ANDRÉ ELIAS LAUER

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PLANEJAMENTO DE LINHAS E FREQUÊNCIAS DO BRT-ABC

São Paulo
2024

ANDRÉ ELIAS LAUER

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PLANEJAMENTO DE LINHAS E FREQUÊNCIAS DO BRT-ABC

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção.

São Paulo
2024

ANDRÉ ELIAS LAUER

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PLANEJAMENTO DE LINHAS E FREQUÊNCIAS DO BRT-ABC

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção.

Orientadora:

Prof.ª Titular Débora Pretti Ronconi

São Paulo
2024

AGRADECIMENTOS

Aos meus familiares e amigos, pelo apoio incondicional e por tornarem esta jornada mais leve e significativa.

À Professora Débora Pretti Ronconi, pela excelente orientação e parceria, e por despertar em mim um grande interesse por um tema tão fascinante. Seu apoio foi fundamental para o sucesso deste projeto.

A todos os professores que contribuíram para minha formação, especialmente ao Professor Mauro Munsignatti Junior, por inspirar e motivar minha busca pelo conhecimento na matemática.

À Escola Politécnica, pela formação de excelência e pelas experiências valiosas que contribuíram para meu crescimento pessoal e profissional.

"Mathematics, rightly viewed,
possesses not only truth, but supreme
beauty."
- Bertrand Russell

RESUMO

O transporte público desempenha um papel crucial para a sociedade, impactando significativamente a qualidade de vida de seus usuários. Reconhecendo essa importância, a Empresa Metropolitana de Transportes Urbanos (EMTU) está desenvolvendo o BRT-ABC, um sistema de ônibus elétricos que conectará a região do Grande ABC à capital paulista. Este trabalho tem como objetivo estudar e otimizar as linhas e frequências do BRT-ABC, visando aprimorar a eficiência e a qualidade do transporte público urbano.

O problema foi classificado como um TNDFSP (*Transit Network Design and Frequency Setting Problem*). Sua formulação matemática, baseada no estudo de Martínez et al. (2017), adota uma abordagem multiobjetivo que considera dois objetivos de interesse do usuário e um objetivo de interesse da EMTU. Devido à complexidade e não-linearidade do problema, foram utilizados algoritmos heurísticos, implementados em Python, para sua resolução. A coleta de dados foi realizada através do Estudo Funcional e Conceitual do empreendimento, elaborado pela empresa Systra. Posteriormente, os dados foram tratados por meio do desenvolvimento de modelos matemáticos.

A partir dos resultados obtidos, foi possível sugerir um conjunto de linhas e frequências que apresentam um desempenho superior à solução atual da EMTU. Essa melhoria contribui significativamente para a eficiência do sistema de transporte público na região do ABC. Os próximos passos do projeto incluem o refinamento dos dados do BRT-ABC após o início das operações e a aplicação de um algoritmo genético para garantir a convergência da solução.

Palavras-Chave –Pesquisa Operacional, Heurística Construtiva, Problema de Transporte Urbano, Transporte Público, BRT.

ABSTRACT

Public transportation plays a crucial role in society, significantly impacting the quality of life of its users. Recognizing this importance, the Empresa Metropolitana de Transportes Urbanos (EMTU) is developing the BRT-ABC, an electric bus system that will connect the Greater ABC region to São Paulo's capital. This work aims to study and optimize the lines and frequencies of the BRT-ABC to improve the efficiency and quality of urban public transportation.

The problem has been classified as a TNDFSP (Transit Network Design and Frequency Setting Problem). Its mathematical formulation, based on the study by Martínez et al. (2017), adopts a multi-objective approach that considers two user-centric objectives and one objective of interest to EMTU. Due to the problem's complexity and non-linearity, heuristic algorithms implemented in Python were used for its resolution. Data collection was conducted through the Functional and Conceptual Study of the project, prepared by the company Systra. Subsequently, the data were processed through the development of mathematical models.

Based on the obtained results, it was possible to suggest a set of lines and frequencies that demonstrate superior performance compared to EMTU's current solution. This improvement significantly contributes to the efficiency of the public transportation system in the ABC region. The next steps of the project include refining the BRT-ABC data after the beginning of operations and applying a genetic algorithm to ensure the convergence of the solution.

Keywords – Operational Research, Constructive Heuristic, Urban Transportation Problem, Public Transportation, BRT.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A temática do transporte público é de grande importância para o desenvolvimento e funcionamento sustentável da sociedade, sendo o meio de transporte utilizado por milhões de pessoas no Estado de São Paulo. Diante dessa relevância, o Governo de São Paulo, junto à EMTU (Empresa Metropolitana de Transportes Urbanos de São Paulo), iniciou em fevereiro de 2022 as obras do BRT-ABC, sistema rápido de ônibus elétricos que conectará a região do Grande ABC à capital. O novo empreendimento visa atender aproximadamente 173 mil passageiros por dia (EMTU 2021).

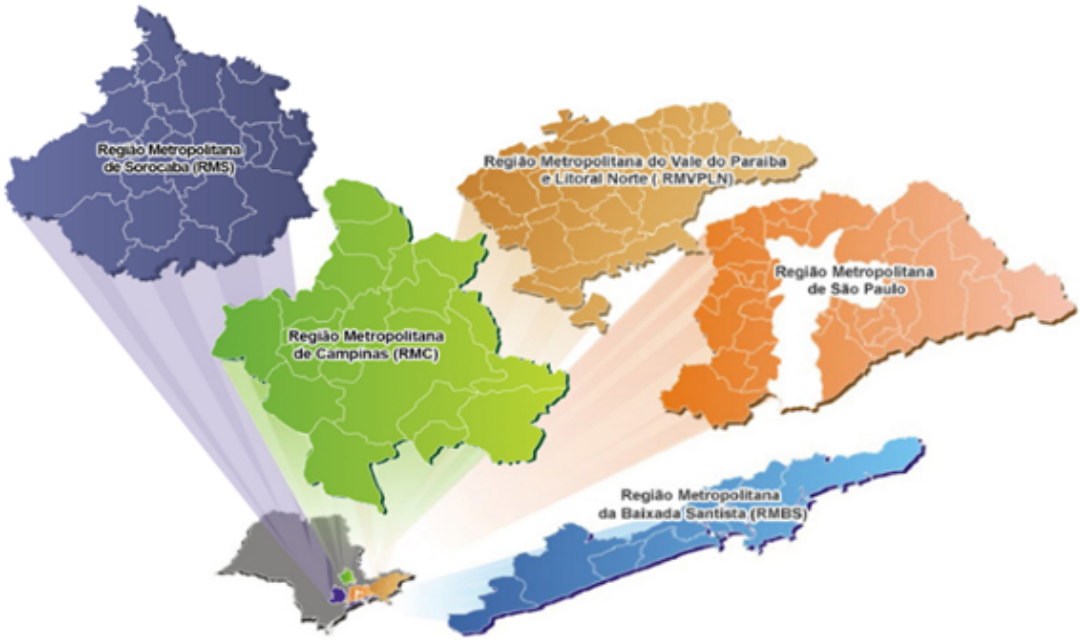
Considerando-se a relevância deste tema para a população, este trabalho busca realizar o estudo de otimização do BRT-ABC, através da aplicação de técnicas de otimização para as linhas e frequências do corredor.

Neste capítulo, apresenta-se a empresa responsável pela construção do BRT, a definição do problema, os objetivos deste trabalho e a sua estrutura.

1.1 Descrição da Empresa

A Empresa Metropolitana de Transportes Urbanos de São Paulo (EMTU/SP) é uma entidade pública sob o controle do Governo do Estado de São Paulo e vinculada à Secretaria de Estado dos Transportes Metropolitanos (STM). Sua principal função é fiscalizar e regulamentar o transporte metropolitano de baixa e média capacidade nas cinco Regiões Metropolitanas do Estado de São Paulo: São Paulo, Campinas, Sorocaba, Baixada Santista e Vale do Paraíba e Litoral Norte. Ao todo, essas áreas abrangem 134 municípios, cuja rede de transportes intermunicipais é gerida pela EMTU/SP.

Figura 1: Regiões Metropolitanas do Estado de São Paulo



Fonte: EMTU/SP, 2024

A criação da Empresa Metropolitana de Transportes Urbanos de São Paulo (EMTU/SP) foi parte de um processo maior de transformação urbana e social nas décadas de 1970 e 1980. Nesse período, a industrialização começou a conectar municípios vizinhos, pois muitos trabalhadores moravam em uma cidade e se deslocavam para trabalhar em outra, fomentando o surgimento das Regiões Metropolitanas.

Nos anos 70, a necessidade de regulamentar e organizar o transporte intermunicipal tornou-se evidente. Inicialmente, esses deslocamentos eram realizados por ônibus regulamentados pelo Departamento Estadual de Estradas de Rodagem (DER). Em resposta à crescente demanda e complexidade do transporte entre cidades, o Governo do Estado de São Paulo estabeleceu a EMTU em 13 de dezembro de 1977, através da Lei nº 1.492, instituindo o Sistema Metropolitano de Transportes Urbanos. A EMTU foi criada para coordenar e regulamentar os serviços de transporte prestados por concessionárias, que operam as linhas de ônibus intermunicipais.

Originalmente focada na Região Metropolitana de São Paulo, a atuação da EMTU foi expandida para outras regiões ao longo do tempo. Em 1996, com a criação da Região Metropolitana da Baixada Santista (RMBS), a EMTU passou a regulamentar o transporte nessa área. Em 2000, a Região Metropolitana de Campinas (RMC) foi instituída, incluindo mais municípios sob a gestão da EMTU. Em 2012, a Região Metropolitana do

Vale do Paraíba e Litoral Norte (RMVPLN) foi criada, seguida pela Região Metropolitana de Sorocaba em 2014. Atualmente, a EMTU gerencia e fiscaliza o transporte metropolitano em 134 municípios, abrangendo uma população significativa e contribuindo para a mobilidade urbana de milhões de pessoas.

A EMTU não opera diretamente os ônibus, mas regula o serviço prestado por diversas concessionárias, assegurando que os padrões de qualidade e segurança sejam mantidos. Sua função é crucial para garantir que o transporte público intermunicipal atenda às necessidades da população de forma eficiente e integrada. A seguir, são detalhados os serviços oferecidos pela empresa:

Corredores e terminais: A EMTU administra diversos corredores e terminais para melhorar a mobilidade na Região Metropolitana de São Paulo. O Corredor Metropolitano ABD, com 32 km de faixas exclusivas, liga São Mateus a Jabaquara. A Extensão Diadema – São Paulo (Morumbi/Berrini) adiciona 12 km de faixa exclusiva. O Corredor Metropolitano Noroeste conecta Campinas a municípios vizinhos, enquanto outros corredores como Guarulhos – São Paulo e Itapevi – São Paulo também são importantes. A EMTU também gerencia terminais como o Terminal Cotia e a Estação de Transferência Munhoz Júnior, equipados com infraestrutura de acessibilidade.

Sistema Regular: O sistema metropolitano de transporte gerido pela EMTU/SP é operado por empresas privadas e inclui dois tipos de serviços: comum e seletivo. O Serviço Comum utiliza ônibus urbanos para transportar passageiros sentados e em pé entre municípios de uma mesma região metropolitana. O Serviço Seletivo emprega ônibus rodoviários que transportam apenas passageiros sentados. Na RMSP, a EMTU também gerencia o sistema de Reserva Técnica Operacional (RTO), com linhas intermunicipais operadas por micro-ônibus ou vans com até 20 lugares. Este sistema inclui nove terminais metropolitanos, cinco estações de transferência, 156 paradas e uma frota de cerca de 260 veículos, transportando mais de 200 mil passageiros diariamente. Desde 1997, a operação é concedida à empresa Metra (Sistema Metropolitano de Transportes Ltda).

Fretamento: trata-se de uma modalidade de transporte gerida pela Secretaria de Estado dos Transportes Metropolitanos (STM), realizada por empresas registradas e que transportam pessoas a destinos pré-estabelecidos nas regiões metropolitanas do Estado de São Paulo. Esse serviço é regulamentado por diversos decretos e resoluções da STM. Os serviços de fretamento são classificados em quatro tipos: Contínuo, Eventual, Próprio e Escolar.

Airport Bus Service: foi criado para facilitar o deslocamento entre a Região Metropolitana de São Paulo e os aeroportos locais. Operado pelo Consórcio Internorte de

Transportes, oferece linhas executivas que conectam o Aeroporto Internacional de São Paulo em Guarulhos ao Aeroporto de Congonhas, Terminais Rodoviários do Tietê e da Barra Funda, e hotéis na Paulista/Augusta. Há também uma linha suburbana específica para trabalhadores na região do Aeroporto de Guarulhos.

VLT da Baixada Santista: começou a operar em abril de 2015, ligando o Terminal Barreiros, em São Vicente, à Estação Porto, em Santos, ao longo de 11,5 km com 15 estações. O percurso inclui ciclovia e paraciclos, com um bicicletário no Terminal Barreiros. Além de ser moderno e não poluente, o VLT reduz o tempo de viagem e a poluição sonora. Desde 2016, oferece integração com linhas municipais e metropolitanas da Baixada Santista.

A EMTU está envolvida em vários empreendimentos que visam melhorar a infraestrutura de transporte público nas regiões metropolitanas de São Paulo. Esses projetos incluem a renovação de corredores de ônibus, modernização de terminais e a introdução de novas tecnologias para melhorar a eficiência e o conforto dos usuários.

Destaca-se o projeto do BRT-ABC, que será o tema de estudo deste trabalho. Este sistema rápido de ônibus elétricos conectará a região do Grande ABC à capital paulista. Com um investimento de R$ 860 milhões, o BRT-ABC terá uma extensão de aproximadamente 17,3 km, passando por Santo André e São Caetano do Sul, beneficiando cerca 173 mil passageiros por dia. O BRT-ABC contará com ônibus elétricos, que serão articulados e equipados com ar-condicionado e internet wi-fi. As estações serão modernas, com climatização e painéis de previsão de chegada dos ônibus, e a cobrança de tarifa será feita nas estações para agilizar o embarque. Segundo o site technibus, as obras serão finalizadas no final de 2024, com início das operações no começo de 2025.

1.2 Descrição do problema

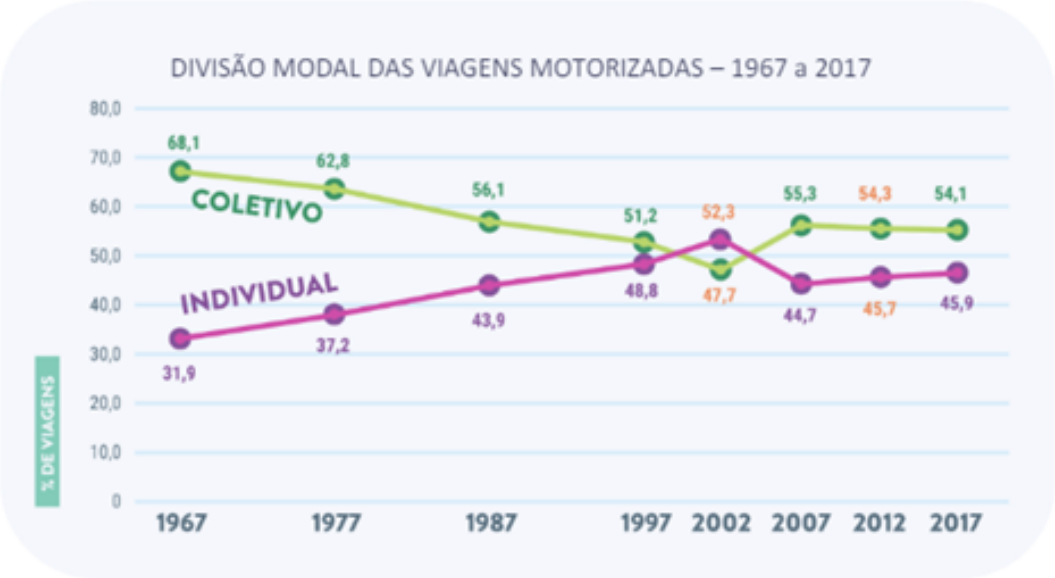
A qualidade do transporte público influencia significativamente a vida de milhares de pessoas que dependem dele diariamente. Para compreender melhor o problema abordado, é essencial entender a situação das famílias que residem na região do ABC, onde o BRT-ABC será implementado. Conhecer as condições socioeconômicas e demográficas dessas famílias ajuda a contextualizar a importância do projeto e sua contribuição para a melhoria da mobilidade urbana na região.

A Pesquisa Origem e Destino 2017, realizada pelo Metrô de São Paulo, tem como objetivo coletar e analisar dados sobre o fluxo de viagens na Região Metropolitana de

São Paulo. Esta pesquisa fornece informações muito relevantes sobre o transporte e a qualidade de vida das famílias. A região do ABC é classificada como Sub-região Sudeste na pesquisa, que abrange os municípios de Santo André, São Bernardo do Campo, São Caetano do Sul, Diadema, Mauá, Ribeirão Pires e Rio Grande da Serra.

De acordo com a Pesquisa OD 2017, o transporte coletivo representa 54,1% das viagens motorizadas na RMSP. No entanto, há uma tendência de queda na utilização do transporte coletivo em comparação ao transporte individual, conforme ilustrado na Figura 2.

Figura 2: Divisão modal das viagens motorizadas - 1967 a 2017



Fonte: Metrô-Pesquisas OD 2007 e 2017

Essa queda no uso do transporte público também é evidente na região Sudeste entre os anos de 2007 e 2017, com uma redução de 2,3% na participação do transporte coletivo no total de viagens durante esse período, conforme Tabela 1.

Tabela 1: Viagens por modo - Região Sudeste

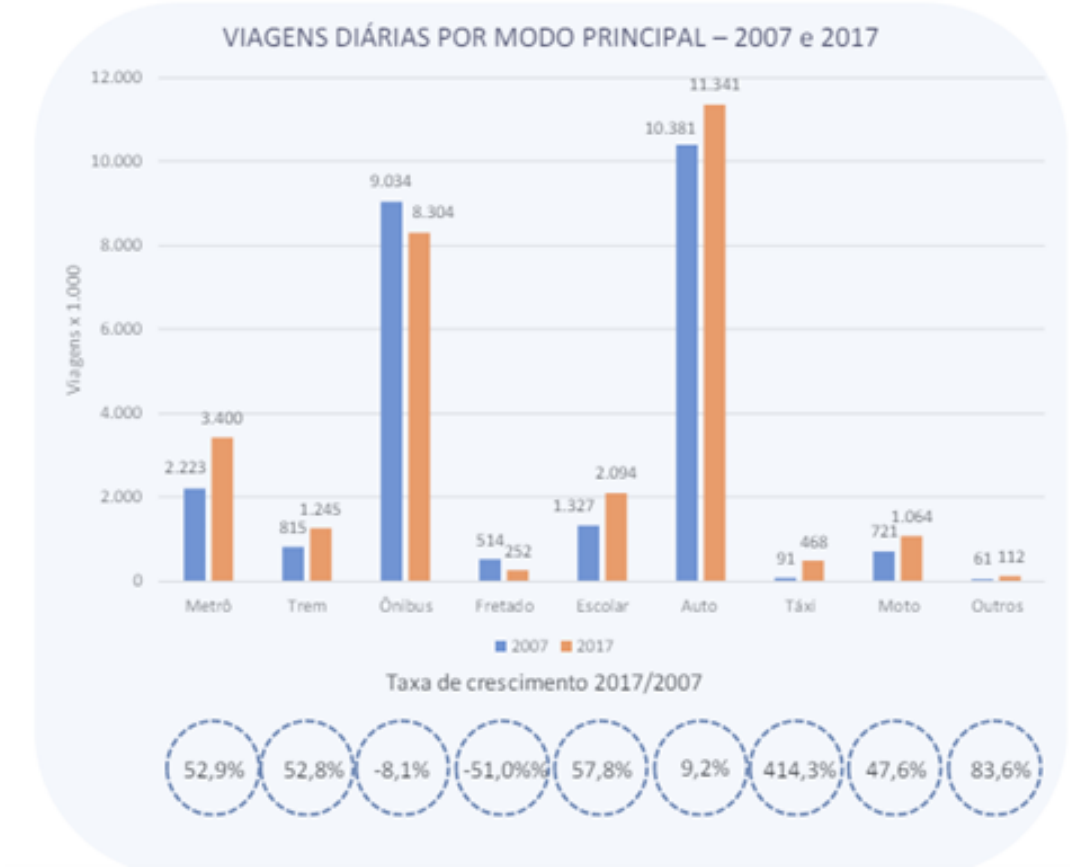
Ano	**Coletivo**		**Individual**	
	(x 1.000)	**%**	**(x 1.000)**	**%**
2007	1.672	46,7	1.905	53,3
2017	1.488	44,4	1.866	55,6

Fonte: adaptado de Metrô-Pesquisas OD 2007 e 2017.

Outro dado importante a ser destacado é a redução no uso de ônibus. Conforme a

Figura 3, a taxa de crescimento das viagens diárias na RMSP utilizando ônibus entre 2007 e 2017 foi de -8,1%.

Figura 3: Viagens diárias por modo principal - 2007 e 2017



Fonte: Metrô-Pesquisas OD 2007 e 2017

Além disso, ao focar na região Sudeste, a redução é ainda mais acentuada. De acordo com a pesquisa, o número de viagens de ônibus em 2007 foi de 1.169 (x1000), enquanto em 2017 foi de 875 (x1000), representando uma diminuição de 25,1%.

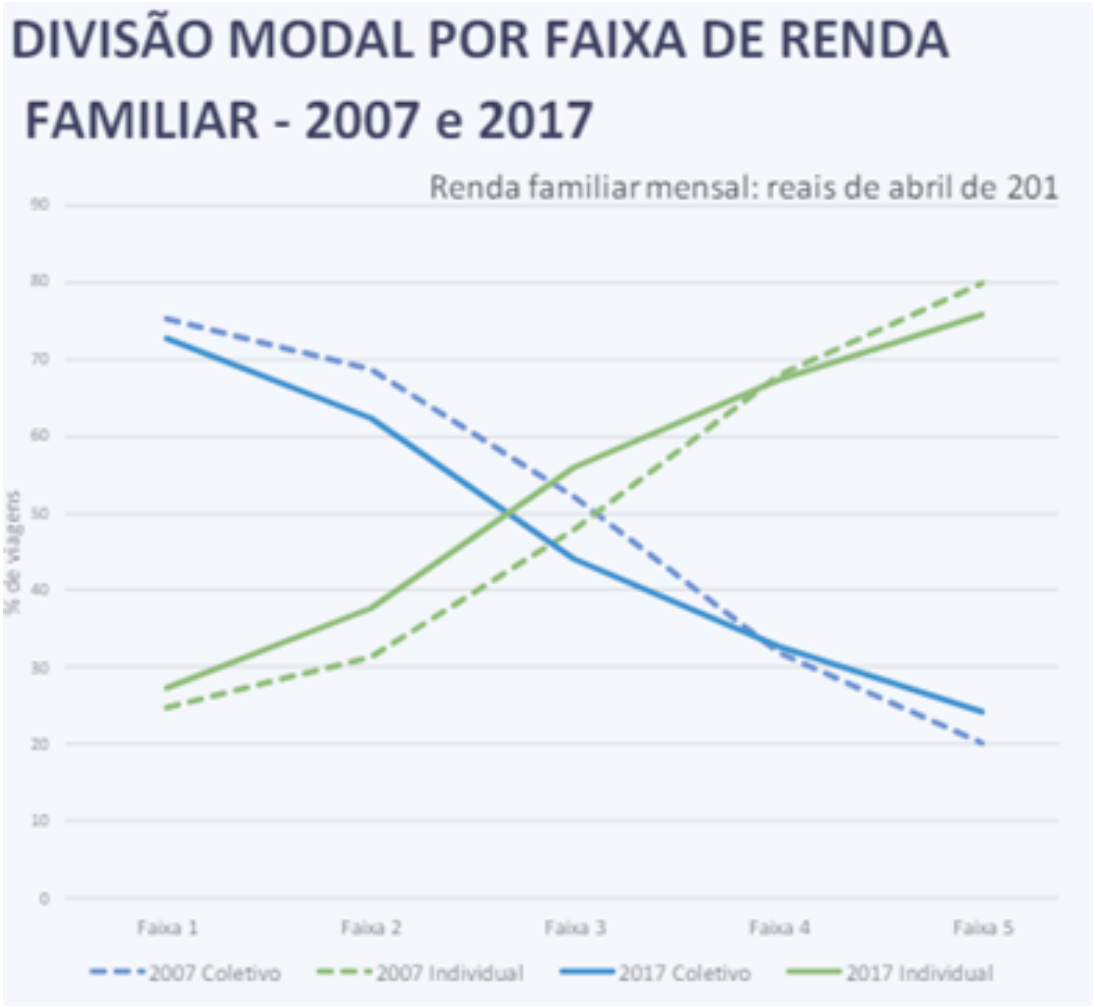
Os estudos de Beirão & Cabral (2007) e Dobbie et al. (2010) buscam entender por que as pessoas optam por não utilizar o transporte por ônibus. Os autores observaram que os usuários de transporte público frequentemente enfrentam várias barreiras ao utilizarem ônibus, afetando negativamente sua percepção e disposição para usá-los. Entre os problemas mais citados estão o comportamento inadequado dos motoristas e passageiros, além de preocupações com a segurança, confiabilidade e acessibilidade dos veículos, bem como a limpeza e conforto a bordo. Há também preocupações com a segurança nas paradas de ônibus e a adequação das informações disponíveis. A percepção de que as viagens são longas, de que os horários não são seguidos pontualmente, e a falta de rotas diretas

desestimulam o uso. Esses problemas são agravados pela sensação de perda de tempo, superlotação, falta de conforto e flexibilidade.

A redução no uso do transporte público, especialmente dos ônibus, contraria a visão de um sistema de transporte eficiente para centros urbanos. A demanda de transporte nas grandes cidades só pode ser atendida por um sistema público de alta qualidade que, quando bem projetado, demonstra elevada viabilidade econômica. Além disso, áreas urbanas com alta dependência de carros enfrentam três principais problemas: segurança, congestionamento e poluição (sonora e do ar) (Schmöcker et al., 2003).

Outra informação de grande importância trazida pela Pesquisa OD é a divisão modal por faixa de renda familiar. Na pesquisa, foram consideradas as seguintes faixas de renda: FAIXA 1: até 1.908 reais, FAIXA 2: de 1.908 a 3.816 reais, FAIXA 3: de 3.816 a 7.632 reais, FAIXA 4: de 7.632 a 11.448 reais e FAIXA 5: mais de 11.448 reais, valores estabelecidos em abril de 2018. A Figura 4 ilustra essa divisão modal por renda familiar.

