Técnicas Metaheurísticas para o Problema de Alocação de Salas de Aula

Maria Eduarda Bessa Teixeira[1], **João Victor Ramalho de Sousa Pereira Jardim**[1]

[1] Universidade Federal de Ouro Preto (UFOP)
Instituto de Ciências Exatas e Biológicas (ICEB)
Campus Morro do Cruzeiro, Ouro Preto – MG, Brasil

maria.ebt@aluno.ufop.edu.br, joao.jardim@aluno.ufop.edu.br

Abstract. *This paper presents a constructive heuristic approach for solving the Classroom Assignment Problem (CAP), a known NP-hard combinatorial optimization problem in the class of timetabling and resource allocation problems. The work focuses on generating high-quality initial solutions using two complementary greedy strategies: a fully greedy Largest-First Best-Fit heuristic and a semi-greedy variant with controlled randomness based on Restricted Candidate Lists (RCL). A formal mathematical model is presented, including set definitions, parameters, decision variables, hard and soft constraints, and the objective function. Both constructive heuristics are detailed with pseudocode, computational complexity analysis, and practical examples. The semi-greedy approach introduces a parameter* $\alpha \in [0, 1]$ *to control the balance between solution quality and diversity for subsequent metaheuristic refinement.*

Resumo. *Este artigo apresenta uma abordagem heurística construtiva para o Problema de Alocação de Salas (PAS), um problema de otimização combinatória classificado como NP-difícil na classe de problemas de timetabling e alocação de recursos. O trabalho concentra-se na geração de soluções iniciais de alta qualidade utilizando duas estratégias gulosas complementares: uma heurística totalmente gulosa Largest-First Best-Fit e uma variação semi-gulosa com aleatoriedade controlada baseada em Listas Restritas de Candidatos (RCL). Um modelo matemático formal é apresentado, incluindo definições de conjuntos, parâmetros, variáveis de decisão, restrições obrigatórias e suaves, e a função objetivo. Ambas as heurísticas construtivas são detalhadas com pseudocódigo, análise de complexidade computacional e exemplos práticos. A abordagem semi-gulosa introduz um parâmetro* $\alpha \in [0, 1]$ *para controlar o equilíbrio entre qualidade de solução e diversidade para refinamento subsequente por metaheurísticas.*

1. Introdução

A alocação de turmas em salas de aula é uma tarefa recorrente e complexa enfrentada por instituições de ensino superior. A cada semestre, é necessário distribuir centenas de encontros (aulas) em um conjunto limitado de salas, respeitando capacidades, horários, características físicas, disponibilidade e preferências de professores e cursos. Esse processo é frequentemente realizado manualmente, sendo demorado, propenso a erros e, muitas vezes, incapaz de gerar alocações eficientes.

Do ponto de vista teórico, esse problema é conhecido como Problema de Alocação de Salas (PAS), e está relacionado a problemas de *timetabling* e alocação de recursos, ambos classificados como NP-difíceis [1]. A dificuldade aumenta conforme a quantidade de encontros cresce, tornando técnicas exatas inviáveis para instâncias reais de médio e grande porte.

1.1. Breve introdução do problema e revisões bibliográficas

Diversos autores estudaram variações do PAS. Carter e Tovey [2] analisaram a complexidade do problema e destacaram que, mesmo em versões simplificadas, a dificuldade permanece elevada. Schaerf [3] apresenta um panorama sobre técnicas de *timetabling* aplicadas a instituições de ensino, incluindo heurísticas e metaheurísticas.

Em contextos brasileiros, trabalhos como os de Nascimento et al. [4] e Souza et al. [5] aplicaram, respectivamente, *Simulated Annealing* e *Variable Neighborhood Search* para melhorar soluções para instâncias acadêmicas reais.

O artigo-base selecionado [6] modela o PAS como um problema multiobjetivo, considerando desperdício, alocação excedente e taxa de não alocação. Neste trabalho, utilizamos essa formulação como inspiração, adotando um enfoque mono-objetivo com componentes de preferências para orientar as heurísticas construtivas desenvolvidas.

