PROBLEMA DE ALOCAÇÃO DE PROFESSORES DO ENSINO MÉDIO CONSIDERANDO RESTRIÇÕES DE INTERVALO

Alisson Segatto de Souza

José Eduardo Pécora Jr

Adriana Alves Fressato

Gustavo Valentin Loch

segatto.souza@gmail.com

pecorajr@gmail.com

adriana.afres@gmail.com

gustavo.gvalentim@gmail.com

Universidade Federal do Paraná

Campus III – Centro Politécnico – Av. Cel. Francisco H. dos Santos – 210 – Jardim das Américas – Curitiba, 81531-970, Paraná, Brasil.

Resumo. *O problema de construir uma grade horária escolar de ensino médio (High School Timetabling - HSTT) consiste em reunir professores, classes e salas em um determinado período, construindo a escala semanal de uma instituição de ensino. O grande número de variáveis e restrições tornam a resolução manual deste problema muito difícil. Além disso, para problemas muito grandes não é possível encontrar boas soluções em um tempo computacional viável via métodos exatos, tornando necessária a utilização de heurísticas. Neste trabalho, usaremos a heurística de fix-and-optimize para resolver um problema variante do HSTT que busca minimizar os períodos de ociosidade e gerar o maior número de aulas geminadas possíveis, considerando os intervalos escolar no processo. Também apresentamos duas novas abordagens ao problema: na primeira adiciona-se uma nova restrição ao modelo diminuindo a área de busca do algoritmo e na segunda modificamos a heurística de fix-and-optimize para diminuir o número de vizinhanças exploradas. As duas novas abordagens foram capazes de obter melhores soluções em menor tempo computacional para os cinco problemas-testes estudados.*

Keywords: *High School Timetabling, fix-and-optimize, heurística.*

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

INTRODUÇÃO

O HSTT é a tarefa de criar uma grade horária escolar, agendando professores, classe e salas em um determinado período e vem sendo exaustivamente estudado há muitos anos. Está grade horária influência na vida de centena de pessoas como professores, funcionários e alunos, de forma que sua composição reflete na qualidade do ensino. Porém, construir esta grade escolar manualmente é uma tarefa muito difícil, chegando a ser impossível para grandes instituições, tornando indispensável o uso de algoritmos para a automatização deste processo.

É comum para muitos países, inclusive no Brasil, que professores trabalhem em mais de uma escola ao mesmo tempo, tornando-se indisponíveis para a instituição de ensino em certos períodos. Portanto é essencial que as aulas de um professor sejam ofertadas no menor número de dias possíveis, eliminando seus períodos de ociosidade. Outro fator importante para a qualidade de ensino é que aulas de uma mesma disciplina que possuam mais de uma ocorrência no mesmo dia sejam aplicadas em sequência, permitindo que os professores explorem melhor o tempo que possui com cada turma, chamadas de aulas geminadas. Este problema é chamado de *Class-Teacher Timetabling Problem with Compactness Requeriments* (CTTPCR), e é classificado como NP-Completo. (Dorneles et al., 2014; Demirovic & Musliu, 2016; Even et al., 1975).

O CTTPCR é um problema de alta complexidade, possuindo inúmeras variáveis e restrições. Estas restrições podem ser separadas em *hard* e *soft*. Restrições *hard* são as regras que precisam ser respeitadas para garantir a factibilidade da solução, estruturando o problema. Já as restrições *soft* se referem as preferencias que existem sobre o problema, podendo serem desrespeitadas quando necessário, mas cumprir estas restrições nos garante uma solução de alta qualidade.

Neste trabalho analisamos a influência do intervalo escolar sobre as aulas geminadas. Este intervalo pode causar uma quebra no ritmo das aulas tanto para alunos quanto para professores, portanto não consideramos as aulas aplicadas nos períodos antecessor e sucessor a estes intervalos como aulas geminadas. Também propomos uma nova restrição ao modelo do CTTPCR proposto por Dorneles et al (2014) buscando diminuir o espaço de busca e uma variação para o método de *fix-and-optimize* diminuindo o número de vizinhanças para cada problema. Como resultado fomos capazes de obter melhores resultados em todas os cinco problemas-testes estudados.

