

Uma Ferramenta Baseada em Busca Tabu para Alocação de Docentes

José Eduardo Saroba Bieco

Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
Av. Trabalhador São-carlense, 400, 13560-970, São Carlos-SP, Brazil
jose.bieco@usp.br

Elias Salomão Helou Neto

Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
Av. Trabalhador São-carlense, 400, 13560-970, São Carlos-SP, Brazil
elias@icmc.usp.br

RESUMO

Este trabalho propõe uma ferramenta baseada na metaheurística Busca Tabu para otimizar a atribuição de docentes no SME do ICMC-USP. A solução foi desenvolvida com uma arquitetura modular, composta por blocos independentes para geração de vizinhança, avaliação, verificação da lista tabu e atualização da solução incumbente, facilitando a manutenção e a extensibilidade. Testes conduzidos com dados reais de dois semestres permitiram calibrar as penalizações aplicadas às violações de restrições flexíveis e ajustar parâmetros operacionais, como o tamanho da lista tabu e o número máximo de iterações. Os resultados demonstraram melhorias significativas na distribuição da carga docente e maior aderência às restrições institucionais. A ferramenta integra processos automatizados de otimização e uma etapa de validação manual, assegurando flexibilidade e eficiência no processo de atribuição, com potencial para apoiar a tomada de decisões em contextos acadêmicos complexos.

PALAVRAS CHAVE. Busca Tabu, Problema de Atribuição de Docentes, Balanceamento de Carga Didática.

MH — Metaheurísticas, OD - Otimização Discreta

ABSTRACT

This work proposes a tool based on the Tabu Search metaheuristic to optimize faculty allocation at SME of ICMC-USP. The solution was developed with a modular architecture, composed of independent blocks for neighborhood generation, evaluation, tabu list verification, and incumbent solution updating, facilitating maintenance and extensibility. Tests conducted with real data from two semesters allowed for the calibration of penalties applied to violations of soft constraints and the adjustment of operational parameters, such as the tabu list size and the maximum number of iterations. The results demonstrated significant improvements in the distribution of teaching workload and greater adherence to institutional constraints. The tool integrates automated optimization processes with a manual validation stage, ensuring flexibility and efficiency in the allocation process, with the potential to support decision-making in complex academic contexts.

KEYWORDS. Tabu Search, Faculty Assignment Problem, Teaching Load Balancing.

MH — Metaheuristics, OD — Discrete Optimization

1. Introdução

A atribuição de docentes a turmas em instituições de ensino superior é um problema de otimização combinatória conhecido como *Faculty Assignment Problem* (FAP) [Moreira e Costa, 2023]. O objetivo é encontrar, dentre as diversas configurações possíveis, a que melhor satisfaça um conjunto de restrições e maximize critérios como as preferências dos professores e o equilíbrio da carga horária. A resolução do FAP exige a consideração simultânea de disponibilidade docente, preferências individuais, normas institucionais e demandas curriculares.

Trata-se de um problema NP-difícil [Khorbotly e White, 2024], o que significa que não existem algoritmos de tempo polinomial capazes de encontrar a solução ótima para todas as suas instâncias [Schaerf, 1999]. O crescimento exponencial do espaço de busca com o aumento do problema inviabiliza o uso de métodos exatos — como programação inteira ou enumeração completa —, especialmente em cenários com múltiplos critérios e restrições.

No Departamento de Matemática Aplicada e Estatística (SME) do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), este processo tradicionalmente era realizado de forma manual ou semiautomática, com apoio de planilhas e formulários, exigindo alto esforço da comissão responsável e estando sujeito a inconsistências e retrabalho. Com o aumento da carga administrativa e da complexidade curricular, tornou-se essencial a adoção de ferramentas que otimizem este processo, assegurando eficácia e equidade nas atribuições.

Para solucionar o problema no contexto do SME, este trabalho propõe a aplicação da metaheurística **Busca Tabu**, considerando as particularidades da instituição. Uma contribuição central do modelo é a incorporação de *saldos*, que representam o histórico de carga de trabalho de cada docente em semestres anteriores e servem como critério para equilibrar a distribuição de turmas ao longo do tempo. Assim, o algoritmo tende a alocar mais turmas para docentes com saldo negativo (aqueles que lecionaram menos que o esperado), fomentando o reequilíbrio da carga de trabalho. Em contrapartida, professores com saldo positivo recebem menos turmas e são direcionados àquelas de maior interesse.

O processo tem início com a coleta de preferências via formulários digitais preenchidos pelos próprios docentes. Estas informações, anteriormente utilizadas apenas como apoio à alocação manual, agora alimentam diretamente o modelo de otimização. A solução obtida busca respeitar restrições como a obrigatoriedade de alocar todas as turmas, evitar conflitos de horário e considerar apenas preferências explicitamente registradas, alinhando as demandas institucionais com os interesses dos docentes.

