Universidade Federal Fluminense

JAQUELINE RODRIGUES FOLLY

Otimização do Problema de Alocação de Aulas em Salas no Âmbito do Ensino Superior

VOLTA REDONDA

2017

JAQUELINE RODRIGUES FOLLY

Otimização do Problema de Alocação de Aulas em Salas no Âmbito do Ensino Superior

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional e Pesquisa Operacional.

Orientador:

Tiago Araújo Neves

Coorientador:

Luís Alberto Duncan Rangel

Universidade Federal Fluminense

VOLTA REDONDA

2017

Ficha catalográfica automática - SDC/BEM

R696o Rodrigues Folly, Jaqueline
Otimização do Problema de Alocação de Aulas em Salas no Âmbito do Ensino Superior / Jaqueline Rodrigues Folly ; Tiago Araújo Neves, orientador ; Luis Alberto Duncan Rangel, coorientador. Volta Redonda, 2018.
126 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Volta Redonda, 2018.

1. Problema de Agendamento de Recursos. 2. Meta-Heurística. 3. Simulated Annealing. 4. Modelagem Computacional Exata. 5. Produção intelectual. I. Título II. Neves,Tiago Araújo, orientador. III. Rangel, Luis Alberto Duncan, coorientador. IV. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda.

CDD -

Bibliotecária responsável: Helania Oliveira Madureira - CRB7/4250

Otimização do Problema de Alocação de Aulas em Salas no Âmbito do Ensino Superior

Jaqueline Rodrigues Folly

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Aprovada por:

Prof. Tiago Araújo Neves, D.Sc. / MCCT-UFF

Prof. Luís Alberto Duncan Rangel, D.Sc. / PPGEM-UFF

Prof. Rossana Aparecida Ravaglia, D.Sc. / UniFOA

Prof. Antônio Iacono, D.Sc. / PPGEM-UFF

Prof. Weslley Luiz da Silva Assis, D.Sc. / MCCT-UFF

Prof. Emerson Souza Freire, D.Sc. / MCCT-UFF

Volta Redonda, 23 de Fevereiro de 2018.

Dedico esta obra à Deus, minha família e a todos que me auxiliaram e apoiaram de alguma maneira ao longo deste trabalho.

Agradecimentos

À Deus, por ter me proporcionado forças e me mantido de pé, diante das tormentas, estando ao meu lado nos momentos difíceis.

Aos meus pais, Sebastião Carlos e Lucia Helena que com seus esforços e apesar da simplicidade me proporcionaram condições para realizar cada meta estabelecida.

Aos meus irmãos, que me apoiaram com carinho e de forma incondicional a cada obstáculo encontrado.

Aos meus sobrinhos, Sophia e Miguel responsáveis por toda minha inspiração para vida.

Ao meu orientador e professor, Tiago Araújo Neves pela contribuição e auxílio na elaboração dos algoritmos e incentivo na realização desta dissertação.

Ao meu coorientador, Luís Alberto Duncan Rangel pela dedicação e ajuda na elaboração da modelagem exata.

À todos os professores do MCCT e os demais que contribuíram para a realização deste trabalho.

Resumo

O problema de alocação de aulas em salas em uma instituição de ensino consiste basicamente no arranjo otimizado de aulas, com tempos pré-determinados, em salas, sendo o número total de salas limitado. Este é um típico problema enfrentado pelas universidades no início de todo período letivo, sendo essa tarefa normalmente realizada de forma manual, consumindo um período de tempo considerável. Sendo assim, o presente projeto de dissertação interdisciplinar foi elaborado com o objetivo de resolver problemas de alocação horária de recursos através técnicas de otimização. Inicialmente, foi tentada uma abordagem heurística, utilizando um algoritmo construtivo, juntamente com a meta-heurística *Simulated Annealing*. Posteriormente, foi elaborado um modelo matemático para representar o problema estudado. Testes computacionais revelam que o modelo matemático proposto apresenta melhores resultados, tanto em termos de qualidade de solução quanto em termos de tempo computacional.

