PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

ALOK

Alocação Automática de Salas de Aula com Auxílio de Algoritmos Genéticos

Bruno Messeder dos Anjos

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Engenharia da Computação

Rio de Janeiro, julho de 2024

Bruno Messeder dos Anjos

ALOK

Alocação Automática de Salas de Aula com Auxílio de Algoritmos Genéticos

Relatório de Projeto Final, apresentado ao programa Engenharia da Computação da PUC-Rio como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Marco Aurélio Cavalcanti Pacheco
Co-Orientadora: Profa. Manoela Rabello Kohler

Rio de Janeiro, julho de 2024

Resumo

Anjos, Bruno M. Pacheco, Marco A. Kohler, Manoela. **ALOK - Alocação Automática de Salas de Aula com Auxílio de Algoritmos Genéticos**. Rio de Janeiro, 2024. 31 p. Relatório Final de Projeto Final de Graduação I – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto tem o intuito de automatizar a alocação de salas de aula para as disciplinas da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). A motivação para o desenvolvimento deste sistema advém da complexidade e da carga de trabalho envolvida no processo manual atualmente realizado pelos departamentos e pela Coordenação Central de Graduação (CCG), que precisa alocar aproximadamente 4500 disciplinas a cada semestre. Esse processo manual é repetitivo, suscetível a erros e consome um tempo considerável dos profissionais envolvidos, o que impacta negativamente a eficiência e a precisão da alocação das salas. O *software* desenvolvido em *Python* utiliza Algoritmos Genéticos (AG), uma metaheurística de Inteligência Artificial, para encontrar uma alocação otimizada, levando em consideração requisitos e preferências de cada disciplina, como o número de vagas ofertadas e prédios e andares preferenciais. A utilização de AG no software desenvolvido é uma escolha estratégica, pois essa técnica de otimização é capaz de lidar com múltiplas restrições e preferências simultaneamente. Isso inclui considerar o número de vagas ofertadas por cada disciplina, as preferências por prédios e andares específicos, e a necessidade de recursos adicionais nas salas, como computadores. As vantagens da automação desse processo são numerosas. Primeiramente, a eficiência é significativamente aumentada, liberando os departamentos e a CCG do fardo do trabalho manual e permitindo que esses recursos humanos sejam alocados para tarefas mais estratégicas e intelectualmente desafiadoras. Em segundo lugar, a precisão na alocação das salas é aprimorada, uma vez que o algoritmo pode considerar um maior número de variáveis, que seria impraticável manualmente. O sistema desenvolvido é implementado em Python e pode ser executado em servidores web, acessível através de qualquer navegador. O software pode realizar a alocação de forma integral ou parcial, adaptando-se às necessidades específicas dos departamentos. Os testes realizados com dados reais de semestres anteriores demonstram que o sistema atende às expectativas dos departamentos e da CCG, proporcionando uma solução eficiente e precisa para o problema da alocação de salas de aula.

Palavras-chave

Alocação de Salas; Algoritmos Genéticos; Otimização; Inteligência Artificial; Python.

Abstract

Anjos, Bruno M. Pacheco, Marco A. Kohler, Manoela. **ALOK - Automatic Classroom Allocation with the Aid of Genetic Algorithms**. Rio de Janeiro, 2024. 31 p. Relatório Final de Projeto Final de Graduação I – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This project aims to automate the allocation of classrooms for the disciplines at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). The motivation for developing this system arises from the complexity and workload involved in the current manual process carried out by the departments and the Central Graduation Coordination (CCG), which needs to allocate approximately 4500 disciplines each semester. This manual process is repetitive, error-prone, and consumes a considerable amount of the professionals' time, negatively impacting the efficiency and accuracy of classroom allocation. The software developed in Python uses Genetic Algorithms (GA), an Artificial Intelligence metaheuristic, to find an optimized allocation, taking into account the requirements and preferences of each discipline, such as the number of offered seats and preferred buildings and floors. The use of GA in the developed software is a strategic choice, as this optimization technique can handle multiple constraints and preferences simultaneously. This includes considering the number of seats offered by each discipline, preferences for specific buildings and floors, and the need for additional resources in the rooms, such as computers. The advantages of automating this process are numerous. Firstly, efficiency is significantly increased, freeing the departments and CCG from the burden of manual work and allowing these human resources to be allocated to more strategic and intellectually challenging tasks. Secondly, the accuracy of classroom allocation is enhanced, as the algorithm can consider a greater number of variables, which would be impractical manually. The developed system is implemented in Python and can be run on web servers, accessible through any browser. The software can perform the allocation in either a full or partial manner, adapting to the specific needs of the departments. Tests conducted with real data from previous semesters demonstrate that the system meets the expectations of the departments and CCG, providing an efficient and accurate solution to the problem of classroom allocation.

Keywords

Classroom Allocation; Genetic Algorithm; Optimization; Artificial Intelligence; Python.

Lista de Figuras

Sumário

1. Introdução

Dentre os diversos processos necessários para o funcionamento de uma universidade, a alocação de disciplinas em salas de aula é uma etapa fundamental, com o objetivo de garantir que as salas possuam os recursos necessários para as disciplinas, como número de vagas oferecidas, e para que não haja disciplinas sem sala disponível.

Na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), os departamentos e a Coordenação Central de Graduação (CCG) definem todo semestre, de forma totalmente manual, as salas de cada uma das disciplinas ofertadas, que costuma ser em torno de 4500. Esse processo é bastante repetitivo e demorado. Por conta disso, há o interesse por parte da CCG em automatizar parte desse processo, visando diminuir o trabalho dela e dos departamentos.

Um programa de alocação de salas de aula não pode ser feito de forma "ingênua", alocando as disciplinas em qualquer sala disponível, pois devem ser levados em consideração diversos requisitos, como o número de vagas ofertadas, e preferências das disciplinas, como a proximidade com o departamento, prédios e andares prioritários e a necessidade de recursos adicionais, como computadores nas salas. Com isso, é necessário um programa de alocação inteligente, com a utilização de técnicas como Algoritmos Genéticos (HOLLAND, 1992), para que seja possível alocar todas as disciplinas em salas adequadas de forma a agradar os departamentos, a CCG, os alunos e os professores.

Este projeto visa descrever o desenvolvimento do programa de alocação de disciplinas em salas de aula, explicando o seu funcionamento, como é usado na prática e quais etapas do processo de alocação de sala serão substituídas por ele.

O programa foi escrito na linguagem *Python*, compatível com os sistemas operacionais *Windows* e distribuições *Linux*, e é executado em um servidor *web*, podendo ser acessado em qualquer navegador, com a utilização das bibliotecas '*deap*' (DEAP, 2023), de Algoritmos Genéticos, e '*dash*' (PLOTLY, 2023), de criação de *dashboards* interativos, para o acompanhamento do andamento do algoritmo, a visualização dos resultados obtidos e métricas, como percentual de disciplinas alocadas.

A alocação de disciplinas pode ser feita de forma integral ou parcial. Na

forma integral, o programa substitui quase todo o trabalho dos departamentos e da CCG, exceto eventuais ajustes, e define as salas de aulas de todas as disciplinas. Na forma parcial, o programa substitui o trabalho da CCG de escolha de salas para uma parte das disciplinas, sem alterar o trabalho dos departamentos. O projeto visa incluir essas duas formas de funcionamento.

2. Situação Atual

2.1 Processo de alocação manual

A alocação de salas de aula da PUC-Rio atualmente é feita pelos departamentos e pela CCG de forma totalmente manual, com diversas etapas que se repetem todo semestre.