Após a geração automática das atribuições, realiza-se uma etapa de validação manual que permite ajustes em situações excepcionais. O fluxo completo, incluindo tanto as etapas automatizadas quanto os momentos de intervenção da comissão, está representado na Figura 1. A integração entre automação e intervenção humana é amplamente recomendada na literatura, uma vez que o problema incorpora aspectos qualitativos e subjetivos que nem sempre são passíveis de formalização computacional. Além disso, a natureza combinatória da tarefa e a diversidade de critérios institucionais frequentemente exigem decisões estratégicas que se beneficiam do julgamento humano [Schaerf, 1999].

Dessa forma, este trabalho se posiciona com uma dupla contribuição: (i) uma contribuição de engenharia, ao entregar uma solução de software modular e robusta para um problema administrativo real e complexo; e (ii) uma contribuição científica inicial, ao formalizar este problema no contexto de uma instituição real e estabelecer uma solução base, abrindo portas para futuras pesquisas e comparações metodológicas neste cenário.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a metaheurística Busca Tabu e contextualiza o problema no SME; a Seção 4

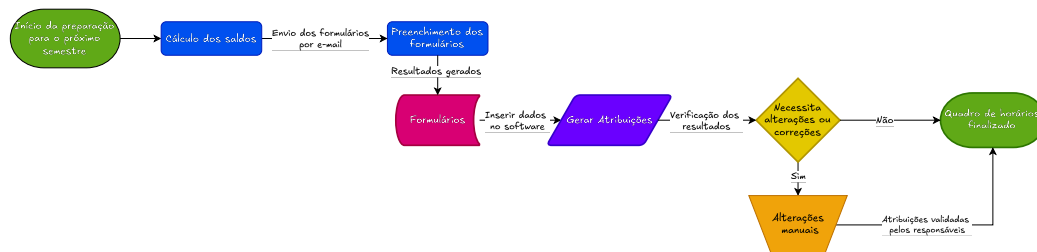


Figura 1: Fluxo do processo de atribuição de docentes no SME - ICMC, incluindo etapas automáticas e manuais.

detalha a metodologia de implementação; a Seção 5 discute os experimentos e seus resultados; e, por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

A literatura recente apresenta diversas abordagens computacionais para a alocação de recursos acadêmicos. Anthony e Mueller [2024] propõem uma ferramenta web de código aberto que, embora focada na geração de horários, utiliza programação inteira (via *PyGLPK*) para validar alterações e planejam expandir seu escopo para a atribuição de docentes. A interface simplificada do sistema visa ampliar a participação de diferentes *stakeholders*, oferecendo *feedback* imediato sobre a consistência das soluções.

No estudo de Khorbotly e White [2024], o FAP é tratado via programação linear (PL) em uma ferramenta web alimentada por planilhas. O objetivo é maximizar a correspondência entre as disciplinas e as preferências dos docentes, respeitando a carga horária e garantindo a cobertura completa das turmas. A solução promove transparência e auxilia na gestão estratégica de pessoal, permitindo também ajustes manuais.

De forma similar à nossa abordagem, o sistema HORARIS [Valdes et al., 2002] utiliza *Busca Tabu* como núcleo para a geração de horários acadêmicos. Sua arquitetura é estruturada em três fases (construção, melhoria e refinamento) e se destaca pela alta configurabilidade e pela interação contínua com o usuário.

Embora esses estudos demonstrem a relevância de sistemas de apoio à gestão acadêmica, este trabalho se diferencia por duas contribuições principais. A primeira é a adaptação da metaheurística *Busca Tabu* às particularidades institucionais do FAP no SME, incorporando regras de negócio específicas, como o sistema de *saldos* de carga horária para garantir equidade de longo prazo. Em segundo lugar, enquanto sistemas como o HORARIS oferecem configurabilidade, nossa solução avança ao apresentar uma arquitetura modular e extensível. Essa flexibilidade permite a configuração dinâmica de parâmetros do algoritmo — como pesos de penalizações, definição de restrições rígidas ou flexíveis, e seleção de critérios de parada e estratégias de vizinhança —, além de ter sido projetada para a fácil adição de novos mecanismos de busca, constituindo uma base robusta e adaptável para diferentes contextos do FAP.

3. Busca Tabu

A estratégia de otimização deste trabalho fundamenta-se na *Busca Tabu*, especificamente na abordagem *First Level Tabu Search Procedure* [Glover e Laguna, 1997]. A escolha desta versão foi motivada por seu comprovado equilíbrio entre simplicidade e eficácia na fuga de mínimos locais. Sua estrutura conceitual robusta e compreensível a torna particularmente adequada para implementações práticas. Além disso, sua implementação autocontida alinhou-se ao objetivo de desenvolver uma ferramenta web leve e de fácil implantação, sem a dependência de *solvers* de otimização externos ou que exigiriam um *backend* dedicado.

