Calendário Unificado: Otimização no Agendamento de Avaliações Acadêmicas

Karen Silva Pacheco
DRE:123476904
Disciplina: Otimização
UFRJ — 2025.2

3 de dezembro de 2025

1 Introdução

O Calendário Unificado aborda um problema recorrente em períodos letivos: a concentração de avaliações (provas, apresentações, entregas) em dias ou semanas específicas, especialmente no meio e no fim do semestre, gerando picos de carga e conflitos entre disciplinas cursadas pelos mesmos alunos.

Evidências empíricas demonstram que a distribuição das avaliações afeta diretamente o desempenho.

O estudo *Marathon, Hurdling or Sprint? The Effects of Exam Scheduling on Academic Performance* (Goulas & Megalokonomou, IZA DP 11624) analisou dados de milhares de estudantes com sorteio aleatório das datas de exames. Os resultados mostram que, quando um exame ocorre após uma sequência de provas próximas, o desempenho do aluno cai entre 7% e 10% em áreas STEM devido à fadiga cognitiva acumulada.

Um estudo britânico (*Exam Scheduling and Student Performance*, 2013) mostrou que mudanças no calendário que concentraram exames finais provocaram redução estatisticamente significativa nas notas, especialmente entre estudantes com maior carga de disciplinas.

Diante desses problemas, este trabalho propõe formular o agendamento de avaliações como um problema de otimização.

Para cada turma e para cada avaliação, o sistema recebe:

- um intervalo de datas permitido pelo docente;
- o calendário de provas já marcadas em turmas que compartilham alunos;
- restrições de dias de aula, feriados e datas bloqueadas.

Com base nisso, o modelo escolhe uma data principal e produz datas alternativas quase-ótimas, buscando minimizar a sobrecarga dos alunos, evitar conflitos e distribuir melhor a carga de avaliações ao longo das semanas.

2 Definição do Problema

2.1 Decisão

O modelo é aplicado localmente, isto é, para uma turma por vez.

Dada uma turma c e um intervalo de datas $[d_{\min}, d_{\max}]$, o sistema deve realizar duas tarefas principais:

- selecionar uma data principal dentre as datas elegíveis;
- gerar uma lista de datas alternativas quase-ótimas (Top-K), ordenadas pela função objetivo, permitindo remarcação ou ajustes manuais pelo docente.

2.2 Objetivo

A função objetivo combina diversos critérios:

- **Minimizar o pico de carga diária (T)**: representa a pior situação de um estudante em termos de número de avaliações naquele dia.
- **Minimizar conflitos**: reduz o número de avaliações coincidentes no mesmo dia para os estudantes afetados.
- **Maximizar o espaçamento entre avaliações**: favorece datas mais distantes de outras avaliações já realizadas ou agendadas.
- **Reduzir a variabilidade semanal**: evita inserir avaliações em semanas já sobrecarregadas.

Esses componentes são ponderados por parâmetros α, β, γ, compondo a função objetivo:

$$\min \left(T + \alpha \cdot \text{conflitos} - \beta \cdot \text{espacamento} + \gamma \cdot \text{variabilidade} \right).$$

2.3 Restrições

O modelo respeita as seguintes restrições:

- **Exclusividade:** cada turma recebe exatamente uma data dentro do intervalo elegível.
- **Janela pedagógica:** apenas datas entre $d_{\min}$ e $d_{\max}$ são consideradas.
- **Dias de aula:** avaliações só podem ocorrer nos dias semanais em que a turma tem aula.
- **Calendário acadêmico:** feriados e datas institucionalmente bloqueadas são excluídos.
- **Compatibilidade com outras avaliações:** conflitos e semanas carregadas são penalizados pela função objetivo.

2.4 Conjuntos e Entidades

Para uma chamada do modelo:

- **Turma** c: oferta da disciplina.
- **Estudantes** S: estudantes da turma e das turmas que compartilham alunos com c.
- **Datas elegíveis** D: datas úteis, dias de aula, não feriados e não bloqueadas.

