PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

LPBasic.jl - Planejamento de Linhas de Transporte Público em Julia/JuMP

André Mazal Krauss

Relatório Final de PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Ciência da Computação

Rio de Janeiro, 01 de Dezembro de 2019

André Mazal Krauss

LPBasic.jl - Planejamento de Linhas de Transporte Público em Julia/JuMP

Relatório Final de Projeto Final, apresentado ao curso Bacharelado em Ciência da Computação da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Marcus Poggi

Rio de Janeiro

Dezembro de 2019.

Resumo

A partir de um modelo selecionado da literatura relativa à otimização do planejamento de linhas de transporte público, descritos como problemas de otimização linear-inteira *(Mixed Integer Linear Programming/MILP)*, e da leitura de dados abertos disponibilizados em formato GTFS: são elaborados um pequeno número de variantes sobre o mesmo modelo base; é implementado um sistema em Julia/JuMP para a obtenção de soluções ótimas; são analisados os resultados para um número de instâncias.

Abstract

Based on a Mixed Linear Integer Programming(MILP) model selected from the literature on line planning for public transportation networks, and using openly available GTFS data: a small number of variants are derived from the initial model; an implementation for such optimization problems is developed using Julia/JuMP; and a number of results from different problem instances are analysed.

palavras-chave: planejamento de linhas, transporte público, programação linear-inteira

keywords: line planning, public transportation, mixed integer linear programming

Sumário

1 Introdução

1.1 Motivação e Domínio do Sistema

Há, na literatura científica, uma grande variedade de teses e trabalhos que se propõem a modelar e otimizar situações reais relacionadas ao setor de transportes. Entre esses problemas e suas respectivas modelagens matemáticas estão aqueles que dizem respeito ao planejamento, operação e utilização de redes de transportes públicos. A pesquisa científica nesta área é amplamente justificada: há o interesse público em uma rede que melhor contemple às necessidades da população; há o interesse dos operadores privados ou governamentais em minimizar custos operacionais; por último, há a necessidade de uma abordagem matemática-computacional pela natureza complexa do problema. No Brasil e, em particular, no Rio de Janeiro, a mobilidade urbana permanece um desafio.

Nesse contexto, proponho com este trabalho um estudo, uma implementação e a experimentação sobre um algoritmo para o problema do **planejamento de linhas** *(line planning)*, uma das etapas do processo de modelagem e otimização de linhas de transporte público.[1, p. 6-7] [2, p. 498]. Desenvolvi uma ferramenta para a leitura de dados de entrada e a subsequente resolução do sistema, resultando em uma solução comprovadamente ótima(ou em uma comprovação de insolubilidade). Tipicamente representa-se o problema como um de programação inteira mista (*Mixed Integer Linear Programming*/MILP). A ferramenta foi aplicada a instâncias do problema obtidas a partir de dados GTFS disponibilizados abertamente e, com isso, almejo sugerir a aplicabilidade das soluções obtidas nas redes consideradas.

Espera-se que a complexidade inerente e a grande aplicabilidade do problema descrito sejam, em parte, evidenciadas pela própria definição do mesmo, abaixo. Nela estão definidos conceitos que são, de maneira mais informal, já conhecidos por qualquer usuário de transportes públicos, o que ajuda a comunicar esta importância àqueles de fora do meio acadêmico.

Sobre o ambiente computacional escolhido para o desenvolvimento da solução, optei pela linguagem Julia e o framework JuMP[3] para otimização matemática. Os motivos para a escolha, inicialmente, limitaram-se a recomendações por parte do orientador do projeto e de supervisores meus no estágio. Posteriormente vim a confirmar o acerto da decisão ao constatar que consegui não só utilizar corretamente as funcionalidades da linguagem e do framework, como também tive acesso fácil a uma multitude de outras ferramentas e pacotes que auxiliaram no desenvolvimento do projeto, todas elas disponibilizadas abertamente(*open source*) e a maior parte correntemente desenvolvidas pela comunidade. Com isso, pude usufruir do ambiente open source da linguagem Julia e ainda possivelmente agregar valor com o meu próprio trabalho.

1.2 Situação Atual

Há hoje uma extensa bibliografia que estuda o problema do planejamento de linhas, da qual uma pequena parcela foi usada para fundamentar os conceitos acima. Para além dela, M. Bussieck [1, p. 15-17] e A. Schöbel[2, p. 499] fazem, em seus respectivos trabalhos, uma extensa compilação de referências sobre o tema. Estas datam a partir de 1925, e são, dentre outras:

1. PATZ [4], propõe um modelo que determina um Plano de Linhas com um *pênalti* reduzido
2. WEGEL [5], introduz a noção de requerimentos de frequência para as linhas
3. TORRES et al. , propõem um modelo em que não há restrições de frequência máxima e em que são representadas *linhas expressas*, que não têm obrigação de parar em todas estações.
4. SCHÖBEL E SCHOLL , apresentam uma abordagem que procura minimizar o tempo total gasto pelos passageiros, considerando a rede de transporte por inteiro

Sem entrar em maiores detalhes, pretende-se sugerir a dimensão da pesquisa já existente. Vale também notar que muitos dos modelos propostos são problemas de programação inteira mista(MILP).

Porém, as ferramentas existentes que implementam este tipo de modelagem costumam ser proprietárias, exigindo pagamentos recorrentes pelo uso. Tipicamente são ferramentas de grande escopo, que podem ser usadas para modelar e solucionar múltiplos problemas relacionados ao setor de transportes públicos [6, Planning and Travel Demand]. Esta flexibilidade e nível de polimento justificam os altos preços cobrados por esse tipo de serviço, e sua grande complexidade de uso pode exigir treinamentos e consultorias, às vezes não inclusas [6, Pricing].

Tomo como exemplo dois softwares que se encaixam no perfil descrito acima: TransCAD[6] e Optibus:[7]

1. O TransCAD [6] é um SIG(Sistema de Informação Geográfica, ou GIS em inglês) feito especificamente para o uso por profissionais do setor de transporte. Ele combina as funções de mapeamento e visualização geográfica com a modelagem e otimização de diversos problemas, oferencendo métodos distintos. O sistema requer instalação, e oferece suporte a chamadas a partir das linguagens da família .NET e Python, além de fornecer uma *scripting language* própria para implementar customizações sobre o sistema. O preço para uma licensa de utilização por um único usuário começa em $4000, chegando até $12000 iniciais, somados a $800-$1200 de manutenção anual para se ter acesso às atualizações.
2. O Optibus [7] também se propõe a fornecer soluções para planejamento de linhas de transportes urbanos, porém se diferencia por ser uma plataforma *software-as-a-service*, ou seja, a otimização sobre os modelos é executada nos servidores da empresa e não localmente. Também se diferencia por integrar técnicas de *machine learning* aos seus variados modelos disponíveis. Não são fornecidas informações sobre preço no site da plataforma, porém é informado que as taxas são mensais.

Ao mesmo tempo, existem diversos *solvers* capazes de solucionar eficientemente problemas MILP. Gurobi, CPLEX, CbC, dentre outros, são ferramentas disponíveis, livremente ou não, que podem ser aplicadas a modelos de otimização em transportes em geral e ao problema do planejamento de linhas em particular. Há também uma grande similaridade entre modelos deste tipo no tocante aos dados de entrada: tipicamente são usados uma rede de transportes e formas variadas de requerimentos, representados como atributos das linhas(requerimentos de frequência) ou como uma matriz VxV(*Origin-Destination Matrix*,

Matriz Origem-Destino).[2, p. 493-499]. Apesar disso, não conhecemos um framework unificado que sirva de intermediário entre os dados de entrada e os solvers, se responsabilizando pela entrada/saída e pela correta construção do problema junto à interface do solver.

Com isso, creio ser justificado o sistema desenvolvido: uma ferramenta que seria mais simples, de menor escopo e, sobretudo, de código e uso aberto. Apesar de contemplar, hoje, somente uma pequena parcela dos modelos listados, sugiro a possibilidade de futuras adições para facilitar o trabalho com diferentes dados de entrada e diferentes modelos.

1.3 Proposta e Objetivos do Trabalho

Objetiva-se com este trabalho produzir os seguintes artefatos:

1. Implementação em Julia/JuMP dos sistemas MILP que resolvem algumas variantes de um problema de planejamento de linhas, gerando uma saída ótima a partir dos dados de entrada. O objetivo é servir de intermediário entre os dados de entrada e as solvers, contemplando a leitura dos dados, sua organização em estruturas coerentes com a natureza do problema, e a modelagem correta junto à interface do JuMP. O JuMP então é responsável por disparar o uso dos solvers específicos, como o Gurobi, CPlex ou CbC.
2. Análise de resultados da implementação, levando em consideração as diferentes variantes, os valores obtidos e o tempo de processamento necessário para instâncias de diferentes tamanhos, inclusive instâncias de dimensão compatível a de grandes cidades. Utilizar para tal instâncias de redes reais como dado de entrada. Procurar avaliar a aplicabilidade e generalidade do método. Sugerir melhorias futuras.
3. Análise de resultados mais aprofundada para uma instância específica. Investigar a viabilidade e aplicabilidade da metodologia implementada.

Por meio destes, deseja-se disponibilizar livremente uma ferramenta de auxílio ao planejamento urbano e à análise objetiva dos modais de transporte, voltada à utilização por equipes profissionais e pesquisadores interessados no setor. Simultaneamente, deseja-se embasar teoricamente a metodologia utilizada e mostrar sua viabilidade prática. Com isso, estaria criada uma alternativa às opções já existentes.

Tal sistema deve ser desenvolvido utilizando o framework Julia/JuMP, supondo o formato GTFS [8] para os dados de entrada necessários e uma escolha de solver. Porém, sua arquitetura de código deve ser planejada de maneira a tornar simples uma adaptação para outros dados de entrada e a escolha por outros solvers. O projeto se restringe à mais simples modelagem deste tipo, a *Feasible Line Plan* [1, p. 26], ou *LP-Basic*[2, p. 498]; e às variantes deste problema, como definidas abaixo.

