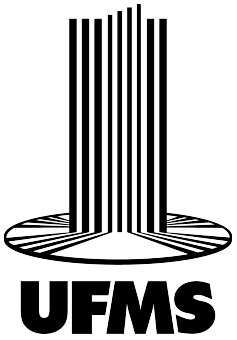
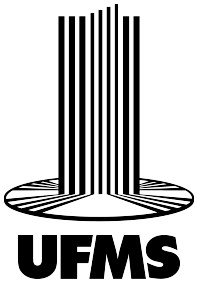


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Uma Abordagem Matheurística baseada em Large Neighborhood Search para o Problema da Distribuição de Disciplinas

Deivid Reinke Schutz

Campo Grande - MS
23 de Janeiro de 2026



UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Uma Abordagem Matheurística baseada em Large Neighborhood Search para o Problema da Distribuição de Disciplinas

Deivid Reinke Schutz

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharelado em Ciência da Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Orientador: Profa. Dr. Edna Ayako Hoshino

Campo Grande - MS
23 de Janeiro de 2026

Uma Abordagem Matheurística baseada em Large Neighborhood Search para o Problema da Distribuição de Disciplinas

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharelado em Ciência da Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Banca Examinadora:

Profa. Dr. Edna Ayako Hoshino

Prof. Dr. Membro da banca 1

Prof. Dr. Membro da banca 2

Campo Grande - MS

23 de Janeiro de 2026

Agradecimentos

Resumo

Este trabalho aborda o Problema da Distribuição de Disciplinas (DPD), que consiste em atribuir disciplinas a professores respeitando restrições de alocação e limites de carga horária, visando maximizar a qualidade da atribuição. O problema é modelado como um programa linear inteiro e resolvido no SCIP via Branch-and-Bound.

Como contribuição, propõe-se e avalia-se uma matheurística baseada em Large Neighborhood Search (LNS) no esquema fix-and-relax, que explora uma solução incumbente e define uma vizinhança ao fixar a maior parte das variáveis binárias $x_{p,c}$, liberando uma fração controlada para reotimização. A taxa de destruição (`lns_perc`), a ordenação de candidatos (`avgPreferenceWeight`) e o tempo máximo do sub-MIP são analisados quanto ao impacto na qualidade da solução e nas métricas de desempenho observadas: tempo, gap e nós remanescentes (nodes-left).

Palavras-chave: Distribuição de Disciplinas, Programação Linear Inteira, Branch-and-Bound, Matheurística, Large Neighborhood Search, Fix-and-Relax, SCIP.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contextualização

O planejamento acadêmico semestral envolve diversas decisões interdependentes, entre as quais se destaca a atribuição de disciplinas a professores. Em cursos de graduação, essa distribuição precisa respeitar restrições institucionais (por exemplo, alocação única de cada disciplina, limites de carga horária por período e por ano e regras internas de oferta), ao mesmo tempo em que busca atender critérios de qualidade associados à afinidade do docente com o conteúdo, histórico de atuação e preferências.

Neste trabalho considera-se o Problema da Distribuição de Disciplinas (DPD), no qual o objetivo é construir uma atribuição disciplina–professor que maximize a qualidade da alocação sob restrições operacionais. O DPD pode ser modelado como um problema de otimização combinatória e, em particular, como um programa linear inteiro (PLI) com variáveis binárias indicando se um professor p é designado para uma disciplina c. Modelos desse tipo podem ser resolvidos por métodos exatos, como Branch-and-Bound, disponíveis em solvers modernos como o SCIP.

Apesar de a modelagem via PLI permitir incorporar restrições e critérios de forma sistemática, instâncias de maior porte e/ou com restrições mais detalhadas podem tornar a obtenção de boas soluções em tempo limitado um desafio prático. Isso motiva o uso de estratégias que combinem a força dos métodos exatos com mecanismos heurísticos de intensificação.

1.2 Motivação e Justificativa

Na prática, a distribuição de disciplinas é um processo recorrente e sensível: pequenas mudanças em regras, em ofertas ou no quadro docente podem alterar significativamente o espaço de soluções. Além disso, a necessidade de respostas em prazos curtos torna desejável obter soluções viáveis de boa qualidade rapidamente, mesmo quando a prova de otimalidade não é alcançada.

Uma linha de pesquisa que atende a essa necessidade é o uso de matheurísticas, isto é, heurísticas baseadas em programação matemática, que exploram subproblemas de PLI (ou relaxações) como operadores dentro de um procedimento de busca. Em particular, a Large Neighborhood Search (LNS) é uma matheurística de destruição e reparação: parte-se de uma solução incumbente, “destrói-se” uma parte controlada das decisões e, em seguida, reotimiza-se o subproblema resultante para “reparar” a solução. Esse paradigma

tem sido utilizado com sucesso em diferentes problemas combinatórios e também em contextos que integram técnicas exatas e heurísticas (**??**).

No contexto do DPD, a LNS é especialmente atrativa por permitir explorar vizinhanças grandes preservando a maior parte da estrutura da incumbente. Nesta monografia, implementa-se uma LNS no estilo fix-and-relax no SCIP: fixa-se a maior parte das variáveis binárias conforme a incumbente e libera-se uma fração das atribuições para reotimização, gerando um subproblema inteiro (sub-MIP) resolvido com limite de tempo. A estratégia é integrada ao processo de Branch-and-Bound, buscando intensificação por meio de melhorias sucessivas da incumbente.

Além da análise interna da LNS, este trabalho também considera uma comparação com uma heurística GRASP desenvolvida especificamente para o DPD, bem como uma estratégia híbrida na qual a LNS é aplicada como etapa de melhoria a partir de soluções construídas pelo GRASP (GRASP|LNS). Essa comparação visa evidenciar, nas métricas de tempo, gap e nós remanescentes, em quais cenários a intensificação via LNS agrega valor em relação a uma abordagem construtiva/melhorativa dedicada (JHONATAN, 2025).

1.3 Objetivos

1.3.1 Objetivo Geral

Propor, implementar e avaliar uma matheurística baseada em LNS, integrada ao solver SCIP, para obter soluções de boa qualidade para o DPD em tempo limitado.

1.3.2 Objetivos Específicos

- modelar o DPD como um PLI e integrá-lo ao framework de solução do SCIP, utilizando Branch-and-Bound como abordagem exata de referência;
- implementar uma matheurística LNS do tipo fix-and-relax, baseada em destruição parcial da incumbente e reparação via resolução de um sub-MIP em uma instância secundária (sub-SCIP);
- analisar o impacto de parâmetros internos da LNS, incluindo a taxa de destruição, o sentido de ordenação dos candidatos e o tempo máximo do subproblema;
- avaliar o desempenho por meio das métricas observadas nos experimentos: tempo, gap e nós remanescentes (nodes-left);
- comparar a estratégia LNS com abordagens alternativas consideradas no trabalho (relaxação e a heurística GRASP), incluindo a variação híbrida GRASP|LNS, discutindo cenários em que a LNS é mais efetiva.

1.4 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 é apresentada a fundamentação teórica, cobrindo conceitos de otimização inteira e heurísticas/matheurísticas relevantes ao estudo. O Capítulo 3 define o DPD e descreve o modelo matemático exato adotado. O Capítulo 4 apresenta os resultados experimentais e a análise computacional, incluindo a avaliação das configurações internas da LNS e comparações de desempenho. Por fim, o Capítulo 5 conclui o trabalho e discute limitações e possibilidades de trabalhos futuros.

2 Fundamentação Teórica

2.1 Otimização Combinatória e Programação Linear Inteira (PLI)

Problemas de alocação em ambientes acadêmicos podem ser formulados como problemas de otimização combinatória, nos quais se busca selecionar uma atribuição que maximize uma medida de qualidade (por exemplo, preferências) e, ao mesmo tempo, respeite restrições operacionais (carga horária, conflitos e requisitos mínimos). Na literatura, problemas próximos incluem o *class–faculty assignment* e variantes de alocação docente com preferências (AL-YAKOOB; SHERALI, 2006; AL-YAKOOB; SHERALI, 2013; DOMENECH; LUSA, 2016), bem como estudos em *course timetabling* (CARTER; LAPORTE, 1998; AVELLA; VASILYEV, 2005).

Uma forma padrão de modelar tais problemas é por Programação Linear Inteira (PLI), em que variáveis de decisão (frequentemente binárias) representam escolhas discretas, e a função objetivo quantifica a qualidade da solução. De modo geral, um modelo pode ser escrito como

$$\max \sum_{p \in P} \sum_{c \in C} w_{p,c}\, x_{p,c} \quad \text{sujeito a restrições lineares e } x_{p,c} \in \{0,1\},$$

onde $x_{p,c} = 1$ indica que o professor p é designado à disciplina c, e $w_{p,c}$ representa o peso/-preferência. A PLI permite representar restrições de compatibilidade e limites de capacidade de forma explícita, mantendo uma interpretação direta do modelo (VANDERBEI, 2014; WOLSEY, 2020).

2.2 O Método Branch-and-Bound e Relaxação Linear

Modelos de PLI são, em geral, NP-difíceis, de modo que solucionadores utilizam métodos exatos baseados em limites e enumeração inteligente. Um componente central é a relaxação linear, obtida ao substituir a integralidade por $0 \leq x \leq 1$, produzindo um problema de Programação Linear (PL) que fornece um limite para a solução inteira ótima (VANDERBEI, 2014; WOLSEY, 2020).

O Branch-and-Bound (B&B) explora uma árvore de subproblemas: a cada nó, resolve-se uma relaxação para obter um limite e, quando necessário, ramifica-se em subproblemas impondo decisões de integralidade (por exemplo, $x_{p,c} = 0$ ou $x_{p,c} = 1$). Nós cujos limites não superam a melhor solução inteira conhecida podem ser podados, reduzindo a busca (WOLSEY, 2020). Na prática, solucionadores modernos integram B&B com

técnicas adicionais; neste trabalho, a resolução exata é realizada com o SCIP (GAMRATH et al., 2020).

