UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

ITALOS ESTILON DA SILVA DE SOUZA

PROGRAMAÇÃO DE TRIPULAÇÃO NO TRANSPORTE DE ÔNIBUS URBANO: UMA ABORDAGEM UTILIZANDO PROGRAMAÇÃO POR RESTRIÇÕES

QUIXADÁ – CEARÁ

2016

ITALOS ESTILON DA SILVA DE SOUZA

PROGRAMAÇÃO DE TRIPULAÇÃO NO TRANSPORTE DE ÔNIBUS URBANO: UMA ABORDAGEM UTILIZANDO PROGRAMAÇÃO POR RESTRIÇÕES

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Orientador: Prof. Dr. Críston Pereira de Souza

QUIXADÁ – CEARÁ

2016

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)

D32p de Souza, Italos Estilon da Silva.
Programação de Tripulação no Transporte de Ônibus Urbano: Uma Abordagem Utilizando Programação por Restrições / Italos Estilon da Silva de Souza. – 2016.
47 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Dr. Críston Pereira de Souza.

1. PROGRAMAÇÃO POR RESTRIÇÕES. 2. TRANSPORTE PÚBLICO. 3. ALOCAÇÃO DE RECURSOS. I. Título.

CDD 005.1

ITALOS ESTILON DA SILVA DE SOUZA

PROGRAMAÇÃO DE TRIPULAÇÃO NO TRANSPORTE DE ÔNIBUS URBANO: UMA ABORDAGEM UTILIZANDO PROGRAMAÇÃO POR RESTRIÇÕES

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

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

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

Prof. MSc. Lucas Ismaily Bezerra Freitas
Campus Quixadá
Universidade Federal do Ceará - UFC

A todos que lerem.

AGRADECIMENTOS

Ao Prof. Dr. Críston Pereira de Souza, pela sugestão do tema e pela excelente orientação.
Aos professores participantes da banca examinadora Prof. Dr. Paulo de Tarso Guerra e Prof. Msc. Lucas Ismaily Bezerra Freitas pelo tempo, pelas valiosas colaborações e sugestões. Suas sugestões foram muito importante para o trabalho e para o meu amadurecimento.
Aos meus colegas de turma, pela troca de experiência e apoio.
Aos meus pais, pelo apoio em todos os sentidos.

"There's nothing glorious in dying. Anyone can do it."

(Johnny Rotten)

RESUMO

Este trabalho define um modelo em Programação por Restrições para o Problema de Programação de Tripulação com o objetivo de produzir soluções viáveis para este problema. A definição do problema é baseada em (NUNES; NEPOMUCENO, 2015), mas uma restrição foi removida. Essa restrição, faz parte do modelo de Nunes e Nepomuceno (2015) para melhorar o tempo de execução e tornar possível encontrar soluções, mas soluções melhores podem ser encontradas sem essa restrição. O modelo foi implementado utilizando o Gecode e foram utilizados dados reais como entrada para analisar a capacidade do modelo de encontrar soluções viáveis. O trabalho analisa ainda como diferentes configurações do Gecode para realizar a busca pode influenciar na qualidade da solução.

Palavras-chave: Programação por Restrições. Problema de Programação de Tripulação. Gecode. Transporte Público.

ABSTRACT

This paper defines a model in Constraint Programming to the Crew Scheduling Problem with the objective of producing feasible solutions to this problem. The problem definition is based on (NUNES; NEPOMUCENO, 2015), but a constraint has been removed. This constraint is part of the Nunes e Nepomuceno (2015)'s model to improve execution time and make it possible to find solutions, but better solutions can be found without this constraint. The model was implemented using Gecode and real data was used as input to analyze the model's ability to find feasible solutions. The paper also analyzes how different configurations of the Gecode to perform the search can influence the quality of the solutions.

Keywords: Constraint Programming. Crew Scheduling Problem. Gecode. Public Transportation

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

PPT	Problema de Programação de Tripulação
PR	Programação por Restrições
PLI	Programação Linear Inteira

SUMÁRIO

1 INTRODUÇÃO

O serviço de transporte público é um importante meio de locomoção de pessoas em centros urbanos. Financeiramente, serviços de transporte públicos são mais atrativos que o uso individual de outros meios, como o carro. Dentre os serviços públicos o ônibus é o mais comum, sendo que apenas oito dos grandes centros urbanos do país contam com rede metroviárias (NUNES; NEPOMUCENO, 2015).

A privatização de bens e instituições públicas podem ser utilizadas pelo o governo para transferir gastos e atrair investimentos para determinados setores. Instituições privadas buscam maximizar seus lucros e, para isso, devem investir em redução de custo. A privatização das empresas que administram os serviços de transporte público tom que esse investimento aconteça. Dentre todos os custos operacionais, o custo com mão de obra representa a maior parcela (SILVA; SOUZA; ATZINGEN, 2006).

O planejamento das operações dessas empresas é complexo e compreende quatro etapas (NUNES; NEPOMUCENO, 2015):

- definição das linhas de operação;
- definição das tabelas de horários das linhas de operação;
- definição da programação dos veículos que serão utilizados;
- alocação dos motoristas para os veículos.

A última etapa é relevante para a empresa pois trata da escala de trabalho dos motoristas. O processo de definir essa escala de trabalho é conhecido na literatura como o Problema de Programação de Tripulação (PPT). O PPT consiste em alocar jornadas de trabalho pré-definidas para um conjunto de tripulações de modo que todas as jornadas sejam atendidas (NUNES; NEPOMUCENO, 2015). Ele visa reduzir os custos, respeitando legislações trabalhistas e culturas empresariais. Essas características tornam o PPT um problema pertencente à classe NP-Difícil (CARRARESI; GALLO, 1984) (MARTELLO; TOTH, 1986).

Para minimizar o número de motoristas na escala e agilizar o processo de alocação, Nunes e Nepomuceno (2015) propõem um modelo em Programação Linear Inteira para resolver o PPT para uma empresa de transporte público da cidade de Fortaleza. Os autores obtiveram sucesso para um determinado conjunto de restrições e para um tipo de escopo em que um motorista só pode ser alocado para uma tabela em um dado dia.

São ditos problemas de Programação Linear Inteira (PLI) aqueles problemas de otimização que podem ser definidos como maximizar ou minimizar uma função que está sujeita

a um conjunto de restrições lineares e todas ou parte das variáveis são inteiras. Essa classe de problemas é menor que as que podem ser modelados com Programação por Restrições.

Programação por Restrições (PR) é um paradigma declarativo que se utiliza da definição de restrições entre variáveis para descrever computações. Para se resolver um problema utilizando tal paradigma é preciso especificá-lo em termo de variáveis e restrições que limitam os possíveis valores que estas podem assumir. Computações passíveis de serem modeladas como um Problema de Satisfação de Restrições podem ser realizadas utilizando esse paradigma.

O objetivo desse trabalho é propor um modelo em Programação por Restrições capaz de produzir soluções viáveis para o PPT utilizando instâncias da empresa mencionada acima removendo-se uma restrição da definição do problema definido por Nunes e Nepomuceno (2015).

Neste trabalho visa-se produzir apenas soluções viáveis e não soluções ótimas. Soluções viáveis podem ser utilizadas, por exemplo, em uma abordagem utilizando Programação Linear Inteira para reduzir o espaço de busca a ser considerado para se encontrar soluções ótimas. Abordagens híbridas de Programação por Restrições e Programação Linear Inteira podem ser encontradas na literatura como em Yunes, Moura e Souza (2000).

O restante desse trabalho está estruturado da seguinte forma: no Capítulo 2 estão descritos os conceitos relevantes para o entendimento deste trabalho; no Capítulo 3 são apresentados dois trabalhos que tratam de problemas semelhantes ao deste trabalho; em seguida, são detalhadas as etapas da metodologia 4; os resultados atingidos estão descritos no Capítulo 5 e são discutidos no Capítulo **??**; considerações sobre todo o trabalho são feitas no Capítulo 6 e por fim tem-se as referências utilizadas por este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções que seguem, são apresentados os conceitos relevantes para esse trabalho. Na Seção 2.1 é apresentada a técnica utilizada para resolver o PPT neste trabalho. Na Seção 2.2 é apresentada a técnica utilizada nos dois trabalhos relacionados. O entendimento desse conceito é relevante para compreender as restrições descritas por Nunes e Nepomuceno (2015) assim como a solução por ele proposta. Na Seção 2.3 é descrito o problema que este trabalho busca resolver.

