UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANA PAULA DOS SANTOS DANTAS

MODELOS DE PROGRAMAÇÃO LINEAR INTEIRA PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÃO

QUIXADÁ

2017

ANA PAULA DOS SANTOS DANTAS

MODELOS DE PROGRAMAÇÃO LINEAR INTEIRA PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÃO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Ciência da Computação.

Orientador: Dr. Críston Pereira de Souza

QUIXADÁ

2017

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

D21m Dantas, Ana Paula dos Santos.
Modelos de programação linear inteira para o problema de programação de tripulação / Ana Paula dos Santos Dantas. – 2017.
63 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2017.
Orientação: Prof. Dr. Críston Pereira de Souza.

1. Otimização combinatória. 2. Programação inteira. 3. Programação (Matemática). I. Título.
CDD 004

ANA PAULA DOS SANTOS DANTAS

MODELOS DE PROGRAMAÇÃO LINEAR INTEIRA PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÃO

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em: ___ / ___ / _____

BANCA EXAMINADORA

Dr. Críston Pereira de Souza (Orientador)
Universidade Federal do Ceará – UFC

Me. Lucas Ismaily Bezerra Freitas
Universidade Federal do Ceará - UFC

Dr. Paulo de Tarso Guerra Oliveira
Universidade Federal do Ceará - UFC

Aos meus pais.

AGRADECIMENTOS

Agradeço à minha mãe, Ivanilda, e ao meu pai, Francisco, pelos sacrifícios que fizeram para que eu pudesse usufruir de uma boa educação, pelo apoio e carinho que sempre me foi dado durante toda a minha caminhada e por serem meu alicerce.

Agradeço ao Prof. Dr. Críston Pereira de Souza, por ter me convidado a participar deste projeto, pela paciência, e pela excelente orientação que permitiu a realização deste trabalho.

Agradeço aos meus irmãos, Maria Letícia e Paulo Filipe, que, apesar de todas as brigas, sempre estiveram ao meu lado quando precisei.

Agradeço a Alexsandro Oliveira Alexandrino, pelo companheirismo e carinho, por sempre escutar minhas reclamações e me oferecer um ombro pra descansar.

Agradeço aos professores Paulo de Tarso Guerra Oliveira e Lucas Ismaily Bezerra Freitas, pela disponibilidade em participar da banca desse trabalho e pelas excelentes colaborações e sugestões.

Agradeço aos professores Ricardo Reis, Críston Pereira, Paulo de Tarso, Arthur Araruna, Tânia Pinheiro, Ênyo Gonçalves, Davi Romero, Marcos Antônio pelas aulas, pelos conselhos e pelo trabalho que dedicam para melhorar o campus.

Agradeço aos meus professores da escola profissional, Renato Souza e Cássio Pinheiro, por terem me iniciado na área da TI. Em especial ao Cássio por estar sempre ao meu lado nos grandes passos da minha caminhada acadêmica.

Agradeço Joe Brockginton, Jeanne Hess e Alyce Brady pela amizade, apoio e hospitalidade durante o intercâmbio em Kalamazoo.

Agradeço às minhas amigas e meus amigos Alexandro Oliveira, André Davys, Anh Lam, Daiane Mendes, Deives Batista, Dieinison Jack, Jhonata Adam, João Vitor, Jonas Costa, Marisângela Cândido, Micaele Vieira, Raul de Araújo, Rômulo Costa, Sérgio Freitas, Tiago Nascimento e Isac Cavalcante pela amizade, pelos momentos que passamos juntos e por me darem forças para continuar nessa jornada.

“Computer science is not about machines, in the same way that astronomy is not about telescopes. There is an essential unity of mathematics and computer science.”

(Edsger W. Dijkstra)

RESUMO

Neste trabalho, propomos mudanças para o modelo apresentado em Nunes (2015), de modo a conseguir uma alocação de viagens quando consideramos todas as linhas da empresa de uma só vez. Foi feita uma análise do tempo de execução e da qualidade da solução do modelo de Nunes (2015) sem a inclusão de algumas restrições, comparando-a com o modelo quando é incluída uma solução viável inicial obtida por um modelo de Programação por Restrições desenvolvido em SOUZA (2016). Percebemos a necessidade de trabalhar individualmente as suas subdivisões do problema: Problema de Determinação de Tarefas (PDT) e Problema de Alocação de Escalas (PAE). Assim, o modelo de Nunes (2015) foi divido em dois, o modelo para o PDT é usado para criar tarefas, que são alocadas pela segunda parte, PAE. Para melhorar o PDT, foram desenvolvidas algumas heurísticas, com o intuito de diminuir o tamanho das entradas e para fornecer uma solução viável inicial ao modelo. Essas heurísticas conseguiram melhorar consideravelmente os resultados do modelo PDT. Por sua vez, o modelo PAE, que ficou mais simples devido a separação e às novas decisões de modelagem, apresentou bons resultados.

Palavras-chave: Programação (Matemática). Programação inteira. Otimização combinatória. Problema de determinação de tarefas. Problema de alocação de escalas

ABSTRACT

In this study, we propose adaptations to the model shown in Nunes (2015), so that this model creates workshift allocations considering all lines operated by the company. It was made an analysis on the execution time and on the quality of the solution given by the model without some of the restrictions, and comparing it to the same model with an initial feasible solution obtained by a Constraint Programming developed in SOUZA (2016). We recognize the necessity to work individually on each of the subproblems: Tasks Determination Problem (TDP) and Workshift Allocation Problem (WAP). Threfore, the model presented on Nunes (2015) was divided in two, the model for the TDP is used to create the tasks that will be allocated on the second part, WAP. To improve the TDP model, we developed two heuristics to reduce the size of the instances and provide the model with an initial feasible solution. These heuristics were able to improve the TDP model. The WAP model due to the separation and the modeling decisions, and thus, presented good results.

Keywords: Programming (Mathematical). Integer programming. Combinatorial optimization. Tasks determination problem. Workshift allocation problem

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A montagem de turnos de trabalhos para motoristas é uma atividade complexa. Tal atividade requer conhecimento sobre as leis trabalhistas, que regulam as condições de trabalho, e sobre o funcionamento das rotas que os motoristas devem atender. Montar turnos se torna ainda mais difícil quando há um número grande de tarefas a serem alocadas, como acontece em Fortaleza, onde existem atualmente mais de 200 linhas de ônibus urbano, cada linha sendo percorrida por vários ônibus simultaneamente. A operação dessas linhas é terceirizada e cada empresa deve contratar motoristas e fornecer os ônibus que serão usados nas linhas, além de serem responsáveis pela definição de onde e quando cada um deles deve trabalhar.

As empresas de ônibus, ao criarem um turno de trabalho, devem seguir tanto as restrições definidas pelas leis quanto as que foram acordadas com os sindicatos. Essa tarefa é delicada e requer bastante atenção quando feita manualmente para que não sejam alocados turnos que desrespeitem as leis, pois isso pode se tornar algo perigoso para os motoristas e passageiros, devido à sobrecarga de trabalho que pode ser ocasionada por uma alocação errada.

Por outro lado, quando bem elaborados, os turnos de trabalhos podem trazer grandes benefícios para a empresa, como a diminuição do número de horas extras pagas, e para seus funcionários, pois terão seus direitos garantidos e consequentemente uma melhor qualidade de vida. Por esses motivos, o problema é bastante estudado em busca de soluções otimizadas (NUNES, 2015), (BOAS; SILVA, 2014) e (SOUZA et al., 2004).

A criação dos turnos é tratada na computação como um problema de alocação, conhecido como Problema de Programação de Tripulação (PPT) ou *Crew Scheduling Problem*. Problemas de alocação são uma classe de problemas de tomada de decisão e geralmente são abordados usando modelos matemáticos de otimização. Tais modelos buscam alocar recursos escassos entre tarefas concorrentes, de modo a encontrar o melhor resultado para um objetivo mensurável. O tipo mais comum dos modelos matemáticos para problemas de alocação são aqueles que seguem as características da Programação Linear (PL), Programação Linear Inteira (PLI) ou Programação Linear Mista (TURBAN; SHARDA; DELEN, 2011).

Em Nunes (2015) são propostos dois modelos utilizando PLI considerando leis trabalhistas no contexto da cidade de Fortaleza, CE. Em Boas e Silva (2014) também são propostos dois modelos de PLI, com a diferença de que esses são adequados ao contexto da cidade de Belo Horizonte, MG. Outras abordagens para o problema são apresentadas em Souza et al. (2004), utilizando as metaheurísticas *Simulated Annealing*, Método de Pesquisa em

Vizinhança Variável e Busca Tabu.

Este trabalho tem como objetivo apresentar mudanças para os modelos apresentados em Nunes (2015), a fim de que os modelos representem de forma mais fiel a realidade das empresas que operam os ônibus de transporte urbano de Fortaleza, CE. De início, foram replicados os experimentos em Nunes (2015). Foram feitas análises do comportamento do modelo quanto ao tempo e qualidade da solução quando o modelo é alterado e quando são fornecidas soluções viáveis iniciais ao *solver* do modelo. Foram identificadas novas restrições a partir das leis e dos acordos mais atuais.

O modelo também foi modificado para alocar todas as linhas da empresa juntas, que é chamado de escopo de empresa. Dessa mudança, surgiu a necessidade de trabalhar o problema em partes, por causa do aumento da complexidade imposto pelo tamanho das entradas. Assim, o problema passou a ser tratado como dois subproblemas: o Problema de Determinação de Tarefas e o Problema de Alocação de Escalas (GOMES; GUALDA, 2011). O modelo de Nunes (2015) foi separado em dois, um para cada subproblema, criando assim dois novos modelos. Esses novos modelos foram implementados e foram executados experimentos para se medir o desempenho e qualidade da solução de ambos.

Os próximos capítulos estão organizados da seguinte maneira: no Capítulo 2 são apresentados os conceitos base; no Capítulo 3 são apresentados os trabalhos relacionados, de modo mais extensivo o trabalho de Nunes (2015); no Capítulo 4 são apresentados os modelos específicos para o Problema de Determinação de Tarefas e o Problema de Alocação de Escalas; no Capítulo 5 os experimentos realizados são descritos e discutidos; e no Capítulo 6 são apresentadas as considerações finais e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos utilizados neste trabalho. Na Seção 2.1 são abordados os conceitos envolvendo as características do Transporte Público Urbano na cidade de Fortaleza, na Seção 2.2 os conceitos de Pesquisa Operacional e na Seção 2.3 os de Programação Linear e suas variações.

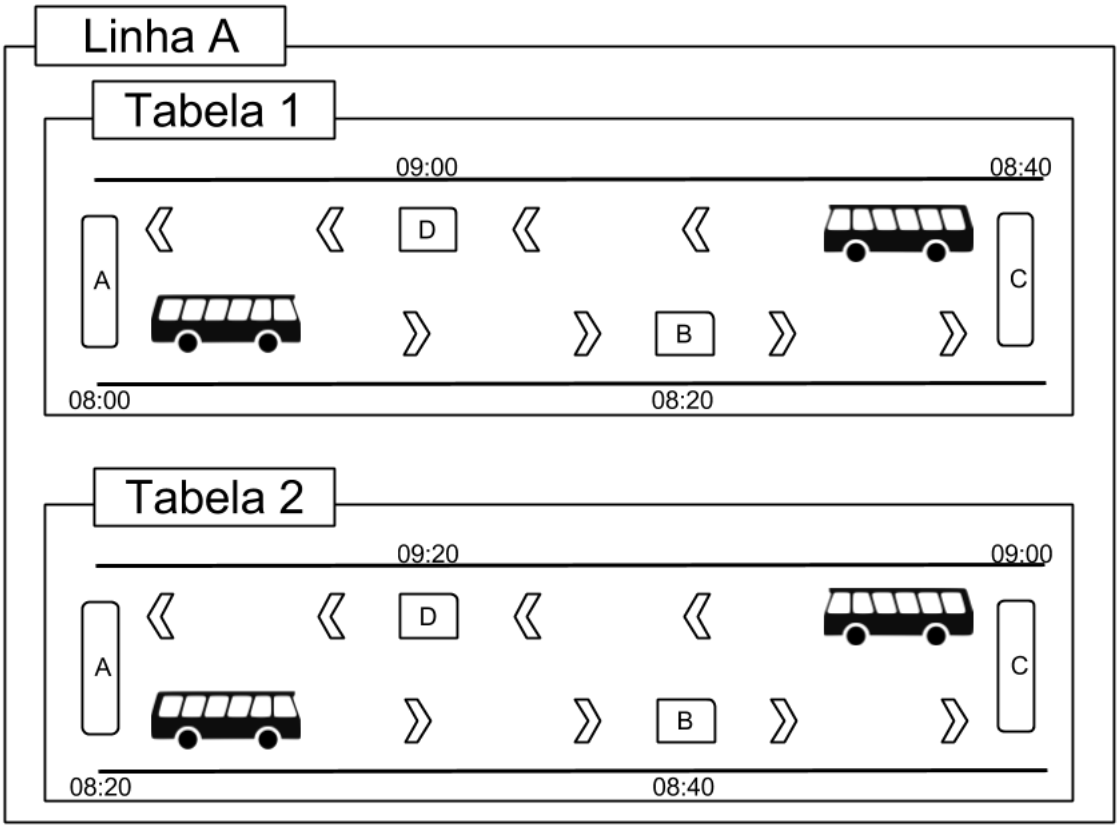
2.1 Transporte Público Urbano

Cada **linha** representa uma rota que os motoristas devem percorrer e são executadas diversas vezes por dia. Esta quantidade é influenciada pela demanda do local e pelo tipo de dia: dia útil, sábado ou domingo (um feriado é tratado como um dia de domingo). Uma linha possui várias **tabelas** de horários que indicam que um veículo está operando nesta linha, o que possibilita o compartilhamento de linhas entre empresas diferentes. Cada tabela é dividida em **viagens** de um ponto de parada a outro. Dentre os pontos de parada existem alguns pontos especiais: os **postos de controles** e os **terminais**, que possibilitam a troca de motoristas ou mesmo de veículos que trabalharão na tabela. Chamamos de **tarefa** um conjunto de viagens. Uma **pegada** é uma divisão do turno de trabalho. Um motorista pode trabalhar um turno sem intervalos, ou com um intervalo. Este último é chamado de **pegada-dupla**.

A Figura 1 mostra o exemplo de uma linha que tem o ponto A como origem e destino, passando pelos demais pontos B,C e D. Acima e abaixo de cada ponto de parada estão demarcados os horários em que o ônibus deve passar em cada ponto no sentido A/C e C/A respectivamente. Esta linha possui duas tabelas de horários: a Tabela 1 começando às 8:00h que passa pelo ponto A em direção ao ponto C e de volta ao ponto A, e a Tabela 2, começando às 8:20h, que faz o mesmo trajeto. Os pontos A e C representam postos de controle onde podem acontecer as trocas de motoristas. Um exemplo de tarefa pode ser a junção das viagens AB+BC que forma a tarefa AC, outro exemplo é a junção de CD+DA formando as tarefa CA.

O processo de programação de tripulação é a construção de turnos legais, incluindo horas extras, que juntos cubram todas as viagens de uma programação (WREN; ROUSSEAU, 1995). Essa alocação está sujeita às leis trabalhistas bem como aos acordos com as cooperativas e sindicatos. A rede de transporte público de Fortaleza é composta por 295 linhas regulares e 22 linhas complementares que transportam aproximadamente um milhão de passageiros por dia (FORTALEZA, 2016). Estas linhas são operadas por mais de 20 empresas, havendo a

Figura 1 – Exemplo de uma linha de ônibus



Fonte: Elaborada pela autora.

possibilidade de linhas compartilhadas, ou seja, operadas por mais de uma empresa.

É de responsabilidade de cada empresa contratar motoristas e fornecer os ônibus que compõem a sua frota, bem como alocar os seus recursos, humanos e materiais, para atender a todas as rotas que lhes foram designadas. Entretanto, não cabe à empresa alterar as rotas, a frequência do trajeto ou mesmo os horários de execução das viagens. Tais alterações são de responsabilidade do poder público. No caso de Fortaleza, a determinação das linhas é feita pela ETUFOR - Empresa de Transporte Urbano de Fortaleza S/A (FORTALEZA, 2016).

Devido ao grande número de viagens, e às diversas formas como elas podem ser organizadas em um grupo, temos um grande número de tarefas a serem executadas por um número limitado de motoristas, o que torna o problema mais difícil de se resolver. Os conceitos sobre linhas, viagens e tarefas são adaptados e utilizados no modelo como parâmetros de entrada e variáveis de decisão.

