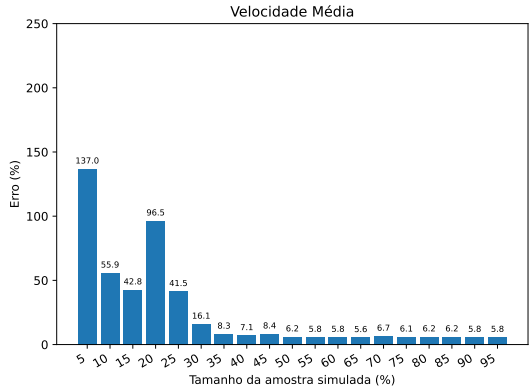
(b) Resultado do algoritmo Min



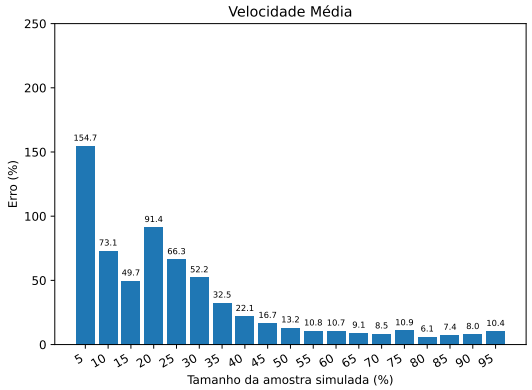
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



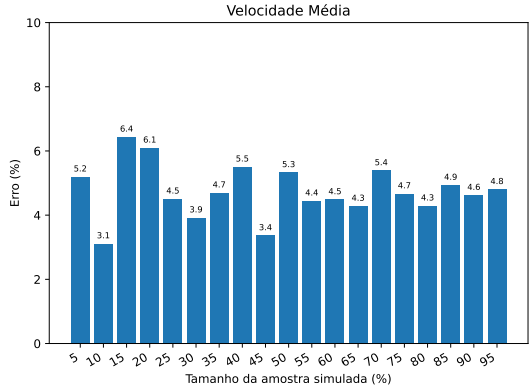
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

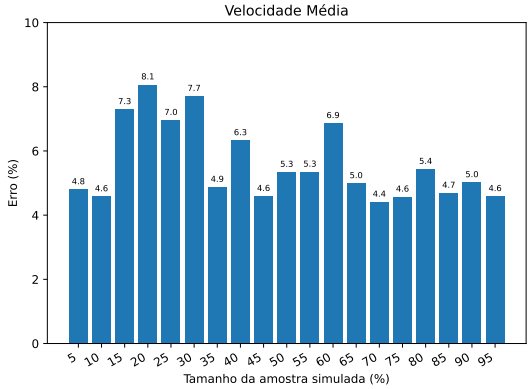
Na Figura 28, diferentemente da Figura 27, os erros apresentados para os algoritmos apresentam um padrão decrescente de acordo com o aumento da tamanho da amostra. Isso é por conta que a quantidade de séries temporais ou *simulation points* não gerados devido ao tamanho da amostra vai diminuindo a media que a amostra cresce. No geral, para esses experimentos, os algoritmos Max e Seleção Aleatória de Viagens Divididas por *Cluster* apresentaram um melhor desempenho que os demais.

A Figura 29 mostra os resultados para métrica de Velocidade Média usando a média dos dados de normalização da simulação base e simulação parcial.

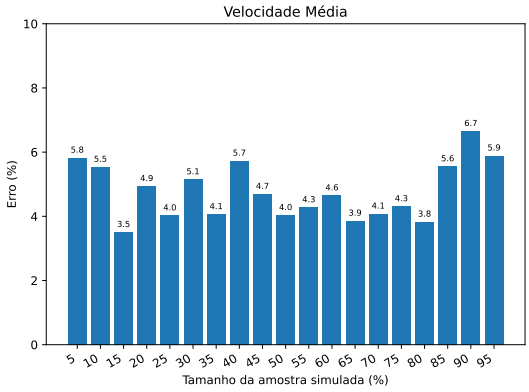
Figura 29 – Resultados para estimativas de velocidades médias das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais.



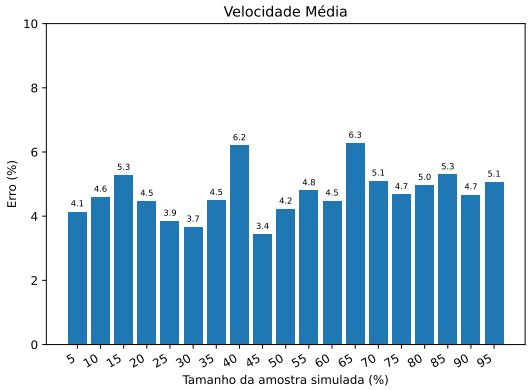
(a) Resultado do algoritmo Max



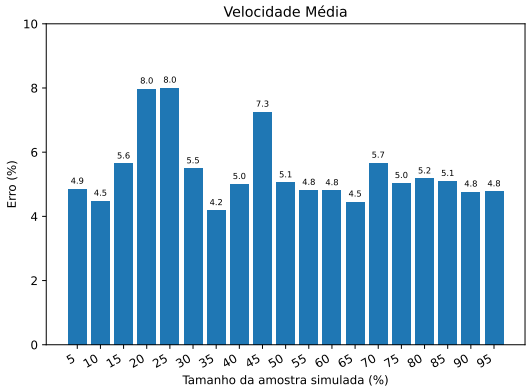
(b) Resultado do algoritmo Min



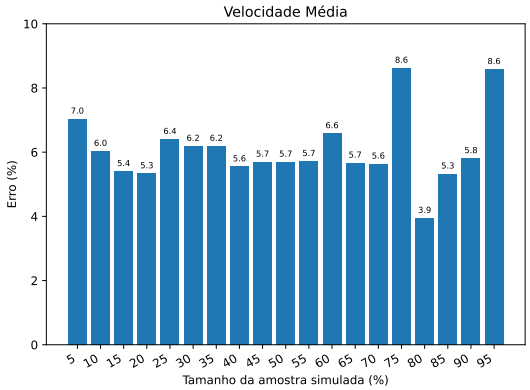
(a) Resultado do algoritmo de Seleção Aleatória de Viagens



(b) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(c) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



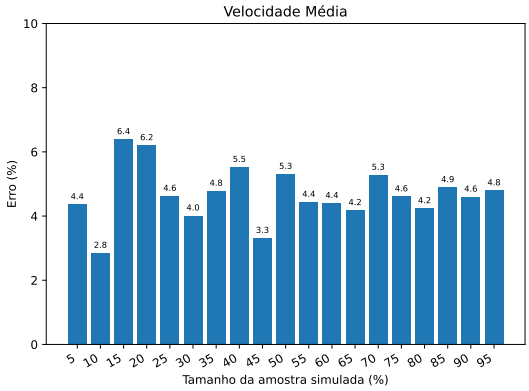
(d) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

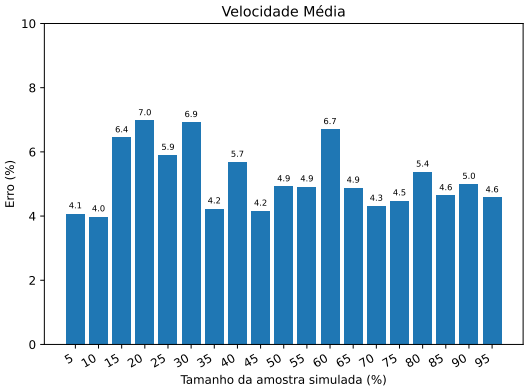
Ao usar os dados da simulação base com as da simulação parcial as estimativas apresentaram um comportamento diferente das apresentadas na Figura 28. Os resultados não apresentam um padrão, assim como os da Figura 27. Além disso, os resultados para todos os experimentos ficaram abaixo de 20%. O algoritmo Seleção Aleatória de Viagens Divididas por *Clusters* apresenta o melhor desempenho, com o maior erro 6,3%, porém, não há uma discrepância do valor para o maior valor dos demais algoritmos.

Na Figura 31 são mostrados os resultados da métrica Velocidade Média para todos os algoritmos. Todas as estimativas desses resultados foram realizadas com média ponderada dos dados de normalização da simulação base com os da simulação parcial. Os dados são aplicados usando a Equação 17.

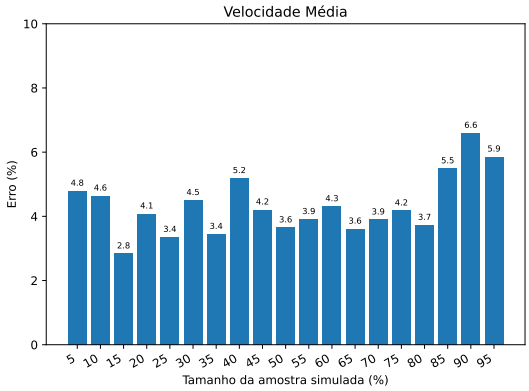
Figura 31 – Resultados para estimativas de velocidades médias das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais.



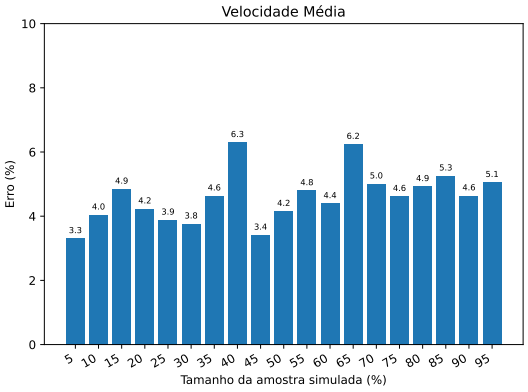
(a) Resultado do algoritmo Max



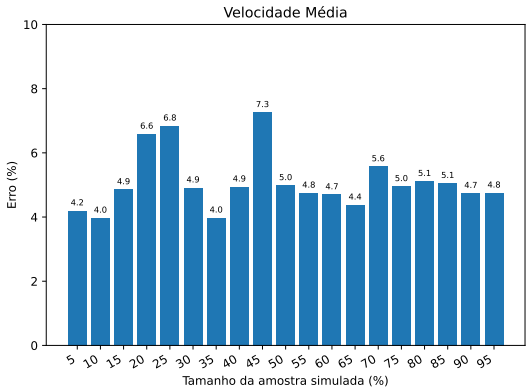
(b) Resultado do algoritmo Min



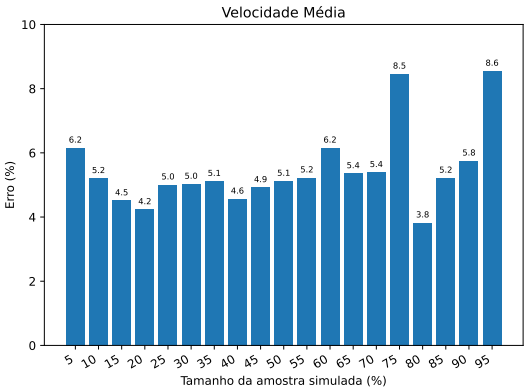
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



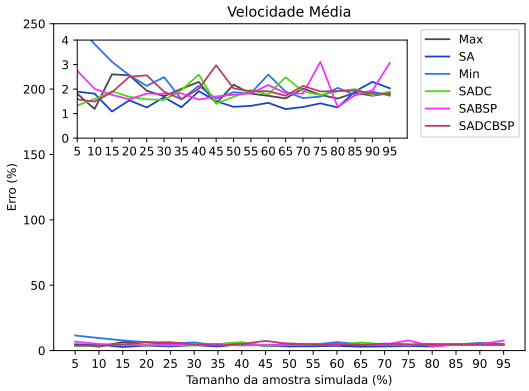
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

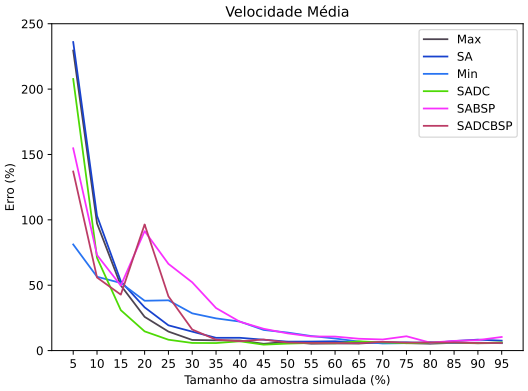
Assim como nos resultados mostrados nas Figuras 27 e 29, os erros para as estimativas das simulações parciais não apresentaram um padrão de comportamento. Além disso, todos os erros ficaram abaixo dos 10%. Com destaque ao algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster*, com o maior erro com valor de 6,3%.

