LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

Um Algoritmo Heurístico aplicado ao Problema de Programação de Horários Universitários do DECSI/UFOP

Gabriel A. Mafia
Programa de Pós-Graduação em Engenharia de Produção
Universidade Federal de Ouro Preto
Rua 37, 115 - Vera Cruz, João Monlevade - MG, 35931-008
gabrielmafia@gmail.com

George H.G. Fonseca
Departamento de Computação e Sistemas
Universidade Federal de Ouro Preto
Rua 37, 115 - Vera Cruz, João Monlevade - MG, 35931-008
george@ufop.edu.br

RESUMO

O problema de programação de horários educacionais consiste em alocar recursos e horários a eventos de modo a atender um conjunto de restrições. Esse trabalho aborda o problema de programação de horários do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto (DECSI/UFOP). Esse problema tem algumas características peculiares que não foram ainda abordadas em conjunto na literatura. Por isso, o presente trabalho tem como principal objetivo propor uma abordagem heurística para o problema. A abordagem se utiliza de um algoritmo construtivo de duas etapas baseado, respectivamente em backtracking e força-bruta e da metaheurística Multi-Start numa estrutura multi-vizinhança para o refinamento das soluções iniciais. Os resultados obtidos foram satisfatórios e o sistema foi adotado pelo departamento para auxiliar na resolução do problema, automatizando por completo o processo de construção dos horários.

PALAVRAS CHAVE. Programação de Horários Universitários, Metaheurísticas, Multi-Start.

AdP&ED – PO na Administração Pública e Educação
MH – Metaheurísticas

ABSTRACT

The educational timetabling problem consists of assigning resources and times to events in order to attend a set of constraints. This work addresses the timetabling problem of the Department of Computing and Systems at the Federal University of Ouro Preto (DECSI/UFOP). This problem has some peculiar characteristics that have not yet been addressed together in the literature. Therefore, the present work aims to propose a heuristic approach to the problem. The approach uses a two-step constructive algorithm based, respectively, on backtracking and brute force, and on the Multi-Start metaheuristic in a multi-neighborhood structure for the refinement of the initial solutions. The results obtained were satisfactory and the system was adopted by the department to assist in solving the problem, completely automating the process of building timetables.

KEYWORDS. University Timetabling. Metaheuristics. Multi-Start.

AdP&ED – OR in Public Administration and Education
MH – Metaheuristics

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introdução

O problema de programação de horários educacionais consiste essencialmente em alocar recursos e horários a eventos de modo a respeitar um determinado conjunto de restrições e maximizar o atendimento de um conjunto de preferências Fonseca et al. [2017]. Os recursos a serem alocados geralmente consistem de professores, classes, estudantes e salas. As restrições mais comuns desse problema são: não permitir conflitos de horários para os professores e estudantes e não alocar eventos em horários nos quais um recurso está indisponível.

Os primeiros estudos para solucionar o problema com o auxilio de computadores ocorreram na década de 1960, com o trabalho precursor de Csima e Gotlieb [1961]. Antes disso, a solução era obtida de forma manual, o que poderia demorar semanas, dependendo da complexidade do sistema, além de não haver garantia de obtenção de um resultado satisfatório, sem conflitos ou indisponibilidade de recursos [Souza, 2000]. Este problema é classificado como $\mathcal{NP}$-difícil em praticamente todas suas variantes, pois pode ser reduzido ao problema da coloração de grafos [Neufeld e Tartar, 1974], o que justifica o uso de métodos heurísticos para sua resolução.

Devido à variedade de modelos de problemas de programação de horários disponíveis na literatura, foram propostas três competições internacionais dedicadas ao assunto. A mais recente, a ITC2011/12 adotou um formato flexível para especificação de uma diversos de problemas de programação de horários educacionais, reunindo um vasto conjunto de instâncias com variadas características [Post et al., 2011]. O resolvedor melhor classificado utilizou um algoritmo de emparelhamento de custo mínimo em grafos para geração de soluções iniciais e a uma metaheurística híbrida de *Simulated Annealing* e *Iterated Local Search* numa estrutura multi-vizinhança para o refinamento [Fonseca et al., 2016]. Boa parte da literatura recente sobre programação de horários se concentra em propor algoritmos mais eficientes para os problemas *benchmark* das competições, como Fonseca et al. [2017] e Bagger et al. [2018] e em propor algoritmos que resolvem problemas específicos de uma certa universidade ou escola, como Skoullis et al. [2017] e Saviniec et al. [2020].

Apesar do formato proposto pela ITC2011 ser genérico e flexível, não foi possível representar o problema de programação de horários do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto (DECSI/UFOP) no mesmo. Em especial, não foi possível especificar a restrição de padrões de alocação indesejáveis para os professores (veja a Seção 2). Dessa forma, o presente trabalho tem como objetivo principal propor uma abordagem de solução para o problema de programação de horários do DECSI/UFOP. A abordagem proposta é baseada em algoritmos de *backtracking* e força-bruta para gerar soluções iniciais e na meteheurística *Multi-Start* [Martí, 2003] com uma estrutura multi-vizinhança para refinamento dessas soluções.

O restante do trabalho está organizado da seguinte forma. A Seção 2 apresenta o problema de programação de horários do DECSI/UFOP. A Seção 3 apresenta a abordagem de solução proposta ao problema. A Seção 4 apresenta as características das instâncias abordadas, os resultados obtidos e uma discussão acerca desses resultados. Por fim, a Seção 5 apresenta as considerações finais, contribuições do presente trabalho à literatura e direções de trabalhos futuros no tema.

