CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS DIVINÓPOLIS

Wallace Ketler Melo de Morais

GERAÇÃO AUTOMÁTICA DE HORÁRIOS EDUCACIONAIS: UM ESTUDO DE CASO DO CEFET-MG, CAMPUS DIVINÓPOLIS

Divinópolis-MG
2023

WALLACE KETLER MELO DE MORAIS

GERAÇÃO AUTOMÁTICA DE HORÁRIOS EDUCACIONAIS: UM ESTUDO DE CASO DO CEFET-MG, CAMPUS DIVINÓPOLIS

Trabalho de Conclusão de Curso apresentado no curso de Graduação em Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. André Luiz Maravilha Silva

DIVINÓPOLIS-MG

2023

WALLACE KETLER MELO DE MORAIS

GERAÇÃO AUTOMÁTICA DE HORÁRIOS EDUCACIONAIS: UM ESTUDO DE CASO DO CEFET-MG, CAMPUS DIVINÓPOLIS

Trabalho de Conclusão de Curso apresentado no curso de Graduação em Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Aprovado em 8 de fevereiro de 2024.

Título Nome

Instituição

Título Nome

Instituição

Título Nome

Instituição

Dedico à minha família, que me manteve em tantos aspectos durante toda minha vida.

AGRADECIMENTOS

Agradeço aos professores e colegas de sala que, durante toda graduação, se mantiveram animados e aptos a ajudarem com quaisquer que sejam as dificuldades enfrentadas.

“É preciso ter um caos dentro de si para dar à luz uma estrela cintilante”

Friedrich Nietzsche

RESUMO

A alocação de horários é um problema de otimização combinatória que faz parte do conjunto de problemas NP-difícil. Desta forma, instituições públicas de ensino que demandam a constante geração semestral de horários, necessitam de uma opção gratuita e eficiente de alocação de turmas. Portanto, o trabalho propõe uma aplicação *desktop*, por meio dos *frameworks Vue* e *Electron*, que integra um algoritmo heurístico baseado no *Simulated Annealing*, implementado em C++ para uma maior eficiência. Desta forma, é possível comparar numericamente o valor da função objetivo gerada por esta solução com a solução gerada pelo software proprietário adquirido pelo Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), dadas as restrições escolhidas e implementadas especificamente para atender as demandas desta instituição. Com isso, dadas as execuções realizadas, foi possível se aproximar do valor da função objetivo do software gerado pelo CEFET-MG, o que denota a qualidade das abordagens heurísticas para o problema de alocação de horários.

Palavras-chave: Programação de horários educacionais; Heurísticas; *Simulated Annealing*; Otimização.

ABSTRACT

Schedule allocation is a combinatorial optimization problem that is part of the NP-hard problem set. Therefore, public educational institutions that require constant semesterly schedule generation need a free and efficient option for class allocation. Thus, this work proposes a desktop application using the Vue and Electron frameworks, integrating a heuristic algorithm based on Simulated Annealing, implemented in C++ for increased efficiency. In this way, it is possible to numerically compare the objective function value generated by this solution with the solution generated by the proprietary software acquired by the Federal Center for Technological Education of Minas Gerais (CEFET-MG), given the constraints chosen and specifically implemented to meet the demands of this institution. Consequently, based on the performed executions, it was possible to approach the objective function value of the software generated by CEFET-MG, indicating the quality of heuristic approaches for the schedule allocation problem.

Keywords: Educational schedule programming; Heuristics; Simulated Annealing; Optimization.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diagrama de casos de uso 34
Figura 2 – Página inicial da aplicação 35
Figura 3 – Tutorial de uso 35
Figura 4 – Erro ao gerar horário 36
Figura 5 – Horário gerado com possibilidade de exportação em PDF 36
Figura 6 – Organização dos horários disponíveis 37
Figura 7 – Organização de professores disponíveis 38
Figura 8 – Cadastro de professores 38
Figura 9 – Confirmação de remoção 39
Figura 10 – Gerência de classes 39
Figura 11 – Gerência de disciplinas 40
Figura 12 – Gerência de salas 40
Figura 13 – Atualização cadastral de disciplina 41
Figura 14 – Gráfico do valor das funções objetivo iniciais 43
Figura 15 – Gráfico de funções objetivo finais 43

LISTA DE TABELAS

Tabela 1 – Tabela de trabalhos 25
Tabela 2 – Tabela de quantidade de itens na instância do segundo semestre de 2023 42
Tabela 3 – Resultados 42

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Considerações Iniciais 1
1.2 Objetivos do Trabalho 2
1.2.1 Objetivo geral 2
1.2.2 Objetivos específicos 2
1.3 Contribuições 3
1.4 Estrutura do Texto 3
2 REFERENCIAL TEÓRICO 4
2.1 Programação Linear 4
2.2 Otimização Combinatória 5
2.3 Estratégias para resolver problemas de otimização combinatória . . . 6
2.3.1 Algoritmos Exatos 7
2.3.2 Algoritmos Aproximativos 7
2.3.3 Algoritmos Heurísticos 8
2.3.3.1 Heurísticas Construtivas 10
2.3.3.2 Heurísticas de Busca Local 10
2.3.3.3 Meta-Heurísticas 12
2.4 Função objetivo 13
2.5 Simulated Annealing 13
2.6 Considerações Finais 15
3 REFERENCIAL BIBLIOGRÁFICO 17
3.1 Algoritmos da literatura 17
3.1.1 Busca Tabu 17
3.1.2 *Simulated Annealing* 18
3.1.3 *Greedy Randomized Adaptative Search Procedure* 18
3.1.4 *Iterated Local Search* 19
3.1.5 *Variable Neighborhood Search* 20
3.2 Restrições usadas na Literatura 21
4 O PROBLEMA DE ALOCAÇÃO DE HORÁRIOS 26
4.1 Introdução ao problema 26

4.2 Definição do Problema e Formulação Matemática 26
4.2.1 Função objetivo 29
4.3 Discussões finais 30
5 METODOLOGIA 31
5.1 Considerações Iniciais 31
5.2 Coleta e armazenamento de dados 31
5.3 Restrições adotadas 31
5.3.1 Função de Vizinhança 32
5.4 Planejamento Experimental 33
5.5 Aplicação Desktop 33
6 RESULTADOS 34
6.1 Projeto e Implementação do Software 34
6.2 Soluções Alcançadas 41
7 CONCLUSÃO 46
REFERÊNCIAS 47

1 INTRODUÇÃO

1.1 Considerações Iniciais

A otimização e a pesquisa operacional abordam problemas de minimização ou maximização de recursos em geral e se definem como um ramo tradicional que assimila diversas técnicas de modelagem matemática (GOLDBARG, 2005). Tal modelagem, se baseia, essencialmente, na tentativa de satisfazer critérios pré-definidos de qualidade para a geração de uma solução aceitável (BELLIO et al., 2016).

Com o crescente avanço no estudo de algoritmos e técnicas de pesquisa operacional, é possível a aplicação de seus métodos nas mais diversas áreas. Referente a isso, a implementação de técnicas de pesquisa operacional pode ser exemplificada em projetos e manutenção de redes elétricas ao redor do mundo (STEINER et al., 2006), processos manufaturados em geral (PAIVA, 2008), mercado de capitais (SAMPAIO et al., 2016), minereração de ferro (BARBOSA; MAPA, 2017), otimização de rotas rodoviárias (BARREIRA, 2016) e no fluxo de pacientes em hospitais (SCARPIN et al., 2007).

Dentre as áreas de aplicação da pesquisa operacional, o problema de alocação de horários (no inglês, *Timetabling*) possui potencial para a exploração de técnicas de otimização e consiste na criação de soluções viáveis para a disposição de períodos de tempo e recursos. Essa classe de problemas engloba desde a alocação de jogos esportivos em estádios e horários, bem como a alocação de salas para avaliações e aulas escolares em determinados períodos de tempo.

No contexto da otimização de horários universitários se encontra como uma das vertentes do problema de *Timetabling* e se caracteriza como a pesquisa por um método de alocação de eventos em salas e horários pré-definidos (BABAEI; KARIMPOUR; HADIDI, 2015). Complementarmente, a disposição de horários para exames em geral, descrito como uma série de avaliações alocadas em diversos períodos de tempo (ARBAOUI; BOUFFLET; MOUKRIM, 2019) se mostra um problema recorrente na área.

Sobre tal tema, a dimensão do tempo como ferramenta de trabalho do estudante universitário e do professor de determinada instituição rege, muitas vezes, a capacidade física e mental na produção de um trabalho de qualidade (SILVA, 2019). Dessa forma, a otimização do tempo em instituições de ensino se mostra um fator determinante na

qualidade do ensino tanto no espectro da sequência adequada de disciplinas quanto na economia de tempo para o discente e docente.

Ainda assim, há de se atentar para o fato de que em muitas instituições utiliza-se o método manual para a elaboração dos horários, o que demanda tempo e recursos em geral. Simultaneamente, dentre o número de configurações de horário possíveis, a maior parte não atende às restrições e aspectos desejados, o que demanda um tempo considerável para obtenção de soluções viáveis. Portanto, a alocação adequada dos horários favorece todos os âmbitos de um centro educacional.

1.2 Objetivos do Trabalho

1.2.1 Objetivo geral

Busca-se, enquanto objetivo geral desse trabalho, disponibilizar uma ferramenta baseada em otimização para alocar horários de disciplinas ofertadas pelo Centro Federal de Educação Tecnológica de Minas Gerais (Campus Divinópolis) para um dado semestre. Para tanto, implementou-se um software que constrói, de forma automática, quadros de horários seguindo indicadores de qualidade.

1.2.2 Objetivos específicos

Para que o objetivo geral do trabalho seja alcançado, existe a necessidade do cumprimento de alguns objetivos específicos, descritos como:

a) Projetar um algoritmo de otimização para *timetabling* considerando aspectos demandados pelo CEFET-MG;
b) Coletar e usar dados reais do CEFET-MG, Campus Divinópolis;
c) Verificar a viabilidade do algoritmo escolhido;
d) Fazer uma análise comparativa que permita verificar numericamente a qualidade da solução apresentada;
e) Implementar um software *desktop* para uso do método proposto.

1.3 Contribuições

Este trabalho busca contribuir na verificação da aplicabilidade de um algoritmo de otimização conhecido na literatura em um problema real com restrições específicas do CEFET-MG, Campus Divinópolis. Ainda assim, a implementação do aplicativo *desktop* pode permitir uma maior facilidade para o CEFET-MG, na geração de horários escolares, de nível técnico e superior em comparação com uma geração manual. Logo, tende-se a produzir uma alternativa à maneira com que se gera os horários na atualidade do Campus.

1.4 Estrutura do Texto

O presente capítulo buscou contextualizar o tema abordado neste trabalho, de modo que possibilite ao leitor entender a importância, bem como a motivação para a otimização da alocação de horários educacionais.

Subsequentemente, o Capítulo 2, Referencial Teórico, fornece definições a respeito da pesquisa operacional, otimização combinatória e algoritmos usados na literatura. A partir disso, é possível o aprofundamento no tema tratado.

No Capítulo 3 é feita uma revisão da literatura, a fim de fornecer uma visão de como a pesquisa na área de *Timetabling* se encontra, de modo que sejam exploradas condições e soluções usadas.

Uma vez que os conceitos técnicos são definidos, o Capítulo 4, explica e explora com maiores detalhes, o problema de alocação de horários educacionais.

Após a revisão da literatura e a contextualização do problema, o Capítulo 5 apresenta a metolodogia utilizada para a execução deste trabalho. Para tanto, é descrito em detalhes as atividades realizadas.

Complementarmente, a fim de explicitar o que foi desenvolvido, o Capítulo 6, Resultados, demonstra tanto a estética final da aplicação *desktop* como o resultado dos experimentos executados no algoritmo *Simulated Annealing*.

Por fim, o Capítulo 7 apresenta as considerações finais e as sugestões de trabalhos futuros que podem ser desenvolvidos.

2 REFERENCIAL TEÓRICO

Este capítulo serve como base para as informações que serão apresentadas adiante no trabalho. Ele desempenha um papel crucial ao estabelecer o suporte teórico necessário para compreensão dos conteúdos subsequentes. Para tanto, serão contemplados assuntos como programação linear, otimização combinatória e estratégias para a resolução de problemas de otimização.

2.1 Programação Linear

Um problema de otimização pode ser expresso genericamente como:

$$\begin{aligned} &\text{Minimizar } f(x) \\ &\text{sujeito a: } g(x) \geq b \\ &x \in \mathbb{R}^n \end{aligned} \tag{2.1}$$

De modo que $f : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo que deve ser otimizada, $g : \mathbb{R}^n \to \mathbb{R}^m$ é uma função de restrições, $b \in \mathbb{R}^m$ é um vetor coluna dos termos independentes das restrições e $x \in \mathbb{R}^n$ é um vetor coluna de variáveis de decisão. Sobre isso, os conjuntos de valores x que satisfazem todas as restrições são denominadas pontos viáveis ou soluções viáveis. O conjunto de todas soluções viáveis constitui a região viável ou espaço de soluções do problema (BAZARAA; JARVIS; SHERALI, 2009).

Um problema de programação linear (PL) constitui um tipo especial do problema de otimização em que f e g são lineares em x (BAZARAA; JARVIS; SHERALI, 2009). Um PL, em sua forma matricial canônica, é escrito como:

$$\begin{aligned} &\text{Minimizar } c^T x \\ &\text{sujeito a: } Ax \geq b \\ &x \geq 0 \end{aligned} \tag{2.2}$$

De modo que $A \in \mathbb{R}^{m \times n}$ é a matriz de restrições e $c \in \mathbb{R}^n$ é um vetor coluna de coeficientes de custos. Complementarmente, sobre a natureza de um problema de programação linear de minimização, sua forma é canônica se todas restrições são do tipo $\geq$ e todas variáveis são não negativas, de modo que qualquer problema de programação linear pode ser escrito

desta forma (BAZARAA; JARVIS; SHERALI, 2009).

Uma vantagem de modelos de programação linear está na eficiência de algoritmos de solução existentes, como algoritmo Simplex de Dantzig, proposto em 1947 (DANTZIG, 1963). Este algoritmo, apesar de ter complexidade exponencial no pior caso, é usualmente bastante rápido na prática (CORMEN et al., 2009).

Se for adicionada a restrição de que todas as variáveis devem assumir valores inteiros, tem-se um problema de programação linear inteira (PLI), escrito como:

$$\begin{gathered} \text{Minimizar } c^T x \\ \text{sujeito a: } Ax \geq b \\ x \geq 0 \text{ e } x \in \mathbb{Z}^n \end{gathered} \tag{2.3}$$

Mas, se apenas algumas, e não todas, variáveis são restritas a valores inteiros, tem-se um problema de programação linear inteira mista (PLIM), escrito como:

$$\begin{gathered} \text{Minimizar } c^T x + h^T y \\ \text{sujeito a: } Ax + Gy \geq b \\ x \geq 0 \\ y \geq 0 \text{ e } y \in \mathbb{Z}^p \end{gathered} \tag{2.4}$$

Onde $y \in \mathbb{Z}_+^p$ é um vetor coluna de variáveis inteiras, $G \in \mathbb{R}^{m \times p}$ é a matriz de restrições associada às variáveis inteiras e $h \in \mathbb{R}^p$ é um vetor coluna de coeficientes de custo.