Com a finalidade de facilitar a comparação do desempenho entre os algoritmos de seleção de viagens, a Figura 32 mostra o comparativo dos erros das estimativas da Velocidade Média feitas por cada algoritmo de seleção de viagens. Nela é possível ver o comportamento das estimativas ao usar um algoritmo de seleção de viagens com a seleção dos dados para reverter a normalização seguindo o uso da *z-score* inversa mostrada na Equação 15.

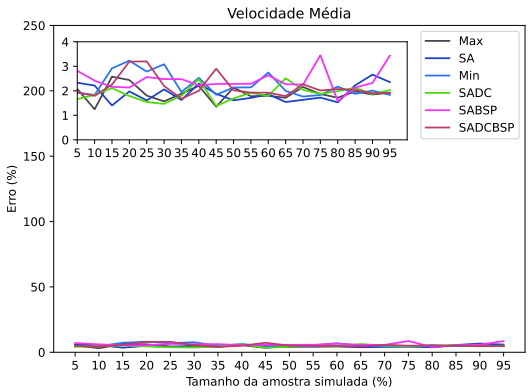
Figura 32 – Comparação das estimativas da métrica de Velocidade Média realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização.



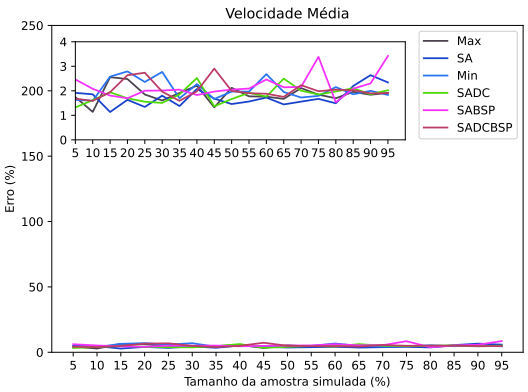
(a) Usando somente dados da simulação base



(b) Usando somente dados da simulação parcial



(c) Usando a média dos dados da simulação base e das parciais



(d) Usando a média ponderada dos dados da simulação base e das parciais

Fonte: Francisco Wallison Carlos Rocha (2023).

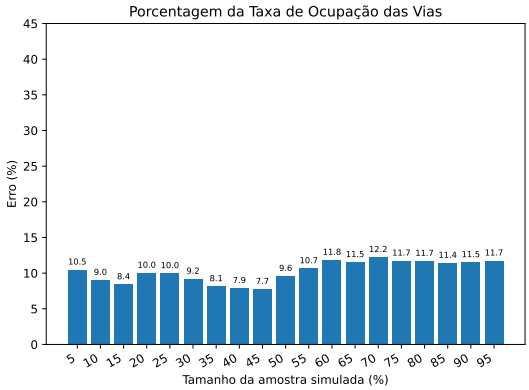
Na Figura 32c, é possível ver que as formas de estimativas que usam os dados da simulação base não apresentam um padrão de comportamento. Já a forma que usa somente os dados simulação parcial, apresenta um comportamento decrescente. No entanto, o uso dos dados da simulação base para estes experimentos executados apresenta um erro abaixo dos 10% para todos os algoritmos, embora não der uma ideia de qual seja o erro, mas o esperado é que seja um erro baixo. Já no uso somente da simulação parcial, amostras de menor tamanho apresentam erros maiores, porém, o formato da curva nos permite ter um erro mais controlado.

Esse comportamento ocorre porque o erro da estimativa usando somente os dados da simulação parcial acompanha o tamanho da amostra. Quanto maior o tamanho da amostra maior a quantidade de séries temporais e *simulation points* gerados e que podem ser estimados. Já no outro caso, os dados da simulação base compensam a falta desses elementos não gerados na simulação parcial, por isso apresentam erros pequenos da menor para maior amostra para esse experimento com essas simulações.

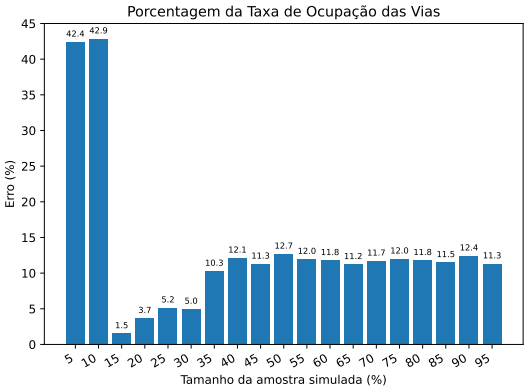
5.1.2 Porcentagem da Taxa de Ocupação das Vias

Nesta seção, são mostrados todos os resultados dos erros em porcentagem para métrica de Porcentagem da Taxa de Ocupação das Vias usando os seis algoritmos de seleção de viagens com as formas de reverter a normalização das séries temporais. A Figura 33 mostra os resultados dos erros das simulações parciais para cada um dos algoritmos combinados com a aplicação da Equação 15 somente com as médias e os desvios padrão da simulação base.

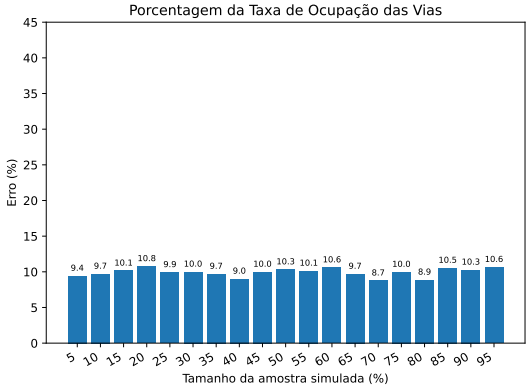
Figura 33 – Resultados para estimativas da Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando somente dados de normalização da simulação base.



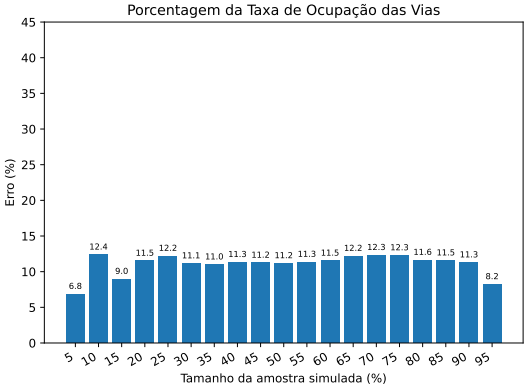
(a) Resultado do algoritmo Max



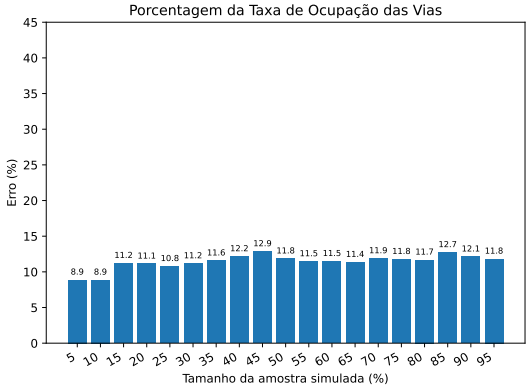
(b) Resultado do algoritmo Min



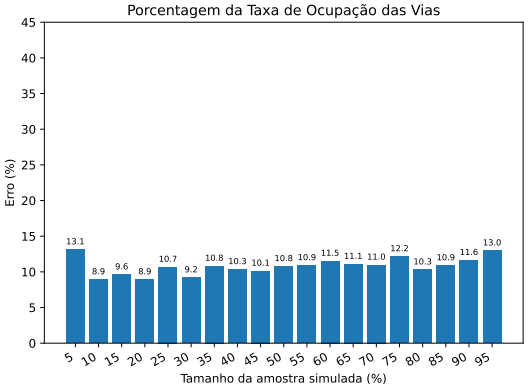
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



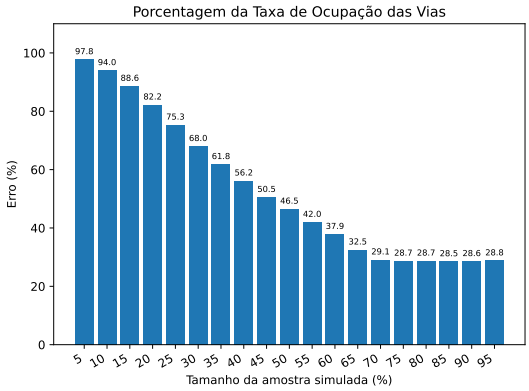
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

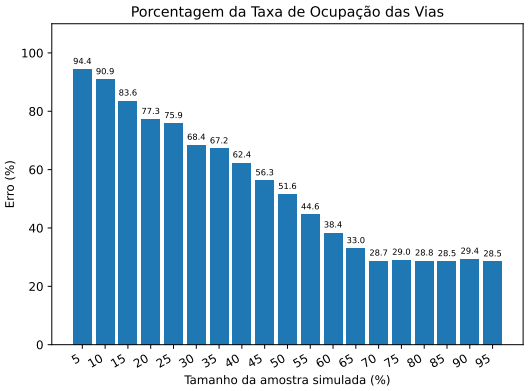
Os resultados das estimativas apresentados pela Figura 33 também não apresentem um padrão como alguns casos da Velocidade Média. Porém, a maioria dos erros para as estimativas estão abaixo dos 15%, exceto para as amostras do algoritmo Min. As amostras 5% e 10% mostram erros acima dos 40%. Isso pode ocorrer devido a não formação de vários *simulation points*, impedindo a estimativa de muitas séries temporais que tenham impactado direto na métrica. No geral, para esses experimentos, o algoritmo de Seleção Aleatória de Viagens demonstrou um melhor desempenho.

Na Figura 34 são mostrados os resultados da métrica Porcentagem da Taxa de Ocupação das Vias para todos os algoritmos. Todas as estimativas desses resultados foram realizadas com uso dos dados de normalização somente da simulação parcial.