2.3 Heurísticas e Metaheurísticas

Embora métodos exatos forneçam garantias de otimalidade, instâncias reais podem apresentar grande dimensão e múltiplas restrições, tornando necessário recorrer a estratégias aproximadas. Heurísticas constroem boas soluções em tempo reduzido, enquanto metaheurísticas organizam mecanismos de exploração e intensificação para escapar de ótimos locais e melhorar soluções iterativamente.

Para problemas de alocação docente, há abordagens heurísticas e metaheurísticas que exploram diferentes representações e movimentos de vizinhança, como algoritmos genéticos e buscas locais especializadas (GUNAWAN; NG; ONG, 2008; HOSNY, 2012; SZWARC; BACH-DÁBROWSKA; BOCEWICZ, 2018). Esse corpo de técnicas é especialmente útil quando se deseja produzir soluções de boa qualidade sob limites de tempo, ou quando o modelo exato é utilizado como referência/validação.

2.3.1 Greedy Randomized Adaptive Search Procedure (GRASP)

O GRASP é uma metaheurística em duas fases: (i) uma construção gulosa aleatorizada, que produz uma solução inicial por escolhas guiadas por um critério de qualidade com componente probabilística; e (ii) uma busca local, que tenta melhorar a solução por movimentos em uma vizinhança definida. A aleatoriedade permite amostrar diferentes regiões do espaço de soluções, enquanto a busca local promove intensificação.

No contexto do Problema da Distribuição de Disciplinas (DPD), uma estratégia GRASP é um caminho natural por combinar regras gulosas (por exemplo, priorizar pares professor–disciplina com maior peso) com fases de melhoria que reatribuem disciplinas para reduzir violações e aumentar a qualidade (JHONATAN, 2025).

2.4 Matheurísticas

Matheurísticas combinam programação matemática com componentes heurísticos/metaheurísticos, buscando aproveitar simultaneamente a força de modelagem e os limites fornecidos por modelos exatos, e a flexibilidade de estratégias de busca em grandes espaços combinatórios (BALL, 2011; BOSCHETTI; LETCHFORD; MANIEZZO, 2023; MANIEZZO; STÜTZLE; VOSS, 2021). Em geral, a ideia é usar submodelos (ou modelos restritos) como operadores de melhoria, resolver subproblemas por MIP/CPLEX/SCIP, ou ainda usar informações do solucionador (limites, incumbentes, cortes) para orientar a busca.

Em problemas do tipo mochila e variantes, por exemplo, há linhas de trabalho que empregam resoluções exatas em subestruturas, gerando melhorias sistemáticas em torno de incumbentes (D'AMBROSIO et al., 2023; JOVANOVIC; VOSS, 2024). Essa mesma filosofia se estende ao DPD: parte-se de um modelo de PLI e explora-se uma vizinhança definida sobre variáveis binárias, reotimizando subproblemas com tempo controlado.

2.4.1 Large Neighborhood Search (LNS) e Fix-and-Relax

O Large Neighborhood Search (LNS) alterna etapas de *destroy* e *repair*: uma porção da solução corrente é liberada (ou removida) e, em seguida, o problema restrito é resolvido para reparar e potencialmente melhorar a solução. A principal vantagem é permitir movimentos de grande alcance, mantendo a viabilidade por meio de um procedimento de reparo eficaz (AHUJA et al., 2002; PISINGER; ROPKE, 2010). Uma implementação prática e amplamente adotada é usar um solucionador MIP na etapa de reparo, o que transforma o LNS em uma matheurística.

O esquema *fix-and-relax* pode ser visto como uma forma estruturada de definir vizinhanças: fixa-se a maior parte das variáveis (mantendo decisões da incumbente) e relaxa-se um subconjunto selecionado para reotimização, normalmente sob um limite de tempo. Esse desenho permite controlar o tamanho da vizinhança e o custo computacional de cada iteração, ao mesmo tempo em que preserva a capacidade do MIP de recombinar decisões de modo consistente. Assim, o método proposto neste trabalho utiliza o SCIP (GAMRATH et al., 2020) como motor de reotimização em vizinhanças geradas a partir de uma solução incumbente.

Por fim, este TCC se insere na linha de abordagens para o DPD que exploram tanto modelos exatos quanto estratégias de melhoria, dialogando com trabalhos recentes no problema (SILVA, 2024a) e com a motivação geral de matheurísticas de melhoria para problemas combinatórios de grande porte (SILVA, 2024b).

3 O Problema da Distribuição de Disciplinas (DPD)

3.1 Definição do Problema

O Problema da Distribuição de Disciplinas (DPD) consiste em atribuir professores a disciplinas de modo a respeitar restrições operacionais (carga horária mínima anual e máximas por semestre para cada professor, cobertura obrigatória de todas as disciplinas) e, simultaneamente, maximizar a qualidade da alocação. A qualidade de cada atribuição é mensurada por um peso que combina a preferência do professor pela disciplina e a compatibilidade entre a área de atuação do professor e a área da disciplina. Quando um professor é designado a uma disciplina fora de suas áreas de especialização, o modelo aplica uma penalidade (`area_penalty`), resultando em um coeficiente negativo na função objetivo.

Formalmente, temos um conjunto de P professores e um conjunto de C disciplinas. Cada disciplina $j \in C$ possui uma carga horária associada ch_j (em horas/aula ou créditos) e deve ser atribuída a exatamente um professor. Cada professor $i \in P$ possui uma carga horária mínima anual $\min_i$ e cargas horárias máximas $\max_{i,1}$ e $\max_{i,2}$ para o primeiro e segundo semestres, respectivamente. O problema é NP-difícil e sua modelagem como Programação Linear Inteira permite explorar tanto métodos exatos quanto heurísticos/matheurísticos para obtenção de soluções de boa qualidade.

3.2 Modelo Matemático Exato

3.2.1 Conjuntos e Índices

- $P = \{1, \ldots, |P|\}$: conjunto de professores, indexado por i;
- $C = \{1, \ldots, |C|\}$: conjunto de disciplinas, indexado por j;
- $C_1 \subseteq C$: disciplinas do primeiro semestre;
- $C_2 \subseteq C$: disciplinas do segundo semestre.

3.2.2 Parâmetros e Constantes

- $w_{i,j}$: peso/preferência da atribuição do professor i à disciplina j. Quando o professor possui competência na área da disciplina e alguma preferência, $w_{i,j}$ assume o valor

da preferência; caso contrário, aplica-se $w_{i,j} = -\text{area\_penalty}$;

- ch_j: carga horária (créditos) da disciplina j;
- $\min_i$: carga horária mínima anual do professor i;
- $\max_{i,1}$: carga horária máxima do professor i no primeiro semestre;
- $\max_{i,2}$: carga horária máxima do professor i no segundo semestre.

3.2.3 Variáveis de Decisão

$$x_{i,j} \in \{0,1\}, \quad i \in P,\ j \in C,$$

onde $x_{i,j} = 1$ indica que o professor i é designado à disciplina j, e $x_{i,j} = 0$ caso contrário.

3.2.4 Função Objetivo

O objetivo é maximizar a soma dos pesos das atribuições:

$$\max\ Z = \sum_{i \in P} \sum_{j \in C} w_{i,j}\, x_{i,j}.$$

Valores positivos de $w_{i,j}$ indicam preferência ou adequação, enquanto valores negativos (resultantes da penalidade por incompatibilidade de área) desencorajam atribuições inadequadas.

3.2.5 Restrições

3.2.5.1 Cobertura Obrigatória de Disciplinas

Cada disciplina deve ser atribuída a exatamente um professor:

$$\sum_{i \in P} x_{i,j} = 1, \quad \forall j \in C.$$

3.2.5.2 Carga Horária Mínima Anual

A soma das cargas horárias das disciplinas atribuídas ao professor i deve ser pelo menos $\min_i$:

$$\sum_{j \in C} ch_j\, x_{i,j} \geq \min_i, \quad \forall i \in P.$$

3.2.5.3 Carga Horária Máxima no Primeiro Semestre

A soma das cargas horárias das disciplinas do primeiro semestre atribuídas ao professor i não pode exceder $\max_{i,1}$:

$$\sum_{j \in C_1} ch_j\, x_{i,j} \leq \max_{i,1}, \quad \forall i \in P.$$

3.2.5.4 Carga Horária Máxima no Segundo Semestre

Analogamente, para o segundo semestre:

$$\sum_{j \in C_2} ch_j \, x_{i,j} \leq \max_{i,2}, \quad \forall i \in P.$$

3.3 Matheurística LNS Proposta

A matheurística desenvolvida neste trabalho baseia-se no Large Neighborhood Search (LNS) no esquema fix-and-relax. A cada iteração, a heurística parte de uma solução incumbente $\bar{x}$ e define uma vizinhança ao fixar a maior parte das variáveis binárias e liberar um subconjunto controlado de atribuições para reotimização. A estratégia é implementada no arquivo `heur_lns.c` e integra-se ao framework de heurísticas primais do SCIP.

3.3.1 Definição de Candidatos e Ordenação

A partir da solução incumbente $\bar{x}$, o algoritmo identifica todas as atribuições ativas (i.e., $\bar{x}_{i,j} = 1$), formando um conjunto de candidatos. Cada candidato corresponde a um par (professor, disciplina) e possui associado o valor `avgPreferenceWeight` do professor, uma métrica calculada como:

$$\text{avgPreferenceWeight}_i = \frac{\sum_{j \in J_i} w_{i,j}}{|J_i|} + \frac{|C|}{|J_i|},$$

onde J_i é o conjunto de disciplinas para as quais o professor i declarou preferência no arquivo de entrada, e $|C|$ é o número total de disciplinas. O primeiro termo representa a média aritmética dos pesos de preferência, enquanto o segundo termo ($|C|/|J_i|$) introduz um bônus inversamente proporcional ao grau de versatilidade do professor: professores mais especializados (com menos preferências declaradas) recebem valores maiores, indicando maior prioridade para reotimização devido às suas restrições de alocação.