Abstract

This work is about University Course Timetabling Problem. It basically consists in the great process allocation of grade course class, with schedules already established, to a set of classrooms considering the room capacity and the needs of professors and students. Every beginning of semester higher education institutions face the same trouble, to allocate courses to classrooms subject to certain restrictions. This process is usually solved by the institutions manually which not only take several days to complete besides does not guarantee the efficient allocation of spaces. Thus, there is a necessity to automatize the assignment process and to resort to computational strategies capable of yielding quality solutions at low costs. Therefore, this paper addresses the classroom assignment problem motivated by a real problem case in a EEIMVR university. For the solution of this problem, use was made of the meta-heuristic Simulated Annealing starting from a random algorithm, which has proven satisfactory for resolving these kinds of problems. Next, it was created an appropriate mathematical model for the allocation of classrooms in the same university, as an alternative that heuristic, and it provided better than results when compared each other. Computational tests revealed both approaches are capable to obtain promising solutions. But, the mathematical model obtains better solutions, in quality solution and processament time. Besides that, both model provided high quality solutions when compared to the manual solution.

Palavras-chave

1. Otimização Combinatória
2. Problema de Alocação de Aulas em Salas
3. Meta-heurísticas
4. *Simulated Annealing*
5. Modelagem Computacional Exata

Glossário

UFF	:	Universidade Federal Fluminense
EEIMVR	:	Escola de Engenharia Industrial Metalúrgica de Volta Redonda
NP-hard	:	Não Polinomial-Difícil
PAAS	:	Problema de Alocação de Aulas em Salas
SA	:	*Simulated Annealing*
TS	:	*Tabu Search*
STP	:	*School Timetabling Problem*
CTP	:	*Course Timetabling Problem*
ETP	:	*Examination Timetabling Problem*
CO	:	*Combinatorial Optimization*
OFISP	:	*Problema Operacional de Agendamento de Horários Fixos*
FSP	:	*Problema de Agendamento de Horários Fixos*
MaxFSP	:	*Problema de Agendamento de Horários Fixos Máximo*
AGc	:	*Algoritmo Genético Compacto*
TSP	:	*Travelling Salesman Problem*
VLSI	:	*Very-large-scale integration*
URL	:	*Uniform Resource Locator*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Considerações Iniciais

O uso de algoritmos na resolução de problemas reais é uma prática que tem se mostrado cada vez mais comum entre programadores e pesquisadores da área. Entende-se por algoritmo qualquer procedimento computacional bem sucedido que tomado algum valor, ou conjunto de valores, como dado de entrada, *input*, é então processado e obtido algum dado de saída, *output*.

Dentre os muitos algoritmos de destaque estão aqueles que fornecem soluções ótimas para os modelos de otimização relacionados à Pesquisa Operacional (programação linear, não-linear e inteira). Porém, alguns desses problemas podem ser tão complicados que, no estado da arte atual em termos de algoritmos e *hardware*, não é possível obter a solução ótima desejada (soluções exatas). Diante disso, se faz necessário o uso de alguma técnica que forneça pelo menos uma solução próxima à ótima, ou concluir que tais soluções, na verdade, não existem. Na prática, os métodos heurísticos, em alguns casos os meta-heurísticos, geralmente atendem nossas expectativas com "boas" soluções [19]. Porém, o inconveniente no uso da heurística está na dificuldade que esta possui em escapar de ótimos locais, daí a escolha das meta-heurísticas, visto que, ao contrário das heurísticas convencionais,

possui a capacidade de explorar ao máximo o espaço de busca até atingir o ótimo global.

