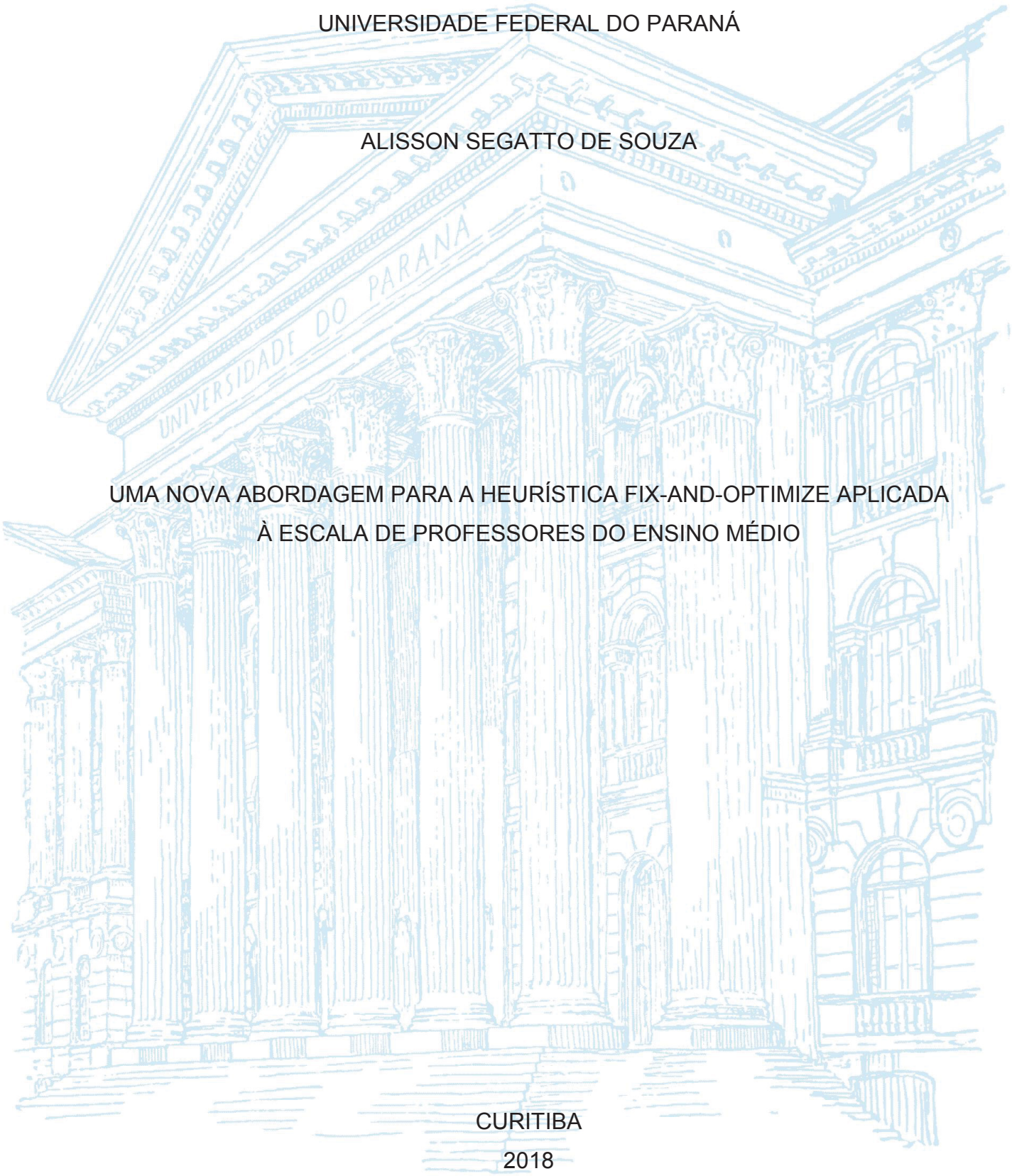
UNIVERSIDADE FEDERAL DO PARANÁ

ALISSON SEGATTO DE SOUZA



UMA NOVA ABORDAGEM PARA A HEURÍSTICA FIX-AND-OPTIMIZE APLICADA À ESCALA DE PROFESSORES DO ENSINO MÉDIO

CURITIBA

2018

ALISSON SEGATTO DE SOUZA

UMA NOVA ABORDAGEM PARA A HEURÍSTICA FIX-AND-OPTIMIZE APLICADA À ESCALA DE PROFESSORES DO ENSINO MÉDIO

Dissertação apresentada ao curso de Pós-Graduação em Métodos Numéricos em Engenharia, Setor de Ciências Exatas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Métodos Numéricos para Engenharia.

Orientador: Prof. Dr. José Eduardo Pécora Junior
Coorientador: Prof. Dr. Gustavo Valentim Loch

CURITIBA

2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

S279n Souza, Alisson Segatto de
Uma nova abordagem para a heurística fix-and-optimize aplicada à escala de professores do ensino médio / Alisson Segatto de Souza. – Curitiba, 2018.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, 2018.

Orientador: José Eduardo Pécora Junior – Coorientador: Gustavo Valentim Loch.

1. Programação heurística. 2. Programação (Matemática). 3. Programação linear. 4. Calendário escolar. I. Universidade Federal do Paraná. II. Pécora Junior, José Eduardo. III. Loch, Gustavo Valentim. IV. Título.

CDD: 519.64

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR CIÊNCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO MÉTODOS NUMÉRICOS EM ENGENHARIA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **ALISSON SEGATTO DE SOUZA** intitulada: **UMA NOVA ABORDAGEM PARA A HEURÍSTICA FIX-AND-OPTIMIZE APLICADA À ESCALA DE PROFESSORES DO ENSINO MÉDIO**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 08 de Fevereiro de 2018.

GUSTAVO VALENTIM LOCH
Presidente da Banca Examinadora (UFPR)

ROBERTO ZANETTI FREIRE
Avaliador Externo (PUC/PR)

CASSIUS TADEU SCARPIN
Avaliador Interno (UFPR)

RESUMO

Um dos maiores desafios enfrentados pelas instituições de ensino no início do ano letivo é a de distribuir as turmas e dias em que cada professor deverá atuar. Apesar de todo avanço computacional, a maioria das escolas ainda realiza a construção da grade escolar manualmente, tornando o trabalho demorado e envolvendo praticamente todo o corpo docente e pedagógico, secretaria e direção na busca de otimizar e satisfazer as preferências de seus professores. O problema de construir uma grade horária escolar de ensino médio (*High School Timetabling* - HSTT) consiste em reunir professores, turmas e salas em um determinado período, construindo a escala semanal da instituição de ensino. Para grandes escolas, mesmo utilizando *softwares,* não é possível encontrar boas soluções em um tempo computacional viável utilizando métodos exatos, tornando necessária a utilização de heurísticas. A heurística *fix-and-optimize* decompõe o problema original em subproblemas menores ao fixar grande parte das variáveis, gerando regiões de busca menores que podem ser facilmente exploradas por *softwares* de forma exata, encontrando ótimos locais para o problema. A maneira com que estas regiões são exploradas impactam na velocidade de convergência do algoritmo. Neste trabalho é apresentado dois modelos matemáticos de programação inteira mista para resolver cinco instâncias clássicas da literatura buscando diminuir o número de dias trabalhados pelos professores, eliminar períodos de ociosidade e aumentar o número de aulas geminadas tanto quanto possível. A heurística *fix-and-optimize* foi utilizada juntamente com cinco diferentes métodos de busca em vizinhança, com o objetivo de diminuir a complexidade do algoritmo e alcançar melhores resultados. Desta forma foram encontradas soluções superiores para duas das instâncias estudadas e igualando os resultados já conhecidos para as demais instâncias.

Palavras Chaves: *High School Timetabling, fix-and-optimize*, heurística, busca em vizinhança.

ABSTRACT

One of the biggest challenges faced by educational institutions at the beginning of the school year is to distribute the classes and days when each teacher should act. Despite all the computational advances, most schools still carry out the construction of the school grid manually, making the work time consuming and involving practically all the teaching and pedagogical staff, secretariat and direction in the search to optimize and satisfy the preferences of their teachers. The problem of building a high school timetable (HSTT) consists of bringing together teachers, classes and classrooms in a given period, building the weekly scale of the teaching institution. For grades schools, even using software, it is not possible to find good solutions in a viable computational time using exact methods, making it necessary to use heuristics. The fix-and-optimize heuristic decomposes the original problem into smaller subproblems by fixing a large number of variables, generating smaller search regions that can be easily exploited by software accurately, finding optimal locations for the problem. The way in which these regions are exploited impacts on the convergence speed of the algorithm. In this work two mathematical models of mixed integer programming are presented to solve five classic instances of literature aiming to reduce the number of days worked by teachers, eliminate periods of idleness and increase the number of twin classes as much as possible. The fix-and-optimize heuristic was used along with five different neighborhood search methods, in order to reduce the complexity of the algorithm and achieve better results. In this way superior solutions were found for two of the studied instances and matching the results already known to the other instances.

Keywords: High School Timetabling, fix-and-optimize, heuristic, neighborhood search.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 . INTRODUÇÃO

Todas as instituições de ensino possuem a tarefa de agendar suas aulas, combinando alunos, professores, salas e disciplinas em períodos de tempo viáveis, respeitando as particularidades físicas, pessoais, pedagógicas e institucionais.

Construir esta escala manualmente é uma tarefa difícil e suscetível a erro, chegando a ser impossível para grandes instituições devido a sua complexidade, por possuir muitas variáveis e restrições. Estas restrições podem ser classificadas em *hard* e *soft*. Restrições *hard* são aquelas que precisam ser respeitadas para garantir a factibilidade da solução, enquanto as restrições *soft* representam as preferências sobre o problema, podendo ser desrespeitadas se necessário. O cumprimento das restrições *soft* reflete diretamente na qualidade da solução.

A qualidade da escala horária das escolas impacta diretamente a vida de um grande número de pessoas, professores e alunos, durante todo um semestre ou até mesmo ano letivo. Para os professores, uma grade horária de má qualidade pode resultar em dias a mais de trabalho, muitos períodos de ociosidade durante a semana e até mesmo prejudicar o método de ensino caso as aulas sejam distribuídas de maneira esparsa. Para alunos, uma má grade horária pode influenciar não apenas em seu aprendizado, mas também em sua saúde já que o peso das mochilas e livros acabam aumentando caso suas disciplinas sejam má distribuídas, características que são dificilmente abordadas nos problemas de *high school timetabling*. (Dorneles, 2015).

Mesmo hoje em dia, muitas instituições ainda criam a grade horária manualmente, sendo mais comum usarem *softwares* especializados. Existem uma grande variedade de opções no mercado, sendo algumas delas gratuitas e disponíveis na internet. Porém, poucas delas são realmente independentes, necessitando de ajustes manuais após o processamento; ou então simplesmente não modelam todas as características e restrições das instituições.

De fato, também é comum que cada instituição possua suas próprias diretrizes a serem consideradas durante a construção da grade horária, dificultando a criação de um método genérico que aborde todas estas restrições. Deste modo, os pesquisadores do *high school timetabling* acabaram focando suas pesquisas em suas próprias regiões e

países, dificultando a comparação dos resultados obtidos por falta de uma base comum de dados e instâncias de testes.

Foi para resolver este problema que os pesquisadores desenvolveram o XHSTT, uma base de dados com problemas de diversos países de forma a servir como base para qualquer pesquisa na área[1]. (POST *et al*, 2011). O XHSTT conta com aproximadamente 50 instâncias e foi bem aceita por todos os pesquisadores, sendo adotada como base para o *Third International Timetabling Competition* (ITC2011). Os problemas que podem ser adotados pelo XHSTT são chamados de *Generalized High School Timetabling Problem*. (DORNELES et al, 2014; DORNELES, 2015; DEMIROVIC E MUSLIU, 2016).

Este trabalho tem como foco o cenário brasileiro, onde é comum que os professores lecionem em mais de uma escola ao mesmo tempo, tornando necessário deixar sua grade a mais compacta possível. Outro fator importante que foi considerado são as aulas geminadas, onde o professor pode optar se deseja suas aulas em sequência ou não. Este é um subproblema do *high school timetabling* e é chamado de *Class-Teacher Timetabling Problem with Compactness Requeriments* (CTTPCR). (SOUZA E MACULAN, 2000; DORNELES *et al.*, 2014.; DEMIROVIC E MUSLIU, 2016).

Dorneles *et al*. (2014) adaptou cinco instâncias do XHSTT para o CTTPCR. Neste trabalho apresentamos dois modelos matemáticos baseados nos trabalhos de Dorneles *et al* (2014) e Veenstra e Vis (2016) utilizados para resolver estas instâncias buscando atender as preferências dos professores com relação a carga horária e aulas geminadas, e, e possível, eliminando períodos de ociosidade. Além destes, serão apresentados três métodos de busca em vizinhança que tem como objetivo evitar vizinhos que não possuam penalidades, evitando fazer buscas em regiões irrelevantes para o problema.

Como resultado, as abordagens apresentadas foram capazes de encontrar soluções superiores ao da literatura para duas das cinco instâncias estudadas e diminuir o tempo de busca ao comparadas ao método de busca básico do *fix-and-optimize*.

---

[1] http://www.utwente.nl/ctit/hstt/archives/XHSTT-2012/

1.1 Objetivo Geral

O objetivo geral deste trabalho é apresentar um modelo de programação inteira mista para resolver o problema de High School Timetabling e obter novas melhores soluções conhecidas para cinco instâncias da literatura, retiradas do XHSTT.

1.2 Objetivos Específicos

A fim de atingir o objetivo geral, os objetivos específicos são:

- Estudar as características dos problemas de Timetabling;
- Estudar as instâncias do XHSTT;
- Compreender e implementar o algoritmo de fix-and-optimize;
- Realizar testes computacionais na para identificar onde ocorre o maior gasto de tempo e propor novos métodos para resolução em menor tempo computacional;
- Desenvolver modelos matemáticos que atendam às necessidades das instituições.

1.3 Importância do Trabalho

A distribuição das turmas para os professores do ensino médio possui uma grande complexidade e precisa ser completamente reformulado cada vez que as preferências de qualquer um dos professores muda, justificando a necessidade de uma ferramenta ágil e que apresente boas soluções.

Como a grande maioria dos professores trabalham em mais de uma instituição, é necessário minimizar seu tempo de permanência na escola, assim como seus dias de trabalho, possibilitando melhores condições de trabalho. Já o atendimento das aulas geminadas ajuda no aprendizado e na forma como o professor expõe seu conteúdo.

Desta maneira os modelos apresentados otimizarão o problema em poucos minutos de processamento e ainda possibilitará que mudanças sejam facilmente realizadas durante o ano letivo.

1.4 Estrutura do Trabalho

Este trabalho encontra-se dividido em seis capítulos, incluindo esta introdução.

No segundo capítulo são apresentados o problema de *Timetabling* e *school timetabling,* assim como alguns dos principais trabalhos sobre o tema apresentados na literatura e uma breve descrição sobre o algoritmo de *fix-and-optimize.*

No terceiro capítulo são apresentados os dois modelos adaptados, suas variáveis e restrições.

No quarto capítulo é apresentado o algoritmo *fix-and-optimize* adaptado ao problema de *high school timetabling,* as três estratégias de decomposição e os cinco métodos de busca em vizinhança.

No quinto capítulo é apresentado os resultados obtidos para cada modelo, instância, decomposição e método de busca em vizinhança. Após será feita uma comparação entre os modelos com intenção de avaliar qual deles obteve o melhor desempenho.

Por fim, o sexto capítulo apresenta as principais conclusões sobre o estudo realizado.

2 . REVISÃO DE LITERATURA

Em 1954 o matemático George B. Dantzig aceitou o desafio de modelar o problema de grade horária de agentes rodoviários na taxação de pedágio do porto de Nova York, que até então havia sido resolvido por Edie (1954) utilizando técnicas estatísticas. Assim havia nascido um novo ramo de pesquisa dentro da Programação Linear chamada de *Timetabling*. No decorrer dos anos, o *timetabling* foi exaustivamente usado para resolver problemas de grade horária escolar e seus derivativos. Neste capitulo iremos apresentar os referencias teóricos usados na construção do *High School Timetabling* e os métodos utilizados para sua resolução.

2.1 O problema de *Timetabling*

A construção de grades horárias (*timetabling*) para instituições de ensino vem sido explorado desde a década de 1960. Atualmente, muitas meta-heurísticas têm surgido para resolução deste problema. (BURKE *et al.*, 2003).

O problema de *timetabling* possui quatro parâmetros: intervalos de tempo, recursos, reuniões e restrições. O problema consiste em unir um grupo de recursos e um intervalo de tempo para formar uma reunião (evento), buscando obedecer às restrições da melhor forma possível. (BURKE *et al.*, 2003).

Os parâmetros de intervalo de tempo são os períodos em que os eventos acontecem. Normalmente, é dividido em períodos de mesma duração, podendo ser dividido em períodos com tempos diferentes, porém, este é mais raro. Estes intervalos de tempo normalmente possuem sua duração pré-fixada, repetindo de forma cíclica em intervalos semanais, mensais, anuais, entre outras. Como exemplo, escolas e universidades repetem os intervalos de tempo semanalmente. Mas também existem alguns problemas acíclicos, como por exemplo vestibulares e outros exames de qualificação. (BURKE *et al.*, 2003).

Os recursos podem ser representados tanto por objetos, pessoas ou espaços físicos, como equipamentos especializados, professores, alunos e salas de aula. Estes recursos também são normalmente pré-fixados, e representam as variáveis com as quais desejamos fazer a designação do problema. Como restrição, os recursos nunca podem

ser designados juntos para a mesma reunião, o que seria equivalente a dois professores dando aula para uma mesma turma ou o fato de alguns recursos não estarem disponíveis em alguns intervalos de tempo, como professores que não podem trabalhar em certo dia da semana. (BURKE *et al*, 2003).