Figura 4: Divisão modal por faixa de renda familiar - 2007 e 2017

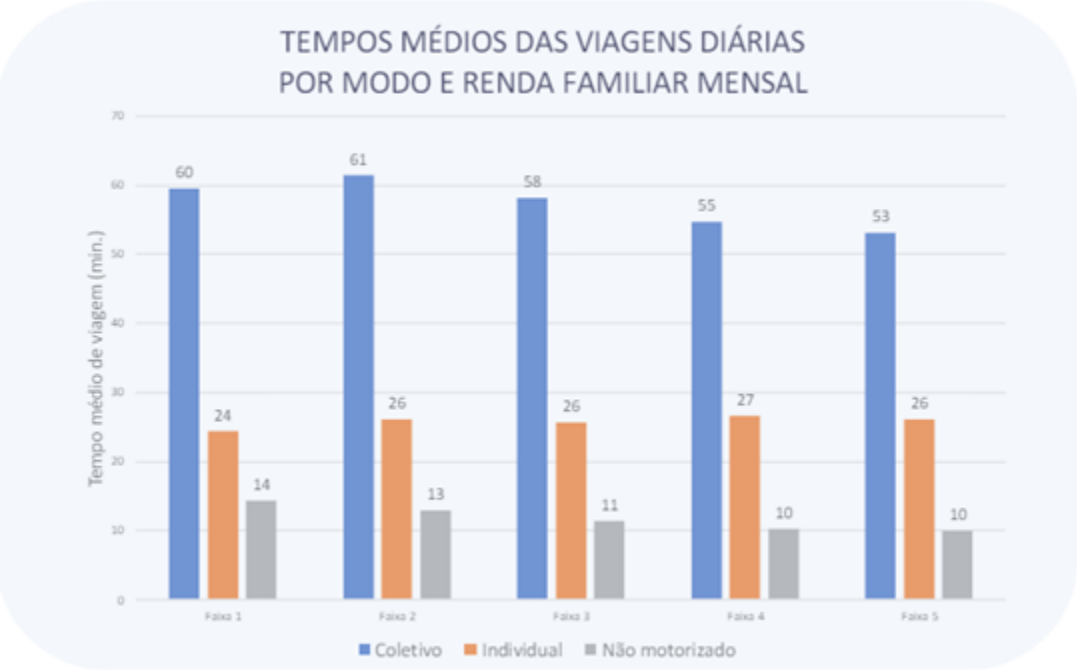


Fonte: Metrô-Pesquisas OD 2007 e 2017

Portanto, observa-se que o transporte coletivo desempenha um papel fundamental, especialmente para a população mais carente, com uma participação de 72,7% na FAIXA 1 e 62,4% na FAIXA 2 em 2017. Além disso, no mesmo ano, o ônibus foi o principal meio de transporte, excluindo as viagens a pé, para as Faixas 1 e 2, representando respectivamente 23,0% e 23,2% do total de viagens dessas faixas. Nas Faixas 3 e 4, embora o número de viagens de carro supere o de ônibus, este ainda é o meio de transporte coletivo mais utilizado, representando 16,2% e 11,2%, respectivamente. Juntas, as faixas 1, 2 e 3 representam 95,4% do total de viagens de ônibus na região da RMSP.

Por fim, nota-se que o transporte público afeta diretamente a qualidade de vida das pessoas, pois possui tempos de viagem muito maiores se comparados aos modais individuais e não motorizados, conforme mostra a Figura 5.

Figura 5: Tempos médios das viagens diárias por modo e renda familiar mensal



Fonte: Metrô-Pesquisas OD 2007 e 2017

Portanto, conclui-se que o sistema de transporte público é essencial para a qualidade de vida das pessoas e para o bom funcionamento do transporte na Região Metropolitana de São Paulo, especialmente os ônibus, que são o modal coletivo mais utilizado. No entanto, entre 2007 e 2017, observou-se uma diminuição no uso do transporte coletivo, com destaque para os ônibus. Essa tendência vai na contramão da necessidade de um sistema de transporte eficiente e sustentável em áreas urbanas densamente povoadas. O problema a ser abordado, portanto, é a melhoria do transporte público, com o objetivo de estimular seu uso e, assim, elevar a qualidade de vida das milhares de pessoas que dependem desse serviço diariamente.

1.3 Objetivo do trabalho

O BRT-ABC desempenhará um papel crucial na melhoria da mobilidade na Região Metropolitana de São Paulo (RMSP). Segundo o Estudo Funcional e Conceitual do projeto, elaborado pela Systra, este sistema visa estabelecer uma ligação vital entre o sistema metroferroviário de alta capacidade, especificamente a Linha 2 – Verde do Metrô e a Linha 10 – Turquesa da CPTM, integradas na Estação Tamanduateí, e a região do Grande ABC Paulista, que inclui os municípios de São Caetano do Sul, Santo André e São Ber-

nardo do Campo. Adicionalmente, o BRT ABC conectará o Terminal/Estação Sacomã e o Expresso Tiradentes da SPTrans, facilitando uma nova integração metropolitana.

O Estudo Funcional e Conceitual também destaca que este empreendimento proporcionará à população desses municípios um sistema de transporte seguro, rápido e acessível, com alto padrão de confiabilidade. A implantação do BRT ABC atenderá a demanda utilizando vias subutilizadas e através de uma política integrada de transporte entre esses municípios. Além disso, o dinamismo econômico da região sudeste da RMSP resulta em um maior número de viagens internas na região, aumentando a necessidade de um sistema de transporte coletivo eficiente.

Diante do contexto apresentado anteriormente, o objetivo deste trabalho é analisar e propor melhorias para o sistema de transporte público na Região Metropolitana de São Paulo, através da otimização das linhas e frequências do BRT-ABC. Através de uma abordagem baseada em técnicas de otimização, busca-se aplicar um modelo eficiente que atenda às necessidades de mobilidade da população, promovendo um transporte mais rápido, confiável e acessível. Este estudo visa não apenas melhorar a eficiência operacional do BRT-ABC, mas também aumentar sua atratividade e uso, revertendo a tendência de diminuição na utilização do transporte coletivo observada nos últimos anos.

Para alcançar esse objetivo, serão utilizados métodos de modelagem matemática e algoritmos heurísticos para determinar as melhores configurações de linhas e frequências do BRT-ABC. A análise levará em consideração fatores como a demanda de passageiros, tempos de viagem e custos do operador (neste trabalho, operador será o nome atribuído à empresa/instituição responsável por gerenciar a rede de ônibus). Espera-se que, ao implementar as melhorias propostas, o sistema de transporte público se torne mais eficiente e competitivo, satisfazendo tanto o usuário, como o operador.

1.4 Estrutura do trabalho

Este trabalho está estruturado de forma a abordar de maneira abrangente e detalhada a otimização das linhas e frequências do BRT-ABC. Inicialmente, na Introdução, o problema é contextualizado, destacando sua importância e relevância para a Região Metropolitana de São Paulo. Em seguida, o próximo capítulo apresenta a revisão bibliográfica, na qual se realiza uma análise de trabalhos anteriores relacionados ao tema, destacando os principais métodos de solução utilizados.

O capítulo subsequente é dedicado à formulação e ao método de solução propostos, no

qual é definida a modelagem e as técnicas de otimização que serão aplicadas para resolver o problema específico do BRT-ABC. Depois, é apresentado o capítulo sobre a Coleta e Tratamento dos Dados, no qual são detalhados os dados coletados e os procedimentos adotados para preparar esses dados para análise.

A seguir, é feita a validação do método proposto. Este capítulo expõe a definição de parâmetros essenciais para o projeto, além da apresentação das soluções encontradas. Finalmente, o trabalho é encerrado com um capítulo dedicado às conclusões, no qual são apresentadas as principais contribuições do estudo, além dos próximos passos a serem seguidos, visando a continuidade e aprofundamento das pesquisas e melhorias no sistema de transporte público da região.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, são apresentados os principais tópicos relacionados ao problema do planejamento de redes de transporte, com foco no *Transit Design and Frequencies Setting Problem* (TNDFSP). Inicialmente, discute-se a amplitude e a complexidade do planejamento de transporte, destacando a decomposição do problema em subproblemas menores e as diferentes classificações do problema de design de redes de transporte urbano (UTNDP) conforme a literatura.

Em seguida, a revisão aborda o processo de planejamento de redes de transporte e a definição das frequências das linhas de ônibus, considerando as restrições e os objetivos envolvidos. Além disso, categoriza os métodos de solução encontrados na literatura em modelos matemáticos, heurísticos e meta-heurísticos, abordando algumas de suas aplicações e contribuições.

Por fim, é apresentado um estudo de caso específico sobre a aplicação de um Algoritmo Genético para o planejamento de linhas e frequências de um corredor de BRT na Colômbia.

2.1 O problema do planejamento do transporte

O planejamento do transporte abrange uma área de pesquisa muito ampla, pois envolve muitos fatores que estão relacionados a seu estudo e solução dos seus problemas. Um problema geral de planejamento do transporte possui um grau de dificuldade e complexidade bem alto, sendo normalmente decomposto em problemas menores. Ainda assim, os sub-problemas também são de difícil resolução, sendo considerados *NP-Hard* do ponto de vista computacional. (Guihaire & Hao, 2008; Magnanti & Wong, 1984)

Os problemas de design de redes de transporte urbano (UTNDP – *Urban Transportation Network Design Problems*), nomenclatura utilizada em Farahani et al. (2013), possuem pelo menos três diferentes classificações na literatura, de acordo com os autores. São elas:

1. Definição comum na literatura e utilizada em Dantzig et al. (1979), considera-se que o UTNDP se preocupa em construir novas ruas ou aumentar a capacidade das ruas existentes. Outra nomenclatura comum para essa definição é a de RNDP –*Road Network Design Problem*, que se aplica a problemas relacionados ao estudo de rodovias;

2. O UTNDP tem como objetivo definir as localizações ótimas que devem ser adicionadas a uma rede de transportes, ou determinar as melhorias ótimas de capacidade das instalações existentes em uma rede (Friesz, 1985). Esta definição é bem mais abrangente que a anterior, na qual as instalações são consideradas como pontos/nós para a modelagem;

3. Esta definição é a mais completa e abrangente das três, configurando o UTNDP como uma hierarquia completa de processos de tomada de decisão no planejamento do transporte, incluindo decisões estratégicas, táticas e operacionais (Magnanti & Wong, 1984). Para essa definição, é necessário estabelecer o que são as decisões estratégicas, táticas e operacionais.

Segundo Farahani et al. (2013), pode-se classificar estas decisões como:

- Estratégicas: decisões de longo-prazo relacionadas à infraestrutura das redes de transporte como um todo, incluindo tanto a rede de transporte e rede rodoviária;

- Táticas: decisões relacionadas à utilização efetiva das infraestruturas e recursos já existentes da rede de transportes urbana;

- Operacionais: decisões de curto-prazo, relacionadas ao controle do fluxo de tráfego, gestão de demanda e problemas de cronograma.

A Tabela 2 ilustra exemplos para as decisões estratégicas, táticas e operacionais.

Tabela 2: Decisões estratégicas, táticas e operacionais

Estratégica	**Tática**	**Operacional**
Construir novas ruas	Determinar a orientação de vias de mão única	Determinar a programação dos semáforos
Projetar linhas de ônibus	Alocar faixas exclusivas de ônibus	Determinar o cronograma dos transportes
Expandir ruas existentes	Determinar a alocação de faixas para vias de mão dupla	Determinar o cronograma de reparos às vias urbanas
	Determinar a frequência dos serviços de trânsito	

Fonte: Farahani et al. (2013)

Considerando as três definições da literatura, estabelece-se uma definição para o UTNDP que se aplica bem ao presente trabalho. Adota-se a terceira definição, mas se limita principalmente às decisões especificadas na segunda definição, relacionadas à topologia da rede de transportes. Dessa forma, essa definição inclui duas grandes classes de UTNDP: (1) o problema de desenvolver uma nova rede através da adição de novos elementos, relacionado a decisões estratégicas, e (2) o problema de melhorar ou gerenciar a rede atual, relacionado a decisões táticas e operacionais (Farahani et al., 2013). Nota-se que o projeto do BRT-ABC se inicia extremamente correlacionado à grande classe (1), pois representa uma decisão estratégica do Governo de São Paulo visando melhorar as condições de transporte público na região do ABC, ao introduzir um sistema rápido de ônibus elétricos em faixas exclusivas. Somado a isso, o BRT-ABC também abrange a grande classe (2), já que requer um estudo conceitual e funcional para implementá-lo e integrá-lo a outros meios de transporte da região do ABC, aprimorando a rede de transporte existente. Este trabalho foca principalmente na grande classe (2), buscando otimizar o BRT-ABC.

Segundo Ceder & Wilson (1986), o processo de resolução de um problema de planejamento do transporte pode ser decomposto em 5 componentes, conforme Tabela 3.

Tabela 3: Componentes para a resolução de um problema de planejamento do transporte

***Inputs* Independentes**	**Atividades de Planejamento**	**Outputs**
Dados de demanda Dados de fornecimento Indicadores de performance das linhas	**Design da rede de transportes**	Mudanças nas linhas; Novas linhas; Estratégias de operação
Subsídio disponível Número de ônibus disponíveis Políticas de serviço Patrocínio atual	**Definição das frequências**	Frequências do serviço
Demanda por hora do dia Horários para a primeira e última viagem Tempos de viagem	**Desenvolvimento do cronograma**	Horários de partida das linhas Horários de chegada das linhas
Tempo de deslocamento vazio Tempo de recuperação Restrições de cronograma Estrutura de custos	**Desenvolvimento do cronograma dos ônibus**	Cronograma dos ônibus
Regras de trabalho para os motoristas Estrutura de custos operacionais	**Desenvolvimento do cronograma dos motoristas**	Cronograma dos motoristas

Fonte: Ceder & Wilson, 1986

O cenário ideal seria resolver os cinco problemas simultaneamente e de forma integrada, garantindo uma boa interação e conexão entre eles. No entanto, na prática, é impossível abordar o planejamento de transporte dessa maneira. Portanto, a abordagem mais eficaz é fragmentá-lo em diversos subproblemas. No trabalho desenvolvido para o BRT-ABC, apenas as duas primeiras etapas são tratadas e modeladas, ou seja, considera-se apenas o design da rede de transportes e a definição das frequências. A utilização de apenas essas etapas caracteriza o problema como um TNDFSP – *Transit Design and Frequencies Setting Problem* (Oliveira & Barbieri, 2015).

2.2 Planejamento da rede de transportes

De acordo com Guihaire & Hao (2008), o objetivo dessa componente é determinar um conjunto de linhas de ônibus, para uma área específica, na qual cada linha de ôni-

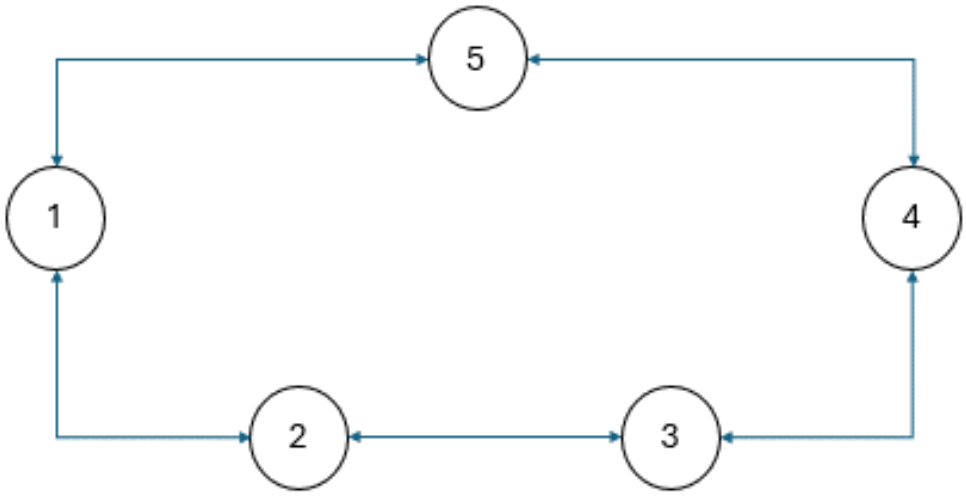
bus é formada por uma sequência de estações de ônibus. Para determiná-las, os *inputs* considerados são a topologia e a matriz de Origem-Destino.

A topologia refere-se às ruas e estradas da área, locais possíveis para pontos de ônibus e estações de transferência, além de possíveis armazéns que possam servir como os terminais da linha. Já a matriz de Origem-Destino é uma matriz que possui os índices das estações como coordenadas, e seus valores representam o número de passageiros que viajam de um índice até o outro em determinado período do dia. A matriz OD é fundamental para uma boa modelagem do problema, pois traduz as demandas da comunidade em que está se projetando a rede de transportes. Para obter-se uma matriz de Origem-Destino fiel a realidade, recomenda-se conduzir uma pesquisa por toda a comunidade, e não apenas pelos usuários atuais do transporte público.

Para ilustrar o funcionamento de uma matriz OD, considera-se o exemplo a seguir:

Foi projetada uma linha de ônibus que possui as estações de índice 1,2,3,4,5:

Figura 6: Exemplo da linha de ônibus



Fonte: Autor

Suponha que essa linha foi projetada em uma comunidade com 100 pessoas. Para o período de pico da manhã, a seguinte matriz OD foi obtida:

A matriz Origem-Destino do exemplo mostra que há 5 pessoas que viajariam da estação 1 até a estação 2, 10 pessoas que viajariam da estação 1 até a estação 3, e assim por diante.

Além dos *inputs* mencionados, é necessário considerar as principais restrições e obje-

Tabela 4: Matriz OD para o exemplo

	1	**2**	**3**	**4**	**5**
1	0	5	10	10	20
2	2	0	5	5	10
3	3	0	0	5	5
4	2	2	1	0	5
5	5	2	2	1	0

Fonte: Autor

tivos para a modelagem da rede de transportes. Em Guihaire & Hao (2008), as restrições e objetivos são descritos conjuntamente, pois, dependendo do projeto sendo desenvolvido, eles podem se intercambiar. Portanto, os seguintes fatores devem ser considerados:

- **Contexto histórico:** É fundamental considerar a rede de transportes já existente, pois ela impacta diretamente na modelagem de novas redes. Além disso, podem existir razões políticas que influenciem a decisão de modificar ou desativar as linhas antigas.

- **Cobertura de área:** Calcula-se a porcentagem da demanda estimada que será atendida pelo transporte público. Em geral, considera-se que as pessoas que moram a até 500 metros de uma estação de ônibus fazem parte dessa porcentagem. Usualmente, esse cálculo leva em conta fatores como o comprimento da linha, a densidade populacional, o espaçamento entre os pontos de ônibus e a distância entre as linhas. (Murray, 2003; Benn, 1995).

- **Eficiência das linhas e viagens:** Do ponto de vista dos usuários, uma rede de ônibus deve permitir que eles se desloquem da origem ao destino de forma mais direta possível, com a maior eficiência de linha e a menor distância a pé até as estações iniciais e finais. Há diversas definições que permitem avaliar esse fator de acordo com a literatura. Segundo Benn (1995), que nomeou este fator como *Route Directness*, uma abordagem matemática deve ser usada para avaliar o desvio da linha em relação a um caminho linear, considerando um ou mais dos seguintes fatores: tempo de viagem adicional para uma viagem de ônibus de ida; tempo adicional de viagem considerando a mesma viagem feita de carro; limitações de quilometragem que permitem um desvio máximo por linha; aumento do tempo limite no tempo médio de viagem por passageiro; limite para o número total de desvios do percurso; os desvios de percurso não devem reduzir a produtividade média da rota, ou o desvio

deve ter uma classificação de produtividade superior para a linha como um todo. Já em Martínez (2017), o conceito de eficiência das linhas é tratado como um dos objetivos a serem considerados, calculando-se o desvio médio das linhas em relação ao tempo ideal de viagem, ponderado pelo número de usuários.

- **Satisfação do usuário:** É fundamental que a rede de ônibus satisfaça o usuário. Em geral, se uma viagem requer mais de duas transferências de linha, assume-se que o usuário buscará outros meios de locomoção. Quando a eficiência das linhas é insuficiente, ou quando as estações são muito distantes da origem e do destino dos passageiros, considera-se que a comunidade está insatisfeita.

- **Tamanho e quantidade de linhas:** É de interesse do operador minimizar o tamanho e a quantidade total das linhas, reduzindo o número de pessoas e veículos necessários para providenciar o serviço de transportes para a região. No entanto, as linhas não podem ser nem muito curtas, nem muito longas, devido a questões de lucratividade.

- **Objetivos específicos do operador:** o operador pode ter particularidades que precisam ser satisfeitas no momento de desenvolvimento da rede de ônibus. Por exemplo, a empresa pode requerer que a rede possua algum formato específico, como radial, retangular ou triangular (Van Nes, 2002).

2.3 Definição das frequências

O objetivo desta etapa é definir as frequências das linhas da rede de ônibus para um determinado período. Nota-se que, ao definir a frequência, outras variáveis da rede também são estabelecidas. O inverso da frequência é o tempo de *headway*, que corresponde ao intervalo entre as partidas de ônibus em uma linha. Com o tempo de *headway* definido, calcula-se o número de viagens por hora de uma linha. Para esta componente, segundo Guihaire & Hao (2008), o principal input é a rede de linhas do transporte público. Além das linhas, é necessário conhecer a demanda dos usuários, a partir das matrizes de Origem-Destino. Com as demandas e as linhas, é possível definir as frequências de cada uma delas, de modo a atender da melhor forma a comunidade. É importante notar que as demandas variam em função de algumas variáveis:

- **Dias da semana:** a demanda difere significativamente entre os dias úteis (segunda a sexta-feira) e os fins de semana (sábado e domingo).

- **Horário do dia:** a demanda varia conforme os horários do dia, com destaque aos horários de pico, que apresentam uma demanda muito maior. Nota-se que também há diferença entre os horários de pico da manhã e da tarde, pois o sentido do trajeto dos usuários é oposto em cada período.
- **Período do ano:** alguns períodos do ano, como as férias escolares, podem afetar diretamente a demanda de usuários.

Portanto, observa-se que a matriz OD varia consideravelmente ao longo do tempo. Idealmente, as agências de transporte deveriam fornecer esse detalhamento de informações para um melhor dimensionamento das linhas e frequências de ônibus. No entanto, a coleta desses dados pode ser bastante custosa e trabalhosa para as empresas.

Por fim, é importante considerar como input o número de veículos disponíveis para as linhas, pois as frequências dependem do tamanho da frota e da capacidade dos ônibus. Para redes de ônibus com frotas heterogêneas, ou seja, compostas por ônibus de diferentes capacidades, é necessário conhecer a capacidade de cada tipo de ônibus e a disponibilidade de cada um para cada linha.

Em relação aos principais objetivos e restrições para a definição das frequências, os seguintes fatores devem ser considerados:

- **Satisfação do usuário:** as frequências das linhas devem ser coerentes à demanda dos usuários, de forma a evitar tempos de *headway* muito longos ou superlotação. A frequência bem dimensionada garante tempos de espera e de transferência eficientes.
- **Número de viagens por período:** neste fator, os interesses dos usuários e do operador entram em conflito. Do ponto de vista do usuário, quanto maior o número de viagens disponíveis por linha em cada período, melhor. Já o operador deseja minimizar o uso de recursos, reduzindo o número de viagens por período para maximizar seus lucros. Portanto, é fundamental considerar ambos os interesses na modelagem do problema.
- **Limite de *headway*:** podem existir limites máximos ou mínimos de tempo de *headwa* impostos ao operador para determinadas linhas. Em Martínez (2017), a frequência das linhas não pode ultrapassar o limite de veículos que cada estação suporta, restringindo, portanto, o tempo de *headway*.
- **Contexto histórico:** conforme já mencionado, podem existir motivos políticos que afetem a definição da frequência.

2.4 *Transit network design and frequencies setting problem* - TNDFSP

O TNDFSP (*Transit Network Design and Frequencies Setting Problem*) tem como objetivo encontrar um conjunto de linhas e suas respectivas frequências, de modo a atender as demandas de viagem dos usuários em uma área urbana, criando um sistema de transporte público eficiente. Esse problema é aplicado em redes de transporte público, como ônibus, BRTs (*Bus Rapid Transit*) e bondes (Oliveira & Barbieri, 2015).

De acordo com Oliveira & Barbieri (2015), os problemas de *design* de redes de transporte são tratados como multiobjetivos, pois os interesses dos usuários e operador são conflitantes, isto é, a otimização dos objetivos dos usuários normalmente leva a uma piora nos objetivos dos operadores, e vice-versa. Portanto, a formulação do problema deve considerar os objetivos de ambos os grupos:

- **Usuários:** o objetivo é viajar da origem ao destino no menor tempo possível, o que implica minimizar o tempo de viagem dentro do veículo, o tempo de espera nas estações e o número de transferências. Além disso, a redução de outras inconveniências, como o tempo de caminhada até as estações, também é do interesse dos passageiros.
- **Operador:** o objetivo é minimizar os custos de operação das linhas, aumentando consequentemente os lucros. Para isso, o operador busca reduzir a frota total necessária, que é relacionada às frequências e tempos de viagem das linhas.

Para o sucesso dos sistemas de transporte público, é fundamental determinar uma solução eficiente tanto para os usuários quanto para os operadores. No entanto, problemas classificados como TNDFSP são NP-hard, apresentando diversas variáveis de decisão e restrições a serem consideradas, o que os torna desafiadores e demorados de resolver.

Historicamente, conforme mostra Fan et al. (2009), os planejadores das linhas de ônibus têm se apoiado principalmente em experiências passadas, diretrizes simples, conhecimento local e procedimentos ad hoc (processos criados especificamente para resolver um problema de maneira imediata e sem planejamento prévio). No entanto, com o avanço da tecnologia e dos recursos computacionais, diversos estudos recomendam o uso de novas ferramentas e metodologias para o planejamento e avaliação das redes de transporte público.

A partir dos estudos de Cancela et al. (2015) e Farahani et al. (2013), os métodos

de solução para um TNDFSP são classificados em três categorias: (1) métodos exatos ou matemáticos, (2) heurísticas e (3) meta-heurísticas. A seguir, apresenta-se uma revisão literária para cada categoria.

2.4.1 Métodos exatos

Há um número limitado de estudos que aplicam métodos exatos para solucionar um problema classificado como TNDFSP (Farahani et al., 2013). Em geral, esses métodos conseguem encontrar soluções apenas para redes de transporte de pequena escala. Isso ocorre porque a busca de soluções exatas para o TNDFSP pode ser extremamente desafiadora, dada a complexidade de resolução deste tipo de problema.

Em Bussieck (1998), o autor desenvolve uma abordagem de programação matemática para o problema de planejamento de linhas. A pesquisa aborda tanto os aspectos matemáticos quanto práticos da otimização de linhas. Bussieck utiliza programação linear inteira e métodos de relaxação, como *branch-and-bound*, para obter soluções viáveis. A tese destaca a importância dos resultados teóricos na melhoria prática dos algoritmos aplicados a dados reais de grande escala.

No trabalho de Wan & Lo (2003), os autores formularam o problema da rede de transportes com múltiplas linhas como um problema de programação inteira mista. Utilizaram técnicas de linearização para as restrições, transformando-o em um problema de programação linear inteira mista, resolvido através do solver MIP no software CPLEX. A formulação permite determinar múltiplas linhas de transporte simultaneamente, oferecendo serviços diretos para todas as demandas de viagem. Os autores destacam que essa abordagem é um bom ponto de partida para pesquisas futuras em planejamento de sistemas de transporte e enfatizam a importância de heurísticas e algoritmos para problemas de grande escala.

Portanto, observa-se que tanto no trabalho de Bussieck (1998) quanto no de Wan & Lo (2003), embora os autores utilizem métodos exatos para resolver problemas de TNDFSP, ambos recomendam a aplicação de técnicas mais sofisticadas para problemas de maior complexidade. Dessa forma, embora esses métodos contribuam significativamente para a melhoria dos algoritmos e técnicas de resolução dos TNDFSPs, nota-se que existem abordagens mais eficientes atualmente.

Na Tabela 5, apresentam-se os estudos de TNDFSP que utilizaram métodos exatos.

Tabela 5: Estudos de TNDFSP que utilizaram métodos exatos

Referência	**Restrições**	**Objetivos**	**Método de Solução**
Hasselström (1979, 1981)	Orçamento	Minimizar o número de transferências Maximizar o número de passageiros	Método exato
Van Nes et al. (1988)	Tamanho da frota	Maximizar a satisfação dos usuários	Método exato + heurística
Bussieck (1998)	Número de veículos Frequências Capacidade das linhas e veículos	Maximizar o número de passageiros com viagens sem transferência Minimizar os custos do operador	Método exato
Wan & Lo (2003)	Frequência de serviço das linhas Capacidade dos ônibus	Minimizar os custos do operador	MIP Solver no software CPLEX

Fonte: Adaptado de Farahani et al. (2013)

2.4.2 Métodos heurísticos

A heurística é fundamental na resolução de problemas TNDFSP devido à sua capacidade de lidar com a complexidade combinatória e a escala desses problemas. Em vez de buscar soluções exatas, que podem ser computacionalmente inviáveis, as heurísticas oferecem métodos eficientes para explorar grandes espaços de solução e encontrar resultados viáveis e de alta qualidade.

