UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA CIVIL

Daniel Rodrigues Acosta

DEFINIÇÃO DE ESCALA DE TRIPULAÇÃO DE TRANSPORTE COLETIVO UTILIZANDO UM ALGORITMO CONSTRUTIVO-EVOLUTIVO

Porto Alegre
dezembro 2019

DANIEL RODRIGUES ACOSTA

DEFINIÇÃO DE ESCALA DE TRIPULAÇÃO DE TRANSPORTE COLETIVO UTILIZANDO UM ALGORITMO CONSTRUTIVO-EVOLUTIVO

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Civil da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Engenheiro Civil

Orientador: Fernando Dutra Michel

Porto Alegre
dezembro 2019

DANIEL RODRIGUES ACOSTA

DEFINIÇÃO DE ESCALA DE TRIPULAÇÃO DE TRANSPORTE COLETIVO UTILIZANDO UM ALGORITMO CONSTRUTIVO-EVOLUTIVO

Este Trabalho de Diplomação foi julgado adequado como pré-requisito para a obtenção do título de Engenheiro Civil e aprovado pela banca examinadora e, em sua forma final, pelo Professor Orientador.

Porto Alegre, dezembro de 2019

Prof. Fernando Dutra Michel
Dr. pela UFRGS
Orientador

BANCA EXAMINADORA

Eng. Maria Cristina Molina Ladeira
(UFRGS)
Me. pela UFRGS

Eng. Douglas Zechin
(UFRGS)
Me. pela UFRGS

Dedico esse trabalho a meus pais.

AGRADECIMENTOS

Agradeço à minha família por me oferecer mais do que posso conceber e expressar em palavras.

Agradeço aos meus melhores amigos, que me aguentam.

Muito obrigado aos meus professores: aos que me ensinaram da UFRGS, aos da rede pública da minha cidade, aos do IFSul, à minha mãe, às suas companheiras do CPERS. Muitos não imaginam o quanto me marcaram positivamente, mas todos sabem ou deveriam saber que são fundamentais pra humanidade. Obrigado por me instigarem desde cedo com o encanto por entender, por trazer sentido ao mundo, por lutar acima de tudo pelo que é certo. E por me inspirarem a de algum modo seguir seus passos.

Obrigado a todos os lugares diferentes por onde passei: Relações Internacionais, Arquitetura, Mandarim, Karatê, Japonês, Canto Coral, Teatro, Economia, porque numa caminhada em linha reta perdemos paisagens demais. E conhecimento, e visão.

Agradeço aos que me acolheram tão bem no estágio no DMAE, na iniciação científica no NETT e no Lastran. Obrigado por me mostrarem que eu conseguia fazer bem mais do que imaginava, e que o importante é nunca esquecer que estou lidando com pessoas.

Agradeço a todos que conheci através dos Encontros de Engenharia Popular e Solidária, por me darem vontade de continuar nessa graduação e por me darem a certeza de que é meu direito ser o engenheiro que eu quiser ser. O resto é questão de coragem.

RESUMO

A queda na demanda por transporte coletivo por ônibus observado no Brasil nas últimas décadas, aliada a outros fatores, tem causado o aumento da tarifa desse modal. A sociedade brasileira tem tratado essa pauta como prioridade e tem se posicionado contra esses aumentos firmemente, tendo em vista o seu impacto no cotidiano da população, especialmente sua parcela de renda mais baixa. Para atender às demandas de barateamento é necessário encontrar pontos onde a operação do modal possa ser aperfeiçoada, para que possa haver diminuição nos custos. Uma das lacunas observadas no transporte coletivo por ônibus na cidade de Porto Alegre diz respeito ao Problema de Programação de Tripulações (PPT), que é uma etapa do planejamento do sistema de transporte coletivo. Nesse âmbito o PPT consiste em gerar jornadas de trabalho a serem cumpridas por motoristas e cobradores, utilizando como dado de entrada a Tabela Horária de uma linha de ônibus. Essa jornada, a fim de minimizar custos de mão de obra, deve ter o menor tempo ocioso possível entre as viagens. Além disso, deve atender às particularidades operacionais e trabalhistas do setor, como compatibilizar terminais das viagens e atribuir intervalos para descanso. Na literatura da área, o PPT tem sido resolvido utilizando tanto métodos computacionais exatos quanto aproximativos. Na cidade de Porto Alegre algumas empresas resolvem o PPT sem nenhum tipo de sistematização, o que pode gerar perda de eficiência. Esse trabalho se propôs a encontrar uma solução para o PPT utilizando um Algoritmo Construtivo-Evolutivo, que se baseia no Algoritmo Construtivo AAO* e no Algoritmo Genético. Para isso, foi criada e executada uma rotina de programação na linguagem Python. Usando a Tabela Horária de uma linha de ônibus de Porto Alegre, foi encontrada uma solução que atende a todas as 301 viagens da linha e a todos os requisitos de operação e de legislação trabalhista.

Problema de Programação de Tripulações, Algoritmo Construtivo-Evolutivo, Algoritmo Genético, Transporte Coletivo

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

PPT – Problema de Programação de Tripulações

ACE – Algoritmo Construtivo-Evolutivo

AG – Algoritmo Genético

TH – Tabela Horária

PRTC – Projeto de Redes de Transporte Coletivo

PPV – Problema de Programação de Veículos

PRT – Problema de Rodízio de Tripulações

PPVT – Problema de Programação Integrada de Veículos e Tripulações

FC – Função Custo

CA – Custo Atual de uma solução

CM – Custo Meta de uma solução

BC – Viagem Bairro-Centro

CB – Viagem Centro-Bairro

SUMÁRIO

1 INTRODUÇÃO

A proporção de habitantes em área urbana no Brasil em relação aos que habitam a área rural passou de 75% em 1991 para 84% em 2012, o que levou à diversificação das atividades e, por consequência, dos padrões de viagens (OLIVEIRA et al., 2013). O uso de transporte coletivo é preferível para atender a essa demanda, visto que, em comparação ao transporte particular, causa menos congestionamentos e representa um uso mais eficiente de energia. Ainda assim, no Brasil tem sido observada nas últimas décadas uma forte queda no uso de transporte público, consequência das dificuldades desse sistema em manter uma boa posição de mercado (GUEDES et al., 2019).

A queda na demanda pelo transporte coletivo urbano por ônibus tem causado o aumento de suas tarifas em um ritmo sem precedentes, visto que o sistema é geralmente sustentado pelos passageiros pagantes (GUEDES et al., 2019).

A tarifa do transporte público é uma pauta que mobiliza a sociedade, sendo uma das motivações dos protestos que ocorreram por todo o Brasil em junho de 2013 (BARBOSA; KERBAUY, 2016). Para atender à justa pressão de barateamento, tem se tornado cada vez mais necessário encontrar pontos que permitam a melhoria da operação desse sistema. O investimento em resolver essas lacunas é uma oportunidade para a redução de custos que impactaria positivamente na tarifa.

Um dos pontos passíveis de melhoria é o gerenciamento da mão-de-obra. Na cidade de Porto Alegre, as despesas com pessoal representam 48,84% da tarifa do modal ônibus (REIS; COSTA, 2017).

A área que estuda esses problemas é a engenharia de transportes, ramo responsável pelo planejamento, construção, operação e manutenção de infraestrutura para a mobilidade de cargas e pessoas. Na literatura da área, o estudo da otimização da escala de trabalho de motoristas e cobradores é conhecido como o Problema de Programação de Tripulações (PPT).

O PPT consiste em distribuir as viagens de uma Tabela Horária (TH) em escalas de trabalho a serem cumpridas pelos motoristas e cobradores. Nessa solução, é necessário também minimizar características como o tempo em que motoristas estão ociosos. Devido ao fato de

as tabelas horárias serem irregulares e por ser necessário atender a inúmeras particularidades como requisitos trabalhistas e operacionais, a resolução do PPT não é trivial.

Atualmente em muitas empresas esse procedimento não é sistematizado: utiliza-se escalas feitas manualmente, o que gera uma organização menos eficiente, com maior ocorrência de tempos ociosos. O presente trabalho busca propor uma sistematização para essa a resolução do PPT de acordo com as particularidades do sistema de transporte coletivo por ônibus da cidade de Porto Alegre.

1.1 DESCRIÇÃO DO PROBLEMA

Este trabalho busca obter uma solução para o Problema de Programação de Tripulações (PPT) para a Tabela Horária (TH) de uma linha de ônibus da cidade de Porto Alegre, considerando todas as particularidades trabalhistas e operacionais da cidade.

Uma solução do problema consiste em viagens agrupadas em serviços. Um serviço corresponde a um período de jornada de trabalho em um veículo, a ser cumprido por uma tripulação formada por um motorista e um cobrador. Conforme os acordos trabalhistas em vigor na cidade de Porto Alegre, o serviço não pode ultrapassar o período de sete horas e meia e deve contar com um intervalo de descanso de no mínimo meia hora (SINDICATO DAS EMPRESAS DE ÔNIBUS DE PORTO ALEGRE, 2019).

Busca-se uma solução que tenha a menor quantidade possível de serviços e de tempos ociosos dentro destes serviços (tempos entre viagens ou tempo restante até o fim das sete horas e meia de jornada). Minimizando a quantidade de serviços e de tempos ociosos, o custo de operação é reduzido e isto contribui para a diminuição da tarifa, tornando-a mais barata para este modal.

1.2 SOLUÇÃO PROPOSTA

Esse trabalho se propõe a encontrar uma solução factível para o PPT, com o menor custo possível, através de um modelo computacional heurístico, que é um método cujo objetivo não é atingir a solução ótima, e sim atingir uma solução suficientemente próximas da ótima utilizando recursos computacionais menores.

O modelo heurístico usado é o Algoritmo Construtivo-Evolutivo (ACE), utilizado nos trabalhos de Elizondo et al. (2009) e Fuentes (2010). Este algoritmo é baseado no método do Algoritmo Genético (AG) e pode ocorrer em duas fases. A primeira fase corresponde à construção de uma solução a partir do zero, adicionando viagens aos poucos e aplicando nessa solução os operadores do AG. Quando todas as viagens da Tabela Horária já constam na solução e todas as restrições de intervalo de descanso foram satisfeitas, se encerra a fase construtiva e pode se iniciar a segunda fase, que consiste no AG puro. Nessa fase o objetivo é encontrar uma solução que tenha um custo menor que a encontrada na fase construtiva.

1.3 OBJETIVOS E DELIMITAÇÕES DO TRABALHO

Os objetivos deste trabalho estão divididos em objetivo geral e objetivos específicos.

1.3.1 Objetivo Geral

Implementar e executar um programa para encontrar uma solução viável e de menor custo para o PPT em uma linha do transporte coletivo por ônibus de Porto Alegre através do método do Algoritmo Construtivo-Evolutivo (ACE), utilizando-se da linguagem de programação Python.

1.3.2 Objetivos Específicos

São objetivos específicos desse trabalho:

a) Compreender o problema do PPT;

b) Compreender os métodos do ACE e do AG;

c) Compreender a linguagem de programação Python;

d) Criar e executar um algoritmo na linguagem Python capaz de resolver o problema utilizando o método do ACE;

e) Executar o Algoritmo desenvolvido para dados reais de uma linha de transporte coletivo da cidade de Porto Alegre.

1.3.3 Delimitações

Esse trabalho se aplica ao âmbito do transporte coletivo em modal ônibus, na cidade de Porto Alegre. A Tabela Horária de cada linha é pré-determinada e usada como dado de entrada do problema. As exigências trabalhistas a serem atendidas são referentes à legislação e acordos trabalhistas aplicáveis à cidade em questão, bem como as exigências operacionais.

1.4 FERRAMENTAS UTILIZADAS

O trabalho utilizou para a resolução do problema proposto o método formulado por Elizondo et al. (2009) e Fuentes (2010), que resolvem o PPT usando o ACE. O presente trabalho utilizará dados de tabelas horárias de uma linha de ônibus e aplicará o ACE para obter escalas de tripulação.

A aplicação computacional foi construída utilizando a linguagem de programação *Python*, através do ambiente de desenvolvimento (IDE) *Spyder*. Foi utilizado um computador com sistema operacional *Windows 10* de 64 bits, processador *Intel Core i5* de 1.70GHz, Disco SSD 480GB 6GB/s e Memória RAM de 8 GB.

1.5 RESULTADOS OBTIDOS

A partir da Tabela Horária (TH) da Linha 165 (Cohab), o algoritmo construído nesse trabalho foi capaz de gerar dez diferentes escalas de tripulações factíveis, que atendem a todas as viagens da TH, com intervalo de descanso atribuído e pouco tempo ocioso. Entre as dez opções factíveis, a mais eficiente distribuiu as 310 viagens da TH em 45 jornadas de trabalho.

1.6 ESTRUTURA DO TRABALHO

Esse documento é composto da presente Introdução, o Capítulo 2, que que apresenta o referencial teórico utilizado, o Capítulo 3, que explica detalhadamente o algoritmo proposto, o Capítulo 4, com a discussão dos resultados da aplicação desse algoritmo a dados reais e por fim o Capítulo 5 com as considerações finais sobre o trabalho.

2 REFERENCIAL TEÓRICO

A revisão que serve de base para esse trabalho teve como objetivo estabelecer o estado da arte em três temáticas: Problema de Programação de Tripulações (PPT), Algoritmo Construtivo-Evolutivo (ACE) e Algoritmo Genético (AG). A primeira se insere no campo de Sistema de Transporte Coletivo e as últimas se inserem no campo de Métodos de Resolução Computacional.

2.1 SISTEMA DE TRANSPORTE COLETIVO

O Sistema de Transporte Coletivo é composto por todo e qualquer modo de transporte que permita o deslocamento coletivo de passageiros, a fim de permitir a realização das atividades diárias de cada indivíduo. Como exemplos de sistemas de transporte coletivo de passageiros, pode-se citar o ferroviário, o metroviário e o rodoviário (REIS, 2008).

Um dos desafios do planejamento do sistema de transporte coletivo de uma região é a determinação de linhas e suas respectivas tabelas horárias para o atendimento da demanda de passageiros. Neste trabalho chamamos esse problema de Projeto de Redes de Transporte Coletivo (PRTC) (*Bus Network Design*). O processo de resolução do PRTC pode ser dividido em seis etapas (adaptado de IBARRA-ROJAS et al., 2015):

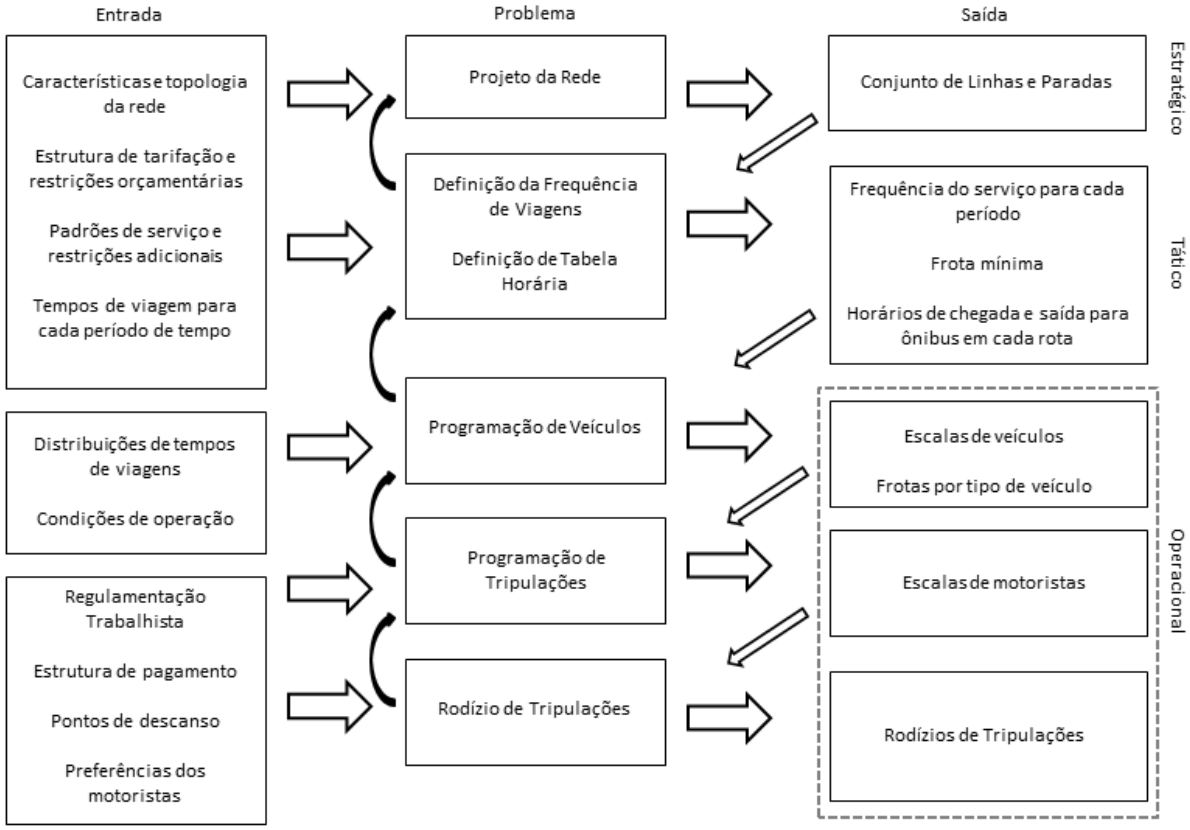
1. Projeto da Rede (*Network Design*) - definir o traçado das linhas dos ônibus e características relacionadas como distância entre paradas e tipo de frota, visando minimizar custos de operação;
2. Definição da Frequência de Viagens (*Frequency Setting*) - estipular o período de operação da linha e o número de viagens a serem realizadas por hora com base na análise da demanda e dos horários de pico;
3. Definição de Tabela Horária (TH) (*Network Timetabling*) - estabelecer os horários de saída dos veículos e duração das viagens, visando atender à frequência definida para a linha, facilitar a transferência de passageiros e minimizar o tempo de espera;
4. Problema de Programação de Veículos (PPV) (*Vehicle Scheduling Problem*) - atribuir as viagens a veículos de modo a cobrir toda a Tabela Horária, buscando minimizar custos relacionados ao uso da frota;
5. Problema de Programação de Tripulações (PPT) (*Driver Scheduling Problem*) - criar jornadas de trabalho diárias a serem cumpridas por motoristas e

cobradores. As jornadas devem cobrir todas as viagens da Tabela Horária e minimizar custos oriundos de mão-de-obra, atendendo às regulamentações trabalhistas;

6. Problema de Rodízio de Tripulações (PRT) (*Driver Rostering Problem*) - distribuir as jornadas diárias à tripulação de modo que sejam atendidas suas folgas e férias em um dado horizonte de tempo (semanas, meses, etc.).

