SERVIÇO PÚBLICO FEDERAL

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

PRÓ-REITORIA DE ENSINO

CÂMPUS GOIÂNIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DIEGO TALIATELI DO COUTO

FELIPE CRISPIM PAULINO

Estudo do problema de alocação de salas aplicado ao IFG - Câmpus Goiânia

Goiânia, 2019.

INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Diego Tabatelli do Couto e Felipe Crispim Paulino
Matrícula: 20131011090303 20131011090087
Título do Trabalho: Estudo do problema de alocação de salas aplicado ao IFG - Câmpus Goiânia

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG;
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____;
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG.

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Goiânia, 09/07/19.
Local Data

Diego Tabatelli do Couto Felipe Crispim Paulino
Assinatura do Autor e/ou Detentor dos Direitos Autorais

SERVIÇO PÚBLICO FEDERAL

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

PRÓ-REITORIA DE ENSINO

CÂMPUS GOIÂNIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DIEGO TALIATELI DO COUTO

FELIPE CRISPIM PAULINO

Estudo do problema de alocação de salas aplicado ao IFG - Câmpus Goiânia

Trabalho de Conclusão apresentado à Coordenação do Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Goiás, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Raphael de Aquino Gomes

Goiânia, 2019.

C8373e Couto, Diego Taliateli do.
Estudo do problema de alocação de salas aplicado ao IFG – Câmpus Goiânia / Diego Taliateli do Couto; Felipe Crispim Paulino – Goiânia: Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia, 2019.
65 f. : il.

Orientador: Prof. Dr. Raphael de Aquino Gomes.

TCC (Trabalho de Conclusão de Curso) – Curso Superior de Bacharelado em Sistemas de Informação, Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia.
Inclui apêndice.

1. Algoritmos. 2. Alocação de salas - Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) - Câmpus Goiânia. 3. Simulated Annealing. I. Paulino, Felipe Crispim. II. Gomes, Raphael de Aquino (orientador). III. Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia. IV. Título.

CDD 005.1

Ficha catalográfica elaborada pelo Bibliotecário Alisson de Sousa Belthodo Santos CRB1/ 2.266
Biblioteca Professor Jorge Félix de Souza,
Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia.

SERVIÇO PÚBLICO FEDERAL

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

PRÓ-REITORIA DE ENSINO

CÂMPUS GOIÂNIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DIEGO TALIATELI DO COUTO

FELIPE CRISPIM PAULINO

Estudo do problema de alocação de salas aplicado ao IFG - Câmpus Goiânia

Trabalho de Conclusão apresentado à Coordenação do Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Goiás como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação, aprovada em 02 de Julho de 2019, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Raphael de Aquino Gomes
Departamento IV – IFG / Câmpus Goiânia
Presidente da Banca

Prof. Dr. Sirlon Diniz de Carvalho
Departamento IV – IFG / Câmpus Goiânia

Prof. Me. Carlos Augusto da Silva Cabral
Departamento IV – IFG / Câmpus Goiânia

Dedicatória

Eu Diego, dedico este trabalho de conclusão de curso primeiramente à minha esposa Ana Priscilla que desde o início da graduação sempre me apoiou, me motivou a chegar até o fim apesar de todas as dificuldades do percurso. Também dedico ao meu falecido pai Divino que com certeza estaria muito feliz em ver seu filho estudando e concluindo essa etapa da vida, e à minha mãe Tuca por ter criado eu e meu irmão sozinha não deixando que nada nos faltasse.

Eu, Felipe, dedico este trabalho de conclusão de curso à minha noiva Eudiane, minha mãe Olinda, meu pai Homero, meu padrinho Arinaldo e todos os meus amigos que me ajudaram nessa trajetória no IFG.

Agradecimentos

Eu Diego, agradeço primeiramente à Deus por conceder forças para seguir em frente e não desistir de concluir esta etapa da minha vida. Agradeço ao nosso orientador Raphael que sempre foi muito prestativo e que fez muito mais do que esperávamos para nos apoiar a entregar um bom trabalho. E também agradeço aos meus amigos Kellerman e Ramon que impulsionaram minha carreira na área de TI sempre muito pacientes com as minhas dificuldades, e por fim agradeço à minha esposa Ana Priscilla que teve um papel fundamental de suporte, apoio e fé no meu potencial me mantendo engajado com meu propósito de concluir a graduação.

Eu, Felipe, agradeço primeiramente ao nosso orientador Raphael pela paciência e presteza na orientação deste trabalho. Agradeço a minha noiva Eudiane, que me incentivou a enfrentar esse desafio de realizar uma graduação e esteve sempre ao meu lado em toda a trajetória. E também agradeço a todos os amigos que fiz no IFG que acreditaram no meu potencial e me ajudaram chegar até aqui.

"O sucesso é ir de fracasso em fracasso sem perder entusiasmo."

Winston Churchill,
1874-1965.

Resumo

Título: Estudo do problema de alocação de salas aplicado ao IFG - Câmpus Goiânia
Autores: Diego Taliateli do Couto e Felipe Crispim Paulino
Orientador: Dr. Raphael de Aquino Gomes

O presente estudo aborda o Problema de Alocação de Salas na realidade do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) - Câmpus Goiânia e as técnicas utilizadas para solucioná-lo. O problema consiste em alocar turmas com horários pré-definidos em salas de aula que possam atender melhor os seus requisitos, e isso deve ser feito em um tempo de processamento satisfatório. Portanto, fizemos um estudo acerca das principais meta-heurísticas utilizadas em problemas de designação e otimização combinatória, e elegemos aquela que entendemos ser a que melhor se aplicaria na resolução do problema. Foi desenvolvida uma solução computacional que gera uma solução inicial a partir do algoritmo guloso e aplica a meta-heurística Simulated Annealing para o refinamento da mesma. Os dados reais utilizados no trabalho foram fornecidos pela gestão do IFG, os mesmos apresentam informações de salas e turmas. O trabalho se propõe a validar se a solução apresentou otimizações nas alocações de salas em relação às alocações manuais feitas atualmente.

Palavras–chave

Problema de Alocação de Salas, Metaheurísticas, Simulated Annealing

Abstract

Title: Solution to the room allocation problem applied to the Instituto Federal de Goiás
Authors: Diego Taliateli do Couto and Felipe Crispim Paulino
Adviser: Dr. Raphael de Aquino Gomes

The present study deals with the Problem of Room Allocation in the reality of the Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) - Câmpus Goiânia and the techniques used to solve it. The problem is to allocate pre-defined classes in classrooms that can best meet your requirements, and this must be done in a satisfactory processing time. Therefore, we did a study about the main meta-heuristics used in combinatorial designation and optimization problems, and chose the one that we consider to be the one that would best apply in solving the problem. We developed a computational system that generates an initial solution from the greedy algorithm and applies the Simulated Annealing meta-heuristic to the solution refinement. The data used are actual data provided by the IFG management that present room and class information. The paper proposes to validate that the system presented optimizations in room allocations in relation to the manual allocations currently made.

Keywords

Classroom Assigment Problem, Metaheuristics, Simulated Annealing

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Códigos de Programas

Lista de Abreviaturas e Siglas

AG	Algoritmos Genéticos
GAAAE	Gerência de Administração e Acadêmica e Apoio ao Ensino
GRASP	Greedy Randomized Adaptive Search Procedures
IFG	Instituto Federal de Goiás
NP	Tempo polinomial não determinístico
PAS	Problema de Alocação de Salas
SA	Simulated Annealing

Sumário

CAPÍTULO 1

Introdução

Antes do início de cada semestre nas universidades existe a necessidade de alocar as salas para as turmas, onde na maioria das vezes esse processo é feito de forma manual ou através de planilhas. Além de ser bastante trabalhoso fazer a alocação dessa forma, geralmente a alocação não fica otimizada, desperdiçando recursos ou não atendendo corretamente a quantidade de vagas requerida para cada turma. Isso acontece pois, dependendo do número de combinações, uma solução manual não vai conseguir contemplar todas as exigências. O resultado pode provocar insatisfação tanto de alunos quanto de professores, e demais envolvidos (coordenação, monitores, etc).

Segundo [Schaerf 1999] a distribuição de aulas, com seus horários estabelecidos previamente para as salas que serão usadas, sendo necessário respeitar um conjunto de restrições de diversas naturezas se caracteriza como Problema de Alocação de Salas (PAS). PAS é um problema conhecido e bastante estudado na computação, mas que não possui uma solução ótima, já que a complexidade apenas tende a crescer de acordo com o número de entradas, gerando assim mais possibilidades combinatórias.

Assim como em outras instituições, no Instituto Federal de Goiás, Ciência e Tecnologia de Goiás (IFG) - Campus Goiânia há a necessidade de otimizar a gestão de salas e recursos, pois atualmente essa alocação é feita de forma manual, não trazendo resultados otimizados e demandando muito tempo para ser formulada.

O Câmpus Goiânia do IFG é dividido em quatro departamentos, sendo que cada um deles possui suas salas e laboratórios exclusivos. As coordenações dos cursos de cada departamento são responsáveis por controlar a alocação e utilização dos espaços disponíveis. Caso uma coordenação queira solicitar uma sala/laboratório de outro departamento, deve-se enviar um memorando para o responsável por aquele local no outro departamento.

O instituto possui salas com diversos tamanhos e formatos e também variedade de recursos, sendo que, cada turma possui disciplinas com várias especificidades, necessitando de salas com recursos mínimos, para que as aulas sejam ministradas com qualidade. Outro ponto é a otimização da capacidade das salas, onde é importante a alocação das turmas em salas que satisfaçam a capacidade de alunos e também evitando desperdício de vagas.

A sugestão para essa necessidade vem da pesquisa de uma solução de alocação de salas que realize o emparelhamento da turma com a sala de forma automatizada e otimizada. Para isto, abordaremos ao longo do trabalho os diversos estudos encontrados, com o objetivo de identificar a melhor forma de implementar a solução de alocação de salas no cenário do IFG - Câmpus Goiânia. O objetivo do trabalho é a solução do PAS do IFG - Câmpus Goiânia, sendo que para resolver forma computacional, será desenvolvido um protótipo de software.

Neste trabalho apresenta-se todo o processo de investigação da realidade do IFG, busca por dados reais do campus com pesquisa *in loco*, um estudo sobre o PAS e as técnicas que resolvem a questão, a eleição da melhor solução para o nosso problema, o passo a passo da criação de um protótipo baseado na técnica escolhida, o funcionamento geral do protótipo e as etapas do processo de alocação e refinamento da solução. Também será feita uma avaliação da nova solução comparando os resultados obtidos, além de propormos sugestões para novos trabalhos.

1.1 Objetivos

Nesta seção são apresentados os objetivos deste trabalho.

1.1.1 Objetivo Geral

Este trabalho tem por objetivo geral desenvolver uma solução computacional que busca solucionar o problema de alocação de salas no Instituto Federal de Goiás - Campus Goiânia, baseado nos estudos e trabalhos acadêmicos acerca do tema. Procuraremos relatar todo o processo de conhecimento adquirido sobre métodos de otimização combinatória que possibilitou a construção da solução e os resultados obtidos.

1.1.2 Objetivos Específicos

- Vivenciar o cotidiano inerente ao problema do instituto, através de entrevistas e obtenção de dados com os servidores.
- Realizar uma pesquisa buscando o máximo de trabalhos que abordam o tema e também temas relacionados, assim como os métodos utilizados na solução.
- Identificar uma solução do PAS que seja possível adaptar e aplicar ao cenário do IFG - Câmpus Goiânia.
- Realizar uma modelagem do problema através de dados fornecidos pelo IFG - Câmpus Goiânia.
- Aplicar uma solução baseada no algoritmo elegido durante este trabalho.

- Apresentar o protótipo (núcleo do sistema) que permite solucionar o problema através do algoritmo adotado.
- Testar e analisar a solução obtida através do protótipo.
- Identificar pontos de melhoria e apresentar ideias para trabalhos futuros.

1.2 Metodologia

Para entender melhor sobre o cenário do IFG - Campus Goiânia, foram realizadas entrevistas com o gerente de Administração e Acadêmica e Apoio ao Ensino (GAAAE) do Câmpus Goiânia. O mesmo nos explicou que a alocação era realizada usando planilha a cada início de semestre letivo, onde cada departamento ficava responsável por alocar as turmas da sua grade. Foi nos fornecido a planilha que contém a lista de salas do IFG com informações relevantes para o desenvolvimento do nosso trabalho, como capacidade, bloco e recursos disponíveis. As mesma está contida no Apêndice A.

A pesquisa referente ao PAS foi realizada em literaturas correlatas ao tema. O objetivo foi entender mais sobre o problema e encontrar a solução ideal para o nosso cenário, levando em conta as características do mesmo.

Como apresentaremos no Capítulo 2 os métodos para solução do problema de alocação de salas trata-se de meta-heurísticas que tem o objetivo de chegar o mais próximo da solução ótima. Para isso selecionamos os principais algoritmos dentre vários de meta-heurística que mais foram estudados e utilizados para problemas de otimização combinatória como o PAS. Com a realização do estudo serão comparados os algoritmos levando em consideração restrições funcionais e não funcionais e com isso elegeremos o método mais eficiente para nosso cenário.

Feito o estudo das soluções já desenvolvidas para este problema, pretendemos utilizar os conceitos e adaptá-los ao nosso cenário desenvolvendo uma solução específica para o IFG utilizando dados fornecidos pela gestão do Câmpus, regras e restrições obtidos através de entrevistas e análises. A solução desenvolvida deve realizar a alocação de salas em um tempo satisfatório e validá-las com alocações anteriores garantindo que a solução automatizada seja mais assertiva e realize de fato uma otimização combinatória.