As meta-heurísticas representam uma família de técnicas de otimização aproximada que ganhou notoriedade nas últimas décadas, por gerar soluções aceitáveis, em tempos computacionais razoáveis, na resolução de problemas complexos de ciência e engenharia, o que explica o crescimento no interesse por seu domínio. Ao contrário dos algoritmos de otimização exatos, as meta-heurísticas não garantem a otimalidade das soluções obtidas, mas, provavelmente encontrarão excelentes soluções viáveis.

No passado, algumas grandes variedades de soluções eram consideradas aceitáveis. Em projetos de engenharia, por exemplo, eram comuns incluir um grande fator de segurança. Hoje, no entanto, devido a competição econômica acirrada, já não é mais suficiente desenvolver um projeto aceitável sem que este faça uso de forma mais eficiente de seus recursos escassos. Nesse sentido, o impacto da tecnologia aliado ao advento da era do computador tem fornecido oportunidades de resolver problemas de grandes proporções que poderiam ter sido mal resolvidos no passado, mas, que atualmente permite-se fazê-los com maior "refinamento". A resolução de problemas relacionados ao planejamento de recursos é comum no dia a dia operacional dos gestores, e podem ser encontrados em diversas áreas como: telecomunicações, elaboração de integração em grande escala (VLSI), análise financeira, **agendamento de recursos**, **planejamento de espaços**, distribuição de energia, engenharia molecular, logística, elaboração de referenciais, manufatura flexível, gerenciamento de desperdício, exploração de minérios, análises biomédicas, conservação ambiental e dezenas de outras [18]. E é no contexto da associação dos temas, agendamento de recursos e planejamento de espaços, que se encontra o problema assunto desse trabalho.

O problema de alocação de aulas em salas, ou simplesmente PAAS, em uma instituição de ensino consiste basicamente no arranjo otimizado de aulas, com tempos pré-determinados, em salas, sendo o número total de salas limitado. Algumas

restrições devem ser respeitadas como capacidade das salas, demanda de alunos e número de encontros diários, de modo que duas aulas distintas não sejam alocadas simultaneamente na mesma sala e no mesmo horário, enquanto outras questões devem ser otimizadas, como por exemplo a disparidade entre as demandas das turmas e a capacidade das salas, que deve ser minimizada.

Apesar da definição simples, este problema possuí inter-relações que o tornam pertencente à classe de problemas NP-Difícil (Não-Polinomial-Difícil) [35]. Ou seja, não se conhecem algoritmos capazes de obter a solução do problema em tempo razoável.

Sendo assim, o presente trabalho se propõe a resolver o problema de alocação de aulas em salas de uma instituição de ensino superior através de uma ferramenta computacional sugerida, para auxiliar seus decisores no momento da montagem do quadro de horários.

Para isso, inicialmente, propôs-se uma solução inicial construída a partir de um algoritmo aleatório. Ao perceber que a solução fornecida pelo algoritmo não possuía qualidade satisfatória, utilizou-se a meta-heurística *Simulated Annealing* (SA) no refinamento dessa solução inicial, visto que, este método é frequentemente usado quando o espaço de busca é discreto, exatamente como o caso estudado neste trabalho. Para atingir este fim, foram utilizadas duas estruturas de vizinhança distintas na exploração do espaço de soluções. Apesar do algoritmo SA possibilitar a obtenção de uma solução viável em um curto tempo computacional, quando comparado à forma manual, sua construção mostrou que diversos casos devem ser analisados para construção e refinamento de soluções. Como a enumeração de todos casos se tornou complexa, uma abordagem matemática, que visa modelar o problema através de equações, também foi proposta. Os resultados computacionais mostram que ambas as abordagens podem gerar soluções viáveis para o problema em tempo razoável, sendo que o modelo matemático oferece as soluções de melhor qualidade.

Para tanto, foram utilizados os dados disponibilizados pela Escola de Enge-

nharia Industrial Metalúrgica de Volta Redonda, uma unidade da Universidade Federal Fluminense. Os dados são referentes ao segundo período letivo do ano de 2014.