Programação por Restrições

Programação por Restrições é um paradigma de programação declarativo que define restrições para relacionar variáveis. A definição de Programação por Restrições é tão ampla que engloba outras áreas como Programação Linear e Inteira, Otimização Global e Álgebra Linear (APT, 2003). **Restrição**, o conceito chave nesse paradigma, é a relação entre os domínios de um conjunto de variáveis sobre as quais a restrição está definida (APT, 2003). Uma restrição pode ser vista como um requisito que determina quais combinações de valores as variáveis podem assumir (APT, 2003). Assim, as restrições reduzem os domínios das variáveis.

Para que um problema seja resolvido utilizando essa técnica, é preciso modelá-lo como o **Problema de Satisfação de Restrições** (PSR).

Definição 2.1.1. O Problema de Satisfação de Restrições consiste de um conjunto de variáveis $X = \{x_0, x_1, x_2, \ldots, x_n\}$; um conjunto de domínios finitos $D = \{d_0, d_1, d_2, \ldots, d_n\}$ com os possíveis valores que as variáveis de X podem assumir; e um conjunto de restrições $C = \{c_0, c_1, c_2, \ldots, c_m\}$.

As restrições de C restringem os valores que as variáveis de X podem assumir simultaneamente. Ou seja, $c_i(x_{i_1}, x_{i_2}, \ldots, x_{i_k}) \subseteq d_{i_1} \times d_{i_2} \times \ldots \times d_{i_k}$, onde c_i é i-ésima restrição sobre k variáveis de X.

O PSR pode ser representado como um grafo, ou rede de restrições, onde um vértice corresponde a uma variável e uma aresta a uma restrição binária. Um hiper grafo pode ser utilizado para representar restrições com aridade maior que dois, mas também é possível representá-las em termos de restrições binárias (BARTÁK, 1998) e dessa forma utilizar um grafo.

Uma solução para o PSR consiste em atribuir para cada variável x_i um valor de d_i de

forma que satisfaça a todas as restrições (NIEDERLIŃSKI, 2009). A solução pode ou não ser única. Para soluções ótimas é necessário definir uma função objetivo em termos de uma ou mais variáveis do problema.

Para compreender melhor as características do PSR, segue um pequeno exemplo retirado de em Niederliński (2009).

Exemplo 2.1.1. Seja $X = \{x_0, x_1\}$, $D = \{d_0, d_1\}$ e C = $\{c_0\}$, sendo $d_0 = \{0, 1, 2, ..., 10\}$, d_1 = {2,3,4,5} e c_0 é uma relação binária definida por $x_0 + 2 \cdot x_1 \leq 8$. A solução para esse problema seria $x_0 = \{0, 1, 2, 3, 4\}$ e $x_1 = \{2, 3, 4\}$. O valor 5 é removido de d_1 porque $2 \cdot 5 > 8$. Os valor 9 e 10 são removidos de d_0 porque ambos são maiores que 8. Os outros valores, 6, 7, 8, 9 são removidos porque $\forall x \in \{6, 7, 8, 9\}$ e $\forall y \in d_1$, $x + 2 \cdot y > 8$. Para os valores que sobraram em d_0 e d_1 vale as seguintes propriedades: $\forall x \in d_0, \exists y \in d_1$ tal que $x + 2 \cdot y \leq 8$ e $\forall y \in d_1, \exists x \in d_0$ tal que $x + 2 \cdot y \leq 8$.

Note que é possível encontrar uma solução se ela existir ou determinar se não existe, dado que os domínios são finitos (NIEDERLIŃSKI, 2009). Um algoritmo para resolver o PSR, assumindo-se que seja dado tempo suficiente, poderia enumerar todas as possíveis atribuições de valores e verificar quais delas respeitam todas as restrições. Sendo assim, podemos dizer que o PSR é um problema decidível.

Existem formas mais elaboradas de resolver o PSR, dado que encontrar soluções da forma dita acima pode ser impraticável. Segundo Niederliński (2009), um problema com 30 variáveis com domínios de tamanho 100 levaria 10^{54} anos para analisar todas as possíveis atribuições de valores para as variáveis.

Rossi, Beek e Walsh (2006) descrevem duas categorias de estratégias para resolver o PSR: inferência e busca. As duas abordagens podem ser combinadas. As técnicas de inferência podem utilizar propagação de restrições locais para reduzir o espaço de soluções possíveis, enquanto que, a busca explora o espaço de soluções possíveis e elimina sub-espaços com falha (ROSSI; BEEK; WALSH, 2006).

Uma abordagem descrita por Rossi, Beek e Walsh (2006) é o *backtracking*, um método de busca completa, ou seja, garante encontrar uma solução se ela existir. O *backtracking* de forma incremental tenta expandir uma solução parcial, onde algumas variáveis estão atribuídas, em uma solução completa repetidamente atribuindo às variáveis, ainda não atribuídas, valores consistentes com os que já estão na solução parcial. Esse tipo de busca pode ser combinado com

as técnicas de consistência, métodos de inferência, que identificam inconsistências e reduzem os casos em que o algorítimo de busca chega a um estado de falha e precisa desfazer suas escolhas.

As técnicas de consistência realizam propagação de restrições, que é o procedimento de explicitamente eliminar valores ou combinações de valores dos domínios de algumas variáveis que, de outra forma, um subconjunto das restrições não seriam satisfeitas (ROSSI; BEEK; WALSH, 2006). A propagação das restrições é realizado através de algorítimos de consistências. Esses algorítimos são de tempo polinomial e seu esforço depende do nível de consistência e do tamanho do subconjunto de variáveis do contexto local (ROSSI; BEEK; WALSH, 2006). Aqui serão mencionados os três níveis de consistência: de nó, de arco e de caminho.

Uma variável $x_i \in X$, x_i é **nó-consistente** se e somente se cada valor de $d_i \in D$ satisfaz todas as restrições unárias em C sobre x_i. Para se alcançar a consistência desse nó basta remover todos os valores que não satisfazem a todas as restrições unárias sobre x_i. Se todos os nós forem consistentes, então, o grafo é dito **nó-consistente** (ROSSI; BEEK; WALSH, 2006), e se o for, todas as restrições unárias foram satisfeitas (BARTÁK, 1998).

A consistência de arco é outra técnica que consegue reduzir mais ainda os domínios das variáveis (ROSSI; BEEK; WALSH, 2006). Se uma restrição $c_k \in C$ está definida sobre duas variáveis x_i e $x_j \in X$, dizemos que o arco (x_i, x_j) é consistente se e somente se, para cada valor em d_i existe um valor em d_j tal que c_k sobre x_i e x_j é satisfeita. Para tornar o arco (x_i, x_j) consistente, basta remover de d_i todos os valores que não possuem um valor em d_j que satisfaz c_k (ROSSI; BEEK; WALSH, 2006). O conceito de consistência de arco é direcional e, portanto, se o arco (x_i, x_j) é consistente não garante que o arco (x_j, x_i) seja (BARTÁK, 1998) (ROSSI; BEEK; WALSH, 2006). O grafo é dito **arco-consistente** se todos os arcos forem consistentes.

Um caminho entre x_i e x_k passando por x_j é dito consistente se e somente se para cada tupla de valores de d_i e d_j que satisfaz todas as restrições sobre x_i e x_j existe um valor em d_k que satisfaz todas as restrições sobre x_i, x_j e x_k. Um grafo é dito caminho-consistente se todos os caminhos de tamanho 2 forem consistentes (ROSSI; BEEK; WALSH, 2006) (BARTÁK, 1998). A garantia de consistência de caminhos de tamanho 2 garante a consistência de caminhos de qualquer tamanho (MONTANARI, 1974).

Neste trabalho, está sendo usado o Gecode, que utiliza *backtraking* e propagação de restrições. Entender como o *solver* trabalha para encontrar uma solução para o PSR é útil para que sejam escolhidos níveis de consistências adequados a cada restrição e assim o desempenho não seja impactado.

