UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CURSO DE ENGENHARIA DE PRODUÇÃO

JOÃO PEDRO CORRÊA DA CONCEIÇÃO

AUTOMATIZAÇÃO DO PROCESSO DE GERAÇÃO DA GRADE HORÁRIA DO CURSO DE ENGENHARIA DE PRODUÇÃO DA UFRJ-MACAÉ

Macaé-RJ
2023

JOÃO PEDRO CORRÊA DA CONCEIÇÃO

AUTOMATIZAÇÃO DO PROCESSO DE GERAÇÃO DA GRADE HORÁRIA DO CURSO DE ENGENHARIA DE PRODUÇÃO DA UFRJ-MACAÉ

Trabalho de Conclusão de Curso de Graduação em Engenharia de Produção, Universidade Federal do Rio de Janeiro – UFRJ Campus Macaé, apresentado como requisito necessário para obtenção do grau de Bacharel em Engenharia de Produção.

Orientador: D.Sc. Thiago Gomes de Lima

Macaé-RJ
2023

CIP - Catalogação na Publicação

C744
Conceição, João Pedro Corrêa da
Automatização do processo de geração da grade horária do curso de Engenharia de Produção da UFRJ Macaé / João Pedro Corrêa da Conceição - Macaé, 2023.
66 f.

Orientador(a): Thiago Gomes de Lima.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto Politécnico, Bacharel em Engenharia Civil, 2023.

1. Pesquisa operacional. 2. Automatização. 3. Horário de aula.
4. Programação por restrições. I. Lima, Thiago Gomes de , orient. II. Título.

CDD 620

Ficha catalográfica elaborada pela Biblioteca com os
dados fornecidos pelo(a) autor(a)
Biblioteca Central do Centro Multidisciplinar UFRJ-Macaé
Bibliotecário: Anderson dos Santos Guarino CRB7 – 5280

JOÃO PEDRO CORRÊA DA CONCEIÇÃO

AUTOMATIZAÇÃO DO PROCESSO GERAÇÃO DA GRADE HORÁRIA DO CURSO DE ENGENHARIA DE PRODUÇÃO DA UFRJ-MACAÉ

Trabalho de Conclusão de Curso de Graduação em Engenharia de Produção, Universidade Federal do Rio de Janeiro – UFRJ, apresentado como requisito necessário para obtenção do grau de Bacharel em Engenharia de Produção.

Aprovado em Macaé, 23 de Janeiro de 2023

BANCA EXAMINADORA:

Thiago Gomes de Lima

Prof. D.Sc. Thiago Gomes de Lima (UFRJ)

Prof. D.Sc. Milena Estanislau (UFRJ)

Prof. Ma. Isabella Fischer (UFRJ)

Macaé-RJ
2023

AGRADECIMENTOS

Agradeço inicialmente à minha família, que sempre foi minha base e me apoiou em todas as minhas decisões até aqui. Minha mãe, que nunca mediu esforços para que eu pudesse realizar meus sonhos, mesmo que de outra cidade. Ao meu pai, meu exemplo de superação e trabalho que me inspirou durante toda a minha trajetória.

Sou grato às amizades construídas nessa jornada, as melhores que eu poderia ter, responsáveis por deixar a caminhada muito mais leve e oferecer o suporte necessário quando eu mais precisei.

Além disso, agradeço a esta Universidade e a toda a equipe de docentes, direção e administração, responsáveis por tantas experiências enriquecedoras que ajudaram a formar o profissional que eu me tornei.

Por último, agradeço aos professores que aceitaram compor a banca avaliadora e disponibilizaram seu tempo e conhecimento para contribuir com a minha formação. De maneira especial, ao meu orientador Thiago Gomes, parceiro que pude contar diversas vezes durante a minha passagem pela UFRJ.

RESUMO

É uma tradição nas universidades, o planejamento da grade horária para o próximo período, uma atividade atribuída aos coordenadores de curso. Na literatura, é uma demanda conhecida como *school timetabling*, parte dos estudos de Pesquisa Operacional. Trata-se de uma atividade que requer dedicação, tempo e estratégias assertivas para atender os diversos critérios acadêmicos bem como as necessidades das partes interessadas, tais como: professores, professores substitutos, alunos e coordenação. Nesse sentido, a presente monografia objetiva automatizar esse processo, de modo que ele se torne rápido e prático. Para isso, um script em python foi escrito, que importa os elementos professores, horários e matérias do período e exporta a grade horária em uma planilha de Excel, utilizando um modelo de programação por restrições, da área de estudo da Pesquisa Operacional. Os resultados mostram que a grade gerada atende a todas as restrições que uma grade horária da UFRJ-Macaé exige; e é gerada em minutos, ao invés de dias como no processo tradicional. Portanto, observou-se que os alunos também serão beneficiados por esse script, visto que o modelo prioriza alocar as matérias nos horários noturnos, para que os que estagiam possam se inscrever na maior quantidade de matérias possível.

Palavras-chave: Pesquisa operacional. Automatização. Grade horária. Programação por restrições. Python.

ABSTRACT

At the beginning of every semester on the Production Engineering course in the Federal University of Rio de Janeiro, Macaé campus, it is necessary that the course coordinator draws up a schedule for the next period. This activity, besides being stressful, takes a long time and generates a lot of rework, since, many times, it is necessary to adjust the schedule more than once per period, due to changes in the availability of professors, substitute professors, and several other reasons. In this sense, the present monograph aims to automate this process, so that it becomes fast and practical. For this, a python script was written, which imports the teachers, schedules and subjects of the period and exports the schedule in an Excel spreadsheet, using a constraint programming model, from the Operations Research study area. The generated schedule meets all the constraints that the UFRJ-Macaé schedule requires, and is generated in minutes, instead of days as in the traditional process. Furthermore, students will also benefit from this script, since the model prioritizes the allocation of subjects in the evening hours, so that those in training can enroll in as many subjects as possible.

Keywords*:* Operational Research. Automation. Timetable. Constraint programming. Python.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A proposta deste capítulo é apresentar os assuntos que serão abordados no decorrer do projeto. Portanto, foi dividido em cinco tópicos. No primeiro tópico, é apresentado o problema de *school timetabling* e como a área de Pesquisa Operacional pode ajudar a resolver esse problema. No segundo tópico, são apresentados o objetivo geral e os objetivos específicos do trabalho. Após isso, são evidenciadas a justificativa e a motivação para a realização do estudo. Por fim, o quinto tópico se concentra em apresentar a estrutura do trabalho.

1.1 Contextualização

Tradicionalmente, nas instituições de ensino superior, antes do início do período letivo, gestores de instituições de ensino se encontram frente a uma tarefa recorrente, que é montar a grade horária do curso. Esse processo (montar grades horárias) é conhecido na literatura como *school timetabling*, e é definido por Cooper e Kingston (1993) como alocar professores, alunos e salas de aula em uma coleção de turmas.

Em instituições de ensino que não dispõe de algum tipo de ferramenta computacional que auxilie nesse processo, dias e até semanas são despendidos nesta tarefa, demandando tempo e exigindo exaustivas negociações entre professores com diferentes disponibilidades horária e preferências de disciplinas (Poulsen, 2012).

A dificuldade dos problemas de *school timetabling* está no fato de sua natureza combinatória e no tamanho dos problemas. Sabe-se que, em muitos casos reais, as turmas podem ter centenas de participantes e, por outro lado, aulas com poucos horários disponíveis, bem como restrições de vários tipos (Cooper e Kingston, 1993). Isto implica a necessidade de se pensar em uma quantidade enorme de possíveis soluções. Segundo Andrade et al. (2019), os problemas de *timetabling* são tão complexos que na maioria das vezes achar uma solução factível (que respeite todas as restrições impostas) faz com que o problema seja considerado solucionado, mesmo que a qualidade da solução não seja boa. Uma solução ruim implica alunos deixando de cursar matérias por não conseguirem se inscrever; professores com horários ruins, ou mesmo, lecionando matérias que não são de sua preferência.

Ainda de acordo com Andrade et al. (2019), a criação de um sistema capaz de automatizar a solução do problema de *timetabling* permitiria às instituições de ensino prepararem suas grades horárias de maneira mais rápida e com qualidade superior, fazendo com que coordenadores pudessem investir mais tempo em outros assuntos importantes, tais como apoiar projetos extra curriculares, desenvolvimento de pesquisas, projetos de extensão ou orientar alunos em trabalhos de conclusão de curso.

Frente aos fatos apresentados até aqui, o presente trabalho pretende desenvolver um programa que automatize a geração de grade horária para o curso de Engenharia de Produção. Portanto, para tanto, pretende-se utilizar o curso de Engenharia de Produção da Universidade Federal do Rio de Janeiro (UFRJ) localizado no Campus Macaé, como um estudo de caso.

1.2 Objetivos

Nesta seção, serão apresentados o objetivo geral e os objetivos específicos do estudo.

1.2.1 Objetivo Geral

Promover a automatização da geração da grade horária do curso de Engenharia de Produção, na UFRJ Campus Macaé por meio de um script em Python. Para isso, espera-se que esse script possa ler os inputs necessários (como nome dos professores, matérias e horários) em uma planilha do *Google Sheets,* bem como, exportar grade horária final na mesma planilha, em uma aba separada;

1.2.2 Objetivos Específicos

- Contextualizar teoricamente a programação por restrições e o *school timetabling*
- Definir os critérios necessários para a construção de uma grade horária válida para o curso de Engenharia de Produção da UFRJ Macaé;
- Modelar matematicamente em variáveis, restrições e função objetivo os critérios definidos no primeiro ponto;

1.3 Justificativa

A dificuldade para criar uma grade horária diferente a cada novo semestre letivo é bastante conhecida nos bastidores das universidades. Segundo Alvarez-Valdes, Martin e Tamarit (1996) o problema de *school timetabling* varia muito de país para país, devido às diferentes regras e características que cada sistema educacional possui. Dentro de um mesmo país as restrições de uma grade horária muda bastante: em algumas instituições de ensino as matérias já tem um professor específico, em algumas as salas de aula já são fixas para uma turma de alunos, em outras os alunos se deslocam de uma sala para outra, e por aí vai.

Na UFRJ Macaé, mais especificamente no curso de Engenharia de Produção, é especialmente desafiador montar uma grade horária. Alguns fatores contribuem para dificultar esse processo, entre eles, o fato de não existir professores fixos para ministrar alguma matéria, a cada novo período é preciso decidir qual professor irá ministrar qual matéria; não é possível ofertar todas as matérias do curso em todos os semestres, logo, uma matéria que era prioridade de ser alocada na grade horária em uma período pode não ser prioridade no próximo; adicionalmente, o curso conta com as contratações de professores

substitutos, no qual cada um dos respectivos professores apresentam disponibilidades de horários diferentes e específicas.

Percebe-se ainda, que a coordenação tem que gerenciar o conflito de interesses entre professores com disponibilidade de horário na parte da manhã e alunos que possuem estágio/ trabalho nesse mesmo período, e por isso precisam que as matérias sejam alocadas no período da noite. Por esses motivos, a cada novo semestre o processo de construção de uma grade horária começa basicamente do zero.

Desta maneira, o trabalho se justifica devido ao alto investimento de tempo e energia por parte da coordenação do curso em questão, somado a necessidade dos alunos de terem uma grade horária que atenda os seus interesses e particularidades. Neste sentido, essa problemática poderia ser minimizada, através de um processo automatizado, que pode contribuir para atender as necessidades de diferentes partes interessadas, como os alunos, docentes e a coordenação do curso.

1.4 Motivação pessoal

Pelo contato próximo do autor com a coordenação e professores do curso, sempre ficou claro como é desgastante e difícil o processo de montar a grade horária do semestre letivo. Além disso, por diversas vezes o autor deste trabalho foi um dos alunos que apresentou dificuldades em organizar as disciplinas que precisava cursar ante a grade horaria proposta. Somado a isso, o autor sempre procurou retribuir um pouco todos os aprendizados e coisas boas que a UFRJ proporcionou para ele ao longo da sua trajetória acadêmica. Este trabalho é uma forma de deixar algo concreto que possa ser usado em benefício da universidade. Adicionalmente, uniu-se a complexidade matemática do trabalho com o interesse do autor em programação para a escolha da solução aqui proposta. A linguagem de programação Python foi uma decisão lógica, pois está presente na grade curricular do curso e possui bibliotecas específicas da área como *o Google OR-Tools*.

1.5 Estrutura do trabalho

O presente trabalho está dividido em 6 seções. Na primeira sessão, foi apresentado a introdução do trabalho, o objetivo geral e os específicos, a justificativa e a motivação do autor. Na segunda seção será contextualizado o referencial teórico, os conceitos matemáticos usados quanto a definição e apresentação das ferramentas computacionais utilizadas. Na terceira seção, discorre sobre o estudo de caso, apresentando a universidade e o curso em questão e esclarecendo a aplicação do modelo matemático e computacional, além de discutir os resultados. Na quarta seção, será apresentado as conclusões do trabalho, indicando

caminhos para futuros trabalhos na área. Por fim, serão disponibilizadas as referências bibliográficas e anexos.

2 Referencial Teórico

Nesta seção será elucidado a Pesquisa Operacional, além de detalhar o método de Pesquisa Operacional utilizado na resolução do problema, a Programação por Restrições. Assim, será introduzida a linguagem de programação Python e suas bibliotecas utilizadas na elaboração do script final. Por último, são expostos os principais conceitos acerca do *school timetabling*.

2.1 Pesquisa Operacional

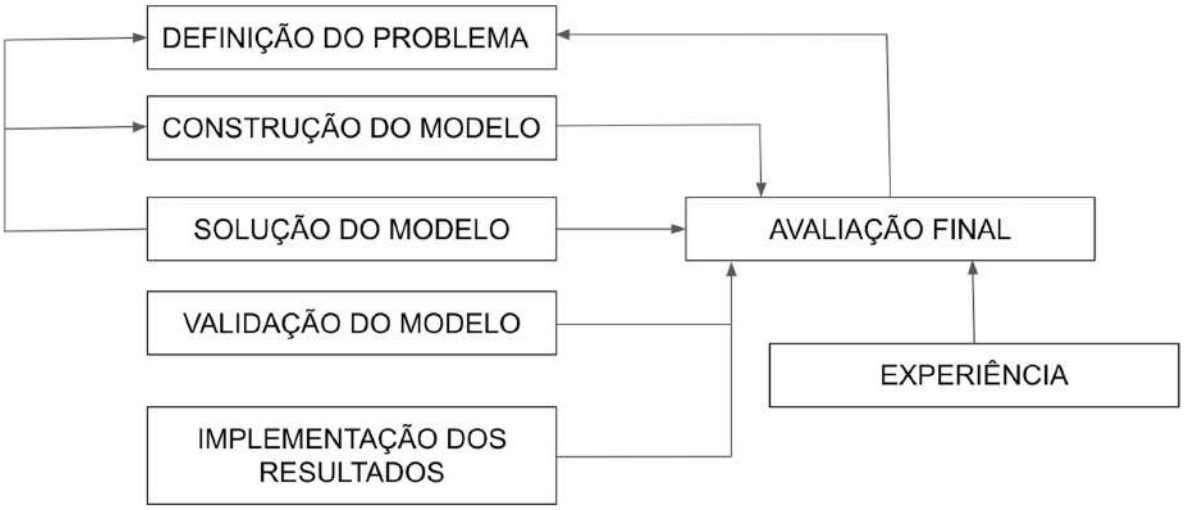
De acordo com Marins (2011), a Pesquisa Operacional surgiu pela primeira vez durante a Segunda Guerra Mundial, quando equipes de pesquisadores desenvolveram métodos para solucionar problemas de operações militares relacionados a alocação de recursos escassos, como aviões, submarinos, radares, para um grande número de alvos e operações militares. Diante do sucesso dessas aplicações e com o crescimento econômico pós-guerra, o mundo acadêmico e empresarial passou a utilizar esses métodos e ferramentas desenvolvidos nos ramos comercial, industrial e governamental, em que os recursos a serem alocados eram matérias-primas, pessoas e máquinas.