1.2 Justificativa

Este é um típico problema enfrentado pelas universidades no início de todo período letivo, e geralmente, este processo de alocação é realizado de forma manual por estas instituições, consumindo um período de tempo considerável, além de frequentemente não fornecer soluções que atendam, de forma satisfatória, às necessidades das instituições. Além disso, o elevado número de combinações possíveis de alocação dada a limitação humana, torna esta tarefa demorada, além de não garantir uma alocação eficiente dos espaços devido a complexidade envolvida, o que inviabiliza a alocação manual. Somado a isso, a escassez de salas de aula, o aumento da oferta por novos cursos universitários, a capacidade das salas dada a demanda dos alunos acrescida à disponibilidade dos docentes das disciplinas devido suas restrições horárias, são outros fatores que aumentam a complexidade do problema estudado.

Diante disso, a automatização do processo de alocação é uma tarefa eficiente que auxilia na descoberta de soluções viáveis e de qualidade, apresentando uma redução significativa no tempo de resposta para obtenção da solução desejada.

1.3 Objetivos

O presente trabalho tem como objetivo resolver o problema de alocação de aulas em salas da Escola de Engenharia Industrial Metalúrgica de Volta Redonda, uma das noves unidades pertencentes à Universidade Federal Fluminense, através de duas ferramentas computacionais desenvolvidas, método exato e aproximado, possibilitando decisões mais assertivas, por parte dos tomadores de decisão, no ato

da montagem do quadro de horários desta unidade específica.

1.3.1 Objetivo Geral

Automatizar a resolução do problema de alocação de aulas em salas dessa escola de engenharia, fazendo com que esta tarefa possa ser desempenhada de maneira menos árdua por seus responsáveis e melhorando a qualidade da solução apresentada.

1.3.2 Objetivos Específicos

Como objetivos específicos tem-se:

1. O modelamento do problema por meio de ferramentas de otimização e;
2. A construção de ferramentas computacionais para tratar os modelos propostos.

1.4 Processo Metodológico

O desenvolvimento deste trabalho se baseou em questões teóricas e práticas, bem como, nas experiências prévias dos autores sobre a modelagem de sistemas classificados como NP-Difícil, visando possibilitar ao leitor, um alicerce bem fundamentado nas resoluções de problemas relacionados ao tema principal, agendamento de recursos.

Com esse objetivo em mente, realizou-se uma vasta pesquisa em busca de assuntos que retratassem problemas similares ao estudado neste trabalho. Nesta fase, além da leitura de artigos, utilizou-se as principais literaturas, como os clássicos [18] e [19], na elaboração de conceitos e teorias utilizadas nesta dissertação. A partir daí, segundo algumas características singulares de alguns artigos estudados,

optou-se por selecionar quatro dentre todos, para serem apresentados de forma sintetizada na Seção 3.

Adicionalmente ao embasamento teórico, realizou-se alguns procedimentos experimentais como forma de fundamentar toda teoria desenvolvida ao longo do texto. Como o objetivo do presente trabalho é resolver o problema de alocação de encontros, através da modelagem exata e aproximada, foi necessário a utilização de algumas ferramentas, na interface entre a máquina e o programador, na obtenção da solução desejada, tais como: software de otimização *CPLEX*, Versão 12.6 e o ambiente de desenvolvimento integrado (IDE) - CODE::BLOCKS, que permitiu escrever todo o código em linguagem de programação C.

O uso do *CPLEX* permitiu desenvolver um modelo exato viável, em termos de solução e tempo computacional, para o problema principal abordado. Além disso, com auxílio do CODE::BLOCKS foi possível modelar a meta-heurística *Simulated Annealing* adaptada para o mesmo problema em questão. Todas as informações necessárias na elaboração do experimento são oriundas do ano letivo de 2014.

Como forma complementar, ao fim dos experimentos, foi feita uma comparação entre as soluções obtidas pelos dois diferentes métodos e pela forma manual, atualmente utilizada pela instituição de ensino. E a partir daí, foram realizadas as devidas considerações, conclusões e possíveis sugestões para trabalhos futuros.