O artigo está organizado da seguinte maneira. Na seção 1 teremos uma revisão de literatura do problema de timetabling. Na seção 2 apresentamos o problema, as notações usadas, restrições e o modelo usado. Na seção 3 apresentamos o método usado para a resolução do problema. Na seção 4 apresentamos os principais resultados obtidos. Por fim, na seção 5 faz-se uma breve conclusão.

1 REVISÃO DE LITERATURA

O HSTT é um problema bem conhecido da literatura. Vários métodos de solução já foram propostos para tentar resolvê-lo, sendo que os métodos heurísticos têm se destacado devido ao alto número de variáveis e restrições que caracterizam este problema.

De Werra (1971) apresentou um modelo simples para a resolução do HSTT aplicado as escolas suíças, contendo 50 classes e 80 professores. O problema consistia em não alocar

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering
P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

professores e classes em uma mesma aula ao mesmo tempo, e pode ser resolvido em tempo polinomial. Os resultados foram considerados muito bons para a época deixando apenas 5% das aulas sem serem atendidas. Mais tarde, Even et al. (1975) estudam a complexidade do HSTT e o resolvem utilizando um algoritmo de fluxo integral multi-commodity. Com isso os autores mostraram que para os problemas-teste em que os professores podem não estar disponível em todos os períodos o problema é classificado como NP-Completo, mesmo sendo resolvido em tempo polinomial no problema-teste estudado.

Drexl & Salewski (1997) propõem um novo modelo de HSTT, considerando aulas com diferentes tempos de duração. Para analisar o modelo foram gerados problemas randômicas e resolvidos usando algoritmo genético. Os autores destacam que os problemas de HSTT podem variar dependendo da instituição ou país, restringindo o trabalho de muitos pesquisadores. Devido a isto, Schaerf (1999) apresenta várias heurísticas para a solução do HSTT obtendo bons resultados.

Souza e Maculan (2000) são os primeiros a abordar o CTTPCR, que é um problema originário das escolas brasileiras. Para soluciona-lo os autores utilizam um algoritmo de busca local aliado à algumas metaheurísticas, como Busca Tabu, GRASP, Simulated Annealing, entre outras. Mais tarde, Santos et al. (2010) apresentam um algoritmo de corte e geração de colunas para resolver o CTTPCR que foi capaz de encontrar lower bounds menores para o problema estudado.

Devido às 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 proposta 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, com novos trabalhos utilizando parte ou todos os problemas-teste da competição, propondo novas abordagens e heurísticas para solucionar os problemas da melhor forma, o mais rápido possível. (Dorneles et al., 2014; Kristiansen et al., 2014; Fonseca et al., 2015; Demirovic E Musliu, 2016; Dorneles et al., 2017).

Gunawan et al. (2012) propõem uma solução para o problema de grade horária universitária, onde professores devem ser designados e cursos agendados. Uma solução inicial é gerada através de um algoritmo de relaxação lagrangeana, e então a solução é melhorada usando o simulated anealing. O algoritmo proposto é testado nos problemas-teste da 2º ITC e tem resultados muito bons.

Dorneles et al. (2014) apresentaram uma nova abordagem aplicando o algoritmo Fix-and-Optimize juntamente ao método de Variable Neighborhood Descent para solucionar doze problemas-teste, sendo sete delas da ITC-2011 e as outras cinco provenientes de adaptações dos problemas-teste anteriores para o caso do CTTPCR. Dos doze problemas estudados, ele foi capaz de encontrar sete das melhores soluções conhecidas, sendo que três destas foram 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 os problemas-teste, focando no CTTPCR. O método encontrou cinco novos lower bounds para os problemas estudados. Desta forma, duas das soluções encontradas por Dorneles et al. (2014) se mostraram soluções ótimas.

Demirovic e Musliu (2016) propõem um algoritmo de MaxSAT-based Large Neighborhood para solucionar o High School Timetabling (HSTT). Este algoritmo direciona uma solução inicial até um ótimo local e então realiza uma 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 utilizados 27 problemas da ITC-2011. O método se mostrou muito veloz e foi capaz de encontrar novas melhores soluções para 4 dos problemas estudados.