2. Modelo Matemático Formal

2.1. Caracterização do Problema

O Problema de Alocação de Salas (PAS) é um problema de otimização combinatória pertencente à classe de *Timetabling Problems* e *Resource Allocation Problems*. Especificamente, é um *Class Timetabling Problem* com restrições de capacidade, compatibilidade de tipo de sala e preferências de localização e equipamentos.

- **Classe**: Problema de Otimização Combinatória NP-Difícil;
- **Objetivo**: Maximizar alocação de encontros minimizando desperdício de capacidade e violações de preferências;
- **Tipo**: Problema de Satisfação com Preferências (*Constraint Satisfaction with Preferences*).

2.2. Conjuntos e Índices

Definem-se os seguintes conjuntos:

- M: conjunto de encontros (meetings), com $|M| = m$;
- C: conjunto de salas (classrooms), com $|C| = c$;
- H: conjunto de horários (schedules), com $|H| = h$;
- D: conjunto de dias da semana, com $|D| = 7$;
- P: conjunto de professores;
- S: conjunto de disciplinas (subjects);
- B: conjunto de prédios (buildings);
- Pref: conjunto de preferências.

Os índices utilizados são: $i \in M$ (encontro), $j \in C$ (sala), $k \in H$ (horário), $d \in D$ (dia).

2.3. Parâmetros

2.3.1. Parâmetros de Encontros

Para cada encontro $i \in M$, definem-se:

- demand_i: número de alunos requeridos (demanda do encontro);
- dayOfWeek_i: dia da semana em que o encontro ocorre (fixo, $0 \leq \text{dayOfWeek}_i < 7$);
- $S_i \subseteq H$: conjunto de horários disponíveis para o encontro i;
- $\text{isPractical}_i \in \{0, 1\}$: indicador se o encontro é prático (requer laboratório);
- $\text{prof}_i \subseteq P$: conjunto de professores responsáveis pelo encontro;
- $\text{subj}_i \in S$: disciplina associada ao encontro.

2.3.2. Parâmetros de Salas

Para cada sala $j \in C$, definem-se:

- capacity_j: capacidade (número máximo de alunos) da sala;
- $\text{isLab}_j \in \{0, 1\}$: indicador se a sala é laboratório;
- $\text{building}_j \in B$: prédio onde a sala está localizada;
- floor_j: andar da sala;
- board_j: tipo de quadro disponível (lousa, whiteboard, etc.);
- $\text{projector}_j \in \{0, 1\}$: indicador se a sala possui projetor.

2.3.3. Parâmetros de Preferências

Para cada preferência $p \in \text{Pref}$, definem-se:

- category_p: categoria da preferência (professor, disciplina ou turma);
- categoryCode_p: código identificador da categoria;
- building_p: prédio preferido (ou vazio se não especificado);
- floor_p: andar preferido (ou -1 se não especificado);
- board_p: tipo de quadro preferido (ou vazio);
- $\text{projector}_p \in \{0, 1\}$: preferência por projetor.

2.3.4. Parâmetros de Penalidades

- W_{pref}: penalidade por preferência violada. Na heurística gulosa, $W_{\text{pref}} = 10\,000$; na heurística semi-gulosa, $W_{\text{pref}} = 1\,000$;
- $\alpha \in [0, 1]$: parâmetro de aleatoriedade para a heurística parcialmente gulosa, controlando a largura da Lista Restrita de Candidatos.

2.4. Variáveis de Decisão

A variável de decisão binária é definida como:

$$x_{i,j,k,d} \in \{0, 1\}$$

Onde $x_{i,j,k,d} = 1$ se o encontro i é alocado à sala j, no horário k, no dia d; e $x_{i,j,k,d} = 0$ caso contrário.

Restrição implícita: $d = \text{dayOfWeek}_i$ (o dia é fixo por encontro).

Definem-se também variáveis auxiliares para cálculo de métricas:

- $\text{waste}_{i,j} = \text{capacity}_j - \text{demand}_i$: desperdício de capacidade quando encontro i é alocado à sala j;
- $\text{penalty}_{i,j}$: penalidade total por preferências violadas na alocação de i a j.