Deve-se notar também que, em relação à proposta de projeto entregue no primeiro semestre, houve mudanças nos objetivos relativos à análise dos resultados. Inicialmente me propunha a examinar principalmente os tempos de execução para diferentes instâncias, e a partir deles avaliar o sistema. Porém, ao longo do segundo semestre e seguindo as sugestões propostas pelo orientador do projeto, optei por focar na análise qualitativa sobre os próprios resultados finais, e não nos detalhes da sua obtenção.

2 Definição do Problema

2.1 Definições fundamentais e modelo base

Abaixo, descrevo o problema denominado *Feasible Line Plan* [1, p. 26], ou *LP-Basic*[2, p. 498]. A partir dele elaborei algumas simples variantes. Porém, antes de apresentá-las é necessário apresentar alguns conceitos relativos à representação das redes de transporte em si.

Definição 1. Define-se **Rede de Transporte Público** (RTP) como um grafo (direcionado) G=(V,E) em que os vértices v $\in$ V são denominados *paradas* e cada aresta e $\in$ E, e = (i,j) representa uma conexão direta de i para j, com i,j $\in$ V. [2, p. 493]

Definição 2. Define-se **Linha** como um caminho em em uma RTP.
A **frequência** $f \in \mathbb{N}$ de uma linha determina quantas vezes em um intervalo de tempo T a linha é atendida por um veículo(trem, onibus etc.)
Denomina-se **Plano de Linhas** $(\mathscr{L}, \bar{f})$ um conjunto de linhas $\mathscr{L}$ junto às suas frequências $f_l, \forall l \in \mathscr{L}$

Vale notar que todos os problemas descritos no presente documento se enquadram na etapa de planejamento de linhas (*line planning*) e, mais especificamente, otimização de linhas (*line optimization*), como descritas em [1, p.6-17] e reproduzida na figura 1. Com isso, assume-se algumas premissas importantes:

1. a infraestrutura da RTP já está definida de antemão, ou seja, não podemos adicionar, remover ou alterar paradas ou conexões.
2. considera-se somente um único modal de transporte. Dificuldades e custos extra advindos da divisão e integração entre diferentes modais devem ser considerados anteriormente no processo de *System Split* descrito em [1, p.14]
3. mesmo considerando um plano de linhas com frequências definidas para um intervalo de tempo T, não nos preocupamos em definir os horários específicos em que cada parada é atendida. Esse processo, o de encontrar uma tabela de horários(*timetable*) é posterior [1, p.6][2, p. 492] e foge ao escopo da otimização aqui definida.

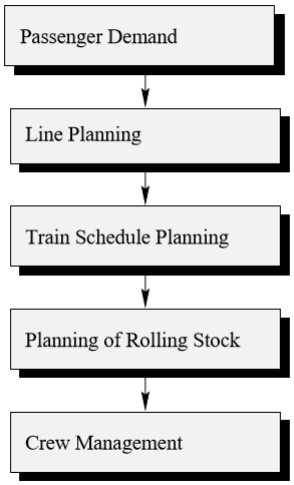


Figura 1: Processo de Planejamento Hierárquico [1, p.6]

Seguimos, definindo o objeto em que resulta nosso modelo de otimização:

Definição 3. Um Plano de Linhas é dito **viável** se, dados *requerimentos de frequência mínima* $f^{min} \in \mathbb{N}$ e *requerimentos de frequência máxima* $f^{max} \in \mathbb{N}$ para cada aresta $e \in E$, a soma da frequência das linhas que atendem a e é maior ou igual a f^{min} e menor ou igual a f^{max}:

$$\forall e \in E : f_e^{min} \leq \sum_{l \in \mathscr{L} : e \in l} f_l \leq f_e^{max} \tag{1}$$

[1, p. 24], [2, p. 499]

Com a definições acima, definimos o **Problema da Viabilidade do Plano de Linhas**(*Basic Line Planning Feasibility Problem*) como o problema de, dados como entrada uma RTP e um conjunto de linhas $\mathscr{L}_0$, determinar um plano de linhas $\mathscr{P} = (\mathscr{L}, \bar{f})$ tal que $\mathscr{P}$ seja viável.

E definimos o **Problema do Plano Viável de Menor Custo** como um Problema da Viabilidade do Plano de Linhas acrescido da meta de minimizar uma função objetivo dada pela soma dos *custos de operação* de cada linha usada no Plano.

$$\sum_{l \in \mathscr{L}} c_l f_l \qquad c_l \in \mathbb{R} \geq 0 \tag{2}$$

A este problema denominamos *LP-Basic* ou, simplesmente, *Basic*.

Repare que os *inputs* definidos para o problema em questão são os *outputs* do processo de determinação da demanda dos passageiros. Além disso, os *outputs* poderiam ser usados para determinar posteriormente um cronograma de veículos. Apesar de no presente projeto focarmos somente na etapa de planejamento de linhas, a utilidade prática é mais evidente quando consideramos o processo completo definido em [1, p.6]. Além disso, para de fato emular o uso do sistema desenvolvido, precisei estimar o que seriam os resultados da primeira etapa, como descrito mais abaixo na seção 3.

2.2 Formulação como MILP

A partir das definições acima, pode-se formular o LP-Basic como um problema de programação linear inteira. Dadas as entradas

uma RTP $G = (V, E)$

um vetor f^{min} de frequências mínimas por conexão $e \in E$

um vetor f^{max} de frequências máximas por conexão $e \in E$

um conjunto de linhas $\mathscr{L}_0$

um vetor c do custo por frequência para cada linha $l \in \mathscr{L}$

Deseja-se obter, como saída:

um vetor solução y, com as frequências ótimas para cada linha

Montamos o problema de otimização linear inteira:

$$\begin{aligned}
\min_{y} \quad & \sum_{l \in \mathscr{L}} c_l y_l & & & \\
\text{s.t.} \quad & \sum_{l \in \mathscr{L}: e \in l} y_l & \geq & f_e^{min} & \forall e \in E \\
& \sum_{l \in \mathscr{L}: e \in l} y_l & \leq & f_e^{max} & \forall e \in E \\
& y_l & \geq & 0 & \forall l \in \mathscr{L} \\
& y_l & \in & \mathbb{Z}^+ & \forall l \in \mathscr{L}
\end{aligned} \tag{3}$$

Para a resolução de um problema neste formato, usamos o framework JuMP integrado a *solvers* especializados, como já discutido. Uma vez solucionado, conclui-se trivialmente que $f_l = y_l, \forall l \in \mathscr{L}$

Antes de descrevermos as variantes do modelo-base, destaco que cada uma delas aplica uma alteração não relacionada diretamente com as demais. Assim sendo, é possível combinar variações, e isso de fato é feito pelo modelo desenvolvido, como será melhor explicado posteriormente.

2.3 Variante: Binary Basic

Esta variante é um formato mais restrito da LP-Basic. Nela, restringimos o modelo a aceitar completamente ou rejeitar completamente cada linha. Ou seja, para cada linha $l \in \mathscr{L}$, a frequência da linha, na solução, ou é 0, ou é uma frequência-base f^{base} pré-determinada.

$$f_l = 0 \quad ou \quad f_l = f_l^{base} \qquad \forall l \in \mathscr{L} \tag{4}$$

Ao se traduzir esta restrição para um modelo MILP, é mais adequado se usar variáveis de decisão binárias combinadas a um vetor entrada adicional f^{base}. Assim, obtemos:

$$\begin{array}{lrcll} \min\limits_{y} & \sum\limits_{l \in \mathscr{L}} c_l y_l f_l^{base} & & & \\ \text{s.t.} & \sum\limits_{l \in \mathscr{L}: e \in l} y_l f_l^{base} & \geq & f_e^{min} & \forall e \in E \\ & \sum\limits_{l \in \mathscr{L}: e \in l} y_l f_l^{base} & \geq & f_e^{max} & \forall e \in E \\ & y_l & \in & [0,1] & \forall l \in \mathscr{L} \end{array} \tag{5}$$

Solucionado este problema, a multiplicação abaixo basta para se obter a frequência desejada para cada linha.

$$f_l = y_l f_l^{base} \quad \forall l \in \mathscr{L} \tag{6}$$

2.4 Variante: Slack

Para a variante com folga, ou *slack*, cria-se um mecanismo que permite à otimização optar por não atender às restrições de frequência para uma conexão, sob o custo de pagar um pênalti por unidade de frequência violada. Essa variante foi bastante usada para experimentações sobre as instâncias, com diferentes valores para o pênalti. Intuitivamente, pênaltis baixos resultam em soluções que optam por simplesmente não usar diversas linhas, adicionando ao plano de linhas somente as com um bom custo-benefício. Já pênaltis altos resultam em soluções parecidas às do modelo básico, com algumas linhas de baixo custo-benefício possivelmente removidas. Enquanto essa variante pode sim ser usada para tentar identificar este suposto custo-benefício de cada linha, este tipo de interpretação é complexo, e só pode ser feito caso a caso, ainda assim com ressalvas.

Na prática, isto se traduz em mais uma entrada, um valor $p \in \mathbb{R}^+$, indicando o pênalti por unidade de frequência violada; e mais dois vetores de variáveis de decisão, $slack^{lo}$ e $slack^{up}$ com $slack_l^{up}, slack_l^{lo} \in \mathbb{Z}^+, \forall l \in \mathscr{L}$. Esses últimos são, respectivamente, a folga usada para violar os requisitos de frequência para menos ou para mais. Construímos assim o modelo:

$$
\begin{aligned}
\min_{y, slack^{lo}, slack^{up}} \quad & cost + penalty \\
\text{where} \quad & cost = \sum_{l \in \mathscr{L}} c_l y_l \\
& penalty = \sum_{e \in E} (slack^e_{lo} + slack^e_{up}) p \\
\text{s.t.} \quad & slack^{lo}_e + \sum_{l \in \mathscr{L}: e \in l} y_l \geq f^{min}_e \quad \forall e \in E \\
& (-slack^{up}_l) + \sum_{l \in \mathscr{L}: e \in l} y_l \leq f^{max}_e \quad \forall e \in E \\
& y_l \geq 0 \quad \forall l \in \mathscr{L} \\
& slack^{lo}_e \geq 0 \quad \forall e \in E \\
& slack^{up}_e \geq 0 \quad \forall e \in E \\
& y_l \in \mathbb{Z}^+ \quad \forall l \in \mathscr{L} \\
& slack^{lo}_e, slack^{hi}_e \in \mathbb{Z}^+ \quad \forall e \in E
\end{aligned} \tag{7}
$$

Nessa variante, temos uma função objetivo composta por duas componentes. A primeira exprime o custo pago pela uso das linhas, assim como no modelo básico. A segunda exprime uma penalização para restrições de frequência que não estejam atendidas pela seleção do plano de linhas.