De acordo com Zanakis et al. (1989), heurísticas construtivas são aquelas que constroem soluções incrementais, adicionando componentes individuais, um a um, até que uma solução viável seja encontrada. Em problemas de transporte, esses componentes podem incluir estações de linhas, veículos alocados às linhas, entre outros. Um exemplo clássico mencionado pelo autor é a heurística do vizinho mais próximo na resolução do problema do caixeiro-viajante. No contexto dos problemas de transporte, o método do canto noroeste (*north-west corner rule*) e o método de aproximação de Vogel, que visam otimizar os custos de transporte, são exemplos de heurísticas construtivas. A análise de 442 artigos por Zanakis et al. (1989) revelou que as heurísticas construtivas são as mais utilizadas entre os métodos heurísticos.

Rayward-Smith et al. (1996) destacam diversas vantagens significativas na utilização

de métodos heurísticos:

- As heurísticas são notavelmente mais fáceis de implementar e proporcionam soluções quase-ótimas de forma rápida.
- São flexíveis e altamente adaptáveis, permitindo modificações eficientes em resposta a mudanças nas condições do problema, o que é essencial em ambientes dinâmicos.
- As heurísticas conseguem incorporar de maneira mais eficiente as complexidades dos problemas do mundo real, abordando aspectos que poderiam ser ignorados por outros métodos de solução.

O primeiro algoritmo heurístico desenvolvido para um problema de planejamento da rede de transportes foi proposto por Lampkin e Saalmans (1967), através de um estudo de caso de uma empresa de ônibus municipal. Este estudo focou na reorganização das linhas, frequências, cronogramas e na minimização do número de ônibus necessários. Os autores iniciaram com um esqueleto básico para cada linha de ônibus, adicionando as estações uma a uma a esse esqueleto inicial. Após a construção das linhas, as frequências foram atribuídas a cada uma delas, com o objetivo de maximizar os níveis de serviço para os usuários (Oliveira & Barbieri, 2015). O modelo resultante da reorganização foi implementado na rede de transportes municipal, com a expectativa de manter o nível de serviço anterior, mas com um custo operacional significativamente reduzido. Esse resultado evidenciou a eficiência dos métodos heurísticos na resolução de TNDFSPs.

No trabalho de Ceder & Wilson (1986), os autores desenvolveram uma heurística utilizando uma abordagem de dois níveis. O nível I considera exclusivamente o ponto de vista dos usuários, enquanto o nível II abrange tanto os impactos para os passageiros quanto para os operadores. Essa abordagem abrangeu cinco objetivos principais:

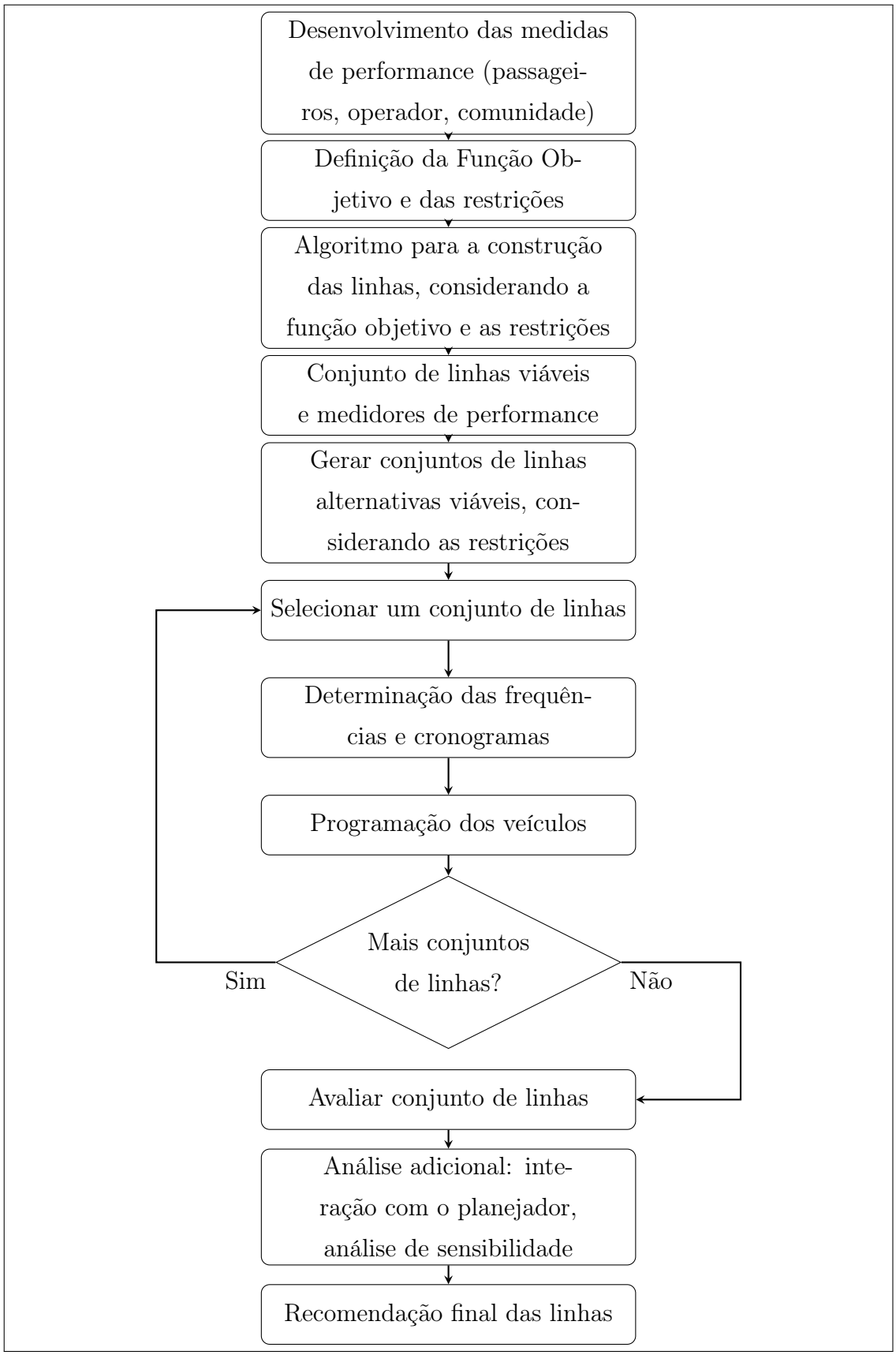
1. Desenvolver um algoritmo que forneça uma solução otimizada para o planejamento da rede de transportes, garantindo que as linhas transportem passageiros de qualquer par origem-destino, respeitando restrições de viagem, tempos de transferência e espera.
2. Estabelecer medidas de desempenho considerando as perspectivas dos passageiros, operadores e da comunidade em geral.
3. Integrar componentes de cronograma das linhas e veículos aos procedimentos de design da rede de transportes.

4. Implementar uma análise de sensibilidade para avaliar a robustez das soluções frente a possíveis mudanças na demanda e nas restrições.

5. Criar uma interface interativa pessoa-máquina que permita a modificação das restrições durante o processo de planejamento do sistema de transportes.

A Figura 7 ilustra a metodologia heurística para o nível II utilizada por Ceder & Wilson.

Figura 7: Heurística para a resolução de TNDFSP



Fonte: adaptado de Ceder & Wilson (1986)

A partir da metodologia heurística desenvolvida, Ceder & Wilson conseguem entregar soluções viáveis e de boa qualidade para o TNDFSP. Ao considerar tanto as perspectivas dos passageiros quanto dos operadores, a abordagem permite a construção de linhas eficientes que atendem às restrições operacionais e melhoram a satisfação do usuário. Esta metodologia se destaca por sua capacidade de gerar e avaliar múltiplas alternativas, promovendo uma e seleção criteriosa das melhores opções para implementação prática.

Na Tabela 6, apresentam-se os estudos de TNDFSP que utilizaram apenas métodos heurísticos de solução.

Tabela 6: Estudos de TNDFSP que utilizaram métodos heurísticos

Referência	**Restrições**	**Objetivos**	**Método de Solução**
Lampkin & Saalmans (1967)	Tamanho da frota	Maximizar o número de passageiros com viagens sem transferência Minimizar o tempo total de viagem	Heurístico
Silman et al. (1974)	Orçamento	Minimizar o tamanho da frota Minimizar o tempo de viagem Minimizar superlotação	Heurístico
Dubois et al. (1979)	Orçamento	Minimizar o tempo total de viagem	Heurístico
Ceder & Wilson (1986)	Frequência mínima Tamanho da frota Tamanho da rota	Minimizar o excesso dos tempos de viagem, transferência e espera Minimizar os custos com veículos	Heurístico

Referência	**Restrições**	**Objetivos**	**Método de Solução**
Shih & Mahmassani (1994) Shih et al. (1998)	Não há	Minimizar o tempo de viagem Maximizar a satisfação do usuário Minimizar o tamanho da frota	Heurístico
Baaj & Mahmassani (1995)	Tempo de headway Tamanho da frota Capacidade	Maximizar o número de viagens sem transferência Minimizar os tempos de espera e transferência	Heurístico
Carrese & Gori (2004)	Satisfação dos usuários Tamanho da rota Número de transferências Tempo total de viagem Tamanho da frota	Minimizar o tempo em excesso comparado ao caminho mínimo Minimizar os custos do operador	Heurístico
Fusco et al. (2002)	Nível de serviço Satisfação dos usuários configuração das linhas Frequência Tamanho da rota	Minimizar o custo total	Heurístico

Referência	**Restrições**	**Objetivos**	**Método de Solução**
Ceder (2003)	Tamanho da rota Desvio em relação ao caminho mínimo	Minimizar os custos do usuário e do operador Minimizar o tamanho da frota	Heurístico

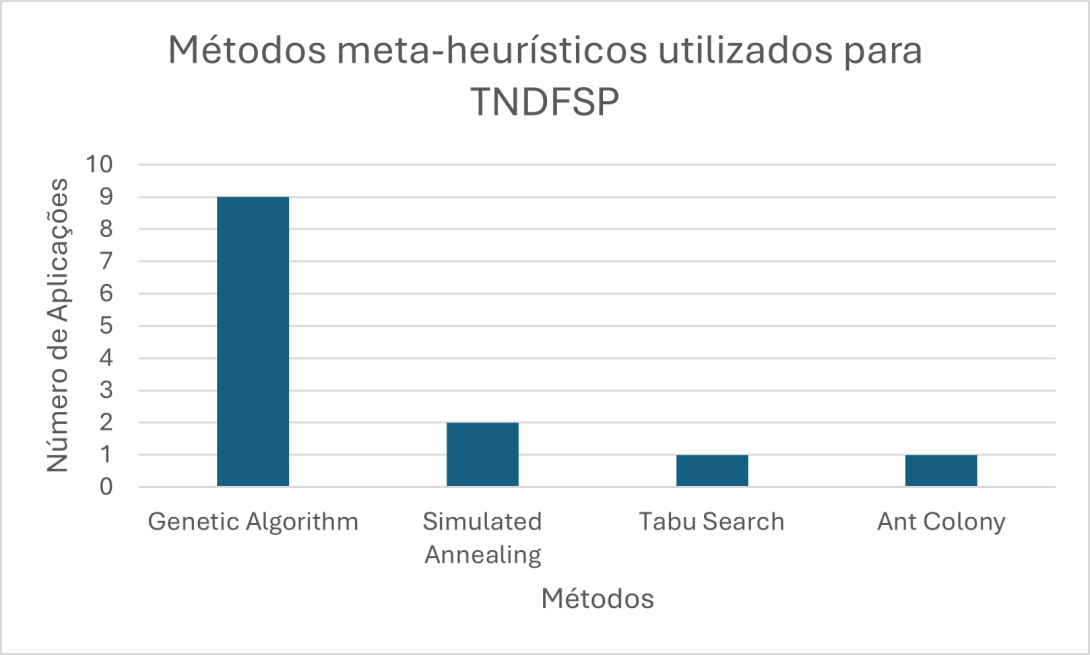
Fonte: Adaptado de Farahani et al. (2013)

2.4.3 Meta-heurísticas

A natureza *NP-hard* do TNDFSP exige o desenvolvimento de métodos inovadores para obter soluções quase ótimas de maneira eficiente para problemas de tamanho prático. Por essa razão, heurísticas e meta-heurísticas são frequentemente utilizadas, e às vezes, estratégias de computação paralela são incorporadas. Normalmente, são utilizados métodos para gerar rotas candidatas usando algoritmos heurísticos e definir a configuração das rotas utilizando heurísticas ou meta-heurísticas. A geração de rotas candidatas é geralmente realizada usando algoritmos baseados no caminho mais curto, que adicionam estações de ônibus à rota até que alguma das restrições definidas, como o comprimento da linha ou o tempo de viagem, sejam violadas. A configuração das rotas geralmente é definida pela seleção e melhoria das rotas geradas, acompanhada pela determinação da frequência das linhas, que pode ser feita sequencialmente ou simultaneamente à definição das linhas, dependendo do método. A atribuição de valores de demanda de passageiros à rede é realizada nessa etapa (Farahani et al., 2013).

Apesar de existirem diversos métodos meta-heurísticos na literatura para a resolução de TNDFSPs, há um pequeno número de abordagens que são utilizadas com muito mais frequência do que as outras. Os métodos meta-heurísticos podem ser divididos em clássicos e não-clássicos. Entre os métodos clássicos, *Genetic Algorithm* (GA) e *Simulated Annealing* (SA) são os mais prevalentes, com o GA tendo o maior número de aplicações para o TNDFSP (Farahani et al., 2013). Na Figura 8, observa-se um resumo das meta-heurísticas utilizadas na resolução de TNDFSPs até 2013.

Figura 8: Métodos meta-heurísticos utilizados para TNDFSP



Fonte: Adaptado de Farahani et al. (2013)

Na próxima seção, será estudado um Algoritmo Genético (GA) para a modelagem e solução do problema de definição de linhas e frequências do BRT na Colômbia.

2.4.4 Modelagem e solução para o BRT da Colômbia

Nesta seção, será analisado o trabalho intitulado "Model and Solution Method to a Simultaneous Route Design and Frequency Setting Problem for a Bus Rapid Transit System in Colombia", de Fabián Martínez, María Gulnara Baldoquín e Antonio Mauttone. Este artigo é destacado na revisão bibliográfica por fornecer a fundamentação teórica e metodológica para várias das ideias desenvolvidas neste trabalho. O estudo aborda uma resolução heurística por meio de um Algoritmo Genético para o planejamento simultâneo das linhas e frequências de um dos principais corredores do BRT (*Bus Rapid Transit*) na Colômbia. O Artigo baseia-se na modelagem feita em Szeto & Wu (2011), com as devidas adaptações ao caso do BRT.

No artigo, os autores desenvolvem uma formulação matemática para o problema, considerando uma função objetivo com três objetivos, dois relacionados ao usuário e um relacionado ao operador. São eles:

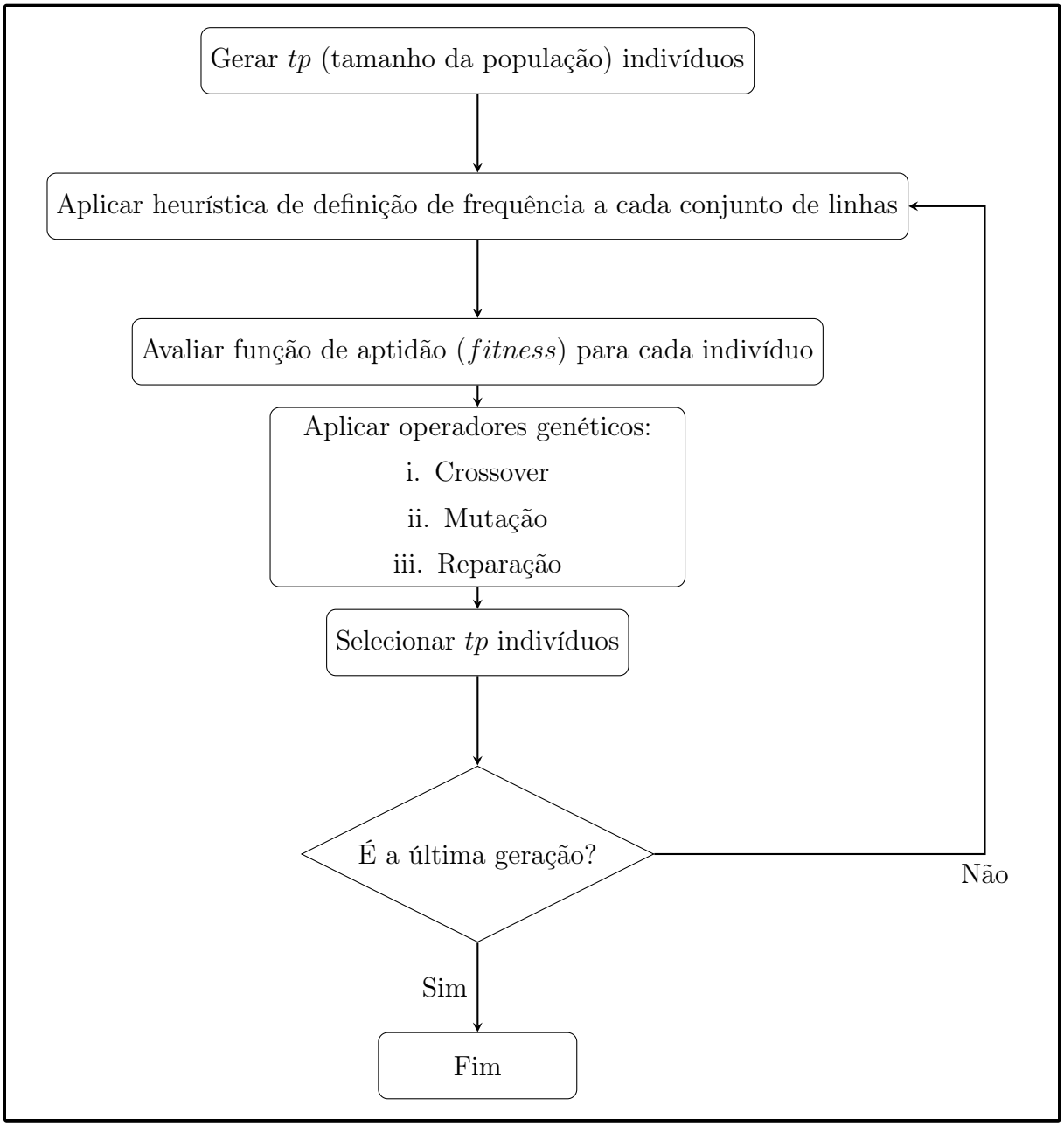
- Tempo total esperado de viagem, considerando os tempos de parada, de espera, de

veículo em movimento e de espera para transferência;

- O desvio médio das linhas em proporção ao tempo ideal de viagem, ponderado pelo número de usuários de cada viagem;
- O número de veículos necessário para operar as linhas.

O Algoritmo Genético utilizado para a solução do problema é representado na Figura 9.

Figura 9: Estrutura do Algoritmo Genético proposto.



Fonte: Adaptado de Martínez et al. (2017)

Portanto, no modelo proposto, uma população inicial de conjuntos de linhas é gerada por meio de um algoritmo, respeitando as restrições do problema. Com essa população inicial, aplica-se um algoritmo heurístico para definir as frequências de cada conjunto de linhas. Em seguida, utiliza-se a função de aptidão para avaliar os indivíduos gerados. Após essa etapa, aplicam-se os operadores genéticos de *Crossover*, Mutação e Reparação. Por fim, seleciona-se um número de indivíduos igual ao tamanho da população. Avalia-se

então se esta é a última geração; caso seja, o algoritmo é encerrado. Após a aplicação do GA, os autores realizam um controle de diversidade da população, conforme proposto por Szeto & Wu (2011). Essa abordagem garante a convergência do Algoritmo Genético, o que é altamente benéfico para a definição da melhor configuração de linhas e frequências.

O método de solução e a modelagem proposta foram validados utilizando dados reais de um dos principais corredores de BRT da Colômbia. Os resultados demonstraram uma melhoria significativa no nível de serviço dos usuários, validada pelo operador do corredor (Martínez et al., 2017).

3 FORMULAÇÃO MATEMÁTICA E MÉTODO DE SOLUÇÃO PROPOSTOS

O capítulo aborda a formulação matemática do problema de otimização do BRT-ABC e os métodos de solução aplicados. São definidos os elementos principais, incluindo variáveis, restrições e a função objetivo que balanceia o tempo de viagem dos usuários e os custos operacionais. Em seguida, são detalhados os algoritmos heurísticos utilizados para determinar as configurações ideais de linhas e frequências de ônibus, além de um exemplo prático que ilustra a aplicação desses métodos. Por fim, a estrutura de implementação dos algoritmos é descrita, apresentando o processo de seleção da melhor solução para o sistema de transporte.

A etapa de formulação é fundamental para a modelagem e consideração do problema de otimização do BRT-ABC. Ela estabelece os objetivos e restrições que o modelo deve seguir, além de identificar as variáveis de decisão. Uma formulação bem delineada assegura que todos os aspectos relevantes do problema, como a configuração das linhas e os tempos de viagem, sejam representados de maneira precisa. Sem uma formulação rigorosa, as soluções propostas podem falhar em capturar a complexidade e as especificidades do sistema de transporte, comprometendo sua eficácia. Portanto, a formulação e a modelagem matemática são cruciais para a obtenção de resultados que sejam aplicáveis e benéficos na prática.

O BRT-ABC possui um traçado que passará por 20 estações e 3 terminais, conforme Figura 10.

Figura 10: Terminais e estações do BRT-ABC

Fonte: Estudo Conceitual e Funcional – BRT-ABC, Systra

Para a formulação, considera-se o Terminal São Bernardo como ponto Sul e o Terminal Sacomã como ponto Norte. Será utilizada a formulação proposta por Martínez et al. (2017), com as devidas adaptações para este estudo de caso. A abordagem envolve uma função multiobjetivo, na qual dois objetivos refletem os interesses dos usuários e um objetivo reflete os interesses do operador.

A aplicação de uma função multiobjetivo ao problema do BRT é extremamente vantajosa, pois possibilita a avaliação da função objetivo a partir dos interesses de diferentes partes envolvidas. Em Damm (2016), o autor ressalta que a consideração de objetivos distintos e conflitantes gera uma variedade de soluções para o problema, proporcionando ao tomador de decisão um papel mais proativo na escolha da solução mais adequada.

De acordo com Branke et al. (2008), um problema de otimização multiobjetivo passa por três fases principais: modelagem, otimização e tomada de decisão. Os autores argumentam que converter um problema de otimização multiobjetivo em um problema simplista mono-objetivo antecipa a fase de tomada de decisão para antes da otimização, ou seja, antes de se conhecerem as alternativas de solução. Essa abordagem pode comprometer a obtenção de um resultado ótimo, já que a solução final pode não ser a preferida pelo decisor entre o conjunto de soluções que a otimização multiobjetivo geraria.

Branke et al. (2008) destacam que a modelagem multiobjetivo posiciona a fase de tomada de decisão após a otimização ou a intercala com esta. Esse método permite que o usuário desenvolva uma compreensão mais profunda do problema e das alternativas disponíveis, resultando em uma tomada de decisão mais consciente e eficaz.

Para a modelagem, as seguintes hipóteses são consideradas:

- Trata-se de um, e apenas um, corredor de BRT;
- O corredor possui duas faixas, de sentidos contrários. Cada linha para nas mesmas estações, independentemente do sentido. Há um subconjunto de estações que são o começo e fim da linha, as quais permitem que os veículos façam um retorno em direção à estação de onde vieram. As estações que não fazem parte deste conjunto, não permitem retorno em direção de onde o ônibus veio;
- Não há restrições em relação às estações em que os usuários podem transferir de linha. Se duas linhas param na mesma estação, a transferência é permitida;
- O tempo total de viagem é composto por 3 fatores: tempo de espera na estação, tempo no veículo e tempo para transferência. O tempo no veículo é dividido em dois

componentes: o tempo do ônibus em movimento e os tempos de parada em cada estação. O tempo de transferência corresponde ao tempo até a chegada do ônibus da próxima linha;

- Deve existir pelo menos uma linha que leve o usuário de uma estação até a outra, quaisquer sejam as estações, sem necessidade de transferência;
- Não se considera a situação que o usuário pode perder um veículo por ele estar cheio;
- Define-se uma frequência mínima para as linhas operando no corredor;
- Define-se um número máximo de veículos operando no BRT e um número máximo de linhas disponíveis;
- Ao realizar uma transferência, o usuário opta **sempre** pela estação de transferência mais próxima da estação final do seu trajeto;
- Ao realizar uma transferência, o usuário escolhe **sempre** a linha que o levará mais rapidamente da estação de transferência até a estação final do seu trajeto.

3.1 Notação e modelagem

3.1.1 Conjuntos

- U: conjunto representando estações do corredor $U \subseteq \mathbb{N}\{0\}$;
- V: conjunto das estações (terminais) que permitem retorno sul-sul (início) $V \subseteq U$;
- Y: conjunto das estações (terminais) que permitem retorno norte-norte (final) $Y \subseteq U$;
- i, j, k, e, p: índices para estações $\{1, 2, 3, \ldots, 23\}$;
- n: índice para linhas $\{1, 2, \ldots, R_{\max}\}$.

3.1.2 Parâmetros

- r_{ij}: distância entre as estações i e j (km);
- vm_k: velocidade média para as linhas com k estações (km/hora), $2 \leq k \leq 23$;

- c_{ij}: tempo médio no veículo se deslocando de i para j (horas);
- s_i: tempo médio de parada em i (horas);
- d_{ie}: número de passageiros viajando de i para e (passageiros/hora);
- $f_{\min}$: frequência mínima para linhas operando no corredor (número de veículos/hora);
- W: tamanho da frota disponível no corredor;
- $R_{\max}$: número máximo de linhas no corredor;
- B_1, B_2, B_3: fatores de ponderação na função objetivo para: tempo total de viagem dos usuários (B_1), desvio máximo de viagem (B_2) e o número de veículos atribuídos às linhas (B_3).

3.1.3 Variáveis de Decisão

- X_{ij}^n: 1 se a linha n chega à estação $j \neq i$ imediatamente após a estação i, e 0 caso contrário. Indica a trajetória da linha n de i para j em ambas as direções: quando vai de sul para norte ($i < j$) e quando vai de norte para sul ($j < i$);
- X_{i0}^n: 1 se a linha n começa na estação i e 0 caso contrário;
- X_{0j}^n: 1 se a linha n termina na estação j e 0 caso contrário;
- X_{00}^n: 1 se a linha n está desabilitada e 0 caso contrário;
- RT_{ij}^n: 1 se a linha n permite viagem sem transferências das estações i até j, consecutivas ou não na linha n; 0 caso contrário;
- W_{ij}^n: 1 se a linha n tem uma parada em i, não importando que não pare em j, e tem uma parada em comum entre i e j com pelo menos outra linha que tem uma parada na estação j, 0 caso contrário;
- e_n: número de estações da linha n;
- v_n: velocidade média da linha n (km/hora). É igual à velocidade média vm_k, na qual k é o número de estações da linha n;
- C_{ij}^n: tempo no veículo se deslocando de i para j (horas) para a linha n;
- f_n: frequência da linha n (veículos/hora);

- T_n: tempo de ciclo da linha n em horas (tempo do veículo em movimento mais o tempo de cada parada em toda a trajetória da linha n de sul a norte e norte a sul);
- V_n: número de veículos necessários para operar a linha;
- O_{ij}^n: 1 se a linha n possui uma trajetória entre i e j com ou sem transferência. 0 caso contrário;
- T_{ie}: tempo de viagem esperado de i para e (horas);
- t_{ij}^n: tempo de viagem de i para j na linha n, sendo i e j dois pontos de parada na linha (horas);
- T_{ij}^n: tempo de viagem esperado de i para j usando n como linha inicial, onde i é um ponto de parada da linha n; j pode ou não ser um ponto de parada na linha n (horas);
- Z_{ij}^n: 1 se n é a linha que fornece a trajetória mais rápida entre i e j sem transferência e 0 caso contrário;
- G_{ij}: menor tempo de viagem entre i e j, considerando todas as linhas que passam por i e j;
- F_{ij}: frequência da linha que possui o menor tempo de viagem sem transferência entre i e j.

