SMNE UFPR
Simpósio de Métodos Numéricos em Engenharia
25 a 27 de outubro, 2017

Performance de dois *solvers* na Resolução da Metaheurística Fix and Optimize Aplicado ao Problema de High School Timetabling

Alisson Segatto de Souza
Mestranda do Programa de Pós-graduação em Métodos Numéricos em Engenharia
UFPR
Curitiba, Brasil

José Eduardo Pécora Junior
Professor Doutor do departamento de Administração Geral e Aplicada
UFPR
Curitiba, Brasil

Gustavo Valentin Loch
Professor Doutor do departamento de Administração Geral e Aplicada
UFPR
Curitiba, Brasil

Adriana Alves Fressato
Mestranda do Programa de Pós-graduação em Métodos Numéricos em Engenharia
UFPR
Curitiba, Brasil

***Resumo*— O problema da Grade Horária Escolar do Ensino Médio (High Scholl Timetabling – HSTT) consiste em construir a escala semanal de uma instituição de ensino. O problema é caracterizado por um grande número de variáveis e restrições, fazendo que para instancias de médio e grande porte torna-se impossível resolve-lo manualmente, ou em alguns casos até métodos exatos são incapazes de encontrar boas soluções em um tempo computacional viável, obrigando os pesquisadores adotarem o uso de heurísticas. Neste trabalho testaremos dois *solvers* de programação inteira mista para a solução da heurística de *fix-and-optimize*. Esta heurística é um método que consiste em fixar uma parte das variáveis, criando subproblemas muito menores, possibilitando sua solução através de métodos exatos e então repetindo o processo de fixação até todas as variáveis serem deixadas livres para a otimização. Para solucionar a parte exata da heurística usaremos os *solvers* GUROBI 7.0.2 e CPLEX 12.6.2 e então comparar os resultados para avaliar qual seria a melhor opção a ser usada neste tipo de problema. Como resultado o *solver* GUROBI teve um melhor desempenho em quatro das cinco instancias estudadas.**

Palavras-chave*— *Comparação; Timetabling; Fix-and-optimize;

I. INTRODUÇÃO

O *High Schoool Timetabling* (HSTT) é o problema de construção da uma grade escolar, relacionando professores, alunos e salas de aula em um período. A qualidade desta grade horaria influencia tanto a vida dos professores quanto a dos alunos, o que torna seu planejamento essencial para qualquer instituição de ensino.

É muito comum que professores trabalhem em mais de uma escola simultaneamente. Por este motivo é importante que as aulas dos professores sejam ministradas no menor número possível de dias. O *Class-Teacher Timetabling Problem with Compactness Requeriments* (CTTPCR) é um subproblema do HSTT e busca não apenas compactar os dias de trabalho dos professores, mas também eliminar seus períodos de ociosidade, buscando aproveitar da melhor forma possível o tempo que este permanece na escola. Outra

característica importante deste problema é que quando o mesmo professor tem mais de uma aula na mesma turma, estas devem ser ministradas em sequência, permitindo que o tempo que os professores passam com a turma sejam melhor explorados. [11] e [14].

O CTTPCR pertence à classe dos NP-completo, possuindo alta complexidade e um grande número de variáveis e restrições. [2]. Estas restrições podem ser separadas em *hard*: aquelas que precisam ser respeitadas para garantir a factibilidade da solução e *soft*: que representam as preferências sobre o problema, podendo ser desrespeitadas se necessário, causando uma penalidade na função objetivo.

Neste trabalho iremos avaliar o rendimento de dois *softwares* de programação inteira mista (MIP), GUROBI 7.0.2 e CPLEX 12.6.2, para a solução do problema de CTTPCR utilizando a heurística de *fix-and-optimize*. Esta heurística consiste em fixar o valor de parte das variáveis do problema para diminuir o tamanho do problema, e então repetir o processo alternando as variáveis a serem fixadas.

O artigo está organizado da seguinte maneira. Na seção II tem-se uma revisão de literatura do problema de timetabling. Na seção III descreve-se o problema, as notações usadas e o modelo usado. Na seção IV apresenta-se o método de *fix-and-optimize* usado para a resolução do problema. Na seção V são expostos os principais resultados obtidos. Por fim, na seção VI faz-se uma breve conclusão.

II. Revisão Bibliografica

O HSTT já vem sendo estudado há décadas e muitos métodos heurísticos e exatos foram propostos para sua resolução, 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.