2.1.1 Restrições legislativas do problema

A Lei 13.103 de 2 de Março de 2015 (BRASIL, 2015) determina as seguintes restrições sobre as jornadas de trabalhos de motoristas profissionais, que atuam na categoria de transporte urbano rodoviário coletivo de passageiros:

- A jornada diária de trabalho é de 8 horas, podendo ser acrescida de 2 horas extraordinárias, ou 4 horas em casos que houverem acordo prévio com os motoristas;
- O motorista tem direito a pelo menos 1 hora para refeição, que pode coincidir com outros tempos de descanso;
- Os horários reservados para refeições podem ser fracionados em dois períodos de 30 minutos;
- Em um período de 24 horas, 11 delas devem ser destinadas ao descanso dos motoristas. Dessas 11 horas, 8 devem ser ininterruptas e o tempo restante pode ser fracionado;
- Um motorista não pode dirigir mais de 5 horas e 30 minutos ininterruptas;
- A cada 4 horas de trabalho, ininterruptas ou não, o motorista tem direito a 30 minutos de descanso.

Além das restrições impostas pela lei, também devem ser respeitados os acordos com os sindicatos. Tais acordos ainda prevalecem sobre alguns pontos da legislação citada acima, como por exemplo sobre os descansos e pausas para refeições podem ser fracionados em intervalos menores de 30 minutos. O acordo define o tamanho padrão da jornada como sendo 7 horas e 20 minutos. É também nesse acordo onde fica definido que o motorista tem direito à folgas aos domingos (CEARÁ, 2016).

Todos esses pontos devem ser respeitados quando se está criando um turno de trabalho para motoristas. Logo, eles serão transformados em restrições matemáticas para que sejam utilizadas nos modelos apresentados neste trabalho.

2.2 Pesquisa Operacional

Pesquisa Operacional (PO) é aplicada a problemas que envolvem a organização de atividades ou operações. É um estudo sobre as operações e como elas podem ser melhor realizadas. A PO surgiu durante a Segunda Guerra Mundial para suprir a necessidade de se

alocar de forma efetiva recursos escassos para a realização de operações militares (HILLIER; LIEBERMAN, 2013).

Desde então, a pesquisa operacional vem sendo usada em diversos problemas que necessitam de uma alocação eficiente de recursos, como no planejamento logístico de entregas, na melhoria do serviço de caixas de bancos e, mais comumente, na alocação de turnos de trabalhos.

Um bom exemplo de uso ideal da PO é o da empresa americana de aviação *United Airlines*. A empresa desenvolveu um programa que se utilizava Programação Linear Inteira e técnicas de otimização de redes para planejar os turnos dos atendentes das centrais e dos balcões de venda dos aeroportos. Durante os anos de 1983 a 1986 a empresa reportou uma economia de 6 bilhões de dólares por ano em decorrência do uso desse sistema (HOLLORAN; BYRN, 1986).

A pesquisa operacional utiliza modelos matemáticos para resolver problemas de tomadas de decisões. Hillier e Lieberman (2013) apontam a modelagem do problema como a terceira fase do processo de um estudo de PO, logo após a definição do problema e da coleta de dados. Para se modelar um problema, é necessário identificar quais as decisões devem ser tomadas e quais as regras, ou restrições, essas decisões devem obedecer.

Existem diversos procedimentos para encontrar soluções de modelos matemáticos, que são escolhidos de acordo com a natureza do modelo. O mais utilizado é o método Simplex, para a Programação Linear, e algoritmos enumerativos e de planos de corte, para a Programação Linear Inteira (PAPADIMITRIOU; STEIGLITZ, 2013).

2.3 Programação Linear

Programação Linear (PL) consiste no problema de minimização de uma função de custos linear sujeita a restrições, também lineares, de igualdade ou desigualdade (BERTSIMAS; TSITSIKLIS, 1997). O substantivo *programação* não se refere a softwares de computadores, e é usado aqui com o sentido de planejamento, ou seja, a programação linear envolve o planejamento de atividades de modo a se conseguir o melhor resultado possível da função objetivo (HILLIER; LIEBERMAN, 2013).

Um programa linear pode ser dividido em dois componentes básicos: a função objetivo e um conjunto de restrições. Na forma canônica de um programa linear todas as restrições são de desigualdade com $\leq$ e a função objetivo é sempre de maximização. Um exemplo de um problema de programação linear na forma canônica é mostrada nas equações em

(2.1), onde c, x são vetores em $\mathbb{R}^n$, b é um vetor em $\mathbb{R}^m$ e A uma matriz em $\mathbb{R}^{m\times n}$.

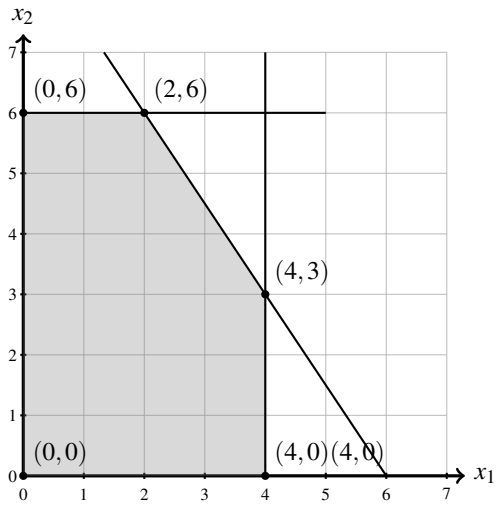
$$\begin{aligned} &\max\ c^T x \\ &\text{sujeito a} \\ &\qquad Ax \leq b \\ &\qquad x \geq 0 \end{aligned} \tag{2.1}$$

O algoritmo Simplex foi criado por G.B. Datzing e é amplamente utilizado para resolver programas lineares. Esse algoritmo otimiza a função objetivo melhorando o resultados ao mover a solução pelos vértices do problema (PAPADIMITRIOU; STEIGLITZ, 2013). Como a função objetivo de um PL é linear, se o conjunto de soluções ótima é não vazio, então ele contém pelo menos um vértice. Um modelo simples de programação linear pode ser resolvido por meio da avaliação da sua forma gráfica, checando-se os valores dos vértices. A seguir é mostrado um exemplo de um modelo com apenas duas variáveis.

$$\begin{aligned} \text{Max} \quad & 3x_1 + 5x_2 \\ \text{sujeito a} \quad & x_1 \leq 4 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2.2}$$

A Figura 2 mostra o gráfico gerado pelas restrições do problema anterior. A região sombreada representa a região permissível, ou viável, e os pontos demarcados são chamados vértices. Pelo algoritmo Simplex temos que cada um desses vértices é candidato a solução ótima (PAPADIMITRIOU; STEIGLITZ, 2013).

Figura 2 – Solução gráfica de um programa linear



Fonte: Adaptado de Hillier e Lieberman (2013).

Assim podemos encontrar o valor ótimo para o exemplo na Equação (2.2) apenas substituindo os valores dos vértices na função objetivo. Ao fazer isso encontramos que o valor ótimo para a função é 36, e os valores de x_1 e x_2 são 2 e 6, respectivamente.

Vale lembrar que o método Simplex não avalia todos os vértices, mas percorre vértices adjacentes, sempre melhorando a solução objetivo até que não seja mais possível melhorar.

2.3.1 Programação Linear Inteira

Um problema de programação linear inteira se caracteriza pelo fato de conter todas as variáveis com valores discretos, ou seja, inteiras. No caso em que apenas algumas das variáveis são inteiras, o problema é chamado de Programa Linear Inteira Mista. Outro caso particular é quando as variáveis são inteiras e restritas aos valores 0 ou 1, esse caso é conhecido como Programa Linear Inteiro Binário (WOLSEY, 1998).

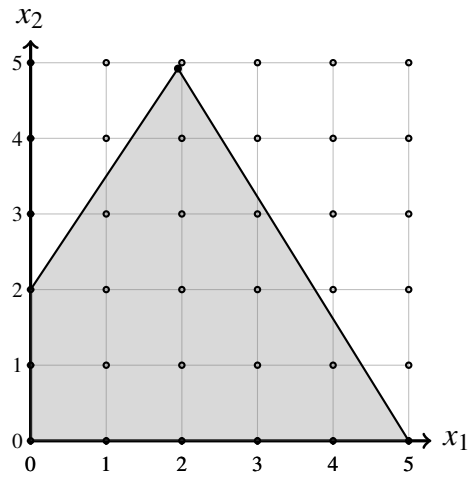
Apesar do fato de muitas características da PL serem compartilhadas com a PLI, não podemos simplesmente resolver o modelo considerando as variáveis como contínuas e depois arredondar para o inteiro mais próximo. O exemplo a seguir – Equação 2.3 e Figura 3 –, adaptado

de Wolsey (1998), mostra o porquê.

$$\begin{array}{lrcl} \text{Max} & 1.00x_1 + 0.64x_2 & & \\ \text{sujeito a} & 50x_1 + 31x_2 & \leq & 250 \\ & 3x_1 - 2x_2 & \geq & -4 \\ & x_1, x_2 & \geq & 0 \quad \text{e inteiros} \end{array} \tag{2.3}$$

A Figura 3 mostra o gráfico das restrições na Equação (2.3). Ao resolver o modelo como um modelo PL de variáveis contínuas temos que a solução ótima $x_1 = 376/193$ e $x_2 = 950/193$ não é inteira, e por isso não é válido como solução do PLI. Ao arredondar para os inteiros mais próximos encontramos os valores $x_1 = 2$ e $x_2 = 5$, entretanto essa também não pode ser a solução ótima, pois não se encontra dentro do espaço de soluções viáveis do problema, e de fato está relativamente longe da solução ótima inteira, que é $x_1 = 5$ e $x_2 = 0$.

Figura 3 – Solução Gráfica de um PLI



Fonte: Adaptado de Wolsey (1998).

Mesmo sendo bastante estudada, ainda não foi desenvolvido um algoritmo eficiente para resolver um PLI de tamanho razoável, ou seja, não existe um algoritmo que o resolva com um número de passos polinomial no tamanho da entrada. De fato, resolver um PLI é um problema da classe NP-Completo. Acredita-se que não exista solução eficiente para um problema dessa classe e qualquer solução correta do problema necessita de tempo exponencial no pior caso. Contudo, existem dois métodos aceitáveis, em termos computacionais, que resolvem problemas grandes. São eles: algoritmos de Planos de Corte e algoritmos Enumerativos (PAPADIMITRIOU;

STEIGLITZ, 2013).

Um importante conceito utilizado por ambas as abordagens é o de relaxação linear. Uma relaxação linear de um PLI é a remoção da restrição que diz que as variáveis devem ser inteiras, ou seja, as variáveis discretas são transformadas em variáveis contínuas. A relaxação não é utilizada para se conseguir a solução final, como na tentativa anterior, e sim para se obter um limitante para o modelo. A relaxação é uma importante ferramenta para a obtenção de limitantes para os problemas de programação linear inteira.

Os algoritmos de Plano de Corte seguem uma ideia básica, em que primeiro se resolve a relaxação linear do problema utilizando o método Simplex. Em seguida, é verificado se a solução é inteira e, em caso afirmativo, a solução da relaxação é também a solução do problema original; se não for inteira, uma nova restrição é adicionada, que deve ser satisfeita por todas as soluções viáveis inteiras, mas não pela solução da relaxação. Depois de adicionada a restrição, os passos anteriores – relaxação, checagem e adição da restrição – são executados novamente até que se encontre uma solução inteira (BERTSIMAS; TSITSIKLIS, 1997).

Já os algoritmos enumerativos consistem em enumerar as soluções viáveis de forma inteligente. Em outras palavras, o que um algoritmo enumerativo tenta fazer é provar de forma construtiva que uma solução é ótima, baseando-se em um número de partições do problema. Resolver a relaxação linear, como no algoritmo de Plano de Corte, é o primeiro passo de um algoritmo enumerativo. Também é feita a checagem da solução da relaxação, entretanto o próximo passo não é adicionar novas restrições e sim dividir o problema em dois subproblemas ao adicionar restrições que são mutuamente exclusivas e exaustivas. Em seguida é escolhido um dos subproblemas para ser relaxado e resolvido para passar pelo processo novamente (PAPADIMITRIOU; STEIGLITZ, 2013).

O *solver* ILOG CPLEX Optimization Studio da IBM, que será utilizado neste trabalho para resolver os modelos e modificações apresentados, faz uso de algoritmos de ambas as classes (CPLEX, 2009).

3 TRABALHOS RELACIONADOS

Em Boas e Silva (2014) são apresentados dois modelos de PLI para o PPT que são executados em sequência para obtenção de uma escala final. O primeiro destes modelos minimiza o número de motoristas usados em uma escala, o segundo recebe como entrada a solução do primeiro e tenta encontrar uma nova solução, de modo a minimizar o número de horas extras usadas pela empresa. O primeiro modelo já tem como entrada jornadas de trabalho dos tipos *pegada simples*, *dupla pegada* ou *noturna*. As jornadas do tipo *pegada simples* são de tamanho máximo permitido pela legislação e são contínuas, e as do *pegada dupla* também tem o tamanho máximo mas possuem um intervalo entre elas. Já as do tipo *noturna* indicam que são executadas no turno da noite. Essa característica torna o problema um pouco mais fácil de se resolver, pois diminui o número de tarefas a serem alocadas.

Uma característica que decorre diretamente do fato de serem usadas jornadas, e não atividades, como entrada é a adição de algumas restrições ao modelo referente às jornadas, como por exemplo os motoristas serem alocados para jornadas que estejam no mesmo turno de trabalho. Essa restrição é também uma influência das próprias características culturais da cidade de Belo Horizonte, MG, enquanto que este trabalho irá adequar-se às características da cidade de Fortaleza, CE.

Em sua dissertação, Nunes (2015) apresentou dois modelos de PLI, o primeiro tem como objetivo diminuir o número de motoristas utilizados em uma escala, e o segundo modelo, semelhante ao de Boas e Silva (2014), visa reduzir o número de horas extras utilizadas. Ambos os modelos de Nunes (2015) serão abordados com mais detalhe nas Seções 3.1 e 3.2.

Devido à sua característica combinatória, o PPT é tratado na literatura como dois subproblemas: Problema de Determinação de Tarefas (PDT) e Problema de Alocação de Escalas (PAE) (GOMES; GUALDA, 2011). O PDT consiste em criar tarefas a partir das viagens de cada linha e o PAE consiste em delegar cada uma dessas tarefas a um motorista. Os modelos apresentados em Nunes (2015) tratam os dois problemas em um só modelo.

Uma abordagem com busca Tabu para o PPT utilizando diversas restrições logísticas é encontrado em Marinho et al. (2004). Um exemplo dessas restrições é que um motorista não pode realizar duas tarefas consecutivas se o ponto final da primeira é diferente do ponto de início da segunda. Eles assumem também que os motoristas não podem trocar de linhas, o que é uma característica compartilhada com o trabalho de Nunes (2015).

3.1 Modelo de minimização do número de motoristas

Esta seção detalha o primeiro modelo de Nunes (2015), apresentando as variáveis, os conjuntos, as funções de indexação e por último as restrições. As restrições, apresentadas na Seção 3.1.3, fazem uso dos dados apresentados nas Seções 3.1.1 e 3.1.2.

3.1.1 Variáveis de instância, conjuntos e funções de indexação

O modelo necessita de diversas variáveis de instância que fornecem ao modelo informações sobre as tarefas, sobre os motoristas e também sobre parâmetros legislativos. A seguir essas variáveis são listadas e seus significados descritos:

- $QMAHPS$: quantidade máxima de horas semanais não aproveitadas por motoristas, ou seja, horas que são pagas ao motorista pela empresa, mas que o motorista não trabalha. Isso acontece pois a lei garante que todos os motoristas devem receber por turnos completos, mesmo que não sejam alocados para tarefas que formam turnos completos. Vale lembrar que no modelo todas as horas são convertidas para minutos;
- $QMDSF$: quantidade máxima de dias sem folga. O valor dessa variável é definido pela legislação;
- $QHED$ - quantidade máxima de horas extras diárias. O valor usado no modelo é o equivalente a duas horas;
- $HT_{s,d,t,f}$: horário de término da tarefa f da tabela t do dia d da semana s;
- $HI_{s,d,t,f}$: horário de início da tarefa f da tabela t do dia d da semana s;
- $QMDEJ$: quantidade mínima de descanso entre jornadas, cujo valor é definido pela legislação;
- $HTUD_m$: hora em que motorista m terminou a jornada de trabalho no dia anterior ao dia de início do planejamento atual. Caso o motorista não tenha trabalhado nesse dia o valor é o equivalente a zero horas;
- $TF_{s,d,t,f}$: tamanho, em minutos, da tarefa f da tabela t do dia d da semana s;
- $QMIHTT$: quantidade mínima de horas de um turno de trabalho;
- $QMAHTT$: quantidade máxima de horas de um turno de trabalho;
- $QHJS$: quantidade padrão de horas de uma jornada semanal. Esse valor que define quantas horas semanais devem ser pagas ao motorista, independente de

ele ser alocado para tarefas em todas elas ou não;

- $QMHES$: quantidade máxima de horas extras semanais da escala por motorista;
- $QMSSFD$: quantidade máxima de semanas sem folga de domingo. O valor desse parâmetro é definido por meio de acordos com os sindicatos de motoristas;
- $QDUF_m$: quantidade de dias desde a última folga do motorista m;
- $QSUFD_m$: quantidade de semanas desde a última folga de domingo do motorista m.