Uma reunião, ou um evento, é a união de recursos em um intervalo de tempo. Se uma reunião é definida de forma que satisfaça as restrições, então a designação destes recursos está completa. No *high school timetabling* a reunião seria a união de uma sala de aula, um grupo de alunos e um professor em um determinado período. (BURKE *et al*, 2003).

As restrições representam as limitações do problema e são separadas em restrições *hard* e *soft*. Sendo as restrições *hard* as restrições que modelam o problema e garantem factibilidade da solução e as restrições *soft* as que representam as preferências sobre o problema e o cumprimento dessas restrições definem o nível de qualidade da solução (BURKE *et al*, 2003; DORNELES *et al,* 2014; DORNELES, 2015; DEMIROVIC E MUSLIU, 2016).

As restrições também podem ser divididas em três categorias: restrições básicas de agendamento, restrições sobre eventos e restrições sobre recursos. As restrições básicas de agendamento são aquelas que atribuem recursos e eventos a um determinado período ou que apresenta alguma preferência relacionando uma aula e um período. As restrições sobre os eventos são aquelas que limitam o número de vezes que uma reunião pode se repetir durante o dia, uma carga horária máxima para cada reunião e a necessidade de algum evento ser geminado (duas ou mais reuniões de um mesmo evento em sequência). As restrições de recursos são as que evitam atribuir um recurso a mais de um evento ao mesmo tempo. Se os recursos estão disponíveis para serem usados, períodos de ociosidade e intervalos de tempo na carga horária, como intervalos para almoço ou descanso. (FONSECA e SANTOS, 2014).

Existem inúmeras variações do problema de *timetabling*. Estas variações diferem de acordo com os objetivos almejado e cada uma possui suas próprias características e restrições. Algumas destas variações são o *school timetabling, physician scheduling, sports timetabling, trasportation timetabling,* etc. (MIRHASSANI E HABIBI, 2013).

Há duas variações para o problema de geração de grade horária, chamados de *planning* e *scheduling* (*timetabling).* O *planning* busca identificar quais são as atividades necessárias para alcançar algum objetivo. Já o *scheduling* busca atribuir a atividade já conhecidas (aulas) aos recursos previamente disponíveis (professores e turmas) dentro de um intervalo de tempo. (BARTAK, 2002). O *timetabling* é o mesmo problema do *Scheduling* cujo a nomenclatura varia de acordo com a região e área de pesquisa.

2.2 *High School Timetabling* (HSTT)

O HSTT consiste em realizar a designação da grade horária escolar de ensino médio. Esta tarefa consiste em agendar professores, alunos e salas em um determinado período, buscando sempre encontrar a melhor grade horária que não prejudique nem alunos nem professores. (DE WERRA, 1971). A grade horária tem grande influência no sistema educacional pois define como serão ministradas as aulas durante todo o semestre e sua qualidade tem grande impacto sobre milhares de pessoas.

O problema é bastante complexo, existindo muitos parâmetros a serem avaliados como disponibilidade, preferência e o balanceamento da carga horária dos professores, capacidade das salas de aula e disciplinas por turma. (VEENSTRA E VIS, 2016). O problema geral do HSTT é classificado como NP-completo. (EVEN *et al*., 1975).

As restrições do HSTT podem variar para cada instituição. Devido a este fato, o trabalho dos pesquisadores acabam ficando restritos a uma única instituição, ou país, fazendo com que um método que se mostre efetivo para um problema não tenha o mesmo desempenho se aplicado em uma instância diferente (DREXL E SALEWSKI, 1997; SCHAERF, 1999). Desta forma, o HSTT é apenas um dos três ramos do chamado *Educational Timetabling Problems*, sendo que os outros dois são o *Course Timetabling Problem* (Universidades) e o *Examination Timetabling Problem* (exames admissionais). (SKOULLIS *et al*., 2017). Portanto, muitas heurísticas foram testadas obtendo boas soluções para casos gerais (SCHAERF, 1999).

Muitas formulações e variações deste problema já foram apresentadas na literatura, contendo variações tanto referente a nação como pela própria instituição que pode possuir suas próprias restrições quanto a carga horária. Muitos métodos heurísticos e exatos foram propostos, sendo os heurísticos os preferidos pelos pesquisadores por

obterem bons resultados em pouco tempo se comparado aos métodos exatos, porém, estes não garantem o resultado ótimo. (DORNELES *et al*, 2014; FONSECA E SANTOS, 2014; DEMIROVIC E MUSLIU, 2016).

Devido as muitas pesquisas realizadas sobre o HSTT, criou-se uma competição para os pesquisadores da área chamada *Third International Timetabling Competition* (ITC-2011) (DEMIROVIC E MUSLIU, 2016). Para padronizar a formatação dos dados usados pelos pesquisadores, foi proposto uma formulação geral para o HSTT chamada XHSTT e foi adotada pelo ITC-2011 (POST *et al.*, 2011).

Após a ITC-2011 as pesquisas na área têm se intensificado. Vários trabalhos utilizando parte ou todas as instâncias da competição surgiram, propondo novas abordagens e heurísticas para solucionar os problemas o mais rápido possível. (DORNELES *et al.*, 2014; KRISTIANSEN *et al.*, 2014; DORNELES, 2015), FONSECA *et al.*, 2015; DEMIROVIC E MUSLIU, 2016; DORNELES *et al.*, 2017).

Dentre os métodos mais usados na competição estão os métodos de busca em vizinhança onde o *Simulated Annealing* tem se destacado como uma excelente ferramenta de solução e alcançando os melhores resultados. (FONSECA E SANTOS, 2014).

2.3 Class-Teacher Timetabling Problem with Compactness Requeriments (CTTPCR)

Em muitos países é comum que professores lecionem em mais de uma escola, o que faz com que não estejam indisponíveis em certos períodos. Por isto, é essencial que as aulas de cada professor sejam ministradas no menor número de dias possíveis, eliminando seus períodos de ociosidade. Outro fator importante para um bom aprendizado é que disciplinas que possuem mais de uma aula no mesmo dia sejam ministradas em sequência, permitindo que os professores explorem melhor o tempo que possuem com cada turma. Este problema especifico é denominado *Class-Teacher Timetabling Problem with Compactness Requeriments* (CTTPCR) (SOUZA E MACULAN, 2000; DORNELES *et al.*, 2014; DEMIROVIC E MUSLIU, 2016). O problema é classificado como um problema NP-completo. (EVEN *et al.,* 1975).

O CTTPCR é um problema originário do Brasil. O problema foi abordado por Souza e Maculan (2000) que utiliza um algoritmo de busca local (Local Search) aliado a algumas metaheurísticas, como Busca Tabu, GRASP, Simulated Annealing, entre outras. Seu método consiste em uma técnica que, partindo de uma solução inicial, gera soluções melhores ao detectar ciclos com custo negativo nos grafos associados ao quadro horário de cada turma. Mais tarde, Santos *et al.* (2012) apresentam um algoritmo de corte e geração de colunas para resolver o CTTPCR. Sua formulação foi capaz de encontrar *lower bounds* menores para o problema estudado.

2.4 Trabalhos Relacionados

Schmidt e Strohlein (1979) apresenta uma heurística construtiva, baseada na simulação manual de resolver o problema de *school timetabling*. Esta técnica consiste do preenchimento gradual da grade horária, onde cada aula vai sendo atribuída por vez. Esse processo continua até que todas as aulas tenham sido preenchidas ou que haja um conflito de horários. Sempre que ocorre um conflito, as aulas já designadas são rearranjadas de maneira que o conflito desapareça. A ideia principal da heurística é sempre alocar as aulas mais difíceis por primeiro, sendo que o conceito de uma aula 'difícil' pode variar de acordo com os objetivos e restrições do problema.

Tripathy (1984) apresentou uma técnica de relaxação lagrangeana para o problema de *Course Timetabling*. A técnica consiste em um método de otimização por subgradientes combinado com o método de *branch-and-bound*. O objetivo do problema é agendar certa matéria de um curso em determinado horário especifico, relaxando as restrições de espaço que impedem estudantes que possuam um mesmo currículo a assistirem a mais de uma aula em um mesmo horário e as anexando a função objetivo. A técnica foi aplicada a um programa de pós-graduação com 900 disciplinas.

De Werra (2003) apresenta uma análise teórica sobre alguns modelos básicos de *timetabling* aplicados ao *Class-Teacher Timetabling* para encontrar formas de classificar os problemas como pequeno, médio e grande porte. Para isso faz-se uma analogia aos problemas de tomografia e reconstrução de imagem. Por fim, o autor foi capaz de encontrar algumas características simples e concluindo que problemas pequenos podem ser resolvidos em tempo polinomial.

Zhang *et al.* (2010) resolve o HSTT usando o *Simulated Anealing* e propõe uma nova estratégia de busca em vizinhança para a heurística. Sua estratégia de busca executar uma sequência de trocas de pares durante dois períodos, acelerando o processo de convergência do *Simulated Anealing*. Como característica do problema temos que o número de aulas dos professores devem ser distribuídas de forma equilibrada durante a semana, não podendo dar mais de uma aula por dia e impedido que professores tenham períodos de ociosidade. Como resultado a nova abordagem se mostrou bem competitiva ao comparada com métodos da literatura, aumentando a eficiência e o desempenho do *Simulated Anealing*.

Sorensen e Dahms (2014) apresentam uma decomposição em dois estágios adaptando um problema de *Curse Timetabling Problem* para o HSTT, dando foco ao sistema educacional dinamarquês. No primeiro estágio o modelo foca em atribuir as aulas em determinado período. Está solução parcial serve como entrada para o estágio dois em que as salas são atribuídas as aulas com períodos já fixados, gerando a solução para o problema original. A desvantagem desta abordagem é que o resultado obtido no estágio um não pode ser modificado no estágio dois, o que pode impedir uma alocação ótima. No entanto, ao adaptar algumas restrições *soft* é possível fazer com que a atribuição de salas as aulas já fixadas podem ser feitas de maneira implícita, sendo que se fosse possível fazer está atribuição a todas as aulas, a solução obtida no estágio um seria exata.

Porém, realizar a atribuição de todas as salas implicitamente é infactível, portanto o estágio um retorna apenas um *lower bound* do problema. Além disso, dividir o problema em dois estágios diminuiu o número de variáveis do problema, deixando claro que a abordagem possui mais vantagens do que desvantagens. Para validar o método os autores usam 100 instâncias reais do sistema de educação dinamarquês. Os resultados foram promissores superando os métodos exatos em tempo e *lower bounds*, encontrando bons resultados para todas as 100 instâncias sendo 18 delas soluções ótimas.

Dorneles *et al.* (2014) apresentam uma nova abordagem aplicando o algoritmo *Fix-and-Optimize* juntamente ao método de *Variable Neighborhood Descent* para solucionar doze instâncias, sendo sete delas da ITC-2011 e as outras cinco provenientes de adaptações das instâncias anteriores para o caso do CTTPCR. Das doze instâncias

estudadas, ele foi capaz de encontrar sete das melhores soluções conhecidas, sendo que três destas são novas melhores soluções conhecidas. Posteriormente, Dorneles *et al.* (2017) utilizam a modelagem de *multicommodity flow* e o método de geração de colunas para encontrar rapidamente fortes *lower bounds* para as instâncias, focando no CTTPCR. O método encontrou cinco novos *lower bounds* para as instâncias estudadas. Desta forma, duas das soluções encontradas por Dorneles *et al.* (2014) se mostraram soluções ótimas.

Já Fonseca e Santos (2014) fazem um estudo do HSTT usando a metaheurística de *Variable Neighborhood Search* e três variações deste método, incentivado pelo alto desempenho obtido por elas, em especial o *Simulated Annealing* no ITC-2011. As três variações foram as seguintes: *Reduced Variable Neighborhood Search, Sequential Variable Neighborhood Descent* e *Skewed Variable Neighborhood Search*. Por fim, os autores conseguiram resultados superiores aos descritos na literatura e apontam duas vantagens do método em relação ao *Simulated Annealing*: o *Variable Neighborhood Search* possui menos parâmetros e estes parâmetros não são sensíveis ao tamanho do problema.

Posteriormente, Fonseca et al (2016) apresentou uma metaheurística hibrida composta por três partes: primeiro, utiliza o solver *Kingston High School Timetabling* (KHE) para gerar uma solução inicial para o problema. Por segundo, faz uma busca em vizinhança usando o *Skewed Variable Neighborhood Search* tendo como base seis possíveis movimentos. Por último, utiliza o solver GUROBI para resolver o modelo matemático de cada vizinhança até que um tempo limite seja atingido. Atualizando a função objetivo a cada iteração. O modelo foi capaz de encontrar novas melhores soluções para 15 das 17 instâncias testadas.

Demirovic e Musliu (2016) propõem um algoritmo de *MaxSAT-based Large Neighborhood* para solucionar o HSTT. Este algoritmo direciona uma solução inicial até um ótimo local e então realiza uma poderosa técnica de pesquisa de vizinhança. O algoritmo encontra ótimos locais e então destrói parte da solução fazendo uma busca na vizinhança, semelhante ao algoritmo genético. Para testar o método foram utilizadas 27 instâncias da ITC-2011. O método se mostrou veloz e foi capaz de encontrar novas melhores soluções para 4 das instâncias estudadas.

Veenstra e Vis (2016) propõe um modelo de HSTT aplicado ao modelo educacional dos Países Baixos. Neste problema, o número de períodos de cada dia não é fixo, podendo haver até nove períodos de aula em um dia. Desta forma os autores propõem um modelo de substituição, onde os períodos do dia podem ser rearranjados caso um professor precise faltar, de modo que as classes não possuam períodos de ociosidade e repondo a aula com o professor faltante em outro dia da semana. O modelo se baseia em um sistema de troca, que busca alocar as aulas dos professores faltantes no primeiro ou último período em que a classe possui aula no dia, buscando realizar o menor número de trocas possíveis. Para a solução do problema é proposto uma heurística de fixação, onde o problema é reduzido em problemas menores para que se possam ser otimizados através de um solver, semelhante a heurística de *fix-and-optimize*. Como resultado foi possível eliminar de 77,6% a 97,2% dos períodos de ociosidade.

Saviniec *et al* (2017) busca utilizar metaheurísticas em paralelo para a solução do HSTT em que diferentes métodos de solução são aplicados simultaneamente em diferentes núcleos do processador. As metaheurísticas utilizadas pelos autores foram busca tabu, *simulated annealing* e *late acceptance strategy*. Para o controle das atividades paralelas são adotadas duas estratégias. A primeira estratégia aplica metaheurísticas diferentes paralelamente a uma mesma solução inicial e após um período de execução o processo é finalizado, trocando as metaheurísticas usadas em cada solução. O novo método que será aplicado a determinada solução encontrada irá depender de parâmetros pré-estabelecidos no que é chamado de memória central, podendo permanecer o mesmo método inicial. A segunda estratégia é semelhante a primeira, porém, esta estratégia guarda soluções mais fracas do problema para ser usada como diversificação para as soluções de elite do problema, possibilitando escapar de ótimos locais. Como resultado os métodos foram capazes de diminuir o tempo computacional de sete instancias da literatura calculadas até a otimalidade, validando seu modelo.

2.5 Fix-And-Optimize

Uma maneira de lidar com problemas grandes é utilizar métodos heurísticos que são capazes de encontrar boas soluções rapidamente, mas não são capazes garantir o ótimo global do problema. O fix-and-optimize é uma heurística que reduz o tamanho do modelo através da fixação de uma parte das variáveis originais, de modo que é possível resolver o problema reduzido na otimização. (DORNELES *et al*, 2015)

Está heurística consiste em decompor o problema iterativamente em subproblemas menores. Nos subproblemas gerados, uma grande parte das variáveis do modelo tem seus valores fixados, enquanto as demais são livres para serem otimizadas. Desta forma, os subproblemas podem ser resolvidos com a utilização de *solvers* de programação linear. A nova solução obtida passa a ser a solução geral caso possua uma função objetivo melhor do que a solução atual do problema completo. Este processo é repetido, selecionando um novo grupo de variáveis para ficarem livres a cada iteração, até que uma condição de parada seja satisfeita. (DORNELES, 2015).

3 . DESCRIÇÃO DOS MODELOS

Nesta seção apresenta-se uma variação do modelo apresentado por DORNELES *et al.* (2014), onde é proposto um modelo para o CTTPCR e uma alternativa para as restrições de controle de período de ociosidade proposto por VEENSTRA e VIS (2016).

3.1 Introdução

O problema é definido para um conjunto de classes C e um conjunto de professores T. Uma classe $c \in C$ é um grupo de alunos pertencentes a um mesmo curso e que seguem o mesmo cronograma de lições. O objetivo do problema é construir uma grade horária semanal que é dividida em um conjunto de dias D, cada dia separado em um conjunto de períodos P. Cada período letivo é a combinação entre o dia e o período de uma determinada classe (d, p), com $d \in D$ e $p \in P$, onde cada período tem a mesma duração. Outra característica do problema é que o professor T pode não estar disponível em certos períodos (DORNELES *et al*, 2014).

Como entrada para o algoritmo é dado um conjunto de eventos E, que são as aulas que um professor e uma classe possuem em determinada sala. No Brasil é comum que estes eventos já sejam pré-definidos, com cada professor, classe, sala e carga horária já agendados, faltando apenas designá-los ao devido período. Cada evento possui uma carga de trabalho máxima diária, sendo que em um dia pode existir mais de uma aula sendo ministrada pelo mesmo professor em uma mesma classe. Quando isso ocorre, estas lições devem ser consecutivas e são chamadas lições duplas ou geminadas. (DORNELES et al, 2014). Nas instâncias estudadas neste trabalho, vamos considerar um número máximo de aulas duplas para cada evento.