1.3 Resultados esperados

Este trabalho tem como resultado esperado a produção de um núcleo de sistema que seja capaz de solucionar e otimizar o processo de alocação de salas do IFG - Câmpus Goiânia. O mesmo terá capacidade de alocar as turmas nas salas de maneira otimizada, se baseando em soluções já consideradas na literatura.

Esperamos que a solução apresente alocações comprovadamente melhores que as alocações manuais feitas anteriormente de acordo com a função objetiva definida. De maneira complementar, esperamos um tempo de execução aceitável e um custo mínimo para operação do sistema. Do mesmo modo, almeja-se com os conhecimentos adquiridos poder evoluir a solução operacionalmente e, posteriormente, atacar também os demais subproblemas do PAS.

1.4 Organização do trabalho

Este trabalho está dividido em 6 capítulos, no Capítulo 2 discorremos sobre os estudos realizados acerca do PAS referenciando os principais autores e trabalhos científicos relacionados ao tema, e as estratégias utilizadas, além de fazermos um breve resumo de cada meta-heurística relevante ao problema. No Capítulo 3 vamos apresentar o cenário do IFG relatando todas especificidades, desenhar a modelagem do problema e eleger uma técnica para refinamento da solução. No Capítulo 4 vamos descrever o fluxo da solução, as regras envolvidas, códigos desenvolvidos e apresentar como foi aplicado as técnicas elegidas na solução. No Capítulo 5 vamos validar os resultados gerados pelo sistema com as alocações feitas anteriormente de forma manual, demonstrando as melhorias e ganho com a solução computacional. No Capítulo 6 vamos resumir o esforço do trabalho apontando as principais dificuldades, as lições aprendidas e os trabalhos futuros que pretendemos realizar a partir deste problema.

CAPÍTULO 2

Fundamentação teórica

Com base nas pesquisas sobre o problema de alocação de salas, nos deparamos com diversos estudos sobre problemas de agendamento de horários e emparelhamento, dentre eles o PAS. Identificamos que é um assunto bastante estudado e que possui uma vasta quantidade de obras literárias a respeito. Neste capítulo, vamos abordar os conceitos obtidos através da pesquisa sobre Timetabling e seu sub-problema tema deste trabalho, o PAS, assim como as técnicas para solução destes problemas que abrange o nicho de pesquisa do trabalho, sendo elas as heurísticas. Vamos discorrer sobre as definições de cada heurística buscando entender a relação com a complexidade do problema, e ampliar os conhecimentos para futuros problemas que estas técnicas possam vir a contribuir com a solução.

2.1 Timetabling

Timetabling, em Pesquisa Operacional, se refere à alocação horária de recursos. [Wren 1995] deu a seguinte definição: "*Timetabling* é a designação, sujeito a restrições, na entrega de recursos a objetos que estão sendo alocados em um determinado espaço de tempo, de tal modo que se consiga satisfazer, da melhor maneira possível, um conjunto de objetivos desejáveis."

Para [Qu et al. 2009], o problema também é conhecido como *educational timetabling*, *nurse scheduling*, *sports timetabling*, e também *transportation timetabling*. Segundo [Schaerf 1999], *educational timetabling* "consiste no agendamento de uma sequência de aulas entre professores e alunos em um período prefixado de tempo (tipicamente a semana) satisfazendo um conjunto de restrições de vários tipos". O tema é bastante relevante, sendo muito estudado ao longo dos últimos 25 anos [Alvarez-Valdes, Crespo e Tamarit 2002].

Para [Schaerf 1999] o *timetabling* é classificado em três classes:

1. *school timetabling*: programação semanal dos horários das turmas, para evitar que as turmas e professores tenham duas aulas no mesmo horário;

2. *course timetabling*: programação semanal de horários de disciplinas para todos os períodos dos cursos universitários, diminuindo sobreposição de disciplinas nos cursos que tem alunos em comum;
3. *examination timetabling*: problemas de provas para cursos universitários em que se busca evitar sobreposições de datas e provas de disciplinas que possuem estudantes em comum e também distanciar as datas das provas dos estudantes o máximo possível.

2.2 Problema de Alocação de Salas (PAS)

O problema de alocação de salas (PAS) é referente à distribuição de aulas, em que se tem horários estabelecidos previamente para as salas que serão usadas e onde deve se respeitar um conjunto de restrições de diversas naturezas [Schaerf 1999]. A alocação de salas faz parte do problema de programação de cursos universitários (*course timetabling*) [Bardadym 1996].

Em PAS se considera que as disciplinas e horários disponíveis já estejam definidos, sendo que o problema seja alocar as turmas em salas de aula. A solução manual dessa questão pode ser complexa e necessitar de um longo tempo de trabalho e ainda não ser otimizada de maneira satisfatória, causando problemas como o grande fluxo de alunos e a perturbação do ambiente. A solução desse problema não é fácil, pois o PAS é classificado como um problema NP-Difícil [Even, Itai e Shamir 1976], pois trata-se de um problema que ainda não possui um algoritmo para resolvê-lo em tempo polinomial.

Segundo a literatura no PAS a função objetivo é composta pelo somatório das restrições, nos quais são atribuídos pesos conforme sua relevância na resolução do problema. Segundo [Souza, Martins e Araújo 2002] existem dois tipos de restrições no PAS, que são as restrições essenciais e as restrições não essenciais. Como restrições essenciais podemos citar o cenário onde um professor não pode estar alocado em duas salas no mesmo horário, e também duas turmas alocadas na mesma sala no mesmo horário de aula. As restrições não essenciais são aqueles que não afetam na solução caso não sejam atendidos, mas que se forem implementados, certamente ocasionará em uma solução mais viável, como por exemplo, alocar uma turma em uma sala que possui maior nível de refrigeração ou ventilação, ou menor ruído durante a aula.

2.3 Otimização Combinatória

Imagine um conjunto que possui regras, e usando essas regras seja possível extrair novos subconjuntos. Se cada elemento retirado tem seu custo, o novo subconjunto formado também terá o seu, que é fornecido pela soma do custo de cada objeto. O

problema de Otimização Combinatória, em geral, tem o objetivo de encontrar, dentre as possibilidades de subconjuntos, aquele que forneça o menor custo possível.

Segundo [Resende, Mateus e Silva 2012], a otimização combinatória pode ser definida por um conjunto finito de soluções viáveis e uma função objetivo, sendo modelado para cada problema específico. Os métodos heurísticos são utilizados para resolver problemas de otimização como o PAS, onde não é possível encontrar a solução em tempo polinomial, porém os mesmos têm que ser transformados em modelos de minimização ou maximização de pesquisa operacional, para que possam ser solucionados através de solução computacional.

2.4 Estratégias de solução do PAS (Heurísticas)

Na computação existem problemas que não podem ser resolvidos por algoritmos convencionais pois possuem complexidades que ainda podem ser dinâmicas. Estes algoritmos estão sempre em busca de duas características: obter um tempo de execução aceitável ou ser uma solução ótima ou próxima disso para o problema em todos casos.

Entende-se que a heurística na computação se caracteriza por métodos que vão encontrar uma solução viável, mas não uma solução ótima. Também podemos dizer que se espera no máximo resultados aproximados do ideal [Hillier e Lieberman 2013]. Normalmente é um algoritmo iterativo que a cada iteração envolve a condução da procura de uma nova solução que poderia ser o melhor resultado encontrado por ele previamente, sendo assim após o término em um tempo aceitável, ele fornece a melhor solução obtida naquela iteração [Hillier e Lieberman 2013]. A seguir será descrita a principal heurística para o problema considerado.

2.4.1 Algoritmo Guloso

Segundo [Martins 2010] o algoritmo guloso se baseia em um algoritmo chamado "melhor escolha", em que dada uma disciplina, o mesmo elenca a melhor sala que pode ser associada a tal disciplina no momento.

Para resolver um problema, o algoritmo guloso escolhe o objeto mais atrativo que encontra pela frente. Esse objeto passa a fazer parte da solução que o algoritmo vai construindo.

O algoritmo guloso não enxerga a distância, ou seja, ele só toma decisões baseadas na iteração atual, sem olhar as consequências futuras que essas decisões venham causar. As decisões tomadas por um algoritmo guloso numa iteração são definitivas, ou seja, o mesmo não se arrepende ou volta atrás.

A vantagem de se usar um algoritmo guloso é que geralmente eles são rápidos e eficientes, mas nem sempre trazem a melhor solução para o problema.

2.5 Estratégias de solução do PAS (Meta-heurísticas)

Com o objetivo de obter alternativas aos ótimos locais foram desenvolvidas as meta-heurísticas, pois essa é a grande desvantagem das heurísticas. De acordo com [Vieira et al. 2006] meta-heurística trata-se "De um processo iterativo ou de refinamento de soluções do problema que organiza e direciona a heurística pela combinação de diferentes conceitos, podendo manipular uma solução completa, incompleta ou um conjunto de soluções, tentando evitar parada prematura em ótimo local através de mecanismos que permitem escapar do mesmo[...] tendo como objetivo explorar características de boas soluções e até novas regiões promissoras, saindo de um ótimo local".

As meta-heurísticas são um conjunto de métodos heurísticos que buscam uma maneira de explorar o conjunto de soluções a fim de encontrar a solução ótima para o problema. [Nassiffe, Santos e Gubetti 2018] diz que "As meta-heurísticas são estratégias que orientam o processo de busca com o objetivo de explorar eficientemente o espaço à procura de soluções quase ótimas". Não é garantido que se encontre a solução ótima, mas na maioria dos casos é encontrada uma solução viável. Este método é geralmente aplicado aos problemas NP-Completo e NP-Difícil. Segundo [Prado e Souza 2014] a forma pela qual a meta-heurística trabalha o espaço de busca é classificada em duas categorias:

- Busca populacional: Partindo de um conjunto de soluções prontas, é realizada a aplicação de um conjunto de operadores, com a intenção de melhora.
- Busca local: Consiste em definir uma vizinhança, onde a mesma possui um conjunto de soluções com características "parecidas". Em um problema de minimização, tendo-se a solução atual, o algoritmo de busca local percorre a vizinhança buscando uma com valor menor. Se for encontrada, a mesma se torna a solução atual e algoritmo continua seu ciclo. Para o caso de não encontrar, a solução atual é considerada um ótimo local em relação à vizinha escolhida.

Na Figura 2.1 podemos ver a hierarquia a partir da heurística aos métodos de meta-heurística que vamos descrever a seguir.

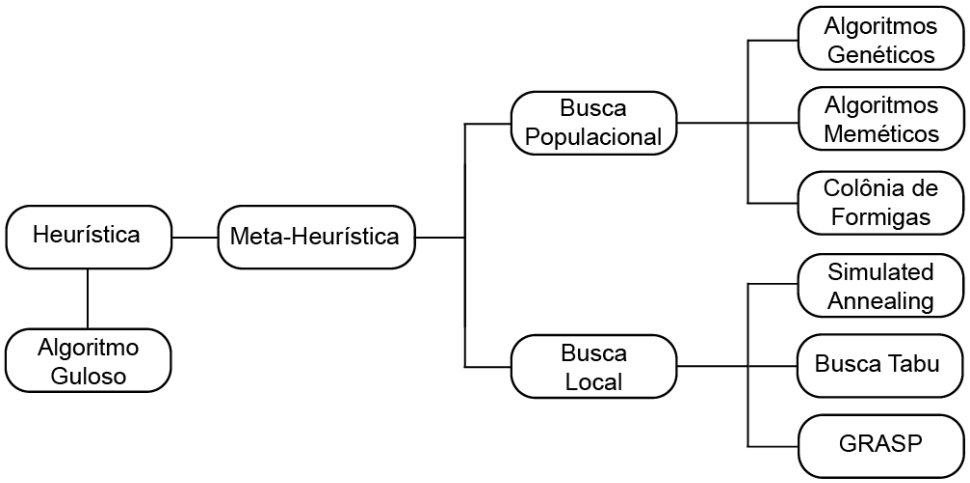

Figura 2.1: *Representação de hierarquia das heurísticas*

2.5.1 Algoritmos Genéticos

Conhecido como aplicação de técnicas heurísticas, os algoritmos genéticos (AG) se fundamentam através de uma analogia com os conceitos da Biologia. Uma definição de Algoritmos Genéticos bem clara pode ser dada pelo autor Ricardo Linden. Segundo [Linden 2006], "Algoritmos genéticos (GA) são um ramo dos algoritmos evolucionários e como tal podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural".

Para [Aytug et al. 2003], geralmente, os AG possuem oito componentes básicos, que são: sua representação genética, a população inicial, a função avaliação, o método de seleção de reprodução, os operadores genéticos, o método de seleção de gerações, os critérios de parada e os parâmetros de configuração dos AG.

Com base nos estudos é conhecido que os AG são altamente sensíveis aos seus principais parâmetros de configuração, sendo eles: o tamanho da população, o número de gerações, a taxa de cruzamento e a taxa de mutação. No entanto, são técnicas muito eficazes em problemas de otimização para descobrimento da melhor solução. Em nosso cenário podemos destacar as vantagens em aplicar os AG para direcionar os alunos às salas com um menor deslocamento. Abaixo no algoritmo 2.1 segue exemplo do pseudo-código do Algoritmo Genético.