Memória de Curto Prazo

O principal componente da memória de curto prazo é a **Lista Tabu**, que armazena as últimas soluções visitadas para impedir o retorno imediato a elas e, assim, evitar ciclos. Diferentemente de implementações que memorizam movimentos específicos, nosso modelo armazena soluções completas em uma estrutura *FIFO* (*First In, First Out*) de tamanho configurável.

Nessa abordagem, o conceito de **Tabu Tenure** — o período em que uma solução é considerada tabu — é controlado implicitamente pelo tamanho da lista (*tabu size*). Uma solução permanece na lista e, portanto, "tabu", até ser naturalmente removida pela política FIFO. Consequentemente, o *tenure* de uma solução é efetivamente igual ao tamanho da lista.

Geração de Vizinhança

A geração de vizinhanças produz novas soluções candidatas a partir da modificação da solução corrente. Esse processo se dá pela aplicação de três operadores básicos: **adição**, **remoção** ou **troca** de atribuições. A solução inicial pode estar vazia ou conter alocações pré-definidas pelo usuário. A cada iteração, a vizinhança é gerada a partir da melhor solução encontrada na iteração anterior, promovendo uma busca progressiva.

O modelo manipula os seguintes componentes: o conjunto de docentes $D = \{D_1, \dots, D_n\}$, o conjunto de turmas $T = \{T_1, \dots, T_m\}$, os horários associados a cada turma $H = \{H_1, \dots, H_m\}$, e os saldos de carga horária de cada docente $S = \{S_1, \dots, S_n\}$.

Três matrizes auxiliares são utilizadas: a de preferências $P = \{p_{i,j}\}$, que armazena a prioridade do docente i pela turma j ; a de atribuições $X = \{x_{i,j}\}$, que indica com valor 1 se o docente i está alocado à turma j ; e a de travamento $F = \{f_{i,j}\}$, que fixa atribuições que não podem ser alteradas.

Durante a geração de vizinhos, apenas movimentos que respeitam duas **restrições rígidas** são considerados válidos. Primeiramente, um docente i só pode ser alocado a uma turma j se tiver manifestado interesse prévio ($p_{i,j} > 0$). Em segundo lugar, atribuições travadas ($f_{i,j} = 1$) não podem ser modificadas.

Avaliação da Solução

Cada solução candidata é avaliada por uma **função mérito**, que o algoritmo busca maximizar. Essa função quantifica a qualidade de uma atribuição a partir de dois componentes: um **custo**, que reflete o alinhamento com as preferências docentes, e um conjunto de **penalizações**, aplicadas quando restrições flexíveis são violadas. Essa abordagem permite que o algoritmo explore soluções temporariamente inviáveis (penalizadas) para escapar de mínimos locais e alcançar resultados globalmente superiores.

O custo da solução é calculado com base nas preferências $p_{i,j}$. Como a otimização visa maximização, as prioridades são invertidas para que as preferências mais altas (valores menores) resultem em maiores ganhos. Para isso, define-se $P_{\max} = \max(P) + 1$. O custo total é dado por:

$$\text{Custo} = \sum_{j \in T} \sum_{i \in D} K_1 x_{i,j} (P_{\max} - p_{i,j}),$$

onde K_1 é um fator de ponderação.

As penalizações, por sua vez, são subtraídas do custo e incidem sobre três tipos de violações:

Penalização 1: Conflito de Horários

A primeira penalização trata da alocação de um mesmo docente a turmas com horários conflitantes. Sendo X_i o conjunto de turmas atribuídas ao docente i , a penalização é calculada como:

$$\text{Pen}_H = \sum_{i \in D} \sum_{j \in X_i} \sum_{\substack{k \in X_i, \\ k > j}} K_2 x_{i,j} x_{i,k} \delta(H_j, H_k),$$

onde K_2 é o fator de penalização e $\delta(H_j, H_k)$ é uma função que retorna 1 se os horários das turmas j e k se sobrepõem, e 0 caso contrário.

Embora essa restrição seja, em muitos contextos, classificada como rígida, no cenário do SME ela foi tratada como *flexível*. Isso se deve a situações institucionais em que, apesar de turmas distintas constarem separadamente no planejamento acadêmico, por pertencerem a diferentes cursos ou grades curriculares, elas são efetivamente ministradas em conjunto, no mesmo horário e local. Dessa forma, conflitos aparentes no papel podem não representar problemas práticos, o que justifica o tratamento mais permissivo por meio de penalização.

Penalização 2: Turmas sem Docente

Para garantir que todas as disciplinas sejam cobertas, uma penalização é aplicada a cada turma que permanece sem docente. Utilizando a função indicadora $\mathbf{1}_{\{\cdot\}}$, que retorna 1 se a condição em seu subscrito é verdadeira e 0 caso contrário, a penalização é:

$$\text{Pen}_T = \sum_{j \in T} K_3 \mathbf{1}_{\sum_{i \in D} x_{i,j} = 0},$$

onde a condição $\sum_{i \in D} x_{i,j} = 0$ identifica que a turma j não possui nenhum docente atribuído. K_3 é o fator de penalização.