2 DESCRIÇÃO DO PROBLEMA E MODELO

O CTTPCR tem o objetivo de unir professores T, classes C e salas em uma aula, sendo que uma classe é formada pelo conjuntos de alunos de um determinado curso que seguem o mesmo cronograma diário. Para isto foi criado um conjunto de eventos E que é formado por um professor $t \in T$ e uma classe $c \in C$, considerando que cada classe, professor, sala e carga horária permanecem fixos, apenas faltando designa-los ao respectivo turno, que são formados pelos períodos $p \in P$ que compõem um dia $d \in D$, onde cada período possui a mesma duração.

Cada evento tem uma carga horária máxima diária, o que permite que um professor e uma classe possam se reunir em uma aula mais de uma vez por dia, mas limita este número de encontros, sendo estas chamadas de aulas geminadas. (Dorneles et al., 2014).

Nas escolas brasileiras é comum que cada dia seja composto por cinco períodos, existindo um intervalo entre o período três e quatro. Portanto, aulas consecutivas envolvendo estes dois períodos não podem ser consideradas aulas geminadas, sendo que há uma quebra no ritmo da aula devido a este intervalo. Ao gerar seu modelo, Dorneles et al. (2014) considera os cinco períodos do dia como sendo continuo, e releva este intervalo entre os períodos três e quatro. Neste trabalho iremos adicionar a impossibilidade de aulas geminadas entre estes períodos e analisar a qualidade da solução obtida.

Neste trabalho iremos considerar as seis seguintes restrições *hards*, propostas por Dorneles et al. (2014):

i) A carga horária do evento deve ser respeitada

ii) Um professor não pode ser agendado em mais de uma sala no mesmo período

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

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

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

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

Além destas, são consideradas as seguintes restrições *soft*, responsáveis pela qualidade da solução.

a) Evitar os períodos de ociosidade dos professores.

b) 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.

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

Tabela 1. Notação usada

Símbolo	Definição
Conjuntos	
$d \in D$	Dias da semana
$p \in P$	Períodos do dia
p'	p sem os últimos dois períodos do dia
$t \in T$	Conjunto de professores
$c \in C$	Conjunto de classe
$e \in E$	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', $n \in P : n \geq m + 2$
Q	Conjunto de tuplas (m, n) para p', $n \in P : n \geq m$
SG_e	Conjunto de períodos em que cada evento e pode começar uma aula geminada $\left(SG_e = \{(d,p): d \in D, p \in P \; e \; p < \lvert P \rvert \; e \; p \neq 3, \; V_{edp} + V_{edp+1} = 2\}\right)$. 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)

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

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

Adaptado de Dorneles et al. (2014)

O modelo apresentado por Dorneles et al. (2014) foi o primeiro a apresentar (vi) como restrição *hard* obtendo um ótimo resultado comparado aos resultados já existentes na literatura. Também iremos incluir ao modelo a restrição (19), onde minimizamos previamente os problemas-teste 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

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

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

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

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

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

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

$$\sum_{e \in E_c} x_{edp} \leq 1 \ \forall\, c, d, p \tag{7}$$

$$b_{edp} \geq x_{edp} - x_{edp-1} \ \forall\, e, d, p : p > 1 \tag{8}$$

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

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

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

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

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

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

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

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

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

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

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

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

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

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

A função objetivo (1) é compostas de três partes, que representam as três restrições soft do problema (a), (b) e (c), respectivamente, sendo que a restrição (a) 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 (vi). O conjunto de restrições (10) e (11) força aulas geminadas quando a variável g_{edp} for igual a um, desconsiderando o período $p = 3$, pois aulas lecionadas nos períodos {3,4} não são considerados como aulas geminadas. 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 possua aula no período n.

A restrição (19) foi proposta por nós. Ela é calculada através do modelo original, mas considerando apenas a segunda parte da função objetivo, referente a (b), para encontrar LB, parâmetro que representa o número mínimo de dias que devem ser trabalhados pelos professores. Este resultado será usado na restrição proposta com o objetivo de reduzir a área 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.