Para que haja factibilidade na solução, DORNELES *et al* (2014) propõe que cumpramos os seguintes requerimentos (restrições *hard*):

H1 A carga horária do evento deve ser respeitada.

H2 Um professor não pode ser agendado em mais de uma aula em um mesmo período.

H3 Duas aulas não podem ser agendadas a uma mesma classe em um mesmo período.

H4 Um professor não pode ser agendado em um período em que ele não está disponível.

H5 O número máximo de aulas diárias de cada evento deve ser respeitado.

H6 Duas aulas de um mesmo evento devem ser consecutivas se agendadas para um mesmo dia, caso seja requerido.

Além destas, os seguintes requerimentos devem ser cumpridos sempre que possível (restrições *soft*):

S1 Evitar períodos de ociosidade dos professores.

S2 Minimizar o número de dias trabalhados pelos professores, onde o dia trabalhado é aquele em que o professor tem pelo menos uma aula no dia.

S3 Buscar cumprir o número mínimo de aulas duplas exigidas para cada evento.

3.2 Modelo 1

Em DORNELES *et al* (2014) propôs-se pela primeira vez um modelo que trata o requerimento H6 como restrição *hard*. Esta abordagem, que também contempla a restrição *soft* S1, trouxe resultados de relevantes em pouco tempo computacional.

TABELA 1 – NOTAÇÃO USADA PARA A CONSTRUÇÃO DO MODELO 1.

Símbolo	Definição
Conjuntos	
$d \epsilon D$	Dias da semana
$p \epsilon P$	Períodos do dia
p'	Conjunto de períodos onde $p' = \|p\| - 2$
$m \epsilon P$	Origem do arco z_{tdmn}
$n \epsilon P$	Fim do arco z_{tdmn}
$t \epsilon T$	Conjunto de professores
$c \epsilon C$	Conjunto de classe
$e \epsilon E$	Conjunto de eventos

E_t	Conjunto de eventos designados ao professor t
E_c	Conjunto de eventos designados a classe c
U	Conjunto de tuplas (m, n) para p', $n \epsilon P : n \geq m + 2$
Q	Conjunto de tuplas (m, n) para p', $n \epsilon P : n \geq m$
SG_e	Conjunto de períodos em que cada evento e pode começar uma aula geminada $\left(SG_e = \{(d,p): d \epsilon D, p \epsilon P \; e \; p < \lvert P \rvert, \; V_{edp} + V_{edp+1} = 2\}\right)$. O parâmetro V_{edp} será definido em seguida.

Parâmetros

ω_t	Custo de cada período de ociosidade do professor t
γ_t	Custo do dia de trabalho do professor t
δ_e	Custo de cada lição geminada do evento e não dada sequencialmente
R_e	Carga de trabalho do evento e
L_e	Número máximo de aulas que podem ser dadas por dia no evento e
V_{edp}	Parâmetro binário que indica se o professor designado para o evento e está disponível no período (d, p)
MG_e	Quantidade mínima de aulas geminadas requisitadas para o evento e
LB	Mínimo de dias trabalhados no problema-teste atual

Variáveis

x_{edp}	Variável binária que indica se o evento e foi designado para o período (d, p)
y_{td}	Variável binária que indica se o professor t foi escalado para ao menos uma aula no dia d
g_{edp}	Variável binária que indica se o evento e possui uma aula geminada começando no período (d, p)
G_e	Variável inteira que indica o número de aulas geminadas faltam para ser atendidas
b_{edp}	Variável binária que indica se o evento e possui uma aula no período (d, p) e não no período $(d, p-1)$
z_{tdmn}	Variável binária que indica se o professor t possui períodos de ociosidade no dia d entre o período m e n

FONTE: ADAPTADO DE DORNELES ET AL (2014)

O modelo a seguir foi desenvolvido por DORNELES *et al.* (2014). Também iremos incluir ao modelo a restrição (19), onde minimizamos previamente as instâncias para encontrar o número mínimo de dias trabalhados pelos professores, e então usamos este valor como parâmetro, diminuindo a área de busca para o algoritmo.

Min $$\sum_{t \in T} \sum_{d \in D} \sum_{(m,n) \in U} \omega_t (n - m - 1) z_{tdmn} + \sum_{t \in T} \sum_{d \in D} \gamma_t y_{td} + \sum_{e \in E} \delta_e G_e \quad (1)$$

Sujeito à

$$\sum_{d \in D} x_{edp} = R_e \quad \forall\, e \quad (2)$$

$$\sum_{p \in P} x_{edp} = L_e \;\; \forall\, e, d \quad (3)$$

$$x_{edp} \leq V_{edp} \;\; \forall e, d, p \quad (4)$$

$$\sum_{e \in E_t} x_{edp} \leq y_{td} \;\; \forall\, t, d, p \quad (5)$$

$$\sum_{e \in E_t} \sum_{p \in P} x_{edp} \geq y_{td} \;\; \forall\, t, d \quad (6)$$

$$\sum_{e \in E_c} x_{edp} \leq 1 \;\; \forall\, c, d, p \quad (7)$$

$$b_{edp} \geq x_{edp} - x_{edp-1} \;\; \forall\, e, d, p : p > 1 \quad (8)$$

$$\sum_{p \in P: p>1} b_{edp} + x_{ed1} \leq 1 \;\; \forall\, e, d \quad (9)$$

$$g_{edp} \leq x_{edp} \;\; \forall e, (d,p) \in SG_e \quad (10)$$

$$g_{edp} \leq x_{edp+1} \;\; \forall\, e, (d,p) \in SG_e \quad (11)$$

$$G_e \geq MG_e - \sum_{(d,p) \in SG_e} g_{edp} \;\; \forall\, e \quad (12)$$

$$\sum_{d \in D} y_{td} \geq \max\left\{ \left\lceil \frac{\sum_{e \in E_t} R_e}{|P|} \right\rceil, \max_{e \in E_t} \left\{ \left\lceil \frac{R_e}{L_e} \right\rceil \right\} \right\} \;\; \forall\, t \quad (13)$$

$$\sum_{(m,n) \in Q} z_{tdmn} = y_{td} \;\; \forall\, t, d, m \in P : m \leq 3 \quad (14)$$

$$\sum_{(m,n)\,\epsilon\,Q} z_{tdmn} \leq y_{td} \quad \forall\, t,d,n\,\epsilon\,P : n \geq 3 \tag{15}$$

$$z_{tdpp} \leq 1 + \sum_{e\,\epsilon\,E_t} \left(x_{edp+1} - x_{edp}\right) \quad \forall t,d,p\,\epsilon\,P \tag{16}$$

$$z_{tdmn} \leq 1 - \sum_{e\,\epsilon\,E_t} x_{edn} \quad \forall t,d,(m,n)\epsilon U \tag{17}$$

$$z_{tdmn} \leq \sum_{e\,\epsilon\,E_t} x_{edn} \quad \forall t,d,(m,n)\,\epsilon\,U \tag{18}$$

$$\sum_{t\,\epsilon\,T} \sum_{d\,\epsilon\,D} y_{td} = LB \tag{19}$$

$$x_{edp}, b_{edp}\,\epsilon\,\{0,1\}, g_{edp}, G_e \geq 0 \quad \forall e,d,p \tag{20}$$

$$y_{td} \geq 0, z_{tdmn}\,\epsilon\,\{0,1\} \quad \forall t,d,(m,n)\,\epsilon\,Q \tag{21}$$

A função objetivo (1) é compostas de três partes, que representam as três restrições soft do problema (S1), (S2) e (S3), respectivamente, sendo que a restrição (S1) é proporcional pelo número de períodos de ociosidade dos professores.

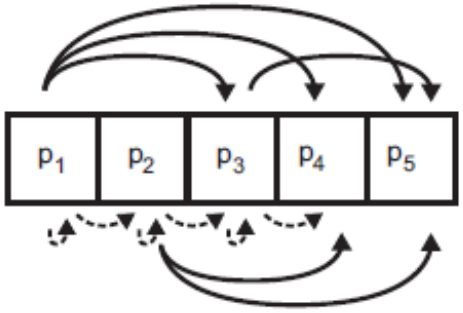
O conjunto de restrições (2) garante que a carga de trabalho de cada evento seja agendada. O conjunto de restrições (3) proporciona o limite diário de aulas para cada evento. O conjunto de restrição (4) determina se as aulas serão agendadas em períodos disponíveis. O conjunto de restrições (5) e (7) garantem que professores e classes estejam agendados somente para uma única aula por vez. O Conjunto de restrições (5) e (6) os dias em que cada professor trabalha. O conjunto de restrições (8) e (9) garantem que as aulas de um evento sejam agendadas sequencialmente de acordo com a restrição (H6). O conjunto de restrições (10) e (11) força aulas geminadas quando é existe a possibilidade de aulas duplas iniciando no período p. O conjunto de restrição (12) determina G_e, variável que controla quantas aulas geminadas faltam para alcançar MG_e. A variável G_e está presa a terceira parte da função objetivo, sendo que o lado direito da inequação tende a aumentar, garantindo o atendimento de aulas geminadas. O conjunto de restrições (13) é um corte que define o número mínimo de dias trabalhados para cada professor.

Os conjuntos de restrições de (14) a (18) foram propostos por Dorneles et al. (2014) para o controle e minimização dos períodos de ociosidade dos professores. Para isto foram considerados arcos para cada dia de trabalho de um professor. Estes arcos, com $(m,n)\ \epsilon\ U$, são penalizados na função de acordo com o número de períodos de ociosidade, representados pelas variáveis z_{tdmn} onde m é a calda de cada arco e n é a cabeça do arco, ou seja, representa qual foi o intervalo entre aulas de cada professor. Os conjuntos de restrições (14) e (15) garantem que existam apenas um arco deixando e chegando em cada período, podendo ser o início ou fim de um período de ociosidade. Para representar os períodos onde não há ociosidade foram criados arcos auxiliares com $(m,n)\ \epsilon\ Q\backslash U$. Estes arcos auxiliares são usados em apenas dois casos: quando o professor não possui aula no período m, ou quando o professor possui aula no período m e $m+1$. O controle sobre estes arcos auxiliares é feito pelo conjunto de restrições (16). O conjunto de restrições (17) garantem que o arco auxiliar $(m,m+1)$ só seja ativado quando a ultima aula do professor seja aplicada no período m. O conjunto de restrição (18) garante que um arco de período de ociosidade (m,n) seja ativado apenas se o professor possui aula no período n.

A restrição (19) é uma adaptação proposta neste trabalho. Ela é calculada através do modelo original, mas considerando apenas a segunda parte da função objetivo, referente a (S2), para encontrar LB, parâmetro que representa o número mínimo de dias que devem ser trabalhados pelos professores. Este parâmetro coincide com os melhores valores de *lower bounds* encontrados na literatura e *será* usado na restrição proposta com o objetivo de reduzir o espaço de busca.

Na Figura 1 vemos o grafo de períodos de ociosidade onde as linhas continuas representam os arcos de ociosidade e as linhas tracejadas os arcos auxiliares.

FIGURA 1 – GRÁFICO DOS PERÍODOS DE OCIOSIDADE



FONTE: DORNELES *ET AL* (2014)

3.3 Modelo 2

Como opção, as restrições de (14) a (18) para o controle dos períodos de ociosidade temos o modelo proposto por VEENSTRA E VIS (2016).

TABELA 2 – NOTAÇÃO USADA PARA A CONSTRUÇÃO DO MODELO 2.

Símbolo	Definição
Variáveis	
h_{td}	Número inteiro de períodos de ociosidade do professor t no dia d
$\bar{a}_{td}$	Número inteiro que indica qual foi o período da primeira aula do professor t no dia d
$\underline{a}_{td}$	Número inteiro que indica qual foi o período da última aula do professor t no dia d

FONTE: ADAPTADO DE VEENSTRA & VIS (2016).

O modelo completo proposto por VEENSTRA E VIS (2016) foi aplicado em um problema de HSTT de uma escola dos países baixos. Porém, é utilizado apenas as restrições relacionadas ao controle dos períodos de ociosidade, juntamente com as restrições de (1) a (13) do modelo apresentado por DORNELES *et al* (2014). A função objetivo também precisou ser modificada pois neste segundo modelo não utilizaremos as variáveis z_{tdmn}, e sim as variáveis h_{td}. Devido a estas modificações, foram necessárias pequenas adaptações nas restrições de VEENSTRA E VIS (2016), as quais podem ser apresentadas da seguinte forma:

$$\text{Min} \quad \sum_{t \in T} \sum_{d \in D} \omega_t h_{td} + \sum_{t \in T} \sum_{d \in D} \gamma_t y_{td} + \sum_{e \in E} \delta_e G_e \tag{22}$$

$$\overline{a}_{td} \leq (|P| + 1) - (|P| + 1 - p) \sum_{e \in E_T} x_{edp} \quad \forall\, t \in T, d \in D, p \in P \tag{23}$$

$$\underline{a}_{td} \geq p \sum_{e \in E_T} x_{edp} \quad \forall\, t \in T, d \in D, p \in P \tag{24}$$

$$h_{td} \geq \underline{a}_{td} - \overline{a}_{td} + 1 - \sum_{p \in P} \sum_{e \in E_T} x_{edp} \quad \forall\, t \in T, d \in D \tag{25}$$

A expressão (22) é a adaptação de (1) para o novo conjunto de restrições. As restrições (23) e (24) definem o primeiro e o último período de aula de cada professor em cada dia. A restrição (23) apresenta $|P|$ que representa o número de períodos do dia, neste trabalho sempre será cinco ($|P| = 5$). A restrição (25) define a quantidade de períodos de ociosidade de cada professor em cada dia.

4 . *FIX-AND-OPTIMIZE* COM BUSCA DE REDUÇÃO DE SUBPROBLEMAS

Neste capítulo iremos apresentar a heurística de *fix-and-optimize* adaptado ao problema do CTTPCR. Também apresentaremos dois métodos de busca em vizinhança com redução de subproblemas, usadas com a intenção de diminuir o número de iterações desta heurística reduzindo o tempo computacional.

4.1 Introdução

Os problemas de CTTPCR são complexos e com um grande número de variáveis (DORNELES *et al.,* 2014) sendo que, para problemas de médio e grande porte, se torna difícil encontrar soluções factíveis usando *software* de programação inteira mista. Buscando superar este problema será usada a heurística de *fix-and-optimize*.

As variáveis a serem fixadas a cada iteração são as variáveis de decisão x_{edp}, pois todas as outras são dependentes desta. A escolha da quantidade e de quais variáveis serão fixadas variam em cada subproblema, segundo a decomposição escolhida. Neste trabalho, foram consideradas decomposições de classe e professores, denotadas por CD e TD respectivamente. Na decomposição de classes, um certo número de classes é fixado possibilitando a otimização das classes restantes. De forma semelhante é realizada a decomposição por professores. (DORNELES *et al.*, 2014)

Considerando k como o parâmetro que indica a cardinalidade do conjunto de variáveis a serem liberadas em cada subproblema, é possível criar um conjunto de vizinhanças $\mathcal{N}$ que consiste em combinar uma decomposição do tipo τ com um certo número k (HANSEN, 2001). Combinando as vizinhanças, obtemos uma sequência de vizinhanças a ser seguida pelo algoritmo. Por exemplo, a sequência (CD,2), (TD,2), ... , (CD,|C|),(TD,|T|) consiste em liberar inicialmente duas classes e, em seguida, dois professores, incrementando o número de classes e professores livres até atingir o número total desses conjuntos (DORNELES *et al.*, 2014).

O primeiro passo do algoritmo consiste em obter uma solução factível para o problema. Esta solução inicial pode ser obtida pela resolução do modelo sem considerar

a função objetivo. Caso não seja possível encontrar uma solução inicial factível, o algoritmo é finalizado e retorna uma solução nula.

A partir da solução inicial, o algoritmo percorre todas as vizinhanças definidas *a priori*. Para cada vizinhança considerada, o problema é decomposto em um número finito de subproblemas. Cada subproblema tem o tempo de resolução limitado em 30 segundos, conforme descrito em Dorneles *et al.*, (2014). Se, após a otimização do subproblema, o valor da função objetivo for melhor do que a solução anterior então a solução é atualizada e a variável $SemMelhorias$ é zerada. Caso contrário, $SemMelhorias$ é incrementada (QUADRO 1). Esses passos devem ser repetidos até que a quantidade de subproblemas sem melhorias seja igual a quantidade de subproblemas possíveis na vizinhança. Como não há garantia que a heurística nos retorne o valor ótimo do problema, define-se um critério de parada de execução do algoritmo. Neste trabalho, o critério de parada é o tempo limite (TL) e varia conforme a instância estudada de acordo com os melhores tempos encontrados na literatura (DORNELES *et al.*, 2014).

QUADRO 1 – PSEUDO CÓDIGO DA HEURÍSTICA *FIX-AND-OPTIMIZE*

Algoritmo FixAndOptimize(ℕ, TL)
1. $x^* \leftarrow SoluçãoInicial$
2. **Se** $x^* = \emptyset$ **então**
3. Retorna ∅
4. **Fim**
5. **Para todo** $(\tau, k) \in$ ℕ **faça**
6. $count \leftarrow QtdeSubproblemas(\tau, k)$
7. $s \leftarrow 1$
8. $SemMelhorias \leftarrow 0$
9. **Repita**
10. ℝ $\leftarrow Decompor(\tau, k, s)$
11. $x \leftarrow Resolver(x^*, \mathbb{R})$
12. **Se** x é $melhor\ do\ que\ x^*$ **então**
13. $x^* \leftarrow x$
14. $SemMelhorias \leftarrow 0$
15. **Senão**
16. $SemMelhorias = SemMelhorias + 1$
17. **Fim**
18. **Se** $TL\ foi\ atingido$ **então**
19. Retorna x^*
20. **Fim**
21. $s \leftarrow (s\ \boldsymbol{mod}\ count) + 1$

22. **até** $SemMelhorias = count$
23. **Fim**
24. **Retorna** x^*

FONTE: DORNELES ET AL (2014).