3.2 Função Objetivo e Restrições

A função objetivo a ser minimizada é dada por:

$$\min z = B_1 \sum_{i \in U} \sum_{e \in U} d_{ie} T_{ie} + B_2 \frac{\sum_{i \in U} \sum_{e \in U} d_{ie} \frac{T_{ie}}{c_{ie}}}{\sum_{i \in U} \sum_{e \in U} d_{ie}} + B_3 \sum_{n=1}^{R_{\max}} V_n \tag{1}$$

s.t.:

$$\sum_{i \in V \cup \{0\}} X_{i0}^n = 1 \quad \forall n \tag{2}$$

$$\sum_{j \in Y \cup \{0\}} X_{0j}^n = 1 \quad \forall n \tag{3}$$

$$X_{ij}^n - X_{ji}^n = 0 \quad \forall i, j \in U, \forall n \tag{4}$$

$$\sum_{i \in V \cup \{0\}, i<j} X_{i0}^n \geq X_{0j}^n \quad \forall j \in Y, \forall n \tag{5}$$

$$\sum_{j \in U, j>i} X_{ij}^n \leq 1 \quad \forall i \in U \cup \{0\}, \forall n \tag{6a}$$

$$\sum_{i \in U, i>j} X_{ij}^n \leq 1 \quad \forall i \in U \cup \{0\}, \forall n \tag{6b}$$

$$f_n \geq f_{\min}(1 - X_{00}^n) \quad \forall n \tag{7}$$

$$e_n = 1 + \sum_{i \in U, i<j} \sum_{j \in U} X_{ij}^n \tag{8}$$

$$v_n = vm_{e_n} \quad \forall n \tag{9}$$

$$C_{ij}^n = \frac{r_{ij}}{v_n} \quad \forall i, j \in U, i \neq j, \forall n \tag{10}$$

$$T_n = \sum_{i \in U, i \neq j} \sum_{j \in U} X_{ij}^n (C_{ij}^n + s_i) + \sum_{i \in U} s_i (X_{i0}^n + X_{0i}^n) \quad \forall n \tag{11}$$

$$V_n = T_n f_n (1 - X_{00}^n) \quad \forall n \tag{12}$$

$$\sum_{n=1}^{R_{\max}} V_n \leq W \tag{13}$$

$$RT_{ik}^n = X_{ik}^n + \sum_{j \in U, k>j>i} X_{ij}^n RT_{jk}^n \quad \forall i, k \in U, k > i, \forall n \tag{14a}$$

$$RT_{ik}^n = X_{ik}^n + \sum_{j \in U, i>j>k} X_{ij}^n RT_{jk}^n \quad \forall i, k \in U, k < i, \forall n \tag{14b}$$

$$\sum_{n=1}^{R_{\max}} RT_{ij}^n (1 - X_{00}^n) \geq 1 \quad \forall i, j \in U, i \neq j \tag{15}$$

$$t_{ik}^n = X_{ik}^n(C_{ik}^n + s_i) + \sum_{j\in U, i<j<k} X_{ij}^n RT_{jk}^n(t_{jk}^n + C_{ij}^n + s_i) \quad \forall i,k \in U, k > i, \forall n \tag{16a}$$

$$t_{ik}^n = X_{ik}^n(C_{ik}^n + s_i) + \sum_{j\in U, i>j>k} X_{ij}^n RT_{jk}^n(t_{jk}^n + C_{ij}^n + s_i) \quad \forall i,k \in U, k < i, \forall n \tag{16b}$$

$$W_{ij}^n = 1 - \prod_{\forall m, m\neq n} \prod_{k\in U, i<k<j} \left(1 - RT_{ik}^n RT_{kj}^m\right) \quad \forall i,j \in U, j > i, \forall n \tag{17a}$$

$$W_{ij}^n = 1 - \prod_{\forall m, m\neq n} \prod_{k\in U, i>k>j} \left(1 - RT_{ik}^n RT_{kj}^m\right) \quad \forall i,j \in U, j < i, \forall n \tag{17b}$$

$$O_{ij}^n = \max(RT_{ij}^n, W_{ij}^n) \quad \forall i,j \in U, j \neq i, \forall n \tag{18}$$

$$\sum_{n=1}^{R_{\max}} Z_{ij}^n = 1 \quad \forall i,j \in U, i \neq j \tag{19}$$

$$G_{ij} = \sum_{n=1}^{R_{\max}} Z_{ij}^n t_{ij}^n \quad \forall i,j \in U, i \neq j, \forall n \tag{20}$$

$$F_{ij} = \sum_{n=1}^{R_{\max}} Z_{ij}^n f_n \quad \forall i,j \in U, i \neq j, \forall n \tag{21}$$

$$\begin{aligned} T_{ij}^n = {} & \frac{1}{f_n} + RT_{ij}^n t_{ij}^n \\ & + \left(1 - RT_{ij}^n\right)\left(\sum_{i\leq e<j}\left(\left(W_{ej}^n - \sum_{k=1}^{j-e-1} X_{ee+k}^n W_{e+kj}^n\right)\left(\sum_{e<p<j} X_{ep}^n\left(t_{ip}^n + \frac{1}{F_{pj}} + G_{pj}\right)\right)\right)\right) \\ & + \left(1 - RT_{ij}^n\right)\left(\sum_{j<e\leq i}\left(\left(W_{ej}^n - \sum_{k=1}^{e-j+1} X_{ee-k}^n W_{e-kj}^n\right)\left(\sum_{j<p<e} X_{ep}^n\left(t_{ip}^n + \frac{1}{F_{pj}} + G_{pj}\right)\right)\right)\right) \\ & \forall i,j \in U, j \neq i, \forall n \end{aligned} \tag{22}$$

$$T_{ie} = \sum_{n=1}^{R_{\max}} \frac{f_n O_{ie}^n (1 - X_{00}^n)}{\sum_{m=1}^{R_{\max}} f_m O_{ie}^m (1 - X_{00}^m)} T_{ie}^n \quad \forall i,e \in U, i \neq e \tag{23}$$

$$B1, B2, B3 \geq 0 \tag{24}$$

$$B1 + B2 + B3 = 1 \tag{25}$$

$$X_{ij}^{n} \in \{0,1\} \quad \forall i,j \in U \cup \{0\}, \forall n \tag{26}$$

$$RT_{ij}^{n}, Z_{ij}^{n} \in \{0,1\} \quad \forall i,j \in U, i \neq j, \forall n \tag{27}$$

Para a função objetivo (1) consideram-se três objetivos ponderados por B_1, B_2 e B_3:

1. $\sum_{i \in U} \sum_{e \in U} d_{ie} T_{ie}$: representa o tempo total de viagem. Observa-se que pares de estações (i, e) com maior demanda serão priorizados pela função objetivo, resultando em uma melhor otimização do seu tempo de viagem, enquanto viagens entre pares com menor demanda serão menos favorecidas. A unidade para este objetivo é passageiros × horas;
2. $\frac{\sum_{i \in U} \sum_{e \in U} d_{ie} \frac{T_{ie}}{c_{ie}}}{\sum_{i \in U} \sum_{e \in U} d_{ie}}$: representa o desvio médio em relação ao tempo ideal de viagem considerando a velocidade média do modelo. O tempo ideal de viagem é apenas o tempo de deslocamento dentro do veículo. Faz-se a ponderação pela demanda da viagem entre (i, e), favorecendo novamente linhas com maior demanda dos usuários. Nota-se que o desvio é medido proporcionalmente (T_{ie}/c_{ie}). Este objetivo é adimensional;
3. $\sum_{n=1}^{R_{\max}} V_n$: representa o número de veículos utilizados pelo conjunto de linhas. Nota-se que uma redução no número de veículos melhora o número de passageiros/km e reduz os custos para o operador. A unidade para este objetivo é o número de veículos.

Como as unidades dos objetivos são diferentes, utiliza-se o método de normalização para a função objetivo proposto em Martínez et al.:

$$\max .z = B_1[\frac{\sum_{i \in U} \sum_{e \in U} d_{ie} T_{ie} - sa_1}{-\delta sa_1}] + B_2[\frac{\frac{\sum_{i \in U} \sum_{e \in U} d_{ie} \frac{T_{ie}}{c_{ie}}}{\sum_{i \in U} \sum_{e \in U} d_{ie}} - sa_2}{-\delta sa_2}] + B_3[\frac{\sum_{n=1}^{R_{\max}} V_n - sa_3}{-\delta sa_3}] \tag{28}$$

Na função objetivo normalizada, sa_1, sa_2 e sa_3 correspondem aos valores da solução atual da EMTU para os objetivos 1, 2 e 3, respectivamente. Nota-se que, como a modelagem busca minimizar os objetivos, é desejável que os valores dos objetivos sejam menores

que os valores de sa_i para $i = 1, 2, 3$, o que resultaria em objetivos normalizados negativos, caso não existissem os sinais de menos no denominador. Portanto, esse sinal inverte o sinal dos objetivos normalizados, garantindo que quanto mais distante os objetivos estiverem dos sa_i, melhor. Isso explica por que a função se torna de maximização.

O valor de δ funciona meramente para trazer os valores resultantes da função objetivo próximos a 0, na qual valores maiores que zero indicam soluções melhores que a da EMTU e valores menores que zero indicam soluções piores. Para este trabalho, considerou-se $\delta = 0.05$, mesmo valor utilizado em Martínez et al. (2017).

A seguir, detalha-se o significado de cada uma das restrições:

2. Garante que existe uma, e apenas uma, estação de início na linha. Observa-se que o caso $i = 0$ também está contemplado, o que significaria que a linha está desabilitada;

3. Garante que existe uma, e apenas uma, estação de fim da linha. Observa-se que o caso $j = 0$ também está contemplado, o que significaria que a linha está desabilitada;

4. Garante que as linhas passem pelas mesmas estações tanto no sentido sul-norte quanto no sentido norte-sul. Nota-se que se $X_{ij}^n = 1$, X_{ji}^n necessariamente precisa ser igual a 1 também. O mesmo vale para $X_{ij}^n = 0$;

5. Garante que para cada linha habilitada, a estação inicial seja anterior à estação final;

6. Garante que para cada linha habilitada, tanto no sentido sul-norte quanto no sentido norte-sul, a linha pode passar no máximo uma vez por cada estação;

7. Garante que a frequência das linhas habilitadas sempre será maior ou igual à frequência mínima estabelecida;

8. Calcula o número de estações da linha. Nota-se que o somatório percorre todos os pares X_{ij}^n com $i < j$. Como $X_{ij}^n = 1$ caso o caminho esteja habilitado, então o número de estações vai ser exatamente este somatório adicionado de 1 unidade. Por exemplo: considera-se uma linha que passa por 1, 3, 4: $X_{ij}^n = 1$ para os pares (1, 3) e (3, 4). Portanto, o número de estações será $2 + 1 = 3$.

9. Calcula a velocidade média para a linha com base no número de estações, pois vm_k é um parâmetro que é igual à velocidade média a depender do número k de estações;

10. Calcula o tempo em movimento do ônibus com base na distância entre i e j e a velocidade média da linha;

11. Calcula o tempo de ciclo da linha n considerando os tempos do veículo em movimento e os tempos de parada nas estações;

12. Determina o número de veículos para a linha com base no tempo de ciclo e na frequência. Nota-se que se a linha estiver desabilitada, o número de veículos é igual a 0;

13. Garante que o número total de veículos para o conjunto de linhas é menor ou igual ao tamanho da frota disponível;

14. Calcula a variável binária RT_{ik}^{n}. Nota-se que os termos somados são mutuamente excludentes. Se $X_{ik}^{n} = 1$, então k é a estação consecutiva de i, fazendo com que $RT_{ik}^{n} = 1$. Caso $X_{ik}^{n} = 0$, então k não é consecutiva de i e o somatório procura a possível estação j consecutiva de i. Caso essa estação exista, calcula-se o valor de RT_{jk}^{n} seguindo a mesma fórmula. Observa-se que o processo é necessariamente encerrado na estação de índice imediatamente anterior a k;

15. Garante que exista pelo menos uma linha que realize a viagem entre as estações i e j sem transferência;

16. Calcula o tempo de viagem entre as estações i e k. O tempo de viagem é o tempo do veículo em movimento somado ao tempo de parada da estação inicial. Nota-se que os dois termos somados são mutuamente excludentes. $X_{ik}^{n}(C_{ik}^{n} + s_i)$ refere-se ao caso em que k é a estação consecutiva de i para a linha n, enquanto o somatório de $X_{ij}^{n} RT_{jk}^{n}(t_{jk}^{n} + C_{ij}^{n} + s_i)$ refere-se ao caso em que k não é consecutivo de i;

17. Determina o valor da variável binária W_{ij}^{n}. Nota-se que para o produtório ser igual a 0, então deve existir alguma estação k tal que $RT_{ik}^{n} = 1$ e $RT_{kj}^{m} = 1$ com n e m sendo linhas diferentes. Quando o produtório é igual a 0, resulta em $W_{ij}^{n} = 1$;

18. Calcula o valor da variável binária O_{ij}^{n}. Observa-se que é possível viajar da estação i até j diretamente ($RT_{ij}^{n} = 1$) ou realizando uma transferência (W_{ij}^{n}). Como $O_{ij}^{n} = 1$ quando é possível viajar de i até j com ou sem transferência, basta que uma das variáveis RT_{ij}^{n} ou W_{ij}^{n} seja igual a 1. Caso ambas sejam 0, então O_{ij}^{n} também será zero;

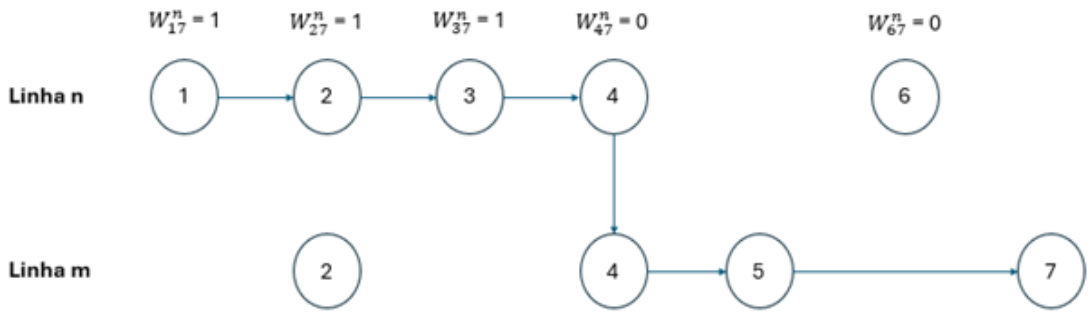
19. Garante que só exista uma linha que seja a mais rápida para a viagem sem transferência entre as estações i e j. Essa restrição é especialmente importante para o caso de existir mais de uma linha com o mesmo tempo de viagem entre i e j, forçando o algoritmo a escolher uma delas;

20. Determina o menor tempo t_{ij}^n utilizando a variável Z_{ij}^n;

21. Armazena a frequência da linha com o menor tempo t_{ij}^n utilizando a variável Z_{ij}^n;

22. Estima o tempo total de viagem entre as estações i e j utilizando n como a linha inicial. O termo $\frac{1}{f_n}$ calcula o tempo de espera (*headway*) para o ônibus chegar. Nota-se que os três termos restantes são todos excludentes entre si. O termo $RT_{ij}^n t_{ij}^n$ calcula o tempo de viagem caso as estações i e j pertençam à linha n. Caso contrário, $RT_{ij}^n = 0$ e, portanto, $1 - RT_{ij}^n = 1$. Os dois somatórios seguintes são mutuamente excludentes por conta de seus limites superiores e inferiores. Observa-se que o primeiro somatório percorre para todos os valores de e tal que $i \leq e < j$, enquanto o segundo somatório percorre para todos os valores de e tal que $j < e \leq i$. Ou seja, o primeiro somatório calcula o tempo total de viagem no caso de $i < j$ (sul-norte) e o segundo somatório calcula o tempo total de viagem no caso de $i > j$ (norte-sul). Os dois somatórios possuem a mesma lógica, dessa forma será analisado apenas o primeiro (sul-norte). Nota-se que $(W_{ej}^n - \sum_{k=1}^{j-e-1} X_{ee+k}^n W_{e+kj}^n)$ só será igual a 1 no caso em que e é a última estação que consegue chegar a j com transferência. Considera-se p como sendo a estação de transferência. Nota-se que $W_{p,j}^n = 0$ pois não é mais possível chegar de i a j com transferência partindo da linha n e, portanto, o termo será igual a 1, já que $W_{ej}^n = 1$. Observa-se também que para quaisquer outros casos, $(W_{ej}^n - \sum_{k=1}^{j-e-1} X_{ee+k}^n W_{e+kj}^n)$ será igual a 0, pois ou ambas as estações são capazes de realizar a transferência, resultando em 1 - 1 = 0, ou ambas as estações não são capazes de realizar a transferência, resultando em 0 - 0 = 0. A Figura 11 ilustra essa lógica para o exemplo de uma viagem com transferência de 1 até 7, iniciando na linha n:

Figura 11: Exemplo para a restrição 22



Fonte: Autor

Nota-se que para o exemplo $W_3^n - W_4^n = 1$ é o único momento em que a subtração de estações consecutivas é diferente de 0.

Portanto, uma vez identificada a estação de transferência, basta calcular o tempo total de viagem expresso pelo somatório:

$$\sum_{e<p<j} \left(X_{ep}^n \left(t_{ip}^n + \frac{1}{F_{pj}} + G_{pj} \right) \right)$$

que calcula o tempo até a estação de transferência (t_{ip}^n), o tempo de headway para a linha de transferência ($\frac{1}{F_{pj}}$) e o tempo de viagem da estação p até a j (G_{pj}). Esta formulação está de acordo com as últimas duas hipóteses definidas.

23. Determina o tempo total de viagem esperado para o par de estações (i, e). Nesta restrição, consideram-se as probabilidades de chegada dos veículos que operam em cada uma das linhas que realizam a viagem da estação inicial até a final com ou sem transferência. Tal modelagem das probabilidades foi utilizada em Martínez et al. (2017) e definida com base no trabalho de Spiess & Florian (1989).

24. Garante que os pesos atribuídos a cada objetivo não são negativos;

25. Garante que a soma dos pesos atribuídos a cada objetivo é igual a 1;

26. Determina o conjunto para a variável X_{ij}^n;

27. Determina o conjunto para as variáveis RT_{ij}^n e Z_{ij}^n.

3.3 Método de solução

Devido à complexidade e não-linearidade do problema, sua solução não pode ser obtida através de softwares solvers convencionais. Portanto, é necessário utilizar algoritmos heurísticos, implementados em Python, para resolver o problema. Utilizando a notação definida na Seção 3.1, são aplicados dois algoritmos: o primeiro para a geração do conjunto de linhas e o segundo para a definição das frequências. Ambos os algoritmos são executados simultaneamente, permitindo determinar as linhas e frequências de maneira conjunta. Estes algoritmos são os mesmos utilizados no trabalho de Martínez et al. (2017).

Uma diferença em relação ao trabalho de Martínez et al. (2017) é a adição de uma etapa de verificação após a execução do segundo algoritmo para garantir que as restrições (7) e (15) sejam respeitadas. Essa verificação é necessária porque o algoritmo pode alterar

o número de veículos em cada linha, potencialmente violando a restrição de frequência (restrição 7), ou desabilitar uma linha ao zerar o número de veículos nela, possivelmente violando a restrição 15.

A aplicação de heurísticas na resolução de problemas proporciona uma eficiência notável. Como destacado na revisão bibliográfica, as heurísticas são rápidas em encontrar soluções e conseguem incorporar diversos aspectos complexos na modelagem do problema, o que é particularmente vantajoso para o TNDFSP, dada a sua complexidade. Além disso, a adaptabilidade dos métodos heurísticos permite que a modelagem possa ser ajustada futuramente, caso sejam necessárias mudanças na implementação ou operação do BRT-ABC.

3.3.1 Algoritmo para a geração do conjunto de linhas

O algoritmo gera as linhas para o BRT-ABC começando pela primeira linha que inicia na primeira estação e termina na última. Para as linhas subsequentes, uma estação inicial é escolhida aleatoriamente entre as estações permitidas e uma estação final é selecionada aleatoriamente entre as estações posteriores. Em seguida, o algoritmo verifica todos os pares de estações e se um par não está presente em nenhuma linha, ele é inserido em uma linha que tenha uma estação inicial anterior ao par e uma estação final posterior, garantindo que a restrição (15) seja respeitada. Finalmente, as linhas são ordenadas e as estações duplicadas são removidas. O algoritmo para a geração do conjunto de linhas é desenvolvido respeitando as restrições (2) – (6) e a restrição (15).

Algoritmo 1 - Geração do Conjunto de Linhas

Para a linha $n = 1$, Estação inicial = 1;
Para a linha $n = 1$, Estação final = última estação de U;
End

Para as linhas $n = 2 \dots R_{\max}$:
    Selecione aleatoriamente uma estação inicial $i \in V$ e uma estação de retorno $j > i \in Y$ para a linha n;
    Próxima linha.
End

Para $i \in U$:
    Para $j > i \in U$:
        Se i e j não estão presentes em nenhuma linha:

Selecione aleatoriamente uma linha com estação inicial antes de i e estação final depois de j;

Insira a estação i e j nesta linha.

End

Próximo j.

End

Próximo i.

End

Para as linhas $n = 1 \ldots R_{\max}$:

Ordene as estações e delete estações repetidas.

End

3.3.2 Heurística para a determinação das frequências

Para a determinação heurística das frequências, utiliza-se um algoritmo que busca otimizar o número de veículos em cada linha. Inicialmente, o algoritmo aloca a quantidade de veículos necessária para cada linha, garantindo que a frequência mínima seja atendida e que o total de veículos não ultrapasse a capacidade da frota disponível. Em seguida, realiza a otimização da solução variando o número de veículos por linha e avaliando a função objetivo após cada alteração. Se a alteração resultar em uma melhoria, ela é mantida; caso contrário, é descartada. Durante esse processo, existe a possibilidade de que as restrições (7) e (15) sejam violadas após a aplicação da heurística, conforme mencionado anteriormente. Se a solução gerada violar alguma dessas restrições, ela é descartada. Todas as demais restrições são respeitadas.

Algoritmo 2 – Heurística para a determinação das frequências

Seja $S = (L, F)$ em que L é o conjunto de linhas e F é o conjunto de frequências de L e $F[q]$ é o elemento na posição q do vetor F.
Seja VN um vetor com $R_{\max}$ posições em que $VN[q]$ é o elemento de posição q do vetor.
Seja VS um vetor com $R_{\max} + 1$ posições em que $VS[k]$ é o elemento de posição k do vetor; valores a partir da posição 2 são iguais aos do vetor VN.
Seja nvn a soma de todos os valores de VN.
$q = 1$

Enquanto $q \leq R_{\max}$:

$VN[q] = \min\{k \in \mathbb{N} \mid k \geq f_{\min} * T_q\}$

$F[q] = VN[q]/T_q$

$q = q + 1$

End

Se nvn é menor que o tamanho da frota disponível W, então $VS[1] = W - nvn$, caso contrário $VS[1] = 0$.

$vae = nvn - W$;

Enquanto $vae > 0$:

Selecione aleatoriamente um número k do conjunto $\{k \in \mathbb{N} \mid 2 \leq k \leq R_{\max} + 1\}$;

$VS[k] = VS[k] - 1$;

$vae = vae - 1$;

End

Avalia S usando (28);

Para $m = 1$:

$VP = \{m + 1 \ldots R_{\max} + 1\}$

Enquanto $VP \neq \{\}$:

Selecione aleatoriamente um elemento n do conjunto VP;

$VP = VP - \{n\}$;

Enquanto o valor da função objetivo (28) for melhorado ou igual e o número $VS[n]$ é maior que zero:

$VS[n] = VS[n] - 1$;

$VS[m] = VS[m] + 1$;

$F[n-1] = VS[n]/T_{n-1}$

Avalia S usando (28);

End

End

Próximo m.

End

Para $m = 2 \ldots R_{\max}$:

$VP = \{m + 1 \ldots R_{\max} + 1\}$;

Enquanto $VP \neq 0$:

Selecione aleatoriamente um elemento n do conjunto VP;

$VP = VP - \{n\}$;

Enquanto o valor da função objetivo (28) é melhorado ou igual e o número $VS[m]$ é maior que 0 e as linhas $(m-1)$ e $(n-1)$ são diferentes:

$VS[n] = VS[n] - 1;$
$VS[m] = VS[m] + 1;$
$F[n-1] = VS[n]/T_{n-1}$
Avalia S usando (28);
End
End
Próximo m.
End

Para cada par de rotas idênticas, alocar o número de veículos atribuídos a ambas as rotas em apenas uma delas;
End

Se a restrição (7) ou a restrição (15) for violada:
Descartar a solução.
End

3.3.3 Exemplo para um caso reduzido

Para ilustrar o funcionamento dos algoritmos, o exemplo a seguir é apresentado para um caso reduzido.

Algoritmo 1:

Estações (U): 1, 2, 3, 4, 5
Estações de início (V): 1
Estações de Retorno (Y): 3, 5
$R_{\max} = 3$ linhas. Ou seja, gera-se um conjunto de 3 linhas ($n = 1$, $n = 2$, $n = 3$)

Para $n = 1$, Estação Inicial = 1
Para $n = 1$, Estação Final = última estação de $U = 5$

Para $n = 2, n = 3$:
Seleciona uma estação de início aleatória $i \in V$ e outra estação de retorno $j > i \in Y$
Inicial(2) = 1
Inicial(3) = 1
Final(2) = 5
Final(3) = 3

#Portanto, definiu-se as estações de início e retorno para as linhas:

$n = 1$: (1, 5)

$n = 2$: (1, 5)

$n = 3$: (1, 3)

Para $i \in U$:

Para $j \in U$ e $j > i$:

Se o par i e j não está presente em nenhuma linha:

Adicione i e j a uma linha que possui estação de início antes de i e estação de retorno antes de j.

$i = 1$ e $j = 2 \rightarrow$ não está presente, adiciona na linha 1: (1, 1, 2, 5)

$i = 1$ e $j = 3 \rightarrow$ está presente na linha 3

$i = 1$ e $j = 4 \rightarrow$ não está presente, adiciona na linha 2: (1, 1, 4, 5)

$i = 2$ e $j = 3 \rightarrow$ não está presente, adiciona na linha 3: (1, 2, 3, 3)

$i = 2$ e $j = 4 \rightarrow$ não está presente, adiciona na linha 1: (1, 1, 2, 2, 4, 5)

$i = 2$ e $j = 5 \rightarrow$ está presente na linha 1

$i = 3$ e $j = 4 \rightarrow$ não está presente, adiciona na linha 2: (1, 1, 3, 4, 5)

$i = 3$ e $j = 5 \rightarrow$ está presente na linha 2

$i = 4$ e $j = 5 \rightarrow$ está presente na linha 1 e linha 2

Remove duplicatas:

linha 1: (1, 2, 4, 5)

linha 2: (1, 3, 4, 5)

linha 3: (1, 2, 3)

#Portanto, o algoritmo gera um conjunto de linhas que respeita as restrições. Nota-se que é possível ir de i para j sem trocas de linhas para quaisquer i e j.

Algoritmo 2:

A solução é definida por $S = (L, F)$, na qual L é o conjunto de linhas e F é o conjunto de frequências para essas linhas.

Linhas definidas:

linha 1 = [1, 2, 4, 5]

linha 2 = [1, 3, 4, 5]

linha 3 = [1, 2, 3]

Para o exemplo, utiliza-se $f_{\min} = 8$ e $T_1 = 1,3$; $T_2 = 1,5$; $T_3 = 1,8$.

#O looping abaixo garante que as linhas possuem veículos suficientes para atender à frequência mínima estabelecida.

Enquanto $q \leq 3$:

$VN[1] = \min(k \in \mathbb{N} | k \geq f_{\min} * T_1) \rightarrow VN[1] = 11$

$F[1] = VN[1]/T_1 = 8.46$

$VN[2] = \min(k \in \mathbb{N} | k \geq f_{\min} * T_2) \rightarrow VN[2] = 12$

$F[2] = VN[2]/T_2 = 8$

$VN[3] = \min(k \in \mathbb{N} | k \geq f_{\min} * T_3) \rightarrow VN[3] = 15$

$F[3] = VN[3]/T_3 = 8.33$

Se $nvn < W$:

$VS[1] = W - nvn = 40 - 38 = 2$

Em outro caso ($nvn \geq W$):

$VS[1] = 0$

#Nota-se que no caso que $nvn > W$ o número de veículos alocados para as linhas é maior do que o número máximo de veículos definido (W). Portanto, usa-se uma variável auxiliar $vae = nvn - W$ de forma a retirar os veículos em excesso. O looping abaixo faz exatamente isso.

Para exemplificar esse looping, considera-se $W = 37$:

$vae = 38 - 37 = 1$

Enquanto $vae > 0$:

Selecione-se um número aleatório k entre 2 e $R_{\max} + 1$ ($3 + 1 = 4$). $k = 3$

$VS[3] = VS[3] - 1 = VN[2] - 1 = 12 - 1 = 11$

$vae = vae - 1$

$vae = 1 - 1 = 0$

#Após esses loopings, garante-se que a frequência de cada linha é maior que a mínima e que o número de veículos não ultrapassa o máximo definido.

Para o restante do exemplo, considera-se o caso em que $W = 40$:

$VS[1] = 2$;

$VS[2] = VN[1] = 11$;

$VS[3] = VN[2] = 12$;

$VS[4] = VN[3] = 15$;

#Após essa parte do algoritmo, tem-se uma solução inicial S pois já foram definidas tanto as linhas como as frequências para as linhas. Com essas informações, é possível calcular a Função Objetivo.

Continuação do Algoritmo:
Avalia S usando a Função Objetivo Normalizada.

#Os loopings abaixo têm como objetivo variar as quantidades de veículos em cada linha, consequentemente alterando as frequências. Uma vez alteradas as frequências, avalia-se a Função Objetivo Normalizada e observa-se se as alterações melhoraram ou pioraram a FO Normalizada. Mantém-se no looping enquanto a FO é melhorada ou continua a mesma. Caso seja feita alguma alteração que piore o valor da FO, o algoritmo sai do looping.

Considera-se um vetor auxiliar $VP = [m+1, m+2, \ldots, R_{\max}+1]$. $R_{\max} = 3 \rightarrow VP = [m+1, m+2, \ldots, 4]$.
Para $m = 1$:

$VP = [2, 3, 4]$

#A condição abaixo garante que o algoritmo passa por todos os valores de VP:
Enquanto $VP \neq \emptyset$:

Selecione-se um elemento aleatório n de VP e remove-se do conjunto:

Exemplo para $n = 3 \rightarrow VP = [2, 4]$

Enquanto o valor da FO normalizada é melhorado ou igual e o número $VS[1]$ é maior que zero:

$VS[1] = VS[1] - 1 = 2 - 1 = 1$
$VS[3] = VS[3] + 1 = 12 + 1 = 13$
$F[2] = VS[3]/T_2 = 13/1,5 = 8.67$
Avalia S usando a FO normalizada

#Nota-se que o looping retira veículos da posição de VS que guarda os veículos não utilizados e aloca na linha 2. Nota-se que o looping contém a condição "o número $VS[1]$ é maior que zero:" que garante que o valor em $VS[1]$ nunca será menor que 0. Observa-se que o vetor VS possui uma posição adicional em relação aos outros vetores destinada a armazenar o número de veículos não utilizados na sua primeira posição. Portanto, ao se referir ao índice de uma linha, o vetor VS sempre será maior em uma unidade em comparação com os outros vetores.

#O looping abaixo faz exatamente a mesma coisa, porém retirando os veículos da linha e avaliando se isso melhora ou piora a FO.

Enquanto o valor da FO normalizada é melhorado ou igual e o número $VS[3]$ é maior que zero:

$VS[3] = VS[3] - 1 = 13 - 1 = 12$
$VS[1] = VS[1] + 1 = 1 + 1 = 2$

$F[2] = VS[3]/T_2 = 12/1,5 = 8$

Avalia S usando a FO normalizada

#Observa-se que a condição "e o número $VS[3]$ é maior que zero" garante que o número de veículos da linha 2 não será negativo.

#Os loopings abaixo possuem exatamente a mesma lógica, porém variam os valores de veículos entre as linhas de ônibus. O algoritmo passa por todas as combinações de linhas. Para o exemplo abaixo, alteram-se as linhas 1 ($m = 2$) e linha 3 ($n = 4$). O primeiro looping retira veículos da linha 1 e aloca na linha 4 – avaliando se isso melhorou ou piorou a FO. O segundo looping retira veículos da linha 4 e aloca na linha 1 avaliando se isso melhorou ou piorou a FO.

Para $m = 2 \dots R_{\max} \rightarrow m = 2, 3$

#Para o exemplo, considera-se o caso $m = 2$:

Para $m = 2$:

$VP = [3, 4]$

Enquanto $VP \neq \emptyset$:

Selecione-se um elemento aleatório n de VP e remove-se do conjunto:

#Exemplo para $n = 4 \rightarrow VP = [3]$

Enquanto o valor da FO normalizada é melhorado ou igual e o número $VS[2]$ é maior que zero e as linhas $m - 1$ (1) e $n - 1$ (3) são diferentes:

$VS[2] = VS[2] - 1 = 11 - 1 = 10$

$VS[4] = VS[4] + 1 = 15 + 1 = 16$

$F[3] = VS[4]/T_3 = 16/1,8 = 8.89$

$F[1] = VS[2]/T_1 = 10/1,3 = 7.69$

Avalia S usando a FO normalizada

Enquanto o valor da FO normalizada é melhorado ou igual e o número $VS[2]$ é maior que zero e as linhas $m - 1$ (1) e $n - 1$ (3) são diferentes:

$VS[4] = VS[4] - 1 = 16 - 1 = 15$

$VS[2] = VS[2] + 1 = 10 + 1 = 11$

$F[3] = VS[4]/T_3 = 15/1,8 = 8.33$

$F[1] = VS[2]/T_1 = 11/1,3 = 8.46$

Avalia S usando a FO normalizada

Caso haja 2 linhas iguais, o algoritmo desabilita uma delas e aloca todos os veículos em apenas uma.

#O algoritmo roda o mesmo looping para o restante dos valores de m. Dessa forma, o

algoritmo 2 busca variar o número de veículos para cada linha, consequentemente alterando a frequência da linha de modo a otimizar o valor da Função Objetivo. A forma em que o algoritmo é modelado permite que sejam atribuídos 0 veículos para uma linha, desabilitando-a.

#Por fim, o algoritmo verifica se alguma restrição foi violada.

Verifica se as soluções respeitam a frequência mínima (restrição (7)).

#Como o algoritmo varia os números de veículos para cada linha, é possível que gere alguma(s) linha(s) que não respeitem o mínimo para a frequência. Nesse caso, a solução é descartada.

Verifica se as soluções respeitam a restrição de que é possível viajar de i a j sem transferência (restrição (15)).

#Como o algoritmo é capaz de desabilitar linhas, essa restrição pode acabar sendo violada. Nesse caso, a solução é descartada.

#A seguir apresenta-se a solução obtida considerando apenas as iterações mostradas. Nota-se que este é apenas um exemplo para a resolução do caso real. É necessário que todas as iterações sejam realizadas e que a Função Objetivo seja calculada de forma a avaliar as mudanças nos números de veículos.

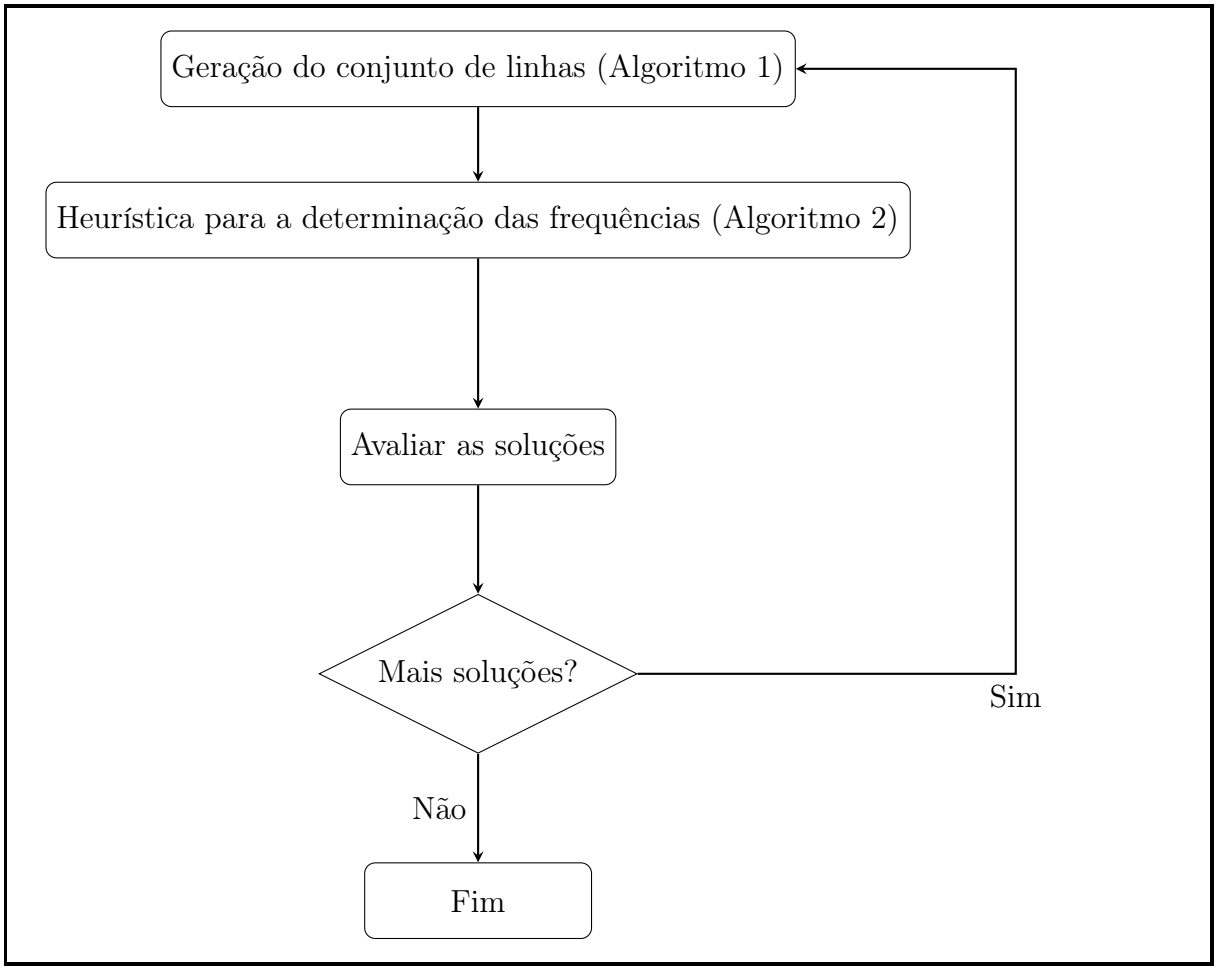
Linhas da solução:
$L = [[1, 2, 4, 5], [1, 3, 4, 5], [1, 2, 3]]$
Frequências para cada linha da solução:
$F = [8.46, 8.00, 8.33]$
Solução final:
$S = ([[1, 2, 4, 5], [1, 3, 4, 5], [1, 2, 3]], [8.46, 8.00, 8.33])$

3.3.4 Implementação do método de solução

De forma a implementar os algoritmos propostos e definir uma solução para o problema, estabelece-se uma estrutura de resolução, conforme ilustrado na Figura 12.

Neste método, gera-se um número de soluções utilizando os Algoritmos 1 e 2. Em seguida, realiza-se a análise dessas soluções, verificando se a demanda dos usuários e operadores foi atendida, gerando mais soluções caso necessário. Por fim, conclui-se o método com a definição da melhor solução encontrada para o problema da definição das linhas e frequências do BRT-ABC.

Figura 12: Estrutura de resolução proposta



Fonte: Autor

4 COLETA E TRATAMENTO DE DADOS

O capítulo trata da coleta e tratamento dos dados necessários para a modelagem do sistema de transporte BRT-ABC. Inicialmente, são apresentadas as velocidades médias para diferentes tipos de linhas, utilizando dados do Estudo Conceitual e Funcional da Systra, que modela três linhas com velocidades variando conforme o número de estações. A partir desses dados, é formulada uma equação que descreve a velocidade em função do número de estações, permitindo calcular os tempos de viagem entre estações. Em seguida, são detalhados os métodos utilizados para dimensionar o número de passageiros viajando entre diferentes estações, com base em dados de embarque e desembarque, e o tratamento das discrepâncias nos dados coletados. Por fim, são definidos os tempos de parada em cada estação.

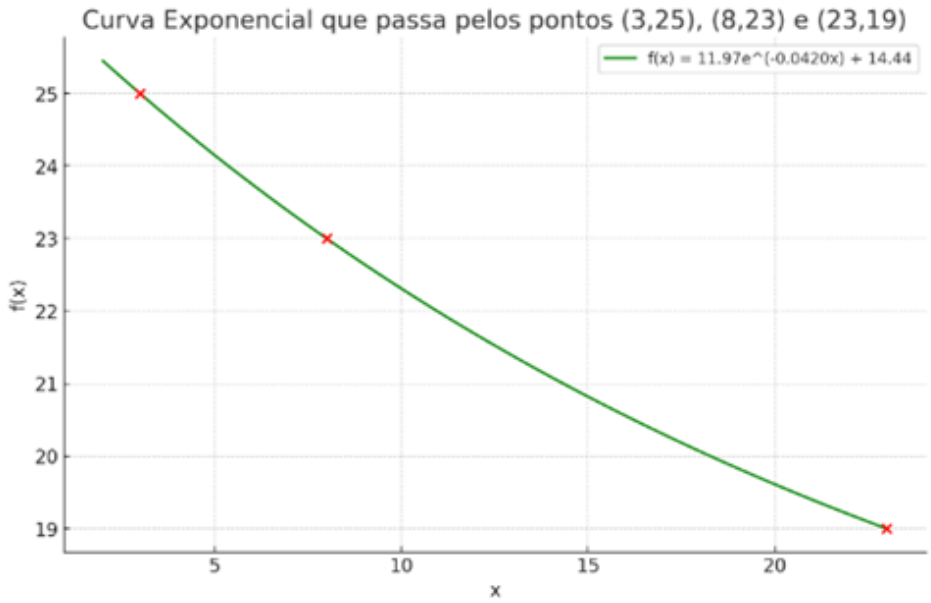
4.1 Cálculo das velocidades e dos tempos médios de trajeto

Para a modelagem da velocidade (incluindo a velocidade média), utilizou-se o Estudo Conceitual e Funcional da empresa Systra. No estudo, foram modeladas três linhas para o BRT-ABC, cada uma com diferentes velocidades. A linha Expressa passa apenas pelos terminais, possui 3 estações e uma velocidade média de 25 km/h. A linha Semi-Expressa passa por 8 estações e tem uma velocidade média de 23 km/h. Por fim, a linha Paradora passa por todas as 23 estações e possui uma velocidade média de 19 km/h.

Utilizando esses dados, considera-se os pares (3,25), (8,23) e (23,19), nos quais x representa o número de estações da linha e y a velocidade média (km/h) da linha para o par (x,y). Observa-se que a curva é sempre decrescente no intervalo de x = 2 até x = 23, pois quanto maior o número de estações, menor será a velocidade. A partir desses pontos, a equação que descreve a velocidade em função do número de estações é a seguinte:

$$f(x) = 11,97e^{-0,0420x} + 14,44$$

Figura 13: Curva da velocidade em função do número de estações



Fonte: Elaborado pelo autor com a utilização de inteligência artificial

Os pares (x,y), que representam a velocidade média em função do número de estações da linha, são apresentados na Tabela 7.

Dessa forma, foi possível modelar a velocidade para todos os casos possíveis. Com a equação da curva, é possível calcular o valor médio, através da fórmula:

$$\bar{y} = \frac{1}{b-a} \int_a^b y(x)\, dx$$

Obtendo-se $V_{\text{media}} \simeq 21{,}78$ km/h.

Com a velocidade média, é possível calcular o tempo médio de trajeto entre quaisquer estações, desde que a distância entre as estações seja conhecida. Para obter essas distâncias, foi utilizado o Google Maps. Algumas rotas entre estações ainda não foram construídas, portanto, nesses casos, foram utilizadas ruas e avenidas paralelas às que serão construídas para obter as distâncias equivalentes. Os tempos e distâncias são apresentados na Tabela 8.

Tabela 7: Pares (x, $f(x)$) para as estações e velocidades médias

x	$f(x)$
2	25,45
3	25,00
4	24,56
5	24,14
6	23,74
7	23,36
8	23,00
9	22,64
10	22,30
11	21,98
12	21,67
13	21,37
14	21,09
15	20,82
16	20,55
17	20,30
18	20,06
19	19,83
20	19,61
21	19,40
22	19,19
23	19,00

Fonte: Autor

Tabela 8: Distâncias e tempos de trajeto entre as estações do BRT

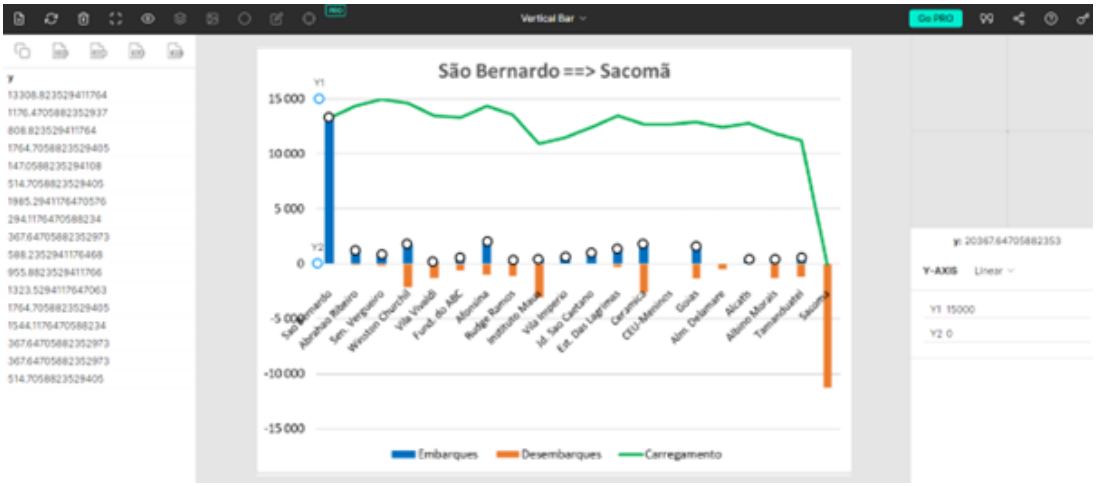
Estação de partida (EP)	**Índice EP**	**Estação de chegada (EC)**	**Índice EC**	**Distância (km)**	**Tempo (horas)**	**Tempo (minutos)**
Term. São Bernardo	1	Metrópole	2	0,80	0,037	2,20
Metrópole	2	Aldino Pinotti	3	0,60	0,028	1,65
Aldino Pinotti	3	Abrahão Ribeiro	4	0,75	0,034	2,07
Abrahão Ribeiro	4	Sen. Vergueiro	5	0,75	0,034	2,07
Sen. Vergueiro	5	Winston Churchill	6	0,75	0,034	2,07
Winston Churchill	6	Vila Vivaldi	7	0,40	0,018	1,10
Vila Vivaldi	7	Fund. Do ABC	8	0,60	0,028	1,65
Fund. Do ABC	8	Afonsina	9	1,10	0,051	3,03
Afonsina	9	Rudg Ramos	10	0,80	0,037	2,20
Rudg Ramos	10	Inst. Mauá	11	1,00	0,046	2,75
Inst. Mauá	11	Vila Império	12	0,75	0,034	2,07
Vila Império	12	Jd. São Caetano	13	0,45	0,021	1,24
Jd. São Caetano	13	Est. das Lágrimas	14	0,70	0,032	1,93
Est. das Lágrimas	14	Cerâmica	15	0,70	0,032	1,93
Cerâmica	15	CEU - MENINOS	16	0,60	0,028	1,65
CEU - MENINOS	16	Goiás	17	0,65	0,030	1,79
Goiás	17	Alm. Delamare	18	0,70	0,032	1,93
Alm. Delamare	18	Alcatéis	19	0,80	0,037	2,20
Alcatéis	19	Albino de Morais	20	0,55	0,025	1,52
Albino de Morais	20	Term. Tamanduateí	21	0,80	0,037	2,20
Term. Tamanduateí	21	Rua do Grito	22	0,85	0,039	2,34
Rua do Grito	22	Term. Sacomã	23	1,40	0,064	3,86

Fonte: Autor

4.2 Cálculo do número de passageiros viajando entre estações

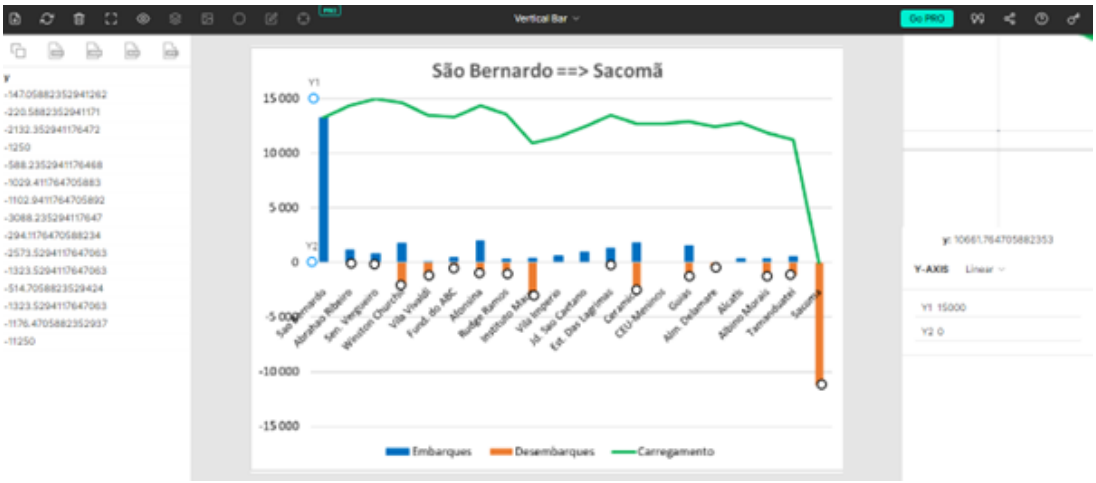
Para dimensionar o número de passageiros viajando entre duas estações quaisquer i e j, utilizou-se novamente o Estudo da Systra. No entanto, no estudo, não é fornecida a matriz de origem-destino para o BRT-ABC. Para fazer esse dimensionamento, utilizou-se o número de Embarques e Desembarques para o horário de pico para cada estação. Os dados foram extraídos utilizando o software PlotDigitizer. Devido ao uso deste software para capturar os valores dos gráficos, é esperado obter valores com casas decimais, mesmo que os valores originais sejam números inteiros, uma vez que se referem ao número de passageiros.

Figura 14: Embarques sentido sul-norte



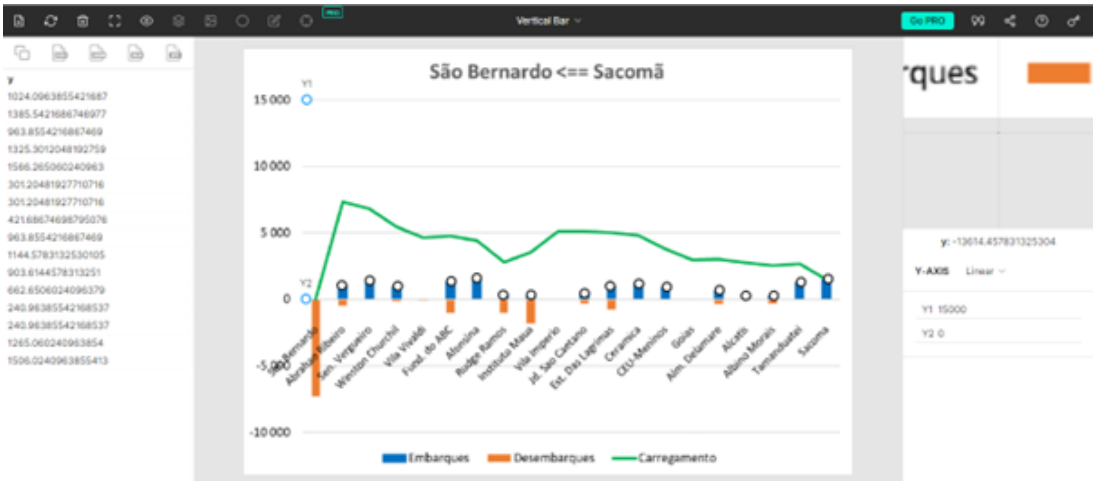
Fonte: Elaborado pelo autor utilizando o PlotDigitizer

Figura 15: Desembarques sentido sul-norte



Fonte: Elaborado pelo autor utilizando o PlotDigitizer

Figura 16: Embarques sentido norte-sul



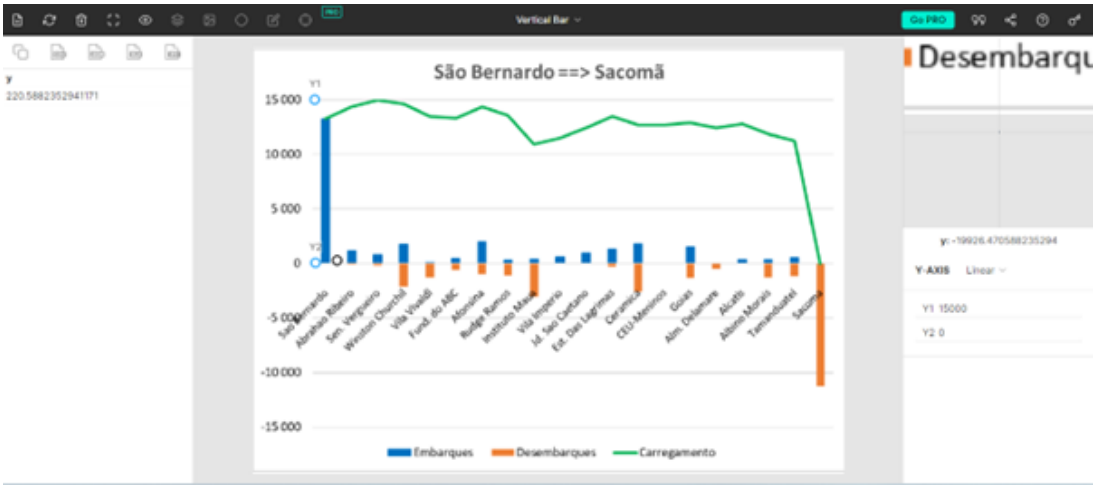
Fonte: Elaborado pelo autor utilizando o PlotDigitizer

Figura 17: Desembarques sentido norte-sul

Fonte: Elaborado pelo autor utilizando o PlotDigitizer

No entanto, o PlotDigitizer não é totalmente preciso. O número de embarques e desembarques deve ser igual, e através das medições, houve valores remanescentes. Para o sentido sul-norte, esse valor foi de 220,59 desembarques a mais, e para a volta foi de 361,45 embarques a mais. Para o sentido sul-norte, os valores remanescentes de desembarque foram distribuídos proporcionalmente para os embarques das estações. Para o sentido norte-sul, os valores remanescentes de embarque foram distribuídos proporcionalmente para os desembarques das estações. Nota-se que esses desvios representam uma variação muito baixa em relação ao valor real, os quais são representados pelos pontos de preenchimento branco nas Figuras 18 e 19.