2.5. Restrições (Hard Constraints)

2.5.1. Alocação Única

Cada encontro é alocado no máximo uma vez:

$$\sum_{j \in C} \sum_{k \in S_i} x_{i,j,k,d_i} \leq 1 \quad \forall i \in M$$

2.5.2. Compatibilidade de Horário

Um encontro só pode ser alocado em horários admissíveis:

$$x_{i,j,k,d} = 0 \quad \text{se} \quad k \notin S_i$$

2.5.3. Compatibilidade de Tipo de Sala

Encontros práticos só podem ser alocados em laboratórios:

$$x_{i,j,k,d} = 0 \quad \text{se} \quad \text{isPractical}_i = 1 \text{ e } \text{isLab}_j = 0$$

2.5.4. Restrição de Capacidade

A sala deve ter capacidade suficiente para o encontro:

$$x_{i,j,k,d} = 0 \quad \text{se} \quad \text{demand}_i > \text{capacity}_j$$

2.5.5. Sem Conflito de Sala-Horário-Dia

Cada sala em cada horário em cada dia pode ter no máximo um encontro:

$$\sum_{i \in M} x_{i,j,k,d} \leq 1 \quad \forall j \in C, k \in H, d \in D$$

2.6. Restrições (Soft Constraints / Preferências)

Para cada encontro i e sala j, calcula-se a penalidade de preferências violadas:

$$\text{penalty}_{i,j} = \sum_{p \in \text{Pref}} \delta_{i,j,p} \cdot W_{\text{pref}}$$

Onde $\delta_{i,j,p} = 1$ se a preferência p aplicável ao encontro i é violada pela sala j. Exemplos de violações:

- Prédio preferido $\neq$ prédio da sala;
- Andar preferido $\neq$ andar da sala;
- Tipo de quadro preferido $\neq$ quadro da sala;
- Projetor preferido mas sala não possui.

2.7. Função Objetivo

2.7.1. Objetivo Primário

Maximizar o número de encontros alocados:

$$\text{maximize} \quad \sum_{i \in M} \sum_{j \in C} \sum_{k \in S_i} x_{i,j,k,d_i}$$

2.7.2. Objetivo Secundário

Para cada alocação válida, minimizar a função de custo (score):

$$\text{score}_{i,j} = \text{waste}_{i,j} + \text{penalty}_{i,j}$$

Onde:

- $\text{waste}_{i,j} = \text{capacity}_j - \text{demand}_i$: desperdício de capacidade;
- $\text{penalty}_{i,j}$: penalidade por violações de preferências.

A heurística gulosa escolhe a sala com menor $\text{score}_{i,j}$, enquanto a heurística semi-gulosa constrói uma lista restrita de candidatos com scores próximos ao mínimo e seleciona aleatoriamente.

3. Metodologia

3.1. Representação Computacional da Solução

3.1.1. Estrutura de Dados da Solução

Uma solução é formalmente representada como um conjunto de tuplas (i, j, k, d) indicando que o encontro $i \in M$ foi alocado à sala $j \in C$, no horário $k \in S_i$, no dia $d = \text{dayOfWeek}_i$.

Computacionalmente, a solução é mantida por meio de duas estruturas principais, garantindo a rápida verificação das restrições obrigatórias:

1. **Alocações por Encontro (Allocations)**: Um mapeamento (ou vetor) que associa cada encontro i à tupla de alocação (j, k). Isso facilita a verificação da restrição de Alocação Única ($\sum_{j,k} x_{i,j,k,d_i} \leq 1$).
2. **Reservas de Salas (Reservations)**: Uma estrutura de dados (e.g., um dicionário aninhado ou matriz) que registra o encontro alocado para cada tripla única (sala j, horário k, dia d).

O exemplo de métricas coletadas em um cenário simplificado ilustra essa representação de forma agregada. Por exemplo, a estrutura classroom_occupancy armazena o *estado* de uma sala após a alocação, indicando que a **sala de id 11** está com 2 encontros (meetings), uma demanda total de 36 alunos, capacidade de 40 e uma utilização de 90%.