No caso do problema de programação linear inteira, se todas as variáveis são restritas ao conjunto de valores $\{0, 1\}$, tem-se um caso especial, denominado problema de programação linear inteira binária (PLIB), escrito como:

$$\begin{gathered} \text{Minimizar } c^T x \\ \text{sujeito a: } Ax \geq b \\ x \in \{0, 1\}^n \end{gathered} \tag{2.5}$$

2.2 Otimização Combinatória

Dado um conjunto finito e discreto $\mathbb{E} = \{e_1, e_2, ..., e_n\}$, o conjunto $P(\mathbb{E})$ é o conjunto contendo todos os subconjuntos possíveis de $\mathbb{E}$. Um subconjunto $\mathbb{X} \subseteq \mathbb{E}$ é viável se,

e somente se, $\mathbb{X}$ satisfaz todas as restrições impostas pelo problema que se propõe a resolver, formando assim, um conjunto de soluções viáveis para o problema.

Assim, o espaço de soluções de um problema de otimização combinatória é definido sobre o conjunto de subconjuntos viáveis (WOLSEY, 1998), escrito como:

$$\chi = \{\mathbb{X} \in P(\mathbb{E}) : \mathbb{X} \text{ é viável}\} \tag{2.6}$$

O conjunto χ é finito e discreto, portanto, é um conjunto enumerável. Considerando uma função objetivo $f : \chi \rightarrow \mathbb{R}$, um problema de otimização combinatória pode ser escrito como:

$$\text{Encontrar } \mathbb{X}^* = \arg\min\{f(\mathbb{X}) : \mathbb{X} \in \chi\} \tag{2.7}$$

É relevante constatar que problemas de otimização combinatória podem ser escritos como PLIM e consequentemente, fazer uso do arcabouço de técnicas de resolução de PLIM.

2.3 Estratégias para resolver problemas de otimização combinatória

Como o espaço de soluções em problemas de otimização combinatória é um conjunto enumerável, uma estratégia simplista para resolver problemas desta classe é enumerar e avaliar todas as possíveis soluções e assim, determinar a solução ótima.

Entretanto, para o problema do caixeiro viajante, que se trata de um clássico problema de otimização que objetiva que um viajante visite um conjunto de cidades apenas uma vez, retornando à cidade de origem com a menor rota possível, por exemplo, existem $n!$ permutações possíveis, porém, como vetores rotacionados representam a mesma solução, existem $(n-1)!$ soluções diferentes e viáveis. Para uma instância com n = 101 cidades, esse valor é de aproximadamente $9,33 \times 10^{157}$ (WOLSEY, 1998).

Este rápido crescimento de soluções possíveis é chamado de *explosão combinatória*. Enumerar todas as possíveis soluções para resolver problemas de otimização combinatória é viável apenas para problemas de pequena dimensão. Com isso, são necessárias estratégias mais eficientes para resolver problemas de grandes dimensões (WOLSEY, 1998).

2.3.1 Algoritmos Exatos

Algoritmos exatos garantem encontrar a solução ótima para um problema de otimização. Como o número de soluções em um problema de otimização combinatória cresce exponencialmente em função de seu tamanho, a completa enumeração das soluções se torna impraticável (WOLSEY, 1998). Logo, para a implementação, deve-se evitar a enumeração completa do conjunto de soluções sem perder a garantia de encontrar a solução ótima.

Dentre os algoritmos exatos, os mais usados são: *Branch-and-Bound*, Plano de Corte, *Branch-and-Cut*, Geração de Colunas, *Branch-and-Price* e *Branch-and-Cut-and-Price* (WOLSEY, 1998).

2.3.2 Algoritmos Aproximativos

Para problemas pertencentes à classe NP-Difícil, não é conhecido nenhum algoritmo em máquina determinística que garanta encontrar a solução ótima com complexidade de tempo polinomial em função do tamanho do problema. De fato, tal algoritmo só existe se $P = NP$, que é uma questão ainda em aberto (CORMEN et al., 2009).

Se $P \neq NP$, então não podemos ter um algoritmo capaz de garantir, simultaneamente, que (i) a solução ótima é encontrada (ii) com complexidade de tempo polinomial (iii) para qualquer instância. Ao menos um destes requisitos deve ser relaxado ao lidar com problemas NP-difíceis (WILLIAMSON; SHMOYS, 2011).

Algoritmos que relaxam o requisito (iii) são projetados para casos específicos de algum problema de otimização. Assim, seu uso é bastante restrito, podendo não funcionar adequadamente ou até mesmo não funcionar para as demais instâncias do problema.

Os algoritmos exatos relaxam o requisito (ii). Ou seja, eles abrem mão de uma complexidade de tempo polinomial para garantir a solução ótima em qualquer instância. No entanto, ao se tratar de problemas de otimização de difícil solução ou problemas de grande porte, o tempo demandado por estes algoritmos é impraticável. Assim, os algoritmos exatos ficam restritos a problemas mais simples e/ou de dimensões reduzidas.

A opção restante é relaxar o requisito (i), no qual se abre mão da solução ótima

do problema. É nesta abordagem em que os algoritmos heurísticos e os aproximativos de tempo polinomial se encontram.

Algoritmos α-aproximados são algoritmos com complexidade de tempo polinomial, que, para todas instâncias de um problema, gera uma solução cujo valor da função objetivo satisfaça uma razão α em relação ao valor da função objetivo da solução ótima (WILLIAMSON; SHMOYS, 2011).

Desta forma, o objetivo dos algoritmos aproximativos é de relaxar o requisito de garantia de otimalidade o mínimo possível (WILLIAMSON; SHMOYS, 2011). O fator α é denominado fator de aproximação e diz respeito à qualidade esperada, no pior caso, das soluções geradas pelo algoritmo.

2.3.3 Algoritmos Heurísticos

O termo heurística deriva da palavra grega *heurisken* que significa encontrar ou descobrir. Na área de otimização, a palavra heurística é utilizada para caracterizar os algoritmos de otimização que fazem uso de informações específicas ou informações obtidas pelo próprio algoritmo para decidir qual próxima solução candidata deve ser testada ou como uma solução deve ser construída (GOLDBARG; LUNA, 2005).

Diferente dos algoritmos exatos que garantem encontrar a solução ótima e dos algoritmos aproximativos que dão uma garantia de qualidade para o pior caso, os algoritmos heurísticos não dão nenhuma garantia de qualidade da solução gerada (MARTÍ; REINELT, 2011). Contudo, o tempo demandado pelos algoritmos heurísticos pode ser muito menor em ordens de magnitude em relação ao tempo demandado pelos algoritmos exatos, principalmente quando se trata da solução de problemas NP-Difíceis (MARTÍ; REINELT, 2011).

A fim de diferenciar algoritmos aproximativos e algoritmos heurísticos, define-se uma heurística como uma técnica específica para um problema de otimização, ou classe de problemas de otimização, utilizada no projeto de algoritmos, os quais não é possível garantir, ou não é conhecida nenhuma garantia da qualidade das soluções geradas por esses algoritmos. Tais algoritmos são denominados algoritmos heurísticos (WEISE, 2009).

Apesar da definição adotada especificar que uma heurística é um componente utilizado no projeto de algoritmos heurísticos, neste trabalho, o termo heurística será

utilizado como sinônimo de algoritmo heurístico.

Como as heurísticas não tem nenhuma garantia de qualidade das soluções por elas geradas, elas podem ter um desempenho muito ruim se o pior caso for considerado, retornando soluções muito distantes da solução ótima. No entanto, heurísticas bem projetadas superam até mesmo os melhores algoritmos aproximativos (AUSIELLO et al., 1999). Complementarmente, uma boa heurística deve contemplar as seguintes características (MARTÍ; REINELT, 2011):

a) Uma solução deve ser obtida com um esforço computacional aceitável.
b) Considerando o desempenho médio da heurística, a solução retornada deve ser próxima à solução ótima.
c) A probabilidade da heurística produzir soluções ruins (distantes da solução ótima) deve ser baixa.

Além da necessidade de algoritmos que, em tempo razoável, sejam capazes de encontrar boas soluções para problemas de otimização de grande porte ou de difícil solução, outras razões que levam ao uso de heurísticas para um dado problema são (MARTÍ; REINELT, 2011):

a) Não é conhecido nenhum algoritmo exato capaz de encontrar a solução ótima do problema.
b) Embora seja conhecido algum algoritmo exato para o problema, nos computadores atuais, tal algoritmo demanda um tempo impraticável para encontrar a solução ótima.
c) As heurísticas são mais flexíveis que os métodos exatos, permitindo, por exemplo, a incorporação de condições difíceis de modelar.
d) As heurísticas podem ser utilizadas como parte de um procedimento global que garanta a solução ótima.

Existem diferentes heurísticas, e assim, é difícil fornecer uma classificação completa que abranja todos os tipos de heurísticas. As subseções seguintes classificam as heurísticas em três grupos: heurísticas construtivas, heurística de busca local e meta-heurísticas.

2.3.3.1 Heurísticas Construtivas

Heurísticas construtivas são algoritmos de otimização que derivam uma única solução viável. Problemas de otimização combinatória são formados por um subconjunto de um conjunto de possíveis elementos candidatos a formar a solução. Assim, heurísticas construtivas para problemas de otimização combinatória geralmente constroem uma solução a partir de um fragmento de uma solução (geralmente um conjunto vazio) e, iterativamente, adicionando elementos ao conjunto ou combinando subconjuntos até que uma solução completa e viável seja obtida (MARTÍ; REINELT, 2011). É fácil notar que heurísticas construtivas possuem uma estrutura simples e são fortemente dependentes do problema que se propõem a resolver requerendo geralmente, um tempo polinomial (AUSIELLO et al., 1999).

Muitas vezes, as heurísticas construtivas são determinísticas e a escolha dos componentes é baseada na melhor alternativa para a iteração. As heurísticas construtivas com essa propriedade são muitas vezes denominadas heurísticas gananciosas ou gulosas.

As heurísticas construtivas são amplamente utilizadas na otimização combinatória, podendo retornar diretamente a solução por ela gerada, mas, mais frequentemente, tais soluções são utilizadas como soluções de partida para heurísticas mais robustas, como as heurísticas de busca local.

2.3.3.2 Heurísticas de Busca Local

Diferentemente das heurísticas construtivas, as heurísticas de busca local iniciam o processo de otimização a partir de uma solução viável já conhecida e tentam, iterativamente, melhorá-la (MARTÍ; REINELT, 2011). Antes de definir uma heurística de busca local, é necessário definir os seguintes conceitos:

a) ***Função de vizinhança:*** Dado χ o conjunto de soluções viáveis de um problema de otimização combinatória, $P(\chi)$ é o conjunto de todos os subconjuntos de χ. Uma função de vizinhança é uma função $n : \chi \rightarrow P(\chi)$ que, para cada solução viável $\mathbb{B} \in \chi$ associa um único conjunto de soluções vizinhas, ou vizinhança, $N_{\mathbb{B}} \in P(\chi)$. Em outras palavras, a função de vizinhança pode ser definida como

uma maneira de se explorar as soluções.

b) ***Estrutura de vizinhança:*** Uma estrutura de vizinhança é uma estrutura topológica sobre χ induzida por uma função de vizinhança $n : \chi \rightarrow P(\chi)$ que pode ser descrita como um grafo direcionado $G = \{\mathbb{V}, \mathbb{A}\}$, onde $\mathbb{V} = \chi$ é o conjunto de vértices e $\mathbb{A} = \{(\mathbb{X}_i, \mathbb{X}_j) \in \chi \times \chi : \mathbb{X}_j \in n(\mathbb{X}_i)\}$. Textualmente, pode-se definir a estrutura de vizinhança como a forma com que as soluções estão dispostas em relação às demais.

c) ***Ótimo local:*** Dada uma função objetivo $f : \chi \rightarrow \mathbb{R}$ para um problema de minimização, sem perda de generalidade, uma solução $\mathbb{X}_i \in \chi$ é dita ser ótimo local para uma função de vizinhança n se, e somente se, não existe $\mathbb{X}_j \in n(\mathbb{X}_i)$ tal que $f(\mathbb{X}_j) < f(\mathbb{X}_i)$. De maneira semelhante, um ótimo local pode ser expresso como determinada solução que, dada as funções de vizinhança, não é possível se mover para uma solução vizinha melhor que a atual, o que não significa, entretanto, que tal solução é a melhor entre todas as possíveis.

Uma heurística de busca local percorre, iterativamente, uma estrutura de vizinhança a partir de uma solução inicial $\mathbb{X}_0 \in \chi$ onde, a cada iteração ela se move da solução corrente $\mathbb{X}_i$ para uma outra solução $\mathbb{X}_j \in n(\mathbb{X}_i)$ tal que $f(\mathbb{X}_j) < f(\mathbb{X}_i)$, encerrando a busca quando a solução corrente for ótima local (AUSIELLO et al., 1999).

Para uma determinada instância de um problema de otimização combinatória, o desempenho de uma heurística de busca local depende de três aspectos: da função de vizinhança, da solução inicial e da estratégia de escolha da solução para a qual a heurística deve se mover (AUSIELLO et al., 1999).

Apesar de permitir a obtenção de soluções melhores a partir de uma solução já conhecida, as heurísticas de busca local podem ficar presas em uma solução ótima local que não necessariamente é a solução ótima global do problema de otimização. No entanto, as heurísticas de busca local juntamente com as heurísticas construtivas formam a base para as meta-heurísticas, estratégias para a construção de heurísticas mais robustas, apresentadas a seguir.

2.3.3.3 Meta-Heurísticas

Meta-heurísticas são algoritmos de propósito geral que coordenam o uso de procedimentos de melhorias locais, como as heurísticas de busca local, juntamente com estratégias de alto nível para a exploração do espaço de busca com o propósito de projetar heurísticas capazes de escapar de soluções que são ótimos locais e, ao mesmo tempo, realizar uma busca robusta pelo espaço de soluções (GLOVER; KOCHENBERGER, 2003).

O desenvolvimento de meta-heurísticas tem avançado rapidamente e, com isso, inúmeras metodologias tem surgido ao longo dos últimos anos. Apesar de ser um assunto controverso, algumas meta-heurísticas baseadas em população podem ser vistas como extensões ou simplificações de meta-heurísticas mais gerais, ou em alguns casos, sendo exatamente iguais a meta-heurísticas anteriormente propostas, alterando apenas a metáfora utilizada na sua descrição (MARTÍ; REINELT, 2011). As meta-heurísticas podem ser classificadas basicamente em duas classes (OSMAN, 2003):

a) ***Meta-heurísticas baseadas em busca local:*** Essa classe de meta-heurísticas geralmente lida com uma única solução por vez, mapeando diversas soluções ótimas locais através de algoritmos heurísticos de busca local ao longo da sua execução. Alguns exemplos são: métodos *multi-start*, *Greedy Randomize Adaptive Search Procedure* (GRASP), *Simulated Annealing* (SA), *Tabu Search* (TS), *Iterated Local Search* (ILS) e *Variable Neighborhood Search* (VNS).
b) ***Meta-heurísticas baseadas em população:*** Essa classe de meta-heurísticas geralmente tem inspiração na natureza e lidam com um conjunto de soluções simultaneamente, gerando, a cada iteração, um novo conjunto de soluções (possivelmente melhores) utilizando, de alguma forma, informações das soluções de conjuntos anteriores. Alguns exemplos são: Algoritmos Genéticos, *Scatter Search*, Algoritmos de Estimação de Distribuição, Otimização por Enxame de Partículas e Otimização por Colônia de Formigas.