2. Problema de Programação de Horários Universitários

Esse trabalho aborda o problema específico de programação de horários do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto (DECSI/UFOP). O DECSI está inserido no Instituto de Ciências Exatas e Aplicadas, campus João Monlevade e oferta 71 disciplinas para os cursos do instituto: 32 para o curso de Sistemas de Informação; 31 para Engenharia de Computação; 5 para Engenharia Elétrica; e 3 para Engenharia de Produção. As aulas do instituto são distribuídas de segunda a sexta-feira (5 dias), com quatro *slots* de horário em cada dia: (i)

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

13:30h às 15:10h; (ii) 15:25h às 17:05h; (iii) 18:50h às 20:30h; e (iv) 20:45h às 22:25h. Os horários (i) e (ii) são usados pelos cursos vespertinos e os horários (iii) e (iv) pelos cursos noturnos.

Nesse problema, cada evento consiste de uma disciplina a ser cursada pelos estudantes. Um evento pertence a uma determinada classe (período) de um curso e tem um determinado professor responsável. Pode haver mais de uma opção de professor para um determinado evento. Nesse caso, o resolvedor deve decidir qual professor assumirá o evento dentro de suas disponibilidades e preferências.

Uma classe consiste de um conjunto de eventos que certos alunos devem atender. Por exemplo, a classe do 6^o período de Sistemas de Informação é composta por 5 eventos (disciplinas), as quais os alunos desse período devem cursar concomitantemente. Para permitir que os alunos se matriculem em todas as disciplinas do período em que estão cursando, não é permitido que eventos pertencentes a uma mesma classe ocorram no mesmo horário.

Os professores podem definir ainda algumas preferências, é possível: (i) definir um conjunto de horários indesejáveis, por exemplo os horários da sexta-feira; (ii) definir o número mínimo e máximo de dias nos quais preferem lecionar; e (iii) definir um conjunto de padrões indesejáveis de alocação de horários ocupados/livres, por exemplo um padrão com alocações na segunda, quarta e sexta feiras pode ser indesejável para professores que moram fora da cidade.

A principal decisão a ser tomada pelo resolvedor é em quais horários irão ocorrer cada evento de modo a não haver conflitos nas tabelas de horários das classes e dos professores e qual professor será alocado para cada evento. As subseções a seguir objetivam a definir mais formalmente o problema de programação de horários abordado nesse trabalho.

2.1. Dados de entrada

O problema de programação de horários abordado nesse trabalho é composto pelos seguintes conjuntos e parâmetros:

$\mathcal{E}$	Eventos
$\mathcal{P}$	Professores
$\mathcal{C}$	Classes
$\mathcal{D}$	Dias
$\mathcal{T}$	*Slots* de horário
$\mathcal{CE}_c$	Eventos que compõem a classe c
$\mathcal{CPE}_e$	Professores compatíveis com o evento e
$\mathcal{LE}$	Pares de eventos (e_1, e_2) que devem ser ligados
$\mathcal{UC}_c$	Horários indisponíveis para a classe c
$\mathcal{UP}_p$	Horários indisponíveis para o professor p
$\mathcal{NP}_p$	Horários indesejados pelo professor p
$\mathcal{NPP}_p$	Padrões indesejáveis para o professor p

d_e	Duração do evento e
$\overline{dur}_p$	Duração máxima de eventos que pode ser alocada ao professor p
$\underline{dias}_p$	Número mínimo de dias com aulas desejado pelo professor p
$\overline{dias}_p$	Número máximo de dias com aulas desejado pelo professor p

2.2. Restrições

O problema possui as seguintes restrições fortes (i.e. cujo atendimento é obrigatório para uma solução ser considerada factível):

1. Para cada evento $e \in \mathcal{E}$, deve-se alocar um número de horários igual à sua duração d_e;
2. Para cada evento $e \in \mathcal{E}$, deve-se alocar um professor dentre seu conjunto de professores compatíveis $p \in \mathcal{CPE}_e$;
3. Para cada classe $c \in \mathcal{C}$, não é permitido alocar eventos nos horários em que a classe não está disponível $(d, t) \in \mathcal{UC}_c$;
4. Para cada professor $p \in \mathcal{P}$, não é permitido alocar eventos nos horários em ele não está disponível $(d, t) \in \mathcal{UP}_p$;
5. Para cada classe $c \in \mathcal{C}$, não pode haver conflito de horário entre os eventos que a compõem;
6. Para cada professor $p \in \mathcal{P}$, não pode haver conflito de horário entre os eventos aos quais ele foi alocado;
7. Para cada professor, a carga horária de eventos a ele alocada não pode ultrapassar seu limite dur_p;
8. Para cada evento $e \in \mathcal{E}$, deve haver um espaçamento mínimo de um dia entre seus encontros;
9. Para cada par de eventos ligados $(e_1, e_2) \in \mathcal{LE}$, seus encontros devem ocorrer no mesmo horário.

O problema considerado possui ainda as seguintes restrições fracas (i.e. cujo atendimento é desejável porém não obrigatório para a factibilidade de uma solução):

1. Uma penalidade s_p^D é computada para a quantia de dias em que um professor $p \in \mathcal{P}$ é alocado a mais dias que o máximo de dias com aulas desejado $\overline{dias}_p$ ou menos que o mínimo de dias com aulas desejado $\underline{dias}_p$;
2. Uma penalidade s_p^{UT} é computada para cada professor $p \in \mathcal{P}$ que tiver que atender algum evento em um de seus horários indesejáveis $(d, t) \in \mathcal{NP}_p$;
3. Uma penalidade s_p^{UP} é computada para professor $p \in \mathcal{P}$ que tem alocações seguindo um padrão indesejado de dias ocupados/livres.