Além disso, o modelo recebe:

- **Avaliações existentes**: conjunto de provas já marcadas e alunos impactados.
- **Matrículas estudante–turma**: mapeia quais avaliações afetam os mesmos estudantes.

2.5 Entradas

- **Informações da turma**: identificador e dias de aula.
- **Intervalo informado pelo docente**: $intervaloInicial$, $intervaloFinal$.
- **Avaliações existentes**: turma, data, professor, alunos impactados.
- **Inscrições**: relação estudante–turma.
- **Estudantes relevantes**: alunos da turma analisada.
- **Calendário institucional**: feriados e datas bloqueadas.

2.6 Saídas

- **Data principal recomendada (*melhorData*)**: data que minimiza a função objetivo.
- **Top-K alternativas (*melhoresAlternativas*)**: datas elegíveis adicionais ordenadas pela função objetivo.

3 Métricas Usadas na Avaliação

Para analisar o comportamento do modelo, são consideradas as seguintes métricas:

Pico máximo diário por estudante (T)

Representa o maior número de avaliações que qualquer estudante enfrenta em um único dia após incluir a nova prova no calendário.

Número de conflitos evitados

Quantifica situações em que a escolha de data pelo modelo reduz casos em que dois ou mais exames ocorreriam no mesmo dia para o mesmo estudante, comparando-se com datas alternativas possíveis.

Espaçamento entre avaliações

- **Espaçamento mínimo:** distância (em dias) entre a nova avaliação e a avaliação já existente mais próxima para qualquer estudante (pior caso).
- **Espaçamento médio:** média das distâncias entre a nova avaliação e as avaliações anteriores para o conjunto de estudantes impactados.

Variabilidade semanal da carga

Avalia a distribuição de avaliações ao longo das semanas:

- **Alta variabilidade:** semanas muito carregadas e semanas quase vazias.
- **Baixa variabilidade:** distribuição mais uniforme da carga semanal.

O modelo foi construído de forma que essas métricas estejam:

- explicitamente presentes na função objetivo (como T, conflitos e espaçamento); ou
- implicitamente representadas por variáveis auxiliares (como a variabilidade semanal).

4 Conjunto de Dados

O modelo é testado sobre instâncias fictícias construídas a partir de estruturas similares às do SIGA, porém simplificadas para foco exclusivo no problema de agendamento.

4.1 Turma

Para uma execução do solver, considera-se apenas um identificador de turma, por exemplo:

- `turma` (ex.: "COS360-A").

Esse identificador é utilizado para:

- determinar quais estudantes pertencem à turma, via `inscricoes` e `estudantes_turma`;
- permitir, na camada de aplicação, distinguir avaliações "minhas" e "de outros professores" no calendário exibido.

Não há necessidade, na formulação matemática, de atributos como nome da disciplina, código ou curso, pois tais informações não influenciam a escolha da data. Assim, tais campos não aparecem no modelo nem no JSON de entrada do solver.

4.2 Inscrições Efetivadas (estudante–turma)

As inscrições são representadas por pares:

- `estudante_id`
- `turma_id`

Elas são utilizadas para:

- identificar quais avaliações existentes (`avaliacoes_existentes`) impactam os estudantes listados em `estudantes_turma`;
- construir a carga fixa diária carga_fixa$[s, d]$ para cada estudante s em cada data d.

Assume-se que esses dados já representam a matrícula consolidada, isto é, após encerramento das movimentações acadêmicas, não contendo adições ou remoções temporárias de estudantes.

4.3 Intervalo de Datas Elegíveis

O intervalo permitido para a nova avaliação é definido por:

- `intervaloInicial`
- `intervaloFinal`

Esse intervalo é refinado por:

- `dias_aula`: define os dias da semana em que a turma pode realizar avaliações;
- `feriados`: exclui datas classificadas como feriado;
- `datas_bloqueadas`: exclui datas vetadas pela instituição.

A partir dessas informações, o solver gera internamente o conjunto D de datas elegíveis, consistindo apenas de dias válidos segundo todas as restrições listadas.