O conjunto de candidatos é ordenado segundo o parâmetro `lns_order`:

- `"decrescente"` (padrão): candidatos com maior `avgPreferenceWeight` aparecem primeiro, priorizando professores mais especializados (menor número de preferências). Estes professores possuem menos opções alternativas de atribuição, sendo candidatos naturais para reotimização;
- `"crescente"`: ordena em ordem ascendente, priorizando professores mais versáteis (maior número de preferências). Esta estratégia busca explorar a flexibilidade de realocação dos professores generalistas.

3.3.2 Destruição (Destroy)

Após a ordenação, o algoritmo seleciona os primeiros $\lfloor \text{nCands} \times \text{lns\_perc} \rfloor$ candidatos, onde `lns_perc` $\in (0, 1)$ é a taxa de destruição configurada pelo usuário. Para cada candidato selecionado, a variável $x_{i,j}$ correspondente é liberada (i.e., seu valor é desfixado), removendo a atribuição da solução incumbente. As demais variáveis permanecem fixadas nos valores da incumbente.

3.3.3 Reparo (Repair) via Sub-MIP

Com o subconjunto de variáveis liberado, o algoritmo resolve um sub-MIP no SCIP auxiliar (mantido em memória pela estrutura `heurdata->subscip`). O sub-MIP possui:

- Todas as restrições do problema original;
- Variáveis fixadas conforme o vetor `fixed[]`;
- Limite de tempo configurado pelo parâmetro `lns_time`;
- Limite de função objetivo (`objlimit`) ajustado para parar assim que uma solução melhor que a incumbente for encontrada.

A resolução do sub-MIP busca reatribuir as disciplinas liberadas de modo a melhorar a função objetivo, respeitando todas as restrições de carga horária e cobertura. Se uma solução melhor é encontrada, ela é transferida para o SCIP principal e passa a ser a nova incumbente.

3.3.4 Integração com o SCIP

A heurística LNS é registrada como uma heurística primal do SCIP com prioridade e frequência configuráveis. A cada chamada, a heurística:

1. Verifica se existe uma solução incumbente viável;
2. Extrai os candidatos ativos e ordena conforme `lns_order`;
3. Libera uma fração `lns_perc` das atribuições;
4. Resolve o sub-MIP com limite de tempo `lns_time`;
5. Caso o sub-MIP retorne uma solução de melhor qualidade, adiciona a nova solução ao pool do SCIP principal.

A abordagem permite explorar vizinhanças de tamanho controlado, equilibrando a intensificação (ao focar em atribuições específicas) e a diversificação (ao permitir reatribuições significativas). Os parâmetros `lns_perc`, `lns_order` e `lns_time` oferecem controle sobre o trade-off entre qualidade da solução e tempo computacional, sendo objeto de análise experimental detalhada no próximo capítulo.

4 Resultados e Análise Computacional

4.1 Instâncias e Ambiente de Testes

Os experimentos computacionais foram conduzidos utilizando um conjunto de 42 instâncias de teste geradas sinteticamente, abrangendo diferentes cenários do Problema da Distribuição de Disciplinas (DPD). As instâncias variam em tamanho (62 a 214 disciplinas), número de professores (22 a 69), complexidade das restrições de carga horária e distribuição de preferências entre áreas de conhecimento.

Para cada experimento, foram selecionadas aleatoriamente 10 instâncias dentre as 42 disponíveis, garantindo representatividade estatística e mantendo o tempo total de experimentação viável. As mesmas instâncias foram utilizadas em todas as configurações testadas de um mesmo parâmetro, assegurando comparabilidade direta dos resultados.

Todas as execuções foram realizadas com limite de tempo global de 400 segundos por instância. O solver utilizado foi o SCIP 7.0 (GAMRATH et al., 2020), integrado com as heurísticas LNS e GRASP desenvolvidas neste trabalho. A calibração dos parâmetros da heurística LNS foi conduzida de forma incremental, conforme a abordagem *one-factor-at-a-time*: primeiro determinou-se a melhor taxa de destruição, em seguida o melhor sentido de ordenação e, finalmente, o melhor limite de tempo para o sub-MIP. Essa estratégia permite isolar o efeito de cada parâmetro e identificar a configuração mais adequada de forma sistemática.

Para a seleção do melhor valor de cada parâmetro, adotou-se a seguinte hierarquia de critérios de desempate: (i) tempo de execução, (ii) gap de otimalidade e (iii) número de nós restantes na árvore de Branch-and-Bound (*nodes left*). Quando os critérios de maior prioridade apresentaram empate entre as configurações, o critério subsequente foi utilizado como desempate.

4.2 Análise Interna das Configurações da LNS

4.2.1 Impacto da Taxa de Destruição (10%, 25%, 50% e 75%)

A primeira análise experimental investigou o impacto da taxa de destruição (`lns_perc`) na qualidade das soluções e no desempenho computacional. Foram testadas quatro configurações: 10%, 25%, 50% e 75% de destruição, mantendo fixos o tempo máximo do LNS (30s por chamada), a ordenação (crescente) e o tempo total de execução (400s).

Tempo de Execução. A Tabela 1 apresenta os tempos de execução para as 10

Tabela 1 – Tempo de execução (segundos) para diferentes taxas de destruição do LNS. Configuração: ordenação crescente, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**10%**	**25%**	**50%**	**75%**	**Melhor**
input2	79	24	4,04	17,07	17,66	16,98	17,04	Relax.
input5	84	29	113,03	145,27	145,04	145,71	143,63	Relax.
input6	113	31	1,07	2,47	2,40	2,33	2,22	Relax.
input7	99	28	400,02	400,02	400,02	400,02	400,02	Empate
input9	133	41	17,06	72,52	72,33	72,57	73,48	Relax.
input13	125	39	29,13	60,21	59,56	59,04	59,43	Relax.
input19	114	39	2,11	1,39	1,44	1,42	1,47	LNS 10%
input20	113	40	0,95	4,69	4,56	4,59	5,02	Relax.
input26	185	60	33,73	25,50	24,67	24,81	24,38	LNS 75%
input34	137	42	21,92	26,68	26,08	26,38	26,41	Relax.
Média	114	39	19,49	25,38	25,38	25,59	25,39	Relax.
Total	1.296	412	**623,05**	755,82	753,76	753,84	753,10	**Relax.**

Tabela 2 – Número de nós restantes na árvore de Branch-and-Bound para diferentes taxas de destruição. Configuração: ordenação crescente, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**10%**	**25%**	**50%**	**75%**	**Melhor**
input2	79	24	0	0	0	0	0	–
input5	84	29	0	0	0	0	0	–
input6	113	31	0	0	0	0	0	–
input7	99	28	66.230	62.628	60.540	61.056	**58.921**	**75%**
input9	133	41	0	0	0	0	0	–
input13	125	39	0	0	0	0	0	–
input19	114	39	0	0	0	0	0	–
input20	113	40	0	0	0	0	0	–
input26	185	60	0	0	0	0	0	–
input34	137	42	0	0	0	0	0	–
Média	114	39	6.623	6.263	6.054	6.106	5.892	LNS 75%
Total	1.296	412	66.230	62.628	60.540	61.056	**58.921**	**LNS 75%**

Tabela 3 – Gap de otimalidade (%) para diferentes taxas de destruição. Configuração: ordenação crescente, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**10%**	**25%**	**50%**	**75%**	**Melhor**
input2	79	24	0,00	0,00	0,00	0,00	0,00	Empate
input5	84	29	0,00	0,00	0,00	0,00	0,00	Empate
input6	113	31	0,00	0,00	0,00	0,00	0,00	Empate
input7	99	28	0,60	0,60	0,60	0,60	0,60	Empate
input9	133	41	0,00	0,00	0,00	0,00	0,00	Empate
input13	125	39	0,00	0,00	0,00	0,00	0,00	Empate
input19	114	39	0,00	0,00	0,00	0,00	0,00	Empate
input20	113	40	0,00	0,00	0,00	0,00	0,00	Empate
input26	185	60	0,00	0,00	0,00	0,00	0,00	Empate
input34	137	42	0,00	0,00	0,00	0,00	0,00	Empate
Média	114	39	0,06	0,06	0,06	0,06	0,06	Empate
Total	1.296	412	0,60	0,60	0,60	0,60	0,60	**Empate**

Tabela 4 – Tempo de execução (segundos) comparando ordenação crescente e decrescente. Configuração: 75% de destruição, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**Crescente**	**Decrescente**	**Melhor**
input4	123	46	400,05	400,04	400,05	Crescente
input5	214	69	114,18	140,64	141,47	Crescente
input12	115	37	5,79	25,70	25,84	Crescente
input15	94	30	400,03	400,02	400,02	Decrescente
input16	89	28	3,07	0,72	0,72	Decrescente
input17	108	38	400,04	400,04	400,03	Decrescente
input20	104	37	0,90	4,40	4,32	Decrescente
input33	104	36	400,03	400,03	400,03	Decrescente
input37	152	53	68,00	31,61	31,62	Crescente
input38	62	22	0,16	0,28	0,28	Decrescente
Média	106	37	91,09	180,31	180,38	Crescente
Total	1.271	433	**1.792,24**	1.803,48	1.804,48	**Relax.**

Tabela 5 – Número de nós restantes comparando ordenação crescente e decrescente. Configuração: 75% de destruição, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**Crescente**	**Decrescente**	**Melhor**
input4	123	46	19.797	19.541	19.507	Decrescente
input5	214	69	0	0	0	–
input12	115	37	0	0	0	–
input15	94	30	64.711	**61.546**	62.452	**Crescente**
input16	89	28	0	0	0	–
input17	108	38	4.673	4.667	4.652	Decrescente
input20	104	37	0	0	0	–
input33	104	36	50.279	**46.243**	47.063	**Crescente**
input37	152	53	0	0	0	–
input38	62	22	0	0	0	–
Média	106	37	13.946	13.200	13.367	Crescente
Total	1.271	433	139.460	**131.997**	133.674	**Crescente**

Tabela 6 – Gap de otimalidade (%) comparando ordenação crescente e decrescente. Configuração: 75% de destruição, tempo LNS = 30s, limite global = 400s.