Programação Linear Inteira

Um problema é tido como um problema de Programação Linear se ele pode ser descrito em termos de minimizar ou maximizar uma função linear sujeita a um conjunto finito de restrições lineares (CORMEN, 2009). Formalmente, podemos definir um Problema de Programação Linear da seguinte forma:

- Seja $a_0, a_1, \ldots, a_n$ um conjunto de números reais e $x_0, x_1, \ldots, x_n$ um conjunto de variáveis reais.
- Seja $f(x_0, x_1, \ldots, x_n) = a_0 \cdot x_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n = \sum_{i=0}^{n} a_i \cdot x_i$. Dizemos que f é uma função linear.
- $f(x_0, x_1, \ldots, x_n) \leq b$ é uma desigualdade linear, onde b é um número real.
- $f(x_0, x_1, \ldots, x_n) \geq b$ é uma desigualdade linear, onde b é um número real.
- $f(x_0, x_1, \ldots, x_n) = b$ é uma igualdade linear, onde b é um número real.

O termo restrição linear é utilizado para denotar igualdades e desigualdades lineares.

Na Programação Linear, as variáveis de decisão podem ser reais. No entanto, existem problemas onde é necessário que ao menos uma das variáveis de decisão seja inteira, esse tipo de problema é chamado de Problema de Programação Inteira.

$$\sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i \quad (i = 1, 2, \ldots, n) \tag{2.2}$$

$$x_j \geq 0 \quad (j = 1, 2, \ldots, n) \tag{2.3}$$

$$x_j \text{ inteiro} \quad (\text{para algum } j = 1, 2, \ldots, n) \tag{2.4}$$

Na formulação definida pelas Equações 2.1, 2.2, 2.3 e 2.4, a Equação 2.4 garante a propriedade necessária para o problema ser considerado de Programação Inteira. Chamamos um programa de inteiro misto quando alguma variável de decisão deve ser inteira, mas não todas, e dizemos que é puro quando todas as variáveis de decisão são inteiras.

Problema de Programação de Tripulação

Nesta seção está descrito o problema para o qual este trabalho busca encontrar soluções viáveis.

Definição 2.3.1. O Problema de Programação de Tripulação (PPT) consiste em atribuir um conjunto de tarefas para um conjunto de tripulações (PORTUGAL; LOURENÇO; PAIXÃO, 2009).

O problema recebe como entrada as leis trabalhistas, regras da empresa e as tarefas a serem executadas durante o período do planejamento (MAYRINK; SILVA, 2012). As tarefas podem ser consideradas diferentes dependendo do tipo do dia, se é útil, sábado ou domingo. Todas as tarefas devem ser atendidas, mas nem todas as tripulações precisam estar alocadas para alguma tarefa. Nesse trabalho, as tripulações serão motoristas e as tarefas conjuntos de viagens em uma linha de trânsito.

As empresas de ônibus recebem, dos órgãos públicos, a definição de quais linhas devem operar, quais os horários e um conjunto de regras relacionadas ao trabalho dos motoristas. O horário de entrada e saída de cada linha, os horários para pausa dos motoristas e o trajeto dos ônibus são exemplos de definições que as empresas devem seguir sob pena de perderem o direito de atuar nas linhas (NUNES; NEPOMUCENO, 2015).

Esse trabalho tem como objetivo resolver o problema definido em Nunes e Nepomuceno (2015) sem uma das restrições da definição original. Para melhor entendimento, serão apresentados aqui alguns termos utilizados e definidos em Nunes e Nepomuceno (2015):

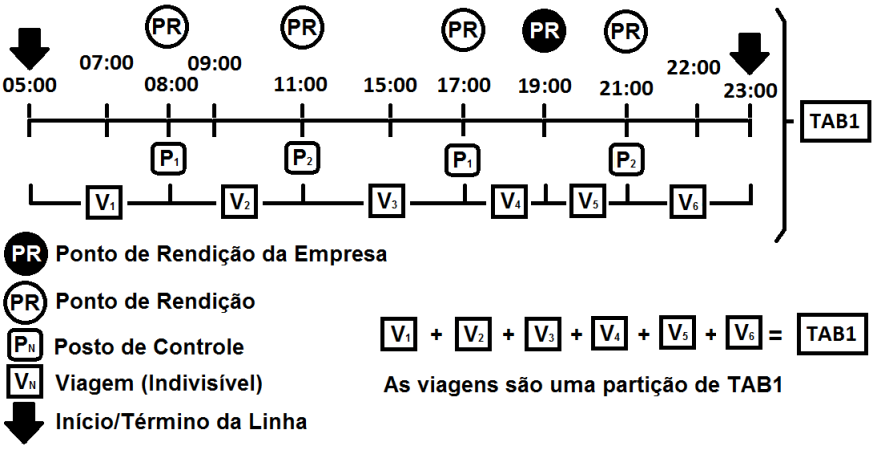
1. **Linha** - Define um trajeto que deve ser seguido pelos veículos de uma empresa;
2. **Tabela** - Define um veículo está atuando em uma determinada linha;
3. **Tipo de Dia** - Indica se o dia é útil, sábado ou domingo. Diferentes tipos de dias podem conter diferentes quantidades de tabelas;
4. **Horário** - Indica um detalhe operacional de uma tabela, como o horário que um ônibus entra na linha ou quando passa em um posto de controle;
5. **Viagem** - Representa o tempo decorrido entre um dois horários.

As tabelas de horários e os horários são suficientes para que um veículo execute corretamente seu trajeto dentro de uma linha.

A Figura 1 apresenta os detalhes para uma tabela de horários denominada “TAB1”. Nela, pode-se ver que as viagens são delimitas pelos pontos de rendição (PR). Existem dois

pontos de controle, P_1 e P_2, com pontos de rendição associados a eles, e um terceiro ponto de rendição estabelecido pela empresa. Esses pontos de rendição definem as seis viagens que, juntas, correspondem a todo o horário da tabela.

Figura 1 – Postos de controle, pontos de rendição e viagens da tabela de horários TAB1.



Fonte – Nunes (2015)

Comumente, abordagens para esse problema buscam soluções que minimizem o número de motoristas ou custo operacional em dinheiro. Esse custos podem ser com horas extras, pessoal adicional ou horas não aproveitadas. Nesse trabalho, o foco é encontrar apenas soluções viáveis que são aquelas que respeitam todas as restrições do problema.

Restrições do Problema

O problema que este trabalho se propõem a resolver é uma instância do Problema de Programação de Tripulação. Ele consiste em definir uma escala de trabalho para motoristas dado um horizonte de tempo determinado pela quantidade de semanas da escala (PORTUGAL; LOURENÇO; PAIXÃO, 2009). A definição da escala deve seguir as restrições da legislação e da cultura da empresa. Abaixo, estão as restrições que serão consideradas neste caso, as quais são definidas por Nunes e Nepomuceno (2015), com exceção da restrição 10.

1. Sejam f e g duas tarefas em dias consecutivos. Sejam s_k e e_k os instantes de início e fim em minutos, respectivamente, de uma tarefa k qualquer. Se $s_g - e_f < 11 \cdot 60$ então f e g devem ser alocadas para motoristas diferentes. Essa restrição garante o descanso mínimo de onze horas entre jornadas.
2. O descanso mínimo entre jornadas é contínuo, ou seja, deve-se considerar a última jornada dos motoristas na escala anterior. Assim, sejam t_i e f a última

tarefa do motorista i na escala anterior e uma tarefa qualquer do primeiro dia da nova escala, respectivamente. Se $s_f - e_{t_i} < 11 \cdot 60$ então o motorista i não poderá ser atribuído para a tarefa f.

3. A jornada de trabalho diária não será maior que o máximo dado como entrada. Seja T_{ij} o conjunto das tarefas alocadas para o motorista i em um dia j, d_k a duração da tarefa k e QDH a quantidade limite de horas diárias que um motorista pode trabalhar, então $\sum_{t \in T_{ij}} d_t \leq \text{QHD}$.
4. A jornada de trabalho será maior ou igual ao mínimo dado como entrada.
5. A quantidade de horas trabalhadas semanalmente por um motorista deve ser igual ao máximo permitido para a jornada semanal. As hora extras e horas não aproveitadas são levadas em consideração. Se o motoristas fez hora extra, o limite máximo será somado com a quantidade de horas extras, e se ele teve horas não aproveitadas, o limite é diminuído. Horas extras anulam horas não aproveitas e vice versa.
6. A quantidade de horas extras dadas pela empresa deve ser menor ou igual ao máximo dado como entrada.
7. Nenhum motorista deve trabalhar mais que a quantidade limite de dias sem folga. Esse limite é dado como entrada. A última folga da escala anterior deve ser considerada.
8. Nenhum motorista deve trabalhar mais que a quantidade limite de domingos sem folga. Esse limite é dado como entrada e a última folga em domingo na escala anterior deve ser considerada.
9. Cada tarefa deve ser atendida por um único motorista.