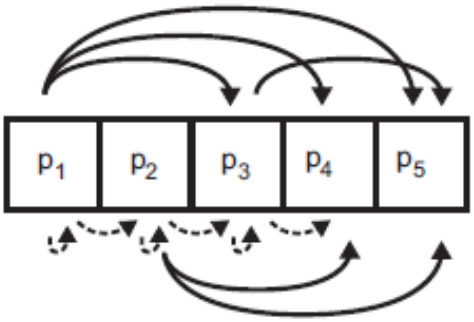

Figura 1. Grafo dos periodos de ociosidade. (Dorneles et al. 2014).

3 FIX-AND-OPTIMIZE COM REDUÇÃO DE SUBPROBLEMAS

O *fix-and-optimize* utilizado neste trabalho é baseado nas decomposições propostas por Dorneles et al. (2014), sendo feitas para classes e professores. Nesta seção iremos apresentar uma variação da heurística de *fix-and-optimize* aplicada ao problema de HSTT.

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. Combinando as vizinhanças, obtemos uma sequência de vizinhanças a ser seguida pelo algoritmo. Por exemplo, a sequência $(CD,2),(TD,2),\ldots,(CD,|C|),(TD,|T|)$ 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).

A decomposição tem início partindo de uma solução factível, gerada a partir de um *solver* que gera uma solução factível para o problema sem considerar a função objetivo. Após esta etapa todas as variáveis x_{edp} tem seu valor fixado e em seguida um conjunto destas variáveis são liberadas para a otimização, gerando o primeiro subproblema. Então este processo de fixação e liberação é repetido a cada iteração, de acordo com a estratégia de decomposição escolhida. As decomposições são feitas em sequência, de modo que ao final do período de processamento todas as vizinhanças tenham sido exploradas, e dura o tempo que o solver levar para encontrar a primeira solução factível do subproblema ou o tempo máximo de processamento, que neste trabalho foi fixado em 30 segundos. Porém, esta forma de exploração busca melhores soluções em combinações que já estão cumprindo todos os requisitos *soft* do problema, não havendo melhorias a serem encontradas. O *fix-and-optimize* proposto irá evitar subproblemas onde não há períodos de ociosidade a serem corrigidos.

Algoritmo FixAndOptimize

1. $x^* \leftarrow SoluçãoInicial$
2. **Se** $x^* = \emptyset$ **então**
3. Retorna $\emptyset$
4. Fim

5. Para todo $(\tau, k)\ \epsilon$ 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. **Se** (τ, k) $possui\ periodos\ de\ ociosidade$ **então**
12. $x \leftarrow Resolver(x^*, R)$
13. Fim
14. Se $x < x^*$ então
15. $x^* \leftarrow x$
16. $SemMelhorias \leftarrow 0$
17. Senão
18. $SemMelhorias \leftarrow SemMelhorias + 1$
19. **Fim**
20. **Se** $TL\ for\ atingido$ **então**
21. $Retorna\ x^*$
22. Fim
23. $s \leftarrow (s\ \boldsymbol{mod}\ count) + 1$
24. **Até** $SemMelhorias = count$
25. Retorna x^*

Algoritmo 1. Pseudo código do *fix-and-optimize* com redução de subproblemas. (Adaptado de Dorneles et al. 2014).

Algoritmo $Decompor(\tau, k, s)$

1. **Caso** $\tau = CD$
2. $R \leftarrow \{x_{edp}: c\ \epsilon\ Subconjuntos(C, k, s), e\ \epsilon\ E_c, d\ \epsilon\ D, p\ \epsilon\ P\}$
3. **Caso** $\tau = TD$
4. $R \leftarrow \{x_{edp}: c\ \epsilon\ Subconjuntos(C, k, s), e\ \epsilon\ E_T, d\ \epsilon\ D, p\ \epsilon\ P\}$
5. Retorna R

Algoritmo 2. Função de decomposição. (Adaptado de Dorneles et al. 2014).

Algoritmo $QtdeSubProblema(\tau, k)$

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

Algoritmo 3. Função que calcula a quantidade de subproblemas de uma vizinhança. (Adaptado de 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.

