

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

ABORDAGENS MULTI-OBJETIVO PARA PROBLEMAS DE ALOCAÇÃO DE SALAS

Carlos Eduardo Gonzaga Romaniello de Souza

Orientador: Prof. Dr. TÚLIO ÂNGELO MACHADO TOFFOLO

Coorientador: Prof^a. Dr^a. RODRIGO PEDROSA SILVA

Relatório referente ao período de agosto de 2020 a agosto de 2021 apresentado à
Universidade Federal de Ouro Preto, como parte das exigências do Programa
Institucional de Bolsistas de Iniciação Científica – Edital XX/2020/PIBIC

Data: XX de fevereiro de 2021

Local: Ouro Preto – Minas Gerais – Brasil

Sumário

Resumo	3
1 Introdução	4
2 Objetivos	6
3 Problema abordado	7
3.1 Problema mono-objetivo	8
3.2 Problema multi-objetivo	9
4 Metodologia	11
4.1 Organização dos dados	11
4.2 Construção da solução inicial	11
4.3 Estruturas de vizinhança	12
4.3.1 Movimento trocar	13
4.3.2 Movimento deslocar	13
4.3.3 Movimento substituir	14
4.3.4 Movimento alocar	15
4.4 <i>Learning Automaton</i>	16
4.5 Abordagem mono-objetiva	17
4.5.1 <i>Lower bounds</i>	17
4.5.2 LAHC mono-objetivo	19
4.6 Abordagem multi-objetiva	20
4.6.1 LAHC multi-objetivo	20
4.6.2 <i>Non-dominated Sorting Genetic Algorithm II</i> (NSGA)	21
5 Experimentos computacionais	23
5.1 Descrição das instâncias	23
5.2 Problema mono-objetivo	23
5.2.1 <i>Lower bounds</i>	23
5.2.2 Solução inicial	24
5.2.3 Solução refinada	25
5.3 Problema multi-objetivo	29
5.3.1 LAHC multi-objetivo	29
5.3.2 NSGAI ^I I	33
5.3.3 NSGAI ^I I vc LAHC	37

6 Conclusões	41
Referências bibliográficas	42

Resumo

Abordagens multi-objetivo para problemas de alocação de salas

O presente trabalho tem como objetivo apresentar um algoritmo que tem como finalidade resolver o Problema de Alocação de Salas (PAS) em cursos universitários. Este problema consiste em alocar turmas de disciplinas, com horários predefinidos, em salas de aula distribuídas em vários prédios, considerando as determinações da universidade. Os algoritmos desenvolvidos nesse trabalho são: (i) um algoritmo construtivo guloso, (ii) a meta-heurística *Late Acceptance Hill-Climbing* (LAHC) utilizando uma busca local estocástica, e o algoritmo genético Non-dominated Sorting Genetic Algorithm II (NSGAII). A exploração do espaço de soluções é feita por meio de cinco estruturas de vizinhança. O algoritmo foi testado usando dados reais de alocações de salas de uma instituição de ensino superior e comparado com outro algoritmo meta-heurístico baseado em *Simulated Annealing* (SA). Este relatório disserta sobre os primeiros cinco meses do projeto, que demonstram dados suficientes para apresentar resultados preliminares promissores.

1 Introdução

Semestralmente, inúmeras instituições de ensino superior precisam realizar a alocação de turmas de disciplinas em salas de aula obedecendo a uma série de regras e restrições, por exemplo, que aulas da mesma disciplina ocorram na mesma sala durante a semana. Essa tarefa, devido a sua complexidade, é um desafio para as instituições, pois o elevado número de combinações possíveis para essas alocações inviabiliza uma solução manual, além disso normalmente uma solução manual não consegue contemplar todas as restrições impostas o que pode gerar insatisfação por parte dos professores e alunos.

A distribuição de aulas previamente definidas com horários estabelecidos, atentando-se a diversas particularidades relacionadas ao espaço físico, possibilidade de acesso, infraestrutura e recursos necessários são fatores que caracterizam o Problema de Alocação de Salas (PAS) (Schaerf, 1999).

Devido ao fato de o PAS ser um problema da classe NP-difícil (Even et al., 1976; Cartere Tovey, 1992), ele tem sido normalmente resolvido por meio de técnicas heurísticas, como as metaheurísticas. Sendo assim, vários trabalhos foram apresentados para a solução do PAS por meio dessas técnicas, como o Algoritmo Genético (Ueda et al., 2001), Busca Tabu (Subramanian et al., 2011), Simulated Annealing (Beyrouthy et al., 2006).

Neste projeto foi desenvolvido um algoritmo que combina uma fase de construção gulosa para gerar uma solução inicial e a meta-heurística *Late Acceptance Hill-Climbing* (LAHC) para refinamento. Utilizou-se, para fins de validação da solução, um ambiente real, caracterizado pelos ambientes com vários prédios da Universidade Federal de Ouro Preto (UFOP), abrangendo turmas distribuídas em dois prédios. Este projeto, parte do Programa Institucional de Voluntários de Iniciação Científica (PIVIC) da UFOP, foi interrompido em Julho/2020, após cinco meses de execução (de um total de doze meses inicialmente previstos). Neste período, foram avaliados algoritmos para resolução do problema mono-objetivo. A adaptação dos algoritmos para o problema multi-objetivo foi realizada durante a segunda metade do projeto.

Resultados preliminares demonstram que o algoritmo implementado é promissor. As soluções obtidas pelo LAHC foram comparadas com soluções obtidas por outra metaheurística, *Simulated Annealing*, implementada como parte do projeto “Abordagens heurísticas para problemas de alocação de salas” pelo aluno Vinícius Gabriel Angelozzi Verona de Resende, que gentilmente nos cedeu os resultados obtidos em seu trabalho.

O restante deste relatório está organizado da seguinte forma: Os objetivos do projeto são listados na Seção 2. Na Seção 3 é apresentada a descrição completa do problema abordado. Na Seção 4 o algoritmo desenvolvido é apresentado em detalhes. Na Seção 5

são apresentados os resultados preliminares obtidos durante o projeto. Por fim, a Seção 6 conclui o relatório, discutindo brevemente os resultados preliminares e trabalhos futuros.

2 Objetivos

O objetivo geral deste projeto é a aplicação de métodos de otimização para abordar Problemas de Alocação de Salas, em particular o problema enfrentado pelo ICEB/UFOP. Para atingir este objetivo, os seguintes objetivos específicos foram elencados:

1. Fazer uma revisão da literatura sobre os métodos utilizados para resolver o problema abordado.
2. Modificar, adaptar e aperfeiçoar algoritmos para os problemas abordados.
3. Avaliar o(s) algoritmo(s) desenvolvido(s).
4. Contribuir com a divulgação de técnicas de otimização mono- e multi-objetivo aplicadas à resolução do problema.
5. Contribuir com a formação de recursos humanos especializados nessa área do conhecimento.
6. Contribuir para a consolidação do grupo de pesquisa GOAL (Grupo de Otimização e Algoritmos) da Universidade Federal de Ouro Preto.

3 Problema abordado

O presente trabalho aborda o Problema de Alocação de Salas (PAS) do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). Este problema foi tratado em duas versões, a mono-objetivo e o multi-objetivo, que divergem apenas na forma como as soluções são avaliadas. O ICEB oferece vários cursos de graduação e pós-graduação para seus alunos, além de ofertar aulas do currículo de outros cursos da instituição. As aulas ocorrem em 36 salas divididas em dois prédios: o próprio ICEB e o Pavilhão Central de Aulas (PCA). Estas aulas ocorrem em 16 horários diários divididos em três turnos (manhã, tarde e noite). Cada aula, que doravante denominaremos *encontro* por razões de generalidade, consiste de uma turma (conjunto de alunos), professor e disciplina, e possui ainda um conjunto de dias e horários pré-definidos. Assim, o problema consiste em alocar os encontros às salas disponíveis, considerando situações como:

- **Pré-requisitos dos encontros:** podem ter um conjunto de pré-requisitos que a sala deve atender, tais como necessidade de projetor, quadro negro, e/ou outros equipamentos específicos;
- **Preferências** de docentes e discentes por uma determinada sala, que podem acontecer por motivos diversos. A título de exemplo, citamos que alguns docentes têm alergia a giz, e que alguns discentes possuem capacidade limitada de locomoção, preferindo portanto salas no andar térreo.
- **Capacidade das salas:** diferentes encontros podem ter número diferente de alunos, e a demanda das aulas devem ser atendidas sempre que possível.

Há ainda restrições mais óbvias, como: (i) cada sala pode alocar apenas um encontro em um período de tempo, e (ii) cada encontro somente pode ser alocado a uma sala em cada período de tempo.

A solução gerada por meio dos algoritmos propostos neste trabalho tiveram a finalidade de melhorar a atual utilização das salas pela UFOP. Para tal, foram levadas em consideração diversas características que tem como objetivo criar uma solução mais adequada para alunos e professores. São elas:

- Evitar alocar encontros de pequena demanda em salas com alta capacidade;
- Tentar alocar o máximo de encontros possíveis;
- Alocar, preferencialmente, encontros de uma turma usual na mesma sala;

- Evitar alocar encontros em salas que possuem capacidade menor que a demanda;
- Respeitar as reservas de salas feitas previamente;
- Tentar satisfazer as preferências sugeridas;
- Evitar que professores troquem de sala ou prédio em horários consecutivos.

3.1 Problema mono-objetivo

As características citadas acima definem os objetivos do problema. Assim, a qualidade de uma solução mono-objetiva do problema abordado é avaliada por meio de uma combinação linear dos vários objetivos que compõem o problema. Nesta combinação linear, pesos são especificados como mecanismo para definir a importância relativa de cada objetivo. Ao todo, sete objetivos são considerados:

1. minimizar a capacidade ociosa total;
2. minimizar o não atendimento de demandas;
3. minimizar alocações de salas a encontros que não tem capacidade suficiente para atender à demanda, deixando de satisfazer mais de 10% da demanda;
4. minimizar a alocação de salas diferentes para um encontro, ou seja, evitar a alocação de mais de uma sala para uma mesma disciplina/professor durante a semana;
5. minimizar o não atendimento às solicitações de docentes e discentes;
6. minimizar o deslocamento de professores durante o dia;
7. minimizar o deslocamento dos professores entre salas em horários consecutivos;
8. minimizar o deslocamento dos professores entre prédios em horários consecutivos.

A cada objetivo g é associado o peso ω_g . Os seguintes pesos foram utilizados: $\omega_1 = 1, \omega_2 = 100, \omega_3 = 1000, \omega_4 = 10, \omega_5 = 3000$. Note que o objetivo 3 tem a maior prioridade.