Esta divisão não corresponde a uma real separação das etapas e sim a uma simplificação consagrada no ramo, tendo em vista a alta complexidade de solucionar simultaneamente tais problemas. As entradas e saídas das etapas do PRTC de acordo com Ibarra-Rojas et al. (2015) podem ser vistas na Figura 1.

Figura 1 - Interação entre os estágios do processo de planejamento



(fonte: adaptado de IBARRA-ROJAS et al., 2015)

Ainda que a abordagem básica seja a sequencial, com o refinamento dos estudos e a maior disponibilidade de recursos computacionais, abordagens integradas têm sido cada vez mais utilizadas, das quais a mais relevante é a resolução integrada das etapas 4 e 5, denominada Problema da Programação Integrada de Veículos e Tripulações (PPVT).

Mesquita e Paias (2008) propõem o uso de uma técnica de programação linear chamada Partição de Conjuntos para a resolução do PPVT. Pelo ponto de vista do problema prático, essa formulação leva em conta múltiplos depósitos para os veículos e utiliza a noção de viagens mortas (viagens de ônibus cuja única função é levar a tripulação de um terminal ou depósito a outro, sem carregar passageiros). Esse foi o modelo matemático no qual se baseou Fuentes (2010), que por sua vez adicionou a manipulação de frota heterogênea com base na demanda.

O trabalho de Simões (2009) realiza uma comparação entre a abordagem integrada PPVT e as abordagens separadas e sequenciais do PPV e PPT, utilizando o Método de Busca Local Iterada. Executa as diferentes abordagens para o mesmo grupo de dados reais de uma empresa de transporte público da cidade de Belo Horizonte, comparando as soluções obtidas e confirmando a vantagem da abordagem integrada.

Mesquita et al. (2013), que resolvem de maneira integrada o PPV, o PPT e o PRT para dados de companhias de ônibus de Portugal, formulando o problema através de um modelo de programação linear e resolvendo-o com o uso de uma heurística baseada na Decomposição de Benders. Os resultados trazidos por esta implementação também confirmam a vantagem das abordagens integradas quando há recursos computacionais adequados.

Guedes et al. (2019) resolvem o PPV levando em conta a diminuição na demanda no modal ônibus observada nas últimas décadas no Brasil, realizando uma otimização que permite o ajuste de uma frota heterogênea (escolha entre ônibus simples ou biarticulados) de acordo com essa diminuição. A abordagem utilizou dados de companhias de ônibus da região sul do país e seus resultados permitem reduzir os custos na programação de veículos.

Moreno et al. (2019a) resolvem o PPT para um sistema de ônibus de trânsito rápido (BRT) em uma heurística de duas fases. Primeiramente é utilizado um algoritmo recursivo baseado em *Dynamic Scheduling*. No segundo estágio, é utilizado um Algoritmo de Acoplamento baseado em Teoria dos Grafos, classificado como um algoritmo construtivo. O estudo foi aplicado a um sistema de trânsito em massa na Colômbia e permitiu a diminuição de quantidade de motoristas necessários para cobrir a TH.

O presente trabalho busca resolver o PPT tendo como dados fixos de entrada as etapas anteriores. Parte de um cenário com rotas, paradas, frequência e tabelas horárias já estabelecidas e tem como objetivo atribuir viagens a tripulações.

O Problema de Programação de Tripulações (PPT) consiste em agrupar viagens de uma Tabela Horária em jornadas de trabalho de modo que possam ser atribuídas a tripulações, minimizando custos e atendendo aos critérios trabalhistas e de operação. A natureza de tais critérios tem um efeito profundo na facilidade com a qual uma programação de tripulações pode ser processada por um computador. (WREN; ROUSSEAU, 1995)

Na abordagem básica sequencial são definidas as tripulações logo depois dos veículos. Um dos métodos mais usados é o agrupamento de várias viagens em um bloco que será atendido por um único veículo, iniciando por uma viagem *pull-out* (quando um veículo sai de sua garagem) e terminando com uma viagem *pull-in* ou "recolhe" (quando o veículo volta para a garagem). Depois de definidos os blocos, estes são então atribuídos a tripulações, mas não necessariamente correspondem a uma jornada completa de tripulação, posto que uma jornada pode ser formada por alguns blocos. (WREN; ROUSSEAU, 1995)

Elizondo et al. (2009) resolveram apenas o PPT e utilizou uma abordagem construtivo-evolutiva para sua solução, que será detalhada na próxima seção. Foram utilizadas viagens individuais, ou seja, não agrupadas em blocos. Fuentes (2010) se baseou no trabalho acima, utilizando a também a abordagem construtivo-evolutiva e viagens individuais, porém para o modal ônibus. Atribuiu as viagens aos veículos posteriormente, o que a difere das abordagens mais tradicionais.

O presente trabalho buscará uma solução apenas para o PPT no modal ônibus, utilizando como base os trabalhos de Elizondo et al. (2009) e Fuentes (2010) e atendendo às restrições trabalhistas e operacionais da cidade de Porto Alegre.

2.2 MÉTODOS DE RESOLUÇÃO COMPUTACIONAL

Sendo um problema de natureza combinatória e tendo sua complexidade computacional classificada como NP-difícil (BERTOSSI; CARRARESI; GALLO, 1987), o maior desafio do PPT é a grande quantidade de escalas possíveis dentro do espaço de soluções que atendem às restrições do problema. Encontrar a solução ótima cobrindo exaustivamente o espaço de soluções se torna impraticável à medida que aumenta o tamanho do espaço de soluções. Existem diferentes métodos para superar esse desafio, podendo ser divididos em exatos e heurísticos.

Os métodos exatos abordam os problemas com um tratamento matemático e computacional bastante complexo, explorando com ferramental analítico a estrutura do problema a fim de restringir o espaço de soluções. Desde o desenvolvimento das bases teóricas do PPT sabe-se que o problema pode ser formulado matematicamente através de modelos exatos de Programação Linear Inteira, porém a tecnologia da época não permitia a solução de tais modelos em qualquer escala realista, e até o final dos anos 1970 a maior parte da pesquisa se concentrou em desenvolver soluções heurísticas. Esses métodos eram em algumas vezes tentativas de simular os processos adotados por profissionais que criavam manualmente as programações de tripulações. (WREN; ROUSSEAU, 1995)

Entre os métodos exatos mais utilizados figuram técnicas como geração de colunas (BARBOSA; RESPÍCIO; ALVELOS, 2003), relaxação lagrangiana (CARRARESI; NONATO; GIRARDI, 2011) (KANG; CHEN; MENG, 2019) e *branch-and-price* (HORVÁTH; KIS, 2019).

Como alternativa aos métodos exatos, as heurísticas, também chamadas métodos aproximativos, procuram uma solução dentro do espaço de soluções de modo aleatório. Com calibração adequada, uma heurística converge mais rapidamente para uma boa solução do que os algoritmos exatos, ainda que não garanta que seja encontrada a solução ótima (BLUM; ROLI, 2003).

Dentre as meta-heurísticas existe outra gama de abordagens para a resolução do PPT como Busca Tabu (CHEN; NIU, 2012) (WANG et al., 2014), Busca de Vizinhança Variável (*Variable Neighborhood Search* – VNS) (SILVA; PRATES, 2014) (CEDER et al., 2016) (REIS, 2008), além de vários métodos com inspiração em fenômenos da natureza como Têmpera Simulada (*Simulated Annealing*) (CIANCIO et al., 2018) e o utilizado neste trabalho, o Algoritmo Construtivo-Evolutivo, que tem como base o Algoritmo Genético (DIAS; DE SOUSA; CUNHA, 2002) (SONG et al., 2015).

A pesquisa na área desde os anos 1970 se voltou também à produção de combinações de heurísticas e métodos exatos (WREN; ROUSSEAU, 1995), abordagem utilizada até os dias de hoje. Ciancio et al. (2018) criam uma solução inicial para o PPVT utilizando algoritmo guloso e então refina a solução utilizando a meta-heurística do Têmpera Simulada. Uvaraja, Lee e Nor (2019) resolvem o PPVT com um híbrido da meta-heurística da Busca Tabu Múltipla e uma abordagem exata de Cobertura de Conjuntos Multiobjetiva. Guo, Wang e Zuo

(2019) utilizam um método híbrido de Geração de Colunas e Algoritmo Genético para resolver o PPV.

Lourenço, Paixão e Portugal (2003) resolveram o PPT usando uma abordagem híbrida com o AG e Busca Tabu. O estudo se propôs a ser incorporado no Sistema de Planejamento de Transportes GIST como uma ferramenta que auxilie nos processos de decisão. Para isso, em vez das tradicionais otimizações com uma única função objetiva, utilizou uma abordagem multiobjetiva, buscando compatibilizar as funções objetivas que conflitam entre si. O trabalho cumpriu este objetivo e é usado na prática por várias companhias.

Prata (2015) resolve o PPVT usando uma abordagem híbrida com GRASP (*Greedy Randomized Adaptive Search Procedure*) e AG, aplicando o método a dados do sistema de transporte coletivo de Fortaleza. Obteve melhoria computacional frente a outros métodos.

Entre os avanços mais recentes desse campo está o estabelecimento das hiper-heurísticas: a aplicação de heurísticas de nível mais alto sobre heurísticas de nível mais baixo, generalizando a possibilidade de uso desses métodos e comparando resultados entre uma e outra (BURKE et al., 2006). Li et al. (2015) propõem um método de geração de colunas aliado a uma abordagem hiper-heurística para encontrar uma solução quase ótima do PPT.

O Algoritmo Genético vem sendo usado e aperfeiçoado desde seu surgimento na década de 1960. Segundo a revisão de Johar, Jain e Garg (2016), a mecânica básica do Algoritmo Genético (AG) envolve gerar uma população, aplicar operadores genéticos de mutação e cruzamento e realizar a seleção da população de acordo com uma função de desempenho (*fitness*). Os indivíduos da população que forem aprovados pela função de desempenho sobreviverão para gerar outros indivíduos na geração seguinte. Pela analogia com a seleção natural, os sobreviventes passarão seu código genético adiante.

Dias, de Sousa e Cunha (2002) empregam o AG no PPV, usando formulações matemáticas de Cobertura e Partição de Conjuntos. Aplicam o método a dados reais de companhias de ônibus e de aviões, utilizando a abordagem de blocos de viagens. Concluem a vantagem do método para a obtenção de soluções suficientemente próximas da solução ótima com maior rapidez.

Song et al. (2015) resolvem o PPT através do AG, também utilizando uma abordagem de viagens em blocos e propondo um método mais específico dentro do AG, a Recombinação de Genes. Aplicam o método a dados de uma companhia de ônibus de Pequim, mostrando que a

abordagem se torna cada vez mais vantajosa em relação a outras abordagens à medida que cresce o número de viagens do problema a ser resolvido.

Cao e Ceder (2019) utilizam o AG integrado com o método exato da Iteração Variável Binária para resolver o PPV para uma linha de ônibus sem motoristas, de modo integrado à definição da Tabela Horária (TH). Utiliza-se de dados de demanda em tempo real e da tática operacional *skip-stop,* que consiste em permitir que um veículo atrasado não pare em algumas paradas. A formulação matemática é baseada no conceito de Função de Déficit (FD). O modelo é testado em um estudo de caso em Auckland, Nova Zelândia, trazendo resultados mais satisfatórios que sem a tática *skip-stop*. O trabalho mostra que a abordagem do modelo FD com o AG pode contribuir para pesquisas futuras de transporte autônomo de passageiros.

Moreno et al. (2019b) resolvem o PPV utilizando uma abordagem híbrida construtiva, inicializando o método com três procedimentos construtivos, aplicando o AG e depois refinando a solução usando modelos de Partição de Conjuntos, resolvido em um software de Programação Linear. Aplicando o modelo a dados de sistemas de trânsito colombianos, foi possível reduzir a frota necessária para o PPV.

O AG é um método interessante para a resolução do PPT porque não impõe quaisquer restrições particulares na função objetivo, permitindo exigir que a solução tenha características mais complexas, o que é geralmente difícil de manejar através de algoritmos tradicionais (DIAS; DE SOUSA; CUNHA, 2002). Assim, o sucesso do algoritmo depende menos de como a solução será criada e mais de como ela será avaliada depois de sua criação, sendo a prioridade dessa abordagem uma adequada definição da função de desempenho.

Esse trabalho utiliza como base Elizondo et al. (2009) e Fuentes (2010), que utilizaram uma abordagem chamada Algoritmo Construtivo-Evolutivo (ACE) (PALMA, 1997) uma união do AG com um algoritmo construtivo baseado na teoria de grafos. Nesta abordagem, o AG funciona como uma meta-heurística do grupo dos métodos evolutivos, e tem como base abstrata o uso de populações de indivíduos que cruzam entre si, sofrem mutações e passam por uma seleção com base em uma função fitness.

O Algoritmo Construtivo utilizado por Elizondo et al. (2009) como base para o ACE será detalhado na próxima seção. A implementação do AG será detalhada no capítulo 3, onde é apresentado o algoritmo proposto para esse trabalho.

2.3 ALGORITMO CONSTRUTIVO

De acordo com Elizondo et al. (2009), o problema do PPT pode ser representado por um grafo, como definido a seguir: seja *G(N,A)* um grafo Gerador de Jornadas de Trabalho. *N* é o conjunto de nós que representa as viagens, onde |*N*| = número de viagens. *A* é o conjunto de arcos, onde cada arco entre um par de nós, *i,j (j>i)* indica que uma mesma tripulação pode executar a viagem *i* e logo depois a viagem *j*. Cada um desses arcos possui um custo c_{ij}, o qual está associado ao tempo que a tripulação permanece ociosa entre as viagens unidas pelo arco. Existe um nó "0" que representa o depósito, no qual iniciam todas as jornadas de trabalho. Também existe um nó "N+1", nó considerado depósito, no qual são encerradas todas as jornadas.

O problema representado por este grafo consiste em encontrar *K* caminhos diferentes desde o nó "0" até o nó "N+1" que cumpram as seguintes condições:

a) Todas as viagens pertencem exatamente a um caminho;
b) O tempo total de trabalho em cada caminho não pode ser maior que um tempo máximo de jornada;
c) O custo total dos caminhos é mínimo.

Como cada viagem deve ser executada uma única vez, o custo associado à sua execução será constante para qualquer conjunto de caminhos que possuam todas as viagens. Portanto, se pode descartar esse custo, considerando unicamente o custo dos arcos de transição entre tarefas.

Os arcos de transição cumprem um papel importante na formulação do PPT, posto que, como se estabeleceu anteriormente, para qualquer par de viagens *i* e *j (j>i)* somente existe um arco de transição de custo c_{ij} quando é possível que a mesma tripulação execute primeiramente a viagem *i* e logo depois execute a viagem *j*, respeitando colisão de horários e compatibilidade de terminais.

O método construtivo provém da Inteligência Artificial e se baseia na representação do problema através de grafos. Seu algoritmo de solução é conhecido na literatura como AAO* (NILSSON, 1981) (PEARL, 1990), um método de busca construtiva. Neste método, são

utilizados dois tipos de representação de problemas da área da Inteligência Artificial (NILSSON, 1981). O primeiro tipo de representação, conhecida como Espaço de Estados, representa o problema por meio de um grafo onde cada nó constitui um estado solução ou uma parte dele e os arcos deste grafo correspondem a um conjunto de transformações ou regras que permitem modificar um estado, transformando-o em outro. O segundo tipo de representação utiliza uma redução do problema em subproblemas de menor dificuldade, que podem ser resolvidos de forma independente. Essa decomposição se representa como um grafo AND/OR, de forma que se decompõe cada nó até chegar a um ponto onde não se pode seguir decompondo (PEARL, 1990).