2.4 Função objetivo

A função objetivo no problema de geração de horários escolares, trata como restrições rígidas qualquer quesito pedagógico fundamental para o funcionamento da instituição, logo, devem possuir um maior peso em relação às restrições suaves (BELLIO et al., 2016). Por outro lado, as restrições suaves são assimiladas como questões relevantes pedagogicamente, mas não essenciais para o funcionamento da instituição de ensino. Desta forma, a função objetivo pode ser escrita como:

$$f = \sum w_s \times R_s + \sum w_r \times R_r \tag{2.8}$$

Na qual w_s e R_s representam, respectivamente os pesos e as restrições suaves e o w_r e R_r representam, em ordem, os pesos e as restrições rígidas.

2.5 Simulated Annealing

O *Simulated Annealing*, algoritmo utilizado neste trabalho, é uma técnica de inteligência computacional descrita como uma meta-heurística. Desta forma, busca soluções boas, mas não necessariamente ótimas (SOUZA, 2008). Tal algoritmo se baseia em propriedades da termodinâmica e simula o recozimento de materiais. Para tanto, o método utiliza da probabilidade para exemplificar que, a medida que determinado objeto se esfria, menos ele se altera e se torna maleável. Semelhantemente, no início, quando a temperatura simulada é considerada alta, a probabilidade de alterações também se mostra elevada, o que evita mínimos locais, mas a medida que o tempo de execução do algoritmo passa, menos alterações são permitidas, o que evita movimentações que piorem a solução (SOUZA, 2008).

A partir de uma solução inicial s qualquer gerada aleatoriamente, exploramos a vizinhança s'. Tal exploração se dá por meio da comparação de um valor, de acordo com a Equação (2.9) (AMARAL; PAIS, 2016). Quanto à solução inicial, sua geração inicial ser aleatória não prejudica a execução do algoritmo, uma vez que o peso das restrições rígidas é alto, logo, as soluções não factíveis são altamente penalizadas e não se manterão por muito tempo.

$$\Delta = f(s') - f(s) \tag{2.9}$$

Caso Δ seja menor que 0, a nova solução é considerada melhor e automaticamente assume a solução atual, caso contrário, a solução vizinha assumirá a solução atual com uma probabilidade p descrita pela Equação (2.10), na qual T representa a temperatura atual da simulação.

$$p = e^{(-\Delta/T)} \tag{2.10}$$

A temperatura se inicia com valor elevado T_0 que decai depois de um número fixo de iterações com uma taxa de resfriamento α. Nesse sentido, α assume valores definidos entre $0 < \alpha < 1$. Desta forma, o novo valor de T é definido como descrito em (2.11).

$$T_K = \alpha \times T_{K-1} \tag{2.11}$$

O pseudocódigo do algoritmo pode ser visto no Algoritmo 1.

Algoritmo 1: Simulated Annealing($f(\cdot)$, $N(\cdot)$, α, $SAmax$, T_0, s)

```
1  s* ← s;
2  IterT ← 0;
3  T ← T_0;
4  enquanto T > 0 faça
5      enquanto IterT < SAmax faça
6          IterT ← IterT + 1;
7          Gera vizinho em s' ∈ N(s);
8          Δ ← f(s') − f(s);
9          se Δ < 0 então
10             s ← s';
11             se f(s') < f(s*) então
12                 s* ← s';
13             fim
14         fim
15         senão
16             Tome x ∈ [0, 1];
17             se x < e^{−Δ/T} então
18                 s ← s';
19             fim
20         fim
21     fim
22     T ← α × T;
23     IterT ← 0;
24 fim
25 s ← s*;
26 retorna s;
```

2.6 Considerações Finais

Grande parte dos problemas de otimização combinatória de interesse prático pertencem à classe de problemas NP-Difícil, e assim, até então não é conhecido nenhum

algoritmo eficiente capaz de encontrar a solução ótima com complexidade de tempo polinomial em máquina determinística. Isso, de certa forma, é uma das principais motivações para o contínuo desenvolvimento de técnicas algorítmicas para resolvê-los.

Neste capítulo foram apresentadas diferentes estratégias para a resolução de problemas de otimização combinatória. Os algoritmos exatos, apesar de garantirem a otimalidade, podem demandar tempos impraticáveis, o que os torna inviáveis para os problemas mais difíceis e de grande porte. Entretanto, isso não os tornam menos importantes que as outras estratégias. Com o avanço do poder de processamento dos computadores e do aprimoramento dos métodos de programação matemática, instâncias maiores e problemas mais difíceis, que não podiam ser resolvidos de maneira exata, tornaram-se possíveis. No entanto, ainda para muitos outros problemas, os métodos exatos são impraticáveis, fazendo com que algoritmos aproximativos e heurísticos sejam frequentemente utilizados como estratégias de resolução.

3 REFERENCIAL BIBLIOGRÁFICO

3.1 Algoritmos da literatura

As diversas soluções heurísticas para o problema de alocação de horários educacionais são amplamente exploradas pela literatura. Dessa forma, uma vez que o problema abordado no Centro Federal de Educação Tecnológica de Minas Gerais em questão é do tipo baseado em currículos (CB), cabe a análise dos algoritmos usados neste tipo de problema.

Dentre as estratégias de solução propostas, se destacam as exatas e as baseadas em heurísticas (CESCHIA; DI GASPERO; SCHAERF, 2023). Neste referencial, são analisados os trabalhos que fazem uso de métodos heurísticos.

3.1.1 Busca Tabu

A Busca Tabu (TS, do inglês *Tabu Search*) é uma meta-heurística proposta por Fred Glover (GLOVER, 1986) e Pierre Hansen (HANSEN, 1986). A aplicação desse método ocorre por meio da exploração e movimentação das soluções que sejam os melhores vizinhos, associado a um armazenamento das soluções anteriores para que os ótimos locais sejam evitados (SOUZA, 2008). Desta forma, parte-se de uma solução inicial s, e busca-se, dentro de uma parte da vizinhança de s, uma solução que, de acordo com a função objetivo $f(.)$ seja melhor que a corrente (SOUZA, 2008). O nome do algoritmo se dá pela lista de movimentos que não devem ser feitos durante a execução, denominada *lista tabu*. Essa lista é necessária devido à geração de soluções já criadas anteriormente ser recorrente, dada a estratégia de escolha do melhor vizinho (SOUZA, 2008).

O algoritmo de busca Tabu foi aplicado em conjunto com sistemas *Fuzzy* ao problema de alocação de horários para provas com ajustes de pesos, o que proporcionou resultados satisfatórios (AMARAL; PAIS, 2016). Além disso, tal algoritmo para o problema de alocação de horários baseados em currículo já foi usado no formato adaptativo em três fases (BELLIO et al., 2016).

No caso de alocação de horários escolares pós-matrícula, o algoritmo de busca tabu foi usado como primeira fase de uma combinação de algoritmos de busca local no

formato de *Busca Tabu com Amostragem e Perturbação* (TSSP). Desta forma, a partir da segunda fase com o *Simulated Annealing*, foi possível obter resultados satisfatórios (GOH; KENDALL; SABAR, 2017).

Para além disso, após a implementação do novo sistema de ensino chinês, que permite a adição de classes optativas, um modelo baseado em coloração de grafos para eliminação de cálculos redundantes foi criado através de uma implementação da *Busca Tabu* de duas fases. (SUN; WU, 2023).

3.1.2 *Simulated Annealing*

O *Simulated Annealing* é uma meta-heurística probabilística desenvolvida por Kirkpatrick (KIRKPATRICK; GELLAT; VECCHI, 1983). Os detalhes envolvidos em sua implementação e aplicação são tratados com detalhes na Metodologia.

Para a formulação de alocação de horários baseada em currículos, este algoritmo foi usado em combinação com um método de ajuste de parâmetros que funciona por meio da análise entre a relação da instância e os argumentos da técnica (BELLIO et al., 2016).

No âmbito da formulação pós-matrícula, o *Simulated Annealing* se apresenta como segunda fase de um algoritmo combinado de buscas locais, em forma da variação *Simulated Annealing com Reaquecimento* (SAR) (GOH; KENDALL; SABAR, 2017).

Uma variante do *Simulated Annealing* convencional, conhecida como *FastSA*, é aplicado ao *Timetabling* para exames. Em tal abordagem, o critério de aceitação é alterado, de modo que movimentos aceitos em temperaturas imediatamente anteriores são levados em consideração para a aceitação de próximas mudanças.(LEITE; MELÍCIO; ROSA, 2019).

3.1.3 *Greedy Randomized Adaptative Search Procedure*

O algoritmo *Greedy Randomized Adaptative Search Procedure* (GRASP) ou Procedimento de Busca Adaptativa Gulosa e Aleatória é um algoritmo criado por Resende (FEO; RESENDE, 1995). Tal algoritmo pode ser descrito em fases distintas de maneira que, em primeiro plano, gera-se uma solução s. Posteriormente, explora-se, iterativamente, a vizinhança de s por meio de busca local para a obtenção de ótimos

locais. Desta forma ao fim das iterações o melhor resultado é dado como o resultado final (SOUZA, 2008).

O GRASP foi aplicado ao problema de alocação de horários para frotas de ônibus dada a dificuldade inerente ao problema NP-Difícil com grandes massas de dados que *solvers* comerciais possuem (ZAMANI KAFSHANI; MIRHASSANI; HOOSHMAND, 2020).

O algoritmo também já foi aplicado ao problema de *timetabling* escolar por meio de uma combinação de algoritmo guloso para a geração de uma solução inicial e uma busca tabu complementar para melhorias (RESENDE et al., 2004).

3.1.4 *Iterated Local Search*

O algoritmo *Iterated Local Search* (ILS) é descrito como uma série de perturbações da solução inicial em busca de melhorias (SOUZA, 2008). Dessa forma, deve-se inicialmente gerar uma solução s e, a partir desta, realizar uma busca local para se obter uma solução vizinha melhor. Após tal processo, é necessária uma forma de perturbar as soluções e, por meio de determinado critério de aceitação, decidir qual solução será aceita para a próxima iteração (SOUZA, 2008).

No que tange à aplicabilidade, durante o *ITC 2011*, a solução ganhadora combinou *Simulated Annealing* com o *Iterated Local Search* na fase de melhoria, dada uma solução inicial gerada por *Kingston High School Timetabling Engine* (KHE) (FONSECA; SANTOS; CARRANO, 2016).

Por outro lado, no âmbito da geração de horários universitários, o algoritmo em questão é aplicado em um modelo de três fases, compostas pela geração de uma solução inicial, uma intensificação por meio da aplicação do *Simulated Annealing* e um método iterativo por meio do *Iterated Local Search* (SONG et al., 2018).

O ILS é também combinado com uma hiper-heurística (meta-heurística que escolhe outras meta-heurísticas) e testado em bases de dados da segunda edição do *International Timetabling Competition* com resultados efetivos.(SORIA-ALCARAZ et al., 2016).

3.1.5 *Variable Neighborhood Search*

O *Variable Neighborhood Search* (VNS), é uma meta-heurística baseada em busca local que foi proposta por Nenad Mladenović e Pierre Hansen (HANSEN; MLADENOVIĆ, 2005). Sua operação acontece por meio de trocas sucessivas nas estruturas de vizinhança (SOUZA, 2008). Entretanto, sua peculiaridade se baseia no fato de que sua exploração é feita por meio das estruturas mais distantes da solução vigente, ou seja, as alterações da atual solução em comparação com a nova são significativas. Com isso, a nova solução se mantém somente se for melhor que a solução atual (SOUZA, 2008). A exploração ocorre por meio do algoritmo *Variable Neighborhood Descent* (VND), de maneira que, partindo de uma solução inicial s, se itera por soluções vizinhas s', nas quais a busca local é aplicada. Assim, quando a busca local s'' for superior a s, retorna-se para o início das estruturas (SOUZA, 2008).

O VNS apresenta resultados superiores ao *Simulated Annealing* em algumas instâncias, como as do *ITC*, além de possuir implementação simples, com poucos parâmetros (FONSECA; SANTOS, 2014).

Esta técnica é aplicada na geração de horários de ensino médio com variações, como o *Reduced VNS*, que exclui a parcela de *descent* do algoritmo, *Sequential VNS (SVNS)* que altera a forma de exploração das estruturas vizinhas e o *Skewed VNS*, que realiza uma relaxação na escolha de melhorias, de modo que o *SVNS* apresentou melhores resultados (FONSECA; SANTOS, 2014).

O VNS foi também incorporado ao *Iterated Local Search* na geração de horários escolares de ensino médio, obtendo resultados satisfatórios em comparação ao vencedor do *ITC 2011 (Simulated Annealing)* (SAVINIEC; CONSTANTINO, 2017).

Além disso, o algoritmo VNS foi aplicado aos cursos de Sistemas de Informação e Análise e Desenvolvimento de Sistemas do Centro Universitário do Espírito Santo com grande melhora em relação à estratégia vigente da instituição (XAVIER et al., 2013).

Em outra aplicação semelhante, o algoritmo VNS foi usado para a alocação de salas de aula (SOUZA et al., 2002).

Em complemento, dadas as instâncias da terceira edição do *ITC*, a equipe finalista utilizou uma combinação de *Simulated Annealing* e *Variable Neighborhood Search* após a geração inicial provida pelo *software Kingston's High School Timetabling Engine* (KHE)

(BRITO et al., 2012).

3.2 Restrições usadas na Literatura

As restrições usadas nos trabalhos explorados nas diversas instâncias dos problemas de programação de horários educacionais podem ser divididas em dois grandes grupos: rígidas e suaves. Entretanto, alguns autores levam em consideração outros aspectos para a definição das funções objetivos. A partir disso, a numeração e organização delas se mostra necessária para a compreensão do uso das restrições na literatura.

Para Fonseca e Santos (2014), as restrições são divididas, além de rígidas e suaves, entre básicas, de eventos e de recursos. Dentre as restrições básicas, se encontram: cada evento deve ter um intervalo de tempo (1), cada evento deve ter recursos atribuídos (2), alguns eventos tem preferência por determinados horários (3), alguns eventos tem preferência por determinados recursos (4). Por outro lado, quanto às restrições de eventos, se encontram: agendamento de um conjunto de eventos para mesmo horário de começo (5), limite diário de aulas (6), definir sala padrão para evento ocorrer (7), quantidade de ocorrências de eventos e sua duração (8), número máximo de ocorrências de eventos (9). Por fim, as restrições de recursos aparecem como: evitar conflitos de recursos (10), evitar atribuir horários em que salas não estão disponíveis (11), definir valor mínimo e máximo de carga horária de um recurso (12) e definir limite de inatividade de recurso (13).