Instância	**Disciplinas**	**Profs**	**Relax.**	**Crescente**	**Decrescente**	**Melhor**
input4	123	46	0,20	0,20	0,20	Empate
input5	214	69	0,00	0,00	0,00	Empate
input12	115	37	0,00	0,00	0,00	Empate
input15	94	30	0,73	0,73	0,73	Empate
input16	89	28	0,00	0,00	0,00	Empate
input17	108	38	0,20	0,20	0,20	Empate
input20	104	37	0,00	0,00	0,00	Empate
input33	104	36	0,49	0,49	0,49	Empate
input37	152	53	0,00	0,00	0,00	Empate
input38	62	22	0,00	0,00	0,00	Empate
Média	106	37	0,16	0,16	0,16	Empate
Total	1.271	433	1,62	1,62	1,62	**Empate**

Tabela 7 – Tempo de execução (segundos) para diferentes limites de tempo do sub-MIP. Configuração: 75% de destruição, ordenação crescente, limite global = 400s.

Inst.	**Disc.**	**Prof.**	**Relax.**	**10s**	**50s**	**100s**	**200s**	**Melhor**
input6	79	24	1,19	2,40	2,14	2,15	**2,14**	**200s**
input14	99	28	14,21	22,31	68,47	118,44	168,12	Relax.
input15	94	30	400,03	400,02	400,02	**76,31**	77,09	**100s**
input21	104	31	6,98	**3,02**	3,04	3,05	3,10	**10s**
input22	133	41	400,05	400,04	400,04	400,04	400,04	10s
input23	118	35	1,55	**11,41**	12,80	12,66	12,50	**10s**
input27	125	39	4,93	7,45	7,34	7,33	**7,31**	**200s**
input29	103	39	400,03	400,03	19,42	19,40	**19,39**	**200s**
input33	104	36	400,03	400,03	400,03	400,03	400,03	200s
input38	62	22	0,16	0,28	0,28	**0,28**	0,28	**100s**
Média	104	33	162,92	164,77	162,11	160,27	**159,46**	**200s**
Total	1.125	358	1.629,16	1.647,01	1.313,60	**1.039,68**	1.090,00	**100s**

Tabela 8 – Número de nós restantes para diferentes limites de tempo do sub-MIP. Configuração: 75% de destruição, ordenação crescente, limite global = 400s.

Inst.	**Disc.**	**Prof.**	**Relax.**	**10s**	**50s**	**100s**	**200s**	**Melhor**
input6	79	24	0	0	0	0	0	–
input14	99	28	0	0	0	0	0	–
input15	94	30	54.355	63.056	59.819	**0**	**0**	**100s**
input21	104	31	0	0	0	0	0	–
input22	133	41	21.528	29.161	**25.100**	50.411	33.000	**50s**
input23	118	35	0	0	0	0	0	–
input27	125	39	0	0	0	0	0	–
input29	103	39	12.643	4.347	**0**	**0**	**0**	**50s**
input33	104	36	47.460	48.110	45.505	1.524	**539**	**200s**
input38	62	22	0	0	0	0	0	–
Média	104	33	13.599	14.467	13.042	5.194	**3.354**	**200s**
Total	1.125	358	135.986	144.674	130.424	51.935	**33.539**	**200s**

Tabela 9 – Gap de otimalidade (%) para diferentes limites de tempo do sub-MIP. Configuração: 75% de destruição, ordenação crescente, limite global = 400s.

Inst.	**Disc.**	**Prof.**	**Relax.**	**10s**	**50s**	**100s**	**200s**	**Melhor**
input6	79	24	0,00	0,00	0,00	0,00	0,00	Empate
input14	99	28	0,00	0,00	0,00	0,00	0,00	Empate
input15	94	30	0,73	0,73	0,73	**0,00**	**0,00**	**100s**
input21	104	31	0,00	0,00	0,00	0,00	0,00	Empate
input22	133	41	0,32	0,32	0,32	0,48	0,48	Relax.
input23	118	35	0,00	0,00	0,00	0,00	0,00	Empate
input27	125	39	0,00	0,00	0,00	0,00	0,00	Empate
input29	103	39	0,19	0,19	**0,00**	**0,00**	**0,00**	**50s**
input33	104	36	0,49	0,49	0,49	**0,24**	**0,24**	**100s**
input38	62	22	0,00	0,00	0,00	0,00	0,00	Empate
Média	104	33	0,17	0,17	0,15	0,07	**0,07**	**100s**
Total	1.125	358	1,73	1,73	1,54	**0,72**	**0,72**	**100s**

Tabela 10 – Tempo de execução (segundos) comparando Relaxação, LNS, GRASP e GRASP|LNS. Configuração LNS: 75% destruição, crescente, 200s. Configuração GRASP: max_iter=1, alpha=0.8, local_search=0.

Instância	Disciplinas	Profs	Relax.	LNS	GRASP	GRASP|LNS
input5	214	69	112,63	**54,46**	400,12	83,09
input11	100	31	1,54	2,59	3,90	**0,83**
input15	94	30	400,03	**79,90**	400,02	400,02
input16	89	28	3,07	0,75	8,38	**0,36**
input19	95	29	2,03	**1,45**	5,74	2,73
input21	104	31	5,93	**3,16**	13,45	4,42
input25	107	34	15,63	**9,27**	37,64	9,54
input30	109	36	3,21	**1,36**	7,94	2,36
input35	169	53	186,72	**63,27**	400,07	71,20
input42	142	35	**6,91**	8,18	20,19	13,39
Média	106	33	73,76	**22,44**	129,75	58,79
Total	1.329	409	744,13	**230,05**	1.314,27	594,92

Tabela 11 – Número de nós restantes comparando Relaxação, LNS, GRASP e GRASP|LNS. Configuração LNS: 75% destruição, crescente, 200s. Configuração GRASP: max_iter=1, alpha=0.8, local_search=0.

Instância	Disciplinas	Profs	Relax.	LNS	GRASP	GRASP|LNS
input5	214	69	0	**0**	8.037	**0**
input11	100	31	0	**0**	0	**0**
input15	94	30	64.364	**0**	7.526	2.324
input16	89	28	0	**0**	0	**0**
input19	95	29	0	**0**	0	**0**
input21	104	31	0	**0**	0	**0**
input25	107	34	0	**0**	0	**0**
input30	109	36	0	**0**	0	**0**
input35	169	53	0	**0**	20.740	**0**
input42	142	35	0	**0**	0	**0**
Média	106	33	6.436	**0**	3.630	232
Total	1.329	409	64.364	**0**	36.303	2.324

instâncias selecionadas. A configuração de relaxação (sem LNS) obteve o menor tempo médio, com 19,49 segundos, contra 25,39 segundos da melhor configuração LNS (75%). Esse resultado é esperado, pois o *overhead* introduzido pelas chamadas ao sub-MIP do LNS aumenta o tempo total, sobretudo em instâncias que já são resolvidas de forma ótima pelo Branch-and-Bound puro. Nove das dez instâncias foram resolvidas até a otimalidade por todas as configurações, e apenas a instância input7 atingiu o limite de tempo. Assim, o tempo de execução não foi suficiente para diferenciar as configurações LNS entre si.

Gap de Otimalidade. Em termos de gap, todas as configurações — incluindo a relaxação — apresentaram desempenho equivalente (0,60% de gap total), com apenas a instância input7 permanecendo sem solução ótima comprovada (ver Tabela 3). Esse empate impediu o uso do gap como critério de seleção.

Tabela 12 – Gap de otimalidade (%) comparando Relaxação, LNS, GRASP e GRASP|LNS. Configuração LNS: 75% destruição, crescente, 200s. Configuração GRASP: max_iter=1, alpha=0.8, local_search=0.

Instância	**Disciplinas**	**Profs**	**Relax.**	**LNS**	**GRASP**	**GRASP|LNS**
input5	214	69	0,00	**0,00**	1,01	**0,00**
input11	100	31	0,00	**0,00**	0,00	**0,00**
input15	94	30	0,73	**0,00**	0,24	0,24
input16	89	28	0,00	**0,00**	0,00	**0,00**
input19	95	29	0,00	**0,00**	0,00	**0,00**
input21	104	31	0,00	**0,00**	0,00	**0,00**
input25	107	34	0,00	**0,00**	0,00	**0,00**
input30	109	36	0,00	**0,00**	0,00	**0,00**
input35	169	53	0,00	**0,00**	2,21	**0,00**
input42	142	35	0,00	**0,00**	0,00	**0,00**
Média	106	33	0,07	**0,00**	0,35	0,02
Total	1.329	409	0,73	**0,00**	3,46	0,24

Nós Restantes (*Nodes Left*). Uma vez que tanto o tempo quanto o gap resultaram em empate, o critério de desempate utilizado foi a quantidade de nós restantes na árvore de Branch-and-Bound ao final da execução, métrica que reflete o progresso em direção à prova de otimalidade. A Tabela 2 revela que a configuração com 75% de destruição alcançou o menor total de nós restantes: **58.921 nós**, comparado a 66.230 (relaxação), 62.628 (10%), 60.540 (25%) e 61.056 (50%). Todas as configurações obtiveram resultado idêntico em 9 das 10 instâncias (resolvidas até a otimalidade), mas na instância crítica input7, a taxa de 75% demonstrou maior capacidade de exploração e poda da árvore. A Figura 1 apresenta visualmente a comparação dos totais, e a Figura 2 exibe a média por instância, evidenciando a superioridade da configuração de 75%.