É importante destacar que nesse tipo de grafo se pode associar um custo pela geração de cada novo nó. A geração de um novo nó representa no grafo a inserção de uma nova viagem (nó) de modo a construir uma jornada (caminho) que pertence a uma solução. Assim, se define *g(n)* como o custo devido à geração de um novo nó *n*. Se na geração deste nó participam os nós $n_1,...,n_k$, o custo total *g(n)* é o somatório do custo de todos esses nós.

Por outro lado, a cada nó se associa uma função de custo heurístico *h(n)*, que permite estimar o custo que falta para, a partir deste nó, atingir o nó final "N+1", isto é, completar o caminho do nó "0" ao "N+1". Essa função se conhece como heurística, que se define de acordo com cada problema específico.

A busca nesse tipo de grafo pode ser feita por meio do algoritmo AAO* (GOMES; PARADA, 1991), que trabalha com listas de nós: *Abertos* e *Fechados.* Dentro dos nós que se encontram na lista de *Abertos*, escolhe o nó que tenha o melhor valor da função custo para expandí-lo, logo translada esse nó à lista de nós *Fechados* e avalia a todos os nós recém-expandidos, completando um ciclo do algoritmo. A função $f(n) = g(n) + h(n)$ aplicada a cada nó inclui as duas funções vistas anteriormente: a função de custo g(n) e a função heurística h(n).

O algoritmo construtivo AAO* chega à solução ótima por meio da união de outras soluções parciais, realizando cortes para que não ocorra a ampliação exagerada do espaço de soluções. No algoritmo proposto nesse trabalho, as listas de nós *Abertos* e *Fechados* são representadas por conjuntos de soluções.

3 ALGORITMO COMPUTACIONAL PROPOSTO

O algoritmo criado para resolver o problema proposto foi escrito na linguagem de programação Python. O mesmo foi baseado nos trabalhos de Elizondo et al. (2009) e de Fuentes (2010), o que inclui os componentes básicos do algoritmo, os operadores genéticos, as funções de custo e o andamento geral do algoritmo.

O ACE baseia sua arquitetura sobre o conceito de populações, que são grupos de indivíduos. Nesse trabalho, um indivíduo corresponde a uma solução para o PPT, sendo uma solução qualquer agrupamento de viagens, seja ele parcial ou completo (com todas as viagens da TH já alocadas). No ACE, os indivíduos das populações interagem entre si através de cruzamentos, são passíveis de sofrer mutações e passar por seleções, enquanto são construídas as soluções, interagindo soluções parciais entre si para poder chegar numa completa. Desse modo o algoritmo ao se encerrar gera um grupo de soluções em vez de uma única. Salienta-se que ele sempre guarda a solução de menor valor da Função Custo.

Em cada iteração do algoritmo, de modo análogo a diferentes gerações de uma espécie, as populações recebem indivíduos gerados por cruzamentos das iterações anteriores, que podem ou não ter sofrido mutações, e que tenham sido aprovados em uma seleção. Essas ações, chamadas operadores genéticos, são apresentadas na seção 4.2.

Nesse capítulo serão apresentados os componentes básicos do algoritmo, os operadores genéticos, a Função Custo usada para avaliar as soluções, a interação entre as populações que permite que ocorra a evolução do sistema e o algoritmo simplificado em pseudocódigo.

3.1 COMPONENTES BÁSICOS

O código se organiza em torno dos seguintes componentes: viagens, serviços, soluções e populações. Na implementação em Python cada um destes componentes corresponde a uma classe e tem vinculados a si atributos (valores que ficam armazenados e podem ser alterados) e métodos (instruções de cálculos que a serem feitos sempre que solicitado, mas sem que seja armazenado seu resultado). A decisão entre quais informações serão tratadas como atributos e

quais como métodos não é rígida e universal, sendo que neste trabalho buscou-se utilizar a menor quantidade possível de atributos.

Nessa seção serão apresentados, pelo ponto de vista computacional, os componentes e seus atributos.

3.1.1 Viagens

As viagens são as peças fundamentais do problema. É o único componente que não sofre qualquer alteração interna, ficando armazenadas em uma base de dados fixa, disponível para a leitura dos atributos para que sejam utilizados pelo algoritmo. A definição de uma viagem tal qual sua representação no algoritmo computacional está esquematizada na Figura 2.

idv	*hi*	*hf*	*ti*	*tf*	*dur*	*chp*
54	07/Jun/2019 23:45	08/Jun/2019 00:20	04	05	00:35	0,25

Figura 2 - Representação esquemática dos atributos de uma viagem

Os atributos de uma viagem são:

a) *idv* – número identificador da viagem
b) *hi* – data e horário de início da viagem
c) *hf* – data e horário de fim da viagem
d) *ti* – terminal de início da viagem
e) *tf* – terminal de fim da viagem
f) *dur* – duração da viagem
g) *chp* – coeficiente de horário de pico

Os campos de horário de início e fim da viagem armazenam também uma data, apenas como convenção, para permitir a definição de viagens como a exemplificada, que começam em um dia e terminam em outro. O coeficiente de horário de pico é um atributo que aponta o quão próxima essa viagem está dos horários da TH com maior frequência de viagens. Seu cálculo é detalhado na seção 3.3.1.3, bem como sua aplicação no cálculo da Função Custo.

3.1.2 Serviços

Os serviços são grupos de viagens a serem atribuídas a tripulações. Um serviço deve ter um intervalo de descanso preferencialmente próximo à sua metade. A representação de um serviço no algoritmo computacional está esquematizada na Figura 3.

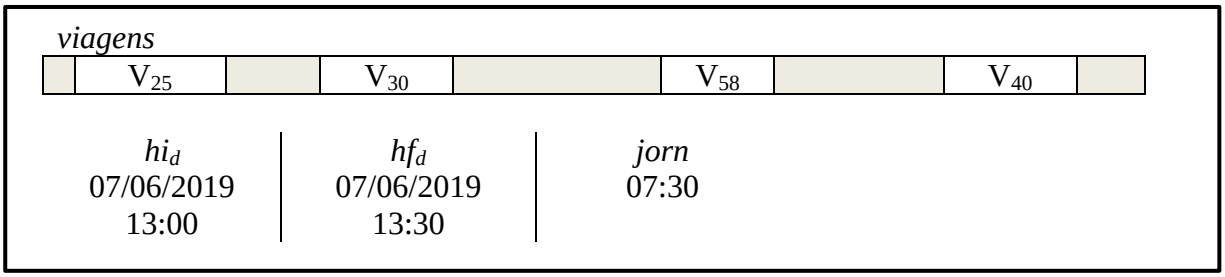


Figura 3 - Representação esquemática dos atributos de um serviço

Os atributos de um serviço são:

a) *viagens* – uma lista dos atributos *idv* das viagens que são cumpridas nesse serviço, organizadas por ordem do horário de início
b) hi_d – horário de início do intervalo de descanso
c) hf_d – horário de fim do intervalo de descanso
d) *jorn* – duração fixa da jornada para esse serviço

A jornada de trabalho de um motorista ou cobrador, pelo ponto de vista do pagamento da mão-de-obra, é fixada em 7h30min e um serviço necessariamente deve ter esse tempo para sua execução. Isso significa que se, por um lado, o tempo que transcorre da primeira à última viagem não pode ultrapassar esse limite, por outro lado, um serviço que tenha viagens a cumprir apenas durante 6h também receberá o pagamento de 7h30min. Assim, para maior economia é mais vantajoso que o serviço esteja o mais completo possível.

A construção de um serviço se dá ao longo do algoritmo, à medida que os operadores genéticos (seção 3.2) atribuem novas viagens a ele. Durante esse processo, o horário de início e fim do serviço não é fixo, sendo estabelecido apenas de modo dependente às suas viagens. Quando um serviço é criado, ele conta com apenas uma viagem. A cada nova viagem adicionada, seu horário de início e fim é atualizado, bastando que todas as viagens estejam contidas num intervalo menor ou igual à duração *jorn*.

O intervalo de descanso deve ter no mínimo meia hora e não pode ser atribuído antes que o serviço tenha um dado número de viagens. Essa condição é testada sempre que uma nova viagem é adicionada. A atribuição do intervalo de descanso segue as seguintes diretrizes:

a) Pode ser atribuído o intervalo desde que seja atendido um número mínimo de viagens, parâmetro a ser calibrado;
b) Quando atendida a condição (a), há uma probabilidade de 50% de atribuir o intervalo;
c) Quando atendidas as condições acima, tenta-se colocar o intervalo na metade do serviço;
d) Caso haja colisão com alguma viagem na metade do serviço, tenta-se adicionar o intervalo nos períodos livres entre viagens, priorizando os mais próximos da metade;
e) Caso não haja períodos livres entre viagens de tamanho suficiente, adiciona-se no início ou no fim do serviço.

A condição (e) é uma característica indesejada para uma solução completa, porém é aceitável para soluções parciais, considerando que esses serviços receberão novas viagens. Ainda assim, uma solução pode chegar ao seu estágio de solução completa (ver definições dos tipos de solução na próxima seção) contendo serviços com o intervalo atribuído no início ou fim, ou mesmo sem nenhum intervalo atribuído. Nesse caso, tais serviços são excluídos e a solução continua sua construção.

Na operação de transporte de ônibus da cidade de Porto Alegre existem *viagens mortas*, que consistem em deslocamentos do ônibus que não atendem aos usuários. Tais viagens são usadas sempre que o ônibus ou a tripulação precisam ser transportados entre terminais ou entre a garagem e um terminal.

Quando a TH tem mais viagens Bairro-Centro (BC) do que Centro-Bairro (CB), são necessárias viagens mortas CB entre duas viagens BC. Esse tipo de viagem morta é denominado *viagem expressa*.

Sempre que um serviço se inicia, a tripulação deve retirar o veículo na garagem e leva-lo ao terminal. Já quando um serviço se encerra, a tripulação deve entregar o veículo na garagem. Para atender a essa restrição, neste algoritmo é adicionado um intervalo fixo de meia hora no início e no fim dos serviços para permitir todas as possíveis viagens mortas de início e fim de serviço. Não são permitidas neste algoritmo outras situações de viagens mortas, como viagens

expressas. Como consequência disso, uma viagem só é adicionada se seu terminal de início for o terminal de fim da viagem anterior, e vice-versa para a viagem posterior.

3.1.3 Soluções

Uma solução consiste em um agrupamento de serviços que não tenham viagens repetidas entre si, de modo que cada serviço possa ser realizado por uma tripulação diferente. Essa solução, pela perspectiva do ACE, corresponde a um indivíduo de uma população.

As soluções nesse algoritmo podem ser classificadas em:

a) solução parcial – uma solução que está em processo de construção e ainda não possui todas as viagens da TH;
b) solução completa – uma solução que já possui todas as viagens da TH;
c) solução factível – uma solução que, além de possuir todas as viagens da TH, atende aos seguintes requisitos: ter intervalos de descanso atribuídos em todos seus serviços e nenhum desses intervalos estarem no início ou no fim de um serviço.

O objetivo do algoritmo é encontrar uma população com no mínimo uma solução factível. Uma solução parcial não pode ser considerada uma resposta para o PPT: antes disso precisa completar a fase construtiva do ACE, na qual continuará recebendo viagens.

Para chegar até uma solução factível, o algoritmo manipula soluções parciais, fazendo nelas alterações ao longo das iterações. Sempre que uma solução parcial se torna completa, mas não atende aos requisitos para ser considerada uma solução factível, os serviços que não o atenderam são excluídos da solução e esta volta ao ciclo de construção. A fim de permitir variabilidade na recombinação, também são excluídos 10% dos serviços que atenderam aos requisitos. Sem essas medidas de exclusão de serviços, a solução completa nunca viria a se tornar factível.

A representação computacional de uma solução está esquematizada na Figura 4.

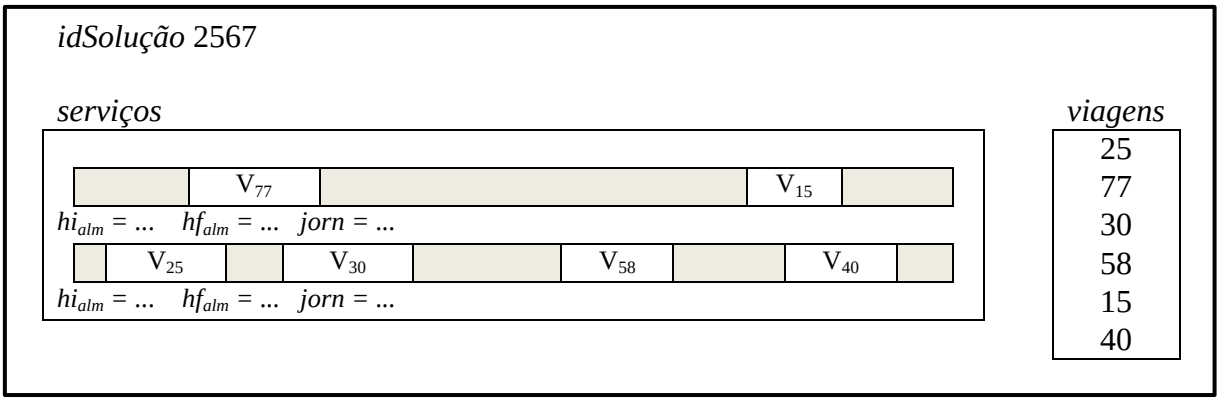


Figura 4 - Representação esquemática dos atributos de uma solução

Os serviços existem apenas como parte de soluções, não tendo significado fora delas pelo ponto de vista do algoritmo. Por esse motivo, o campo *ids* de um serviço serve apenas para identificá-lo entre os outros serviços da solução. Assim, não se faz necessária uma contagem global de serviços, diferentemente do caso das viagens.

3.1.4 Populações

Populações correspondem a um grupo de soluções, sejam elas factíveis, completas ou parciais, sobre o qual são realizadas as operações do algoritmo. São abstrações que dizem respeito apenas à lógica básica do ACE e não têm significado concreto no PPT.

Uma população tem o atributo n_{pop} de população máxima, que guarda o número máximo de soluções que ela pode conter. No algoritmo algumas populações são fixas durante toda a execução, enquanto outras são totalmente zeradas a cada iteração. Como o PPT é um problema de natureza combinatória, as populações fixas estão sempre recebendo novas soluções, motivo pelo qual algumas soluções devem também ser excluídas. Para essa exclusão é utilizado o operador da Seleção, que é detalhado na seção 4.2.3. Sendo n a quantidade atual de soluções dessa população, a cada iteração, se $n > n_{pop}$, o operador Seleção permitirá permanecerem apenas n_{pop} indivíduos e excluirá $n - n_{pop}$ soluções sobressalentes.

A definição de uma população no algoritmo está esquematizada na Figura 5.