Inicialmente, os departamentos definem quais disciplinas serão disponibilizadas para um determinado semestre junto ao dia e a hora dessas disciplinas. Após isso, cada departamento escolhe as salas de algumas das suas disciplinas, usando uma lista de salas preferenciais daquele departamento. Essa lista contém salas que, idealmente, são usadas para as aulas apenas daquele departamento.

Em seguida, a CCG recebe a lista de disciplinas com e sem sala definidas de todos os departamentos. As disciplinas que estão sem sala são alocadas em qualquer sala disponível, podendo ser sala de uso geral ou sala preferencial dos departamentos, se não houver outras salas disponíveis. Para que uma disciplina possa ser atribuída a uma sala preferencial de outro departamento, a CCG entra em contato com o departamento ao qual a sala pertence para pedir permissão de uso.

Após isso, caso ainda haja disciplinas sem sala disponível, a CCG notifica os departamentos aos quais essas disciplinas pertencem, informando que será necessário mudar os dias e/ou horários, ou, eventualmente, cancelar algumas disciplinas. Após as mudanças serem feitas, todas as disciplinas têm uma sala de aula.

Com o quadro de disciplinas completo, as disciplinas são divulgadas no *site* de horários da PUC-Rio, com um dia, hora e sala de aula definidos, que pode ser consultado pelos alunos. Em seguida, há o período de matrícula dos alunos, que escolhem quais disciplinas irão cursar no próximo semestre. Após o término do período de matrícula, eventuais mudanças podem ser feitas nas salas definidas para as disciplinas. Caso haja uma quantidade grande de alunos que não conseguiram se matricular em uma determinada disciplina por falta de vagas, essa disciplina pode passar a ofertar mais vagas e a sala pode ser alterada para comportar o novo número de alunos. Ao final de todas essas etapas, a lista de disciplinas estará completa, com todas as disciplinas associadas a uma sala.

2.2 Abordagens Existentes

Atualmente, existem diversos artigos utilizando Algoritmos Genéticos para a alocação de salas de aula. O artigo *Design and Application of an Improved Genetic Algorithm to a Class Scheduling System* (CHEN *et al*, 2021) propõe a otimização de aulas de colégios e universidades a partir de restrições predefinidas, como ocupação máxima das salas e recursos de multimídia, enquanto o artigo *A genetic algorithm approach to school timetabling* (BELIGIANNIS *et al*, 2009) permite que as restrições sejam mais dinâmicas, podendo se adaptar a diferentes universidades. Apesar disso, essas soluções não podem ser usadas para resolver o problema da PUC-Rio, uma vez que o horário das aulas é fixo, não podendo ser alterado pela otimização, enquanto os artigos otimizam o dia e hora das aulas, além das salas.

Devido às características específicas da alocação de salas da PUC-Rio, também não há *softwares* comerciais que possam ser usados de forma a seguir todos os requisitos e preferências das disciplinas. A PUC-Rio possui sistemas antigos de alocação automatizada de salas de aula, porém, esses sistemas foram descontinuados, e não atendem aos novos requisitos dos departamentos e da CCG.

2.3 Algoritmo Genético

O Algoritmo Genético é uma técnica computacional de otimização que se baseia na teoria de evolução (HOLLAND, 1992). Nele, um indivíduo representa uma solução possível para o problema em questão. No caso desse projeto, um indivíduo representa uma disposição das disciplinas nas salas de aula. Assim como na teoria da evolução, indivíduos podem ter características que os fazem melhores ou piores que os outros. A cada indivíduo, é atribuído um valor numérico que representa o quão apto ele é, ou seja, o quão bem ele resolve o problema proposto.

O algoritmo genético é um processo iterativo, ou seja, são necessárias várias etapas para que o algoritmo dê resultados satisfatórios. Cada etapa é chamada de geração. O processo inteiro é chamado de otimização.

No início do algoritmo, diversos indivíduos são criados aleatoriamente ou

seguindo uma lógica preestabelecida. A cada geração, há um processo de “reprodução”, em que novos indivíduos são gerados a partir de pares de indivíduos aleatórios. Cada indivíduo possui um DNA, que o identifica e é usado para gerar novos indivíduos, a partir de operações de cruzamento, ou recombinação, em que os DNAs de dois indivíduos são misturados, e mutação, em que partes do DNA de um indivíduo são alteradas aleatoriamente, gerando um novo DNA, e, portanto, um novo indivíduo que será “filho” de dois indivíduos “pai”. Os indivíduos “filhos”, farão parte da geração seguinte, e os indivíduos “pai” são descartados. Os pares de indivíduos são selecionados a partir de operadores, como torneio ou roleta. Esses operadores escolhem aleatoriamente indivíduos se baseando em suas aptidões. Um indivíduo com maior aptidão tem mais chance de ser selecionado. Com o passar das gerações, há a tendência dos indivíduos terem melhores aptidões, até que se chegue a um resultado ótimo.

Há diversas técnicas adicionais para melhorar o desempenho do algoritmo genético, como a criação de uma elite, que é uma lista dos melhores indivíduos de qualquer geração, para que indivíduos muito bons não sejam descartados na geração seguinte. A cada geração, a elite é atualizada, podendo incluir indivíduos dessa geração, caso eles sejam melhores que indivíduos da elite. Esses indivíduos são usados em todas as gerações para criar novos indivíduos, aumentando a chance dos indivíduos “filhos” terem aptidão maior.

3. Objetivos do Trabalho

O objetivo deste projeto é desenvolver um sistema de alocação automática de disciplinas em salas de aula da PUC-Rio para reduzir o trabalho manual repetitivo dos departamentos e da CCG. O sistema é capaz de escolher as salas de aula de forma integral, alocando todas as disciplinas, ou parcial, alocando apenas as disciplinas que não foram alocadas pelos departamentos. Para isso, foi implementado em *Python* um algoritmo genético, para fazer a alocação das disciplinas, e um *dashboard*, para acompanhamento, visualização e *download* dos resultados em forma de aplicação *web*.

Há restrições em quais disciplinas serão otimizadas. As disciplinas precisam ser de graduação, ter horário fixo e não ter sala compartilhada com outra disciplina.

4. Atividades Realizadas

4.1 Estudos Preliminares

Antes de desenvolver o sistema, o aluno já possuía conhecimento de algoritmos genéticos e aplicações web.

4.2 Estudos Conceituais e de Tecnologia

Embora já possuísse conhecimento de algoritmos genéticos, não havia aplicado esses conhecimentos em um problema de alocação, sendo necessário estudar métodos específicos para esse tipo de problema, como tipos de *crossover* e mutação. Além disso, foi necessário estudar mais a fundo o *framework dash*, para poder implementar todas as funcionalidades requeridas e a biblioteca *deap*, de algoritmo genético.

4.3 Método

O sistema foi desenvolvido inicialmente como um MVP (*Minimum Viable Product*), que consistia em uma versão funcional mínima do sistema, para que pudesse ser testada pela CCG, para ter um *feedback* do sistema. Em seguida, foram feitas as modificações sugeridas até que o sistema atendesse às necessidades da CCG.

4.4 Cronograma

A figura 1 mostra o cronograma original, enquanto a figura 2 mostra o cronograma revisado.