Partindo da solução inicial, o algoritmo percorre todas as vizinhanças previamente definidas e o decompõe em um número finito de subproblemas. Estes subproblemas são verificados para descobrir se existem períodos de ociosidade a serem corrigidos e, caso haja, o *solver* tem 30 segundos para buscar uma solução que melhore a função objetivo. Após este período, caso o valor da função objetivo tenha sido melhorada, o algoritmo guarda esta nova solução para ser usada nas próximas iterações e a variável $SemMelhorias$ é reiniciada com o valor 0. Caso contrário, $SemMelhorias$ é incrementada. (Algoritmo 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 o ptoblema-teste estudado.

Nas vizinhanças em que $k \geq 3$, um grande número de variáveis são liberadas, causando assim um aumento no tempo computacional de resolução dos subproblemas. Em alguns casos, o solver não é capaz de encontrar uma solução factível dentro do tempo limite de 30 segundos. Dado que o critério de parada do algoritmo é o tempo total atingido, é desejável obter o maior número de subproblemas resolvidos 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, as restrições soft envolvidas são a redução de janelas e minimização de dias de trabalho. 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 as janelas entre as aulas. Caso os professores a serem liberados não possuam janelas, não há melhoria a ser obtida no valor da função objetivo e a resolução do subproblema gerado pode ser evitada. Para economizar tempo computacional de verificação de janelas, foi criado um procedimento para que, caso um subproblema seja evitado por não haver janelas, o próximo subproblema tenha pelo menos um professor com janelas liberado para otimização. Desta forma, o subproblema só será resolvido caso algum professor tenha ao menos uma janela.

4 RESULTADOS COMPUTACIONAIS

As modificações propostas neste trabalho possuem o objetivo de diminuir o tempo computacional do modelo proposto por Dorneles et al. (2014). Para isto, usa-se o solver GUROBI 7.0.2 com os algoritmos implementados em VB.net. Os testes foram feitos em um computador com CPU 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 usados no modelo matemático são 9, 3 e 1 respectivamente.

Os problemas-teste usados foram retiradas do ITC2011, e foram adaptadas para a aplicação do CTTPCR por Dorneles et al. (2014). A Tabela 2 apresenta as características de cada problema-teste, onde as colunas $|D|$, $|P|$, $|T|$, $|C|$ e $|E|$ representam o número de dias, períodos, professores, classes e eventos, respectivamente. E por último as colunas $\sum_{e \in E} MG_e$ e $\sum_{e \in E} R_e$ representam a quantidade mínima de aulas geminadas exigidas e a carga total de trabalho, sendo que a quantidade mínima de aulas geminadas consideradas é o maior possível para cada evento. Todos os professores se encontram disponíveis em todos os períodos

Tabela 2. Características dos problemas-teste estudados. Adaptado de Dorneles et al. (2014).

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

Para a aplicação do *fix-and-optimize* adotou-se a estratégia de fixar classes e em seguida professores, descrito por Dorneles et al. (2014) como estratégia F8, e que gerou os melhores resultados.

- F8- (CD, 2); (TD, 2); ...; (CD; ∞); (TD; ∞)

Está estratégia consiste em explorar as vizinhanças do conjunto de classes com cardinalidade $k = 2$, até que toda a vizinhança tenha sido explorada, e então fazer o mesmo para o conjunto de professores. Após as vizinhanças terem sido completamente exploradas dois a dois aumentamos $k = 3$, e repete-se o processo até que o tempo limite do problema-teste seja atingido. Repare que toda vez que aumentamos k o tamanho do problema e o número de vizinhanças crescem, aumentando o esforço computacional e o tempo de processamento. O

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

tempo total de processamento foi fixado em 10 minutos para o problema-teste A, 30 minutos para os problemas-teste D e F e 60 minutos para os problemas-teste E e G.

Para comparação e avaliação dos resultados obtidos usamos o modelo proposto por Dorneles et al. (2014) resolvido de forma clássica, o modelo adaptado por nós com o acréscimo da restrição (19) e o modelo adaptado por nós com a estratégia de redução de subproblemas (RRS).

Tabela 3. Resultados computacionais.