Figura 18: Desvio para o sentido sul-norte



Fonte: Elaborado pelo autor utilizando o PlotDigitizer

Figura 19: Desvio para o sentido norte-sul

Fonte: Elaborado pelo autor utilizando o PlotDigitizer

Dessa forma, constroem-se as Tabelas 9 e 10 para o número de embarques e desembarques para ambos os sentidos.

Tabela 9: Embarques e desembarques para o sentido sul-norte

Estação	**Embarques**	**Desembarques**
Term. São Bernardo	13414,45	0,00
Metrópole	0,00	0,00
Aldino Pinotti	0,00	0,00
Abrahão Ribeiro	1185,81	147,06
Sen. Vergueiro	815,24	220,59
Winston Churchill	1778,71	2132,35
Vila Vivaldi	148,23	1250,00
Fund. Do ABC	518,79	588,24
Afonsina	2001,05	1029,41
Rudge Ramos	296,45	1102,94
Inst. Mauá	370,56	3088,24
Vila Império	592,90	0,00
Jd. São Caetano	963,47	0,00
Est. das Lágrimas	1334,03	294,12
Cerâmica	1778,71	2573,53
CEU - MENINOS	0,00	0,00
Goiás	1556,37	1323,53
Alm. Delamare	0,00	514,71
Alcatis	370,56	0,00
Albino de Morais	370,56	1323,53
Term. Tamanduateí	518,79	1176,47
Rua do Grito	0,00	0,00
Term. Sacomã	0,00	11250,00

Fonte Autor

Tabela 10: Embarques e desembarques para o sentido norte-sul

Estação	**Embarques**	**Desembarques**
Term. São Bernardo	0,00	7541,12
Metrópole	0,00	0,00
Aldino Pinotti	0,00	0,00
Abrahão Ribeiro	1024,10	494,50
Sen. Vergueiro	1385,54	0,00
Winston Churchill	963,86	185,44
Vila Vivaldi	0,00	123,62
Fund. Do ABC	1325,30	1050,81
Afonsina	1566,27	0,00
Rudge Ramos	301,20	1050,81
Inst. Mauá	301,20	1854,37
Vila Império	0,00	0,00
Jd. São Caetano	421,69	370,87
Est. das Lágrimas	963,86	803,56
Cerâmica	1144,58	0,00
CEU - MENINOS	903,61	0,00
Goiás	0,00	0,00
Alm. Delamare	662,65	370,87
Alcatis	240,96	0,00
Albino de Morais	240,96	370,87
Term. Tamanduateí	1265,06	0,00
Rua do Grito	0,00	0,00
Term. Sacomã	1506,02	0,00

Fonte: Autor

Com os números de embarque e desembarque, é possível seguir com o dimensionamento do número de passageiros viajando entre as estações.

Nota-se que os desembarques em uma estação qualquer representam todas as viagens que tiveram como destino essa estação. Além disso, todas essas viagens partiram de estações anteriores a essa estação de destino. Dessa forma, é possível estimar o número de passageiros viajando de i até j através das fórmulas de recorrência abaixo:

Para o sentido sul-norte:

$$d_{ij} = \frac{E_i - \sum_{k=i+1}^{j-1} d_{ik}}{\text{Total Passageiros } j} \cdot D_j$$

Para o sentido norte-sul:

$$d_{ij} = \frac{E_i - \sum_{k=i-1}^{j+1} d_{ik}}{\text{Total Passageiros } j} \cdot D_j$$

Observa-se que as fórmulas possuem a seguinte lógica:

O total de passageiros viajando de i até j é o número de passageiros que embarcaram em i (Ei), subtraído do número de passageiros que viajam até k (k antes de j) partindo de i, dividido pelo total de passageiros em j (antes dos desembarques e embarques em j), multiplicado pelo número de desembarques em j (Dj). Ou seja, realiza-se a divisão proporcional dos desembarques em j entre todas as possíveis viagens de i até j, para qualquer i anterior a j, considerando as viagens que partem de i mas desembarcam antes de j.

4.3 Cálculo dos tempos de parada

Por fim, define-se os tempos de parada em cada estação. Esse dado também não é fornecido pelo estudo. Portanto, faz-se o cálculo proporcional ao Terminal São Bernardo, que possui tempo de parada fornecido pelo estudo, de 3 minutos, e é o Terminal/Estação com o maior número de Embarques/Desembarques. Define-se também um tempo de parada mínimo de 10 segundos. Dessa forma, o tempo de parada em segundos de uma estação i é dado pela fórmula:

$$s_i = \max\left(\frac{\max(E_i, D_i)}{E_{\text{São Bernardo}}} \cdot 3 \cdot 60, 10\right)$$

Observa-se que é utilizado o valor máximo entre o tempo de embarque e o tempo de desembarque para a estação i, pois ambos os tempos são independentes, e o gargalo será determinado pelo maior dos dois. Os tempos de parada médios para cada estação são mostrados na Tabela 11.

Tabela 11: Tempos médios de parada para as estações

Estação	**Sul-Norte**	**Norte-Sul**	**Média (segundos)**	**Média (horas)**
Term. São Bernardo	180,00	180,00	180,00	0,0500
Metrópole	0,00	0,00	10,00	0,0028
Aldino Pinotti	0,00	0,00	10,00	0,0028
Abrahão Ribeiro	15,91	24,44	20,18	0,0056
Sen. Vergueiro	10,94	33,07	22,01	0,0061
Winston Churchil	28,61	23,01	25,81	0,0072
Vila Vivaldi	16,77	10,00	13,39	0,0037
Fund. Do ABC	10,00	31,63	20,82	0,0058
Afonsina	26,85	37,39	32,12	0,0089
Rudge Ramos	14,80	25,08	19,94	0,0055
Inst. Mauá	41,44	44,26	42,85	0,0119
Vila Império	10,00	0,00	10,00	0,0028
Jd. São Caetano	12,93	10,07	11,50	0,0032
Est. das Lágrimas	17,90	23,01	20,45	0,0057
Cerâmica	34,53	27,32	30,93	0,0086
CEU - MENINOS	0,00	21,57	10,78	0,0030
Goiás	20,88	0,00	10,44	0,0029
Alm. Delamare	10,00	15,82	12,91	0,0036
Alcatis	10,00	10,00	10,00	0,0028
Albino de Morais	17,76	10,00	13,88	0,0039
Term. Tamanduateí	120,00	120,00	120,00	0,0333
Rua do Grito	0,00	0,00	10,00	0,0028
Term. Sacomã	180,00	180,00	180,00	0,0500

Fonte: Autor

Para estações sem demanda nenhuma, o tempo de parada foi considerado como 0 para os ambos os sentidos. Caso a demanda exista, porém gere um tempo de parada menor do que 10 segundos, considerou-se o tempo de parada de 10 segundos.

5 VALIDAÇÃO DO MÉTODO PROPOSTO

O capítulo trata da validação do método proposto para a otimização do sistema de transporte BRT-ABC. Inicialmente são definidos os conjuntos de estações e parâmetros essenciais como a frequência mínima de operação e o número máximo de linhas. Em seguida apresenta-se a solução atual proposta pelo Estudo Funcional e Conceitual. Após isso é realizada uma análise de viabilidade utilizando os pesos propostos pela EMTU para os objetivos de tempo de viagem, desvio médio e número de veículos. O processo de validação inclui a geração de múltiplas soluções e a avaliação dessas soluções para verificar sua conformidade com as restrições estabelecidas. Por fim, o capítulo discute a escolha da melhor solução com base na análise dos resultados obtidos e nas exigências operacionais do sistema.

Para aplicar e validar o método proposto, é necessário definir alguns dos conjuntos e parâmetros a serem utilizados. Os conjuntos de estações para os corredores são:

$$U = \{1, 2, 3, \ldots, 23\}$$

$$V = \{1\}$$

$$Y = \{21, 23\}$$

O método para a definição da frequência mínima foi o mesmo utilizado em Martínez et al. (2017), sendo a frequência mínima igual à menor frequência do conjunto de linhas da solução inicial, que neste estudo de caso é a da EMTU. Portanto, através do Estudo Conceitual e Funcional, consegue-se determinar o valor de $f_{\min} = 8$ veículos/hora (Linha Expressa).

Para a determinação do valor de $R_{\max}$, utilizou-se inicialmente uma adaptação do método proposto em Martínez et al. (2017). No estudo, um dos fatores que os autores utilizam na definição do $R_{\max}$ é o número máximo de chegadas na estação que suporta o número mínimo de chegadas (número de veículos/hora). No entanto, não foi considerado

tal parâmetro na modelagem para o BRT-ABC pois não havia essa informação disponível. Portanto, para este estudo de caso foi utilizada apenas uma parte da modelagem utilizada por Martínez et al. (2017) em que:

$$R_{\max} = \min \left\{ k \in \mathbb{N} \mid k = \frac{W}{nv_{\max}} \right\}$$

Em que n_{vmax} é o número de veículos necessário para operar a linha paradora com a frequência mínima. Para o BRT-ABC, $n_{\text{vmax}} = 18$ e, portanto, $R_{\max} = 5$. No entanto, o algoritmo não se comportou adequadamente, encontrando muita dificuldade para identificar linhas viáveis que não violassem a restrição de frequência. Dessa forma, foi testada a utilização de $R_{\max} = 4$, mas este valor também não produziu bons resultados. Por fim, adotou-se o mesmo valor para $R_{\max}$ que o número de linhas definido no Estudo Conceitual e Funcional, ou seja, $R_{\max} = 3$. Este também é o valor utilizado no trabalho de Martínez et al. (2017). Para $R_{\max} = 3$, o algoritmo comporta-se muito bem.

A definição do tamanho disponível da frota também foi com base no Estudo Funcional e Conceitual, que prevê a utilização de 76 veículos entre as 3 linhas.

Para a solução atual da EMTU, o tempo total esperado de viagem é de $s_{a1} = 19898,60$, o desvio médio é de $s_{a2} = 1,71$ e o total de veículos utilizados é de $s_{a3} = 76$. Nesta solução, são propostos três tipos de linhas: expressa, semi-expressa e paradora.

- A linha expressa atende exclusivamente aos terminais, utilizando uma frota de 12 veículos e operando com uma frequência de 8 ônibus por hora.
- A linha semi-expressa inclui os terminais e cinco estações intermediárias, com uma frota de 34 veículos e uma frequência de 20 ônibus por hora.
- A linha paradora atende todos os terminais e estações com uma frota de 30 veículos e uma frequência de 15 ônibus por hora.

As configurações das linhas estão ilustradas nas Figuras 20, 21 e 22.

Figura 20: Linha expressa

Fonte: Estudo Funcional e Conceitual - Systra

Figura 21: Linha semi-expressa

Fonte: Estudo Funcional e Conceitual - Systra

Figura 22: Linha paradora

Fonte: Estudo Funcional e Conceitual - Systra

Para a definição dos pesos B_1, B_2 e B_3, foram consultados os representantes da EMTU que auxiliaram no desenvolvimento deste trabalho, perguntando quais deveriam ser esses pesos do ponto de vista da empresa. Dessa forma, definiu-se inicialmente $B_1 = 0.4$, $B_2 = 0.2$ e $B_3 = 0.4$. No entanto, tais pesos não produzem soluções viáveis conforme será analisado na seção a seguir.

5.1 Análise de viabilidade

Foram geradas 1000 soluções utilizando o método proposto com $B_1 = 0.4$, $B_2 = 0.2$ e $B_3 = 0.4$. Foi utilizado um computador com o processador Intel® Core™ i7-1185G7 de 11ª geração (frequência base de 1.80 GHz, frequência máxima de 3.00 GHz) e 16 GB de

RAM. Com essas configurações, as 1000 soluções foram geradas em aproximadamente 35 minutos, resultando em um tempo médio de cerca de 2,1 segundos por solução.

Para as 1000 soluções geradas, apenas 10 respeitam as restrições (7) e (15). Entre essas 10, a melhor solução encontrada pelo algoritmo foi a utilização exclusiva da linha paradora. No entanto, como foi atribuído um peso de 0.4 para a redução do número de veículos, o algoritmo priorizou excessivamente esse objetivo, prejudicando significativamente os outros objetivos. O melhor conjunto de linhas para essa configuração de pesos é apresentado na Tabela 12.

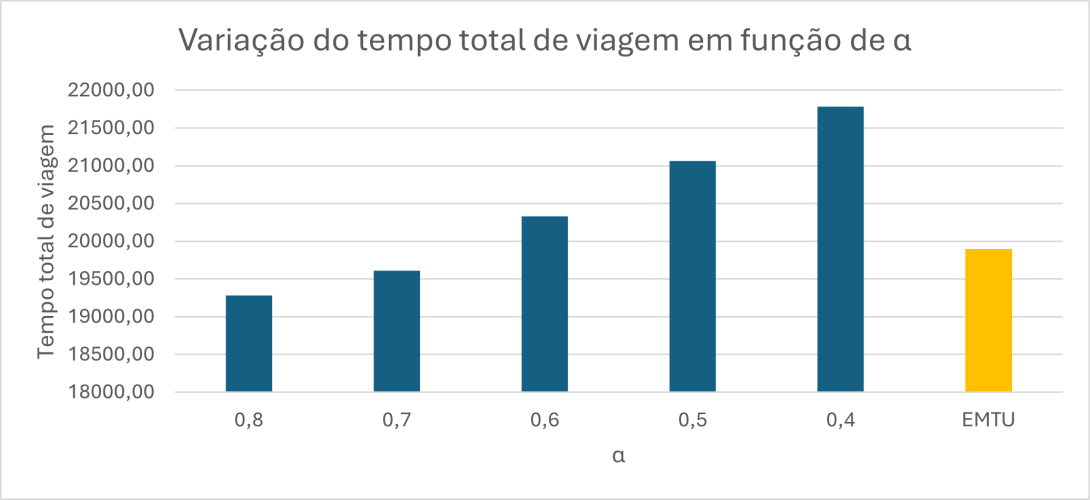
Tabela 12: Melhor solução para os pesos $B_1 = 0.4$, $B_2 = 0.2$ e $B_3 = 0.4$

FO Normalizada	4,19
Tempo total esperado de viagem	21780,57
Desvio médio	1,90
Número de veículos	25
Linha	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23
Frequência	11,35

Fonte: Elaborado pelo autor

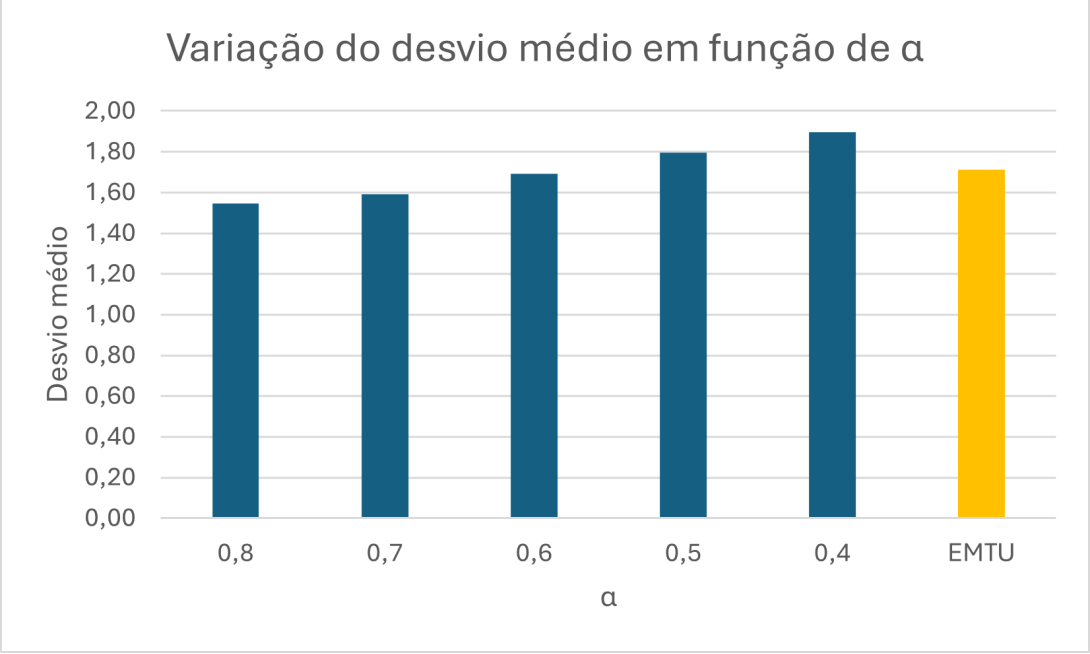
Portanto, observa-se que o número de veículos é extremamente reduzido considerando esses pesos, prejudicando excessivamente o tempo total esperado de viagem e o desvio médio. Assim, é necessário realizar um estudo de viabilidade para definir quais pesos serão utilizados. Para realizar tal análise, considerou-se α tal que o Tempo total esperado de viagem $= \alpha$ e o Número de veículos $= 0.8 - \alpha$. Nota-se que o caso proposto pela EMTU corresponde a $\alpha = 0.4$. Dessa forma, considerou-se apenas a linha paradora para esta análise, já que foi a solução encontrada para $\alpha = 0.4$. As Figuras 23, 24, 25 e 26 e a Tabela 13 ilustram o comportamento desta linha em função da variação de α. Os valores dos objetivos para a solução da EMTU também foram considerados para efeito de análise.

Figura 23: Variação do tempo total de viagem em função de α



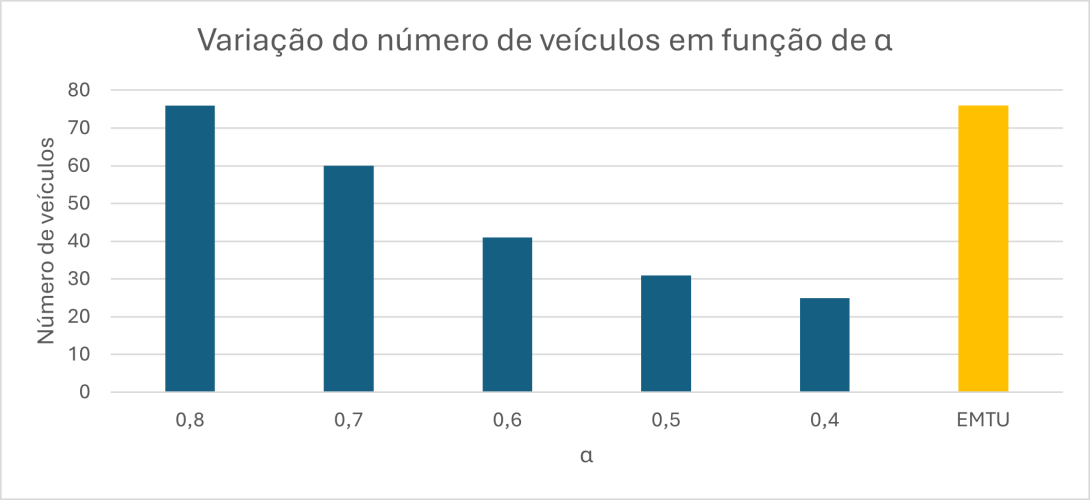
Fonte: Autor

Figura 24: Variação do desvio médio em função de α



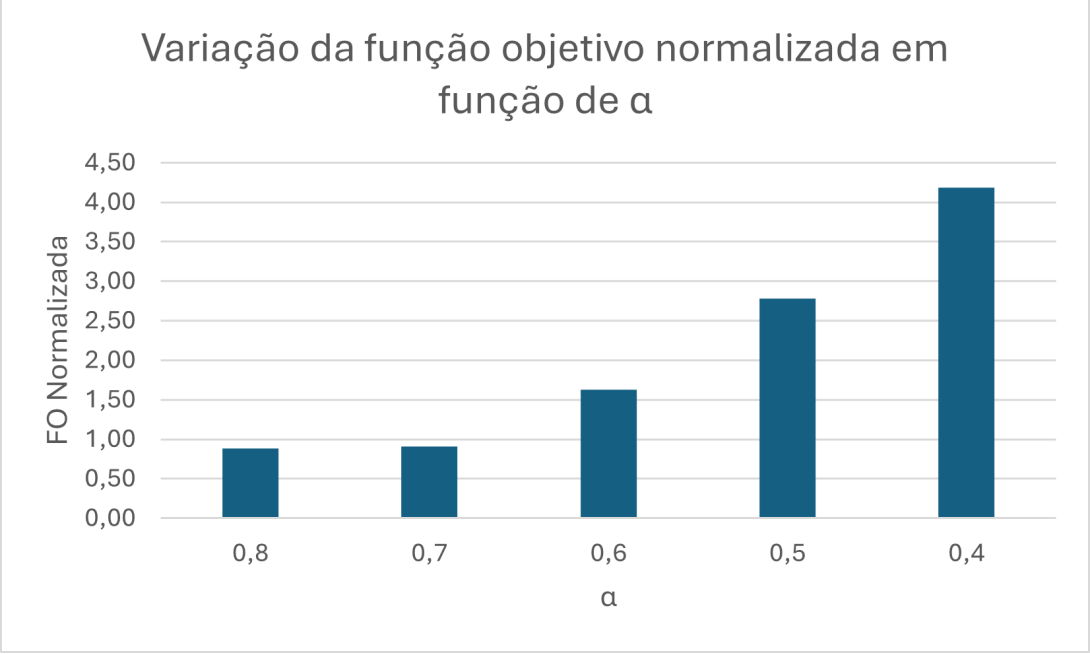
Fonte: Autor

Figura 25: Variação do número de veículos em função de α



Fonte: Autor

Figura 26: Variação da função objetivo normalizada em função de α



Fonte: Autor

Tabela 13: Resultados para diferentes valores de α

α	**FO Normalizada**	**Tempo total esperado de viagem**	**Desvio médio**	**Número de veículos**
0,8	0,88	19283,97	1,55	76
0,7	0,91	19610,32	1,59	60
0,6	1,63	20328,70	1,69	41
0,5	2,78	21060,49	1,79	31
0,4	4,19	21780,57	1,90	25
EMTU	-	19898,60	1,71	76

Fonte: Elaborado pelo autor

Portanto, observa-se que a partir de $\alpha = 0.7$ a utilização apenas da linha paradora é superior à solução da EMTU em todos os objetivos. Este resultado leva à utilização de $\alpha = 0.7$ para a modelagem deste estudo de caso. Dessa forma, os pesos utilizados para o restante da modelagem serão $B_1 = 0.7$, $B_2 = 0.2$ e $B_3 = 0.1$.

Além disso, este resultado é um forte indício de que a utilização apenas da linha paradora é uma solução candidata. O valor de $\alpha = 0.8$ foi considerado apenas para meios de comparação, já que $\alpha = 0.8$ implica em $B_3 = 0$, atribuindo peso nulo ao número de veículos e, portanto, não considerando os interesses do operador na modelagem.

5.2 Resultados

Considerando a análise de viabilidade, foram geradas 10.000 soluções com os pesos $B_1 = 0.7$, $B_2 = 0.2$ e $B_3 = 0.1$. A partir deste grupo de soluções, observou-se uma tendência do algoritmo em preferir soluções com apenas uma linha (paradora) ou duas linhas. Dessa forma, foram geradas mais 10.000 soluções considerando $R_{\max} = 2$.

Em conversas com a EMTU, foi informado que as linhas paradora e semi-expressa eram indispensáveis. Portanto, a solução recomendada respeitará essa restrição. No entanto, também será apresentada a solução com o melhor valor da função objetivo normalizada. A seguir são apresentadas as soluções sugeridas nas Tabela 14 e 15 e Figuras 27, 28, 29 e 30.

Tabela 14: Linhas e frequências para as soluções

Solução	**Linhas**	**Frequências**
Paradora	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 16,17,18,19,20,21,22,23	27,24
2 linhas	1,2,3,4,5,6,7,8,9,10,11,12, 13,14,15,16,17,18,19,20,21,22,23	26,33
	1,8,11,15,23	11,12
EMTU	1,21,23	8,00
	1,6,9,11,15,17,21,23	20,00
	1,2,3,4,5,6,7,8,9,10,11,12, 13,14,15,16,17,18,19,20,21,22,23	15,00

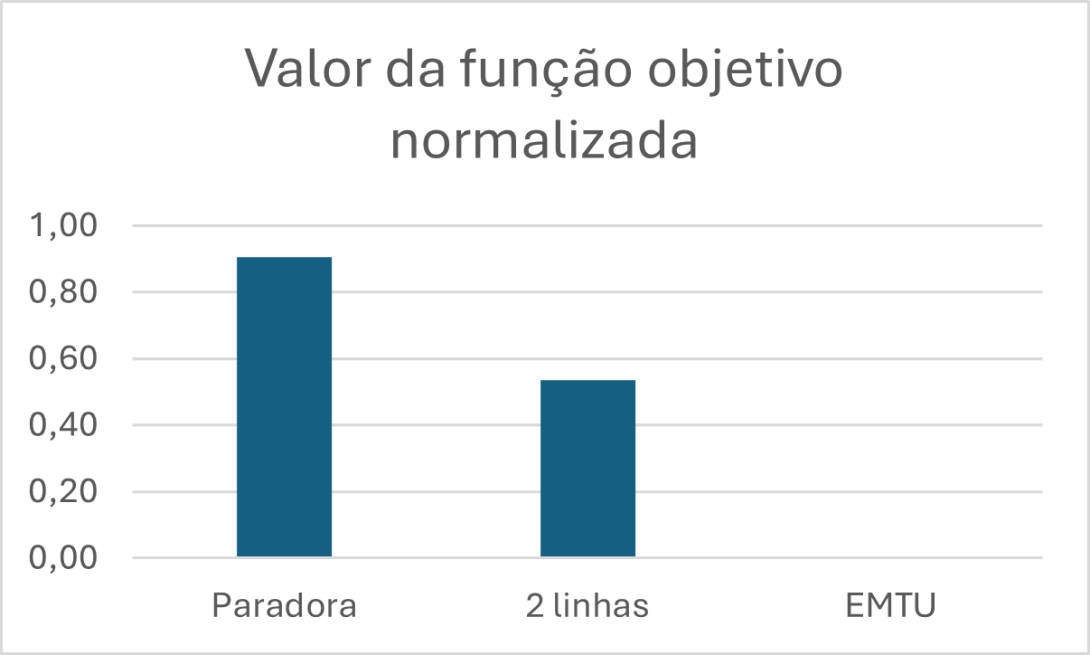
Fonte: Elaborado pelo autor

Tabela 15: Resultados obtidos para as soluções

Solução	**FO Normalizada**	**Tempo total esperado de viagem**	**Desvio médio**	**Número de veículos**
Paradora	0,91	19.610,32	1,59	60
2 linhas	0,53	19.522,47	1,60	76
EMTU	0,00	19.898,60	1,71	76

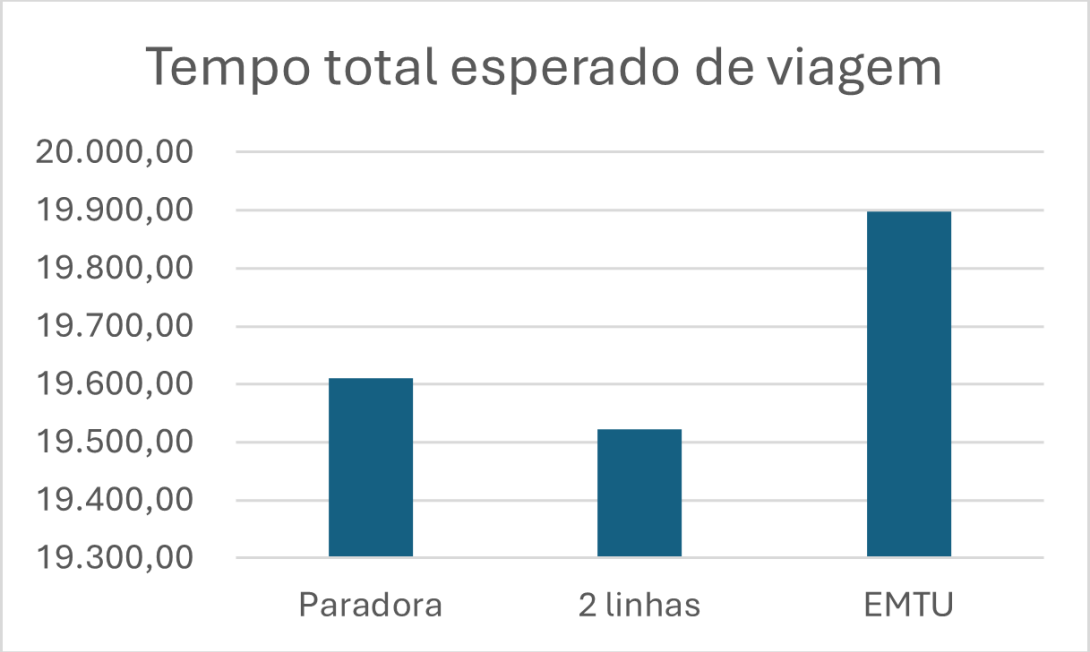
Fonte: Elaborado pelo autor

Figura 27: Valor da função objetivo normalizada



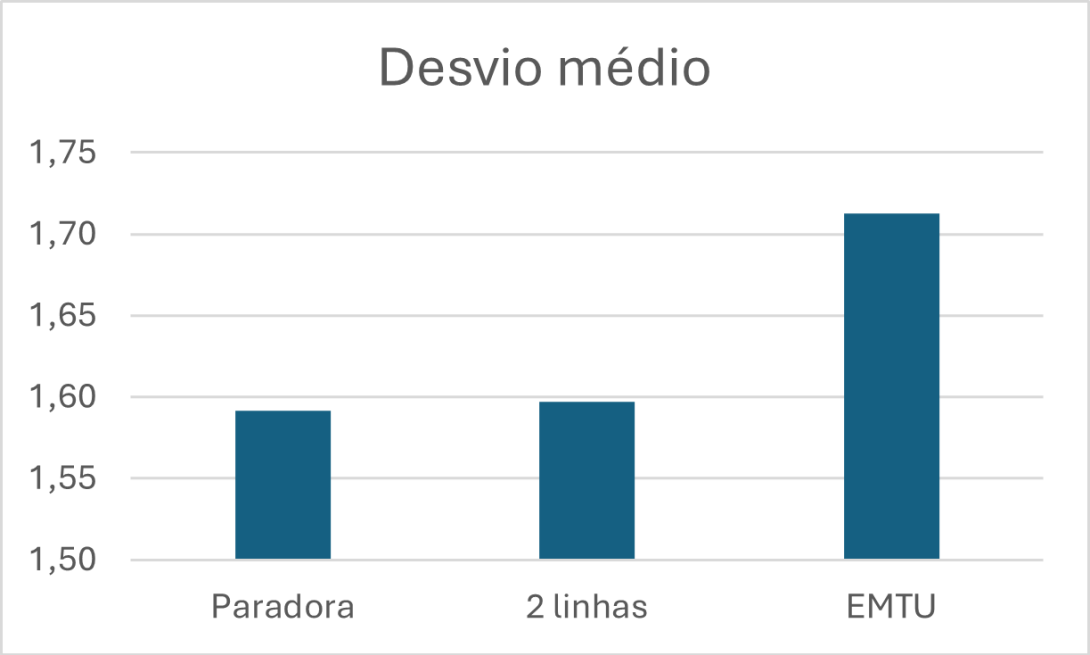
Fonte: Autor

Figura 28: Tempo total esperado de viagem



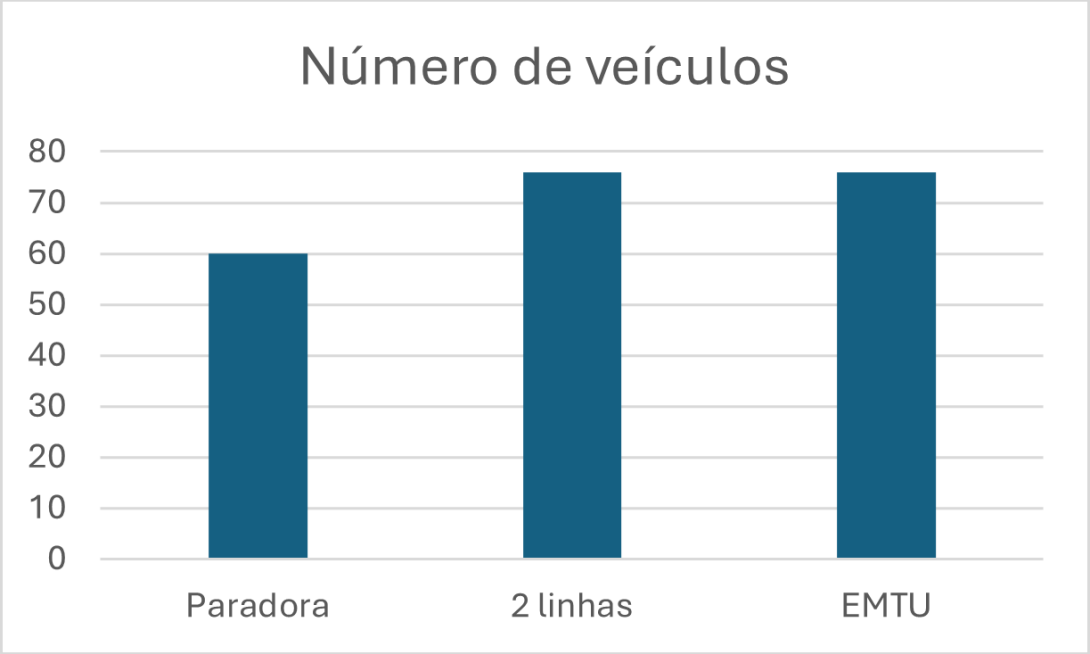
Fonte: Autor

Figura 29: Desvio médio



Fonte: Autor

Figura 30: Número de veículos



Fonte: Autor

A melhor solução encontrada pelo método proposto é a utilização exclusiva da linha paradora com 60 veículos. Essa solução apresenta um valor significativamente melhor

da função objetivo normalizada pois consegue reduzir os três objetivos. A solução com duas linhas também se mostrou muito eficiente, pois, embora mantenha o número de veículos utilizado pela EMTU, reduz consideravelmente os outros dois objetivos (voltados ao usuário do transporte público), com destaque ao tempo total esperado de viagem sendo o menor entre as três soluções.

Portanto, a recomendação final é a solução com duas linhas. A solução consiste em uma linha que passa por todas as estações (com 58 veículos) e outra que percorre as estações Terminal São Bernardo – Fund. Do ABC – Inst. Mauá – Cerâmica – Terminal Sacomã (com 18 veículos). Essas estações apresentam uma boa demanda e uma distância adequada entre elas.

6 CONCLUSÃO E PRÓXIMOS PASSOS

Conforme visto neste trabalho, o transporte público desempenha um papel fundamental na Região Metropolitana de São Paulo, impactando diretamente a vida de milhares de pessoas, especialmente aquelas de classes mais baixas, que dependem desse serviço para suas atividades diárias. A qualidade e eficiência do transporte público influenciam significativamente a mobilidade urbana, o acesso a oportunidades de emprego, educação e serviços de saúde. No entanto, a utilização de meios de transporte coletivo tem apresentado uma tendência de queda nos últimos anos, refletindo a necessidade urgente de melhorias e otimizações para reverter esta situação. Melhorar o transporte público não é apenas uma questão de conveniência, mas uma necessidade vital para promover a inclusão social e o bem-estar da sociedade.

Neste contexto, estudou-se o BRT-ABC de modo a realizar a modelagem de suas linhas e frequências através de métodos de otimização voltados à resolução de problemas do tipo TNDFSP. Este estudo visou a melhoria da eficiência operacional do sistema, e o consequente aumento de sua atratividade para os usuários, incentivando o uso do transporte público em detrimento do transporte individual. A implementação de um sistema de transporte público eficaz pode reduzir congestionamentos, diminuir a poluição ambiental e melhorar a qualidade de vida urbana de forma geral.

A fundamentação teórica do estudo baseou-se em referências bibliográficas relevantes, com uma revisão literária que permitiu o entendimento dos principais métodos de resolução para esse tipo de problema. Utilizou-se boa parte da modelagem proposta por Martínez et al. (2017), que também trata de um problema de definição das linhas e frequências de um BRT. Este tipo de problema é complexo e multiobjetivo, exigindo uma abordagem que considere os múltiplos objetivos e as diversas restrições para encontrar soluções viáveis e eficientes.

A metodologia proposta incluiu a modelagem matemática do problema e a utilização de algoritmos heurísticos implementados em Python para a sua resolução. O método implementado considerou uma função multiobjetivo, com o intuito de minimizar o tempo

total esperado de viagem, o desvio médio e o número de veículos. As soluções foram avaliadas com base nos valores gerados para a função objetivo e para seus objetivos, considerando as ponderações estabelecidas.

Os resultados demonstraram que foi possível encontrar soluções superiores às inicialmente propostas pela EMTU. A utilização exclusiva da linha paradora, com 60 veículos, mostrou-se eficiente em termos de redução dos três objetivos. No entanto, a solução com duas linhas, que inclui a linha paradora e uma linha semi-expressa, foi recomendada por atender melhor às necessidades operacionais da EMTU e melhorar significativamente os objetivos voltados ao usuário do transporte público. Essa solução conseguiu atingir um ótimo nível de serviço aos usuários, reduzindo o tempo total esperado de viagem e o desvio médio em relação à solução inicialmente proposta pela EMTU.

Os próximos passos consistem na implementação do algoritmo genético para a convergência da solução. A utilização de algoritmos genéticos, conforme visto na revisão bibliográfica, é a técnica mais utilizada para resolução de problemas de TNDFSP. Esta preferência da literatura é justificada pela eficiência deste método na resolução de problemas desse tipo. Os algoritmos heurísticos, embora eficazes na obtenção de boas soluções, não garantem a convergência para a solução ótima, destacando a importância de uma continuação do trabalho, com a aplicação do algoritmo genético.

Além disso, o refinamento dos dados utilizados é essencial para a melhora da modelagem, especialmente após o início das operações do BRT, quando dados mais confiáveis e detalhados poderão ser obtidos para uma análise mais precisa. Esse refinamento permitirá uma melhor precisão do modelo em relação às soluções propostas. Um exemplo de parâmetro que poderia ser melhorado é o tempo de parada, que foi estimado unicamente com base no tempo de parada no Terminal de São Bernardo.

Por fim, este trabalho buscou contribuir para a melhoria da logística urbana e do transporte público, visando garantir uma melhor qualidade de vida para as pessoas e beneficiar a sociedade como um todo. A longo prazo, espera-se que as soluções e métodos propostos ajudem a assegurar a operação do BRT-ABC em sua máxima eficiência, contribuindo para uma rede de transporte público mais resiliente e adaptável às necessidades crescentes da população, promovendo maior inclusão e acesso para todos.

7 REFERÊNCIAS BIBLIOGRÁFICAS

Baaj, M. H., & Mahmassani, H. S. (1995). Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research Part C, 3*(1), 31–50.

Beirão, G., & Cabral, J. A. S. (2007). Understanding attitudes towards public transport and private car: A qualitative study. *Transport Policy, 14*(6), 478-489.

Benn, H. P. (1995). Bus route evaluation standards. Tech. rep., Transportation Research Board, Washington.

Branke, Juergen & Deb, Kalyan & Miettinen, Kaisa & Roman, Slowinski. (2008). Multiobjective Optimization, Interactive and Evolutionary Approaches outcome of Dagstuhl seminars.

Bussieck, M. R. (1998). Optimal Lines in Public Rail Transport. PhD thesis, Technische Universitat Braunschweig, 1998.

Cancela, H., Mauttone, A., & Urquhart, M. E. (2015). Mathematical programming formulations for transit network design. *Transportation Research Part B: Methodological, 77*, 17–37.

Carrese, S., & Gori, S. (2004). Chapter 11: an urban bus network design procedure. In: Patriksson, M., & Labbé, M. (Eds.), *Transportation Planning: State of the Art (Applied Optimization)*, Springer, 64, pp. 177–196.

Ceder, A., & Wilson, N. H. M. (1986). Bus network design. *Transportation Research B, 20*, 331–344.

Ceder, A. (2003). Chapter 3: designing public transport network and routes. In: Lam,

W., & Bell, M. (Eds.), *Advanced Modeling for Transit Operations and Service Planning.* Elsevier Science Ltd. Pub., Pergamon Imprint, pp. 59–91.

Companhia do Metropolitano de São Paulo – Metrô. (2019). Pesquisa Origem Destino 2017: síntese. São Paulo: Metrô. Disponível em: *https://transparencia.metrosp.com.br/dataset/pesquisa-origem-e-destino*. Acesso em: 25 maio 2024.

DAMM, Ricardo de Brito. Métodos mono e multiobjetivo para o problema de escalonamento de técnicos de campo. 2016. *Tese (Doutorado em Engenharia de Produção)* - Escola Politécnica, Universidade de São Paulo, São Paulo, 2016. doi:10.11606/T.3.2016.tde-23062016-154154. Acesso em: 2024-06-10.

Dantzig, G. B., Harvey, R. P., Lansdowne, Z. F., Robinson, D. W., & Maier, S. F. (1979). Formulating e solving the network design problem by decomposition. *Transportation Research Part B, 13*(1), 5–17.

Dobbie, F., McConville, S., & Ormston, R. (2010). Understanding why some people do not use buses. Scottish Centre for Social Research (ScotCen).

Dubois, D., Bel, G., & Llibre, M. (1979). A set of methods in transportation network synthesis and analysis. *The Journal of the Operational Research Society, 30*(9), 797–808.

Empresa Metropolitana de Transportes Urbanos de São Paulo (EMTU). EMTU/SP - Site Oficial. Disponível em: *https://www.emtu.sp.gov.br/*. Acesso em: 18 maio 2024.

Fan, L., Mumford, C. L., & Evans, D. (2009). A simple multi-objective optimization algorithm for the urban transit routing problem. In: *Evolutionary Computation 2009. CEC'09. IEEE Congress on Evolutionary Computation*, pp. 1–7.

Farahani, R. Z., Miandoabchi, E., Szeto, W. Y., & Rashidi, H. (2013). A review of urban transportation network design problems. *European Journal of Operational Research, 229*(2), 281–302.

Friesz, T. L. (1985). Transportation network equilibrium, design and aggregation: key developments and research opportunities. *Transportation Research Part A, 19*(5–6),

413–427.

Fusco, G., Gori, S., & Petrelli, M. (2002). A heuristic transit network design algorithm for medium size towns. In: *Proceedings of the 13th Mini-EURO Conference*.

Guihaire, V., & Hao, J-K. (2008). Transit Network Design and Scheduling: a Global Review.

H., & Florian, M. (1989). Optimal strategies: A new assignment model for transit networks. *Transportation Research Part B: Methodological, 23*(2), 83–102.

Hasselström, D. (1979). A Method for Optimization of Urban Bus Route Networks. Volvo Bus Corporation, Göteborg, Sweden.

Hasselström, D. (1981). Public Transportation Planning – A Mathematical Programming Approach. Göteborg, Sweden.

Lampkin, W., & Saalmans, P. D. (1967). The design of routes, service frequencies, and schedules for a municipal bus undertaking: a case study. *OR*, 375–397.

Magnanti, T. L., & Wong, R. T. (1984). Network design and transportation planning: models and algorithms. *Transportation Science, 18*(1), 1–55.

Martínez, F., Baldoquin, M., & Mauttone, A. (2017). Model and solution method to a simultaneous route design and frequency setting problem for a bus rapid transit system in Colombia. *Pesquisa Operacional, 37*, 403-434.

Murray, A. T. (2003). A coverage model for improving public transit system accessibility and expanding access. *Annals of Operations Research, 123*, 143–156.

Oliveira, R., & Barbieri, C. (2015). Efficient transit network design and frequencies setting multiobjective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological, 81*(2), 355–376.

Rayward-Smith, V. J., Osman, I. H., Reeves, C. R., & Smith, G. D. (1996). *Modern Heuristic Search Methods*. Wiley, New York.

Schmöcker, J.-D., Bell, M., & Lam, W. (2003). Importance of public transport.

Transportation, 38, 1-4.

Shih, M., & Mahmassani, H. S. (1994). A Design Methodology for Bus Transit Networks with Coordinated Operations. No. SWUTC/94/60016-1. Center for Transportation Research, University of Texas, Austin.

Shih, M., Mahmassani, H. S., & Baaj, M. H. (1998). Planning and design model for transit route networks with coordinated operations. *Transportation Research Record, 1623*, 16–23.

Silman, L. A., Barzily, Z., & Passy, U. (1974). Planning the route system for urban buses. *Computers & Operations Research, 1*(2), 201–211.

Szeto, W. Y., & Wu, Y. (2011). A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong. *European Journal of Operational Research, 209*(2), 141–155.

Systra. (2019). Estudo Funcional e Conceitual - BRT ABC, 12 de agosto de 2019 (Número de Referência 1.ROM.27619). São Bernardo do Campo – Tamanduateí – Sacomã: BRT ABC.

Technibus. BRT-ABC e consórcio do Grande ABC criarão grupo de trabalho para acompanhar obras. Disponível em: *https://technibus.com.br/2024/05/16/brt-abc-e-consorcio-do-grande-abc-criarao-grupo-de-trabalho-para-acompanhar-obras*. Acesso em: 18 maio 2024.

Van Nes, R. (2002). Design of multimodal transport networks, a hierarchical approach. TRAIL Thesis Series T2002/5, DUP, Delft University, The Netherlands.

Wan, Q., & Lo, H. K. (2003). A mixed integer formulation for multiple-route transit network design. *Journal of Mathematical Modelling and Algorithms, 2*(4), 299–308.

Zanakis, S. H., Evans, J. R., & Vazacopoulos, A. A. (1989). Heuristic methods and applications: A categorized survey. *European Journal of Operational Research, 43*, 88-110.

8 ANEXO 1 - CÓDIGO IMPLEMENTADO EM PYTHON

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  9 20:15:16 2024

@author: andre.lauer
"""

def main():

    ## Algoritmo 1 ##
    #Criando o Conjunto U, com todas as estações:

    U = []

    for i in range(23):
        U.append(i)



    #Y = [0] #conjunto de estações de início
    V = [20,22] #conjunto de estações de retorno-fim


    Rmax = 2 #número de linhas - usando Rmax = 3 como exemplo

    #Criando o Conjunto L, que possui todas as linhas.
    L = []
    for i in range(Rmax):
        L.append([])

    #Definindo as estações iniciais e finais para a primeira rota:
    L[0].append(0)
```

```
    L[0].append(U[-1])



    #Definindo as estações iniciais para as linhas restantes:
    for i in range(1,Rmax):
        L[i].append(0)



    #Definindo as estações retorno-fim para as linhas restantes:
    import random
    for i in range(1,Rmax):
        L[i].append(random.choice(V))


    #adicionando as estações
    #Criando variáveis de True (1) ou False(0) como flags auxiliares
para Linha
    #A para a estação inicial i
    #B para a estação final j
    #C para o par i e j
    A = []
    B = []
    C = []
    for i in range(Rmax):
        A.append(0)
        B.append(0)
        C.append(0)


    for i in U:
        for j in U:
            if j>i:
                for k in range(Rmax):
```

```
                    for e in range(len(L[k])):
                        if i == L[k][e]:
                            A[k] = 1
                        if j == L[k][e]:
                            B[k] = 1
                    C[k] =min(A[k],B[k])


                if max(C) == 0:
                    #Criar uma Flag para selecionar uma rota que
tenha estação-fim maior que j
                    flag = False
                    while flag == False:
                        Aleatorio = random.randint(0,Rmax-1)
                        if j < max(L[Aleatorio]):
                            L[Aleatorio].append(i)
                            L[Aleatorio].append(j)
                            flag = True
                for r in range(Rmax):
                    A[r] = 0
                    B[r] = 0
                    C[r] = 0



    #Remover duplicatas
    def remover_duplicatas(lista_de_listas):
        # Usar list comprehension para iterar sobre cada sublista
        # e remover duplicatas convertendo cada sublista em um
dicionário e de volta para uma lista
        return [list(dict.fromkeys(sublista)) for sublista in
lista_de_listas]
```

```
    L_final = remover_duplicatas(L)

    #Ordena os elementos

    for i in range(Rmax):

        L_final[i] = sorted(L_final[i])


    ###-----------------------------------------------------------------
------###


    #### Input dos parâmetros ####


    ## Matriz dos tempos para o parâmetro cij.


    import numpy as np


    # Número de estações

    n = 23


    #Distância entre estações consecutivas:

    Distancia_est = np.array([0.8, 0.6, 0.75, 0.75, 0.75, 0.4, 0.6,
1.1, 0.8, 1, 0.75, 0.45, 0.7, 0.7, 0.6, 0.65, 0.7, 0.8, 0.55, 0.8,
0.85, 1.4])


    #Criando a lista de velocidades médias para cada linha

    V = []

    for i in range(Rmax):

        V.append(0)


    #Criando a lista de velocidades médias a depender do número de
estações da linha. Mínimo = 2 estações. Máx = 23 estações.

    V_media_est = [25.445592135458643, 25, 24.558885401175367,
24.14269342426314, 23.743619514685285, 23.360959602576166, 23,
22.642209092970468, 22.30485043318623, 21.98136740910469,
21.6711893127847, 21.373768909627962, 21.08858147291528,
20.81512385805256, 20.55291361489327, 20.301488136571262,
```

```
20.060403843342307, 19.82923539999439, 19.607574965446084,
19.39503147320915, 19.191229941445812, 19]



    #definindo os V[i]s:


    for i in range(Rmax):

        V[i] = V_media_est[len(L_final[i])-2]


    # Calculando os tempos entre estações consecutivas para cada
linha:

    Tempos_consecutivos = []

    for linha in L_final:

        velocidade = V_media_est[len(linha) - 2]

        tempos_linha = Distancia_est / velocidade

        Tempos_consecutivos.append(tempos_linha.tolist())


    # Função para criar a matriz de tempos de viagem entre todas as
estações para uma linha de ônibus

    def criar_matriz_tempos(tempos_consecutivos):

        # Inicializa a matriz de tempos com zeros

        matriz_tempos = np.zeros((n, n))


        # Preenche a matriz

        for i in range(n):

            for j in range(i+1, n):  # Calcula apenas a metade
superior devido à simetria

                # Soma os tempos consecutivos entre a estação i e j

                matriz_tempos[i][j] = matriz_tempos[i][j-1] +
tempos_consecutivos[j-1] if j-1 < len(tempos_consecutivos) else
matriz_tempos[i][j-1]

                matriz_tempos[j][i] = matriz_tempos[i][j]  #
Aproveita a simetria
```

```
        return matriz_tempos


    # Lista para armazenar as matrizes de tempos de todas as linhas
    matriz_cij = []


    # Calcula a matriz de tempos para cada linha
    for tempos_linha in Tempos_consecutivos:
        matriz = criar_matriz_tempos(tempos_linha)
        matriz_cij.append(matriz)



    # Tempos entre as estações consecutivas usando Vmédia da
modelagem
    tempos_consecutivos = [0.036730946, 0.027548209, 0.034435262,
0.034435262, 0.034435262, 0.018365473, 0.027548209, 0.050505051,
0.036730946, 0.045913682, 0.034435262, 0.020661157, 0.032139578,
0.032139578, 0.027548209, 0.029843893, 0.032139578, 0.036730946,
0.025252525, 0.036730946, 0.03902663, 0.064279155]



    # Criar a matriz para os cij que serão usados como parâmetros


    matriz_cij_media = np.zeros((n, n))


    # Preencher a matriz com os tempos de viagem
    for i in range(n):
        for j in range(i+1, n):
            matriz_cij_media[i][j] = matriz_cij_media[i][j-1] +
tempos_consecutivos[j-1]


    # Como a matriz é simétrica, copie os valores para a parte
inferior
    for i in range(n):
        for j in range(i):
```

```
            matriz_cij_media[i][j] = matriz_cij_media[j][i]


    ## listas com os tempos médios de parada (stop time) por estação


    Stop_time = [0.05, 0.002777778, 0.002777778, 0.005604985,
0.006112627, 0.007169317, 0.003718466, 0.00578247, 0.008921696,
0.005539116, 0.011902968, 0.002777778, 0.003193538, 0.00568152,
0.008590646, 0.002995624, 0.002900552, 0.00358568, 0.002777778,
0.0038555, 0.033333333, 0.002777778, 0.05]


    ## Listas com os Embarques e Desembarques para Ida e Volta


    Embarques_Ida = [13414.45, 0.00, 0.00, 1185.81, 815.24, 1778.71,
148.23, 518.79, 2001.05, 296.45,

        370.56, 592.90, 963.47, 1334.03, 1778.71, 0.00, 1556.37,
0.00, 370.56, 370.56,

        518.79, 0.00, 0.00]


    Desembarques_Ida = [0, 0, 0, 147.0588235, 220.5882353, 2132.35,
1250, 588.2352941, 1029.41, 1102.94,

        3088.24, 0, 0, 294.1176471, 2573.53, 0, 1323.53,
514.7058824, 0, 1323.53, 1176.47,

        0, 11250]


    Embarques_Volta = [0, 0, 0, 1024.10, 1385.54, 963.8554217, 0,
1325.30, 1566.27, 301.2048193,

        301.2048193, 0, 421.686747, 963.8554217, 1144.58,
903.6144578, 0, 662.6506024,

        240.9638554, 240.9638554, 1265.06, 0, 1506.02]


    Desembarques_Volta = [7541.121006, 0, 0, 494.4997381, 0,
185.4374018, 123.6249345, 1050.811943, 0,

        1050.811943, 1854.374018, 0, 370.8748036, 803.5620744, 0, 0,
0, 370.8748036, 0,

        370.8748036, 0, 0, 0]
```

```
    ## Criando a matriz Dij com o número de passageiros viajando de
i até j


    # Precisam-se criar uma lista auxiliar com o valor total dos
passageiros

    # a ser considerado no cálculo de Dij.

    # Note que o valor a ser considerado precisa ser o total até j-
1, pois

    # quem embarca em j não desembarca em j.


    #Para a Ida:


    Total_passageiros_ida = []

    Total_passageiros_ida.append(0) #O primeiro elemento é 0.


    for i in range(n-1):


Total_passageiros_ida.append(Total_passageiros_ida[i]+Embarques_Ida[
i]-Desembarques_Ida[i])




    # Para a Volta:


    Total_passageiros_volta = []

    Total_passageiros_volta.append(0) # O primeiro elemento é 0.