Figura 1 – Cronograma original

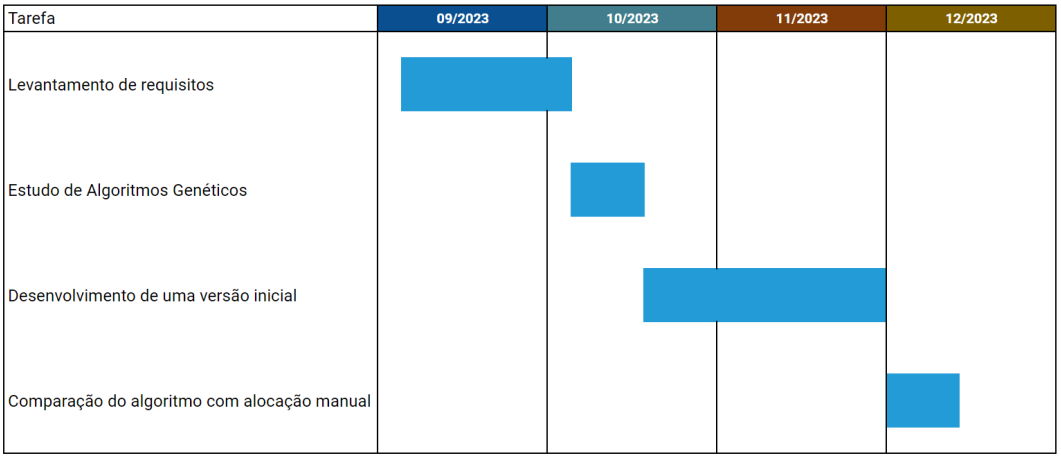
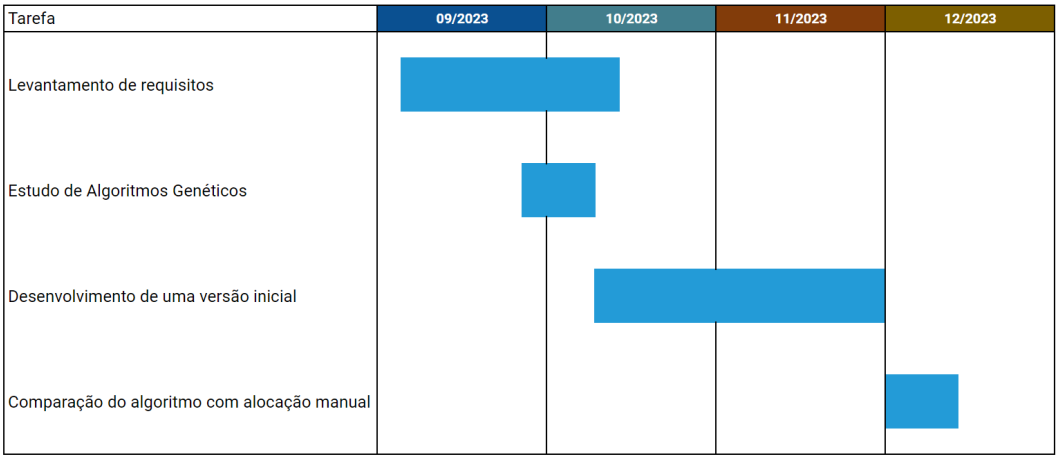


Figura 2 – Cronograma revisado



5. Projeto e Especificação do Sistema

5.1 Visão Geral do Sistema

O sistema possui como funcionalidade principal a otimização de disciplinas em salas de aulas. Para isso, há diversas funcionalidades implementadas, como recebimento de arquivos do usuário, otimização, alteração manual de salas de aula, visualização de estatísticas geradas a partir do resultado da otimização e *download* do resultado.

Como o sistema inclui um processo de otimização, que é um processo demorado e demanda bastante do processador, foi necessário desenvolvê-lo de forma eficiente. Para isso, durante o desenvolvimento, diversas operações implementadas manualmente foram substituídas por funcionalidades da biblioteca *numpy*, que é ideal para operar sobre vetores e matrizes de forma rápida. Além disso, o processo de otimização foi paralelizado, com a utilização de múltiplos processos para o cálculo da aptidão dos indivíduos. Essa paralelização foi implementada utilizando a biblioteca *multiprocessing*, padrão do *Python*.

5.2 Arquivos do Usuário

Para a otimização ser feita, o usuário precisa fornecer arquivos contendo as informações referentes às disciplinas e salas de aula. Todos os arquivos são no formato *Excel*. São eles:

- Arquivo do planejamento acadêmico, que contém todas as disciplinas de um determinado semestre da PUC-Rio, incluindo disciplinas que já foram alocadas anteriormente pelos departamentos, com informações como dia, hora, departamento e vagas oferecidas pela disciplina
- Arquivo de salas, que informa em que prédio e andar está cada sala, além de sua ocupação máxima
- Arquivo de *slots* de salas, que indica quais salas estão disponíveis em quais dias e horários
- Arquivo de prédios e andares preferenciais, que contém os prédios e andares preferenciais de todos os departamentos

A partir dos arquivos fornecidos pelo usuário, é criado um ambiente virtual que será usado durante a otimização. Esse ambiente possui formato

similar a um banco de dados SQL, porém utilizando *dataclasses* de *Python*. Isso foi feito pois os dados do usuário são armazenados em seu navegador e nenhuma funcionalidade específica de SQL é necessária para o funcionamento do sistema, além de reduzir o uso de disco do servidor.

As informações do arquivo de planejamento acadêmico são divididas em duas tabelas: Disciplina e Classe. As figuras 3 e 4 mostram a modelagem das tabelas Disciplina e Classe respectivamente.

Figura 3 – Esquema da tabela disciplina

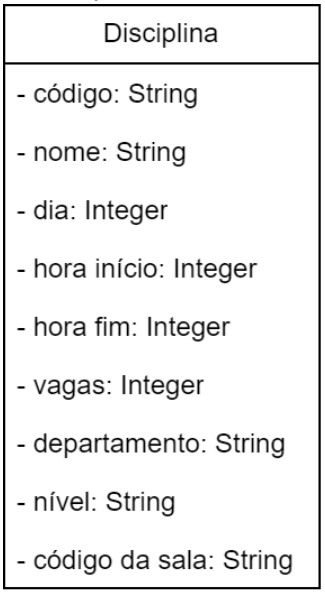
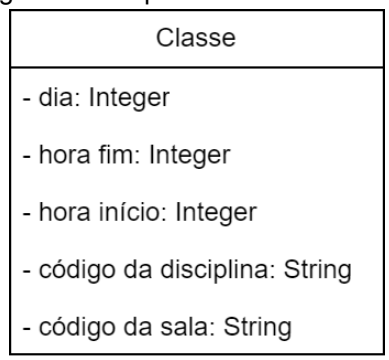


Figura 4 – Esquema da tabela classe



O arquivo de salas é dividido em duas tabelas, Prédio e Sala. As figuras 5 e 6 mostram a modelagem das tabelas Prédio e Sala respectivamente:

Figura 5 – Esquema das salas de prédio

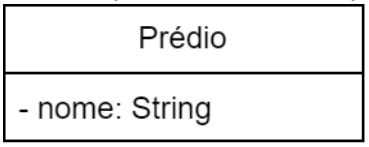
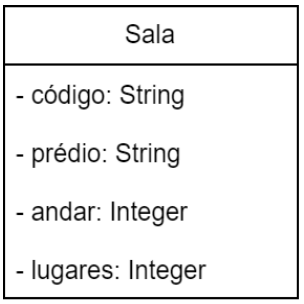
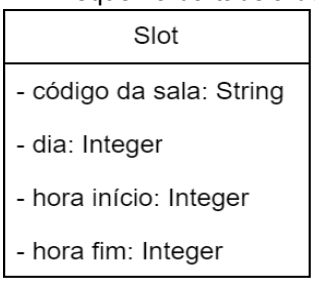


Figura 6 – Esquema das salas de aula



O arquivo de *slots* contém os horários disponíveis das salas da PUC-Rio. A figura 7 mostra a modelagem da tabela de Slots.