2.3. Função objetivo

O objetivo do problema abordado consiste em minimizar a ocorrência de penalidades por violação às restrições fracas. Mais especificamente, a função objetivo consiste de uma somatória onde, para cada professor $p \in \mathcal{P}$, é calculada e somada sua penalidade individual, com base nas penalidades s_p^D, s_p^{UT} e s_p^{UP}. A cada penalidade por não-atendimento de uma restrição fraca é aplicada um multiplicador, que guia sua prioridade em uma solução ideal. Os multiplicadores $w_D = 1$, $w_{UT} = 1$ e $w_{UP} = 1$ foram considerados para as restrições fracas 1, 2 e 3 respectivamente. Por fim, cada professor tem ainda um multiplicador de prioridade pr_p. Nesse trabalho foi assumido $pr_p = 1,0$ para professores efetivos e $pr_p = 0,9$ para professores substitutos. A equação 1 apresenta formalmente a função objetivo do problema:

$$min \sum_{p \in P} (w^D \times s_p^D + w^{NP} \times s_p^{NP} + \sum_{up \in UPP_p} (w^{UP} \times s_{p,up}^{UP}) + \sum_{(s,c) \in CPE_p} (w^{EP} \times costEP_{s,c,p} \times z_{s,c,p})) \times priority_p \quad (1)$$

2.4. Solução

A solução para o problema consiste do conjunto de tabelas de horários dos eventos de cada classe bem como a definição do professor alocado a cada evento. A Figura 1 apresenta um exemplo de tabela de solução, em específico para o 6^o período do curso Sistemas de Informação, composto por cinco eventos, todos de duração 2. Note que os horários da noite são indisponíveis e que foi respeitado o espaçamento mínimo de um dia entre encontros do mesmo evento.

Tabela 1: Exemplo de solução para a classe 6^o período do curso Sistemas de Informação.

6^o período Sistemas de Informação					
Horário	Seg	Ter	Qua	Qui	Sex
13:30h	CSI450 Tiago	CSI477 Fernando	CSI478 Diego	CSI450 Tiago	CSI433 Theo
15:10h	CSI417 Luiz	CSI433 Theo	CSI419 Luiz	CSI477 Fernando	CSI478 Diego
18:50h	-	-	-	-	-
20:45h	-	-	-	-	-

3. Abordagem Proposta

Esse trabalho propõe uma abordagem heurística para solucionar o problema. A abordagem consiste de algoritmos construtivos baseados em *backtracking* e em força-bruta para gerar soluções iniciais e da metaheurística Multi-Start para o refinamento dessas soluções numa estrutura de vizinhança composta por três movimentos. Cada uma das subseções a seguir detalha um desses aspectos da abordagem proposta.

3.1. Algoritmo Construtivo

O algoritmo construtivo executa em duas etapas: primeiro é realizada a alocação dos professores aos eventos e em seguida a alocação dos horários aos eventos. É importante que o algoritmo construtivo seja capaz de gerar soluções com considerável variabilidade para eficiente aplicação posterior da mateheurística Multi-Start, por isso a aleatoriedade foi introduzida em algumas etapas desse algoritmo.

3.1.1. Alocação dos Professores

O Algoritmo 1 apresenta o pseudocódigo do procedimento que aloca professores aos eventos. Esse algoritmo recebe como entrada: (i) uma instância do problema $\mathbb{P}$, contendo todos os dados de entrada definidos na seção 2.1; (ii) a alocação evento-professor corrente $\mathcal{Y}$; e (iii) a carga horária corrente alocada aos professores $\mathcal{CH}$. O algoritmo retorna os vetores $\mathcal{Y}$ e $\mathcal{CH}$ de acordo com as alocações realizadas por ele até o momento. Na chamada inicial, o vetor $\mathcal{Y}$ é composto por valores nulos e o vetor $\mathcal{CH}$ é composto por zeros.

Esse algoritmo se baseia na estratégia de *backtracking*, que consiste em construir progressivamente um candidato a solução e abandonar (*backtrack*) esse candidato assim que identificar que ele não pode gerar uma solução completa válida [Knuth, 2014]. Nesse caso, a construção continua do próximo candidato a solução não descartado. Isso é feito através de chamadas recursivas, que constroem a solução progressivamente.

O Algoritmo 1 aloca um professor para um evento por iteração (linha 2). Para realizar a alocação ele ordena os professores compatíveis com o evento ($\mathcal{CPE}_e$) de acordo com sua disponibilidade. Essa ordenação se dá de forma crescente sobre o número (cardinalidade do conjunto) de

horários indisponíveis do professor p ($\mathcal{UT}_p$); assim, professores com maior disponibilidade são preferíveis (linha 3). Se não há professor alocado ao evento e (linha 4), para cada professor candidato $p \in \mathcal{CPE}_e$ é verificado se é possível adicionar o evento à sua carga horária sem extrapolar seu limite dur_p (linha 6). Caso possível, a alocação é feita (linha 7) e a carga horária alocada incrementada (linha 8). Se e for o último evento a ser alocado (linha 9), então a alocação está completa e temos uma solução válida. Nesse caso a flag $alocacaoCompleta$ recebe **true** e o algoritmo retorna os vetores $\mathcal{Y}$ e $\mathcal{CH}$, encerrando a última chamada recursiva. Senão, é feita uma chamada recursiva à função ALOCA_PROFESSORES, passando os vetores $\mathcal{CH}$ e $\mathcal{Y}$ atualizados (linha 13). Se a alocação estiver completa, o algoritmo retorna os vetores $\mathcal{Y}$ e $\mathcal{CH}$, encerrando a chamada recursiva (linha 15), o que evita a execução das linhas 16 e 17, responsáveis por desfazer uma alocação do professor p ao evento e (*backtrack*). Caso uma alocação seja desfeita, a próxima iteração do laço da linha 5 será executada, testando o próximo candidato a professor do evento.