Penalização 3: Desequilíbrio da Carga de Trabalho

A terceira penalização incentiva a distribuição equilibrada do trabalho com base no saldo (S_i) de cada docente. Ela atua em duas frentes: penaliza a **sobrecarga** (docente com muitas turmas) e a **ociosidade** (docente sem turmas), com pesos modulados pelo saldo histórico. A fórmula é:

$$\text{Pen}_S = \sum_{i \in D} K_4 \left(\mathbf{1}_{\sum_{j \in T} x_{i,j} > 2} f(S_i) + \mathbf{1}_{\sum_{j \in T} x_{i,j} < 1} g(S_i) \right),$$

onde as funções indicadoras detectam sobrecarga (> 2 turmas) e ociosidade (< 1 turma). As funções adaptativas $f(S_i)$ e $g(S_i)$ ajustam a intensidade da penalização:

$$f(S_i) = \begin{cases} 0.75, & \text{se } S_i < -1.0 \\ 1.0, & \text{caso contrário} \end{cases} \quad g(S_i) = \begin{cases} 0.75, & \text{se } S_i > 1.0 \\ 1.0, & \text{caso contrário} \end{cases}.$$

A função $f(S_i)$ reduz a penalização por sobrecarga se o docente possui um saldo devedor ($S_i < -1.0$), enquanto $g(S_i)$ reduz a penalização por ociosidade se o docente já havia trabalhado acima da média ($S_i > 1.0$), promovendo uma distribuição mais justa ao longo do tempo.

É importante notar que, embora implementada como um componente de uma função de mérito mono-objetivo, a penalização associada aos saldos atua como um instrumento para um critério de equidade de longo prazo [Donahue e Kleinberg, 2020]. Essa abordagem traduz uma meta conceitualmente multi-critério [Miettinen, 1999] — maximizar as preferências atuais e, simultaneamente, garantir a justiça na distribuição de carga ao longo do tempo — em um único objetivo quantificável, alinhando-se aos princípios de *fairness-aware optimization* [Mehrabi et al., 2021] de maneira pragmática e computacionalmente eficiente para o escopo deste trabalho.

Solução Incumbente

Para garantir que a melhor solução encontrada durante todo o processo não seja perdida, o algoritmo mantém uma **solução incumbente**. Ela representa a melhor solução (com o maior valor na função mérito) identificada até o momento atual da busca. Ao final de cada iteração, a melhor solução vizinha (que não seja tabu ou que satisfaça um critério de aspiração) é comparada com a incumbente. Caso sua avaliação seja superior, ela se torna a nova solução incumbente. Ao término da execução, é esta solução que é retornada como o resultado final da otimização.

Critérios de Aspiração

O critério de aspiração permite que uma solução, mesmo que esteja na lista tabu, seja selecionada sob certas condições excepcionais. No entanto, na presente implementação, a aplicação do critério padrão — conhecido como *aspiração por objetivo* [Glover e Laguna, 1997] — se mostra redundante ou contraproducente devido à estratégia de armazenar soluções completas na lista tabu, em vez de movimentos.

A principal interpretação deste critério sugere quebrar o tabu se uma solução candidata for melhor que a solução incumbente. No nosso modelo, essa situação é uma impossibilidade lógica: como a incumbente já é, por definição, a melhor solução encontrada até a iteração atual, uma solução visitada anteriormente (e, portanto, tabu) não pode superá-la. Uma interpretação alternativa — permitir a quebra do tabu se um vizinho tiver avaliação superior à da solução que o gerou (a solução da iteração anterior) — também foi descartada. O risco desta abordagem é a criação de ciclos curtos: se o algoritmo faz um movimento não-excelente de uma solução S_A para uma solução S_B de menor qualidade, S_A se torna tabu. No passo seguinte, o critério de aspiração permitiria o retorno imediato a S_A , já que sua avaliação é superior à de S_B . Uma vez de volta a S_A , como o processo de geração de vizinhança é determinístico, o algoritmo inevitavelmente repetiria o mesmo passo para S_B , ficando preso no laço $S_A \rightarrow S_B \rightarrow S_A$. Por essas razões, embora o mecanismo de aspiração esteja previsto na arquitetura, ele não é utilizado ativamente na estratégia de busca atual.

Critérios de Parada

A execução da Busca Tabu é encerrada por um de dois critérios. O primeiro é um número máximo de iterações, previamente configurado pelo usuário. O segundo é a interrupção manual por meio da interface do sistema. Neste caso, para garantir a consistência do resultado, o algoritmo não para abruptamente; ele conclui a iteração corrente (incluindo geração, avaliação e possível atualização da incumbente) antes de finalizar a execução.