Conclusão Parcial. A taxa de destruição de **75%** foi selecionada para as análises subsequentes, pois demonstrou melhor capacidade de redução da árvore de busca na instância difícil. Esse resultado indica que vizinhanças maiores permitem ao LNS realizar explorações mais agressivas e produtivas do espaço de soluções, reforçando a estratégia de intensificação adotada pela matheurística.

4.2.2 Impacto do Sentido de Ordenação (Crescente vs. Decrescente)

Com a taxa de destruição fixada em 75%, investigou-se o impacto do sentido de ordenação dos candidatos pelo valor de `avgPreferenceWeight`. Conforme descrito no Capítulo 3, a ordenação crescente prioriza a liberação de professores mais versáteis (maior número de preferências declaradas), enquanto a decrescente prioriza professores mais especializados (menor número de preferências).

Tempo de Execução. A Tabela 4 revela desempenho temporal praticamente idêntico entre as duas estratégias: 180,31 segundos (crescente) versus 180,38 segundos

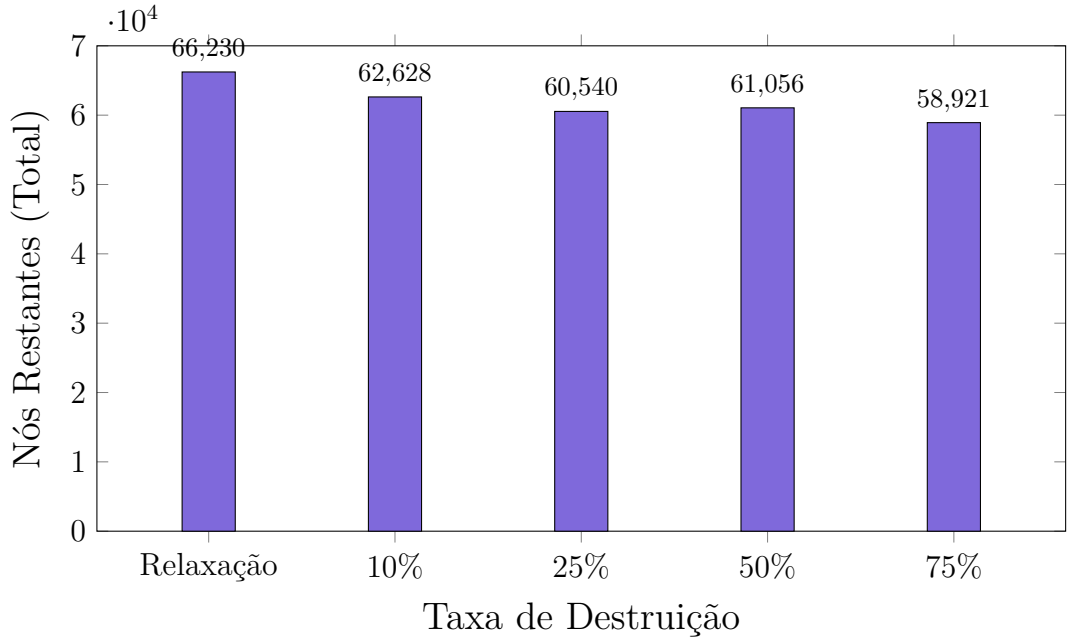


Figura 1 – Número total de nós restantes na árvore de Branch-and-Bound para diferentes taxas de destruição do LNS. A configuração de 75% apresentou o melhor desempenho com 58.921 nós restantes.

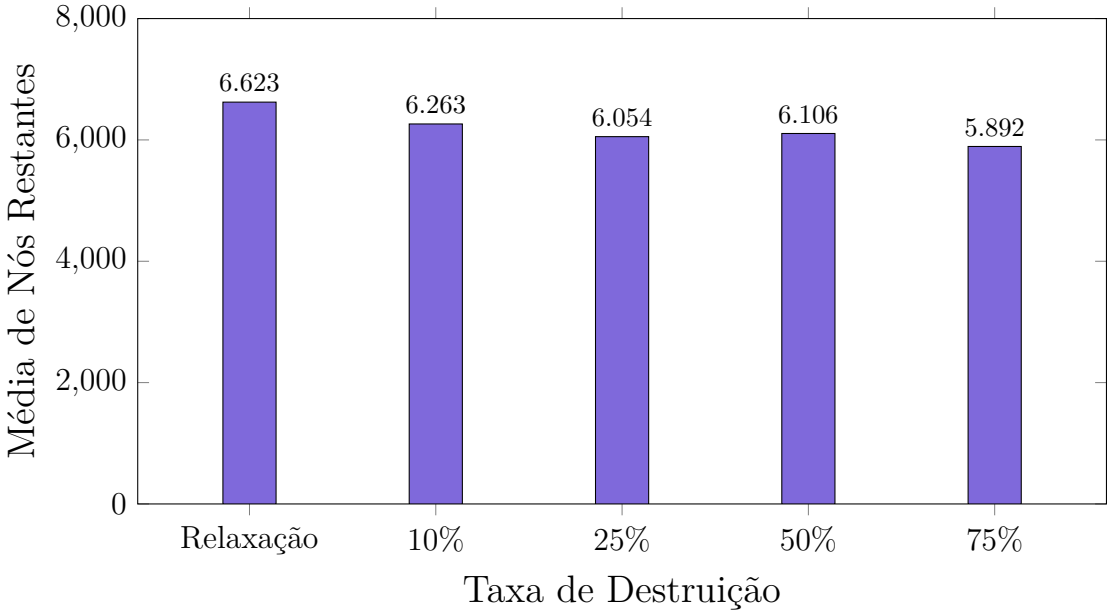


Figura 2 – Média de nós restantes na árvore de Branch-and-Bound para diferentes taxas de destruição do LNS. A configuração de 75% apresentou a menor média, com 5.892 nós restantes por instância.

(decrescente) no tempo médio, com diferença inferior a 0,1 segundo. Das 10 instâncias testadas, quatro atingiram o limite de tempo em ambas as configurações (input4, input15, input17 e input33), e as demais apresentaram variações inferiores a 1 segundo. O tempo, portanto, não permitiu diferenciar as estratégias.

Gap de Otimalidade. Ambas as configurações mantiveram gap total idêntico de 1,62%, distribuído entre quatro instâncias não resolvidas de forma ótima (input4, input15, input17 e input33), com gaps individuais variando de 0,20% a 0,73% (ver Tabela 6). O empate no gap impediu, novamente, seu uso como critério de seleção.

Nós Restantes (*Nodes Left*). A métrica de nós restantes revelou vantagem para a ordenação crescente. A Tabela 5 mostra que a configuração crescente resultou em **131.997 nós restantes** no total, enquanto a decrescente deixou 133.674 nós — uma redução de 1.677 nós (1,25%). A diferença se concentra em duas instâncias críticas: input15 (61.546 vs. 62.452 nós) e input33 (46.243 vs. 47.063 nós). A Figura 3 apresenta a comparação visual dos totais, e a Figura 4 exibe a média por instância.

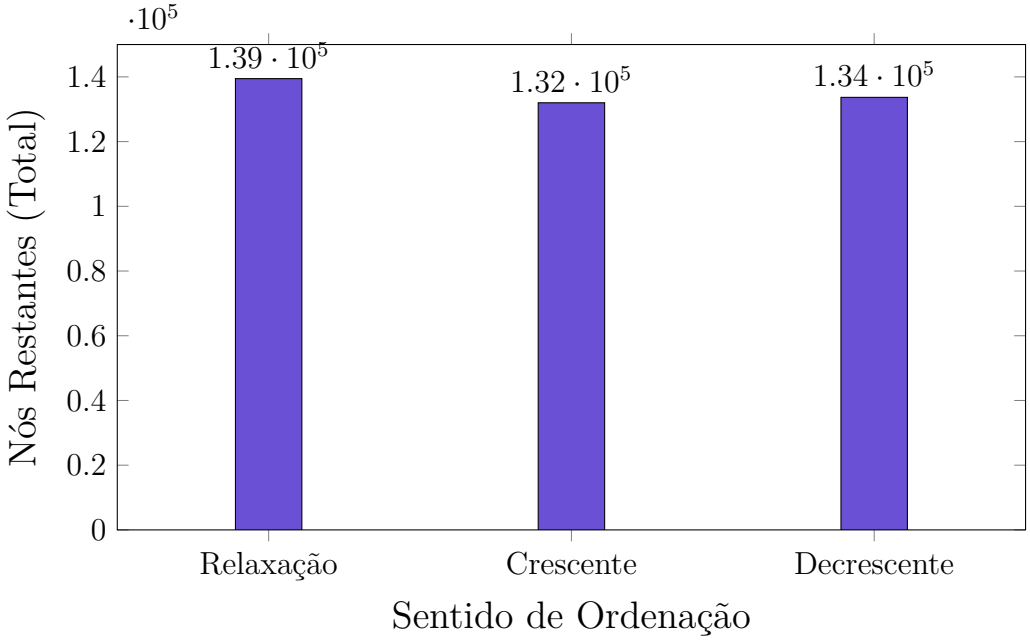


Figura 3 – Comparação do número total de nós restantes entre ordenação crescente e decrescente. A ordenação crescente (priorizando professores versáteis) obteve melhor desempenho com 131.997 nós restantes.