Algoritmo 1: ALOCA_PROFESSORES: Algoritmo construtivo baseado em *backtracking* para alocação de professores aos eventos.

Entrada: (i) Instância do problema $\mathbb{P}$ ($\mathcal{E}$, $\mathcal{P}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{H}$, $\mathcal{T}$, $\mathcal{CE}_c$, $\mathcal{CPE}_e$, $\mathcal{LE}$, $\mathcal{UC}_c$, $\mathcal{UP}_p$, $\mathcal{NP}_p$, $\mathcal{NPP}_p$, d_e, $\overline{dur}_p$, $\underline{dias}_p$, $\overline{dias}_p$),
(ii) Alocação evento-professor $\mathcal{Y}$,
(iii) Carga horária alocada aos professores $\mathcal{CH}$.

Saída: Vetores $\mathcal{Y}$ e $\mathcal{CH}$ atualizados com a alocação de professores corrente.

```
1  alocacaoCompleta ← false;
2  para cada e ∈ E faça
3      CPE_e ← Ordenar crescentemente os professores p ∈ CPE_e por |UP_p|;
4      se Y_e = null então
5          para cada p ∈ CPE_e faça
6              se CH_p + d_e ≤ dur_p então
7                  Y_e ← p;                          ▷ Professor p é alocado ao evento e
8                  CH_p = CH_p + d_e;      ▷ Incrementa CH alocada do professor p em d_e
9                  se e é o último evento de E então
10                     alocacaoCompleta ← true;          ▷ Solução válida encontrada
11                     retorna (Y, CH);              ▷ Encerra última chamada recursiva
12                 senão
13                     ALOCA_PROFESSORES(P, CH, Y);          ▷ Chamada recursiva
14                     se alocacaoCompleta = true então
15                         retorna (Y, CH);              ▷ Encerra chamada recursiva
16                     Y_e ← ∅;      ▷ Alocação do professor p para o evento e é desfeita
17                     CH_p = CH_p − d_e;      ▷ Decrementa CH alocada do prof. p em d_e
```

3.1.2. Alocação de Horários

O Algoritmo 2 apresenta o pseudocódigo do procedimento que aloca horários aos eventos. Esse algoritmo recebe como entrada: (i) uma instância do problema $\mathbb{P}$, contendo todos os dados de entrada definidos na seção 2.1; (ii) a alocação evento-professor $\mathcal{Y}$ obtida pelo Algoritmo 1; (iii) uma matriz de alocações de horários dos eventos $\mathcal{X}^{\mathcal{E}}$; e (iv) uma matriz de alocações de horários

dos professores $\mathcal{X}^{\mathcal{P}}$. As matrizes $\mathcal{X}^{\mathcal{E}}$ e $\mathcal{X}^{\mathcal{P}}$ são binárias e assumem, respectivamente:

$$\mathcal{X}^{\mathcal{E}} = \begin{cases} 1 & \text{se o evento } e \in \mathcal{E} \text{ tem encontro no horário } (d,t) \in \mathcal{D} \times \mathcal{T}. \\ 0 & \text{caso contrário.} \end{cases}$$

$$\mathcal{X}^{\mathcal{P}} = \begin{cases} 1 & \text{se o professor } p \in \mathcal{P} \text{ leciona no horário } (d,t) \in \mathcal{D} \times \mathcal{T}. \\ 0 & \text{caso contrário.} \end{cases}$$

As matrizes $\mathcal{X}^{\mathcal{E}}$ e $\mathcal{X}^{\mathcal{P}}$ são, inicialmente preenchidas por zeros. O algoritmo atualiza e retorna as matrizes $\mathcal{X}^{\mathcal{E}}$ e $\mathcal{X}^{\mathcal{P}}$ de acordo com as alocações realizadas. Esse algoritmo se baseia na estratégia de força bruta, que consiste em computar sistematicamente todas as combinações possíveis de alocação (horários) para encontrar uma solução. Para cada classe $c \in \mathcal{C}$, o procedimento encerra assim que uma alocação viável é encontrada, poupando esforço computacional. Ao final do processamento de todas as classes, uma solução completa factível (caso exista) é encontrada.

O Algoritmo 2 aloca os horários dos eventos de uma classe $c \in \mathcal{C}$ por iteração (linha 2). Uma etapa de pré-processamento ordena as classes de modo que as classes que têm menor disponibilidade de horários (i.e. têm mais restrições no conjunto $\mathcal{UC}$) sejam resolvidas primeiro (linha 1). As linhas 3 a 19 são responsáveis por enumerar as combinações de horários em que cada evento $e \in \mathcal{CE}_c$ pode ocorrer. O professor associado ao evento e é recuperado na variável p (linha 5). Na sequência, um conjunto, inicialmente vazio, das possíveis combinações de horários para o evento e é criado (linha 6). O conjunto dos horários indisponíveis $\mathcal{ITE}_e$ para o evento e é criado pela união dos conjuntos de horários indisponíveis da classe à qual o evento pertence $\mathcal{UC}_c$, do professor alocado ao evento $\mathcal{UP}_p$ e dos horários já alocados na tabela do professor p ((d,t) caso $\mathcal{X}_{pdt}$ = 1). Caso o evento seja ligado a outro evento e_2, as indisponibilidades de e_2 são consideradas em conjunto (linha 7). Para cada horário (d,t) (linhas 8 e 9), se o horário não está indisponível ($(d,t) \notin \mathcal{ITE}_e$) (linha 10) há dois casos: (i) o evento tem duração 1 (linha 11) e o horário (d,t) já constitui uma combinação para o evento e e é adicionada a $\mathcal{Q}_e$ (linha 12); ou (ii) o evento tem duração 2 (linha 13), nesse caso uma combinação requer um par de horários $((d,t),(d_2,t_2))$. d_2 é inicializado de modo a garantir um dia de folga após d conforme a restrição forte 8 (linha 14). O laço da linha 16 itera sobre os possíveis horários e, caso (d_2,t_2) não seja indisponível para o evento e (linha 17) a combinação $((d,t),(d_2,t_2))$ é adicionada a $\mathcal{Q}_e$. Não há eventos de duração maior que 2 no problema considerado.