Para introduzir a função objetivo do problema, a seguinte notação é utilizada:

- S : conjunto de salas;
- S_E : conjunto de encontros (aulas);
- S_P : conjunto de preferências;
- S_{Pr} : conjunto de professores;

- o_i : capacidade ociosa da sala i considerando a alocação atual, $o_i \geq 0$. Considera-se capacidade ociosa apenas se a demanda de vagas é inferior a 50% da capacidade da sala;
- d_j : demanda não atendida do encontro j , $d_j \geq 0$;
- d_j^- : demanda não atendida do encontro j que respeita 10% da capacidade da sala, $d_j^- \geq 0$;
- d_j^+ : demanda não atendida do encontro j que ultrapassa 10% da capacidade da sala, $d_j^+ \geq 0$;
- h_j : quantidade de horários do encontro j , para $h_j \geq 0$;
- d_j^s : demanda do encontro j alocada a uma sala diferente da sala mais frequente da turma pertencente, para $d_j^s > 0$;
- e_ℓ : números de exigências da preferência ℓ não atendidas, para $e_i \geq 0$;
- ts_i : quantidade de vezes que o professor i troca de salas em horários consecutivos;
- tp_i : quantidade de vezes que o professor i troca de prédios em horários consecutivos.
- $\omega_1 = 1, \omega_2 = 1000, \omega_3 = 100, \omega_4 = 10, \omega_5 = 3000$

A Equação (3.1) apresenta a função objetivo considerada inicialmente para o problema. Note que trata-se de um problema de minimização, em que uma combinação linear dos cinco objetivos elencados anteriormente é utilizada como única função de avaliação:

$$\begin{aligned} & \omega_1 \sum_{i \in S} o_i + \omega_2 \sum_{j \in S_E} d_j h_j + \omega_2 \sum_{j \in S_E} d_j^+ h_j + \omega_3 \sum_{j \in S_E} d_j^- h_j \\ & + \omega_4 \sum_{j \in S_E} d_j^s + \omega_5 \sum_{i \in S_P} e_\ell h_j + \omega_2 \sum_{i \in S_{P_r}} ts_i + \omega_2 \sum_{i \in S_{P_r}} tp_i \end{aligned} \quad (3.1)$$

3.2 Problema multi-objetivo

Esta versão do problema busca melhorar a solução para cada objetivo separadamente e não agrupando-os em uma única função objetivo. As duas versões compartilham as mesmas características, porém no multi-objetivo a heurística do programa (LAHC) foi ajustada para buscar soluções não dominadas no espaço de soluções. Uma explicação sobre as soluções não dominadas pode ser observada pela imagem a baixo.

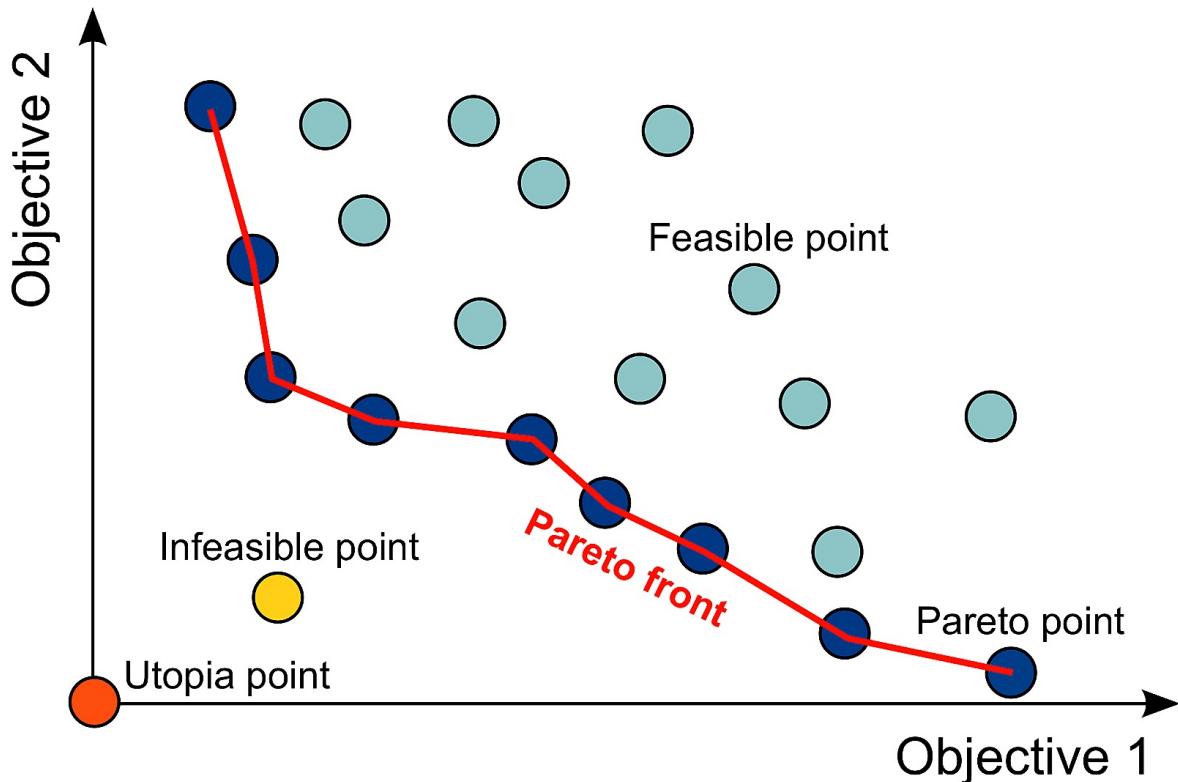


Figura 3.1: Exemplo de soluções não dominadas

Observando a imagem que trata de um problema com apenas dois objetivos, pode-se perceber que há um conjunto de soluções que formam uma barreira na parte inferior (pontos cortados pela linha vermelha). Esses pontos são as soluções não dominadas, ou seja, não existem outras soluções que são melhores que elas em todos os objetivos.

A versão multi-objetiva do PAS é uma adaptação do problema retratado na imagem capaz de abranger todos os objetivos citados anteriormente. Com isso, o programa desenvolvido nessa pesquisa busca encontrar esse conjunto de soluções não dominadas para o problema.

4 Metodologia

Com o objetivo de gerar soluções de alta qualidade para os problemas abordado, três algoritmos foram implementados: (i) um algoritmo construtivo guloso para geração da solução inicial, a (ii) a meta-heurística *Late Acceptance Hill Climbing* (LAHC), e o (iii) *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) utilizando cinco diferentes estruturas de vizinhança. Esta seção detalha os procedimentos implementados, apresentando inicialmente as estruturas de dados utilizadas para representação dos dados de entrada e de uma solução do problema. Em seguida, os dois algoritmos implementados são apresentados.

4.1 Organização dos dados

Os dados do problema foram fornecidos através de um arquivo no formato *JavaScript Object Notation* (JSON) que contém informações como salas, horários, encontros, turmas, entre outros. Os campos mais utilizados no algoritmo foram os citados acima, todos separados em estruturas de dados para tornar seu manuseio mais fácil.

Para melhor visualização do problema, os encontros foram separados por dia, de segunda a sábado, sendo que cada dia também contém uma matriz de tamanho *quantidade de horários* \times *quantidade de salas*, que armazena as informações do encontro alocado naquela célula e o status da célula (alocado, não alocado ou reservado). Há também a estrutura `alocacao` que contém todas as informações de uma turma e todos os encontros pertencentes a ela junto com a sala na qual cada um está alocado.

4.2 Construção da solução inicial

A solução inicial é produzida por um algoritmo guloso que opera da seguinte forma: inicialmente tenta-se alocar os encontros que possuem alguma preferência na primeira sala disponível encontrada. Logo após, a estrutura `alocacao` é ordenada de forma decrescente em relação à demanda de cada turma e a estrutura `sala` é ordenada também de forma decrescente em relação às capacidades. Após isso, todo o vetor de `alocacao` é percorrido e o algoritmo tenta alocar os encontros de cada turma seguindo os passos: verifica-se se algum encontro da mesma turma já está alocado; em caso afirmativo, tenta-se alocar esse encontro na mesma sala, caso contrário, o vetor de salas ordenado é percorrido tentando alocar esse encontro. Caso não seja possível alocar o encontro, o algoritmo passa para o próximo encontro. Vale ressaltar que para a solução inicial, alocações inviáveis podem

acontecer. O Algoritmo 1 apresenta o pseudocódigo do método implementado.

Algoritmo 1: *Algoritmo guloso*

Entrada: Matriz de alocações M , vetor de turmas T ordenado de acordo com a demanda e vetor de salas S ordenado de acordo com a capacidade

GREEDY(M, T, S)

```
1    $E_p \leftarrow$  encontros em  $T$  que possuem preferências
2   para cada  $e \in E_p$  faça
3       para cada  $s \in S$  faça
4           se  $s$  está disponível no dia/horário de  $e$  então
5                $M \leftarrow$  alocação de  $e$  em  $s$ 
6               sair do loop
7   para cada  $t \in T$  tal que  $t \notin E_p$  faça
8        $D \leftarrow$  demanda de  $t$ 
9       se  $t$  possui algum encontro e  $D > 0$  então
10       $E_t \leftarrow$  encontros em  $t$ 
11       $H \leftarrow$  salas utilizadas pelos encontros da turma  $t$ 
12      para cada  $e \in E_t$  faça
13          se  $H \neq \emptyset$  então
14              para cada sala  $s \in H$  faça
15                  se  $s$  está disponível no dia/horário de  $e$  então
16                       $M \leftarrow$  alocação de  $e$  em  $s$ 
17                      sair do loop
18          se  $H = \emptyset$  ou  $e$  não foi alocado em alguma sala  $s \in H$  então
19              para cada  $s \in S$  faça
20                  se  $s$  está disponível no dia/horário de  $e$  então
21                       $M \leftarrow$  alocação de  $e$  em  $s$ 
22                      sair do loop
23   retorne  $M$ 
```

4.3 Estruturas de vizinhança

Para percorrer o espaço de soluções foram implementados 4 movimentos que são utilizados por todos os algoritmos desse projeto.

4.3.1 Movimento trocar

Para o movimento trocar, o algoritmo procura aleatoriamente um segundo encontro da mesma forma que o primeiro para tentar fazer uma troca, como mostrado na Figura 4.1. O segundo encontro precisa obedecer alguns critérios:

- ser no mesmo dia que o primeiro encontro,
- estar alocado,
- ter os mesmos horários de aula que o primeiro encontro.

Ao localizar algum encontro que respeita esses critérios o algoritmo calcula o custo para realizar a movimentação de ambos e aplica essa diferença no custo da solução atual. Caso o custo seja menor que o item da lista a alteração é aceita atualizando a solução e o custo. Caso contrário a alteração é descartada e o loop passa para a próxima iteração.



SEGUNDA				
	Sala 1	Sala 2	Sala 3	
Antes	8:00 - 9:00	Encontro 1		Encontro 4
	9:00 - 10:00			
	10:00-11:00			
	11:00 - 12:00		Encontro 3	
	12:00 - 13:00	Encontro 2		
	13:00 - 14:00			

SEGUNDA				
	Sala 1	Sala 2	Sala 3	
Depois	8:00 - 9:00	Encontro 4		Encontro 1
	9:00 - 10:00			
	10:00-11:00			
	11:00 - 12:00		Encontro 3	
	12:00 - 13:00	Encontro 2		
	13:00 - 14:00			

Figura 4.1: Exemplo do movimento Trocar

4.3.2 Movimento deslocar

Para o movimento deslocar, o algoritmo procura aleatoriamente uma sala para tentar deslocar o primeiro encontro, como mostrado na Figura 4.2. A sala precisa obedecer alguns critérios:

- estar situada no mesmo dia que o primeiro encontro.

- não possuir nenhum encontro alocado.
- não estar reservada.

Ao encontrar alguma sala que respeite esses critérios o algoritmo calcula o custo para realizar o deslocamento e aplica essa diferença no custo da solução atual. Caso o custo seja menor que o item da lista, a alteração é aceita atualizando a solução e o custo. Caso contrário, a alteração é descartada e o loop passa para a próxima iteração.

Antes
→
Depois

SEGUNDA			
	Sala 1	Sala 2	Sala 3
8:00 - 9:00			
9:00 - 10:00	Encontro 1		Encontro 4
10:00-11:00			
11:00 - 12:00		Encontro 3	
12:00 - 13:00	Encontro 2		
13:00 - 14:00			

SEGUNDA			
	Sala 1	Sala 2	Sala 3
8:00 - 9:00	Encontro 1		Encontro 4
9:00 - 10:00			
10:00-11:00		Encontro 3	
11:00 - 12:00			
12:00 - 13:00			Encontro 2
13:00 - 14:00			

Figura 4.2: Exemplo do movimento Deslocar

4.3.3 Movimento substituir

Para o movimento substituir, o algoritmo procura aleatoriamente um encontro para tentar substituir o primeiro encontro, como mostrado na Figura 4.3. O encontro precisa obedecer alguns critérios:

- ser do mesmo dia que o primeiro encontro.
- não estar alocado alocado.
- ter os mesmos horários de aula que o primeiro.