(NUNES, 2015) considera mais uma restrição: um motorista só pode estar alocado para tarefas de uma única tabela em um dado dia. Essa restrição foi adicionada porque ela melhora a execução de seu modelo, mesmo os autores acreditando que a remoção dessa restrição poderia produzir soluções melhores.

A expressão $\left|\bigcup_{i=1}^{|T|} T_i\right|$ define uma função objetivo para o problema. Na qual T_i é o motorista que está alocado para a tarefa i. A expressão é igual ao número de motoristas utilizados nas tarefas T.

O escopo escolhido para esse trabalho é o de linha, ou seja, tarefas de diferentes tabelas da linha podem ser atribuídas a um mesmo motorista. Como (NUNES, 2015) afirma,

o escopo de empresa pode adicionar complexidade extra para considerar o deslocamento dos motoristas de uma linha para outra, e também que o escopo de linha é satisfatório para as empresas.

3 TRABALHOS RELACIONADOS

O Problema de Programação de Tripulação (PPT) é amplamente estudado nos países desenvolvidos e seus resultados possuem larga aplicação (AGGARWAL, 1982). O PPT pode ser dividido em duas partes: o Problema de Determinação de Tarefas e o Problema de Alocação de Escala, e a abordagem para a primeira parte pode influenciar na qualidade e velocidade das soluções para a segunda parte (NUNES, 2015).

Nunes e Nepomuceno (2015) seguem a abordagem de dividir o PPT e resolvem a primeira parte como um *Set Partition Problem*. Sua decisão é justificada pelo fato dessa abordagem reduzir o espaço de busca para a segunda parte do problema que eles resolvem com um modelo em Programação Linear Inteira. Nunes e Nepomuceno (2015) restringem o problema a um escopo menor. Em sua instância, os motoristas só podem ser alocados para tarefas de uma mesma tabela. Os diferentes tipos de dias não são considerados assim como os conceitos de motoristas efetivos e folguistas.

O objetivo do trabalho é minimizar o número de motoristas utilizados nas escalas, e portanto, custos variáveis com adicionais noturnos e horas extras não são levados em consideração. No caso do adicional noturno, o mesmo acontece com a solução proposta por esse trabalho: o custo não será considerado. Para o modelo, uma hora noturna é igual a uma hora diurna. Entretanto, horas extras são tratadas por causa das legislações trabalhistas que restringem a quantidade de horas extras seguidas, mas apenas no aspecto da quantidade e não na variabilidade de seu valor em relação à hora normal. Este trabalho remove a restrição de que um motorista pode trabalhar em apenas uma tabela por dia.

Mayrink e Silva (2012) apresentam uma abordagem dividida em duas etapas. Na primeira etapa, é gerada uma sequência de jornadas diárias sem considerar as restrições das folgas legais das tripulações. Dessa forma, no primeiro dia do horizonte de planejamento as jornadas são atribuídas de forma sequencial às tripulações. Na segunda etapa, as jornadas a serem atribuídas no próximo dia são definidas resolvendo um modelo de designação que busca minimizar o acumulado de horas extras considerando como essas horas podem ser compensadas com horas ociosas no dia seguinte. Para isso, é proposto um modelo de Programação Inteira para completar as folgas não atribuídas na etapa anterior.

A necessidade de contratar pessoal extra, denominados folguistas, para atender demandas quando for necessário dar folga a uma tripulação em dias úteis é considerada em um outro modelo proposto por Mayrink e Silva (2012). Esse modelo tem como objetivo minimizar o

número de folguistas na escala.

Diferentemente dos trabalhos mencionados acima, este propõe um modelo em Programação com Restrições com o objetivo de reduzir o tempo para se encontrar soluções viáveis. Este trabalho também seguirá a abordagem de dividir o PPT em duas partes, mas assume que a primeira parte já foi resolvida e receba como entrada.

4 MODELO DE PROGRAMAÇÃO POR RESTRIÇÕES

Variáveis

São essas as variáveis do modelo.

- TM_t: define qual motorista está alocado para a tarefas t.
- MT_{md}: define quais tarefas são atribuídas ao motorista m no dia d.
- HM_{md}: define a quantidade de horas extras cumpridas pelo motorista m no dia d.
- C: define o custo de um nó no espaço de busca.
- T_d: é o conjunto de índices de tarefas no dia d.
- M: é o conjunto dos índices dos motoristas.
- D: são os índices dos dias da escala.
- QMHD: é o máximo de horas por dia que um motorista pode trabalhar.
- QMHED: é o máximo de horas extras permitidas por dia.
- DT: é a duração da tarefa t.
- QMHES: é quantidade máxima de horas extras por semana.
- L_m: é o horário da última tarefa to motorista m na escala anterior.
- F_m: é a quantidade de domingos desde a última folga do motorista m em um domingo
- JMS: é tamanho da jornada semanal de um motorista.

Restrições

Na Equação 4.1 define o custo de uma solução, ou seja, o número de motorista utilizados. A operação de *channel* está definida na Seção 2.3.1.

$$\mathrm{C} = |channel(\mathrm{TM})| \tag{4.1}$$

A Equação 4.2 assegura que duas tarefas de um mesmo dia que não são de uma mesma tabela devem ser atribuídas para motoristas diferentes, isso será alterado no novo modelo. O predicado binário *tabela* diz se duas tarefas pertencem a mesma tabela.

$$\forall t_1 \forall t_2 \in \mathbf{T}_d, \quad \neg tabela(t_1, t_2) \implies \mathrm{TM}_{t_1} \neq \mathrm{TM}_{t_2} \tag{4.2}$$

A Equação 4.3 relaciona as variáveis TM_t e TM_{md}, para cada dia d da escala.

$$\mathrm{TM}_t = m \iff t \in \mathrm{TM}_{md} \tag{4.3}$$

A Equação 4.4 limita a jornada diária.

$$\forall i \in \mathrm{M}, \forall j \in \mathrm{D}, \quad \sum_{t}^{\mathrm{M}_{ij}} DT_t \leq \mathrm{QMHD} + \mathrm{QMHED} \tag{4.4}$$

Para cada motorista m e para cada semana s:

$$\sum_{d=1}^{|s|} \mathrm{HM}_{ms_d} \leq \mathrm{QMHES} \tag{4.5}$$

Para cada par de tarefas, t_1 e t_2, em dias consecutivos:

$$inicio_{t2} - fim_{t1} \geq 11 \cdot 60 \tag{4.6}$$

A Equação 4.7 concede folgas entre jornadas dias consecutivos de trabalho. A operação de $count(x,y)$ consiste de quantas vezes y se repete em x. Para cada motorista m e para cada 7-uplas de dias consecutivos $(\mathrm{MT}_{m_{d_i}}, \mathrm{MT}_{md+1} \ldots \mathrm{MT}_{md_{i+5}}, \mathrm{MT}_{md_{i+6}})$:

$$count((\mathrm{MT}_{md_i}, \mathrm{MT}_{md+1} \ldots \mathrm{MT}_{md_{i+5}}, \mathrm{MT}_{md_{i+6}}), \emptyset) \geq 1 \tag{4.7}$$

Para cada tarefa t do primeiro dia da escala e para cada motorista m:

$$inicio_t - L_m < 11 \cdot 60 \implies t \neq m \tag{4.8}$$

A Equação 4.9 considera a folga na escala anterior para conceder as folgas na escala atual. Para cada motorista m, e para $(7 - D_m)$-upla $(\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+1}} \ldots \mathrm{MT}_{md_{i+7-D_m-2}}, \mathrm{MT}_{md_{i+7-D_m-1}})$, sendo que D_m é de dias desde a última folga do motorista m na escala anterior:

$$count((\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+1}} \ldots \mathrm{MT}_{md_{i+7-D_m-2}}, \mathrm{MT}_{md_{i+7-D_m-1}}), \emptyset) \geq 1 \tag{4.9}$$

A Equação 4.10 concede folgas nos domingos. Para cada motorista m, e para z-upla $(\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+7}} \ldots \mathrm{MT}_{md_{i+7\cdot(z-2)}}, \mathrm{MT}_{md_{i+7\cdot(z-1)}})$, sendo que z é número máximo de domingos sem folgas, e i é um domingo:

$$count((\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+7}} \ldots \mathrm{MT}_{md_{i+7\cdot(z-2)}}, \mathrm{MT}_{md_{i+7\cdot(z-1)}}), \emptyset) \geq 1 \tag{4.10}$$