3.1.2. Análise da Representação Computacional

O **estado final** da solução (as tuplas (i, j, k, d)) é processado para gerar métricas de avaliação. O exemplo de saída JSON abaixo ilustra o resultado da avaliação de uma solução em uma micro-instância de teste:

Exemplo de Saída de Avaliação (JSON)


```
{
  "metrics": [
    {"metric": "Encontros Alocados", "value": 2.0},
    {"metric": "Taxa Alocacao (%)", "value": 100.0},
    {"metric": "Demanda Alocada", "value": 36.0},
    {"metric": "Desperdicio Medio", "value": 22.0}
  ],
  "classroom_occupancy": [
    {
      "classroom_id": 11, "meetings": 2,
      "demand": 36, "capacity": 40, "util_pct": 90.0
    }
  ],
  "waste_distribution": [22, 22]
}
```

3.1.3. Visualização da Solução e Definição

A qualidade da solução é definida pela sua capacidade de satisfazer o Objetivo Primário (maximizar a taxa de alocação) enquanto minimiza o **score** do Objetivo Secundário (minimizar desperdício e penalidades por preferências).

Solução Válida (Viável): Uma solução é considerada válida se satisfizer **todas as Restrições Obrigatórias (Hard Constraints) definidas na Seção 2.5, independentemente do score de custo.

Solução Ideal (Ótima): Uma solução ideal é uma solução válida que maximiza o número de encontros alocados e, entre as soluções com o máximo de encontros alocados, minimiza o custo total (soma dos $\text{score}_{i,j}$ para todos os encontros alocados).

A imagem acima representa o conceito de uma solução viável: um **Timetable** onde cada bloco de aula é um encontro (i), alocado a uma sala (j), em um tempo específico (k, d). A ausência de sobreposição de encontros em qualquer célula (sala, horário, dia) garante a validade da alocação.

3.2. Heurística 1: Largest-First Best-Fit (Totalmente Gulosa)

A primeira heurística construtiva é totalmente gulosa, seguindo o paradigma *Largest-First Best-Fit*. O procedimento é resumido nas seguintes etapas:

1. **Ordenação dos encontros**: Cria-se um vetor de índices dos encontros e ordena-se em ordem decrescente de demanda. Encontros com maior número de alunos são alocados primeiro, pois são geralmente mais difíceis de acomodar.
2. **Varredura de horários permitidos**: Para cada encontro, percorre-se sua lista de horários admissíveis S_i, respeitando o dia da semana dayOfWeek_i.
3. **Busca pela melhor sala**: Para cada par (encontro, horário), todas as salas físicas são avaliadas. Descartam-se salas que:
    - Já estão ocupadas naquele dia e horário;
    - Têm capacidade $\text{capacity}_j < \text{demand}_i$;
    - Não atendem requisitos obrigatórios (e.g., laboratório para aulas práticas).
4. **Cálculo do custo local**: Para cada sala viável, calcula-se:

$$\text{score}_{i,j} = (\text{capacity}_j - \text{demand}_i) + \text{penalty}_{i,j}$$

5. **Escolha best-fit**: Seleciona-se a sala com menor score. Se encontrada, adiciona-se a alocação a reservations e passa-se ao próximo encontro.

Tratamento de preferências. Antes de avaliar as salas, constrói-se para cada encontro i o conjunto de preferências aplicáveis. Esse conjunto contém apenas as preferências cuja categoria (professor, disciplina ou turma) combina com os códigos presentes no encontro. Na avaliação de uma sala j, essas preferências são comparadas com os atributos da sala, e cada violação contribui com $W_{\text{pref}} = 10\,000$ à penalidade.

Pseudocódigo.