Para o primeiro encontro que respeita esses critérios o algoritmo calcula o custo para realizar a substituição e aplica essa diferença no custo da solução atual. Caso o custo seja menor que o item da lista a alteração é aceita atualizando a solução e o custo. Caso contrário, a alteração é descartada e o loop passa para a próxima iteração.



Antes

SEGUNDA			
	Sala 1	Sala 2	Sala 3
8:00 - 9:00			
9:00 - 10:00	Encontro 1		
10:00-11:00			
11:00 - 12:00		Encontro 3	
12:00 - 13:00			
13:00 - 14:00	Encontro 2		

Depois

SEGUNDA			
	Sala 1	Sala 2	Sala 3
8:00 - 9:00			
9:00 - 10:00	Encontro 5		
10:00-11:00			
11:00 - 12:00		Encontro 3	
12:00 - 13:00			
13:00 - 14:00	Encontro 2		

Figura 4.3: Exemplo do movimento Substituir

4.3.4 Movimento alocar

Para o movimento alocar, o primeiro encontro não é utilizado já que é necessário um encontro que não esteja alocado para completar o movimento, como mostrado na Figura 4.4. Sendo assim, o algoritmo procura um encontro e uma sala aleatória que devem obedecer esses critérios:

- o encontro deve estar desalocado.
- a sala deve estar vazia.
- os dois devem pertencer ao mesmo dia.

Para o encontro e a sala que respeitem esses critérios o algoritmo calcula o custo para realizar a alocação e aplica essa diferença no custo da solução atual. Caso o custo seja menor que o item da lista, a alteração é aceita atualizando a solução e o custo. Caso contrário, a alteração é descartada e o loop passa para a próxima iteração.

The diagram illustrates a movement labeled 'Alocar' (Allocate) with a circular arrow pointing from 'Antes' (Before) to 'Depois' (After). Below the arrows are two tables representing room schedules for 'SEGUNDA' (Monday).

Antes (Before):

	Sala 1	Sala 2	Sala 3
8:00 - 9:00	Encontro 1		Encontro 4
9:00 - 10:00			
10:00-11:00		Encontro 3	
11:00 - 12:00			
12:00 - 13:00	Encontro 2		
13:00 - 14:00			

Depois (After):

	Sala 1	Sala 2	Sala 3
8:00 - 9:00	Encontro 1		Encontro 4
9:00 - 10:00			
10:00-11:00		Encontro 3	
11:00 - 12:00			
12:00 - 13:00	Encontro 2		Encontro 6
13:00 - 14:00			

Figura 4.4: Exemplo do movimento Alocar

4.4 *Learning Automaton*

As probabilidades de seleção dos diferentes movimentos são atualizadas a cada it iterações por meio do algoritmo *Learning Automaton* (Narendra e Thathachar, 1989). Trata-se de um algoritmo simples de aprendizado de máquina que mantém e atualiza uma distribuição de probabilidades γ de todas as ações possíveis.

No algoritmo implementado, cada vizinhança k tem probabilidade $\gamma_{k,e}$ de ser selecionada na iteração e . O valor de $\gamma_{k,e}$ é atualizado de acordo com o resultado da iteração anterior, $e - 1$. A probabilidade do movimento escolhido na iteração e é atualizado de acordo com a Equação (4.1), enquanto as probabilidades dos demais movimentos são atualizadas de acordo com a Equação (4.2). Nestas equações, $|N|$ é o número de movimentos, α é o coeficiente de variação aplicado em cada iteração, e β_e indica se houve melhora, ou seja, assume valor 1 caso o movimento escolhido na iteração e tenha resultado em melhora e valor 0 caso contrário.

$$\gamma_{k,e+1} = \gamma_{k,e} + \alpha\beta_e(1 - \gamma_{k,e}) - \alpha(1 - \beta_e)\gamma_{k,e} \quad (4.1)$$

$$\gamma_{k,e+1} = \gamma_{k,e} - \alpha\beta_e\gamma_{k,e} + \alpha(1 - \beta_e) \left(\frac{1}{|N| - 1} - \gamma_{k,e} \right) \quad (4.2)$$

Esta abordagem foi bem sucedida e auxiliou o algoritmo desenvolvido por Toffolo et al.

(2018) a vencer a primeira competição internacional de roteamento de veículos organizada pela *VeRoLog*¹ (*First VeRoLog International Vehicle Routing Challenge*).

4.5 Abordagem mono-objetiva

4.5.1 *Lower bounds*

Para assegurar a qualidade da solução encontrada pelo LAHC, foi calculado o valor de uma solução hipotética para o problema. Nela algumas restrições são ignoradas para que se possa gerar o *lowerbound*, um limite inferior para o resultado onde é impossível encontrar um valor menor que ele.

Para isso foi utilizado uma biblioteca implementada em **Python** chamada **Python-MIP**². Nela está implementado um algoritmo de modelagem e resolução de Programas Lineares Inteiros Mistos. Para o problema em questão foi utilizado o seguinte modelo matemático:

¹ Working Group on Vehicle Routing and Logistics Optimization within EURO, the Association of the European Operational Research Societies.

²Mais informações sobre em <https://www.python-mip.com>

$$\min \sum_{e \in E} \sum_{s \in S'} \sum_{h \in H_e} C_{es} * X_{esh} \quad (4.3)$$

Restrições:

$$\sum_{e \in E} X_{esh} = 1, \forall s \in S', \forall h \in H_e \quad (4.4)$$

$$\sum_{s \in S} X_{esh} \leq 1, \forall h \in H, \forall e \in E \quad (4.5)$$

$$X_{esh} = X_{esh'}, \forall e \in E, \forall s \in S', \forall (h, h') \in G_e \quad (4.6)$$

Onde:

- E : conjunto de encontros;
- H : conjunto de horários;
- H_e : conjunto de horários de um encontro específico;
- S : conjunto de salas disponíveis;
- S' : conjunto de salas disponíveis incluindo uma sala virtual para os desalocados;
- C_{es} : matriz bidimensional com o custo de alocação de cada encontro e em cada sala s respeitando a equação:

$$(\omega_1 \times o_{es}) + (\omega_2 \times d_e) + (\omega_2 \times d_{es}^+) + (\omega_3 \times d_{es}^-) + \sum_{p \in P_e} (\omega_4 \times p) \quad (4.7)$$

onde:

- o_{es} : capacidade ociosa do encontro e na sala s ;
- d_e : demanda não atendida do encontro e ;
- d_{es}^+ : demanda não atendida do encontro e que ultrapassa 10% da capacidade da sala s ;
- d_{es}^- : demanda não atendida do encontro e que não ultrapassa 10% da capacidade da sala s ;
- P_e : conjunto de preferências não atendidas do encontro e ;
- $\omega_1 = 1, \omega_2 = 1000, \omega_3 = 100, \omega_4 = 3000$.
- X_{esh} : matriz binária tridimensional que assume o valor 0 quando o encontro e , na sala s , no horário h não está alocado e 1 quando está alocado;

- G_e : conjunto dos horários do encontro e agrupados dois a dois (h, h').

Com esse modelo matemático foi possível calcular o valor do *lowerbound*. Como dito anteriormente, algumas restrições do problema original foram ignoradas para poder gerar uma solução relaxada que garante a impossibilidade de obter um valor menor do que o *lowerbound*. Porém, algumas foram mantidas, como a restrição representada na equação 4.4 que garante que todo encontro deve ser alocado, a restrição da equação 4.5 que limita todas as salas a terem no máximo um encontro alocado por horário e a restrição da equação 4.6 que assegura que todos os encontros com horários geminados sejam alocados na mesma sala.

4.5.2 LAHC mono-objetivo

Nessa fase do problema ocorre o refinamento da solução inicial gerada pelo algoritmo guloso por meio da heurística LAHC. Inicialmente, é gerada uma lista com 1000 posições, onde cada posição possui o *custo da solução inicial* $\times 1, 1$, e também uma cópia da solução, da estrutura `alocacao` e da estrutura `sub_encontros`. Logo após, o algoritmo entra em um laço de repetição que possui como critério de parada o tempo total de processamento.

No laço de repetição são sorteados um dia, uma sala, um horário e um movimento aleatório. O dia, a sala e o horário servem para buscar de forma aleatória um encontro já alocado na matriz do dia para que se possa realizar o movimento. Essa busca é feita para todos os movimentos com exceção do movimento *substituir 3* onde isso é feito 3 vezes. Para cada movimento o procedimento é diferente.

Para percorrer o espaço de soluções foram utilizados os movimentos citados na seção 4.3 que são sorteados aleatoriamente respeitando uma probabilidade pré estabelecida. Em seguida, esta probabilidade é atualizada por um algoritmo de aprendizado de máquinas denominado *Learning Automaton* (vide Seção 4.4). Assim, após a realização de cada movimento, é verificado o sucesso ou não do movimento e o *Learning Automaton* (LA) é aplicado. Dado um vetor com probabilidades, o LA é capaz de aumentar uma probabilidade e diminuir as outras proporcionalmente ou diminuir uma probabilidade e aumentar as outras proporcionalmente. Caso o movimento melhore a solução o LA é aplicado para aumentar a probabilidade de que o movimento seja escolhido e caso contrário ele é aplicado para diminuir essa probabilidade. O Algoritmo ?? apresenta o pseudocódigo do método implementado.

Algoritmo 2: Late Acceptance Hill-Climbing mono-objetivo

Entrada: Solução inicial S , tamanho da lista ℓ e probabilidades γ

LAHC(S, ℓ, γ)

```
1  para cada  $p \in \{0, \dots, \ell - 1\}$  faça
2     $F_p \leftarrow f(S)$ 
3     $S^* \leftarrow S$ 
4     $p \leftarrow r \leftarrow 0$ 
5  enquanto tempo limite não for alcançado faça
6     $N \leftarrow$  selecione um movimento considerando as probabilidades  $\gamma$ 
7     $S' \leftarrow$  vizinho aleatório  $N(S)$ 
8    se  $f(S') \leq f(S)$  ou  $f(S') \leq F_p$  então
9      se  $f(S') < f(S)$  então
10         $r \leftarrow 0$ 
11         $S \leftarrow S'$ 
12        se  $f(S) < f(S^*)$  então
13           $S^* \leftarrow S$ 
14     $\gamma \leftarrow$  probabilidades atualizadas pelo Learning Automaton
15     $F_p \leftarrow f(S)$ 
16     $p \leftarrow (p + 1) \bmod l$ 
17     $r \leftarrow r + 1$ 
18  retornne  $S^*$ 
```

4.6 Abordagem multi-objetiva

4.6.1 LAHC multi-objetivo

O LAHC na versão multi-objetiva possui a mesma lógica de funcionamento da sua versão mono-objetiva. A única divergência entre as duas é na aceitação e armazenamento de soluções, ele possui uma lista que contém todas as soluções não dominadas encontradas

durante sua execução. A diferença pode ser percebida no pseudocódigo a baixo:

Algoritmo 3: *Late Acceptance Hill-Climbing multi-objetivo*

Entrada: Solução inicial S , tamanho da lista ℓ e probabilidades γ , [lista de soluções não dominadas \$N_d\$](#)

LAHC(S, ℓ, γ, N_d)

```

1   para cada  $p \in \{0, \dots, \ell - 1\}$  faça
2      $F_p \leftarrow f(S)$ 
3    $S^* \leftarrow S$ 
4    $p \leftarrow r \leftarrow 0$ 
5   enquanto tempo limite não for alcançado faça
6      $N \leftarrow$  selecione um movimento considerando as probabilidades  $\gamma$ 
7      $S' \leftarrow$  vizinho aleatório  $N(S)$ 
8     se  $f(S') \leq f(S)$  ou  $f(S') \leq F_p$  então
9       se  $f(S') < f(S)$  então
10         $r \leftarrow 0$ 
11         $S \leftarrow S'$ 
12        se  $f(S) < f(S^*)$  então
13           $S^* \leftarrow S$ 
14        se  $S'$  não é dominada por nenhuma  $S \in N_d$  então
15           $N_d \leftarrow S'$ 
16        se alguma  $S \in N_d$  é dominada por  $f(S')$  então
17          Remove  $S$  de  $N_d$ 
18       $\gamma \leftarrow$  probabilidades atualizadas pelo Learning Automaton
19       $F_p \leftarrow f(S)$ 
20       $p \leftarrow (p + 1) \bmod l$ 
21       $r \leftarrow r + 1$ 
22  retorno  $N_d$ 

```

4.6.2 Non-dominated Sorting Genetic Algorithm II (NSGAII)

O NSGAII é um algoritmo genético utilizado para resolver problemas de otimização multi-objetivo. Nesse projeto ele foi implementado para realizar uma última verificação no conjunto de soluções encontradas anteriormente pelo algoritmo 3.