O Quadro 2 decompõe os subproblemas de forma crescente para evitar que exploremos a mesma vizinhança várias vezes, permitindo que um ciclo completo de exploração seja realizado. Portanto, R sempre será construído a partir de $\mathrm{R} = \{\mathrm{R}_1\}$, que é quando a cardinalidade $\tau = 1$, passando para $\mathrm{R} = \{\mathrm{R}_2, \mathrm{R}_1\}$ para $\tau = 2$, e assim por diante. A maior cardinalidade obtida neste trabalho será $\tau = 4$ para a instância A. Nenhumas das instâncias restantes completam o ciclo da cardinalidade $\tau = 3$.

QUADRO 2 – FUNÇÃO DE DECOMPOSIÇÃO

Algoritmo Decompor(τ, k, s)

1. **Caso** τ =CD
2. $\mathrm{R} \leftarrow \{x_{edp} : c \in subconjuntos(C, k, s), e \in E_c, d \in D, p \in P\}$
3. **Caso** τ =TD
4. $\mathrm{R} \leftarrow \{x_{edp} : t \in subconjuntos(T, k, s), e \in E_t, d \in D, p \in P\}$
5. **Retorna** R

FONTE: DORNELES ET AL (2014).

QUADRO 3 – FUNÇÃO QUE CALCULA O NÚMERO DE SUBPROBLEMAS DE UMA VIZINHANÇA

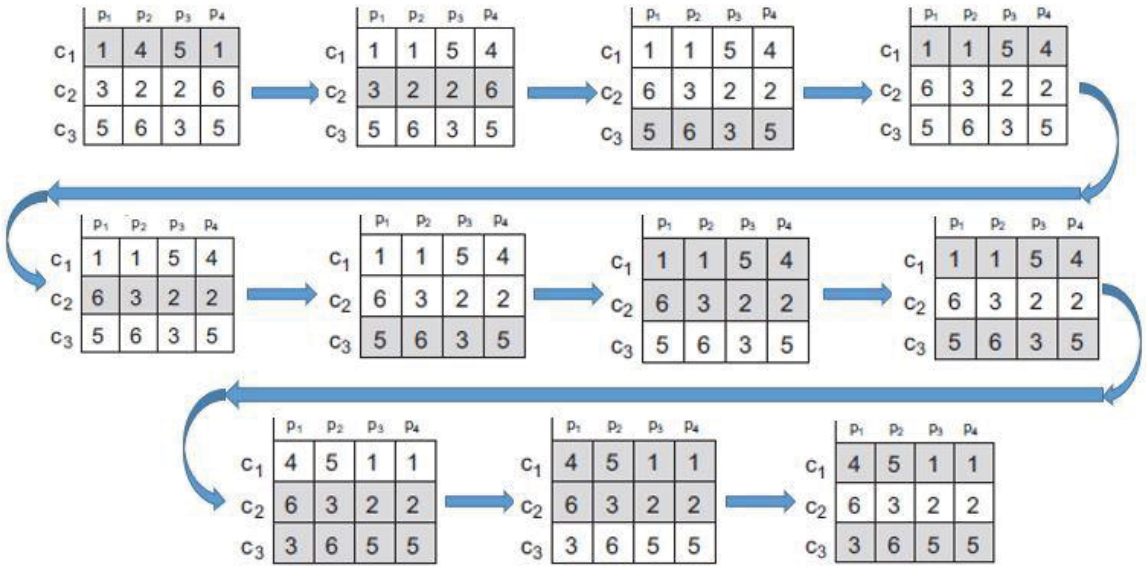
Algoritmo QtdeSubproblemas(τ, k)

1. **Caso** τ =CD
2. $count \leftarrow \binom{|C|}{k}$
3. **Caso** τ =TD
4. $count \leftarrow \binom{|T|}{k}$
5. **Retorna** $count$

FONTE: DORNELES ET AL (2014).

O exemplo a seguir ilustra o processo iterativo do algoritmo.

FIGURA 2 – EXEMPLO DO COMPORTAMENTO DA HEURISTICA DE *FIX-AND-OPTIMIZE* APLICADO AO HSTT



FONTE: ADAPTADO DE DORNELES *ET AL* (2014)

Na figura 2 temos um exemplo do *fix-and-optimize* aplicado ao problema de HSTT onde as linhas representam as classes, as colunas representam os períodos e os valores representam os professores. O exemplo se refere a uma decomposição por classe (CD) e inicia-se fixando as classes 2 e 3, C2 e C3, mantendo apenas a classe 1, C1, livre para otimização, que como pode ser visto no segundo quadro eliminou os períodos de ociosidade do professor 1. Então fixa C1 e passa para C2, onde elimina os períodos de ociosidade dos professores 3 e 6 passando para C3, onde não foi capaz de realizar melhorias. Então o algoritmo retorna a C1 e C2, dando continuidade ao processamento. Ao chegar em C3 completasse um ciclo sem melhorias, aumentando a cardinalidade, k, para 2. Então C1 e C2 ficam livres para otimização, onde não é realizada melhorias, e em seguida C1 e C3 onde os períodos de ociosidade do professor 5 é eliminado. Então realizasse mais um ciclo sem melhorias encerrando o algoritmo.

4.2 Estratégia de redução de Subproblemas

Nas vizinhanças em que $k \geq 3$, um grande número de variáveis é liberado, causando assim um aumento no tempo computacional de resolução dos subproblemas.

Em alguns casos, o *solver* tem dificuldade de encontrar uma solução factível, levando vários segundos. Dado que o critério de parada do algoritmo é o tempo total atingido, é desejável resolver o maior número de subproblemas em que se tenha melhorias e evitar subproblemas que não tragam ganho no valor da função objetivo. Por exemplo, quando é feita a decomposição de professores, todas as restrições *soft* influenciam na decisão, já para a decomposição por classe apenas as restrições referentes a aulas geminadas influenciam na tomada de decisão, pois dias de trabalho e períodos de ociosidade são variáveis diretamente relacionada aos professores. Com a utilização da restrição (19), o número de dias trabalhados por cada professor é fixado previamente e, neste caso, o objetivo se reduz em minimizar o número de janelas entre as aulas e o número de geminadas não atendidas. Caso os professores a serem liberados não possuam nenhuma destas duas penalidades, não há melhoria a ser obtida no valor da função objetivo e a resolução do subproblema pode ser evitada.

Para economizar tempo computacional, foram aplicados dois procedimentos, um para verificar a existência de períodos de ociosidade dos professores e outro para verificar a existência de aulas geminadas não atendidas. Caso um subproblema seja evitado por não possuir possíveis melhorias, o próximo subproblema deverá conter ao menos um professor que receba penalização.

O estudo é realizado de três formas separadas, na primeira evitaremos os subproblemas relacionados aos professores que não possuam períodos de ociosidade, a segunda evitando os subproblemas que não possuam eventos com aulas geminadas a serem atendidas e o terceiro subproblemas que não possuam nem professores com períodos de ociosidade nem eventos que possuam aulas geminadas a serem atendidas.

4.2.1 Estratégia de redução de Subproblemas para decomposição por professores

Para a decomposição por professores foi criado sub-rotinas para auxiliar na busca por subproblemas relevantes para a melhoria da função objetivo. Este procedimento é baseado no *fix-and-optimize* apresentado por Dorneles *et al.* (2014) e está detalhado nos algoritmos a seguir.

QUADRO 4 – PSEUDO CÓDIGO DA HEURÍSTICA *FIX-AND-OPTIMIZE* COM RESTRIÇÃO DE SUBPROBLEMAS VINCULADO A PROFESSORES

Algoritmo FixAndOptimizeParaProfessores($\mathbb{N}, TL$)

1. $x^* \leftarrow SoluçãoInicial$
2. **Se** $x^* = \emptyset$ **então**
3. Retorna $\emptyset$
4. **Fim**
5. $ListaDeProblemasRelevante \leftarrow IdentificaProblemaRelevante(x^*)$
6. **Para todo** $(\tau, k) \in \mathbb{N}$ **faça**
7. $count \leftarrow QtdeSubproblemas(\tau, k)$
8. $s \leftarrow 1$
9. $SemMelhorias \leftarrow 0$
10. **Repita**
11. **Se** $TL\ foi\ atingido$ **então**
12. Retorna x^*
13. **Fim**
14. $\mathrm{R} \leftarrow Decompor(\tau, k, s)$
15. **Se** $\mathrm{R} \in ListaDeProblemasRelevantes$
16. $x \leftarrow Resolver(x^*, \mathrm{R})$
17. **Se** x é $melhor\ do\ que\ x^*$ **então**
18. $x^* \leftarrow x$
19. $SemMelhorias \leftarrow 0$
20. **Senão**
21. $SemMelhorias = SemMelhorias + 1$
22. **Fim**
23. $ListaDeProblemasRelevante \leftarrow IdentificaProblemaRelevante(x^*)$
24. **Senão**
25. $\mathrm{R} \leftarrow EncontrarProblema(\mathrm{R}, ListaDeProblemasRelevantes)$
26. **Fim**
27. $s \leftarrow (s\ \boldsymbol{mod}\ count) + 1$
28. **até** $SemMelhorias = count$
29. **Fim**
30. **Retorna** x^*

FONTE: O AUTOR (2018).

QUADRO 5 – FUNÇÃO QUE ERNCONTRA OS PROFESSORES QUE POSSUEM PERÍODOS DE OCIOSIDADE OU AULAS GEMINADAS A SEREM ATENDIDAS

Algoritmo IdentificaProblemaRelevante(x^*)

1. **Caso** $\tau = TD$
2. **Para todo** $t \in T$ **Faça**
3. **Se** $t \in \mathrm{R}\ possui\ alguma\ penalidade$ **então**
4. $ListaDeProblemasRelevantes \leftarrow t$
5. **Fim**
6. **Fim**
7. **Retorna** $ListaDeProblemasRelevantes$

FONTE: O AUTOR (2018).

QUADRO 6 – FUNÇÃO QUE ENCONTRA A PRÓXIMA VIZINHANÇA RELEVANTE PARA $k \leq 3$

Algoritmo EncontrarProblema($\mathrm{R}, ListaDeProblemasRelevantes$)

1. **Para todo** $t \in ListaDeProblemasRelevantes$ **Faça**
2. **Se** $\mathrm{R}_3 < \mathrm{t}$ **então**
3. $Px \leftarrow PosiçaoSubProblemaRelevante(\mathrm{R}_1, \mathrm{R}_2, t)$
4. $SemMelhorias \leftarrow SemMelhorias + Px - Posição(\mathrm{R})$
5. $Posição(\mathrm{R}) \leftarrow Px$
6. $Flag \leftarrow 1$
7. **Fim**
8. **Fim**
9. **Se** $Flag = 0$ **então**
10. **Para todo** $t \in ListaDeProblemasRelevantes$ **Faça**
11. **Se** $\mathrm{R}_2 < \mathrm{t}$ **e** $t < QuantidadeProfessores$ **então**
12. $Px \leftarrow PosiçaoSubProblemaRelevante(\mathrm{R}_1, t, t+1)$
13. $SemMelhorias \leftarrow SemMelhorias + Px - Posição(\mathrm{R})$
14. $Posição(\mathrm{R}) \leftarrow Px$
15. $Flag \leftarrow 1$
16. **Fim**
17. **Fim**
18. **Se** $Flag = 0$ **então**
19. **Para todo** $t \in ListaDeProblemasRelevantes$ **Faça**
20. **Se** $\mathrm{R}_1 < \mathrm{t}$ **e** $t < QuantidadeProfessores - 1$ **então**
21. $Px \leftarrow PosiçaoSubProblemaRelevante(t, t+1, t+2)$
22. $SemMelhorias \leftarrow SemMelhorias + Px - Posição(\mathrm{R})$
23. $Posição(\mathrm{R}) \leftarrow Px$
24. $Flag \leftarrow 1$
25. **Fim**
26. **Fim**
27. **Se** $Flag = 0$ **então**
28. $SemMelhorias \leftarrow SemMelhorias + NumeroTotaldeVizinhos - Posição(\mathrm{R})$
29. $Posição(\mathrm{R}) \leftarrow 0$
30. **Fim**
31. **Fim**
32. **Fim**
33. **Retorna** R

FONTE: O AUTOR (2018).

QUADRO 7 – FUNÇÃO PARA ENCONTRAR A POSIÇÃO DO PRÓXIMO PROBLEMA RELEVANTE PARA $k \leq 3$

Algoritmo PosiçãoSubProblemaRelevante(S_3, S_2, S_1)

1. $Px = PosInicio(k_3, S_3) - PosInicio(k_2, S_3 + 1) + PosInicio(k_2, S_2) + S_1 - S_2 - 1$
2. **Retorna** Px

FONTE: O AUTOR (2018).

A lista $ListaDeProblemasRelevantes$ é composto por professores que possuam penalidades com relação a períodos de ociosidade ou que pertençam a eventos com aulas geminadas a serem atendida, de forma que na linha 12 do QUADRO 4 é analisado se os professores que compõe a decomposição R estão contidos em $ListaDeProblemasRelevantes$. Já o QUADRO 5 apresenta a função usada para identificar se o professor ou classe possui alguma penalidade.

O Algoritmo 6 faz uma busca em R para ir direto aos subproblemas em que há melhorias a serem realizadas. Para isto utiliza-se o fato de que $\mathrm{R} = \{\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3\}$ sempre é representado de forma crescente, com o professor que possua o de menor índice (posição em que o professor é cadastrado) em R_1 e o maior índice na posição R_3. O processo se inicia verificando se os professores de R estão contidos em $ListaDeProblemasRelevantes$. Caso existam professores relacionados o subproblema é processado buscando melhorias, mas caso nenhum professor estiver relacionado se busca a posição Px do próximo professor com índice maior que R_3 que esteja contido em $ListaDeProblemasRelevantes$. O algoritmo irá buscar a posição deste elemento sendo $PosInicio(k_3, S_3)$ a posição de início do elemento S_3 na cardinalidade k_3, e assim por diante. Após ser encontrado a posição desejada o $SemMelhorias$ precisa ser atualizado de forma que conte a quantidade de subproblemas evitados para que não se perca o controle do número de vizinhanças visitadas pois, apesar de não ter sido feita a busca em certo subproblema, esta precisa ser contado para que o critério de parada do Algoritmo 4 continue valendo (linha 28). Caso nenhuma das condições seja cumprida, significa que os professores que constam em $ListaDeProblemasRelevantes$ possuem índices inferiores do que aos índices dos elementos de R, sendo necessário voltar a busca-los desde o índice 0.

O algoritmo de $EncontrarProblema$ para $k = 4$ está descrito no QUADRO 8.

QUADRO 8 – FUNÇÃO QUE ENCONTRA A PRÓXIMA VIZINHANÇA RELEVANTE PARA $k = 4$

```
Algoritmo EncontrarProblema(R, ListaDeProblemasRelevantes)
1.  Para todo t ∈ ListaDeProblemasRelevantes Faça
2.      Se R4 < t então
3.          Px ← PosiçaoSubProblemaRelevante(R1, R2, R3, t)
4.          SemMelhorias ← SemMelhorias + Px − Posição(R)
5.          Posição(R) ← Px
6.          Flag ← 1
7.      Fim
8.  Fim
9.  Se Flag = 0 então
10.   Para todo t ∈ ListaDeProblemasRelevantes Faça
11.     Se R3 < t e t < QuantidadeProfessores então
12.         Px ← PosiçaoSubProblemaRelevante(R1, R2, t, t + 1)
13.         SemMelhorias ← SemMelhorias + Px − Posição(R)
14.         Posição(R) ← Px
15.         Flag ← 1
16.     Fim
17.   Fim
18.   Se Flag = 0 então
19.     Para todo t ∈ ListaDeProblemasRelevantes Faça
20.       Se R2 < t e t < QuantidadeProfessores − 1 então
21.           Px ← PosiçaoSubProblemaRelevante(R1, t, t + 1, t + 2)
22.           SemMelhorias ← SemMelhorias + Px − Posição(R)
23.           Posição(R) ← Px
24.           Flag ← 1
25.       Fim
26.     Fim
27.     Se Flag = 0 então
28.       Para todo t ∈ ListaDeProblemasRelevantes Faça
29.         Se R1 < t e t < QuantidadeProfessores − 2 então
30.             Px ← PosiçaoSubProblemaRelevante(t, t + 1, t + 2, t + 3)
31.             SemMelhorias ← SemMelhorias + Px − Posição(R)
32.             Posição(R) ← Px
33.             Flag ← 1
34.         Fim
35.       Fim
36.       Se Flag = 0 então
37.         SemMelhorias ← SemMelhorias + NumeroTotaldeVizinhos − Posição(R)
38.         Posição(R) ← 0
39.       Fim
40.     Fim
41.   Fim
42. Retorna R
```

FONTE: O AUTOR (2018).

QUADRO 9 – FUNÇÃO PARA ENCONTRAR A POSIÇÃO DO PRÓXIMO PROBLEMA RELEVANTE PARA $k = 4$

Algoritmo PosiçãoSubProblemaRelevante(S_4, S_3, S_2, S_1)

1. $Px = PosInicio(k_4, S_4) - PosInicio(k_3, S_4 + 1) + PosInicio(k_3, S_3) + PosInicio(k_2, S_3 + 1) + PosInicio(k_2, S_2) + S_1 - S_2 - 1$
2. **Retorna** Px

FONTE: O AUTOR (2018).