Para considerar a última folga em um domingo na escala anterior tem-se a Equação 4.11. Para cada motorista m, e para $(z - F_m)$-upla $(\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+7}} \ldots \mathrm{MT}_{md_{i+7\cdot(z-F_m-3)}}, \mathrm{MT}_{md_{i+7\cdot(z-F_m-2}})$:

$$count((\mathrm{MT}_{md_i}, \mathrm{MT}_{md_{i+7}} \ldots \mathrm{MT}_{md_{i+7\cdot(z-F_m-3)}}, \mathrm{MT}_{md_{i+7\cdot(z-F_m-2}}), \emptyset) \geq 1 \tag{4.11}$$

A Equação 4.12 calcula a quantidade de horas não aproveitadas e de horas extras na jornada semanal de um motorista. HN_{ms} é negativo se houveram horas não aproveitada e positivo se foram utilizadas horas extras. Para cada motorista m e para cada semana s da escala:

$$\mathrm{HN}_{ms} = \left(\sum_{d=1}^{|s|} \sum_{t \in \mathrm{MT}_{ms_d}} \mathrm{DT}_t \right) - \mathrm{JMS} \tag{4.12}$$

Para cada motorista m e para cada semana s, sendo que DT é duração da tarefa t e D é o conjunto dos índices dos dias da escala:

$$\sum_{d=1}^{|s|} \sum_{t \in \mathrm{MT}_{ms_d}} \mathrm{DT}_t = \mathrm{JMS} + \mathrm{HN}_{ms}, \forall m \exists x \in \mathrm{D}, \text{ tal que } \mathrm{MT}_{mx} \neq \emptyset \tag{4.13}$$

5 RESULTADOS EXPERIMENTAIS

A implementação do modelo foi testada para verificar se consegue produzir soluções viáveis, com tempo limite de uma hora. Os testes foram executados em um computador com processador Intel Xeon E31240, 8 GB de memória RAM e sistema operacional Ubuntu 14.04.5.

Também foi analisado o comportamento do modelo quando se buscava minimizar a quantidade de motoristas utilizados nas soluções, com o tempo limite de dois minutos. Esse tempo é justificado pelo fato de que o modelo deve produzir soluções inciais e não necessariamente soluções ótimas.

Além disso, foi analisado como a estratégia de *branching* utilizada pelo Gecode influência na qualidade das soluções e no tempo de execução. Foram utilizados as seguintes estratégias:

- atribuir a primeira variável ainda não atribuída o menor valor possível;
- atribuir a primeira variável ainda não atribuída o valor médio dentre os possíveis.
- atribuir a primeira variável ainda não atribuída um valor aleatório dentre os possíveis;
- atribuir a uma variável ainda não atribuída escolhida aleatoriamente um valor aleatório dentre os possíveis;
- atribuir a uma variável ainda não atribuída escolhida aleatoriamente o menor valor possível;
- atribuir a uma variável ainda não atribuída escolhida aleatoriamente o valor médio dentre os possíveis;

As soluções produzidas pelo modelo inicial foram validadas usando uma implementação do modelo de Nunes e Nepomuceno (2015). As soluções foram dadas como solução inicial para o modelo em PLI implementado usando o CPLEX. O CPLEX consegue determinar então se a solução inicial é viável.

Preparação das Entradas

As entradas para o modelo foram fornecidas por uma empresa de atuante no ramo, e são dados reais de linhas, tabelas e motoristas, sendo os mesmo dados utilizados por Nunes e Nepomuceno (2015). A definição das tarefas é um problema à parte que neste trabalho foi assumido como resolvido. Os dados foram pré-processados para estarem compatíveis com a

forma que o modelo espera as entradas.

Nas entradas estão especificadas as seguintes informações:

- a quantidade de semanas da escala;
- o número de motoristas;
- tamanho máximo da jornada de trabalho;
- a última folga de cada motorista;
- última folga em um domingo de cada motorista;
- hora final da jornada de cada motorista no dia imediatamente anterior ao início da escala;
- a quantidade de tabelas em dias úteis, sábados e domingos.
- a data e horário de inicio e fim das tarefas, assim como a duração, o índice do dia, o código da tabela, o código do tipo de dia e de tipo de horário.

Análise da Execução do Modelo

O modelo implementado com o Gecode 4.4.0 foi executado utilizando entradas de Nunes e Nepomuceno (2015). Foram utilizadas 56 entradas, variando número de motoristas e tamanho máximo de tarefa. A Tabela 1 detalha as entradas utilizadas, apresentando o horizonte da escala, a quantidade de motoristas disponíveis para definir a escala e o tamanho máximo das tarefa.

Quadro 1 – Correspondência

Código	Descrição
RND MIN	Atribuir a uma variável ainda não atribuída escolhida aleatoriamente o menor valor possível.
RND MED	Atribuir a uma variável ainda não atribuída escolhida aleatoriamente o valor médio dentre os possíveis.
RND RND	atribuir a uma variável ainda não atribuída escolhida aleatoriamente um valor aleatório dentre os possíveis
NONE RND	Atribuir a primeira variável ainda não atribuída um valor aleatório dentre os possíveis
NONE MED	Atribuir a primeira variável ainda não atribuída o valor médio dentre os possíveis.
NONE MIN	Atribuir a primeira variável ainda não atribuída o menor valor possível.

Fonte – Elaborado pelo autor

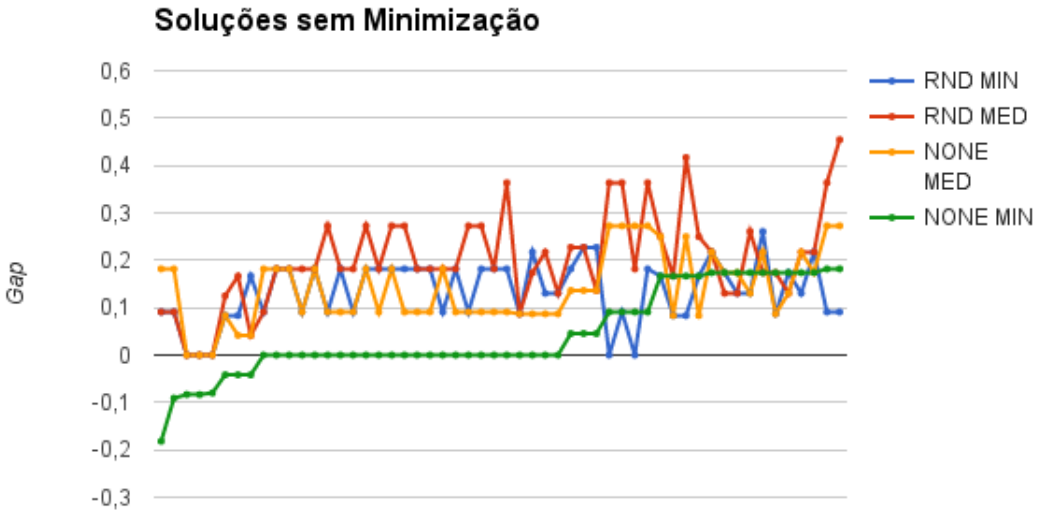
A nomenclatura descrita no Quadro 1 foi utilizada para se referir as estratégias de *branching* utilizadas na implementação do modelo.

Nas tabelas no Apêndice A, soluções com 0 motoristas significam que não foi encontrada uma solução para aquela entrada. Não é apresentado nessa seção o tempo de execução quando a minimização foi realizada pois todas as entradas atingiram o *time-out*. As

entradas 8 e 16 não possuem soluções viáveis.

Na Tabela 2 estão descritas as soluções quando a busca parou na primeira solução viável variando a estratégia de *branching*, o que teve influencia em todas as entradas. Nas primeiras 15 entradas essas diferença foi bastante acentuada, sendo que com RND RND e NONE RND as soluções utilizaram mais do dobro de motoristas que as outras. Nas últimas 23 entradas esse comportamento se repetiu. Para as entradas 37 e 47 o *branching* não foi possível encontrar respostas dentro do tempo limite utilizando NONE MIN (para a 37) , NONE RND e RND RND (para a 47). As outras soluções ficaram próximas evidenciando pouca influência do *branching*.