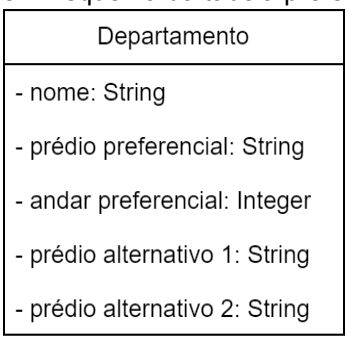
Figura 7 – Esquema da tabela de *slots*



Após o arquivo de *slots* ser carregado, ele é convertido em um tensor de bits sala x dia x hora, em que cada elemento do tensor indica, com zero ou um, se a sala está disponível em um dia e hora específicos. Isso é feito para agilizar a procura de salas disponíveis durante a otimização.

O arquivo de prédios e andares preferenciais contém, para cada departamento, até um prédio preferencial e até dois prédios alternativos, sendo que o prédio preferencial pode ter também um andar preferencial. A figura 8 mostra a modelagem da tabela de preferências.

Figura 8 – Esquema da tabela preferências

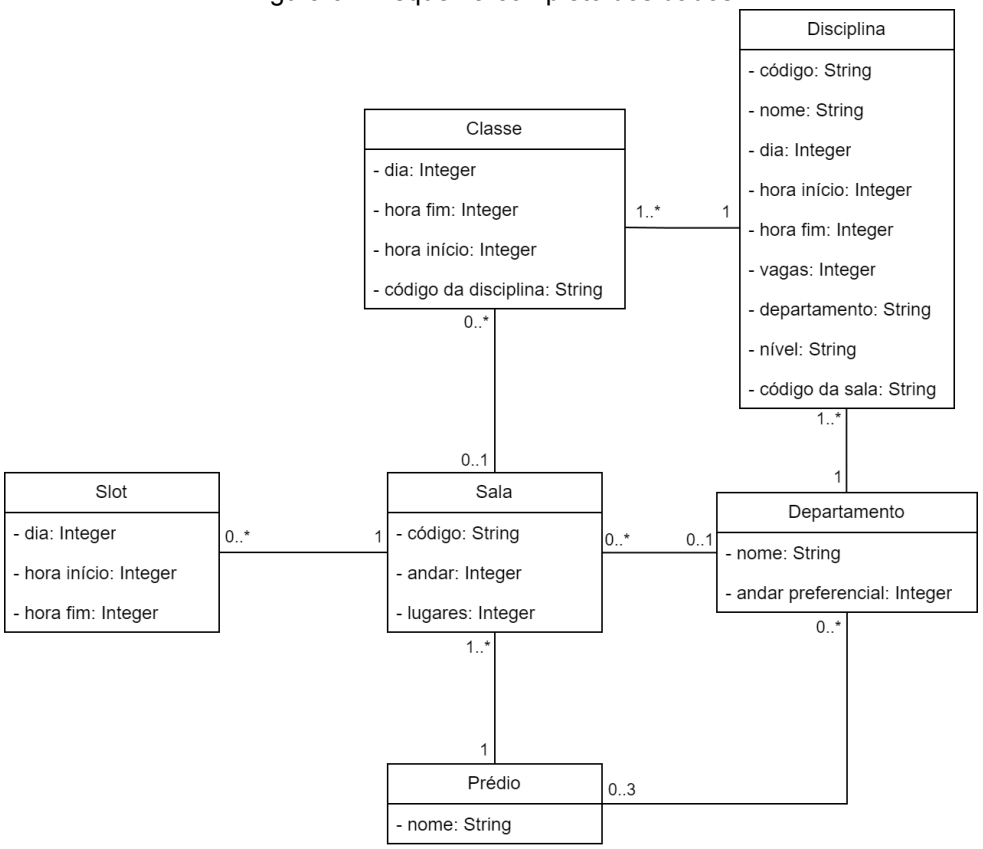


Durante a otimização, o algoritmo primeiro procurar salas disponíveis seguindo as preferências dos departamentos, caso tenham alguma. Se nenhuma sala preferencial estiver disponível, será procurada uma sala em qualquer prédio da PUC-Rio. Para agilizar a procura de salas preferenciais durante a otimização, o arquivo de preferências é convertido em uma matriz departamento x sala, em que cada elemento é uma sequência de bits que indicam a preferência do departamento. O significado de cada bit é:

- bit 0: sala no prédio e andar preferencial
- bit 1: sala no prédio preferencial (em qualquer andar)
- bit 2: sala no primeiro prédio alternativo
- bit 3: sala no segundo prédio alternativo
- bit 4: sala em qualquer prédio

A figura 9 mostra a modelagem completa dos dados enviados pelo usuário.

Figura 9 – Esquema completo dos dados



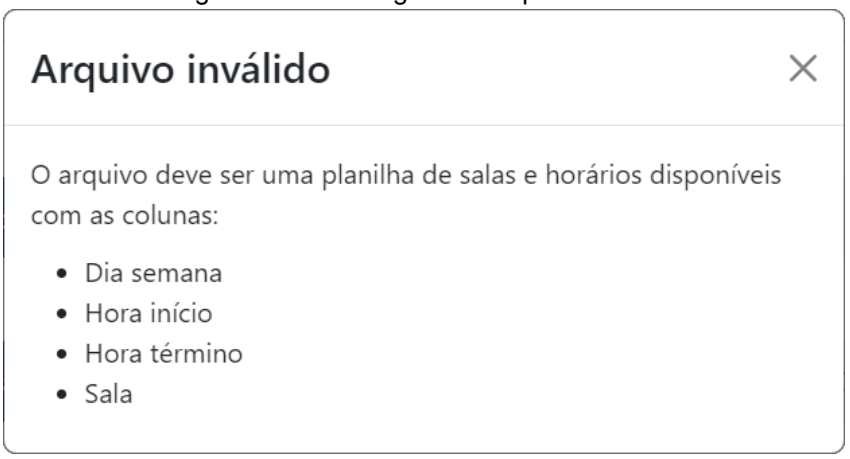
O usuário envia esses arquivos na página inicial do *site*, que contém botões para incluir os arquivos, como mostra a figura 10.

Figura 10 – Interface de envio de arquivos

Após os arquivos serem enviados, eles são verificados para garantir que são do tipo correto (*Excel*) e estão organizados da forma esperada. A figura 11 mostra uma mensagem de erro que aparece caso um arquivo inválido seja

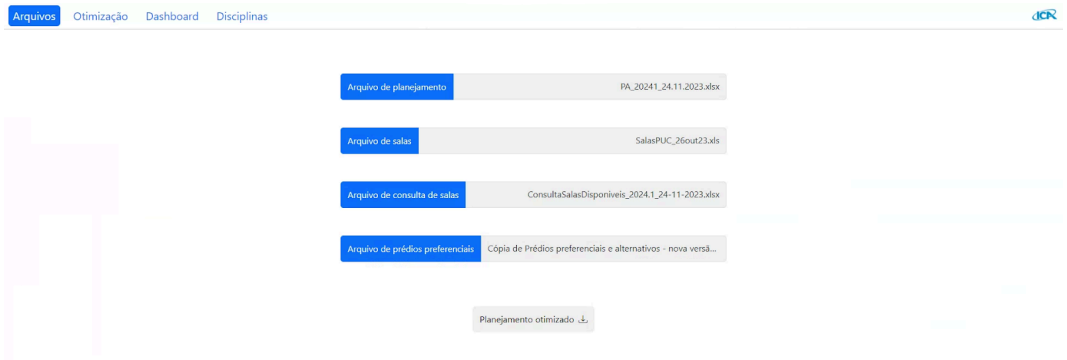
enviado.