4.4 Avaliações Existentes

A lista `avaliacoes_existentes` contém, para cada prova já agendada:

- `turma`
- `data`
- `professor_id`

Combinando essa tabela com `inscricoes`, é possível reconstruir:

- a carga fixa diária carga_fixa$[s, d]$: quantas avaliações cada estudante s possui em cada data d;
- as datas de avaliações anteriores necessárias para calcular o espaçamento mínimo por estudante;
- a carga semanal carga_semana(d): número de avaliações existentes na semana associada à data d.

5 Modelo Matemático

O problema é formulado como um modelo de Programação Linear Inteira Binária (0–1 MILP), resolvido para uma turma por vez. O modelo utiliza variáveis binárias para a escolha da data, além de variáveis contínuas auxiliares, compondo uma função objetivo linear que combina múltiplos critérios.

5.1 Conjuntos, Parâmetros e Notação

Conjunto de datas elegíveis:

$$D: \text{ conjunto de datas elegíveis}$$

Ou seja, conjunto de datas elegíveis, derivado de intervaloInicial, intervaloFinal, dias_aula, feriados e datas_bloqueadas.

Conjunto de estudantes:

$S:$ conjunto de estudantes considerados, obtido a partir de *estudantes_turma* e *inscricoes*.

Parâmetros:

$\text{carga\_fixa}_{s,d}:$ número de avaliações já existentes para o estudante s na data d.

$\Delta_{s,d}:$ distância (em dias) entre a data d e a avaliação mais próxima do estudante s.

$\text{carga\_semana}(d):$ carga de avaliações na semana correspondente à data d.

α, β, $\gamma:$ pesos da função objetivo para conflitos, espaçamento e variabilidade.

Esses parâmetros são derivados automaticamente a partir dos dados de entrada do JSON; não são fornecidos diretamente pelo usuário.

5.2 Variáveis de Decisão

Variáveis binárias:

$$x_d \in \{0,1\}, \qquad \forall d \in D,$$

onde $x_d = 1$ significa que a nova avaliação será marcada na data d.

Variáveis auxiliares contínuas:

$$T \geq 0, \qquad \text{conflitos} \geq 0, \qquad \text{espacamento} \geq 0, \qquad \text{variabilidade} \geq 0.$$

5.3 Função Objetivo

$$\min\left(T + \alpha \cdot \text{conflitos} - \beta \cdot \text{espacamento} + \gamma \cdot \text{variabilidade}\right).$$

Interpretações:

- T penaliza picos de avaliações diários.
- conflitos penaliza datas com carga pré-existente elevada.
- espacamento entra com sinal negativo, favorecendo datas mais afastadas.
- variabilidade penaliza semanas já muito carregadas.

5.4 Restrições

Escolha exclusiva de uma única data:

$$\sum_{d \in D} x_d = 1.$$

Definição do pico máximo por estudante:

$$\text{carga\_fixa}_{s,d} + x_d \leq T, \qquad \forall s \in S,\, \forall d \in D.$$

Definição da variável de conflitos:

$$\text{conflitos} = \sum_{d \in D} x_d \left(\sum_{s \in S} \text{carga\_fixa}_{s,d} \right).$$

Definição da variável de espaçamento:

$$\text{espacamento} = \sum_{d \in D} x_d \left(\sum_{s \in S} \Delta_{s,d} \right).$$

Definição da variabilidade semanal:

$$\text{variabilidade} = \sum_{d \in D} x_d \cdot \text{carga\_semana}(d).$$

Domínio das variáveis:

$$x_d \in \{0,1\}, \quad T \geq 0, \quad \text{conflitos} \geq 0, \quad \text{espacamento} \geq 0, \quad \text{variabilidade} \geq 0.$$

5.5 Categoria do Modelo e Implementação

O modelo pertence à classe de Programação Linear Inteira Binária (Binary Integer Linear Programming — BILP) e é resolvido com o solver CBC (Coin-or Branch-and-Cut) via biblioteca OR-Tools em Python.