Saviniec e Constantino (2017) se limitam em seu problema a definir restrições suaves e rígidas. Desta forma, as restrições rígidas são: cada evento deve ser atribuído exatamente n vezes por semana (14), cada turma deve atender a exatamente uma reunião por período de tempo (15), cada professor deve lecionar no máximo uma aula por período de tempo(16), professores não devem ser atribuídos a intervalos de tempo que estão indisponíveis (17). Quanto às restrições suaves, encontram-se: cada evento deve ter pelo menos m aulas duplas por semana (18), horários ociosos devem ser evitados no cronograma dos professores (19), os cronogramas dos professores devem ser concentrados em um número mínimo de dias (20).

Além disso, Xavier et al. (2013) define quatro restrições para a geração de horário:

um mesmo professor não pode ministrar mais de uma aula por período de tempo (21), uma disciplina deve cumprir uma carga horária semanal (22), uma turma deve ter somente um professor por horário de aula (23) e uma disciplina deve ter apenas um professor em uma mesma turma (24).

Complementarmente, para Souza et al. (2002), as restrições foram definidas como: na mesma sala deve haver apenas uma aula por vez (24), uma sala deve respeitar sua capacidade quanto à quantidade de alunos (25), tentar sempre alocar na mesma sala alunos de mesmo curso e período (26), tentar sempre alocar turmas que demandam recursos às salas que possuem tais ferramentas (27), respeitar restrições de uso de salas (28), evitar alocar turmas com poucos alunos em salas com grande capacidade (29), respeitar salas reservadas para determinadas atividades (30) e, para fins de limpeza, tentar deixar ao menos um horário ocioso por sala em um dia (31).

Amaral e Pais (2016), por sua vez, definem para o problema de alocação de horários para exames as restrições de evitar provas em períodos consecutivos em um mesmo dia (32), bem como provas em um mesmo dia (33), em dias consecutivos (34) e em períodos consecutivos durante uma mesma noite (35).

Para Bellio et al. (2016), as restrições também podem ser simplesmente divididas entre suaves e rígidas. Para tanto, as restrições rígidas são: duas aulas não podem ser realizadas simultaneamente na mesma sala (36), aulas de turmas com mesmo currículo ou dadas pelo mesmo professor devem ser ajustadas em períodos diferentes (37) e um curso pode não estar disponível em determinado período (38). Por outro lado, as restrições suaves são definidas como: a quantidade de alunos em uma turma deve ser menor que a capacidade da sala (39), as aulas de uma determinada turma devem ser espalhadas em um número mínimo de dias (40), aulas de eixos parecidos devem ser alinhados em horários próximos (41) e, finalmente, todas as aulas de uma turma devem ser dadas em uma mesma sala (42).

Goh, Kendall e Sabar (2017), em seu trabalho para o problema pós-matrícula, definem suas restrições rígidas como: nenhum estudante pode ser matriculado em dois cursos distintos (43), as salas devem atender os requisitos do curso (44), o número de estudantes deve ser menor ou igual à capacidade da sala (45), não podem existir mais de uma turma em determinada sala no mesmo intervalo de tempo (46). Já as restrições suaves foram colocadas como: um estudante não deve ter apenas uma aula em um dia

(47), um estudante não deve ter mais de duas aulas consecutivas (48) e, por fim, um estudante não deve ter uma aula alocada no último intervalo de tempo do dia (49).

Sun e Wu (2023), definem as restrições rígidas como: aulas devem ser atribuídas de forma uniforme em todas semanas (50), cada turma deve atender a uma aula por período (51), cada professor deve dar no máximo uma aula por período (52), aulas pertencentes ao mesmo eixo devem ser atribuídas no mesmo intervalo de tempo da semana (53) e aulas não devem ser atribuídas a períodos de tempo nos quais o professor esteja indisponível (54). Em relação às restrições suaves, tem-se: cada turma não pode ser alocada mais que n vezes por dia (55), para cada turma, um número mínimo de aulas consecutivas devem ser respeitadas (56), deve-se evitar tempo ocioso no cronograma do professor (57), não devem haver lacunas entre os horários de uma mesma turma que estejam no mesmo dia (58), os planos de ensino dos professores devem ser arranjados de forma síncrona se eles lecionam em mais de uma turma (59) e, em conclusão, evitar alocar aulas em períodos de pesquisa do professor (60).

Leite, Melício e Rosa (2019), no problema de alocação de exames, delimitam as restrições rígidas como: não devem existir conflitos entre dois exames para um mesmo estudante (61), para todas salas e intervalos de tempo, o número de cadeiras deve ser maior que o número de ocupantes (62), para cada exame, a duração deve ser menor ou igual ao intervalo de tempo alocado (63) e exames devem ser realizados em salas únicas (64). Complementarmente, as restrições suaves são: exames consecutivos devem ser minimizados (65), caso existam mais de 3 períodos de provas, as lacunas entre horários de provas devem ser minimizados (66), os horários de provas devem ser distribuídos ao longo dos intervalos disponíveis (67), deve-se evitar provas em uma mesma sala que tenham duração distintas entre si (68), as provas com maior quantidade de alunos devem ser alocadas primeiro (69) e, por fim, a utilização das salas e períodos de tempo devem ser minimizadas (70).

Para Resende et al. (2004), as restrições são postas como: um professor não pode dar mais de uma aula ao mesmo tempo (71), uma turma não pode ter uma aula com mais de um professor ao mesmo tempo (72), cada professor deve preencher seu número semanal de aulas (73), um professor não pode ser alocado a um horário no qual ele não esteja disponível (74), uma turma não pode ter o mesmo conteúdo por mais de duas aulas em um mesmo dia (75), o pedido de professores em dar duas aulas consecutivas deve

ser respeitado sempre que possível (76) e os horários dos professores deve ser o mais compacto o possível (77).

Song et al. (2018), em sua formulação, defende três restrições rígidas iniciais para o problema de alocação de horários universitários: todos eventos devem ser alocados em um determinado horário e em determinada sala (78), nenhum estudante deve estar envolvido em mais de um evento ao mesmo tempo (79) e as salas devem receber apenas um evento por vez (80).

Por fim, Soria-Alcaraz et al. (2016) também divide as restrições entre rígidas e suaves. Entretanto, divide restrições úteis para o problema pós-matrícula e o baseado em currículos. Para tanto, as restrições suaves no modelo pós-matrícula são: estudantes não devem ser alocados para aulas no último intervalo de horário (81), alunos não devem ter três aulas consecutivas (82), estudantes não devem ter apenas uma aula em determinado dia (83). Por outro lado, as restrições rígidas são: nenhum estudante deve ser requisitado a participar de mais de uma aula no mesmo horário (84), para cada caso as salas devem atender os requisitos da turma (85), apenas um evento deve ser colocado em uma sala em cada horário (86), aulas devem ser colocadas em intervalos de horários pré-definidos como disponíveis (87), quando especificados, eventos devem ser realizados de forma consecutiva (88). Enquanto isso, para o modelo baseado em currículos, as restrições suaves se encontram como: A quantidade de alunos em determinada sala não deve exceder a capacidade da sala (89), aulas de cada turma devem ser espalhadas em um número mínimo de dias (90), aulas do mesmo eixo devem ser alocadas de forma adjacente ou consecutivas (91) e todas as aulas de determinada turma devem ser feitas na mesma sala (92). Complementarmente, as restrições rígidas são: todas aulas devem ser alocadas e em horários distintos (93), duas aulas não devem ser feitas na mesma sala e no mesmo horário (94), aulas de cursos do mesmo eixo devem ser alocadas em horários diferentes (95) e as aulas devem satisfazer os requisitos e disponibilidade dos professores (96).

Brito et al. (2012), Zamani Kafshani, Mirhassani e Hooshmand (2020) e Fonseca, Santos e Carrano (2016), não explicitam as restrições usadas em seus problemas.

Percebe-se, portanto, que os algoritmos heurísticos descritos são amplamente explorados na literatura para o problema de alocação de horários escolares. A Tabela 1 sintetiza e associa o uso das restrições e técnicas aos respetivos autores.

Algoritmos	Trabalhos	Restrições
VNS	(FONSECA; SANTOS, 2014), (SAVINIEC; CONSTANTINO, 2017), (XAVIER et al., 2013), (SOUZA et al., 2002), (BRITO et al., 2012)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
TS	(GOH; KENDALL; SABAR, 2017), (AMARAL; PAIS, 2016), (BELLIO et al., 2016), (SUN; WU, 2023)	32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55
SA	(GOH; KENDALL; SABAR, 2017), (BELLIO et al., 2016), (LEITE; MELÍCIO; ROSA, 2019)	36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70
GRASP	(RESENDE et al., 2004), (ZAMANI KAFSHANI; MIRHASSANI; HOOSHMAND, 2020)	71, 72, 73, 74, 75, 76, 77
ILS	(SONG et al., 2018), (SORIA-ALCARAZ et al., 2016)	78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96

Tabela 1 – Tabela de trabalhos

4 O PROBLEMA DE ALOCAÇÃO DE HORÁRIOS

4.1 Introdução ao problema

O problema de alocação de horários é um clássico problema computacional de otimização pertencente à classe NP-Difícil (BABAEI; KARIMPOUR; HADIDI, 2015). Sobre ele, deve-se destacar que a sua resolução se mostra relevante em diversas áreas. Dentre elas, destacam-se as alocações de frotas de ônibus e transportes em geral, de jogos esportivos em estádios e ginásios, de exames nacionais ou de cunho local, bem como a alocação de horários educacionais em instituições de nível técnico e superior. Sobre isso, percebe-se que a alocação de horários em escolas e universidades desempenha papel fundamental e cíclico, uma vez que a cada semestre ou ano novos horários devem ser elaborados (BELLIO et al., 2016).

O problema de alocação de horários educacionais pode ser particionado em dois tipos clássicos: *Post-Enrollment Timetabling* (PE) e *Curriculum-based* (CB). O primeiro, representa os tipos de gerações realizadas após a matrícula dos alunos em disciplinas. Levando isso em consideração, a complexidade do problema tende a aumentar. O segundo caso, por sua vez, define uma geração baseada em currículos, logo, não se considera a quantidade de matrículas nas turmas. (BELLIO et al., 2016).

Uma vez que o interesse das soluções geradas pelas pesquisas acerca do assunto abordado é global, o *International Timetabling Competition* (ITC) foi criado a fim de fomentar as pesquisas da área (VAN BULCK; GOOSSENS, 2023). Além disso, a implantação da competição gera a possibilidade da comparação entre diferentes soluções abordadas ao redor do mundo (FONSECA; SANTOS, 2014). Entretanto, as instâncias utilizadas nas competições são artificiais e se distanciam de aplicações reais, de modo que as restrições rígidas e suaves são pré-definidas (VAN BULCK; GOOSSENS, 2023).

4.2 Definição do Problema e Formulação Matemática

Antes de se apresentar a definição e formulação matemática para o problema de programação de horários educacionais, é introduzida a terminologia adotada na literatura para este problema.

a) **Horário**: O horizonte de tempo é dividido em dias, e cada dia é subdividido em blocos de horários (geralmente, um mesmo número de blocos é atribuído a cada dia). Um *período* é um bloco par (*dia*, *bloco de horário*).
b) **Eventos**: Um evento é um encontro entre os estudantes e um ou mais professores. Eventos podem ser de diferentes tipos: aulas ou avaliações de uma disciplina, laboratório ou seminários.
c) **Recursos**: São considerados três tipos principais de recursos: estudantes, professores e salas. Os eventos precisam ser agendados levando em consideração as restrições de recursos, como matrículas dos estudantes, demandas e preferências dos professores, além da disponibilidade das salas.
d) **Restrições**: As restrições são divididas em obrigatórias (*hard constraints*) e desejáveis (*soft constraints*). As restrições obrigatórias devem ser sempre atendidas, enquanto as desejáveis contribuem para a função objetivo, que é soma ponderada de todas as penalidades das restrições desejáveis não atendidas.
e) **Conflito**: É uma colisão entre alocações de recursos como salas ou turmas (BABAEI; KARIMPOUR; HADIDI, 2015).

Para o problema de programação de horários educacionais são dados um conjunto de eventos, um conjunto de períodos (dias divididos em blocos de horário) e um conjunto de salas. São dados, também, um conjunto de eventos conflitantes e, portanto, não podem ser alocados em um mesmo período, pois são eventos associados a um mesmo grupo de estudantes ou professor. Alguns eventos também não podem ser alocados em determinados períodos devido a questões de compatibilidade entre os eventos e os recursos associados a ele. Há, também, relações de precedência entre os eventos, que impõem que alguns eventos devem ser agendados antes de outros. Por fim, também são dados um conjunto de restrições de compatibilidade entre os eventos e salas, ou seja, alguns eventos requerem tipos específicos de salas (por exemplo, um laboratório, auditório ou quadra esportiva).

Considere os seguintes conjuntos: $E = \{1, 2, ..., n_E\}$ como o conjunto de eventos a serem agendado; $T = \{1, 2, ..., n_T\}$ como o conjunto de períodos (blocos de horário ao longo dos dias da semana); $R = \{1, 2, ..., n_R\}$ como o conjunto de salas; $C_1 = \{(i_1, i_2)\} \subset E^2$ como o conjunto de eventos conflitantes que não podem ser alocados em um mesmo

período; $C_2 = \{(i_1, i_2)\}$ como o conjunto de regras de precedência em que o evento i_1 deve preceder o evento i_2; $C_3 = \{(i, j)\} \subset E \times T$ como o conjunto de pares indicando que o evento i não pode ser agendado no período j; $C_4 = \{(i, k)\} \subset E \times R$ como o conjunto de pares indicando que o evento i não pode ser agendado para ocorrer na sala k; $C_5 = \{(k, j)\} \subset R \times T$ como o conjunto de pares indicando que a sala k não está disponível para uso no período j.

Dadas as variáveis de decisão:

$$x_{i,j,k} = \begin{cases} 1, & \text{se o evento } i \text{ é agendado para o período } j \text{ na sala } k \\ 0, & \text{caso contrário} \end{cases} \tag{4.1}$$

Para todo evento $i \in E$, período $j \in T$ e sala $k \in R$, o problema de programação de horários educacionais, abordado neste projeto, pode ser formulado como um problema de programação linear inteira mista da seguinte forma:

Minimizar $f(x)$

sujeito a:

$$\sum_{j=1}^{n_r} \sum_{k=1}^{n_R} x_{i,j,k} = 1, \qquad \forall i \in E$$

$$\sum_{i=1}^{n_E} x_{i,j,k} \leq 1, \qquad \forall j \in T; \forall k \in R$$

$$\sum_{k=1}^{n_R} x_{i_1,j,k} + \sum_{k=1}^{n_R} x_{i_2,j,k} \leq 1, \qquad \forall (i_1, i_2) \in C_1; \forall j \in T$$

$$\sum_{j=1}^{n_r} \sum_{k=1}^{n_R} j x_{i_2,j,k} \geq 1 + \sum_{j=1}^{n_r} \sum_{k=1}^{n_R} j x_{i_1,j,k}, \qquad \forall (i_1, i_2) \in C_2$$

$$\sum_{k=1}^{n_R} x_{i,j,k} \leq 0, \qquad \forall (i, j) \in C_3$$

$$\sum_{j=1}^{n_r} x_{i,j,k} \leq 0, \qquad \forall (i, k) \in C_4$$

$$\sum_{i=1}^{n_E} x_{i,j,k} \leq 0, \qquad \forall (k, j) \in C_5$$

$$x_{i,k,k} \in \{0, 1\}, \qquad \forall i \in E; \forall j \in T; \forall k \in R$$

A função objetivo $f(x)$ representa a penalidade total associada às restrições desejáveis que não são atendidas pela solução. Esta função está sujeita as seguintes restrições (enumeradas na mesma ordem em que são apresenadas na formulação acima): todo evento deve ser agendado a uma única sala em um único período; uma sala, em um dado período, não pode ser agendado para mais de um evento; tarefas conflitantes não podem ser agendadas para ocorrerem em um mesmo período; as regras de precedência entre os eventos devem ser atendidas; os eventos não podem ser agendados para certos períodos devido às restrições dos recursos associados ao evento; os eventos não podem ser agendados para salas incompatíveis com seus requisitos; as salas não podem ter eventos agendados a elas nos períodos em que estiverem indisponíveis; e, por fim, o domínio das variáveis de decisão devem ser satisfeitos.