Algoritmo 2.1: Algoritmos Genéticos

$X_0 \leftarrow$ *conjunto de n soluções*;
Alterar cada $x \in X_0$ *por melhoria iterativa*;
repita
    Selecionar $X_t \subseteq X_{t-1}$;
    Aumentar X_t *acrescentando os descendentes*;
    Alterar cada $x \in X_t$ *por melhoria iterativa*;
até *critério de parada;*

2.5.2 Algoritmos Meméticos

É uma meta-heurística pertencente à classe dos algoritmos populacionais. Algoritmos desse tipo utilizam várias soluções na busca por soluções viáveis no espaço de busca. Esse termo foi introduzido por [Moscato et al. 1989]. Para [Moscato e Cotta 2003], o que define os algoritmos meméticos é a utilização de várias meta-heurísticas. Para [Moscato e Norman 1992] a ideia principal do algoritmo é explorar as vizinhanças dos resultados obtidos por um algoritmo genético, buscando o local ótimo para cada solução. O termo memético é derivado da palavra “meme” [Radcliffe e Surry 1994], pois se refere à unidade de informação de um processo de argumentação, ou seja, refere-se à memória, análogo ao termo "gene"na genética.

Todas as etapas dos algoritmos genéticos estão presentes nos algoritmos meméticos, como operadores de seleção, recombinação e mutação. A diferença ocorre pela inclusão de uma etapa de otimização dos agentes, onde são adicionados agentes de busca local independentes, sendo que cada um possui seu aprendizado particular.

Uma diferença vista entre genes e memes é a etapa de transmissão aos seus descendentes. Ao transmitir o meme, ocorre uma adaptação baseada no seu conhecimento, buscando atender melhor às suas necessidades. Pode se dizer que os algoritmos genéticos emulam de forma computacional, a evolução biológica, sendo que os algoritmos meméticos emulam a evolução cultural. Abaixo no algoritmo 2.2 segue exemplo do pseudo-código do Algoritmo Memético.

Algoritmo 2.2: Algoritmos Meméticos

```
Input: ProblemSize, Pop_size, MemePop_size
Output: S_melhor
x ← InicializarPopulacao(ProblemSize, Pop_size);
repita
    para S_i ∈ Populacao faça
        Si_custo ← Custo(S_i);
    fim
    S_melhor ← ObtemMelhorSolucao(x);
    x ← BuscaGlobalEstocastica(x);
    xMemetico ← SelecionaPopulacaoMemetica(x, MemePop_size);
    para S_i ∈ PopulacaoMemetica faça
        S_i ← BuscaLocal(S_i);
    fim
até critério de parada;
retorna S_melhor
```

2.5.3 Ant Colony Systems (Colônia de Formigas)

Como o nome da meta-heurística já sugere, este método foi criado com base na observação do comportamento das formigas, que apesar das limitações físicas, têm a capacidade de realizar a melhor rota da colônia até a fonte de alimento. Isto se dá por um fenômeno chamado feromônio, substância química que as formigas liberam enquanto se deslocam.

Imaginemos um grupo de formigas em direção a uma fonte de alimentos, e há obstáculos neste caminho fazendo com que elas desviem, existe uma rota mais longa e uma mais curta, logo, as formigas se dividem entre os dois caminhos levando as que fizeram o caminho mais curto a chegar primeiro na fonte de alimento, e então no caminho de volta a liberação de feromônio fica mais forte, o que induz as demais formigas a realizar a rota mais curta.

Esta técnica baseia-se no mecanismo conhecido como reforço positivo (*positive feedback*) [Dorigo, Maniezzo e Colorni 1991], como mencionado, é a rota que possui o maior rastro de feromônio. Abaixo no algoritmo 2.3 segue exemplo do pseudo-código do algoritmo Colônia de Formigas.

Algoritmo 2.3: Colônia de Formigas

```
x ← alguma solução inicial;
x' ← x;
repita
    para todas as n trilhas faça
        x ← GerarNovasSoluções(x);
        AtualizarFeromônio(x);
    fim
    AvaliarSoluções(x);
    x' ← x;
até critério de parada;
```

2.5.4 Simulated Annealing

O Simulated Annealing é uma meta-heurística que utiliza probabilidades no momento da escolha da solução vizinha com intuito de escapar de ótimos locais de qualidade baixa [Martins 2010].

O SA é conhecido por Recozimento Simulado, o mesmo é um algoritmo de busca local que se baseia no processo de aquecer um metal até o ponto de fusão para então resfriá-lo, alterando as moléculas para um ponto de baixa energia formando uma estrutura cristalina e livre de defeitos. O Simulated Annealing conecta este comportamento termodinâmico à busca pelo máximo/mínimo global [Luzia e Rodrigues 2009].

Esta técnica começa o processo de otimização através de uma solução inicial qualquer, considerada como solução atual, e para a busca de valores otimizados são geradas soluções aleatórias para o problema. O processo também considera inicialmente um alto valor para o parâmetro da temperatura T, para a qual uma nova solução é gerada na vizinhança da solução atual. Se no processo surgirem soluções melhores, estas naturalmente são aceitas como soluções e como novos centros de busca, onde o processo será reiniciado a partir da nova solução encontrada. Abaixo no algoritmo 2.4 segue exemplo do pseudo-código do algoritmo Simulated Annealing.

Algoritmo 2.4: Simulated Annealing

```
x ← alguma solução inicial;
k ← 1;
repita
    gerar x' ∈ N(x);
    se f(x') ≤ f(x) então
        x ← x';
    senão
        se e^((f(x)−f(x'))/C_k) > random[0,1) então
            x ← x';
        fim
    fim
    k ← k + 1;
até critério de parada;
```

2.5.5 Tabu Search (Busca Tabu)

A Busca Tabu foi desenvolvida inicialmente por [Glover e Kochenberger 1996], a mesma tinha uma proposta para solucionar problemas de programação inteira. A partir daí a técnica foi formalizada pelo autor, que publicou diversos trabalhos com várias aplicações da mesma. Os testes têm mostrado que a Busca Tabu tem sido eficiente para solucionar vários problemas de diferentes naturezas, [Glover et al. 1993], podendo ser dito que trata-se de uma técnica consagrada.

Com o artigo de [Newell, Shaw e Simon 1958] foram iniciados os métodos para solucionar os problemas de competição (*challenging problems*), em um trabalho que conjugava a inteligência artificial e a Pesquisa Operacional. Mas logo as áreas se separaram, pois, a Pesquisa Operacional se concentrou nos resultados matemáticos das soluções e a Inteligência Artificial foi para o lado da análise qualitativa e simbólica.

Resumindo, a busca tabu é adaptativa com a busca local, em que existe uma estrutura de memória, onde se aceita movimentos de piora para escapar de locais ótimos, porém somente quando não existe possibilidade de melhora [Souza, Maculan e Ochi 2000]. Abaixo no algoritmo 2.5 segue exemplo do pseudo-código do algoritmo Busca Tabu.

Algoritmo 2.5: Busca Tabu

$T \leftarrow []$;
$x \leftarrow$ *alguma solução inicial;*
$\hat{x} \leftarrow$ x;
repita
  encontrar a melhor $x' \in N(x) \backslash T$;
  se $f(x') < f(\hat{x})$ **então**
    $\hat{x} \leftarrow x'$;
  fim
  $x \leftarrow x'$;
  atualizar lista tabu T;
até *critério de parada;*

2.5.6 GRASP (*Greedy Randomized Adaptive Search Procedures* - Procedimentos de busca aleatória, adaptativa e gulosa)

O GRASP (*Greedy Randomized Adaptive Search Procedures* - Procedimentos de busca aleatória, adaptativa e gulosa) é uma meta-heurística que realiza múltiplos reinícios, onde as iterações são compostas por uma solução construída de forma gulosa, sendo que posteriormente é feita uma busca local, com o intuito de encontrar um ótimo local. Essas são duas fases definidas por [Feo e Resende 1989]: Uma fase de construção, na qual uma solução é gerada, elemento a elemento; uma fase de busca local, na qual um ótimo local na vizinhança da solução construída é pesquisado.

A meta-heurística GRASP foi iniciada por [Feo e Resende 1989] onde se tratava o problema de cobertura de conjuntos. Desde então, a heurística já foi utilizada em vários problemas de otimização como: planarização de grafos [Resende e Ribeiro 1997], roteamento de circuitos virtuais [Resende e Ribeiro 2003] entre outros. O algoritmo possui iterações independentes, onde cada uma delas constrói a solução inicial e depois faz uma busca local para melhorar o resultado. Abaixo no algoritmo 2.6 segue exemplo do pseudo-código do algoritmo GRASP.

Algoritmo 2.6: GRASP

repita
  $x \leftarrow$ *GreedyRandomizedConstruction*();
  $x \leftarrow$ *BuscaLocal*(x);
  atualizar a melhor solução;
até *critério de parada;*

2.6 Considerações finais

Neste capítulo foi realizada a abordagem dos principais conceitos dos problemas de agendamento e foram descritas as estratégias relevantes para solucionar o PAS, desde a geração de uma solução inicial até os métodos que operam em um refinamento. No próximo capítulo será feita uma análise das especificidades do nosso problema e quais serão os melhores métodos para construção da solução.

CAPÍTULO 3

Abordagem do problema

Demonstrado os estudos de técnicas para solução do PAS no capítulo anterior, buscamos neste capítulo nos aprofundar sobre o problema real do IFG entendendo suas especificidades e definir a técnica que melhor possa nos auxiliar na resolução do problema. Para isso, vamos descrever o problema de alocação de salas no IFG e determinar a modelagem do problema, com a definição da função objetivo e as restrições que destacamos no decorrer do tempo de pesquisa do trabalho. Será escolhida uma técnica a ser utilizada para a construção da solução e que será justificada levando em consideração as características de uma forma geral das heurísticas apresentadas.

3.1 Especificidades do IFG

O Campus Goiânia é dividido em quatro departamentos, sendo que cada um deles possui suas salas e laboratórios exclusivos. As coordenações dos cursos de cada departamento são responsáveis por controlar a alocação e utilização dos espaços disponíveis. Caso uma coordenação queira solicitar uma sala/laboratório de outro departamento, deve-se enviar um memorando para o responsável por aquele local no outro departamento.

O IFG possui 125 salas, sendo divididas entre salas comuns e laboratórios. As mesmas são classificadas entre *S* e *T*, sendo as salas *T* localizadas no térreo e as salas S no primeiro andar. Também existe uma classificação da sala por bloco, onde a numeração muda de 100 em 100 para cada bloco. Exemplo: Sala no térreo, localizada no segundo bloco = T-201.

O instituto não possui salas padronizadas, ou seja, cada sala possui um formato específico e uma capacidade. Existem salas com quadro de giz ou de pincel, salas com ar condicionado e outras com ventilador. Algumas dispõem de data show e outras não, sendo que algumas não suportam a utilização de data show, pois não possuem o suporte adequado e a tela de projeção.

Os dias e horários de aula são previamente definidos em reunião dos professores. Alguns cursos possuem prioridades de salas de acordo com os recursos necessários para cada disciplina. Um exemplo são os laboratórios S-400, os mesmos possuem prioridade

de alocação para o curso superior de Bacharelado em Sistemas de informação, devido a maior parte das disciplinas do mesmo necessitarem ser ministradas nesse tipo de sala.

3.2 Abordagem proposta

Para solução do PAS do IFG - Campus Goiânia, foi realizada uma modelagem do problema, identificando restrições essenciais e não essenciais, a fim da realização do mapeamento. Logo após isso, foram elencadas características das heurísticas e meta-heurísticas e feita a argumentação sobre a escolha para a composição da nossa solução. Também exibimos uma tabela que mostra o comparativo dos critérios que as meta-heurísticas contemplam, a fim de exemplificar a opção escolhida.

3.2.1 Modelagem

Entendemos o PAS como um problema de designação, onde temos um conjunto de salas e um conjunto de horários disponíveis para ser feita a alocação das turmas. Sendo assim listamos as restrições que são significativas no processo de alocação de salas dividindo-as em restrições essenciais e restrições não essenciais. Restrições essenciais são as restrições que devem ser obrigatoriamente atendidas no processo de alocação, diferente das restrições não essenciais que são as restrições que podem ou não ser consideradas no processo. A Tabela 3.1 abaixo apresenta as restrições especificadas pela coordenação de curso do IFG, sendo *Re* as restrições essenciais e *Rn* as restrições não essenciais.

Tabela 3.1: *Conjunto de restrições da alocação de salas no IFG*

restrições	Descrição
Re_1	Não pode haver em uma mesma sala e horário mais de uma turma.
Re_2	Poderá alocar somente uma sala por turma que possui capacidade de suporte à quantidade de alunos da turma.
Rn_3	Cada turma poderá ter necessidade de recursos especiais como: Tela de projeção, Computadores, Softwares entre outros.
Rn_4	Cada turma poderá ser alocada em uma sala próxima ao bloco do seu curso.
Rn_5	Cada turma pode ser alocada na mesma sala durante a semana.
Rn_6	Turmas do mesmo curso poderão ser alocadas em salas do mesmo bloco na semana.