É importante notar que, apesar de não haver nenhuma restrição que espefique $f^{lo}_e = 0$ ou $f^{up}_e = 0 \; \forall e \in E$, temos essa garantia para qualquer solução ótima. Isso acontece porque utilizar a folga superior e inferior simultaneamente somente acrescenta à função objetivo, sem ajudar a atender ambas restrições, só a uma no máximo.

2.5 Variante: Relaxado

Esta variante não difere em nada do modelo básico, excetuando que relaxamos a integralidade para a frequência de cada linha. Com isso, é possível obter soluções ótimas melhores do que as do modelo base, seja para comparação direta, seja para análises relativas exclusivamente a problemas de otimização linear, como análise do valor dual.

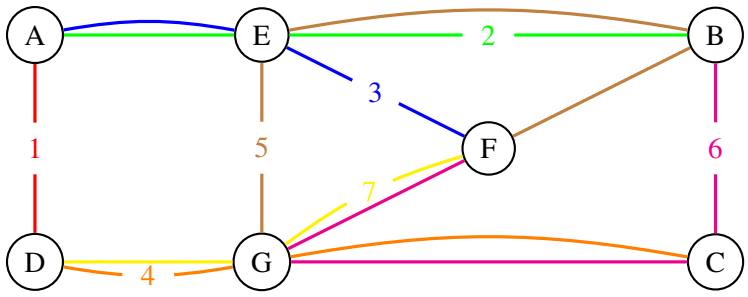
$$
\begin{aligned}
\min_{y} \quad & \sum_{l \in \mathscr{L}} c_l y_l \\
\text{s.t.} \quad & \sum_{l \in \mathscr{L}: e \in l} y_l \geq f^{min}_e \quad \forall e \in E \\
& \sum_{l \in \mathscr{L}: e \in l} y_l \leq f^{max}_e \quad \forall e \in E \\
& y_l \geq 0 \quad \forall l \in \mathscr{L} \\
& y_l \in \mathbb{R}^+ \quad \forall l \in \mathscr{L}
\end{aligned} \tag{8}
$$

2.6 Exemplo

Para facilitar o entendimento do problema, discuto abaixo uma instância exemplo, e construo as suas formulações MILP para cada variante. Para todos os exemplos abaixo, optei por exprimir em uma só linha restrições do tipo $b \leq a$, $b \geq c$, no formato $a \leq b \leq c$.

Informações adicionais:
$C_l \equiv 1, \;\; \forall l \in \mathscr{L}$: o custo de cada linha é igual a 1

Figura 2: Instância exemplo com 7 paradas e 7 linhas



$f_e^{min} \equiv 1, \quad \forall e \in E$: a frequência mínima para cada conexão é igual a 1
$f_e^{max} \equiv 2, \quad \forall e \in E$: a frequência máxima para cada conexão é igual a 2

Para esta instância, temos então problema de otimização linear-inteira:

Modelo básico

$$
\begin{array}{lrcl}
\min\limits_{y} & y_1+y_2+y_3+y_4+y_5+y_6+y_7 & & \\
\text{s.t.} & 1 \leq y_1 \leq 2 & & (A-D) \\
& 1 \leq y_2+y_3 \leq 2 & & (A-E) \\
& 1 \leq y_4+y_7 \leq 2 & & (D-G) \\
& 1 \leq y_5 \leq 2 & & (E-G) \\
& 1 \leq y_2+y_5 \leq 2 & & (E-B) \\
& 1 \leq y_4+y_6 \leq 2 & & (G-C) \\
& 1 \leq y_6 \leq 2 & & (B-C) \\
& 1 \leq y_3 \leq 2 & & (E-F) \\
& 1 \leq y_6+y_7 \leq 2 & & (G-F) \\
& 1 \leq y_5 \leq 2 & & (B-F) \\
& y_1,y_2,y_3,y_4,y_5,y_6,y_7 & \geq & 0 \\
& y_1,y_2,y_3,y_4,y_5,y_6,y_7 & \in & \mathbb{Z}^+
\end{array}
\tag{9}
$$

Cuja solução ótima é: $\{y_1 = 1,\ y_2 = 0,\ y_3 = 1,\ y_4 = 1,\ y_5 = 1,\ y_6 = 1,\ y_7 = 0\ \}$, com valor 5 para a função objetivo.

Para o **modelo binário**, se considerarmos o vetor de entrada f^{base} de maneira que $f_l^{base} \equiv 1, \forall l \in \mathscr{L}$, obtemos o mesmo problema com a mesma solução.

Para o **modelo com slack/folga**, considerando $penalty \equiv 10/unid.freq.$, obtemos:

$$
\begin{aligned}
\min_{y} \quad & custo + penalty \\
\text{where} \quad & custo = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 \\
& penalty = 10((s_1^{lo} + s_1^{up}) + (s_2^{lo} + s_2^{up}) + (s_3^{lo} + s_3^{up}) + (s_4^{lo} + s_4^{up}) + (s_5^{lo} + s_5^{up}) + \\
& (s_6^{lo} + s_6^{up}) + (s_7^{lo} + s_7^{up}) + (s_8^{lo} + s_8^{up}) + (s_9^{lo} + s_9^{up}) + (s_{10}^{lo} + s_{10}^{up})) \\
\text{s.t.} \quad & 1 \leq 1y_1 + s_1^{lo} - s_1^{up} \leq 2 \\
& 1 \leq y_2 + y_3 + s_2^{lo} - s_2^{up} \leq 2 \\
& 1 \leq y_4 + y_7 + s_3^{lo} - s_3^{up} \leq 2 \\
& 1 \leq y_5 + s_4^{lo} - s_4^{up} \leq 2 \\
& 1 \leq y_2 + y_5 + s_5^{lo} - s_5^{up} \leq 2 \\
& 1 \leq y_4 + y_6 + s_6^{lo} - s_6^{up} \leq 2 \\
& 1 \leq y_6 + s_7^{lo} - s_7^{up} \leq 2 \\
& 1 \leq y_3 + s_8^{lo} - s_8^{up} \leq 2 \\
& 1 \leq y_6 + y_7 + s_9^{lo} - s_9^{up} \leq 2 \\
& 1 \leq y_5 + s_1^{lo} 0 - s_1^{up} 0 \leq 2 \\
& y_1, y_2, y_3, y_4, y_5, y_6, y_7 \geq 0 \\
& s_1^{lo}, s_2^{lo}, s_3^{lo}, s_4^{lo}, s_5^{lo}, \\
& s_6^{lo}, s_7^{lo}, s_8^{lo}, s_9^{lo}, s_{10}^{lo} \geq 0 \\
& s_1^{up}, s_2^{up}, s_3^{up}, s_4^{up}, s_5^{up}, \\
& s_6^{up}, s_7^{up}, s_8^{up}, s_9^{up}, s_{10}^{up} \geq 0
\end{aligned}
\tag{10}
$$

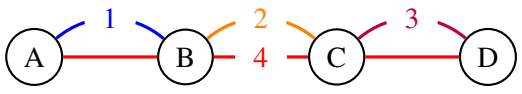
Onde optamos por abreviar *slack* para s. Nota-se que, mesmo para um exemplo com somente 7 paradas, 10 conexões e 7 linhas, já somamos 27 variáveis de decisão. Como escolhemos um valor de pênalti bastante elevado para o problema em questão, acabamos por obter um vetor solução y idêntico ao dos problemas anteriores, enquanto obtivemos $s_{lo}, s_{up} = \{0,0,0,0,0,0,0,0,0,0\}$.

Para a variante relaxada linearmente, visualmente obteríamos novamente um problema idêntico à variante básica, porém o vetor y poderia assumir valores fracionários. A solução ótima, ainda assim, se mantém idêntica.

2.7 Limitações

Mesmo antes de considerar instâncias reais, é possível identificar limitações inerentes ao modelo. Gostaria de destacar uma delas, a possibilidade de, ao minimizar custos, se optar por uma solução que aumenta o número de transferências entre linhas para um passageiro. Demonstro esta possibilidade através de uma instância exemplo.

Figura 3: Instância com solução ótima problemática



Informações adicionais:
As cores e números na imagem indicam cada linha, com quatro ao todo.

$C_4 \equiv 4$, : o custo da linha 4, vermelha, é igual a 4
$C_l \equiv 1$, $\forall l \in \mathscr{L} \setminus \{l_4\}$: o custo das demais linhas é 1
$f_e^{min} \equiv 1$, $\forall e \in E$: a frequência mínima para cada conexão é igual a 1
$f_e^{max} \equiv 2$, $\forall e \in L$: a frequência máxima para cada conexão é igual a 2

Não é difícil deduzir que, para este exemplo, a solução de menor custo seria $\{f_1 = 1, f_2 = 1, f_3 = 1, f_4 = 0\}$, com custo total igual a $1C_1 + 1C_2 + 1C_3 + 0C_4 = 3$. Nesta solução, um passageiro viajando de A para D precisará realizar duas transferências. Enquanto isso, a solução $\{f_1 = 0, f_2 = 0, f_3 = 0, f_4 = 1\}$ atende aos requisitos de frequência, com custo 4, e possibilitaria o deslocamento entre todas as paradas sem qualquer transferências.

Diversas soluções que remediam essa limitação são listadas em [2, p. 493-499], mas elas fogem ao escopo do presente projeto.