Figura 2 – Soluções sem minimização - *branching*



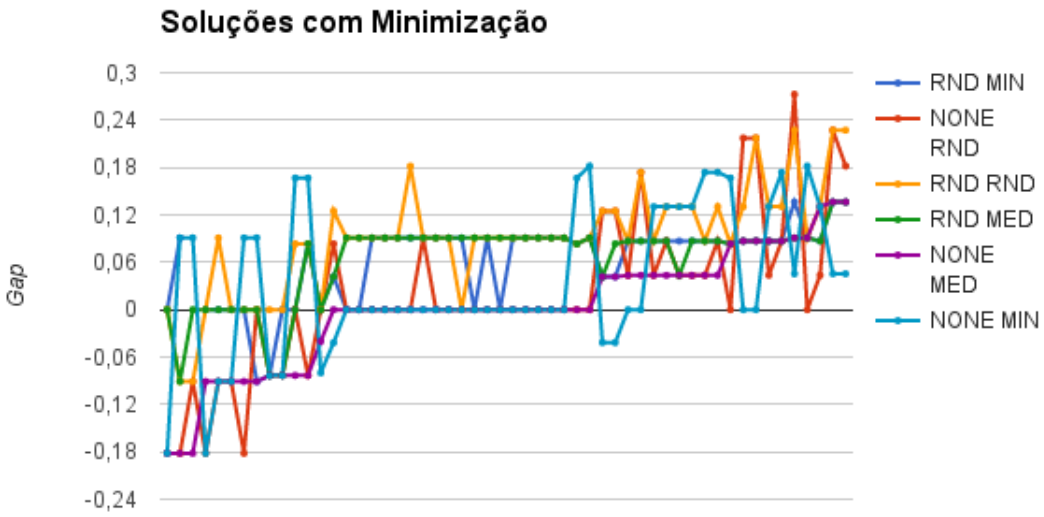
Fonte – O próprio autor.

A Figura 2 mostra um gráfico do *gap* entre as soluções encontradas, sem minimização, em relação as soluções de (NUNES; NEPOMUCENO, 2015) para cada um das estratégias de textitbranching. As estratégias RND RND e NONE RND não aparecem nos gráfico pois as soluções encontradas utilizando essas estratégias foram muito próximas do número máximo de motoristas disponíveis para a escala. A estratégia NONE MIN se sai melhor na maioria dos casos, mas falhou em encontrar solução para uma entrada.

Na Tabela 3 estão as soluções encontradas para cada entrada quando foi realizada a etapa de minimização variando a estratégia de *branching*. Essa etapa de minimização permitiu encontrar soluções melhores que as da Tabela 2. A diferença no número de motoristas em cada

solução foi bem menor que na Tabela 6, isso se deu pela minimização que melhora as soluções, justificando essa etapa. No entanto, para as entradas 12, 37 e 47 não foi possível encontrar solução dentro do tempo limite com RND MIN (para a 12), NONE MIN (para a 37) , NONE RND e RND RND (para a 47).

Figura 3 – Soluções com minimização- *branching*



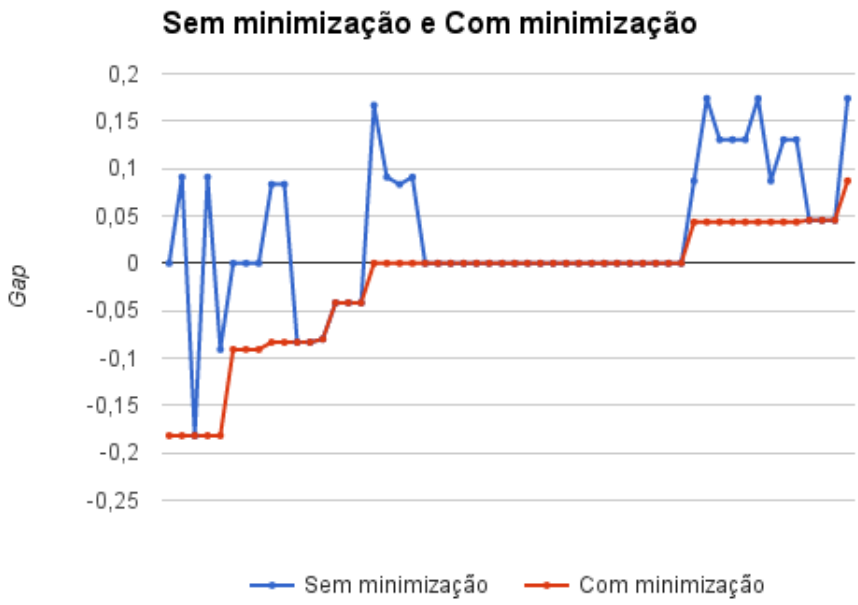
Fonte – O próprio autor.

A Figura 3 mostra um gráfico do *gap* entre as soluções encontradas, com minimização, em relação as soluções de (NUNES; NEPOMUCENO, 2015) para cada um das estratégias de textitbranching. As estratégias que se saem melhor em mais casos são NONE MIN e NONE MED.

Na Tabela 4 estão descritos os tempos de execução para cada entrada e para cada estratégia de *branching* para encontrar a primeira solução viável. Em seguida, na Tabela 5 são mostradas as melhores soluções encontradas quando houve minimização para cada entrada e o tipo de *branching* utilizado. Por último, a Tabela 6 apresentas as melhores soluções para cada entrada, sem minimização, o tempo de execução e o tipo de *branching*.

A Figura 4 mostra um gráfico comparando o *gap* entre as melhores soluções com minimização e sem minimização para as soluções de Nunes e Nepomuceno (2015). É possível perceber que mesmo sem minimização, algumas soluções foram melhores que as de Nunes e Nepomuceno (2015). Com a minimização, a maioria das soluções foram melhores ou

Figura 4 – Comparativo das melhores soluções com e sem minimização



Fonte – O próprio autor.

equivalentes as soluções de Nunes e Nepomuceno (2015), justificando essa etapa adicional em 22 casos.

6 CONSIDERAÇÕES FINAIS

Neste trablaho, foram encontradas soluções viáveis para o Problema de Programação de Tripulação. Os resultados obtidos evidenciam que ao remover a restrição de que um motorista pode trabalhar em apenas um tabela por dia, ainda é possível encontrar soluções viáveis. Algumas dessas soluções utilizam menos motoristas que as soluções encontradas por Nunes e Nepomuceno (2015), o que justifica sua remoção. No entanto, é preciso verificar se após remover essa restrição, a solução ainda faz sentido no mundo real, onde seria aplicada.

Durante o desenvolvimento desse trabalho, foi identificado que a estratégia de *branching* escolhida pode impactar bastante no tempo de execução e na qualidade das soluções viáveis.

Executar o modelo várias vezes, alterando a maneira como o *branching* é realizado pode demandar tempo. Então, uma abordagem com reinicio de busca pode ser uma alternativa mais eficiente. Nessa abordagem, a busca para quando atinge um limite especificado, como número de nós com falha ou profundidade da árvore, e é reiniciada utilizando o que foi aprendido na tentativa anterior. Durante o reinicio, o modo de realizar o *branching* pode ser alterado, permitindo assim, que várias estratégias possam ser utilizadas de forma mais simples.

Outra abordagem, poderia ser definir uma estratégia específica para o problema, considerando suas particularidades. Essa abordagem exige um entendimento mais profundo do problema. Heurísticas podem ser úteis para tornar o *branching* mais eficiente.

Fica como trabalho futuro verificar se utilizar as soluções viáveis encontradas neste trabalho pode melhorar o desempenho de outros modelos. Fica ainda modificar a definição do problema para considerar mais de uma linha na definição da escala e adicionar os conceitos de motoristas folguistas e efetivos.

REFERÊNCIAS

AGGARWAL, S. C. A focussed review of scheduling in services. **European Journal of Operational Research**, Elsevier, v. 9, n. 2, p. 114–121, 1982.

APT, K. **Principles of constraint programming**. Amsterdã: Cambridge University Press, 2003. 404 p.

BARTÁK, R. On-line guide to constraint programming. Roman Barták, 1998.

CARRARESI, P.; GALLO, G. A multi-level bottleneck assignment approach to the bus drivers' rostering problem. **European Journal of Operational Research**, Elsevier, v. 16, n. 2, p. 163–173, 1984.

CORMEN, T. H. **Introduction to algorithms**. Terceira edição. Boston: MIT press, 2009. 1292 p.