O fluxo de implementação consiste em:

1. ler o JSON de entrada;

2. construir o conjunto D de datas elegíveis;
3. calcular os parâmetros derivados (carga_fixa, $\Delta_{s,d}$, carga_semana(d));
4. montar o modelo MILP usando a API `pywraplp`;
5. resolver o modelo e identificar a data ótima (`melhorData`);
6. reavaliar todas as demais datas elegíveis com a mesma função objetivo para produzir um ranking das alternativas (`melhoresAlternativas`).

O resultado final consiste na data ótima e em uma lista de alternativas ordenadas de acordo com a "qualidade" segundo a função objetivo do modelo.

6 Desenvolvimento da Aplicação Web

Para ilustrar o uso prático do modelo de otimização, foi desenvolvida uma aplicação web que permite ao docente visualizar o calendário de avaliações da turma e solicitar sugestões de datas. A solução foi implementada em uma arquitetura cliente–servidor, composta por três módulos principais: frontend, backend e solver de otimização.

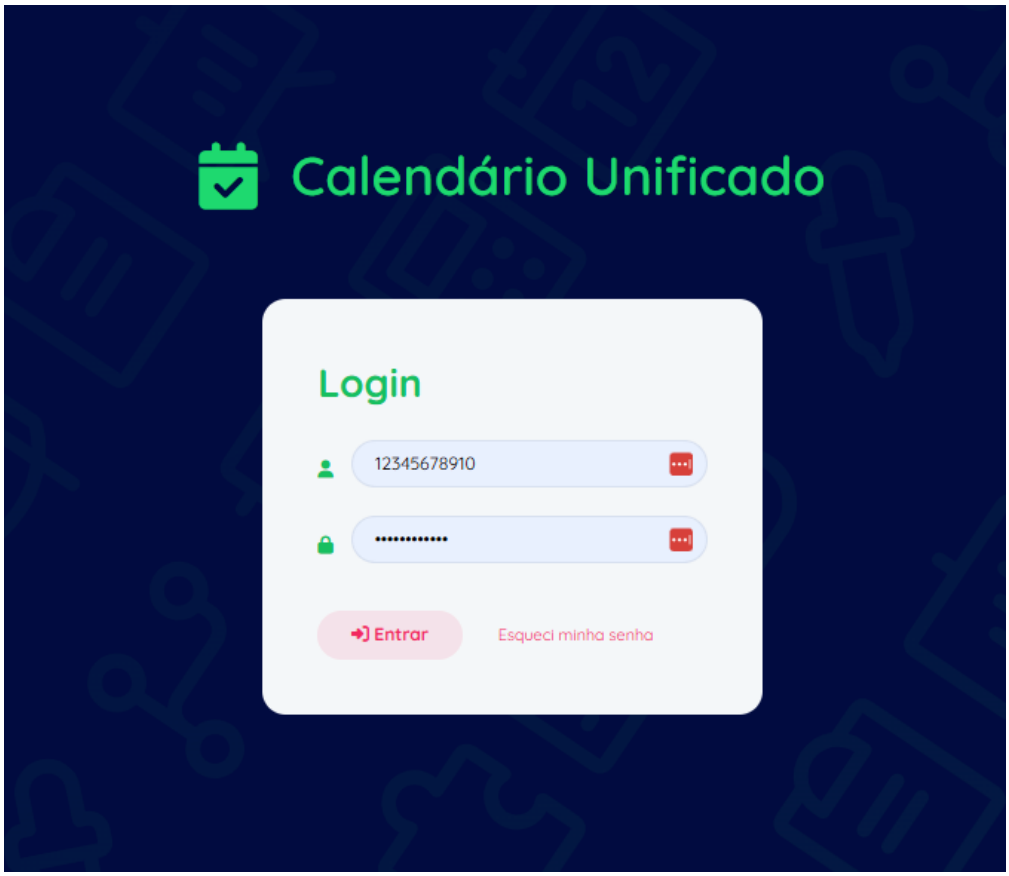

Figura 1: Exibição da Interface – Tela de Login

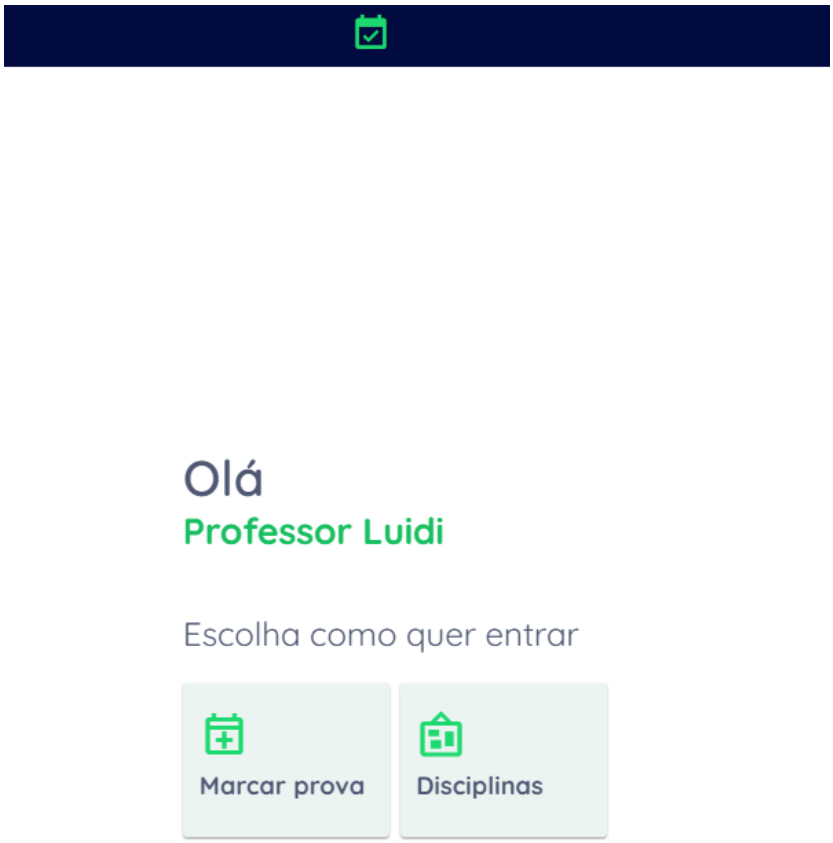

Figura 2: Exibição da Interface – Seletor de Turmas

Figura 3: Exibição da Interface – Tela de Turmas

Figura 4: Exibição da Interface – Calendário

6.1 Frontend (Cliente)

O frontend foi desenvolvido com tecnologias web modernas para proporcionar uma interface simples, reativa e responsiva.