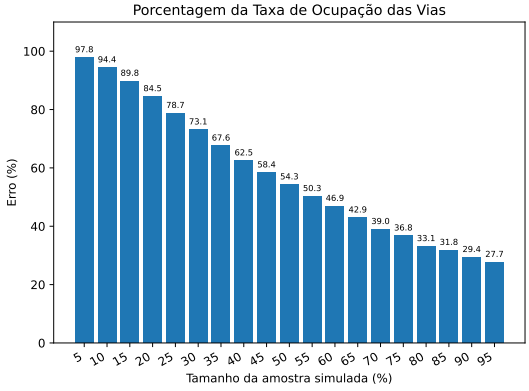
Figura 34 – Resultados para estimativas da Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando somente dados de normalização da simulação parcial.



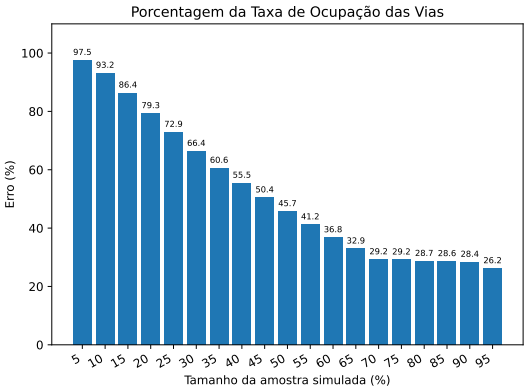
(a) Resultado do algoritmo Max



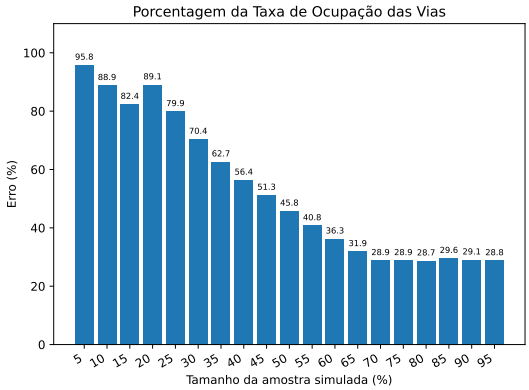
(b) Resultado do algoritmo Min



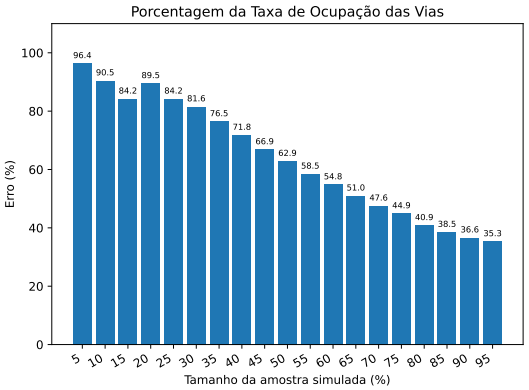
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



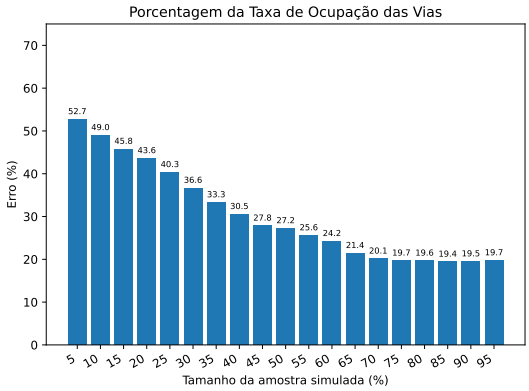
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

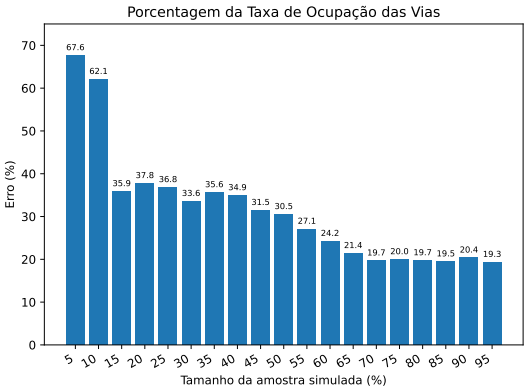
Assim como as estimativas mostradas na Figura 28, o comportamento das estimativa da Figura 34 usando os dados da simulações parciais na normalização apresentam um comportamento decrescente. No entanto, para esta métrica os erros são maiores.

A Figura 35 mostra os resultados para métrica de Porcentagem da Taxa de Ocupação das Vias usando a média dos dados de normalização da simulação base e simulação parcial. Os dados são aplicados usando a Equação 16.

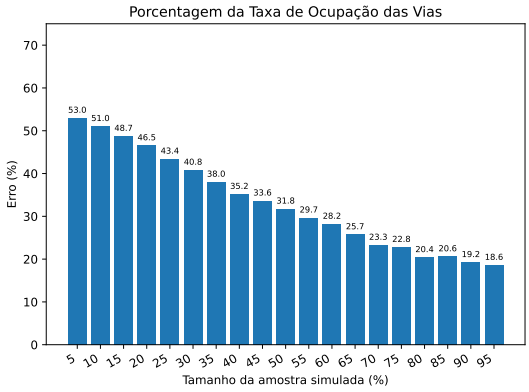
Figura 35 – Resultados para estimativas de Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais.



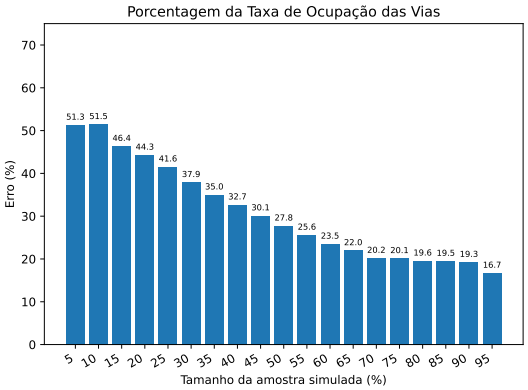
(a) Resultado do algoritmo Max



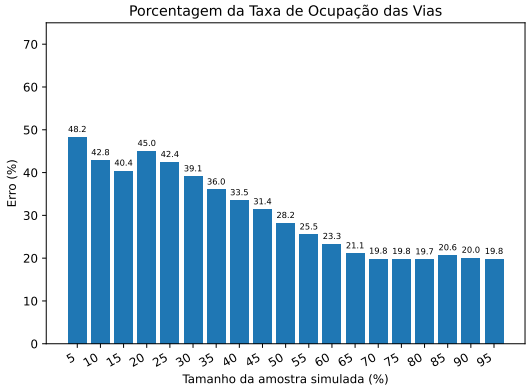
(b) Resultado do algoritmo Min



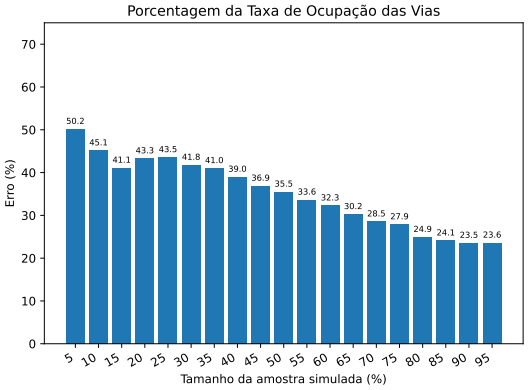
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



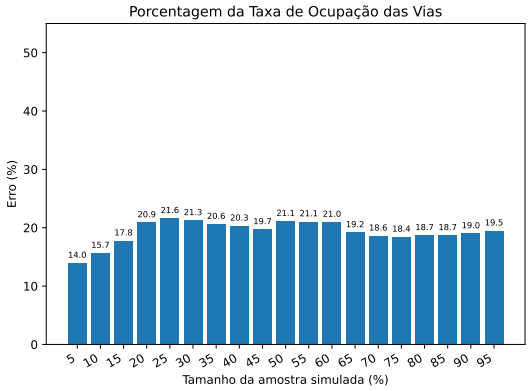
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

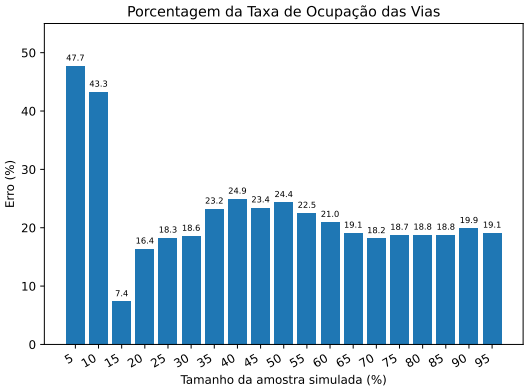
Para estimativa usando as médias dos dados das simulações parciais e base, as estimativas parciais apresentam um comportamento decrescente. A medida que se aumenta o tamanho da simulação, o valor do erro diminui como mostra a Figura 35.

Na Figura 36 são mostrados os resultados da métrica Porcentagem da Taxa de Ocupação das Vias para todos os algoritmos. Todas as estimativas desses resultados foram realizadas com média ponderada do dos dados de normalização da simulação base com os da simulação parcial. Os dados são aplicados usando a Equação 17.

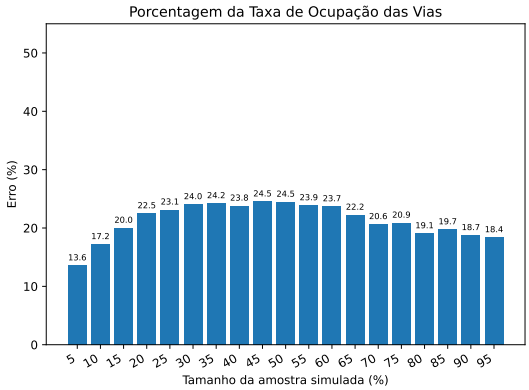
Figura 36 – Resultados para estimativas de Porcentagem da Taxa de Ocupação das Vias das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais.