Apesar do trabalho aqui proposto ser contextualizado para um cenário específico, por simplicidade apresentamos a modelagem matemática do problema tratado usando uma representação genérica. Nesse caso, é definido x_{ij}, i = 1,2,...,m (enésima turma) e j = 1,2,...,n (enésima sala), como sendo as variáveis de decisão que se pretende encontrar, caso exista, segue especificação:

$$x_{ij} = \begin{cases} 1, & \text{se a turma } i \text{ é designada para a sala } j, \\ 0, & \text{caso contrário.} \end{cases}$$

A partir da primeira restrição essencial Re_1, somente uma turma poderá ser alocada em um determinado horário na mesma sala, ou seja, não haverá duas turmas ou mais em um mesmo horário e mesma sala. Para representar os horários considere *K, k* = 1,2,...,*t* (enésimo horário). Como representação do conjunto de turmas que se encontram no horário k, define-se P_k. Sendo assim, deve-se submeter:

$$\sum_{i \in P_k} x_{ij} \leq 1, \quad j = 1,2,...,n, k = 1,2,...,t.$$

A segunda restrição essencial Re_2 deve garantir que cada turma seja alocada para exatamente uma sala, respeitando sua capacidade. Para isso, define-se um conjunto de salas S_i que respeite a capacidade da turma i, i=1,2,...,m. Deve-se submeter:

$$\sum_{j \in S_i} x_{ij} = 1, \quad i = 1,2,...,m.$$

O objetivo aqui exposto é o de minimizar a quantidade de assentos vazios, o que se chamará c_{ij}, i=1,2,...,m e j=1,2,...,n, como podemos representar abaixo:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Com base nas restrições não essenciais *Rn*, para cada um deles é determinado um custo baseado em: Distância percorrida pelos alunos, preferências de coordenação por um certo bloco e nível de necessidade de recursos ou objetos que uma sala possui. Ao final, o ganho deverá ser calculado pela soma da quantidade de atributos das restrições não essenciais dividido pelo peso que é a subtração da capacidade da sala pela quantidade de alunos matriculados na turma. A formulação deste problema de designação de salas de aula é dado pelo problema de programação linear inteira 0-1,

$$\begin{aligned} \text{Minimizar} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{sujeito a:} \quad & \sum_{i \in P_k} x_{ij} \leq 1, \quad j = 1,2,...,n, k = 1,2,...,t, \\ & \sum_{j \in S_i} x_{ij} = 1, \quad i = 1,2,...,m, \\ & x_{ij} \in \{0,1\}, \quad i = 1,2,...,m \ \ e \ \ j = 1,2,...,n \end{aligned}$$

3.2.2 Meta-Heurística escolhida para a solução do problema

Considerando nosso estudo sobre o tema e o problema do Instituto Federal de Goiás – Campus Goiânia, concluímos no momento que o algoritmo ideal para solução do mesmo é o Simulated Annealing, pois o mesmo admite que já exista uma solução inicial, que é o nosso caso, e vai à procura de novos valores otimizados. Para construção dessa solução inicial, utilizaremos o algoritmo guloso, devido o mesmo conseguir obter uma solução parcialmente otimizada e demandar pouco tempo de execução. A Heurística oferece como recurso a parametrização de suas características que nos dá certa liberdade para manipular os valores conforme a demanda, por exemplo, se possuirmos uma grande quantidade de turmas para alocar em salas, podemos aumentar a quantidade de reaquecimento, caso contrário com poucas turmas, diminuímos esse valor de iteração, favorecendo a busca.

Consideramos como ideal utilizar a geração de vizinhança para realizar o movimento de realocação de salas e turmas, buscando por resultados de alocação mais satisfatórios analisando o contexto global da solução. Esta proposta visa minimizar a quantidade de computadores não utilizados nos laboratórios e tentar agrupar todas as aulas semanais de uma mesma turma em um único laboratório, assim como reduzir os assentos vazios nas salas comuns.

A escolha também se deve ao fato de ser o método mais optado pelos pesquisadores para a solução do problema de alocação de salas. Há uma vasta aplicação deste algoritmo neste caso devido a sua capacidade de otimização combinatorial e a pouca quantidade de parâmetros de controle se comparado às outras meta-heurísticas, e pela característica única de movimento de piora que visa escapar de ótimos locais.

Como justificativa da escolha da meta-heurística também ilustramos na Tabela 3.2 os critérios que cada meta-heurística contempla e não contempla.

Características das meta-heurísticas	*Algoritmos Genéticos*	*Algoritmos Meméticos*	*Colônia de Formigas*	*Simulated Annealing*	*Busca Tabu*	*GRASP*
Aceita movimento de piora	□	□	□	■	□	□
Necessita de solução inicial	■	■	■	■	■	☑
Utiliza busca local	□	□	□	■	■	■
Utiliza busca populacional	■	■	■	□	□	□
Busca a melhor solução global	■	■	■	■	■	■
Maior relevância na literatura pesquisada sobre o PAS	□	□	□	■	■	☑

Legenda: ■ Contempla □ Não Contempla ☑ Contempla Parcialmente.

Tabela 3.2: *Análise das características das meta-heurísticas para aplicação no PAS.*

3.3 Considerações finais

A modelagem do problema é de fundamental importância para entender os pontos nos quais a solução vai atuar, com o objetivo de prever o comportamento do mesmo. Para isso neste capítulo destacamos as restrições essenciais, não essenciais e a função objetivo, conceitos de otimização combinatória que nos possibilitaram sua aplicação neste trabalho. Foi definido o SA como a técnica para realizar o refinamento da solução a partir dos critérios definidos na Tabela 3.2. No próximo capítulo vamos explanar e detalhar os passos da solução desenvolvida e como o SA contribuiu para que conseguíssemos um resultado otimizado.

CAPÍTULO 4

Solução implementada

Conforme apresentado no capítulo anterior, realizamos um estudo e elegemos a meta-heurística Simulated Annealing para aplicar o refinamento da solução. Igualmente, ilustramos a modelagem do problema para o entendimento da função objetivo e as restrições mapeadas que são fundamentais para o desenvolvimento da aplicação.

Avaliando o cenário do IFG, propomos como solução o desenvolvimento de um sistema de alocação de salas visando reduzir os transtornos que eventualmente ocorrem devido ao processo de alocação manual por parte da coordenação. O sistema foi desenvolvido na linguagem Java, que permitiu abstrair bem o problema com a orientação a objetos. Devido ao atributo de ser multiplataforma, temos a garantia de que o sistema poderá ser implantado em qualquer sistema operacional. Para o desenvolvimento utilizamos a IDE Eclipse e o plugin ObjectAid para a geração do diagrama de classe, conforme a figura 4.1 abaixo.

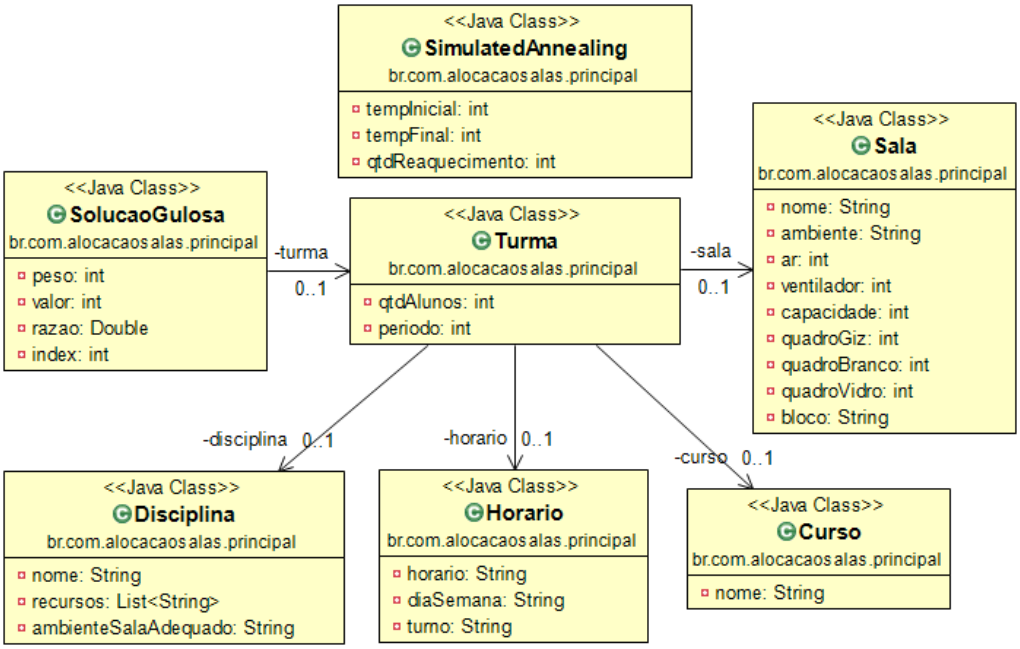

Figura 4.1: *Diagrama de classe*

A solução é representada por uma matriz multidimensional de dimensões |*T*| X

|*S*| X |*D*| X |*H*|, onde *T* é o conjunto de turmas de disciplinas, *S* é o conjuntos de salas, *D* é o conjunto de dias da semana (segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado) e *H* é o conjunto dos 16 horários disponíveis para o acontecimento das aulas em cada sala.

Para o desenvolvimento da solução adotamos o algoritmo guloso para geração da solução inicial e a meta-heurística Simulated Annealing para refinar o resultado em um tempo hábil. Nos tópicos a seguir, vamos descrever a estrutura da solução assim como os dados que obtivemos da gestão do Câmpus. A Figura 4.2 abaixo apresenta um fluxograma da aplicação.

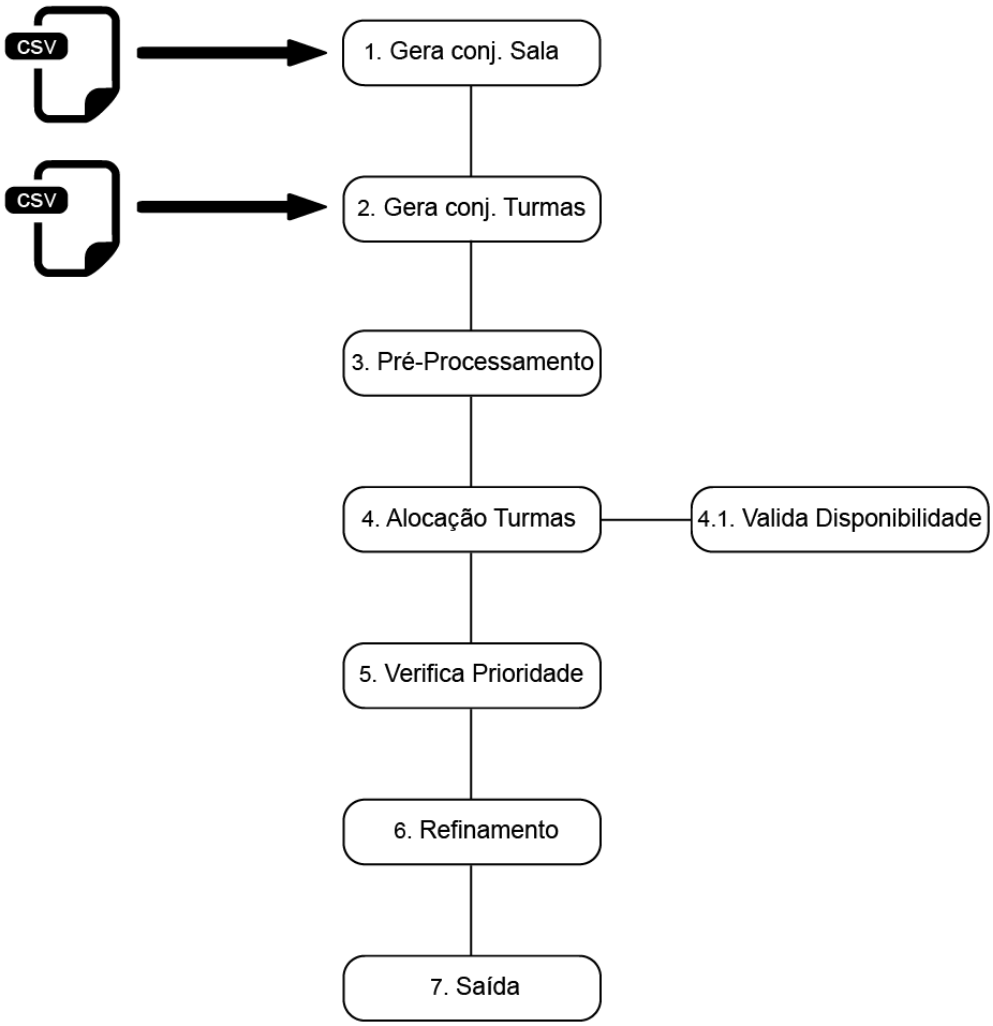

Figura 4.2: *Fluxo da aplicação*

Na Tabela 4.1 apresentaremos uma breve descrição de cada etapa da solução, o que será aprofundado nos próximos tópicos, assim como os métodos que foram utilizados para conseguirmos êxito na proposta de analisar a saída da aplicação e gerar resultados de comparação com o modelo atual do IFG.

Tabela 4.1: *Etapas da solução proposta*

Etapa	Descrição
1	Obtém os dados de sala e turma realizando a leitura das planilhas.
2	Realiza o pré-processamento das turmas para unir turmas com aulas sequenciais.
3	Inicia o processo de alocação nas salas baseando-se nas restrições essenciais e não essenciais gerando um conjunto de alocações factíveis para cada turma.
4	Seleciona a melhor alocação para cada conjunto de turmas através do algoritmo guloso.
5	Gera uma solução inicial baseada nas turmas informadas na planilha.
6	Itera sobre a solução inicial para buscar turmas prioritárias e realiza a desalocação se necessário.
7	Realiza o refinamento da solução inicial através da meta-heurística Simulated Annealing gerando uma solução otimizada.