O problema básico do *timetabling* consiste em reunir professores e salas, no que seria uma aula, de forma em que nenhum professor ou classe seja designado para mais de uma aula ao mesmo tempo, sendo resolvido em tempo polinomial [1]. No entanto, para casos em que os professores podem não estar disponíveis em alguns períodos, o *timetabling* é classificado como NP-completo [2].

As restrições do *timetabling* podem variar para cada instituição. Devido a isto, trabalhos de muitos pesquisadores acabam ficando restritos a uma única instituição, ou país, fazendo com que um método que se mostre muito efetivo para um problema não tenha o mesmo desempenho se aplicado em uma instância diferente [3]-[4]. Portanto, muitas heurísticas foram apresentadas obtendo boas soluções para casos gerais [4].

O CTTPCR é um problema originário do Brasil e foi inicialmente abordado por [5] que utilizaram um algoritmo de busca local aliado à 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, [9] 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) [7], [14]. 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 [8].

Após a ITC-2011 as pesquisas na área têm se intensificado, com novos trabalhos utilizando parte ou todas as instâncias da competição, propondo novas abordagens e heurísticas para solucionar os problemas da melhor forma, o mais rápido possível [10] – [14].

Referência [11] apresenta 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 foram novas melhores soluções conhecidas. Posteriormente, em [13] utilizaram 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 [11] se mostraram soluções ótimas.

Referência [14] 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 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 muito veloz e foi capaz de encontrar novas melhores soluções para 4 das instâncias estudadas.

III. Descrição do problema e modelo

Nesta seção apresentaremos o modelo para o CTTPCR apresentado por [11], e que será usado para a análise de desempenho dos *solvers*.

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 cronogramas 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 $t \in T$ pode não estar disponível em certos períodos. [11].

Como entrada para o programa 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 horaria 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 aulas devem ser consecutivas e são chamadas aulas duplas ou geminadas. [11]. Nas instâncias estudadas neste trabalho, consideramos um número mínimo de aulas geminadas para cada evento.

Para que haja factibilidade na solução, [11] 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.

TABELA I. NOTAÇÃO USADA PARA A CONSTRUÇÃO DO MODELO

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 $\left(SG_e = \{(d,p): d \in D, p \in P \ e \ p < \lvert P \rvert, V_{edp} + V_{edp+1} = 2\}\right)$. 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
Variáveis	
x_{edp}	Variável binaria que indica se o evento e foi designado para o período (d,p)
y_{td}	Variável binaria que indica se o professor t foi escalado para ao menos uma aula no dia d
g_{edp}	Variável binaria que indica se o evento e possui uma aula geminada começando no período (d,p)
G_e	Variável inteira que indica o número de lições geminadas faltam para ser atendidas
b_{edp}	Variável binaria que indica se o evento e possui uma aula no período (d,p) e não no período $(d,p-1)$
z_{tdmn}	Variável binaria que indica se o professor t possui períodos de ociosidade no dia d entre o período m e n

Fonte: Adaptado de [11].

O modelo apresentado por [11] foi o primeiro a considerar o requerimento H6 como restrição *hard*, obtendo excelentes resultados quando comparadas aos já existentes na literatura.

$$\text{Min} \quad O_1 + O_2 + O_3 \tag{1}$$

$$O_1 = \sum_{t \in T} \sum_{d \in D} \sum_{(m,n) \in U} \omega_t (n - m - 1) z_{tdmn} \tag{2}$$

$$O_2 = \sum_{t \in T} \sum_{d \in D} \gamma_t y_{td} \tag{3}$$

$$O_3 = \sum_{e \in E} \delta_e G_e \tag{4}$$

Sujeito a

$$\sum_{d \in D} x_{edp} = R_e \quad \forall e \tag{5}$$

$$\sum_{p \in P} x_{edp} = L_e \quad \forall e, d \tag{6}$$

$$x_{edp} \leq V_{edp} \quad \forall e, d, p \tag{7}$$

$$\sum_{e \in E_t} x_{edp} \leq y_{td} \quad \forall t, d, p \tag{8}$$

$$\sum_{e \in E_t} \sum_{p \in P} x_{edp} \geq y_{td} \quad \forall t, d \tag{9}$$

$$\sum_{e \in E_c} x_{edp} \leq 1 \quad \forall c, d, p \tag{10}$$

$$b_{edp} \geq x_{edp} - x_{edp-1} \quad \forall e, d, p : p > 1 \tag{11}$$

$$\sum_{p \in P : p > 1} b_{edp} + x_{ed1} \leq 1 \quad \forall e, d \tag{12}$$

$$g_{edp} \leq x_{edp} \quad \forall e, (d, p) \in SG_e \tag{13}$$

$$g_{edp} \leq x_{edp+1} \quad \forall e, (d, p) \in SG_e \tag{14}$$

$$G_e \geq MG_e - \sum_{(d,p) \in SG_e} g_{edp} \quad \forall e \tag{15}$$

$$\sum_{d \in D} y_{td} \geq \max\left\{ \left\lceil \frac{\sum_{e \in E_t} R_e}{|P|} \right\rceil, \max_{e \in E_t} \left\{ \left\lceil \frac{R_e}{L_e} \right\rceil \right\} \right\} \quad \forall t \tag{16}$$

$$\sum_{(m,n) \in Q} z_{tdmn} = y_{td} \quad \forall t, d, m \in P : m \leq 3 \tag{17}$$

$$\sum_{(m,n) \in Q} z_{tdmn} \leq y_{td} \quad \forall t, d, n \in P : n \geq 3 \tag{18}$$

$$z_{tdpp} \leq 1 + \sum_{e \in E_t} \left(x_{edp+1} - x_{edp} \right) \quad \forall t, d, p \in P \tag{19}$$

$$z_{tdmn} \leq 1 - \sum_{e \in E_t} x_{edn} \quad \forall t, d, (m, n) \in U \tag{20}$$

$$z_{tdmn} \leq \sum_{e \in E_t} x_{edn} \quad \forall t, d, (m, n) \in U \tag{21}$$

$$x_{edp}, b_{edp} \in \{0,1\}, g_{edp}, G_e \geq 0 \quad \forall e, d, p \tag{22}$$

$$y_{td} \geq 0, z_{tdmn} \in \{0,1\} \quad \forall t, d, (m, n) \in Q \tag{23}$$

A função objetivo (1) é compostas de três partes, (2), (3) e (4) 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 (5) garante que a carga de trabalho de cada evento seja agendada. O conjunto de restrições (6) proporciona o limite diário de aulas para cada evento. O conjunto de restrição (7) determina se as aulas serão agendadas em períodos disponíveis. O conjunto de restrições (8) e (10) garantem que professores e classes estejam agendados somente para uma única aula por vez. O Conjunto de restrições (8) e (9) define os dias em que cada professor trabalha. O conjunto de restrições (11) e (12) garantem que as aulas de um evento sejam agendadas sequencialmente de acordo com a restrição H6. O conjunto de restrições (13) e (14) força aulas geminadas quando a variável g_{edp} for igual a um. O conjunto de restrição (15) determina G_e, variável que controla quantas aulas geminadas faltam para alcançar MG_e. A variável G_e está relacionada a (4), sendo que o lado direito da inequação tende a aumentar, garantindo o atendimento de aulas geminadas. O conjunto de restrições (16) é um corte que define o número mínimo de dias trabalhados para cada professor.

Os conjuntos de restrições de (17) a (21) foram propostos por [11] 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 $(m, n) \in U$, 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 (17) e (18) 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 $(m, n) \in Q \backslash U$. 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 (19). O conjunto de restrições (20) 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 (21) garante que um arco de período de ociosidade (m, n) seja ativado apenas se o professor possua aula no período (n).

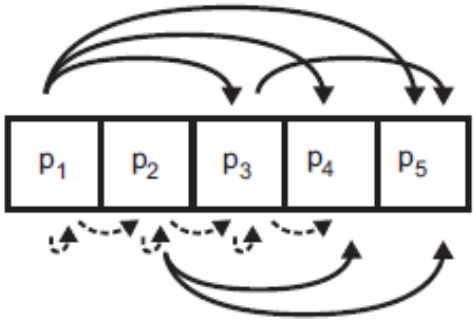

Fig. 1. Grafo dos periodos de ociosidade. Fonte: [11].

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

IV. HEURÍSTICA DE *FIX-AND-OPTIMIZE*

O CTTPCR é um problema muito complexo e que possui um número muito grande de variáveis e restrições, fazendo com que muitas vezes seja impossível resolve-los utilizando *solvers* MIP. Portanto é comum utilizar

heurísticas para solucionar estes problemas. Neste trabalho utilizaremos a heurística de *fix-and-optimize*.