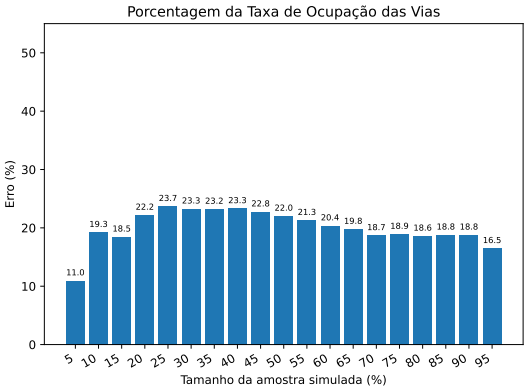
(a) Resultado do algoritmo Max



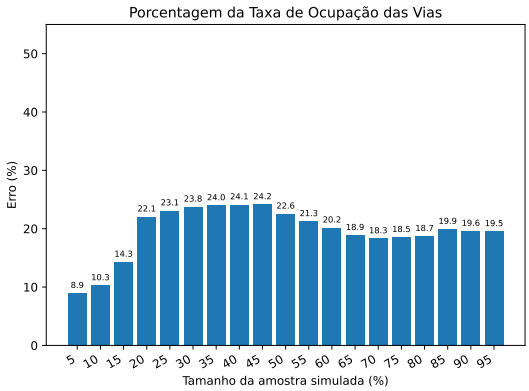
(b) Resultado do algoritmo Min



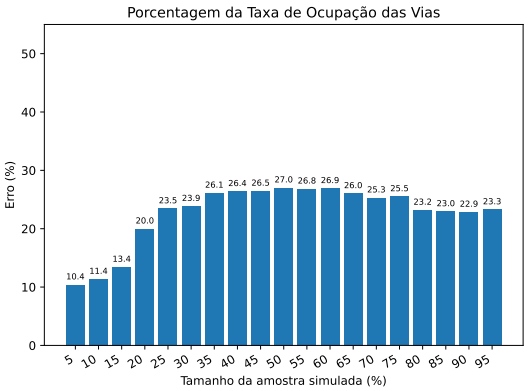
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



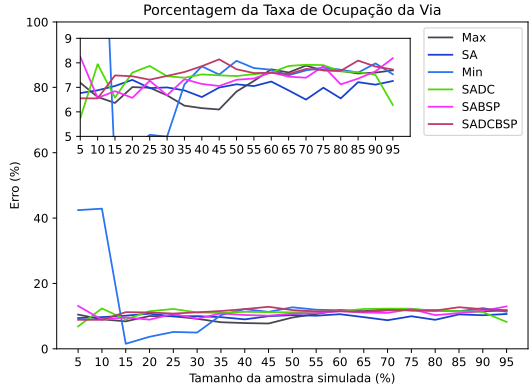
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

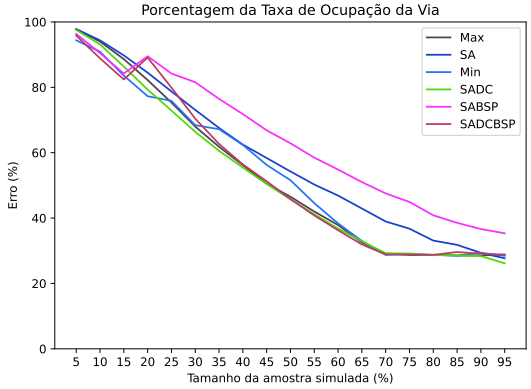
Na Figura 36, com a diminuição da proporção dos dados das simulações parciais e o integral dos dados da simulação base, as estimativas não apresentam um padrão de comportamento. Isto também ocorre com o uso somente dos dados da simulação base.

Na Figura 37 apresenta o comparativo dos erros das estimativas da Porcentagem da Taxa de Ocupação das Vias realizada por cada algoritmo de seleção de viagens.

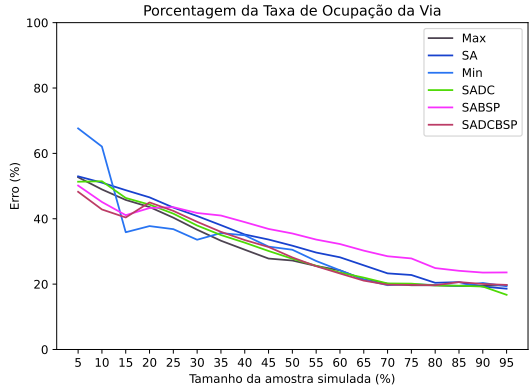
Figura 37 – Comparação das estimativas da métrica de Porcentagem da Taxa de Ocupação das Vias realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização.



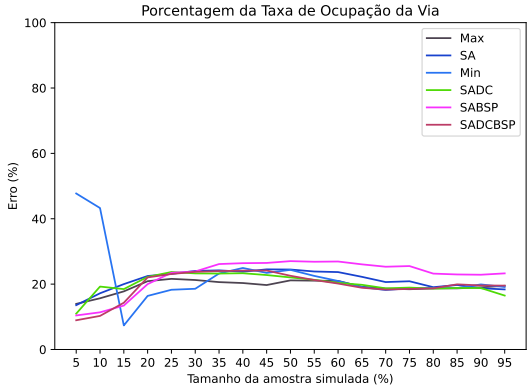
(a) Usando somente dados da simulação base



(b) Usando somente dados da simulação parcial



(c) Usando a média dos dados da simulação base e das parciais



(d) Usando a média ponderada dos dados da simulação base e das parciais

Fonte: Francisco Wallison Carlos Rocha (2023).

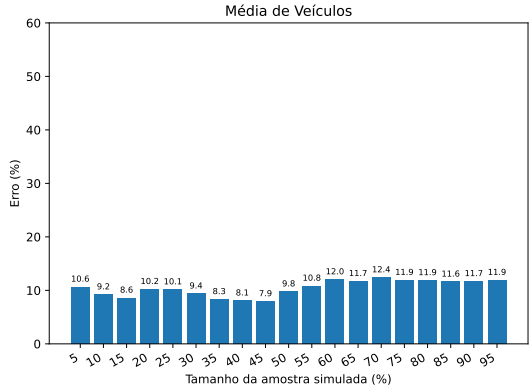
Os resultados da Figura 37 mostram que no uso das quatro formas de reverter as normalizações para esta métrica os algoritmos apresentam um comportamento semelhante. Um detalhe importante é que para as amostras menores o algoritmo Min apresenta um erro bem maior que os demais, devido a ausência de *simulation points* que representam grandes quantidades de séries temporais não foi possivel estimar essas séries temporais.quantidades

de séries temporais. Outro detalhe é que o uso dos dados da simulação base apresentam erros menores do que o uso somente da simulação parcial, assim como ocorre na estimativa da Velocidade Média.

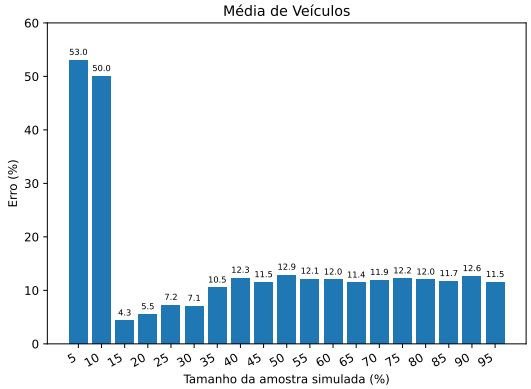
5.1.3 Média de Veículos

Nesta seção, são mostrados todos os resultados dos erros em porcentagem para métrica de Média de Veículos usando os seis algoritmos com as formas de reverter a normalização das séries temporais. A Figura 38 mostra os resultados dos erros das simulações parciais para cada um dos algoritmos combinados com a aplicação da Equação 15 somente com as médias e os desvios padrão da simulação base.

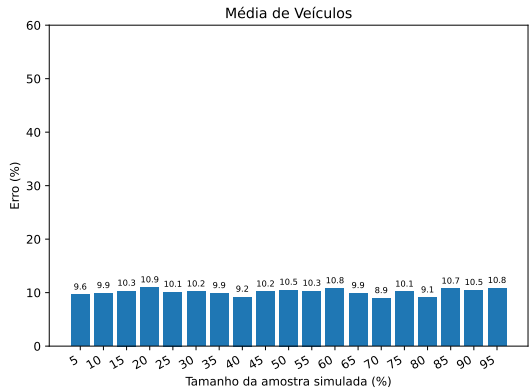
Figura 38 – Resultados para estimativas da Média de Veículos das simulações parciais usando somente dados de normalização da simulação base.



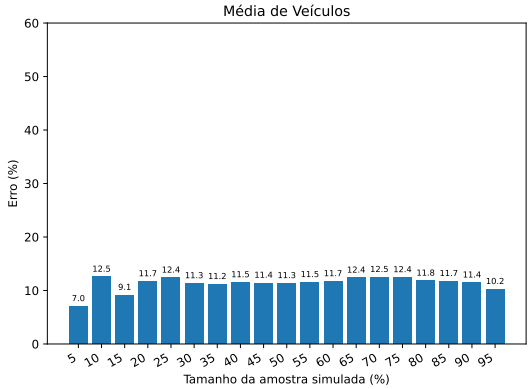
(a) Resultado do algoritmo Max



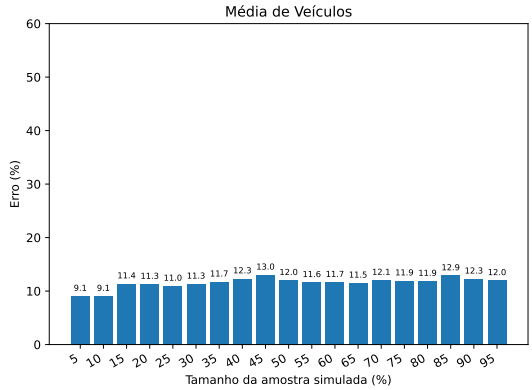
(b) Resultado do algoritmo Min



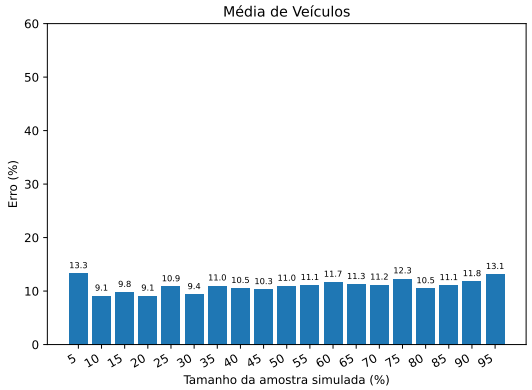
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*

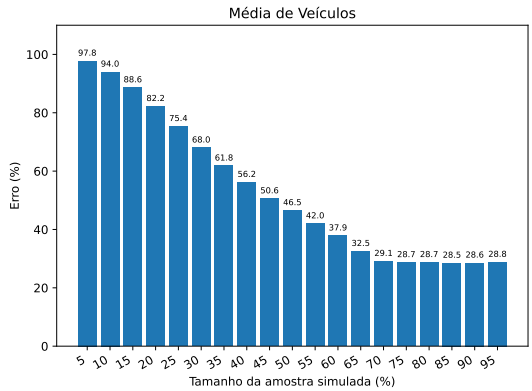


(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

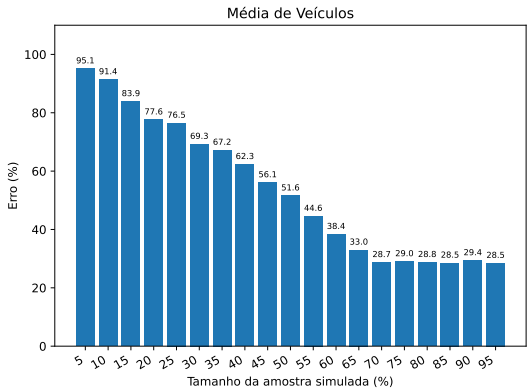
Fonte: Francisco Wallison Carlos Rocha (2023).

Na Figura 38, embora as estimativas não apresentem um padrão, elas apresentam um erro abaixo de 15%, exceto para dois casos no algoritmo Min. No algoritmo Min, as amostras de 5% e 10% demonstram um erro acima de 50%. Isto ocorre por conta que vários *simulation points* não foram gerados, impedindo uma melhor estimativa. Já na Figura 39 são mostrados os resultados para as estimativas usando somente dados da simulação parcial para reverter a normalização. No geral, o algoritmo com melhor desempenho foi o de Seleção Aleatória de Viagens se saiu melhor, porém, os demais estão bem próximos.