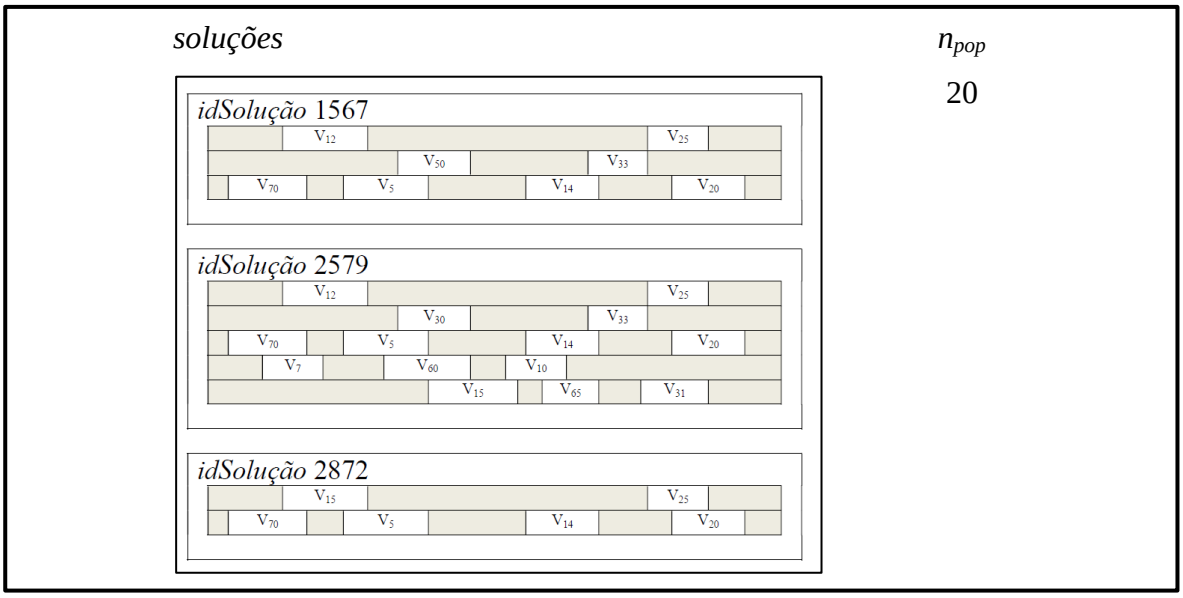


Figura 5 - Representação esquemática de uma população

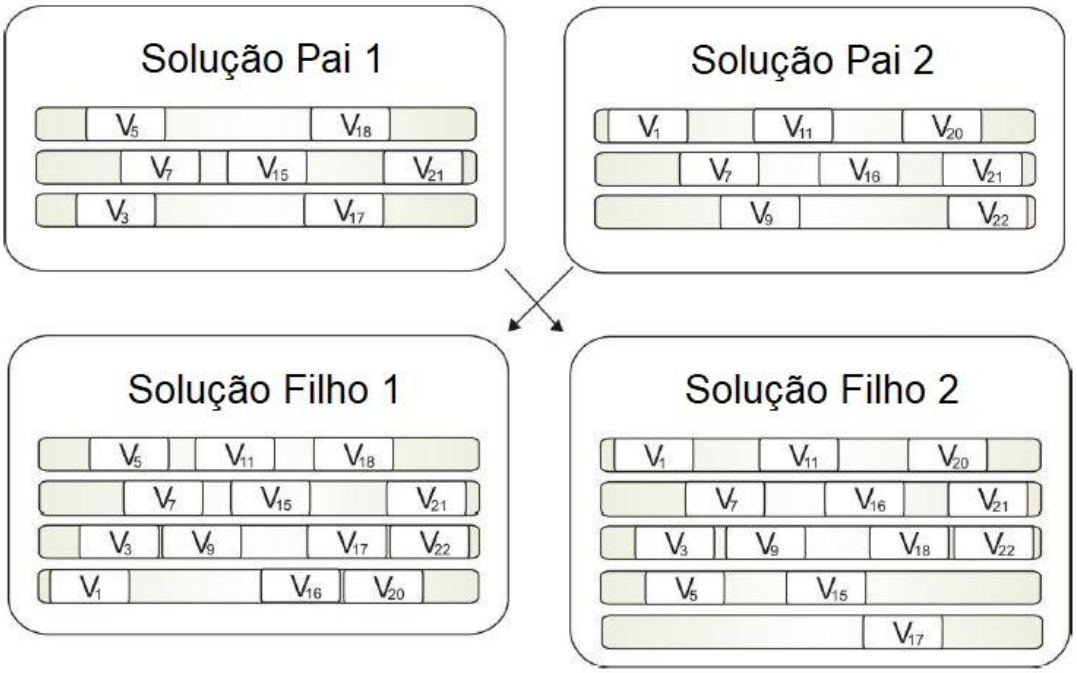
3.2 OPERADORES GENÉTICOS

Parte fundamental da teoria do AG, herdada pelo ACE, é a utilização dos operadores genéticos, sendo eles o Cruzamento, a Mutação e a Seleção. Esses operadores são alterações que os indivíduos das populações devem sofrer para que progrida o algoritmo de modo análogo ao que ocorre com a seleção natural, e para que desse modo possam ser exploradas diferentes soluções do espaço de soluções. No algoritmo em Python, essas operações são representadas por métodos, sendo estes atrelados às classes (componentes básicos da seção anterior). Serão apresentados nas próximas seções os operadores segundo a metodologia de Elizondo et al. (2009) e Fuentes (2010), tais quais são utilizados no presente trabalho.

3.2.1 Cruzamento

Um cruzamento é uma operação aplicada em duas soluções, que consiste na união entre dois indivíduos da população e à geração de indivíduos descendentes. Neste algoritmo, um cruzamento entre os indivíduos Pai 1 e Pai 2 gera os indivíduos Filho 1 e Filho 2. O Filho 1 é gerado fazendo uma cópia do Pai 1 e adicionando a ela, uma a uma, as viagens do Pai 2. Já o Filho 2 tem como base o Pai 2 e recebe as viagens do Pai 1. Um esquema do funcionamento do cruzamento pode ser visto na Figura 6.

Figura 6 - Representação esquemática de um cruzamento



(fonte: FUENTES, 2010)

Pelo ponto de vista das informações inseridas no Filho 1, os serviços vindos do Pai 1 têm sua estrutura mantida e os do Pai 2 são ignorados, já que as viagens são inseridas independentemente. Aqui, para o Filho 1, o Pai 1 é chamado Pai Base. Nesse processo, também são mantidos os horários de intervalo e de duração de jornada dos serviços do Pai 1 e desconsiderados os do Pai 2. Para o Filho 2, a situação é oposta, sendo mantidos os serviços do Pai 2, que será o Pai Base deste caso.

A inserção de viagens ocorre tentando adicionar a viagem a um serviço e, caso não seja possível, tenta-se no próximo, seguindo a ordem em que os serviços estão dispostos na solução. Caso a viagem não possa ser inserida em nenhum dos serviços, deve ser criado um novo, o que na Figura 6 ocorre, por exemplo, com a Viagem 17 do Filho 2.

Como cada viagem deve constar apenas uma vez em cada solução, uma viagem que consta em ambos os Pais permanece do modo que estava disposta no Pai Base. Isso pode ser visto na Figura 6 no caso das viagens 7 e 21.

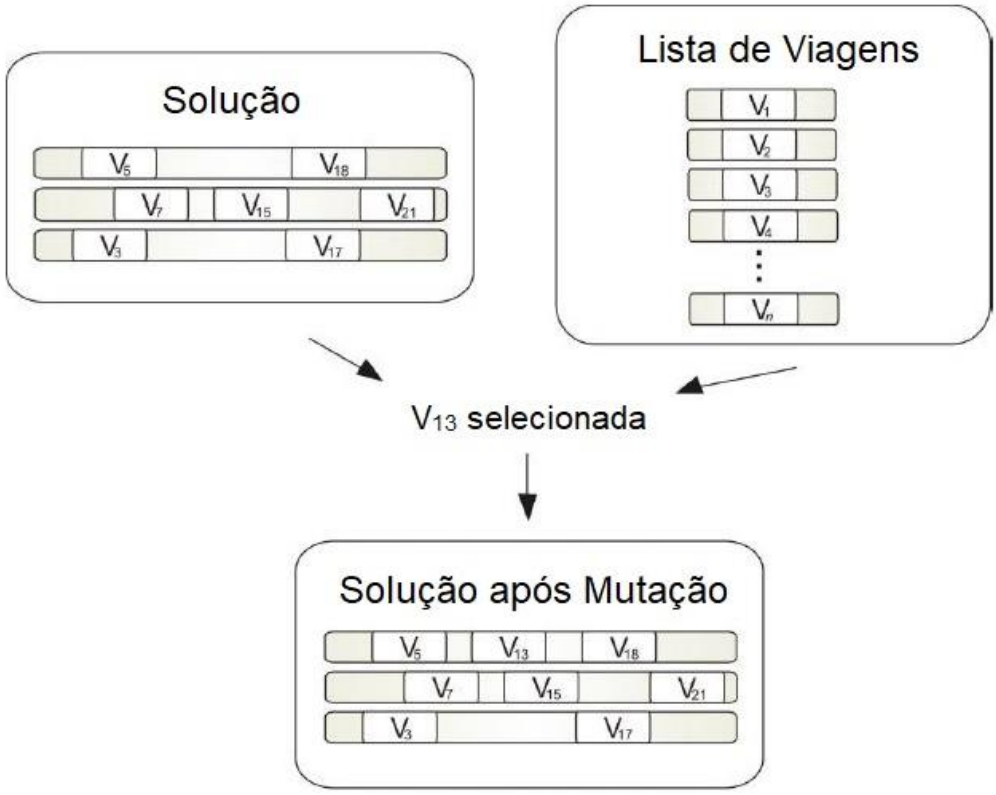
3.2.2 Mutação

Uma mutação corresponde neste algoritmo à adição de uma viagem isolada, que ocorre seguindo os mesmos princípios da adição de viagens que ocorre no Cruzamento. No ACE uma mutação cumpre a função de adicionar aleatoriedade à solução, fazendo com que ela não fique limitada a pontos ótimos locais.

Esse operador é de natureza probabilística, o que neste algoritmo é representado por uma probabilidade de ocorrência da mutação. O operador é aplicado sempre que um gerador de números aleatórios resulta num valor x que pertença ao intervalo $0 < x < p_m$, onde p_m é a probabilidade de ocorrência de mutação, parâmetro global estipulado. Esse teste é realizado a cada geração (iteração do algoritmo).

Uma representação do processo de mutação é apresentada na Figura 7.

Figura 7 - Representação esquemática de uma mutação



(fonte: FUENTES, 2010)

3.2.3 Seleção

O operador da seleção é aplicado sobre uma população para definir quais soluções serão escolhidas para gerar descendentes na próxima geração. Analogamente à seleção natural, os indivíduos mais aptos terão mais chances de passar seu material genético adiante.

O operador seleção é utilizado nas seguintes situações:

a) escolher indivíduos para realizar cruzamentos entre eles;
b) escolher os melhores indivíduos gerados pelos cruzamentos da iteração atual, para serem passados para a geração seguinte;
c) escolher indivíduos para serem excluídos quando uma população ultrapassou seu número máximo de indivíduos (n_{pop}).

Para definir qual indivíduo é mais apto em uma população, utiliza-se a Função de Desempenho (*Fitness Function*). Como o objetivo desse trabalho é minimizar características como o tempo ocioso e a demanda por tripulações, essas características são atreladas a um custo e a Função de Desempenho é chamada de Função Custo (FC). A definição dessa função será apresentada na próxima seção. O processo da seleção, portanto, consiste em escolher em uma população as soluções com a FC de menor valor.

Outra característica importante do AG é que, como na natureza, a seleção tem caráter probabilístico, portanto o método conta com dois tipos de seleção: Seleção Determinística e Seleção Roleta. A Seleção Determinística consiste simplesmente em escolher de uma população os indivíduos com menor FC. A Seleção Roleta consiste em fazer uma escolha dos indivíduos utilizando um gerador aleatório, atribuindo maior probabilidade de seleção aos indivíduos com menor FC.

A Seleção Roleta é aplicada sobre uma população de acordo com as seguintes etapas:

a) Obter uma lista *pesos* que contém os valores de FC de todas as soluções da população;
b) Obter o valor máximo da lista $maxp_1$;
c) Inverter os valores p da lista *pesos* aplicando $p \leftarrow maxp_1 - p$ para cada um dos valores. Invertida, a lista *pesos* tem valor zero para as soluções com maior FC e valor máximo para as soluções com menor FC;
d) Obter o novo valor máximo $maxp_2$ da lista *pesos;*

e) Somar uma probabilidade mínima $p_{roleta} \cdot maxp_2$ a todos os valores *p* da lista *pesos*, aplicando $p \leftarrow p + p_{roleta} \cdot maxp_2$ para que as soluções de maior FC não sejam totalmente excluídas (p_{roleta} é um parâmetro calibrado);

f) Obter a soma *somap* de todos os valores da lista *pesos*;

g) Normalizar a lista *pesos*, aplicando $p \leftarrow p / somap$ de modo que os valores *p* representem probabilidades que se acumulam em um total 1;

h) Utilizar a função *random.choice* da biblioteca *Numpy* (SCIPY.ORG, 2018) para obter as soluções selecionadas probabilisticamente com base nos seus valores de FC. Utilizar como entrada da função os atributos:

· *a* – lista de *ids* das soluções que constam na população

· *size* – número de soluções a serem selecionadas

· *p* – as probabilidades atribuídas a cada solução de *a*. Atribuir a essa entrada a lista *pesos.*

i) A função *random.choice* retornará a lista de soluções selecionadas e com isso obtém-se o resultado desejado da Seleção Roleta.

3.3 FUNÇÃO CUSTO

A Função Custo (FC), definida geralmente como Função de Desempenho (*Fitness Function)* é parte essencial do funcionamento do AG, que está incluído no ACE. Serve para comparar soluções, sejam elas factíveis, completas ou parciais, escolhendo as que mais se adequam aos requisitos do problema. O uso da FC junto ao operador seleção promove o melhoramento das soluções, atribuindo grandezas mensuráveis às características preferíveis ou não.

A utilização da FC para avaliar os indivíduos de uma população é vantajosa porque exige menos controle fino de métodos exatos de construção para atingir uma solução complexa. Se a FC é bem construída e avalia adequadamente as soluções, ela guia o algoritmo na exploração do espaço de soluções, usando a aleatoriedade favoravelmente. Deixa-se que a aleatoriedade gere as soluções dentro de uma base mínima de princípios e, depois de geradas, elas são avaliadas, sendo excluídas ou não. Por outro lado, é necessário ficar atento nas calibrações para que a avaliação não seja muito restritiva nem muito branda. Uma avaliação muito restritiva pode excluir soluções demais e nunca conseguir encontrar alguma que a satisfaça. Uma restrição muito branda pode gerar soluções de baixa qualidade.

A FC é formada por duas parcelas: o Custo Atual e o Custo Meta. O Custo Atual representa as características da solução no momento em que é avaliada, no que diz respeito à quantidade de tempos ociosos, à quantidade de serviços e à priorização de viagens em horários de pico. O

Custo Meta representa quanto falta para uma solução atingir o estágio de solução completa, através de estimativas relativas a quais características uma solução ideal teria.

O detalhamento das Funções Custo Atual e Custo Meta será apresentado nas seções seguintes.

3.3.1 Função Custo Atual

A primeira parcela do custo representa o custo relacionado às viagens e serviços já alocados em uma solução. Esse custo depende do tempo ocioso dentro dos serviços e do número de serviços, que são as características a minimizar nesse problema. Além disso, é atribuído um custo referente às viagens fora da hora de pico, a fim de fazer com que o algoritmo priorize as viagens de horário de pico em sua alocação, para que os horários de intervalo preferencialmente não sejam colocados nesse período.

O tempo ocioso, que neste algoritmo é chamado *folga*, é definido como o tempo em que a tripulação está cumprindo um serviço, mas não está desempenhando uma viagem. A folga de um serviço pode ser de dois tipos: folga interna e folga externa.

Uma jornada tem uma duração fixa de 7h30min, não sendo permitido que seja efetuado um pagamento de mão de obra proporcional para períodos menores. Caso um serviço não tenha viagens atribuídas até o final, o tempo ocioso até que as 7h30min sejam completadas é denominado *folga externa*. Já o tempo ocioso restante, que ocorre entre as viagens, é denominado *folga interna*.

Cabe ressaltar que, em uma solução, as viagens da TH não significam um custo. Isso se deve ao fato de que a FC não representa um custo monetário e sim uma ferramenta para a resolução do problema. O objetivo do trabalho é minimizar os custos citados acima (tempos ociosos e número de serviços) em comparação com uma situação ideal onde todas as viagens são cumpridas com zero tempo ocioso e um número mínimo de serviços. A minimização do custo monetário ocorre como decorrência natural disso, mas a escala de tripulações não tem seus valores financeiros apreciados e comparados dentro dessa resolução.

A fim de expressar qual preponderância um custo tem em relação ao outro, ou seja, quais características são preferíveis a outras, cada um dos custos tem um coeficiente que o relaciona aos outros.

O cálculo do Custo Atual (CA) de uma solução, em minutos, é dado pela Equação (1).

$$CA = FE + \tau \cdot FI + \alpha \cdot QS + \delta \cdot HP \quad (1)$$

Onde:

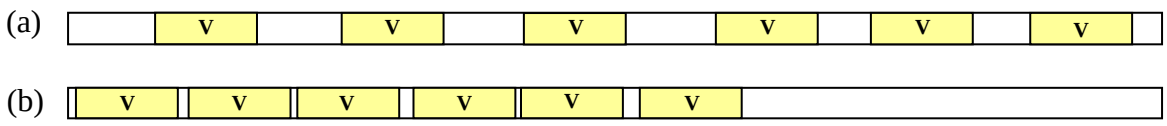
FE é o custo das folgas externas dos serviços da solução, em minutos
FI é o custo das folgas internas dos serviços da solução, em minutos
NS é o custo relativo à quantidade de serviços da solução, em minutos
HP é o custo relativo ao horário de pico, em minutos
τ é o coeficiente de ponderação do custo das folgas internas
α é o coeficiente de ponderação do custo do número de serviços
δ é o coeficiente de ponderação do custo do horário de pico

As parcelas da Equação (1) serão definidas e detalhadas nas seções seguintes.

3.3.1.1 Duração de Folgas dos Serviços

As primeiras duas parcelas da Equação (1) se referem às folgas do serviço, ou seja, a seus tempos ociosos. A divisão da duração de folgas entre internas e externas se deve ao fato de que são preferíveis serviços com viagens mais compactas. Na Figura 8 são ilustrados dois serviços para comparação quanto à compacidade das viagens.

Figura 8 – Representação de dois serviços, um com viagens menos compactas (a) e outro com viagens mais compactas (b)



(fonte : ELIZONDO et al., 2009)

O tratamento diferenciado para folgas internas e externas é dado pelo coeficiente τ, que no trabalho de FUENTES (2010) tem o valor de $\tau = 1{,}5$. Multiplicadas por esse coeficiente, as folgas internas resultam num valor de custo proporcionalmente maior, o serviço *(a)* se torna

mais caro que o serviço *(b)* e por fim o serviço *(b)* é priorizado quando um operador genético *seleção* é aplicado.