Os conjuntos que são utilizados no modelo são: $\mathcal{M} = \{0,1,...,M-1\}$, onde M é o número de motoristas disponíveis para a escala; $\mathcal{S} = \{0,1,...,S-1\}$, onde S é o número de semanas da escala; $\mathcal{D} = \{0,1,...,6\}$, que indica os dias da semana; $\mathcal{T}_d = \{0,1,...,T_d-1\}$, onde d é o índice de um dia, $d \in \mathcal{D}$ e T_d indica a quantidade de tabelas de um dia d; $\mathcal{F}_t = \{0,1,...,F_t-1\}$ onde t é o índice de uma tabela, $t \in \mathcal{T}_d$ e F_t indica a quantidade de tarefas da tabela t; $\mathcal{E} = \{0,1,...,E-1\}$ e indexa os dias da escala, onde E é o número total de dias da escala, ou seja, a quantidade de dias que serão planejados.

O modelo também conta com duas funções de indexação: $SC(e)$ e $DC(e)$. A função $SC(e)$ retorna $e/7$ (quociente da divisão de e por 7) e indica a semana da escala em que se encontra o e-ésimo dia da escala. Já a função $DC(e)$ retorna $e\%7$ (resto da divisão de e por 7) e indica em qual dia da semana se encontra o e-ésimo dia da escala.

3.1.2 Variáveis de decisão

As variáveis de decisão usadas no modelo são listadas a seguir:

- $x_{m,s,d,t,f} \in \{0,1\}$ - indica se o motorista m foi alocado para a tarefa f da tabela t do dia d da semana s.
- $r_{m,s,d,t,f} \in \{0,1\}$ - indica se o motorista m encerrou o turno na tarefa f da tabela t do dia d da semana s.
- $y_{m,s,d} \in \{0,1\}$ - indica se o motorista m trabalhou no dia d da semana s.
- $u_{m,s} \in \{-QMAHPS,...,QHED*QMDSF\}$ - indica a quantidade de minutos a mais ou a menos que o tamanho da jornada semanal mínima trabalhados pelo motorista m na semana s.
- $v_{m,s} \in \{0,...,QHED*QMDSF\}$ - indica a quantidade de minutos extras a serem pagos ao motorista m na semana s.
- $z_m \in \{0,1\}$ - indica se o motorista m foi alocado em alguma tarefa da escala que

está sendo programada.

3.1.3 Restrições do modelo

A função objetivo, apresentada na Equação (3.1), minimiza o número de motoristas utilizados no modelo. A variável z_m tem valor 1 quando o motorista m é usado na escala e 0 caso contrário.

$$\text{Min} \sum_{m=0}^{M-1} z_m \tag{3.1}$$

As Equações de (3.2) a (3.8) apresentam as restrições de modelagem. O conjunto de restrições na Equação (3.2) determina que todas as tarefas devem ser atendidas por exatamente um motorista. Como a variável $x_{m,s,d,t,f}$ é igual a 1 apenas quando o motorista m é alocado para a tarefa f da tabela t do dia d da semana s, se houver mais de um motorista alocado para a mesma tarefa essa solução se torna inviável.

$$\begin{gathered} \sum_{m=0}^{M-1} x_{m,s,d,t,f} = 1 \\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}_d,\ \forall f \in \mathcal{F}_t \end{gathered} \tag{3.2}$$

As restrições na Equação (3.3) mostram a dedução dos valores das variáveis $y_{m,s,d}$. Sempre que um motorista for alocado para uma tarefa, o valor de $x_{m,s,d,t,f}$ será igual a 1, o que faz com que o valor de $y_{m,s,d}$ também seja igual a 1, ou seja, esse motorista trabalhou no dia d da semana s.

$$\begin{gathered} y_{m,s,d} \geq x_{m,s,d,t,f} \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}_d,\ \forall f \in \mathcal{F}_t \end{gathered} \tag{3.3}$$

Para identificar se um motorista foi alocado para a escala, as restrições na Equação (3.4) dizem que sempre que o motorista for alocado em pelo menos um dia da escala, ele está sendo usado nessa escala. O valor de z_m será 1 apenas quando existir pelo menos um dia d de uma semana s em que esse motorista m trabalhou, ou seja, se existir um $y_{m,s,d}$ igual a 1 para um

dia d de uma semana s.

$$\begin{gathered} z_m \geq y_{m,s,d} \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D} \end{gathered} \tag{3.4}$$

As restrições na Equação (3.5) são usadas para encontrar os valores de $r_{m,s,d,t,f}$. Essas variáveis terão valor 1 sempre que $x_{m,s,d,t,f-1}$ for igual a 1, significando que o motorista está alocado para a tarefa $f-1$, e $x_{m,s,d,t,f}$ for igual a 0, ou seja, o mesmo motorista não foi alocado para as duas tarefas consecutivas. A partir daí se deduz que ele terminou o turno de trabalho.

$$\begin{gathered} r_{m,s,d,t,f} \geq x_{m,s,d,t,f-1} - x_{m,s,d,t,f} \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}_d, \\ \forall f \in \{1,2,...,F_t - 1\} \end{gathered} \tag{3.5}$$

As restrições na Equação (3.6) fazem um trabalho semelhante ao das restrições na Equação (3.5), pois identificam fins de turno que acontecem na última tarefa da tabela. Vale ressaltar que, diferente de $x_{m,s,d,t,f}$, o índice f de $r_{m,s,d,t,f}$ começa em 1 e vai até F_t, por isso a necessidade de separar em dois conjuntos de restrições.

$$\begin{gathered} r_{m,s,d,t,F_t} \geq x_{m,s,d,t,F_t-1} \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}_d \end{gathered} \tag{3.6}$$

As restrições na Equação (3.7) garantem que um motorista só tenha um fim de turno por dia, além de assegurar que um motorista não vai trabalhar em mais de uma tabela em um dado dia. Essa restrição justifica a adição da variável $r_{m,s,d,t,f}$ no modelo e ainda diminui consideravelmente o espaço de soluções viáveis, sendo adicionada no modelo para melhorar o tempo de execução do modelo.

$$\begin{gathered} \sum_{t=0}^{T_d-1} \sum_{f=1}^{F_t} r_{m,s,d,t,f} \leq 1 \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D} \end{gathered} \tag{3.7}$$

As restrições na Equação (3.8) garantem aos motoristas um descanso mínimo entre jornadas, impedindo que duas atividades que tenham um intervalo menor que *QMDEJ* entre o término da primeira e início da segunda sejam alocadas para o mesmo motorista.

$$\begin{gathered} x_{m,SC(e),DC(e),t_1,f_1} + x_{m,SC(e+1),DC(e+1),t_2,f_2} \leq 1 \\ \forall m \in \mathcal{M},\ \forall e \in \{0,1,...,E-2\},\ \forall t_1 \in \mathcal{T}_{DC(e)}, \\ \forall f_1 \in \mathcal{F}_{t_1} : HT_{SC(e),DC(e),t_1,f_1} > 24*60 - QMDEJ, \\ \forall t_2 \in \mathcal{T}_{DC(e+1)}, \\ \forall f_1 \in \mathcal{F}_{t_2} : HI_{SC(e+1),DC(e+1),t_2,f_2} + 24*60 - HT_{SC(e),DC(e),t_1,f_1} < QMDEJ \end{gathered} \tag{3.8}$$

Semelhante às restrições anteriores, as restrições contidas na Equação (3.9) garantem o descanso entre duas jornadas, com a diferença de que a primeira atividade é anterior à data do início da escala. Essa é uma das restrições que incorporam uma memória da escala anterior para a escala atual, o que faz com que o modelo realmente possa ser usado pelas empresas. Este tipo de restrição garante a continuidade das programações.

$$\begin{gathered} x_{m,0,0,t,f} = 0 \\ \forall m \in \mathcal{M} : HTUD_m > 24*60 - QMDEJ,\ \forall t \in \mathcal{T}_0, \\ f \in \mathcal{F}_t : HI_{0,0,t,f} + 24*60 - HTUD_m < QMDEJ \end{gathered} \tag{3.9}$$

O conjunto de restrições na Equação (3.10) delimita uma quantidade mínima de horas em que um motorista deve trabalhar por turno. Esse valor é particular de cada empresa. A utilização dessa restrição pode ajudar a diminuir as horas não aproveitadas, pois define um limite inferior para o tempo trabalhado por um motorista em um determinado dia. Em contrapartida, essas restrições podem tornar o modelo mais difícil de resolver.

$$\begin{gathered} \sum_{t=0}^{T_d-1} \sum_{f=0}^{F_t-1} (TF_{s,d,t,f} * x_{m,s,d,t,f}) \geq QMIHTT * y_{msd} \\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D} \end{gathered} \tag{3.10}$$

As restrições na Equação (3.11) garantem que um motorista trabalhará no máximo o número de horas permitidas em um turno, acrescido de horas extras. Os valores desses

parâmetros são definidos pelas legislações que regem a profissão dos motoristas.

$$\sum_{t=0}^{T_d-1}\sum_{f=0}^{F_t-1}(TF_{s,d,t,f} * x_{m,s,d,t,f}) \leq QMAHTT + QHED$$
$$\forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall d \in \mathcal{D} \tag{3.11}$$

As restrições na Equação (3.12) deduzem os valores das variáveis $u_{m,s}$, que representam a diferença entre horas trabalhadas e as horas mínimas que são pagas pela empresa. O valor de $u_{m,s}$ será zero quando o motorista não for alocado na escala, ou seja, $z_m = 0$. A soma dos tamanhos das tarefas é multiplicado por $x_{m,s,d,t,f}$ para que sejam contabilizadas apenas as tarefas que foram alocadas para o motorista.

$$\sum_{d=0}^{6}\sum_{t=0}^{T_d-1}\sum_{f=0}^{F_t-1}(TF_{s,d,t,f} * x_{m,s,d,t,f}) = QHJS * z_m + u_{m,s}$$
$$\forall m \in \mathcal{M},\ \forall s \in \mathcal{S} \tag{3.12}$$

A Equação (3.13) são usadas para deduzir os valores das variáveis $v_{m,s}$, que guardam quantas horas extras por semana devem ser pagas ao motorista m. O valor dessa variável será zero quando o motorista tiver horas não aproveitadas, ou tiver trabalhado exatamente o valor padrão de uma jornada.

$$v_{m,s} \geq u_{m,s}$$
$$\forall m \in \mathcal{M},\ \forall s \in \mathcal{S} \tag{3.13}$$

As restrições na Equação (3.14) garantem um limite para a quantidade de horas extras semanais de um motorista. Essa restrição é importante, pois pagar muitas horas extras é mais oneroso do que contratar um novo motorista. Esse valor também é definido pela empresa. Não foram utilizadas os valores das variáveis $u_{m,s}$, pois esses valores guardam também os valores das horas não aproveitadas.

$$\sum_{m=0}^{M-1} v_{m,s} \leq QMHES$$
$$\forall s \in \mathcal{S} \tag{3.14}$$

As restrições na Equação (3.15) garantem pelo menos um dia de folga para os motoristas a cada período de $QMDSF$ dias. Essa restrição é decorrente de leis trabalhistas.

$$\sum_{e_2=e}^{e+QMDSF} y_{m,SC(e_2),DC(e_2)} \leq QMDSF$$
$$\forall m \in \mathcal{M},\ \forall s \in \mathcal{S},\ \forall e \in \{0,1,...,E-QMDSF-1\} \tag{3.15}$$

As restrições na Equação (3.16) garantem que a escala que está sendo planejada irá continuar a respeitar as restrições impostas pelos acordos com os sindicatos. Elas levam em consideração os valores da escala anterior, guardando os dias decorridos desde a última folga do motorista m.

$$\sum_{e=0}^{QMDSF-QDUF_m} y_{m,SC(e),DC(e)} \leq QMDSF - QDUF_m$$
$$\forall m \in \mathcal{M} \tag{3.16}$$

As restrições na Equação (3.17) garantem uma folga de domingo a cada período de tempo, conforme definido através de acordos com os sindicatos.

$$\sum_{s_2=s}^{s+QMSSFD} y_{m,s_2,6} \leq QMSSFD$$
$$\forall m \in \mathcal{M},\ \forall s \in \{0,1,...,S-QMSSFD-1\} \tag{3.17}$$

O conjunto de restrições na Equação (3.18) garantem a memória da escala anterior para as folgas de domingo. Nessas restrições também são necessários os valores de entrada com informações específicas de cada motorista.

$$\sum_{s=0}^{QMSSFD-QSUFD_m} y_{m,s_2,6} \leq QMSSFD - QSUFD_m$$
$$\forall m \in \mathcal{M} \tag{3.18}$$

Por fim, o modelo de Nunes (2015) contém também uma constante M que fornece um limite superior para o número de motoristas que podem ser utilizados em uma escala. O valor da constante M é definido pela Equação (3.19). A função soma o tamanho de todas as tarefas da

primeira semana e divide esse valor pelo número de horas que um motorista pode trabalhar, sem a adição de horas extras. O fator g possibilita que seja adicionada uma folga na estimativa.

$$M = \left\lceil g * \frac{\sum_{d=0}^{6} \sum_{t=0}^{T_d-1} \sum_{f=0}^{F_t} (TF_{0,d,t,f})}{QHJS} \right\rceil \tag{3.19}$$

3.2 O segundo modelo de Nunes (2015) – Minimização de horas não aproveitadas

Após seus experimentos, Nunes (2015) percebeu que as soluções encontradas, apesar de conseguirem um número de motoristas reduzido, ainda continham uma grande quantidade de horas não aproveitadas. Horas não aproveitadas são aquelas que são pagas pela empresa mas o motorista não é alocado para trabalhar. Em outras palavras, essas horas representam a diferença entre o tamanho da jornada padrão de trabalho e a jornada alocada para o motorista pelo modelo.

Partindo da ideia de que essas horas não são desejáveis para a empresa, foi criado um novo modelo. Esse segundo modelo é um complemento do modelo anterior, sendo executado após a execução do primeiro e tendo como entrada a solução do primeiro. Foram adicionados ao modelo uma nova variável, uma constante e duas restrições, além de uma nova função objetivo.

A nova variável é $w_{m,s} \in \{-QMAHPS, 0\}$, que indica a quantidade de horas não aproveitadas do motorista m na semana s. A nova constante é K, que indica a quantidade de motoristas a serem utilizadas na alocação, cujo valor é definido pelo primeiro modelo.

A nova função objetivo, Equação (3.20), que maximiza a soma dos valores $w_{m,s}$. Vale ressaltar que como esses valores são negativos, a maximização da soma diminui minimiza o valor real das horas complementares.

$$\text{Max} \sum_{m=0}^{M-1} \sum_{s=0}^{S-1} w_{m,s} \tag{3.20}$$

A restrições na Equação 3.21 são usadas para deduzir os valores de $w_{m,s}$. Como o limite superior das variáveis de $w_{m,s}$ é 0, elas guardarão apenas os valores negativos de $u_{m,s}$ que indicam a quantidade de horas a menos que a jornada padrão alocadas para o motorista m na semana s. Quando o motorista tiver trabalhado um número de horas maior ou igual à jornada padrão, o valor de $u_{m,s}$ será não negativo e o de $v_{m,s}$ será 0, indicando que o motorista não tem

horas desperdiçadas nesta alocação.

$$w_{m,s} \leq u_{m,s},\ \forall m \in \mathcal{M},\ \forall s \in \mathcal{S} \tag{3.21}$$

A outra restrição adicionada ao segundo modelo é apresentada na Equação (3.22). Ela faz com que o número de motoristas usados seja o mesmo encontrado pelo primeiro modelo, de modo que a solução encontrada pelo segundo modelo possa apenas reduzir as horas não aproveitadas. Para alcançar esse objetivo, as horas extras e desperdiçadas serão rearranjadas entre o número de motoristas da escala, K, que é esperado que seja ótimo pela solução do primeiro modelo.

$$\sum_{m=0}^{M-1} z_m = K \tag{3.22}$$

Este trabalho foca apenas no primeiro modelo, para o qual alterações são propostas e implementadas. Essas alterações são discutidas mais detalhadamente nos capítulos seguintes.