4. Metodologia de Implementação

A lógica da Busca Tabu foi implementada seguindo um fluxo de execução dividido em seis etapas principais: geração da vizinhança, avaliação dos vizinhos, verificação da lista tabu, seleção do melhor vizinho, atualização da solução incumbente e verificação dos critérios de parada.

Para traduzir este fluxo em um sistema robusto e flexível, a arquitetura foi projetada com foco em modularidade e extensibilidade. O princípio fundamental é que cada bloco funcional opera como um módulo independente que interage apenas por meio de interfaces bem definidas (entradas e saídas), sem depender do funcionamento interno dos demais. Para gerenciar a complexidade e evitar o uso de variáveis globais ou o repasse excessivo de parâmetros, foram criadas abstrações específicas para os principais componentes configuráveis do algoritmo.

Cada um desses componentes implementa uma interface padronizada. As **Restrições** (1), por exemplo, são classificadas como *rígidas* (verificando a viabilidade de um movimento) ou *flexíveis* (calculando penalizações) e devem expor os métodos apropriados para cada caso (`hard()` e `soft()`). Os componentes de **Movimento** (2) são responsáveis exclusivamente por gerar vizinhos a partir de uma solução base, via método `generate()`.

De forma análoga, os **Critérios de Aspiração** (3) implementam um método `fulfills()` para determinar se um vizinho tabu pode ser aceito, enquanto os **Critérios de Parada** (4) utilizam um método `stop()` para sinalizar o fim da execução. A própria **Lista Tabu** (5) foi abstraída para permitir diferentes estratégias de armazenamento (como soluções completas ou movimentos) por meio de uma interface comum que exige os métodos `has()`, `add()` e `remove()`.

A Figura 2 ilustra essa arquitetura por meio de um diagrama de classes. A figura evidencia a modularidade empregada e destaca a classe central `TabuSearch`, que orquestra a interação entre os diversos componentes configuráveis, encapsulando a lógica principal da busca.

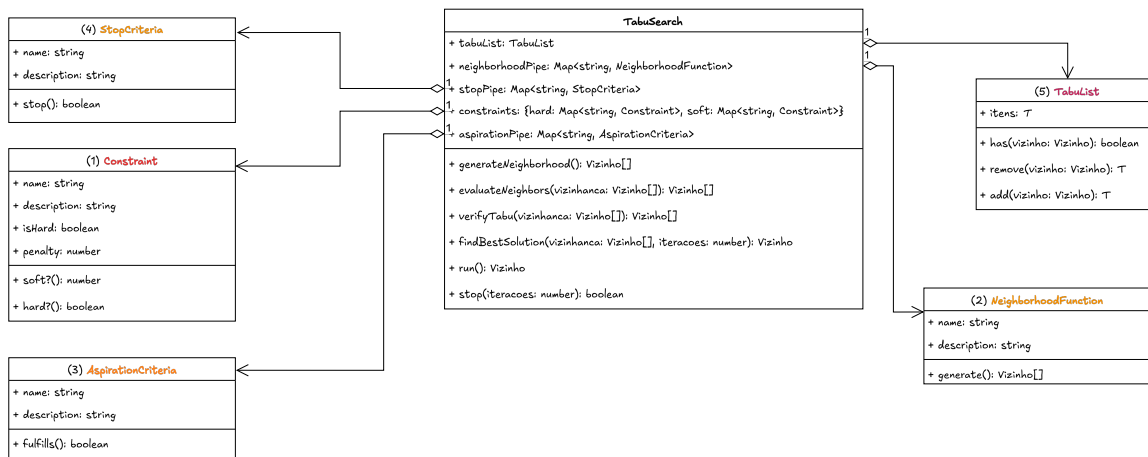


Figura 2: Diagrama de classes da arquitetura do sistema, com destaque para a classe TabuSearch e suas associações com os componentes configuráveis do algoritmo.

5. Experimentos e Resultados

Como discutido, a ferramenta foi desenvolvida para otimizar a alocação docente em um cenário institucional real. Contudo, devido à indisponibilidade de dados históricos de preferências docentes anteriores ao segundo semestre de 2024, os experimentos foram conduzidos nos dois únicos conjuntos de dados disponíveis: o segundo semestre de 2024 e o primeiro de 2025. Embora este escopo limitado de instâncias restrinja uma análise estatística aprofundada, ele oferece uma validação de alta fidelidade do modelo em cenários de aplicação autênticos.

Os dois cenários diferenciam-se principalmente pela quantidade de turmas, uma vez que os semestres ímpares historicamente concentram uma oferta maior. Ambos os conjuntos refletem a complexidade do ambiente real, incluindo inconsistências nos dados, como o não preenchimento do formulário de preferências por parte de alguns docentes. No cenário do segundo semestre de 2024 (Cenário 1), apenas um docente não registrou suas preferências, enquanto no primeiro semestre de 2025 (Cenário 2), esse número aumentou para seis.