- Construído com **HTML**, **CSS** e **JavaScript**, utilizando o framework **Vue.js** na forma de *Single File Components* para organização da lógica de interface.
- Utilização de componentes e ícones no estilo **Material Design** para botões, marcações de prova e elementos de navegação.
- Uso da biblioteca **date-fns** para manipulação de datas: cálculo de dias do mês, mudança de mês, operações de diferença de dias e formatação.
- Comunicação com o backend via **Fetch API**, consumindo endpoints REST:
  - `/api/turmas`

- `/api/avaliacoes`
- `/api/sugestoes`
- `/api/criar-avaliacao`

O frontend exibe um calendário mensal que mostra as avaliações já agendadas e, ao solicitar sugestões de data, apresenta a recomendação do modelo e alternativas quase-ótimas.

6.2 Backend (Servidor Web)

O backend foi implementado em **Node.js** utilizando o framework **Express**. Seu papel é realizar a mediação entre o frontend e o solver de otimização.

- A API REST expõe os mesmos endpoints consumidos pelo frontend.
- Utilização do middleware **cors** para permitir acesso do frontend durante o desenvolvimento local.
- Os dados acadêmicos utilizados no cenário (turmas, inscrições, avaliações já marcadas, feriados etc.) são carregados a partir de um arquivo JSON de *mock*.
- Para resolver o problema de otimização, o backend invoca um script Python (`solver.py`) por meio de:

  `child_process.execFile`

  enviando o JSON de entrada para o modelo MILP.