Figura 11 – Mensagem de arquivo inválido



A figura 12 mostra a tela de arquivos completa.

Figura 12 – Tela de Arquivos



5.3 Armazenamento

Todos os dados são armazenados no navegador do usuário, de forma que não ocupam espaço no disco do servidor. Esses dados são gerados no servidor, serializados e salvos no armazenamento local do navegador. As informações salvas são:

- Arquivo do planejamento
- Arquivo de prédios e sala
- Arquivo de *slots*
- Arquivo de prédios e andares preferenciais
- Parâmetros da otimização, como quantidade de gerações
- Histórico da otimização, caso já tenha sido feita
- Resultado da otimização, caso já tenha sido feita
- Versão dos dados do usuário, para eventuais atualizações do servidor

A serialização dos dados é feita a partir da biblioteca padrão do *Python pickle*, que serializa qualquer objeto em bytes. Como essa biblioteca executa códigos arbitrários durante a desserialização, ela é considerada insegura se usada sem nenhuma verificação dos dados antes de desserializar. Por isso, foi utilizada outra biblioteca padrão, *hmac*, para autenticar o conteúdo serializado, garantindo que não foi alterado pelo usuário.

5.4 Otimização

Após todos os arquivos serem enviados pelo usuário, a página de otimização ficará disponível, onde é possível otimizar a alocação de salas e

acompanhar o processo, que pode demorar segundos ou poucos minutos, visualizando o progresso da otimização, assim como quanto tempo resta.

A otimização por algoritmo genético foi implementada utilizando a biblioteca *'deap*', que possui alguns métodos prontos, como seleção de indivíduos e alguns cruzamentos e mutações. Outros tipos de cruzamentos e mutações específicos para o problema de planejamento foram implementados:

- *Uniform Order Based Crossover*: reordena partes de um indivíduo baseado em partes de outro indivíduo
- *Cycle Crossover*: gera novos indivíduos baseado em ciclos formados a partir dos indivíduos pais
- *Right/Left Rotate Mutation*: rotaciona indivíduos

Um indivíduo representa uma disposição possível das disciplinas em salas. O DNA do indivíduo é uma sequência de classes, que são alocadas em salas seguindo essa sequência. Ou seja, a primeira classe do DNA será alocada primeiro, na melhor sala possível considerando seus requisitos e preferências, enquanto a última classe do DNA será alocada por último, ou pode até ficar sem sala. O objetivo do algoritmo genético é descobrir uma sequência ótima das classes para que tenha o maior número de classes alocadas em salas de aula adequadas.

Cada indivíduo tem um agendador de classes, que tenta alocar as classes nas melhores salas possíveis, baseado nos requisitos e nas preferências de cada classe. Inicialmente, o agendador está vazio, sem nenhuma classe alocada a nenhuma sala. Para cada classe do DNA do indivíduo, o agendador tenta escolher a sala vazia mais apropriada, e marca essa sala como ocupada no determinado dia e horário. Durante o agendamento é calculado a aptidão do indivíduo, que leva em consideração quantas classes foram alocadas, se as preferências de cada classe foram atendidas e o número de vagas disponibilizadas pela classe (classes com mais vagas têm maior preferência).

O agendamento possui diversas etapas. Em cada etapa, as classes que ainda não possuem sala são alocadas seguindo uma regra. Essa regra inclui qual prédio e andar da sala e se todas as classes de uma disciplina devem ser na mesma sala, para disciplinas com mais de uma classe na semana. As etapas executadas pelo agendador são, em ordem:

1. Alocar todas as classes nos prédios e andares preferenciais, com todas as classes de uma disciplina na mesma sala
2. Alocar as classes restantes nos prédios preferenciais, com todas as

classes de uma disciplina na mesma sala

3. Alocar as classes restantes nos prédios e andares preferenciais, com as classes de uma disciplina em salas distintas
4. Alocar as classes restantes nos prédios preferenciais, com as classes de uma disciplina em salas distintas
5. Alocar as classes no prédio alternativo 1, com todas as classes de uma disciplina na mesma sala
6. Alocar as classes no prédio alternativo 1, com as classes de uma disciplina em salas distintas
7. Alocar as classes no prédio alternativo 2, com todas as classes de uma disciplina na mesma sala
8. Alocar as classes no prédio alternativo 2, com as classes de uma disciplina em salas distintas
9. Alocar as classes em qualquer sala disponível, com todas as classes de uma disciplina na mesma sala
10. Alocar as classes em qualquer sala disponível, com as classes de uma disciplina em salas distintas

Como nem todos os departamentos possuem prédios preferenciais e alternativos, durante o agendamento, algumas etapas podem ser puladas por algumas classes. Durante o agendamento, a aptidão é calculada baseado em quantas classes foram alocadas em cada etapa. A primeira etapa possui a maior aptidão, já que atende a todas as preferências dos departamentos, enquanto a última etapa possui a menor aptidão. Ao final é descontado da aptidão o número de classes sem sala, caso haja alguma.

Os primeiros indivíduos, que farão parte da primeira geração, são criados aleatoriamente, com a exceção de um, que terá como DNA as classes em ordem decrescente do número de vagas oferecidas. Esse indivíduo é uma boa estimativa inicial, pois aloca primeiro as classes com mais vagas, que têm menos salas com capacidade disponíveis.

A cada geração, são aplicados operadores de seleção, cruzamento e mutação para gerar os indivíduos da geração seguinte. Para isso, são selecionados indivíduos a partir do operador de torneio, que escolhe aleatoriamente alguns indivíduos e “sobrevive” aquele com maior aptidão. Com os indivíduos selecionados, são aplicados os cruzamentos uniforme baseado em ordem, parcialmente correspondido, ordenado e cíclico. As mutações feitas nos

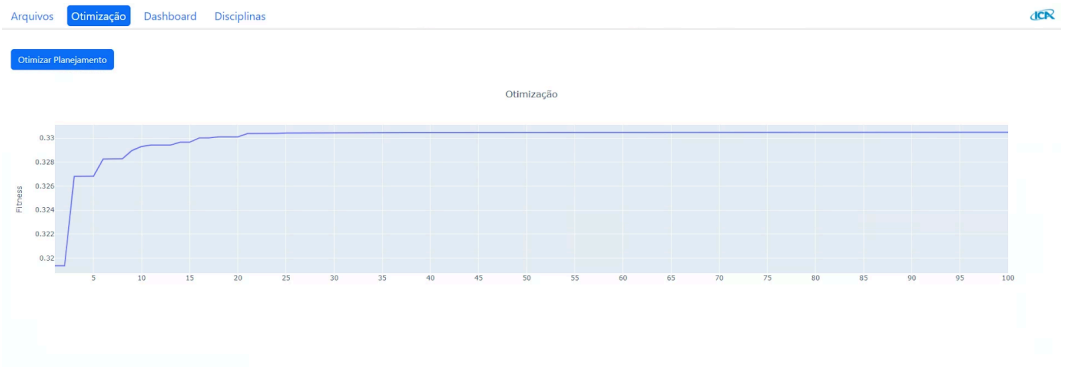
indivíduos são de aleatorização de índices e rotação para a esquerda e para a direita. Esses operadores são adequados para algoritmos de agendamento, pois garantem que todas as classes estarão presentes nos indivíduos gerados.