4.2.1 Função objetivo

O objetivo do problema de programação de horários educacionais é minimizar o número de restrições desejáveis que não são atendidas por uma solução. Dessa forma, considerando o contexto do CEFET-MG, essas restrições desejáveis são:

a) Evitar períodos vagos (isso é, períodos sem eventos agendados) entre os eventos agendados para uma mesma turma (grupo de alunos de alguma forma relacionados);
b) Evitar eventos consecutivos de aulas de disciplinas consideradas desgastantes ou de um mesmo eixo (essa relação entre disciplinas é definida pelo usuário/instituição);
c) Evitar dias sem eventos alocados a uma turma (esse requisito é definido, turma a turma, pelo usuário/instituição quando necessário);
d) Evitar a alocação de eventos vinculados a um professor nos períodos que ele tiver indicado que não deseja a alocação de eventos

Assim, considere os conjuntos: $W = \{1, 2, ..., 7\}$ representando os dias da semana; $B(d) \subset T$ como sendo os períodos do dia d, para todo $d \in W$; $G = \{1, 2, ..., n_G\}$ como o conjunto das diferentes turmas (grupos de alunos relacionados); $E_G(g) \subset E$ como o conjunto de eventos associados à turma $g \in G$; $C_6 = \{(i_1, i_2)\} \subset E^2$ como o conjunto de pares $(i_1, i_2) \in E^2$ indicando que os eventos i_1 e i_2 não podem ser agendados um

imediatamente após o outro; $P = \{1, 2, ..., n_P\}$ como o conjunto dos professores; $p \in P = \{1, 2, ..., n_P\}$ como o conjunto de professores, $E_p(p) \subset E$ como o conjunto de eventos associados ao professor $p \in P$; e $T_P(p) \subset T$ como o conjunto de períodos em que o professor deseja que não sejam alocadas aulas a ele. Dessa forma, a função objetivo pode ser escrita como:

$$\text{Minimizar: } \sum_{g \in G} \sum_{j \in B(d)} e^{(1)}_{g,j} + \sum_{(i_1,i_2) \in C_6} e^{(2)}_{i_1,i_2} + e^{(3)}_{i_1,i_2} + \sum_{g \in G} \sum_{d \in W} e^{(4)}_{g,d} + \sum_{p \in P} \sum_{i \in E_p(p)} e^{(e)}_{p,i}$$

$$\begin{aligned}
\text{sujeito a: } & \sum_{i \in E_G(g)} \sum_{k=1}^{n_R} x_{i,j-1,k} + x_{i,j+1,k} \leq e^{(1)}_{g,j} + \sum_{i \in E_G(g)} \sum_{k=1}^{n_R} x_{i,j,k}, && \forall g \in G; \forall d \in W; \forall j \in B(d) : j \\
& \sum_{k=1}^{n_R} x_{i_1,j,k} + x_{i_2,j+1,k} \leq 1 + e^{(2)}_{i_1,i_2}, && \forall (i_1, i_2) \in C_6; \forall j \in T : j \\
& \sum_{k=1}^{n_R} x_{i_1,j+1,k} + x_{i_2,j,k} \leq 1 + e^{(3)}_{i_1,i_2}, && \forall (i_1, i_2) \in C_6; \forall j \in T : j \\
& \sum_{i \in E_G(g)} \sum_{j \in B(d)} \sum_{k=1}^{n_R} x_{i,j,k} \geq 1 - e^{(4)}_{g,d}, && \forall g \in G; \forall d \in W \ \{1, 7\} \\
& \sum_{j \in T_P(p)} \sum_{k=1}^{n_R} x_{i,j,k} \leq 0 - e^{(5)}_{p,i}, && \forall p \in P; \forall i \in E_P(p) \\
& e^{(.)}_{,,} \geq 0
\end{aligned}$$

em que $e^{(.)}_{,,}$ são variáveis de decisão que assumirão o valor 1 para toda restrição desejada que for violada. Dessa forma, a função objetivo modela a minimização dessas restrições desejadas que não são atendidas.

4.3 Discussões finais

Desta forma, percebe-se que a alocação de horários é uma tarefa essencial para o funcionamento de instituições empresariais e educacionais ao redor do mundo. Com isso, esforços globais são feitos para o desenvolvimento de ferramentas que facilitem tal processo, como o ITC e demais pesquisas.

5 METODOLOGIA

5.1 Considerações Iniciais

A geração de soluções viáveis para o problema de programação de horários escolares é realizado através de uma heurística baseada no *Simulated Annealing*. Tal escolha foi realizada, em detrimento das soluções exatas, devido à extensa carga literária acerca de tais métodos, diferentemente das demais técnicas de resolução que se mostram menos presentes na literatura e são exploradas em menor quantidade (FONSECA; SANTOS, 2014).

Acerca do algoritmo, selecionou-se o *Simulated Annealing*, algoritmo vencedor de diversas edições do *International Timetabling Competition* (ITC), como em 2003, 2007, 2011 e 2012 e, ainda hoje, é amplamente explorado com diversas variações (FONSECA; SANTOS, 2014).

5.2 Coleta e armazenamento de dados

Para a avaliação da estratégia de resolução adotada, são necessárias instâncias escolares disponíveis. Nesse sentido, a coleta de dados ocorre em dois âmbitos. Em primeiro plano, o site oficial do Centro Federal de Educação Tecnológica de Minas Gerais, Campus Divinópolis, fornece os horários gerados no período do primeiro semestre de 2020 até o primeiro semestre de 2023, de modo que, a coleta das instâncias ocorre manualmente, sendo armazenado em um arquivo no formato *JavaScript Object Notation* (JSON). Em segunda instância, a coleta ocorre através de um aplicativo *Desktop* que permite o cadastro de professores, salas, turmas e cursos. Tais dados, quando cadastrados, são salvos igualmente em um arquivo JSON, o que permite que os dados coletados sejam mantidos de maneira uniforme.

5.3 Restrições adotadas

As restrições de um problema de alocação de horários educacionais dependem fortemente das demandas da instituição em questão (BRITO et al., 2012). Logo, define-

se, abaixo, as demandas do Centro Federal de Educação Tecnológica de Minas Gerais, campus Divinópolis. Referente às restrições rígidas, arbitrariamente coloca-se o mesmo peso 10 para todas e peso 1 para as suaves, uma vez que as restrições rígidas devem ser maiores o suficiente para que a solução gerada dê prioridade às factíveis (BELLIO et al., 2016). Quanto às restrições rígidas, define-se:

a) Uma sala não pode ser usada por duas disciplinas em um mesmo horário;
b) Respeitar turnos de determinadas disciplinas;
c) Um professor deve ser alocado em uma única disciplina e em uma única sala em determinado horário;
d) Professores não podem dar aula em turnos alternados(manhã e noite), deve ser manhã e tarde ou tarde e noite.

Por outro lado, as restrições suaves são colocadas como:

a) Não deve haver espaço entre aulas maior que 2 horários na mesma classe em qualquer dia;
b) Evitar aulas de eixos parecidos de forma consecutiva para uma mesma turma;
c) Evitar dias sem aulas para determinadas turmas;
d) Preferências de professores;
e) Indisponibilidades de professores;
f) Indisponibilidades de salas.

5.3.1 Função de Vizinhança

Para a heurística proposta, é utilizada uma única função de vizinhança para a obtenção de novas soluções candidatas a cada iteração do *Simulated Annealing*. Essa função de vizinhança funciona da seguinte forma: dada uma solução s, o seu conjunto de soluções vizinhas $N(s) \subset \Omega$ é definido por todas aquelas soluções que se diferenciam de s em uma única atribuição de um *evento* a um *período*.

Como, a cada iteração de heurística, é escolhida uma solução $s' \in N(s)$ aleatoriamente, esta etapa da heurística pode ser implementada de maneira simples como a escolha de um evento aleatório para que tenha a sua atribuição a um *período* alterado, também de forma aleatória.

A cardinalidade dessa função de vizinhança é igual a $n \times m$, em que n é o número

de eventos na instância do problema e m é o número de $slots$ disponíveis.

5.4 Planejamento Experimental

A execução e teste do algoritmo consiste na comparação da função objetivo criada para o problema nas diferentes execuções. Para tanto, dados os horários fornecidos pelo CEFET-MG Campus Divinópolis, compara-se o valor da função objetivo da solução gerada pelo software utilizado pelo CEFET-MG, com o valor da função objetivo gerada pelo *Simulated Annealing*.

O algoritmo *Simulated Annealing* é executado 6 vezes com variação de parâmetros, de modo que seja possível verificar a influência destes no resultado final.

Finalmente, a melhor solução adquirida por meio da variação dos parâmetros será usada para comparar com o horário implantado no campus, de modo a verificar quais restrições foram violadas em maior volume e propor hipóteses das razões por trás disso.

5.5 Aplicação Desktop

O aplicativo *desktop* foi desenvolvido por meio dos *frameworks Vue*, que permite um desenvolvimento *front-end* para navegadores através da linguagem *Javascript* e o *Electron*, que possibilita que uma aplicação nativamente produzida para ser executada em um navegador *web*, gere uma aplicação *desktop* que execute em diferentes sistemas operacionais, como *Linux*, *Windows* e *MacOS*. Desta forma, o aplicativo conta com um sistema de inclusão, deleção e atualização de dados de professores, salas de aula, departamentos, disciplinas e turmas. Dada a coleta dos dados por meio da aplicação em formato JSON, a aplicação possibilita a geração de uma solução através do algoritmo *Simulated Annealing*.

6 RESULTADOS

6.1 Projeto e Implementação do Software

Dada a necessidade do Centro Federal de Educação Tecnológica de Minas Gerais, Campus V, e das demais instituições públicas de ensino em gerar semestralmente os horários educacionais, uma aplicação *desktop* foi criada no intuito de permitir a inserção personalizada de dados, preferências e outras restrições. Tal aplicação foi criada por meio do editor de código *Visual Studio Code*.

Quanto a isso, o diagrama de casos de uso descrito na Figura 1 permite visualizar as funcionalidades do software.

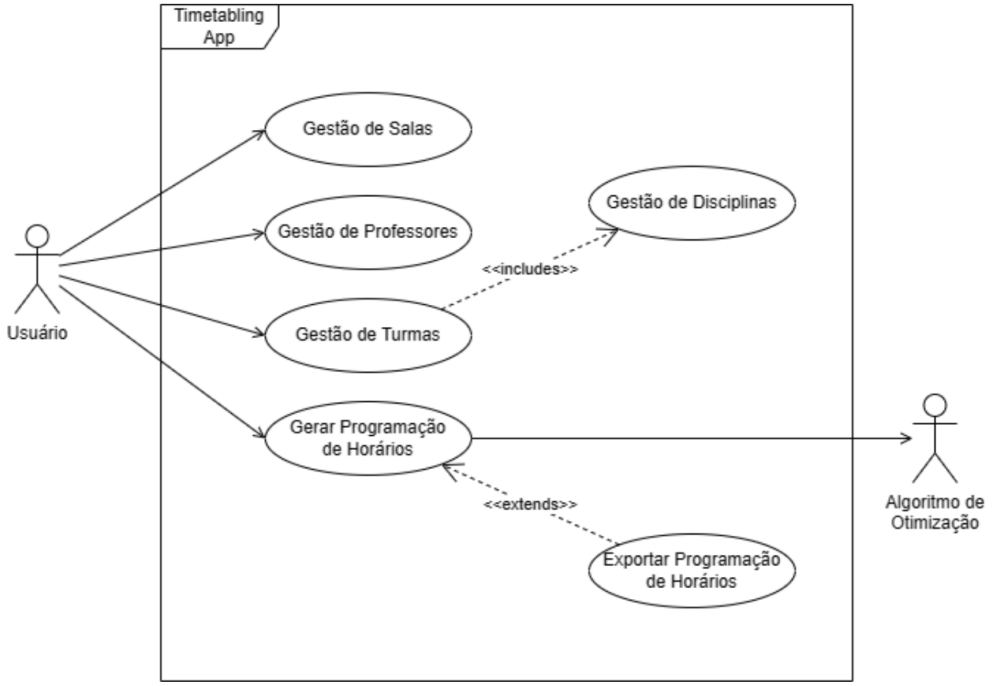

Figura 1 – Diagrama de casos de uso

Por meio concepção das funcionalidades, tais necessidades foram transformadas nas telas do software demonstradas a partir da Figura 2, na qual é possível visualizar a tela inicial do software, que permite tanto o acesso à barra de navegação quanto a botões de tutorial e de geração de horário.

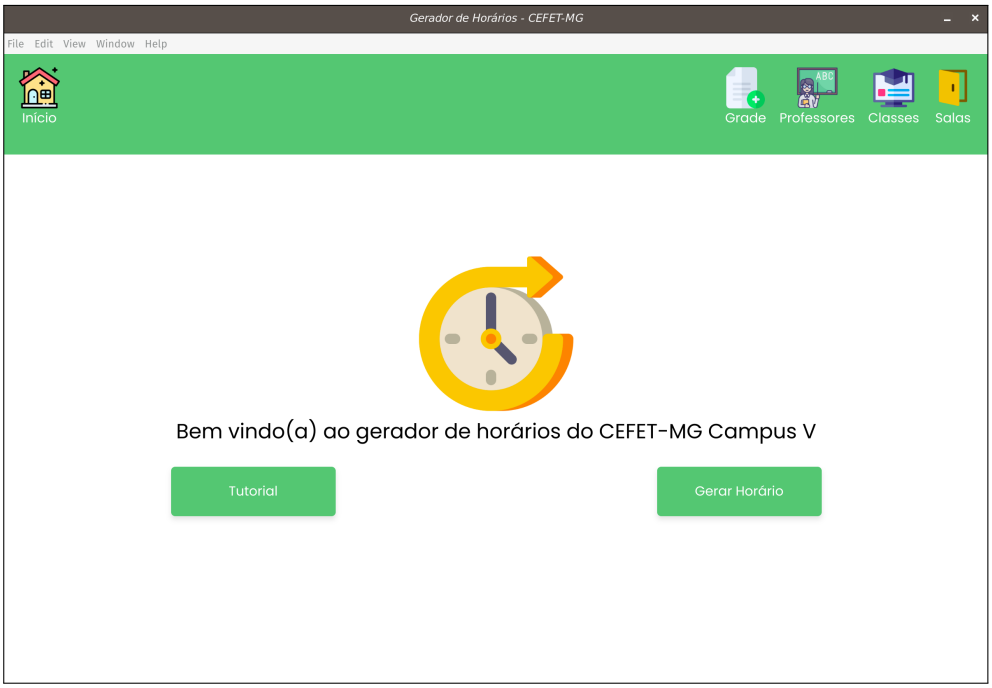

Figura 2 – Página inicial da aplicação

Ao clicar no botão de Tutorial, pode-se ver uma breve explicação do funcionamento da aplicação, como pode ser visto na Figura 3.