Interpretação. A superioridade da ordenação **crescente** sugere que, para o DPD, priorizar a realocação de professores mais versáteis (generalistas) oferece maior flexibilidade para o sub-MIP encontrar rearranjos que melhorem a solução global. Professores generalistas possuem preferências por um maior número de disciplinas, o que amplia o espaço de busca do sub-MIP e facilita a satisfação simultânea de múltiplas restrições de carga horária. A ordenação crescente foi, portanto, adotada para as análises subsequentes.

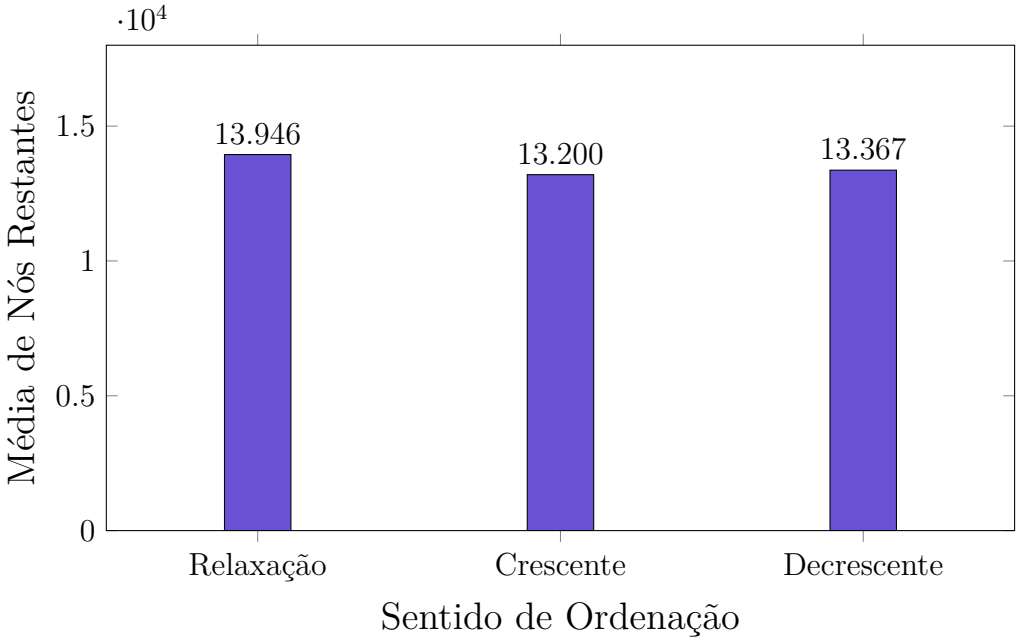


Figura 4 – Média de nós restantes por instância comparando ordenação crescente e decrescente. A ordenação crescente obteve a menor média, com 13.200 nós restantes.

4.2.3 Impacto do Tempo Máximo do LNS (10s, 50s, 100s e 200s)

Com os parâmetros de taxa de destruição (75%) e ordenação (crescente) definidos, investigou-se o impacto do limite de tempo do sub-MIP (`lns_time`). Foram testados quatro valores: 10s, 50s, 100s e 200s, mantendo o limite global de 400 segundos por instância.

Tempo de Execução. A Tabela 7 apresenta resultados contraintuitivos: embora se esperasse que limites maiores de LNS aumentassem o tempo total, observou-se comportamento não monotônico. O tempo médio foi de 164,77s (10s), 162,11s (50s), 160,27s (100s) e **159,46s (200s)**. O tempo total agregado reforça essa tendência: 1.629,16s (relaxação), 1.647,01s (10s), 1.313,60s (50s), **1.039,68s (100s)** e 1.090,00s (200s). Destaca-se a instância input15, em que as configurações de 100s e 200s resolveram o problema em $\sim$77s, contra 400s nas demais. Contudo, os resultados de tempo médio e tempo total apontam para configurações diferentes (200s e 100s, respectivamente), não permitindo uma conclusão unívoca por este critério.

Essa redução de tempo com limites maiores de LNS pode ser explicada pela capacidade de encontrar soluções de melhor qualidade mais cedo: uma incumbente mais forte permite podas mais agressivas na árvore de Branch-and-Bound, acelerando a convergência. Configurações com tempo insuficiente (10s) produzem melhorias marginais que não compensam o *overhead* da heurística.

Gap de Otimalidade. A Tabela 9 mostra que as configurações de 100s e 200s empataram com gap total de **0,72%**, substancialmente inferior ao obtido pela relaxação e pelo LNS com 10s (ambos com 1,73%). O destaque vai para a instância input15, resolvida

até a otimalidade (gap 0,00%) apenas pelas configurações de 100s e 200s, e para a instância input33, cujo gap caiu de 0,49% (relaxação) para 0,24% com 100s e 200s. Entretanto, o empate entre 100s e 200s impediu o uso do gap como critério final.

Nós Restantes (*Nodes Left*). A Tabela 8 foi o critério de desempate definitivo. O total de nós restantes foi: 135.986 (relaxação), 144.674 (10s), 130.424 (50s), 51.935 (100s) e **33.539 (200s)**. A configuração de 200s reduziu os nós em 75,3% em relação à relaxação e em 76,8% em relação ao LNS com 10s. As instâncias input15 e input29 foram resolvidas completamente (0 nós restantes) pelas configurações de 100s e 200s, enquanto permaneceram com dezenas de milhares de nós nas configurações inferiores. Na instância input33, a configuração de 200s deixou apenas 539 nós, contra 1.524 com 100s. A Figura 5 apresenta os totais, e a Figura 6 ilustra a média por instância, evidenciando a clara superioridade do limite de 200s.

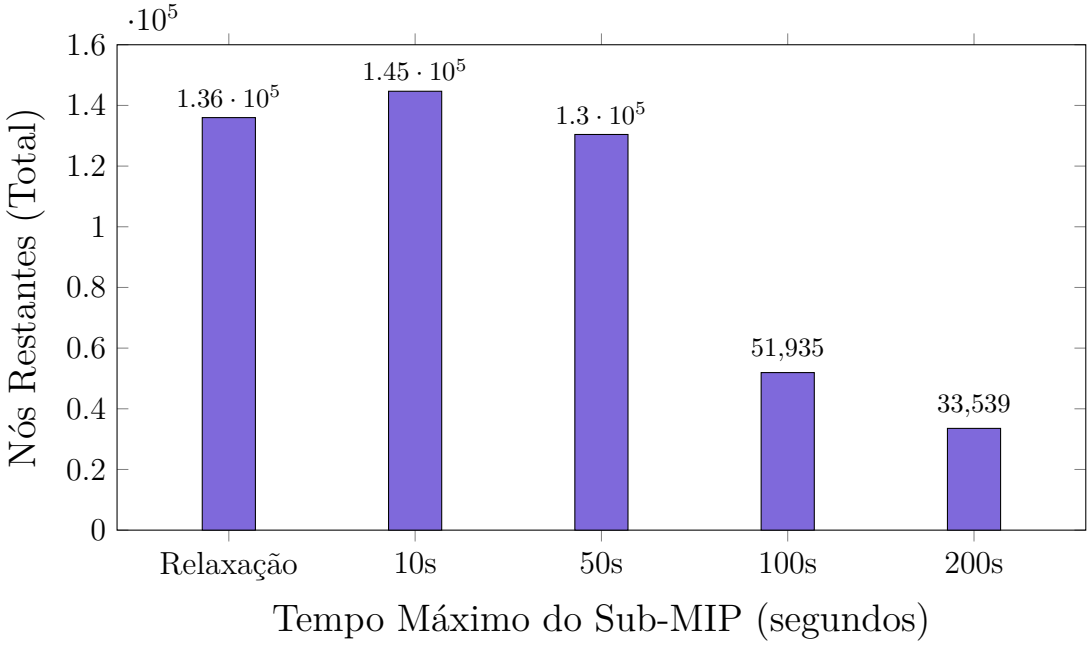


Figura 5 – Impacto do limite de tempo do sub-MIP no número de nós restantes. A configuração de 200s demonstrou o melhor desempenho com 33.539 nós, representando uma redução de 75,3% em relação à relaxação pura.

Conclusão Parcial. O limite de **200 segundos** demonstrou o melhor desempenho global, reduzindo drasticamente a árvore de busca e resolvendo instâncias que outras configurações não conseguiram. Para problemas combinatórios como o DPD, investir mais tempo em cada chamada do LNS é compensado por soluções de maior qualidade, que aceleram a convergência do Branch-and-Bound. O *trade-off* entre intensidade (tempo por chamada) e frequência (número de chamadas) favorece claramente a intensidade para este problema.