O backend recebe a solução do solver, formata o resultado e devolve ao frontend as recomendações de datas para exibição ao docente.

6.3 Solver de Otimização

O módulo de otimização foi implementado em **Python**, utilizando a biblioteca **OR-Tools** (módulo `pywraplp`) para resolver o modelo inteiro-binário definido na Seção 5.

- Recebe um JSON contendo informações da turma, estudantes, avaliações existentes e intervalo desejado.
- Constrói o conjunto de datas elegíveis e calcula os parâmetros derivados (carga fixa, espaçamento, carga semanal).
- Resolve o modelo MILP utilizando o solver CBC.
- Retorna:
  - a data ótima (*melhorData*);
  - um ranking de alternativas quase-ótimas (*melhoresAlternativas*).

Os resultados retornados são utilizados diretamente na interface web para apoiar o docente na escolha da data de avaliação.

7 Resultados

Esta seção apresenta dois cenários de demonstração realizados diretamente na interface web. Cada cenário ilustra como o modelo reage a diferentes distribuições de avaliações existentes e como o frontend exibe a melhor data e as alternativas quase-ótimas.

7.1 Exemplo 1: Maior espaçamento levando em conta cargas distintas

O intervalo permitido para a avaliação é:

$$\text{intervaloInicial} = 10/12/2025, \qquad \text{intervaloFinal} = 20/12/2025.$$

As avaliações já registradas para o estudante da turma são:

- 10/12 — duas avaliações já existentes;
- 19/12 — uma avaliação existente.

As datas elegíveis da turma são:

$$\{10/12,\ 12/12,\ 17/12,\ 19/12\}.$$

A data de 10/12 concentra duas provas, gerando uma carga elevada e sendo fortemente penalizada pelo modelo.

A data de 19/12 possui apenas uma prova, porém encontra-se próxima demais da prova já existente no mesmo dia. Em contraste, a data de 17/12 apresenta maior distância em relação às datas carregadas (10/12 e 19/12), resultando no melhor balanço entre:

- redução de conflitos;
- maximização do espaçamento mínimo;
- suavização da variabilidade semanal.

Assim, a data mais vantajosa é:

$$\boxed{17/12/2025}$$

O solver retorna:

$$\texttt{melhorData} = 17/12/2025$$

$$\texttt{melhoresAlternativas} = \{12/12,\ 19/12,\ 10/12\}.$$

Figura 5: Interface exibindo recomendação e alternativas para o Exemplo 1.