4.1 Pré-processamento

Em consulta com a gestão do IFG Campus Goiânia, tivemos acesso aos dados de salas de aula representados em uma planilha que contém informações como: Capacidade da sala, quantidade de ar condicionado e ventilador, nome, bloco, quantidade de quadros (branco, negro e vidro) e ambiente da sala que define se é laboratório ou sala comum. Esta planilha, convertida para a extensão CSV, é a base de dados de entrada para nosso objeto de sala. Foi necessário realizar um pré-processamento também nessa planilha para transformar dados de texto para números, por exemplo, nas informações de recursos havia `Ar(2)`, isso foi convertido para 2 como uma espécie de discretização, sendo necessário criar colunas específicas para cada recurso.

Para a construção e validação da solução, elaboramos uma outra planilha que seria a entrada dos dados da turma, que é constituída por:

- Nome da disciplina.
- Ambiente da sala (laboratório ou sala comum).
- Período.
- Curso.
- Dia.
- Horário.
- Turno.
- Quantidade de alunos matriculados.

*C*omma Separated Values (CSV) [1]

[1] É um formato de texto que pode ser usado para trocar dados de uma planilha entre aplicativos.

Conforme citado no Capítulo 3 os dias e horários são definidos previamente pelos professores no início de cada semestre de acordo com a disponibilidade de cada um, sendo esta uma especificidade do IFG.

Considerando diferentes formas de entrada de dados, foi necessário realizar o pré-processamento das turmas levando em consideração que os registros podem estar descritos como horário integral ou parcial. Por exemplo, uma turma de Algoritmos pode ser solicitada das 19:00 às 22:15, como também pode ser solicitada das 19:00 às 20:30, e no mesmo dia das 20:45 às 22:15. Pensando na praticidade e em evitar deslocamento de sala desnecessário, o pré-processamento trata de unificar os horários da mesma turma no mesmo dia para ser alocada na mesma sala, considerando como um objeto só. Segue abaixo o código implementado para realizar o pré-processamento.

Código 4.1: *Pré-processamento*

```
1  /**
2   * Metodo que realiza o pre-processamento das turmas com base na planilha, para
3   * unificar os horarios das turmas evitando assim a troca de sala em horarios
4   * consecutivos
5   *
6   * @param turmas
7   * @param turmasProcessadas
8   * @return
9   */
10 private static Set<Turma> preProcessamento(Set<Turma> turmas, Set<Turma> turmasProcessadas)
       {
11   // Itera sobre as turmas e seta null nos horarios da lista de turmasProcessadas
12   turmas.forEach(x -> {
13     x.getHorario().setHorario(null);
14     turmasProcessadas.add(x);
15   });
16   // Cria lista de turmas auxiliar com base no metodo que realiza a leitura da planilha
17   Set<Turma> turmasAux = TurmaFactory.criaListaDeTurmas();
18   for (Turma turma : turmasProcessadas) {
19     int countAulas = 0;
20     int countElement = 0;
21     for (Turma t : turmasAux) {
22       countElement++;
23       if (turma.getDisciplina().equals(t.getDisciplina()) && turma.getCurso().equals(t.
             getCurso())
24           && turma.getPeriodo() == t.getPeriodo()
25           && turma.getHorario().getDiaSemana().equals(t.getHorario().getDiaSemana())
26           && turma.getHorario().getTurno().equals(t.getHorario().getTurno())
27           && turma.getQtdAlunos() == t.getQtdAlunos()) {
28         // Contabiliza a quantidade de aulas consecutivas
29         countAulas++;
30         turma.getHorario().setHorario(t.getHorario().getHorario());
31
32       }
33       if (countElement == turmasAux.size()) {
34         // Compara o horario final e ajusta o horario com base na quantidade aulas
               consecutivas
35         if (countAulas == 4 && turma.getHorario().getHorario().endsWith("22:15")) {
36           turma.getHorario().setHorario("19:00 - 22:15");
37         } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("22:15")) {
38           turma.getHorario().setHorario("20:45 - 22:15");
39         } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("20:30")) {
40           turma.getHorario().setHorario("19:00 - 20:30");
41         } else if (countAulas == 4 && turma.getHorario().getHorario().endsWith("18:00")) {
42           turma.getHorario().setHorario("14:45 - 18:00");
43         } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("18:00")) {
44           turma.getHorario().setHorario("16:30 - 18:00");
45         } else if (countAulas == 4 && turma.getHorario().getHorario().endsWith("16:15")) {
46           turma.getHorario().setHorario("13:00 - 16:15");
```

```
47            } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("16:15")) {
48              turma.getHorario().setHorario("14:45 - 16:15");
49            } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("14:30")) {
50              turma.getHorario().setHorario("13:00 - 14:30");
51            } else if (countAulas == 4 && turma.getHorario().getHorario().endsWith("12:00")) {
52              turma.getHorario().setHorario("08:45 - 12:00");
53            } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("12:00")) {
54              turma.getHorario().setHorario("10:30 - 12:00");
55            } else if (countAulas == 4 && turma.getHorario().getHorario().endsWith("10:15")) {
56              turma.getHorario().setHorario("07:00 - 10:15");
57            } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("10:15")) {
58              turma.getHorario().setHorario("08:45 - 10:15");
59            } else if (countAulas == 2 && turma.getHorario().getHorario().endsWith("08:30")) {
60              turma.getHorario().setHorario("07:00 - 08:30");
61            }
62
63          }
64        }
65      }
66      return turmasProcessadas;
67    }
```

4.2 Processo de alocação

O processo de alocação das turmas nas salas de aula se inicia partindo da premissa de que os dados estão organizados, que cada turma já possui seu dia e horário definido. A integridade da relação turma, sala e horário deve ser mantida, ou seja, o espaço físico alocado em um determinado horário deve ser ignorado nas demais iterações.

Primeiramente, as restrições essenciais precisam ser satisfeitas, não sendo possível considerar uma sala em que a capacidade de alunos é inferior à quantidade de alunos matriculados na disciplina, como também na circunstância de uma turma em que a disciplina seja de computação, ela não pode ser alocada em uma sala comum. Neste processo é criado um conjunto de possíveis alocações para cada turma, de modo que o algoritmo guloso possa definir; baseado em critérios de peso e valor, qual sala será alocada dentre o conjunto de possibilidades. Esta primeira etapa das alocações pode ser analisada pelo código implementado abaixo:

Código 4.2: *Processo de alocação*

```
1   /**
2    * Metodo responsavel por realizar a alocacao das turmas na sala que melhor
3    * atende aos criterios
4    * @param turmas
5    * @param salas
6    */
7   private static void alocaTurmas(Set<Turma> turmas, Set<Sala> salas) {
8     List<Turma> turmasAlocadas = new ArrayList<Turma>();
9     List<Turma> turmasNaoAlocadas = new ArrayList<Turma>();
10    // Inicia iteracao das turmas
11    for (Turma turma : turmas) {
12      List<Turma> turmasSolucaoGulosa = new ArrayList<Turma>();
13      int countSalas = 0;
14      for (Sala sala : salas) {
15        countSalas++;
16        // Validacoes de criterios/restricoes para alocacao
17        if (turma.getDisciplina().getAmbienteSalaAdequado() != null
```

```
18            && turma.getDisciplina().getRecursos() != null) {
19          if (turma.getDisciplina().getAmbienteSalaAdequado().equals(sala.getAmbiente())) {
20            if (turma.getQtdAlunos() <= sala.getCapacidade()) {
21              // Chama o metodo para validar se ja possui alocacao
22              if (jaAlocado(turmas, turma.getHorario(), sala) == false) {
23                if (turma.getSala() == null) {
24                  Turma t = new Turma(turma.getDisciplina(), turma.getQtdAlunos(), turma.
                        getCurso(),
25                      turma.getHorario());
26                  t.setSala(sala);
27                  // add solucao na lista de solucao gulosa
28                  turmasSolucaoGulosa.add(t);
29                } else {
30                  turma.setSala(sala);
31                }
32              }
33            }
34          }
35        } else if (turma.getDisciplina().getAmbienteSalaAdequado() != null
36            && turma.getDisciplina().getRecursos() == null) {
37          if (turma.getDisciplina().getAmbienteSalaAdequado().equals(sala.getAmbiente())) {
38            if (turma.getQtdAlunos() <= sala.getCapacidade()) {
39              if (jaAlocado(turmas, turma.getHorario(), sala) == false) {
40                if (turma.getSala() == null) {
41                  Turma t = new Turma(turma.getDisciplina(), turma.getQtdAlunos(), turma.
                        getCurso(),
42                      turma.getHorario());
43                  t.setSala(sala);
44                  turmasSolucaoGulosa.add(t);
45                } else {
46                  turma.setSala(sala);
47                }
48              }
49            }
50          }
51        } else if (turma.getDisciplina().getRecursos() != null
52            && turma.getDisciplina().getAmbienteSalaAdequado() == null) {
53          if (turma.getQtdAlunos() <= sala.getCapacidade()) {
54            if (jaAlocado(turmas, turma.getHorario(), sala) == false) {
55              if (turma.getSala() == null) {
56                Turma t = new Turma(turma.getDisciplina(), turma.getQtdAlunos(), turma.
                      getCurso(),
57                    turma.getHorario());
58                t.setSala(sala);
59                turmasSolucaoGulosa.add(t);
60              } else {
61                turma.setSala(sala);
62              }
63            }
64          }
65        }
66        if (countSalas == salas.size() && !turmasSolucaoGulosa.isEmpty()) {
67          // Ao obter todas as solucoes para a turma corrente, envia para o metodo da solucao
                gulosa
68          definePesoValor(turmasSolucaoGulosa, turma);
69        }
70      }
71    }
72    // Itera sobre as turmas e adiciona no conjunto correspondente de turmas alocas ou nao
        alocadas
73    turmas.forEach(t -> {
74      if (t.getSala() == null) {
75        turmasNaoAlocadas.add(t);
76      } else {
77        turmasAlocadas.add(t);
78      }
79    });
80    // Aplica regra de prioridade de curso sobre salas
81    verificaSalaComPrioridade(turmasNaoAlocadas, turmasAlocadas);
82  }
```

O algoritmo guloso nesse caso foi definido para eleger a melhor sala para a turma atual da iteração. Dessa forma, a primeira turma a ser alocada ganha vantagem em relação às próximas turmas que venham a ter o mesmo horário e atender as mesmas restrições. A estratégia gulosa foi definida da seguinte maneira:

$$R = \frac{V}{P}$$

onde o Valor V, é a soma de todas as características positivas que a sala possui, e o Peso P é a subtração da capacidade da sala com a quantidade de alunos matriculados. A Razão R é o resultado utilizado pela estratégia gulosa responsável por eleger a melhor sala para a turma. Isso é feito através de iteração das turmas onde cada uma recebe um valor de Razão R, e em seguida essa lista de turmas é ordenada de forma decrescente pela razão e a melhor turma é o primeiro elemento da lista, caracterizando-se pela solução ótima local. O código abaixo apresenta a solução implementada do algoritmo guloso.

Código 4.3: *Algoritmo Guloso*

```
1  /**
2   * Metodo responsavel por Realizar a alocacao caracterizando como a solucao
3   * gulosa
4   *
5   * @param turmasSolucaoGulosa
6   * @param turma
7   */
8  private static void definePesoValor(List<Turma> turmasSolucaoGulosa, Turma turma) {
9    List<SolucaoGulosa> solucaoGulosaList = new ArrayList<SolucaoGulosa>();
10   // Inicia iteracao das solucoes construidas
11   for (Turma t : turmasSolucaoGulosa) {
12     SolucaoGulosa sg = new SolucaoGulosa();
13     sg.setTurma(t);
14     // Peso e a subtracao da capacidade da sala com a qtd de alunos matriculados
15     sg.setPeso(t.getSala().getCapacidade() - t.getQtdAlunos() + 1);
16     // Valor e todos recursos possiveis que torna a sala mais vantajosa
17     sg.setValor(t.getSala().getAr() + t.getSala().getVentilador() + 1);
18     // Razao e o criterio usado para alocacao: Peso dividido pelo valor
19     double razao = sg.getValor() / sg.getPeso();
20     sg.setRazao(razao);
21     solucaoGulosaList.add(sg);
22   }
23   // Ordena a lista pela Razao
24   Collections.sort(solucaoGulosaList, Collections.reverseOrder(new Comparator<SolucaoGulosa
         >() {
25
26     @Override
27     public int compare(SolucaoGulosa sg1, SolucaoGulosa sg2) {
28       return sg1.getRazao().compareTo(sg2.getRazao());
29     }
30
31   }));
32   // Aloca a primeira sala do array devido a ordenacao por razao
33   turma.setSala(solucaoGulosaList.get(0).getTurma().getSala());
34 }
```

4.3 Prioridades na alocação

Uma característica importante na alocação de salas que mapeamos, é a priorização de espaço físico para determinados cursos. Através de uma estatística intuitiva, certos cursos possuem predominância de tipos de salas que necessariamente as disciplinas precisam ser direcionadas. Como exemplo, no curso de Bacharelado em Sistemas de Informação, a grande maioria de disciplinas são necessariamente ministradas em laboratórios de informática. Do mesmo modo, salas de música devem ser utilizadas de preferência para o curso de Licenciatura em Música, entre outros exemplos que poderíamos citar.