O *fix-and-optimize* decompõe o problema em subproblemas menores fixando o valor de parte das variáveis, permitindo que *solvers* resolvam estes subproblemas rapidamente. Este processo se repete sucessivamente, fixando novas variáveis a cada iteração. A escolha da quantidade e de quais variáveis são fixadas variam em cada subproblema, segundo a decomposição escolhida. Neste problema, x_{edp} são as variáveis a serem fixadas, pois todas as outras são dependentes desta.

Escolhemos fazer as decomposições de classes 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. [11].

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 $\mathbb{N}$ 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), ..., (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. [11].

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

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

Fig. 2. Pseudo codigo da heuristica *fix-and-optimize.* Adaptado de [11].

Algoritmo Decompor(τ, k, s)

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

Fig. 3. Função de decomposição. Adaptado de [11]

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** $count$

Fig. 4. Função que calcula o número de subproblemas de uma vizinhança.

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 teve o tempo de resolução limitado em 30 segundos. 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. Fig. 2. 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.

V. Resultados Computacionais

Para avaliar o desempenho dos *solvers* GUROBI 7.0.2 e CPLEX 12.6.2 para a solução do CTTPCR a partir da heurística de *fix-and-optimize* implementamos os algoritmos 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, 6 e 3 respectivamente.

As instancias usadas para testes foram retiradas de [7], são referentes as escolas brasileiras e estão adaptadas para a aplicação do CTTPCR. A Tabela II 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. E por último 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. [11].

TABELA II. CARACTERISTICAS DAS INSTANCIAS ESTUDADAS

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

Fonte: Adaptado de [11].

Várias estratégias de decomposição foram consideradas para avaliar o método proposto. Utilizamos duas das estratégias que tiveram os melhores resultados das dez sugeridas por [11], definidas a seguir:

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

Para a comparação do desempenho dos dois *solvers* citados foram utilizados todas as cinco instancias através das duas estratégias acima citadas. Comparamos os melhores valores da função objetivo em que os solvers encontraram dentro do tempo limite de cada instancia.

TABELA III. RESULTADOS COMPUTACIONAIS

Instancia	F7		F8	
	CPLEX	GUROBI	CPLEX	GUROBI
A	203	200	203	200
D	683	651	666	651
E	830	777	817	777
F	815	796	789	796
G	1131	1091	1114	1091

É possível ver pela TABELA III que o solver GUROBI obteve resultados superiores para quatro das cinco instancias estudadas, sendo que o CPLEX obteve melhor resultado somente para a instancia F usando a estratégia F8.

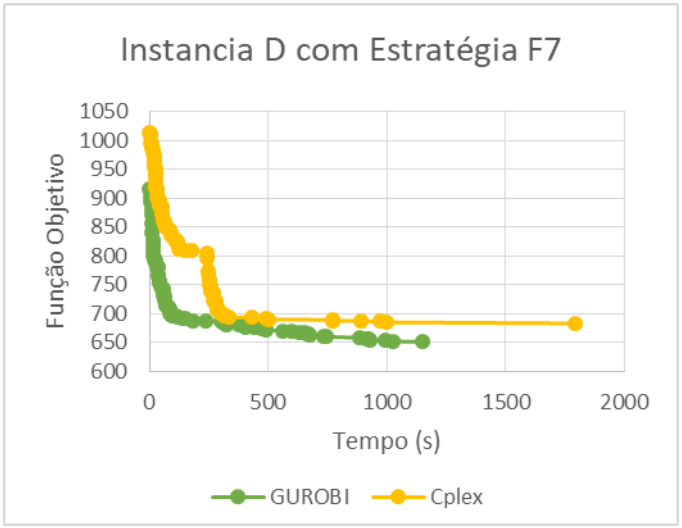

Fig. 5. Grafico de desempenho para a Instancia D utilizando a estrategia F7.

Usando a estratégia F7 todas as instancias possuem o mesmo padrão apresentado na Fig. 5. Podemos perceber que para curva do CPLEX o decaimento da função objetivo começa a estabilizar em aproximadamente 250 segundos, depois volta a cair bruscamente. Isto ocorre na mudança da cardinalidade k, que acontece quando todas as combinações dois a dois são otimizadas, e começa a realizar as combinações três a três. Após isto ele volta a estabilizar, seguindo quase horizontalmente até atingir o tempo limite de processamento. O mesmo não acontece com o GUROBI que tem um decaimento continuo durante todo o processo, se mantendo abaixo da curva do CPLEX por todo o tempo de processamento.