3 Leitura de Dados GTFS

Para poder de fato executar os modelos implementados sobre dados de cidades reais, fez-se necessário implementar, em Julia, um leitor de dados GTFS. Esta leitura instancia uma *struct* do tipo *LPBasic*, que poderá ser utilizada posteriormente para construção de modelos, visualização etc. Porém, esta construção não é trivial, já que os dados GTFS não estão estruturados de maneira a possibilitar uma equivalência direta com a nossa representação, baseada em frequências. De fato, existem estimações e imperfeições neste processo, e estas influenciam o resultado final da otimização. Assim sendo, destaco alguns destes fatores abaixo.

Primeiramente, é crucial notar que, ao passo que a definição dos requisitos de frequência não especifica os cronogramas de cada linha, no GTFS esta é a única representação disponível. Por isso, para se obter a frequência base de uma linha qualquer(f_l^{base}) para um dado intervalo de tempo, deve-se calcular a quantidade de viagens na linha para o mesmo. Como consideramos sempre o intervalo de tempo que cobre o *feed* por inteiro, temos que f_l^{base} é a simples contagem da ocorrência de viagens da linha l no *feed* GTFS em questão.

Além disso, os requisitos de frequência mínima e máxima de cada aresta devem ser estimados a partir da soma das frequências bases das linhas que as atendem. Para isso, definimos os estimadores $est_{inf}, est_{sup} \in \mathbb{R}^+$, de maneira que:

$$\forall e \in E : f_e^{min} = \lceil est_{min} \sum_{l \in \mathscr{L} : e \in l} f_l^{base} \rceil \tag{11}$$

$$\forall e \in E : f_e^{max} = \lfloor est_{max} \sum_{l \in \mathscr{L} : e \in l} f_l^{base} \rceil \tag{12}$$

Onde $\lceil x \rceil$ indica a operação *ceil*, ou arredondamento para o próximo valor inteiro, e $\lfloor x \rceil$ indica *round*, ou arredondamento para o inteiro mais próximo. O uso do *ceil* para o cálculo de f_e^{min} visa evitar situações em que $f_e^{min} = 0$, o que poderia causar a existência de paradas inacessíveis na solução final.

Os valores usados para os estimadores est_{inf} e est_{sup} influenciam enormemente nas soluções obtidas. Um valor baixo para est_{inf} tende a reduzir o custo total da solução, já que os requisitos mínimos se tornam menores. Valores de est_{inf} e est_{sup} muito próximos podem tornar o modelo inviável, enquanto selecionar valores tais que $est_{sup} \leq est_{inf}$ certamente o fazem. Para entender tais conclusões basta verificar o impacto que est_{inf} e est_{sup} têm nas restrições do modelo MILP.

Por último, destaco que há inúmeras situações, específicas a cada cidade, que podem gerar ruídos na leitura das linhas e conexões. Destaco algumas das mais comuns:

1. Caminhos que são diferentes entre si podem, para o provedor do GTFS, estar associados ao mesmo identificador de linha. Isso acontece porque, em diferentes horários do dia, é comum que o traçado de linhas de ônibus seja ligeiramente adaptado para as condições de trânsito daquela hora(com faixas reversíveis, por exemplo). Na nossa leitura não tratamos estes casos, e simplesmente tomamos cada caminho único como uma única linha. Isso pode fazer com que haja mais ”linhas”na instância lida do que no próprio GTFS.
2. Um importante caso particular da situação 1 é que linhas de ida e volta sejam tratadas como duas linhas independentes.
3. O custo assumido para cada linha é calculado simplesmente pela soma das distâncias ponto a ponto entre as paradas. A distância é calculada baseada nas latitudes e longitudes lidas do *feed*. Usar esta estimação simplista captura o fato de que conexões mais longas em distância custem mais caro, mas não representa fielmente fatores como tempo de deslocamento e custos com funcionários. Para além disso, mesmo a estimação de distância é imperfeita porque não leva em consideração o caminho real tomado pelo veículo, somente a linha ortodrômica que liga os dois pontos [9, GreatCircleDistance].

4 Projeto e Especificação do Sistema

O sistema desenvolvido foi realizado sob a forma de um módulo da linguagem Julia. Durante o processo, foram identificadas as diversas funcionalidades a serem implementadas, que foram subsequentemente divididas em arquivos fonte/submódulos diferentes,objetivando um maior desacoplamento das utilidades providas. Estas subcomponentes do todo são enumeradas mais abaixo. De forma geral, podemos dividir as funcionalidades implementadas em:

1. **Definições**: Definições das estruturas de dados que representam uma instância do problema. A mais importante delas é a struct ProblemInstance
2. **Leitura de Dados**: Funções para ler dados a partir de *feeds* em formatos padrão e instanciar as *structs* do tipo *ProblemInstance*
3. **Criação de modelos JuMP**: A partir de uma instância da struct ProblemInstance, gerar o modelo JuMP que resolve a otimização para uma variante específica.
4. **Visualização**: A partir das soluções obtidas e das structs ProblemInstance, gerar dados e mapas para facilitar a sua interpretação.
5. **Miscelânea**: funções acessórias e auxiliares.

As funções e structs para cada uma destas categoria são melhor detalhadas nas respectivas seções da documentação de código(em inglês)[9]. Respectivamente, nas seções *Data Structures*, *IO Functionality*, *Model Construction*, *Visualization* e *Miscelaneous*. Essas categorias também equivalem naturalmente às transformações/processos mostrados na figura 4.

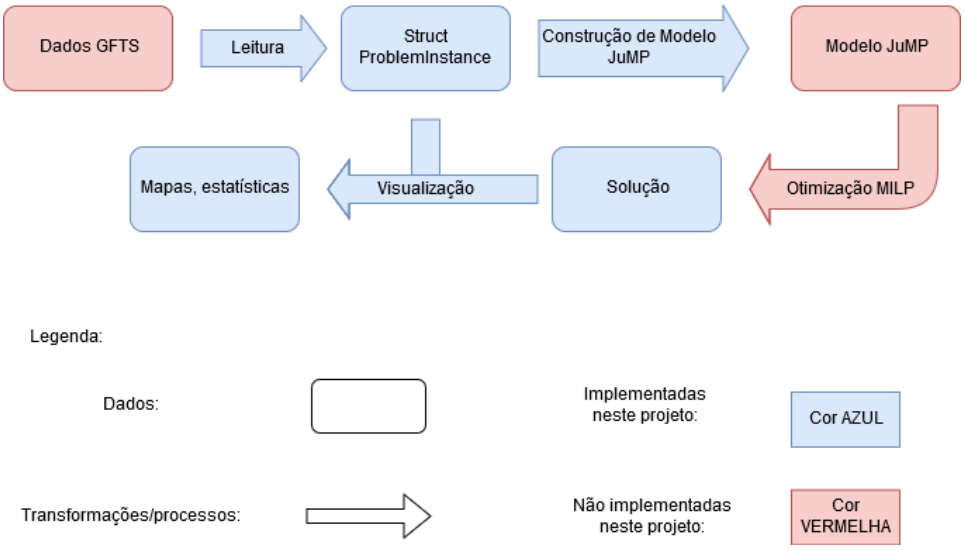


Figura 4: Pipeline de uso da solução.

Não houve um trabalho especificamente focado em interface de usuário, já que, pela natureza do projeto, o usuário final deverá necessariamente programar seus próprios *scripts* usando a linguagem Julia. Assim sendo, a documentação da API disponibilizada já é suficiente. Esta documentação seguiu os padrões especificados no manual da linguagem Julia [10, Documentation].

Para completar a seção de Especificação do Sistema, faz-se necessário pontuar algumas importantes dependências e o papel que elas desempenham no contexto deste projeto. A lista completa de dependências está disponível na documentação [9, Dependencies]

1. **JuMP**: Realiza o papel essencial de intermediar o presente módulo com os solvers MILP.
2. **DataFrames** e **CSV**: usados para leitura/escrita de arquivos, ler arquivos GTFS e exportar resultados.
3. **LightGraphs** e **MetaGraphs**: usados para representação da rede de transporte em forma de grafo direcionado
4. **Luxor**, **OpenStreetMapX**, **OpenStreetMapXPlot** e **Plots**: usados para gerar a visualização de instâncias e soluções em mapas.

5 Atividades Realizadas

Foram realizados, no primeiro semestre, atividades iniciais relativas à aquisição de material e conhecimento teórico e também a testes de tecnologia.

Em termos teóricos, eu já detinha algum conhecimento sobre o tema devido a uma disciplina cursada por mim quando em intercâmbio acadêmico na *Freie Universität Berlin*. Revisitei estes estudos prévios, em particular no que se tratava às diferentes modelagens para otimização do traçado de linhas, e os agreguei a outras variantes, principalmente as presentes em [2]. O estudo destas variantes e do diagrama em [2, p. 492] me permitiram orientar o projeto para o problema definido acima, sem grandes indefinições e, ao mesmo tempo, sabendo onde ele se encaixa em relação aos processos maiores. Com isso, foram essenciais para determinar o escopo do projeto.

Em termos de tecnologia, também no primeiro semestre foram realizados testes iniciais da *pipeline* de produção com as ferramentas propostas, através do desenvolvimento de uma implementação ingênua. Para tal, foi necessário instalar e usar corretamente componentes importantes do projeto: a linguagem Julia, o framework JuMP, e um solver MILP, o Cbc. Com isso, foi possível resolver um problema exemplo, com 7 estações e 10 linhas, e preparar a solução para problemas de maior porte, lidos de *feeds* de cidades reais.

O segundo semestre foi mais focado na implementação de fato do sistema proposto. Este trabalho foi pautado pelos aprendizados adquiridos ao longo do curso, como modularidade de código, reusabilidade etc., conjugados com as particularidades da linguagem Julia. Estas particulares me obrigaram a recontextualizar práticas de organização de código que já havia consolidado em outras linguagens, principalmente no que diz respeito ao conceito de *objeto*, que não está diretamente presente em Julia. Justamente por isso avalio que a linguagem não se adequa bem à maior parte dos modelos conceituais vistos no curso, que costumam se basear em classes de objetos e seus respectivos métodos. Em vez disso, optei por organizar o projeto baseado no fluxograma da figura 4, que é baseado em estruturas de dados e transformações sobre as mesmas.