Dado uma população inicial que, nesse caso, é a lista de soluções não dominadas retornada pelo LAHC multi-objetivo, o NSGAII realiza cruzamentos e mutações entre a população inicial e então realiza uma ordenação de não dominância entre as soluções e seleciona as melhores para a próxima iteração. Encontra-se abaixo o pseudocódigo do

algoritmo implementado

Algoritmo 4: *NSGAII*

Entrada: População inicial P

NSGA(P)

```
1   enquanto quantidade de iterações não atingida faça
2       para cada par aleatório  $(S_1, S_2) \in P$  faça
3            $F \leftarrow$  cruzamento( $S_1, S_2$ )
4           se condição atendida então
5                $F \leftarrow$  mutacao( $F$ )
6            $P \leftarrow F$ 
7        $P \leftarrow$  selecaoElitista( $P$ )
8   retorne  $P$ 
```

Gráficos contendo os testes estarão na seção de resultados/experimentos

5 Experimentos computacionais

Os algoritmos descritos na seção anterior foram implementados na linguagem de programação **Julia**¹ versão 1.6.1. As instâncias de teste foram geradas a partir das demandas do Instituto de Ciências Exatas e Biológicas (ICEB) e Pavilhão Central de Aulas (PCA) da UFOP. Ao todo foram geradas cinco instâncias considerando as demandas de 2017/01, 2017/02, 2018/01, 2018/02, 2019/01, 2020/01.

O algoritmo foi executado 5 vezes para cada instância considerando o tempo limite de 300 segundos em um computador com processador Intel i7-9750H 2.6Ghz e 16GB de memória RAM. Dada a característica estocástica do algoritmo, cada execução utilizou uma semente diferente para geração de números pseudo-aleatórios.

Vale ressaltar que, para a instância do primeiro semestre do ano de 2020, foram utilizados dados reais utilizados pela UFOP durante o respectivo período letivo, o que proporcionou a validação da solução final do algoritmo desenvolvido nesse projeto.

5.1 Descrição das instâncias

Para executar os experimentos desse trabalho foram utilizadas instâncias reais referentes aos dados do ICEB (Instituto de Ciências Exatas e Biológicas) da UFOP (Universidade Federal de Ouro Preto). Essa instância é um arquivo JSON que separa em campos informações importantes para o problema como: salas disponíveis, horário de funcionamento dos prédios, os encontros a serem alocados e os professores da universidade.

Além da instância principal, também foi fornecido um outro arquivo JSON que contém regras para pré alocação de salas e regras para agrupamento de turmas.

5.2 Problema mono-objetivo

Nas seções seguintes serão apresentadas uma revisão dos métodos utilizados e uma visualização dos resultados obtidos para a abordagem mono-objetiva do problema. Nessa versão todos os objetivos são agrupados em uma única função objetivo que mede a qualidade das soluções.

5.2.1 *Lower bounds*

Como dito na seção 4.5.1, a biblioteca **Python-MIP** foi utilizada para calcular uma solução hipotética para o problema. Essa solução ignora algumas restrições da função

¹Mais informações sobre a linguagem em <https://julialang.org>

objetivo (função 3.1) para torná-la um limite inferior para o problema, o *Lower Bound*.

Esses valores foram calculados para todas as intâncias e utilizados como parâmetro de qualidade da solução para o problema mono-objetivo. Os resultados obtidos foram os seguintes:

Instância	Lowerbound	Distância média em relação a melhor solução
2017-01	234.794	0,5468%
2017-02	295.846	12,5468%
2018-01	569.736	7,3606%
2018-02	550.991	2,3133%
2019-01	431.665	3,4705%
2019-02	575.167	0,2358%
2020-01	459.918	0,2749%

Tabela 5.1: Resultados Lowerbound

Os melhores resultados serão apresentados na seção 5.2.3. A imagem abaixo representa os dados da tabela 5.2.1.

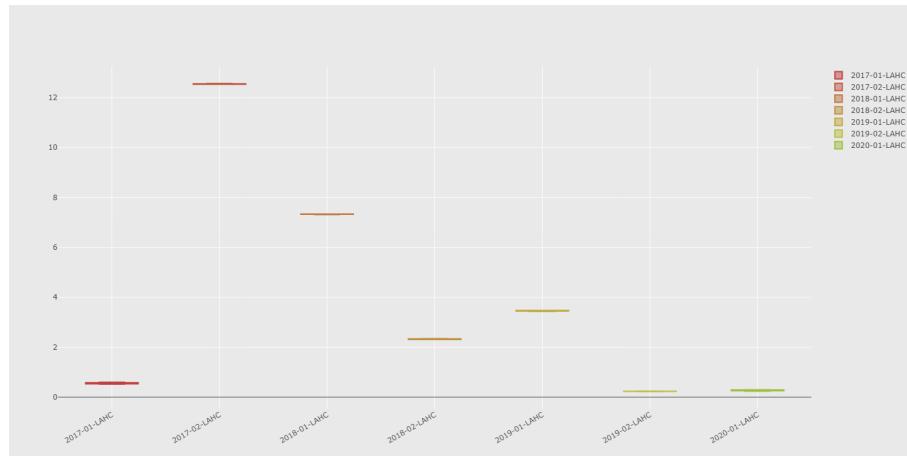


Figura 5.1: Boxplot das soluções

5.2.2 Solução inicial

Dado os valores dos *Lower Bounds*, o próximo passo é encontrar uma solução inicial para o algoritmo heurístico. Para isso, foi utilizado o algoritmo 1 citado na seção 4.2. Note que, uma vez que o algoritmo não possui nenhuma componente aleatória, ele tem natureza determinística e sempre gera a mesma solução para uma dada instância. Assim, o resultado independe da semente de números aleatórios. Os valores obtidos são os seguintes:

Instância	Solução Inicial
2017-01	268.805
2017-02	366.583
2018-01	660.871
2018-02	606.752
2019-01	487.706
2019-02	612.653
2020-01	498.198

Tabela 5.2: Resultados do algoritmo guloso

5.2.3 Solução refinada

Após a geração da solução inicial, inicia-se a fase de refinamento que utiliza o algoritmo heurístico LAHC (algoritmo 3). Ao final, a solução é melhorada ainda mais, porém dessa vez os custos alcançados variam dependendo da semente utilizada. A Tabela 5.3 summariza os resultados obtidos pelo algoritmo. Para cada instância são apresentados: (i) o valor médio da função objetivo, (ii) a melhor solução encontrada, e (iii) o desvio padrão dos resultados, na coluna “Desv.P.”.

Instância	Algoritmo LAHC		
	Média	Melhor	Desv.P.
2017-01	236.109,0	236.019	40,2492
2017-02	338.294,0	338.240	24,1495
2018-01	614.794,4	614.676	52,9500
2018-02	564.104,6	564.031	32,9149
2019-01	447.147,8	447.015	59,3899
2019-02	576.529,4	576.471	26,1172
2020-01	461.185,0	461.022	72,8958

Tabela 5.3: Resultados obtidos pelos algoritmos LAHC mono-objetivo

Observa-se pela Tabela 5.3 que o LAHC mono-objetivo proposto e implementado neste projeto resultou em soluções de melhor qualidade. Além disso, destaca-se ainda a robustez do algoritmo, com menor variação entre o valor da função objetivo obtido em diferentes execuções.

As tabelas 5.2 e 5.3 permitem concluir, também, que a fase de refinamento foi capaz de melhorar a qualidade das soluções produzidas pelo algoritmo construtivo. Por tentar realizar sempre a melhor alocação local possível, o algoritmo guloso produziu soluções de qualidade razoável para o problema. Na fase heurística, o algoritmo conseguiu melhorar a solução inicial de todas as instâncias. Visando explicitar a evolução do valor da

função objetivo e o desempenho do algoritmo de refinamento (LAHC), as figuras 5.2 a 5.8 apresentam a variação do *GAP* ao longo da execução do algoritmo. Note que o *GAP* é calculado como $GAP = \frac{f(S) - f(BKS)}{f(S)}$, sendo $f(S)$ o valor da função objetivo da solução corrente e $f(BKS)$ o valor da função objetivo da melhor dentre todas as soluções geradas para a instância. Cada linha nos gráficos representa uma execução diferente, utilizando uma semente de números pseudo-aleatórios diferente.