Em termos de escala, o Cenário 1 compreende 48 turmas e 36 docentes elegíveis, com $P_{\max} = 14$. O Cenário 2, por sua vez, abrange 59 turmas e 37 docentes, com $P_{\max} = 13$. Em ambos os casos, não havia atribuições pré-fixadas (travas).

Valores das Penalizações

A definição dos valores dos multiplicadores da função objetivo (K_i) foi realizada de forma empírica, buscando refletir a importância relativa de cada restrição flexível. Adotou-se o multiplicador do custo principal (preferências), $K_1 = 10^3$, como valor de referência. A sobreposição de horários, considerada a violação mais crítica, recebeu o maior peso, $K_2 = 10^5$. As penalizações por ausência de docente em turmas e por desequilíbrio na carga de trabalho foram definidas, respectivamente, com $K_3 = 10^4$ e $K_4 = 10^3$. Tal parametrização visou direcionar a busca de forma eficaz, sem comprometer excessivamente a diversidade das soluções exploradas.

Ambiente de Teste

O algoritmo foi implementado em *TypeScript* (v5.8.3) e executado sobre *Node.js* (v20.17.0). A interface gráfica para interação e visualização dos resultados foi desenvolvida com o *framework* *Next.js* (v14.2.28).

As execuções foram realizadas em um computador pessoal com processador *AMD Ryzen 5 7535HS* e 16 GB de memória RAM, sob o sistema operacional *Windows 10*, a partir do navegador

Opera GX. Ainda que não represente um servidor dedicado, este ambiente mostrou-se suficiente para avaliar o desempenho e a viabilidade da ferramenta em condições análogas às de uso real.

Resultados Obtidos

Com os valores de penalização definidos inicialmente, o algoritmo foi executado por 300 iterações com uma lista tabu de tamanho 25. Nos resultados, observou-se que as restrições flexíveis foram violadas com uma frequência acima da esperada. Embora já se previssem violações na restrição de carga de trabalho (devido aos docentes que não preencheram o formulário), também ocorreram violações na alocação de turmas, com algumas ficando sem nenhum docente atribuído. Apesar destas falhas, a maior parte das atribuições concentrou-se em preferências de alto valor, conforme ilustra a Figura 3.

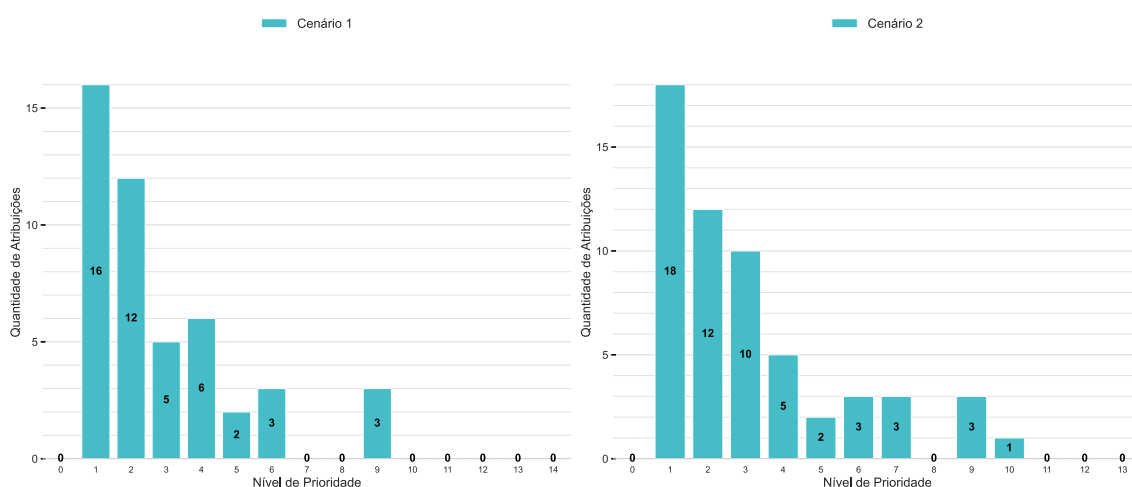


Figura 3: Distribuição da quantidade de atribuições por prioridade na configuração inicial: Cenário 1 (esquerda) e Cenário 2 (direita).

Em números, o Cenário 1 registrou uma turma sem docente e três violações de carga de trabalho (um docente sobrecarregado e dois sem atribuição). O Cenário 2 apresentou um quadro mais crítico: duas turmas sem docentes e dez violações de carga, sendo seis docentes sem atribuição (um número já esperado) e quatro com sobrecarga. A quantidade de professores sobrecarregados, especialmente no Cenário 2, indicou a necessidade de ajustes nos pesos das penalizações.