Dada este fluxograma, a implementação na prática procurou priorizar, no primeiro momento, possibilitar um ciclo completo, trabalhando também nos pontos de conexão entre o presente sistema e as suas várias dependências. Uma vez com o ciclo consolidado para alguns casos de uso, se tornava mais fácil expandir os usos seguindo a mesma estrutura organizacional.

5.1 Plano de Ação

Dividi o planejamento do projeto final em quatro grandes áreas, que trabalhadas por mim em paralelo e de maneira complementar ao longo do ano. São elas:

1. **DOCUMENTOS** - Trabalho de confecção, revisão e entrega de documentos ao orientador e ao Departamento de Informática, possibilitando um correto acompanhamento do andamento do projeto e a adequação aos prazos estipulados pelo departamento.

2. **IMPLEMENTAÇÃO** - Trabalho de programação e desenvolvimento do código para o processamento e resolução dos problemas computacionais levantados pelo projeto.
3. **AVALIAÇÃO** - Trabalho de mensurar e avaliar o desempenho e os resultados obtidos pelas soluções implementadas, de modo a identificar pontos-críticos, detectar problemas, orientar o desenvolvimento e prover material para os documentos.
4. **PESQUISA** - Trabalho de pesquisa e embasamento teóricos que devem permear todas as demais áreas. Lida com a busca, leitura e incorporação de trabalhos acadêmicos pertinentes ao tema em questão.

É evidente que as quatro áreas, em realidade, devem caminhar juntas, se complementando. Porém, julguei útil a divisão para tangibilizar os cronogramas apresentados.

	ABRIL	MAIO	JUNHO	JULHO
DOCUMENTOS	Entrega da Proposta de Projeto Final	Revisão da Proposta de Projeto Final	Escrita do Relatório de Projeto Final I	
IMPLEMENTAÇÃO	Implementação ingênua variante 1	Implementação ingênua variante 2		
AVALIAÇÃO		Análise de desempenho variante 1	Análise de desempenho variante 2	
PESQUISA	Levantamento de material	Mapeamento de variantes	Especialização em variantes específicas	

Figura 5: Cronograma inicialmente proposto para o primeiro semestre

	JULHO	AGOSTO	SETEMBRO	OUTUBRO	NOVEMBRO	DEZEMBRO
DOCUMENTOS	Entrega e Revisão do Relatório I		Confecção do relatório de Projeto Final 2			
IMPLEMENTAÇÃO	Pipeline Entrada/Saída dos Dados	Desenvolvimento do sistema			Estabilização e refino	Preparação da Entrega Final
AVALIAÇÃO	Avaliação da implementação ingênua	Identificação contínua de pontos-críticos			Testes	
PESQUISA		Pesquisa sobre desempenho em MIP				

Figura 6: Cronograma inicialmente proposto para o segundo semestre

	JULHO	AGOSTO	SETEMBRO	OUTUBRO	NOVEMBRO	DEZEMBRO
DOCUMENTOS	Entrega e Revisão do Relatório I				Confecção do relatório de Projeto Final 2	Revisão/Apresentação
IMPLEMENTAÇÃO	Pipeline Entrada/Saída dos Dados	Desenvolvimento do sistema			Estabilização e refino. Revisão da Documentação	Preparação da Entrega Final
AVALIAÇÃO			Testes com diferentes instâncias GTFS		Agregação e comparação de resultados	
PESQUISA		Pesquisa sobre MILP e problemas duais				

Figura 7: Cronograma real para o segundo semestre

Para a primeira metade da disciplina, pretendia-se seguir o cronograma da figura 5, obtendo: uma base sólida em termos de pesquisa bibliográfica; uma versão inicial da implementação do sistema; uma avaliação desta implementação inicial, identificando os potenciais problemas e pontos-críticos; e, por último, um relatório que agregasse todo o conhecimento adquirido até então. Destes, foi possível realizar o estudo teórico, a versão inicial da implementação e o relatório. Note que que somente a avaliação de resultados não foi realizada. A maior dificuldade que impediu a realização plena desta foi o gerenciamento do meu tempo frente às demais obrigações acadêmicas, estágio e vida pessoal. Esta situação afetou o cronograma proposto para o segundo semestre(figura 6).

Neste, me propunha a realizar, na segunda metade do ano: a implementação das funcionalidades de entrada/saída da aplicação; a avaliação da implementação ingênua já existente; o desenvolvimento e avaliação de performance e pesquisa contínuas e integradas objetivando melhorar o desempenho da aplicação; testes sobre o produto final e correção de erros; e a confecção do relatório final, que sumarize a experiência e conhecimento adquiridos. Destes, todos foram realizados, com algumas ressalvas. Primeiramente, a análise de desempenho proposta inicialmente, que seria mais focada em tempo de execução, se tornou uma análise mais geral, focada primariamente em avaliar qualitativamente os resultados obtidos. Ela é apresentada na seção 6. Além disso, como se observa na figura 7, as tarefas foram realizadas de maneira mais sequencial, e não paralelamente como inicialmente proposto. Avalio que o maior ônus deste ocorrido tenha sido o fato de que a avaliação mais profunda dos resultados obtidos tenha sido feita tardiamente. Com isso, somente percebi certas limitações existentes e potenciais melhorias quando já não havia tempo hábil para implementá-las. Falo mais sobre isso na seção 7.

6 Análise de Resultados

6.1 Resultados Gerais

Exibo abaixo resultados agregados de instâncias de onze cidades distintas: As cidades foram escolhidas inicialmente de acordo com o seu tamanho. Desejava-se cidades pequenas, cujos problemas fossem rapidamente resolvidos. Estas são: Águeda, Córdoba, Estelí, Kochi, Kuopio e Nairobi. Posteriormente, optei por incluir cidades de maior porte e de especial interesse: Belo Horizonte, São Paulo e Rio de Janeiro são cidades brasileiras de grande importância. Berlim é uma cidade de relevância internacional. Além destas, há o município de Bagé, no Rio Grande do Sul, que for incluído aqui e também selecionado por mim para ser estudado mais profundamente, abaixo. Evidentemente, a escolha de cidades também foi pautada pela disponibilidade de dados GTFS.

Todos os resultados aqui expostos foram obtidos usando o modelo básico e estimadores $f_e^{min} = 1f_e^{base}$ e $f_e^{max} = 2f_e^{base}$, $\forall e \in E$,à exceção de Belo Horizonte, em que estes estimadores resultavam em um modelo inviável. Para este caso foi usado $f_e^{max} = 3f_e^{base}$. Foi utilizado o solver Clp em uma máquina Dell/Inspiron OAK14_HSW com processador Intel(R) Core(TM) i7-4500 CPU @1.80 GHz e 8.0 GB de RAM instalada.

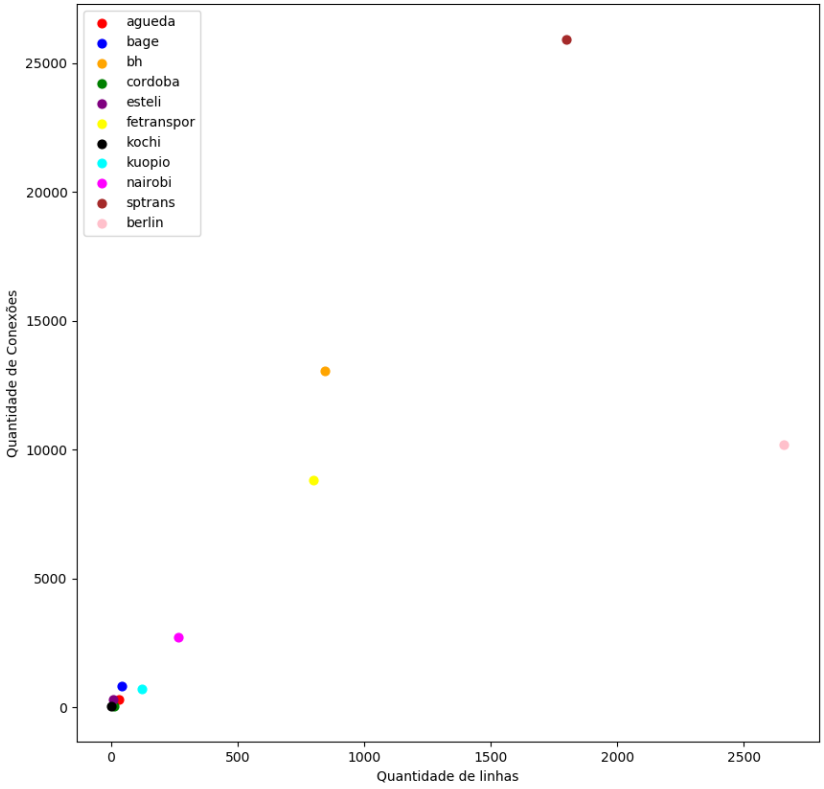


Figura 8: Linhas X Conexões

Primeiramente, é útil visualizarmos as diferentes dimensões dos problemas. Na figura 8 comparamos a quantidade de linhas e de conexões em diferentes instâncias, notando que, em geral, há uma relação aproximadamente linear entre uma e outra. Estes valores merecem um destaque especial porque eles impactam, respectivamente, a quantidade de variáveis de decisão e a quantidade de restrições no modelo MILP. Berlim se destaca da curva, porém é difícil avaliar o porquê sem um estudo mais específico. Poderia sugerir que a malha da cidade é mais densa, no sentido de que há mais linhas que cobrem a mesma conexão.

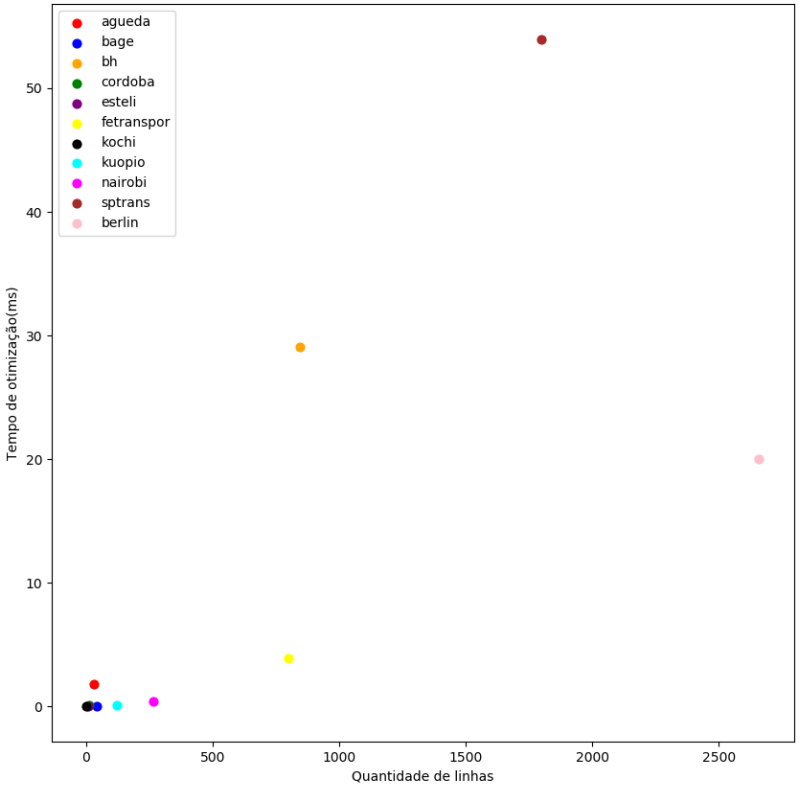


Figura 9: Linhas X Tempo(segundos)

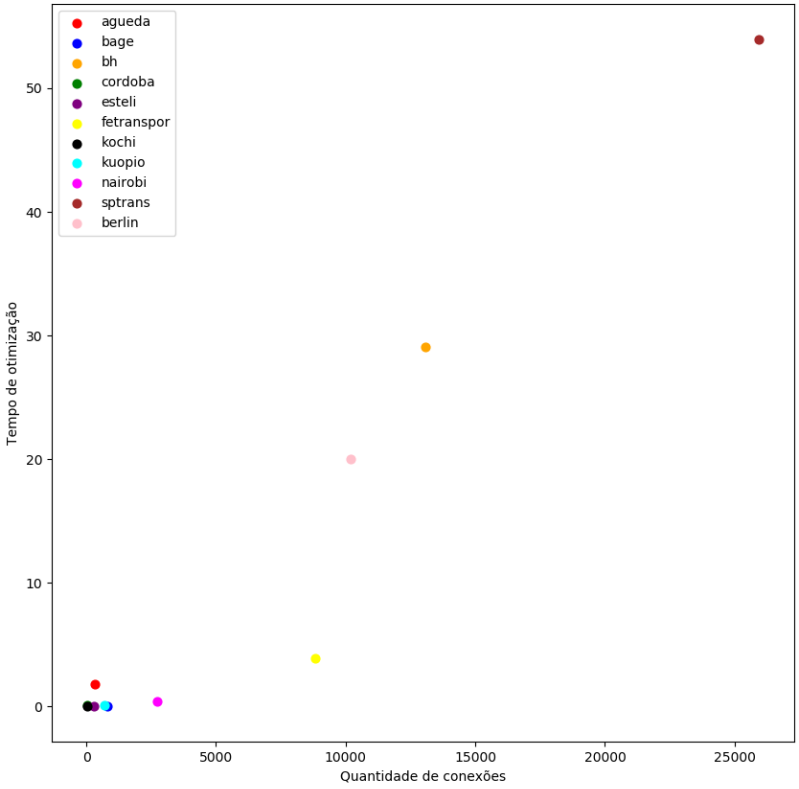


Figura 10: Conexões X Tempo(segundos)

Em seguida, observamos como este tamanho de instância influencia no tempo de execução dos modelos, nas figuras 9 e 10. De forma geral percebe-se uma maior correlação entre o número de conexões com o tempo de execução, e não do número de linhas. Ou seja, da quantidade de restrições, e não de variáveis de decisão. É evidente que esta suposta relação só poderia ser explicada pelas técnicas utilizadas pelo solver, levando atentando principalmente ao algoritmo Simplex. Isto, porém, foge ao escopo do presente documento.

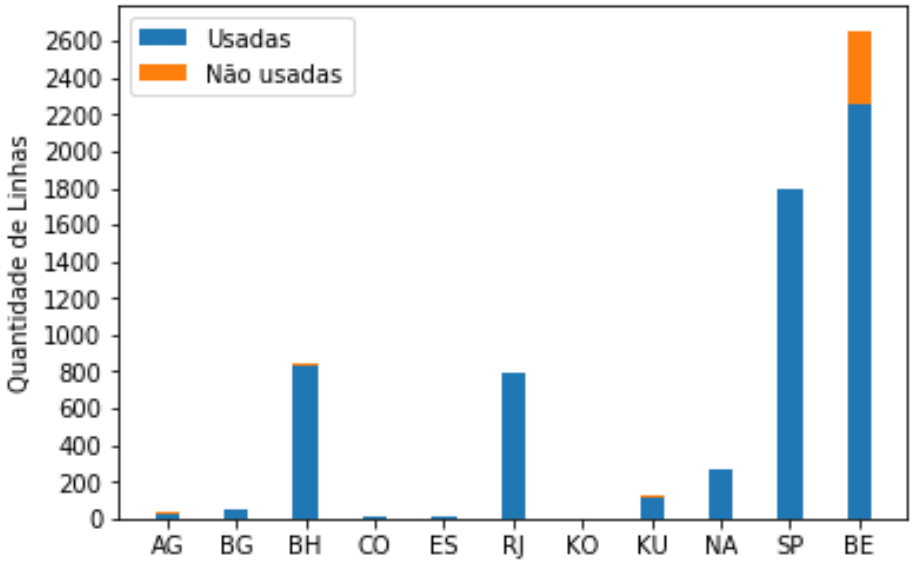


Figura 11: Linhas selecionadas X não selecionadas

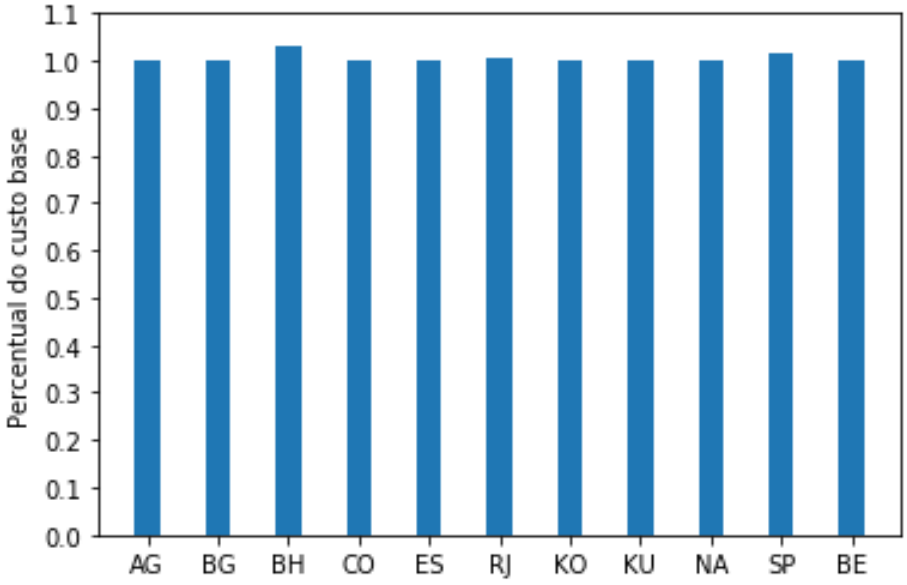


Figura 12: Percentual do custo ótimo em relação ao custo base

Na imagem 11, deseja-se mostrar que, de maneira geral, poucas linhas acabam por serem excluídas da solução ótima. Novamente, Berlim se destaca, possivelmente pelo mesmo motivo já elencado acima, de que exista um grande número de linhas que cobrem as mesmas conexões e que seriam assim potencialmente descartadas.

Já na imagem 12, vê-se que solução ótima obtida, em geral, mantém um preço próximo ao preço-base, calculado como o preço obtido pela solução trivial de manter cada linha operando com a sua frequência-base, exatamente como lida da instância GTFS. Vale notar que, mesmo que o custo da solução ótima seja similar à da solução original, é difícil avaliar se as soluções de fato são similares baseado somente neste gráfico comparativo. Para isso, teríamos que comparar diretamente as soluções, individualmente.

De forma geral, podemos concluir que, ainda que tenhamos implementado efetivamente a problema de otimização descrito na literatura, os resultados obtidos até agora não demonstraram uma óbvia aplicabilidade prática, e pareceram somente reproduzir, com ligeiras alterações, a instância GTFS lida.

6.2 Município de Bagé

Para demonstrar as possibilidades de análise que o sistema desenvolvido oferece foi selecionado o município de Bagé, no estado brasileiro do Rio Grande do Sul. Os motivos para escolha são o fato de ela ser uma cidade localizada no Brasil, para qual há dados GTFS disponíveis e, diferente de Belo Horizonte, Rio de Janeiro e São Paulo, é uma cidade pequena. Assim o processo de experimentar com diversos valores e com as diversas variantes implementadas torna-se dramaticamente mais rápido, o que era necessário dado o escopo deste projeto.

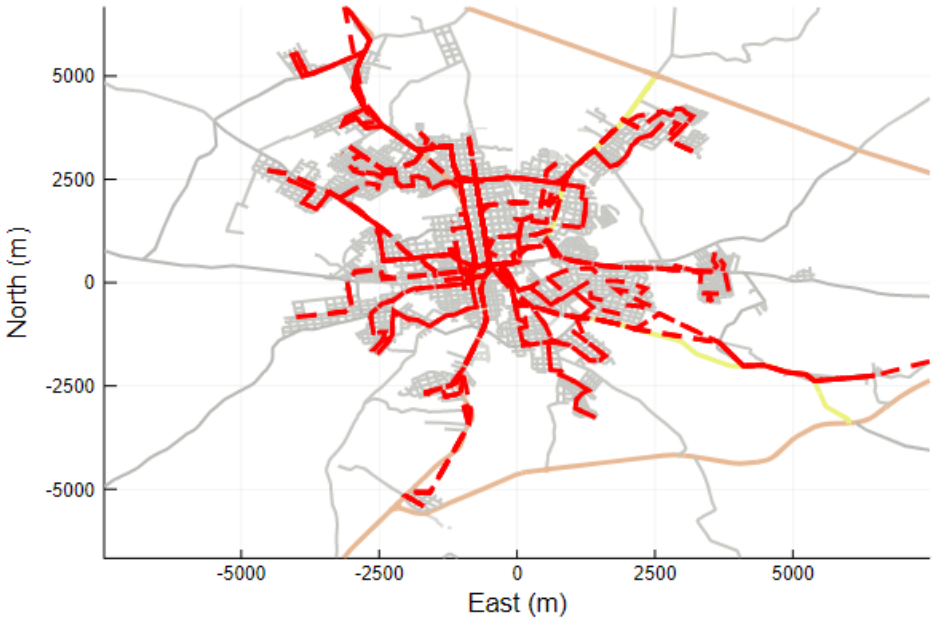


Figura 13: Visualização da rede de transportes do município de Bagé(RS)

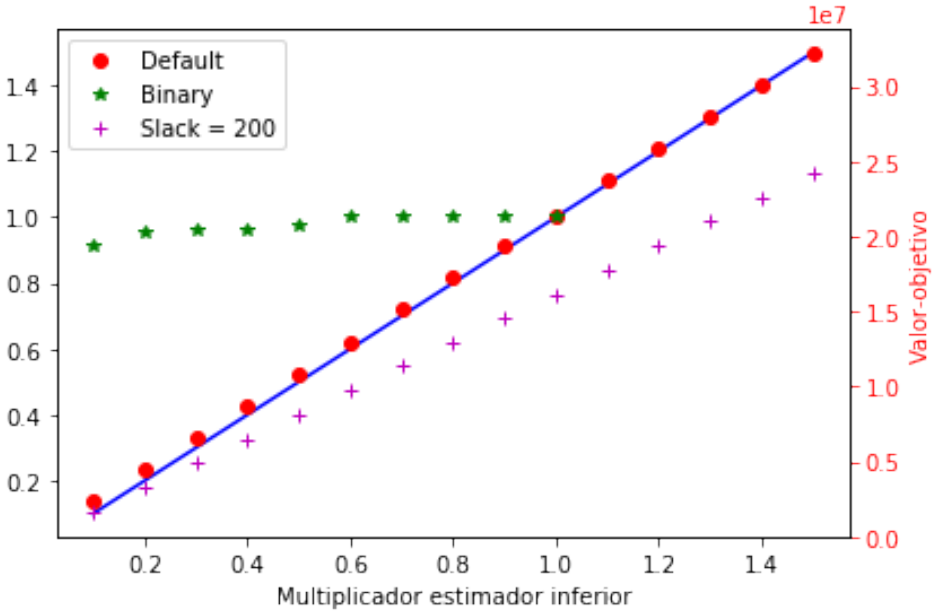


Figura 14: Comparação do custo total para diferentes valores de est_{inf} e diferentes variantes

Para começar, expomos, na figura 14 uma comparação entre o valor objetivo obtido para diferentes variantes e valores para est_{min}. A linha azul representa a progressão linear de est_{min} de 0.1 até 1.5. Os pontos vermelhos, que mostram os valores-objetivo para a variante básica, claramente seguem a mesma progressão linear. Isso acontece porque os requisitos mínimos foram estimados linearmente a partir das frequências base das linhas, então é simples ao modelo de otimização escolher frequências para cada linha de maneira a ”desfazer”a estimação.

Para a variante binária, vemos que a menor flexibilidade tem um grande impacto no custo obtido, que é sempre maior ou igual ao da variante básica. Para além disso, a variação deste valor é substancialmente menor, e somente poucas linhas foram não escolhidas, mesmo para valores de est_{min} baixos. Também é notável que, para o valor $est_{min} = 1.0$, a otimização trivialmente opta por incluir todas as linhas. Para valores $est_{min} > 1.0$, não há entradas no gráfico porque o modelo se torna inviável.

Para a variante com slack e $penalty = 200$, vemos que o valor-objetivo é, como esperado, sempre menor ou igual ao da variante básica. Além disso, a sua progressão também cresce linearmente, acompanhando est_{min}.

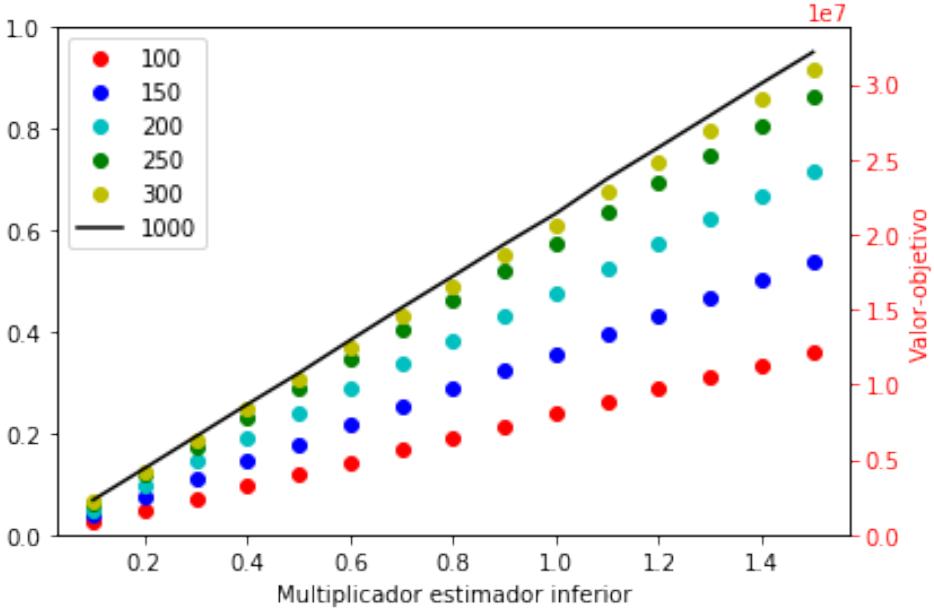


Figura 15: Diferentes linhas de crescimento de acordo com o valor de pênalti por unidade de frequência violada(p). Cada linha está associada a uma cor e um valor de pênalti diferente.

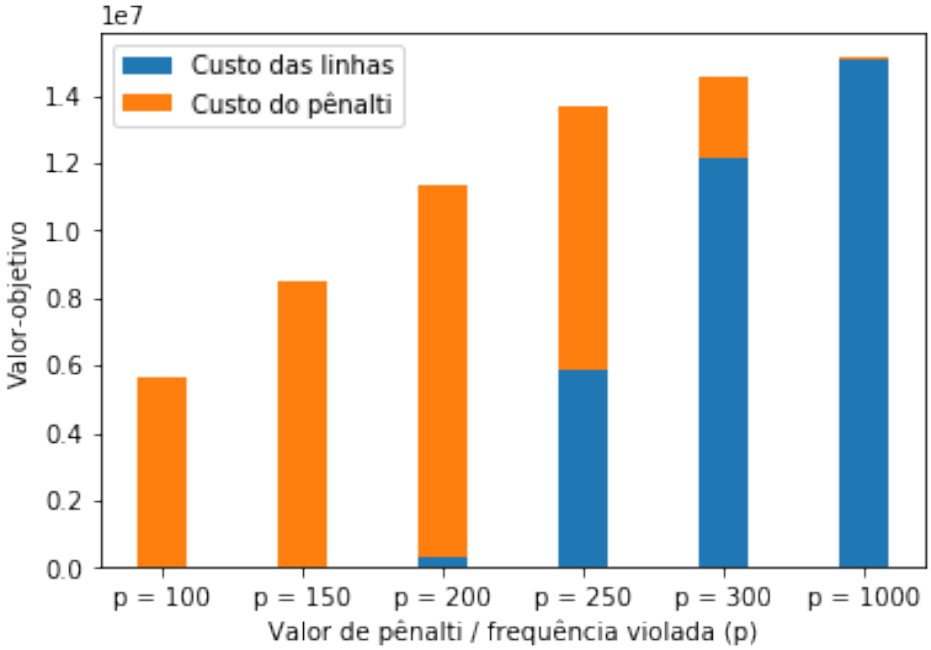


Figura 16: Diferentes razões entre custo total usado em linhas e custo total assumido em pênalti, para valores diferentes valores de p. O valor de est_{min} é fixo em 0.7

Os gráficos 15 e 16 detalham melhor a relação entre diferentes valores para p, o valor-objetivo, e as suas duas componentes *custo em linhas* e *pênalti*. No primeiro, nota-se como valores baixos para slack fazem com que o valor-objetivo cresça em uma taxa menor do que a observada na variante default, porém ainda de maneira marcadamente linear. A representação para o cenário em que $p = 1000$ foi propositalmente alterado para uma linha sólida para ressaltar o fato de que, conforme aumenta o valor de p, a variante com slack fica mais próxima à default. No segundo gráfico vê-se como, para valores baixos de p, a otimização opta por não incluir linha alguma, ou seja, custo em linhas = 0. Por outro lado, valores altos para p aproximam o resultado ao da variante default conforme o valor-objetivo é dominado pelo custo em linhas e a penalidade se aproxima de 0.

Gostaria ainda de destacar que, para os valores de $p = 200$ incluímos somente poucas linhas(figura 17), enquanto no valor de $p = 1000$ excluímos poucas linhas(figura 18). Identificar quais são essas linhas pode ser útil para sugerir quais as linhas que geram maior atendimento com o menor custo. Para isso, podemos facilmente inspecionar o vetor solução, buscando por linhas onde $y_l = 0$, para não usadas, ou $y_l \neq 0$ para usada. Esse foi justamente o processo usado para gerar as imagens.