A solução desenvolvida se propõe a validar que quando um curso tem preferência por determinado tipo de sala, esta esteja disponível para alocação. De acordo com a ordem de execução, caso a alocação seja feita para outra turma, o sistema deverá desalocar e atribuir para a turma adequada. Este processo será feito até que todas as turmas prioritárias estejam alocadas, e as que foram desalocadas são disponibilizadas novamente para alocação.

Este processo é realizado por uma iteração da lista de turmas alocadas e por uma lista de turmas não alocadas, que foi gerada na etapa de alocação no qual contém as turmas que exigem um ambiente de sala específico, porém, no horário definido da aula não houve salas disponíveis que pudessem atender. Logo, é feito uma comparação entre o curso e o ambiente de sala e dos elementos de ambas as listas para identificar se a turma alocada não possui prioridade, caso não possua, esta turma é realocada para o conjunto de turmas não alocadas, havendo assim uma troca de elementos entre conjuntos. Na Figura 4.3 é demonstrado essa troca de elementos em que ambas as turmas são de uma disciplina que necessariamente precisa ser alocada em laboratório de informática, mas que o curso Bacharelado em Sistemas de Informação possui prioridade.

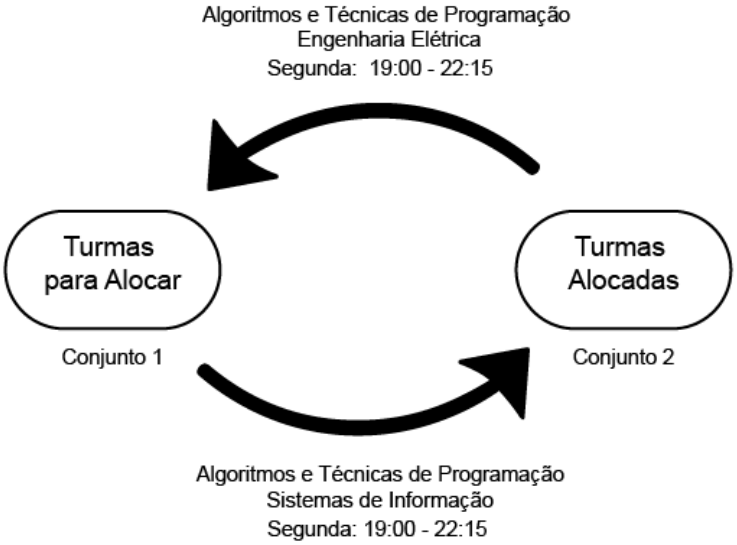

Figura 4.3: *Desalocação de turma por prioridade de curso*

4.4 Refinamento da solução

Com base na solução obtida através da heurística algoritmo guloso, temos um conjunto de turmas alocadas em uma sala em um horário específico, caracterizando-se como a solução ótima até o momento. No entanto, para conseguir um melhor resultado em relação a função objetivo, se faz necessário analisar todo o espaço de busca para verificar se é possível uma troca, um movimento. Este movimento de troca pode ser demonstrado na Figura 4.4 abaixo.

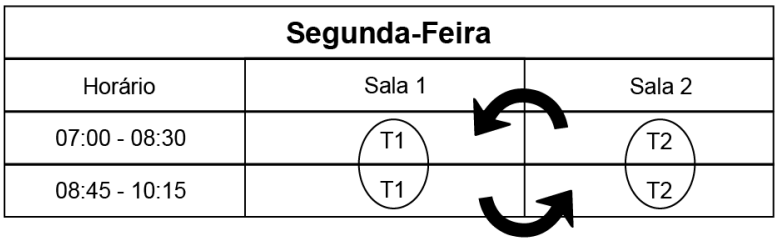

Figura 4.4: *Movimento de troca*

Para realizar este refinamento, utilizamos a meta-heurística Simulated Anneling que nos possibilita através da geração de vizinhos que atendem as restrições essenciais, selecionar um vizinho aleatoriamente e comparar se a relação custo e valor possui vantagem em relação à solução corrente. Desta forma a meta-heurística nos permite configurar a quantidade de iterações em busca de um ótimo global, e caso haja sucesso realizar o movimento de troca. O esquema de resfriamento foi realizado com uma taxa

de 0,95 que corresponde a 5% de resfriamento que a cada iteração a temperatura inicial reduz até o ponto de reaquecimento.

$$Temp = Temp * 0,95$$

Segue abaixo o código implementado da meta-heurística Simulated Annealing:

Código 4.4: *Simulated Annealing*

```
/**
 * Metodo que aplica a meta heuristica Simulated Annealing
 * @param melhorObjetivo = Solucao Inicial
 */
private static void SimulatedAnnealing(List<SolucaoGulosa> melhorObjetivo) {
  // Cria objeto Simulated Annealing com os parametros (tempInicial, tempFinal,
      qtdReaquecimento)
  SimulatedAnnealing sa = new SimulatedAnnealing(10, 2, 5);
  Set<Turma> melhorSolucao = new LinkedHashSet<Turma>();
  // Inicia iteracao da solucao inicial
  for (SolucaoGulosa turmaCorrente : melhorObjetivo) {
    SolucaoGulosa sg = new SolucaoGulosa();
    // Itera ate a quantidade de reaquecimento definida
    for (int i = 0; i < sa.getQtdReaquecimento(); i++) {
      double temp = sa.getTempInicial();
      while (temp > sa.getTempFinal()) {
        // metodo que resfria na taxa de 5%
        temp = sa.esquemaResfriamento(temp);
        // chama metodo que retorna uma solucao vizinha aleatoriamente
        sg = geraVizinho(melhorObjetivo, turmaCorrente);
        // Valida se o ganho da solucao vizinha e maior que a solucao atual
        if (turmaCorrente.getRazao() < sg.getRazao()) {
          Sala sala = sg.getTurma().getSala();
          if (turmaCorrente.getTurma().getQtdAlunos() >= sala.getCapacidade()) {
            // Turma sugerida recebe a sala da turma da iteracao
            melhorObjetivo.get(sg.getIndex()).getTurma().setSala(turmaCorrente.getTurma().
                getSala());
            // Turma da iteracao recebe a sala da turma sugerida
            turmaCorrente.getTurma().setSala(sala);
          }
          // Aplica probabilidade para decidir se realiza movimento de troca e piora
        } else if (Math.exp((sg.getRazao() - turmaCorrente.getRazao()) / temp)
            > Math.random()) {
          Sala sala = sg.getTurma().getSala();
          if (turmaCorrente.getTurma().getQtdAlunos() >= sala.getCapacidade()) {
            // Turma sugerida recebe a sala da turma da iteracao
            melhorObjetivo.get(sg.getIndex()).getTurma().setSala(turmaCorrente.getTurma().
                getSala());
            // Turma da iteracao recebe a sala da turma sugerida
            turmaCorrente.getTurma().setSala(sala);
          }
        }
      }
    }
    // Add a solucao na lista de melhor solucao
    melhorSolucao.add(turmaCorrente.getTurma());
  }
}
```

O resultado da solução pode ser representado por uma matriz de agendamento semanal para cinco turmas, conforme exemplo na Tabela 4.2 a seguir.

	2ª	3ª	4ª	5ª	6ª
07:00 - 08:30	Algoritmos **S-401A**	Fundamentos de SI **S-401B**		Português **S-107**	Português **S-107**
08:45 - 10:15	Algoritmos **S-401A**	Fundamentos de SI **S-401B**	Lógica **S-403C**	Matemática Discreta **S-107**	Matemática Discreta **S-107**
10:30 - 12:00	Tecnologia e Sociedade **S-401B**	Tecnologia e Sociedade **S-401B**	Lógica **S-403C**		

Tabela 4.2: *Resultado da alocação para o sistema acadêmico.*

4.5 Considerações finais

O desenvolvimento da solução foi parcialmente complexo devido à necessidade de entendimento das heurísticas, de abstrair as ideias obtidas através dos pseudo-códigos e conseguir aplicar ao problema com todas as particularidades que apenas no cenário real podemos nos deparar.

A solução teve grande exploração das estruturas de dados para trabalhar de forma eficiente o tratamento de conjuntos ordenados, não repetidos, e a utilização de cópias que não alteram o estado original dos conjuntos. A parte das restrições não essenciais de distância a percorrer não foram implementadas devido à ausência de dados suficientes, mas procuramos desenvolver a solução para estar apta a receber estas melhorias. No próximo capítulo vamos descrever os resultados obtidos e a validação da solução comparando-a com os dados de alocações anteriores fornecidos pela gestão do IFG.

CAPÍTULO 5

Avaliação da solução

O objetivo do trabalho foi desenvolver uma solução para resolver o problema de alocação de salas no IFG - Câmpus Goiânia conforme demonstrado no Capítulo 4. Com isso em mente, torna-se necessário avaliar esta solução para identificarmos os pontos de melhoria e validarmos os resultados. Com este protótipo desenvolvido, esperamos entregar ao instituto uma ideia relevante e comprovadamente mais eficiente que a solução atual.

A partir da solução implementada, conforme exposto no capítulo anterior, propomos neste capítulo relatar os resultados obtidos, como, o tempo gasto na execução e a avaliação de ganho das alocações feitas pela aplicação. O objetivo também é comparar os resultados da solução com as alocações feitas manualmente pela gestão no processo atual. Com base na função objetivo definida que é a minimização de assentos vazios, poderemos analisar o ganho ou a perda da solução automatizada.

Os experimentos foram realizados em um notebook Dell com processador Intel Core i7-8750H (cache 9M, 2,20 GHz até 4,10 GHz), com 8GB de memória RAM, SSD 240GB M.2, sob o sistema operacional Windows 10 64bits.

5.1 Descrição dos experimentos

Para cumprir com os objetivos de validação e comparação de resultados foi necessário criar uma planilha com os dados das alocações feitas no IFG do período 1/2019, baseada em documentos fornecidos pela coordenação que contém esses dados. Na planilha criada, realizamos o cálculo de ganho manualmente definido pela função objetivo abordada neste trabalho em cada registro de turma, devido essas turmas já possuírem as salas de aulas definidas. Feito isso, utilizamos os mesmos dados, porém sem a informação de sala de aula, e executamos na solução para obter os ganhos que a aplicação gerou para essas mesmas turmas. Com isso, temos um modelo de comparação com a solução gerada computacionalmente e a manual.

Os experimentos foram realizados alocando todas as turmas do semestre 2019/1 no turno noturno, dessa maneira podemos validar o maior número de alocações possíveis

nos horários estabelecidos desse período. Foi necessário limitar os experimentos neste semestre e turno, devido a inviabilidade de elaborar as planilhas das alocações feitas pela gestão do câmpus para os outros turnos, pois para gerar um registro de turma é preciso navegar entre dois documentos com informações dispersas e realizar o cálculo de ganho manualmente. Também se faz necessário ponderar a quantidade de experimentos para que não se estenda demasiadamente o trabalho, nesse caso escolhemos três cursos que identificamos como melhores exemplos, devido as ações da heurística terem apresentado variações. Dessa forma, os experimentos foram feitos a partir da alocação de 780 turmas, ou seja, todas as turmas de 2019/1 e três cursos foram selecionados para demonstrar os experimentos.

Outra métrica que propomos validar é quantificar o tempo de processamento da solução, desconsiderando o tempo de leitura de dados. Essa medição foi realizada em duas partes, no processamento do algoritmo guloso e e outra na Simulated Annealing, para que possamos mensurar o esforço de refinar a solução avaliando se é viável, ou se apenas a solução gulosa já é suficiente em relação ao custo benefício de tempo e resultado.

5.2 Resultados obtidos

É importante ressaltar que os resultados que serão aqui apresentados desconhecem o cenário do processo de alocação realizado pela coordenação do câmpus, não considerando possíveis restrições adicionadas após o acesso aos dados utilizados, ou seja, a atual alocação de salas tida na realidade será processada de acordo com os critérios e restrições implementados na solução.

Para realizar os experimentos é necessário definir uma métrica de avaliação para entendermos qual solução será mais eficiente. Com isso, realizamos alguns testes que podem ser observados na Tabela 5.1 que apresenta um exemplo de duas turmas em que a heurística realizou um movimento de troca.

Solução Gulosa				
Turma	**Qtd. Alunos**	**Sala**	**Capacidade**	**Ganho**
Teoria Geral da Administração	32	S-102	33	2
Matemática Discreta	31	S-107	32	3
Simulated Annealing				
Turma	**Qtd. Alunos**	**Sala**	**Capacidade**	**Ganho**
Teoria Geral da Administração	32	S-107	32	6
Matemática Discreta	31	S-102	33	1,33

Tabela 5.1: *Movimento de troca realizado no refinamento.*

O ponto a se observar é que o refinamento trouxe um menor ganho para a turma de Matemática Discreta, porém elevou consideravelmente o ganho de Teoria Geral da

Administração, com isso chegamos à conclusão que a solução que obter maior média será considerada mais eficaz, o que nesse caso demonstra que o refinamento realizou uma otimização, pois a média da Solução Gulosa foi de 2,5 e a média da Simulated Annealing foi de 3,66. Sendo assim, nos experimentos a seguir utilizaremos a média de ganho como métrica de definição da melhor solução obtida.

Os dados representados nos gráficos abaixo caracterizam-se da seguinte maneira: O eixo y representa os ganhos das alocações de forma numérica; O eixo x representa as turmas do semestre na nomenclatura de sigla. A Figura 5.1 apresenta os ganhos obtidos com as três soluções para o curso Técnico Integrado em Informática para Internet - Proeja que possui 29 alunos matriculados.