Refinamento dos Parâmetros da Busca

Com base no diagnóstico inicial, foram realizadas modificações nos parâmetros para melhorar a **qualidade** da solução, entendida aqui como a capacidade de satisfazer tanto as preferências docentes quanto as necessidades institucionais. Primeiramente, os valores das penalizações foram ajustados: o peso para turmas sem docente (K_3) foi aumentado para 10^6 , e o da carga de trabalho (K_4) para 10^4 , reforçando a importância dessas restrições. O multiplicador para conflito de horários (K_2) foi mantido em 10^5 , pois já apresentava desempenho satisfatório.

Uma segunda análise revelou que a busca estagnava prematuramente. No Cenário 1, a partir da iteração 63, e no Cenário 2, a partir da 104, os melhores vizinhos encontrados passaram a ter avaliações idênticas. Uma inspeção ao final da execução mostrou um grande número de vizinhos com o mesmo valor da solução incumbente (71 no Cenário 1 e 100 no Cenário 2). Esse platô de soluções de mesma qualidade indicava que a lista tabu de tamanho 25 era insuficiente para forçar a diversificação da busca. Diante disso, seu tamanho foi ampliado para 400.

Como consequência, o número máximo de iterações, antes fixado em 300, mostrou-se inadequado para explorar o espaço de busca. Para garantir que o algoritmo tivesse tempo suficiente para escapar de ótimos locais e explorar novas regiões, o limite foi ajustado para 1500 iterações.

Análise dos Resultados Finais

O novo conjunto de configurações impactou positivamente a qualidade das soluções. No Cenário 1, o total de violações foi reduzido de quatro para apenas uma (o docente que já era esperado não ser alocado). No Cenário 2, as violações caíram de doze para sete (os seis docentes sem formulário e apenas um com sobrecarga). Embora o Cenário 2 ainda apresente uma sobrecarga pontual, a melhoria é evidente, reduzindo de quatro docentes sobrecarregados para um e equilibrando melhor a distribuição geral.

No que diz respeito à distribuição por prioridade, os cenários apresentaram rebalanceamentos distintos. No Cenário 1, a otimização promoveu ajustes sutis, com uma leve redução nas atribuições de mais alta prioridade (prioridade 1 de 16 para 14, e prioridade 2 de 12 para 11). Essa diminuição foi compensada por um fortalecimento nas faixas intermediárias, com destaque para a prioridade 3, que aumentou de 5 para 7 alocações. Essa movimentação sugere que o algoritmo realizou trocas pontuais, sacrificando um pequeno volume de alocações de topo por uma maior diversidade, resultando no aumento da qualidade.

Em contrapartida, o Cenário 2 exibiu mudanças mais expressivas. Houve uma redução acentuada nas faixas de maior preferência, com a prioridade 1 caindo de 18 para 15 atribuições e a prioridade 3 diminuindo de 10 para 7. Enquanto faixas intermediárias como a prioridade 7 (de 3 para 5) e a 8 (de 0 para 2) tiveram aumentos, o ajuste mais significativo ocorreu na extremidade inferior da escala, com a prioridade 10 saltando de 1 para 6 alocações. Este comportamento indica que, para este cenário mais complexo, a busca por uma solução globalmente melhor exigiu um sacrifício considerável nas preferências mais altas.

As alterações descritas podem ser observadas na Figura 4. As linhas interpoladas indicam os resultados obtidos com as configurações iniciais, permitindo uma comparação visual direta.

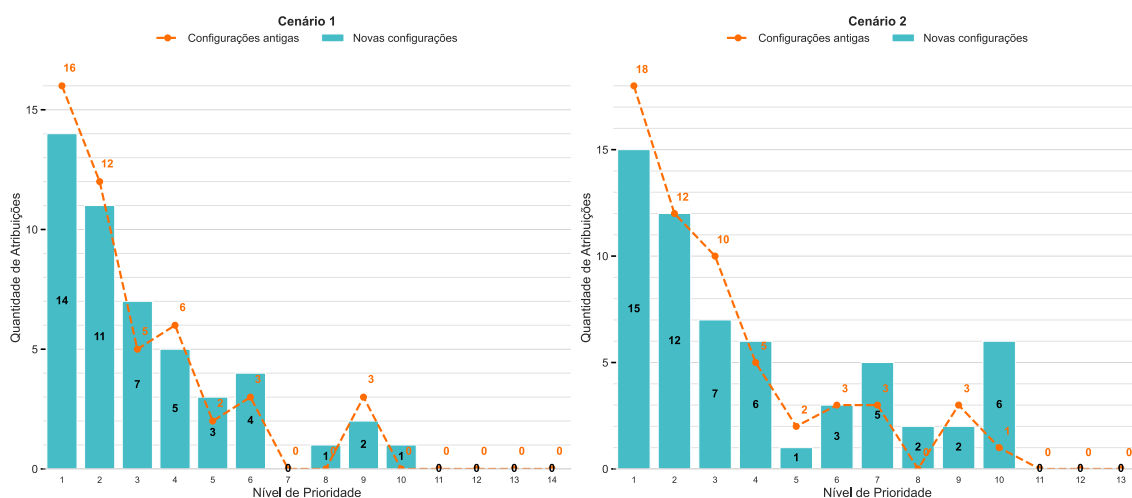


Figura 4: Distribuição da quantidade de atribuições por prioridade após o refinamento dos parâmetros. A interpolação sobreposta indica os resultados da configuração inicial para comparação.