4 MODELOS PROPOSTOS

Neste capítulo são apresentados os modelos que propomos, que são resultados da adição do escopo de empresa, a principal alteração no modelo.

Nessa parte do projeto foi escolhida uma empresa, a partir das disponíveis na aplicação *web* Gerenciador Integrado do Sistema de Transporte [1] (GIST). Dentre as empresas listadas pela aplicação, foi escolhida uma que operava o menor número de linhas. Também do GIST, foram coletados todas as viagens de todas as linhas operadas por essa empresa para os dias 01/01/2017, 02/01/2017 e 07/01/2017. Esses dias são representativos dos tipos de dia domingo, útil e sábado, respectivamente.

Notando-se a grande quantidade de viagens dessa empresa, entre 1100 e 1800 viagens por dia, foi decidido dividir, de fato, o problema em dois subproblemas. Assim o modelo apresentado na Seção 3.1 foi separado em duas partes, uma para o PDT e o PAE, que são apresentados nas Seções 4.1 e 4.2.

4.1 Modelo para o Problema de Determinação de Tarefas (PDT)

Para este modelo foram usadas todas as restrições do modelo em Nunes (2015) que se relacionam com a criação de um turno, como as restrições de tamanho do turno e de horas extras. Entretanto, essas restrições foram adaptadas para atender a algumas novas restrições que não ficaram explícitas no modelo 3.1.3.

4.1.1 Restrições do modelo PDT

O modelo foi adaptado para trabalhar com o conceito de pegadas, que são uma divisão na jornada, ou turno de trabalho. Nesse contexto, as pegadas são tratadas como tarefas, de modo que o tamanho ideal de uma tarefa gerada modelo PDT está entre 220 e 340 minutos. Esses valores foram escolhidos para restringir que o motorista tenha exatamente duas pegadas a serem alocadas pelo modelo PAE. A razão pela qual esses valores foram escolhidos está relacionada ao valor do tamanho padrão da jornada diária (440 minutos) e tamanho padrão acrescido do número máximo de horas extras (680 minutos).

O modelo recebe um conjunto de viagens como entrada e retorna um conjunto de tarefas. Para que qualquer par de viagens estejam na mesma tarefa elas não podem ter

[1] Disponível em http://gistapis.etufor.ce.gov.br

sobreposições de tempo, ou seja, uma delas deve terminar antes da outra começar. O intervalo entre essas viagens deve ser de até 15 minutos e a soma de todos esses intervalos dentro de uma tarefa não pode ultrapassar 60 minutos. Essas restrições são impostas, pois não é desejável que a tarefa tenha muito tempo de espera, já que a empresa deve pagar pelo tempo de espera maior que 120 minutos.

4.1.2 Conjuntos e índices

No modelo, a variável T guarda o número estimado de tarefas a serem criadas. Para estimar esse valor é somado o tamanho de todas as viagens e dividido pelo tamanho mínimo de cada tarefa, multiplicado ainda por um fator de folga g, como mostra a Equação (4.1). O conjunto $\mathcal{T}$, definido por $\{0,1,...,T-1\}$, é utilizado para indexar as tarefas do modelo. A variável V guarda o número de viagens da entrada e o conjunto $\mathcal{V} = \{0,1,...,V-1\}$ indexa essas viagens.

$$T = g * \left\lceil \frac{\sum_{v=0}^{V-1} L[v]}{220} \right\rceil \tag{4.1}$$

4.1.3 Informações de entrada

O modelo recebe como entrada informações básicas de cada viagem: $HI[v]$ e $HT[v]$, que indicam o horário de início e de término da viagem v, respectivamente; $PI[v]$ e $PT[v]$, que armazenam o posto inicial e terminal da viagem v, respectivamente; $L[v]$, que indica a quantidade de horas trabalhadas em cada viagem. Por fim, o modelo também recebe as informações de custo da hora de trabalho, C, e custo da hora extra C_e. Além de uma estimativa de limite inferior lb para a solução, calculado como mostra a Equação (4.2), ou seja, o valor de lb é o menor custo entre usar, ou não, horas extras nas tarefas.

$$lb = \min\left(\left(\sum_{v=0}^{V-1} L[v] - \left\lfloor \frac{\sum_{v=0}^{V-1} L[v]}{220} \right\rfloor * 220\right) * C_e, \left(\left\lceil \frac{\sum_{v=0}^{V-1} L[v]}{220} \right\rceil * 220 - \sum_{v=0}^{V-1} L[v]\right) * C\right) \tag{4.2}$$

4.1.4 Variáveis de decisão

As variáveis de decisão do modelo PDT são listada a seguir:

- $z_t \in \{0,1\}$: indica se a tarefa t tem viagens alocadas a ela, ou seja, se a tarefa t foi utilizada na solução;

- $x_{t,v} \in \{0,1\}$: indica se a viagem v foi alocada à tarefa t;
- $b_t \in \mathbb{Z}$: guarda a diferença de tempo entre o tamanho da tarefa t e o tamanho ideal de 220 min. Essa variável é utilizada para deduzir os valores de horas extras e horas não aproveitadas.
- $e_t \in \mathbb{Z}^+$: indica a quantidade de horas extras alocadas a tarefa t;
- $d_t \in \mathbb{Z}^+$: indica a quantidade de horas não aproveitadas da tarefa t;
- $i_t \in \mathbb{R}$: indica o horário inicial da tarefa t;
- $f_t \in \mathbb{R}$: indica o horário final da tarefa t.

As variáveis b_t, e_t e d_t são semelhantes às variáveis $u_{m,s}$, $v_{m,s}$ e $w_{m,s}$ do modelo de Nunes (2015).

4.1.5 Modelagem do PDT

A Equação (4.3) mostra a função objetivo do modelo PDT. Diferente dos modelos de Nunes (2015), nessa modelagem o objetivo é encontrar um equilíbrio entre as horas desperdiçadas e as horas extras pagas a cada motorista. Vale ressaltar que, como essas tarefas ainda serão alocadas pelo modelo PAE, o custo ainda pode ser reduzido ao combinar uma tarefa com horas extras para uma das pegadas com uma tarefa com horas desperdiçadas. Ou seja, os custos aqui são apenas uma estimativa.

$$\text{Min} \sum_{t=0}^{T-1} (d_t * C + e_t * C_e) \tag{4.3}$$

As restrições apresentadas na Equação (4.4) estabelecem o limite inferior para o valor da função objetivo, utilizando o valor da variável lb, que é dado como entrada ao modelo. Essa restrição foi adicionada ao modelo para tentar melhorar os resultados e os valores GAP da solução. O GAP é a diferença percentual entre a melhor solução viável e o melhor limite inferior obtido através da relaxação linear.

$$\sum_{t=0}^{T-1} (d_t * C + e_t * C_e) \geq lb \tag{4.4}$$

As restrições mostradas nas Equações em (4.5) restringem que cada viagem só pode estar alocada a uma única tarefa. Essa restrição também define que nenhuma viagem pode ficar

sem tarefa.

$$\sum_{t=0}^{T-1} x_{t,v} = 1 \quad \forall v \in \mathcal{V} \tag{4.5}$$

As restrições nas Equações (4.6) são usadas para deduzir os valores das variáveis z_t, que são similares as do modelo de Nunes (2015).

$$z_t \geq x_{t,v} \quad \forall t \in \mathcal{T},\ \forall v \in \mathcal{V} \tag{4.6}$$

As restrições em (4.7) definem o tamanho máximo de uma tarefa, 220 minutos somados 120 minutos de horas extras. Uma restrição de tamanho mínimo das tarefas não foi adicionada, porque devido aos tamanhos variados das viagens e das restrições de logística apresentadas a seguir. Tal restrição de tamanho mínimo das tarefas tornaria o modelo inviável para muitas instâncias.

$$\sum_{v=0}^{V-1} L[v] * x_{t,v} \leq 340 \quad \forall t \in \mathcal{T} \tag{4.7}$$

As restrições nas Equações (4.8) definem o horário inicial de cada tarefa. Para fazer isso, essas restrições definem que o valor de i_t deve ser menor ou igual a todos os horários iniciais das viagens alocadas à tarefa t. Quando a viagem não está alocada à tarefa t, ou seja, $x_{t,v} = 0$, o valor de i_t é restringido apenas pelo valor 1440, que é o valor máximo de minutos que i_t pode receber.

$$i_t \leq HI[v] * x_{t,v} + 1440 * (1 - x_{t,v}) \quad \forall t \in \mathcal{T},\ \forall v \in \mathcal{V} \tag{4.8}$$

As Equações (4.9), de modo similar às Equações em (4.8), definem o valor da variável f_t. Elas definem que o valor deverá ser maior que horário de término de todas as viagens alocadas a t. Quando a viagem não estiver alocada à t, é adicionada a restrição de que o valor deve ser maior que zero.

$$f_t \geq HI[t] * x_{t,v} \quad \forall t \in \mathcal{T},\ \forall v \in \mathcal{V} \tag{4.9}$$

As restrições em (4.10) delimitam que a diferença entre o tempo trabalhado e tempo decorrido entre o início e o fim da tarefa seja no máximo 60 minutos, de modo que o tempo de

espera entre viagens de uma mesma tarefa não seja muito grande. As restrições em (4.8), (4.9) e (4.10) foram adicionadas após ser percebido que o modelo gerava tarefas com o tamanho correto, mas bem espaçadas durante o dia. Tornando inviáveis o uso dessas tarefas para a alocação dos motoristas, pois de acordo com a lei o máximo de tempo de espera dos motoristas deve ser de no máximo 2 horas.

$$\sum_{v=0}^{V-1} x_{t,v} * L[v] \geq f_t - i_t - 60 \quad \forall t \in \mathcal{T} \tag{4.10}$$

As restrições apresentadas nas Equações (4.11) deduzem os valores das variáveis b_t. Como essas variáveis guardam a diferença entre o tamanho da tarefa t e o tamanho ideal, 220 minutos, seus valores são utilizados para deduzir valores de horas extras e horas desperdiçadas.

$$\sum_{v=0}^{V-1} L[v] * x_{t,v} = 220 * z_t + b_t \quad \forall t \in \mathcal{T} \tag{4.11}$$

As restrições em (4.12) fazem a dedução dos valores de d_t. Note que, como d_t deve ser maior que zero, estabelecido pelo seu domínio, apenas os valores negativos de b_t serão armazenados em d_t, ou seja, a quantidade de minutos a menos que 220 minutos.

$$d_t \geq -b_t \quad \forall t \in \mathcal{T} \tag{4.12}$$

As restrições mostradas nas Equações (4.13) foram adicionadas para se deduzir a quantidade de horas extras do modelo. Como a variável também é limitada a valores positivos, apenas os valores não negativos de b_t são armazenados em e_t. Ou seja, a quantidade de minutos a mais que 220 minutos.

$$e_t \geq b_t \quad \forall t \in \mathcal{T} \tag{4.13}$$

As Equações (4.14) garantem que não haverão viagens com sobreposição de tempo na mesma tarefa, ou seja, deve haver uma sequência nos horários das viagens. Essas restrições definem que para cada par de viagens, se e segunda começa antes de a primeira terminar, então elas não podem estar na mesma tarefa.

$$x_{t,v_1} + x_{t,v_2} \leq 1 \quad \forall t \in \mathcal{T},\ \forall v_1 \in \mathcal{V},\ \forall v_2 \in \mathcal{V} : HI[v_2] < HT[v_1] \tag{4.14}$$

As restrições apresentadas nas Equações (4.15) são usadas para garantir que um motorista não tenha que mudar de posto no meio de uma tarefa, a não ser que essa mudança ocorra durante uma viagem. Elas definem que, em um par de viagens, se a segunda começa até 15 minutos depois e em um posto de controle diferente daquele em que a primeira viagem termina, então essa duas viagens não podem estar na mesma tarefa.

$$\begin{gathered} x_{t,v_1} + x_{t,v_2} \leq 1 \\ \forall t \in \mathcal{T},\ \forall v_1 \in \mathcal{V}, \\ \forall v_2 \in \mathcal{V} : PI[v_2] \neq PT[v_1], HI[v_2] - HT[v_1] >= 15 \end{gathered} \tag{4.15}$$

As restrições nas Equações (4.16) definem que viagens na mesma tarefa não podem ter intervalo entre o início da primeira e o término da segunda maior que 400 minutos (340 minutos do tamanho máximo somado a 60 do intervalo máximo). Essas restrições são desigualdades válidas para tentar reduzir o espaço de busca do problema.

$$\begin{gathered} x_{t,v_1} + x_{t,v_2} \leq 1 \\ \forall t \in \mathcal{T},\ \forall v_1 \in \mathcal{V}, \\ \forall v_2 \in \mathcal{V} : HT[v_2] - HI[v_1] > 340 + 60 \end{gathered} \tag{4.16}$$

4.2 Modelo para o Problema de Alocação de Escalas (PAE)

Este modelo também é adaptado de Nunes (2015), usando apenas as restrições referentes à alocação de tarefas aos motoristas. Ou seja, as restrições que tratam da carga de trabalho dos motoristas e de suas folgas.

4.2.1 Restrições do modelo PAE

O modelo PDT fornece a entrada para este modelo. Sendo assim, este modelo não trata de otimizar o tamanho das tarefas ou a sua forma de criação. Entretanto, ainda existem algumas semelhanças entre os modelos, como a função objetivo, as restrições de dedução de horas, dentre outras. Um paralelo entre os dois pode ser feito de modo que o PDT aloca viagens a tarefas e o PAE aloca tarefas a motoristas, e cada umas dessas alocações tem suas peculiaridades.

O conjunto de tarefas recebido como entrada é menor, em comparação com o PDT. Nesse modelo são adicionadas restrições sobre o tamanho e tempo de espera de uma jornada

de trabalho, e sobre o descanso entre jornadas. Além dessas, também foi tratada a questão do translado dos motoristas entre uma pegada e outra, que não é algo desejável que aconteça. As folgas não são tratadas diretamente no modelo, pois como as entradas são de alocações para um único dia, é esperado que o software que alimenta o modelo faça o preprocessamento da entrada, de modo a não colocar motoristas para a alocação no modelo PAE se esses já trabalharam a quantidade máxima de dias sem folga.

O modelo faz alocações utilizando também o conceito de pegada dupla. Essa estilo foi adotado para modelar os requisitos da lei, que diz que um motorista tem direito a um intervalo para descanso a cada 240 minutos trabalhados. Também é tratada a questão do tempo de espera, ou seja, quando o tempo de espera entre a execução de tarefas for maior que 120 minutos, esse excedente deve ser considerado como hora trabalhada. Essa característica foi abordada por meio do uso de duas variáveis de alocação, a e b, que são apresentadas mais detalhadamente na Seção 4.2.4.

4.2.2 Conjuntos e índices

As variáveis usadas para construir os conjuntos de índices do modelo são M, que guarda o número estimado de motoristas do modelo, e T, o número de tarefas que serão alocadas. O valor de T é dado pelo número exato de tarefas da entrada, já o valor de M é uma estimativa de motoristas, que é calculado pela Equação (4.17), onde g é um fator de multiplicação utilizado para dar uma folga na alocação. O valor de T é dividido por 2, pois como a alocação utilizará o conceito pegada dupla, cada motorista poderá ser alocado para no máximo duas tarefas. Com essas variáveis são criados dois conjuntos para indexar as variáveis de dedução e de entrada do modelo, o conjunto $\mathcal{M} = \{0, 1, 2, \ldots, M-1\}$ e o conjunto $\mathcal{T} = \{0, 1, 2, \ldots, T-1\}$.

$$M = g * \left\lceil \frac{T}{2} \right\rceil \tag{4.17}$$

4.2.3 Informações de entrada

O modelo PAE recebe como entrada as informações sobre as tarefas que foram criadas pelo modelo PDT e as informações sobre os motoristas. As informações sobre as tarefas são as seguintes: $HI[t]$ e $HT[t]$, que indicam os horários de início e término da tarefa respectivamente; $PI[t]$ e $PT[t]$, que indicam os postos onde as tarefas inciam e terminam

respectivamente; $L[t]$, que indica o tamanho da tarefa em minutos; $O[t]$, que indica os minutos de espera da tarefa; e por fim, também é fornecido como entrada o tipo de dia TD, que pode ser 1 para dias do tipo DOMINGO e 0 para os demais tipos, UTIL e SABADO.