Gecode Team. **Gecode: Generic Constraint Development Environment**. 2006. Acesso em: 21 de jun. 2016. Disponível em: <http://www.gecode.org>.

MARTELLO, S.; TOTH, P. A heuristic approach to the bus driver scheduling problem. **European Journal of Operational Research**, Elsevier, v. 24, n. 1, p. 106–117, 1986.

MAYRINK, V. T. d. M.; SILVA, G. P. Otimização do rodízio de tripulações do sistema de transporte público. **Journal of Transport Literature**, Scielo, v. 7, n. 3, p. 192–203, 2012.

MONTANARI, U. Networks of constraints: Fundamental properties and applications to picture processing. **Information sciences**, Elsevier, v. 7, p. 95–132, 1974.

NIEDERLIŃSKI, A. **A Gentle Guide to Constraint Logic Programming**. Terceira edição. Gliwice: pkjs.com.pl, 2009. 509 p.

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.

NUNES, R. de P.; NEPOMUCENO, N. V. Programação de tripulação no transporte de ônibus urbano: Uma abordagem utilizando programação linear inteira. **Simpósio Brasileiro de Pesquisa Operacional**, p. 1604–1615, 2015.

PORTUGAL, R.; LOURENÇO, H. R.; PAIXÃO, J. P. Driver scheduling problem modelling. **Public transport**, Springer, v. 1, n. 2, p. 103–120, 2009.

ROSSI, F.; BEEK, P. V.; WALSH, T. **Handbook of constraint programming**. [S.l.]: Elsevier, 2006.

SILVA, G. P.; SOUZA, M. J. F.; ATZINGEN, J. von. Métodos exatos para resolver o problema de programação da tripulação. **TRANSPORTES**, v. 14, n. 1, 2006.

YUNES, T. H.; MOURA, A. V.; SOUZA, C. C. de. Solving very large crew scheduling problems to optimality. In: ACM. **Proceedings of the 2000 ACM symposium on Applied computing-Volume 1**. Como, Itália, 2000. p. 446–451.

APÊNDICE A – TABELAS

Tabela 1 – Descrição das Entradas

Entrada	Quantidade de Semanas	Quantidade de Motoristas	Tamanho Máximo da Tarefa
1	4	64	360
2	4	64	420
3	4	64	480
4	4	64	540
5	4	49	360
6	4	49	420
7	4	49	480
8	4	49	540
9	4	64	360
10	4	64	420
11	4	64	480
12	4	64	540
13	4	49	360
14	4	49	420
15	4	49	480
16	4	49	540
17	4	12	440
18	4	17	440
19	4	22	440
20	4	27	440
21	4	32	440
22	4	37	440
23	4	42	440
24	4	47	440
25	4	52	440
26	4	57	440

27	4	12	440
28	4	17	440
29	4	22	440
30	4	27	440
31	4	32	440
32	4	37	440
33	4	42	440
34	4	47	440
35	4	52	440
36	4	57	440
37	4	25	440
38	4	30	440
39	4	35	440
40	4	40	440
41	4	45	440
42	4	50	440
43	4	55	440
44	4	60	440
45	4	65	440
46	4	70	440
47	4	25	440
48	4	30	440
49	4	35	440
50	4	40	440
51	4	45	440
52	4	50	440
53	4	55	440
54	4	60	440
55	4	65	440
56	4	70	440

Tabela 2 – Soluções sem Minimização

Entrada	RND MIN	NONE RND	RND RND	RND MED	NONE MED	NONE MIN
1	14	63	64	15	15	14
2	12	64	64	15	14	13
3	11	64	64	15	14	12
4	12	63	61	15	14	12
5	13	48	49	14	13	14
6	12	49	49	12	13	11
7	12	49	49	12	13	9
8	0	0	0	0	0	0
9	13	64	64	17	15	14
10	12	64	64	16	14	13
11	11	63	64	13	14	12
12	13	64	63	15	14	12
13	14	48	49	15	13	14
14	13	49	49	13	13	11
15	12	49	49	12	13	10
16	0	0	0	0	0	0
17	12	12	12	12	12	11
18	13	17	17	13	13	11
19	12	22	22	13	12	11
20	13	27	27	13	13	11
21	12	32	32	14	12	11
22	13	37	37	13	12	11
23	12	42	42	13	12	11
24	13	47	47	14	13	11
25	13	51	52	13	12	11
26	13	57	57	14	13	11
27	12	12	12	12	12	11
28	13	17	17	14	12	11
29	13	22	22	13	12	11

30	13	27	27	13	12	11
31	12	32	32	13	13	11
32	13	37	37	13	12	11
33	12	42	42	14	12	11
34	13	47	47	14	12	11
35	13	52	52	13	12	11
36	13	57	56	15	12	11
37	25	25	25	25	25	0
38	28	30	30	28	28	27
39	27	35	35	26	27	27
40	26	40	40	26	27	27
41	26	45	45	29	26	27
42	29	50	50	27	28	27
43	25	55	55	27	25	27
44	27	60	60	26	26	27
45	26	65	65	28	28	27
46	28	70	70	28	27	27
47	25	0	0	25	25	23
48	26	30	30	27	26	23
49	26	35	35	28	25	23
50	28	40	40	25	25	23
51	28	45	45	27	25	23
52	26	50	50	28	25	23
53	26	55	55	26	25	23
54	26	60	60	27	25	23
55	27	65	65	27	25	23
56	27	70	70	25	25	23

Tabela 3 – Soluções com Minimização

Entrada	RND MIN	NONE RND	RND RND	RND MED	NONE MED	NONE MIN

1	13	12	13	13	13	14
2	12	11	12	12	12	13
3	11	9	11	11	10	12
4	12	9	10	10	9	12
5	12	12	13	12	11	14
6	11	10	12	11	10	10
7	11	9	11	11	10	9
8	0	0	0	0	0	0
9	13	12	13	13	12	14
10	12	11	12	12	11	13
11	10	11	11	11	10	12
12	0	10	10	11	9	12
13	13	11	13	13	11	14
14	11	10	11	11	10	10
15	11	9	11	11	9	9
16	0	0	0	0	0	0
17	11	11	12	11	11	11
18	11	11	12	12	11	11
19	11	11	12	12	11	11
20	12	11	12	12	11	11
21	12	11	12	12	11	11
22	12	11	12	12	11	11
23	12	11	13	12	11	11
24	12	12	12	12	11	11
25	12	11	12	12	11	11
26	12	11	12	12	11	11
27	12	11	12	11	11	11
28	12	11	11	12	11	11
29	11	11	12	12	11	11
30	12	11	12	12	11	11
31	11	11	12	12	11	11
32	12	11	12	12	11	11

33	12	11	12	12	11	11
34	12	11	12	12	11	11
35	12	11	12	12	11	11
36	12	11	12	12	11	11
37	25	24	25	25	24	0
38	25	24	25	25	24	26
39	25	24	26	25	25	26
40	25	24	25	25	24	27
41	25	25	26	25	24	26
42	25	25	26	25	24	27
43	25	24	26	24	24	26
44	25	24	26	25	24	26
45	25	24	26	25	26	26
46	25	25	26	25	25	27
47	25	0	0	25	24	23
48	25	27	27	25	25	23
49	25	27	27	26	25	23
50	25	26	27	25	24	23
51	25	28	26	25	25	23
52	25	28	28	25	25	23
53	25	27	27	25	24	23
54	25	27	27	25	25	23
55	25	26	27	25	25	23
56	25	28	27	24	24	23

Tabela 4 – Soluções sem Minimização (Tempo de Execução)

Entrada	RND MIN	NONE RND	RND RND	RND MED	NONE MED	NONE MIN
1	3.56907	3.45891	5.77788	6.81137	4.3049	0.973142
2	6.82657	0.516314	4.93853	4.70572	0.732487	4.80641
3	4.41577	3.12046	1.5234	1.57038	3.39226	3.30443