As linhas 22 a 38 avaliam cada combinação de possíveis horários de alocação dos eventos pertencentes à classe c (23 a 32) e realizam a primeira alocação viável encontrada (33 a 38). Mais especificamente, a flag inf indica se a combinação $\mathcal{Q}'_{ce}$ é infactível (linha 24) e $\mathcal{OT}_c$ armazena os horários ocupados da classe c (linha 25). Para cada alocação de horário (d,t) do evento e na combinação $\mathcal{Q}'_{ce}$ (linhas 26 e 27), se o horário (d,t) ainda não está ocupado (linha 28), ele é adicionado ao conjunto dos horários ocupados da classe c, $\mathcal{OT}_c$ (linha 29); caso contrário, a combinação é infactível (usa o mesmo horário (d,t) mais de uma vez) (linha 31) e a execução do laço é interrompida de modo a continuar pela próxima combinação possível (linha 32). Caso a combinação $\mathcal{Q}'_{ce}$ tenha sido validada como factível (linha 33), procede-se à alocação dos horários conforme $\mathcal{Q}'_{ce}$. Para cada alocação de horário (d,t) do evento e na combinação $\mathcal{Q}'_{ce}$, as variáveis $\mathcal{X}^{\mathcal{E}}_{edt}$ e $\mathcal{X}^{\mathcal{P}}_{pdt}$ são ativadas, indicando a realização da alocação respectivamente para o evento e e para o professor p (linhas 37 e 38). Por fim, as matrizes $\mathcal{X}^{\mathcal{E}}$ e $\mathcal{X}^{\mathcal{E}}$indicando as alocações realizadas são retornadas (linha 39).

Algoritmo 2: ALOCA_HORARIOS: Algoritmo construtivo de força-bruta para alocação de horários aos eventos.

Entrada: (i) Instância do problema $\mathbb{P}$ ($\mathcal{E}$, $\mathcal{P}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{H}$, $\mathcal{T}$, $\mathcal{CE}_c$, $\mathcal{CPE}_e$, $\mathcal{LE}$, $\mathcal{UC}_c$, $\mathcal{UP}_p$, $\mathcal{NP}_p$, $\mathcal{NPP}_p$, d_e, $\overline{dur}_p$, $\underline{dias}_p$, $\overline{dias}_p$),
(ii) Vetor de alocações evento-professor $\mathcal{Y}$,
(iii) Matriz de alocações de horário dos eventos $\mathcal{X}^{\mathcal{E}}$ (preenchida por zeros),
(iv) Matriz de alocações de horário dos professores $\mathcal{X}^{\mathcal{P}}$ (preenchida por zeros).

Saída: Matrizes $\mathcal{X}^{\mathcal{E}}$ e $\mathcal{X}^{\mathcal{P}}$ atualizadas com as alocações de horários.

$\mathcal{C} \leftarrow$ Ordenar crescentemente as classes $c \in \mathcal{C}$ por $|\mathcal{UC}_c|$;
para cada $c \in \mathcal{C}$ **faça**
  $\mathcal{Q}_{ce} \leftarrow \emptyset$; ▷ Combinações de horário para cada evento $e \in \mathcal{CE}_c$
  para cada $e \in \mathcal{CE}_c$ **faça**
    $p \leftarrow \mathcal{Y}_e$; ▷ p recebe o professor alocado ao evento e
    $\mathcal{Q}_e \leftarrow \emptyset$; ▷ Combinações de horário do evento e
    $\mathcal{ITE}_e \leftarrow$ Horários indisponíveis para o evento e de acordo com $\mathcal{UC}_c$, $\mathcal{UP}_p$, $\mathcal{X}^{\mathcal{P}}_p$ e $\mathcal{LE}$
    para cada $d \in \mathcal{D}$ **faça**
      para cada $t \in \mathcal{T}$ **faça**
        se $(d,t) \notin \mathcal{ITE}_e$ **então**
          se $d_e = 1$ **então**
            $\mathcal{Q}_e \leftarrow \mathcal{Q}_e \cup \{(d,t)\}$;
          senão se $d_e = 2$ **então**
            $d_2 \leftarrow d + 2$; ▷ Garante um dia de folga entre encontros do evento
            enquanto $d_2 \leq |\mathcal{D}|$ **faça**
              para $t_2 \in \mathcal{T}$ **faça**
                se $(d_2, t_2) \notin \mathcal{ITE}_e$ **então**
                  $\mathcal{Q}_e \leftarrow \mathcal{Q}_e \cup \{((d,t),(d_2,t_2))\}$;
              $d_2 \leftarrow d_2 + 1$;
    Embaralhe o conjunto $\mathcal{Q}_e$;
    $\mathcal{Q}_{ce} \leftarrow \mathcal{Q}_{ce} \cup \{\mathcal{Q}_e\}$;
  para cada Combinação $\mathcal{Q}'_{ce}$ de elementos em $\mathcal{Q}_{ce}$ **faça**
    ▷ Checa se a combinação $\mathcal{Q}'_{ce}$ constitui uma tabela livre de conflitos (factível) para c
    $inf \leftarrow$ **false**; ▷ Flag que indica se a combinação $\mathcal{Q}'_{ce}$ é infactível
    $\mathcal{OT}_c \leftarrow \emptyset$; ▷ Horários ocupados da classe c
    para cada $e \in \mathcal{CE}_c$ **faça**
      para cada $(d,t) \in \mathcal{Q}'_{ce}$ **faça**
        se $(d,t) \notin \mathcal{OT}_c$ **então**
          $\mathcal{OT}_c \leftarrow \mathcal{OT}_c \cup \{(d,t)\}$;
        senão
          $inf \leftarrow$ **true**;
          break;
    se $inf =$ **false então**
      ▷ Aloca os horários conforme a combinação $\mathcal{Q}'_{ce}$
      para cada $e \in \mathcal{CE}_c$ **faça**
        para cada $(d,t) \in \mathcal{Q}'_{ce}$ **faça**
          $\mathcal{X}^{\mathcal{E}}_{edt} \leftarrow 1$; ▷ Marca o horário (d,t) como alocado no evento e
          $\mathcal{X}^{\mathcal{P}}_{pdt} \leftarrow 1$; ▷ Marca o horário (d,t) como alocado no professor p