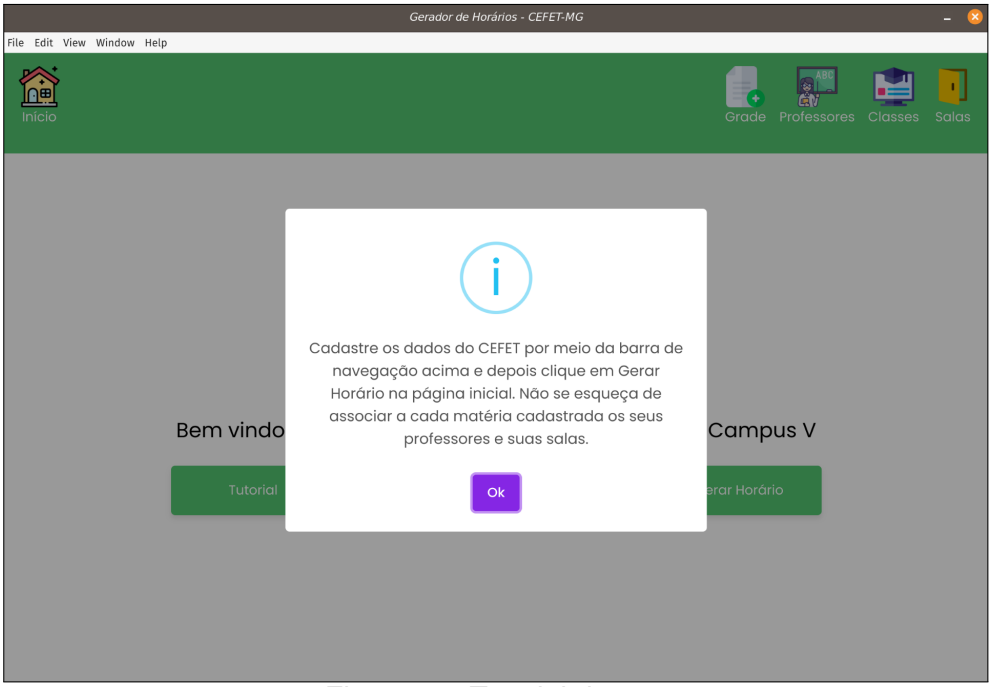

Figura 3 – Tutorial de uso

Por outro lado, ao clicar no botão de geração de horários, existem dois comportamentos possíveis: em primeiro plano, caso a instância cadastrada esteja

incompleta, ocorrerá um erro como demonstrado na Figura 4.

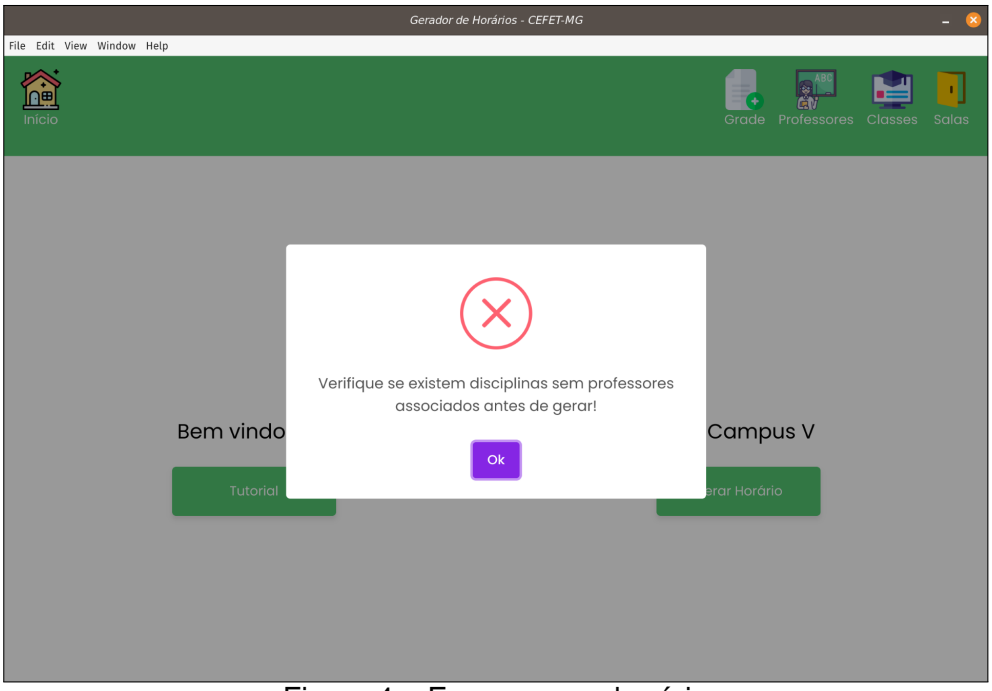

Figura 4 – Erro ao gerar horário

Por outro lado, caso os dados cadastrados estejam corretos, o algoritmo *Simulated Annealing* implementado em C++ é executado e o horário é gerado, como pode ser visto na Figura 5.

Figura 5 – Horário gerado com possibilidade de exportação em PDF

Ainda assim, a aplicação permite o acesso a importantes informações para a geração. Quanto a isso, o software permite a manipulação dos horários disponíveis na instituição, de modo que podem ser excluídos horários específicos ou ainda dias inteiros da geração final. Tal gerência pode ser vista na Figura 6.

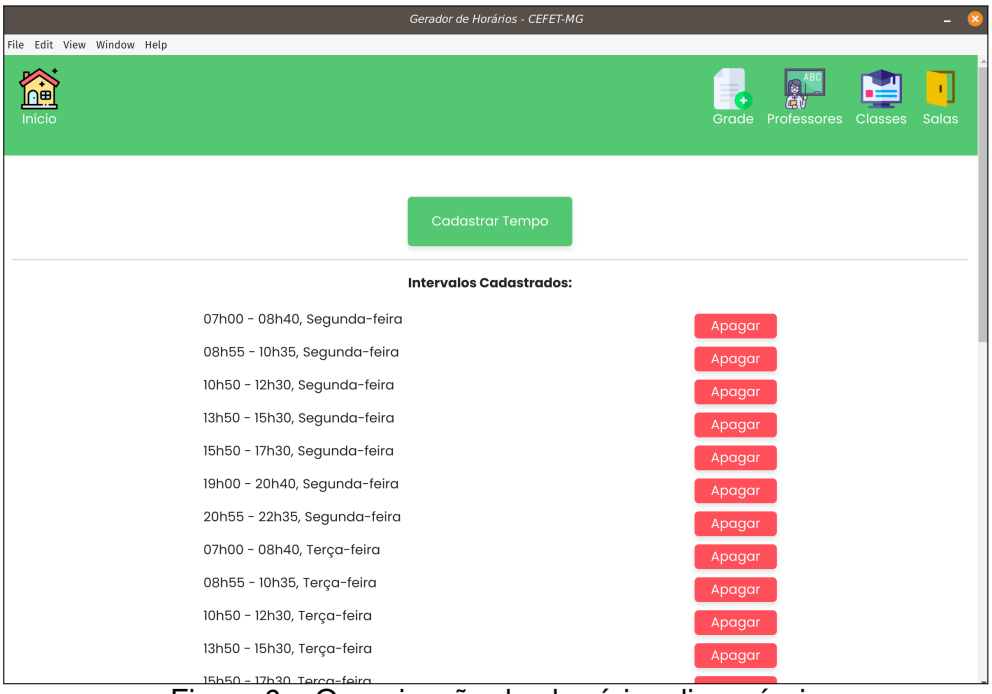

Figura 6 – Organização dos horários disponíveis

Ademais, o cadastro, edição e remoção de professores pode ser feito como consta na Figura 7.

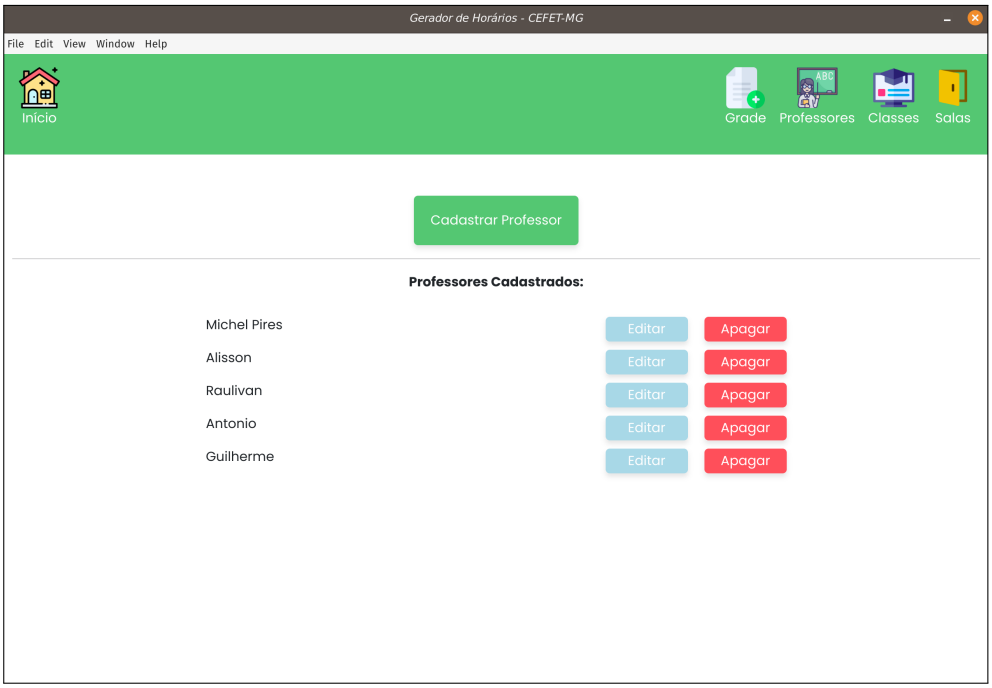

Figura 7 – Organização de professores disponíveis

Sobre isso, o cadastro de professores pode ser visto na Figura 8 e se assemelha à atualização e ao cadastro dos demais itens presentes na base de dados.

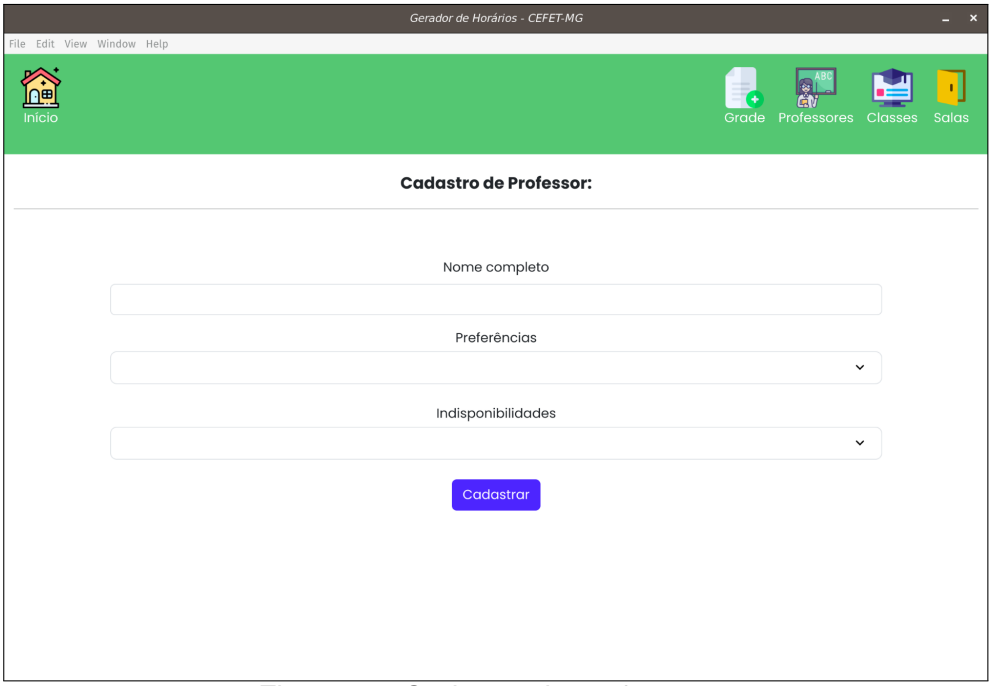

Figura 8 – Cadastro de professores

Em caso de remoção, existe a necessidade de uma confirmação, uma vez que não há recuperação, como pode ser visto na Figura 9.

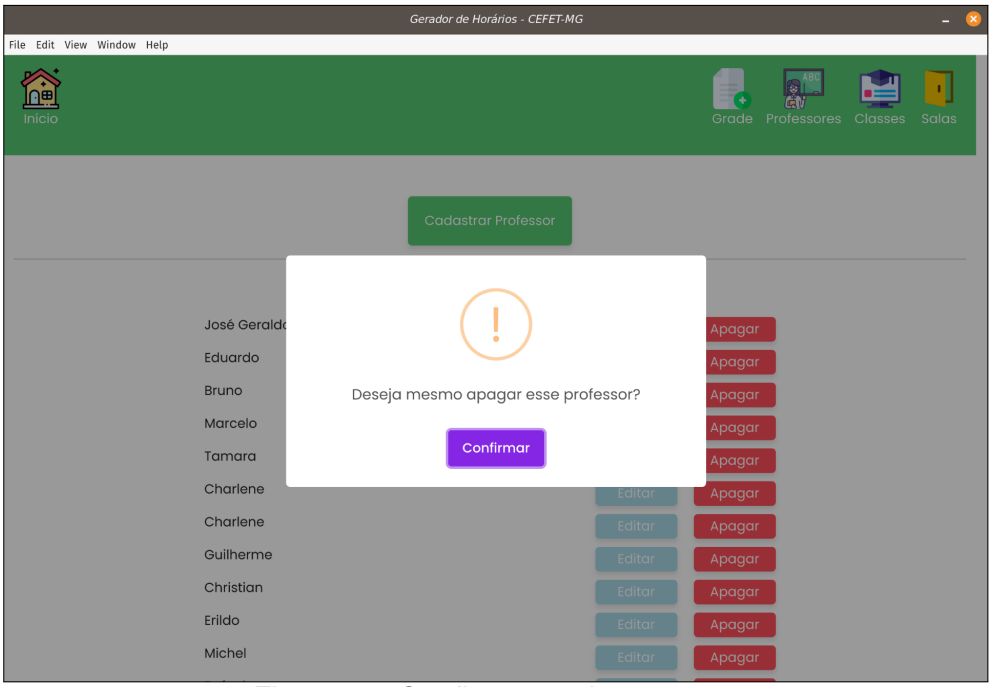

Figura 9 – Confirmação de remoção

Semelhantemente, a gerência de classes é vista na Figura 10, de disciplinas na Figura 11 e, finalmente, de salas na Figura 12.