Fig. 6. Grafico de desempenho para a Instancia D utilizando a estrategia F8.

A Fig. 6 apresenta a instancia D com estratégia F8. Assim como mostrado na Fig. 5, todas as instâncias seguem o mesmo padrão para as curvas criadas pelos dois *solvers*, iniciando com um grande decaimento logo de início, sendo que, em alguns casos, o CPLEX chega a decair mais rápido do que o GUROBI. Porém, a curva do CPLEX para de decair primeiro, enquanto o GUROBI continua decaindo lentamente, exceto para a instancia F, em que o oposto ocorre, e o GUROBI se estabiliza um pouco antes.

VI. CONCLUSÃO

Neste trabalho fizemos uma comparação entre dois *solvers* de programação inteira mista para resolver o CTTPCR a partir da heurística de *fix-and-optimize*, os *solvers* GUROBI 7.9.2 e CPLEX 12.6.2. Para cada instancia foram fixados limites de tempo de processamento para avaliar o desempenho obtido pelos respectivos *solvers* em relação a função objetivo. Por fim o GUROBI obteve melhores resultados para quatro das cinco instancias apresentadas, se consolidando como melhor opção para a solução deste tipo de problema.

Como proposta para trabalhos futuros indicamos que sejam feitas novas abordagens aplicadas ao *fix-and-optimize*, buscando melhorar o tempo de processamento da heurística.

AGRADECIMENTOS

Agradecemos a CAPES pelo suporte financeiro especialmente ao grupo de pesquisa GTAO por todo apoio, suporte e amizade.

REFERÊNCIAS

[1] D. de Werra, “Construction of school timetables by flow methods”. vol. 9, pp. 12-22, 1971.

[2] S. Even, A. Itai, A. Shamir, “On the Complexity of Time Table and Multi-commodity Flow Problems”. In: Proceedings of the 16th annual symposium on foundations of computer science. SFCS ’75, Washington, DC, USA: IEEE Computer Society, pp. 184-193, 1975.

[3] A. Drexl, F. Salewski, “Distribution Requirements and Compactness Constrains in School Timetabling”. European Journal of Operational Research, vol. 102, pp. 193-214, June 1996.

[4] A. Scharf, “A Survey of Automated Timetabling”. Artificial Intelligence Review, vol. 13, pp. 87-127, 1999.

[5] M. Souza, N. Maculan, “Melhorando quadros de Horario de Escolas Atraves de Caminhos Mínimos”. Tendencias em Matematica Aplicada e Computacional, vol. 1, pp. 515-524, 2000.

[6] T. Birbas, S. Daskalaki, E. Housos, “School Timetabling and Teacher Schedules”. Journal of Scheduling, vol. 12, pp. 177-197, August 2008.

[7] ITC2011, “Third International Timetabling Competition”. 2011. URL (http://www.utwente.nl/ctit/hstt/itc2011).

[8] G. Post, J. Kingston, S. Ahmadi, S. Daskalaki, C. Gogos et al, “XHSTT: an XML archive for High School Timetabling Problems in Different Countries”. Annals of Operation Research, pp. 295-301, November 2011.

[9] H. G. Santos, E. Uchoa, L. S. Ochi, N. Maculan, “Strong Bounds with Cut and Column Generation for Class-teacher Timetabling”. Annals of Operation Research, vol. 194, pp. 399-412, 2012.

[10] S. Kristiansen, M. Sorensen, T. R. Stidsen, “Integer Programing for the Generalized High School Timetabling Problem”. Journal of Scheduling, vol. 18, pp. 377-392, 2014.

[11] A. P. Dorneles, O. C. B. Araujo, L. S. Buriol, “A fix-and-optimize Heuristic for the High School Timetabling Problem”. Computer & Operations Research, vol. 52, pp. 29-38, July 2014.

[12] G. H. G. Fonseca, H. G. Santos, E. G. Carrano, “Late acceptance Hill-climbing for High Scholl Timetabling”. Journal of Scheduling, November 2015.

[13] A. P. Dorneles, O. C. B. Araujo, L. S. Buriol, “A Column Generation Approach to High School Timetabling Modeled as a Multicommodity Flow Problem”. European Journal of Operational Research, vol. 256, pp. 685-695, July 2016.

[14] E. Demirovic, N. Musliu, “MaxSAT-based large neighborhood search for high school timetabling”. Computer & Operation Research, vol. 78, pp. 172-180, August 2016.