3.2. Estrutura de Vizinhança

Foi desenvolvida nesse trabalho uma estrutura multi-vizinhança composta por três movimentos distintos: (i) realoca eventos; (ii) troca professores; e (iii) realoca eventos ligados. Para gerar um novo vizinho a estrutura de vizinhança é escolhida de acordo com as seguintes probabilidades: 50% para a estrutura (i); 25% para a estrutura (ii); e outros 25% para a estrutura (iii). Esses valores foram ajustados empiricamente. As subseções a seguir descreveram cada uma dessas estruturas.

3.2.1. Realoca Eventos

Essa vizinhança sorteia uma classe $c \in \mathcal{C}$ e dois eventos $e_1 \in \mathcal{CE}_c$ e $e_2 \in \mathcal{CE}_c$ dentro dessa classe. Todas as alocações desses eventos são desfeitas (i.e. $\forall d \in \mathcal{D}, \forall t \in \mathcal{T}, \mathcal{X}^{\mathcal{E}}_{edt} \leftarrow 0$). Todas as combinações alternativas factíveis de realocação de horários para e_1 e e_2 são avaliadas e aquela que implicar no menor custo de função objetivo é retornada como vizinha. As matrizes de alocação de horários dos professores que lecionam os eventos e_1 e e_2 também são atualizadas de acordo.

3.2.2. Troca Professores

Essa vizinhança sorteia um evento $e_1 \in \mathcal{E}$ que tenha mais de uma opção de professor compatível ($|\mathcal{CPE}_{e_1}| > 1$) e um segundo evento $e_2 \in \mathcal{E}$ que tenha os mesmos professores compatíveis que o evento e_1, ou seja, $\mathcal{CPE}_{e_1} = \mathcal{CPE}_{e_2}$. Os professores p_1 de $\mathcal{Y}_{e_1}$ e p_2 de $\mathcal{Y}_{e_2}$ são trocados ($\mathcal{Y}_{e_1} \leftarrow p_2$ e $\mathcal{Y}_{e_2} \leftarrow p_1$). Caso a troca leve a uma solução factível (i.e. não cause conflito de horário nas tabelas dos professores p_1 e p_2) o vizinho é retornado.

3.2.3. Realoca Eventos Ligados

Essa vizinhança sorteia um par de eventos ligados $(e_1, e_2) \in \mathcal{LE}$ e um terceiro evento pertencente à mesma classe do evento e_1 ($e_3 \in \mathcal{CE}_{e_1}$). Um quarto evento e_4 que pertence à mesma classe do evento e_2 ($e_4 \in \mathcal{CE}_{e_2}$) que tenha ao menos uma alocação de horário em comum com e_3 também é selecionado. As alocações de horário dos eventos e_1, e_2, e_3 e e_4 são desfeitas (i.e. $\forall d \in \mathcal{D}, \forall t \in \mathcal{T}, \mathcal{X}^{\mathcal{E}}_{edt} \leftarrow 0$). Na sequência, todas as combinações alternativas factíveis de realocação de horários para e_1, e_2, e_3 e e_4 são avaliadas e aquela que implicar no menor custo de função objetivo é retornada como vizinha. As matrizes de alocação de horários dos professores que lecionam os eventos e_1, e_2, e_3 e e_4 também são atualizadas de acordo.

3.3. Multi-Start

O Algoritmo 3 apresenta o pseudocódigo da implementação desenvolvida da metaheurística Multi-Start nesse trabalho. O algoritmo recebe como entrada: (i) uma instância do problema $\mathbb{P}$ contendo os dados de entrada; (ii) um tempo limite, em segundos, para sua execução $time$; e (iii) o número máximo de iterações sem melhoria para execução da busca local $iMax$. A saída do algoritmo é a melhor solução encontrada $\mathbb{S}^*$ para o problema $\mathbb{P}$ dentro de $time$ segundos.

No Algoritmo 3, a linha 1 inicializa a melhor solução encontrada, $\mathbb{S}^*$, como vazia e a linha 2 assume o valor de função objetivo dessa solução como infinito. Observe que a função $f()$ calcula o custo da função objetivo de uma dada solução. As linhas 4 a 8 criam uma solução inicial com base no algoritmo construtivo da subseção 3.1. Primeiramente, as estruturas que contêm informações sobre uma solução para o problema são criadas e inicializadas com zeros (matrizes $\mathcal{X}^{\mathcal{E}}$, $\mathcal{X}^{\mathcal{P}}$ e vetor $\mathcal{CH}$) ou valores nulos (vetor $\mathcal{Y}$) (linha 5). Na linha 6 uma alocação evento-professor é obtida pelo algoritmo ALOCA_PROFESSORES (Algoritmo 1) e na linha 7 a alocação de horários é obtida de acordo com o algoritmo ALOCA_HORARIOS (Algoritmo 2). As linhas 9 a 15 realizam uma descida randômica não ascendente sobre a solução inicial $\mathbb{S}$. A cada iteração, um vizinho $\mathbb{S}'$ de $\mathbb{S}$ é gerado de acordo com a estrutura de vizinhança da subseção 3.2 (linha 11). Caso o vizinho $\mathbb{S}'$ tenha um custo de função objetivo menor que a solução corrente $\mathbb{S}$ (linha 12), o vizinho $\mathbb{S}'$ passa a ser a solução corrente $\mathbb{S}$ (linha 13). Ao final, se a solução corrente $\mathbb{S}$ tive um valor de função objetivo menor que o da melhor solução encontrada $\mathbb{S}^*$, a melhor solução encontrada $\mathbb{S}^*$ é substituída por $\mathbb{S}$.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