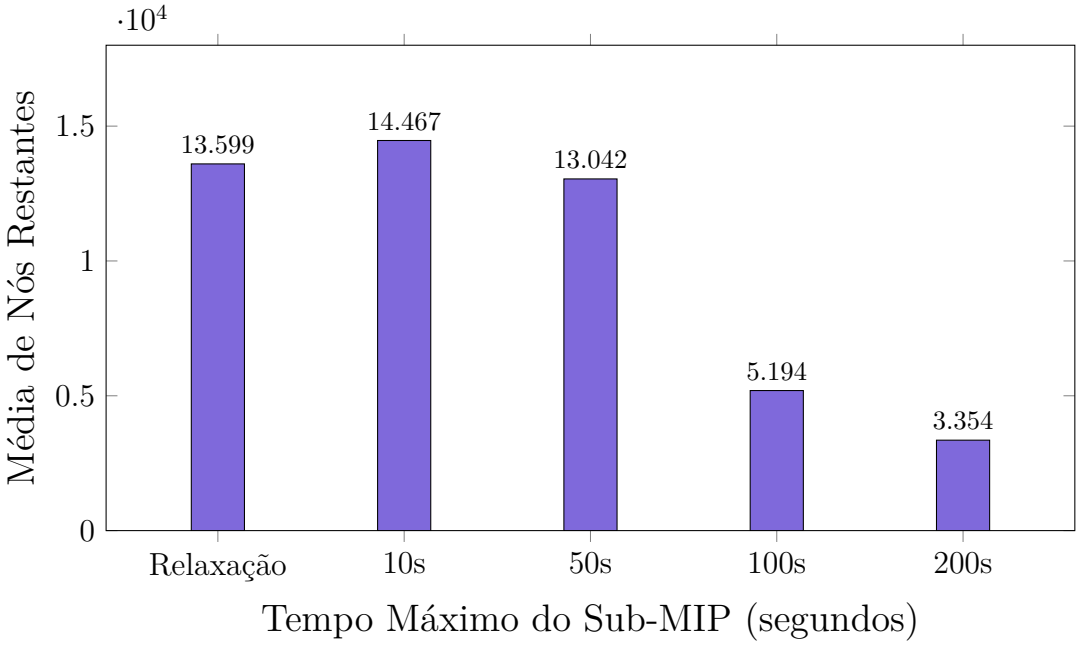


Figura 6 – Média de nós restantes por instância para diferentes limites de tempo do sub-MIP. A configuração de 200s alcançou a menor média, com 3.354 nós, representando uma redução de 75,3% em relação à relaxação pura.

4.2.4 Resumo da Configuração Selecionada

A Tabela 13 resume os parâmetros selecionados ao longo da análise incremental.

Tabela 13 – Configuração final da heurística LNS selecionada pela análise incremental.

Parâmetro	**Valores Testados**	**Selecionado**	**Critério Decisivo**
Taxa de destruição (`lns_perc`)	10%, 25%, 50%, 75%	75%	Nós restantes
Ordenação (`lns_order`)	Crescente, Decrescente	Crescente	Nós restantes
Tempo do sub-MIP (`lns_time`)	10s, 50s, 100s, 200s	200s	Nós restantes

Nota-se que, em todas as etapas, o tempo de execução e o gap de otimalidade não foram suficientes para diferenciar as configurações, sendo a métrica de nós restantes o critério de desempate definitivo. Esse padrão sugere que, para as instâncias avaliadas, o LNS atua predominantemente como acelerador da prova de otimalidade, reduzindo o esforço do Branch-and-Bound por meio de incumbentes de melhor qualidade.

4.3 Comparativo de Desempenho: LNS, GRASP e Modelo Exato

Com os parâmetros da LNS calibrados (75% de destruição, ordenação crescente e tempo de sub-MIP de 200s), realizou-se um comparativo final entre quatro configurações: (i) **Relaxação** (Branch-and-Bound puro, sem heurísticas primais customizadas), (ii) **LNS** (apenas a heurística LNS proposta), (iii) **GRASP** (apenas a heurística GRASP, com `max_iter=1`, $\alpha = 0{,}8$ e sem busca local (JHONATAN, 2025)) e (iv) **GRASP|LNS** (ambas

as heurísticas ativadas simultaneamente). As mesmas 10 instâncias foram utilizadas em todas as configurações.

4.3.1 Análise do Tempo de Execução

A Tabela 10 apresenta os tempos de execução para cada configuração. O **LNS** obteve o menor tempo médio: **22,44 segundos**, representando uma redução de 69,6% em relação à relaxação (73,76s) e de 82,7% em relação ao GRASP (129,75s). A configuração GRASP|LNS alcançou 58,79s de tempo médio, posicionando-se entre a relaxação e o LNS.

Dentre as 10 instâncias, o LNS foi o mais rápido em 7 delas, com destaque para a instância input15: enquanto a relaxação e o GRASP atingiram o limite de 400s, o LNS resolveu a instância em 79,90s. A instância input35, com 169 disciplinas e 53 professores, exemplifica o impacto da heurística: o LNS concluiu em 63,27s contra 186,72s da relaxação e 400,07s do GRASP. A Figura 7 ilustra a comparação dos tempos médios.

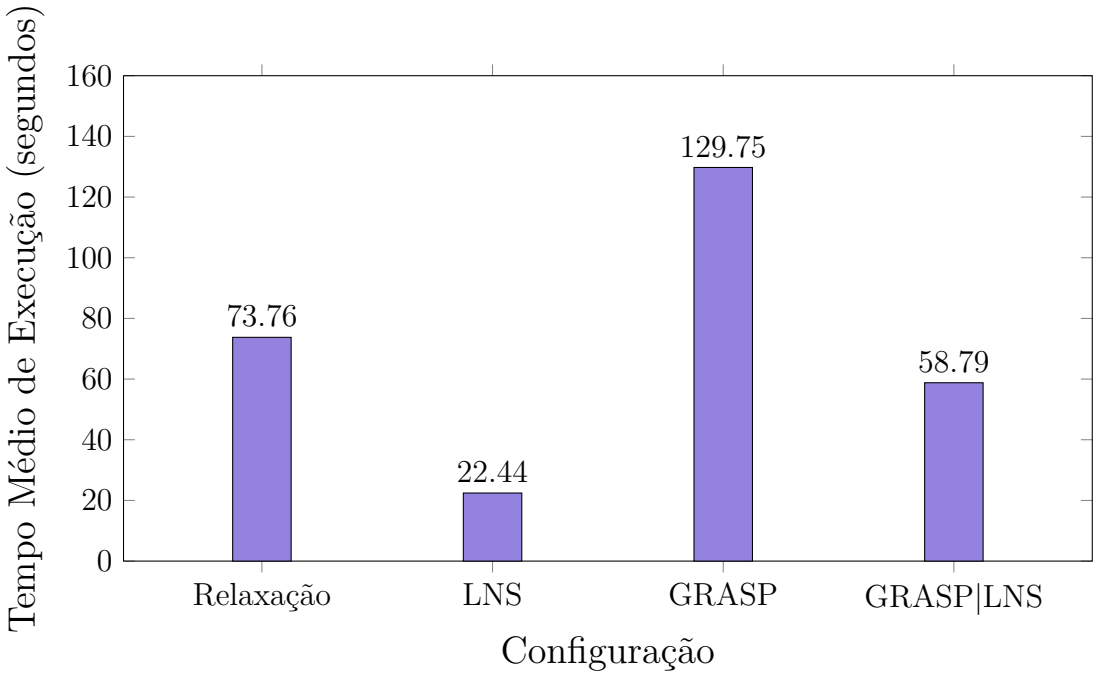


Figura 7 – Tempo médio de execução (segundos) para cada configuração. O LNS obteve o menor tempo médio (22,44s), representando uma redução de 69,6% em relação à relaxação pura e de 82,7% em relação ao GRASP.

O GRASP apresentou o pior desempenho temporal, atingindo o limite de tempo em 3 das 10 instâncias (input5, input15 e input35). Esse resultado sugere que a heurística construtiva do GRASP, embora forneça soluções iniciais viáveis rapidamente, não contribui de forma eficaz para a redução da árvore de busca do Branch-and-Bound.

4.3.2 Análise do Gap de Otimalidade

A Tabela 12 apresenta o gap de otimalidade para cada configuração. O **LNS** atingiu **gap total de 0,00%**, provando a otimalidade de todas as 10 soluções encontradas. A relaxação obteve 0,73% (devido à instância input15 não resolvida), a GRASP|LNS obteve 0,24% (input15 com gap residual) e o GRASP apresentou o pior resultado com 3,46% de gap total, com três instâncias sem solução ótima comprovada (input5 com 1,01%, input15 com 0,24% e input35 com 2,21%). A Figura 8 apresenta a comparação visual.

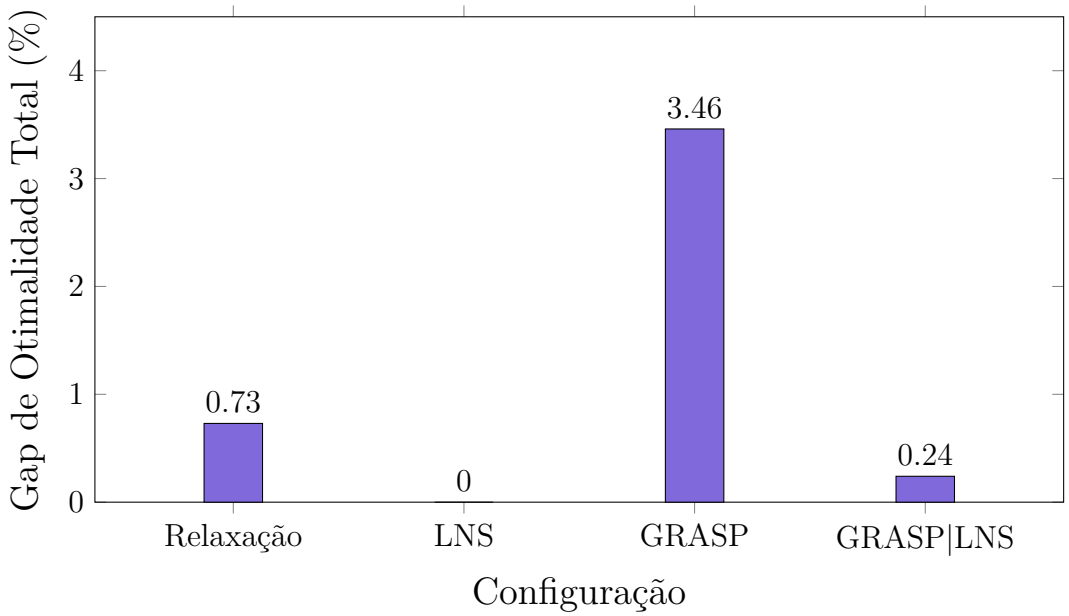


Figura 8 – Gap de otimalidade total (%) para cada configuração. O LNS atingiu gap zero, provando a otimalidade de todas as soluções encontradas.