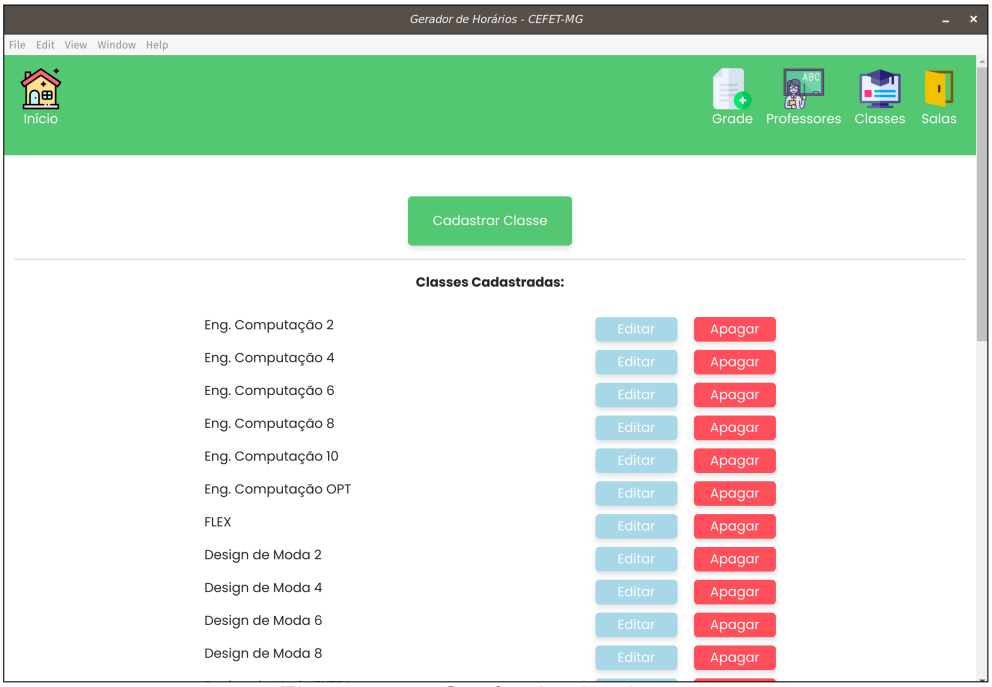

Figura 10 – Gerência de classes

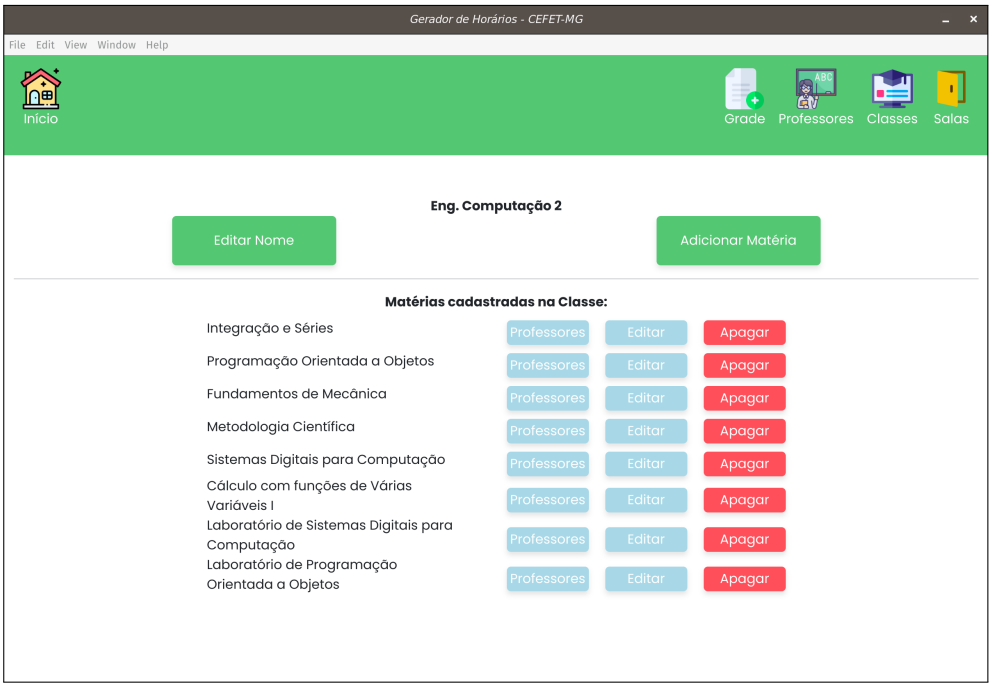

Figura 11 – Gerência de disciplinas

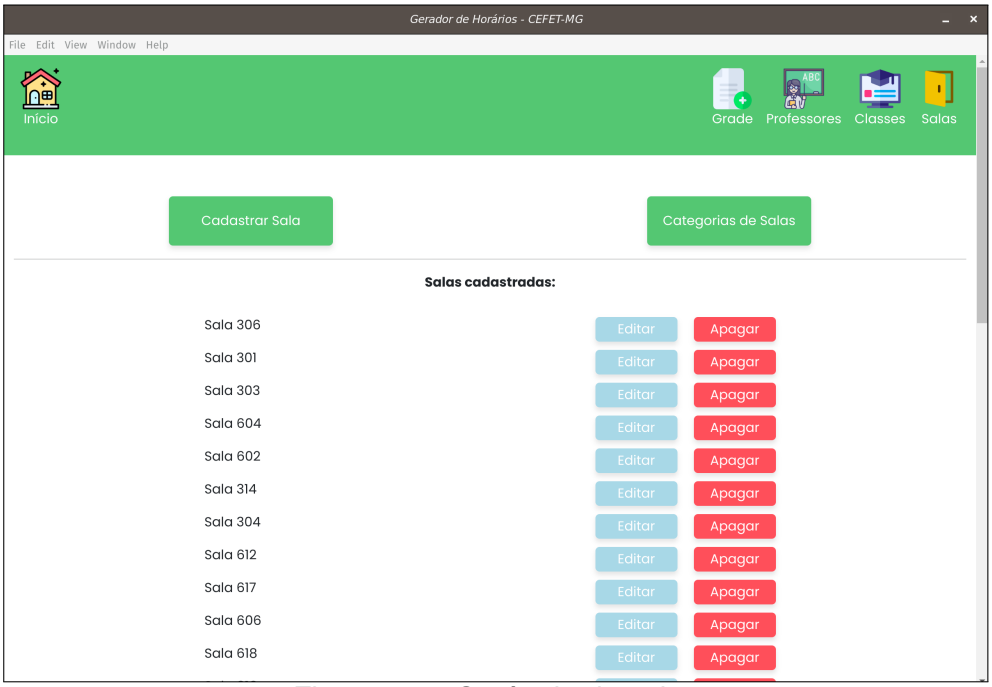

Figura 12 – Gerência de salas

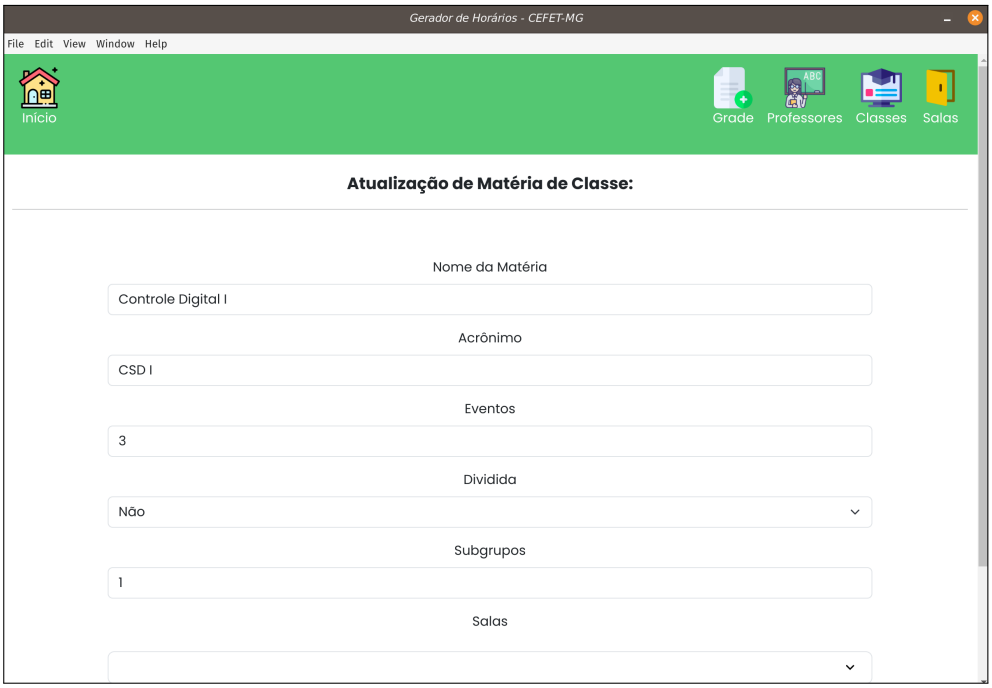

Figura 13 – Atualização cadastral de disciplina

6.2 Soluções Alcançadas

Por meio da coleta dos dados de horários do segundo semestre de 2023 para as aulas do Centro Federal de Educação Tecnológica de Minas Gerais, Campus Divinópolis e da implementação do software integrado ao algoritmo *Simulated Annealing*, foi possível obter um valor equivalente a *2305* da função objetivo do horário gerado pelo software proprietário por meio do cálculo da soma das restrições aqui definidas que foram violadas. Tais restrições contam, nas rígidas, duas violações ao turno adequado, 39 horários repetidos, 76 salas usadas por duas disciplinas diferentes em um mesmo horário e 52 professores alocados em mais de uma disciplina ao mesmo tempo. A quantidade de professores alocados em turnos alternados é 0. Quanto às restrições rígidas quebradas, ocorrem, majoritariamente devido à existência de eventos de laboratórios que dividem horários em duas disciplinas, de modo que cada disciplina possui dois ou mais grupos, o que não é permitido na restrição criada. Por outro lado, nas restrições suaves, 550 violações foram por haver espaço significativo entre dois eventos de uma mesma classe, 34 por algum dia sem aula em determinada turma e 31 violações às restrições de horários de professores. É importante salientar que tal horário contou com grande tempo hábil de execução para ser gerado, o que contribui para uma solução melhor.

Complementarmente, após a geração final, o horário é alterado manualmente para o atendimento específico de demandas que a geração do software não foi capaz de atingir.

Referente aos dados, a instância conta com 34 classes cadastradas, divididas entre graduação, curso técnico e curso técnico integrado com ensino médio, ainda assim, podendo ser classes de matérias optativas, flexibilizadas ou obrigatórias. Cada classe, conta com uma média de *9,14 disciplinas*, de modo que os cursos de graduação tem uma média de 6,73 disciplinas, os cursos técnicos integrados contam com uma média de *16,11 disciplinas* e os cursos técnicos com uma média de *6,33 disciplinas* por semestre. No semestre avaliado, a instituição possuia *85 professores cadastrados*, *37 salas de aula e laboratórios*, além de *34 horários de aula* disponíveis para alocação, distribuídos entre *7* horas da manhã de segunda-feira até *22* horas e *40* minutos de sexta-feira. Cabe salientar, ainda, que determinados cursos como a graduação em Design de Moda e os cursos técnicos noturnos possuem restrições rígidas de turno e não podem utilizar todos os *34* espaços de horário para sua alocação. Os dados sintetizados podem ser visualizados na Tabela 2.

Professores	**Salas**	**Classes**	**Eventos**
85	37	34	501

Tabela 2 – Tabela de quantidade de itens na instância do segundo semestre de 2023

Por meio desses dados o algoritmo foi executado 6 vezes com diferentes parâmetros. Quanto a isso, como consta no Algoritmo 1, O *Simulated Annealing* conta com uma temperatura inicial, um número de iterações por temperatura e um fator de resfriamento *alfa*. Desta forma, a Tabela 3 descreve os parâmetros utilizados em cada execução e qual o valor da função objetivo da solução inicial aleatória (FO inicial) e final (FO final) para cada uma delas.

Execução	**Temperatura Inicial**	**Iterações**	**Alfa**	**FO Inicial**	**FO Final**
1	50	10	0.02	7239	7010
2	50	20	0.04	7229	6990
3	50	30	0.08	7375	6674
4	50	50	0.1	7148	6205
5	50	100	0.1	7446	5555
6	50	300	0.1	6788	3544
7	50	300	0.25	7302	2937

Tabela 3 – Resultados

Comparativamente, a Figura 14 mostra a variação do valor da função objetivo inicial gerada aleatoriamente por execução e a média aritmética deste valor.

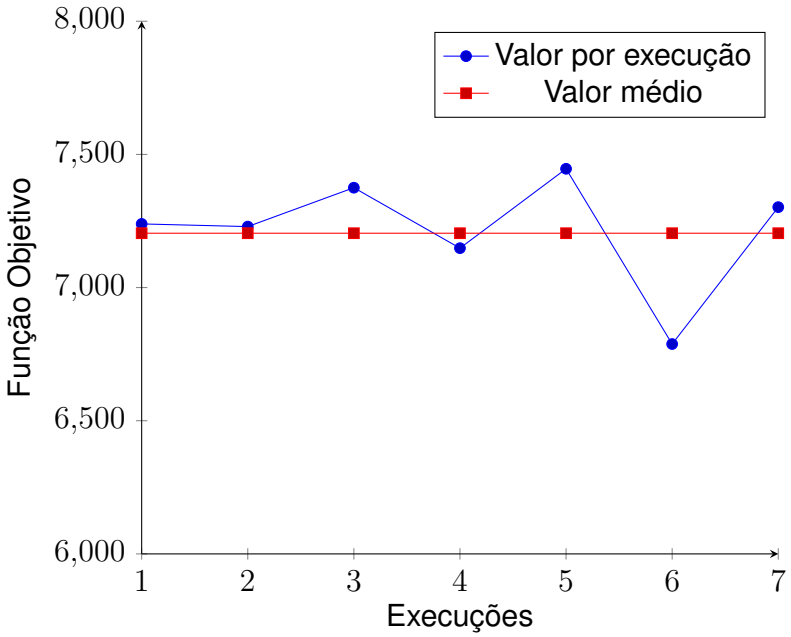

Figura 14 – Gráfico do valor das funções objetivo iniciais

Além disso, a Figura 15 demonstra a variação do valor da função objetivo final com o aumento do valor dos parâmetros e compara com a solução gerada pelo software proprietário utilizado pelo CEFET-MG, campus Divinópolis.