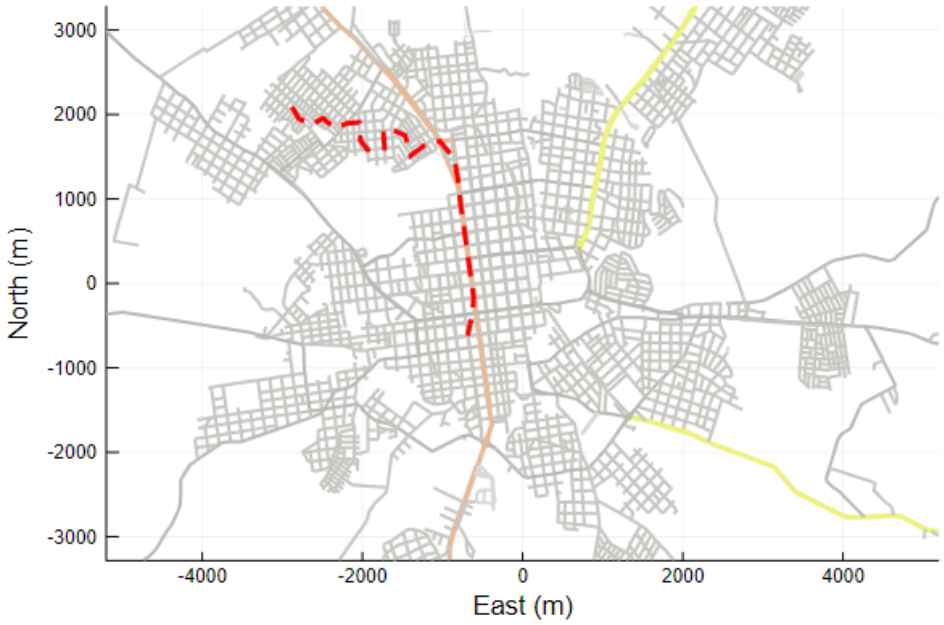


Figura 17: Linha usada para solução com $p = 200$ e $est_{min} = 0.7$

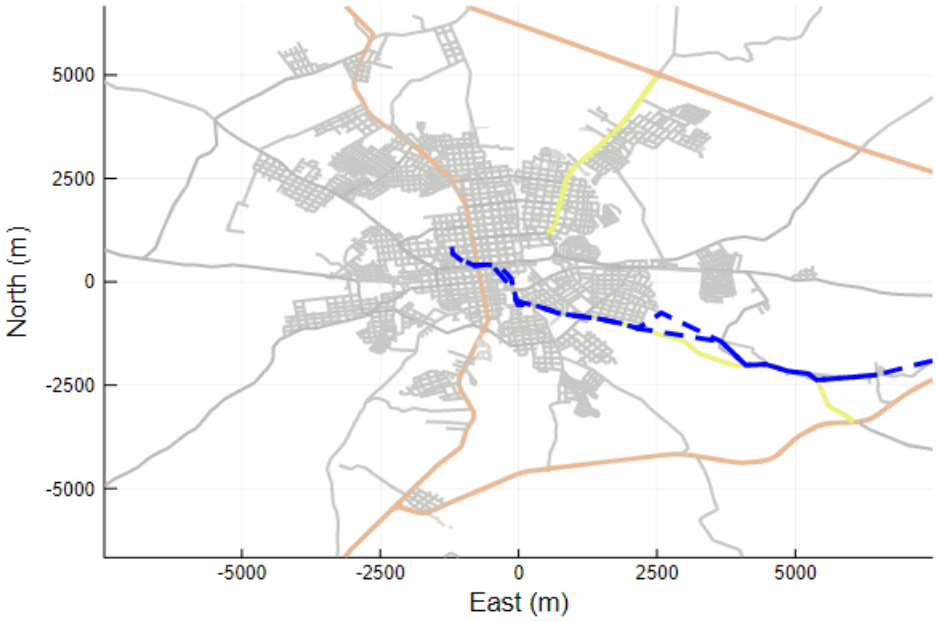


Figura 18: Linhas não usadas para solução com $p = 1000$ e $est_{min} = 0.7$

Por outro lado, pode ser útil identificar quais conexões são as mais críticas para a rede. Para este tipo de análise devemos inspecionar os valores das restrições para cada conexão,

onde é provável que identifiquemos uma ou mais arestas que estão no seu limite inferior de frequência, e que, assim, possivelmente estão justificando a adição de mais linhas ou pênalti, aumentando assim o custo total da solução obtida. Porém, ao na prática tentar identificar as conexões nesta situação, descobri que 733 das 815 restrições do tipo $f_e^{min} \leq \sum_{l \in \mathscr{L}: e \in l} y_l \leq f_e^{max}$ estavam precisamente no limite inferior. Assim sendo, esta informação acabou por ter pouco valor.

Havendo explorado as diversas informações obtidas através das ferramentas implementadas, pretendo haver demonstrado, com um exemplo mais específico, como poderiam ser realizados estudos iniciais para uma cidade a partir dos seus dados GTFS.

7 Considerações Finais

7.1 Contribuições

Não avalio que o presente projeto tenha realizado contribuições expressivas às comunidades acadêmicas, de desenvolvedores Julia ou de profissionais do setor de transportes. Para os primeiros, os resultados obtido já certamente foram reproduzidos em outras linguagens de programação, e o presente projeto não inovou em técnicas ou algoritmos. Para o segundo grupo, usar diretamente o framework JuMP provavelmente permitiria uma maior flexibilidade, melhor documentação e suporte mais amplo, enquanto as facilidades oferecidas pelo presente projeto poderiam ser reproduzidas relativamente rápido. Na perspectiva do terceiro grupo, provavelmente as ferramentas aqui oferecidas seriam insuficientes para ajudar a formular propostas de melhorias reais para as redes de transporte.

7.2 Trabalhos Futuros

Parte das deficiências listadas acima poderiam ser remediadas expandindo e melhorando funcionalidades já implementadas. Por exemplo:

1. Oferecer filtros customizáveis para as funções de leitura de dados, objetivando reduzir ou eliminar ruídos e imperfeições.
2. Prover facilidades para configurar livremente os valores utilizados, para incentivar a substituição de valores estimados(requisitos de frequência, custos das linhas etc.) por valores reais providos por agentes do setor.
3. Criar facilitadores para explorar e gerar resultados mais rapidamente para diferentes valores dos estimadores est_{min}, est_{max}, $penalty$ etc.
4. Desenvolver mecanismos para permitir que modelos parecidos(com diferentes estimadores, por exemplo) sejam resolvidos sequencialmente de maneira ótima(reotimização).
5. Desenvolver ferramentas para ler/gravar diretamente dados do tipo *struct ProblemInstance*
6. Melhorar e dar mais flexibilidade às funções de visualização. Possibilitar a visualização baseada nas conexões individuais e não só nas linhas individuais.
7. Expandir as funcionalidades de avaliação de resultados.
8. Otimizar forma de construir o modelo JuMP, que hoje demora mais do que o necessário devido a realocações internas.

Além destas, há inúmeras expansões ao projeto que poderiam de fato aumentar a sua utilidade prática e assim agregar valor:

1. Permitir a representação de uma *matriz origem-destino*(Origin Destination Matrix), comumente usada para problemas de planejamento de linhas[2, p. 493-499].
2. Implementar mais modelos além do LPBasic, em que, por exemplo, se considerasse a efetividade dos trajetos ponto-a-ponto pela rede, objetivando mitigar a limitação referente a transferências levantada na seção 2.7.
3. Implementar heurísticas para, em uma etapa de pré-processamento, gerar mais linhas a partir das linhas lidas inicialmente. Com uma maior gama de linhas novas e distintas para utilizar, a otimização teria maior possibilidade de diminuir o custo total ou de melhorar o atendimento dos requisitos de frequência.

Dos pontos levantados acima, diversos foram pensados por mim ou levantados pelo orientador durante o andamento do projeto, mas não foram concretizados por limitações de escopo e tempo hábil.

7.3 Pessoais

Creio que a maior contribuição de um projeto final desta natureza seja de fato na experiência e vivência adquirida pelo graduando, no caso, eu. Dito isso, avalio que o projeto tenha sido bem sucedido neste aspecto.

O projeto final no curso de Ciência da Computação na PUC propicia ao aluno um ano bastante distinto dos demais. Em nenhum outro momento do curso somos incumbidos de criar sozinhos uma ideia e levá-la a cabo. Geralmente preocupamo-nos como alunos somente com a implementação de sistemas cujos requisitos e objetivos já foram definidos pelo professor da disciplina em questão, e a própria duração de um ano gera desafios inexistentes nos trabalhos, que costumam durar de 1 a 2 meses.

Além disso, raras vezes é pedido de nós qualquer auto-avaliação, sendo o resultado de qualquer projeto anterior determinado unicamente pela nota que o mesmo recebe. Assim sendo, escrever este tipo de relatório e avaliar objetivamente os resultados do próprio trabalho é por si só uma vivência particular.

Dito isso, avalio que o presente projeto obteve sucesso em agregar à minha formação acadêmica e profissional. Não só me motivou a investir em começar e finalizar um projeto acadêmico, em que há a vivência de propor um estudo, buscar material, usufruir de orientação etc., como também se provou um desafio prático adequado ao término de um curso em Ciência da Computação.

8 Referências bibliográficas

Referências

[1] M. Bussieck, “Optimal lines in public rail transport,” 1998.

[2] A. Schöbel, "Line planning in public transportation: models and methods," *Springerlink.com*, 2011.

[3] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[4] A. Patz, "Die richtige auswahl von verkehrslinien bei großen straßenbahnnetzen," *Verkehrstechnik*, 1925.

[5] H. Wegel, *Fahrplangestaltung für taktbetriebene Nahverkehrnetze*. PhD thesis, Technische Universität Braunschweig, 1974.

[6] Caliper Corporation, "Transcad transportation planning software's website." `https://www.caliper.com/transcad/applicationmodules.htm`, 2019. Acesso em: 04/07/2019.

[7] Optibus, "Optibus' website." `https://www.optibus.com/`, 2019. Acesso em: 04/07/2019.

[8] Google, "GTFS reference." `https://developers.google.com/transit/gtfs/reference/`, 2019. Acesso em: 14/11/2019.

[9] A. M. Krauss, "Documentation for LPBasic module," *PUC-Rio*, 2019.

[10] V. B. S. Jeff Bezanson, Stefan Karpinski and other contributors, "Julia language manual." `https://docs.julialang.org/en/v1/`, 2019. Acesso em: 29/11/2019.

Lista de Figuras