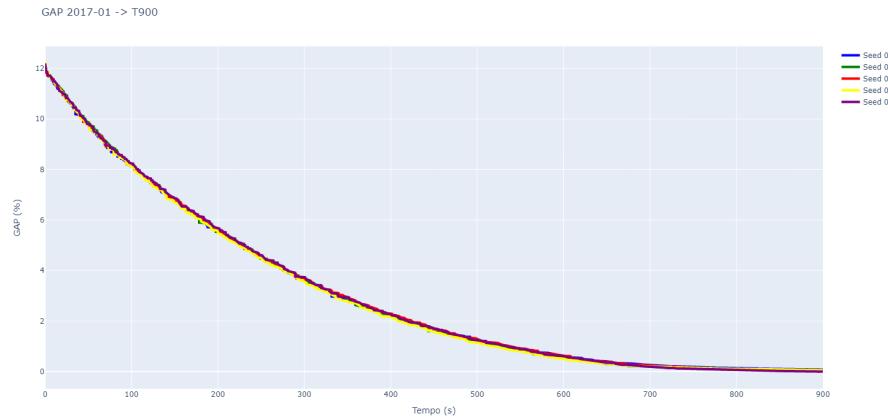


Figura 5.2: 1º semestre de 2017

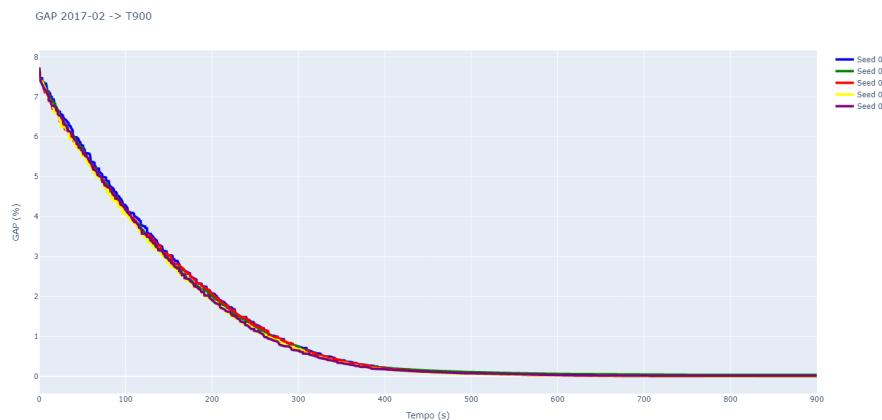


Figura 5.3: 2º semestre de 2017

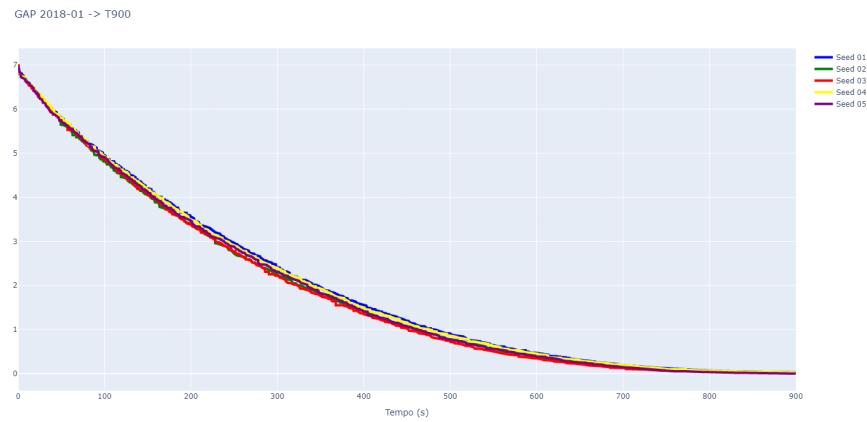


Figura 5.4: 1º semestre de 2018

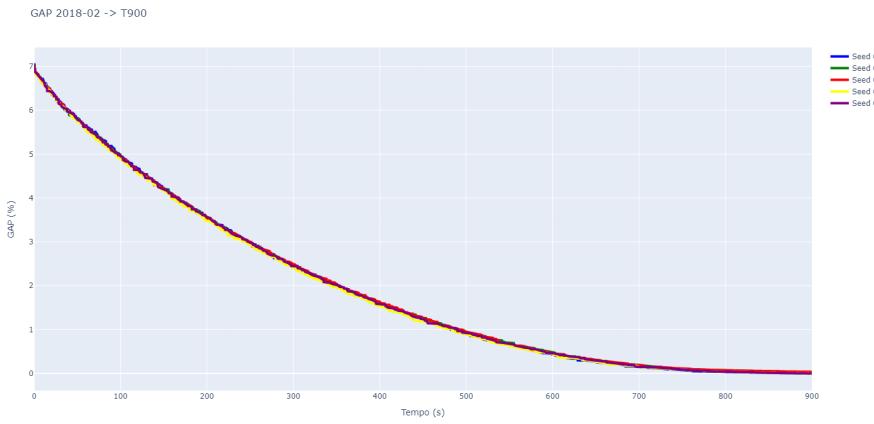


Figura 5.5: 2º semestre de 2018

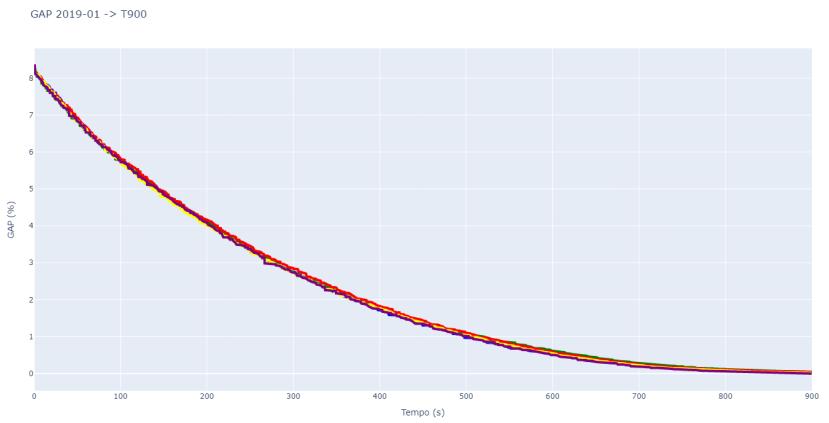


Figura 5.6: 1º semestre de 2019

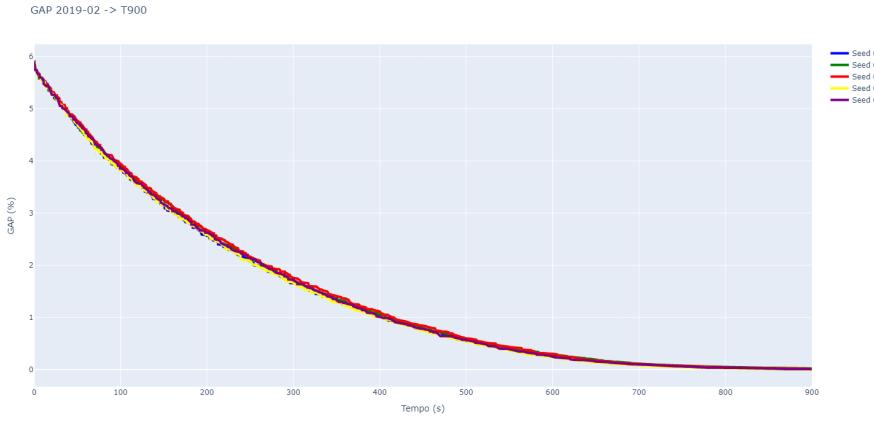


Figura 5.7: 2º semestre de 2019

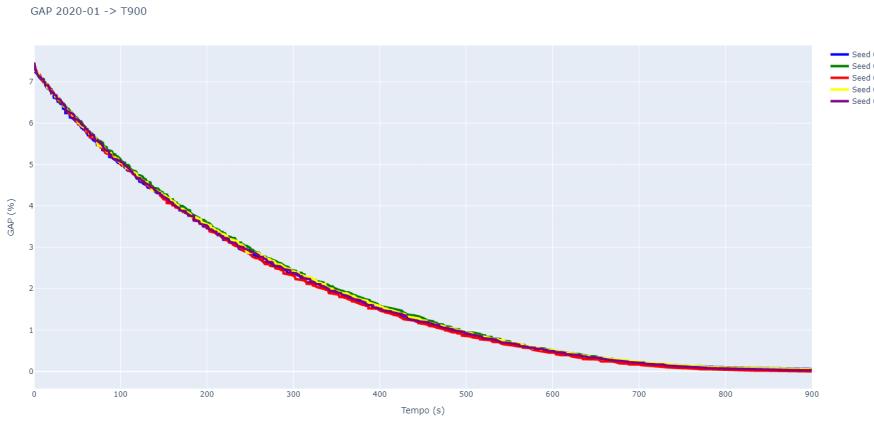


Figura 5.8: 1º semestre de 2020

Os gráficos mostram o comportamento do LAHC. No início da fase de refinamento, a solução inicial melhora bastante. Este processo começa a se estabilizar próximo a 3/4 do tempo de execução, sugerindo que uma solução ótima local foi encontrada.

A Tabela 5.2.3 sumariza os resultados obtidos pelo algoritmo e os compara com os resultados obtidos por um algoritmo *Simulated Annealing* (SA). Para essa comparação, algumas restrições foram ignoradas pois o algoritmo SA não as considerava na função objetivo, são elas ts_i e ts_p (professores que trocam de sala ou prédio em horários consecutivos).

Instância	Algoritmo LAHC			Algoritmo SA		
	Média	Melhor	Desv.P.	Média	Melhor	Desv.P.
2017-01	235.961,2	235.767	86.8488	603.379	602.503	392
2017-02	338.361,2	338.274	38.9970	766.851	766.597	113
2018-01	614.765,4	614.523	108.4045	1.807.779	1.807.507	121
2018-02	563.927,4	563.837	40.4281	1.253.806	1.252.855	425
2019-01	446.886,4	446.796	40.4281	1.647.632	1.647.374	115
2019-02	576.391,8	576.202	84.8811	1.430.094	1.429.603	219
2020-01	460.998,2	460.835	72.9852	1.597.164	1.596.496	299

Tabela 5.4: Resultados obtidos pelos algoritmos LAHC e SA

5.3 Problema multi-objetivo

Nas seções seguintes serão apresentadas uma revisão dos métodos utilizados e uma visualização dos resultados obtidos para a abordagem multi-objetiva do problema. Nessa versão todos os objetivos são avaliados separadamente, onde a qualidade de uma solução é avaliada com base na sua dominância em relação as outras soluções da vizinhança.

5.3.1 LAHC multi-objetivo

Para a abordagem multi-objetivo do problema, a solução inicial que é utilizada para iniciar o algoritmo LAHC é, também, a solução gerada pelo algoritmo guloso 1. Dada essa solução o algoritmo funciona da mesma forma que o anterior, porém ele armazena todas as soluções não dominadas encontradas durante sua execução. A grande diferença entre os dois algoritmos LAHC implementados nesse trabalho é que no multi-objetivo o impacto de cada objetivo na solução é alterado para que seja possível explorar ainda mais o espaço de solução em busca de soluções melhores.

Dado o exposto, os experimentos foram executados como mencionado na seção 5 e foram gerados os gráficos para representarem as soluções encontradas. Nesses gráficos, cada linha representa uma solução não dominada encontrada pelo algoritmo multi-objetivo. As abreviações representam os objetivos abordados nesse problema que foram descritos na equação 3.1, são elas:

- *OCI*: quantidade da ociosidade das salas;
- *DES*: quantidade da demanda desalocada;
- *DNA*: quantidade da demanda não atendida em um encontro alocado;
- *DNAE*: quantidade da demanda não atendida especial em um encontro alocado;

- SD : quantidade dos encontros que trocam de sala durante a semana;
- $PREF$: quantidade das preferências não atendidas;
- PTS : quantidade de professores que trocam de salas em horários consecutivos;
- PTP : quantidade de professores que trocam de prédio em horários consecutivos;