As informações necessárias dos motorista são apenas as variáveis $DA[m]$, que guardam o horário de término da jornada do motorista m no dia anterior. O modelo também recebe como entrada as variáveis C e C_e, custo da hora e custo da hora extra, respectivamente. Para cada motorista o valor das variáveis $DA[m]$ é o $HT[t]$ de alguma tarefa aleatória do tipo de dia anterior, essa escolha é feita na hora da criação das entradas do modelo. A escolha aleatória foi feita pois não temos acesso a informações reais sobre os motoristas.

4.2.4 Variáveis de decisão

As variáveis de decisão usadas no modelo são listadas a seguir:

- $a_{m,t} \in \{0,1\}$: indica se motorista m foi alocado para a tarefa t na primeira parte do seu turno, ou seja, a primeira pegada.
- $b_{m,t} \in \{0,1\}$: indica se motorista m foi alocado para a tarefa t na segunda parte do seu turno, ou seja, a segunda pegada.
- $t_m \in \{0,1\}$: indica se o motorista m está sendo utilizado na alocação.
- $b_m \in \mathbb{Z}$: guarda a diferença de tempo entre as horas trabalhadas do motorista m e o tamanho de turno ideal de 440 min. Como no modelo PDT, essa variável é utilizada para deduzir os valores de horas extras e horas não aproveitadas.
- $e_m \in \mathbb{Z}^+$: indica a quantidade de horas extras alocadas ao motorista m;
- $d_m \in \mathbb{Z}^+$: indica a quantidade de horas não aproveitadas do motorista m;

4.2.5 Modelagem do PAE

A função objetivo, apresentada na Equação (4.18), é a mesma do modelo PDT. Entretanto, ao contrário do modelo para o PDT, neste modelo a expressão fornece o custo real da alocação e não apenas uma estimativa. Assim, o modelo tentará balancear a alocação aos motoristas, colocando uma tarefa que tem horas extras em uma pegada e uma tarefa que tem horas desperdiçadas na outra, para diminuir o custo geral da jornada.

$$\text{Min} \sum_{t=0}^{T-1} d_t * C + e_t * C_e \tag{4.18}$$

Da mesma forma que os outros modelos, as Equações (4.19) definem que todas as

tarefas devem ser atendidas por exatamente um motorista, e também definem que uma tarefa não pode estar alocada em mais de uma pegada.

$$\sum_{m=0}^{M} (a_{m,t} + b_{m,t}) = 1$$
$$\forall t \in \mathcal{T} \tag{4.19}$$

As restrições nas Equações (4.20) definem o valor das variáveis t_m, ou seja, elas definem se o motorista foi ou não utilizado na alocação. Diferente do modelo de Nunes (2015), essa variável não é usada para a função objetivo, apenas nas definições de outras restrições.

$$t_m \geq a_{m,t} + b_{m,t}$$
$$\forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{4.20}$$

As restrições em (4.21) são desigualdades válidas para o problema. Elas definem que se um motorista está sendo usado na alocação, então ele deve ter sido alocado para no máximo duas pegadas. Essas restrições são derivadas de uma outra desigualdade válida que seria: sempre que o motorista é usado na alocação ele deverá ser alocado para as duas pegadas. Entretanto essa segunda restrição tornaria o modelo inviável quando o número de tarefas for ímpar.

$$\sum_{t=0}^{T-1} (a_{m,t} + b_{m,t}) \leq 2 * t_m$$
$$\forall m \in \mathcal{M} \tag{4.21}$$

As restrições apresentadas nas Equações (4.22), (4.23) e (4.24) são iguais às do modelo PDT (Equações 4.11, 4.12, e 4.13). A diferença entre elas está apenas no significado: as restrições em 4.22 deduzem a diferença entre o que foi alocado ao motorista e tamanho ideal do turno, que é maior que o tamanho ideal de tarefa; as restrições em 4.23 deduzem a quantidade de horas extras; e as restrições em 4.24 deduzem a quantidade de horas desperdiçadas.

$$\sum_{m=0}^{M-1} L[v] * (a_{m,t} + b_{m,t}) = 440 * t_m + b_m$$
$$\forall m \in \mathcal{M} \tag{4.22}$$

$$d_t \geq -b_t$$
$$\forall m \in \mathcal{M} \tag{4.23}$$

$$e_t \geq b_t$$
$$\forall m \in \mathcal{M} \tag{4.24}$$

As Equações em (4.25) definem que a primeira pegada deve ser indicada nas variáveis a e que não podem haver sobreposições entre as tarefas indicadas nas variáveis a e b. Isso é feito a partir da definição de que para um mesmo motoristas, para cada duas tarefas, t_1 e t_2, se a segunda começa antes da primeira terminar, então essas tarefas não podem estar alocadas ao mesmo motorista.

$$\begin{gathered} a_{m,t_1} + b_{m,t_2} \leq 1 \\ \forall m \in \mathcal{M}, \forall t_1 \in \mathcal{T}, \\ \forall t_2 \in \{t_1 + 1..T - 1\}, HI[t_2] < HT[t_1] \end{gathered} \tag{4.25}$$

As restrições apresentadas nas Equações (4.26) foram adicionadas ao modelo para que o mesmo ficasse em conformidade com o que foi estabelecido na lei: a cada 4 horas de trabalho o motorista tem direito a 30 minutos de descanso, que pode ser fracionado. Assim as restrições definem que se deve ter um intervalo mínimo de 30 minutos entre a primeira pegada e a segunda, levando em consideração o tempo de espera que já alocado dentro de cada tarefa.

$$\begin{gathered} a_{m,t_1} + b_{m,t_2} \leq 1 \\ \forall m \in \mathcal{M}, \forall t_1 \in \mathcal{T}, \forall t_2 \in \{t_1 + 1..T - 1\}, \\ HI[t_2] - HT[t_1] < 30 - O[t_1] - O[t_2] \end{gathered} \tag{4.26}$$

Também para garantir a conformidade com a lei referente ao intervalo entre pegadas, foram adicionadas as restrições em (4.27). Essas restrições delimitam o intervalo máximo entre as jornadas, também levando em consideração o tempo de espera de cada tarefa. Foi estabelecido o limite máximo de 2 horas, para evitar que o turno de trabalha se extenda por um período de tempo muito longo.

$$\begin{gathered} a_{m,t_1} + b_{m,t_2} \leq 1 \\ \forall m \in \mathcal{M}, \forall t_1 \in \mathcal{T}, \forall t_2 \in \{t_1 + 1 \ldots T - 1\}, \\ HI[t_2] - HT[t_1] > 120 - O[t_1] - O[t_2] \end{gathered} \tag{4.27}$$

As Equações (4.28) restringem que a segunda pegada deve começar no posto em que a primeira termina. Essa restrição é uma restrição de logística e foi implementada para que não haja atrasos, devido à necessidade de locomoção do motorista de um posto a outro para a

execução da segunda parte da jornada de trabalho.

$$\begin{gathered} a_{m,t_1} + b_{m,t_2} \leq 1 \\ \forall m \in \mathcal{M},\ \forall d \in \mathcal{D}, \forall t_1 \in \mathcal{T}, \\ \forall t_2 \in \{t_1 + 1 \ldots T - 1\} : PI[t_2] \neq PT[t_1] \end{gathered} \tag{4.28}$$

Como o modelo PDT trabalha apenas com a criação de tarefas de um dia específico, devido ao tamanho da entrada, o PAE trata apenas as restrições que usam os dados do dia anterior para garantir a continuidade das alocações. Isso é tratado nas restrições a seguir.

As restrições apresentadas em (4.29) garantem o descanso mínimo entre jornadas do motorista, definindo que o mesmo não pode iniciar sua jornada em uma tarefa que começa antes das suas horas de descanso estarem completas.

$$\begin{gathered} a_{m,t} = 0 \\ \forall m \in M, \forall t \in T, HI[t] - \mathrm{DA}[m] > 660 \end{gathered} \tag{4.29}$$

5 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados e discutidos os resultados obtidos através da execução deste trabalho. Na Seção 5.1 são apresentados os resultados obtidos durante a replicação dos experimentos de Nunes (2015). Na Seção 5.2 foram feitas análises do desempenho do modelo quando são removidas algumas restrições e quando são adicionadas soluções viáveis iniciais. Na Seção 5.3 são discutidas algumas características do problema que não estão inseridas no modelo de Nunes (2015). Na Seção 5.4 são apresentados os resultados dos experimentos feitos com os modelos do Capítulo 4.

Todos os experimentos reportados aqui foram executados em um servidor com um processador Intel(R) Xeon(R) CPU E31240, 8GB de memória RAM e sistema operacional Ubuntu 14.04.5 LTS.

5.1 Replicação dos experimentos de Nunes (2015)

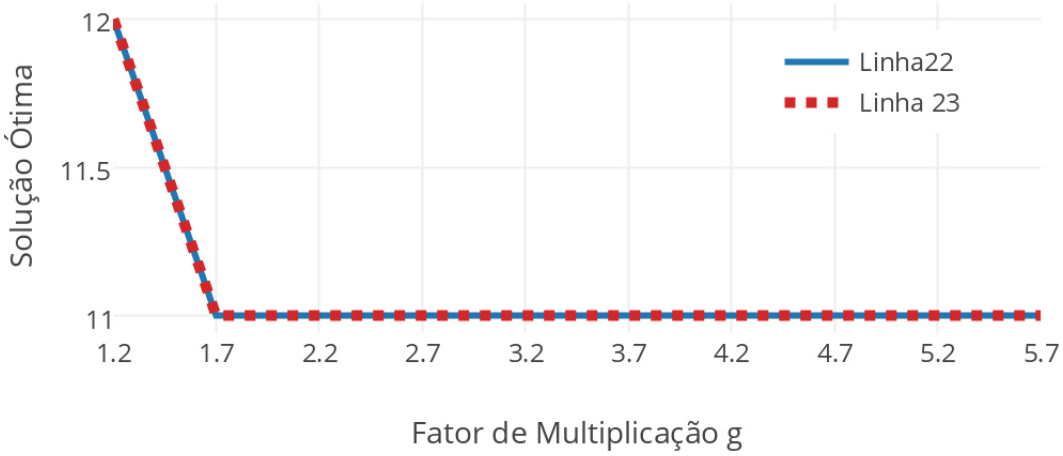
O modelo foi implementado utilizando a linguagem C++ e a biblioteca do *solver* ILOG CPLEX Optimization Studio da IBM (CPLEX, 2009). As instâncias utilizadas na replicação dos experimentos foram cedidas pelo autor de Nunes (2015). Essas instâncias têm tarefas de até 440 minutos, variando apenas o valor de g na estimativa do número de motoristas disponíveis para a alocação. Segundo Nunes (2015) as tarefas foram criadas a partir das viagens reais das linhas 22 e 23 de ônibus urbanos de Fortaleza. Entretanto, as informações sobre os motoristas são fictícias.

Os resultados obtidos a partir da implementação do modelo original são mostrados na Figura 4. O número de motoristas usados na escala encontrado pela replicação é o mesmo relatado por Nunes (2015). O mesmo se repetiu para as demais instâncias de outras linhas. Vale ressaltar que nos testes apresentados nesta seção e na seguinte não foram utilizadas as restrições nas Equações (3.10), pois Nunes (2015) também não as utilizou nessas fases de estudo.

5.2 Análise das restrições e da adição de uma solução viável inicial no modelo de Nunes (2015)

Com o objetivo de deixar o modelo mais realista, as restrições (3.5), (3.6) e (3.7) foram retiradas do modelo, pois como Nunes (2015) citou em sua dissertação, em alocações feitas pelas empresas, os motoristas podem trabalhar em qualquer linha. Sem essas restrições o

Figura 4 – Gráfico com as soluções do modelo replicado



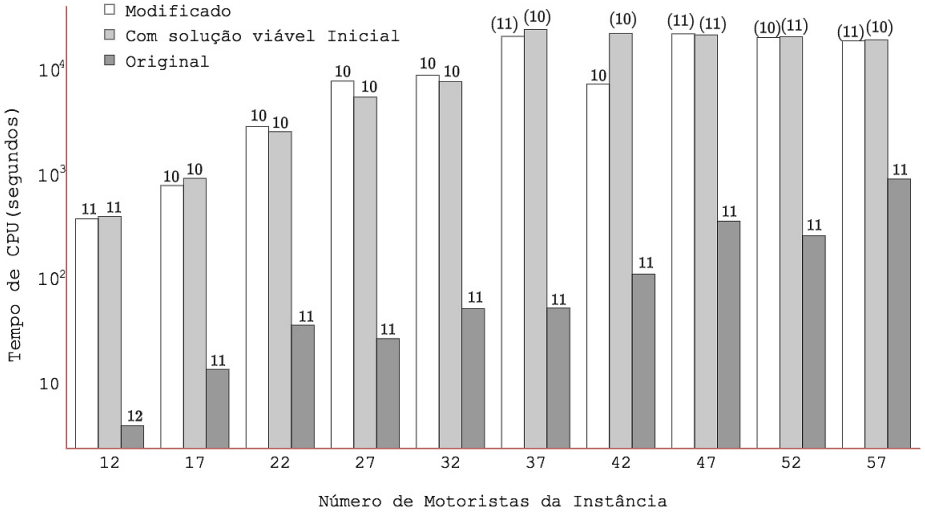
Fonte: Elaborada pela autora.

modelo fica mais próximo da realidade das empresas, pois os motoristas poderão trabalhar em qualquer tabela de uma linha. Uma comparação do desempenho do modelo com e sem essas restrições são discutidas a seguir. Também é comparado o comportamento do modelo quando adicionado solução viável inicial, obtida através de um modelo de programação por restrições desenvolvido em projeto de pesquisa paralelo (SOUZA, 2016).

Os resultados dos experimentos desta Seção referem-se apenas à linha 22, com tamanhos de tarefa fixos e alterando-se o valor de M. Foi utilizado um *timeout* de 1 hora para a execução do *solver*. Na Figura 5 são mostrados, para cada valor de M no eixo x, o tempo de CPU para o modelo sem as restrições (3.5), (3.6) e (3.7), o modelo sem as restrições (3.5), (3.6) e (3.7) e com uma solução viável inicial, e o modelo original, respectivamente. Os números acima de cada barra representam o número de motoristas usados em cada instância, e os que estão entre parênteses indicam que a solução não foi provada como ótima dentro do tempo limite. Já na Figura 6 são mostrados o número de motoristas usados na escala.

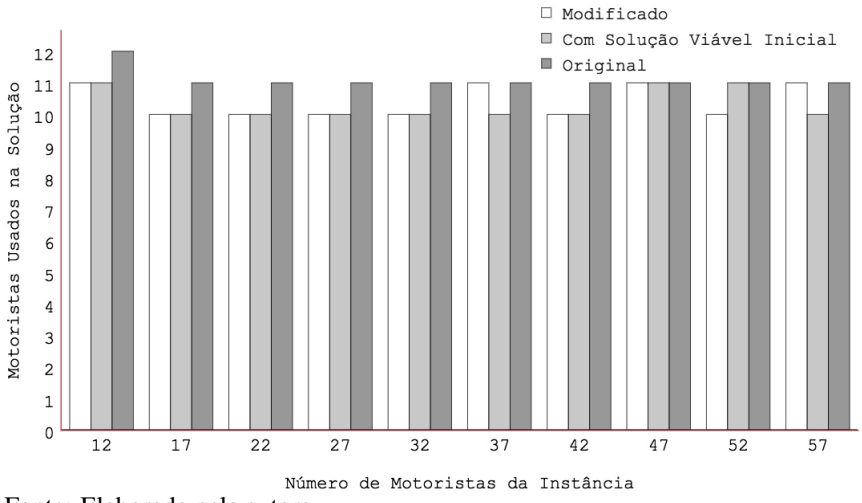
Com os resultados mostrados nas Figuras 5 e 6 temos que, ao retirar as restrições (3.5), (3.6) e (3.7) o modelo perde desempenho computacional, mas encontra soluções que, mesmo quando não provada a otimalidade, são melhores que uma solução ótima com essas restrições. Ao fornecer uma solução viável inicial ao modelo, em alguns casos a resposta pôde ser melhorada e o tempo de CPU diminuído. Um caso que vale a pena ser notado é o da instância com 52 motoristas, onde o modelo com a solução viável inicial resultou em um valor maior que o modelo sem essa solução. Acreditamos que o motivo do ocorrido seja porque a busca se iniciou em ramo ruim da árvore.

Figura 5 – Gráfico com tempos de CPU dos modelos comparados.



Fonte: Elaborada pela autora.