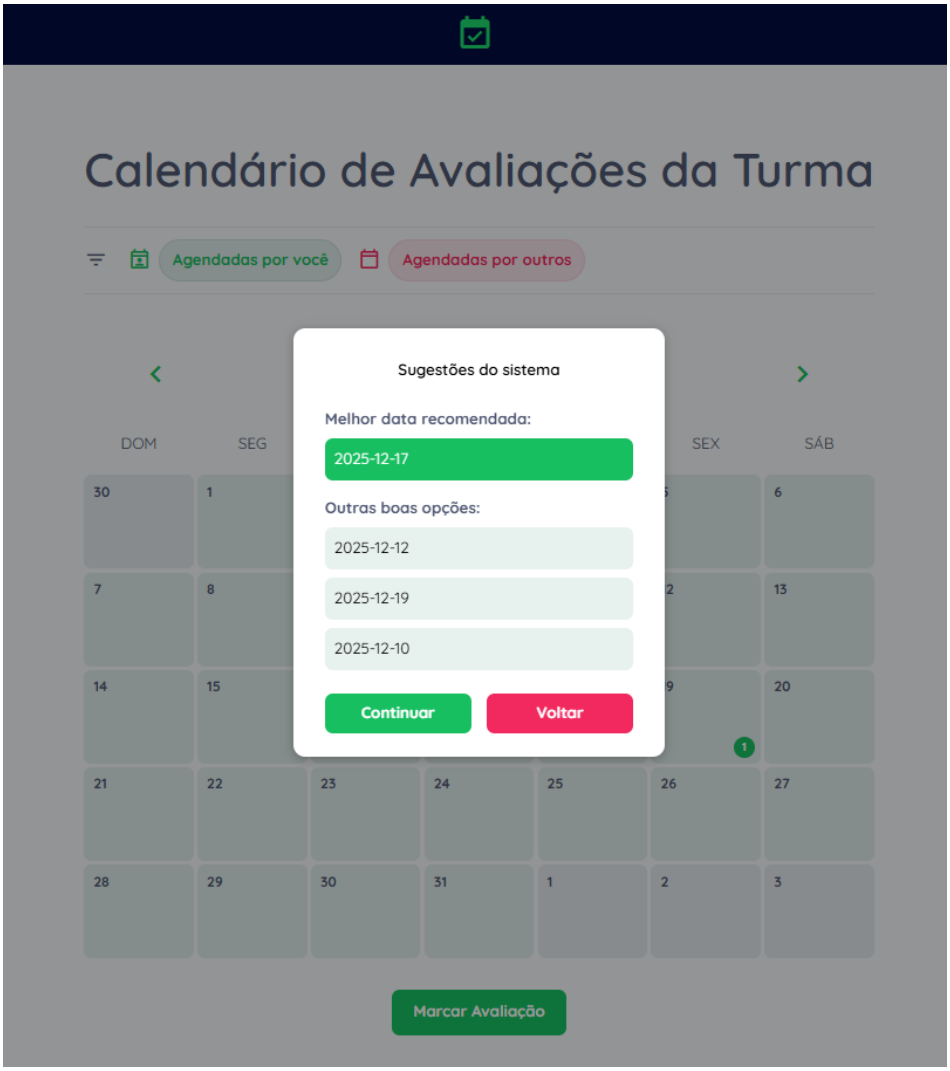

Figura 6: Interface exibindo recomendação e alternativas para o Exemplo 1.

7.2 Exemplo 2: Escolha guiada pelo maior afastamento da última avaliação

Neste cenário, o estudante possui duas avaliações recentes na mesma semana:

- 10/12 — uma avaliação;
- 12/12 — uma avaliação.

As datas elegíveis da turma são:

$$\{10/12,\ 12/12,\ 17/12,\ 19/12\}.$$

As datas de 10/12 e 12/12 coincidem com avaliações já registradas, sendo penalizadas pelo componente de conflito. As datas de 17/12 e 19/12, por outro lado, estão livres e pertencem à semana seguinte.

A decisão do solver decorre da distância em dias até a última avaliação (12/12):

$$\text{distância}(12/12,\ 17/12) = 5 \text{ dias}, \qquad \text{distância}(12/12,\ 19/12) = 7 \text{ dias}.$$

Como o modelo busca maximizar o espaçamento e ao mesmo tempo minimizar conflitos acumulados na semana corrente, a data mais favorável é:

$$\boxed{19/12/2025}$$

O sistema retorna:

$$\texttt{melhorData} = 19/12/2025$$

$$\texttt{melhoresAlternativas} = \{17/12,\ 10/12,\ 12/12\}.$$

A Figura 8 apresenta a captura de tela correspondente.

Figura 7: Interface exibindo recomendação e alternativas para o Exemplo 2.