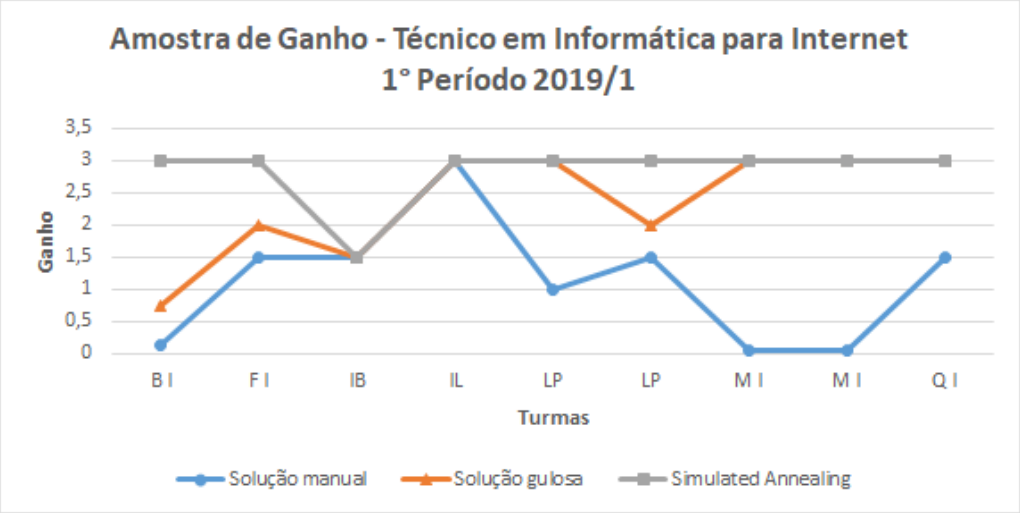

Figura 5.1: *Gráfico de ganho obtido para a turma do Técnico Integrado em Informática para Internet - Proeja*

Nota-se que a solução manual atingiu menor ganho em relação as demais soluções. O refinamento da Simulated Annealing se mostrou eficiente para o curso Técnico Integrado em Informática para Internet - Proeja. A média de ganho das soluções foram de 1,14 para a Solução Manual, 2,36 para a Solução Gulosa e 2,83 para a Simulated Annealing. A Tabela 5.2 abaixo relata as salas que cada solução alocou para as turmas.

A Figura 5.2 apresenta o experimento de alocação das soluções para o curso de Bacharelado em Engenharia Elétrica e possui 28 alunos matriculados.

Técnico em Técnico Integrado em Informática para Internet - Proeja 2019/1				
		Salas		
Sigla	**Nome**	**SM**	**SG**	**SA**
B I	Biologia I	T-109	S-502 B	S-508
F I	Física I	S-104	T-502 A	S-508
IB	Informática Básica	S-401 C	S-402 B	S-402 B
IL	Introdução à Lógica	S-401 C	S-401 D	S-401 D
LP	Língua Portuguesa I	S-105	S-508	S-508
LP	Língua Portuguesa I	S-104	T-502 A	S-503 B
M I	Matemática I	S-205	S-508	S-508
M I	Matemática I	S-205	S-508	S-503 B
Q I	Química I	S-104	S-503 B	S-503 B

Tabela 5.2: *Tabela de turmas do curso Técnico Integrado em Informática para Internet - Proeja 2019/1*

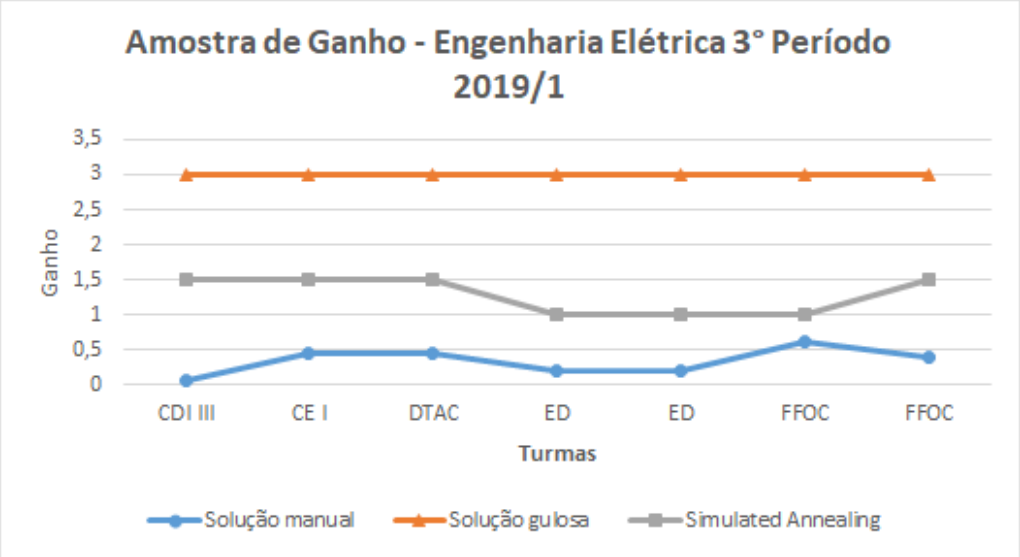

Figura 5.2: *Gráfico de ganho obtido para a turma de Bacharelado em Engenharia Elétrica*

O experimento da Figura 5.2 demonstra um novo ponto de observação. A alocação realizada pela Solução Gulosa se destacou em relação à Simulated Annealing. Com isso, analisamos e identificamos que devido a característica da heurística de avaliar o espaço de busca global, poderá haver movimentos de troca entre cursos e turmas. Sendo assim para um determinado curso haverá perda de ganho em detrimento do aumento de ganho para outra turma que a solução considera mais relevante. A média de ganho das alocações são de 0,37 para a Solução Manual, 3 para a Solução Gulosa e 1,28 para a Simulated Annealing.

Na Tabela 5.3 podemos analisar as informações de turmas e salas que cada solução alocou.

No experimento da Figura 5.3 abaixo, será demonstrado as alocações para o

Bacharelado em Engenharia Elétrica 2019/1				
		Salas		
Sigla	**Nome**	**SM**	**SG**	**SA**
CDI III	Cálculo Diferencial e Integral III	S-205	T-502 D	S-508
CE I	Circuitos Elétricos I	S-308	T-502 D	S-503 B
DTAC	Desenho Técnico Assistido por Computador	S-310	T-502 D	S-503 B
ED	Equações Diferenciais	S-210	T-502 D	T-502 A
ED	Equações Diferenciais	S-204	T-502 D	T-502 A
FFOC	Física: Fluídos, Ondas e Calor	S-107	T-502 D	T-502 A
FFOC	Física: Fluídos, Ondas e Calor	S-108	T-502 D	S-503 B

Tabela 5.3: *Tabela de turmas do curso de Bacharelado em Engenharia Elétrica 2019/1*

curso de Bacharelado em Turismo, que comumente as turmas são alocadas apenas em salas comuns. Essa turma possui 28 alunos matriculados.

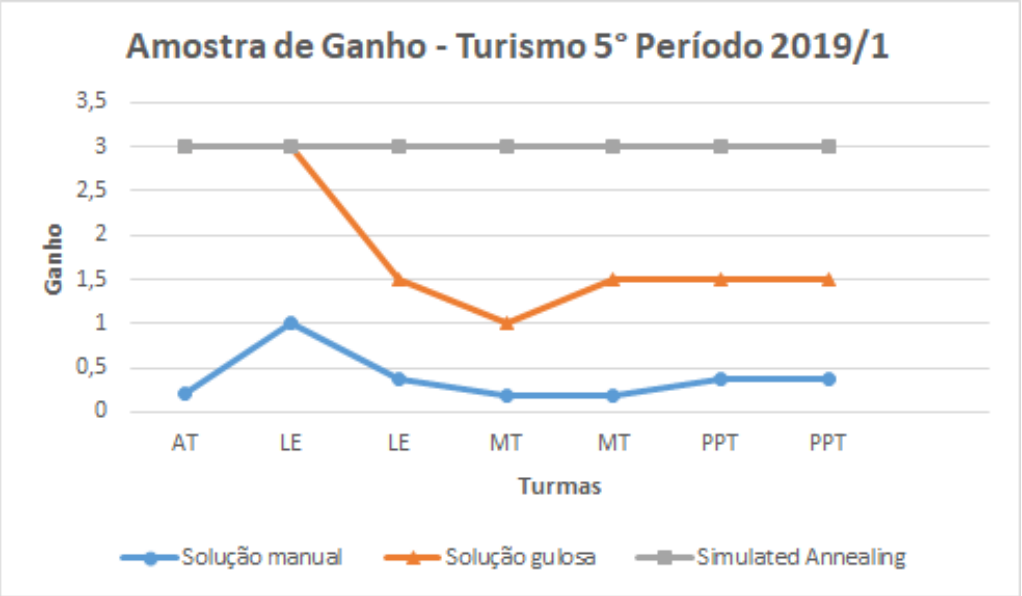

Figura 5.3: *Gráfico de ganho obtido para a turma de Turismo*

Os resultados da Figura 5.3 apresentam que a heurística realizou otimização em relação à Solução Gulosa. Isso ocorre quando a heurística obtém soluções vizinhas que o ganho é superior à solução atual. O fenômeno que podemos observar neste experimento é que foi feito movimentos de troca possivelmente com outro curso, devido à Solução Gulosa não ter encontrado as melhores salas nos horários disponíveis. Na Figura 5.4 abaixo, podemos observar que a heurística determinou a mesma sala para todas as turmas.

Bacharelado em Turismo 2019/1				
		Salas		
Sigla	**Nome**	**SM**	**SG**	**SA**
AT	Agências de Turismo	S-201	T-502 D	T-502 D
LE	Língua Estrangeira II	S-213 B	T-502 D	T-502 D
LE	Língua Estrangeira II	S-406	S-508	T-502 D
MT	Marketing Turístico	S-212	T-502 A	T-502 D
MT	Marketing Turístico	S-212	S-508	T-502 D
PPT	Políticas Públicas do Turismo	S-406	S-508	T-502 D
PPT	Políticas Públicas do Turismo	T-203	S-503 B	T-502 D

Tabela 5.4: *Tabela de turmas do curso de Bacharelado em Turismo 2019/1*

Outro experimento importante de ser realizado é o tempo de execução gasto para obter a solução gulosa e a solução otimizada pela Simulated Annealing. De acordo com a especificação da máquina informada no início do capítulo, realizamos o teste com um total de 780 turmas para alocar em salas, com as variedades de cursos apresentados nas avaliações acima e de períodos diferentes. O resultado da execução do algoritmo guloso foi de 52 milissegundos, e o refinamento com a heurística foi de 201 milissegundos.

5.3 Considerações finais

Concluímos com os experimentos e avaliações que o tempo de execução da heurística em relação à solução gulosa cresce exponencialmente de acordo com o volume de dados para realizar a alocação. Considerando que as alocações podem ser feitas de turma em turma, o custo computacional é mínimo e pode ocasionar em uma solução otimizada com a heurística, mas os testes demonstraram que a solução gulosa já oferece um resultado satisfatório com um custo menor. De fato, ao aplicar no cenário real do IFG poderíamos validar melhor a solução com os recursos do Instituto, e caso necessário a heurística nos permite manipular seus parâmetros para conseguirmos otimizações com menos tempo de execução.

CAPÍTULO 6

Conclusões

Neste trabalho apresentamos os estudos relacionados ao problema de alocação de salas e as principais técnicas utilizadas para solução, com o propósito de implementar uma solução computacional baseada neste referencial teórico. Os problemas que tratam da otimização combinatória são vastos. Este trabalho foi um bom ponto de partida para nos aprofundarmos nestes problemas que podem ser aplicados em diversas situações. Foi realizado uma modelagem matemática partindo das restrições essenciais e não essenciais definidos para a solução do problema do IFG - Câmpus Goiânia, e o estudo das heurísticas e meta-heurísticas que foram desenvolvidas especialmente para estes problemas considerados NP-Difícil.

Para cumprir com nosso objetivo de resolver o problema de alocação de salas no IFG - Câmpus Goiânia, foi desenvolvido uma solução que aplica um algoritmo de otimização que gera uma solução inicial, denominado como *Algoritmo Guloso*. Ademais, foi feito um refinamento da solução com a meta-heurística *Simulated Annealing*. A solução realiza um pré-processamento dos dados e aplica a regra de priorização de cursos para determinadas salas.

Conseguimos bons experimentos desenvolvendo esta solução com a demonstração e a validação dos resultados obtidos. O objetivo é incentivar que cada coordenação de curso do IFG - Câmpus Goiânia venha a utilizar esta solução para melhoria do processo de alocação no início de semestre. A intenção no presente momento com o apoio de recursos do instituto é prosseguir com o trabalho com a idealização do sistema e com a construção de uma interface de usuário, além de avançar nas especificidades do IFG.

6.1 Trabalhos futuros

Com a pesquisa realizada notamos que o ambiente da universidade oferece várias possibilidades para a aplicação de soluções de otimização, podendo ser o PAS apenas o ponto de partida se estendermos o nicho da pesquisa para os demais problemas de *Timetabling* como: *Problema de Programação de Horários em Escolas* (PPHE),

Problema de Programação de Horários em Universidades (PPHU) e *Problemas de Programação de Horários de Exames* (PPHE).