4.2.2 Estratégia de redução de Subproblemas para decomposição por classe

Ao realizarmos a decomposição por classe, o único problema a ser evitado são os que não possuam aulas geminadas que são encontrados observando o valor da variável G_e. O Algoritmo é descrito a seguir.

QUADRO 10 – PSEUDO CÓDIGO DA HEURÍSTICA *FIX-AND-OPTIMIZE* COM REDUÇÃO DE SUBPROBLEMAS RELACIONADO À CLASSE

Algoritmo FixAndOptimizeParaClasses(ℕ, TL)

1. $x^* \leftarrow SoluçãoInicial$
2. **Se** $x^* = \emptyset$ **então**
3. Retorna ∅
4. **Fim**
5. **Para todo** $(\tau, k) \in$ ℕ **faça**
6. $count \leftarrow QtdeSubproblemas(\tau, k)$
7. $s \leftarrow 1$
8. $SemMelhorias \leftarrow 0$
9. **Repita**
10. **Se** $TL\ foi\ atingido$ **então**
11. Retorna x^*
12. **Fim**
13. ℝ $\leftarrow Decompor(\tau, k, s)$
14. **Se** $G_e \in$ ℝ > 0 **então**
15. $x \leftarrow Resolver(x^*,$ ℝ$)$
16. **Se** x *é melhor do que* x^* **então**
17. $x^* \leftarrow x$
18. $SemMelhorias \leftarrow 0$
19. **Senão**
20. $SemMelhorias = SemMelhorias + 1$
21. **Fim**
22. **Senão**
23. $SemMelhorias = SemMelhorias + 1$
24. **Fim**

25. $s \leftarrow (s \ \boldsymbol{mod}\ count) + 1$
26. **até** $SemMelhorias = count$
27. **Fim**
28. **Retorna** x^*

FONTE: O AUTOR (2018).

5. RESULTADOS

As modificações propostas neste trabalho possuem o objetivo de diminuir o tempo computacional do modelo proposto por DORNELES *et al.* (2014), encontrar soluções superiores às conhecidas na literatura além de apresentar opções de busca em vizinhança mais vantajosas para serem aplicadas junto a heurística de *fix-and-optimize*. Para isto, usa-se o *solver* GUROBI 7.0.2 com os algoritmos implementados em VB.net. Os testes foram feitos em um notebook equipado com Intel Core i5-6200U e 2.4 GHz, 8 GB de RAM e com sistema operacional de 64 bits, Windows 10. Os parâmetros γ_t, ω_t e δ_e , relativos aos pesos das restrições *soft*, usados no modelo matemático são 9, 6 e 3 respectivamente, conforme padronizado pelo ITC-2011.

5.1 Dados de Entrada

As instâncias usadas para testes foram retiradas do ITC-2011, são referentes as escolas brasileiras e estão adaptadas para a aplicação do CTTPCR. A tabela 3 apresenta as características de cada instância, onde as colunas $|D|$, $|P|$, $|T|$, $|C|$ e $|E|$ representam o número de dias, períodos, professores, classes e eventos, respectivamente. As colunas $\sum_{e \in E} MG_e$ e $\sum_{e \in E} R_e$ representam a quantidade mínima de lições geminadas exigidas e a carga total de trabalho, sendo que a quantidade mínima de lições geminadas consideradas é o maior possível para cada evento. Todos os professores se encontram disponíveis em todos os períodos (DORNELES *et al.*, 2014).

TABELA 3 – CARACTERISTICAS DAS INSTÂNCIAS ESTUDADAS.

Id	Nome	$\|D\|$	$\|P\|$	$\|T\|$	$\|C\|$	$\|E\|$	$\sum_{e \in E} MG_e$	$\sum_{e \in E} R_e$
A	BrazilInstance1	5	5	8	3	21	30	75
D	BrazilInstance4	5	5	23	12	127	125	300
E	BrazilInstance5	5	5	31	13	119	132	325
F	BrazilInstance6	5	5	30	14	140	144	350
G	BrazilInstance7	5	5	33	20	205	211	500

FONTE: ADAPTADO DE DORNELES ET AL (2014)

Das dez estratégias de decomposição sugeridas por Dorneles et al. (2014), foram consideradas as três que obtiveram os melhores resultados, são elas:

- F7 – ((TD, 2), (CD, 2); ... ; (TD, ∞), (CD, ∞)).
- F8 – ((CD, 2), (TD, 2); ... ; (CD, ∞), (TD, ∞)).
- F10 - ((CD, 3), (TD, 3); ... ; (CD, ∞), (TD, ∞)).

As estratégias consistem em aumentar o valor do parâmetro k toda vez em que todos os subproblemas forem explorados. Desta forma, o número de variáveis livres dos subproblemas tende a crescer, aumentando o esforço computacional para encontrar uma solução factível. O tempo total de processamento foi fixado em 10 minutos para a instância A, 30 minutos para as instâncias D e F e 60 minutos para as instâncias E e G, conforme apresentado na literatura (DORENELES *et al.*, 2014).

Estas estratégias citadas foram aplicadas em cada uma das cinco instâncias e testadas de forma a apresentar o valor da função objetivo e tempo computacional para as seguintes variações de busca para o *fix-and-optimize*:

- Clássico – o método de busca clássico consta com uma exploração sequencial utilizando os modelos sem a inclusão da restrição (19), conforme descrito por Dorneles *et al.* (2014).
- C19 – o método também realiza uma busca sequencial pelos subproblemas, porém, incluindo a restrição (19) proposta neste trabalho, tendo o mínimo de dias trabalhados (LB) já estabelecido previamente, conforme predefinido e descrito na literatura (DORNELES *et al.*, 2014). Os valores de LB são: 189 para a instancia A, equivalente a 21 dias trabalhados pelos professores; 621 para a instância D, equivalente a 69 dias trabalhados; 756 para a instância E, equivalente a 84 dias trabalhados; 738 para a instância F, equivalente a 82 dias trabalhados; e 999 para a instância G, equivalente a 111 dias trabalhados.
- RPO – Este método é similar ao C19 em relação ao modelo, porém realiza a busca dando foco a redução dos períodos de ociosidade conforme descrito no capítulo 4.2.1 quando realiza a decomposição por professores.

Já para as decomposições por classe é realizada normalmente, de forma sequencial.

- RG – Este método também é idêntico ao C19, porém realiza a busca dando foco as aulas geminadas não atendidas conforme descrito no capítulo 4.2.2 quando realiza ambas as decomposições, por professor e por classe.
- RPOG – Este método é a junção do RPO e o RG realizando a otimização de todos os subproblemas que possui alguma penalidade, tanto por períodos de ociosidade como por aulas geminadas não atendidas.

5.2 Resultados Modelo 1

Todos os cinco métodos de busca foram aplicados ao Modelo 1, adaptado do modelo apresentado de Dorneles *et al.* (2014). Os resultados obtidos estão representados na tabela 4 descrita a baixo.

TABELA 4 – RESULTADOS COMPUTACIONAIS MODELO 1.

Instância	Literatura		Estratégia	Clássico		C19		RPO		RG		RPOG	
	F.O.	L.B.		F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)
A	200	200	F7	200	180,564	200	130,633	200	130,05	200	141,582	200	130,603
			F8	200	188,692	200	153,448	200	152,593	200	234,529	200	236,327
			F10	200	138,618	200	150,614	200	144,656	**200**	**126,604**	200	151,533
			Média	200	169,291	200	144,898	***200***	***142,433***	200	167,571	200	172,821
D	648	646	F7	656	1708,589	651	1633,517	653	1797,707	655	1706,297	655	1728,411
			F8	651	1104,214	653	354,64	653	332,846	653	343,225	653	344,593
			F10	655	1438,538	650	1745,953	650	1571,044	**650**	**1537,337**	650	1716,258
			Média	654	1417,1137	***651,3***	***1244,703***	652	1233,865	652,6	1195,619	652,6	1263,087
E	776	775	F7	786	3262,456	783	3298,399	783	3428,508	783	3297,044	782	3206,836
			F8	777	3246,648	779	2875,59	778	2908,420	778	2744,117	**777**	**2831,625**
			F10	778	3515,074	784	2884,364	784	3421,326	783	3120,744	783	3339,353
			Média	***780,3***	***3341,3926***	782	3019,451	781,6	3252,751	781,3	3053,968	780,6	3125,938
F	779	773	F7	811	1123,099	782	1760,666	784	1451,173	793	1300,466	**782**	**1685,165**
			F8	794	1533,383	789	1131,429	789	1054,755	789	1066,414	789	1073,052
			F10	792	1815,839	787	1188,756	787	1179,758	787	1177,497	787	1201,077
			Média	799	1490,774	786	1360,283	786,6	1228,562	789,6	1181,459	***786***	***1319,764***
G	1066	1039	F7	1130	2740,107	1101	3457,083	1088	2559,174	1090	2192,323	1100	3529,671
			F8	1091	2982,91	1094	3246,921	1087	3527,769	1095	3336,473	1095	3327,190
			F10	1089	1698,054	1081	2158,049	**1079**	**3055,107**	1079	3505,939	1079	3482,627
			Média	1103,33	2473,69	1092	2954,017	***1084,6***	***3047,35***	1088	3011,578	1091,3	3445,496

FONTE: O AUTOR (2018)

As linhas da tabela 4 apresentam as cinco instâncias, com cada uma das três estratégias usadas e um valor médio para cada método afim de realizar uma comparação entre os resultados. Nas colunas estão os cinco métodos aplicados a cada instância e estratégia, sendo que a coluna ‘Literatura’ apresentam os melhores valore de função objetivo (F.O.) e os melhores *lower bounds* conhecidos para cada instância, conforme o ITC-2011, sendo que a instância A é a única que possui o valor ótimo conhecido.

Para a instância A todos os métodos foram capazes de encontrar o valor ótimo, em um tempo de processamento bem inferior aos 10 minutos dados como parâmetro de entrada, revelando o quanto é eficiente a heurística de *fix-and-optimize*. Nenhuma das estratégias levou mais de quatro minutos para encontrar o valor ótimo sendo que o método RG com estratégia F10 foi o que o fez com o menor tempo de 126 segundos. É possível perceber que os métodos de redução de subproblemas RPO e RG diminuíram o tempo computacional comparado ao método de origem C19, cumprindo o objetivo proposto. O mesmo não acontece com o RPOG que aumenta o tempo computacional em duas das três estratégias. Em média, o melhor método foi o RPO, com o tempo médio inferior aos outros.

Em nenhuma das instâncias restantes fomos capazes de reproduzir os melhores valores para a função objetivo conhecidos da literatura no tempo computacional definido.

Para a instância D o melhor valor de F.O. obtido foi 650, todos encontrados utilizando a estratégia F10. Uma característica interessante é que para a estratégia F8 um valor de F.O. muito baixo é encontrado rapidamente, em pouco mais de 300 segundos, porém os métodos aplicados não conseguem reduzir este valor no tempo restante de processamento. Para a estratégia F7 os métodos de redução de subproblemas pioraram o valor de F.O. comparados ao C19, mesmo assim se saíram melhores do que os valores obtidos pelo método clássico, já para as estratégias F8 e F10 o tempo computacional dos métodos de redução de subproblemas foram melhores. Em termos de média o método C19 se saiu melhor devido aos valores de F.O.

Para a instância E o melhor valor de F.O. obtido foi o 777, encontrado utilizando a estratégia F8 para os métodos clássico e RPOG, sendo que o segundo o obteve com o menor tempo. O RPOG obteve um melhor valor objetivo em duas das três estratégias,

porém os métodos RPO e RG se mostraram inferiores em tempo ao C19. Mesmo assim, na média, o método clássico foi o que obteve os melhores valores de F.O.

Para a instância F o melhor valor de F.O. obtido foi de 782 para os métodos C19 e RPOG, tendo o segundo o encontrado em menor tempo. Para esta instância, mais uma vez os métodos RPO e RG se saíram com o desempenho inferior ao C19. O método clássico foi o que obteve o pior desempenho sendo o RPOG o melhor, com uma F.O. média de 786.

Por fim, para a instância G o melhor valor de F.O. encontrado foi de 1079 para os métodos RPO, RG e RPOG, todos com a estratégia F10, sendo o RPO o mais rápido. Novamente o método clássico obteve um desempenho inferior comparado aos outros métodos. Na média o RPO foi o que obteve o melhor resultado devido a F.O.

A vantagem das estratégias de redução de subproblemas é evitar vizinhanças que não possui melhorias, permitindo que se explore um maior número de combinações relevantes e diminua o tempo computacional das instâncias. A quantidade de problemas resolvidos e evitados podem ser vistos na tabela 5.

TABELA 5 – QUANTIDADE DE SUBPROBLEMAS EVITADOS DO MODELO 1

Inst.	Estr.	Cláss	C19	RPO			RG			RPOG		
				Resol.	Evitado	**Total**	Resol.	Evitados	**Total**	Resol.	Evitado	**Total**
A	F7	329	278	257	21	**278**	199	67	**266**	277	1	**278**
	F8	93	221	219	28	**247**	123	113	**236**	224	15	**239**
	F10	101	239	201	69	**270**	***152***	***117***	***269***	397	5	**402**
D	F7	6375	7677	4323	8662	**12985**	6552	8588	**15140**	7632	2235	**9867**
	F8	2129	2024	694	1330	**2024**	458	1566	**2024**	1859	165	**2024**
	F10	2834	1771	631	1140	**1771**	***594***	***1147***	***1741***	1606	165	**1771**
E	F7	15124	20138	2363	37318	**39681**	16334	4082	**20146**	10467	33747	**44214**
	F8	8104	6989	560	4366	**4926**	1565	3340	**4905**	***1611***	***3315***	***4926***
	F10	4495	4609	455	4040	**4495**	956	3649	**4605**	2933	1676	**4609**
F	F7	9318	9359	3864	3379	**7243**	11110	2304	**13414**	***8815***	***544***	***9359***
	F8	821	435	479	452	**931**	491	120	**611**	497	66	**563**
	F10	1537	1236	1292	488	**1780**	1291	439	**1291**	1092	406	**1092**
G	F7	19704	19690	15653	21504	**37157**	17001	6715	**23716**	19393	297	**19690**
	F8	1712	15420	1457	4925	**6382**	13933	1487	**15420**	13934	1486	**15420**
	F10	5456	5765	***1190***	***4572***	***5762***	4296	1469	**5765**	4296	1469	**5765**

FONTE: O AUTOR (2018)

Como os métodos Clássico e C19 não possuem estratégias de redução de subproblemas, a tabela 5 apresenta apenas o número de subproblemas resolvidos nestes métodos. O tempo limite para a resolução de cada subproblema é de 30 segundos, sendo que o processamento termina quando a primeira solução factível é encontrada, conforme definido por Dorneles, *et al.*, 2014). Alguns subproblemas levam milésimos de segundo em processamento para encontrar uma solução factível enquanto outros superam 20 segundos, ficando próximo de estourar o tempo permitido. Isto explica a grande diferença entre os números de subproblemas resolvidos entre os métodos, já que este tempo é longo e pode consumir uma boa parte do tempo de processamento.

Para a instância A o número de subproblemas entre os métodos e estratégias não variam muito entre eles devido ao problema ser pequeno. Em geral, para os métodos RPO e RPOG o número de subproblemas evitados é pequeno e apenas o método RG evita uma boa quantidade de subproblemas, mostrando que para esta instância as aulas geminadas não atendidas são poucas, com a maior parte da penalização relacionada a períodos de ociosidade, com o melhor resultado alcançado pelo método RG com F10, sendo também a combinação que mais evitou subproblemas.

Para a instância D o número de subproblemas evitados pelos métodos RPO e RG foi quase o dobro do número de subproblemas resolvidos. Com destaque a estratégia F7 que teve um número grande de subproblemas analisados. Podemos ver que por mais que na tabela 4 mostre que os resultados obtidos para F8 tenham sido encontrados rapidamente e sem melhorias no restante do período de processamento, um grande número de subproblemas foi analisado, com um total de 2024 subproblemas para todas elas, o que explica o tempo semelhante e o mesmo valor da F.O. para os quatro métodos, mostrando que para esta instância as estratégias de redução de subproblema não trouxe ganho significativo em termos de exploração do problema, o mesmo acontece para a estratégia F10. Porém, mesmo explorando exatamente os mesmos problemas, o método RG foi o que mais evitou subproblemas nos dois casos, alcançando o melhor valor no menor tempo para o método RG com F10.

Para a instância E a estratégia F8 foi que conseguiu os melhores resultados com o método RPOG. O destaque desta instância é que o método C19 explorou um número maior de subproblemas e mesmo assim teve um resultado de F.O. inferior.

Para a instância F os melhores resultados foram obtidos pelos métodos C19 e RPOG que também exploraram exatamente o mesmo número de subproblemas, entretanto o RPOG evitou uma parte destes subproblemas, conseguindo alcançar o valor de F.O. mais rapidamente.

Para a instância G os métodos RPO, RG e RPOG obtiveram o mesmo valor de função objetivo, sendo que o RPO evitou uma quantidade maior de subproblemas, o que explica ele ser o mais rápido.

Em nenhum dos testes houve subproblemas que estouraram o tempo limite de 30 segundos. Porém, ao diminuirmos este tempo para 20 segundos o número de estouros fica alto, principalmente para as instâncias maiores, prejudicando o valor objetivo dos problemas. Devido a isso permanecemos usando o tempo de 30 segundos.