Após essas etapas, os novos indivíduos passam a fazer parte da geração seguinte, e a elite é atualizada, mantendo os melhores indivíduos de todas as gerações. Além disso, são calculadas métricas a cada geração, como melhor aptidão da geração e média das aptidões, para que o usuário possa acompanhar o progresso da otimização. Ao final, o melhor indivíduo de todas as gerações é retornado, contendo a alocação ótima das classes, assim como os novos *slots* das salas desocupadas.

Como o algoritmo genético é bastante caro, com relação ao uso de CPU, a otimização é feita de forma paralela, de forma que as aptidões dos indivíduos é calculada em processos diferentes, utilizando todos os núcleos do processador.

A figura 13 mostra a tela de otimização, com um gráfico para acompanhar o progresso e o tempo restante.

Figura 13 – Tela de otimização



Durante a otimização, o usuário pode sair do *site* e retornar quando quiser, além de poder para a otimização. Ao final, as abas de *dashboard* e disciplinas ficarão disponíveis.

5.5 *Dashboard*

O *dashboard* é uma tela com gráficos e tabelas com o objetivo de visualizar os resultados obtidos após o término da otimização.

A figura 14 mostra uma tabela contendo o número de disciplinas alocadas, por prédio e andar preferencial, ou em qualquer prédio, e o número de assentos alocados por departamento.

Figura 14 – Tabela de departamentos

Departamento	Disciplinas Alocadas ▲	Disciplinas no prédio preferencial	Disciplinas no andar preferencial	Assentos Alocados
UEP	21 / 21 (100.00%)	16 / 21 (76.19%)	16 / 21 (76.19%)	733 / 941 (77.90%)
EXT	4 / 4 (100.00%)	0 / 4 (0.00%)	0 / 4 (0.00%)	80 / 88 (90.91%)
COM	583 / 583 (100.00%)	557 / 583 (95.54%)	557 / 583 (95.54%)	10361 / 17485 (59.26%)
CPE	331 / 331 (100.00%)	280 / 331 (84.59%)	280 / 331 (84.59%)	7840 / 16546 (47.38%)
FLE	31 / 31 (100.00%)	31 / 31 (100.00%)	31 / 31 (100.00%)	1620 / 3100 (52.26%)
FIS	25 / 25 (100.00%)	24 / 25 (97.14%)	24 / 25 (97.14%)	604 / 836 (82.01%)
Total	**4292 / 4601 (93.28%)**	**3653 / 4601 (79.40%)**	**2856 / 4601 (62.07%)**	**121480 / 195666 (62.09%)**

É possível também analisar a quantidade de disciplinas necessitando sala por dia e hora, podendo ser usado como referência para entender quais horários são mais críticos, com uma linha vermelha indicando o número de total de salas disponíveis, como mostram as figuras 15 e 16.

Figura 15 – Gráfico de disciplinas por hora (7h-11h)

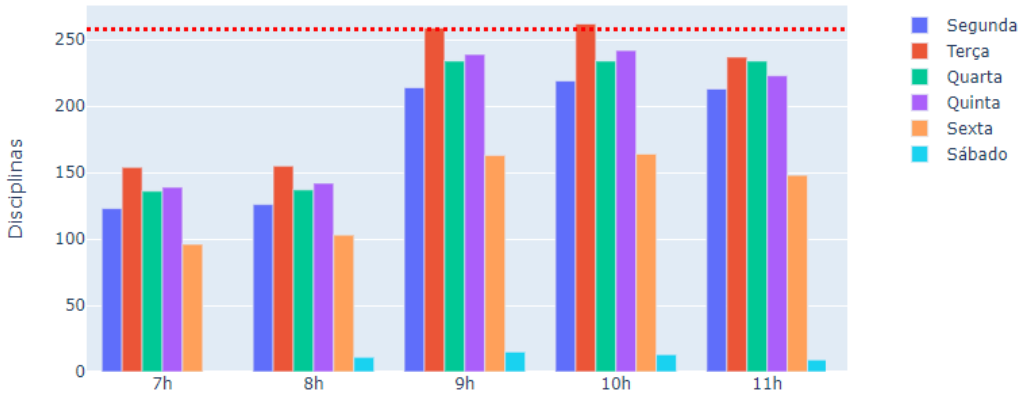
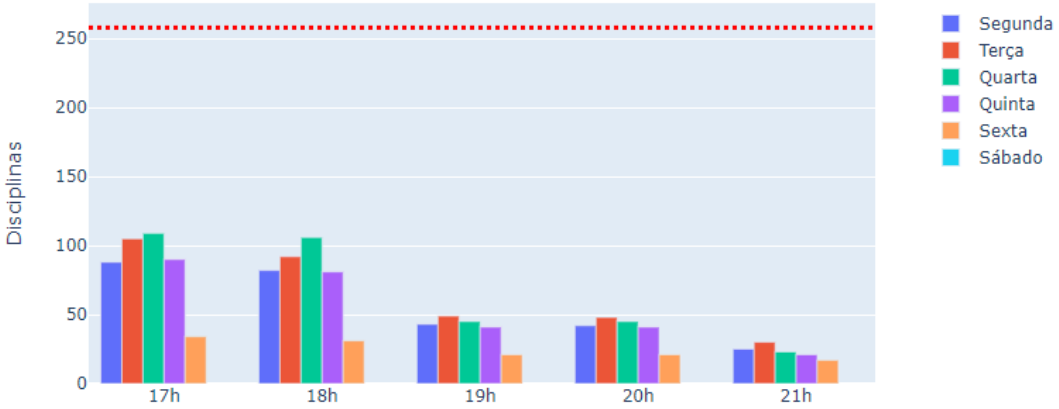


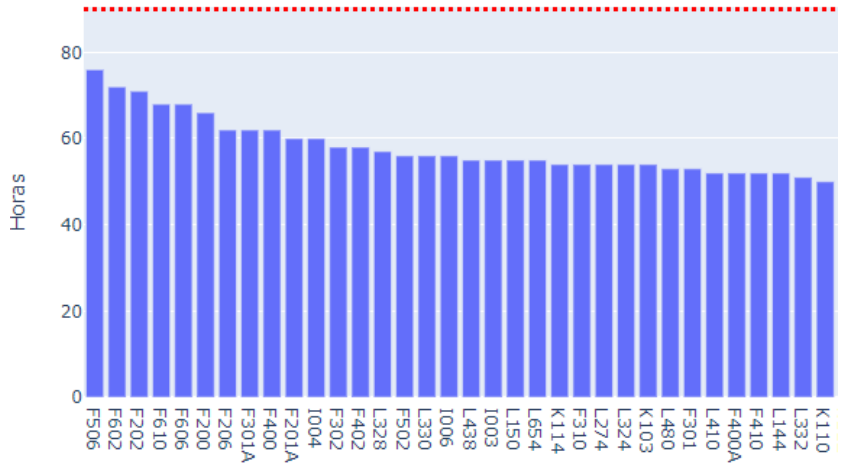
Figura 16 – Gráfico de disciplinas por hora (17h-21h)