	Clássico		Com Restrição		Com Restrição e Redução de Subproblema (RRS)	
Problema	Tempo (s)	F.O.	Tempo (s)	F.O.	Tempo (s)	F.O.
A	201,578	201	121,108	200	121,997	200
D	1801,126	657	1108,439	655	1419,343	654
E	1213,702	782	3430,958	777	3193,768	781
F	1800,404	780	1543,469	780	1432,905	788
G	3218,093	1109	3586,636	1094	3573,373	1081

Podemos ver pela Tabela 3 que a restrição sugerida e a estratégia de evitar subproblemas foram superiores que o *fix-and-optmize* clássico em quase todas os problemas-teste, sendo apenas para o problema-teste F o RRS se mostrou inferior.

Em geral, a restrição proposta faz com que a solução inicial obtida pelo *solver* seja muito melhor do que a do método clássico. A estratégia de evitar subproblemas só seleciona os subproblemas que possuem períodos de ociosidade, porem alguns dos subproblemas evitados podem possuir aulas geminadas a serem atendidas, que acabam prejudicando a função objetivo.

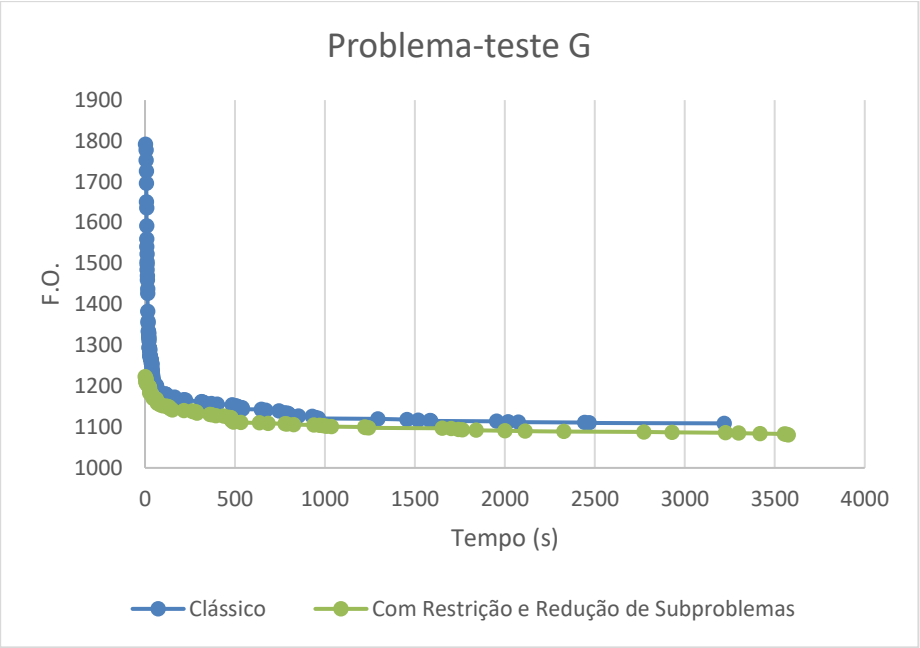

Figura 2. Comparação da curva entre o *fix-and-optimize* clássico e com restrição e redução de subproblemas