Figura 6 – Gráfico com o valor das soluções encontradas pelos modelos.



Fonte: Elaborada pela autora.

5.3 Identificação e implementação de restrições que ainda não são atendidas pelo modelo de Nunes (2015)

Ao avaliar as soluções reportadas na Seção 5.2, percebemos que as jornadas de trabalho estavam muito fracionadas, com muitos períodos de espera entre tarefas. Também percebemos que o modelo não conseguiria tratar do controle desses tempos de espera. Assim, ao adicionar o escopo de empresa ao modelo, esse quesito foi levado em consideração.

Além de não limitar o fracionamento, o modelo não garante aos motoristas os intervalos de descanso previstos por lei. Foi também percebido que o modelo não divide de fato o problema em dois subproblemas, e a criação das tarefas não é exemplificado no trabalho de Nunes (2015). Depois de identificados esses pontos, eles foram incorporados aos modelos que são apresentados nas seções a seguir.

5.4 Modificação do modelo para adicionar o escopo de empresa ao modelo

Foram realizados experimentos com os modelos PDT e PAE separadamente. Os resultados são apresentados e discutidos a seguir.

5.4.1 Modelo PDT

O modelo PDT, apresentado na Seção 4.1, foi implementado utilizando a linguagem matemática *Optimization Programming Language* (OPL). Essa mudança de linguagem foi feita porque a OPL é uma linguagem mais fácil de se escrever e depurar o modelo. Isso é consequência do fato de que a OPL interpreta o modelo a partir de uma linguagem matemática de alto nível. O parâmetro de limite de tempo de execução para todos os experimentos desta seção também são de 1 hora.

5.4.1.1 Primeiro Experimento com o modelo PDT - pré-processamento da entrada

Com o modelo PDT implementado, foram criadas as entradas para o modelo. Inicialmente cada dia foi transformado em uma entrada e o modelo foi executado. Entretanto não foi possível carregar o modelo em memória para nenhuma das instâncias, devido ao número de viagens.

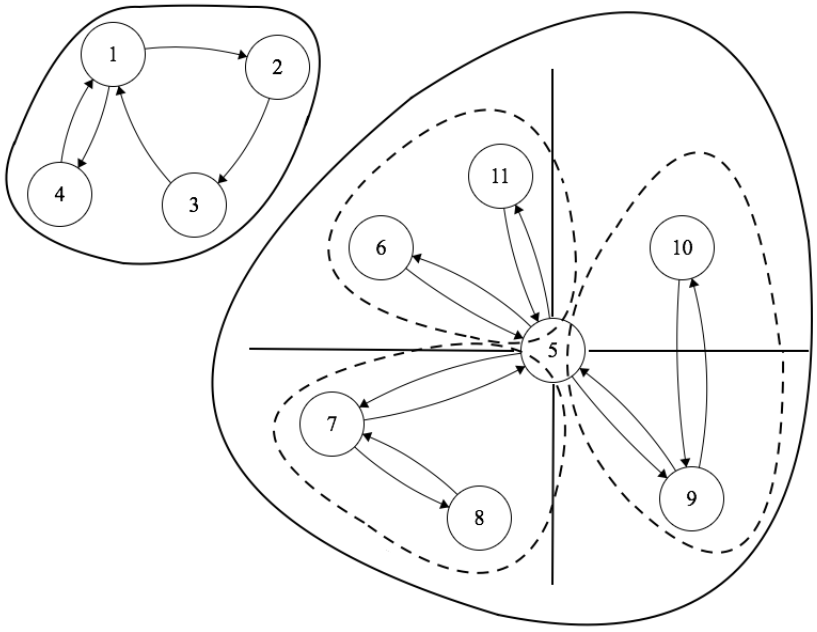
Para contornar esse problema sem comprometer muito a otimalidade da solução, cada entrada foi dividida em partes menores. Para fazer essa divisão, foi criado um grafo direcionado com os nós sendo os postos de controles. Neste grafo, há uma aresta ligando um posto p_1 a um outro posto p_2 se existir uma viagem que começa em p_1 e termina em p_2.

Foi executado um algoritmo, retirado de Cormen et al. (2001), para determinar as componentes fortemente conexas. A partir dessas componentes, foram separadas as viagens que passavam pelos postos de cada componente. Foi ainda escolhida a maior das componentes para ser separada em quadrantes a direita, esquerda, acima e abaixo do posto com o maior

número de viagens passando por ele. Essa separação em quadrantes foi feita com base na localização geográfica dos postos fornecida pela GIST. Cada quadrante representa uma região que foi separada como uma entrada para o PDT.

Na Figura 7 temos um exemplo dessa separação, onde temos 11 postos de controle. Os postos {1,2,3,4} e {5,6,7,8,9,10,11} formam as duas componentes desse exemplo, e estão contornados em linhas contínuas. Dentro da componente com o maior número de postos podemos ver a divisão dos quadrantes, linhas verticais e horizontais, e também as subdivisões de região, em linhas tracejadas. Note que à direita do posto 5 existem dois postos que, apesar de estarem em quadrantes diferentes, estão na mesma região, isso se dá pelo fato de o posto 10 não ter ligação direta com o posto central, posto 5. Ou seja, durante a separação é considerada apenas a localização do posto vizinho ao central. Vale lembrar também que como não existem viagens que começam e terminam no mesmo posto de controle, todas as tarefas que tem início ou fim no posto central serão separadas para a região a qual o segundo posto pertence.

Figura 7 – Exemplo de divisões dos postos de controle em componentes e regiões



Fonte: Elaborada pela autora.

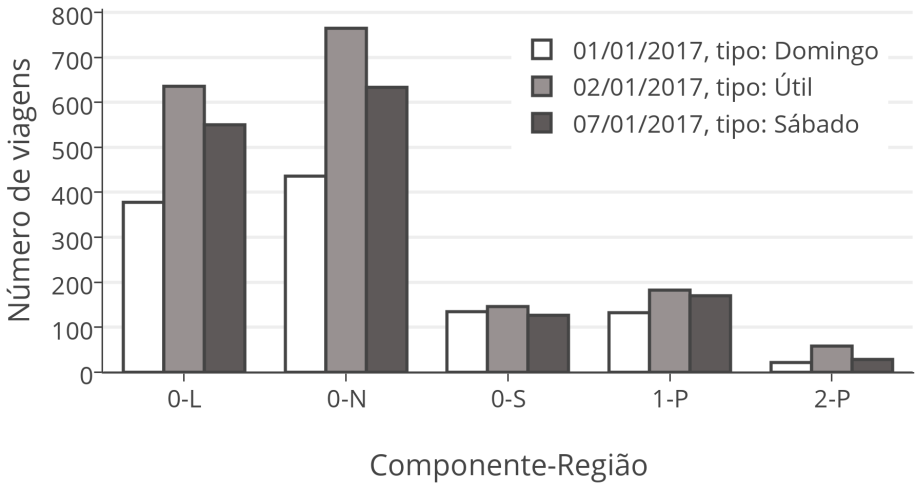
Essas separações em componentes não comprometem a qualidade final da solução, pois como as restrições em 4.15 definem que as viagens dentro de uma mesma tarefa devem ter uma continuidade nos postos de controle, todas as viagens que estão na mesma tarefa devem estar na mesma componente. Já no caso da separação de áreas pode haver um prejuízo no valor

da solução final, mas adiciona ao modelo uma qualidade desejável que é a restrição de manter as tarefas de um motorista dentro de uma mesma região.

Vale notar que essa divisão não será equilibrada. O tamanho das novas instâncias dependerá da demanda de cada linha em determinado tipo de dia. Os dados utilizados para os experimentos neste trabalho geraram 3 componentes. Dessas 3, a maior foi dividida em regiões como no exemplo da Figura 7. [1]

Na Figura 8 são apresentados o número de viagens para cada instância criada para os 3 dias estudados. Podemos ver que a divisão gerou grupos de viagens com tamanhos variados. As duas últimas componentes, que geraram apenas uma região cada, tem bem menos tarefas que a região N ou a S da Componente 0. É também perceptível a diferença no impacto do tipo de dia sobre o número de viagens.

Figura 8 – Gráfico de comparação dos tamanho das instâncias.



Fonte: Elaborada pela autora.

Feito essa separação nas viagens de entrada do modelo, os experimentos foram executados e os resultados são apresentados na Tabela 1. A tabela está dividida em duas partes, de modo que na primeira metade temos os resultados referentes às instâncias com 10% de folga, ($g = 1.1$), no valor estimado de T, e na segunda metade esse valor de folga é de 50%, ($g = 1.5$). Na primeira coluna estão listados os identificadores de cada instância.

Na segunda coluna estão representados os valores da função objetivo, quando

[1] A nomenclatura dessas instâncias tem o seguinte formato: di_j-k, com di representando o dia, j a componente e k a região das viagens dessa entrada.

disponíveis, ou as siglas que indicam o retorno do *solver* para cada instância. A sigla MNC indica que o *solver* não conseguiu encontrar uma solução para relaxação linear do problema, por falta de memória. Neste caso, o modelo chegou a ser carregado em memória. A sigla TBS indica que o modelo conseguiu encontrar uma solução para a relaxação, entretanto, o tempo de execução não foi suficiente para encontrar uma solução inteira. A sigla MNL significa que o modelo não pode ser carregado em memória. Há também *Inf* como possível valor dessa coluna, que significa que o modelo é inviável.

A terceira coluna indica o tempo em segundos que a instância executou. Esse tempo é marcado pelo próprio *solver* e é usado para interromper a execução quando o limite que foi definido antes do início da execução for atingido. A última coluna indica o GAP da solução, quando disponível. Do mesmo modo, estão dispostas as colunas da segunda metade da tabela. As linhas da tabela que estão em cinza escuro significam que o status da solução daquela instância indica que é ótima, e as linhas cinza claro indicam que o modelo encontrou uma solução viável.

Tabela 1 – Resultados do primeiro experimento com o PDT

g = **1.1**				g = **1.5**			
Entrada	**Obj.**	**Tempo**	**GAP**	**Entrada**	**Obj.**	**Tempo**	**GAP**
d1_0-L	MNC	2605.86	-	d1_0-L	MNC	233.48	-
d1_0-N	MNC	922.97	-	d1_0-N	MNC	202.71	-
d1_0-S	TBS	3600.08	-	d1_0-S	TBS	3600.09	-
d1_1-P	TBS	3601.38	-	d1_1-P	TBS	3600.2	-
d1_2-P	530	0.06	0.00%	d1_2-P	530	0.09	0.00%
d2_0-L	MNL	-	-	d2_0-L	MNL	-	-
d2_0-N	MNL	-	-	d2_0-N	MNL	-	-
d2_0-S	*Inf.*	*7.84*	-	d2_0-S	TBS	3600.17	-
d2_1-P	TBS	3600.4	-	d2_1-P	TBS	3600.23	-
d2_2-P	1120	3600.01	4.46%	d2_2-P	1120	3600.01	4.46%
d7_0-L	MNL	-	-	d7_0-L	MNL	-	-
d7_0-N	MNL	-	-	d7_0-N	MNL	-	-
d7_0-S	*Inf.*	*4.46*	-	d7_0-S	TBS	3600.03	-
d7_1-P	TBS	3600.33	-	d7_1-P	TBS	3600.59	-
d7_2-P	820	0.29	0.00%	d7_2-P	820	0.28	0.00%

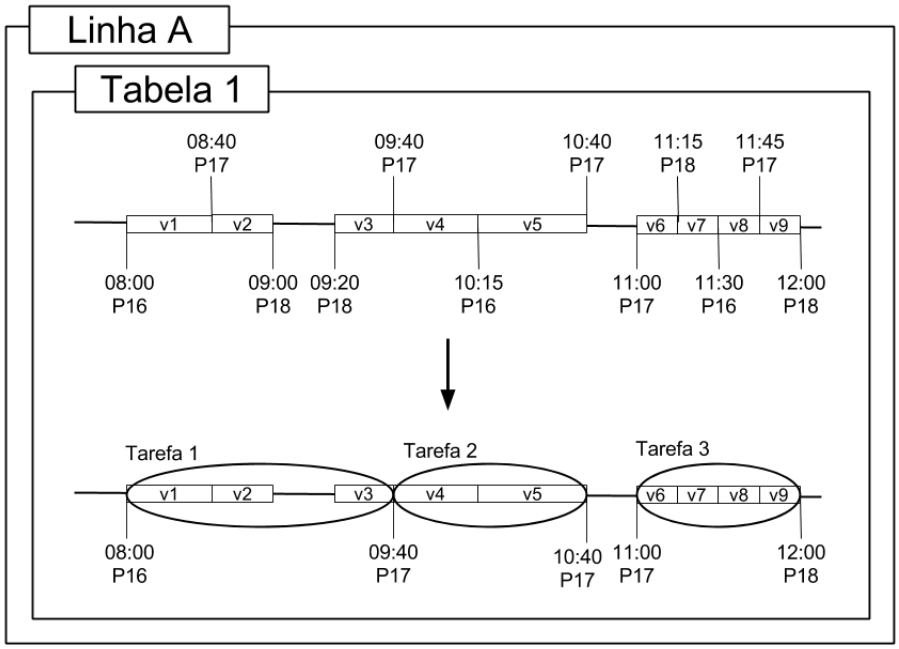
Fonte: Elaborada pela autora.

Podemos ver que mesmo com o tamanho das entradas reduzido, os resultados ainda não são satisfatórios, pois para ambos os valores de g, apenas instâncias pequenas do problema deram resultados, e ainda assim houveram casos em que a solução não é ótima.

Algo que percebemos quando analisamos as soluções dessas pequenas instâncias é que o caso ilustrado na Figura 9 ocorreu em todas as instâncias que retornaram solução. A figura ilustra um caso onde há um pequeno intervalo entre as duas últimas viagens da tarefa que foi construída, que gera um problema quando a viagem que acontece logo após está em outra tarefa.

O problema é que não há garantias de que essas tarefas serão alocadas ao mesmo motorista que executou a primeira. Assim a pode haver uma necessidade de trocar o motorista, essa possível troca pode ocasionar um atraso na execução das tarefas.

Figura 9 – Exemplo de um caso indesejado na criação de tarefas.



Fonte: Elaborada pela autora.

Nesse experimento existem alguns casos curiosos que valem a pena ser analisados, por exemplo, o caso das instâncias $d2\_0-S$ e $d7\_0-S$. Nessas instâncias, com o valor de apenas 10% de folga no valor estimado de T, o modelo é inviável, entretanto, quando se aumenta o número de tarefas, ($g = 1.5$), o *solver* consegue encontrar uma solução para a relaxação. Embora isso não signifique que há uma solução inteira para o modelo, é um caso interessante que mostra que a estimativa do número de tarefas é bem grosseira e não leva em consideração as outras restrições do modelo.

5.4.1.2 Segundo experimento com o modelo PDT - heurística de soma

Os resultados da seção anterior mostram que as entradas, mesmo depois do pré-processamento, ainda são muito grandes. Portanto, foi desenvolvido um novo pré-processamento, a ser executado após o primeiro, que possa diminuir ainda mais a entrada do PDT.

Esse segundo pré-processamento foi feito levando em consideração que o motorista, ao trocar de tabela ou linha dentro da execução da tarefa, deve trocar também de ônibus. Deste

modo, o ideal seria que o motorista executasse o máximo possível de tarefas consecutivas no mesmo ônibus, pois essa troca pode acarretar atrasos.

Pensando nisso, o novo pré-processamento faz uma "soma" todas as **viagens consecutivas**, que são definidas como viagens que são da mesma linha e tabela e que não têm intervalos entre elas. Essa "soma" consiste na criação de uma nova viagem, chamada de **viagem-somada**, que começa no mesmo horário e posto de controle da primeira viagem, e termina no mesmo horário e posto de controle da última viagem. A soma é feita de modo a juntar o maior número de tarefas consecutivas possível. Depois da soma, as viagens-somadas que tem mais de 240 minutos são separadas em viagens com até 220 minutos. Essa divisão é necessário pois o motorista tem direito a intervalos de de descanso a cada 4 horas de trabalho. As viagens-somadas com tamanho entre 220 e 240 minutos foram deixadas com esse tamanho, pois a remoção de última tarefa resultaria em uma viagem de uma tabela de tamanho pequeno ou sem conexão com outras viagens.

Após o uso da heurística de "soma", foram feitos novos experimentos. Foi feita uma alteração no modelo na restrição (4.14), onde a restrição agora define que as viagens consecutivas não podem estar na mesma tarefa. Isso se dá pois é assumido que um motorista pode ter que trocar de ônibus entre essas viagens, de modo que agora haverá uma folga entre o fim de uma viagem e início da seguinte dentro de uma mesma tarefa.