4	227.825	0.46376	1.54845	259.118	3.05679	3.01003
5	0.49993	2.29537	4.16643	1.52142	3.01116	0.549836
6	0.430504	1.87226	3.42093	1.97362	2.49663	2.44178
7	3.17242	0.35839	1.88138	0.394625	2.24466	2.26313
8	0.022818	0.005271	0.080024	0.042927	0.008529	0.067017
9	0.845971	0.61051	0.856772	5.29782	4.69482	4.6059
10	4.49713	3.03965	1.78137	2.88541	2.44266	0.752796
11	4.07552	1.83135	1.59215	0.603526	3.2327	3.35386
12	15576.8	0.428378	0.610297	5.3335	3.07693	0.615507
13	2.50378	2.38542	6.5308	3.56364	3.30608	3.05764
14	0.458219	1.98878	0.44628	2.2517	2.41526	2.53721
15	5.17992	1.72799	0.409991	1.15691	2.34921	0.424532
16	0.060717	0.011544	0.048873	0.075958	0.007346	0.014803
17	0.054382	0.038515	13.043	7.01054	0.040468	0.321215
18	0.075547	0.052718	0.100661	0.07596	0.05693	0.060154
19	0.101466	0.066942	0.616241	0.103329	0.079517	0.643401
20	0.13152	0.086987	1.33425	0.141804	2.41034	0.125162
21	1.76964	0.970228	1.69103	1.74553	0.602041	0.173518
22	1.23861	1.35433	2.08494	1.99546	1.40489	1.40562
23	3.93725	1.47666	0.310899	0.820521	1.9931	1.97597
24	2.72665	0.258484	0.89762	0.345448	2.41345	2.22715
25	4.04031	2.01688	2.44429	3.09458	0.483461	2.54955
26	0.502725	2.42912	0.547577	0.49722	2.86299	2.81506
27	0.910616	0.050664	5.69645	0.059773	0.044751	0.072109
28	0.086027	0.057152	0.088742	0.088985	0.060522	0.067814
29	1.09891	0.078497	0.119263	0.11403	0.08511	0.113631
30	0.462509	0.924797	0.155353	0.149681	0.968884	1.01693
31	1.11623	0.129222	3.31968	1.87174	0.223608	1.30271
32	0.263944	1.42899	0.2601	2.27655	1.64742	1.5971
33	1.77374	0.254934	0.897971	2.66343	1.94753	2.07232
34	1.95859	1.87671	1.2473	0.39995	2.28002	2.42907
35	0.496412	2.468	0.493063	4.57128	2.75379	2.49527

36	1.47622	2.46153	0.580444	3.92724	3.13818	0.624939
37	25.7377	11.4318	683.041	49.5032	66.0806	-
38	3.0879	1.26019	0.434027	1.83765	2.43852	4.52997
39	0.50086	2.79575	7.70871	1.52717	0.388069	0.424916
40	1.61014	0.422456	1.85874	15.3379	3.46212	3.53893
41	1.85481	0.524422	8.63002	2.86745	0.686489	0.687956
42	0.853213	6.21025	1.03182	2.33932	7.39414	5.19898
43	2.62231	9.89269	3.44848	2.73773	10.2593	9.09898
44	9.15255	8.07888	3.20291	1.33439	1.44545	7.60313
45	9.84177	8.59023	6.72371	6.4883	15.8227	12.4862
46	8.00233	1.17836	1.67858	4.51032	2.04844	14.427
47	0.366318	-	-	4.40709	3.09024	0.252487
48	1.24666	3.82878	1.37474	1.32551	0.331491	0.354851
49	1.60665	2.94539	1.72298	1.97218	5.69398	6.10929
50	1.89504	2.36634	0.89196	3.95043	2.45853	7.38912
51	7.55842	2.84273	2.63454	2.2291	7.16855	7.75904
52	2.54314	1.62764	5.07941	1.06019	11.2608	0.98842
53	2.97882	0.825716	23.0998	6.21234	11.7455	4.94167
54	3.48186	2.24055	1.56317	1.48817	5.78323	6.1677
55	3.98864	7.49224	4.97121	1.63356	1.91951	10.7538
56	4.45353	2.92445	8.79982	4.62227	21.0733	9.13403

Tabela 5 – Melhores Soluções com Minimização

Entrada	Quantidade de Motoristas	*Branching*
1	12	NONE RND
2	11	NONE RND
3	9	NONE RND
4	9	NONE RND
5	11	NONE MED
6	10	NONE RND

7	9	NONE RND
8	-	-
9	12	NONE RND
10	11	NONE RND
11	10	RND MIN
12	9	NONE MED
13	11	NONE RND
14	10	NONE RND
15	9	NONE RND
16	-	-
17	11	RND MIN
18	11	RND MIN
19	11	RND MIN
20	11	NONE RND
21	11	NONE RND
22	11	NONE RND
23	11	NONE RND
24	11	NONE MED
25	11	NONE RND
26	11	NONE RND
27	11	NONE RND
28	11	NONE RND
29	11	RND MIN
30	11	NONE RND
31	11	RND MIN
32	11	NONE RND
33	11	NONE RND
34	11	NONE RND
35	11	NONE RND
36	11	NONE RND
37	24	NONE RND
38	24	NONE RND

39	24	NONE RND
40	24	NONE RND
41	24	NONE MED
42	24	NONE MED
43	24	NONE RND
44	24	NONE RND
45	24	NONE RND
46	25	RND MIN
47	23	NONE MIN
48	23	NONE MIN
49	23	NONE MIN
50	23	NONE MIN
51	23	NONE MIN
52	23	NONE MIN
53	23	NONE MIN
54	23	NONE MIN
55	23	NONE MIN
56	23	NONE MIN

Tabela 6 – Melhores Respostas sem Minimização

Entrada	Quantidade de Motoristas	*Branching*	Tempo
1	14	RND MIN	3.56907
2	12	RND MIN	6.82657
3	11	RND MIN	4.41577
4	12	RND MIN	227.825
5	13	RND MIN	0.49993
6	11	NONE MIN	0.430504
7	9	NONE MIN	3.17242
8	-	-	-
9	13	RND MIN	0.845971

10	12	RND MIN	4.49713
11	11	RND MIN	4.07552
12	12	NONE MIN	15576.8
13	13	NONE MED	2.50378
14	11	NONE MIN	0.458219
15	10	NONE MIN	5.17992
16	-	-	-
17	11	NONE MIN	0.054382
18	11	NONE MIN	0.075547
19	11	NONE MIN	0.101466
20	11	NONE MIN	0.13152
21	11	NONE MIN	1.76964
22	11	NONE MIN	1.23861
23	11	NONE MIN	3.93725
24	11	NONE MIN	2.72665
25	11	NONE MIN	4.04031
26	11	NONE MIN	0.502725
27	11	NONE MIN	0.910616
28	11	NONE MIN	0.086027
29	11	NONE MIN	1.09891
30	11	NONE MIN	0.462509
31	11	NONE MIN	1.11623
32	11	NONE MIN	0.263944
33	11	NONE MIN	1.77374
34	11	NONE MIN	1.95859
35	11	NONE MIN	0.496412
36	11	NONE MIN	1.47622
37	25	RND MIN	25.7377
38	27	NONE MIN	3.0879
39	26	RND MED	0.50086
40	26	RND MIN	1.61014
41	26	RND MIN	1.85481

42	27	RND MED	0.853213
43	25	RND MIN	2.62231
44	26	RND MED	9.15255
45	26	RND MIN	9.84177
46	27	NONE MED	8.00233
47	23	NONE MIN	0.366318
48	23	NONE MIN	1.24666
49	23	NONE MIN	1.60665
50	23	NONE MIN	1.89504
51	23	NONE MIN	7.55842
52	23	NONE MIN	2.54314
53	23	NONE MIN	2.97882
54	23	NONE MIN	3.48186
55	23	NONE MIN	3.98864
56	23	NONE MIN	4.45353

Tabela 7 – Comparação entre as Melhores Respostas

Entrada	Quantidade de Motoristas com Minimização	Qunatidade de Motoristas sem Minimização
1	12	14
2	11	12
3	9	11
4	9	12
5	11	13
6	10	11
7	9	9
8	-	-
9	12	13
10	11	12
11	10	11
12	9	12

13	11	13
14	10	11
15	9	10
16	-	-
17	11	11
18	11	11
19	11	11
20	11	11
21	11	11
22	11	11
23	11	11
24	11	11
25	11	11
26	11	11
27	11	11
28	11	11
29	11	11
30	11	11
31	11	11
32	11	11
33	11	11
34	11	11
35	11	11
36	11	11
37	24	25
38	24	27
39	24	26
40	24	26
41	24	26
42	24	27
43	24	25
44	24	26

45	24	26
46	25	27
47	23	23
48	23	23
49	23	23
50	23	23
51	23	23
52	23	23
53	23	23
54	23	23
55	23	23
56	23	23