Algoritmo 1: Heurística Gulosa Largest-First Best-Fit

ENTRADA: M, C, H, D, Pref
SAÍDA: solution (alocações)

$indices \leftarrow$ CRIAR_VETOR$(0, 1, \ldots, |M| - 1)$
ORDENAR $indices$ POR demand[indices$[i]$] DECRESCENTE

PARA CADA $idx \in indices$ FAÇA
  $i \leftarrow idx$
  prefs $\leftarrow$ PREFERÊNCIAS_APLICÁVEIS(i)
  melhor_score $\leftarrow \infty$
  melhor_sala $\leftarrow$ NULL

  PARA CADA $k \in S_i$ FAÇA
    PARA CADA $j \in C$ FAÇA
      SE SALA_VIÁVEL(i, j, k) ENTÃO
        score $\leftarrow$ (capacity$_j$ − demand$_i$) + CALCULAR_PENALTY$(i, j,$ prefs$)$
        SE score $<$ melhor_score ENTÃO
          melhor_score $\leftarrow$ score
          melhor_sala $\leftarrow (j, k)$
        FIM SE
      FIM SE
    FIM PARA
    SE melhor_sala $\neq$ NULL ENTÃO QUEBRAR
  FIM PARA

  SE melhor_sala $\neq$ NULL ENTÃO
    ADICIONAR $(i,$ melhor_sala$)$ A $solution$
  FIM SE
FIM PARA

RETORNAR $solution$

(1)

3.3. Heurística 2: Parcialmente Gulosa (Semi-Greedy com RCL)

A segunda heurística introduz aleatoriedade controlada através de uma *Lista Restrita de Candidatos* (RCL). Mantém a estrutura base da Heurística 1, mas modifica a etapa de seleção de sala.

Construção da RCL. Após calcular o score de todas as salas viáveis, calcula-se um threshold (limiar) de qualidade:

$$\text{threshold} = \text{minScore} + \alpha \cdot (\text{maxScore} - \text{minScore})$$

A RCL é então construída como:

$$\text{RCL} = \{j \in C : \text{score}_{i,j} \leq \text{threshold}\}$$

Se a RCL estiver vazia, adiciona-se a sala com menor score para garantir que uma alocação seja sempre feita.

Seleção aleatória. Uma vez construída a RCL, seleciona-se uniformemente ao acaso uma sala:

$$j^* \leftarrow \text{ESCOLHER\_ALEATORIAMENTE}(\text{RCL})$$

Parâmetro α. O parâmetro α controla o comportamento da heurística:

- $\alpha = 0$: comportamento totalmente guloso (RCL = {melhor sala});
- $\alpha = 1$: máxima aleatoriedade (RCL = todas as salas viáveis);
- $0 < \alpha < 1$: aleatoriedade intermediária.

Penalidade reduzida. Na Heurística 2, $W_{\text{pref}} = 1\,000$ (vs. $10\,000$ na Heurística 1), permitindo maior flexibilidade na exploração.

Pseudocódigo.

Algoritmo 2: Heurística Parcialmente Gulosa (Diferenças)

[Mantém etapas 1-3 da Heurística 1]

[Após calcular scores de todas as salas viáveis:]

minScore $\leftarrow \min(\{\text{score}_{i,j} : j \in \text{candidatas}\})$
maxScore $\leftarrow \max(\{\text{score}_{i,j} : j \in \text{candidatas}\})$

threshold $\leftarrow$ minScore $+ \alpha \cdot$ (maxScore $-$ minScore)

RCL $\leftarrow \emptyset$
PARA CADA $j \in$ candidatas FAÇA
  SE $\text{score}_{i,j} \leq$ threshold ENTÃO
    RCL $\leftarrow$ RCL $\cup \{j\}$
  FIM SE
FIM PARA

SE RCL $= \emptyset$ ENTÃO
  RCL $\leftarrow$ {sala com menor score}
FIM SE

$j^* \leftarrow$ ESCOLHER_ALEATORIAMENTE(RCL)

[Continua com adição à solução como na Heurística 1]

(2)

3.4. Análise de Complexidade

3.4.1. Complexidade Temporal

Para a Heurística 1 (Gulosa):

- Ordenação de encontros: $O(m \log m)$;
- Loops aninhados: para cada encontro (m), cada horário ($\leq h$), cada sala (c), calcula-se o score em $O(p)$ (número de preferências aplicáveis);
- Complexidade total: $O(m \log m + m \cdot h \cdot c \cdot p) = O(m \cdot h \cdot c \cdot p)$.