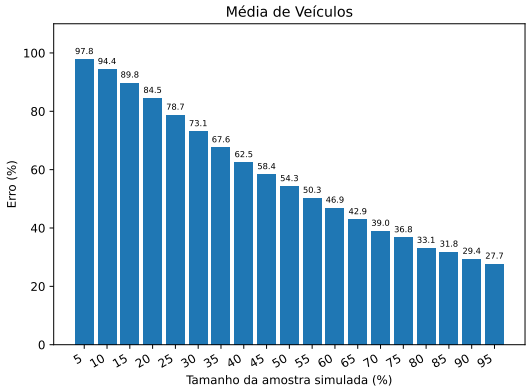
Figura 39 – Resultados para estimativas da Média de Veículos das simulações parciais usando somente dados de normalização da simulação parcial.



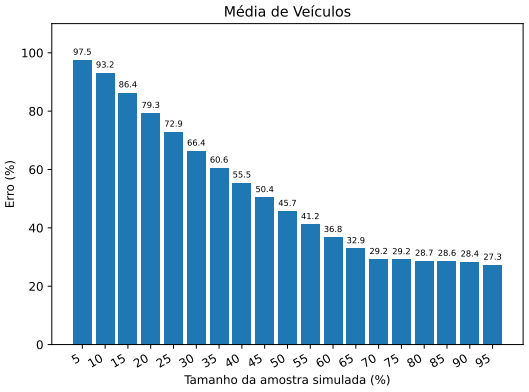
(a) Resultado do algoritmo Max



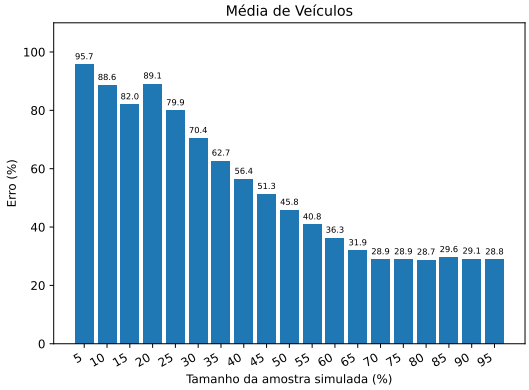
(b) Resultado do algoritmo Min



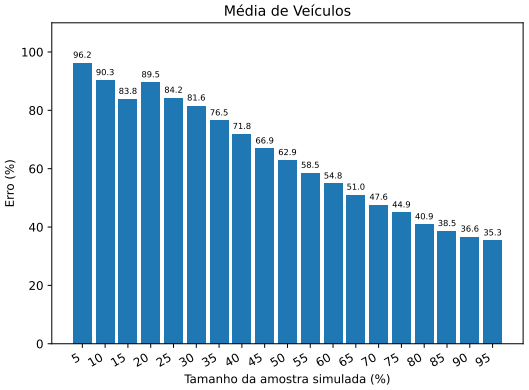
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



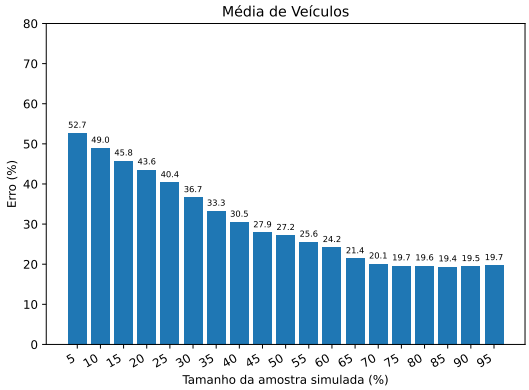
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

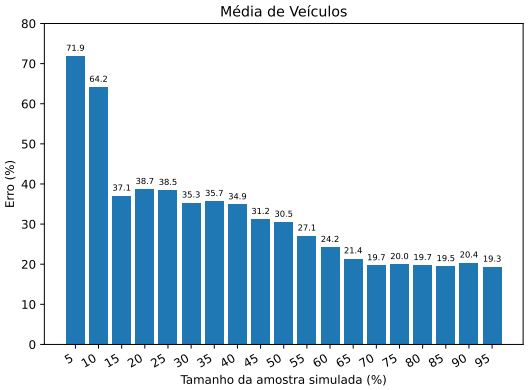
Como em outros resultados mostrados em outras figuras como a Figura 34. O uso somente dos dados da simulação parcial para reverter a normalização, na Figura 39 estimativas apresentam um comportamento decrescente. Porém, o piso do erro é 28% diferente de outras métricas ou uso de somente os dados da simulação base. Isto pode ocorrer devido a falta de *simulation points* e séries temporais geradas nas simulações parciais.

A Figura 40 mostra os resultados para a métrica de Média de Veículos usando a média dos dados de normalização da simulação base e simulação parcial. Os dados são aplicados usando a Equação 16.

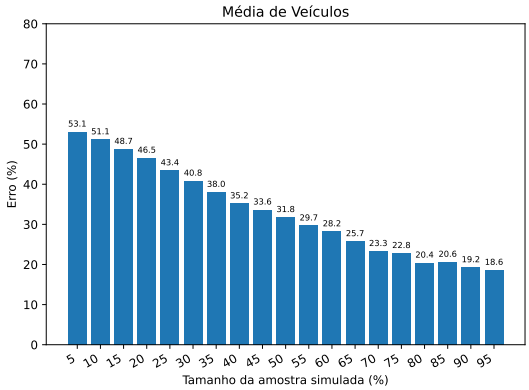
Figura 40 – Resultados para estimativas de Média de Veículos das simulações parciais usando média dados de normalização da simulação base com a os dados das simulações parciais.



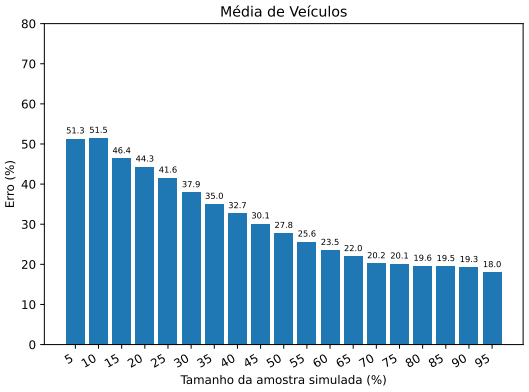
(a) Resultado do algoritmo Max



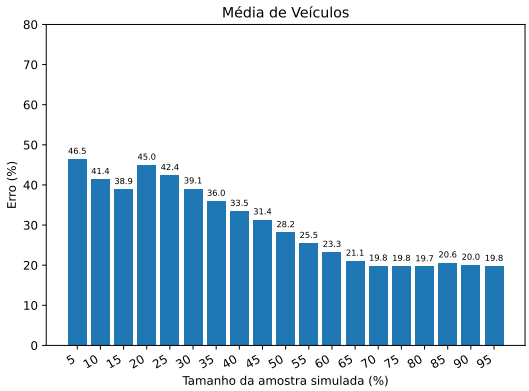
(b) Resultado do algoritmo Min



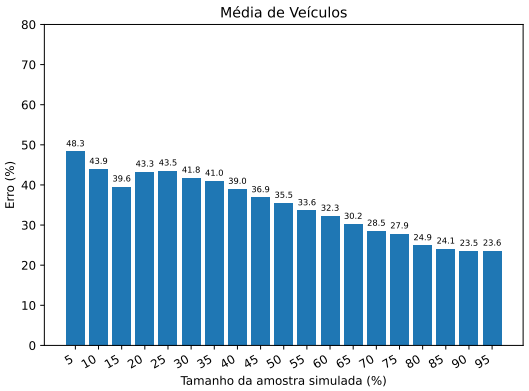
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



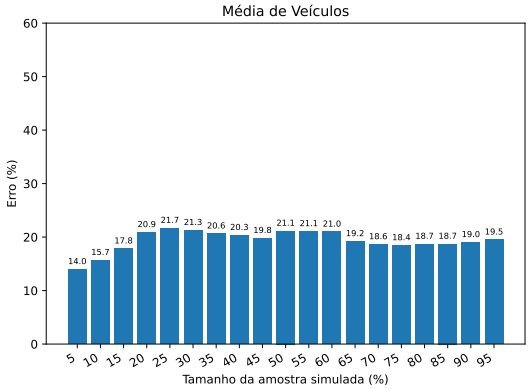
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

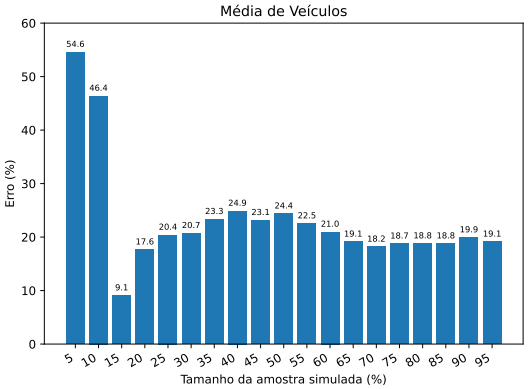
Assim como as estimativas da Figura 39, as estimativas da Figura 40 apresentam um comportamento de crescente a medida que o tamanho da simulação parcial aumenta. Outro detalhe é que o erro diminui ao usar a média dos dados das simulações parciais com a simulação base.

Na Figura 41 são mostrados os resultados da métrica Média de Veículos para todos os algoritmos. Todas as estimativas desses resultados foram realizadas com média ponderada do dos dados de normalização da simulação base com os da simulação parcial. Os dados são aplicados usando a Equação 17.

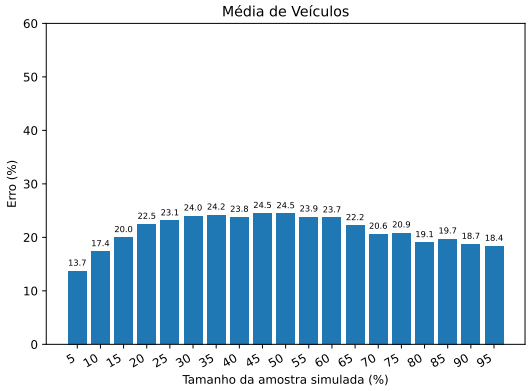
Figura 41 – Resultados para estimativas de Média de Veículos das simulações parciais usando média ponderada dados de normalização da simulação base com a os dados das simulações parciais.