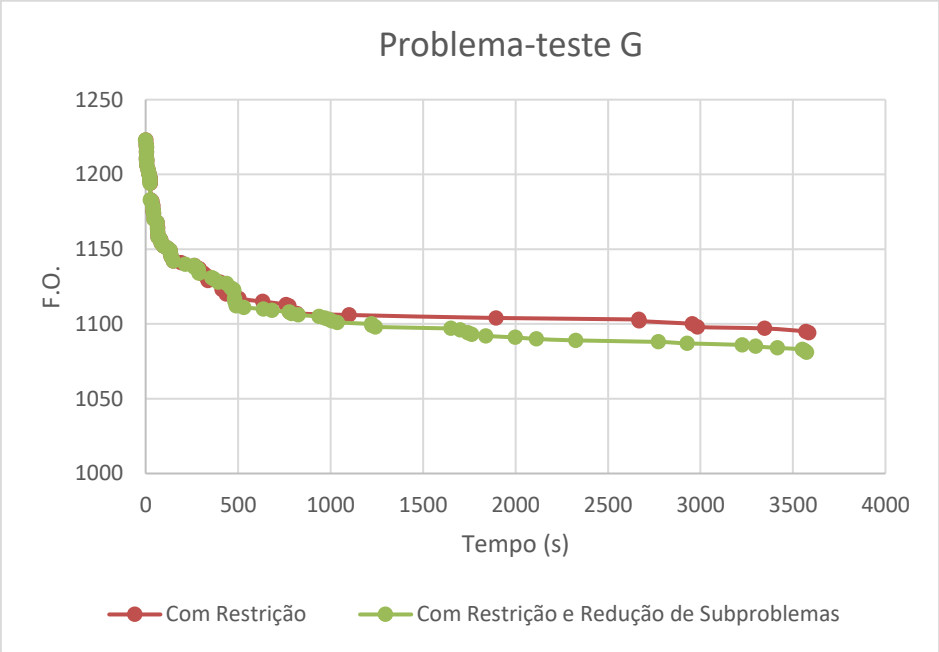

Figura 3. Comparação da curva entre o *fix-and-optimize* com restrição e com restrição e redução de subproblemas para o problema-teste G

Na Figura 2 e Figura 3 podemos ver o decaimento da função objetivo com relação ao tempo de processamento.

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

Na Figura 2 é possível observar que a formulação clássica proposta por Dorneles et al. (2014) possui uma solução inicial com valor mais alto do que a encontrada pelo RSS, isso ocorre devido à restrição (19) que restringe ainda mais a região de busca do modelo. Também é possível observar que mesmo possuindo um declínio muito alto, o modelo clássico não foi capaz de superar o RSS, se mantendo com um valor da função objetivo mais alto durante todo o tempo de processamento.

Já na Figura 3 a comparação é feita em relação ao modelo proposto e ao RRS. Como o modelo usado é o mesmo, divergindo apenas a estratégia de busca, era esperado que a solução inicial fosse a mesma, com as curvas começando a divergir em aproximadamente 250 segundos. Para o problema-teste G apresentado na figura, o RSS começa a decair mais rapidamente, mas isso não é um padrão, variando para cada problema-teste.

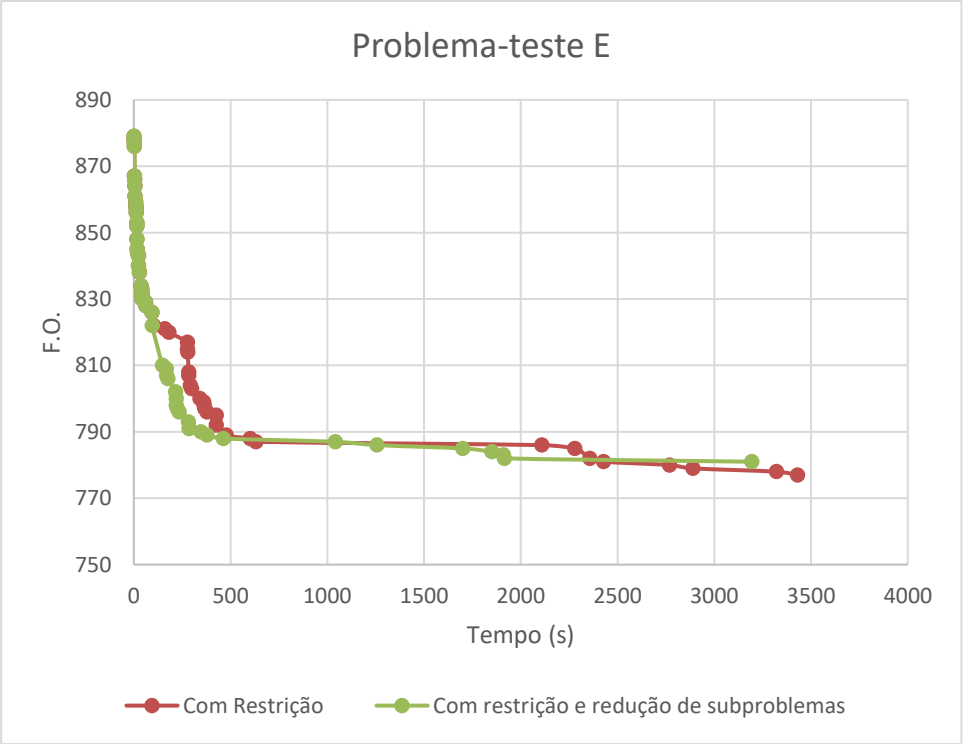

Figura 4. Comparação da curva entre o *fix-and-optimize* com restrição e com restrição e redução de subproblemas para o problema-teste E

Na Figura 4 vemos um caso em que a curva com restrição acaba superando o RRS. Podemos notar que assim como observado na Figura 3 o RRS decai mais rápido do que a curva com restrição, mas por fim a curva com restrição acaba vencendo. Isto pode ocorrer justamente pelo fato dos subproblemas com geminadas não atendidas não estarem sendo processadas, variando de problema para problema.