Para a Heurística 2 (Semi-Gulosa):

- Mesma complexidade da Heurística 1, pois a construção da RCL é linear em relação ao número de candidatos.

3.4.2. Complexidade de Espaço

- Armazenar alocações: $O(m)$;
- Dados de entrada: $O(m + c + h + p)$;
- Complexidade total: $O(m + c + h + p)$.

3.5. Métricas Coletadas

Durante a execução das heurísticas, coletam-se as seguintes métricas:

Métrica	**Descrição**
Encontros Alocados	Número total de encontros com alocação bem-sucedida
Taxa de Alocação (%)	$\frac{\text{Encontros Alocados}}{\lvert M \rvert} \times 100$
Demanda Alocada	Soma das demandas dos encontros alocados
Taxa de Demanda (%)	$\frac{\text{Demanda Alocada}}{\sum_i \text{demand}_i} \times 100$
Desperdício Médio	Média de lugares vazios por sala ocupada
Alunos Desalocados	Demanda total menos demanda alocada
Vagas Ociosas (¡50%)	Vagas em salas com ocupação ¡ 50% da capacidade
Alunos em Pé	Alunos em salas com demanda ¿ capacidade
Taxa de Preferências Atendidas	Percentual de preferências satisfeitas por categoria

Table 1. Métricas de avaliação das heurísticas construtivas.

3.6. Exemplos de Aplicação

Considere um cenário simplificado com 3 encontros e 3 salas:

- **Encontros**: E1 (80 alunos, prefere prédio A), E2 (50 alunos, prefere laboratório), E3 (30 alunos, sem preferências);
- **Salas**: S1 (cap. 90, prédio A, sem lab), S2 (cap. 60, prédio B, com lab), S3 (cap. 40, prédio A, sem lab).

Execução da Heurística 1 (Gulosa):

1. **E1 (80 alunos)**: S1 tem score $(90 - 80) + 0 = 10$ (preferência atendida). Escolhe S1.
2. **E2 (50 alunos)**: S2 tem score $(60 - 50) + 0 = 10$ (laboratório atendido). Escolhe S2.
3. **E3 (30 alunos)**: S3 tem score $(40 - 30) + 0 = 10$. Escolhe S3.

   Resultado: todos os encontros alocados com desperdício mínimo.

Execução da Heurística 2 (Semi-Gulosa) com $\alpha = 0.5$: Para o mesmo cenário, com penalidade reduzida de $1\,000$:

1. **E1 (80 alunos)**: minScore = 10 (S1), maxScore = $1\,020$ (S2 com preferência violada). threshold = $10 + 0.5 \cdot 1\,010 = 515$. RCL = $\{\text{S1}\}$. Escolhe S1.
2. **E2 (50 alunos)**: minScore = 10 (S2), maxScore = $1\,010$ (S3). threshold = 505. RCL = $\{\text{S2}\}$. Escolhe S2.
3. **E3 (30 alunos)**: minScore = 10 (S3), maxScore = 10. threshold = 10. RCL = $\{\text{S3}\}$. Escolhe S3.

Resultado: idêntico ao da Heurística 1 neste caso, mas em cenários mais complexos a Heurística 2 pode explorar alternativas diferentes.

3.7. Plano de Testes

O plano de avaliação das heurísticas construtivas envolve:

1. **Instâncias reais**: dados do ICEB-UFOP de semestres anteriores, contendo aproximadamente 500-600 encontros e 36 salas;
2. **Instâncias sintéticas**: geradas aleatoriamente com diferentes tamanhos (pequenas: 100-200 encontros; médias: 300-500; grandes: 600-1000);
3. **Métricas de avaliação**: conforme Tabela 1;
4. **Comparação**: análise comparativa entre Heurística 1 e Heurística 2 com diferentes valores de α;
5. **Análise de sensibilidade**: impacto de variações nas penalidades de preferência e no parâmetro α.

3.8. Resultados

Para a avaliação de desempenho das heurísticas construtivas, foram feitas análises dos dois algoritmos principais: a heurística gulosa determinística, e a heurística parcialmente gulosa.

Os algoritmos foram comparados em instâncias de teste com base em quatro métricas principais: taxa de alocação de encontros (%), taxa de alocação de demanda (alunos), desperdício médio de vagas e tempo de execução (s).