Algoritmo 3: MULTI-START: Metaheurística Multi-Start baseada nos algoritmos construtivos 1 e 2 e na estrutura de vizinhança da subseção 3.2.

Entrada: (i) Instância do problema $\mathbb{P}$ ($\mathcal{E}$, $\mathcal{P}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{H}$, $\mathcal{T}$, $\mathcal{CE}_c$, $\mathcal{CPE}_e$, $\mathcal{LE}$, $\mathcal{UC}_c$, $\mathcal{UP}_p$, $\mathcal{NP}_p$, $\mathcal{NPP}_p$, d_e, $\overline{dur}_p$, $\underline{dias}_p$, $\overline{dias}_p$),
(ii) Tempo limite $time$ em segundos,
(iii) Máximo de iterações para execução da busca local $iMax$

Saída: (i) Matriz de alocação de horários dos eventos $\mathcal{X}^{\mathcal{E}}$,
(ii) Vetor de alocação evento-professor $\mathcal{Y}$.

```
1  S* ← ∅;
2  f(S*) ← ∞
3  enquanto Tempo limite time não atingido faça
4      ▷ Gera uma solução inicial S.
5      Criar e inicializar as estruturas X^E, X^P, Y e CH com zeros ou valores nulos.
6      (Y, CH) ← ALOCA_PROFESSORES(P, Y, CH);
7      (X^E, X^P) ← ALOCA_HORARIOS(P, Y, X^E, X^P);
8      S ← (X^E, Y);
9      i ← 0;                                  ▷ Realiza busca local sobre a solução S
10     enquanto i < iMax faça
11         S' ← Gerar vizinho de S de acordo com a subseção 3.2;
12         se f(S') ≤ f(S) então
13             S ← S';
14         i ← i + 1;
15     se f(S) < f(S*) então
16         S* ← S;                  ▷ Melhor solução S* é atualizada pela solução corrente S
17 retorna S*;
```

4. Experimentos Computacionais

Os experimentos computacionais foram realizados em um notebook Dell G7-7588 A30P, processador Intel Core i7-8750H (cache 9M, 2.20 GHz até 4,10 GHz) com 16 gigabytes de memória RAM rodando o sistema operacional Windows 10. O código fonte foi desenvolvido em Python na versão 3.8.3. De modo a promover a reprodutibilidade dessa pesquisa, o código-fonte e as instâncias abordadas foram disponibilizadas em [omitido para preservar a anonimidade].

4.1. Caracterização das Instâncias

Para avaliar a abordagem de solução proposta por esse trabalho foram consideradas as instâncias do problema de alocação de horários do DECSI/UFOP dos últimos quatro semestres: 2018/2, 2019/1, 2019/2 e 2020/1. A Tabela 2 apresenta os principais dados das instâncias abordadas. As colunas $|\mathcal{E}|$, $|\mathcal{P}|$ e $|\mathcal{C}|$ representam, respectivamente, o número de eventos, professores e classes da instância. A coluna $|\mathcal{D} \times \mathcal{T}|$ apresenta o número total de horários disponíveis para alocação, enquanto as colunas $\overline{|\mathcal{UC}|}$ e $\overline{|\mathcal{UP}|}$ indicam o número médio de horários indisponíveis para cada classe e para cada professor respectivamente. A coluna $\overline{|\mathcal{NP}|}$ apresenta o número médio de horários indesejáveis por professor e a coluna $\overline{|\mathcal{NPP}|}$ apresenta o número médio de padrões de alocação indesejáveis por professor. A coluna $\overline{|\mathcal{CPE}|}$ indica o número médio de professores compatíveis por evento. Por fim, a coluna $\overline{|\mathcal{CH}|}$ apresenta a carga horária média por professor.

Tabela 2: Dados do problema de alocação de horários do DECSI/UFOP abordados no presente trabalho.

Instância	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{D}\times\mathcal{T}\rvert$	$\overline{\lvert\mathcal{UC}\rvert}$	$\overline{\lvert\mathcal{UP}\rvert}$	$\overline{\lvert\mathcal{NP}\rvert}$	$\overline{\lvert\mathcal{NPP}\rvert}$	$\overline{\lvert\mathcal{CPE}\rvert}$	$\overline{\lvert\mathcal{CH}\rvert}$
DECSI-2018/2	71	20	25	20	10.00	0.00	3.10	0.40	1.75	6.85
DECSI-2019/1	71	26	25	20	14.40	0.00	1.96	1.88	2.68	5.27
DECSI-2019/2	71	30	25	20	14.40	0.27	2.90	2.19	2.24	4.57
DECSI-2020/1	71	29	25	20	14.40	0.00	2.97	2.79	2.34	4.72

4.2. Resultados Obtidos

Para o registro dos resultados obtidos foram consideradas dez execuções do algoritmo proposto sobre cada instância. Dessas execuções são registrados na Tabela 3, respectivamente, o melhor valor de função objetivo encontrado dentre as execuções ($f(\mathbb{S}^*)$), o valor médio de função objetivo das soluções ($\overline{f(\mathbb{S})}$) juntamente com o desvio padrão observado $\pm$ e o valor médio de função objetivo das soluções iniciais obtidas pelo algoritmo construtivo ($\overline{f(\mathbb{S}_0)}$). O tempo limite considerado para execução de cada experimento foi $time = 60$ segundos e o número máximo de iterações sem melhoria foi de $iMax = 5,000$ (parâmetros de entrada do Algoritmo 3). Esses valores foram ajustados empiricamente.