Outra métrica disponível é a quantidade de horas alocadas por sala na

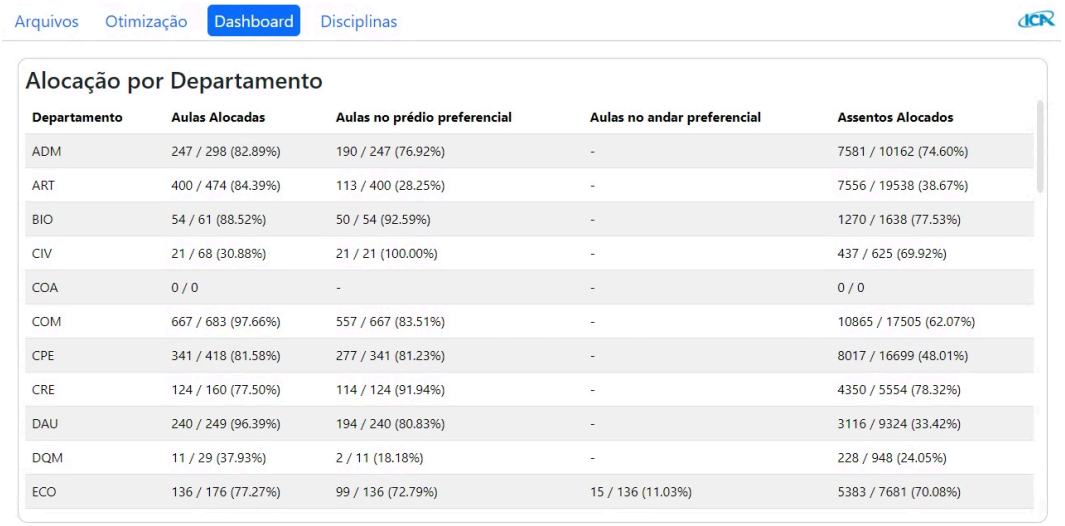
semana, indicando quais salas são mais utilizadas e quais estão livres. A figura 17 mostra as salas mais utilizadas durante a semana.

Figura 17 – Gráfico de ocupação de salas



A figura 18 mostra a tela de dashboard completa.

Figura 18 – Tela de *Dashboard*



5.6 Disciplinas

A página de disciplinas mostra todas as disciplinas incluídas no arquivo de planejamento, com todas as suas informações, como nome, dia, hora e sala. Caso a disciplina não tenha sido otimizada, a disciplina é associada à sala presente no arquivo de planejamento. É possível pesquisar classes específicas para que o usuário altere manualmente a sua sala. A figura 19 mostra opções de filtragem de disciplinas, e a figura 20 mostra o resultado da filtragem.

Figura 19 – Opções de filtragem de disciplina

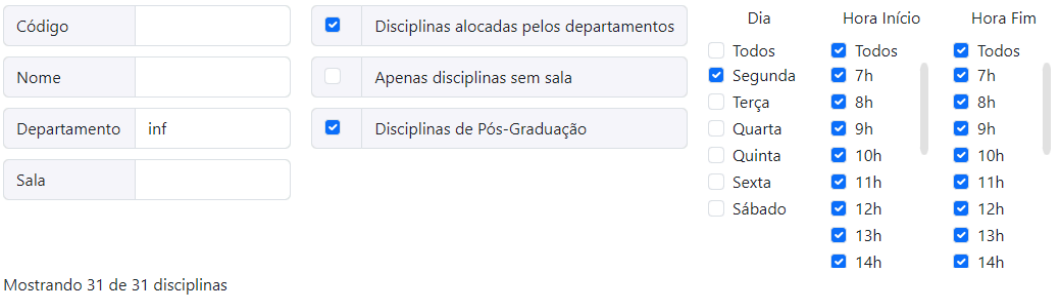
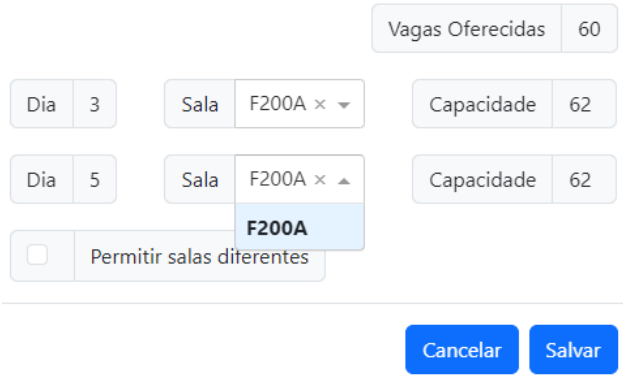


Figura 20 – Resultado da filtragem de disciplinas

Nivel	Centro	Dept.	Código	Nome da Disciplina	Turma	Professor	Dia	Hora Início	Hora Fim	Sala	Capacidade da Sala	Vagas OF	Compartilhada
GRA	3	INF	INF1009	LOGICA PARA COMPUTACAO	3WB		2	13:00	15:00	L481	42	40	N
GRA	3	INF	INF1010	ESTRUTURAS DE DADOS AVANCADAS	3WA		2	09:00	11:00	L546	43	35	N
GRA	3	INF	INF1010	ESTRUTURAS DE DADOS AVANCADAS	3WB		2	15:00	17:00	L546	43	35	N
GRA	3	INF	INF1013	MODELAGEM DE SOFTWARE	3WA		2	17:00	19:00	VIRTUAL	0	35	N
GRA	3	INF	INF1022	ANALISADORES LEXICOS E SINTATICOS	3WA		2	11:00	13:00	L278	48	30	N
GRA	3	INF	INF1022	ANALISADORES LEXICOS E SINTATICOS	3WB		2	09:00	11:00	L654	62	30	N
GRA	3	INF	INF1025	INTRODUCAO A PROGRAMACAO	3WA		2	09:00	11:00	RDC	25	30	N
GRA	3	INF	INF1027	TESTE E MEDICAO DE SOFTWARE	3WA		2	09:00	11:00	L160	62	40	N
GRA	3	INF	INF1032	INTRODUCAO A CIENCIA DOS DADOS	3WA		2	07:00	09:00	L278	48	40	N
GRA	3	INF	INF1034	PROJETO: PRÁTICAS DE PROGRAMAÇÃO	3WA		2	11:00	13:00	L546	43	40	N
GRA	3	INF	INF1034	PROJETO: PRÁTICAS DE PROGRAMAÇÃO	3WB		2	11:00	13:00	L540	40	40	N

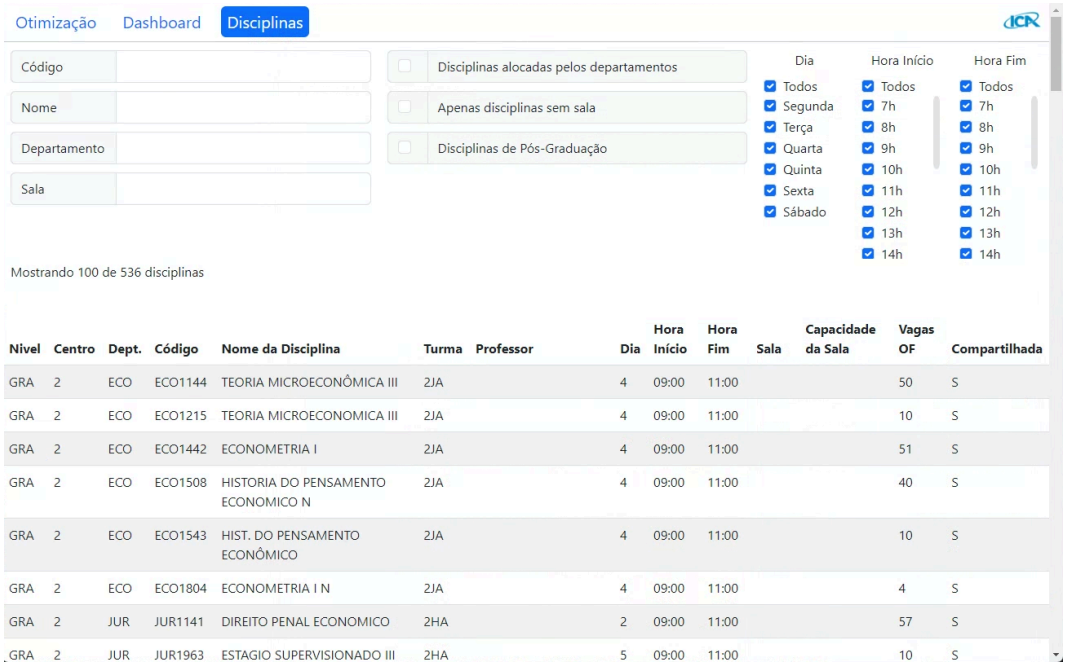
Ao selecionar uma disciplina, é mostrada uma lista com todas as classes dessa disciplina com suas respectivas salas. É possível alterar as salas de cada classe individualmente ou em conjunto, em que todas as classes serão movidas para a mesma sala. Ao clicar no campo “sala”, um *dropdown* é mostrado com as salas disponíveis no horário da aula. Apenas salas com capacidade maior ou igual ao número de vagas oferecidas pela disciplina aparecem na lista. Além disso, essas salas são ordenadas por prioridade, ou seja, salas no prédio e andar preferencial da disciplinas aparecerão no começo da lista. A figura 21 mostra a tela de mudança de sala.

Figura 21 – Mudança de sala manual



A figura 22 mostra a tela de mudança de salas completa.

Figura 22 – Tela de disciplinas



5.7 Resultados

Quando o usuário terminar a otimização e as modificações manuais, poderá baixar o resultado em formato *Excel* idêntico ao formato do arquivo de disciplinas, com duas colunas adicionais. Esses colunas são:

- Alocado pelos departamento: indica se a disciplina já havia sido alocada pelo departamento antes da otimização
- Alocado automaticamente: indica se a disciplina passou pelo processo de otimização, mesmo que, ao final, tenha ficado sem sala

6. Implementação e Avaliação

O sistema foi testado utilizando dados reais de disciplinas e salas de semestres passados. Durante os testes, o feedback contínuo da CCG foi fundamental para identificar pontos fortes e áreas de melhoria. Esse processo iterativo de avaliação e ajuste permitiu aprimorar o software para atender melhor às necessidades específicas dos usuários finais. Além dos testes retrospectivos, o sistema foi utilizado no semestre atual (2024.1) para alocar parte das disciplinas. Esta implementação prática foi crucial para validar a aplicabilidade e robustez do sistema em um ambiente de produção. Durante este período, a utilização do sistema foi monitorada de perto para garantir que ele funcionasse conforme esperado.

Os resultados obtidos até o momento indicam um alto nível de satisfação entre os usuários. Não houve reclamações significativas por parte da CCG nem dos departamentos envolvidos com relação às salas de aula alocadas, sugerindo que o sistema conseguiu atender ou até mesmo superar as expectativas. A alocação automática proporcionou uma distribuição eficiente das disciplinas, respeitando as restrições e preferências especificadas, e resultou em uma utilização otimizada dos espaços disponíveis.

Esses resultados positivos refletem o potencial do sistema para substituir o processo manual, aumentando significativamente a eficiência e precisão da alocação de salas de aula. O sucesso da implementação no semestre atual serve como um forte indicativo de que o sistema pode ser expandido para alocar todas as disciplinas em semestres futuros, trazendo benefícios substanciais em termos de economia de tempo e redução de erros.

7. Trabalhos Futuros

Durante o desenvolvimento do sistema, diversas funcionalidades adicionais foram sugeridas para serem implementadas futuramente. Entre elas estão:

- **Integração com o sistema da CCG:** Uma das principais melhorias propostas é a integração direta com o sistema da CCG. Essa integração permitiria que o acesso aos arquivos de disciplinas e salas fosse feito automaticamente, eliminando a necessidade de inserção manual dos arquivos no site. Isso não apenas reduziria o tempo e o esforço necessários para configurar o sistema, mas também minimizaria o risco de erros humanos na entrada de dados. A automatização deste processo tornaria o sistema mais robusto e amigável para os usuários.
- **Otimização de salas das disciplinas com sala compartilhada:** Atualmente, o sistema é projetado para alocar salas de aula para disciplinas que não compartilham salas. No entanto, muitas disciplinas na universidade utilizam salas compartilhadas, o que adiciona um nível extra de complexidade à alocação. A inclusão de funcionalidades que permitam a otimização de salas compartilhadas ajudaria a garantir que essas disciplinas também sejam alocadas de forma eficiente, respeitando as necessidades de todas as partes envolvidas.
- **Otimização de salas que atualmente são escolhidas pelos departamentos:** Outra melhoria significativa seria a capacidade de otimizar as salas que já foram escolhidas pelos departamentos. Atualmente, o sistema se concentra em alocar disciplinas que não têm sala definida. A expansão do sistema para incluir a reavaliação e otimização das salas escolhidas pelos departamentos poderia levar a uma utilização ainda mais eficiente dos espaços disponíveis, considerando também a possibilidade de trocas benéficas que poderiam atender melhor às necessidades de todos os departamentos.
- **Interface de usuário melhorada**: Melhorias na interface de usuário para tornar a interação com o sistema mais intuitiva e eficiente. Isso incluiria a criação de ferramentas visuais para facilitar a análise e a modificação manual das alocações de salas, permitindo que os usuários façam ajustes rápidos e informados com base nos resultados da otimização.

8. Referências

BELIGIANNIS, G.; MOSCHOPOULOS C.; LIKOTHANASSIS S. A genetic algorithm approach to school timetabling. J Oper Res Soc, v. 60, p. 23-42, 2009.

CHEN, Xiangliu ; YUE, Xiao-Guang. et al. Design and Application of an Improved Genetic Algorithm to a Class Scheduling System. International Journal of Emerging Technologies in Learning (iJET), v. 16, n. 1, p. 44-59, 2021.

DEAP. Deap, 2023. Documentação. Disponível em <https://deap.readthedocs.io/>. Acesso em 23 de out. de 2023.

HOLLAND, John H. Genetic Algorithms. Scientific American, v. 267, n. 1, p. 66-73, Jul. 1992.

INFORMATION TECHNOLOGY. ISO/IEC 19501:2005: Open Distributed Processing - Unified Modeling Language (UML). 2005.

MULTIPROCESSING. Python, 2024. Documentação. Disponível em <https://docs.python.org/3/library/multiprocessing.html>. Acesso em 12 de mai. de 2024.

NUMPY. Numpy, 2024. Página Inicial. Disponível em <https://numpy.org/>. Acesso em 12 de mai. de 2024.

PLOTLY. Dash, 2023. Guia de Usuário e Documentação. Disponível em: <https://dash.plotly.com/>. Acesso em 23 de out. de 2023.

PUC-RIO. PUC-Rio, 2023. Página Inicial. Disponível em <https://www.puc-rio.br/index.html>. Acesso em 23 de out. de 2023.

PYTHON. Python, 2023. Página Inicial. Disponível em <https://www.python.org/>. Acesso em 23 de out. de 2023.