3.8.1. Análise da Heurística Gulosa

A heurística gulosa, por ser determinística, serve como nossa baseline. Conforme ilustrado na Figura 1, a análise de uma instância representativa demonstra a alta eficácia desta abordagem, alcançando uma taxa de alocação de encontros de 97,4% e acomodando 96,1% da demanda total de alunos. O desperdício médio de vagas por encontro alocado foi de 11,69.

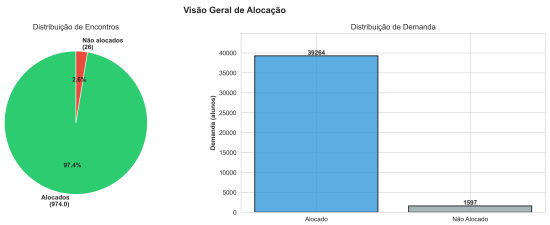

Figure 1. Visão Geral de Alocação para a Heurística Gulosa.

3.9. Análise Comparativa: Gulosa vs. Parcialmente Gulosa

A segunda heurística implementada introduz aleatoriedade controlada pelo parâmetro alpha.

A Figura 2 compara a taxa de alocação entre a heurística gulosa determinística e as variante parcialmente gulosa (denotadas como p-aX-sY, onde a é o alpha e s a semente de aleatoriedade).

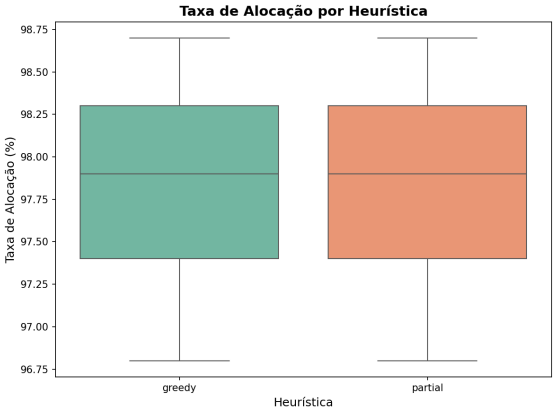

Figure 2. Taxa de Alocação por Heurística.

Observa-se que ambas as heurísticas apresentam a mesma taxa de alocação.

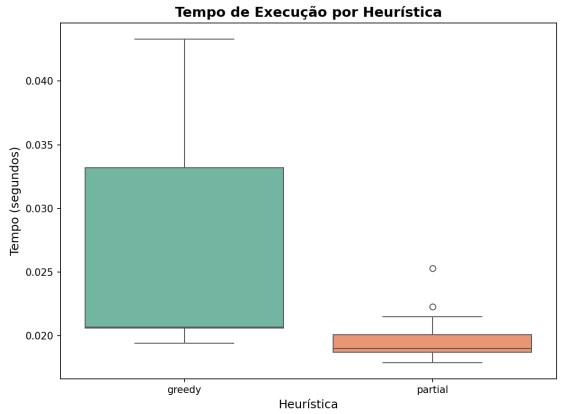

Figure 3. Tempo de Execução por Heurística.

Quanto a eficiência computacional, temos através da Figura 3 que a heurística parcialmente gulosa se mostrou mais rápida. O boxplot evidencia que o Greedy apresenta maior variação e maior tempo médio, enquanto o método parcialmente guloso concentra tempos menores e mais estáveis.

3.9.1. 4.3 Análise do Parâmetro alpha e o Trade-off de Exploração

A análise mais crítica para a heurística parcialmente gulosa é o impacto do parâmetro alpha.