Tabela 3: Resultados obtidos para as instâncias do problema de alocação de horários do DECSI/UFOP.

Instância	$f(\mathbb{S}^*)$	$\overline{f(\mathbb{S})}$	$\overline{f(\mathbb{S}_0)}$
DECSI-2018/2	1.00	1.96±0.65	37.59±4.02
DECSI-2019/1	5.90	8.54±1.35	38.25±3.51
DECSI-2019/2	10.80	13.75±1.29	50.51±4.66
DECSI-2020/1	9.00	12.80±1.87	50.87±4.43

4.3. Discussão dos Resultados

Por se tratar de um problema específico do DECSI/UFOP, é difícil realizar uma comparação dos resultados obtidos pela abordagem proposta com outros trabalhos da literatura. Entretanto, pela Tabela 3 pode-se observar que a busca local de refinamento está sendo efetiva para o problema, melhorando consideravelmente o valor da função objetivo das soluções iniciais encontradas. O desvio padrão observado foi baixo, o que indica robustez do método ao gerar as soluções.

Vale ressaltar que a abordagem proposta foi adotada pelo departamento em detrimento do método anterior de criação dos horários, que utilizava o software FET Free Timetabling Software [1] num processo iterativo para auxiliar na geração das soluções. O software FET, apesar de amigável ao usuário e eficiente, é capaz apenas de determinar se há uma solução factível para o problema ou não e exibi-la em caso afirmativo. Assim, para lidar com as restrições fracas do problema, os responsáveis pelo planejamento têm de inserir, uma a uma, essas restrições de modo a identificar as que não podem ser atendidas simultaneamente. Esse processo demanda estimadas 40 horas-homem de esforço para gerar um horário final e implica em problemas adicionais, como por exemplo a definição da ordem em que as restrições serão inseridas, que pode prejudicar ou favorecer um ou outro professor. Por fim, o software proposto automatiza completamente o processo de geração de soluções, limitando o esforço humano para geração dos horários apenas à definição dos dados de entrada e gera soluções satisfatórias para o problema.

[1] https://lalescu.ro/liviu/fet/

5. Considerações Finais

O presente trabalho apresentou um algoritmo heurístico para a solução do problema de programação de horários do DECSI/UFOP. O algoritmo proposto é baseado em *backtracking* e em força-bruta para gerar soluções iniciais e se utiliza de uma estrutura multi-vizinhança aliada à metaheuristica Multi-Start para refinar as soluções iniciais. O resultados obtidos foram satisfatórios e o sistema foi adotado pelo departamento, em detrimento da abordagem semi-automática assistida pelo software FET usada anteriormente.

Como trabalhos futuros, sugere-se: (i) implementar e avaliar e eficiência de outras metaheurísticas aplicadas ao problema, como GRASP, *Simulated Annealing* e Busca Tabu; (ii) desenvolver um modelo de programação matemática e solucionar o modelo através de um resolvedor comercial de programação inteira para obter limites inferiores para as instâncias do problema e, eventualmente, até mesmo soluções de qualidade; (iii) obter dados e requerimentos de outros departamentos da UFOP, como penalidades por alocar um professor a um evento, ou restrições de ordem para ocorrer eventos e adaptar a abordagem proposta para solucionar esses novos problemas.

Referências

Bagger, N.-C. F., Sørensen, M., e Stidsen, T. R. (2018). Benders' decomposition for curriculum-based course timetabling. *Computers & Operations Research*, 91:178–189.

Csima, J. e Gotlieb, C. (1961). Tests on a computer method for construction of school timetables. *Cornmunications of the ACM*, 7.

Fonseca, G. H. G., Santos, H. G., Toffolo, T. A. M., Brito, S. S., e Souza, M. J. F. (2016). Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, 239(1):77–97. ISSN 1572-9338.

Fonseca, G. H., Santos, H. G., Carrano, E. G., e Stidsen, T. J. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, 262(1):28 – 39. ISSN 0377-2217.

Knuth, D. E. (2014). *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional.

Martí, R. (2003). Multi-start methods. In *Handbook of metaheuristics*, p. 355–368. Springer.

Neufeld, G. A. e Tartar, J. (1974). Graph coloring conditions for the existence of solutions to the timetable problem. *Commun. ACM*, 17(8):450–453. ISSN 0001-0782.

Post, G., Kingston, J., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, K., Musliu, N., Pillay, N., e Santos, H. (2011). XHSTT: An XML archive for high school timetabling problems in different countries. *Annals of Operations Research*, 218:295–301.

Saviniec, L., Santos, M. O., Costa, A. M., e dos Santos, L. M. (2020). Pattern-based models and a cooperative parallel metaheuristic for high school timetabling problems. *European Journal of Operational Research*, 280(3):1064 – 1081. ISSN 0377-2217.

Skoullis, V. I., Tassopoulos, I. X., e Beligiannis, G. N. (2017). Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm. *Applied Soft Computing*, 52: 277 – 289. ISSN 1568-4946.

Souza, M. J. (2000). *Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas*. PhD thesis, Universidade Federal do Rio de Janeiro.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings