MIGUEL RODRIGUES GUIMARÃES DE OLIVEIRA

UM ALGORITMO PARA A RESOLUÇÃO DO PROBLEMA DE DESIGNAÇÃO DE DISCIPLINAS A PROFESSORES EM UM CONTEXTO UNIVERSITÁRIO

LAVRAS – MG

2023

MIGUEL RODRIGUES GUIMARÃES DE OLIVEIRA

UM ALGORITMO PARA A RESOLUÇÃO DO PROBLEMA DE DESIGNAÇÃO DE DISCIPLINAS A PROFESSORES EM UM CONTEXTO UNIVERSITÁRIO

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação.

Prof. Mayron César de Oliveira Moreira
Orientador

Profa. Andreza Cristina Beezão Moreira
Coorientadora

LAVRAS – MG

2023

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Oliveira, Miguel Rodrigues Guimarães de

Um algoritmo para a Resolução do Problema de Designação de Disciplinas a Professores em um Contexto Universitário / Miguel Rodrigues Guimarães de Oliveira. 2^a ed. rev., atual. e ampl. – Lavras : UFLA, 2023.

30 p. : il.

Projeto(bacharelado)–Universidade Federal de Lavras, 2023.

Orientador: Prof. Mayron César de Oliveira Moreira.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

MIGUEL RODRIGUES GUIMARÃES DE OLIVEIRA

UM ALGORITMO PARA A RESOLUÇÃO DO PROBLEMA DE DESIGNAÇÃO DE DISCIPLINAS A PROFESSORES EM UM CONTEXTO UNIVERSITÁRIO
AN ALGORITHMIC SOLUTION TO THE TEACHERS ASSIGNMENT PROBLEM IN UNIVERSITY CONTEXT

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 28 de 07 de 2023.

Prof. Eric Fernandes de Mello Araújo	UFLA
Prof. Paulo Afonso Parreira Junior	UFLA

gov.br
Documento assinado digitalmente
MAYRON CESAR DE OLIVEIRA MOREIRA
Data: 03/08/2023 18:09:38-0300
Verifique em https://validar.iti.gov.br

Prof. Mayron César de Oliveira Moreira
Orientador

gov.br
Documento assinado digitalmente
ANDREZA CRISTINA BEEZAO MOREIRA
Data: 03/08/2023 18:06:19-0300
Verifique em https://validar.iti.gov.br

Profa. Andreza Cristina Beezão Moreira
Co-Orientadora

LAVRAS – MG
2023

Dedico à minha bisa Ruth, que sempre quis me ver formando.

AGRADECIMENTOS

Agradeço a minha família, que me deu apoio durante toda a minha graduação. Agradeço aos meus amigos, pela companhia e momentos de distração.

Agradeço ao meu orientador Mayron César de Oliveira Moreira, por toda a orientação paciente e acolhedora. Assim como agradeço também a co-orientadora Andreza Cristina Beezão Moreira pelo suporte e auxilio.

Por fim, agradeço a Universidade Federal de Lavras e ao Departamento de Ciência da Computação e seus profissionais, os quais além de me proporcionarem um bom meio acadêmico, o proporcionaram com excelência.

Não quero ser um gênio... Já tenho problemas suficientes ao tentar ser um homem.
Albert Camus

RESUMO

O problema de designação de professores (TAP) considerado neste trabalho distribui professores entre as disciplinas disponíveis, considerando preferências dos docentes e as restrições relacionadas a horários, regras de departamento e outras implícitas (como a impossibilidade de um professor ministrar duas aulas ao mesmo tempo). Para resolver esse problema, propomos um algoritmo baseado em programação por restrições e que utiliza ferramentas da Google OR-Tools. Para testar a qualidade do algoritmo, realizamos um experimento computacional utilizando dados de um departamento de uma universidade brasileira. Os resultados mostram que o método resolve o problema em questão e pode ser útil para outros departamentos com condições similares.

Palavras-chave: Programação por restrição. Problema de designação de professores.

ABSTRACT

The teaching assignment problem (TAP) considered in this work involves assigning teachers to courses, taking their preferences into consideration and considering restrictions such as the number of weekly hours taught by them and the limit of two consecutive shifts per day. To tackle this problem, we propose an algorithm based on Constraint Programming techniques and which use Google OR-Tools. To test the algorithm's performance, a computational experiment is carried out using data from from a department of a Brazilian university. Results show that the method solves the problem, and can be useful for other departments with similar requests.

Keywords: Constraint programming. Teacher Assignment Problem.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O problema de designação é um clássico na área de Otimização Combinatória. Esse problema pode se apresentar como a designação de professores, que consiste em distribuir um conjunto de disciplinas para um determinado grupo de professores, considerando uma série de regras, como o número mínimo de horas semanais que um professor deve lecionar, regras trabalhistas, entre outras. Além disso, costuma-se também considerar as preferências dos docentes, objetivando que a maior quantidade de professores lecionem matérias de seu interesse.

A designação de professores para disciplinas pode ser modelado como um problema de programação inteira, composto por variáveis binárias Karp (1972). Problemas dessa natureza se enquadram, em geral, na classe dos problemas NP-difíceis, conforme indicam os trabalhos Sahni e Gonzalez (1976), Bardadym (1996), Hartmanis (1982). Ao adicionarmos particularidades advindas das regras de prioridade, geramos um problema de otimização cuja resolução manual se torna custosa.

Por isso, é importante a elaboração de um algoritmo, tanto para gerar quanto para validar soluções para esse problema. O Problema de Designação de Professores (TAP, do inglês: *Teacher Assignment Problem*) é um problema combinatório e pode ser modelado através de um conjunto de restrições. Considerando isso e por ter resultados computacionais competitivos com outros métodos Demirović e Stuckey (2018), optamos pela técnica de programação por restrição, ou *Constraint Programming*, em inglês, para gerar esse algoritmo.

1.1 Objetivos

Este trabalho possui, como objetivos: (i) o desenvolvimento de um algoritmo baseado em programação por restrições que solucione o problema de designação de professores a disciplinas em um contexto de restrições especifico que será explicado no capítulo 2, e (ii) a criação de uma interface que permita a interação do usuário com o algoritmo proposto e seus resultados. Dessa forma, ao fim, esperamos gerar uma ferramenta que possa ser utilizada por departamentos universitários.

1.2 Revisão bibliográfica

A técnica de programação por restrições tem conseguido espaço como algoritmo para solucionar problemas de designação. Como exemplo disso, Wang, Meskens e Duvivier (2015) usam esta técnica para organizar o agendamento de uma sala de operações. Outro trabalho que aborda um problema e forma de solução semelhantes é o de Tang et al. (2018), que tem como finalidade desenvolver uma solução de *linear scheduling method*. Esse método é utilizado para a alocação de recursos em atividades repetitivas em contexto de construção industrial, como na construção de rodovias, ferrovias, gasoduto. Por fim, citamos o trabalho de Demirović e Stuckey (2018), também sobre ordenação em contexto acadêmico. Os autores usaram *solution-based phase saving* e *hot starts* os quais fornecem, respectivamente, um direcionamento para o *solver* da região próxima da melhor solução encontrada e um ponto de partida para o algoritmo que utilizou programação por restrição. Com essas ferramenta foram obtidos resultados semelhantes aos de abordagens focadas em heurísticas, dentre elas *Simulated Annealing* e *Adaptive Large Neighborhood Search*.

Para melhor compreender e organizar o problema em questão, citamos referências importantes deste contexto. Carter e Laporte (1997) particionam o problema de agendamento acadêmico em cinco tipos:

1. Horário do Curso: é o agendamento respeitando sua ementa e a disponibilidade das salas e dos professores;

2. Horários do Professor-Aula: organiza os encontros de turmas com professores de forma a não ocorrerem conflitos;

3. Horários dos Alunos: faz o agendamento de seções de aula para um determinado número de alunos, de forma a alocá-los em uma sala com capacidade suficiente e sem conflitos;

4. Atribuição do Professor: atribui os professores a cursos, de forma a atender o máximo de preferências possível;

5. Atribuição de Sala de Aula: atribui eventos a determinadas salas, normalmente com os horários já assinalados.

Utilizando essa divisão como base, é perceptível que o problema deste trabalho se enquadra no grupo 4-Atribuição de Professor.

1.3 Organização do trabalho

O Capítulo 2 contém a descrição formal do problema estudado e sua modelagem. No Capítulo 3, apresentamos a metodologia adotada e as ferramentas utilizadas no algoritmo e na interface. O Capítulo 4 sintetiza os resultados advindos do experimento computacional realizado, enquanto o Capítulo 5 finaliza com a conclusão do trabalho e sugestões para contribuições futuras.

2 DESCRIÇÃO DO PROBLEMA

No contexto do problema estudado, os horários das disciplinas, as turmas que frequentarão as disciplinas e os locais de aula já estão determinados. Definimos o termo *composição de turma* como um conjunto de alunos, de diferentes cursos, que, juntos, frequentam as aulas de uma dada matéria em um mesmo horário, local e com o mesmo professor. O objetivo do nosso trabalho é, então, determinar qual docente lecionará para qual composição de turma. Para isso, também é necessário respeitar regras relacionadas à legislação trabalhista e à própria organização do departamento, ao mesmo tempo em que buscamos maximizar a satisfação dos docentes, atendendo suas preferências.

Entre as regras a serem respeitadas, as relacionadas à legislação ditam que: (i) um docente que ministre aulas até as 22h40 poderá voltar a ministrar outra aula apenas após as 10h da manhã do dia seguinte; (ii) em um mesmo dia, um docente pode lecionar em turnos consecutivos (manhã e tarde ou tarde e noite), mas não nos turnos da manhã e da noite; (iii) a carga horária semanal de um docente, em sala de aula, deve ser de no máximo 16 e no mínimo 8 créditos.

Com relação às regras do departamento, são elas:

1. Um professor tem prioridade para lecionar para uma mesma composição de turma por três semestres seguidos. Caso haja disputa por uma composição de turma, vence o professor que melhor atender aos seguintes critérios, por ordem de importância:
    (a) Lecionou para um maior número de disciplinas diferentes no semestre anterior;
    (b) Lecionou para um maior número de alunos no semestre anterior;
    (c) Não ministrou a disciplina em questão no semestre anterior;
    (d) Está há mais tempo sem ministrar a disciplina disputada.
2. A redução de carga horária para 8 créditos pode ocorrer nas seguintes situações:
    (a) Docentes cursando pós-graduação;
    (b) Docentes em cargo de coordenação de curso ou chefia departamental;
    (c) Docentes que não tiveram redução de carga horária no período anterior e que lidera um *ranking* que considera, nesta ordem, o maior número de créditos, o maior número de disciplinas diferentes ou o maior número de estudantes no semestre anterior.

Por último, há algumas regras implícitas, mas que devem ser fornecidas na modelagem, como a impossibilidade de um professor ministrar duas aulas ao mesmo tempo e a necessidade de que toda composição de turma tenha um docente a ela atribuído. Como base para a geração dos dados de entrada e para a verificação da qualidade da solução gerada, foram usados dados públicos de um departamento da Universidade Federal de Lavras (UFLA). Esses dados foram anonimizados por meio da alteração dos códigos das disciplinas e dos nomes dos docentes. Os dados relacionados à preferência dos docentes não nos foram fornecidos, portanto foram gerados a partir da informação das disciplinas lecionadas nos semestres anteriores, considerando que os docentes teriam preferência em permanecer com as mesmas composições de turma.

Os parâmetros, as variáveis de decisão, as restrições e a função objetivo da modelagem matemática estão listados a seguir.

2.1 Parâmetros do problema

$T = \{1, ..., n\}$	Conjunto de composições de turmas (ex.: GMM101-9A).
$P = \{1, ..., m\}$	Conjunto de professores.
$S = \{1, ..., k\}$	Conjunto de *slots* de dia/hora para uma semana (ex.: 01 - Seg/7h-7h50, 02 - Seg/7h-8h40, 03 - Seg/8h-9h40, ..., k - Sex/21h-22h40).
$\hat{T}_p \subset T$	Conjunto das composições de turma priorizadas pelo professor $p \in P$. Note que o primeiro elemento em $\hat{T}_p$ tem maior prioridade em relação ao segundo elemento, o segundo elemento tem maior prioridade em relação ao terceiro elemento, e assim por diante.
$\Delta_i^p \subset T$	Conjunto de composições de turma alocadas ao professor p no período $i \in \{\alpha-3, \alpha-2, \alpha-1\}$, em que α é o próximo período letivo e $\alpha-1$ é o período atual.
$\hat{P}_u^{c'} \subset P$	Conjunto de professores que possuem carga didática reduzida para c', por se enquadrarem na condição u, explicada anteriormente.
$h_t \in S$	Dia/hora (*slot*) da composição de turma $t \in T$.

$c_t \in \{1,2,4,6\}$	Quantidade de créditos de uma turma/disciplina $t \in T$.
$\underline{c}$	Número mínimo de créditos de um professor (no caso, 8 créditos).
$\overline{c}$	Número máximo de créditos de um professor (no caso, 16 créditos).
$\overline{c_m}$	Número máximo de créditos de um professor $p \in P$ com carga reduzida (no caso, 12 créditos).
$S_m \subset S$	Conjunto de *slots* matutinos.
$S_n \subset S$	Conjunto de *slots* noturnos.
R	*Ranking* de prioridade de desempate de professores disputando composições de turmas $d_a, d_b, \ldots$ livres (ou seja, que não estão na regra de prioridade dos três semestres, para qualquer professor do departamento), ordenado por composição: $R = \{d_a = \{p_1, \cdots, p_m\}, d_b = \{p_1, \cdots, p_n\}, \cdots\}$.

2.2 Variáveis de decisão

$x_{pt} \in \{0,1\}$	Variável binária igual a 1 se o professor $p \in P$ é designado à composição de turma $t \in T$.

2.3 Restrições

Apresentamos a modelagem matemática das restrições, seguidas de suas explicações por extenso.

Restrição 1

$$\sum_{p \in P} x_{pt} = 1, \quad \forall t \in T \tag{2.1}$$

Uma composição de turma $t \in T$ deve ser ministrada por exatamente um professor $p \in P$.

Restrição 2

$$\underline{c} \leq \sum_{t \in T} (c_t x_{pt}) \leq \overline{c}, \quad \forall p \in (P - \hat{P}_u^{c'}) \tag{2.2}$$

$$\underline{c} \leq \sum_{t \in T} (c_t x_{pt}) \leq \overline{c_m}, \quad \forall p \in \hat{P}_u^{c'} \tag{2.3}$$

Cada professor $p \in P$ deve ter, como soma de todos os créditos que lecionará, no mínimo $\underline{c}$ créditos. Os que não estão no grupo de professores que possuem carga reduzida ($p \notin \hat{P}_u^{c'}$) devem lecionar no máximo $\overline{c}$ créditos (2.2). Quanto aos docentes que estão no grupo com carga reduzida ($p \in \hat{P}_u^{c'}$), estes deverão lecionar no máximo $\overline{c_m}$ créditos (2.3), sendo $\overline{c_m} \leq \overline{c}$.

Restrição 3

$$x_{pt_1} + x_{pt_2} \leq 1, \quad \forall p \in P, \forall t_1, t_2 \in T, t_1 \neq t_2, h_{t_1}, h_{t_2} \in S, h_{t_1} = h_{t_2} \tag{2.4}$$

Um professor p não pode lecionar para composições de turmas que estejam no mesmo *slot*, isto é, que ocorram ao mesmo tempo.

Restrição 4

$$\begin{aligned} \{s_1 = (s_1^d, s_1^{hi}, s_1^{hf}), s_2 = (s_2^d, s_2^{hi}, s_2^{hf})\} \in S, \\ (((s_1^d = s_2^d + 1) \wedge ((s_1^{hf} = 22:40) \wedge (s_2^{hi} < 10:00))) \vee \\ ((s_2^d = s_1^d + 1) \wedge ((s_2^{hf} = 22:40) \wedge (s_1^{hi} < 10:00)))) \rightarrow \\ x[p][s_1] + x[p][s_2] \leq 1, \\ \forall p \in P \end{aligned} \tag{2.5}$$

A restrição de intervalo de expediente indica que um professor que possui uma disciplina que seu horário de fim (s_x^{hf}, sendo x igual a um ou dois) até às 22h40 do dia s_x^d, no dia s_y^d (sendo $s_x^d + 1 = s_y^d$) o horário de início da aula (s_y^{hi}) deve ser maior que 10h.

Restrição 5

$$t = \Delta_1^p \wedge \neg(t = \Delta_2^p \wedge t = \Delta_3^p) \rightarrow x_{pt} = 1, \quad \forall p \in P, \forall t \in \hat{T}_p \tag{2.6}$$

Um professor p com interesse na composição de turma t, deve lecionar essa disciplina caso a tenha lecionado no último semestre letivo, mas não tendo lecionado-a em todos os últimos três semestres.

Restrição 6

$$\{s_1 = (s_1^d), s_2 = (s_2^d)\} \in S,$$
$$(s_1^d = s_2^d) \wedge (((s_1 \in S_m) \wedge (s_2 \in S_n)) \vee ((s_1 \in S_n) \wedge (s_2 \in S_m))) \rightarrow$$
$$x[p][s_1] + x[p][s_2] \leq 1,$$
$$\forall p \in P \quad (2.7)$$

Duas composições de turmas s_1, s_2 com aulas alocadas para um mesmo dia da semana $s_1^d = s_2^d$, sendo que uma ocorre no turno da manhã e outra no turno noturno, não podem ser lecionadas pelo mesmo p professor.

2.4 Função objetivo

Maximização das preferências dos professores

$$z = \max \sum_{p \in P} \sum_{t \in \hat{T}_p} t x_{pt} \quad (2.8)$$

Cada professor $p \in P$ inicialmente preenche quais conjuntos de composições de turma tem interesse, formando o grupo $\hat{T}_p$. A partir disso, para cada $t \in \hat{T}_p$ atribuída ao tal professor (ou seja, sempre que $x_{pt} = 1$), o peso da preferência é considerado na função objetivo.

3 METODOLOGIA

O desenvolvimento do produto proposto nesse trabalho foi dividido em duas partes: o desenvolvimento do modelo para a resolução via algoritmo de programação por restrições e a criação de uma interface gráfica. Detalhes são apresentados a seguir.

3.1 Resolução por Programação por Restrições

Para implementar as restrições e funções objetivo descritas no Capítulo 2, foi utilizada a biblioteca *Google OR-Tools* em *python*. Essa ferramenta foi adotada por ser gratuita e também por apresentar uma desempenho melhor, quando comparada a outros *solvers* Müller et al. (2022). Um *solver* é um tipo de algoritmo que resolve problemas matemáticos, recebendo descrições genéricas dos problemas e calculando uma solução de acordo com uma série de critérios. Com a *Google OR-Tools* é possível a criação de um modelo com regras que devem ser atendidas. No nosso trabalho, o modelo é representado por uma matriz *booleana* de docentes por composições de turma, em que o valor-verdade 'Verdadeiro' significa que o docente leciona para a composição intercessora. Um exemplo dessa matriz pode ser visto na Figura 3.1 As regras, por sua vez, são representadas por funções, que, no caso das restrições, devem ser respeitadas, e no caso da função objetivo, deve ser maximizada.

Figura 3.1 – Exemplo de matriz de atribuição

	Doc 1	Doc 2
CT 1	0	0
CT 2	0	0

Como exemplo do funcionamento das restrições, podemos observar a Figura 3.2, em que encontra-se a função "res_um_doc_por_dis", que implementa a restrição da Equação 2.1, que diz que cada composição de turma deve ser lecionada por um professor.

Figura 3.2 – Implementação da restrição 2.1

```
def res_um_doc_por_dis(self):
    for dis in self.disciplinas:
        self.modelo.AddExactlyOne(
            self.atribuicao[(doc.pos, dis.pos)] for doc in self.docentes)
```

Já no código da Figura 3.3, podemos ver a implementação da função objetivo 2.8, em que são somadas as preferências atendidas dos docentes.

Figura 3.3 – Implementação da função objetivo 2.8

```
def opt_interesse(self):
    pref_disc = 0
    for doc in self.docentes:
        for dis in self.disciplinas:
            if dis.codigo in doc.preferencia:
                pref_disc += (self.atribuicao[(doc.pos, dis.pos)] * doc.preferencia[dis.codigo])
    return pref_disc
```

A *CP-optimizer*, ferramenta que resolve modelos de Programação por Restrições proveniente da *Google OR-Tools*, trabalha prioritariamente para encontrar soluções viáveis. Isso se deve a natureza de problemas que em geral são resolvidos via Programação por Restrições, caracterizados pela dificuldade de encontrar solução viável devido à intrincada relação entre suas restrições. O algoritmo trabalha com uma árvore de enumeração, incrementada através de regras de redução de domínio de variáveis, resultado da propagação de informações entre os nós da árvore. Isso causa um processo de poda nos ramos da árvore, reduzindo o tempo computacional para a obtenção de soluções viáveis. Sugere-se ao leitor o acesso a obra de Smith (1995) para mais detalhes a respeito da metodologia implementada em resolvedores de Programação por Restrições.

Com o objetivo de respeitar aos critérios de desempate, favorecendo aqueles docentes que pontuarem mais, foi decidido acrescentar mais uma parte a função objetivo, a qual pode ser vista a seguir.

Função objetivo de desempate

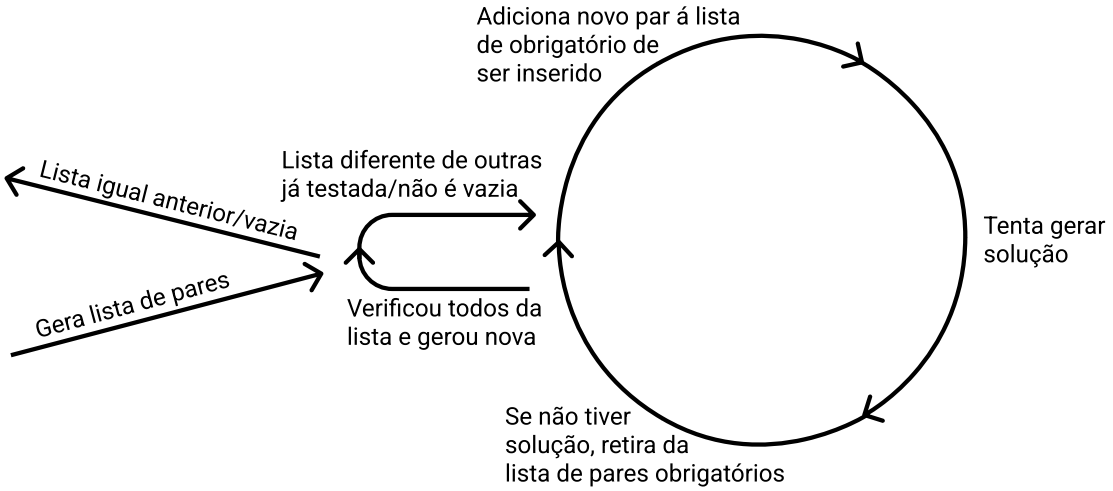
$$\sum_{(d,L)\in R}\sum_{p\in L} x[p][d] * (|L| - p) \tag{3.1}$$

Para cada composição de turma d que foi alocada a um professor p que está em um ranking $p \in L$, o valor da subtração do tamanho da lista com o valor da posição que p possui na lista $(|L| - p)$ é considerado na função objetivo.

Ao fazer uma análise buscando espaços de melhora, percebemos que havia. Essa análise consistiu em criar uma função em *python* que verificaria a quantidade e quais os casos em que um docente no ranking, mas não no topo, foi atribuído para a disciplina em disputa. Com

isso, foi decidido desenvolver também um algoritmo de pós-otimização, o qual tenta certificar que o docente que for designado para uma composição de turma em que houve disputa, é, de fato, quem pontuou mais e quem tem maior preferência pela composição em questão. O funcionamento do algoritmo pode ser visto na Figura 3.5. Primeiramente, uma primeira solução é buscada, comprovando a factibilidade do problema abordado. A partir de tal solução, é gerada uma lista com cada caso em que o *ranking* $\hat{T}_p$ foi infringido. Através dessa lista, são geradas novas soluções em que, a cada nova solução gerada, há uma tentativa forçada de corrigir a infração, trocando o docente da composição de turma. Caso a nova solução seja factível, a solução gerada é mantida. Caso contrário, é descartada. As tentativas são repetidas até que a lista de infrações esteja vazia. Nesse momento, ocorre a geração de uma nova lista de infrações e, caso a nova lista esteja vazia ou seja igual alguma anterior, o programa termina sua execução, fornecendo, como solução definitiva, a última que foi gerada. Um diagrama que representa esse algoritmo pode ser encontrado na Figura 3.4

Figura 3.4 – Diagrama de pós-otimização



3.2 Interface

A interface foi desenvolvida em plataforma *web*, com *front-end* utilizando HTML, Sass e Java Script com JQuery. No *back-end* usamos Python, juntamente com a biblioteca Flask. Essa interface pode ser dividida em duas partes: a de envio/carregamento de arquivo e a de visualização do resultado gerado.

No processo de leitura dos dados de entrada, utilizamos duas possibilidades de formato arquivos: *Json* (Figura 3.6) e CSV (Figura 3.9). O tipo *Json* foi escolhido pela simplicidade em

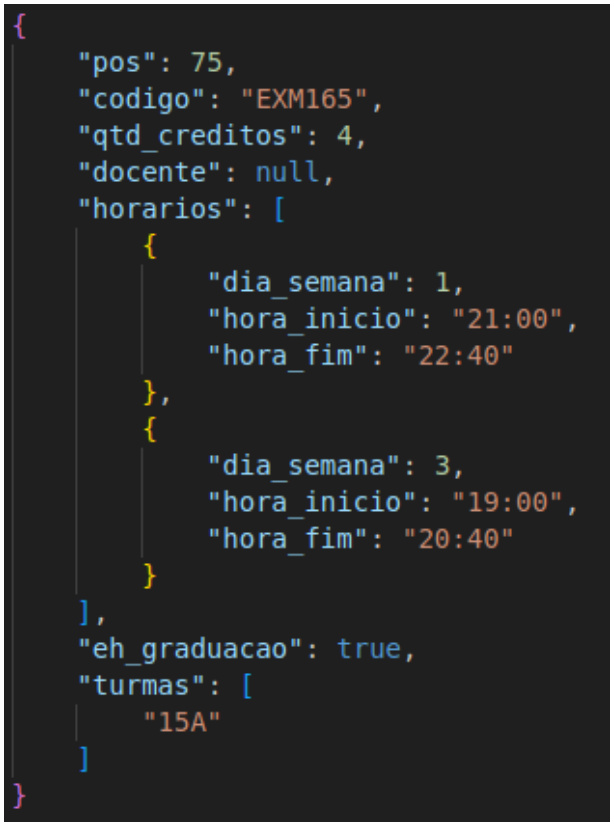
Figura 3.5 – Implementação do algoritmo de pós-otimização

```
def fixa_top_ranking(self, disciplinas, docentes):

    lista_de_infracoes_ocorridas = []
    infracoes = self.lista_restricao_capeoes_de_ranking()
    lista_de_infracoes_ocorridas.append([])

    while infracoes not in lista_de_infracoes_ocorridas:
        lista_de_infracoes_ocorridas.append(infracoes)

        for infracao in infracoes:
            self.fixados.append(infracao)
            resultado = self.soluciona(disciplinas, docentes)

            if resultado == False:
                self.fixados.pop()

        infracoes = self.lista_restricao_capeoes_de_ranking()
```

Figura 3.6 – Tela de envio de arquivos *Json*

salvar e carregar os dados. Um exemplo dos dados de entrada desse tipo pode ser visto nas Figuras 3.8 e 3.7. Já o CSV foi adotado pelo fato de que há departamentos que utilizam planilhas para fazer a distribuição das composições de turma entre os docentes. Dessa forma, para uma fácil transição do modo de trabalho com planilhas para um que envolva o nosso produto, uma interação dos dados das planilhas com o produto é desejável. Além disso, as ferramentas de manipulação de planilhas (*Excel, LibreOffice, Google* Planilhas, entre outros) costumam possuir conversão de seus tipos de arquivo padrão para o CSV, e vice-versa. Igualmente, é um formato que possui uma certa constância em sua configuração, o que pode não ocorrer com os outros tipos de arquivo de planilha, com possibilidade de alteração devido à versão da ferramenta. Por fim, exemplos de entrada dos dados no tipo CSV também podem ser vistos nas Figuras 3.10, 3.11, 3.12, 3.13. Há dois pontos importantes sobre a Figura 3.10, que representa uma única composição de turma. Um deles é que ela representa quatro arquivos de entrada: o de disciplinas no próximo semestre, do ultimo, penúltimo e antepenúltimo. O segundo é que os únicos

Figura 3.7 – Formatação do arquivo *Json* de entrada de docentes



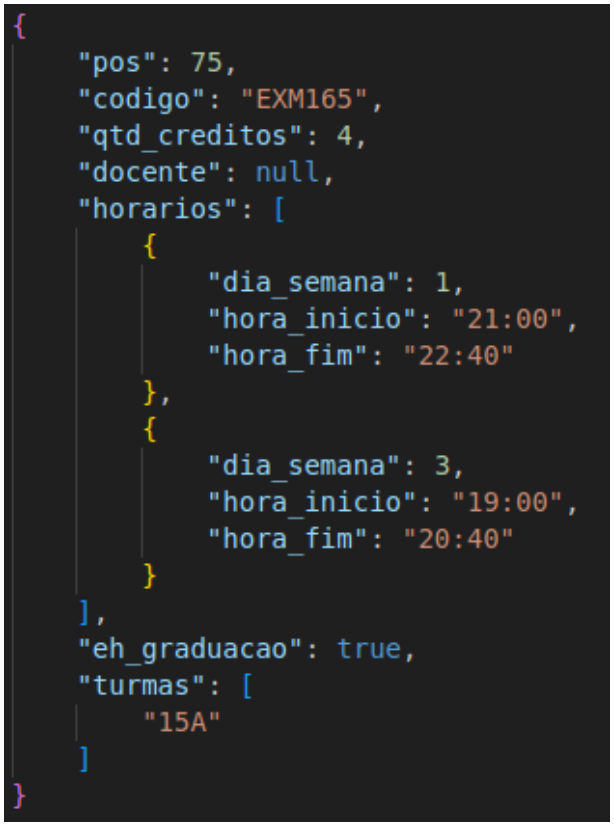
```
{
    "pos": 75,
    "codigo": "EXM165",
    "qtd_creditos": 4,
    "docente": null,
    "horarios": [
        {
            "dia_semana": 1,
            "hora_inicio": "21:00",
            "hora_fim": "22:40"
        },
        {
            "dia_semana": 3,
            "hora_inicio": "19:00",
            "hora_fim": "20:40"
        }
    ],
    "eh_graduacao": true,
    "turmas": [
        "15A"
    ]
}
```


dados utilizados pelo programa são: disciplina, dia, horário, turma e docente (docente não é usado para as disciplinas no próximo semestre).

Feito o envio, é possível gerar a solução usando o botão “Resolver”. O resultado da solução pode ser baixado no formato selecionado (*Json* ou CSV) ou podem ser utilizadas as quatro visualizações da interface. A primeira delas viabiliza uma análise geral de dados relevantes da solução (Figura 3.14), em que é possível saber se foi possível atender às restrições de prioridade e de horário, e dados estatísticos, como a média de créditos e o percentual de aulas que eram preferidas.

Na segunda forma de visualização (Figura 3.15), há um campo do tipo *select* com opções de professores, ordenados pelo nome. Quando um desses nomes é selecionado, são mostradas, em tabela, as composições de turma que serão lecionadas pelo docente, dispostas de forma a informar o horário e dia em que cada aula ocorrerá. Além disso, em cima dessa tabela há também um campo do tipo *checkbox* que, quando selecionado, marca as disciplinas que causam infrações nas restrições de horário, caso isso ocorra. A segunda tabela (Figura 3.16) mostra a lista de composições de turma de cada docente, com estilização do fundo de

Figura 3.8 – Formatação do arquivo *Json* de entrada de disciplinas



```
{
    "pos": 75,
    "codigo": "EXM165",
    "qtd_creditos": 4,
    "docente": null,
    "horarios": [
        {
            "dia_semana": 1,
            "hora_inicio": "21:00",
            "hora_fim": "22:40"
        },
        {
            "dia_semana": 3,
            "hora_inicio": "19:00",
            "hora_fim": "20:40"
        }
    ],
    "eh_graduacao": true,
    "turmas": [
        "15A"
    ]
}
```


acordo com a preferência desse professor: caso a disciplina tenha sido marcada como preferida, o fundo fica verde; caso contrário, o fundo fica vermelho.

Por fim, a última informação a ser visualizada (Figura 3.17) são duas listas que mostram as composições de turma liberadas, ou seja, que não estão designadas a algum professor pela regra de prioridade dos três semestres. A primeira lista mostra as que foram liberadas no semestre atual. A segunda mostra as que serão liberadas no próximo semestre. Essa informação é bastante útil para professores que possuem interesse em lecionar para uma composição de turma diferente das que está lecionando, no momento atual.

Figura 3.9 – Tela de envio de arquivos csv

Figura 3.10 – Formatação do arquivo csv de entrada de disciplinas

Disciplina	Local	Tipo	Tempo	Período	Dia	Horário	Turma	Vagas	Vagas	Vagas	Total	Total	Total	Docente
EXM102	PV4-01	T	S	1	Terça-Feira	07:00 - 08:40	30E	60	56	6	120	112	12	Docente 2
							30G	60	56	6				
EXM102	PV4-01	T	S	1	Sexta-Feira	07:00 - 08:40	30E	60	56	6	120	112	12	
							30G	60	56	6				

Figura 3.11 – Formatação do arquivo csv de entrada de docentes

SIAPE	Nome	Redução
1	Docente 1	
2	Docente 2	1

Figura 3.12 – Formatação do arquivo csv de entrada de preferencias

Siape do Docente	1	2	3	4	5
1	EXM104_3A5A	EXM104_30E30G	EXM104_5A3A	EXM104_21A11A13A	EXM104_19A
2	EXM130_31B	EXM234_32A	EXM130_32B	EXM130_31A31B32A32B	EXM109_15A18A

Figura 3.13 – Formatação do arquivo csv de entrada dos dados do final do último semestre

SIAPE	Professores	Créditos	Número de Disciplinas	nº estudantes fim de período
1	Docente 1	Licença		
2	Docente 2	16	3	109

Figura 3.14 – Dados gerais da solução

Dados da otimização encontrada:

Houve prioridade: Houve

Houve restrição de horario: Houve

Média de créditos: 10.45

Desvio padrão: 2.32

Percentual de preferencias atendidas: 31.66%

Percentual de aulas que eram preferidas: 62.29%

Percentual de preferencias de peso 5 atendidas: 20.83%

Percentual de preferencias de peso 4 atendidas: 20.83%

Percentual de preferencias de peso 3 atendidas: 29.16%

Figura 3.15 – Tabela de horários dos docentes

Escolha um professor: Docente 2 Mostrar infrações de horário

	Segunda	Terça	Quarta	Quinta	Sexta	Sabado
7						
8						
9						
10						
11						
12						
13			EXM166 33A33B34A34B37A	EXM166 33A		
14			EXM166 33A33B34A34B37A	EXM166 33A		
15			EXM166 31B			
16			EXM166 31B			
17			EXM166 31A			
18			EXM166 31A			
19				EXM135 15A18A		
20				EXM135 15A18A		
21			EXM135 15A18A			
22			EXM135 15A18A			

Figura 3.16 – Tabela de composições de turma

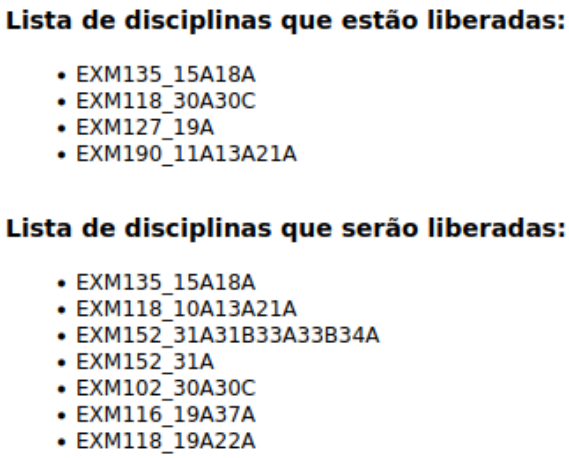
Preferencias atendidas e não atendidas:

Docente 1.	Docente 2.	Docente 3.	Docente 4.	Docente 5.	Docente 6.	Docente 7.	Docente 8.
EXM102 19A	EXM175 15A	EXM191 15A18A24A	EXM102 11A13A21A	EXM190 19A22A	EXM174 9A	EXM118 3A5A	EXM116 19A37A
EXM102 3A5A	EXM166 31A31B32A32B	EXM130 15A18A	EXM102 30A30C	EXM127 19A	EXM174 26A	EXM125 30A30B	EXM195 1A
	EXM166 31B			EXM185 14A	EXM173 15A	EXM125 30C30D	EXM152 31A31B33A33B34A
							EXM152 33A

Docente 9.	Docente 1.0.	Docente 1.1.	Docente 1.2.	Docente 1.3.	Docente 1.4.	Docente 1.5.	Docente 1.6.
EXM116 31A32A	EXM102 AB	EXM190 15A18A	EXM118 30A30C	EXM166 33A33B34A34B37A	EXM102 30E30G	EXM116 11A13A21A	EXM195 12A20A9A
EXM152 32A32B34B37A	EXM116 22A3A	EXM109 15A	EXM190 30A30B		EXM152 34B	EXM123 19A22A22B3A	EXM163 15A
	EXM116 15A18A		EXM140 32A33A		EXM152 31A		
			EXM140 31A34A37A				

Docente 1.7.	Docente 1.8.	Docente 1.9.	Docente 2.0.	Docente 2.1.	Docente 2.2.	Docente 2.3.	Docente 2.4.
EXM118 30E30G	EXM118 19A22A	EXM127 15A18A	EXM161 15A	EXM102 10A22A	EXM118 10A13A21A	EXM190 30C30D	EXM127 11A3A
EXM102 14A15A18A	EXM190 11A13A21A	EXM103 10A15A	EXM166 33A	EXM190 10A3A5A	EXM118 11A30I	EXM116 33A34A	EXM127 10A22A
	EXM195 2A		EXM166 32B				EXM185 10A
	EXM152 32A						

Figura 3.17 – Disciplinas liberadas nesse e no próximo semestre



4 RESULTADOS E DISCUSSÕES

Os testes foram feitos utilizando dados adaptados de um departamento de uma universidade brasileira, que continha 25 docentes e 76 disciplinas. A máquina utilizada para rodar o programa foi um computador com 8GB de RAM, processador Intel Core i5-7200U CPU @ 2.50GHz x 4 e 64 bits, memória principal ATA Disk de 1TB, com o sistema operacional Ubuntu 22.04.2 LTS. A implementação foi feita na linguagem Python 3.10. O código utilizado está disponível em <https://github.com/ElMigu17/TCC>.

Durante a geração do produto, houve dificuldade com relação à factibilidade da solução. Mais especificamente, ocorreu um conflito entre a restrição de prioridade e as restrições de horário, que será explicado na próxima seção. Finalizamos esse capítulo com uma seção sobre os resultados computacionais.

4.1 Incompatibilidade entre prioridades e horários

Durante os testes, percebemos que o algoritmo não encontrava uma solução quando todas as restrições estavam ativas. Investigando o porquê desse comportamento, notou-se que, no contexto atual dos dados reais utilizados, havia incompatibilidade entre a restrição de prioridade (Restrição 2.6) e duas restrições relacionadas a horário (Restrições 2.5 e 2.7).

Como os horários das disciplinas são pouco alterados, por ser um processo que envolve diferentes departamentos e uma reorganização das salas, decidimos por alterar as prioridades para eliminar o conflito. Para isso, foi gerada uma solução em que as regras relacionadas a horários não foram modeladas como restrições, mas sim penalizadas (em caso de infrações, conforme Equação 4.1) na função objetivo. Nesse processo, as preferências que causavam incompatibilidade foram detectadas e alteradas. Dessa forma, uma solução factível foi encontrada, demonstrando a eficácia do algoritmo proposto.

Devido ao risco desse tipo de incompatibilidade ocorrer novamente no futuro, foi decidido que, caso não seja encontrada solução com todas as restrições ativas, o programa tentaria gerar uma solução sem a restrição de prioridade. Caso, ainda assim, nenhuma solução seja encontrada, haveria a tentativa de gerar uma solução sem restrição de prioridade e horário. Caso alguma dessas restrições não sejam utilizadas, isso será informado ao usuário, como pode ser visto na Figura 3.14. Para complementar essas informações, caso a restrição de horário não

seja utilizada, as infrações podem ser resumidas ao selecionar o *checkbox* "Mostrar infrações de horário", conforme a Figura 4.1.

Função de penalização por infração da restrição de horário

$$\{s_1 = (s_1^d, s_1^{hi}, s_1^{hf}), s_2 = (s_2^d, s_2^{hi}, s_2^{hf})\} \in S,$$
$$(((s_1^d = s_2^d + 1) \wedge ((s_1^{hf} = 22:40) \wedge (s_2^{hi} < 10:00))) \vee$$
$$((s_2^d = s_1^d + 1) \wedge ((s_2^{hf} = 22:40) \wedge (s_1^{hi} < 10:00)))) \vee$$
$$((s_1^d = s_2^d) \wedge (((s_1 \in S_m) \wedge (s_2 \in S_n)) \vee ((s_1 \in S_n) \wedge (s_2 \in S_m))))) \rightarrow$$
$$z = z - (x[p][s_1] * x[p][s_2]), \forall p \in P \quad (4.1)$$

A otimização toma as infrações de restrição de horário que podem ter ocorrido e subtrai o peso delas da função objetivo. Essa parte da função objetivo pode ser dividida em duas partes: a primeira é a mesma que a restrição de intervalo de expediente 2.5 e a segunda é a mesma que a restrição de turnos 2.7.

Figura 4.1 – Visualizando infrações de horários

Escolha um professor: Docente 2*

Mostrar infrações de horário

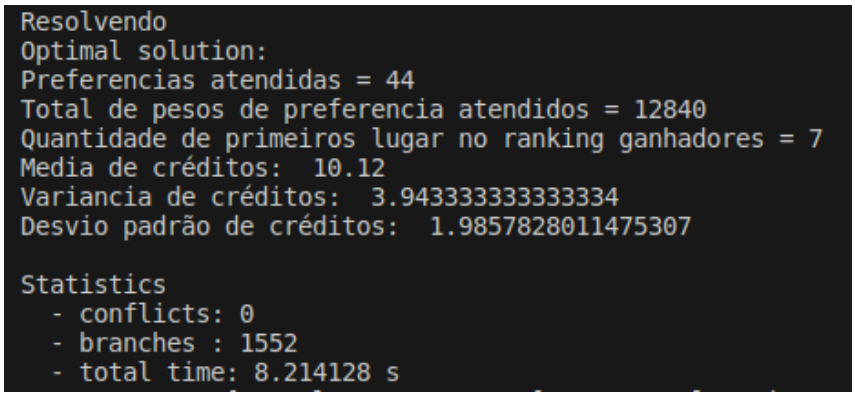
	Segunda	Terça	Quarta	Quinta	Sexta	Sabado
7						
8				EXM166 32A		
9				EXM166 32A		
10						
11						
12						
13			EXM166 33A33B34A34B37A			
14			EXM166 33A33B34A34B37A			
15			EXM166 31B			
16			EXM166 31B			
17						
18						
19				EXM135 15A18A		
20				EXM135 15A18A		
21			EXM135 15A18A			
22			EXM135 15A18A			

4.2 Resultados computacionais

Neste trabalho, foi proposto um algoritmo que produz (com base em 10 rodadas do algoritmo), em cerca de 10 segundos, soluções factíveis e que respeitam a todas as restrições

departamentais e trabalhistas do contexto de distribuição de composições de turmas a professores, em um dado contexto universitário. Além disso, é uma solução que atende a mais da metade das preferências inseridas pelos docentes, com solução ótima para a função objetivo. A Figura 4.2 resume os dados de saída do algoritmo.

Figura 4.2 – Resultados computacionais



5 CONCLUSÕES E TRABALHOS FUTUROS

O objetivo desse projeto foi criar um produto que pudesse ser utilizado por um departamento universitário durante a distribuição das composições de turmas para os docentes. Para ser satisfatório, era necessário que o algoritmo respeitasse as restrições, fosse gerado em tempo hábil e conseguisse atender, da melhor forma possível, as preferências dos docentes. Além disso, era fundamental a criação de uma interface de fácil uso para que a dificuldade de utilizar a ferramenta não fosse um empecilho.

A dificuldade apresentada pela incompatibilidade entre as restrições de prioridade e de horário foi superada, de modo que as soluções geradas atendessem corretamente aos requisitos do contexto usado como base. Para isso, foi usado o *CP-Optimizer* da biblioteca *Google OR-Tools* em um modelo de *Constraint Programming*. Com isso, além de obedecer às restrições, a geração de soluções também possui um certo nível de eficiência. Exemplo dessa eficiência é o fato de a solução gerada para testes ter tido um tempo total de processamento de menos de 10 segundos, com 57.89% das composições de turmas distribuídas para professores que as preferiam.

A interface apresenta espaço para adequações. Como sugestão para trabalhos futuros, propomos a validação da interface junto aos usuários, fazendo um estudo de *design* para melhor adequá-la às suas necessidades. Por fim, uma outra ideia bastante relevante é a adaptação do algoritmo para gerar soluções para outros departamentos. Acreditamos que, com este exemplo já feito, será mais simples criar um novo produto para um contexto semelhante.

REFERÊNCIAS

BARDADYM, V. A. Computer-aided school and university timetabling: The new wave. In: BURKE, E.; ROSS, P. (Ed.). **Practice and Theory of Automated Timetabling**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 22–45. ISBN 978-3-540-70682-3.

CARTER, M. W.; LAPORTE, G. Recent developments in practical course timetabling. In: **Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II**. Berlin, Heidelberg: Springer-Verlag, 1997. (PATAT '97), p. 3–19. ISBN 3540649794.

DEMIROVIĆ, E.; STUCKEY, P. J. Constraint programming for high school timetabling: A scheduling-based model with hot starts. In: HOEVE, W.-J. van (Ed.). **Integration of Constraint Programming, Artificial Intelligence, and Operations Research**. Cham: Springer International Publishing, 2018. p. 135–152. ISBN 978-3-319-93031-2.

HARTMANIS, J. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). **SIAM Review**, v. 24, n. 1, p. 90–91, 1982. Disponível em: <https://doi.org/10.1137/1024022>.

KARP, R. Reducibility among combinatorial problems. In: . [S.l.: s.n.], 1972. v. 40, p. 85–103. ISBN 978-3-540-68274-5.

MüLLER, D. et al. An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. **European Journal of Operational Research**, v. 302, n. 3, p. 874–891, 2022. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221722000728>.

SAHNI, S.; GONZALEZ, T. P-complete approximation problems. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 23, n. 3, p. 555–565, jul 1976. ISSN 0004-5411. Disponível em: <https://doi.org/10.1145/321958.321975>.

SMITH, B. M. A tutorial on constraint programming. University of Leeds, School of Computer Studies, 1995.

TANG, Y. et al. Scheduling optimization of linear schedule with constraint programming. **Computer-Aided Civil and Infrastructure Engineering**, v. 33, n. 2, p. 124–151, 2018. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12277>.

WANG, T.; MESKENS, N.; DUVIVIER, D. Scheduling operating theatres: Mixed integer programming vs. constraint programming. **European Journal of Operational Research**, v. 247, n. 2, p. 401–413, 2015. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221715005226>.