Assim, as parcelas *FE* e *FI* da Equação (1) correspondem simplesmente à duração das folgas medidas nos serviços em minutos, formando o custo $FE + \tau \cdot FI$ referente às folgas dos serviços.

Esse valor pode ser obtido tanto para apenas um serviço quanto para toda uma solução. Porém, como o operador *seleção* é aplicado em soluções, nesse caso a parcela $FE + \tau \cdot FI$ se refere ao somatório das folgas de todos os serviços da solução onde está sendo aplicado o operador.

3.3.1.2 Quantidade de Serviços da Solução

Uma solução é melhor na medida em que ela tem uma menor quantidade de serviços. Uma menor quantidade de serviços corresponde a uma menor quantidade de motoristas, cobradores e veículos para cumprir a TH, o que leva a menores gastos nessas áreas.

Outro motivo que torna necessária a atribuição desse custo é a minimização de folgas. Caso existisse no custo atual apenas a parcela relativa às folgas, o algoritmo atribuiria uma viagem a cada serviço, o que seria avaliado como menos custoso, pois não haveria nenhuma folga interna. Assim, por exemplo, uma TH com 100 viagens geraria uma solução do PPT com 100 serviços. Para evitar isto, atribui-se para a solução um custo relacionado à quantidade de serviços.

O custo correspondente ao número de serviços, representado na Equação (1) por $\alpha \cdot QS$, consiste na soma *QS* das durações das jornadas em minutos, multiplicada pelo coeficiente α de ponderação desse custo em relação ao das folgas externas. Como nessa implementação do algoritmo todas as jornadas têm a duração fixa de sete horas e meia, *QS* se resume à quantidade de serviços da solução multiplicada por essa duração fixa das jornadas.

3.3.1.3 Horário de Pico

A última parcela do custo atual consiste num custo relacionado às horas de pico. É estabelecido um custo menor para as viagens que ocorrem em horário de pico, para que elas sejam alocadas antes. Essa prioridade é interessante para que os intervalos de descanso das tripulações sejam atribuídos aos serviços preferencialmente em períodos fora do horário de pico, fazendo um uso mais eficiente da mão de obra.

Essa parcela está ligada menos aos custos reais e mais a uma priorização para guiar por onde o algoritmo deve começar a alocação de viagens, pois é um custo relativo entre viagens. Por esse motivo, esse custo perde o significado quando são comparadas duas soluções completas, já que necessariamente elas têm as mesmas viagens e, assim, essa parcela de custo será igual em ambas.

Para calcular a parcela de custo de horário de pico, é necessário utilizar o atributo do coeficiente de horário de pico *(chp)* das viagens da TH. O *chp* é uma grandeza que varia 0 a 1 e representa se essa viagem começa em um período de máxima frequência de viagens (0) ou mínima frequência (1). A escala permite que uma viagem gere um menor valor para FC quando está num momento de maior frequência de viagens. O *chp* é calculado através das seguintes etapas:

a) Obter uma série temporal *S* que mostre, a cada 5 minutos e ao longo de toda a TH, a quantidade de viagens *q* que ocorrem ao mesmo tempo;
b) Obter o valor máximo *maxQ* dessa série, ou seja, o número máximo de viagens concomitantes da TH;
c) Inverter a série *S*, aplicando a cada um de seus valores *q* a operação $q \leftarrow maxQ - q$;
d) Normalizar a série *S*, dividindo cada um de seus valores *q* por *maxQ*;
e) Atribuir a cada viagem da TH um valor da série *S*, utilizando para a localização temporal da viagem o atributo de horário de início da viagem *hi*. Esse valor é o atributo *chp*.

O custo de uma solução relacionado ao horário de pico corresponde na Equação (1) à parcela $\delta \cdot HP$, onde δ é o coeficiente de ponderação que relaciona o custo dos horários de pico ao custo das folgas externas e *HP* é o somatório dos custos de todas as viagens da solução, atribuído a partir dos atributos *chp*. A parcela *HP* do custo é apresentada na Equação (2).

$$HP = \sum_{i=0}^{n} \left(jorn_i * \sum_{j=0}^{m} chp_j \right) \quad (2)$$

Onde:

n é o número de serviços da solução
m é o número de viagens do serviço *i*
$jorn_i$ é o atributo *jorn* (duração de jornada) do serviço *i*
chp_j é o atributo *chp* (horário de pico) da viagem *j*

3.3.2 Função Custo Meta

O Custo Meta (CM) serve para fazer com que o algoritmo não fique estagnado em soluções parciais com poucas viagens. Ele representa quantas viagens e quantos serviços ainda devem ser alocados na solução, utilizando-se para isso de estimativas referentes a uma solução idealizada. Essa segunda parcela da FC é necessária para que as soluções recebam novas viagens na fase construtiva (fase 1) do ACE. Se o CM não existisse, nenhuma solução aumentaria de tamanho. Isso se deve ao fato de que uma solução menor, com menos serviços e menos viagens, tem menos folgas, portanto tem menor CA. Assim o algoritmo permaneceria com soluções pequenas e não tenderia a chegar à solução completa. Por isso, se torna necessário atribuir esse custo.

O cálculo do Custo Meta (CM) é demonstrado na Equação (3).

$$CM = \gamma \, (\alpha \cdot SF \cdot jorn_{glob} + FM) \quad (3)$$

Onde:

SF é o número de serviços faltantes
$jorn_{glob}$ é o valor global de duração de jornadas
FM é a duração de folga mínima
γ é o coeficiente de ponderação do Custo Meta em relação ao Custo Atual
α é o coeficiente de ponderação do custo do número de serviços

A fim de compatibilizar o CM com o CA, é utilizado o coeficiente de ponderação γ. Se esse coeficiente for muito baixo, o CM não é preponderante e perde sua função de fazer com que a solução parcial aumente de tamanho. É importante relembrar que para toda solução completa o CM vale zero, já que representa as viagens e serviços que ainda não foram atendidos por uma solução parcial.

As parcelas da Equação (3) serão definidas e detalhadas nas seções seguintes.

3.3.2.1 Quantidade de Serviços Faltantes

A parcela relativa à quantidade de serviços faltantes é estabelecida a partir de uma quantidade estimada de viagens que cabem em um serviço. A partir da TH do problema, é calculada a duração média das viagens a serem alocadas, chamada dur_{med}. A quantidade estimada de viagens que cabem em um serviço é $nv_{serv} = jorn_{glob} / dur_{med}$, onde $jorn_{glob}$ é o valor global de duração das jornadas, que para esse problema é igual a 7h30min. A quantidade de serviços faltantes é definida como $SF = (nv_{TH} - nv_{sol}) / nv_{serv}$, onde nv_{TH} é a quantidade de viagens da TH e nv_{sol} é a quantidade de viagens já alocadas na solução cujo custo está sendo avaliado.

O coeficiente α pelo qual SF é multiplicado é o mesmo usado na Função Custo Atual e tem a mesma função de expressar a proporção do custo dos serviços em relação ao custo das folgas. Além disso, a fim de que seja mantida a compatibilidade das unidades, que para toda a FC é dada em minutos, a parcela relativa à quantidade de serviços faltantes é também multiplicada por $jorn_{glob}$. Assim, obtém-se a parcela $\alpha \cdot SF \cdot jorn_{glob}$.

3.3.2.2 Duração de Folgas

A parcela relativa à duração de folgas mínima se baseia no valor SF definido acima. Supondo uma solução idealizada, a folga mínima possível para um serviço corresponde à soma da duração do intervalo de descanso mínimo (30 minutos) com a duração dos dois intervalos relativos às possíveis viagens expressas no início e fim de serviço (1 hora), totalizando assim $F_{mín}$ = 1h30min. A parcela FM relativa à duração mínima de folgas de uma solução é dada por $SF \cdot F_{mín}$, não sendo feita nenhuma multiplicação por coeficientes de ponderação.

3.3.3 Aplicação da Função Custo

A FC é aplicada sobre uma solução, levando em conta a duração de folgas internas e externas, quantidade de serviços, situação das viagens em relação a horários de pico, além de, para soluções parciais, os serviços faltantes para atingir a solução completa e a folga mínima que teria uma solução completa idealizada. A FC completa está apresentada na Equação (4).

$$FC = CA + CM = FE + \tau \cdot FI + \alpha \cdot QS + \delta \cdot HP + \gamma\,(\alpha \cdot SF \cdot jorn_{glob} + FM) \qquad (4)$$

Onde:

CA é a Função Custo Atual
CM é a Função Custo Meta
FE é o custo das folgas externas dos serviços da solução, em minutos
FI é o custo das folgas internas dos serviços da solução, em minutos
NS é o custo relativo à quantidade de serviços da solução, em minutos
HP é o custo relativo ao horário de pico, em minutos
SF é o número de serviços faltantes
$jorn_{glob}$ é o valor global de duração de jornadas
FM é a duração de folga mínima
τ é o coeficiente de ponderação do custo das folgas internas
α é o coeficiente de ponderação do custo do número de serviços
δ é o coeficiente de ponderação do custo do horário de pico
γ é o coeficiente de ponderação do Custo Meta em relação ao Custo Atual

É importante perceber que as parcelas $FE + \tau \cdot FI$ do CA correspondem à parcela *FM* do CM, pois representam respectivamente os tempos ociosos existentes e os faltantes em relação à solução meta. Do mesmo modo, a parcela $\alpha \cdot QS$ do CA corresponde à parcela $\alpha \cdot SF \cdot jorn_{glob}$ do CM, representando respectivamente os serviços existentes e faltantes. Por fim, apenas a parcela $\delta \cdot HP$ do CA (que representa a situação das viagens em horários de pico) não tem correspondente no CM.

Essa correspondência torna o CM inversamente proporcional ao CA, ou seja, à medida que o algoritmo constrói soluções parciais maiores, o valor de CA das soluções aumenta e o de CM diminui. Por fim, quando uma solução parcial se torna completa, o CM se torna zero.

Os coeficientes de ponderação são utilizados para compatibilizar os módulos das parcelas da FC, bem como estabelecer a prioridade que se deseja dar a cada característica na minimização. Quanto maior o valor de uma parcela no FC, mais custosa é considerada essa característica, portanto o algoritmo minimizará essa característica sempre que buscar uma solução com o menor FC.

Toma-se como base o único valor da CA que não utiliza um coeficiente: a duração das folgas externas. Para que as viagens sejam alocadas mais compactamente, é preferível que um serviço tenha menos folgas internas que externas, portanto é utilizado $\tau > 1$ para tornar as folgas internas mais custosas. Do mesmo modo, é preferível que uma solução tenha serviços bem preenchidos por viagens, em vez de muitos serviços com poucas viagens, portanto é também utilizado $\alpha > 1$ para tornar a quantidade de serviços mais custosa.

Para o caso da parcela de horários de pico, o atributo *chp* de cada viagem já cumpre a função de priorizar características preferíveis, já que seu valor será menor para uma viagem que esteja mais próxima dos horários de pico. Cada viagem alocada tem um *chp* atrelado, que entra no somatório de todas as viagens que compõem a solução, gerando a parcela HP do CA. Assim, o coeficiente δ serve apenas para a compatibilização dos módulos, ou seja, para que esse custo tenha a prioridade considerada adequada no custo. Tendo em vista que a alocação de viagens em horários de pico é considerada menos prioritária do que a minimização de folgas externas e a minimização de quantidade de serviços, o coeficiente é $\delta < 1$.

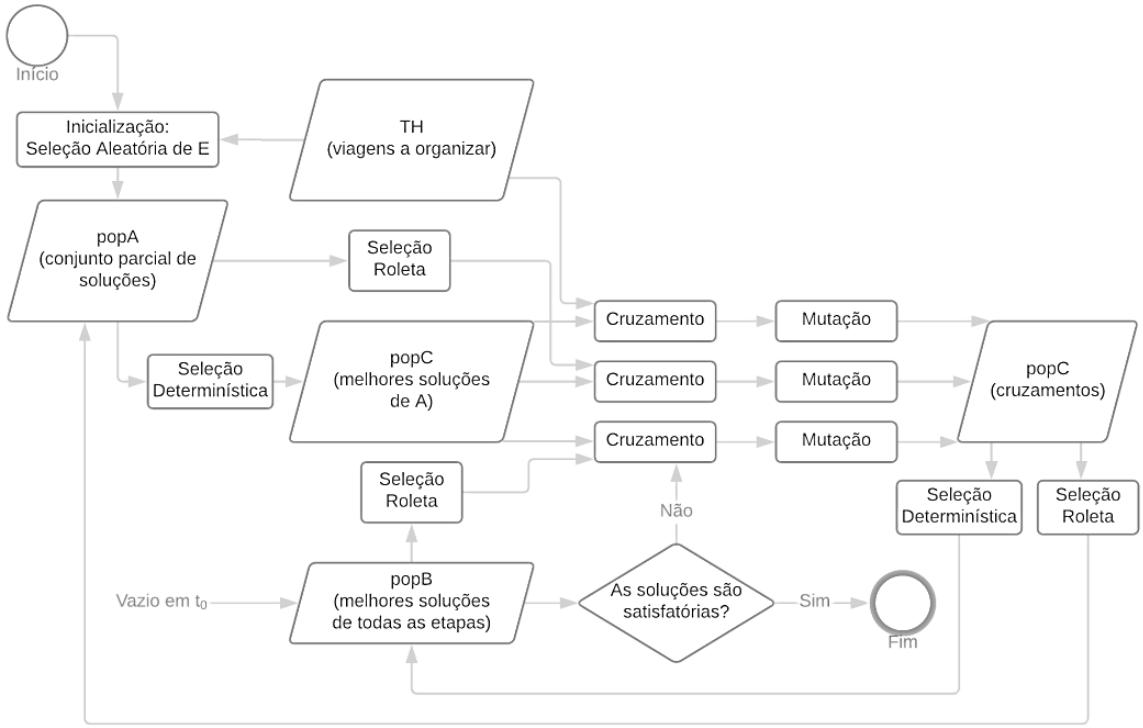
Por fim, o coeficiente γ serve para estabelecer uma proporção entre o CA e o CM que permita a convergência do algoritmo. Esse coeficiente deve ter uma grandeza tal que todo aumento em CA seja acompanhado por uma diminuição de módulo igual ou maior em CM, assim fazendo com que o algoritmo continue adicionando viagens nas soluções parciais. Isso também faz com que nesse momento o FC atinja seu valor mínimo e o CA atinja seu valor máximo. Assim, é utilizado um coeficiente $\gamma > 1$ que seja calibrado de modo a atender às condições acima.

Os valores utilizados nesse trabalho para os coeficientes são os mesmos utilizados por Fuentes (2010) e por Elizondo et al. (2009), exceto o coeficiente γ que foi introduzido nesse trabalho.

3.4 MACROESTRUTURA DO ALGORITMO

O ACE, como definido por Elizondo et al. (2009), consiste numa sequência específica de iterações durante as quais são aplicados operadores genéticos sobre populações. Essa sequência é representada na Figura 9.

Figura 9 - Algoritmo Construtivo-Evolutivo



(fonte: adaptado de FUENTES, 2010)

O algoritmo é iniciado criando as populações *PopA* (população base), *PopB* (melhores soluções) e *PopC* (geração atual), inicialmente vazias. Em seguida, soluções de apenas uma viagem são criadas a partir da TH para inicializar *PopA*, de modo a preencher sua capacidade populacional $n_{pop,A}$.

O laço principal consiste em realizar cruzamentos entre algumas soluções escolhidas de *PopA* para cruzarem (a) com integrantes de *PopA*, (b) com integrantes de *PopB* (que apenas na primeira iteração estará vazia) e (c) com viagens advindas da TH (ocorre o cruzamento como se essa viagem fosse uma solução de apenas um serviço com uma viagem).

Os filhos desses cruzamentos (a) (b) e (c) são armazenados em *PopC*. O operador mutação é aplicado em *PopC*, onde as soluções têm uma dada probabilidade p_m de sofrerem uma mutação.

Caso as populações *PopA* e *PopB* estejam com mais soluções do que suas capacidades populacionais $n_{pop,A}$ e $n_{pop,B}$, é realizada a exclusão das soluções extras utilizando seleção roleta (ver seção 3.2.3).

No fim dessa iteração do laço principal, é realizada uma seleção determinística sobre *PopC* na qual as melhores soluções vão para *PopB*. Além disso, uma seleção roleta é usada sobre *PopC* para escolher soluções para inserir em *PopA*.

Depois de realizadas as seleções, *PopC* é totalmente esvaziada. Então é iniciada a próxima iteração do laço principal.

As seleções diferentes aplicadas sobre a mesma população (*PopC*) (uma roleta para *PopA* e uma determinística para *PopB*) se devem ao fato de que *PopA* e *PopB* têm funções diferentes. A população base *PopA* recebe a seleção roleta para manter suficiente aleatoriedade no algoritmo, permitindo explorar diferentes soluções possíveis. Já *PopB* é utilizada para guardar as melhores soluções, por isso utiliza seleção determinística. Se fosse utilizada apenas *PopB* e fossem realizadas apenas seleções determinísticas, o algoritmo ficaria em um ótimo local com forte dependência das soluções com as quais fora inicializado.