Sugerem-se então os seguintes trabalhos futuros:

- Desenvolver a interface de usuário da solução integrada;
- Desenvolver arquitetura de microsserviços para todas as soluções;
- Implementar na solução uma fusão das meta-heurísticas *Simulated Annealing* e *Busca Tabu*, que alguns estudos dizem ser o estado da arte para este problema;
- Buscar obter um maior número de dados do IFG para evoluir a solução como: Salas com maior ruído em determinados horários, distâncias de blocos e salas a percorrer, softwares instalados em laboratórios de informática, entre outros dados válidos;
- Implementar o refinamento com as demais meta-heurísticas a critério de comparação de desempenho entre elas;
- Criar um cadastro de cursos e salas que possuem relação de prioridade;
- Externar os parâmetros de configuração da heurística;
- Incluir a possibilidade de gerenciar o uso das restrições essenciais de acordo com a necessidade;
- Elaborar uma maneira mais eficiente de entrada de dados para a solução.

Referências Bibliográficas

ALVAREZ-VALDES, R.; CRESPO, E.; TAMARIT, J. M. Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, Elsevier, v. 137, n. 3, p. 512–523, 2002.

AYTUG, S. et al. Impaired irs-1/pi3-kinase signaling in patients with hcv: a mechanism for increased prevalence of type 2 diabetes. *Hepatology*, Wiley Online Library, v. 38, n. 6, p. 1384–1392, 2003.

BARDADYM, V. A. Computer-aided school and university timetabling: The new wave. In: BURKE, E.; ROSS, P. (Ed.). *Practice and Theory of Automated Timetabling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 22–45. ISBN 978-3-540-70682-3.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: An autocatalytic optimizing process. Citeseer, 1991.

EVEN, S.; ITAI, A.; SHAMIR, A. On the complexity of time table and multi-commodity flow problems. In: IEEE. *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*. [S.l.], 1976. p. 184–193.

FEO, T. A.; RESENDE, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, Elsevier, v. 8, n. 2, p. 67–71, 1989.

GLOVER, F.; KOCHENBERGER, G. A. Critical event tabu search for multidimensional knapsack problems. In: *Meta-Heuristics*. [S.l.]: Springer, 1996. p. 407–427.

GLOVER, F. et al. *Tabu search*. [S.l.]: Springer, 1993.

HILLIER, F. S.; LIEBERMAN, G. J. *Introdução à pesquisa operacional*. [S.l.]: McGraw Hill Brasil, 2013.

LINDEN, R. *Algoritmos genéticos: uma importante ferramenta da inteligência computacional*. [S.l.]: Brasport, 2006.

LUZIA, L.; RODRIGUES, M. Introdução ao escalonamento e aplicações: Estudo sobre as metaheurísticas. *São Paulo, SP: Escola de Artes, Ciências e Humanidades da Universidade de São Paulo*, 2009.

MARTINS, J. P. *O Problema do agendamento semanal de aulas*. Dissertação (Mestrado) — Mestrado em Ciência da Computação, 2010. Ciências Exatas e da Terra - Ciências da Computação. Disponível em: <http://repositorio.bc.ufg.br/tede/handle/tde/502>.

MOSCATO, P.; COTTA, C. Una introducción a los algoritmos meméticos. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, Asociación Española para la Inteligencia Artificial, v. 7, n. 19, p. 0, 2003.

MOSCATO, P.; NORMAN, M. G. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel computing and transputer applications*, Amsterdam., v. 1, p. 177–186, 1992.

MOSCATO, P. et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, v. 826, p. 1989, 1989.

NASSIFFE, R. M.; SANTOS, A. C. dos; GUBETTI, M. K. Otimização combinatória: O problema de alocação de professores formulado como um problema binário. *Anais da Mostra de Ensino, Pesquisa, Extensão e Cidadania (MEPEC)*, v. 3, p. 21, 2018.

NEWELL, A.; SHAW, J. C.; SIMON, H. A. Elements of a theory of human problem solving. *Psychological review*, American Psychological Association, v. 65, n. 3, p. 151, 1958.

PRADO, A. S.; SOUZA, S. d. Problema de alocação de salas em cursos universitários: um estudo de caso. *Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional*, p. 11–12, 2014.

QU, R. et al. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, Springer, v. 12, n. 1, p. 55–89, 2009.

RADCLIFFE, N. J.; SURRY, P. D. Formal memetic algorithms. In: SPRINGER. *AISB Workshop on Evolutionary Computing*. [S.l.], 1994. p. 1–16.

RESENDE, M. G.; MATEUS, G. R.; SILVA, R. Grasp: Busca gulosa aleatorizada e adaptativa. *Parte: http://hdl. handle. net/10316.2/5655*, Imprensa da Universidade de Coimbra, 2012.

RESENDE, M. G.; RIBEIRO, C. C. A grasp for graph planarization. *Networks: An International Journal*, Wiley Online Library, v. 29, n. 3, p. 173–189, 1997.

RESENDE, M. G.; RIBEIRO, C. C. A grasp with path-relinking for private virtual circuit routing. *Networks: An International Journal*, Wiley Online Library, v. 41, n. 2, p. 104–114, 2003.

SCHAERF, A. A survey of automated timetabling. *Artificial intelligence review*, Springer, v. 13, n. 2, p. 87–127, 1999.

SOUZA, M. J. F.; MACULAN, N.; OCHI, L. S. Melhorando quadros de horário de escolas através de caminhos mínimos. *Trends in Applied and Computational Mathematics*, v. 1, n. 2, p. 515–524, 2000.

SOUZA, M. J. F.; MARTINS, A. X.; ARAÚJO, C. R. d. Experiências com simulated annealing e busca tabu na resolução do problema de alocação de salas. 2002.

VIEIRA, L. E. et al. Algoritmo evolutivo para o problema do caixeiro viajante com demandas heterogêneas. Universidade Federal de Santa Maria, 2006.

WREN, A. Scheduling, timetabling and rostering—a special relationship? In: SPRINGER. *International conference on the practice and theory of automated timetabling*. [S.l.], 1995. p. 46–75.

APÊNDICE A

Planilha de Salas

Sala	Ambiente	Ar	Ventilador	Capacidade	Quadro-Giz	Quadro-Branco	Quadro-Vidro	Bloco
T-105	Sala Comum		1	31	1	0	0	Bloco 100
T-106	Sala Comum		1	31	1	0	0	Bloco 100
T-107	Sala Comum			35		1	0	Bloco 100
T-108	Sala Comum			35		1	0	Bloco 100
T-109	Sala Comum			35		1	0	Bloco 100
S-101	Sala Comum		3	36	1	0	0	Bloco 100
S-102	Sala Comum		3	33	0	1	0	Bloco 100
S-103	Sala Comum		2	38	0	0	1	Bloco 100
S-104	Sala Comum		2	30				Bloco 100
S-105	Sala Comum		1	30	1	0	0	Bloco 100
S-106	Sala Comum		2	33	1	0	0	Bloco 100
S-107	Sala Comum		2	32	1	0	0	Bloco 100
S-108	Sala Comum		3	37	1	0	0	Bloco 100
S-109	Sala Comum		3	37	1	0	0	Bloco 100
S-110	Sala Comum		2	43	1	0	0	Bloco 100
T-201	Sala Comum							Bloco 200
T-202	Sala Comum					1	0	Bloco 200
T-203	Sala Comum			15		1	0	Bloco 200
T-204	Sala Comum	0	1	22	0	1	0	Bloco 200
T-205	Sala Comum	0	1	22	0	1	0	Bloco 200
T-206	Sala Comum	0	0	21	0	3	0	Bloco 200
T-207	Sala Comum	0	0	23	0	2	0	Bloco 200
T-216	Sala Comum	2	1	36	0	0	1	Bloco 200
T-217	Sala Comum	1	0	29	0	0	1	Bloco 200

T-218	Sala Comum	0	1	36	0	0	0	Bloco 200
S-201	Sala Comum	0	2	41	0	1	0	Bloco 200
S-202	Sala Comum	0	1	19	1	0	0	Bloco 200
S-203	Sala Comum	0	1	18	0	0	1	Bloco 200
S-204	Sala Comum	0	1	37	1	0	0	Bloco 200
S-205	Sala Comum	0	0	46	0	0	0	Bloco 200
S-206	Sala Comum	0	2	40	1	1	0	Bloco 200
S-208	Sala Comum	0	1	32	1	0	0	Bloco 200
S-209	Sala Comum	0	1	28	1	0	1	Bloco 200
S-210	Sala Comum	0	2	42	1	1	0	Bloco 200
S-212	Sala Comum	2	0	44	0	1	0	Bloco 200
S-213 A	Sala Comum	0	2	29	0			Bloco 200
S-213 B	Sala Comum	0	1	29	0	1	0	Bloco 200
S-214	Sala Comum							Bloco 200
S-215	Sala Comum	1	1	36	0			Bloco 200
S-216	Sala Comum							Bloco 200
S-217	Sala Comum							Bloco 200
S-218	Sala Comum							Bloco 200
T-301		0	0	40	0	0	1	Bloco 300
T-301 B		1	0	24	0	1	0	Bloco 300
T-302		0	2	15	0	1	0	Bloco 300
T-303 B		2	0	27	0	0	0	Bloco 300
T-304		2	0	39	1	0	0	Bloco 300
T-305		2	2	45	1	1	0	Bloco 300
T-306		2	0	43	0	0	0	Bloco 300

T-307		2	2	45	1	0	0	Bloco 300
T-308		0	2	40	1			Bloco 300
S-301								Bloco 300
S-302								Bloco 300
S-303								Bloco 300
S-304								Bloco 300
S-305								Bloco 300
S-306								Bloco 300
S-307						1	0	Bloco 300
S-308		2	2	38	0			Bloco 300
S-309								Bloco 300
T-401 A						1	0	Bloco 400
T-401 B		2	2	38	0	1	0	Bloco 400
T-401 C		1	3	44	0	1	0	Bloco 400
T-401 D		1	1	13	0	1	0	Bloco 400
T-401 E		1	2	11	0			Bloco 400
T-403						0	0	Bloco 400
T-404		0	1	7	1	0	0	Bloco 400
T-405 C		1	2	34	1	0	0	Bloco 400
T-407 B		1	2	26	1	0	0	Bloco 400
T-407 C		0	2	7	1	1	0	Bloco 400
T-408		0	2	19	0	1	0	Bloco 400
T-409		0	1	24	0			Bloco 400
S-401A	LAB INFORM	2		30				Bloco 400
S-401B	LAB INFORM	2		30				Bloco 400

S-401C	LAB INFORM	2		30				Bloco 400
S-401D	LAB INFORM	2		30				Bloco 400
S-402A	COORD. LABS	2		30				Bloco 400
S-402B	LAB INFORM	2		30				Bloco 400
S-403A	LAB INFORM	2		30				Bloco 400
S-403B	LAB INFORM	2		30				Bloco 400
S-403A	LAB INFORM	2		30				Bloco 400
S-403B	LAB INFORM	2		30				Bloco 400
S-403C	LAB INFORM	2		30				Bloco 400
S-403D	LAB INFORM	2		30				Bloco 400
S-404	SALA COMUM		2	35				Bloco 400
S-406	SALA COMUM		2	35				Bloco 400
T-502 A		1	0	29	1	1	0	Bloco 500
T-502 B		0	2	23	0	2	0	Bloco 500
T-502 C		3	0	35	1	0	0	Bloco 500
T-502 D		1	1	28	1	1	0	Bloco 500
T-503 C								Bloco 500
T-503 G		1	0	32	1	0	0	Bloco 500
S-501 A		0	2	8	0	1	0	Bloco 500
S-501 B		0	2	30	0	1	0	Bloco 500
S-502 A		0	1	22	0	1	0	Bloco 500
S-502 B		0	2	32	0	1	0	Bloco 500
S-503 A		0	2	35	0	1	0	Bloco 500
S-503 B		0	2	29	0	1	0	Bloco 500
S-504		0	2	34	0	2	0	Bloco 500

S-505		2	4	34	1	2	0	Bloco 500
S-506		1	1	34	0	2	0	Bloco 500
S-507		0	2	33	1	1	0	Bloco 500
S-508		1	1	29	0	0	1	Bloco 500
S-509 ESTUDOS		1	1	14	0	0	0	Bloco 500
S-510		1	1	21	0	0	1	Bloco 500
S-510 B		2	0	21	0	0	1	Bloco 500
DJALMA		1	0	74	0	1	0	Bloco 700
S-702		2	0	31	0	1	0	Bloco 700
S-703 A		1	0	25	0	1	0	Bloco 700
S-703 B		1	0	42	0	1	0	Bloco 700
S-703 C		1	0	24	0	1	0	Bloco 700
LAB 1		0	1	21	0	1	0	Bloco 700
LAB 8		1	0	4	0	1	0	Bloco 700
LAB 5		1	1	4	0	0	0	Bloco 700
LAB 6		0	1	16	0	1	0	Bloco 700
LAB 7		0	1	5	0	0	0	Bloco 700
LAB 9		0	1	2	0	1	0	Bloco 700
LAB 11		0	1	12	0	1	0	Bloco 700
S-801 A		0	1	13	1	0	0	Bloco 800
S-802 A LAB ENSINO		1	0	10	0	0	1	Bloco 800
S-802 D		1	0	12	0	1	0	Bloco 800
S-802 E		1	0	16	0	2	0	Bloco 800
S-803 A		0	1	32	1	0	0	Bloco 800

S-804		1	1	32	1	0	0	Bloco 800
S-805 B		1	0	31	0	0	1	Bloco 800