A superioridade do LNS no gap é particularmente expressiva: enquanto a relaxação pura não conseguiu provar a otimalidade da instância input15 em 400s, o LNS resolveu-a completamente. O GRASP, por sua vez, não apenas falhou em provar otimalidade, como também obteve soluções de pior qualidade em duas instâncias (input5 e input35), conforme indicado pelos gaps de 1,01% e 2,21%.

4.3.3 Análise dos Nós Restantes

A Tabela 11 confirma a superioridade do LNS. O **LNS alcançou 0 nós restantes** no total, indicando que todas as instâncias foram resolvidas até a otimalidade dentro do limite de tempo. A relaxação deixou 64.364 nós (todos da instância input15), o GRASP deixou 36.303 nós (distribuídos entre input5, input15 e input35) e a GRASP|LNS deixou 2.324 nós (apenas input15). A Figura 9 apresenta a comparação.

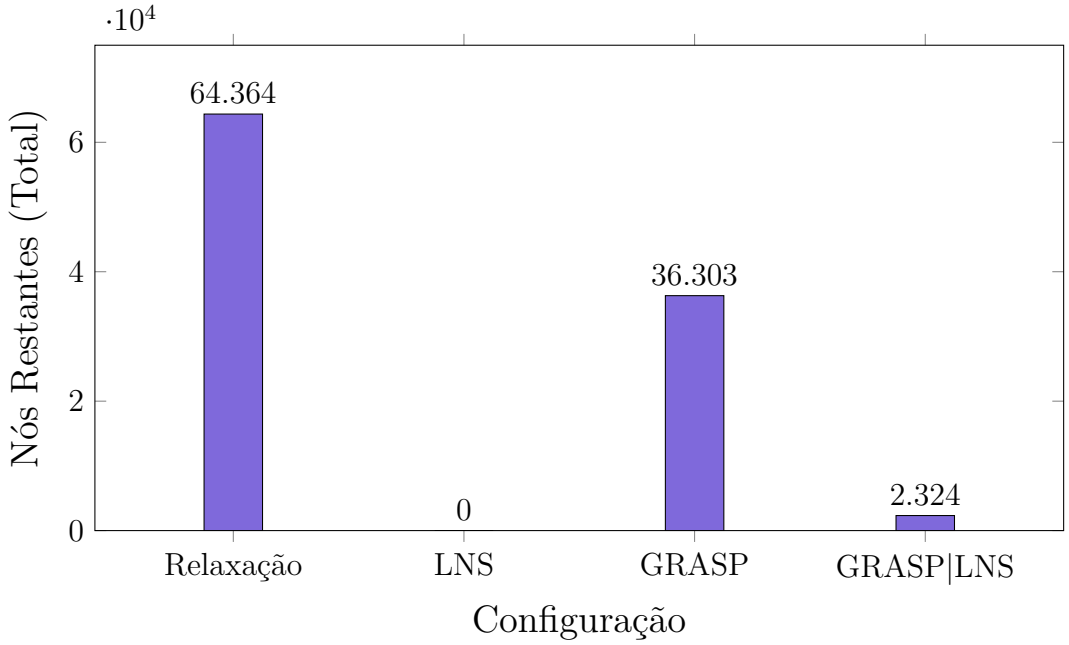


Figura 9 – Total de nós restantes na árvore de Branch-and-Bound para cada configuração. O LNS alcançou 0 nós restantes, resolvendo todas as instâncias até a otimalidade.

4.3.4 Discussão dos Resultados

Os resultados do comparativo evidenciam que a matheurística LNS proposta supera consistentemente tanto o Branch-and-Bound puro quanto o GRASP em todas as métricas avaliadas. A Tabela 14 sintetiza os principais indicadores.

Tabela 14 – Resumo comparativo das quatro configurações avaliadas.

Métrica	**Relaxação**	**LNS**	**GRASP**	**GRASP|LNS**
Tempo médio (s)	73,76	**22,44**	129,75	58,79
Tempo total (s)	744,13	**230,05**	1.314,27	594,92
Gap total (%)	0,73	**0,00**	3,46	0,24
Nós restantes (total)	64.364	**0**	36.303	2.324
Instâncias ótimas	9/10	**10/10**	7/10	9/10

A análise revela três observações principais:

1. **O LNS é o método mais eficaz.** Com a configuração calibrada (75% de destruição, ordenação crescente, 200s de sub-MIP), o LNS resolveu todas as instâncias até a otimalidade, com tempo médio 3,3 vezes menor que a relaxação e 5,8 vezes menor que o GRASP. A estratégia de intensificação via sub-MIP gera incumbentes de alta qualidade que viabilizam podas mais eficientes na árvore de busca.

2. **A combinação GRASP|LNS não supera o LNS isolado.** Embora a GRASP|LNS tenha apresentado desempenho superior à relaxação e ao GRASP, o *overhead* da heurística construtiva GRASP não trouxe benefícios adicionais quando o LNS já

estava ativo. Isso sugere que o LNS é capaz de encontrar incumbentes de qualidade suficiente sem a necessidade de uma solução inicial heurística dedicada.

3. **O GRASP isolado apresenta limitações para o DPD.** O GRASP atingiu o limite de tempo em 30% das instâncias e não conseguiu provar a otimalidade em 3 delas. A heurística construtiva, embora gere soluções viáveis rapidamente, não contribui de forma significativa para a redução do espaço de busca, e o *overhead* computacional de sua execução a cada nó penaliza o desempenho global.

5 Conclusão

5.1 Considerações Finais

5.2 Limitações do Estudo

5.3 Trabalhos Futuros

Referências

AHUJA, R. K. et al. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, v. 123, n. 1-3, p. 75–102, 2002.

AL-YAKOOB, S.; SHERALI, H. A column generation mathematical programming approach for a class-faculty assignment problem with preferences. *Computational Management Science*, v. 12, 2013. Citado na página 6.

AL-YAKOOB, S. M.; SHERALI, H. D. Mathematical programming models and algorithms for a class–faculty assignment problem. *European Journal of Operational Research*, v. 173, n. 2, p. 488–507, 2006. ISSN 0377-2217. Citado nas páginas 6 e 7. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221705002067.

AVELLA, P.; VASILYEV, I. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, v. 8, p. 497–514, 2005. Citado na página 6.

BALL, M. O. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, v. 16, n. 1, p. 21–38, 2011. ISSN 1876-7354. Disponível em: https://www.sciencedirect.com/science/article/pii/S1876735410000036.

BOSCHETTI, M. A.; LETCHFORD, A. N.; MANIEZZO, V. Matheuristics: survey and synthesis. *International Transactions in Operational Research*, v. 30, n. 6, p. 2840–2866, 2023.

CARTER, M. W.; LAPORTE, G. Recent developments in practical course timetabling. In: BURKE, E.; CARTER, M. (Ed.). *Practice and Theory of Automated Timetabling II*. [S.l.]: Springer Berlin Heidelberg, 1998. p. 3–19. ISBN 978-3-540-49803-2. Citado na página 6.

D'AMBROSIO, C. et al. The knapsack problem with forfeit sets. *Computers & Operations Research*, v. 151, p. 106093, 2023.

DOMENECH, B.; LUSA, A. A milp model for the teacher assignment problem considering teachers' preferences. *European Journal of Operational Research*, v. 249, n. 3, p. 1153–1160, 2016. ISSN 0377-2217. Citado na página 6. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221715008139.

GAMRATH, G. et al. *The SCIP Optimization Suite 7.0*. 2020. Technical report, Optimization Online. Citado na página 17. Disponível em: http://www.optimization-online.org/DB_HTML/2020/03/7705.html.

GUNAWAN, A.; NG, K.; ONG, H. A genetic algorithm for the teacher assignment problem for a university in indonesia. *International Journal of Information and Management Sciences*, v. 19, 2008. Citado na página 7.

HOSNY, M. *A heuristic algorithm for the faculty assignment problem*. 2012. Citado na página 6.

JHONATAN. *Uma heurística GRASP para o Problema da Distribuição de Disciplinas (DPD)*. 2025. Referência provisória (completar autores, título, veículo e ano conforme o artigo original).

JOVANOVIC, R.; VOSS, S. Matheuristic fixed set search applied to the multidimensional knapsack problem and the knapsack problem with forfeit sets. *OR Spectrum*, p. 1–37, 2024.

MANIEZZO, V.; STÜTZLE, T.; VOSS, S. *Matheuristics: Algorithms and Implementations*. [S.l.]: Springer, 2021.

PISINGER, D.; ROPKE, S. Large neighborhood search. In: *Handbook of Metaheuristics*. [S.l.]: Springer US, 2010. p. 399–419.

SILVA, L. S. *Uma abordagem exata para o problema da distribuição de disciplinas*. 2024. Trabalho de Conclusão de Curso, Universidade Federal de Mato Grosso do Sul (UFMS), Campo Grande, MS.

SILVA, P. P. A. de Paula e. *Uma Matheurística de Melhoria para o Problema da Mochila com Conjuntos de Penalidades*. 2024. Trabalho de Conclusão de Curso, Universidade Federal de Mato Grosso do Sul (UFMS), Campo Grande, MS.

SZWARC, E.; BACH-DÁBROWSKA, I.; BOCEWICZ, G. Competence management in teacher assignment planning. In: *24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018, Proceedings*. [S.l.]: Springer, 2018. p. 449–460. ISBN 978-3-319-99971-5. Citado na página 6.

VANDERBEI, R. J. *Linear Programming: Foundations and Extensions*. [S.l.]: Springer, 2014. Citado na página 9.

WOLSEY, L. A. *Integer Programming*. [S.l.]: John Wiley & Sons, Ltd., 2020. Citado na página 9.

Apêndices

APÊNDICE A – Códigos

A.1 Apêndices - Materiais do autor

Explicações