Os resultados são apresentados na Tabela 2 e discutidos a seguir. A organização da tabela segue a mesma da tabela anterior. Há apenas uma pequena diferença nos valores possíveis da segunda coluna. Nesses experimentos ocorreu um novo caso, o MBC, que significa que houve estouro de memória durante a fase de *Branch and Cut*. Ou seja, a árvore de busca se expandiu além dos limites de memória da máquina e então a execução foi interrompida.

Tabela 2 – Resultados do segundo experimento com o PDT - Utilizando heurística de soma

g = 1.1				g = 1.5			
Instância	**Obj.**	**Tempo (s)**	**GAP (%)**	**Instância**	**Obj.**	**Tempo**	**GAP**
d1_0-L	TBS	3601.09	-	d1_0-L	TBS	3600.29	-
d1_0-N	22465	3600.2	97.37%	d1_0-N	32515	3600.13	98.19%
d1_0-S	TBS	3600.8	-	d1_0-S	TBS	3637.88	-
d1_1-P	10065	3600.02	27.80%	d1_1-P	10065	3600.03	28.20%
d1_2-P	2155	0.03	0.00%	d1_2-P	61700	0.04	0.00%
d2_0-L	TBS	3600.78	-	d2_0-L	TBS	3605.68	-
d2_0-N	TBS	3601.35	-	d2_0-N	TBS	3607.34	-
d2_0-S	7880	2828.72	0.00%	d2_0-S	MBC	1527.19	71.04%
d2_1-P	16225	3600.06	41.93%	d2_1-P	10890	3600.04	14.14%
d2_2-P	3295	16.8	0.00%	d2_2-P	3295	58.27	0.00%
d7_0-L	TBS	3600.41	-	d7_0-L	TBS	3600.32	-
d7_0-N	TBS	3600.58	-	d7_0-N	TBS	3600.68	-
d7_0-S	7725	2476.92	0.00%	d7_0-S	MBC	2495.2	44.98%
d7_1-P	MBC	1809.28	4.28%	d7_1-P	10805	3600.01	40.70%
d7_2-P	1220	0.13	0.00%	d7_2-P	1220	0.07	0.00%

Fonte: Elaborada pela autora.

Os resultados desse segundo experimento foram bem mais satisfatórios que os anteriores, pois para todas as instâncias e variações do fator g, o modelo pôde ser carregado em memória. Outro ponto positivo do uso dessa heurística foi que o número de instâncias que tiveram soluções ótimas aumentou de 2 para 5 no caso do fator $g = 1.1$, e de 2 para 3 no caso do fator $g = 1.5$. O número de soluções viáveis encontradas também aumentou em ambos os valores de g.

As linhas que indicam os resultados para as instâncias $d7\_1 - P$, com $g = 1.1$, e $d2\_0 - S$ e $d7\_0 - S$ com $g = 1.5$, apresentam como valor objetivo a sigla MBC, que indica que houve um estouro de memória. Contudo podemos ver o GAP da melhor solução inteira conseguida antes da falha.

Com estas mudanças nas entradas podemos perceber o impacto do fator de multiplicação no tempo de execução e no uso da memória. Podemos notar ainda que os valores da função objetivo são os mesmos em todos os casos, com exceção da instância $d_1\_2 - P$ que resultou em um valor objetivo maior, mas que retornou a mesma alocação de viagens à tarefas. Essa mudança se deu apenas na escolha dos valores de horas extras e horas desperdiçadas.

Os valores ótimos da função objetivo desses experimentos também são maiores do que os valores obtidos nos experimentos anteriores. Isso aconteceu porque nesse experimento foi evitado que acontecesse o caso demonstrado na Figura 9. É importante lembrar que mesmo o custo sendo maior, as soluções obtidas nesses experimentos são mais realistas ao considerar as

implicações logísticas de trocas de ônibus e de motoristas.

5.4.1.3 Terceiro experimento com o modelo PDT - analisando o impacto das desigualdades válidas

Nesse terceiro experimento, foi analisado o impacto das desigualdades válidas no tempo de execução e na qualidade da solução do modelo. Foram removidas as restrições apresentadas nas Equações (4.16). Os resultados são apresentados a seguir na Tabela 3.

Tabela 3 – Resultados do terceiro experimento com o PDT - Removendo as desigualdades válidas

g = 1.1				g = 1.5			
Instância	**Obj.**	**Tempo (s)**	**GAP (%)**	**Instância**	**Obj.**	**Tempo**	**GAP**
d1_0-L	TBS	3600.07	-	d1_0-L	TBS	3600.11	-
d1_0-N	TBS	3605.87	-	d1_0-N	TBS	3600.06	-
d1_0-S	MBC	1806.41	-	d1_0-S	MBC	1438.46	-
d1_1-P	10065	3600.36	27.93%	d1_1-P	MBC	3476.14	29.83%
d1_2-P	2155	0.04	0.00%	d1_2-P	2155	0.08	0.00%
d2_0-L	TBS	3600.51	-	d2_0-L	TBS	3600.23	-
d2_0-N	TBS	3600.26	-	d2_0-N	TBS	3600.14	-
d2_0-S	7880	3600.41	81.22%	d2_0-S	7880	3600.2	91.56%
d2_1-P	11475	3600.56	18.08%	d2_1-P	10550	3600.21	11.17%
d2_2-P	3295	10.5	0.00%	d2_2-P	3295	0.97	0.00%
d7_0-L	TBS	3600.17	-	d7_0-L	TBS	3600.1	-
d7_0-N	TBS	3600.09	-	d7_0-N	TBS	3600.09	-
d7_0-S	7725	2251.06	0.00%	d7_0-S	7275	0.2	0.00%
d7_1-P	8680	3600.51	7.49%	d7_1-P	8680	3600.48	7.49%
d7_2-P	1220	0.06	0.00%	d7_2-P	1220	0.07	0.00%

Fonte: Elaborada pela autora.

Com a remoção das restrições das Equações (4.16), tivemos alguns resultados interessantes, como o da instância $d7\_1-P$ com $g = 1.1$, que obteve um GAP maior do que o do experimento na Seção 5.4.1.2, mas que não teve problemas com a memória e conseguiu retornar uma solução viável. Para a $d2\_0-S$ com $g = 1.5$ também não houveram problemas com a memória, retornando uma solução viável, além de o GAP ter sido melhorado nessa execução. Os valores para as instâncias que obtiveram soluções no experimento sem a heurística de soma (**EXP1**, Seção 5.4.1.1), no experimento com a heurística de soma (**EXP2**, Seção 5.4.1.1) e no experimento sem as desigualdades válidas (**EXP3**, Seção 5.4.1.3) são apresentados na Tabela 4.

Na instância $d2\_0-S$ com $g = 1.1$, temos que o valor da função objetivo é o mesmo que o do segundo experimento, sendo que no segundo experimento obteve-se solução ótima. A alteração no modelo fez com o que o solver não conseguisse provar que essa solução é ótima.

Esta remoção não impactou no valor das soluções ótimas, o que era esperado pois

Tabela 4 – Comparação dos resultados dos experimentos com o PDT

Instâncias	Obj.			GAP			Tempo		
	EXP 1	**EXP 2**	**EXP 3**	**EXP 1**	**EXP 2**	**EXP 3**	**EXP 1**	**EXP 2**	**EXP 3**
d1_0-N	MNC	22465	TBS	-	97.37%		922.97	3600.2	3605.87
d1_1-P	TBS	10065	10065	-	27.80%	27.93%	3601.38	3600.02	3600.36
d1_2-P	530	2155	2155	0.00%	0.00%	0.00%	0.06	0.03	0.04
d2_0-S	Inf.	7880	7880	-	0.00%	81.22%	7.84	2828.72	3600.41
d2_1-P	TBS	16225	11475	-	41.93%	18.08%	3600.4	3600.06	3600.56
d2_2-P	1120	3295	3295	4.46%	27.40%	37.87%	3600.01	16.8	10.5
d7_0-S	Inf.	7725	7725	-	0.00%	0.00%	4.46	2476.92	2251.06
d7_1-P	TBS	MBC	8680	-	4.28%	7.49%	3600.33	1809.28	3600.51
d7_2-P	820	1220	1220	0.00%	0.00%	0.00%	0.29	0.13	0.06

Fonte: Elaborada pela autora.

a restrição é uma desigualdade válida. Mas com essa remoção podemos perceber que as desigualdades válidas tiveram um impacto positivo em instâncias maiores, que é o caso das instâncias $d1\_0-N$, mas prejudicam as instâncias menores como a $d7\_1-P$.

5.4.1.4 Quarto experimento com o modelo PDT - analisando o impacto de uma solução viável inicial

Como os experimentos anteriores falharam em retornar pelo menos uma solução viável para cada uma das instâncias, desenvolvemos uma heurística para criar soluções viáveis inciais para o modelo PDT. O objetivo é melhorar os resultados do modelo PDT. Essa heurística recebe como entrada uma lista de viagens-somadas ordenada pelo horário de início e retorna uma conjunto de tarefas, e é apresentada no Algoritmo 1.

A heurística funciona da seguinte forma, enquanto o conjunto de viagens-somadas não estiver vazio, significando que todas as viagens já estão em alguma tarefa, é criado um novo conjunto com a viagem que começa mais cedo, a primeira da lista. É feita uma iteração sobre as viagens restantes, quando é verificado se cada uma das viagens restantes pode ser adicionada à tarefa. Após a iteração são removidas do conjunto de viagens aquelas que foram alocadas para a tarefa atual e a tarefa atual é adicionada ao conjunto de tarefas.

A função *PODE-ADICIONAR*(T, v) verifica se a viagem v pode ser adicionada à tarefa T. Para que uma viagem possa o ser adicionada a uma tarefa é necessário: que o posto inicial da viagem v seja o mesmo que o posto terminal da última viagem da tarefa T; que a viagem comece depois do término da última viagem de T; que a soma do tamanho das viagens da tarefa T com o tamanho da viagem v seja menor que 340; e por último é necessário que o tempo de espera, caso essa viagem seja adicionada, seja menor que 60 minutos.

Algoritmo 1: HEURÍSTICA CONSTRÓI_TAREFAS

Entrada: Conjunto de viagens-somadas, $\mathcal{V}$
Saída: Conjunto de Tarefas
início
  $\mathcal{T} \leftarrow \emptyset$
  enquanto $\mathcal{V}$ *diferente de* $\emptyset$ **faça**
    T = {$\mathcal{V}$.primeiro}
    para $v \in \mathcal{V} - \{\mathcal{V}.primeiro\}$ **faça**
      se *PODE-ADICIONAR(T, v)* **então**
        $T \leftarrow T \cup \{v\}$
      fim
    fimpara
    $\mathcal{V} \leftarrow \mathcal{V} - T$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$
  fim
  retorne $\mathcal{T}$
fim

Fonte: Elaborado pela autora.

Devido a sua simplicidade, a heurística executa muito rápido, por volta de 3 segundos, para criar soluções viáveis iniciais para todas as 15 instâncias do PDT apresentadas nas seções anteriores. Os resultados das entradas foram então inseridos no modelo PDT. Esse experimento é detalhado na Tabela 5 e discutido a seguir.

Tabela 5 – Valor das soluções dadas pela heurística Constrói_Tarefas

g = 1.1				g = 1.5			
Instância	**Obj.**	**Tempo**	**GAP**	**Instância**	**Obj.**	**Tempo**	**GAP**
d1_0-L	69930	3600.22	99.10%	d1_0-L	69930	3600.24	99.10%
d1_0-N	40615	3600.21	98.55%	d1_0-N	44665	3600.21	98.68%
d1_0-S	14075	3601.58	90.94%	d1_0-S	13625	3600.05	90.64%
d1_1-P	10315	3600.02	29.94%	d1_1-P	MBC	2409.34	27.64%
d1_2-P	2155	0.05	0.00%	d1_2-P	2155	0.19	0.00%
d2_0-L	99140	3601.37	100.00%	d2_0-L	99140	3601.06	99.38%
d2_0-N	103100	3602.33	100.00%	d2_0-N	103100	3600.36	99.90%
d2_0-S	MBC	2101.14	70.30%	d2_0-S	MBC	1271.47	70.30%
d2_1-P	12175	3600.07	49.13%	d2_1-P	12500	3600.02	35.23%
d2_2-P	3295	2216.54	0.00%	d2_2-P	3295	1013.86	0.00%
d7_0-L	75230	3601.85	98.90%	d7_0-L	75230	3601.48	98.90%
d7_0-N	76405	3601.48	98.59%	d7_0-N	76405	3600.75	98.59%
d7_0-S	MBC	3178.83	53.20%	d7_0-S	MBC	1431.52	40.80%
d7_1-P	12480	3600.03	46.10%	d7_1-P	9130	3600.03	13.64%
d7_2-P	1220	1.29	0.00%	d7_2-P	1220	0.16	0.00%

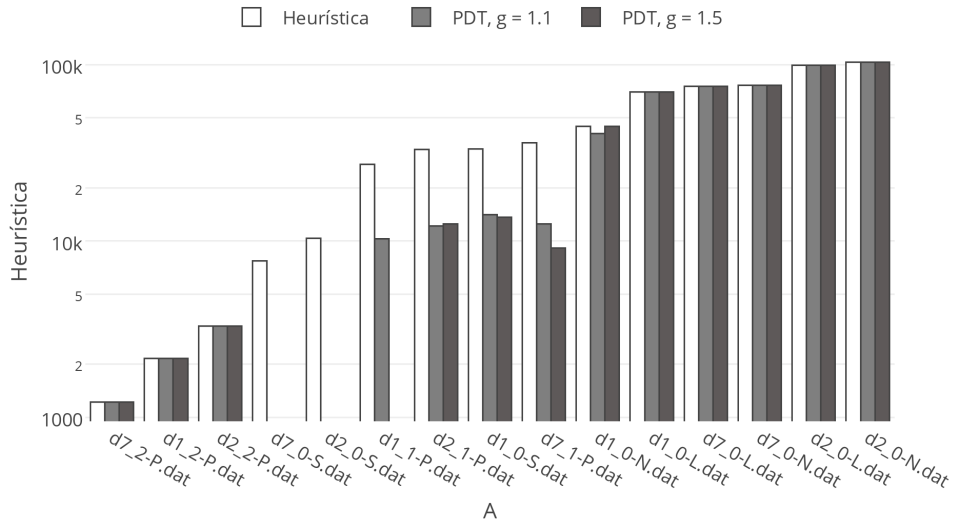
Fonte: Elaborada pela autora.

Podemos ver que apenas cinco instâncias tiveram problemas com a memória. Em

alguns casos o modelo conseguiu melhorar a solução mas ocorreu falha de memória durante a fase de *Branch and Cut*. O que aconteceu na maioria dos casos em que o GAP é entre 98% e 100% foi apenas uma redução no limitante inferior do problema. O único em que isso não é verdade é o da instância $d1\_0-N$, com $g = 1.1$, onde realmente houve uma redução no custo.

O gráfico na Figura 10 mostra os custos da soluções encontradas pela heurística comparado-as aos custos das soluções encontradas pelo modelo de PLI com a solução viável inicial. O gráfico ajuda a visualizar o quanto o modelo pode melhorar a solução da heurística. Pelo gráfico podemos ver que foram poucas as instâncias em que o modelo conseguiu melhorar a solução fornecida pela heurística.

Figura 10 – Gráfico comparando os custos para as tarefas encontradas pela heurística Constrói_Tarefas e pelo modelo PDT.



Fonte: A própria autora.

5.4.2 Modelo PAE

O modelo PAE, apresentado na Seção 4.2, também foi implementado em OPL. Os experimentos realizados com essa implementação são descritos e discutidos a seguir.

5.4.2.1 Primeiro Experimento com o modelo PAE - instâncias da heurística

Nesse experimento do PAE foram utilizados como entradas para o modelo PDT a solução dada pela heurística apresentada no Algoritmo 1. Os resultados são apresentados na

Tabela 6.