Sempre que um operador genético é aplicado sobre uma solução, é verificado se a solução em questão está completa e se é factível. Sempre que uma solução completa é encontrada, mas não é factível, os serviços que não atendem aos requisitos são excluídos (ver seção 3.1.3). Sempre que uma solução factível surge em qualquer uma das populações, ela é armazenada numa população *PopF*.

O laço principal se encerra quando um dado número n_{fac} de soluções factíveis armazenadas em *PopF* é atingido.

3.5 ALGORITMO PRINCIPAL EM PSEUDOCÓDIGO

O algoritmo utilizado é adaptado de Fuentes (2010). Uma população é fixa quando mantém soluções ao longo de todo algoritmo, sendo retiradas apenas quando é aplicada alguma seleção sobre essa população. As definições utilizadas no pseudocódigo são os seguintes:

TH: Tabela Horária

PopA: população fixa geral

PopB: população fixa das melhores soluções

PopC: população dos cruzamentos do laço principal

PopF: população fixa das soluções factíveis encontradas

SR_i*:* seleção roleta

SD_i*:* seleção determinística

$n_{pop,X}$*:* capacidade máxima da população *X*

$n_{fac:}$ número de soluções factíveis para encerrar o algoritmo

f_c: quantidade proporcional de soluções selecionadas deterministicamente de *PopA* para cruzamento

f_b: quantidade proporcional de soluções selecionadas deterministicamente de *PopC* para armazenar em *PopB*

f_a: quantidade proporcional de soluções selecionadas por roleta de *PopC* para armazenar em *PopA*

n_{cruz}: número de soluções selecionadas por roleta de *PopA* e *PopB* para cruzamento

Início

Ler viagens e armazenar em TH

Selecionar aleatoriamente $n_{pop,A}$ viagens diferentes de TH

Preencher *PopA* com $n_{pop,A}$ soluções, cada solução com uma viagem das selecionadas acima

(Laço Principal) Repetir até que sejam encontradas n_{fac} soluções factíveis:

Selecionar deterministicamente $f_c \cdot n_{pop,A}$ soluções a partir de *PopA* (SD_1)

Armazenar em *PopC* as soluções selecionadas em SD_1

Aplicar Cruzamento entre todas as soluções de *PopC* e as seguintes:

n_{cruz} soluções selecionadas por roleta de *PopA* (SR_1)

n_{cruz} soluções selecionadas por roleta de *PopB* (SR_2)

n_{cruz} soluções de uma viagem selecionada aleatoriamente da TH

Retirar de *PopC* as soluções selecionadas em SD_1

Armazenar em *PopC* os filhos dos cruzamentos

Aplicar mutação em *PopC*

Selecionar deterministicamente $f_b \cdot n_{pop,C}$ soluções de *PopC* (SD_2)

Transferir soluções selecionadas em SD_2 para *PopB*

Selecionar por roleta $f_a \cdot n_{pop,C}$ soluções de *PopC* (SR_3)

Transferir soluções selecionadas em SR_3 para *PopA*

Aplicar Seleções Roleta em *PopA* e *PopB* para excluir soluções que estejam acima das capacidades populacionais $n_{pop,A}$ e $n_{pop,B}$ (SR_4, SR_5)

Limpar *PopC*

(Fim do laço principal)

Fim

4 RESULTADOS

O algoritmo proposto foi executado para a linha 165 (COHAB) da cidade de Porto Alegre, cuja TH utilizada consta no Anexo A. Essa TH é a utilizada por Fuentes (2010), referente à operação em dias úteis.

A linha 165 liga a Zona Sul da cidade ao Centro, tendo seu terminal Bairro na Rua Ênio Aveline da Rocha, Bairro Vila Nova, e seu terminal Centro na Av. Salgado Filho. É operada pelo Consórcio Viva Sul. O itinerário da linha é apresentado na Figura 10.

Figura 10 – Itinerário da Linha 165 - Sentido BC (esquerda) e CB (direita)

Foram realizadas quatro execuções independentes do algoritmo, alterando nestas o valor da probabilidade de mutação p_m e o valor do coeficiente de ponderação do custo meta γ a fim de analisar a influência de cada um desses parâmetros no algoritmo. Os diferentes parâmetros utilizados em cada simulação são apresentados na Tabela 1.

Tabela 1 - Parâmetros variáveis utilizados nas execuções do algoritmo

Execução	Valores dos coeficientes	
	γ	p_m
1	3,5	10%
2	3,5	20%
3	2	10%
4	2	20%

Os restantes parâmetros são iguais em todas as execuções e são apresentados na Tabela 2.

Tabela 2 - Parâmetros fixos utilizados nas execuções do algoritmo

Parâmetro		Valor
$n_{pop,A}$	Capacidade máxima da População A	10
$n_{pop,B}$	Capacidade máxima da População B	5
n_{fac}	Número exigido de soluções factíveis	4
f_a	Fator de seleção da PopC para a PopA	0,3
f_b	Fator de seleção da PopC para a PopB	0,3
f_c	Fator de seleção da PopA para a PopC	0,3
n_{cruz}	Quantidade de soluções escolhidas para cruzamento	1
α	Coef. de ponderação do custo do número de serviços	1,5
τ	Coef. de ponderação do custo da folga interna	1,5
δ	Coef. de ponderação do custo do horário de pico	0,5
p_{roleta}	Probab. mínima de seleção roleta das soluções de maior FC	8%
	Duração de jornada	7h30min
	Duração de intervalo de descanso	30min
	Duração de intervalos de início e fim	30min

O resultados das soluções serão apresentados em pares de execuções nas seções a seguir.

4.1 EXECUÇÕES 1 E 2

Buscando analisar qual o impacto no algoritmo de diferentes valores para o parâmetro de probabilidade de mutação p_m, foram utilizados os valores 10% e 20% nas execuções 1 e 2 respectivamente. O algoritmo é executado até serem encontradas n_{fac} = 4 soluções factíveis. A Execução 1 atingiu esse objetivo após 4715 iterações, já a Execução 2 o atingiu em 5610 iterações. A Execução 1 levou 1h12min para ser processada e a Execução 2 levou 1h49min.

A Figura 11 mostra a evolução das funções de custo ao longo dessas duas execuções do algoritmo. A cada iteração foi plotada a solução com o menor valor da FC.

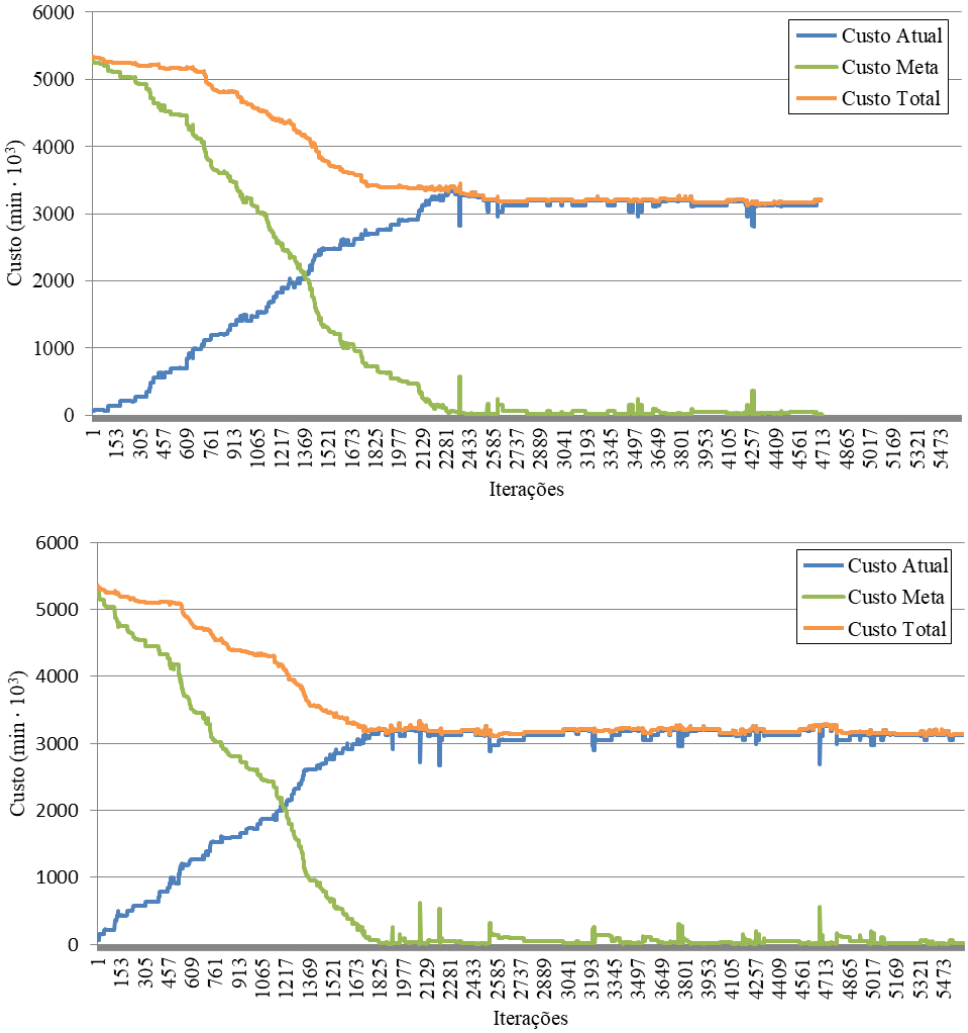


Figura 11 - Convergência do algoritmo com γ = 3,5
Execução 1 (p_m = 10%) (a) e Execução 2 (p_m = 20%) (b)

Percebe-se na Figura 11 que a segunda etapa do algoritmo é atingida em aproximadamente 2281 iterações na Execução 1 e 1825 iterações na Execução 2. Em ambos os casos, quando a segunda etapa é atingida, momento no qual o Custo Meta atinge zero, o Custo Atual e o Total coincidem, o Atual atinge seu máximo e o Total atinge seu mínimo. Assim permanecem em patamares, com pouca alteração, exceto por algumas irregularidades em forma de picos, como próximo às iterações 4257 para (a) e 4713 para (b). Elas correspondem às soluções que atenderam ao critério de quantidade de viagens, mas não tinham o intervalo alocado corretamente, portanto tiveram todos os serviços incorretos retirados.

A pouca variação na segunda fase, com inclusive um leve aumento no custo, mostra que essa fase não é muito interessante de ser aplicada. Utilizar o algoritmo apenas até que se encerre a primeira fase pode levar a soluções suficientemente boas em menos tempo.

A Tabela 3 mostra os custos das soluções factíveis obtidas nas duas execuções do algoritmo, bem como em qual iteração da execução cada solução foi criada e quantos serviços foram criados para cobrir a TH.

Tabela 3 – Dados das soluções obtidas

Exec.	Sol.	Iteração de criação	Nº de serviços	Custo (hh:mm)	Folgas Internas (hh:mm)	Folgas Externas (hh:mm)	Folgas Totais (hh:mm)
1	1	2652	46	908:20	63:36	14:26	78:02
	2	4449	46	906:34	60:05	17:57	78:02
	3	4713	45	886:56	55:18	16:44	72:02
	4	4715	45	886:20	54:07	17:55	72:02
2	5	5044	45	887:09	55:45	16:17	72:02
	6	5073	45	888:39	58:44	13:18	72:02
	7	5250	45	888:00	57:26	14:36	72:02
	8	5610	45	887:00	55:26	16:36	72:02

A Execução 1, com menor valor de p_m, em relação à Execução 2, levou mais iterações para encerrar a etapa construtiva, levou menos iterações para obter as 4 soluções factíveis e obteve algumas soluções com menor qualidade, característica representada por um maior custo (FC).

O custo total é representado em minutos apenas por decorrência de suas principais variáveis serem em minutos. É uma grandeza relativa e não carrega significado temporal real, já que existem coeficientes de preponderância multiplicando as suas parcelas formadoras. Ainda assim, vale lembrar que esse custo representa apenas uma minimização da quantidade de folgas internas, posteriormente uma minimização de folgas externas, e uma minimização de número de serviços. A parcela de horário de pico é constante em todas as soluções completas, podendo ser ignorada. Além disso, para uma solução completa o custo meta é sempre zero. Assim, para todas as soluções que têm a mesma quantidade de serviços, como é o caso das soluções 3 a 8, o tempo real de folgas é sempre o mesmo, já que todo serviço tem a mesma duração de jornada e todas as viagens e intervalos têm durações fixas. Isso faz com que a única diferença entre as soluções de 1 a 6 seja a proporção entre folgas internas e externas.

As soluções factíveis surgem no decorrer do algoritmo, ou seja, não foram criadas todas juntas na mesma iteração. Também não estão aqui apresentadas na ordem em que foram criadas, apenas em ordem de custo e agrupadas por execução do algoritmo.

O fato de o ACE oferecer como saída uma população de soluções é vantajoso para que possam ser comparadas diferentes combinações de custos que resultam num mesmo custo final, podendo o operador realizar a escolha ente características que ainda não são levadas em conta pela FC, como a distribuição das folgas e dos intervalos de descanso.

Tais características podem ser observadas em representações visuais das soluções. As soluções obtidas são representadas da Figura 12 à Figura 21 abaixo usando gráficos de Gantt. Cada linha corresponde a um serviço, retângulos roxos são intervalos iniciais e finais, retângulos vermelhos são viagens e retângulos amarelos são o intervalo de descanso.