    # Corrigindo o range para iniciar de n-1 até 0, ou seja, de 22
até 0, e decrementar 1 a cada iteração

    for i in range(n-1, 0, -1):

        # Aqui, adiciona-se ao último elemento calculado os
embarques e subtrai-se os desembarques do índice atual

        Total_passageiros_volta.append(Total_passageiros_volta[-1] +
Embarques_Volta[i] - Desembarques_Volta[i])
```

```
    # Reverter a lista para ter a ordem correta da viagem de volta
(da estação 22 para a estação 0)
    Total_passageiros_volta.reverse()



    # Número de estações
    n = 23


    # Criar uma matriz nxn inicializada com zeros
    matriz_dij = np.zeros((n, n))


    #Cria uma variável auxiliar Dij
    Soma_dij = 0
    #Preenche a matriz:


    for j in range(n):
        for i in range(n):
            if i < j:
                for k in range(i,j):
                    Soma_dij = Soma_dij + matriz_dij[i][k]
                matriz_dij[i][j] =
Desembarques_Ida[j]*(Embarques_Ida[i] -
Soma_dij)/Total_passageiros_ida[j]
                Soma_dij = 0


    Soma_dij = 0


    for j in range(n-1,-1,-1):
        for i in range(n-1,-1,-1):
            if i > j:
                for k in range(i,j,-1):
```

```
                    Soma_dij = Soma_dij + matriz_dij[i][k]
                matriz_dij[i][j] =
Desembarques_Volta[j]*(Embarques_Volta[i] -
Soma_dij)/Total_passageiros_volta[j]
                Soma_dij = 0



    ## Criação da função t(ijn), que calcula o tempo de viagem entre
i e j para a linha n, i e j pertencem a n ##
    def t(i,j,n):
        if i in L_final[n] and j in L_final[n]:
            auxiliar_stoptime = 0
            if i > j:
                for k in range(i,j,-1):
                    auxiliar_stoptime = auxiliar_stoptime +
Stop_time[k]
            t = matriz_cij[n][i][j] + auxiliar_stoptime
            if i < j:
                for k in range(i,j):
                    auxiliar_stoptime = auxiliar_stoptime +
Stop_time[k]
            t = matriz_cij[n][i][j] + auxiliar_stoptime


            return t
        if i not in L_final[n] or j not in L_final[n]:
##retorna um número grande suficiente caso i ou j não façam parte da
linha n
            return 500


    # Criação da função T(i,j,n) que calcula o tempo de i até j com
ou sem transf.
    def T_(i,j,n):
        list_transf = []                            #lista auxiliar
para a estação de transf.
```

```
        estac_transf = 0                            #variável auxiliar
para estação de transf.

        group_transf = []           ## lista auxiliar para os
tempos após transferência

        tempo_transf = 0            ## variável auxiliar para o
tempo após transferência

        indice_linha = 0            ## variável auxiliar para a
linha que será transferida

        aux_check = 0               ## variável auxiliar para os is
e js consecutivos

        group_linha = []            ## lista auxiliar para
descobrir indice_linha

        if VS[n+1] == 0:            ## desconsidera linhas
desabilitadas

            return 0

        if i in L_final[n] and j in L_final[n]:        ## caso que i
e j pertencem a n

            return 1/F[n] + t(i,j,n)

        if i in L_final[n] and j not in L_final[n]:    ## caso que i
pertence mas j não

            if i < j and i != j:

                for s in range(len(L_final[n])):       ##lógica para
excluir os casos em que não é possível ir por transferência

                    if L_final[n][s] == i:

                        aux_check = s

                if L_final[n][-1] == i:

                    return 0

                if L_final[n][aux_check + 1] > j:

                    return 0

                for k in range(j-1,i,-1):

                    for q in range(Rmax):

                        if k in L_final[q] and j in L_final[q] and k
in L_final[n] and VS[q+1] != 0:   #olha para as linhas habilitadas,
k precisa pertencer a q e n

                            list_transf.append(k)
```

```
                if list_transf == []:     ##caso o algoritmo tenha
desabilitado a linha de transferência
                    return 0
                estac_transf = max(list_transf)          ## para a
ida, temos a última estac. de transf.
                for p in range(Rmax):
                    if VS[p+1] != 0:
                        group_transf.append(t(estac_transf,j,p))
                        group_linha.append(p)
                tempo_transf = min(group_transf)
                for r in range(len(group_transf)):
                    if tempo_transf == group_transf[r]:
                        indice_linha = group_linha[r]
                return 1/F[n] + t(i,estac_transf,n) +
1/F[indice_linha] + tempo_transf
            if i < j and i +1 == j:     #para a ida, estações
consecutivas em que j não pertnce a n
                return 0
            if i > j and i != j:
                for s in range(len(L_final[n])):
                    if L_final[n][s] == i:
                        aux_check = s
                if L_final[n][aux_check - 1] < j:
                    return 0
                for k in range(j+1,i):
                    for q in range(Rmax):
                        if k in L_final[q] and j in L_final[q] and k
in L_final[n] and VS[q+1] != 0:
                            list_transf.append(k)
                if list_transf == []:
                    return 0
                estac_transf = min(list_transf)  ##para a volta,
temos a última estac. de transf.
```

```
                for p in range(Rmax):
                    if VS[p+1] != 0:
                        group_transf.append(t(i,estac_transf,p))
                        group_linha.append(p)
                tempo_transf = min(group_transf)
                for r in range(len(group_transf)):
                    if tempo_transf == group_transf[r]:
                        indice_linha = group_linha[r]
                return 1/F[n] + t(i,estac_transf,n) +
1/F[indice_linha] + tempo_transf
            if i > j and i-1 == j:     #para a volta, estações
consecutivas em quej não pertence a n
                return 0
        if i not in L_final[n]:
            return 0    ## retorna 0 para T_(i,j,n) onde i não
pertence a n



    def X00(n):    #função que retorna 1 se a linha n está
habilitada, 0 caso contrário. Note que aqui está invertido da
formulação, para facilitar o código.
        if VS[n+1] == 0:
            return 0
        if VS[n+1] != 0:
            return 1


    def O(i,j,n):  #função que retorna 1 se da pra ir de i até j por
n, com ou sem transf. 0, caso contrário
        if T_(i,j,n) == 0:
            return 0
        if T_(i,j,n) != 0:
            return 1
```

```
    def Tmedio(i,j):    ##função que calcula o tempo médio de i até
j, considerando todas as possibilidades
        soma_todas = 0   #criando variável auxiliar para somar todas
as frequências habilitadas
        T_in = 0     #variável para calcular o Tmedio
        for k in range(Rmax):
            soma_todas = soma_todas + F[k]*O(i,j,k)*X00(k)
        if soma_todas == 0:   ##Caso o algoritmo tenha produzido uma
solução inviável, retorna um número suficientemente grande para
descartá-la
            return 500000
        for q in range(Rmax):
            T_in = T_in + F[q]*O(i,j,q)*X00(q)*T_(i,j,q)/soma_todas
        return T_in


    ##Definindo os pesos B1, B2, B3 conforme artigo
    B1 = 0.7
    B2 = 0.2
    B3 = 0.1


    #Criando o vetor F que armazena as frequências para cada linha -
setando inicialmente para 0
    F = []
    for i in range(Rmax):
        F.append(8)


    ##Calculando sa1, sa2 e sa3, que representam os valores de cada
Objetivo para a solução atual
    #sa1:
    def sa1():
        Soma_sa1 = 0
        for i in range(len(U)):
            for j in range(len(U)):
```

```
                if i != j:
                    Soma_sa1 = Soma_sa1 +
matriz_dij[i][j]*Tmedio(i,j)
        return Soma_sa1
    #sa2:
    def sa2():
        Soma_sa2den = 0  ##para o denominador
        Soma_sa2 = 0
        for i in range(len(U)):
            for j in range(len(U)):
                if i != j:
                    Soma_sa2den = Soma_sa2den + matriz_dij[i][j]
        for i in range(len(U)):
            for j in range(len(U)):
                if i != j:
                    Soma_sa2 = Soma_sa2 +
matriz_dij[i][j]*Tmedio(i,j)/matriz_cij_media[i][j]
        return Soma_sa2/Soma_sa2den
    #sa3:
    def sa3():
        Soma_sa3 = 0
        for i in range(Rmax):
            Soma_sa3 = Soma_sa3 + VS[i+1]
        return Soma_sa3



    def FO(): # Função Objetivo
        return B1*sa1() + B2*sa2() + B3*sa3()



    ##Solução inicial pela EMTU
    #Linhas:
```

```
    L_emtu =
[[0,20,22],[0,5,8,10,14,16,20,22],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,1
4,15,16,17,18,19,20,21,22]]


    #Frequências

    F_emtu = [8,20,15]


    #Veículos

    VS_emtu = [0,12,34,30]


    ##Gerando a matriz_cij para a EMTU


    # Calculando os tempos entre estações consecutivas para cada
linha:

    Tempos_consecutivos_emtu = []

    for linha in L_emtu:

        velocidade = V_media_est[len(linha) - 2]

        tempos_linha = Distancia_est / velocidade

        Tempos_consecutivos_emtu.append(tempos_linha.tolist())


    # Função para criar a matriz de tempos de viagem entre todas as
estações para uma linha de ônibus

    def criar_matriz_tempos_emtu(tempos_consecutivos):

        # Inicializa a matriz de tempos com zeros

        matriz_tempos_emtu = np.zeros((n, n))


        # Preenche a matriz

        for i in range(n):

            for j in range(i+1, n):  # Calcula apenas a metade
superior devido à simetria

                # Soma os tempos consecutivos entre a estação i e j

                matriz_tempos_emtu[i][j] = matriz_tempos_emtu[i][j-
1] + tempos_consecutivos[j-1] if j-1 < len(tempos_consecutivos) else
matriz_tempos_emtu[i][j-1]
```

```
                matriz_tempos_emtu[j][i] = matriz_tempos_emtu[i][j]
# Aproveita a simetria

        return matriz_tempos_emtu

    # Lista para armazenar as matrizes de tempos de todas as linhas
    matriz_cij_emtu = []

    # Calcula a matriz de tempos para cada linha
    for tempos_linha in Tempos_consecutivos_emtu:
        matriz = criar_matriz_tempos_emtu(tempos_linha)
        matriz_cij_emtu.append(matriz)

    ## Criação da função t_emtu(i, j, n), que calcula o tempo de
viagem entre i e j para a linha n, onde i e j pertencem a n ##
    def t_emtu(i, j, n):
        if i in L_emtu[n] and j in L_emtu[n]:
            auxiliar_stoptime = 0
            if i > j:
                for k in range(i, j, -1):
                    auxiliar_stoptime = auxiliar_stoptime +
Stop_time[k]
            t = matriz_cij_emtu[n][i][j] + auxiliar_stoptime
            if i < j:
                for k in range(i, j):
                    auxiliar_stoptime = auxiliar_stoptime +
Stop_time[k]
            t = matriz_cij_emtu[n][i][j] + auxiliar_stoptime

            return t
        if i not in L_emtu[n] or j not in L_emtu[n]:  ## retorna um
número grande suficiente caso i ou j não façam parte da linha n
            return 500
```

```
    ##Calculando sa1, sa2 e sa3 para a solução da EMTU

    # Criação da função T(i,j,n) que calcula o tempo de i até j com
ou sem transf.

    def T_emtu(i, j, n):

        list_transf = []  # lista auxiliar para a estação de
transferência

        estac_transf = 0  # variável auxiliar para estação de
transferência

        group_transf = []  # lista auxiliar para os tempos após
transferência

        tempo_transf = 0  # variável auxiliar para o tempo após
transferência

        indice_linha = 0  # variável auxiliar para a linha que será
transferida

        aux_check = 0  # variável auxiliar para os is e js
consecutivos

        if i in L_emtu[n] and j in L_emtu[n]:  # caso que i e j
pertencem a n

            return 1 / F_emtu[n] + t_emtu(i, j, n)

        if i in L_emtu[n] and j not in L_emtu[n]:  # caso que i
pertence mas j não

            if i < j and i != j:  # precisa olhar para estações
consecutivas e não somente i e i+1

                for s in range(len(L_emtu[n])):

                    if L_emtu[n][s] == i:

                        aux_check = s

                if L_emtu[n][-1] == i:

                    return 0

                if L_emtu[n][aux_check + 1] > j:

                    return 0

                for k in range(j-1, i, -1):

                    for q in range(3):  # Usando 3 no lugar de Rmax
diretamente
```

```
                        if k in L_emtu[q] and j in L_emtu[q] and k
in L_emtu[n]:
                            list_transf.append(k)
                estac_transf = max(list_transf)  # para a ida, temos
a última estação de transferência
                for p in range(3):  # Usando 3 no lugar de Rmax
diretamente
                    group_transf.append(t_emtu(estac_transf, j, p))
                tempo_transf = min(group_transf)
                for r in range(3):  # Usando 3 no lugar de Rmax
diretamente
                    if tempo_transf == group_transf[r]:
                        indice_linha = r
                return 1 / F_emtu[n] + t_emtu(i, estac_transf, n) +
1 / F_emtu[indice_linha] + tempo_transf
            if i < j and i + 1 == j:  # para a ida, estações
consecutivas em que j não pertence a n
                return 0
            if i > j and i != j:
                for s in range(len(L_emtu[n])):
                    if L_emtu[n][s] == i:
                        aux_check = s
                if L_emtu[n][aux_check - 1] < j:
                    return 0
                for k in range(j + 1, i):
                    for q in range(3):  # Usando 3 no lugar de Rmax
diretamente
                        if k in L_emtu[q] and j in L_emtu[q] and k
in L_emtu[n]:
                            list_transf.append(k)
                estac_transf = min(list_transf)  # para a volta,
temos a última estação de transferência
                for p in range(3):  # Usando 3 no lugar de Rmax
diretamente
                    group_transf.append(t_emtu(i,estac_transf, p))
```

```
                tempo_transf = min(group_transf)
                for r in range(3):  # Usando 3 no lugar de Rmax
diretamente
                    if tempo_transf == group_transf[r]:
                        indice_linha = r
                return 1 / F_emtu[n] + t_emtu(i, estac_transf, n) +
1 / F_emtu[indice_linha] + tempo_transf
            if i > j and i - 1 == j:  # para a volta, estações
consecutivas em que j não pertence a n
                return 0
        if i not in L_emtu[n]:
            return 0  # retorna 0 para T_(i, j, n) onde i não
pertence a n



    def X00_emtu(n):    #função que retorna 1 se a linha n está
habilitada, 0 caso contrário. Note que aqui está invertido da
formulação, para facilitar o código.
        if VS_emtu[n+1] == 0:
            return 0
        if VS_emtu[n+1] != 0:
            return 1


    def O_emtu(i,j,n):  #função que retorna 1 se da pra ir de i até
j por n, com ou sem transf. 0, caso contrário
        if T_emtu(i,j,n) == 0:
            return 0
        if T_emtu(i,j,n) != 0:
            return 1


    def Tmedio_emtu(i,j):    ##função que calcula o tempo médio de i
até j, considerando todas as possibilidades
        soma_todas = 0   #criando variável auxiliar para somar todas
as frequências habilitadas
```

```
        T_in = 0     #variável para calcular o Tmedio
        for k in range(3):
            soma_todas = soma_todas +
F_emtu[k]*O_emtu(i,j,k)*X00_emtu(k)
        for q in range(3):
            T_in = T_in +
F_emtu[q]*O_emtu(i,j,q)*X00_emtu(q)*T_emtu(i,j,q)/soma_todas
        return T_in



    ##Calculando sa1, sa2 e sa3 para a emtu, que representam os
valores de cada Objetivo para a solução atual
    #sa1:
    def sa1_emtu():
        Soma_sa1 = 0
        for i in range(len(U)):
            for j in range(len(U)):
                if i != j:
                    Soma_sa1 = Soma_sa1 +
matriz_dij[i][j]*Tmedio_emtu(i,j)
        return Soma_sa1
    #sa2:
    def sa2_emtu():
        Soma_sa2den = 0  ##para o denominador
        Soma_sa2 = 0
        for i in range(len(U)):
            for j in range(len(U)):
                if i != j:
                    Soma_sa2den = Soma_sa2den + matriz_dij[i][j]
        for i in range(len(U)):
            for j in range(len(U)):
                if i != j:
```

```
                    Soma_sa2 = Soma_sa2 +
matriz_dij[i][j]*Tmedio_emtu(i,j)/matriz_cij_media[i][j]
        return Soma_sa2/Soma_sa2den
    #sa3:
    def sa3_emtu():
        Soma_sa3 = 0
        for i in range(3):
            Soma_sa3 = Soma_sa3 + VS_emtu[i+1]
        return Soma_sa3



    def FO_emtu(): # Função Objetivo para a EMTU
        return B1*sa1_emtu() + B2*sa2_emtu() + B3*sa3_emtu()


    ##Armazenando os valores da solução inicial da EMTU de sa1, sa2,
sa3:
    Current_sa1 = sa1_emtu()
    Current_sa2 = sa2_emtu()
    Current_sa3 = sa3_emtu()


    #Definindo delta para a normalização da função objetivo
    delta = 0.05


    def FO_normalizada(): #Função objetivo normalizada
        return B1*((sa1() - Current_sa1)/(-delta*Current_sa1)) +
B2*((sa2() - Current_sa2)/(-delta*Current_sa2)) + B3*((sa3() -
Current_sa3)/(-delta*Current_sa3))


    #### Algoritmo 2 #####


    #Criando o Vetor T = Tempo de Ciclo para as linhas
```

```
    T = []


    for i in range(Rmax):
        Soma_T = 0
        #no tempo de ciclo, o ônibus para 2x por estação da linha.
        for j in range(len(L_final[i])):
            Soma_T = Soma_T + 2*Stop_time[L_final[i][j]]
        #precisa somar o tempo em trânsito (ida e volta são
simétricas):
        for k in range(len(L_final[i])-1):  #vai até o penúltimo
índice
            Soma_T = Soma_T +
2*matriz_cij[i][L_final[i][k]][L_final[i][k+1]]
        T.append(Soma_T)


    q = 0 # definindo a variável auxiliar q


    #Definindo a frequência mínima conforme caso real:
    fmin = 8


    #criando W conforme o problema real
    W = 76  ##Note que já reserva-se 6 veículos para a reserva
técnica


    #Criando o Vetor VN = número de veículos para as linhas - seta-
se inicialmente para 0
    VN = []
    for i in range(Rmax):
        VN.append(0)


    #Importando math para usar a função teto (ceil)
    import math
```

```
    ## início do algoritmo ##


    while q < Rmax:
        VN[q] = math.ceil(fmin*T[q])
        F[q] = VN[q]/T[q]
        q = q+1


    # criando a variável nvn, igual a soma de todos os VNs
    nvn = 0
    for i in range(len(VN)):
        nvn = nvn + VN[i]


    # Criando o vetor VS de tamanho Rmax +1 -> VS[0] corresponde ao
número de veículos não utilizados e o restante é igual a VN
    VS = [0]
    for i in range(len(VN)):
        VS.append(VN[i])


    #se nvn < W -> VS[0] = W - nvn, else = 0. W é o valor total da
frotal


    if nvn < W:
        VS[0] = W - nvn
    else:
        VS[0] = 0


    #seja vae a diferença entre a variável nvn e W:
    vae = nvn - W


    #Criando a variável aleatória pra ajudar no looping do vae:
    k_aleatorio = 0
    while vae > 0:
```

```
        k_aleatorio = random.randint(1,Rmax)
        VS[k_aleatorio] = VS[k_aleatorio] - 1
        vae = vae - 1

    #Criando S = (L_final,F) -> vetor que armazena as linhas e as
frequências de cada linha

    ##Avaliando S usando 21:

    Evaluate = FO_normalizada()

    #Criando a variável auxiliar m e o vetor auxiliar VP
    m = 0 #note que no nosso caso, m = 0, pois VS[1] é a primeira
linha
    VP = []

    for i in range(m+1,Rmax+1): #Cria o VP conforme o algoritmo,
para m = 1
        VP.append(i)

    #criando a variável auxiliar n aleatória para o looping abaixo
    n_aleatorio= 0
    #Criando a variável auxiliar para comparar os Evaluates
    New_Evaluate = Evaluate


    while VP:
        New_Evaluate = Evaluate
        n_aleatorio = random.choice(VP)
        VP.remove(n_aleatorio)
        #O looping a seguir aloca os veículos que estão sobrando nas
estações
```

```
        while VS[m] > 0 and New_Evaluate >= Evaluate:
            Evaluate = New_Evaluate
            VS[m] = VS[m] - 1
            VS[n_aleatorio] = VS[n_aleatorio] + 1
            F[n_aleatorio-1] = VS[n_aleatorio]/T[n_aleatorio-1]
            New_Evaluate = FO_normalizada()
            ##Precisa reverter as mudanças no último estado, pois o
New Evaluate parou de ganhar
            ## Caso o looping acabe pois VS[m] = 0, o if abaixo não
é satisfeito
        if New_Evaluate < Evaluate:
            VS[m] = VS[m] + 1
            VS[n_aleatorio] = VS[n_aleatorio] - 1
            F[n_aleatorio-1] = VS[n_aleatorio]/T[n_aleatorio-1]
        Evaluate = FO_normalizada() ##Guarda o melhor valor de
FO_normalizada()
        New_Evaluate = Evaluate #Atualiza o New_Evaluate também
        #O looping a seguir remove os veículos de estações que esta
remoção não causa impacto na FO
        while VS[n_aleatorio] > 0 and New_Evaluate >= Evaluate:
            Evaluate = New_Evaluate
            VS[n_aleatorio] = VS[n_aleatorio] - 1
            VS[m] = VS[m] + 1
            F[n_aleatorio-1] = VS[n_aleatorio]/T[n_aleatorio-1]
#note que subtrai-se um pois VS[0] é o número de veículos sem uso
            New_Evaluate = FO_normalizada()
        if New_Evaluate < Evaluate:
            VS[n_aleatorio] = VS[n_aleatorio] + 1
            VS[m] = VS[m] - 1
            F[n_aleatorio-1] = VS[n_aleatorio]/T[n_aleatorio-1]


        Evaluate = FO_normalizada()
        New_Evaluate = Evaluate
```

```
    #Fazendo o mesmo procedimento porém m>=1
    #criando a variável auxiliar m aleatória para o looping abaixo
    m_aleatorio = 0
    #Reatribuindo a lista nula a VP

    for m1 in range(1,Rmax):
        for i in range(m1+1,Rmax+1):
            VP.append(i)
        while VP:
            m_aleatorio = random.choice(VP)
            VP.remove(m_aleatorio)
            while VS[m1] > 0 and L_final[m1-1] !=
L_final[m_aleatorio-1] and New_Evaluate >= Evaluate:
                Evaluate = New_Evaluate
                VS[m1] = VS[m1] - 1
                VS[m_aleatorio] = VS[m_aleatorio] + 1
                F[m_aleatorio-1] = VS[m_aleatorio]/T[m_aleatorio-1]
                F[m1-1] = VS[m1]/T[m1-1]
                New_Evaluate = FO_normalizada()
            if New_Evaluate < Evaluate:
                VS[m1] = VS[m1] + 1
                VS[m_aleatorio] = VS[m_aleatorio] - 1
                F[m_aleatorio-1] = VS[m_aleatorio]/T[m_aleatorio-1]
                F[m1-1] = VS[m1]/T[m1-1]
            Evaluate = FO_normalizada()
            New_Evaluate = Evaluate
            while VS[m_aleatorio] > 0 and L_final[m1-1] !=
L_final[m_aleatorio-1] and New_Evaluate >= Evaluate:
                Evaluate = New_Evaluate
                VS[m_aleatorio] = VS[m_aleatorio] - 1
                VS[m1] = VS[m1] + 1
```

```
                F[m_aleatorio - 1] = VS[m_aleatorio]/T[m_aleatorio-
1]
                F[m1-1] = VS[m1]/T[m1-1]
                New_Evaluate = FO_normalizada()
            if New_Evaluate < Evaluate:
                VS[m_aleatorio] = VS[m_aleatorio] + 1
                VS[m1] = VS[m1] - 1
                F[m_aleatorio - 1] = VS[m_aleatorio]/T[m_aleatorio -
1]
                F[m1-1] = VS[m1]/T[m1-1]
            Evaluate = FO_normalizada()
            New_Evaluate = Evaluate




    # Função para consolidar linhas duplicadas e somar os veículos
    def consolidar_linhas_e_veiculos(linhas, veiculos):
        veiculos_por_linha = {}
        linha_indices = {}  # Para manter o índice da primeira
ocorrência da linha


        # Mapear linhas para a soma dos veículos
        for index, linha in enumerate(linhas):
            chave = tuple(linha)  # Converter a lista em tupla para
usá-la como chave do dicionário
            if chave in veiculos_por_linha:
                veiculos_por_linha[chave] += veiculos[index + 1]  #
index + 1 pois VS[0] é para veículos não usados
            else:
                veiculos_por_linha[chave] = veiculos[index + 1]
                linha_indices[chave] = index
```

```
        # Atualizando VS com os novos valores de veículos,
considerando apenas as linhas únicas
        novos_veiculos = [veiculos[0]]  # Começa-se com os veículos
não usados
        novas_linhas = []
        indices_para_remover = list(range(1, len(linhas) + 1))  #
Índices de VS a serem removidos, exceto o primeiro (0)

        for linha, total_veiculos in veiculos_por_linha.items():
            novas_linhas.append(list(linha))
            novos_veiculos.append(total_veiculos)
            # Mantém apenas o índice da primeira ocorrência
            indices_para_remover.remove(linha_indices[linha] + 1)

        return novas_linhas, novos_veiculos, indices_para_remover

    # Aplicar a função de consolidação
    L_final, VS, indices_para_remover =
consolidar_linhas_e_veiculos(L_final, VS)

    # Atualização das listas de frequência e tempo de ciclo
removendo os elementos duplicados
    def remover_elementos(indices_para_remover, frequencias,
tempos):
        frequencias_atualizadas = [frequencias[i] for i in
range(len(frequencias)) if (i + 1) not in indices_para_remover]
        tempos_atualizados = [tempos[i] for i in range(len(tempos))
if (i + 1) not in indices_para_remover]
        return frequencias_atualizadas, tempos_atualizados

    # Aplicar a função de remoção
    F, T = remover_elementos(indices_para_remover, F, T)

    #Atualizando Rmax:
```

```
    Rmax = len(L_final)

    #Atualizando as velocidades e tempos:
    #definindo os V[i]s:

    for i in range(Rmax):
        V[i] = V_media_est[len(L_final[i])-2]

    # Calculando os tempos entre estações consecutivas para cada
linha:
    Tempos_consecutivos = []
    for linha in L_final:
        velocidade = V_media_est[len(linha) - 2]
        tempos_linha = Distancia_est / velocidade
        Tempos_consecutivos.append(tempos_linha.tolist())



    # Lista para armazenar as matrizes de tempos de todas as linhas
    matriz_cij = []

    # Calcula a matriz de tempos para cada linha
    for tempos_linha in Tempos_consecutivos:
        matriz = criar_matriz_tempos(tempos_linha)
        matriz_cij.append(matriz)

    #Atualizando as frequências:
    for i in range(1,len(F)+1):
        F[i-1] = VS[i]/T[i-1]

    # Função para verificar se um par está presente em alguma lista
de L_final com frequência diferente de 0
    def par_presente_com_frequencia(L_final, F, i, j):
```

```
        for index, linha in enumerate(L_final):
            if F[index] != 0 and i in linha and j in linha:
                return True
        return False
    ##Cálculo do Tempo de Ciclo para a linha que passa por todas as
estações:
    T_ciclo_max = 0

    #no tempo de ciclo, o ônibus para 2x por estação da linha.
    for j in range(len(L_emtu[2])):
        T_ciclo_max = T_ciclo_max + 2*Stop_time[L_emtu[2][j]]
    #precisa somar o tempo em trânsito (ida e volta são simétricas):
    for k in range(len(L_emtu[2])-1):  #vai até o penúltimo índice
        T_ciclo_max = T_ciclo_max +
2*matriz_cij_emtu[2][L_emtu[2][k]][L_emtu[2][k+1]]



    # Criar o flag de verificação
    Descarta_par = False  # Assume que o flag é False inicialmente


    for i in U:
        for j in U:
            if i != j and not par_presente_com_frequencia(L_final,
F, i, j):
                Descarta_par = True
                break
        if Descarta_par:
            break
    if sa1() > 500000:        ##Garante que estamos descartando as
soluções que não possuem rotas de i até j (soma_todas = 0)
        Descarta_par == True
```

```
    #Descarta as soluções que a frequência é menor que a mínima ou
que a restrição acima foi violada
    Descarta_freq = False
    for i in range(Rmax):
        if F[i] > 0 and F[i] < 8:
            Descarta_freq = True

    #Inicializa um dicionário para armazenar as soluções
    results = {}
    if Descarta_freq == False and Descarta_par == False:
        results["FO:"] = FO_normalizada()
        results["sa1:"] = sa1()
        results["sa2:"] = sa2()
        results["sa3:"] = sa3()
        results["L:"] = L_final
        results["F:"] = F
        results["V:"] = VS
    else:
        results["Solução Inviável"] = True

    return results



#lista para armazenar todas as soluções
all_results = []
for i in range(30):
    print(i+1)
    result = main()
    all_results.append(result)

##Cria um DataFrame para os resultados:
```

```
import pandas as pd
df = pd.DataFrame(all_results)

#Salva em um excel:
df.to_excel("results.xlsx",index = True)
```