Em síntese, os resultados demonstram que o refinamento iterativo dos parâmetros foi eficaz para melhorar a qualidade das soluções, especialmente na distribuição da carga docente. Mesmo que

o Cenário 2 ainda apresente desafios, os avanços significativos reforçam a viabilidade da ferramenta para apoiar processos reais de atribuição, ao mesmo tempo em que evidenciam oportunidades para refinar ainda mais os mecanismos de otimização.

6. Conclusão

Este trabalho apresentou o desenvolvimento e a aplicação de uma ferramenta baseada na metaheurística Busca Tabu para otimizar a atribuição de docentes no SME. A solução buscou alinhar as preferências dos docentes às necessidades institucionais por meio de uma função mérito que equilibra restrições rígidas e flexíveis. Os experimentos em cenários reais demonstraram a eficácia e a adaptabilidade da abordagem, que, após a calibração de seus parâmetros, foi capaz de gerar soluções de alta qualidade, mais equilibradas e com menos violações. O sucesso desta aplicação prática não apenas resolve um problema crítico para o departamento, mas também estabelece um *baseline*, abrindo oportunidades para futuras pesquisas em otimização neste cenário institucional.

Apesar dos resultados promissores, o estudo revelou oportunidades para avanços futuros. Uma frente promissora é a incorporação de estocasticidade na geração de vizinhança para aumentar a diversificação da busca, seja pela modificação de um número aleatório de atribuições, seja pela perturbação da solução base. Outra melhoria seria a implementação de uma lista tabu dinâmica, motivada pela observação de que muitos vizinhos possuíam avaliações idênticas, indicando uma oportunidade para refinar a exploração do espaço de busca. Uma terceira vertente, aprofundando a discussão sobre equidade, seria explorar formulações alternativas ao modelo mono-objetivo. Isso incluiria tratar o problema como explicitamente multi-objetivo, buscando a fronteira de Pareto entre a satisfação docente e a equidade de carga [Miettinen, 1999], ou impor métricas de justiça como restrições formais, a exemplo do critério de α -fairness [Donahue e Kleinberg, 2020], em vez de aproximá-las por penalizações.

Por fim, para validar a qualidade e a robustez das soluções de forma mais ampla, planeja-se a criação de um gerador de instâncias sintéticas. A disponibilidade de uma gama maior de cenários permitirá não apenas testar a escalabilidade do algoritmo, mas também conduzir uma análise comparativa extensiva. Esta análise incluirá a implementação de outras metaheurísticas clássicas (como *Simulated Annealing* e Algoritmos Genéticos) e a finalização da validação de um modelo de Programação Linear Inteira, que está em desenvolvimento. A comparação com um método exato será fundamental para avaliar tanto a proximidade da solução da Busca Tabu ao ótimo quanto sua eficiência computacional.

O código-fonte da ferramenta, bem como as instâncias utilizadas, estão publicamente disponíveis no repositório GitHub: `InvOptLab/distribuidor-de-carga`. A plataforma pode ser acessada em: <https://distribuidor-de-carga.vercel.app>.

Agradecimentos

Este trabalho foi apoiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) sob bolsa 88887.083738/2024-00.

Referências

- Anthony, B. M. e Mueller, M. (2024). Course scheduling made easier: A user-friendly web-based timetabling tool using pyglpk. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 15158 LNCS:243–252. ISSN 03029743.
- Donahue, K. e Kleinberg, J. (2020). Fairness and utilization in allocating resources with uncertain demand. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20*, p. 658–668. ACM.

- Glover, F. e Laguna, M. (1997). *Tabu Search*. Springer Book Archive. Springer New York, NY, 1 edition. ISBN 978-0-7923-9965-0.
- Khorbotly, S. e White, D. (2024). A preference-based faculty-assignment tool for course scheduling optimization. In *2024 ASEE Annual Conference & Exposition*, p. 1–8, Portland, Oregon. American Society for Engineering Education. DSA Technical Session 4, Data Science & Analytics Constituent Committee (DSA).
- Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., e Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.
- Miettinen, K. (1999). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.
- Moreira, J. J. e Costa, J. (2023). Automatic resolution of teacher assignment problem: A bibliometric analysis from 1976 to 2022. In *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, volume 1, p. 1–7.
- Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.
- Valdes, R. A., Crespo, E., e Tamarit, J. M. (2002). Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, 137(3):512–523. ISSN 0377-2217.