A Pesquisa Operacional é um método científico para tomadas de decisão e, geralmente, é utilizado para maximizar lucros ou minimizar custos. Esse método consiste basicamente na transformação de situações reais em modelos matemáticos, que servirão como a base. Segundo Marins (2011), um estudo em Pesquisa Operacional consiste em seis fases principais (figura 1).

1. Definição do problema - é uma descrição exata dos objetivos do estudo, identificando as alternativas de decisões existentes, limitações, restrições e exigências do sistema.
2. Construção do modelo - a partir da definição do problema, constrói-se uma representação formal do mesmo.
3. Solução do modelo - essa terceira fase tem por objetivo encontrar a solução para o modelo construído. Para modelos matemáticos, a solução é obtida pelo algoritmo mais adequado, em termos de rapidez, processamento e precisão.
4. Validação do modelo - um modelo é considerado válido se for capaz de fornecer uma previsão aceitável de seu comportamento. Uma forma comum de validação é o teste com dados passados.
5. Implementação dos resultados - nessa fase, a solução ótima obtida é convertida em regras operacionais.

6. Avaliação Final - a avaliação final conta com a experiência do pessoal envolvido no estudo para avaliar e determinar a aplicabilidade da decisão.

Figura 1 - Fases da Pesquisa Operacional



Fonte: Elaborado pelo autor

A utilização dos modelos permite a experimentação, o que significa que a decisão pode ser avaliada e testada antes de ser implementada. Nessa fase de experimentação algumas inconsistências do modelo podem ser evidenciadas, levando a uma redefinição da formulação inicial. Esse processo se repete até que se encontre uma decisão ótima com avaliação final satisfatória.

Pode-se perceber que a qualidade do modelo gerado na pesquisa é de extrema importância para o sucesso de todo o processo de tomada de decisão. Sendo assim, o processo de modelagem e o tipo de modelo a ser feito deve ser escolhido com cautela.

Os modelos podem ser divididos em dois grandes tipos: modelos de simulação e modelos de otimização. Será descrito detalhadamente o modelo de otimização, que é objeto de estudo deste trabalho. Existe uma dificuldade de alocação de recursos, matéria prima, mão de obra e tempo em diversas áreas acadêmicas, industriais e empresariais. Para alocar esses diferentes recursos de forma eficiente e eficaz, aplica-se os processos e modelos de otimização. A área que estuda a otimização é denominada Programação Matemática e descreve as quantidades a serem maximizadas ou minimizadas como uma função matemática dos recursos escassos. As relações entre as variáveis das funções são formalizadas através de restrições e expressas como equações matemáticas.

Como é uma área de estudo ampla, a modelagem de otimização é dividida em duas áreas menores: Programação Linear e Programação Não-Linear. A ferramenta de estudo e

enfoque deste trabalho é a Programação Linear, mais precisamente, a Programação por Restrições (PR).

2.1.1 Programação por restrições

A programação por restrições é formada pela união de dois paradigmas computacionais: a programação lógica e a resolução de restrições. Mesmo sendo bastante recente face a outras estratégias de modelagem, a programação por restrições tem se mostrado promissora na resolução de diversos problemas de otimização combinatória (Cherri, 2018). Hooker & Osório (1999) demonstraram que ela pode ser de 4 a 150 vezes mais rápida do que a Programação Inteira, a técnica mais comumente utilizada em problemas de decisão.

A ideia desse tipo de modelagem é, basicamente, a declaração de restrições sobre as variáveis do problema e busca por uma solução que satisfaça todas as restrições. Na prática, cria-se inicialmente o domínio para cada variável do problema e vai restringindo cada domínio à medida que se avalia cada restrição. No fim do processo, obtém-se uma ou várias soluções que satisfaçam as restrições ou nenhuma solução.

De fato, a escolha da modelagem de problemas combinatórios utilizando a PR pode proporcionar diversas vantagens. Evidentemente, uma delas é a flexibilidade de modelização do problema, podendo utilizar equações lineares e não lineares, operadores lógicos e condições lógicas. Isso permite uma formulação mais intuitiva das características intrínsecas de alguns problemas.

2.1.2 Problema de satisfação por restrição

No centro da PR está o problema da satisfação de restrições, do inglês constraint satisfaction problem (CSP). Brailsford, Potts e Smith (1999) definem CSPs da seguinte maneira: Dado um conjunto de variáveis discretas, juntamente com domínios finitos, e um conjunto de restrições envolvendo essas variáveis, encontre uma solução que satisfaça todas as restrições. Em CSPs, as variáveis podem assumir diferentes tipos, incluindo: booleano, inteiro, simbólico, elementos de conjunto e subconjuntos de conjuntos. (Kanet & Gorman, 2004).

Os métodos utilizados na resolução de CSP são vantajosos em explorar um grande número de possibilidades de soluções de forma eficiente, buscando dar uma solução viável ao invés de buscar a solução ótima. Isso compromete sua aplicabilidade em certos casos, mas no problema de criação de grade horária em questão a modelização em CSP pode ser bastante útil.

Tal abordagem se alinha bem ao problema em questão pois qualquer solução que consiga alocar todos os professores, satisfazendo as restrições, é tão boa quanto qualquer outra que também o faça. Além disso, é possível utilizar restrições fracas (cuja satisfação é desejável, mas não obrigatória) a fim de modelizar certas preferências particulares. Um exemplo comum, que foi explorado neste trabalho, é a preferência por determinados horários pelos professores: existe uma preferência pelos horários da tarde para as disciplinas do ciclo profissional de Engenharia da Produção.

2.2 Problemas de horários de escolas - School Timetabling

O problema da criação de quadro de horário escolar (School Timetabling) pertence à grande família de problemas de agendamento (Scheduling Problems). Segundo Wren (1996), a programação de horários consiste na realização de uma alocação, sujeita a restrições, de recursos a objetos colocados no espaço e no tempo, de modo a satisfazer, na medida do possível, um conjunto de objetivos desejáveis.

Quando se trata de problemas de programação de horários em uma instituição de ensino, a solução é programar uma série de encontros (aulas), entre professores e alunos que satisfaçam um conjunto de restrições envolvendo os recursos disponíveis (professores, horários disponíveis e matérias que cada professor está apto a lecionar). Even et al. (1976), demonstra que esses problemas não são facilmente solucionáveis e então o tempo que necessário para sua encontrar cresce exponencialmente à medida que a escala do problema aumenta.

Dessa forma, para solucionar o problema de criação de grades horárias de forma eficiente e confiável utilizam-se algoritmos fornecidos pela Pesquisa Operacional (PO) que ajudam a se aproximar de um resultado ótimo de maneira eficiente e rápida. A literatura atual contém análises de diversas variantes desse problema, de diferenciando em função da classe da instituição envolvida (universidade, escola, empresa) e das restrições impostas (SCHAERF, 1999).

Diversos métodos foram propostos para a resolução de problemas de programação de horários. Inicialmente, pesquisas sobre o tema se concentravam nos métodos ligados à teoria dos grafos, uma vez que o problema pode ser entendido como um problema de coloração, como é evidenciado nos trabalhos de De Werra (1970) e Dempster (1971). Atualmente, a literatura sobre o assunto é vasta e apresenta diversas abordagens matématicas como a Busca Tabu (Hertz, 1992; Alverez Valdes et al., 2002; Pereira et al., 2007), Algoritmos Evolutivos (Souza et al., 2002; Lobo, 2005; Coelho, 2006), Programação por Restrições (Goltz e Matzken, 1999), Método de Colônias de Formigas (Eley, 2006), Programação Inteira (Crovo

et al., 2007; Daskalaki et al., 2004), Redes Neurais (Carrasco e Pato, 2004; Smith et. al, 2003), além de metaheuristicas como Variable Neghborhood Search (VNS) e Simulated Annealing (Silva Neto, 2005), e Particle Swarm (Fard et al., 2007).

Dado que o problema estudado pode ser descrito em termos de um conjunto de restrições arbitrárias, que podem variar de forma dinâmica, a modelagem escolhida foi a de Programação por Restrições. Isso permite que a formulação das restrições seja feita de uma forma mais declarativa do que em outros métodos. A Programação por Restrições é baseada na viabilidade (encontrar uma solução viável) ao invés da otimização (encontrar uma solução ótima) e foca nas restrições e variáveis ao invés da função objetivo. Na verdade, um problema de CP pode nem mesmo ter uma função objetivo - o objetivo pode ser simplesmente restringir um grande conjunto de soluções possíveis a um subconjunto mais gerenciável, adicionando restrições ao problema.

Desse modo, o presente trabalho visa, primeiramente, a modelagem do problema de criação de grade horária para o curso de Engenharia de Produção da Universidade Federal do Rio de Janeiro, Campus Macaé como um problema de Programação por Restrições e, em seguida, sua resolução com o auxílio de um software de solver.

2.3 Linguagem de Programação Python

A linguagem de programação Python surgiu no início de 1990, sendo criada por Guido van Rossum. É, desde a sua criação, uma linguagem de código aberto, sendo desenvolvida com a contribuição de várias pessoas e mantida pela Python Software Foundation (PFS, 2010).

Com o crescimento da ciência de dados, a linguagem Python vem crescendo no mercado, sendo uma das mais populares da atualidade. Por terem linguagem amigável e possibilidade de colaboratividade, os códigos em Python estão sendo cada vez mais recorrentes.

A linguagem Python conta com uma biblioteca padrão muito extensa, que oferece uma ampla gama de recursos e funções. Além disso, é possível usar outras bibliotecas, criadas por terceiros, que são um conjunto de módulos e funções úteis que reduzem o uso de código no programa. Existem mais de 137 mil bibliotecas Python com diversas finalidades que facilitam a programação dos desenvolvedores.

O modelo desenvolvido neste trabalho foi implementado em linguagem de programação Python, usando a biblioteca OR-Tools. O OR-Tools é uma biblioteca de código aberto para escrever modelos de otimização através de uma linguagem de modelagem algébrica. De acordo com Google (2022), a biblioteca é definida como um pacote de software

de código aberto para otimização, ajustado para enfrentar os problemas mais difíceis do mundo em roteamento de veículos, fluxos, programação inteira e linear e programação de restrição.

Após a implementação do modelo, o programa deve ser executado por um solver. Existem diversos pacotes disponíveis para esse fim, como Gurobi e CPLEX (versões pagas) ou versões de código aberto e gratuitas como SCIP, GLPK, GLOP e CP-SAT.

Para viabilizar a produção deste trabalho, escolheu-se utilizar a linguagem de programação Python aliada ao pacote de otimização OR-Tools, devido a sua versatilidade e fácil utilização. Além disso, para executar o modelo, escolheu-se o solver CP-SAT, que possui grande destaque dentre os de sua categoria. Essas ferramentas juntas permitem a clareza de código e modelagem de alto nível, alto desempenho e extensibilidade e configurabilidade.

3 Procedimentos Metodológicos

Neste capítulo, será apresentado a metodologia do presente trabalho. Portanto, este estudo pode ser classificado de 4 pontos de vista diferentes: em relação a sua natureza, objetivos, procedimentos técnicos e abordagem do problema. Além disso, será apresentado o passo a passo metodológico de como esse projeto foi desenvolvido e será feita a contextualização de onde esse estudo foi feito.

3.1 Classificação da pesquisa

Segundo Silva (2004 apud PRODANOV e FREITAS, 2013), é possível classificar uma pesquisa científica de várias formas diferentes. A Tabela a seguir apresenta cada uma delas, junto com as suas respectivas opções:

Tabela 1 - Classificação de Pesquisa Científica

Ponto de vista	Opções	Classificação da atual pesquisa
Quanto a sua natureza	Pesquisa básica	**Pesquisa aplicada**
	Pesquisa aplicada	
Quanto aos seus objetivos	Pesquisa exploratória	**Pesquisa explicativa**
	Pesquisa descritiva	
	Pesquisa explicativa	
Quanto aos seus procedimentos técnicos	Pesquisa bibliográfica	**Estudo de caso**
	Pesquisa documental	
	Pesquisa experimental	
	Levantamento (survey)	
	Pesquisa de campo	
	Estudo de caso	
	Pesquisa *ex-post-facto*	
	Pesquisa-ação	
	Pesquisa participante	
Quanto à abordagem do problema	Pesquisa quantitativa	**Pesquisa quantitativa**
	Pesquisa qualitativa	

Fonte: Elaborado pelo autor

Do ponto de vista da natureza, este estudo é considerado uma **pesquisa aplicada**, visto que o mesmo gera conhecimentos para a aplicação prática voltado para a solução de um problema específico, a criação da grade horária do curso de Engenharia de Produção da UFRJ Campus Macaé. Além disso, esta pesquisa também envolve interesses locais, característica da pesquisa aplicada.

Segundo Prodanov e Freitas (2013), a pesquisa explicativa necessita manipular e controlar as variáveis, visando identificar qual variável independente que determina o

fenômeno em estudo. Logo, esta pesquisa pode ser considerada como explicativa, pois a mesma visa explicar as restrições que invalidam uma grade horária, além de criar um programa que manipula as variáveis com o intuito de gerar uma grade horária possível.

Para analisar uma pesquisa em relação aos procedimentos técnicos é necessário avaliar a maneira pela qual os dados necessários foram obtidos. O presente documento pode ser classificado como estudo de caso**,** pois "envolve o estudo profundo e exaustivo de um ou poucos objetos de maneira que permita o seu amplo e detalhado conhecimento" (YIN, 2001 apud PRODANOV e FREITAS, 2013). Segundo Gil (2008 apud PRODANOV e FREITAS, 2013), através de estudos de caso, encontra-se ainda a tentativa de solucionar um problema social por meio de aplicação prática de conhecimentos. Um estudo de caso busca analisar o objeto de estudo com maior precisão, coletando e analisando informações sobre determinado indivíduo, família, grupo ou comunidade.

A pesquisa quantitativa transforma informações do mundo real em números, visando classificá-las e analisá-las. No desenvolvimento de uma pesquisa dessa natureza, os pesquisadores formulam hipóteses e classificam a relação entre as variáveis. (PRODANOV e FREITAS, 2013). O atual estudo faz exatamente isso, entende as restrições de um problema do mundo real e transforma em equações matemáticas. Coleta informações como disponibilidade de horários de cada professor, matérias que estão aptos a lecionar, número de aulas por semana de cada matéria, e transforma essas informações em variáveis matemáticas.

3.2 Contextualização do Caso

3.2.1 UFRJ-Macaé

Em 1920, segundo Stallivieri (2007), surge a primeira Universidade do Brasil, a Universidade do Rio de Janeiro, que hoje é conhecida como Universidade Federal do Rio de Janeiro (UFRJ). O início da UFRJ em Macaé começa em 1980, com a criação do NUPEM (Núcleo de Pesquisas Ecológicas de Macaé), que hoje é chamado de Instituto de Biodiversidade e Sustentabilidade. O NUPEM é o primeiro núcleo da UFRJ no norte fluminense, e se deu devido ao interesse de pesquisadores da Universidade em realizar estudos nas lagoas costeiras da região.

Em 2006, ocorreu a criação do primeiro curso de graduação da UFRJ fora da sua sede, na capital do estado, com o curso de Licenciatura em Ciências Biológicas. Em 2008, o Consuni (Conselho Universitário, que é o Órgão máximo de função normativa, deliberativa e de planejamento da Universidade) aprovou a criação do campus UFRJ Macaé. Hoje o

Campus UFRJ-Macaé possui mais de 3 mil estudantes, com 9 cursos de graduação e 4 de pós-graduação.

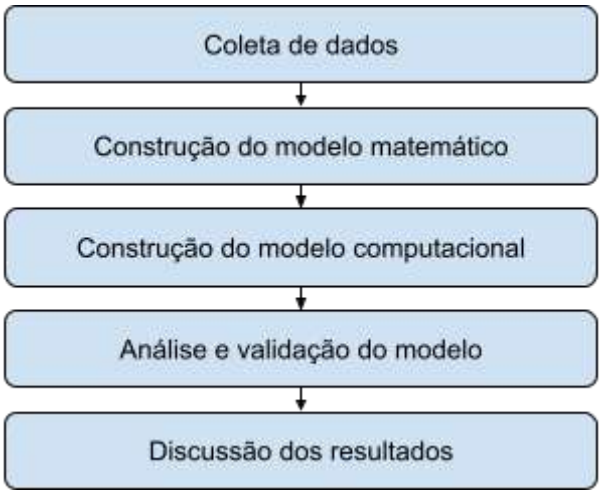
A proposta de implantação do curso de Engenharia de Produção no Campus de Macaé da UFRJ foi aprovada pelo CEG (Conselho de Ensino de Graduação) no dia 07/07/2010 e pelo Consuni em 10/09/2010, junto com outras 2 engenharias: Engenharia Civil e Mecânica.

Para um aluno obter grau e diploma em qualquer uma das 3 engenharias, ele precisa primeiro concluir o ciclo básico (Núcleo Comum), que tem a duração de 4 períodos. A partir daí, ele escolhe qual engenharia ele deseja concluir, no que é conhecido como o ciclo profissional. Cada uma das engenharias presentes no campus Macaé tem suas particularidades no ciclo profissional. O alvo do presente estudo é a Engenharia de Produção.

3.3 Passo a passo metodológico

As etapas metodológicas da pesquisa estão ilustradas no fluxograma da figura 2. Cada uma das etapas abaixo será detalhada no capítulo 4. Abaixo do fluxograma tem-se uma breve descrição do que será realizado em cada uma das etapas.

Figura 2 - Fluxograma metodológico



Fonte: Elaborado pelo autor

O objetivo da coleta de dados nesse trabalho é validar o modelo computacional gerado com dados reais de professores em atividade no curso de Engenharia de Produção da UFRJ Macaé. Para tal, foi passado um formulário que perguntava o nome do(a) professor(a), quais disciplinas do curso de Engenharia de Produção ele(a) é apto(a) a lecionar e quais são os horários que ele(a) está disponível para ministrar aulas.

Com a base de dados de professores, quais horários estão disponíveis e quais matérias se sentem confortáveis em lecionar é possível fazer simulações fidedignas da grade horária gerada pelo programa e fazer as validações necessárias. Além disso, é nesse momento da pesquisa que são coletados todos os requisitos de uma grade horária, que serão transformados

em equações matemáticas no tópico 3.2.2. Requisitos como “Não podem ser ofertadas 2 disciplinas do mesmo período no mesmo horário” que são coletados nessa etapa. Todos os requisitos serão listados no tópico 4 (Aplicação).

Em relação a construção do modelo matemático, conforme Costa (2018) a modelagem matemática tem como objetivo explicar matematicamente situações do cotidiano, e esse é o objetivo desta seção. Conseguir transformar em equações matemáticas todas as regras por trás da criação de uma grade horária da Universidade Federal do Rio de Janeiro.

Segundo Belfiore e Fávero (2013) um modelo matemático “consiste em um conjunto de equações (função objetivo e restrições de igualdade) e inequações (restrições de desigualdade) que tem como objetivo otimizar a eficiência do sistema”. No presente estudo, a função objetivo é maximizar a quantidade de disciplinas ofertadas no período, com as restrições de horários e disciplinas de cada professor. A modelagem matemática (incluindo a função objetivo e as restrições) serão detalhadas no tópico 4.2 (Construção do Modelo Matemático). Além disso, no mesmo tópico será explicado qual tipo de modelo (programação linear, programação binária, entre outros) será usado no estudo.

Em um estudo de Pesquisa Operacional, após a construção do modelo matemático temos a solução do mesmo (Belfiore e Fávero, 2013). No caso da pesquisa em questão, será usado um modelo computacional para a solução do problema, por isso a próxima etapa da pesquisa é a construção do modelo computacional. Cada equação matemática do modelo será implementada ao código do programa (tópico 3.2.3).

Acerca da construção do modelo computacional, essa etapa tem como objetivo mapear o modelo matemático para uma linguagem de programação ou para um *software* específico. A modelagem computacional é vital no trabalho em questão pela complexidade matemática do modelo final, além de trazer praticidade para quem for usufruir do programa final gerado nesta pesquisa. A cada novo semestre letivo a coordenação de Engenharia de Produção da UFRJ Macaé poderá executar o programa com as informações dos professores do semestre em questão e terá automaticamente uma grade horária para o período. Sem um modelo computacional isso não seria possível.

A análise e a validação do modelo, segundo Aragão (2011) é necessário analisar 2 pontos de um modelo computacional: (i) a sua sintaxe e semântica, que em sua maioria são identificados automaticamente pelo próprio editor de texto utilizado para escrever o código do modelo; (ii) comparação dos resultados gerados com o sistema. Segundo Freitas Filhos (2001 apud ARAGÃO, 2011) a proximidade entre os resultados obtidos pelo modelo e os originados pelo sistema real são parâmetros para medir a qualidade e validade do modelo. Na

pesquisa em questão, um bom parâmetro é analisar a grade horária gerada e verificar se alguma regra não está sendo cumprida.

Por fim, a discussão dos resultados deste trabalho será analisada em 2 pontos: (i) a praticidade do programa e (ii) a qualidade das grades horárias geradas. A seguir uma breve explicação da importância de cada item.

i. A praticidade do programa é essencial de ser analisada pois o objetivo desse projeto é facilitar a vida da coordenação de Engenharia de Produção da UFRJ Macaé, que todo semestre investe muito tempo e energia para gerar uma grade horária boa para os alunos;

ii. Não adianta o programa ser prático e não gerar grade horárias de qualidade. Uma grade horária de qualidade é uma que cumpre todos os requisitos definidos na etapa 3.2.1 (Coleta de dados) e na etapa 3.2.2 (Construção do modelo matemático).

4 RESULTADOS

Será descrito neste capítulo a parte prática do projeto, detalhando como foi feito a coleta de dados, a construção do modelo matemático e computacional. Neste capítulo também serão mostrados os passos para a validação do modelo computacional e os resultados do mesmo. Em resumo, as etapas da aplicação do projeto são: coleta de dados do corpo docente do curso de Engenharia de Produção da UFRJ Macaé e dos requisitos de uma grade horária (tópico 4.1), a construção do modelo matemático (tópico 4.2), construção e validação do modelo computacional (tópico 4.3) e por último será a análise e discussão dos resultados obtidos (tópico 4.4).

4.1 Coleta de dados

A coleta de dados foi dividida em 2 partes: na primeira, o objetivo foi entender quais eram todas as restrições e fatores limitantes no processo de gerar uma grade horária para o curso em estudo. A segunda teve o intuito de conseguir os dados do atual corpo docente da Engenharia de Produção da UFRJ-Macaé, a fim de ter uma base de dados sólida para testar o modelo no futuro.

4.1.1 Requisitos de uma grade horária do curso de Engenharia de Produção da UFRJ-Macaé

Para se entender todos os requisitos e restrições, foi preciso primeiro entender quais elementos compõem uma grade horária do curso do presente estudo. Os elementos que compõe uma grade horária são:

- **Matérias (m).** São as disciplinas que serão ofertadas no período. Cada matéria tem um período específico e um número de créditos. Os créditos são o número de horas de aula por semana que a disciplina tem que ter (a grande maioria das disciplinas tem um número par de créditos). Por exemplo, se uma matéria tem 4 créditos, isso significa que ela tem que ter 4 horas de aula por semana;
- **Professores (p).** São os professores que ministrarão as matérias. Cada professor tem um conjunto de matérias que está apto a lecionar (matérias que ele é especializado e se sente confortável em ministrar) e também um conjunto de horários que está disponível para ministrar alguma matéria.
- **Horários (h).** São os intervalos de tempo em que uma matéria pode ser ministrada por um professor. Os horários têm dia da semana, horário de início e horário de fim.

  A **TABELA 2** mostra os horários dos cursos de Engenharia da UFRJ Macaé.

Tabela 2- Horários das Engenharias da UFRJ Macaé

Código	Dia da semana	Horário início	Horário fim
Seg1	Segunda	08h00	10h00
Seg2	Segunda	10h00	12h00
Seg3	Segunda	13h30	15h30
Seg4	Segunda	15h30	17h30
Seg5	Segunda	17h30	19h30
Seg6	Segunda	19h30	21h30
Ter1	Terça	08h00	10h00
Ter2	Terça	10h00	12h00
Ter3	Terça	13h30	15h30
Ter4	Terça	15h30	17h30
Ter5	Terça	17h30	19h30
Ter6	Terça	19h30	21h30
Qua1	Quarta	08h00	10h00
Qua2	Quarta	10h00	12h00
Qua3	Quarta	13h30	15h30
Qua4	Quarta	15h30	17h30
Qua5	Quarta	17h30	19h30
Qua6	Quarta	19h30	21h30
Qui1	Quinta	08h00	10h00
Qui2	Quinta	10h00	12h00
Qui3	Quinta	13h30	15h30
Qui4	Quinta	15h30	17h30
Qui5	Quinta	17h30	19h30
Qui6	Quinta	19h30	21h30
Sex1	Sexta	08h00	10h00
Sex2	Sexta	10h00	12h00
Sex3	Sexta	13h30	15h30
Sex4	Sexta	15h30	17h30
Sex5	Sexta	17h30	19h30
Sex6	Sexta	19h30	21h30

Fonte: Elaborado pelo autor

Sempre que um novo período letivo está para começar, os coordenadores dos cursos de Engenharia da UFRJ Macaé têm que montar a grade horária do novo semestre. O objetivo é conseguir alocar o máximo de disciplinas, para poder ofertar o máximo de matérias possíveis para os alunos. No entanto, alguns requisitos têm que ser cumpridos para que a grade horária seja válida. Esses requisitos são:

(a) Cada professor só pode ser alocado em uma matéria por horário.
(b) Cada professor só pode lecionar no máximo 12 horas de aulas por semana (12 créditos) se for efetivo e 8 horas por semana se for substituto.
(c) Toda matéria alocada deve ter o total de número de aulas alocados por semana.
(d) Não pode ter mais de 1 matéria do mesmo período no mesmo horário.
(e) Cada matéria só pode ser lecionada por 1 único professor.
(f) Professores só podem lecionar matérias que estão aptos a lecionar.
(g) Os professores só podem lecionar matérias nos horários em que eles estão disponíveis.
(h) Todas as aulas de uma mesma matéria devem estar alocadas em horários que comecem na mesma hora (por exemplo, se uma aula de Engenharia do Produto começa às 15h30 da quarta-feira, a outra aula não pode ser às 08h00 da quinta, tem que começar às 15h30 de outro dia).

Além desses requisitos, um ponto que o coordenador deve tentar otimizar ao máximo é em relação aos horários alocados. Como no curso de Engenharia de Produção tem muita gente que já está estagiando, e a maioria dos estágios ocorre no período da manhã e no começo da tarde, o ideal para uma grade horária é que a maioria das matérias sejam ofertadas no período da noite, para que os alunos que tenham estágio possam aproveitar ao máximo. Essa "condição" será introduzida na função objetivo dentro do nosso modelo matemático (tópico 4.2).

4.1.2 Dados do corpo docente do curso de Engenharia de Produção da UFRJ-Macaé

Essa parte da coleta de dados teve inicio com a delimitação de quais dados do corpo docente são necessários para que uma grade horária seja criada. De cada professor é necessário saber quais horários ele estará disponível para ministrar alguma disciplina no próximo semestre, além de saber quais disciplinas ele é apto a ministrar.

Para coletar esses dados foi usada a ferramenta *Google Forms*, um aplicativo de gerenciamento de pesquisas da Google. O *Google Forms* foi escolhido pela experiência do autor com o mesmo, além do aplicativo exportar as informações coletadas em forma de tabela

automaticamente e em tempo real. A pesquisa se encontra no **Anexo A**, assim como as respostas (Anexo B).

A pesquisa tinha 3 perguntas: a primeira pergunta era “Qual o seu nome?”, para eu saber de qual professor era cada resposta na hora de gerar a grade horária. A segunda pergunta era “Quais matérias você se sente confortável em lecionar?”, com a possibilidade do professor selecionar várias opções (todas as que ele se sente apto a lecionar). A terceira e última pergunta era “Quais horários você está disponível para ministrar aulas?”, também com a possibilidade de o professor selecionar várias opções.

Tendo todas disciplinas e todos os horários de cada professor é possível testar o modelo computacional e validar a sua viabilidade.

4.2 Aplicação matemática

O problema deste trabalho foi modelado tendo como objetivo principal a maximização do número de disciplinas ofertadas no período, dada certas restrições. Isso pode ser descrito matematicamente como a maximização de uma função objetivo. No caso, a função objetivo escolhida é uma função do tipo somatório que conta o número de alocações bem sucedidas. Dessa forma, temos uma variável binária para cada combinação possível de professor, matéria e horário, que assume valor 1 quando a alocação é válida e 0 no caso contrário, em função das restrições opostas. Os parâmetros da modelagem são os seguintes:

p_{total} → Quantidade total de professores para serem alocados no período;

m_{total} → Quantidade total de matérias para serem alocadas no período;

h_{total} → Quantidade total de horários; $h_{total} = 30$

P → Conjunto dos professores; $P = \{1,\ 2,\ 3,\ \ldots, p_{total}\}$

M → Conjunto das matérias; $M = \{1,\ 2,\ 3,\ \ldots, m_{total}\}$

H → Conjunto dos horários; $H = \{1,\ 2,\ 3,\ \ldots, h_{total}\}$

S → Conjunto dos períodos; $s = \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\}$

M_p → Conjunto de matérias m que o p-ésimo professor é habilitado a ministrar; $M_p = \{m \mid m \in M \wedge p \in P\}$

M_s → Conjunto de matérias m do s-ésimo período; $M_s = \{m \mid m \in M \wedge s \in S\}$

H_p → Conjunto de horários h que o p-ésimo professor pode ser alocado; $H_p = \{h \mid h \in H \wedge p \in P\}$

$Q_m \rightarrow$ Quantidade de aulas por semana que a matéria m possui; $Q_m = \{1, 2\}$

$Q_p \rightarrow$ Quantidade máxima de aulas a serem ministradas pelo professor p em uma semana; $Q_p = \{4, 6\}$

$s_m \rightarrow$ Período da matéria m; $s_m \in S$

$v_h \rightarrow$ Valor de um horário h, para que possamos maximizar as matérias ofertadas no período da noite. $V_h = \{-1, 1 \mid$ se h começa antes de 15h29, então $v = -1$; se h começa das 15h30 em diante, então $v = 1\}$

$i_h \rightarrow$ Código da hora inicial de um horário. Por exemplo, as aulas que começam às 08h00 (Seg1, Ter1, Qua1, Qui1 e Sex1) têm o código 1. $I_h = \{1, 2, 3, 4, 5, 6\}$.

Antes de definir a função objetivo do nosso problema, é necessário definir a variável de binária de controle:

$x_{p,h,m} = 1$, se o professor p for alocado para lecionar a matéria m no horário h. Caso contrário, 0;

Logo, para se obter uma grade horária ótima, temos que maximizar a alocação de matérias nos horários "corretos" (no período após as 15h30).

$$F_{obj} = Maximizar \sum_{p=1}^{p_{total}} \sum_{m=1}^{m_{total}} \sum_{h=1}^{h_{total}} x_{p,h,m} v_h \quad (1)$$

Agora, é necessário modelar as restrições que devem ser respeitadas em uma grade horária.

$$\sum_{m=1}^{m_{total}} x_{p,h,m} \leq 1 \; \forall p \in P, \; h \in H \quad (2)$$

A equação em (2) garante que cada professor só possa ser alocado uma vez por horário. Ou seja, dado um mesmo professor e um mesmo horário, não pode ter mais de 1 matéria onde o $x_{p,h,m} = 1$.

$$\sum_{h=1}^{h_{total}} \sum_{m=1}^{m_{total}} x_{p,h,m} \leq Q_p \; \forall p \in P \quad (3)$$

A equação acima representa matematicamente que cada professor só pode lecionar no máximo a quantidade máxima de horas de aula por semana dele.

$$\sum_{p=1}^{p_{total}} \sum_{h=1}^{h_{total}} x_{p,h,m} = Q_m \; \forall m \in M \quad (4)$$

Em (4) é garantido que todas as matérias alocadas são alocadas a quantidade necessária de vezes. Se essa restrição não fosse implementada, poderíamos ter, por exemplo, uma matéria com 2 aulas por semana sendo alocada só 1 vez.

$$\sum_{p=1}^{p_{total}} \sum_{m=1}^{m_{total}} x_{p,h,m} \leq 1 \; \forall h \in H, s \in S, s_m = s \; (5)$$

Outra restrição que toda grade horária do curso de Engenharia de Produção na UFRJ-Macaé tem que cumprir é que não pode ter mais de 1 matéria do mesmo período no mesmo horário, e é exatamente isso que a equação (5) garante. Para cada períodos s pertencentes ao conjunto S, para cada horário h pertencentes ao conjunto H, o somatório de todos os $x_{p,h,m}$ para todos os professores p e para todas as matérias m não pode ser maior que 1, se o período de m (s_m) for o mesmo que o período s em questão.

$$\sum_{h'=1}^{h_{total}} \sum_{p' \neq p} x_{p',h',m} = 0 \; se \; x_{p,h,m} = 1 \; \forall m \in M, p \in P, h \in H \; (6)$$

A equação (6) garante que toda matéria só irá ser lecionada por um único professor. Caso ela não fosse adicionada, poderíamos ter o professor X alocado para lecionar a matéria 1 na terça e o professor Y alocado para lecionar a mesma matéria 1 na quinta. Para fazer isso, é preciso que, dado uma mesma matéria, se ela já tiver sido alocada, a soma de $x_{p,h,m}$ seja igual a zero para todo os professores p' diferentes do professor p.

$$\sum_{h=1}^{h_{total}} x_{p,h,m} = 0 \quad se \; m \notin M_p \quad \forall p \in P, \; m \in M \; (7)$$

Para não alocar professores para lecionar matérias que não estão aptos a lecionar, incluímos a equação (7) as restrições. Se a matéria m não está no conjunto M_p do professor p, a soma de todos os $x_{p,h,m}$ tem que ser 0 para esse mesmo professor nessa mesma matéria.

$$\sum_{h=1}^{h_{total}} x_{p,h,m} = 0 \quad se \; (x_{p,h',m} = 1 \; e \; i_h \neq i_{h'}) \quad \forall p \in P, \; m \in M, \; h' \in H \; (8)$$

Em (8) é garantido que, aulas com mais de 2 aulas por semana serão alocadas sempre em horários que começam na mesma hora. Matematicamente temos que a soma de $x_{p,h,m}$ para todos os horários h deve ser igual a zero, se essa matéria m já foi alocada em algum horário h' para algum professor p e se esse horário h não começa na mesma hora que o horário h', para todo professor p pertencente ao conjunto P, para toda matéria m pertencente ao conjunto M e para todo horário h' pertencente ao conjunto H.

A partir disso, o problema está modelado matematicamente e pode-se iniciar a modelagem computacional do problema. É importante lembrar que encontrar uma solução viável ou ótima para essas restrições pode levar muito tempo, dependendo do tamanho do conjunto de variáveis e restrições. Por isso, será utilizado uma abordagem computacional para encontrar a solução de forma mais rápida e eficiente.

4.3 Aplicação computacional

O modelo matemático descrito no capítulo 4.2 foi implementado em Python, na sua versão 3.9.7. A IDE utilizada foi o VSCode, devido a familiaridade do autor com o mesmo. Além disso, foi utilizada uma biblioteca para facilitar a implementação da programação por restrições em Python, o OR-Tools, do Google. Mais especificamente o módulo CP-SAT (CP da sigla em inglês de programação de restrições, e SAT de satisfação). No fluxograma (figura x) estão descritas as etapas do script em Python.

Figura 3 - Fluxograma do script em Python

Fonte: Elaborado pelo autor

Da segunda etapa em diante, foi usado a biblioteca OR-Tools. A primeira etapa, no entanto, tem que ser decidida como será feita. Existem várias formas de importar os dados coletados em Python. Uma das formas mais simples é salvar os dados em um arquivo de texto e, em seguida, ler o arquivo usando a função "open" e armazenar os dados em uma variável. Outra forma comum de importar os dados é através de arquivos em formatos específicos, como por exemplo o formato CSV (Comma Separated Values), que é um formato de arquivo que armazena dados separados por vírgulas.

O python possui a biblioteca "csv" que possui funções para facilitar a leitura de arquivos nesse formato. Também é possível importar os dados a partir de bases de dados, usando bibliotecas como o "psycopg2" para conectar-se a um banco de dados PostgreSQL ou o "pymysql" para conectar-se a um banco de dados MySQL. Outras formas de importar os dados incluem acessar dados através de APIs (Application Programming Interfaces) de websites ou consumir dados a partir de arquivos em formato JSON (JavaScript Object Notation).

O formato escolhido para a importação das informações de professores, matérias e horários foi em um arquivo Excel. Os motivos são:

1. Facilidade de uso: A grande maioria dos coordenadores são familiarizados com o Excel e sabem como trabalhar com ele, o que torna mais fácil coletar os dados e salvá-los em um arquivo Excel no futuro, para rodar o programa nos próximos períodos letivos;
2. Compatibilidade: Muitos programas diferentes podem ler e escrever em arquivos Excel, o que significa que existe a possibilidade de compartilhar os dados com outras sem ter que se preocupar com a compatibilidade, pensando em pesquisas futuras;
3. Análise avançada: O Excel possui uma série de funções e ferramentas de análise avançadas que podem ser úteis para analisar os dados gerados em outras possíveis pesquisas.
4. Visualização de dados: O Excel permite criar gráficos e tabelas para visualizar e apresentar os seus dados de maneira mais clara e intuitiva. Além disso, o Excel já era usado para apresentar a grade horária nos períodos anteriores (**Figura 4**)

Para trabalhar com arquivos Excel em Python foi utilizada a biblioteca openpyxl, que permite a leitura e escrita de arquivos Excel. Um Workbook é um arquivo Excel completo, enquanto que um Worksheet é uma aba específica dentro de um arquivo Excel.

Figura 4 -Grade horária do curso de Engenharia de Produção do semestre 2017.2

Fonte: Elaborado pelo autor

4.3.1 Importar informações coletadas (professores, horários e matérias)

Para importar as informações coletadas dentro do código, foram criadas funções separadas para cada importação, e separadas essas funções em arquivos diferentes. Isso faz com que o código fique mais legível e organizado, o que é especialmente útil quando se trata de um trabalho complexo ou quando o código será visto por outras pessoas. Isso também pode ajudar a evitar conflitos de nomes de variáveis e funções, pois cada arquivo pode ter suas próprias variáveis e funções internas. Todas as informações foram importadas da mesma planilha, que é nomeada como "*Dados - TCC.xlsx*".

A primeira importação foi da lista de professores do período. Essa informação se encontra na aba "*Professores*", que está organizada conforme **Figura 5**:

Figura 5 -Aba "Professores" da planilha "Dados - TCC

Nome	Matérias	Horários	Máximo de aulas por semana
Professor 1	Simulação, Modelagem Estatística Aplicada a Engenharia, Princípios da Economia, Planejamento e Controle da Produção, Pesquisa Operacional I, Pesquisa Operacional II, Engenharia Econômica, Projeto de Métodos Quantitativos, Projeto de Gestão Econômica, Metodologia de Pesquisa na Engenharia de Produção	Qua2, Qua3, Qua4, Qua5, Qua6, Qui2, Qui3, Qui4, Qui5, Qui6, Sex2, Sex3, Sex4, Sex5, Sex6	6
Professor 2	Gestão da Inovação, Contabilidade Gerencial e Custos, Gestão Ambiental, Empreendedorismo	Seg3, Seg4, Ter3, Ter4, Qua3, Qua4, Qui3, Qui4, Sex3, Sex4	6
Professor 3	Planejamento das Unidades Produtivas, Gestão da Qualidade e Produtividade, Gestão de Projetos, Gestão da Manutenção, Logística e Gestão da Cadeia de Suprimentos, Gestão Ambiental, Metodologia de Pesquisa na Engenharia de Produção	Seg4, Seg5, Seg6, Ter1, Ter2, Ter3, Ter4, Ter5, Ter6, Qua1, Qua2, Qua3, Qua4, Qua5, Qua6	6
Professor 4	Estudo de Tempos e Métodos, Estratégia da Produção, Princípios da Economia, Gestão da Inovação, Engenharia do Produto I, Empreendedorismo, Projeto de Engenharia do Produto, Projeto de Engenharia Organizacional, Metodologia de Pesquisa na Engenharia de Produção	Seg3, Seg4, Seg5, Seg6, Ter3, Ter4, Ter5, Ter6, Qua3, Qua4, Qua5, Qua6, Qui3, Qui4, Qui5, Qui6	6
Professor 5	Planejamento e Controle da Produção, Engenharia do Produto I, Planejamento e Controle da Produção II, Logística e Gestão da Cadeia de Suprimentos, Projeto de Operações e Processos de Produção, Projeto de Engenharia do Produto	Ter1, Ter2, Ter3, Ter4, Qua1, Qua2, Qui1, Qui2, Qui3, Qui4, Sex1, Sex2, Sex3, Sex4	6
Professor 6	Planejamento das Unidades Produtivas, Teoria das Organizações, Estratégia da Produção, Gestão da Qualidade e Produtividade, Engenharia do Produto I, Logística e Gestão da Cadeia de Suprimentos, Empreendedorismo	Ter3, Ter4, Qua3, Qua4	6
Professor 7	Organização e Avaliação do Trabalho, Gestão de Projetos, Psicologia e Sociologia do Trabalho, Engenharia do Trabalho, Projeto de Engenharia do Trabalho, Metodologia de Pesquisa na Engenharia de Produção	Seg4, Seg5, Ter4, Ter5	6
Professor 8	Estudo de Tempos e Métodos, Teoria das Organizações, Estratégia da Produção, Organização e Avaliação do Trabalho, Gestão da Qualidade e Produtividade, Gestão de Projetos, Psicologia e Sociologia do Trabalho, Engenharia do Trabalho, Projeto de Engenharia Organizacional, Projeto de Engenharia do Trabalho	Seg2, Seg3, Seg4, Seg5, Ter2, Ter3, Ter4, Ter5, Qua3, Qua4, Qua5	6
Professor 9	Fundamentos da Engenharia do Petróleo	Seg5, Qua5	6
Professor 10	Estudo de Tempos e Métodos, Planejamento das Unidades Produtivas, Teoria das Organizações, Estratégia da Produção, Gestão da Qualidade e Produtividade, Gestão de Projetos, Gestão da Inovação, Controle da Qualidade, Engenharia do Produto I, Empreendedorismo, Projeto de Qualidade, Projeto de Operações e Processos de Produção, Projeto de Engenharia do Produto, Projeto de Engenharia Organizacional, Metodologia de Pesquisa na Engenharia de Produção	Qua4, Qua5, Qua6, Qui4, Qui5, Qui6, Sex3, Sex4, Sex5	6
Professor 11	Teoria das Organizações, Organização e Avaliação do Trabalho, Planejamento e Controle da Produção, Psicologia e Sociologia do Trabalho, Engenharia do Trabalho	Ter3, Ter4, Qua1, Qua2, Qua3, Qua4, Qui1, Qui2, Qui3, Qui4	6
Professor 12	Fundamentos da Termodinâmica, Princípios da Economia	Seg3, Seg4, Ter3, Ter4, Qua3, Qua4, Qui3, Qui4	6
Professor 13	Fenômenos Difusivos, Fundamentos da Termodinâmica	Seg2, Seg3, Ter2, Ter3, Qua2, Qua3	6
Professor 14	Estrutura e Propriedade dos Materiais	Qui5, Qui6, Sex1, Sex3, Sex4	6
Professor 15	Fundamentos da Termodinâmica	Seg2, Ter2, Qua2, Qua4, Sex2, Sex3, Sex4	6
Professor 16	Estrutura e Propriedade dos Materiais	Seg4, Seg5, Seg6, Ter4, Ter5, Ter6, Qua4, Qua5, Qua6	6
Professor 17	Modelagem Estatística Aplicada a Engenharia, Princípios da Economia, Pesquisa Operacional I, Logística e Gestão da Cadeia de Suprimentos, Gestão Ambiental, Engenharia Econômica, Projeto de Métodos Quantitativos, Projeto de Gestão Econômica	Seg4, Seg5, Seg6, Ter2, Ter3, Ter4, Ter5	6

Fonte: Elaborado pelo Autor

O arquivo Python "*importar_professores.py*" possui o código para importar os professores da planilha. Importante lembrar que a importação das informações é feita de forma dinâmica de uma planilha de Excel para que os coordenadores não tenham que editar o código Python e somente editar os professores direto na planilha de Excel. O código da importação pode ser visualizado na **Figura 6**:

Figura 6 -Código do arquivo "importar_professores.py"

```
import openpyxl
from achar_ultima_celula import achar_ultima_celula


def importar_professores():
    professores = []
    materias_por_professor = {}
    horarios_por_professor = {}
    max_aulas_semana_por_professor = {}
    wb = openpyxl.load_workbook('Dados - TCC.xlsx')
    ws = wb['Professores']
    ultima_linha = achar_ultima_celula(ws, direcao="coluna")['linha']
    linha_atual = 2
    while linha_atual < ultima_linha:
        professores.append(ws.cell(row=linha_atual, column=1).value)
        materias_por_professor[ws.cell(
            row=linha_atual, column=1).value] = ws.cell(
            row=linha_atual, column=2).value.split(', ')
        horarios_por_professor[ws.cell(
            row=linha_atual, column=1).value] = ws.cell(
            row=linha_atual, column=3).value.split(', ')
        max_aulas_semana_por_professor[ws.cell(
            row=linha_atual, column=1).value] = ws.cell(
            row=linha_atual, column=4).value
        linha_atual += 1
    return professores, materias_por_professor, horarios_por_professor, \
        max_aulas_semana_por_professor
```

Fonte: Elaborado pelo Autor

A função começou importando a biblioteca *openpyxl* e a função "*achar_ultima_celula*" de um arquivo separado. Em seguida, a função criou uma lista vazia chamada "professores", que será preenchida com os nomes dos professores, um dicionário vazio chamado "*materias_por_professor*", que armazenou as matérias que cada professor está apto a lecionar, outro dicionário vazio chamado "*horarios_por_professor*", que irá conter os horários que cada professor está disponível para ser alocado e outro dicionário vazio chamado "*max_aulas_semana_por_professor*", que vai conter a quantidade máxima de aulas por semana de cada professor. No código, a variável "*wb*" é inicializada como o Workbook

"Dados - TCC.xlsx" e a variável "*ws*" é inicializada como o Worksheet "Professores" do Workbook "*wb*", onde estão as informações dos professores

A função "*achar_ultima_celula*" é chamada para encontrar a última célula preenchida da coluna A do Worksheet "ws". O resultado é armazenado na variável "*ultima_linha*", que representa a última linha preenchida da coluna A. A variável "*linha_atual*" é inicializada como 2, já que a primeira linha (linha 1) do Worksheet é a linha de título.

A lógica aqui é passar por cada linha, começando na linha 2 (a primeira linha contém o cabeçalho da tabela), indo até a linha do último professor (salvo na variável "*ultima_linha*"), e ir adicionando esse professor a lista de professores, além das matérias que ele é apto a lecionar, os horários que ele está disponível e a quantidade de aulas que ele pode ser alocado em uma semana, nos dicionários "*materias_por_professor*", "*horarios_por_professor*" e "*max_aulas_semana_por_professor*" respectivamente.

A função "*achar_ultima_celula*" foi definida em um arquivo separado, chamado "*achar_ultima_celula.py*", pois ela será reutilizada nas outras funções de importação. O código do arquivo está disponível na **Figura 7**:

Figura 7 - Código do arquivo "achar_ultima_celula.py"

```python
def achar_ultima_celula(
    ws,
    direcao="linha",
    primeira_linha=1,
    primeira_coluna=1
):
    linha_atual = primeira_linha
    coluna_atual = primeira_coluna
    if direcao == "linha":
        while ws.cell(row=linha_atual, column=coluna_atual).value is not None:
            coluna_atual += 1
    elif direcao == "coluna":
        while ws.cell(row=linha_atual, column=coluna_atual).value is not None:
            linha_atual += 1

    return {'linha': linha_atual, 'coluna': coluna_atual}

```

Fonte: Elaborado pelo Autor

Essa função tem como objetivo encontrar a última célula ocupada de uma *worksheet*. Ela recebe como parâmetros a worksheet (*ws*) e a direção em que se deseja procurar a última célula preenchida, que pode ser "linha" ou "coluna". Além disso, é possível especificar os valores iniciais da linha (*primeira_linha*) e da coluna (*primeira_coluna*), que são 1 por padrão.

A função começa a iterar a partir da célula especificada pelos parâmetros *primeira_linha* e *primeira_coluna*, e vai avançando a linha ou coluna, dependendo da direção escolhida, até encontrar uma célula vazia. Quando isso acontece, a função retorna um dicionário contendo a linha e coluna da última célula preenchida.

A próxima informação que foi importada são os horários, que estão disponíveis na aba "Horarios" da planilha, conforme apresentado na **Figura 8**:

Figura 8 - Aba "Horarios" da planilha "Dados - TCC"

Horário	Código	Valor
Seg1	1	-1
Seg2	2	-1
Seg3	3	-1
Seg4	4	1
Seg5	5	1
Seg6	6	1
Ter1	1	-1
Ter2	2	-1
Ter3	3	-1
Ter4	4	1
Ter5	5	1
Ter6	6	1
Qua1	1	-1
Qua2	2	-1
Qua3	3	-1
Qua4	4	1
Qua5	5	1
Qua6	6	1
Qui1	1	-1
Qui2	2	-1
Qui3	3	-1
Qui4	4	1
Qui5	5	1
Qui6	6	1
Sex1	1	-1
Sex2	2	-1
Sex3	3	-1
Sex4	4	1
Sex5	5	1
Sex6	6	1

Fonte: Elaborado pelo Autor

A função para importar os horários está definida no arquivo "*importar_horarios.py*", que o código está exposto na **Figura 9**:

Figura 9 - Código do arquivo "importar_horarios.py"

```python
import openpyxl
from achar_ultima_celula import achar_ultima_celula


def importar_horarios():
    horarios = []
    codigo_por_horario = {}
    valor_por_horario = {}
    wb = openpyxl.load_workbook('Dados - TCC.xlsx')
    ws = wb['Horarios']
    ultima_linha = achar_ultima_celula(ws, direcao='coluna')['linha']
    linha_atual = 2
    while linha_atual < ultima_linha:
        horarios.append(ws.cell(row=linha_atual, column=1).value)
        codigo_por_horario[ws.cell(row=linha_atual, column=1).value] = ws.cell(
            row=linha_atual, column=2).value
        valor_por_horario[ws.cell(row=linha_atual, column=1).value] = ws.cell(
            row=linha_atual, column=3).value
        linha_atual += 1
    return horarios, codigo_por_horario, valor_por_horario

```

Fonte: Elaborado pelo Autor

Primeiramente, é importado o módulo openpyxl, e a função "*achar_ultima_celula*" do módulo "*achar_ultima_celula*", definido acima no arquivo "*achar_ultima_celula.py*". Depois, a função cria uma lista vazia de horários (variável "*horarios*"), um dicionário que salva o código de cada horário (variável "*codigo_por_horario*"), um dicionários que vai salvar o valor de cada horário (variável "*valor_por_horario*") e carregar o arquivo de Excel "Dados - TCC.xlsx" e a worksheet "Horários". A variável "*ultima_linha*" corresponde à linha preenchida da *worksheet*. Em seguida, a função cria uma variável "*linha_atual*", iniciada em 2, e entra em um loop while. Enquanto "*linha_atual*" for menor que a última linha preenchida, a função adiciona o horário na lista "*horarios*", o código do horário no dicionário "*codigo_por_horario*", o valor do horário no dicionário "*valor_por_horario*" e incrementa "*linha_atual*" em 1. Quando o *loop* termina, a função retorna a lista de todos os horários e o dicionário com o código de cada horário.

Em seguida, foram importadas as matérias do curso de Engenharia de Produção da UFRJ-Macaé. Elas estão disponíveis na aba "Materias" da planilha, conforme **Figura 10**.

Figura 10 - Aba "Materias" da planilha "Dados - TCC"

Nome	Período	Aulas por semana
Fenômenos Difusivos	5	2
Organização e Avaliação do Trabalho	6	2
Gestão da Inovação	7	2
Engenharia do Trabalho	8	2
Projeto de Qualidade	9	1
Fundamentos da Termodinâmica	5	2
Gestão da Qualidade e Produtividade	6	2
Psicologia e Sociologia do Trabalho	7	2
Gestão da Manutenção	8	2
Projeto de Métodos Quantitativos	9	1
Estrutura e Propriedade dos Materiais	5	2
Simulação	6	2
Controle da Qualidade	7	2
Pesquisa Operacional II	8	2
Projeto de Operações e Processos de Produção	9	1
Estudo de Tempos e Métodos	5	1
Modelagem Estatística Aplicada a Engenharia	6	2
Engenharia do Produto I	7	2
Logística e Gestão da Cadeia de Suprimentos	8	2
Projeto de Gestão Econômica	9	1
Planejamento das Unidades Produtivas	5	1
Gestão de Projetos	6	2
Pesquisa Operacional I	7	2
Gestão Ambiental	8	2
Projeto de Engenharia do Produto	9	1
Teoria das Organizações	5	2
Principios da Economia	6	2
Planejamento e Controle da Produção II	7	2
Engenharia Econômica	8	2
Projeto de Engenharia Organizacional	9	1
Estratégia da Produção	5	2
Planejamento e Controle da Produção	6	2
Contabilidade Gerencial e Custos	7	2
Empreendedorismo	8	2
Projeto de Engenharia do Trabalho	9	1
Fundamentos da Engenharia do Petróleo	6	2
Metodologia de Pesquisa na Engenharia de Produção	9	2

Fonte: Elaborado pelo Autor

A função para importar as matérias, assim como o período e a quantidade de aulas por semana de cada matéria está definida no arquivo "*importar_materias.py*", disponível na **Figura 11**:

Figura 11 - Código do arquivo "importar_materias.py"

```python
import openpyxl
from achar_ultima_celula import achar_ultima_celula


def importar_materias():
    materias = []
    qtdd_aulas_semanal_por_materia = {}
    periodos_por_materia = {}
    wb = openpyxl.load_workbook('Dados - TCC.xlsx')
    ws = wb['Materias']
    ultima_linha = achar_ultima_celula(ws, direcao='coluna')['linha']
    linha_atual = 2
    while linha_atual < ultima_linha:
        materias.append(ws.cell(row=linha_atual, column=1).value)
        periodos_por_materia[ws.cell(
          row=linha_atual, column=1).value] = ws.cell(
          row=linha_atual, column=2).value
        qtdd_aulas_semanal_por_materia[ws.cell(
          row=linha_atual, column=1).value] = ws.cell(
          row=linha_atual, column=3).value
        linha_atual += 1
    return materias, periodos_por_materia, qtdd_aulas_semanal_por_materia

```

Fonte: Elaborado pelo Autor

Para fazer a importação, é necessário abrir o arquivo "Dados - TCC.xlsx" e importar a aba "Materias", na variável "*ws*". Em seguida, encontrar a última linha da planilha utilizando a função "*achar_ultima_celula*" e armazenar o resultado na variável "*ultima_linha*". Depois, criar uma lista vazia: "*materias*" e 2 dicionários vazios: "*qtdd_aulas_semanal_por_materia*" e "*periodos_por_materia*". A lista "*materias*" armazena o nome de cada matéria, "*qtdd_aulas_semanal_por_materia*" armazena a quantidade de aulas semanais de cada matéria e "*periodos_por_materia*" armazena o período de cada matéria. Depois, ela começa a percorrer a planilha "Materias" da linha 2 até a última linha e, em cada iteração, ela adiciona o nome da matéria na lista "*materias*" e armazena o período e a quantidade de aulas semanais da matéria em "*periodos_por_materia*" e "*qtdd_aulas_semanal_por_materia*", respectivamente. Por fim, ela retorna a lista e os 2 dicionários criados.

Com todas as informações para gerar a grade horária, falta construir o modelo, adicionar todas as variáveis, restrições e função objetivo para, depois, rodá-lo.

4.3.2 Criação do modelo matemático em Python

A função que cria e roda o modelo ficará dentro do arquivo chamado "*modelo.py*", cujo código pode ser visualizado na **Figura 12**. Primeiramente, é necessário importar algumas funções e módulos que serão usados no decorrer da implementação. Depois, são

usadas as funções descritas no capítulo acima para importar todas as informações de professores, matérias e horários dentro desse arquivo.

Figura 12 - Código do arquivo "modelo.py"

```
from ortools.sat.python import cp_model
from importar_professores import importar_professores
from importar_horarios import importar_horarios
from importar_materias import importar_materias

importacao_professores = importar_professores()
importacao_horarios = importar_horarios()
importacao_materias = importar_materias()

professores = importacao_professores[0]
materias_por_professor = importacao_professores[1]
horarios_por_professor = importacao_professores[2]
max_aulas_semana_por_professor = importacao_professores[3]
horarios = importacao_horarios[0]
codigo_por_horario = importacao_horarios[1]
valor_por_horario = importacao_horarios[2]
todas_as_materias = importacao_materias[0]
periodos_por_materia = importacao_materias[1]
qtdd_aulas_semanal_por_materia = importacao_materias[2]
materias_por_periodo = importacao_materias[3]

lista_de_periodos = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Fonte: Elaborado pelo Autor

A partir de então, se fez necessário elaborar a função que vai criar e rodar o modelo usando a classe "CpModel", importada de dentro da biblioteca OR-Tools conforme **Figura 13**.

Figura 13 - Código do arquivo "modelo.py"

```
def modelo():
    # Definição do modelo
    model = cp_model.CpModel()

    # Criação das variáveis
    alocacao_horarios = {}
    for p in professores:
        for m in materias:
            for h in horarios:
                # alocacao_horarios[(p, h, m))] = 1 se:
                # o professor p for alocado na matéria m no horário h
                # e 0 caso contrário
                alocacao_horarios[(p, h, m)] = model.NewBoolVar(f'{p}_{h}_{m}')
```

Fonte: Elaborado pelo Autor

Dentro da função "*modelo()*", é usada a classe "CpModel" para instanciar, ou seja, criar uma instância do modelo dentro da variável "*model*". Em seguida, é criada uma variável chamada "*alocacao_horarios*", que representa as alocações de professores em horários específicos para ministrar aulas. Esta variável é inicializada como um dicionário vazio e é preenchida com novas variáveis do tipo booleano criadas a partir do método "*NewBoolVar*" da classe "CpModel". Estas variáveis representam se um professor será ou não alocado para ministrar uma aula em um determinado horário. A chave do dicionário é uma tupla composta pelo nome do professor, nome da matéria e horário, enquanto o valor é 1 ou 0, dependendo se esse professor foi alocado nesse horário para ministrar essa matéria. O próximo passo é adicionar as restrições a esse modelo.

Figura 14 - Código do arquivo "modelo.py"

```
    # Restrições
    # cada professor só pode ser alocado 1 vez por horário
    for p in professores:
        for h in horarios:
            model.Add(sum(alocacao_horarios[(p, h, m)] for m in materias) <= 1)
```

Fonte: Elaborado pelo Autor

A adição de restrições é feita através da função "*Add()*" da classe "CpModel", que adiciona uma nova restrição ao modelo de programação linear. Essa primeira restrição estabelece que cada professor só pode ser alocado em uma matéria por horário, que é dada pelo somatório das variáveis "*alocacao_horarios*". O somatório é feito para todas as matérias e a restrição é que esse somatório deve ser menor ou igual a 1. Dessa forma, a restrição garante que cada professor só pode ser alocado em uma matéria por horário. A **Figura 14** apresenta o código

Figura 15 - Código do arquivo "modelo.py"

```
# cada professor só poder lecionar no máximo
# a quantidade máxima de aulas dele(a) por semana
for p in professores:
    model.Add(sum(alocacao_horarios[
        (p, h, m)] for m in materias for h in horarios) <=
            max_aulas_semana_por_professor[p])
```

Fonte: Elaborado pelo Autor

Essa restrição garante que cada professor só pode lecionar no máximo a quantidade máxima de aulas dele(a) por semana. No caso dos professores do período em que o presente trabalho aconteceu, todos podiam ser alocados em até 6 aulas por semana, pois todos os professores eram titulares. Na UFRJ-Macaé os professores substitutos só podem lecionar 4 aulas por semana. Ela faz isso somando, para cada professor "*p*", a quantidade de aulas que ele é alocado para ministrar (*alocacao_horarios[(p, h, m)]*), para todas as matérias "*m*" e horários "*h*", e garante que esse somatório não seja maior que o valor máximo de aulas por semana do professor "*p*" (*max_aulas_semana_por_professor[p]*).

Figura 16 - Código do arquivo "modelo.py"

```
# não pode ter mais de 1 matéria do mesmo período no mesmo horário
for h in horarios:
    for p in lista_de_periodos:
        model.Add(sum(alocacao_horarios[
            (p, h, m)] for p in professores
            for m in materias if periodos_por_materia[m] == p) <= 1)
```

Fonte: Elaborado pelo Autor

Na **Figura 16**, é mostrado o código que implementa no modelo a restrição de não poder ter mais de uma matéria do mesmo período no mesmo horário. Para isso, é feita a soma de todas as alocações de horários para cada professor "*p*" e matéria "*m*", onde o período da matéria "*m*" é igual ao período "*p*", e essa soma não pode ser maior do que 1.

Figura 17 - Código do arquivo "modelo.py"

```
# toda matéria só pode ser lecionada por um único professor
for m in materias:
    for professor in professores:
        for horario in horarios:
            model.Add(sum(alocacao_horarios[(
                p, h, m)] for h in horarios for p in professores
                if p != professor) == 0).OnlyEnforceIf(
                    alocacao_horarios[(professor, horario, m)])
```

Fonte: Elaborado pelo Autor

A restrição adicionada agora diz que cada matéria só pode ser lecionada por um único professor, conforme **Figura 17**. Para isso, o código percorre cada matéria, professor e horário e verifica se o professor em questão não é o professor que está sendo avaliado. Se esse for o caso, a restrição é adicionada ao modelo. Além disso, há uma condição adicional na restrição, chamada "OnlyEnforceIf", que diz que essa restrição só deve ser aplicada se a variável "*alocacao_horarios[(professor, horario, m)]*" for verdadeira. Isso significa que essa restrição só será aplicada se o professor em questão estiver realmente alocado na matéria "*m*" no horário "*h*".

Figura 18 - Código do arquivo "modelo.py"

```
# os professores só podem lecionar as materias que estão aptos a lecionar:
for p in professores:
    for m in materias:
        if m not in materias_por_professor[p]:
            model.Add(sum(alocacao_horarios[(
                p, h, m)] for h in horarios) == 0)
```

Fonte: Elaborado pelo Autor

Nesta etapa é adicionada a restrição que garante que o professor só pode ser alocado em uma matéria se ele estiver apto a lecioná-la conforme **Figura 18**. Para isso, o código itera sobre cada professor e cada matéria. Se a matéria não estiver presente na lista de matérias que o professor é apto a lecionar (*materias_por_professor[p]*), então é adicionada uma restrição que garante que a soma de "*alocacao_horarios[(p, h, m)]*" para todo "*h*" em "*horarios*" é igual a zero. Isso quer dizer que o professor não pode ser alocado na matéria m em nenhum horário h.

Figura 19 - Código do arquivo "modelo.py"

```
# os professores só pode ser alocados nos horários que estão disponiveis:
for p in professores:
    for h in horarios:
        if h not in horarios_por_professor[p]:
            model.Add(sum(alocacao_horarios[(
                p, h, m)] for m in materias) == 0)
```

Fonte: Elaborado pelo Autor

A restrição da **Figura 19** verifica se o professor "*p*" está disponível no horário "*h*". Se não estiver, a soma das variáveis "*alocacao_horarios*" para esse professor e horário deve ser igual a zero, ou seja, ele não pode ser alocado nesse horário.

Figura 20 - Código do arquivo "modelo.py"

```
# todas as aulas de uma mesma matéria tem que ser alocadas em horários
# que comecem na mesma hora
for m in materias:
    for p in professores:
        for horario in horarios:
            model.Add(sum(alocacao_horarios[
                (p, h, m)] for h in horarios if codigo_por_horario[h]
                != codigo_por_horario[horario]) == 0).OnlyEnforceIf(
                    alocacao_horarios[(p, horario, m)])
```

Fonte: Elaborado pelo Autor

Essa última restrição estabelece que todas as aulas de uma mesma matéria devem ser alocadas em horários que começam na mesma hora conforme pode ser visto na **Figura 20**. Para isso, a restrição verifica se o código do horário atual é diferente do código dos demais horários. Se for diferente, a restrição estabelece que a soma das aulas deve ser igual a 0. Se o código for igual, a restrição não é aplicada.

Nota-se, que todas as restrições foram adicionadas ao modelo. A última parte se refere a adicionar a função objetivo conforme a **Figura 21**

Figura 21 - Código do arquivo "modelo.py"

```
# Função Objetivo
# maximizar o valor de alocacao_horarios * valor_por_horario
model.Maximize(sum(alocacao_horarios[(
    p, h, m)] * valor_por_horario[h] for p in professores
    for h in horarios for m in materias))
```

Fonte: Elaborado pelo Autor

O objetivo da função objetivo é maximizar a soma do produto entre as variáveis de alocação de horários e o valor de cada horário. Isso significa que o modelo está tentando alocar os professores nas matérias e horários que geram o maior valor possível. A variável "*valor_por_horario*" retorna valores maiores para aulas que começam a partir das 15h30 e valores menores para aulas que começam antes deste horário. Isso é feito para tentar atender aos alunos que trabalham/ têm estágio pela manhã, permitindo que eles consigam cursar o máximo de disciplinas possível.

Finalmente, atinge-se a etapa de resolver o problema. Para isso, foi utilizado o solver do OR-Tools, que é uma ferramenta que aplica algoritmos de otimização para encontrar a solução viável ou ótima de um problema de programação linear ou inteira. O solver do OR-Tools é capaz de encontrar soluções para problemas com milhões de variáveis e restrições, e pode ser aplicado em diversas áreas, como planejamento de produção, alocação

de recursos e atribuição de tarefas. Para resolver o modelo, faz-se chamar a função "*solver.Solve(model)*", passando como parâmetro o modelo em questão.

Figura 22 - Código do arquivo "modelo.py"

```
    # Resolução do modelo
    print('Resolvendo o modelo ... ')
    solver = cp_model.CpSolver()
    status = solver.Solve(model)
    print('Modelo resolvido!')

    # Imprimindo a solução
    if status == cp_model.OPTIMAL:
        print('Uma solução ótima foi encontrada!')

    elif status == cp_model.FEASIBLE:
        print('Uma solução viável foi encontrada!')

    else:
        print('Nenhuma solução foi encontrada!')
    return status
```

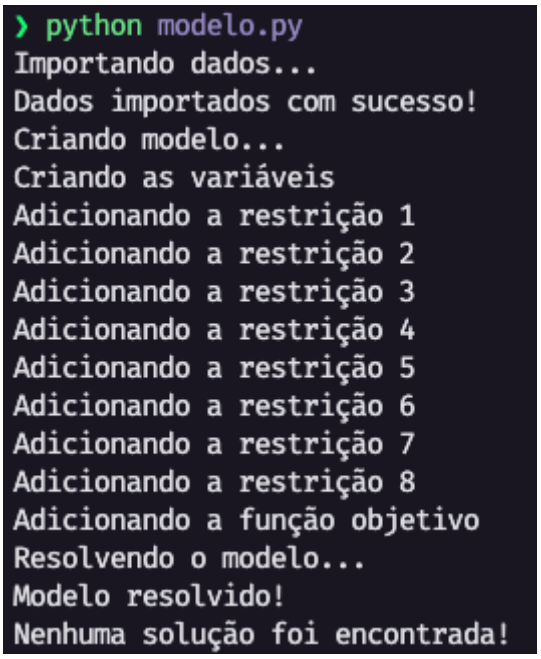
Fonte: Elaborado pelo Autor

O código contido na figura 22 utiliza o solver do OR-Tools para resolver o modelo de programação linear criado. O solver tenta encontrar uma solução viável ou ótima para o modelo, dependendo das restrições e da função objetivo definidas. A variável "status" armazena o resultado da resolução do modelo e pode assumir os seguintes valores:

- "*cp_model.OPTIMAL*": significa que uma solução ótima foi encontrada.
- "*cp_model.FEASIBLE*": significa que uma solução viável foi encontrada, mas não necessariamente ótima.
- "*cp_model.INFEASIBLE*": significa que não foi possível encontrar uma solução viável para o modelo.
- "*cp_model.UNKNOWN*": significa que o solver não conseguiu determinar se existe uma solução viável para o modelo.

Além disso, alguns prints foram adicionados ao código para que quem o executa saiba exatamente em que parte do código a execução está. Ao rodar a função "*modelo()*", nenhuma solução foi encontrada conforme a **Figura 23**:

Figura 23 - Print do terminal ao rodar o programa "modelo.py"



Fonte: Elaborado pelo Autor

Então, o modelo com essas variáveis (professores, horários e matérias) não apresenta uma solução viável. Isso significa que, com essa quantidade de matérias, horários, professores (com essas disponibilidades de horários e com essas matérias que são aptos a lecionar) não existe uma grade horária possível. Existem algumas possibilidades de tornar esse modelo viável:

1. Aumentar a quantidade de professores: pode ser uma opção para resolver o problema, pois isso significa que mais professores estarão disponíveis para lecionar as matérias. No entanto, essa solução pode não ser a mais viável, pois pode haver um limite de professores disponíveis para lecionar e também pode ser necessário investir mais recursos financeiros para contratar novos professores, representando custos adicionais para a instituição de ensino.
2. Aumentar a quantidade de disciplinas que os professores lecionam: pode ajudar a aumentar a capacidade de alocação de aulas, pois haverá mais opções de professores para escolher. No entanto, isso pode requerer que os professores tenham conhecimento e habilidades em disciplinas adicionais, o que pode não ser possível em todas as situações. É preciso considerar se os professores possuem a capacidade de lecionar mais matérias e se isso não compromete a qualidade das aulas.
3. Aumentar os horários disponíveis dos professores: essa opção pode ser viável se os professores disponibilizarem mais horários para ministrar aulas. No

entanto, é preciso levar em consideração que os professores também podem ter outras atividades, como pesquisas, atendimentos a alunos e outras responsabilidades, então talvez não seja possível contar com mais horários disponíveis dos professores.

4. Aumentar a quantidade de horários oferecidos: essa opção pode ser viável se o instituto de ensino conseguir disponibilizar mais horários para ministrar aulas. No entanto, isso pode implicar em mais gastos com locação de espaço e contratação de funcionários para limpeza e segurança. Além disso, pode ser difícil encontrar horários disponíveis em um período de tempo curto, o que pode atrasar o início das aulas.
5. Diminuir a quantidade de matérias oferecidas: essa opção pode ser a mais viável, pois implica em menos demanda por professores e horários. Além disso, pode ser mais fácil de implementar do que as outras opções, pois não envolve gastos adicionais ou mudanças na rotina dos professores. No entanto, é preciso levar em consideração o impacto que a diminuição de matérias pode ter na qualidade do ensino e no atendimento das demandas dos alunos, além de condições impostas por orgãos ligados à Educação como o MEC.

A opção de diminuir a quantidade de matérias oferecidas foi a adotada pelo estudo, pois não envolve custos adicionais à instituição, e esse controle de quais matérias excluir primeiro fica a critério do coordenador. No curso de Engenharia de Produção na UFRJ-Macaé, por exemplo, existem matérias que só são oferecidas em períodos pares, outras que só são oferecidas em períodos ímpares. Logo, essas matérias não teriam tanta prioridade de serem oferecidas em todos os períodos.

Isso pode ser feito passando um parâmetro para a função "*modelo()*" que represente a quantidade de matérias excluídas. Então, a função "*modelo()*" pode ser chamada de forma iterativa, aumentando a quantidade de matérias excluídas a cada iteração. Na primeira vez que a função é chamada, o valor zero é passado como quantidade de matérias excluídas. Se a função não conseguir encontrar uma solução viável, a quantidade de matérias excluídas é aumentada e a função é chamada novamente. Dessa forma, pode-se continuar aumentando a quantidade de matérias excluídas até uma solução viável ser encontrada.

Para facilitar o desenvolvimento, a escrita do código e o controle do coordenador sobre as matérias do período, as matérias serão excluídas começando pela última da lista. Dessa forma, o coordenador pode controlar a prioridade das matérias, bastando posicioná-las na lista do Excel de acordo com a necessidade. A exclusão da última matéria da lista, na

primeira iteração, garante que as matérias mais importantes serão mantidas no planejamento, enquanto que as menos importantes serão excluídas primeiro, caso seja necessário. Isso permite que o modelo consiga encontrar uma solução viável, sem comprometer a qualidade do planejamento.

Para implementar essas atualizações no modelo em Python, será criado mais um arquivo, o "gerar_grade_horaria.py". Ele vai ser responsável por chamar a função "*modelo()*" até encontrar uma solução para o modelo, passando a quantidade de matérias excluídas como parâmetro para função.

Figura 24 - Código do arquivo "gerar_grade_horaria.py"

```
from modelo import modelo
from ortools.sat.python import cp_model

status = cp_model.INFEASIBLE
qtdd_materias_excluidas = 0

while status == cp_model.INFEASIBLE:
    status = modelo(qtdd_materias_excluidas)
    if status == cp_model.INFEASIBLE:
        qtdd_materias_excluidas += 1
    else:
        break
```

Fonte: Elaborado pelo Autor

No código da **Figura 24**, a variável "*status*" é usada para armazenar o retorno da função "*modelo()*". Inicialmente, ela é iniciada com o valor "INFEASIBLE", ou seja, o modelo não tem uma solução viável. Então, é iniciado um *loop* que só será quebrado caso a função "*modelo()*" retorne algum status diferente de "INFEASIBLE", ou seja, uma solução viável ou ótima tenha sido encontrada. A cada iteração do *loop*, a função "*modelo()*" é chamada passando como parâmetro a quantidade de matérias excluídas, que é aumentada em 1 a cada iteração.

Para que esse código funcione, é necessário adaptar a função "*modelo()*" para que ela possa receber o parâmetro "*qtdd_materias_excluidas*". O código está indicado na **Figura 25**:

Figura 25 - -Código do arquivo "modelo.py"

```python
todas_as_materias = importacao_materias[0]
periodos_por_materia = importacao_materias[1]
qtdd_aulas_semanal_por_materia = importacao_materias[2]

lista_de_periodos = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print('Dados importados com sucesso!')


def modelo(qtdd_materias_excluidas):
    if qtdd_materias_excluidas > 0:
        materias_excluidas = todas_as_materias[-qtdd_materias_excluidas:]
        print(f'Materias excluídas: {materias_excluidas}')
        # materias contem todas as materias que não foram excluidas
        materias = todas_as_materias[:-qtdd_materias_excluidas]
    else:
        materias = todas_as_materias

    num_materias = len(materias)
    print(f'Tentando solucionar o modelo para {num_materias} matérias')
```

Fonte: Elaborado pelo Autor

Primeiro, a variável "*materias*" foi renomeado para "*todas_as_materias*". Depois, a função "*modelo()*" foi adaptada para receber o parâmetro "*qtdd_materias_excluidas*". Tendo feito isso, 3 novas variáveis foram criadas: "*materias_excluidas*", que é uma lista com as matérias excluídas, a lista "*materias*" que contém só as matérias que tentarão ser alocadas nessa iteração e a variável "*num_materias*" que contém a quantidade de matérias que pretende-se alocar nessa iteração. Além disso, alguns "*prints*" foram adicionados ao código, para ajudar no acompanhamento da execução do mesmo. Agora, ao rodar o arquivo "*gerar_grade_horaria.py*", tem-se o exposto na **Figura 26**:

Figura 26 - Print do terminal ao rodar o programa "modelo.py"

```
Nenhuma solução encontrada para 27 matérias!
Materias excluídas: ['Principios da Economia', 'Planejamento e Controle da Produção II', 'Engenharia Econômic
a', 'Projeto de Engenharia Organizacional', 'Estratégia da Produção', 'Planejamento e Controle da Produção',
'Contabilidade Gerencial e Custos', 'Empreendedorismo', 'Projeto de Engenharia do Trabalho', 'Fundamentos da
Engenharia do Petróleo', 'Metodologia de Pesquisa na Engenharia de Produção']
Tentando solucionar o modelo para 26 matérias
Criando modelo...
Criando as variáveis
Adicionando a restrição 1
Adicionando a restrição 2
Adicionando a restrição 3
Adicionando a restrição 4
Adicionando a restrição 5
Adicionando a restrição 6
Adicionando a restrição 7
Adicionando a restrição 8
Adicionando a função objetivo
Resolvendo o modelo...
Modelo resolvido!
Solução ótima encontrada para 26 matérias!
```

Fonte: Elaborado pelo Autor

Com as informações coletadas para o período analisado neste trabalho, só é possível encontrar uma solução para 26 matérias, nessa ordem de preferência. Agora, com o modelo resolvido, é necessário resolver mais um desafio na implementação computacional: como mostrar os resultados.

Para acessar os resultados de uma solução encontrada pelo solver do OR-Tools, podem ser utilizados 2 métodos principais dentro da instância:

- "*solver.Value(variable)*": retorna o valor da variável no modelo no momento da solução. Por exemplo, se "*solver.Value(alocacao_horarios[(p, h, m)])*" for 1, significa que o professor *p* foi alocado no horário *h* para ministrar a matéria *m*.
- "*solver.BestObjectiveBound()*": retorna o melhor limite da função objetivo encontrado pelo solver. No caso desse trabalho, esse valor não tem grande importância, pois a pontuação da grade horária não agrega tanto para o coordenador do curso, a grade em si é que agrega.

O primeiro método acima será usado para mostrar a grade horária no mesmo arquivo de importação de dados, o "Dados - TCC.xlsx". Para isso, é necessário criar uma aba dentro da planilha para receber os resultados. A **Figura 27** apresenta a aba em questão:

Figura 27 - Aba "Grade Horária" da planilha "Dados - TCC.xslx"

	Segunda	Terça	Quarta	Quinta	Sexta
08h00 – 10h00					
10h00 – 12h00					
13h30 – 15h30					
15h30 – 17h30					
17h30 – 19h30					
19h30 – 21h30					

Fonte: Elaborado pelo Autor

Para preencher essa aba com as informações do modelo resolvido, foi criado um arquivo que contém a função de exportação. O arquivo foi chamado de: "*exportar_grade_horaria.py*" e seu código está disponível conforme Figura 28:

Figura 28 - Código do arquivo "exportar_grade_horaria.py"

```
import openpyxl
from openpyxl.styles import Alignment

> conversor = {⋯


def exportar_grade_horaria(professores, horarios, materias,
                           periodos_por_materia, alocacao_horarios, solver):
    horarios_com_aulas = {}
    for h in horarios:
        horarios_com_aulas[h] = {}
        for p in professores:
            for m in materias:
                if solver.Value(alocacao_horarios[(p, h, m)]) == 1:
                    horarios_com_aulas[h][p] = m

    wb = openpyxl.load_workbook('Dados - TCC.xlsx')
    ws = wb['Grade Horária']
    for h in horarios_com_aulas:
        horario_completo = ''
        for p in horarios_com_aulas[h]:
            horario_completo += p + " - " + horarios_com_aulas[
                h][p] + ' - ' + str(periodos_por_materia[horarios_com_aulas[
                    h][p]]) + 'º \n'
        ws[conversor[h]].alignment = Alignment(wrapText=True)
        ws[conversor[h]] = horario_completo
    wb.save('Grade Exportada.xlsx')
```

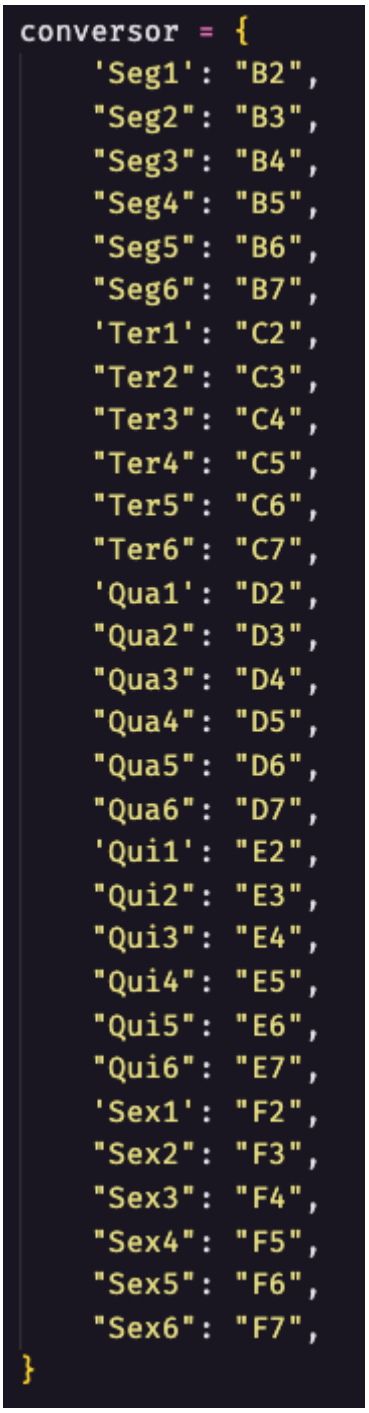
Fonte: Elaborado pelo Autor

A primeira parte da função cria um dicionário chamado "*horarios_com_aulas*", que mapeia cada horário para uma lista de tuplas (professor, disciplina). Essa lista armazena os professores e disciplinas que foram alocados para cada horário, de acordo com os resultados do modelo, que são acessados através da variável "*alocacao_horarios*".

Em seguida, o código carrega o arquivo Excel "Dados - TCC.xlsx" e acessa a sua aba chamada "Grade Horária". O código percorre os horários armazenados no dicionário "*horarios_com_aulas*" e, para cada horário, constrói uma *string* com a informação dos professores e disciplinas alocados para aquele horário. Essa *string* é então escrita na célula correspondente da planilha "Grade Horária" do arquivo Excel. Por fim, o arquivo é salvo com o nome "Grade Exportada.xlsx".

O dicionário "conversor" é usado para mapear os nomes dos horários para os endereços das células correspondentes na planilha "Grade Horária" (**Figura 29**). A propriedade "alignment" é usada para habilitar o "wrap text" na célula, o que permite que o conteúdo da célula seja dividido em várias linhas, caso seja necessário.

Figura 29 - Código do arquivo "exportar_grade_horaria.py"



```
conversor = {
    'Seg1': "B2",
    "Seg2": "B3",
    "Seg3": "B4",
    "Seg4": "B5",
    "Seg5": "B6",
    "Seg6": "B7",
    'Ter1': "C2",
    "Ter2": "C3",
    "Ter3": "C4",
    "Ter4": "C5",
    "Ter5": "C6",
    "Ter6": "C7",
    'Qua1': "D2",
    "Qua2": "D3",
    "Qua3": "D4",
    "Qua4": "D5",
    "Qua5": "D6",
    "Qua6": "D7",
    'Qui1': "E2",
    "Qui2": "E3",
    "Qui3": "E4",
    "Qui4": "E5",
    "Qui5": "E6",
    "Qui6": "E7",
    'Sex1': "F2",
    "Sex2": "F3",
    "Sex3": "F4",
    "Sex4": "F5",
    "Sex5": "F6",
    "Sex6": "F7",
}
```

Fonte: Elaborado pelo Autor

Após a execução da função, o arquivo "Grade Exportada.xlsx" apresenta a grade horária conforme **Figura 30**:

Figura 30 - Aba "Grade Horária" da planilha "Grade Exportada.xlsx"

	Segunda	Terça	Quarta	Quinta	Sexta
08h00 – 10h00					
10h00 – 12h00	Professor 13 - Fenômenos Difusivos - 5º	Professor 13 - Fenômenos Difusivos - 5º			
13h30 – 15h30					
15h30 – 17h30	Professor 3 - Gestão Ambiental - 8º Professor 4 - Projeto de Engenharia do Produto - 9º Professor 8 - Organização e Avaliação do Trabalho - 6º Professor 17 - Projeto de Gestão Econômica - 9º	Professor 4 - Estudo de Tempos e Métodos - 5º Professor 5 - Logística e Gestão da Cadeia de Suprimentos - 8º Professor 6 - Teoria das Organizações - 5º Professor 11 - Psicologia e Sociologia do Trabalho - 7º	Professor 1 - Pesquisa Operacional I - 7º Professor 3 - Gestão Ambiental - 8º Professor 6 - Teoria das Organizações - 5º Professor 8 - Organização e Avaliação do Trabalho - 6º Professor 10 - Planejamento das Unidades Produtivas - 5º Professor 15 - Fundamentos da Termodinâmica - 5º	Professor 5 - Logística e Gestão da Cadeia de Suprimentos - 8º Professor 10 - Projeto de Qualidade - 9º Professor 11 - Psicologia e Sociologia do Trabalho - 7º	Professor 1 - Pesquisa Operacional I - 7º Professor 5 - Projeto de Operações e Processos de Produção - 9º Professor 15 - Fundamentos da Termodinâmica - 5º
17h30 – 19h30	Professor 3 - Gestão da Qualidade e Produtividade - 6º Professor 4 - Gestão da Inovação - 7º Professor 7 - Engenharia do Trabalho - 8º Professor 17 - Modelagem Estatística Aplicada a Engenharia - 6º	Professor 7 - Engenharia do Trabalho - 8º Professor 17 - Modelagem Estatística Aplicada a Engenharia - 6º	Professor 3 - Gestão da Qualidade e Produtividade - 6º Professor 10 - Gestão de Projetos - 6º	Professor 1 - Pesquisa Operacional II - 8º Professor 4 - Gestão da Inovação - 7º	Professor 1 - Pesquisa Operacional II - 8º Professor 10 - Gestão de Projetos - 6º
19h30 – 21h30	Professor 4 - Engenharia do Produto I - 7º Professor 17 - Projeto de Métodos Quantitativos - 9º	Professor 3 - Gestão da Manutenção - 8º Professor 16 - Estrutura e Propriedade dos Materiais - 5º	Professor 3 - Gestão da Manutenção - 8º Professor 4 - Engenharia do Produto I - 7º Professor 10 - Controle da Qualidade - 7º Professor 16 - Estrutura e Propriedade dos Materiais - 5º	Professor 1 - Simulação - 6º Professor 10 - Controle da Qualidade - 7º	Professor 1 - Simulação - 6º

Fonte: Elaborado pelo Autor

Como é possível ver na grade horária acima, todas as restrições foram atendidas. Não existem professores dando aula em mais de uma matéria no mesmo horário, não tem professores com mais de 6 aulas, todas as matérias que foram alocadas, foram alocadas todas as vezes que precisavam, em nenhum horário tem mais de 1 matéria do mesmo período, os professores só foram alocados em horários que estavam disponíveis e em matérias que são aptos e nenhuma matéria tem uma aula em um horário em um dia e outro horário em outro dia. Além disso, fica claro que o período de fim de tarde/ noite foi priorizado, tendo apenas uma aula antes das 15h30, pois a disciplina de Fenômenos Difusivos só possui um professor apto a lecionar (Professor 13).

Com o intuito de medir o tempo de execução do *script*, foram adicionadas algumas linhas de código. Em Python, pode-se usar a biblioteca padrão "*time*" para ter acesso a algumas funções relacionadas a tempos e horários de forma geral.

Figura 31 - Código do arquivo "gerar_grade_horaria.py"

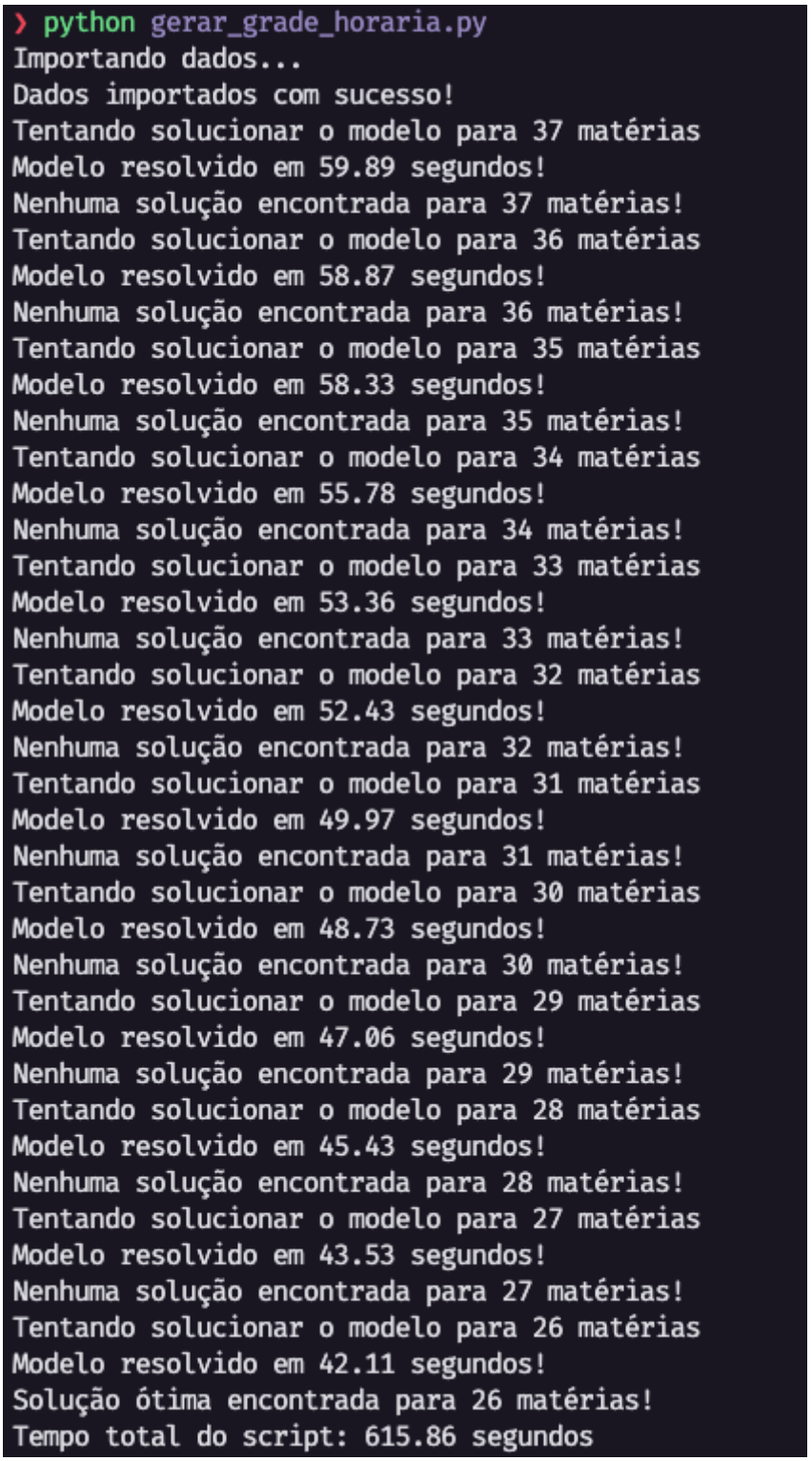
```python
from modelo import modelo
from ortools.sat.python import cp_model
import time

tempo_de_inicio = time.time()

status = cp_model.INFEASIBLE
qtdd_materias_excluidas = 0

while status == cp_model.INFEASIBLE:
    status = modelo(qtdd_materias_excluidas)
    if status == cp_model.INFEASIBLE:
        qtdd_materias_excluidas += 1
    else:
        break

tempo_de_execucao = time.time() - tempo_de_inicio

print(f'Tempo total do script: {round(tempo_de_execucao, 2)} segundos')
```

Fonte: Elaborado pelo Autor

Uma variável chamada "*tempo_de_inicio*" armazena o momento que o script começou a rodar conforme **Figura 31**. Depois que o *script* é finalizado, a variável "*tempo_de_execucao*" é criada para salvar a diferença entre o horário atual e o horário que o *script* começou a rodar (em segundos). Além do tempo total do script, foi medido também o tempo de solução do modelo para cada tentativa, à medida que as matérias são excluídas. Isto possibilita o conhecimento de quanto tempo cada execução gasta, uma vez que em semestres

letivos futuros todas as matérias podem ser alocadas, a depender da quantidade de professores e de quais matérias cada professor é apto a lecionar.

Figura 32 - Print do terminal com os tempos de execução do script



Fonte: Elaborado pelo Autor

Observa-se na **Figura 32**, que com esses parâmetros de professores, matérias e períodos, a execução do *script* durou 615.86 segundos. Algo que é importante perceber é que

levou 58.33 segundos para solucionar o modelo com todas as matérias, e à medida que as matérias vão sendo retiradas o tempo de solução do modelo vai caindo, até 42.11 segundos para solucionar o modelo com 26 matérias.

4.4 Discussão dos resultados

A grade gerada atendeu todas as restrições estabelecidas no modelo de programação linear. Todas as matérias foram alocadas para os professores que estavam aptos a ministrá-las, e todos os professores foram alocados nos horários que estavam disponíveis. Além disso, não houve sobreposição de matérias do mesmo período em horários coincidentes e todas as aulas de uma mesma matéria foram alocadas em horários que começavam na mesma hora.

Ademais, a grade gerada pelo modelo é prática de ser utilizada, pois é exportada no formato de planilha Excel que é o formato utilizado e conhecido pelos alunos e coordenadores. Isso torna a consulta e organização das aulas muito mais fácil, já que não foi necessário realizar nenhum tipo de conversão ou adaptação. Tudo isso contribuiu para a eficiência e praticidade da grade gerada pelo modelo.

Para tal, se fez necessário definir matematicamente as variáveis do modelo, assim como os seus domínios e suas restrições, como apontado por Potts e Smith (1999). Tais definições foram cruciais para a aplicação do modelo computacional, pois facilitaram a construção do mesmo.

Além disso, o uso do módulo OR-Tools para resolver o modelo matemático foi fundamental para tornar a solução viável, já que o problema é muito complexo e demoraria muito tempo para ser resolvido manualmente. Com a utilização do Python e do OR-Tools, foi possível encontrar uma solução ótima para a grade horária de maneira rápida e eficiente, confirmando o que foi dito por Hooker & Osório, em 1999. Soma-se a isso o fato de que a linguagem Python ter sido usada para automatizar o processo de criação da grade e facilitar futuras manutenções da base de código, visto que é uma linguagem presente da grade curricular do próprio curso de Engenharia de Produção.

Por fim, o *script* final torna o processo dos coordenadores muito mais prático e rápido. Basta editar o arquivo de Excel com as informações dos professores do próximo semestre letivo, como a disponibilidade de horário dos mesmos e as matérias que são aptos a lecionar, e rodar o arquivo python "*gerar_grade_horaria.py*". Em poucos minutos, a grade horária já estará disponível em um arquivo Excel chamado "Grade Exportada.xlsx".

4.5 Recomendações

Uma estratégia que se mostrou eficiente utilizada neste trabalho foi a solução do modelo em partes. Devido a pouca experiência do autor com a ferramenta OR-Tools, viu-se

necessária a adição pontual de restrições ao modelo computacional, isto é, o primeiro modelo resolvido em OR-Tools tinha apenas uma restrição. Depois, outra restrição foi adicionada e o modelo foi resolvido. Assim, foi possível aprender sobre a ferramenta gradativamente, à medida que mais restrições eram adicionadas. Isso resultou em uma curva de aprendizado bastante suave e em um modelo completo desenvolvido de maneira mais rápida.

5 CONCLUSÃO

Este estudo teve como objetivo automatizar o desenvolvimento da grade horária do curso de Engenharia de Produção, na UFRJ-Macaé. Para tanto, utilizou-se a linguagem de programação Python e algumas bibliotecas, com ênfase no *openpyxl* para ler os inputs do modelo no Excel e no Google OR-Tools para criação e solução do modelo matemático.

Inicialmente, o estudo debateu sobre o tema de *school timetabling* e PO, entendendo sua origem, principais conceitos e relevância no presente trabalho. Além disso, também foi introduzido o conceito de programação por restrições, que é a abordagem dentro da PO que mais se aplica para resolver o problema. Em sequência, foi necessário compreender o contexto de uma grade horária no curso estudado. Logo, foi preciso entender todas as restrições e critérios que uma grade horária tem que seguir para ser considerada válida, assim como coletar as informações dos professores, matérias e horários para a construção do modelo matemático e computacional.

A partir disso, foi feita a modelagem matemática do problema, definidas todas as variáveis, restrições e também a função objetivo do modelo. Contudo, para solucionar o modelo e automatizar o processo de geração de grade horária, foi necessário, durante o desenvolvimento, passar o modelo matemático para um script em Python que é responsável por usar os dados coletados e gerar a grade em um arquivo Excel em minutos. Em virtude dos fatos, os objetivos desse trabalho de conclusão de curso foram atingidos, visto que o trabalho de fazer a grade horária do período era um processo que poderia levar alguns dias de trabalho do coordenador do curso de Engenharia de Produção, e o script desenvolvido realiza o mesmo trabalho em poucos minutos.

5.1 Limitações do Trabalho

Devido ao contexto em que esse trabalho foi desenvolvido, o da pandemia de COVID19, as aulas eram ministradas de maneira online via Google Meet (aplicativo de videoconferência desenvolvido pela Google) que permite a realização de reuniões online de maneira rápida e fácil. Logo, não houve a necessidade de adicionar as salas de aula e as restrições à grade horária, o que representa, em tempos normais, um fator restritivo para geração de uma grade horária.

5.2 Trabalhos futuros

Indica-se considerar o acréscimo das salas de aula na modelagem, onde os professores teriam que ser alocados para ministrar matérias em horários e em salas de aula específicas, dado que, no contexto de pandemia em que o presente trabalho foi desenvolvido a consideração de salas de aula no modelo não se fez necessário.

Ademais, recomenda-se a criação de uma interface gráfica e um sistema web, onde o coordenador possa adicionar os professores, as matérias e os horários e gerar a grade no próprio sistema web, para que esse resultado possa ficar na nuvem e disponível para todos, ao invés de ser somente um script local.

Referências Bibliográficas

Alvarez-Valdes, R., Martin, G. & Tamarit, J. **Constructing Good Solutions for the Spanish School Timetabling Problem**. J Oper Res Soc 47, 1203–1215 (1996). Disponível em: https://doi.org/10.1057/jors.1996.149

ANDRADE, Eduardo de. Introdução à pesquisa operacional. 2 ed. Rio de Janeiro: LTC, 2000.

ANDRADE, P. R. L., STEINER, M. T. A., GÓES, A. R. T. (2019). Optimization in timetabling in schools using a mathematical model, local search and Iterated Local Search procedures. **Gestão & Produção**, 26(4), e3421.Disponível em: https://doi.org/10.1590/0104-530X3241-19 Acesso em 28/11/2022

BAERÁK, Roman. **Constraint Satisfaction**. Disponível em: http://kti.ms.mff.cuni.cz/~bartak/constraints/constrsat.html Acesso em 29 de nov de 2022
BERROGAIN, Igor. **UFRJ-Macaé torna-se Centro Universitário**: Institutos especializados atuarão em prol de uma formação universitária de qualidade e do amplo desenvolvimento da região. 2021. Disponível em: https://conexao.ufrj.br/2021/07/ufrj-macae-torna-se-centro-universitario/ Acesso em 28 de nov de 2022

BORNIA POULSEN, Camilo José. **Desenvolvimento de um modelo para o school timetabling problem baseado na meta-heurísticasimulated annealing**. Dissertação (Mestrado)--Universidade Federal do Rio Grande do Sul, Escola de Administração, Programa de Pós-Graduação em Administração, 2012. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/39522/000825681.pdf?sequence=1&isAllowed=y Acesso em 28 de nov de 2022

Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). **Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research**, 119, 557-581.

CATUNDA, Heitor. **Bibliotecas do python: conheça as melhores por finalidade!** 2022. Disponível em: Bibliotecas do Python: conheça as melhores por finalidade! (hashtagtreinamentos.com) Acesso em 01 de Dez de 2022

CARDOSO, Andréa. **Fundamentos da pesquisa operacional** Minas Gerais:UNIFAL, 2011. Disponível em: Fundamentos da PESQUISA OPERACIONAL - PDF Free Download (docplayer.com.br) **-** Acesso em 28/11/2022

CHERRI, Luiz Henrique. PROGRAMAÇÃO POR RESTRIÇÕES: UM BREVE TUTORIAL. **Pesquisa Operacional para o Desenvolvimento**, v. 10, n. 1, p. 1-8, 2018.

CONSTRAINT SATISFACTION PROBLEM. 2022. Disponível em: https://en.wikipedia.org/wiki/Constraint_satisfaction_problem Acesso em 29 de nov de 2022

Introdução — documentação Python 3.11.0 **In: A Biblioteca Padrão do Python** Disponível em: https://docs.python.org/pt-br/3/library/intro.html- Acesso em 01de dez de 2022

HIME, Rodrigo. **Uma aplicacão da Programação Inteira no School Timetabling Problem**. 2015. Monografia (Graduação em Sistema de Informação) – Curso de Sistema de Informação – Universidade Federal Rural de Pernambuco, Recife, 2015. Disponível em: http://200.17.137.109:8081/novobsi/bsi-na-ufrpe-recife/trabalhos-de-conclusao-de-curso/2014.2/Hime.pdf - Acesso em 29 de nov de 2022

LEDERMANN, Martin. **Pesquisa operacional**. – Ijuí : Ed. Unijuí, 2012. – 164 p. – (Coleção educação a distância. Série livro-texto).Disponível em: C--_Editora_2012-EaD-Livro Text0011.mdi (unijui.edu.br) Acesso em 28 de nov de 2022

MARINS, Fernando Augusto Silva. **Introdução à Pesquisa Operacional** São Paulo : Cultura Acadêmica : Universidade Estadual Paulista, Pró-Reitoria de Graduação, 2011.

OR-TOOLS. **Otimização de restrições**. Disponível em: https://developers.google.com/optimization/cp Acesso em 28 de nov de 2022

PEREIRA, WILSON INACIO. Pesquisa Operacional: ferramenta para a competitividade. **Instituto Mauá de tecnologia.** v. 31, n. 05, 2014 Disponível em: www. maua. br/arquivos/artigo/h/863f3a2a08f6b7251a2555f56ba01f4e**.** Acesso em 28 de nov de 2022

PERRON, Laurent. FURNON, Vicent. OR-Tools v9.5. Disponível em: https://developers.google.com/optimization/. Acesso em 01 de Dez de 2022

SISTEMA INTEGRADO DE GESTÃO ACADÊMICA. 2022. Disponível em: https://www.siga.ufrj.br/sira/temas/zire/frameConsultas.jsp?mainPage=/repositorio-curriculo/86AA0C62-92A4-F79D-45D8-342907F40115.html Acesso em 01 de Dez de 2022

STICHTING, Copyright & Centrum, Mathematisch; SCHAERF, Andrea. A Survey of Automated Timetabling. Artificial Intelligence Review, v. 13, p. 10.1023/A:1006576209967, 1996.

STALLIVIERI, Luciane. **O Sistema de Ensino Superior do Brasil** - Características, Tendências e Perspectivas. Universidade de Caxias do Sul (UCS), Rio Grande do Sul, 2007.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. **Página Inicial**: Conheça o NUPEM Disponível em: https://nupem.ufrj.br/ Acesso em 01 de Dez de 2022

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. **Consuni - Conselho Universitário da UFRJ** Disponível em: https://consuni.ufrj.br/ Acesso em 01 de Dez de 2022

WREN, A. **Scheduling, timetabling and rostering – A special relationship**? In: BURKE, E. K.; ROSS, P. (Eds.). Practice and theory of automated timetabling: Vol. 1153. Lecture notes in computer science. Springer, 1996.

ANEXO A - FORMULÁRIO

Afinidade de matérias e disponibilidade de horários por professor - Engenharia de Produção - UFRJ Macaé

Esse formulário tem o intuito de coletar informações dos professores de Engenharia de Produção da UFRJ Macaé, afim de avaliar o programa criado no meu TCC.

O meu TCC tem como objetivo final criar um programa que seja capaz de gerar uma grade horária para o curso de Engenharia de Produção, levando em consideração algumas informações iniciais como: quais matérias tem que ser lecionadas, quais professores estão disponíveis em quais horários, quais matérias cada professor se sente confortável em lecionar e por ai vai. Portanto estou coletando essas informações para poder validar o meu TCC com dados reais e montar uma grade para a Engenharia de Produção da UFRJ Macaé.

Obrigado!

*Obrigatório

1. Qual seu nome? *

_____

2. Quais matérias você se sente confortável em lecionar? *

Por favor, selecione todas as opções que você se sente confortável.

Marque todas que se aplicam.

- [ ] Fenômenos Difusivos Fundamentos
- [ ] da Termodinâmica
- [ ] Estrutura e Propriedade dos Materiais
- [ ] Estudo de Tempos e Métodos Planejamento
- [ ] das Unidades Produtivas Teoria das
- [ ] Organizações
- [ ] Estratégia da Produção
- [ ] Organização e Avaliação do Trabalho
- [ ] Gestão da Qualidade e Produtividade
- [ ] Simulação
- [ ] Modelagem Estatística Aplicada a Engenharia
- [ ] Gestão de Projetos
- [ ] Princípios da Economia Planejamento e
- [ ] Controle da Produção
- [ ] Fundamentos da Engenharia do Petróleo
- [ ] Gestão da Inovação
- [ ] Psicologia e Sociologia do Trabalho
- [ ] Controle da Qualidade
- [ ] Engenharia do Produto I
- [ ] Pesquisa Operacional I
- [ ] Planejamento e Controle da Produção II
- [ ] Contabilidade Gerencial e Custos
- [ ] Engenharia do Trabalho
- [ ] Gestão da Manutenção
- [ ] Pesquisa Operacional II
- [ ] Logística e Gestão da Cadeia de Suprimentos
- [ ] Gestão Ambiental
- [ ] Engenharia Econômica
- [ ] Empreendedorismo
- [ ] Projeto de Qualidade
- [ ] Projeto de Métodos Quantitativos
- [ ] Projeto de Operações e Processos de Produção
- [ ] Projeto de Gestão Econômica
- [ ] Projeto de Engenharia do Produto Projeto
- [ ] de Engenharia Organizacional Projeto de
- [ ] Engenharia do Trabalho
- [ ] Metodologia de Pesquisa na Engenharia de Produção

3. Quais horários você está disponível para ministrar aulas? *

Vale ressaltar que esses horários são para um período letivo normal, sem pandemia.

Marque todas que se aplicam.

- [ ] Segunda - 08h00:10h00
- [ ] Segunda - 10h00:12h00
- [ ] Segunda - 13h30:15h30
- [ ] Segunda - 15h30:17h30
- [ ] Segunda - 17h30:19h30
- [ ] Segunda - 19h30:21h30
- [ ] Terça - 08h00:10h00
- [ ] Terça - 10h00:12h00
- [ ] Terça - 13h30:15h30
- [ ] Terça - 15h30:17h30
- [ ] Terça - 17h30:19h30
- [ ] Terça - 19h30:21h30
- [ ] Quarta - 08h00:10h00
- [ ] Quarta - 10h00:12h00
- [ ] Quarta - 13h30:15h30
- [ ] Quarta - 15h30:17h30
- [ ] Quarta - 17h30:19h30
- [ ] Quarta - 19h30:21h30
- [ ] Quinta - 08h00:10h00
- [ ] Quinta - 10h00:12h00
- [ ] Quinta - 13h30:15h30
- [ ] Quinta - 15h30:17h30
- [ ] Quinta - 17h30:19h30
- [ ] Quinta - 19h30:21h30
- [ ] Sexta - 08h00:10h00
- [ ] Sexta - 10h00:12h00
- [ ] Sexta - 13h30:15h30
- [ ] Sexta - 15h30:17h30
- [ ] Sexta - 17h30:19h30
- [ ] Sexta - 19h30:21h30

ANEXO B - RESPOSTAS