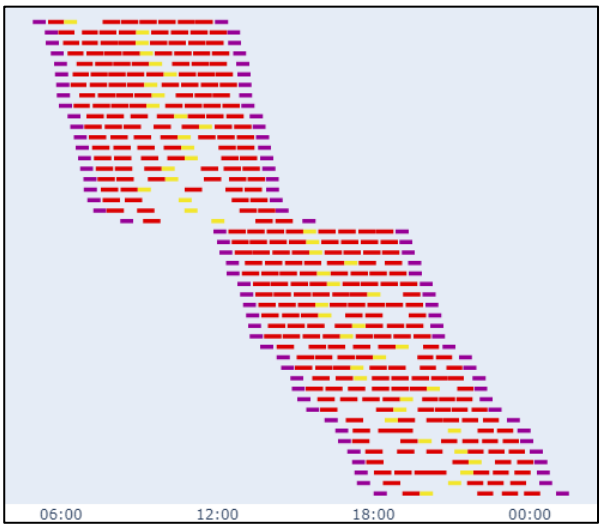


Figura 12 - Execução 1 - Solução 1

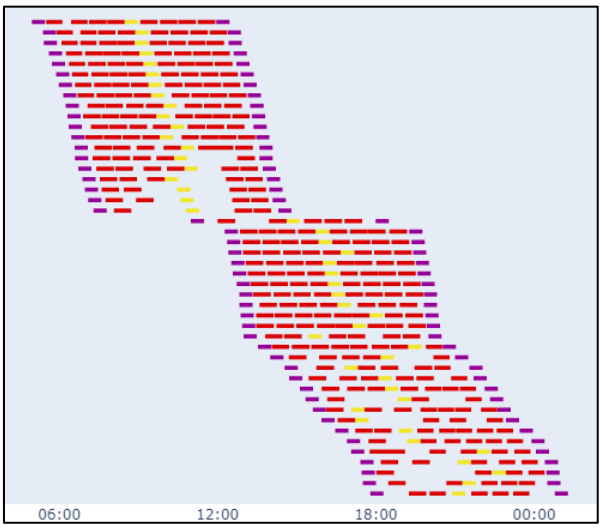


Figura 13 - Execução 1 - Solução 2

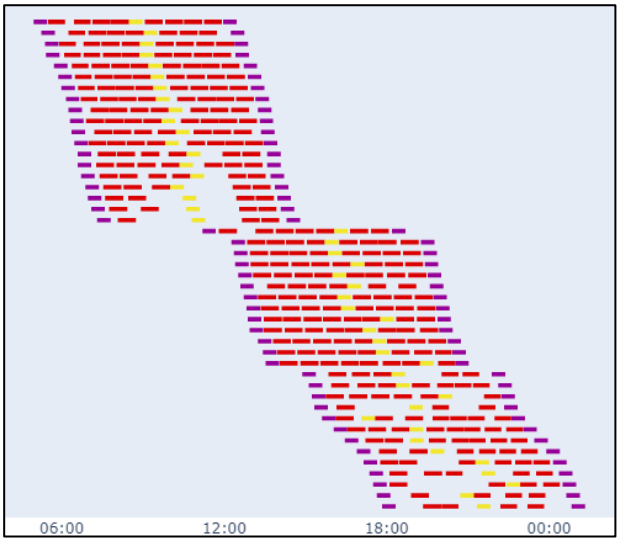


Figura 14 – Execução 1 - Solução 3

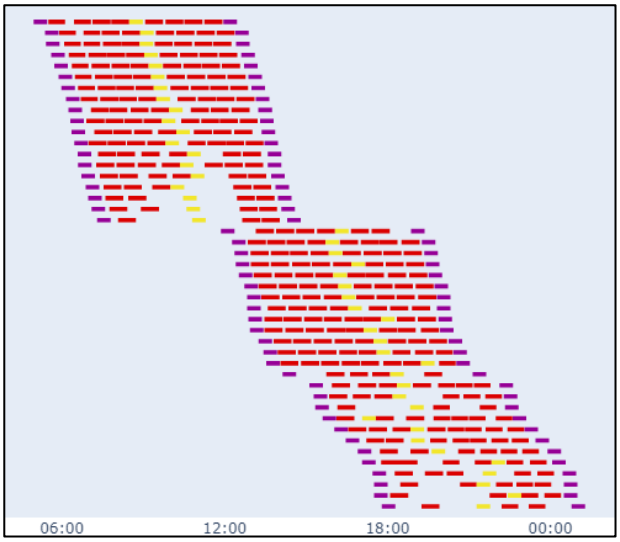


Figura 15 - Execução 1 - Solução 4

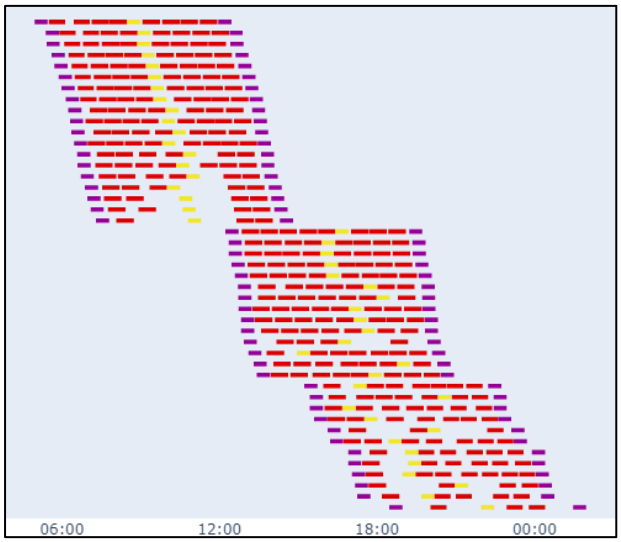


Figura 16 – Execução 2 - Solução 5

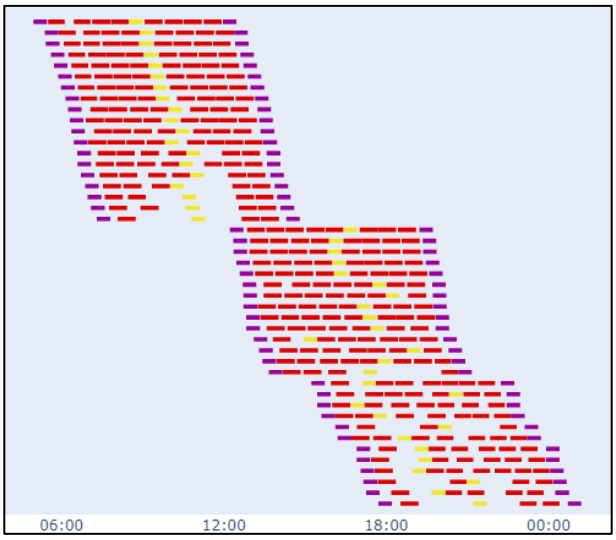


Figura 17 - Execução 2 - Solução 6

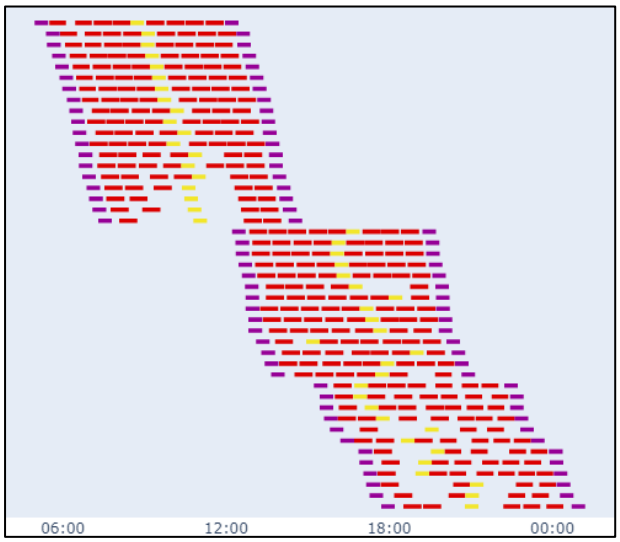


Figura 18 - Execução 2 - Solução 7

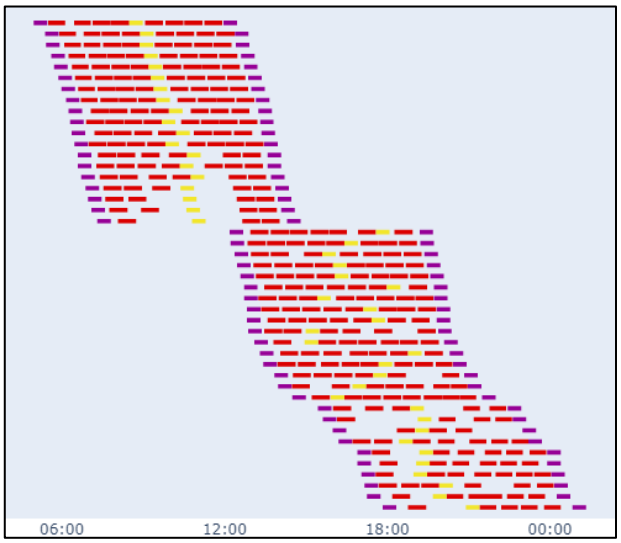


Figura 19 - Execução 2 - Solução 8

Observando os gráficos de Gantt, percebe-se que a FC cumpre seu papel em representar características preferíveis de uma solução como um custo menor. A Solução 4 tem o menor valor para FC e tem seus serviços e viagens mais organizados e compactos.

Ainda assim, tendo em vista que a FC leva em conta na sua minimização apenas quantidade de tempo ocioso e quantidade de serviços, é possível observar outras características que poderiam ser consideradas preferíveis ou não, mas que não são consideradas na FC. Um exemplo é a alocação do horário de descanso: uma solução pode ter custo mais baixo, mas tem horários de descanso muito distantes da metade dos serviços. Outra característica a observar é que todos os serviços tenham aproximadamente a mesma quantidade de viagens. Essas e outras características preferíveis podem ser incorporadas à FC em trabalhos futuros.

4.2 EXECUÇÕES 3 E 4

As execuções 3 e 4 foram realizadas utilizando respectivamente os mesmos valores de p_m utilizados nas execuções 1 e 2, porém foi alterado o valor γ de 3,5 para 2. A Figura 20 mostra a evolução das funções de custo ao longo dessas duas execuções do algoritmo, do mesmo modo que a Figura 11 para as execuções anteriores.

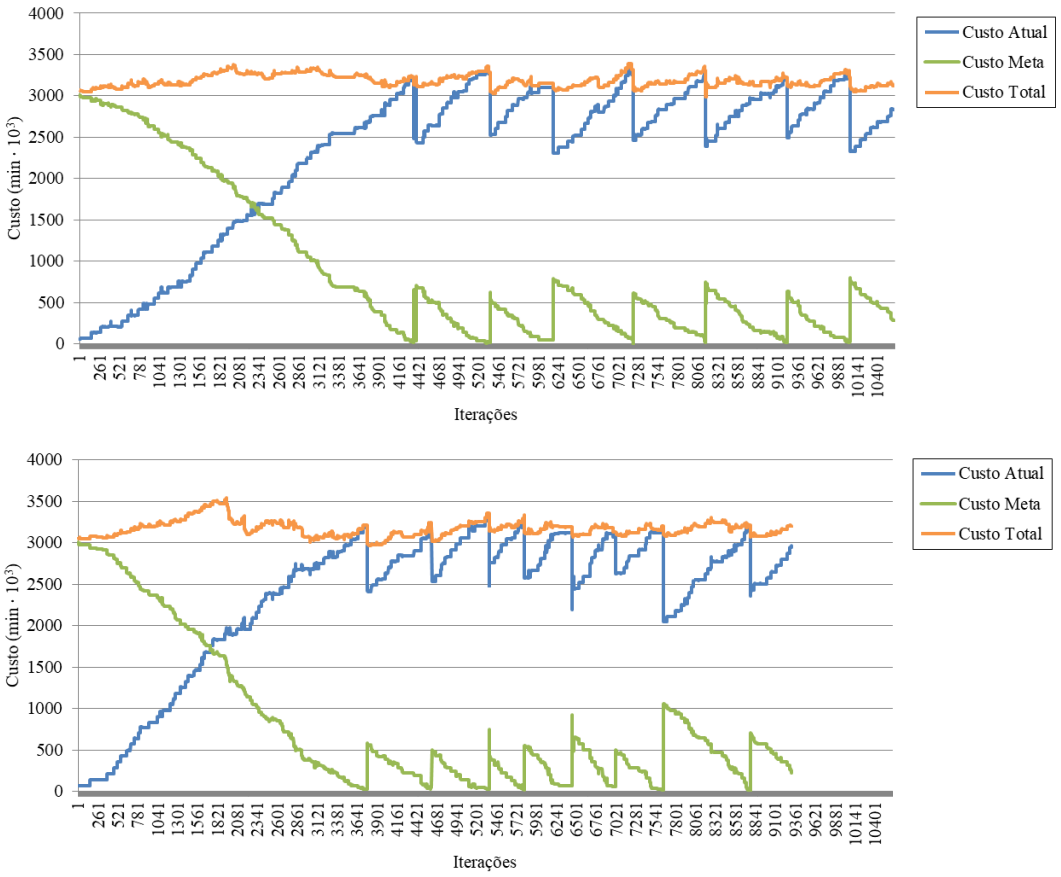


Figura 20 - Convergência do algoritmo com γ = 2
Execução 3 (p_m = 10%) (a) e Execução 4 (p_m = 20%) (b)

Diferentemente das execuções 1 e 2, que levaram menos de 2 horas para encontrarem 4 soluções factíveis, as execuções 3 e 4 após 4 horas de execução ainda não tinham gerado nenhuma solução factível, por isso foram interrompidas. A Figura 20 mostra a diferença drástica dos gráficos da FC para essas execuções em relação às anteriores, demonstrando a necessidade de uma correta calibração do coeficiente γ.

Os vários picos do Custo Meta (CM) na segunda fase do algoritmo, como nas execuções anteriores, correspondem às iterações onde uma solução completa é encontrada, mas não

satisfaz às condições necessárias para ser uma solução factível. Quando isso acontece, alguns de seus serviços são excluídos. Os picos são mais numerosos nessas execuções que nas anteriores, além de serem mais expressivos.

O coeficiente de ponderação γ faz com que o CM tenha um módulo suficientemente alto para que os operadores de seleção considerem vantajosa a adição de viagens aos serviços. Como essa adição aumenta o Custo Atual (CA), é necessário que a diminuição de CM seja maior em módulo que o aumento de CA. Nas execuções 3 e 4, CM não tem um módulo suficientemente grande, pois $\gamma = 2$ é um valor muito baixo. Assim, quando o algoritmo está perto de gerar uma solução completa, essa solução já tem o FC alto demais para que sejam feitas as melhorias necessárias.

5 CONSIDERAÇÕES FINAIS

Nesse trabalho, a metodologia utilizada permitiu resolver adequadamente o problema caracterizado e atingiu os objetivos propostos. A construção do algoritmo em Python permitiu a familiarização com as particularidades dessa linguagem de programação.

Foram realizadas revisões bibliográficas nas áreas de Programação de Tripulações, Algoritmo Construtivo-Evolutivo e Algoritmos Genéticos. É bastante popular o uso de métodos integrados, principalmente o PPVT, e de métodos de resolução híbridos, utilizando abordagens aproximativas e exatas. Uma possibilidade futura para esse trabalho é também fazer uso de uma abordagem híbrida, como por exemplo na etapa do algoritmo onde uma solução completa tem alguns de seus serviços excluídos. Nesse momento é possível utilizar uma abordagem 2-opt. Outra possibilidade é que, para permitir a reconstrução de uma solução que não esteja construída com boas características, pode ser implementada uma variação da mutação que permita a exclusão de viagens.

Durante a realização desse trabalho, ficaram evidentes vários caminhos possíveis para melhorias futuras. Tendo em vista que o trabalho de Fuentes (2010) faz também uma atribuição de veículos, uma próxima etapa natural é a implementação dessa parte do modelo, de modo integrado, bem como a utilização de frota heterogênea.

Outro ponto passível de desenvolvimento é utilizar a FC para uma representação real de custo monetário, já que o objetivo final é que um baixo custo da função de desempenho corresponda a um baixo custo financeiro.

Quanto à operação, ainda há pontos que podem ser mais bem representados, como permitir jornadas de trabalho com durações flexíveis, horas extras, sempre de modo que corresponda ao real custo que isso possa acarretar. Há também a possibilidade de permitir a troca de tripulações entre linhas e, em uma implementação PPVT, troca entre veículos. Além disso, o gerenciamento de viagens garagem-terminal e viagens expressas terminal-terminal, ao ser implementado também permite maior flexibilidade e eficiência no cumprimento da Tabela Horária.

A aplicação dos dados reais de Tabela Horária mostrou que há margem para uma melhor calibração dos parâmetros utilizados, especialmente os coeficientes de ponderação da FC, podendo fazer com que a convergência ocorra mais rapidamente ou até encontre soluções factíveis mais eficientes. Além disso, podem ser utilizadas outras linhas e é possível testar o funcionamento concomitante para duas linhas que compartilhem um terminal. Em Porto Alegre, por exemplo, podem ser utilizadas duas linhas cujo terminal Centro seja na Avenida Salgado Filho, permitindo a troca de tripulações. Seriam apenas necessários ajustes relativos ao tempo necessário para tal transição.

Observando o tempo gasto para esse tipo de operação, fica evidente a ligação desse problema com o da Tabela Horária. Por mais que haja uma alocação muito eficiente e exata das viagens para as tripulações, ela perde o sentido se não vem acompanhada de uma interação adequada com as outras etapas, como a definição de Tabela Horária e o rodízio de tripulações. Do mesmo modo que se observam lacunas na definição de escalas de tripulação, em Porto Alegre também há problemas relacionados a atrasos nas viagens e incorreta medição dos tempos das viagens, causados por eventos como congestionamentos. Isso gera discrepância entre o modelo estabelecido e a realidade, além de causar sobrecarga na tripulação visto a impossibilidade de cumprir as tabelas.

O programa criado pode ser aplicado a qualquer cidade e linha, bastando serem alterados os valores relacionados aos requisitos trabalhistas.

REFERÊNCIAS

BARBOSA, Gisele Heloise; KERBAUY, Maria Teresa Miceli. Os protestos de junho de 2013: movimentos sociais e reivindicações. **10º Encontro da Associação Brasileira de Ciência Política Ciência Política e a Política: memória e futuro**, [s. l.], 2016. Disponível em: <http://www.encontroabcp2016.cienciapolitica.org.br/resources/anais/5/1468352175_ARQUIVO_GiseleHeloiseBarbosaABCP.pdf>

BARBOSA, Vítor; RESPÍCIO, Ana; ALVELOS, Filipe. A Column Generation Based Heuristic for a Bus Driver Rostering Problem. **Artificial Intelligence in Engineering**, [s. l.], v. 1, n. 1, p. 62, 2003.

BERTOSSI, A. A.; CARRARESI, P.; GALLO, G. On some matching problems arising in vehicle scheduling models. **Networks**, [s. l.], v. 17, n. 3, p. 271–281, 1987.

BLUM, Christian; ROLI, Andrea. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. **ACM Computing Surveys**, [s. l.], v. 35, n. 3, p. 268–308, 2003.

BURKE, Edmund et al. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. **Handbook of Metaheuristics**, [s. l.], n. May 2014, p. 457–474, 2006.

CAO, Zhichao; CEDER, Avishai. Autonomous shuttle bus service timetabling and vehicle scheduling using skip-stop tactic. **Transportation Research Part C: Emerging Technologies**, [s. l.], v. 102, n. March, p. 370–395, 2019. Disponível em: <https://doi.org/10.1016/j.trc.2019.03.018>

CARRARESI, Paolo; NONATO, Maddalena; GIRARDI, Leopoldo. Network Models, Lagrangean Relaxation and Subgradients Bundle Approach in Crew Scheduling Problems. [s. l.], p. 188–212, 2011.

CEDER, Avishai et al. A case study of Beijing bus crew scheduling: a variable neighborhood-based approach. **Journal of Advanced Transportation**, [s. l.], v. 50, p. 434–445, 2016.

CHEN, Mingming; NIU, Huimin. A model for bus crew scheduling problem with multiple duty types. **Discrete Dynamics in Nature and Society**, [s. l.], v. 2012, 2012.

CIANCIO, Claudio et al. An integrated algorithm for shift scheduling problems for local public transport companies. **Omega (United Kingdom)**, [s. l.], v. 75, p. 1339–1351, 2018.

DIAS, T. G.; DE SOUSA, J. P.; CUNHA, J. F. Genetic algorithms for the bus driver scheduling problem: A case study. **Journal of the Operational Research Society**, [s. l.], v. 53, n. 3, p. 324–335, 2002.

ELIZONDO, Rafael et al. An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport. **European Biophysics Journal**, [s. l.], 2009.

FUENTES, Karen. **Generación de Servicios de Conducción Para Líneas de Buses Urbanos**. 2010. USACH, [s. l.], 2010.