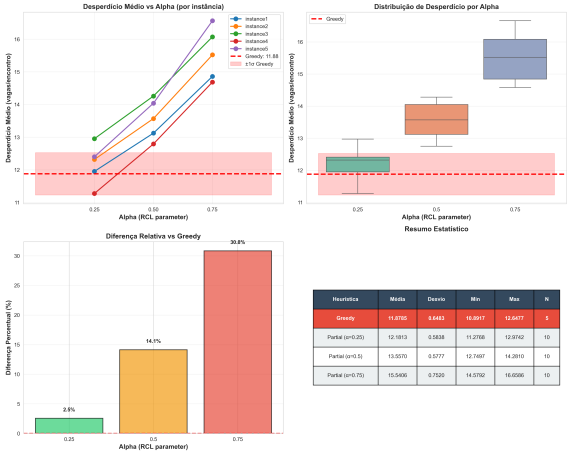

Figure 4. Análise do comportamento da heurística parcilamente gulosa.

A partir da [Figura 4], podemos concluir que: com Alpha baixo (ex: 0.25), o algoritmo se comporta de forma muito similar à heurística gulosa pura, gerando soluções com baixo desperdício médio, mas também com baixa variabilidade. Na medida que alpha cresce, o algoritmo se torna mais exploratório. Ao permitir que salas com maior desperdício entrem na RCL, o desperdício médio da solução final aumenta, mesmo que não seja o suficiente para alterar os resultados de taxa de alocação.

3.10. Extensões e Trabalhos Futuros

Após validação das heurísticas construtivas, pretende-se:

1. Implementar estruturas de busca local para refinamento de soluções (realocação simples, troca, realocação com cascata);
2. Aplicar metaheurísticas como *Variable Neighborhood Search*, *Simulated Annealing* e *Algoritmos Genéticos*;
3. Estender para formulação multiobjetivo explícita, considerando simultaneamente múltiplos objetivos;
4. Investigar técnicas de aprendizado de máquina para ajuste automático de parâmetros.

4. Conclusões

Apresentou-se uma abordagem dual de heurísticas construtivas eficientes para o Problema de Alocação de Salas: uma heurística totalmente gulosa (Largest-First Best-Fit) e uma variação semi-gulosa com aleatoriedade controlada baseada em Listas Restritas de Candidatos. Ambas as heurísticas são capazes de gerar rapidamente soluções iniciais de boa qualidade, considerando simultaneamente desperdício de capacidade e atendimento de preferências, além de serem compatíveis com dados reais de instituições de ensino.

O modelo matemático formal apresentado fornece uma base sólida para compreensão e análise do problema, enquanto as heurísticas propostas oferecem uma abordagem prática e eficiente para sua resolução. A Heurística 1 oferece soluções determinísticas de alta qualidade, enquanto a Heurística 2 fornece diversidade de soluções, permitindo exploração mais ampla do espaço de busca.

As próximas etapas do trabalho envolvem validação experimental em instâncias reais e sintéticas, desenvolvimento de técnicas de refinamento, e aplicação de metaheurísticas para melhoria adicional das soluções.

References

[1] Even, S.; Itai, A.; Shamir, A. On the Complexity of Time Table and Multi-Commodity Flow Problems. In: *Proceedings of the 16th Annual Symposium on Foundations of Computer Science (SFCS)*. IEEE, 1975, p. 184–193.

[2] Carter, M. W.; Tovey, C. A. When Is the Classroom Assignment Problem Hard? *Operations Research*, v. 40, n. 1, p. S28–S39, 1992.

[3] Schaerf, A. A Survey of Automated Timetabling. *Artificial Intelligence Review*, v. 13, p. 87–127, 1999.

[4] Nascimento, A. S.; Sampaio, R. M.; Alvarenga, G. B. Uma aplicação de *simulated annealing* para o problema de alocação de salas. *INFOCOMP Journal of Computer Science*, v. 4, n. 3, p. 59–66, 2005.

[5] Souza, M. J. F.; Martins, A. X.; Araújo, C.; Costa, F. W. A. Métodos de pesquisa em vizinhança variável aplicados ao problema de alocação de salas. In: *Anais do XXII Encontro Nacional de Engenharia de Produção (ENEGEP)*. Fortaleza, Brasil, 2002.

[6] Souza, C. E.; Milagres, B.; Silva, P. Uma abordagem multi-objetivo do problema de alocação de salas de aula. In: *Anais do Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 2024.

[7] Feo, T. A.; Resende, M. G. C. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, v. 6, n. 2, p. 109–133, 1995.

[8] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002.