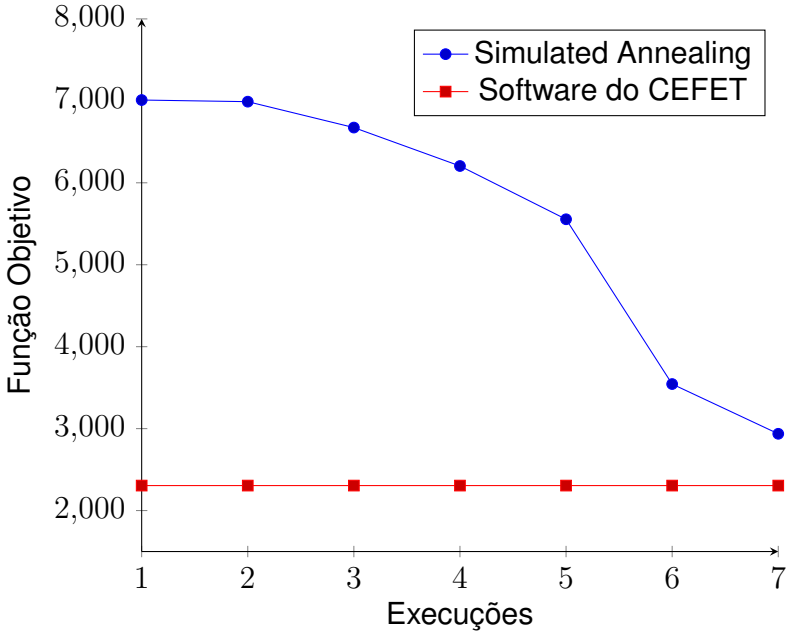

Figura 15 – Gráfico de funções objetivo finais

Desta forma, uma vez que a última execução, com parâmetros mais altos possui o maior desempenho em seu valor final da função objetivo, já que possui um tempo de execução maior, ou seja, com uma maior quantidade de trocas na vizinhança, cabe a análise percentual de restrições respeitadas por esta solução.

Em primeira análise, cabe ressaltar que a última execução quebrou 1402 vezes as restrições postas, de modo que $16,97\%$ dessas vezes foram por restrições rígidas e $83,03\%$ para as suaves. Entretanto, uma vez que o peso das restrições rígidas é dez vezes maior que o das restrições suaves, o valor final da função objetivo é representado por $67,15\%$ das restrições rígidas e $32,85\%$ das restrições suaves, valores que mostram o caráter da preferência que o algoritmo tem que buscar por soluções que evitem quebrar restrições rígidas.

Em segundo plano, faz-se relevante a análise de quais restrições foram quebradas em maior quantidade. Dentre as restrições rígidas quebradas, 135 vezes foram falhas em respeitar os turnos de determinadas turmas, como as turmas do curso de Design de Moda que precisam estar alocadas apenas durante a noite, ou os primeiros semestres de Engenharia de Computação que precisam estar apenas de manhã ou tarde. Ademais, 38 vezes foram erros em garantir que salas fossem usadas por apenas um evento em determinado horário específico independente da classe tratada.

Complementarmente, 45 vezes foram eventos alocados em horários já utilizados, de modo que parte destes casos contemplam horários com mais de dois eventos alocados de turmas que podem dividir horários com outras, como laboratórios com dois grupos, o que ultrapassa o limite de no máximo dois eventos por espaço de horário, enquanto o restante da porcentagem se aplica para alocação indevida de mais de um evento para um certo horário. Ainda assim, 20 vezes a restrição de um professor poder atuar em uma única disciplina e sala em determinado horário. Por fim, a restrição de professores atuarem em turnos alternados foi totalmente atingida e não foi quebrada. Referente a esses dados, conjectura-se que o fato das restrições que envolvem os professores serem as mais fáceis de serem atendidas é reflexo da necessidade de cada disciplina atribuir previamente quais professores nela atuam, de modo que, em geral, os professores atuam em eixos específicos e não invadem horários em turmas que não fazem parte do seu conhecimento. Por outro lado, restrições de turno e de horários, envolvem uma grande gama de possibilidades combinatórias, que contemplam a grande quantidade de salas, de

espaços a serem alocados e de disciplinas. Desta maneira, restrições deste tipo, potencialmente necessitariam de um peso maior durante a execução do algoritmo.

Enquanto isso, as restrições suaves foram quebradas 1008 vezes por haver espaço entre aulas significativo, 34 vezes por possuir algum dia de determinada classe sem aula e 122 vezes por não respeitar alguma indisponibilidade descrita por certo professor. Outras restrições como evitar aulas consecutivas de eixos parecidos para uma mesma classe, preferências de professores e indisponibilidades de salas não foram contabilizadas por falta de dados na instância coletada. Com isso, percebe-se que a maior problemática no que tange às restrições suaves é a dificuldade de conciliar todas as demais restrições rígidas ou não com a preferência de aulas menos afastadas entre si. Tal questão se justifica potencialmente graças à grande quantidade de horários que podem ser alocados em relação à média de disciplinas em cada turma, de modo que inevitavelmente, determinadas soluções boas tendem a possuírem espaços entre eventos.

7 CONCLUSÃO

A problemática da alocação de horários é ampla e de difícil solução, uma vez que a medida que a instância tratada aumenta de tamanho, soluções exatas se tornam inviáveis em relação ao tempo computacional. Desta forma, soluções heurísticas como o *Simulated Annealing* são de grande valia. Isso fica explícito no comparativo do valor da função objetivo atingida pela solução gerada pelo algoritmo em comparação com a solução já existente, dadas as restrições adotadas nesta pesquisa. Entretanto, mesmo soluções heurísticas, quando deparadas com instâncias grandes demandam um tempo de execução alto, logo, a melhor solução encontrada, não é necessariamente a melhor solução possível que o algoritmo possibilita, já que é possível aumentar o valor dos parâmetros indefinidamente, além da possibilidade de alterar a função de vizinhança.

Complementarmente, novas restrições podem ser adotadas ou retiradas a medida que novas demandas aparecerem. Por outro lado, a aplicação *Desktop*, pode incluir, ainda, a opção de gerência manual dos horários gerados, o que, assim como foi feito na solução já existente, permite tanto uma maior flexibilidade, quanto também uma diminuição do valor final da função objetivo.

REFERÊNCIAS

AMARAL, Paula; PAIS, Tiago Cardal. Compromise ratio with weighting functions in a Tabu Search multi-criteria approach to examination timetabling. **Computers and Operations Research**, v. 72, 2016. ISSN 03050548. DOI: 10.1016/j.cor.2016.02.012.

ARBAOUI, Taha; BOUFFLET, Jean Paul; MOUKRIM, Aziz. Lower bounds and compact mathematical formulations for spacing soft constraints for university examination timetabling problems. **Computers and Operations Research**, v. 106, 2019. ISSN 03050548. DOI: 10.1016/j.cor.2019.02.013.

AUSIELLO, G. et al. **Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties**. 1st edition. [S.l.]: Springer, 1999.

BABAEI, Hamed; KARIMPOUR, Jaber; HADIDI, Amin. A survey of approaches for university course timetabling problem. **Computers and Industrial Engineering**, v. 86, 2015. ISSN 03608352. DOI: 10.1016/j.cie.2014.11.010.

BARBOSA, Luciano Wallace Gonçalves; MAPA, Sílvia Maria Santana. Aplicações da pesquisa operacional no setor de mineração: estudo bibliométrico no período de 2006 a 2016. **Revista Latino-Americana de Inovação e Engenharia de Produção**, v. 5, n. 8, p. 166–186, 2017.

BARREIRA, Nuno Manuel Couto. **Sistema Inteligente para otimização de rotas**. 2016. Tese (Doutorado).

BAZARAA, M.S.; JARVIS, J.J.; SHERALI, H.D. **Linear Programming and Network Flows**. 4th. [S.l.]: Wiley, 2009.

BELLIO, Ruggero et al. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. **Computers and Operations Research**, v. 65, 2016. ISSN 03050548. DOI: 10.1016/j.cor.2015.07.002.

BRITO, Samuel S. et al. A SA-VNS approach for the High School Timetabling Problem. **Electronic Notes in Discrete Mathematics**, v. 39, p. 169–176, 2012. EURO Mini Conference. ISSN 1571-0653. DOI: https://doi.org/10.1016/j.endm.2012.10.023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1571065312000248>.

CESCHIA, Sara; DI GASPERO, Luca; SCHAERF, Andrea. Educational timetabling: Problems, benchmarks, and state-of-the-art results. **European Journal of Operational Research**, v. 308, n. 1, p. 1–18, 2023. ISSN 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2022.07.011. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221722005641>.

CORMEN, T.H. et al. **Introduction to Algorithms**. 3rd. [S.l.]: The MIT Press, 2009.

DANTZIG, G.B. **Linear Programming and Extensions**. [S.l.]: Princeton University Press, 1963.

FEO, T.A.; RESENDE, M.G.C. Greedy Randomized Adaptive Search Procedures. **Journal of Global Optimization**, v. 6, p. 109–133, 1995.

FONSECA, George H.G.; SANTOS, Haroldo G. Variable Neighborhood Search based algorithms for high school timetabling. **Computers and Operations Research**, v. 52, 2014. ISSN 03050548. DOI: 10.1016/j.cor.2013.11.012.

FONSECA, George H.G.; SANTOS, Haroldo G.; CARRANO, Eduardo G. Integrating matheuristics and metaheuristics for timetabling. **Computers and Operations Research**, v. 74, 2016. ISSN 03050548. DOI: 10.1016/j.cor.2016.04.016.

GLOVER, F. W.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. 1st edition. [S.l.]: Springer, 2003.

GLOVER, Fred. Future paths for integer programming and links to artificial intelligence. **Computers & operations research**, Elsevier, v. 13, n. 5, p. 533–549, 1986.

GOH, Say Leng; KENDALL, Graham; SABAR, Nasser R. Improved local search approaches to solve the post enrolment course timetabling problem. **European Journal of Operational Research**, v. 261, 1 2017. ISSN 03772217. DOI: 10.1016/j.ejor.2017.01.040.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear: Modelos e Algoritmos**. 2nd edition. [S.l.]: Campus, 2005.

GOLDBARG, Marco. **Otimização Combinatória e Programação Linear**. [S.l.]: Elsevier, jan. 2005. ISBN 85-352-1520-4.

HANSEN, Pierre. The steepest ascent mildest descent heuristic for combinatorial programming. In: CONGRESS on numerical methods in combinatorial optimization, Capri, Italy. [S.l.: s.n.], 1986. P. 70–145.

HANSEN, Pierre; MLADENOVIĆ, Nenad. Variable Neighborhood Search. In: **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. Edição: Edmund K. Burke e Graham Kendall. Boston, MA: Springer US, 2005. P. 211–238. ISBN 978-0-387-28356-2. DOI: 10.1007/0-387-28356-0_8. Disponível em: <https://doi.org/10.1007/0-387-28356-0_8>.

KIRKPATRICK, S.; GELLAT, D.C.; VECCHI, M.P. Optimization by Simulated Annealing. **Science**, v. 220, p. 671–680, 1983.

LEITE, Nuno; MELÍCIO, Fernando; ROSA, Agostinho C. A fast simulated annealing algorithm for the examination timetabling problem. **Expert Systems with Applications**, v. 122, p. 137–151, 2019. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.12.048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417418308169>.

MARTÍ, R.; REINELT, G. **The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization**. 1st edition. [S.l.]: Springer, 2011.

OSMAN, I. H. Preface: Focused issue on applied meta-heuristics. **Computers & Industrial Engineering**, v. 44, n. 2, p. 205–207, 2003.

PAIVA, Emerson José de. Otimização de processos de manufatura com múltiplas respostas baseada em índices de capacidade., 2008.

RESENDE, Mauricio GC et al. A GRASP-Tabu search algorithm for solving school timetabling problems. **Metaheuristics: Computer decision-making**, Springer, p. 659–672, 2004.

SAMPAIO, Agnes Natasha Maciel de et al. Análise de risco e retorno entre diferentes tipos de carteiras de ações: uma abordagem usando a análise gray e a pesquisa operacional, 2016.

SAVINIEC, Landir; CONSTANTINO, Ademir Aparecido. Effective local search algorithms for high school timetabling problems. **Applied Soft Computing Journal**, v. 60, 2017. ISSN 15684946. DOI: 10.1016/j.asoc.2017.06.047.

SCARPIN, Cassius Tadeu et al. Técnicas da Pesquisa Operacional Aplicadas na Otimização do Fluxo de Pacientes do Sistema Único de Saúde do Estado do Paraná. **Trends in Computational and Applied Mathematics**, v. 8, n. 2, p. 299–308, 2007.

SILVA, Robson Ferreira da. **Tempos escolares: os horários escolares e o cotidiano docente**. 2019. Diss. (Mestrado) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo. DOI: 10.11606/D.100.2019.tde-10052019-154848.

SONG, Ting et al. An iterated local search algorithm for the University Course Timetabling Problem. **Applied Soft Computing**, v. 68, p. 597–608, 2018. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2018.04.034. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494618302254>.

SORIA-ALCARAZ, Jorge A. et al. Iterated local search using an add and delete hyper-heuristic for university course timetabling. **Applied Soft Computing**, v. 40, p. 581–593, 2016. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.11.043. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494615007760>.

SOUZA, Marcone Jamilson Freitas. Inteligência computacional para otimização. **Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto**, v. 6, 2008. Disponível em: <http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf>.

SOUZA, Marcone Jamilson Freitas et al. Métodos de pesquisa em vizinhança variável aplicados ao problema de alocação de salas. **XXII Encontro Nacional de Engenharia de Produção-ENEGEP, Curitiba, Brasil. Anais do XXII ENEGEP, CD-ROM**, 2002.

STEINER, Maria Teresinha Arns et al. Técnicas da pesquisa operacional aplicadas à logística de atendimento aos usuários de uma rede de distribuição de energia elétrica. **Sistemas & Gestão**, v. 1, n. 3, p. 229–243, 2006.

SUN, Zhe; WU, Qinghua. Two-phase tabu search algorithm for solving Chinese high school timetabling problems under the new college entrance examination reform. **Data Science and Management**, v. 6, n. 1, p. 55–63, 2023. ISSN 2666-7649. DOI: https://doi.org/10.1016/j.dsm.2023.02.001. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666764923000073>.

VAN BULCK, David; GOOSSENS, Dries. The international timetabling competition on sports timetabling (ITC2021). **European Journal of Operational Research**, v. 308, n. 3, p. 1249–1267, 2023. ISSN 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2022.11.046. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221722009201>.

WEISE, T. **Global Optimization Algorithms: Theory and Applications**. 2nd edition. [S.l.: s.n.], 2009.

WILLIAMSON, D. P.; SHMOYS, D. B. **The Design of Approximation Algorithms**. 1st edition. [S.l.]: Cambridge University Press, 2011.

WOLSEY, L. A. **Integer Programming**. [S.l.]: Wiley, 1998. (Wiley Series in Discrete Mathematics and Optimization).

XAVIER, Bruno Missi et al. Proposta de Alocação de Horários de Professores e Turmas em Instituições de Ensino Superior Utilizando uma Heurística VNS/VND. **Simpósio Brasileiro de Pesquisa Operacional–A Pesquisa Operacional na busca de eficiência nos serviços públicos/privados. Realizado nos dias**, v. 16, 2013.

ZAMANI KAFSHANI, Javad; MIRHASSANI, Seyyed Ali; HOOSHMAND, Farnaz. Adopting GRASP to solve a novel model for bus timetabling problem with minimum transfer and fruitless waiting times. **AUT Journal of Mathematics and Computing**, Amirkabir University of Technology, v. 1, n. 1, p. 125–134, 2020.