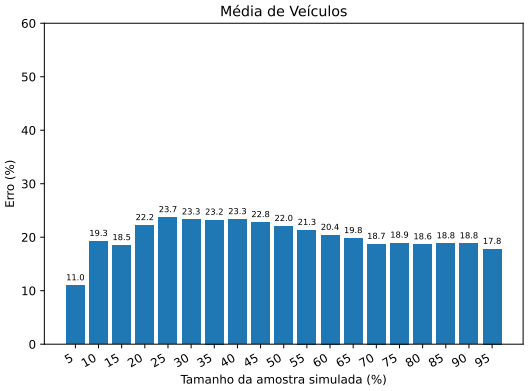
(a) Resultado do algoritmo Max



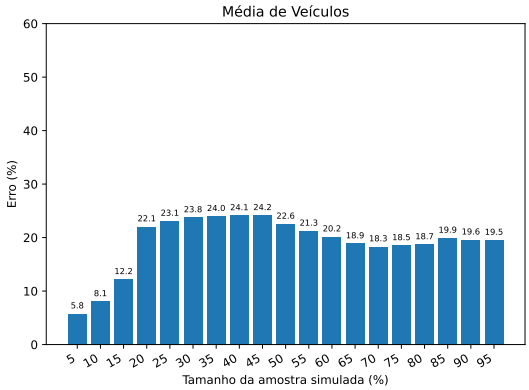
(b) Resultado do algoritmo Min



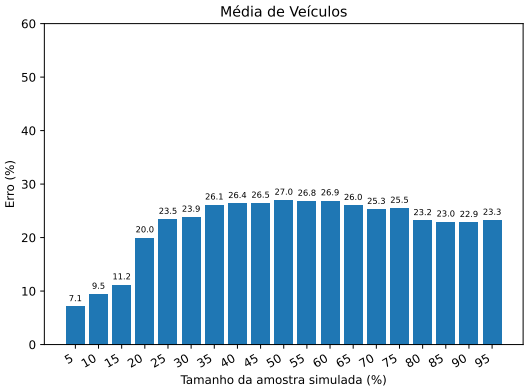
(c) Resultado do algoritmo de Seleção Aleatória de Viagens



(d) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Clusters*



(e) Resultado do algoritmo de Seleção Aleatória de Viagens Divididas por *Cluster* Com Base nos *Simulation Points*



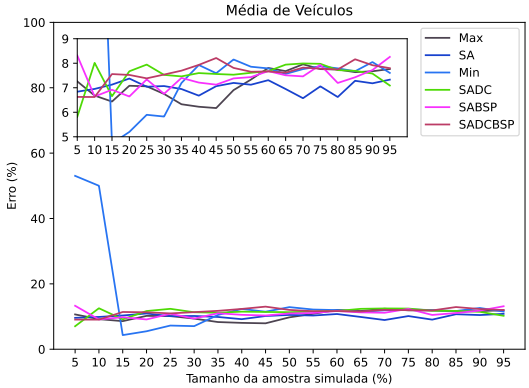
(f) Resultado do algoritmo de Seleção Aleatória de Viagens Com Base nos *Simulation Points*

Fonte: Francisco Wallison Carlos Rocha (2023).

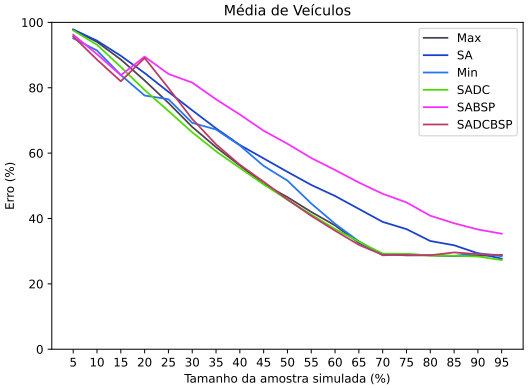
Assim como ocorre com outras métricas, não há um padrão nos erros observados quando a média ponderada é usada. A Figura 41 apresenta esse comportamento. Outra coisa é que ocorreu uma diminuição no erro ao dar mais importância aos dados da simulação base para reverter a normalização. O maior error está baixo de 40%. Como no caso do algoritmo Min, que tem o maior erro pelo motivo já comentado em outras estimativas. O motivo disto é a não geração de *simulation points* ou séries temporais para se realizar a estimativa.

O comparativo dos erros das estimativas da Porcentagem da Taxa de Ocupação das Vias realizada por cada algoritmo de seleção de viagens é mostrado na Figura 42.

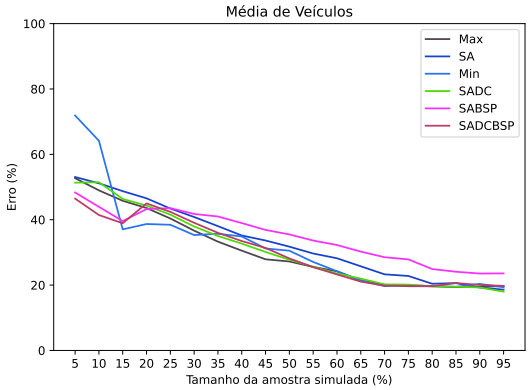
Figura 42 – Comparação das estimativas da métrica de Porcentagem da Taxa de Ocupação das Vias realizadas pelos algoritmos de seleção de viagens em combinação com as 4 formas de reverter a normalização.



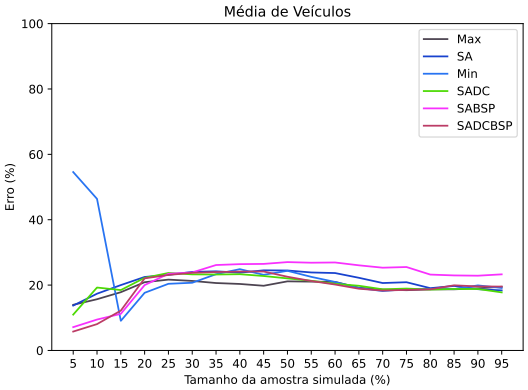
(a) Usando somente dados da simulação base



(b) Usando somente dados da simulação parcial



(c) Usando a média dos dados da simulação base e das parciais



(d) Usando a média ponderada dos dados da simulação base e das parciais

Fonte: Francisco Wallison Carlos Rocha (2023).

Os erros das estimativas mostrados na Figura 42c são semelhantes aos mostrados na Figura 37. Os algoritmos também apresentam comportamento parecido para as 4 formas de usar os dados para reverter a normalização. Os algoritmos usando os dados para reverter a normalização da simulação parcial apresentam um comportamento decrescente, porém, com erros maiores que os que usam os dados da simulação base.

5.2 Considerações Finais

Este capitulo apresentou os experimentos e os resultados obtidos para cada um deles. Os experimentos realizados foram relacionados a principal técnica deste trabalho, a SimEDaPE.

A SimEDaPE mostrou ganhos em relação ao tempo de execução ao diminuir o tamanho da simulação nova simulação. Além disso, nos resultados para medição da acurácia da técnica, a SimEDaPE mostrou erros abaixo dos 10% em algumas combinações de algoritmos com formas de reverter a normalização. Em alguns com a falta de alguns *simulation points* devido ao tamanho e quantidade de algumas simulações parciais, séries temporais não foram estimadas, o que faz com que o erro aumente.

Um outro fato que se nota nas estimativas é que em alguns casos simulações parciais com menor quantidade de viagens na amostra que outras apresentam um erro menor. Um exemplo são as estimativas da Média de Veículos (mostradas na Figura 38d) usando o Algoritmo Seleção Aleatória de Viagens Divididas Por *Cluster* de seleção de viagens usando os dados da simulação base. Nessa figura, pode-se ver que a estimativa para a

amostra de 5% apresenta uma melhor erro que todas as outras. Acredita que os erros intrínsecos a simulação, o uso de dados da simulação base para estimar e a interpolação das séries temporais em combinação com essa e outras simulações parciais possam está diretamente ligada a melhoria ou a piora da estimativa.

6 Considerações Finais

Esse trabalho propôs investigar a aplicação da técnica SimPoint no contexto de cidades inteligentes, com a finalidade de permitir que o *InterSCSimulator* (SANTANA *et al.*, 2018) execute grandes cenários em larga-escala, como o da cidade de São Paulo. A variação da técnica SimPoint no contexto de cidades inteligentes proposta neste trabalho, foi a SimEDaPE.

Na aplicação da técnica SimEDaPE, foi identificado que o que representa os comportamentos da simulação de carros no *InterSCSimulator* é o fluxo de veículos nas vias. Diante disso, foi definido o uso de séries temporais ao invés de *Basic Block Vectors* para representar os comportamentos da simulação. Para comparar e agrupar as séries temporais foi definido o algoritmo *K-shape* com a *Shape Based Distance* no lugar do *K-means* com a distância de *Manhattan*. O *K-shape* foi utilizado para agrupar as séries devido a sua acurácia e menor complexidade assimptótica em relação aos demais algoritmos usados para agrupar séries temporais. O método definido para dada uma entrada na simulação, o simulador execute somente os representantes únicos encontrados previamente, foram os seis algoritmos de seleção de viagem descritos na Seção 4.1.6. Esses algoritmos, apresentam abordagens para reduzir o tamanho da simulação e selecionar viagens que formam os representantes únicos. Por fim, este trabalho definiu um método para estimar os demais resultados de uma simulação parcial. O método definido foi através do uso do *Warping Path* em combinação com as formas de reverter a normalização aplicada pela *z-score*.

Para medir o desempenho da SimEDaPE, foram realizados experimentos para mostrar os ganhos de tempo e acurácia da técnica ao aplicar a estimativa da simulação. Para mostrar os ganhos de tempo foram computados os tempos para cada uma das etapas da SimEDaPE. Os resultados mostram que o uso da SimEDaPE permite diminuir consideravelmente o tempo de execução da simulação, executando apenas parte da simulação e estimando o restante.

O ganho de tempo não pode ser obtido em detrimento do aumento do erro da estimativa da simulação. Além de experimentos para medir os ganhos em relação ao tempo de execução, este trabalho também mostrou resultados para medir a acurácia da SimEDaPE para estimar métricas relacionadas ao contexto de tráfego urbano. Para a métrica Velocidade Média os erros foram baixos ao usar a SimEDaPE. Por outro lado

métricas como Porcentagem da Taxa de Ocupação das Vias que tem o comportamento semelhante a métrica Média de Veículos, mostram erros maiores para métricas que usam de dados da simulação parcial, mas erros pequenos para uso somente da simulação base para reverter a normalização.

Diante disso, o uso da SimEDaPE mostra-se eficaz para reduzir o tempo de execução da simulação e obter uma estimativa próxima da execução da simulação completa em alguns casos.

6.1 Trabalhos Futuros

Existem várias novas oportunidades para trabalhos futuros relacionados a este trabalho. Um exemplo seria investigar o uso de *multithreading* para acelerar algumas etapas da SimEDaPE que para grande volumes possam ser um gargalo. Atualmente a etapa do Cálculo do *Warping Path* usa de *multithreading*. No entanto, outras etapas que demando um grande tempo a depender do tamanho da simulação e *dataset* podem ser o ponto de partida para investigação, são elas: Processamento do *Dataset* e Clusterização.

Além de melhorias na implementação das etapas sugeridas acima, existe a oportunidade de avaliar a SimEDaPE com outras métricas de interesses dos pesquisadores e profissionais.

Atualmente a SimeDaPE usa o *Warping Path* para estimar as demais séries temporais faltantes por ser uma simulação parcial. Investigar outras abordagens alternativas ao uso *Warping Path*, pode ser uma um alternativa para melhorar o desempenho da estimava da SimEDaPE. Uma oportunidade é avaliar o uso de Análise de Componentes Principais (ACP), que é usada no algoritmo de clusterização o *K-shape* no cálculo da *Shape Based Distance*.

Até o presente momento, a SimEDaPE foi aplicada apenas no contexto de cidades inteligentes e no simulador do InterSCSimulator. Como trabalho futuro, existe a possibilidade de aplicar a SimEDaPE em outros simuladores de cidades inteligentes como o SUMO (KRAJZEWICZ *et al.*, 2002) e o MATSim (HORNI; NAGEL; AXHAUSEN, 2016). Além disso, existe a possibilidade de usar em outros contextos de simulações que não sejam relacionadas ao tráfego urbano. A SimEDaPE pode ser aplicada em qualquer

simulação que o seu resultado que tenha seu principal componente no formato ou possível de transformar em séries temporais.

Neste trabalho, a aplicação da SimEDaPE consistiu apenas no modo de viagem carro. Em trabalhos futuros tem-se a intenção de aplicar técnicas em outro modos como bicicleta, caminhando, ônibus e metrô, individualmente em cada um deles ou em combinações.