Após analisar os métodos que obtiveram os melhores desempenhos para cada instância reproduzimos novamente o procedimento dando um tempo de execução de 3 horas para cada um deles, com o objetivo de avaliar qual a melhor solução que pode ser encontrada. Este procedimento não foi realizado para a instância A pois o tempo de 10 minutos foi o suficiente para encontrar seu valor ótimo. Os resultados estão na tabela 6.

TABELA 6 – MELHORES VALORES OBTIDOS EM TEMPO EXTENDIDO PARA O MODELO 1

Instância	Literatura	Estratégia	Método	Valor Obtido	Tempo Computacional (s)
D	648	F10	RG	648	6307,067
E	776	F8	RPOG	777	2831,653
F	779	F7	RPOG	**778**	9685,846
G	1066	F10	RPO	1074	4146,021

FONTE: O AUTOR (2018)

Ao analisarmos a tabela 6 podemos observar que com exceção da instância E, as outras instâncias tiveram seu valor de F.O. reduzido. A instância D alcançou o valor de F.O. igual ao melhor valor conhecido na literatura, levando um tempo de aproximadamente 1:45h. Para as instâncias E e G o tempo não foi o suficiente para encontrar os melhores valores da literatura que seriam 776 e 1066 respectivamente. Já a instância F encontrou um valor objetivo melhor do que o da literatura que era de 779. O tempo que o método levou para encontrar este valor foi de aproximadamente 2:45h,

superior aos 30 minutos dados como parâmetro de entrada para a instância, mas reduziu a penalidade em uma aula geminada não atendida.

5.3 Resultados Modelo 2

O modelo de VEENSTRA E VIS (2016) foi aplicado originalmente a uma escola de ensino médio dos países baixos. Os conjuntos de restrições (22) a (25) referentes aos períodos de ociosidade, são menores do que as apresentadas por DORNELES *et al* (2014), reduzindo o número de variáveis do problema. Em especial, a variável h_{td} possui um menor número de índices, facilitando o controle dos períodos de ociosidade dos métodos RPO, RG e RPOG.

A tabela 7 possui a mesma estrutura da tabela 4, mas aplicada ao modelo 2.

TABELA 7 – RESULTADOS COMPUTACIONAIS MODELO 2.

Instância	Literatura		Estratégia	Clássico		C19		RPO		RG		RPOG	
	F.O.	L.B.		F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)	F.O.	Tempo(s)
A	200	200	F7	200	234,119	200	129,25	200	127,953	200	126,411	200	128,33
			F8	200	178,553	200	255,398	200	254,398	200	254,136	200	254,352
			F10	200	215,158	200	88,292	200	88,566	**200**	**47,062**	200	88,665
			Média	200	209,276	200	157,647	200	156,972	***200***	***142,536***	200	157,115
D	648	646	F7	657	1394,991	659	1700,019	653	1263,413	655	1260,413	653	1339,663
			F8	663	993,337	651	1127,381	650	1540,650	650	1801,244	650	1787,206
			F10	**648**	**1509,447**	650	1048,035	652	682,411	650	938,806	650	983,779
			Média	656	1299,258	653,3	1291,811	651,6	1162,158	651,6	1333,487	***651***	***1370,216***
E	776	775	F7	781	2991,627	781	3385,906	780	2694,017	784	1073,879	778	3358,38
			F8	785	3386,588	778	1206,451	**777**	**2726,983**	777	2838,289	777	2831,625
			F10	777	2910,541	780	1970,06	780	1244,81	780	1352,292	780	1623,411
			Média	781	3096,252	779,6	2187,472	779	2221,936	780,3	1754,82	***778,3***	***2604,472***
F	779	773	F7	810	1362,711	798	1302,433	797	1515,251	785	1725,965	800	1795,995
			F8	782	1774,218	785	1534,946	785	1467,938	785	1490,347	785	1488,571
			F10	792	1454,741	788	1546,416	**781**	**1669,936**	791	1022,369	789	1509,711
			Média	794,6	1530,556	790,3	1461,265	787,6	1551,041	***787***	***1412,893***	791,3	1598,092
G	1066	1039	F7	1084	2868,814	1096	2338,977	1087	1091,606	1078	1311,901	1096	2321,845
			F8	1084	3106,527	1077	3119,142	1075	3524,076	1077	3056,321	1077	3087,966
			F10	1082	3374,492	**1065**	**3595,281**	1069	3210,62	1069	3501,334	1073	2909,441
			Média	1083,3	3116,611	1079,3	3017,8	1077	2608,767	***1074,6***	***2623,185***	1082	2773,084

FONTE: O AUTOR (2018)

Novamente para a instância A todos os métodos foram capazes de encontrar o valor de F.O. ótimo em um período de tempo muito pequeno, em especial o método RG com F10 encontrou este valor em apenas 44 segundos, quase metade do tempo do segundo método mais veloz. Em geral, a diferença de tempo do C19 para os métodos com redução foi pequena, menos de um segundo, mas mesmo assim os métodos de redução foram mais rápidos. Na média o método RG foi o que obteve o melhor desempenho.

Para a instância D o método clássico alcançou o valor F.O. de 648 que é o melhor valor conhecido para a instância. Desta vez os métodos de redução de subproblemas tiveram um melhor desempenho comparados a C19, diminuindo o valor de F.O. nas estratégias F7 e F8 e o tempo computacional em F10. Na média o método RPOG obteve o melhor resultado.

Para a instância E o melhor valor F.O. encontrado foi novamente de 777 para os métodos RPO, RG e RPOG com estratégia F8 e para o método clássico com estratégia F10, sendo o RPO o que o encontrou no menor tempo. Novamente, como esperado, os métodos de redução de subproblemas encontraram valores mais baixos do que o encontrado pelo C19, em F.O. ou em tempo. Em média o método RPOG se saiu melhor.

Para a instância F a F.O. mais baixa foi de 781 usando RPO com F10. Nesta instância o método RPOG tem um resultado inferior ao C19. Na média o RG foi o melhor método com relação a F.O.

Para a instância G o melhor valor de F.O. foi de 1065 usando o método C19 com F10, sendo um valor inferior ao melhor F.O. encontrado na literatura que é de 1066. Por mais que para as estratégias F7 e F8 os métodos de redução de subproblemas tenham se saído melhor, para a estratégia F10 não foram capazes de igualar o novo F.O. encontrado. Entre as cinco instâncias estudadas a G é a maior, e a que possui o maior custo computacional, sendo que o *solver* tem dificuldade até mesmo para encontrar a solução inicial deste problema. Também é a instância com o maior Gap, distância entre o valor objetivo e o melhor *lower bound*, sendo a instância que melhor pode ser trabalhada. Apesar deste valor ter sido obtido pelo método C19, na média foi o método RPO o que obteve o melhor resultado.

A tabela 8 apresenta o número de subproblemas resolvidos e evitados para o modelo 2.

TABELA 8 – QUANTIDADE DE SUBPROBLEMAS EVITADOS DO MODELO 2

Inst.	Estr.	Cláss	C19	RPO			RG			RPOG		
				Resol.	Evitado	**Total**	Resol.	Evitados	**Total**	Resol.	Evitados	**Total**
A	F7	354	270	251	6	**257**	120	134	**254**	257	5	**262**
	F8	247	288	261	27	**288**	162	120	**282**	278	10	**288**
	F10	312	295	274	21	**295**	***122***	***67***	***289***	293	2	**295**
D	F7	6166	10046	3127	9695	**12822**	8836	2394	**11230**	7702	277	**7979**
	F8	2470	2256	418	1838	**2256**	2053	203	**2259**	2076	180	**2256**
	F10	***2726***	2811	304	1449	**1753**	2057	1180	**3237**	2278	515	**2793**
E	F7	15124	20138	2369	37318	**39687**	16334	4082	**20416**	10467	33747	**44214**
	F8	8104	6989	***560***	***4366***	***4926***	1565	3340	**4905**	1611	3315	**4926**
	F10	4499	4702	289	4613	**4902**	1049	3618	**4667**	2943	1759	**4702**
F	F7	9314	11331	7233	3995	**11228**	5554	6421	**11975**	7141	4871	**12012**
	F8	435	435	204	231	**435**	315	120	**435**	344	91	**435**
	F10	954	1116	***651***	***510***	***1161***	632	406	**1038**	570	483	**1053**
G	F7	16773	19371	19169	18998	**38167**	17292	28064	**45356**	19345	138	**19483**
	F8	6340	5984	2373	3611	**5984**	5608	376	**5984**	5852	132	**5984**
	F10	5456	***11112***	2192	3264	**5456**	5863	1099	**6962**	10771	341	**11112**

FONTE: O AUTOR (2018)

A instância A ficou novamente caracterizada pelo baixo número de subproblemas evitados, sendo o método RG o único que tem uma quantidade considerável de subproblemas evitados, confirmando que a maior parte da penalização desta instância é dada pelos períodos de ociosidade. O método RG, em geral, é o que explora o menor número de subproblemas, porém retorna os melhores resultados em tempo, mostrando que por mais que exista uma grade quantidade de penalizações pelos períodos de ociosidade o solver não é capaz de elimina-los, já as penalidades por aulas geminadas não atendidas as penalidades são reduzidas, explicando o melhor desempenho do método RG nesta instância.

Para a instância D o melhor resultado foi encontrado pelo método clássico. Para os métodos de redução de subproblemas é possível ver que o método RPO é o que mais tem subproblemas evitados, assim como acontece no modelo 1, sendo capaz de obter os melhores resultados. Em geral os três métodos de redução de subproblemas se saíram melhor do que C19.

Para a instância E os métodos de redução de subproblemas também tiveram melhores soluções obtendo em geral o mesmo resultado entre eles, com o RPO sendo o mais veloz. Também é o RPO o que explora a maior quantidade de subproblemas, mostrando que para esta instância os períodos de ociosidade é a penalidade mais relevante.

A instância F o RPO foi o que atingiu o valor objetivo mais baixo, sendo a estratégia F10 a que obtém o melhor resultado.

Por fim, para a instância G ao observarmos a estratégia F10 vemos que os métodos C19 e RPOG exploram um número muito maior de subproblemas do que os métodos RPO e RG. Mesmo assim o resultado do RPOG foi muito inferior aos outros em F.O.

Para obter o melhor resultado possível para cada instância apresenta-se a tabela 9 onde os métodos e estratégias que obtiveram os melhores resultados para o modelo 2 são resolvidos com tempo estendido de três horas.

TABELA 9 – MELHORES VALORES OBTIDOS EM UM PARA O MODELO 2

Instância	Estratégia	Método	Valor Obtido	Tempo Computacional (s)
D	F10	Clássico	648	1509,447
E	F8	RPO	777	2726,317
F	F10	RPO	**777**	10043,267
G	F10	C19	**1055**	8660,495

FONTE: O AUTOR (2018)

Para as instâncias D e E nenhuma melhoria foi obtida permanecendo com o menor valor em um tempo computacional semelhante ao encontrado dentro do tempo padrão de cada instância. Já as instâncias F e G novamente foram capazes de encontrar valores de F.O. inferiores aos descritos na literatura, sendo que para a instância F o valor foi ainda mais baixo do que o valor encontrado para o Modelo 1. A instância F encontrou o valor de 777 em um tempo de 2:47h, tempo semelhante ao usado para obter o valor de 778 para o modelo 1, fixando este resultado como novo menor F.O. para esta instância. Já a instância G encontrou um valor de F.O. de 1055, inferior ao melhor resultado da

literatura que era de 1066, com um tempo de processamento de 2:24h. Ambos os resultados podem ser observados no Apêndice A.

5.4 Comparação entre os modelos

Para auxiliar na análise entre os modelos é realizado uma comparação entre as melhores soluções encontradas para cada instância. Os valores serão expressos em valor de função objetivo e tempo usando os melhores resultados encontrados para o tempo pré-estipulado na literatura e para o tempo estendido de três horas.

TABELA 10 – COMPARAÇÃO ENTRE OS MODELOS PARA O TEMPO PREVISTO

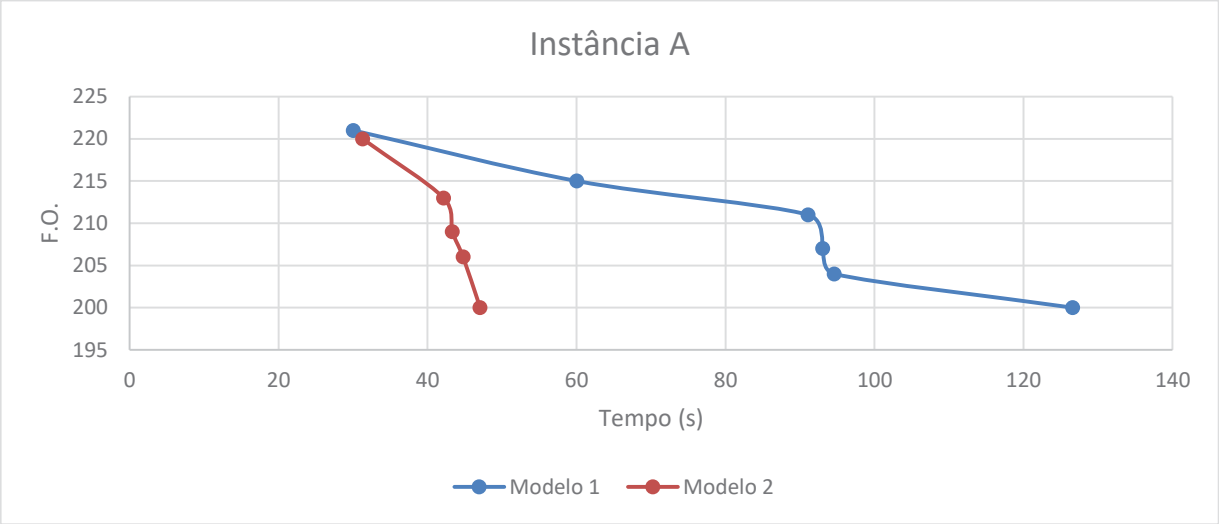
Instância	MODELO 1				MODELO 2			
	Estratégia	Método	F.O.	Tempo	Estratégia	Método	F.O.	Tempo
A	F10	RG	200	126,604	F10	RG	200	47,062
D	F10	RG	650	1537,337	F10	Clássico	648	1509,447
E	F8	RPOG	777	2831,625	F8	RPO	777	2726,983
F	F7	RPOG	782	1685,165	F10	RPO	781	1669,936
G	F10	RPO	1079	3055,107	F10	C19	1065	3595,281

FONTE: O AUTOR (2018)

Pela tabela 10 é possível ver que o Modelo 2 foi superior em todas as cinco instâncias, alcançando um valor de F.O. inferior para as instâncias D, F e G e atingindo a mesma F.O. com um menor tempo as instâncias A e E. Também é possível ver que a estratégia que mais apareceu entre os melhores resultados é a F10. Já os métodos aparecem bem distribuídos, não sendo possível dizer qual deles obteve o melhor desempenho.

As figuras a seguir mostram a convergência dos testes apresentados na tabela 10 para compararmos como o modelo e os métodos se comportam, onde cada ponto é uma melhoria encontrada.

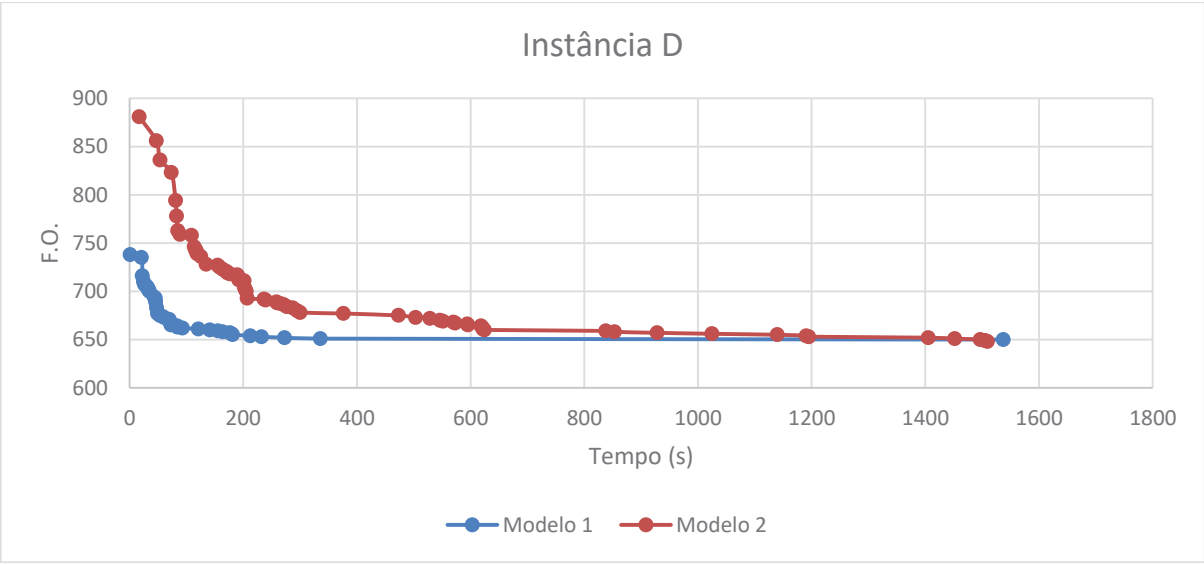
FIGURA 3 – COMPARAÇÃO ENTRE AS SOLUÇÕES DOS MODELOS 1 E 2 PARA A INSTÂNCIA A



FONTE: O AUTOR (2018).

Na figura 3 é possível ver como a curva do Modelo 2 decresce quase em vertical, enquanto a curva do Modelo 1 se mantem mais constante, tendo um decréscimo semelhante à do Modelo 2 próximo aos 90 segundos, mas voltando a decrescer constantemente até o fim. Em geral o Modelo 1 levou o triplo do tempo para encontrar o resultado.