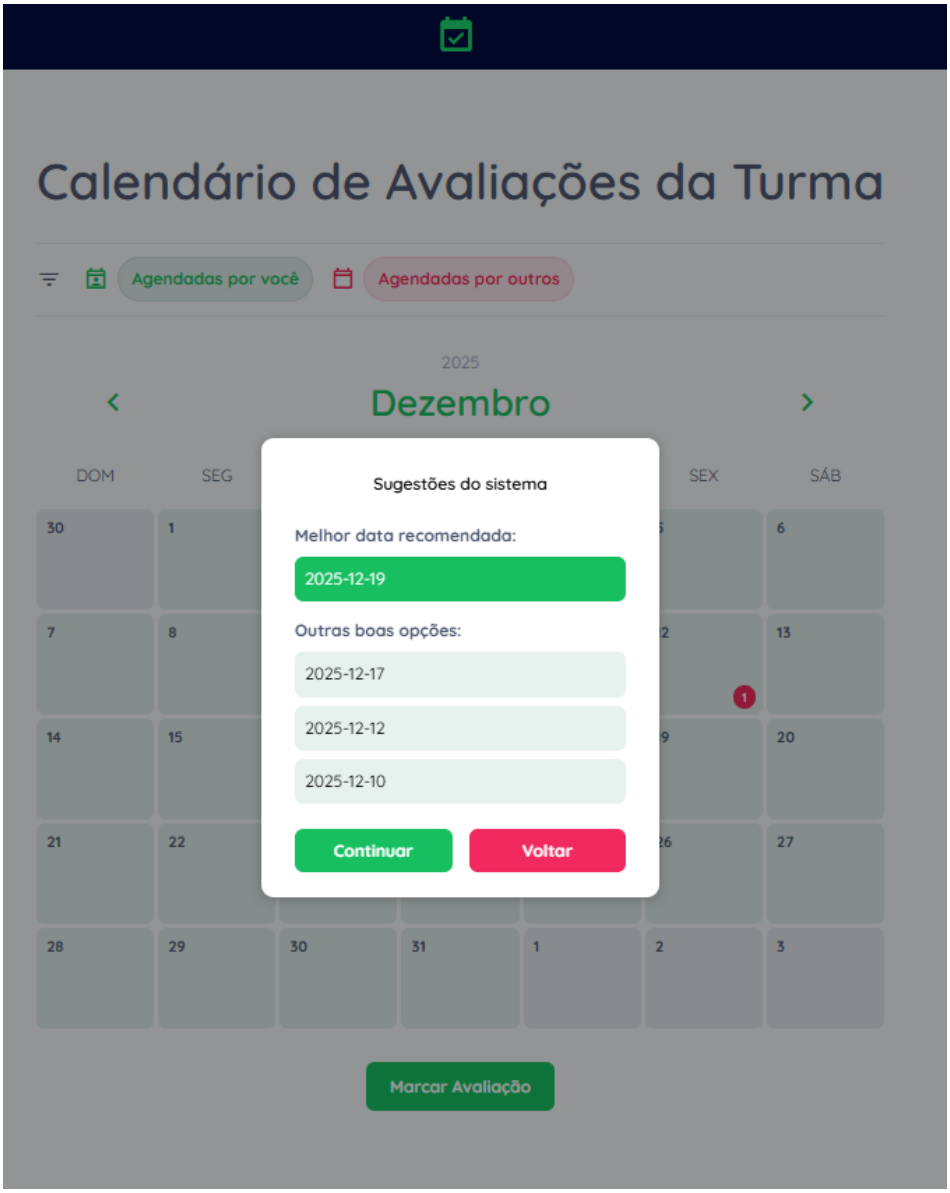

Figura 8: Interface exibindo recomendação e alternativas para o Exemplo 2.

7.3 Síntese dos resultados

Os testes demonstram que:

- o modelo responde adequadamente a diferenças de carga e espaçamento;
- quando há conflitos, ele prioriza a data de menor impacto;
- quando as cargas são equivalentes, ele recorre ao espaçamento e variabilidade para desempate;

Essas demonstrações confirmam que a solução integrada entre frontend, backend e solver oferece suporte prático ao docente na escolha da melhor data de avaliação.

Material

Todo o código-fonte utilizado neste trabalho — incluindo o modelo de otimização, os scripts de execução do solver e os componentes da interface web — está disponibilizado no repositório público:

`https://github.com/srtapacheco/otimizacao`

Além disso, todos os arquivos relevantes (código, dataset de exemplo, JSONs de teste, documentação auxiliar) serão também anexados em formato .zip juntamente à submissão deste relatório