5 CONCLUSÃO

Neste trabalho foram apresentado duas abordagens alternativas para a resolução do CTTPCR a partir da heurística de *fix-and-optimize*, onde consideramos que exista um intervalo entre o terceiro e quarto período do dia, impedindo que aulas geminadas sejam aplicadas nestes períodos.

Na primeira alternativa é proposta uma alteração ao modelo proposto por Dorneles et al. (2014) onde acrescentamos uma restrição ao modelo buscando diminuir o espaço de busca do algoritmo. Na segunda alternativa mudamos a estratégia de busca da heurística de *fix-and-optimize* para evitar subproblemas que já estão cumprindo as restrições *soft* do problema e que tenham pouco a acrescentar a função objetivo.

Conseguimos melhores resultados para todos os cinco problemas-teste estudados sendo que em apenas um deles a estratégia de redução de subproblemas não foi capaz de superar a abordagem clássica, garantindo duas boas alternativas para o estudo do CTTPCR.

Como trabalhos futuros propomos que sejam analisadas mais estratégias para a aplicação do *fix-and-optimize*, buscando melhorias na inteligência por trás do método.

AGRADECIMENTOS

Agradecemos ao grupo de pesquisa GTAO por todo apoio, suporte e amizade.

REFERÊNCIAS

De Werra, D., 1971. Construction of school timetables by flow methods. *INFOR*, vol. 9, pp. 12-22.

Demirovic, E., Musliu, N., 2016. MaxSAT-based large neighborhood search for high school timetabling. *Computers & Operations Research*, vol. 78, pp. 172-180.

Dorneles, A. P., Araujo, O. C. B., Buriol, L. S., 2014. A fix-and-optimize heuristc for the high school timetabling problem. *Computers & Operations Research*, vol. 52, pp. 29-38.

Dorneles, A. P., Araujo, O. C. B., Buriol, L. S., 2016. A column generation approach to high school timetabling modeled as a multicommodity flow problem. *European Journal of Operational Research*, vol. 256, pp. 685-695.

Drexl, A., Salewski, F., 1997. Distribution requirements and compactness constrains in school timetabling. *European Journal of Operational Research*, vol. 102, n.1, pp. 193-214.

Even, S., Itai, A., Shamir, A., 1975. On the complexity of time table and multi-commodity flow problems. *Proceedings of the 16th annual symposium on foundations of computer science. SFCS '75; Wasington, DC, USA: IEEE Computer Society*, pp. 184-193.

Fonseca, G. H. G., Santos, H. G., Carrano, A. G., 2015. Late acceptance hill-climbing for high school timetabling. *Journal of Scheduling*, vol. 19, pp. 453-465.

Gunawan, A., Ming Ng, K., Poh, K. L., 2012. A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem. *Computers & Operations Research*, vol. 39, pp. 3074-3088.

Kristiansen, S., Sorensen, M., Stidsen, T. R., 2014. Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, vol. 18, pp. 377-392.

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

Post, G., Kingston, J., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H. G., Schaerf, A., 2011. XHSTT: an XML archive for high school timetabling problems in different countries. *Annals of Operation Research*, vol 218, pp. 295-301.

Santos, H. G., Uchoa, E., Ochi, L. S., Maculan, N., 2012. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operation Research*, vol. 194, pp. 399-412.

Schaerf, A., 1999. A survey of automated timetabling. *Artificial Intelligence Review*, vol. 13, pp. 87-127.

Souza, M. J. F.; Maculan, N.; Ochi, L. S. Uma Heurística para o Problema de Programação de Horários em Escolas. TEMA: Tendências em Matemática Aplicada e Computacional, SBMAC, vol. 2, n.1, pp. 213-222, 2001.