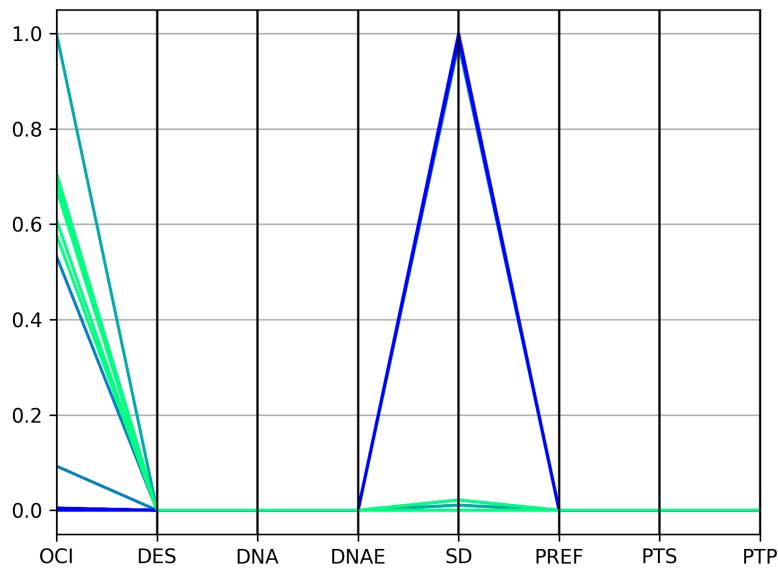


Figura 5.9: 1º semestre de 2017

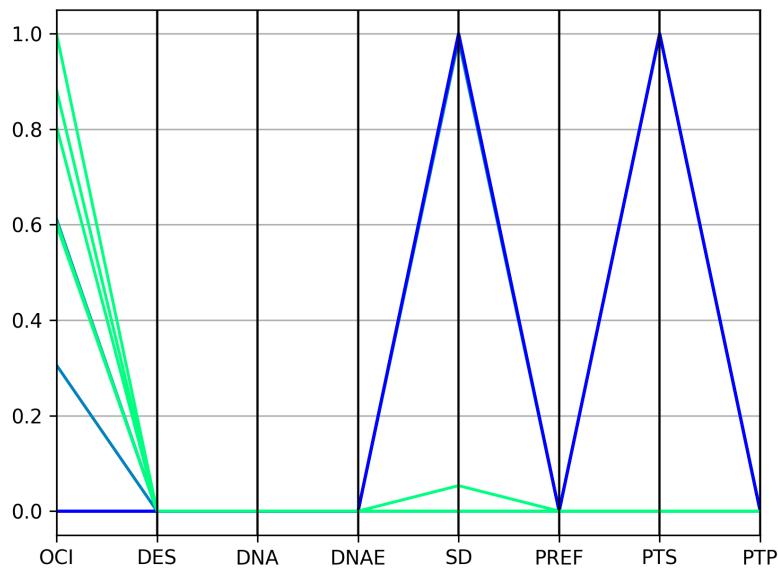


Figura 5.10: 2º semestre de 2017

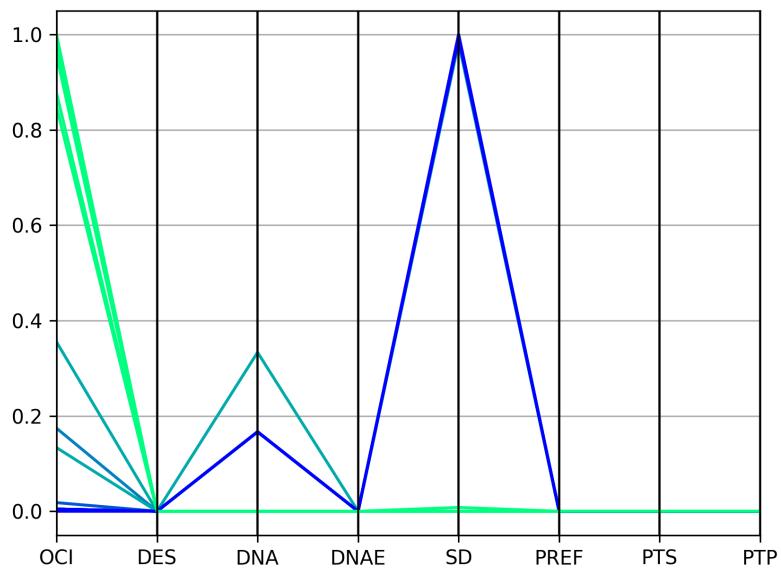


Figura 5.11: 1º semestre de 2018

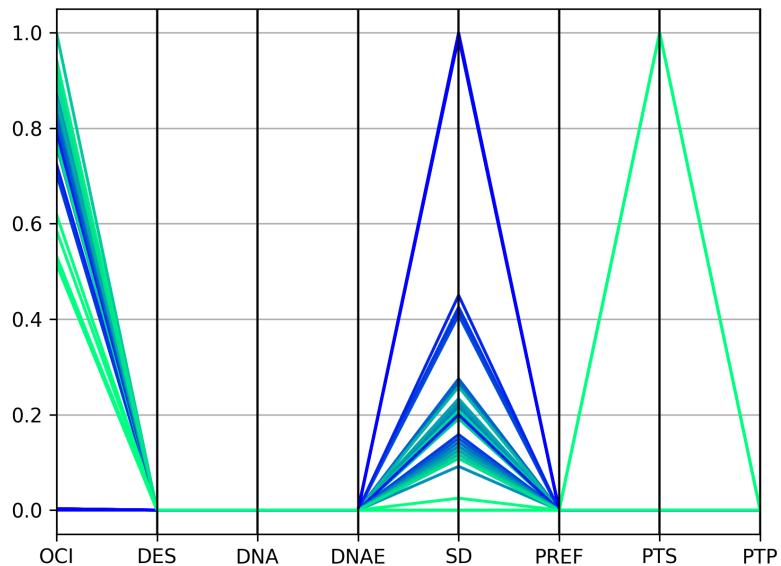


Figura 5.12: 2º semestre de 2018

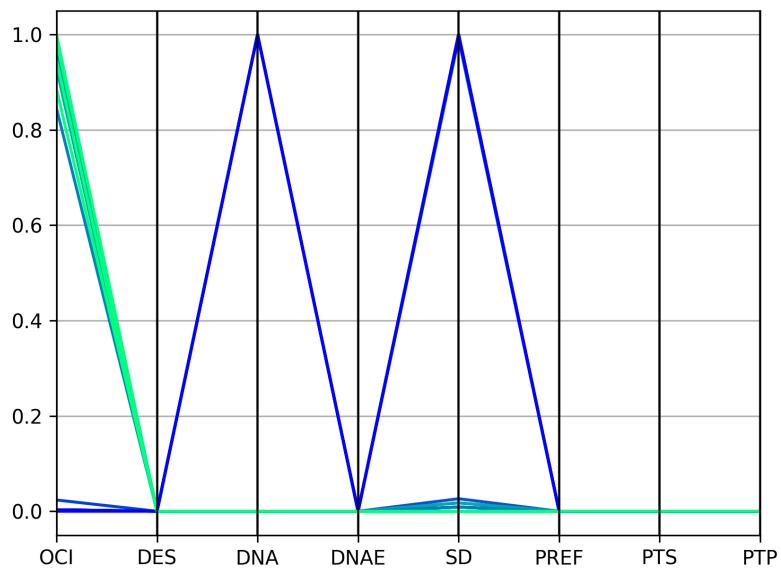


Figura 5.13: 1º semestre de 2019

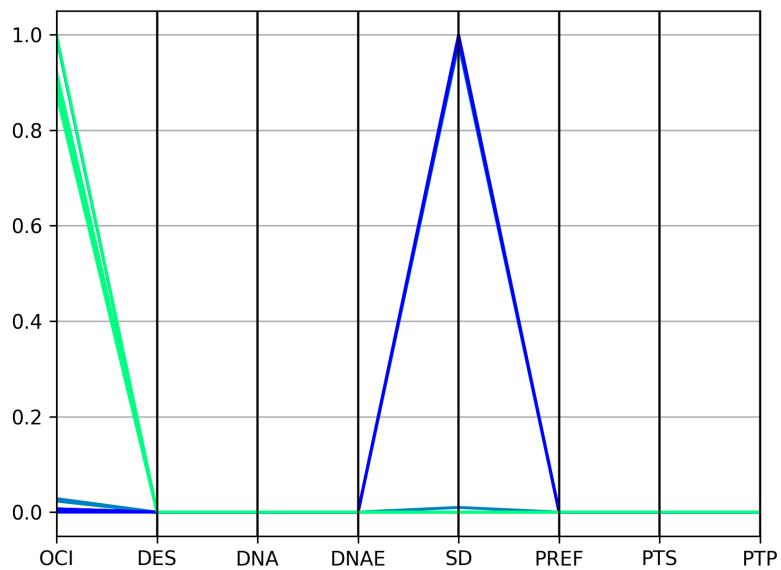


Figura 5.14: 2º semestre de 2019

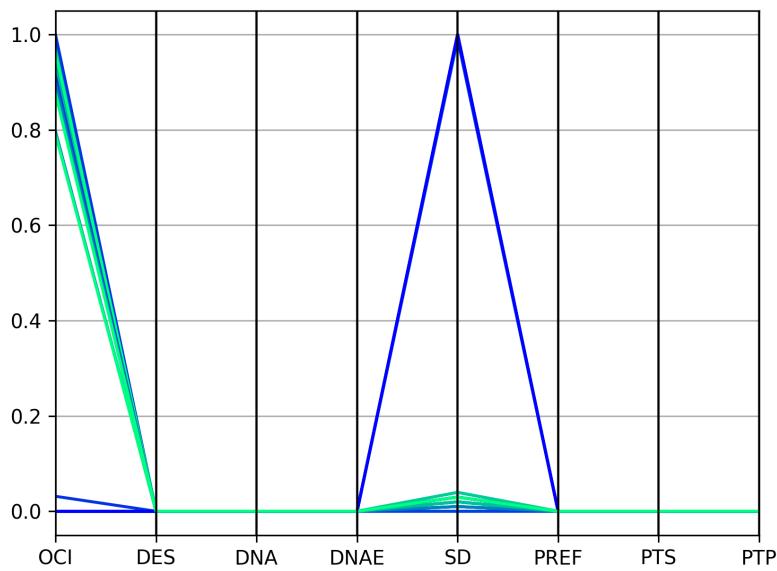


Figura 5.15: 1º semestre de 2020

5.3.2 NSGAII

Após o algoritmo 3 ser executado, são coletadas algumas soluções para servir de população inicial para o algoritmo 4. Com essa população inicial ele executa os passos apresentados na seção 4.6.2 em busca de melhorá-las. Para isso, foi utilizada uma população inicial de tamanho 500 que passa por 50 cruzamentos com uma chance de 30% de mutação.

Os gráficos que representam as soluções não dominadas encontradas pelo algoritmo 4 seguem a mesma lógica dos gráficos da seção anterior e as mesmas abreviações da tabela 5.3.1:

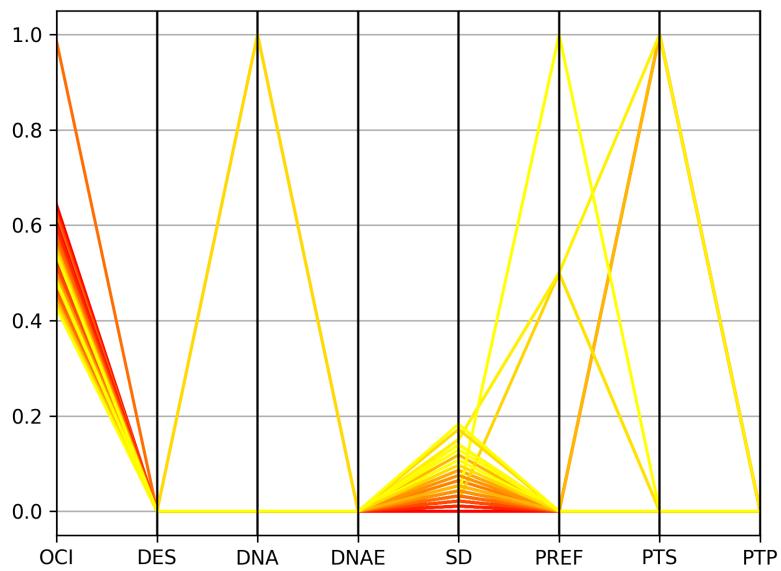


Figura 5.16: 1º semestre de 2017

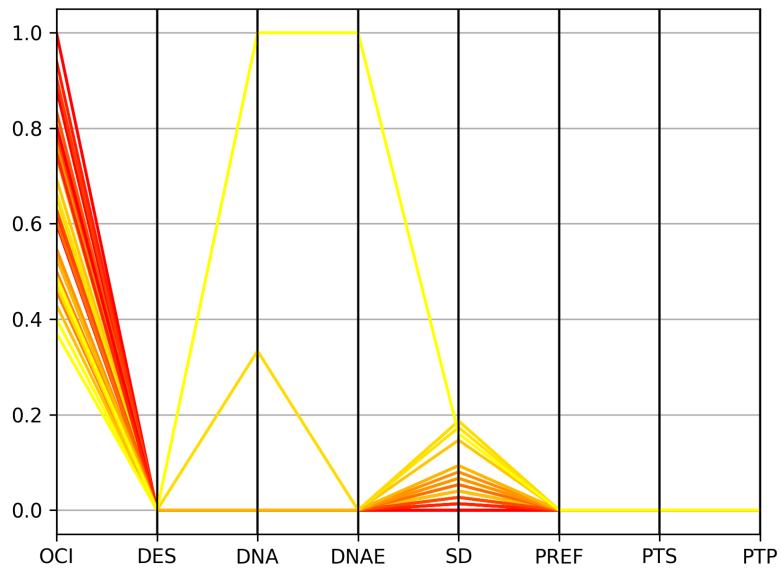


Figura 5.17: 2º semestre de 2017

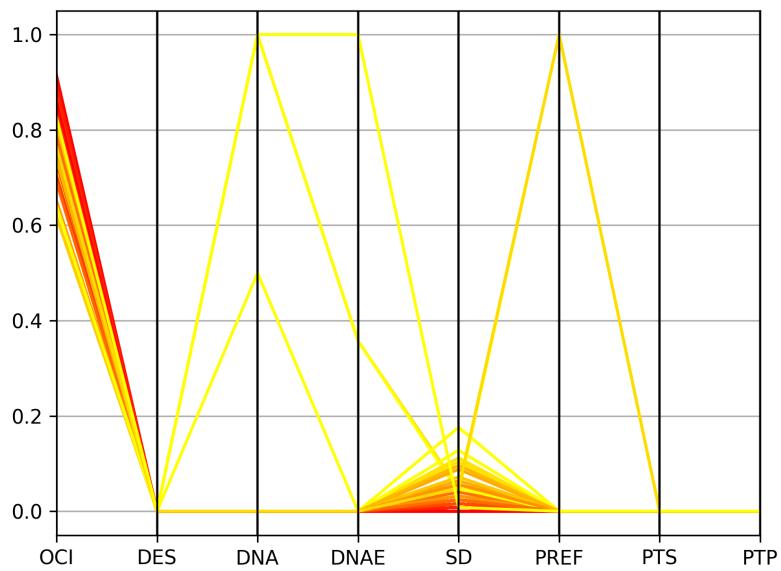


Figura 5.18: 1º semestre de 2018

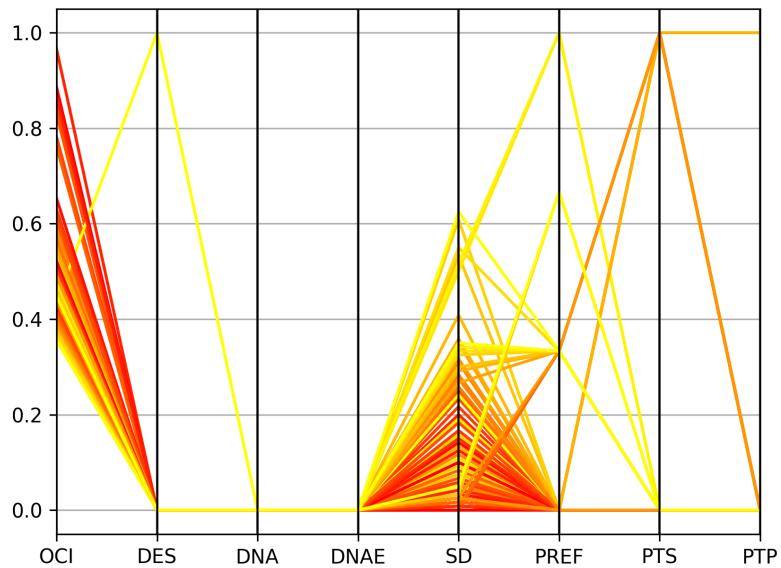


Figura 5.19: 2º semestre de 2018

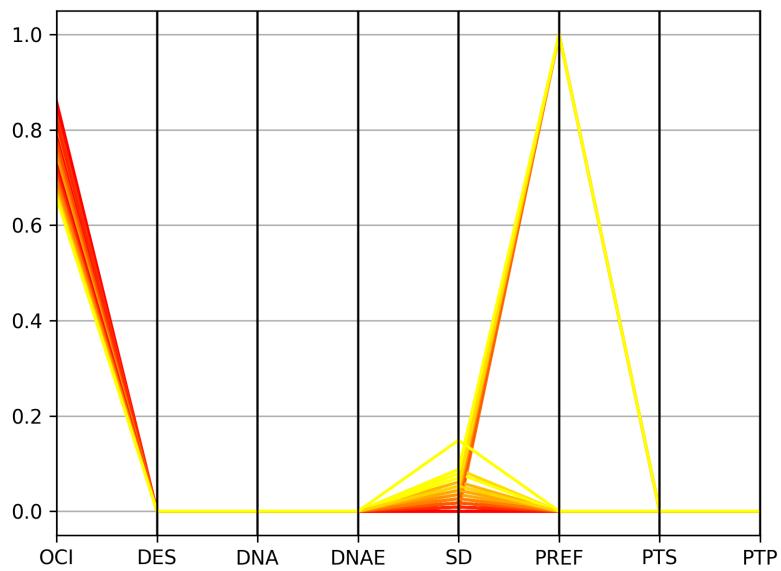


Figura 5.20: 1^o semestre de 2019

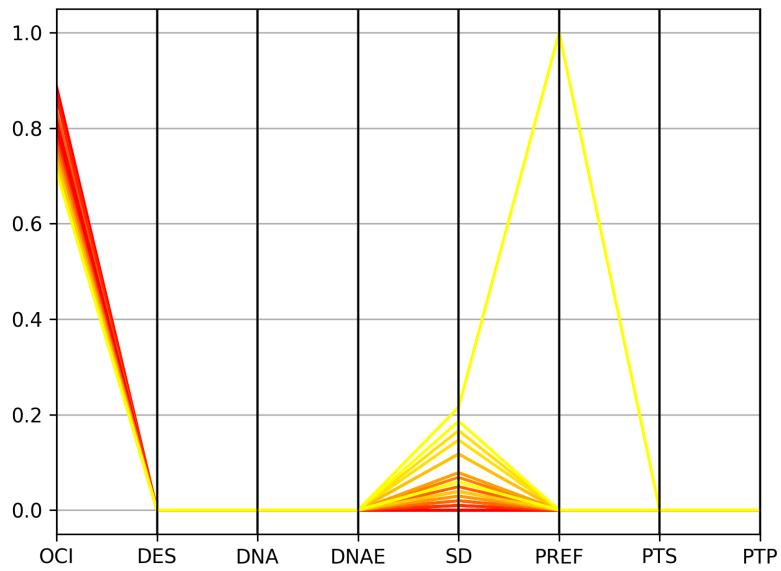


Figura 5.21: 2^o semestre de 2019

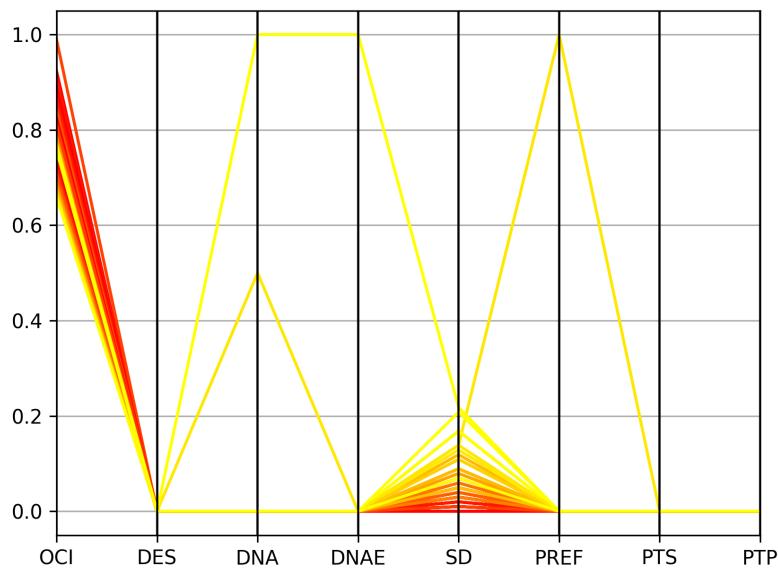


Figura 5.22: 1º semestre de 2020

5.3.3 NSGAII vc LAHC

Os gráficos a seguir foram gerados para facilitar a comparação entre as duas soluções apresentadas anteriormente. As linhas em verde e azul representam as soluções do LAHC multi-objetivo e as linhas em amarelo e vermelho representam as soluções do NSGAII:

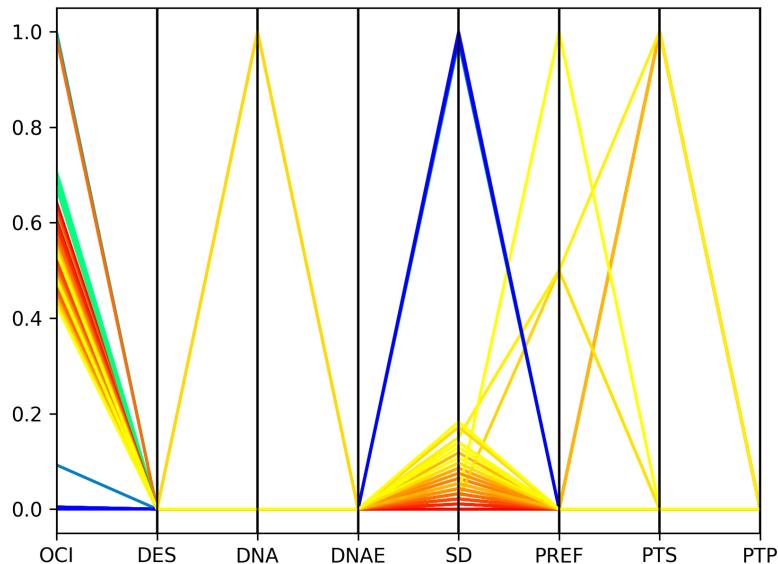


Figura 5.23: 1º semestre de 2017

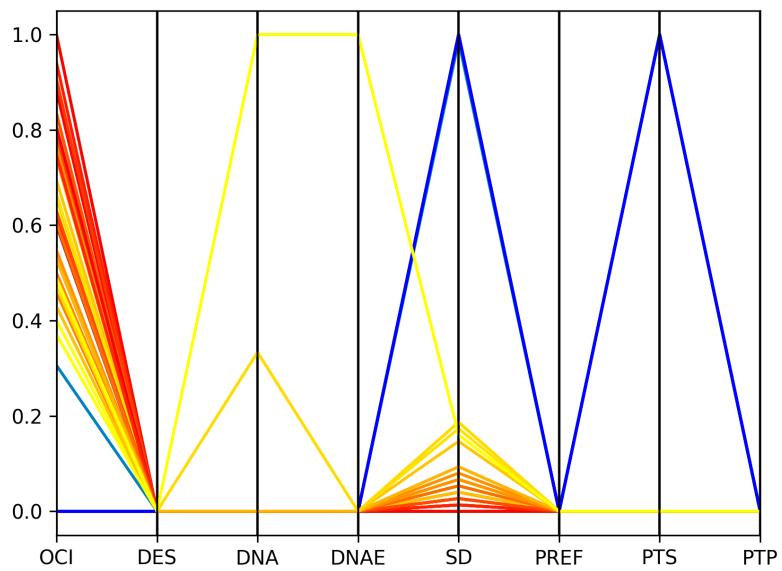


Figura 5.24: 2º semestre de 2017

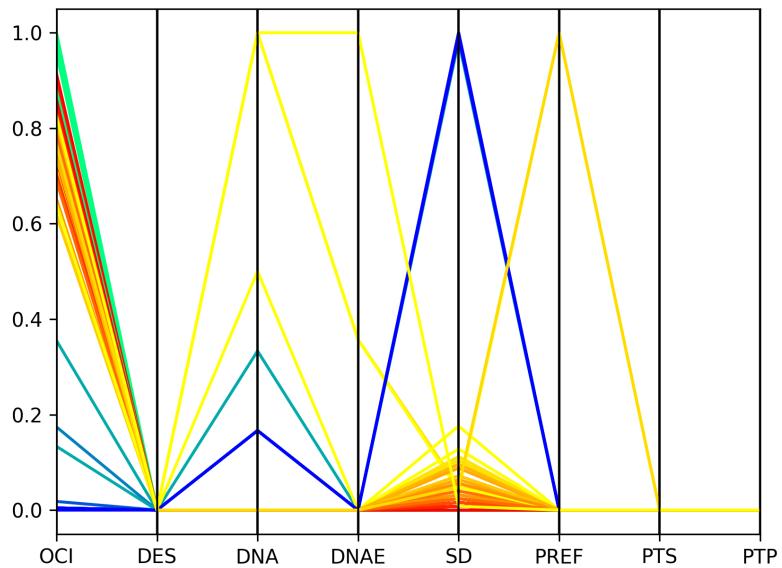


Figura 5.25: 1º semestre de 2018

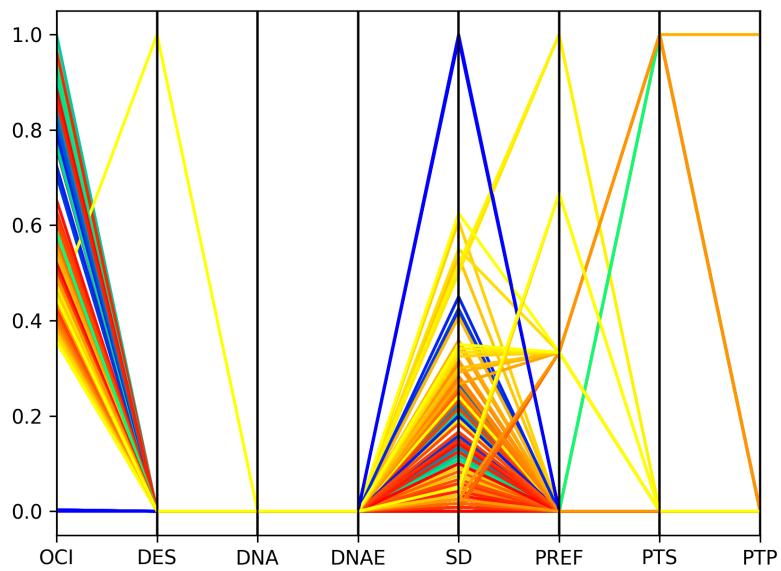


Figura 5.26: 2º semestre de 2018

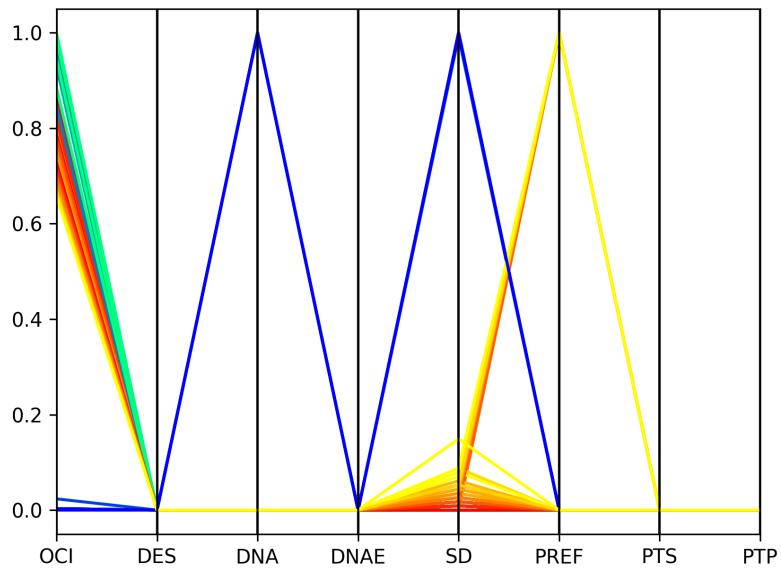


Figura 5.27: 1º semestre de 2019

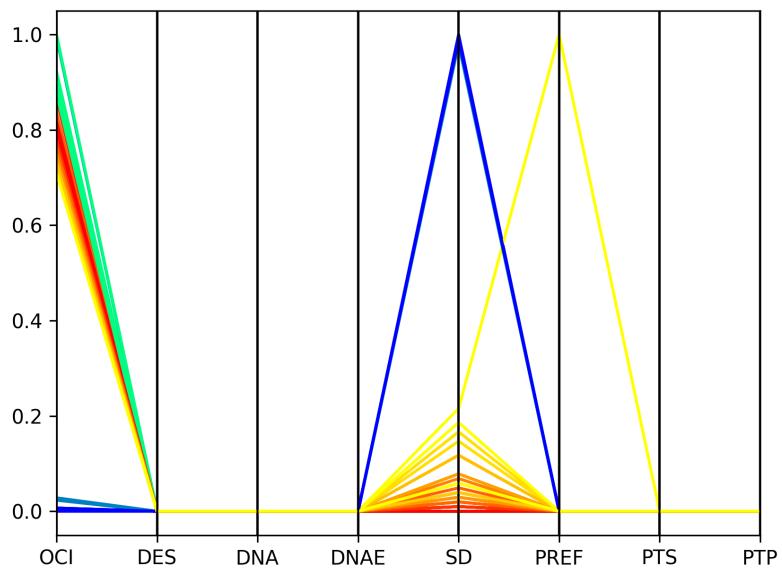


Figura 5.28: 2º semestre de 2019

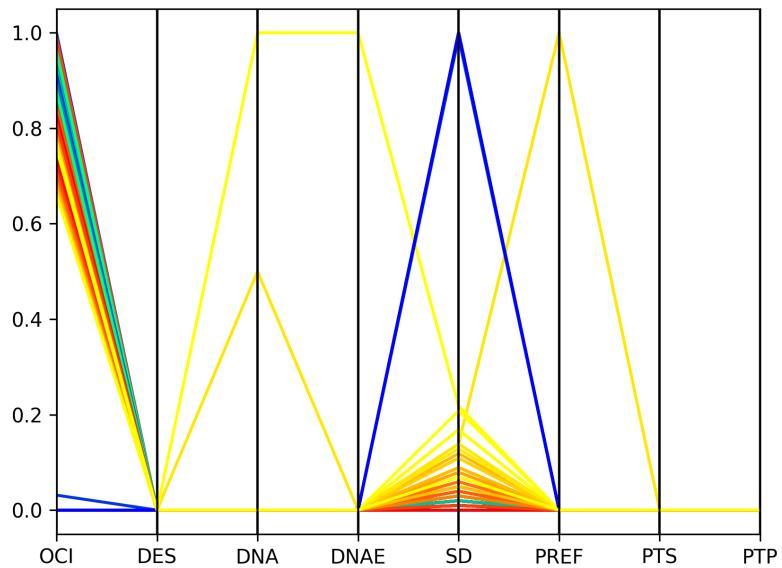


Figura 5.29: 1º semestre de 2020

6 Conclusões

Este projeto aborda o Problema de Alocação de Salas (PAS) considerando os dados, restrições e objetivos do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). Foram implementados, um algoritmo construtivo guloso e a meta-heurística Late Acceptance Hill Climbing (LAHC) em sua forma mono e multi-objetiva, para gerar soluções de qualidade para o problema. Cinco estruturas de vizinhança são utilizadas, sendo exploradas de forma estocástica pelo LAHC. A cada vizinhança foi associada uma probabilidade, atualizada a cada iteração pelo algoritmo de aprendizado de máquina *Learning Automaton*.

Os resultados obtidos pelos algoritmos implementados na versão mono-objetiva são promissores, tendo superado os resultados de uma outra abordagem desenvolvida, baseada no algoritmo *Simulated Annealing*. No entanto, ainda é necessário avaliar as soluções produzidas na prática.

Os resultados obtidos pelos algoritmos na abordagem multi-objetiva também são promissores, visto que para todas as instâncias foram encontradas soluções com muitos dos objetivos totalmente satisfeitos. Além disso, foram encontradas várias outras soluções não dominadas cada uma com suas vantagens, dando uma variedade de escolhas que podem se adaptar aos interesses da instituição.

Referências Bibliográficas

- Al-Yakoob, S.M. & Sherali, H.D. (2006). Mathematical programming models and algorithms for a class-faculty assignment problem. European Journal of Operational Research, 173, 488-507.
- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V.B. (2017). Julia: A fresh approach to numerical computing. SIAM review, 59(1), 65-98.
- Even, S., Itai, A. & Shamir, A. (1976). On the complexity of timetabling and multicommodity flow problems. SIAM Journal of Computation, 5, 691-703.
- Kripka, R.M.L., & Kripka, M. (2012). Alocação de Salas Objetivando a Minimização de Deslocamento dos Alunos pelo Campus Central da Universidade de Passo Fundo. Congresso Nacional de Matemática Aplicada Computacional, Águas de Lindóia, SP, Brasil, 34.
- Prado, A.S. (2014). Problema de Alocação de Salas em Cursos Universitários: Um Estudo de Caso. Dissertação de Mestrado. Centro de Educação Tecnológica de Minas Gerais (CEFET-MG).
- Sales, E.S., Müller, F.M., & Simonetto, E.O. (2015). Solução do problema de alocação de salas utilizando um modelo matemático multi-índice. XLVII Simpósio Brasileiro de Pesquisa Operacional - SBPO, Porto de Galinhas, Pernambuco, Brasil, 2596-2607.
- Silva, D.J. da, & Silva, G.C. da. (2010). Heurísticas Baseadas no Algoritmo de Coloração de Grafos para o problema de alocação de salas em uma instituição de ensino superior. Simpósio Brasileiro de Pesquisa Operacional, Bento Gonçalves, Rio Grande do Sul, Brasil, 42.
- Souza, M.J.F., Martins, A.X. & Araújo, C.R. (2002). Experiências com a utilização de Simulated Annealing e Busca Tabu na resolução do Problema de Alocação de Salas. Simpósio Brasileiro de Pesquisa Operacional - SBPO, Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 34.
- Subramanian, A., Medeiros, J.M.F., Cabral, L.A.F. & Souza, M.J.F. (2006). Aplicação da metaheurística Busca Tabu na resolução do Problema de Alocação de Salas do Centro de Tecnologia da UFPB. Encontro Nacional de Engenharia de Produção Fortaleza, Ceará, Brasil, 26.

Subramanian, A., Medeiros, J.M., Formiga, L, A., & Souza, M.J.F. (2011). Aplicação da metaheurística busca tabu ao problema de alocação de aulas a salas em uma instituição universitária. *Revista Produção Online*, 11(1), 54–75.

Toffolo, T. A. M., Christiaens, J., Van Malderen, S., Wauters, T., & Vanden Berghe, G. (2018). Stochastic local search with learning automaton for the swap-body vehicle routing problem. *Computers & Operations Research*, 89, 68–81.

Qingfu Zhang; Hui Li (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. , 11(6), 0–731.

Shamik Chaudhuri; Kalyanmoy Deb (2010). An interactive evolutionary multi-objective optimization and decision making procedure. , 10(2), 496–511.

Marcos Antonio Alves(2018). Proposta de Agregação Robusta de Múltiplos Métodos com Incertezas em Problemas de Tomada de Decisão Multicritério.

Deb, Kalyanmoy; Jain, Himanshu (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4), 577–601.

Jain, Himanshu; Deb, Kalyanmoy (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation*, 18(4), 602–622.

Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. , 6(2), 0–197.

de Freitas, Alan R.R.; Fleming, Peter J.; Guimarães, Frederico G. (2015). Aggregation Trees for visualization and dimension reduction in many-objective optimization. *Information Sciences*, 298(), 288–314.

Silva, Rodrigo; Salimi, Armin; Li, Min; Freitas, Alan; Guimaraes, Frederico; Lowther, David (2015). Visualization and Analysis of Trade-offs in Many-Objective Optimization: A Case Study on the Interior Permanent Magnet Motor Design. *IEEE Transactions on Magnetics*, (), 1–1.