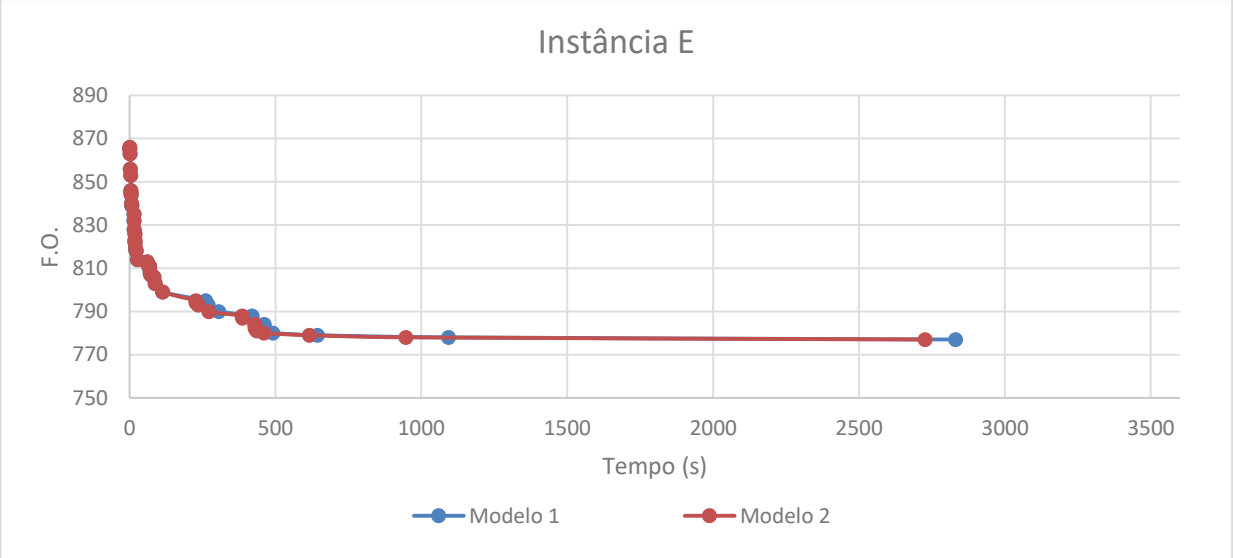
FIGURA 4 – COMPARAÇÃO ENTRE AS SOLUÇÕES DOS MODELOS 1 E 2 PARA A INSTÂNCIA D



FONTE: O AUTOR (2018).

Na figura 4 podemos observar a principal característica do método clássico. Como este método não tem a otimização previa dos dias de trabalho realizado pela restrição (19) sua solução inicial possui um valor de F.O. maior do que os dos outros métodos. Entretanto, apesar de iniciar com um valor F.O. inferior, com o passar do tempo o Modelo 2 foi capaz de reduzir o valor objetivo mais do que o Modelo 1, atingindo o melhor valor de F.O. conhecido para a instância D.

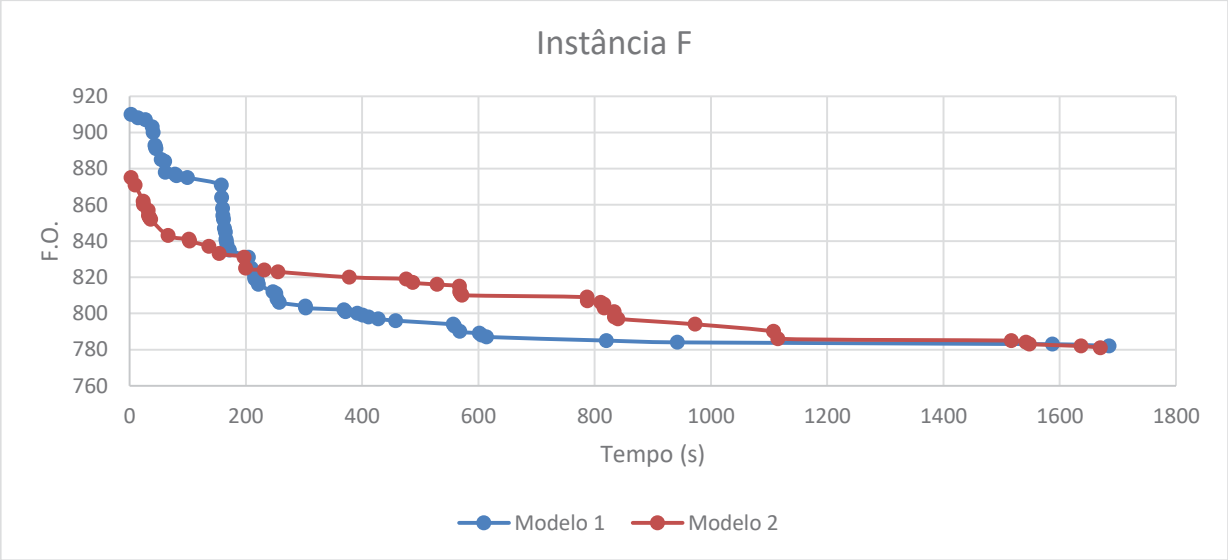
FIGURA 5 – COMPARAÇÃO ENTRE AS SOLUÇÕES DOS MODELOS 1 E 2 PARA A INSTÂNCIA E



FONTE: O AUTOR (2018).

Já na instância E quase não há diferença entre os métodos, sendo a única distinção dada pelo tempo computacional, onde o Modelo 2 se mostra mais rápido que o Modelo 1. Outra observação a ser feita é que os dois modelos param de decair antes do fim do tempo de processamento de 1:00h (3600 segundos).

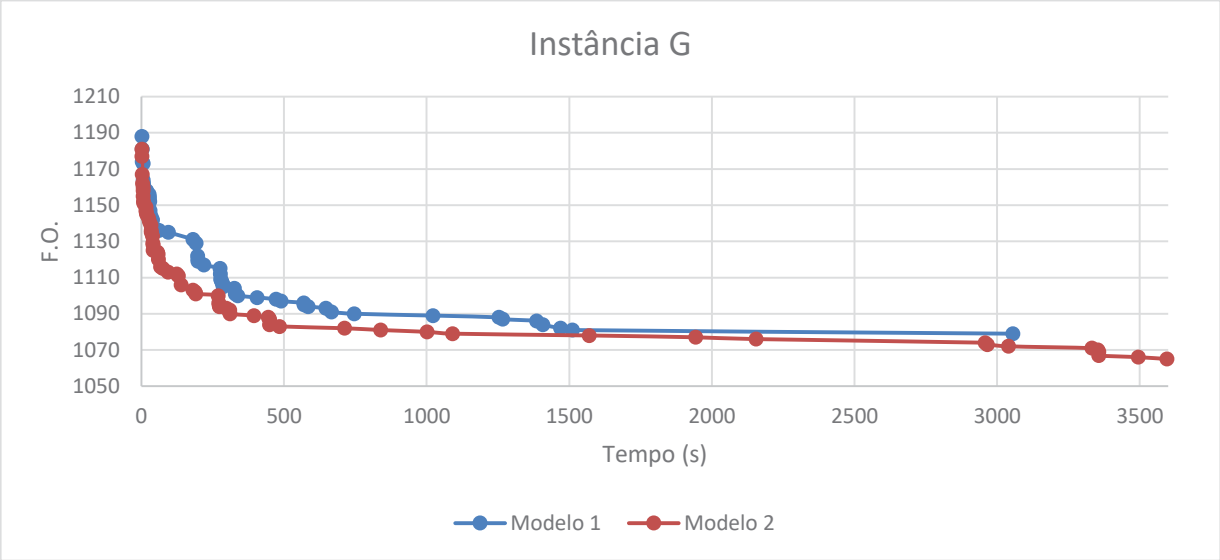
FIGURA 6 – COMPARAÇÃO ENTRE AS SOLUÇÕES DOS MODELOS 1 E 2 PARA A INSTÂNCIA F



FONTE: O AUTOR (2018).

Na figura 6 temos uma grande diferença entre as curvas. Isto é dado principalmente porque esta é a única instância em que as estratégias que encontraram a melhor solução são diferentes entre os modelos, sendo a estratégia F7 para o Modelo 1 e a estratégia F10 para o Modelo 2. O Modelo 2 inicia o processo com um valor inferior ao Modelo 1, porem a curva do Modelo 1 decai de forma rápida próximo aos 200 segundos, voltando a se encontrar em aproximadamente 1100 segundos. A partir daí os dois Modelos se estabilizam, com o valor do Modelo 2 decrescendo 1 ponto no fim do processamento.

FIGURA 7 – COMPARAÇÃO ENTRE AS SOLUÇÕES DOS MODELOS 1 E 2 PARA A INSTÂNCIA G



FONTE: O AUTOR (2018).

A figura 7 mostra que a curva do Modelo 2 permanece abaixo da curva do Modelo 1 por todo o tempo de processamento, decaindo constantemente até o fim do tempo pré-definido de 1:00h.

A comparação também é realizada para os modelos com tempo estendido de 3:00h, que podem ser observados na tabela 11. Lembrando que a instância A não é apresentada por um período estendido pois o seu valor ótimo é encontrado dentro do tempo pré-determinado.

TABELA 11 – COMPARAÇÃO ENTRE OS MODELOS PARA O TEMPO DE 3:00 HORAS

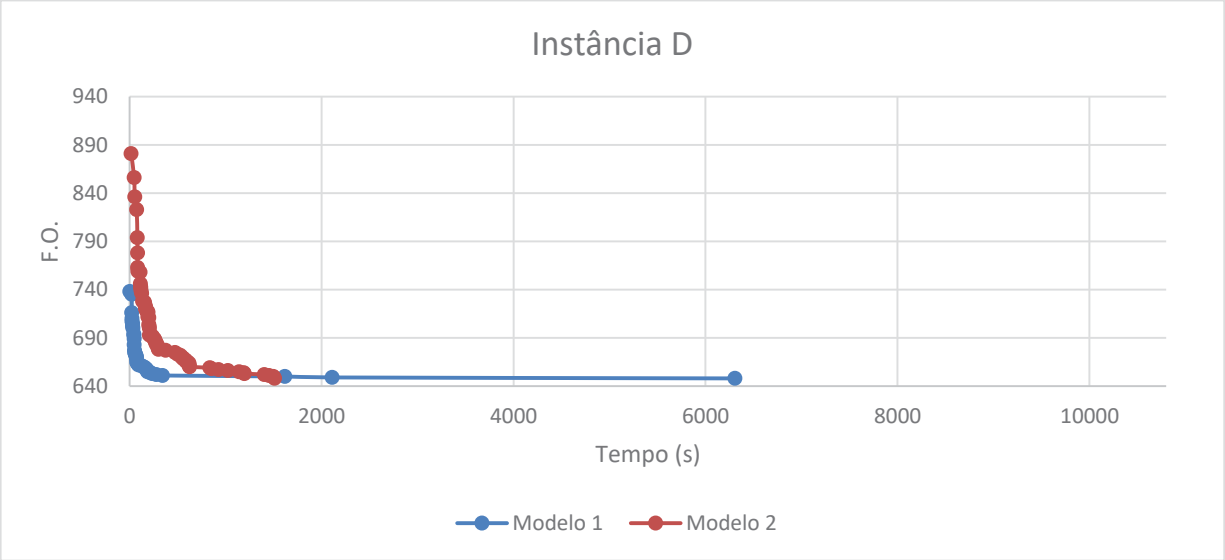
Instância	MODELO 1				MODELO 2			
	Estratégia	Método	F.O.	Tempo	Estratégia	Método	F.O.	Tempo
D	F10	RG	648	6307,067	F10	Clássico	648	1509,447
E	F8	RPOG	777	2831,653	F8	RPO	777	2726,317
F	F7	RPOG	778	9685,846	F8	RPO	777	10043,267
G	F10	RPO	1074	4146,021	F10	C19	1055	8660,495

FONTE: O AUTOR (2018)

Novamente, ao analisarmos a tabela 11 vemos que o Modelo 2 possui um resultado superior ao Modelo 1 em todas as instâncias. Para a instância D o Modelo 2 atinge o valor de 648 dentro do tempo pré-determinado, enquanto o Modelo 1 encontra este valor apenas no tempo de 6307,067 segundos, equivalente a 1:45h. Para a instância E o valor obtido na tabela 10 permanece sendo o mesmo no período estendido, sem alcançar melhorias no tempo extra. Para a instância F encontramos dois valores abaixo do melhor F.O. conhecido, a F.O. de 778 do Modelo 1 e a F.O. de 777 do Modelo 2. Os dois valores são encontrados no fim do tempo de três horas, seis vezes mais tempo do que o passado como parâmetro. Por fim para a instância G o Modelo 1 não consegue alcançar a melhor F.O. conhecida na literatura mesmo com o tempo extra, valor que o Modelo 2 já havia superado na tabela 10.

Segue os gráficos para as instâncias com tempo estendido.

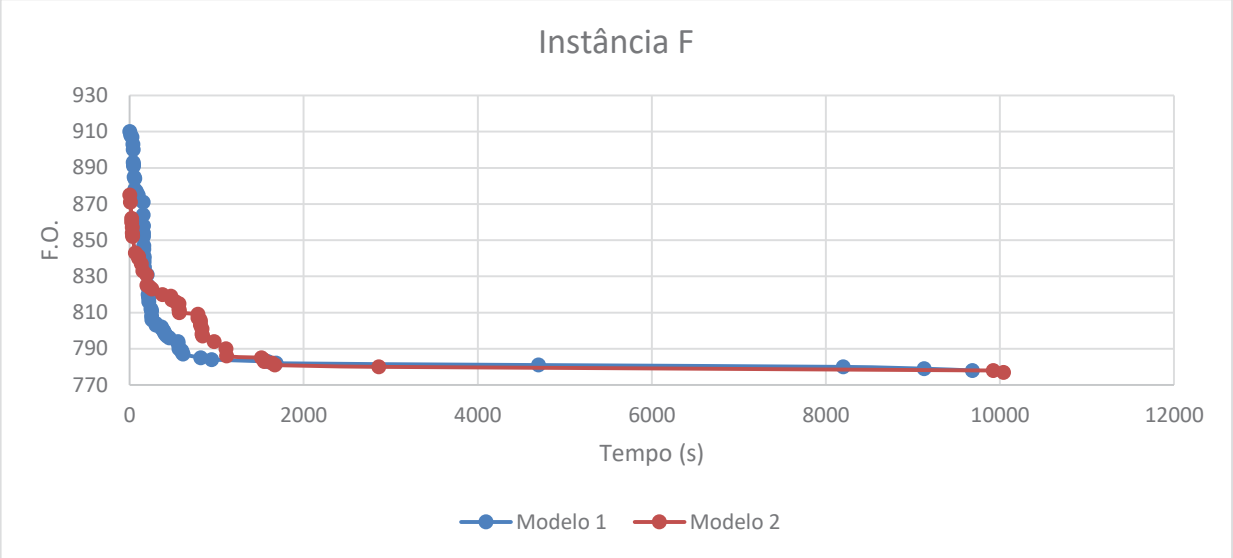
FIGURA 8 – COMPARAÇÃO ENTRE AS SOLUÇÕES ESTENDIDAS DOS MODELOS 1 E 2 PARA A INSTÂNCIA D



FONTE: O AUTOR (2018).

A diferença entre os tempos dos dois modelos é muito grande, sendo que para o Modelo 1 o valor F.O. decai rapidamente no início do processamento, porem estaciona ainda nos primeiros segundos na F.O. de 651. Após isto tem os tempos de redução bem visíveis na figura 8, reduzindo para 650, 649 e 648. Já o Modelo 2 decai continuamente até o valor encontrado sem interrupção.

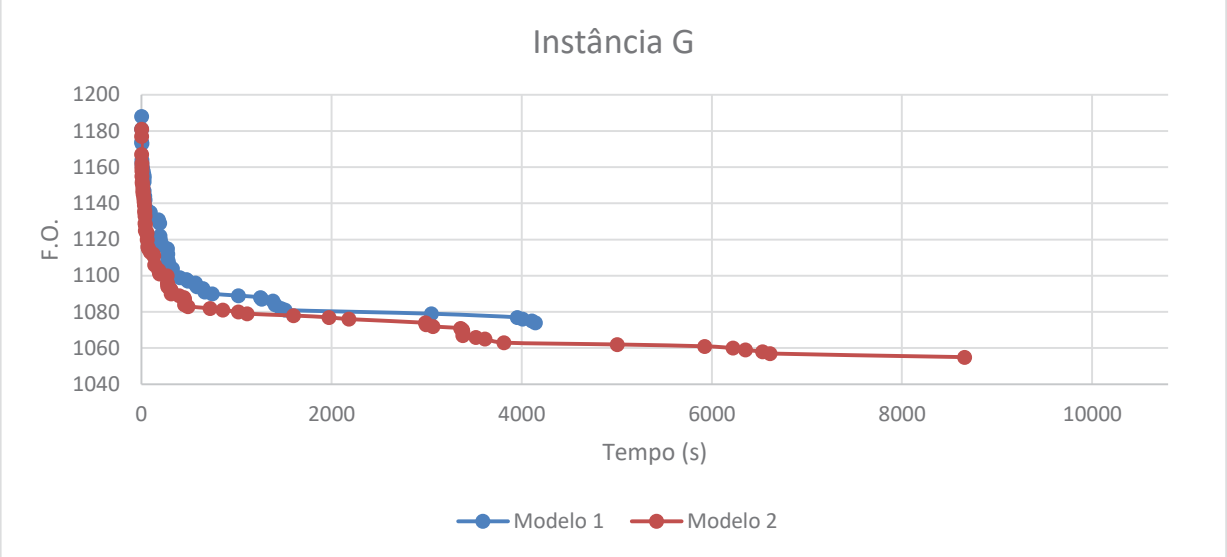
FIGURA 9 – COMPARAÇÃO ENTRE AS SOLUÇÕES ESTENDIDAS DOS MODELOS 1 E 2 PARA A INSTÂNCIA F



FONTE: O AUTOR (2018).

Na figura 9 podemos ver que após o período pré-definido os dois modelos permanecem quase constantes durante todo o período de processamento, voltando a ter reduções na F.O. apenas com mais de 8000 segundos. Os valores finais para ambos os modelos são alcançados apenas no fim do tempo de processamento.

FIGURA 10 – COMPARAÇÃO ENTRE AS SOLUÇÕES ESTENDIDAS DOS MODELOS 1 E 2 PARA A INSTÂNCIA G



FONTE: O AUTOR (2018).

A figura 10 mostra que o Modelo 1 não obteve grandes melhorias após o tempo pré-definido de 3600 segundos, enquanto o Modelo 2 continuou decaindo até 8860 segundos. Mostrando que o Modelo 2 conseguiu explorar de forma mais eficiente o espaço de busca do problema.

Por fim descreveremos os melhores resultados obtidos em cada instância mostrando suas devidas penalidades.

TABELA 12 – PENALIDADE DAS MELHORES SOLUÇÔES OBTIDAS

Instância	F.O.	Gap (%)	Dias Trabalhados	Períodos de Ociosidade	Aulas Geminadas
A	200	0	21	3	2
D	648	0,31	69	3	18
E	777	0,26	84	2	15
F	777	0,52	82	6	21
G	1055	1,54	111	6	37

FONTE: O AUTOR (2018)

A tabela 12 apresenta as penalidades dadas as melhores soluções obtidas neste trabalho. A coluna F.O. mostra os melhores valores de função objetivo encontradas. A coluna Gap apresenta a distância em que a solução encontrada está do melhor *lower bound* conhecido. As demais colunas apresentam o número total de dias trabalhados para os professores, o número de períodos de ociosidade encontrados em cada instância e a quantidade de aulas geminadas não atendidas.

As soluções encontradas ainda podem ser melhoradas, principalmente as maiores instâncias F e G que possui o valor ótimo distante do maior *lower bound* conhecido. Neste trabalho não trabalhamos os *lower bounds*, mas pode ser que alguns destes valores encontrados sejam valores ótimos.

6 . CONCLUSÃO

Na presente dissertação foi apresentado um estudo do problema de timetabling, onde foram encontradas duas soluções que superam os melhores resultados da literatura encontradas em um tempo computacional inferior a três horas.

A motivação inicial do trabalho foi de gerar um modelo para o problema de high school timetabling que apresentasse boas soluções em um tempo computacional viável, respeitando as particularidades de cada professor, dentro do cenário escolar brasileiro. Para isto, foram usadas cinco instâncias retiradas do XHSTT (competição internacional para pesquisadores do ramo) e adaptadas para o Class-Teacher Timetabling Problem with Compactness Requeriments por Dorneles et al. (2014).

O XHSTT foi adotado como formato padrão das instâncias de HSTT, podendo ser usado por pesquisadores em todo o mundo. Seus dados apresentam a quantidade de professores e classes de uma instituição, assim como a quantidade de períodos do dia. Também traz as classes em que cada professor trabalha, o número de aulas por semana de cada professor nas respectivas classes e o número máximo de aulas que o professor pode ensinar por dia em cada classe. Além dos dados apresentados pelo XHSTT foram usados os parâmetros adotados por Dorneles et al. (2014) em que os professores estariam disponíveis para lecionar em todos os períodos, o número mínimo de geminadas para cada classe seria o máximo possível e que caso um professor lecionasse duas vezes no mesmo dia em uma classe, estas aulas devem ser agendadas obrigatoriamente em sequência, assim como eliminar os períodos de ociosidade de cada professor.

Dois modelos foram gerados para o problema. O primeiro é adaptado do trabalho de Dorneles et al. (2014) em que os períodos de ociosidade são modelados em forma de grafos. Foi adicionado uma igualdade valida que fixa previamente o número de dias trabalhados por cada professor, diminuindo o espaço de busca do problema. Já o segundo modelo é a combinação dos modelos apresentados por Dorneles et al. (2014) e Veenstra e Vis (2016) onde a base do modelo é retirado do primeiro trabalho e o tratamento dos períodos de ociosidade é retirado do segundo.

Para a solução dos modelos foi adotado a heurística de fix-and-optimize, método hibrido que consiste em dividir o espaço de busca em subespaços pequenos suficientes para serem resolvidos usando um solver de forma exata. Estes subespaços foram explorados usando três estratégias de decomposição junto a cinco métodos de busca, três dos quais focam sua procura diretamente em vizinhanças que sejam relevantes para o problema, candidatas a receberem melhorias. Estas vizinhanças foram decompostas de duas formas diferentes, explorando professores e classes. Os métodos de busca se mostraram eficazes, obtendo soluções de alta qualidade mais rapidamente do que o método de busca padrão.

Como resultado encontramos novas soluções conhecidas para duas instancias da literatura encontradas no Third International Tametabling Competition (ITC-2011) superando os resultados apresentados na competição. Os resultados obtidos mostram que o modelo e os métodos propostos são muito promissores e podem ser facilmente adaptados para outras instâncias da literatura e variações do HSTT.

Como trabalhos futuros é possível trabalhar com outras instâncias da ITC-2011 e testar outras formas de busca propostos na literatura, assim como outros tipos de decomposições. Outro trabalho interessante seria realizar a escala de aula completa para as instituições de ensino, considerando os turnos da manhã, tarde e noite pois muitos professores trabalham em mais de um turno na instituição e decidir sua escala de trabalho considerando apenas um turno pode não ser o suficiente para uma grade horária otimizada.

REFERÊNCIAS

BARTAK, R. **On Modeling Planning and Scheduling Problems with Time and Resources.** Proceedings of the Twentieth Workshop of the UK Planning and Scheduling Special Interest Group. Edinburgh (UK): Old College, University of Edinburgh, p. 87-98, 2002.

BIRBAS, T.; DASKALAKI, S.; HOUSOS, E. **School timetabling for quality student and teacher schedules**. Journal of Scheduling. Vol.12, p.177-197, 2009.

BURKE, E.; DE WERRA, D.; KINGSTON, J. **Applications to Timetabling.** In: Gross, J. L.; Yellen, J. Graph Theory. [S.1.]: CRC Press, Cap 5, 2003.

DANTZIG, G. B. **A Comment on Edie's "Traffic Delays at Tool Boths".** Operational Research. Vol. 2, p. 339-341, 1954.

DE WERRA, D. **Construction of school timetables by flow methods**. INFOR. n.9, p. 12-22, 1971.

DE WERRA, D. **Constraints of Availability in Timetabling and Scheduling**. Practice and Theory of Automated Timetabling IV. 4th International Conference, PATAT 2002. Gent, Belgium, August 21-23, 2002. Selected Revised Papers. Publisher Springer-Verlag Berlin Heidelberg 2003, p. 3-23, 2003.

DEMIROVIC, E.; MUSLIU, N. **MaxSAT-based Large Neighborhood Search for High School Timetabling**. Computers & Operations Research. Vol. 78, p. 172-180, 2016.

DORNELES, A. P.; ARAUJO, O. C. B.; BURIOL, L. S. **A fix-and-optimize Heuristc for the High School Timetabling Problem**. Computers & Operations Research. Vol. 52, p. 29-38, 2014.

DORNELES, A. P. **A Matheuristic Approach for Solving the High School Timetabling Problem.** Thesis presented in partial fulfillment of the requeriments for the degree of doctor of Computer Science. Universidade Federal do Rio Grande do Sul, 2015.

DORNELES, A. P.; ARAUJO, O. C. B.; BURIOL, L. S. **A Column Generation Approach to High School Timetabling Modeled as a Multicommodity Flow Problem**. European Journal of Operational Research. Vol. 256, p. 685-695, 2017.

DREXL, A.; SALEWSKI, F. **Distribution Requirements and Compactness Constrains in School Timetabling**. European Journal of Operational Research. Vol. 102, n. 1, p. 193-214, 1997.

EDIE, L. C. **Traffic Delays at Tool Boths**. Journal of the Operations Research Society of America. Vol. 2, n. 2, p. 107-138, 1954.

EVEN, S.; ITAI, A.; SHAMIR, A. **On the Complexity of Timetable and Multi-commodity Flow Problems**. In: Proceedings of the 16th annual symposium on foundations of computer science. SFCS ’75; Wasington, DC, USA: IEEE Computer Society; p. 184-193, 1975.

FONSECA, G. H.G.; SANTOS, H. G. **Variable Neighborhood Search Based Algorithms for High School Timetabling.** Computers & Operations Research, Vol. 52, p. 203-208, 2014.

FONSECA, G. H. G.; SANTOS, H. G.; CARRANO, E. G. **Late Acceptance Hill-climbing for High School Timetabling**. Journal of Scheduling, Vol. 19, p. 453-465, 2015.

FONSECA, G. H. G.; SANTOS, H. G.; CARRANO, E. G. **Integrating Matheuristic and Metaheuristics for Timetabling.** Computers & Operations Research, Vol. 74, p. 108-117, 2016.

GINTNER, V.; KLIEWER, N.; SUHL, L. **Solving Large Multiple-depot Multiple-vehicle-type Bus Scheduling Problems in Practice.** Vol. 27, p. 507-523, 2005.

ITC2011. **Third international timetabling competition**. 2011. URL (http://www.utwente.nl/ctit/hstt/itc2011).

KRISTIANSEN, S.; SORENSEN, M.; STIDSEN, T. R. **Integer Programming for the Generalized High School Timetabling Problem**. Journal of Scheduling. Vol. 18, p. 377-392, 2014.

MIRHASSANI, S. A.; HABIBI, F. **Solutions Approaches to the Course Timetabling Problem.** Artificial Intelligence Review. Vol. 32, n. 2, p. 133-149, 2013.

POST, G.; KINGSTON, J.; AHMADI, S.; DASKALAKI, S.; GOGOS, C.; KYNGAS, J.; NURMI, C.; MUSLIU, N.; PILLAY, N.; SANTOS, H.; SCHAERF, A. **XHSTT: an XML Archive for High School Timetabling Problems in Different Countries**. Annals of Operation Research, p. 295-301, 2011.

SANTOS, H. G.; UCHOA, E.; OCHI, L. S.; MACULAN, N. **Strong Bounds with Cut and Column Generation for Class-teacher Timetabling.** Annals of Operation Research, Vol. 194, p. 399-412, 2012.

SAVINIEC, L.; SANTOS, M. O.; COSTA. A. M. **Parallel Local Search Algorithms for High School Timetabling Problems.** European Journal of Operation Research, Vol. 265, p. 81-98, 2018.

SCHAERF, A. **A survey of automated timetabling**. Artificial Intelligence Review, Vol. 13, p. 87-127, 1999.

SCHMIDT, G.; STROHLEIN, T. **Timetabling Construction – an Annotated Bibliography.** The Computer Journal, Vol. 23, p. 307-316, 1979.

SKOULLIS, V. I.; TASSOPOULOS, I. X.; BELIGIANNIS, G. N. **Solving the Hig School Timetabling Problem Using a Hibrid Cat Swarm Optimization Based Algorithm**. Applied Soft Computing, Vol. 52, p. 277-289, 2017.

SORENSEN, M.; DAHMS, F. H. W. **A Two-Stage Decomposition of High School Timetabling Applied to Cases in Denmark**. Computer & Operations Research, Vol. 43, p. 36-49, 2014.

SOUZA, M.; MACULAN, N. **Uma Heurística para o Problema de Programação de Horários em Escolas**. TEMA: Tendências em Matemática Aplicada e Computacional, SBMAC, vol. 2, n.1, p. 213-222, 2001.

TRIPATHY, A. **School Timetabling – A Case in Large Binary Integer Linear Programming.** Management Science, Vol. 30, p. 1473-1489, 1984.

VEENSTRA, M.; VIS, I. F. A. **School Timetabling Problem under Disturbance.** Computers & Industrial Engineering, Vol. 95, p. 175-186, 2016.

ZHANG, D.; LIU, Y.; M'HALLAH, R.; LEUNG, S. C. H. **A Simulated Anealing with a New Neighborhood Structure Based Algorithm for High School Timetabling Problems**. European Journal of Operational Research, Vol. 203, p. 550-558, 2010.

Apêndice A

TABELA 13 – RESULTADO DA INSTÂNCIA F PARA O MODELO 2 COM FUNÇÃO OBJETIVO 777

	Turma																									
Professor	Segunda					Terça					Quarta					Quinta					Sexta					
1	3	3	2	4	4	4	4	3	3	1	1	2	2	5	5	1	1	2	5	5						
2						10	10	8	8		10	10		8	8											
3	6	6	7	7		7	7		6	6																
4						14	14	13	9	9	13	13	9	11	11	12	12	14	14	13	12	12	9	11	11	
5	11	4	4	5	5	5	1	11		2	3	3	4	1	1						3	11	11	2	2	
6	7	7	13	6	6						7	7	13	14	14	13	13	12	12		14	14	12	12	6	
7																	7	7	6	6						
8	9	10	10	8	8						9	9				8	8	10	10	9						
9	1															2	2	1	8	8						
10																5	5	4			10	10	3	3	4	
11						9	9	7	12	12	11	11	14	6	6						13	13		7	14	
12	2	2	3	1	1						2	4	5			4	4	8	3	3	5	5	8	8	1	
13							6	6	7	7														6	7	
14	12	9	9	14	14						12	12	11	13	13	14	11	11	9	10			10	10	13	
15	8	11	11	10	10	2	2	5	5	8						9	3	3	4	4	9	1	1			
16																6	6		7	7						
17						13	13		14	14																
18													6	7	7						6	6	7			
19	14	12	12	9	9						14	14	12	10	10			13	11	11	11	9	13	13	10	
20	5	8	8	3	3			2	4	4	4	1	1	2	2						1	3	5	5	8	
21		13	14	11	12																					
22						1	5	4	2	3																
23	13	14	6	12	11																					
24	10	1	1	13	13			12	11	11						10	9	9	2	2	8	8	14	14	12	
25											5	5	3	4	4	3										
26	4	5	5	2	2	8	8	1	1	5			8	3	3						4	4	2	1	3	
27					7	11	11	9	10	10	6	6	10	12	12						7	7	6	9	9	
28						12	12	14	13	13										14						
29						3	3	10			8	8	7	9	9	7	10	5	1	1	2	2	4	4	5	
30						6										11	14	6	13	12						

GEMINADAS NÃO ATENDIDAS

PERIODOS DE OCIOSIDADE

TABELA 14 – RESULTADO DA INSTÂNCIA G PARA O MODELO 2 COM FUNÇÃO OBJETIVO 1055

	Turma																								
Professor	Segunda					Terça					Quarta					Quinta					Sexta				
1	16	16	17	17	1	2	2	3	3	17	16	16	17	4	2						4	4	1	1	3
2	10	10	9	9	11	11	11	8	10	10						9	7	7		11	7	8	8		
3	14	14	13	13	15						12	12	14	14	15					15	15	13	13	12	12
4	6	6	5	20	20						6	19	19			18	18	20	20	19	19	18	18	5	5
5	4	4	1	1	3						4	4	3	3	14	3	1	1	2	2				2	2
6	7	7	8	6	6						7	7	8	5	5	5	5	12	8	8			6	6	10
7									12	12	9	9	13	12	11	11	9	9	10	10	10	10	12	11	11
8	13	13	15	14	14	13	13	11	15	14						16	16	15	15	14				16	16
9	18	18				20	20	18	18	15	17	17	20	19	19						20	17	17	19	19
10						4	4	1	2	2						2	3	3	5	5	1	1	5	3	4
11	8	8	7	7	10											7	8	6	9	9	6	6	10	10	9
12	11	11	14	12	13						13	13	15	11	12						12	14	14	15	15
13						18	18	17	17	19						19	19	16	17	20	18	16	16	20	20
14						16	16	10		3	2	2	10	1	1	10	17	17	13	13	2	3	3	13	1
15						6	6	19	19	5	5	11	11		4	14	14	5	18	18	14	11	4	4	6
16	9	9	12	15	7	12	12	9	8	8				20	20						8	15	15	7	7
17	2	2	3	16	16	1	1	2	13	13				17	17	1	10	10	3	3					
18	19	19	18	4	4						11	5	18	6	6	6	11	14	14	4	5	5			
19						9	9	20	20	7						15	15	8	12	12	9	7	7	8	8
20	1	1	6	2	2	5		6	4	4	3	3	5	7	7										
21	12	12	10	10	9	8	8	13	11												13	9	11	14	14
22			19	19	17						15	15	16	16							17	20	20	18	18
23	3	3	11	11	19						10	10	2	2		12	12	19	1	1					
24	20	20	4	5	5		15	15	6	6	14	14	4	13	13										
25	17	17	16	18	18	7	7				8	8	9	9	16										
26						19	19	7	16	16	1	1	7	10	10	13	13	4	4						
27	5	5	2	8	8	17	17	14	14	20						20	2	11	11						
28						15	3	12	9	9	18	18	12	15	3				6	6					
29						10	10	16	1	1						4	4	13	7	7	16	19	19		13

30						14	14	5	5	11				8	8						11	2	2	17	17
31	15	15	20	3	12						20	6	6	18	18						3	12	9	9	
32						3	5	4	7	18	19	20	1		9	8	6	2	16	17					
33																17	20	18	19	16					

GEMINADAS NÃO ATENDIDAS

PERIODOS DE OCIOSIDADE