Por fim, atualmente a SimEDaPE tem a possibilidade de usar 6 algoritmos diferentes para selecionar as viagens que serão executadas na simulação parcial. Para melhor selecionar as viagens e formar melhor os *simulation points*, uma abordagem semelhante ao trabalho de Hanai *et al.* (2019) para selecionar as viagens a partir dos eventos que compõe os *simulation points*. No trabalho de Hanai *et al.* (2019), os eventos gerados na simulação que sofreriam modificações em um novo cenário dessa simulação, são identificados para que somente eles sejam executados na nova simulação e os demais eventos sejam reciclados. Identificar todos os eventos responsáveis por formar as viagens que formam os *simulation points* pode ser uma alternativa aos algoritmos de seleção de viagens apresentados neste trabalho.

6.2 Publicações Resultantes Deste Trabalho

Os resultados científicos obtidos durante este mestrado foram divulgados nos seguintes veículos:

ROCHA, F.; FRANCESQUINI, E.; CORDEIRO, D. *Uso de particionamento de dados para acelerar simulações de cidades inteligentes*. In: 11ª Escola Regional de Alto Desempenho de São Paulo. Porto Alegre, RS, Brasil: SBC, 2021. Disponível em: ⟨https://doi.org/10.5753/eradsp.2020.16887⟩.

ROCHA, F.; FRANCESQUINI, E.; CORDEIRO, D. *Uma abordagem inspirada em simulation points para acelerar as simulações de cidades inteligentes*. In: Anais da XII Escola Regional de Alto Desempenho de São Paulo. Porto Alegre, RS, Brasil: SBC, 2021. p. 49–52. Disponível em: ⟨https://doi.org/10.5753/eradsp.2021.16703⟩.

ROCHA, F.; FRANCESQUINI, E.; CORDEIRO, D. *Fast SimEDaPE: Simulation estimation by data patterns exploration*. In: Anais da XIII Escola Regional de Alto Desempenho de São Paulo. Porto Alegre, RS, Brasil: SBC, 2022. p. 37–40. Disponível em: ⟨https://doi.org/10.5753/eradsp.2022.222246⟩.

ROCHA, F.; FRANCESQUINI, E.; CORDEIRO, D. *Improving smart city simulation performance with SimEDaPE and parallelism*. In: Anais do XXI Workshop em Desempenho de Sistemas Computacionais e de Comunicação. Porto Alegre, RS, Brasil: SBC, 2022. p. 108–113. ISSN 2595-6167. Disponível em: ⟨https://doi.org/10.5753/wperformance.2022.223235⟩.

ROCHA, F. W.; FUKUDA, J. C.; FRANCESQUINI, E.; CORDEIRO, D. *Accelerating smart city simulations*. In: Latin American High Performance Computing Conference. 2022. p. 148–162. Springer. Disponível em: ⟨https://doi.org/10.1007/978-3-031-04209-6_11⟩.

6.3 Financiamento

Esta pesquisa foi parte do *INCT da Internet do Futuro para Cidades Inteligentes*, financiado por CNPq (proc. 465446/2014-0), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e FAPESP (procs. 14/50937-1 e 15/24485-9) e do projeto *Aplicações de teoria do escalonamento para otimizar o uso de energia verde em plataformas de nuvens computacionais*, processo nº 2021/06867-2, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências[1]

ANDELFINGER, P.; CHEN, Y.; SU, B.; CAI, W.; ZEHE, D.; ECKHOFF, D.; KNOLL, A. Incremental calibration of seat selection preferences in agent-based simulations of public transport scenarios. In: IEEE. *2018 Winter Simulation Conference (WSC)*. Gothenburg, Sweden, Sweden, 2018. p. 833–844. Citado 2 vezes nas páginas 48 e 50.

ARMSTRONG, J. A history of erlang. In: *Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages*. New York, NY, USA: Association for Computing Machinery, 2007. (HOPL III), p. 6–1–6–26. ISBN 9781595937667. Disponível em: ⟨https://doi.org/10.1145/1238844.1238850⟩. Citado na página 22.

ARMSTRONG, J.; VIRDING, R.; WIKSTRÖM, C.; WILLIAMS, M. Concurrent programming in erlang. Citeseer, 1993. Citado na página 23.

BARCELÓ, J. *et al. Fundamentals of Traffic Simulation*. [S.l.]: Springer, 2010. v. 145. (International Series in Operations Research and Management Science, 978-1-4419-6142-6). Citado na página 50.

BERNDT, D. J.; CLIFFORD, J. Using dynamic time warping to find patterns in time series. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. Seattle, WA: AAAI Press, 1994. (AAAIWS'94), p. 359–370. Citado 2 vezes nas páginas 35 e 65.

BIRDSEY, L.; SZABO, C.; FALKNER, K. Casl: a declarative domain specific language for modeling complex adaptive systems. In: IEEE PRESS. *Proceedings of the 2016 Winter Simulation Conference*. Washington, DC, USA, 2016. p. 1241–1252. Citado 2 vezes nas páginas 48 e 54.

BORETTI, A.; ROSA, L. Reassessing the projections of the world water development report. *NPJ Clean Water*, Nature Publishing Group, v. 2, n. 1, p. 1–6, 2019. Citado na página 17.

BRACEWELL, R. N.; BRACEWELL, R. N. *The Fourier transform and its applications*. [S.l.]: McGraw-Hill New York, 1986. v. 31999. Citado na página 41.

BRAMBILLA, G.; PICONE, M.; CIRANI, S.; AMORETTI, M.; ZANICHELLI, F. A simulation platform for large-scale internet of things scenarios in urban environments. In: *Proceedings of the First International Conference on IoT in Urban Space*. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014. (URB-IOT '14), p. 50–55. ISBN 9781631900372. Disponível em: ⟨https://doi.org/10.4108/icst.urb-iot.2014.257268⟩. Citado 2 vezes nas páginas 48 e 53.

BRITO, A. V.; COSTA, L. F. S.; BUCHER, H.; SANDER, O.; BECKER, J.; OLIVEIRA, H.; MELCHER, E. U. A distributed simulation platform using hla for complex embedded systems design. In: IEEE PRESS. *Proceedings of the 19th International Symposium on Distributed Simulation and Real Time Applications*. Chengdu, China, 2015. p. 195–202. Citado 2 vezes nas páginas 48 e 54.

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

CASSISI, C.; MONTALTO, P.; ALIOTTA, M.; CANNATA, A.; PULVIRENTI, A. Similarity measures and dimensionality reduction techniques for time series data mining. *Advances in data mining knowledge discovery and applications*, p. 71–96, 2012. Citado na página 36.

CHAMOSO, P.; GONZÁLEZ-BRIONES, A.; RODRÍGUEZ, S.; CORCHADO, J. M. Tendencies of technologies and platforms in smart cities: a state-of-the-art review. *Wireless Communications and Mobile Computing*, Hindawi, v. 2018, 2018. Citado na página 42.

D'ANGELO, G.; FERRETTI, S.; GHINI, V. Modeling the internet of things: a simulation perspective. In: IEEE. *2017 International Conference on High Performance Computing & Simulation (HPCS)*. Genoa, Italy, 2017. p. 18–27. Citado 2 vezes nas páginas 48 e 53.

ELBERY, A.; BICHIOU, Y.; RAKHA, H. A.; DU, J.; DVORAK, F.; KLENK, M. City-level agent-based multi-modal modeling of transportation networks: Model development and preliminary testing. In: DONNELLAN, B.; KLEIN, C.; HELFERT, M.; GUSIKHIN, O. (Ed.). *Smart Cities, Green Technologies and Intelligent Transport Systems*. Cham: Springer International Publishing, 2019. p. 279–303. ISBN 978-3-030-26633-2. Citado 4 vezes nas páginas 48, 51, 55 e 56.

FRANCISCO, S.; PINHO, P.; LUíS, M. Improving lora network simulator for a more realistic approach on lorawan. In: *2021 Telecoms Conference (ConfTELE)*. [S.l.: s.n.], 2021. p. 1–6. Citado 2 vezes nas páginas 49 e 53.

FU, Z.; YU, J.; SARWAT, M. Demonstrating geosparksim: A scalable microscopic road network traffic simulator based on apache spark. In: *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*. New York, NY, USA: Association for Computing Machinery, 2019. (SSTD '19), p. 186–189. ISBN 9781450362801. Disponível em: ⟨https://doi.org/10.1145/3340964.3340984⟩. Citado 2 vezes nas páginas 48 e 50.

GARZON, S. R.; DEVA, B.; HANOTTE, B.; KüPPER, A. Catles: A crowdsensing-supported interactive world-scale environment simulator for context-aware systems. In: *Proceedings of the International Conference on Mobile Software Engineering and Systems*. New York, NY, USA: Association for Computing Machinery, 2016. (MOBILESoft '16), p. 77–87. ISBN 9781450341783. Disponível em: ⟨https://doi.org/10.1145/2897073.2897078⟩. Citado 2 vezes nas páginas 48 e 54.

GÜTLEIN, M.; GERMAN, R.; DJANATLIEV, A. Towards a hybrid co-simulation framework: Hla-based coupling of matsim and sumo. In: IEEE. *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. Madrid, Spain, 2018. p. 1–9. Citado 2 vezes nas páginas 48 e 51.

HAMERLY, G.; PERELMAN, E.; LAU, J.; CALDER, B. Simpoint 3.0: Faster and more flexible program phase analysis. *Journal of Instruction Level Parallelism*, v. 7, n. 4, p. 1–28, 2005. Citado 6 vezes nas páginas 19, 20, 31, 32, 35 e 58.

HANAI, M.; SUZUMURA, T.; LIU, E. S.; THEODOROPOULOS, G.; PERUMALLA, K. S. Exact-differential simulation: Differential processing of large-scale discrete event simulations. *ACM Trans. Model. Comput. Simul.*, Association for Computing Machinery, New York, NY, USA, v. 29, n. 3, jun 2019. ISSN 1049-3301. Disponível em: ⟨https://doi.org/10.1145/3301499⟩. Citado 4 vezes nas páginas 49, 51, 56 e 106.

HE, Z.; YOU, L.; LIU, R. W.; YANG, F.; MA, J.; XIONG, N. A cloud-based real time polluted gas spread simulation approach on virtual reality networking. *IEEE Access*, IEEE, v. 7, p. 22532–22540, 2019. Citado 2 vezes nas páginas 48 e 52.

HORNI, A.; NAGEL, K.; AXHAUSEN, K. (Ed.). *Multi-Agent Transport Simulation MATSim*. London: Ubiquity Press, 2016. 618 p. ISBN 978-1-909188-75-4, 978-1-909188-76-1, 978-1-909188-77-8, 978-1-909188-78-5. Citado na página 105.

IANNI, M.; MAROTTA, R.; CINGOLANI, D.; PELLEGRINI, A.; QUAGLIA, F. Optimizing simulation on shared-memory platforms: the smart cities case. In: IEEE. *2018 Winter Simulation Conference (WSC)*. Gothenburg, Sweden, Sweden, 2018. p. 1969–1980. Citado 2 vezes nas páginas 48 e 54.

ICHIMURA, T.; FUJITA, K.; QUINAY, P. E. B.; MADDEGEDARA, L.; HORI, M.; TANAKA, S.; SHIZAWA, Y.; KOBAYASHI, H.; MINAMI, K. Implicit nonlinear wave simulation with 1.08 t dof and 0.270 t unstructured finite elements to enhance comprehensive earthquake simulation. In: IEEE. *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Austin, TX, USA, 2015. p. 1–12. Citado 2 vezes nas páginas 48 e 52.