Tabela 6 – Resultados do primeiro experimento com o PAE - Utilizando as tarefas criadas pela heurística

g = 1.1				g = 1.5			
Instância	**Obj.**	**Tempo (s)**	**GAP (%)**	**Instância**	**Obj.**	**Tempo**	**GAP**
d1_0-L	MBC	779.04	1.07%	d1_0-L	MBC	1257.63	1.84%
d1_0-N	MBC	790.88	0.79%	d1_0-N	MBC	992.65	3.77%
d1_0-S	32150	0.35	0.00%	d1_0-S	32150	0.62	0.00%
d1_1-P	14440	3600.38	0.10%	d1_1-P	14440	29.51	0.00%
d1_2-P	2530	0	0.00%	d1_2-P	2530	0	0.00%
d2_0-L	MBC	1245.89	0.91%	d2_0-L	MBC	1922.65	1.36%
d2_0-N	MBC	3391.5	1.13%	d2_0-N	MBC	2385.32	2.43%
d2_0-S	4980	0.18	0.00%	d2_0-S	4980	0.14	0.00%
d2_1-P	MBC	1161.89	0.31%	d2_1-P	MBC	872.98	0.13%
d2_2-P	1470	0.1	0.00%	d2_2-P	1470	0.06	0.00%
d7_0-L	MBC	729.6	0.64%	d7_0-L	MBC	1156.69	0.79%
d7_0-N	MBC	1063.98	1.69%	d7_0-N	MBC	3165.72	3.37%
d7_0-S	3375	0.24	0.00%	d7_0-S	3375	0.18	0.00%
d7_1-P	27655	3600.25	0.08%	d7_1-P	MBC	846.12	0.21%
d7_2-P	820	0.01	0.00%	d7_2-P	820	0.04	0.00%

Fonte: Elaborada pela autora.

Os resultados desse experimento são bem melhores comparados aos resultados do PDT. Isso acontece por que o problema de alocação fica mais simples que o problema de criar as tarefas. A entrada é menor, pois as viagens já são recebidas agrupadas em tarefas. Além disso a separação em pegadas reduz o espaço de busca. Diferentemente do PDT, que em uma única tarefa pode haver um número variado de viagens no PAE, em cada pegada só pode haver uma tarefa alocada a ela.

Ainda aconteceu em muitos casos deste experimento problemas com a memória. Todos os casos de solução viável não encontrada foram provocadas por este problema. Como essas instâncias apresentaram um GAP muito baixo, novos testes com o parâmetro GAP mínimo de 2%. Esse valor foi escolhido após observarmos os resultados da Tabela 6. Os resultados dessa alteração são apresentados na Tabela 7.

Depois de adicionada a condição de parar a busca quando o GAP atingido for menor ou igual a 2%, todas as instâncias com o parâmetro $g = 1.1$ retornaram uma solução. Entretanto em dois casos as soluções perderam o status de ótimo. Na primeira, $d1\_0 - S$, a solução nesse experimento foi maior que a anterior, o que significa que a busca ainda poderia ter sido melhorada mais um pouco mais antes de ser interrompida. Já no segundo caso, $d7\_0 - S$, a solução foi a mesma e o solver não conseguiu provar no passo seguinte que a solução era de fato ótima.

A Figura 11 mostra uma comparação visual do impacto de parar a busca quando um

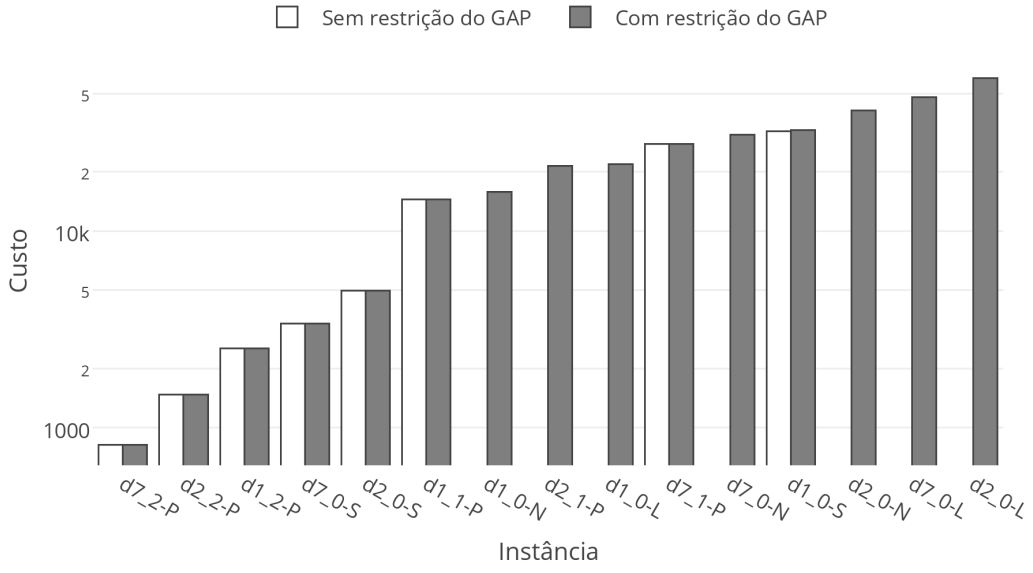
Tabela 7 – Resultados do segundo experimento com o PAE - Encerrando a busca quando o GAP é menor ou igual a 2%

g = 1.1				g = 1.1			
Instância	**Obj.**	**Tempo**	**GAP**	**Instância**	**Obj.**	**Tempo**	**GAP**
d1_0-L	21855	138.4	$\leq$ 2.00%	d1_0-L	21630	499.16	$\leq$ 2.00%
d1_0-N	15865	108.63	$\leq$ 2.00%	d1_0-N	MBC	983.3	3.77%
d1_0-S	32675	0.2	$\leq$ 2.00%	d1_0-S	32150	0.63	0.00%
d1_1-P	14440	3.46	$\leq$ 2.00%	d1_1-P	14440	6.15	$\leq$ 2.00%
d1_2-P	2530	0	0.00%	d1_2-P	2530	0	0.00%
d2_0-L	59990	403.01	$\leq$ 2.00%	d2_0-L	59215	1019.1	$\leq$ 2.00%
d2_0-N	40975	570.03	$\leq$ 2.00%	d2_0-N	MBC	2374.91	2.43%
d2_0-S	4980	0.05	0.00%	d2_0-S	4980	0.15	0.00%
d2_1-P	21500	6.73	$\leq$ 2.00%	d2_1-P	21500	17.92	$\leq$ 2.00%
d2_2-P	1470	0.05	0.00%	d2_2-P	1470	0.02	0.00%
d7_0-L	48105	87.53	$\leq$ 2.00%	d7_0-L	46105	172.76	$\leq$ 2.00%
d7_0-N	30905	130.11	$\leq$ 2.00%	d7_0-N	MBC	3014.03	3.37%
d7_0-S	3375	0.1	$\leq$ 2.00%	d7_0-S	3375	0.27	0.00%
d7_1-P	27655	4.1	$\leq$ 2.00%	d7_1-P	27655	9.67	$\leq$ 2.00%
d7_2-P	820	0	0.00%	d7_2-P	820	0	0.00%

Fonte: Elaborada pela autora.

GAP de 2% for atingido, com o parâmetro $g = 1.1$. O aumento no custo é quase imperceptível.

Figura 11 – Gráfico de comparação entre os custos dos experimentos do PAE com tarefas da heurística.



Fonte: Elaborada pela autora.

5.4.2.2 Segundo Experimento com o modelo PAE - instâncias do PDT

Neste experimento foi realizada a criação das instâncias para o modelo PAE com os valores retornados pelo experimento do PDT, com $g = 1.1$ e solução viável inicial da heurística Constrói_Tarefas, reportados na Seção 5.4.1.4. Esses resultados foram escolhidos por serem os melhores.

Na Tabela 8 podemos ver os resultados desses experimentos. Nesta tabela as instâncias cujo nome estão em negrito indicam que a solução do PDT encontrado pelo modelo de PLI é ótima.

Podemos ver que todas as instâncias com a solução ótima para o PDT tiveram solução ótima para o PAE. Vale lembrar que os valores para o horário de término da jornada do dia anterior dos motoristas não é o mesmo que os do experimento anterior. O que pode explicar o fato de as outras instâncias não terem encontrado uma solução com GAP menor ou igual a 2% como nos casos anteriores.

Tabela 8 – Resultados do terceiro experimento com o PAE - Tarefas criadas pelo modelo PDT

g = **1.1**				g = **1.5**			
Instâncias	**Obj.**	**Tempo**	**GAP**	**Instâncias**	**Obj.**	**Tempo**	**GAP**
d1_0-L	24380	473.78	$\leq 2.00\%$	d1_0-L	MBC	1103.72	90.76%
d1_0-N	21065	30.57	$\leq 2.00\%$	d1_0-N	19840	114.42	$\leq 2.00\%$
d1_0-S	10150	0.07	0.00%	d1_0-S	10150	0.1	0.00%
d1_1-P	MBC	934.17	12.27%	d1_1-P	MBC	957.99	13.57%
d1_2-P	2530	0.06	0.00%	**d1_2-P**	2530	0.03	0.00%
d2_0-L	61615	371.09	$\leq 2.00\%$	d2_0-L	60040	512.9	$\leq 2.00\%$
d2_0-N	MBC	745.2	16.87%	d2_0-N	48000	3612.84	$\leq 2.00\%$
d2_1-P	MBC	652.15	10.32%	d2_1-P	MBC	630.75	10.66%
d2_2-P	1470	0.1	0.00%	**d2_2-P**	1470	0.05	0.00%
d7_0-L	50255	3600.03	$\leq 2.00\%$	d7_0-L	47630	159.12	$\leq 2.00\%$
d7_0-N	MBC	892.23	37.45%	d7_0-N	MBC	1931.87	$\leq 2.00\%$
d7_1-P	MBC	961.72	11.42%	d7_1-P	MBC	1084.46	16.75%
d7_2-P	820	0.05	0.00%	**d7_2-P**	820	0.04	0.00%

Fonte: Elaborada pela autora.

Podemos ver também que houveram muitos casos com estouro de memória. O que não era esperado, já que isso não aconteceu com os experimentos com heurística e os valores das soluções são no máximo a mesma que as da heurística. A provável causa desse fenômeno são os valores diferentes para as informações dos motoristas.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foram apresentadas mudanças para o modelo de Nunes (2015), com o objetivo de torná-lo mais realista. Foram feitos experimentos com o modelo de Nunes (2015) removendo a restrição que determinava que o motorista só podia trabalhar em uma tabela. Percebemos que sem essa restrição, o modelo retornava alocações que utilizavam um número menor de motoristas quando comparados ao modelo original. Entretanto, o modelo ainda não refletia a realidade das empresas de alocar motoristas em quaisquer linhas. Os motoristas podiam apenas serem alocados em mais de uma tabela de uma mesma linha. Também percebemos que com essa modificação, o modelo não tinha controle sobre os intervalo entre a execução de tarefas. Pensando nisso, dividimos o modelo de Nunes (2015) em dois, uma parte para resolver o PDT e a outra o PAE, corrigindo os pontos destacados anteriormente.

O modelo que trata do PDT tem uma carga maior de dificuldade, devido o número de viagens que impacta no número de decisões que devem ser tomadas pelo modelo. Foram feitos vários pré-processamentos para que se pudesse pelo menos carregar o modelo em memória, ainda assim não foi possível encontrar o solução dentro do limite de tempo estabelecido para todas as instâncias. Foi desenvolvida uma heurística para reduzir o tamanho das instâncias e foram obtidos resultados melhores para o PDT. Mesmo adicionando uma solução viável inicial ao modelo PDT não obtivemos resultados tão bons quanto o esperado, pois o modelo não conseguiu melhorar a solução dada pela heurística para a maioria das instâncias. O que nos leva a acreditar que uma abordagem aproximativa seja o melhor caminho para resolver o problema.

Com a criação de turnos sendo tratada por um modelo separado, a alocação feita pelo PAE fica muito mais eficiente de ser realizada. O modelo PAE encontrou soluções ótimas ou muito próximas das soluções ótimas para quase todas as instâncias e em um período de tempo curto, pois nenhuma chegou a ser interrompida pelo *timeout*. Para resolver os problemas de estouro de memória de algumas instâncias seria ideal adicionar mais desigualdades válidas ao problema e realizar experimentos em uma máquina com mais memória.

Como trabalhos futuros sugerimos uma combinação de um modelo de Programação Lógica por Restrições (PLR) para o Problema de Determinação de Tarefas e um modelo em Programação Linear Inteira para o Problema de Alocação de Escalas apresentado na Seção 4.2. Combinar PLI e PLR é uma prática bastante comum, utilizando-se as vantagens de cada uma das abordagens para se encontrar uma solução otimizada. Outro caminho que também pode ser tomado em um trabalho futuro seria modelar o modelo PDT da Seção 4.1 em PLR para fornecer

soluções viáveis iniciais melhores ao modelo e estudar o seu desempenho após essa inserção. Além de realizar testes com ambos modelos em uma máquina que tenha mais memória.

REFERÊNCIAS

BERTSIMAS, D.; TSITSIKLIS, J. **Introduction to Linear Optimization**. [S.l.]: Athena Scientific, 1997. (Athena Scientific series in optimization and neural computation). ISBN 9781886529199.

BOAS, M. G. V.; SILVA, G. P. Modelos de programação inteira para o problema do rodízio de tripulações de ônibus urbano. **Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional**, p. 1–12, 2014.

BRASIL. **Lei nº 13.103, de 2 de março de 2015**. DF: Diário Oficial [da] União, Brasília, 2015. Acesso em: 4 out. de 2016. Disponível em: <http://www.jusbrasil.com.br/diarios/DOU/2012/01/04>.

CEARÁ. **Convenção coletiva de trabalho 2016/2017**. Fortaleza: Ministério do Trabalho e Emprego, 2016. Acesso em: 23 jun. 2017. Disponível em: <http://www3.mte.gov.br/sistemas/mediador/Resumo/ResumoVisualizar?NrSolicitacao=MR047688/2016>.

CORMEN, T.; LEISERSON, C.; RIVEST, R.; STEIN, C. **Introduction To Algorithms**. MIT Press, 2001. ISBN 9780262032933. Disponível em: <https://books.google.com.br/books?id=NLngYyWFl_YC>.

CPLEX, I. I. 12.1 user's manual for cplex. In: . [S.l.]: International Business Machines Corporation, 46(53), 157, 2009.

FORTALEZA. **Mostra Virtual Fortaleza 2040: Fortaleza Hoje**. Fortaleza: Instituto de Planejamento de Fortaleza, 2016. 100 p. Acesso em: 4 out. de 2016. Disponível em: <<http://fortaleza2040.fortaleza.ce.gov.br/site/assets/revistas/i-mostra-virtual/index.html#page/1>>.

GOMES, W. d. P.; GUALDA, N. D. F. Modelagem integrada do problema de programação de tripulantes de aeronaves. **TRANSPORTES**, v. 19, n. 1, p. 23–32, 2011.

HILLIER, F.; LIEBERMAN, G. **Introdução à Pesquisa Operacional**. [S.l.]: AMGH, 2013. ISBN 9788580551198.

HOLLORAN, T. J.; BYRN, J. E. United airlines station manpower planning system. **Interfaces**, INFORMS, v. 16, n. 1, p. 39–50, 1986.

MARINHO, E. H.; OCHI, L. S.; DRUMMOND, L. M. A.; SOUZA, M. J. F.; SILVA, G. P. Busca tabu aplicada ao problema de programação de tripulações de ônibus urbano. **Anais do XXXVI Simpósio Brasileiro de Pesquisa Operacional**, p. 1471–1482, 2004.

NUNES, R. de P. **Programação de tripulação no transporte de ônibus urbano: Uma abordagem utilizando programação linear inteira**. 102 f. Dissertação (Mestrado) — Universidade de Fortaleza, Fortaleza, 2015.

PAPADIMITRIOU, C.; STEIGLITZ, K. **Combinatorial Optimization: Algorithms and Complexity**. [S.l.]: Dover Publications, 2013. (Dover Books on Computer Science). ISBN 9780486320137.

SOUZA, I. E. D. S. D. **Programação de tripulação no transporte de ônibus urbano: uma abordagem utilizando programação por restrições**. Quixadá: [s.n.], 2016.

SOUZA, M. J. F.; CARDOSO, L. X. T.; SILVA, G.; RODRIGUES, M. M. S.; MAPA, S. M. S. Metaheurísticas aplicadas ao problema de programação de tripulações no sistema de transporte público. **Trends in Applied and Computational Mathematics**, v. 5, n. 2, p. 357–368, 2004.

TURBAN, E.; SHARDA, R.; DELEN, D. **Decision Support and Business Intelligence Systems**. [S.l.]: Prentice Hall, 2011. ISBN 9780136107293.

WOLSEY, L. **Integer Programming**. [S.l.]: Wiley, 1998. (Wiley Series in Discrete Mathematics and Optimization). ISBN 9780471283669.

WREN, A.; ROUSSEAU, J.-M. Bus driver scheduling—an overview. In: **Computer-aided transit scheduling**. [S.l.]: Springer, 1995. p. 173–187.