GOMES, A.; PARADA, Victor. Métodos de I.A. Aplicados ao Problema de Cutting-Stock. In: ANAIS DE WORKSHOP INTERNACIONAL EM CONTROLE INTELIGENTE 1991,

Vitória. **Anais**... Vitória

GUEDES, Pablo Cristini et al. Vehicle scheduling problem with loss in bus ridership. **Computers and Operations Research**, [s. l.], v. 111, p. 230–242, 2019. Disponível em: <https://doi.org/10.1016/j.cor.2019.07.002>

GUO, Congcong; WANG, Chunlu; ZUO, Xingquan. A genetic algorithm based column generation method for multi-depot electric bus vehicle scheduling. **GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion**, [s. l.], p. 367–368, 2019.

HORVÁTH, Markó; KIS, Tamás. Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price. **Central European Journal of Operations Research**, [s. l.], v. 27, n. 1, p. 39–67, 2019.

IBARRA-ROJAS, Omar J. et al. Planning, operation, and control of bus transport systems: A literature review. **Transportation Research Part B: Methodological**, [s. l.], v. 77, p. 38–75, 2015. Disponível em: <http://dx.doi.org/10.1016/j.trb.2015.03.002>

JOHAR, Amita; JAIN, S. S.; GARG, P. K. Transit network design and scheduling using genetic algorithm – a review. **An International Journal of Optimization and Control: Theories & Applications (IJOCTA)**, [s. l.], v. 6, n. 1, p. 9–22, 2016.

KANG, Liujiang; CHEN, Shukai; MENG, Qiang. Bus and driver scheduling with mealtime windows for a single public bus route. **Transportation Research Part C: Emerging Technologies**, [s. l.], v. 101, n. February, p. 145–160, 2019. Disponível em: <https://doi.org/10.1016/j.trc.2019.02.005>

LI, Hong et al. A Column Generation Based Hyper-Heuristic to the Bus Driver Scheduling Problem. **Discrete Dynamics in Nature and Society**, [s. l.], v. 2015, p. 1–10, 2015.

LOURENÇO, Helena R.; PAIXÃO, José P.; PORTUGAL, Rita. Multiobjective Metaheuristics for the Bus Driver Scheduling Problem. **Transportation Science**, [s. l.], v. 35, n. 3, p. 331–343, 2003.

MESQUITA, Marta et al. A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern. **European Journal of Operational Research**, [s. l.], v. 229, n. 2, p. 318–331, 2013. Disponível em: <http://dx.doi.org/10.1016/j.ejor.2013.02.055>

MESQUITA, Marta; PAIAS, Ana. Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. **Computers and Operations Research**, [s. l.], v. 35, n. 5, p. 1562–1575, 2008.

MORENO, César Augusto Marín et al. Heuristic constructive algorithm for work-shift scheduling in bus rapid transit systems. **Decision Science Letters**, [s. l.], v. 8, n. 4, p. 519–530, 2019. a.

MORENO, César Augusto Marín et al. A hybrid algorithm for the multi-depot vehicle scheduling problem arising in public transportation. **International Journal of Industrial Engineering Computations**, [s. l.], v. 10, n. 3, p. 361–374, 2019. b.

NILSSON, Nils J. **Principles of Artificial Intelligence**. [s.l: s.n.]. v. PAMI-3

OLIVEIRA, Arthur et al. Panorama da mobilidade urbana: diagnóstico e propostas para o

transporte público por ônibus. **Comunicações Técnicas do 19o. Congresso Brasileiro de Transporte e Trânsito**, [s. l.], p. 1–9, 2013. Disponível em: <http://files-server.antp.org.br/_5dotSystem/download/dcmDocument/2013/10/06/FC408A57-1378-4B7D-B948-42A6C4833224.pdf>

PALMA, R. **Solución al problema de corte de piezas no guillotina utilizando un método híbrido constructivo-evolutivo**. 1997. USACH, [s. l.], 1997.

PEARL, Judea. **Heuristique: Strategies de Recherche Intelligente pour la Resolution de Problemes par Ordinateur**. [s.l.] : Addison-Wesley, 1990.

PRATA, Bruno de Athayde. A hybrid Genetic Algorithm for the vehicle and crew scheduling in mass transit systems. **IEEE Latin America Transactions**, [s. l.], v. 13, n. 9, p. 3020–3025, 2015.

REIS, Jorge von Atzingen Dos. **Heurísticas Baseadas em Busca em Vizinhança Variável para o Problema de Programação Intregrada de Veículos e Tripulações no Transporte Coletivo Urbano por Ônibus**. 2008. Universidade de São Paulo, [s. l.], 2008.

REIS, Walison Reis; COSTA, Abimael de Jesus Barros. A composição dos custos do Sistema de Transporte Coletivo em Municípios. In: XXIV CONGRESSO BRASILEIRO DE CUSTOS 2017, Florianópolis, SC, Brasil. **Anais**... Florianópolis, SC, Brasil

SCIPY.ORG. **numpy.random.choice**. 2018. Disponível em: <https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.choice.html>. Acesso em: 26 dez. 2019.

SILVA, Gustavo Peixoto; PRATES, Raphael Felipe de Carvalho. Otimização da escala mensal de motoristas de ônibus urbano utilizando a heurística Variable Neighborhood Search. **Transportes**, [s. l.], v. 22, n. 1, p. 31, 2014.

SIMÕES, Emiliana Mara Lopes. **Algoritmo para programação integrada de veículos e tripulações no sistema de transporte público por ônibus**. 2009. Universidade Federal de Minas Gerais, [s. l.], 2009.

SINDICATO DAS EMPRESAS DE ÔNIBUS DE PORTO ALEGRE, SEOPA. Processo de Reajuste Tarifário 2019 - Parte 01**Processo SEI 19.16.000004560-1**, 2019. p. 3–20. Disponível em: <http://lproweb.procempa.com.br/pmpa/prefpoa/eptc/usu_doc/01_sei_19.16.000004560_1.pdf>

SONG, Cuiying et al. Improved genetic algorithm with gene recombination for bus crew-scheduling problem. **Mathematical Problems in Engineering**, [s. l.], v. 2015, 2015.

UVARAJA, Vikneswary; LEE, Lai Soon; NOR, Nor Aliza. Multiple Tabu Search for multiobjective Urban Transit Scheduling Problem. **ASM Science Journal**, [s. l.], v. 12, n. Special Issue 1, p. 150–173, 2019.

WANG, Jiao et al. Tabu search for bus crew scheduling with feasible sets of shifts. **Advanced Materials Research**, [s. l.], v. 989–994, p. 2523–2527, 2014.

WREN, Anthony; ROUSSEAU, Jean-Marc. Bus Driver Scheduling -An Overview. **Computer-Aided Transit Scheduling**, [s. l.], p. 173–187, 1995.

ANEXO A – Tabela Horária da Linha 165 (Cohab)

idv	hi	hf	ti	tf
1	05:30	06:07	B	C
2	05:54	06:31	B	C
3	06:15	06:52	B	C
4	06:22	06:59	B	C
5	06:29	07:06	B	C
6	06:36	07:13	B	C
7	06:42	07:19	B	C
8	06:48	07:25	B	C
9	06:54	07:34	B	C
10	06:59	07:39	B	C
11	07:04	07:44	B	C
12	07:08	07:48	B	C
13	07:12	07:52	B	C
14	07:16	07:56	B	C
15	07:20	08:00	B	C
16	07:24	08:04	B	C
17	07:28	08:08	B	C
18	07:32	08:12	B	C
19	07:36	08:16	B	C
20	07:40	08:20	B	C
21	07:45	08:25	B	C
22	07:50	08:30	B	C
23	07:55	08:35	B	C
24	08:00	08:40	B	C
25	08:06	08:46	B	C
26	08:12	08:52	B	C
27	08:18	08:58	B	C
28	08:24	09:04	B	C
29	08:32	09:12	B	C
30	08:40	09:20	B	C
31	08:48	09:28	B	C
32	08:56	09:36	B	C
33	09:04	09:44	B	C
34	09:12	09:52	B	C
35	09:20	10:00	B	C
36	09:28	10:08	B	C
37	09:36	10:16	B	C
38	09:44	10:24	B	C
39	09:52	10:32	B	C
40	10:00	10:40	B	C
41	10:08	10:48	B	C
42	10:16	10:56	B	C
43	10:24	11:04	B	C
44	10:31	11:11	B	C
45	10:38	11:18	B	C
46	10:45	11:25	B	C
47	10:52	11:32	B	C
48	10:58	11:38	B	C
49	11:04	11:44	B	C
50	11:10	11:50	B	C
51	11:16	11:56	B	C
52	11:22	12:02	B	C
53	11:28	12:08	B	C
54	11:35	12:15	B	C
55	11:42	12:22	B	C
56	11:49	12:29	B	C
57	11:56	12:36	B	C

idv	hi	hf	ti	tf
58	12:03	12:43	B	C
59	12:09	12:49	B	C
60	12:15	12:55	B	C
61	12:21	13:01	B	C
62	12:27	13:07	B	C
63	12:33	13:13	B	C
64	12:39	13:19	B	C
65	12:45	13:25	B	C
66	12:51	13:31	B	C
67	12:57	13:37	B	C
68	13:03	13:43	B	C
69	13:09	13:49	B	C
70	13:15	13:55	B	C
71	13:21	14:01	B	C
72	13:28	14:08	B	C
73	13:35	14:15	B	C
74	13:42	14:22	B	C
75	13:50	14:30	B	C
76	13:57	14:37	B	C
77	14:03	14:43	B	C
78	14:10	14:50	B	C
79	14:17	14:57	B	C
80	14:24	15:04	B	C
81	14:31	15:11	B	C
82	14:38	15:18	B	C
83	14:45	15:25	B	C
84	14:52	15:32	B	C
85	14:59	15:39	B	C
86	15:06	15:46	B	C
87	15:13	15:53	B	C
88	15:20	16:00	B	C
89	15:27	16:07	B	C
90	15:33	16:13	B	C
91	15:39	16:19	B	C
92	15:45	16:25	B	C
93	15:51	16:31	B	C
94	15:57	16:37	B	C
95	16:03	16:43	B	C
96	16:09	16:49	B	C
97	16:15	16:55	B	C
98	16:21	17:01	B	C
99	16:27	17:07	B	C
100	16:33	17:13	B	C
101	16:39	17:19	B	C
102	16:45	17:25	B	C
103	16:51	17:31	B	C
104	16:56	17:36	B	C
105	17:01	17:41	B	C
106	17:06	17:46	B	C
107	17:11	17:51	B	C
108	17:12	17:52	B	C
109	17:19	17:59	B	C
110	17:26	18:06	B	C
111	17:31	18:11	B	C
112	17:34	18:14	B	C
113	17:41	18:21	B	C
114	17:42	18:22	B	C

idv	hi	hf	ti	tf
115	17:43	18:23	B	C
116	17:51	18:31	B	C
117	17:56	18:36	B	C
118	18:01	18:41	B	C
119	18:06	18:46	B	C
120	18:11	18:51	B	C
121	18:16	18:56	B	C
122	18:20	19:00	B	C
123	18:22	19:02	B	C
124	18:28	19:08	B	C
125	18:34	19:14	B	C
126	18:41	19:21	B	C
127	18:48	19:28	B	C
128	18:54	19:34	B	C
129	19:01	19:41	B	C
130	19:07	19:47	B	C
131	19:15	19:55	B	C
132	19:23	20:03	B	C
133	19:32	20:12	B	C
134	19:42	20:19	B	C
135	19:52	20:29	B	C
136	20:02	20:39	B	C
137	20:13	20:50	B	C
138	20:24	21:01	B	C
139	20:36	21:13	B	C
140	20:48	21:25	B	C
141	21:00	21:37	B	C
142	21:11	21:48	B	C
143	21:24	22:01	B	C
144	21:37	22:14	B	C
145	21:50	22:27	B	C
146	22:00	22:37	B	C
147	22:10	22:47	B	C
148	22:23	23:00	B	C
149	22:40	23:17	B	C
150	22:56	23:33	B	C
151	23:12	23:49	B	C
152	23:25	00:02	B	C
153	06:05	06:42	C	B
154	06:27	07:04	C	B
155	06:48	07:25	C	B
156	06:59	07:39	C	B
157	07:06	07:46	C	B
158	07:13	07:53	C	B
159	07:19	07:59	C	B
160	07:25	08:05	C	B
161	07:31	08:11	C	B
162	07:36	08:16	C	B
163	07:41	08:21	C	B
164	07:45	08:25	C	B
165	07:49	08:29	C	B
166	07:54	08:34	C	B
167	08:01	08:41	C	B
168	08:02	08:42	C	B
169	08:04	08:44	C	B

idv	hi	hf	ti	tf
170	08:08	08:48	C	B
171	08:13	08:53	C	B
172	08:17	08:57	C	B
173	08:22	09:02	C	B
174	08:27	09:07	C	B
175	08:32	09:12	C	B
176	08:37	09:17	C	B
177	08:43	09:23	C	B
178	08:49	09:29	C	B
179	08:55	09:35	C	B
180	09:01	09:41	C	B
181	09:09	09:49	C	B
182	09:17	09:57	C	B
183	09:25	10:05	C	B
184	09:33	10:13	C	B
185	09:41	10:21	C	B
186	09:49	10:29	C	B
187	09:57	10:37	C	B
188	10:05	10:45	C	B
189	10:13	10:53	C	B
190	10:21	11:01	C	B
191	10:29	11:09	C	B
192	10:37	11:17	C	B
193	10:45	11:25	C	B
194	10:53	11:33	C	B
195	11:01	11:41	C	B
196	11:08	11:48	C	B
197	11:15	11:55	C	B
198	11:22	12:02	C	B
199	11:29	12:09	C	B
200	11:35	12:15	C	B
201	11:42	12:22	C	B
202	11:48	12:28	C	B
203	11:54	12:34	C	B
204	12:00	12:40	C	B
205	12:06	12:46	C	B
206	12:12	12:52	C	B
207	12:19	12:59	C	B
208	12:26	13:06	C	B
209	12:33	13:13	C	B
210	12:40	13:20	C	B
211	12:46	13:26	C	B
212	12:52	13:32	C	B
213	12:58	13:38	C	B
214	13:04	13:44	C	B
215	13:10	13:50	C	B
216	13:16	13:56	C	B
217	13:22	14:02	C	B
218	13:28	14:08	C	B
219	13:34	14:14	C	B
220	13:42	14:22	C	B
221	13:48	14:28	C	B
222	13:54	14:34	C	B
223	14:00	14:40	C	B
224	14:07	14:47	C	B

idv	hi	hf	ti	tf
225	14:14	14:54	C	B
226	14:21	15:01	C	B
227	14:29	15:09	C	B
228	14:36	15:16	C	B
229	14:42	15:22	C	B
230	14:49	15:29	C	B
231	14:56	15:36	C	B
232	15:03	15:43	C	B
233	15:10	15:50	C	B
234	15:17	15:57	C	B
235	15:24	16:04	C	B
236	15:31	16:11	C	B
237	15:39	16:19	C	B
238	15:46	16:26	C	B
239	15:53	16:33	C	B
240	16:00	16:40	C	B
241	16:07	16:47	C	B
242	16:13	16:53	C	B
243	16:19	16:59	C	B
244	16:25	17:05	C	B
245	16:31	17:11	C	B
246	16:37	17:17	C	B
247	16:43	17:23	C	B
248	16:49	17:29	C	B
249	16:55	17:35	C	B
250	17:01	17:41	C	B
251	17:07	17:47	C	B
252	17:13	17:53	C	B
253	17:19	17:59	C	B
254	17:25	18:05	C	B
255	17:31	18:11	C	B
256	17:36	18:16	C	B
257	17:41	18:21	C	B
258	17:46	18:26	C	B
259	17:51	18:31	C	B
260	17:56	18:36	C	B
261	18:01	18:41	C	B
262	18:06	18:46	C	B
263	18:11	18:51	C	B
264	18:16	18:56	C	B

idv	hi	hf	ti	tf
265	18:21	19:01	C	B
266	18:26	19:06	C	B
267	18:31	19:11	C	B
268	18:36	19:16	C	B
269	18:41	19:21	C	B
270	18:46	19:26	C	B
271	18:51	19:31	C	B
272	18:56	19:36	C	B
273	19:02	19:42	C	B
274	19:08	19:48	C	B
275	19:14	19:54	C	B
276	19:21	20:01	C	B
277	19:28	20:08	C	B
278	19:34	20:11	C	B
279	19:41	20:18	C	B
280	19:47	20:24	C	B
281	19:54	20:31	C	B
282	20:02	20:39	C	B
283	20:11	20:48	C	B
284	20:21	20:58	C	B
285	20:31	21:08	C	B
286	20:40	21:17	C	B
287	20:51	21:28	C	B
288	21:02	21:39	C	B
289	21:13	21:50	C	B
290	21:25	22:02	C	B
291	21:36	22:13	C	B
292	21:46	22:23	C	B
293	21:59	22:36	C	B
294	22:12	22:49	C	B
295	22:25	23:02	C	B
296	22:35	23:12	C	B
297	22:45	23:22	C	B
298	22:58	23:35	C	B
299	23:15	23:52	C	B
300	23:31	00:08	C	B
301	23:47	00:24	C	B