ICHIMURA, T.; FUJITA, K.; TANAKA, S.; HORI, M.; LALITH, M.; SHIZAWA, Y.; KOBAYASHI, H. Physics-based urban earthquake simulation enhanced by 10.7 blndof× 30 k time-step unstructured fe non-linear seismic wave simulation. In: IEEE. *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. New Orleans, LA, USA, 2014. p. 15–26. Citado 2 vezes nas páginas 48 e 52.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, v. 31, n. 8, p. 651 – 666, 2010. ISSN 0167-8655. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167865509002323⟩. Citado na página 34.

JANG, K.; VINITSKY, E.; CHALAKI, B.; REMER, B.; BEAVER, L.; MALIKOPOULOS, A. A.; BAYEN, A. Simulation to scaled city: Zero-shot policy transfer for traffic control via autonomous vehicles. In: *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. New York, NY, USA: Association for Computing Machinery, 2019. (ICCPS '19), p. 291–300. ISBN 9781450362856. Disponível em: ⟨https://doi.org/10.1145/3302509.3313784⟩. Citado 2 vezes nas páginas 48 e 50.

KAMINKA, G. A.; FRIDMAN, N. Simulating urban pedestrian crowds of different cultures. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 9, n. 3, p. 27, 2018. Citado 2 vezes nas páginas 48 e 52.

KEOGH, E.; RATANAMAHATANA, C. A. Exact indexing of dynamic time warping. *Knowledge and information systems*, Springer, v. 7, n. 3, p. 358–386, 2005. Citado na página 37.

KHUNAYN, E. B.; KARUNASEKERA, S.; XIE, H.; RAMAMOHANARAO, K. Exploiting data dependency to mitigate stragglers in distributed spatial simulation. In: *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York, NY, USA: Association for

Computing Machinery, 2017. (SIGSPATIAL '17). ISBN 9781450354905. Disponível em: ⟨https://doi.org/10.1145/3139958.3140018⟩. Citado 2 vezes nas páginas 48 e 50.

KRAJZEWICZ, D.; HERTKORN, G.; RÖSSEL, C.; WAGNER, P. Sumo (simulation of urban mobility) - an open-source traffic simulation. In: AL-AKAIDI, A. (Ed.). *4th Middle East Symposium on Simulation and Modelling*. [s.n.], 2002. p. 183–187. LIDO-Berichtsjahr=2004,. Disponível em: ⟨https://elib.dlr.de/6661/⟩. Citado na página 105.

KYRIAZOPOULOU, C. Smart city technologies and architectures: A literature review. In: IEEE. *2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*. Lisbon, Portugal, 2015. p. 1–12. Citado na página 42.

LIN, Z.; YAO, Y. Load balancing for parallel discrete event simulation of stochastic reaction and diffusion. In: IEEE. *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. Chengdu, China, 2015. p. 609–614. Citado 2 vezes nas páginas 48 e 55.

LIN, Z.; YAO, Y. Parallel discrete event simulation of stochastic reaction and diffusion using reverse computation. In: IEEE. *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. Chengdu, China, 2015. p. 643–648. Citado 2 vezes nas páginas 48 e 54.

LOUNIS, M.; BOUNCEUR, A.; EULER, R.; POTTIER, B. Estimation of energy consumption through parallel computing in wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, Springer, p. 1–13, 2017. Citado 2 vezes nas páginas 48 e 53.

MA, M.; PREUM, S. M.; STANKOVIC, J. A. Simulating conflict detection in heterogeneous services of a smart city: Demo abstract. In: *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*. New York, NY, USA: Association for Computing Machinery, 2017. (IoTDI '17), p. 275–276. ISBN 9781450349666. Disponível em: ⟨https://doi.org/10.1145/3054977.3057290⟩. Citado na página 50.

MACQUEEN, J. *et al.* Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.], 1967. v. 1, n. 14, p. 281–297. Citado na página 34.

MANZANILLA-SALAZAR, O.; MELLAH, H.; MALANDRA, F.; SANSò, B. An algorithm for threading assignment in large-scale wireless network mobile simulations. In: *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. [S.l.: s.n.], 2021. p. 1–10. Citado 2 vezes nas páginas 49 e 53.

MOLITOR, C.; GROSS, S.; ZEITZ, J.; MONTI, A. Mescos—a multienergy system cosimulator for city district energy systems. *IEEE Transactions on Industrial Informatics*, IEEE, v. 10, n. 4, p. 2247–2256, 2014. Citado 2 vezes nas páginas 48 e 54.

MONTORI, F.; CORTESI, E.; BEDOGNI, L.; CAPPONI, A.; FIANDRINO, C.; BONONI, L. Crowdsensim 2.0: A stateful simulation platform for mobile crowdsensing in smart cities. In: . New York, NY, USA: Association for Computing Machinery, 2019. (MSWIM '19), p.

289–296. ISBN 9781450369046. Disponível em: ⟨https://doi.org/10.1145/3345768.3355929⟩. Citado 2 vezes nas páginas 49 e 53.

MOTLAGH, O.; LI, J. A model of smart meter time series. In: *Proceedings of the 11th International Conference on Computer Modeling and Simulation*. New York, NY, USA: Association for Computing Machinery, 2019. (ICCMS 2019), p. 79–83. ISBN 9781450366199. Disponível em: ⟨https://doi.org/10.1145/3307363.3307394⟩. Citado 3 vezes nas páginas 48, 55 e 56.

NAM, T.; PARDO, T. A. Conceptualizing smart city with dimensions of technology, people, and institutions. In: *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*. New York, NY, USA: Association for Computing Machinery, 2011. (dg.o '11), p. 282–291. ISBN 9781450307628. Disponível em: ⟨https://doi.org/10.1145/2037556.2037602⟩. Citado 2 vezes nas páginas 17 e 42.

NOSRATABADI, S.; MOSAVI, A.; KEIVANI, R.; ARAM, F. *et al.* State of the art survey of deep learning and machine learning models for smart cities and urban sustainability. Preprints, 2019. Citado na página 42.

PAPARRIZOS, J.; GRAVANO, L. K-shape: Efficient and accurate clustering of time series. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2015. (SIGMOD '15), p. 1855–1870. ISBN 9781450327589. Disponível em: ⟨https://doi.org/10.1145/2723372.2737793⟩. Citado 6 vezes nas páginas 38, 39, 40, 41, 63 e 64.

PARLETT, B. N. The rayleigh quotient iteration and some generalizations for nonnormal matrices. *Mathematics of Computation*, v. 28, n. 127, p. 679–693, 1974. Citado na página 41.

PETROVIć, N.; KONIčANIN, S.; MILIć, D.; SULJOVIć, S.; PANIć, S. Gpu-enabled framework for modelling, simulation and planning of mobile networks in smart cities. In: *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*. [S.l.: s.n.], 2020. p. 280–285. Citado 2 vezes nas páginas 49 e 53.

RAKOW, C. A framework for simulating mobility services in large scale agent-based transportation systems. In: *Proceedings of the 2019 Summer Simulation Conference*. San Diego, CA, USA: Society for Computer Simulation International, 2019. (SummerSim '19). Citado 2 vezes nas páginas 49 e 52.

RAMAMOHANARAO, K.; XIE, H.; KULIK, L.; KARUNASEKERA, S.; TANIN, E.; ZHANG, R.; KHUNAYN, E. B. Smarts: Scalable microscopic adaptive road traffic simulator. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 8, n. 2, p. 26, 2017. Citado 2 vezes nas páginas 48 e 50.

SANTANA, E. F. Z. *InterSCSimulator: a scalable, open source, smart city simulator*. Tese (Doutorado) — Universidade de São Paulo, Instituto de Matemática e Estatística, 3 2019. Citado 11 vezes nas páginas 23, 24, 25, 26, 27, 28, 30, 50, 57, 63 e 75.

SANTANA, E. F. Z.; LAGO, N.; KON, F.; MILOJICIC, D. S. Interscsimulator: Large-scale traffic simulation in smart cities using erlang. In: DIMURO, G. P.; ANTUNES,

L. (Ed.). *Multi-Agent Based Simulation XVIII*. Cham: Springer International Publishing, 2018. p. 211–227. ISBN 978-3-319-91587-6. Citado 14 vezes nas páginas 17, 18, 19, 20, 22, 29, 30, 42, 48, 51, 52, 56, 57 e 104.

SCHIERA, D. S.; BARBIERATO, L.; LANZINI, A.; BORCHIELLINI, R.; PONS, E.; BOMPARD, E. F.; PATTI, E.; MACII, E.; BOTTACCIOLI, L. A distributed platform for multi-modelling co-simulations of smart building energy behaviour. In: *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*. [S.l.: s.n.], 2020. p. 1–6. Citado 2 vezes nas páginas 49 e 55.

SONG, T.; KALESHI, D.; ZHOU, R.; BOUDEVILLE, O.; MA, J.-X.; PELLETIER, A.; HADDADI, I. Performance evaluation of integrated smart energy solutions through large-scale simulations. In: IEEE. *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. Brussels, Belgium, 2011. p. 37–42. Citado 2 vezes nas páginas 18 e 22.

TAKENS, F. Detecting strange attractors in turbulence. In: RAND, D.; YOUNG, L.-S. (Ed.). *Dynamical Systems and Turbulence, Warwick 1980*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981. p. 366–381. ISBN 978-3-540-38945-3. Citado na página 55.

TANG, J.; LIU, S.; WANG, C.; LIU, C. Distributed simulation platform for autonomous driving. In: PENG, S.-L.; LEE, G.-L.; KLETTE, R.; HSU, C.-H. (Ed.). *Internet of Vehicles. Technologies and Services for Smart Cities*. Cham: Springer International Publishing, 2017. p. 190–200. ISBN 978-3-319-72329-7. Citado 2 vezes nas páginas 48 e 54.

WAGSTAFF, K.; CARDIE, C.; ROGERS, S.; SCHRöDL, S. Constrained k-means clustering with background knowledge. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. (ICML '01), p. 577–584. ISBN 1558607781. Citado na página 34.

WEYL, J.; LENFERS, U. A.; CLEMEN, T.; GLAKE, D.; PANSE, F.; RITTER, N. Large-scale traffic simulation for smart city planning with mars. In: *Proceedings of the 2019 Summer Simulation Conference*. San Diego, CA, USA: Society for Computer Simulation International, 2019. (SummerSim '19). Citado 2 vezes nas páginas 49 e 50.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, Elsevier, v. 2, n. 1-3, p. 37–52, 1987. Citado na página 41.

XU, Y.; TAN, G. Sim-tree: Indexing moving objects in large-scale parallel microscopic traffic simulation. In: *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. New York, NY, USA: Association for Computing Machinery, 2014. (SIGSIM PADS '14), p. 51–62. ISBN 9781450327947. Disponível em: ⟨https://doi.org/10.1145/2601381.2601388⟩. Citado 2 vezes nas páginas 48 e 50.

YOVANOF, G. S.; HAZAPIS, G. N. An architectural framework and enabling wireless technologies for digital cities & intelligent urban environments. *Wireless personal communications*, Springer, v. 49, n. 3, p. 445–463, 2009. Citado na página 17.