1.5 Estrutura Organizacional

Do ponto de vista estrutural, o restante do trabalho segue organizado da seguinte maneira.

O Capítulo 2 menciona toda revisão bibliográfica que será utilizada ao longo deste trabalho, incluindo desde, conceitos de problemas de Otimização Combinatória até os Métodos de Resolução existentes, denominados Exatos e Aproximados.

O capítulo subsequente, Capítulo 3, está organizado como trabalhos relacio-

nados, e se destina a fazer uma breve síntese de casos semelhantes ao abordado neste trabalho, *University Course Timetabling Problem*. Neste, encontram-se alguns exemplos que fazem menção ao assunto estudado, e permite ter uma visão holística do problema principal.

Uma vez que os conceitos básicos para problemas de otimização e algoritmos de resolução foram definidos, e exemplos no mesmo contexto foram apresentados, é possível descrever, através do Capítulo 4, o estudo de caso desenvolvido, com as devidas caracterizações do problema, mecanismos de funcionamento e métodos de resolução utilizados.

Então, com os conceitos do problema de alocação de aulas em salas bem definidos, e em posse dos dados e resultados obtidos a partir dos experimentos realizados, elaborou-se o Capítulo 5 que traz a descrição, o desenvolvimento e expõe os resultados obtidos em cada experimento, bem como, uma análise dos mesmos para que se justifique a realização deste trabalho.

Finalmente, no Capítulo 6 encontra-se e discute-se, respectivamente, as conclusões e considerações finais, as quais são usadas para propor estudos de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O Problema de Alocação de Aula em Salas é encontrado na literatura como um problema de *Timetabling*, visto que, seu conceito é definido por uma alocação horária de recursos. Neste contexto, este tipo de problema consiste no agendamento de uma sequencia de encontros entre professores e estudantes, em um período de tempo pré-definido (geralmente semanal), satisfazendo um conjunto de restrições de vários tipos, definição utilizada por Schaerf [30].

Existe um grande número de variantes para o problema de programação de horários encontrados na literatura, no qual, a diferença de um para outro se dá no tipo de instituição envolvida (universidades ou escolas) e suas restrições. Dessa maneira, classificam-se os Problemas de *Timetabling* em três categorias principais:

a) ***School Timetabling Problem - STP*** (Problemas de programação de horários em escolas): se referem à programação horária de aulas em salas, de instituições com atributos típicos de uma escola de nível fundamental ou médio, envolvendo professores, alunos que possuem o mesmo currículo e um conjunto de horários, geralmente semanal, estabelecidos para realização de cada aula, evitando que professores e turmas tenham duas aulas em um mesmo horário e sala. Em problemas dessa natureza, cabe ressaltar alguns aspectos importantes que causam a rigidez dos horários cedidos para a reali-

zação das aulas. Devido a esse aspecto, os horários geralmente são fixos a um determinado turno, sendo possível utilizar parte de outro turno para complementar as aulas que não puderam ser lecionadas no turno selecionado. Além disso, geralmente as aulas possuem a mesma duração, podendo, em alguns casos, possuírem durações diferentes, se necessário. Devido às características das turmas (conjuntos disjuntos) todas as aulas são lecionadas em uma mesma sala, sendo assim, são os professores quem se deslocam para a realização das aulas;

b) ***Course Timetabling Problem - CTP*** (Problemas de programação de horário de cursos): se referem à programação horária semanal de cursos para todos os períodos dos currículos de uma instituição de nível superior, respeitando as disponibilidades e capacidades das salas e evitando que duas aulas distintas sejam alocadas em uma mesma sala num mesmo horário. Nesse problema considera-se um conjunto de cursos (Cálculo I, Cálculo Vetorial, Álgebra Linear etc) que necessitam de um determinado número de encontros para que seja concluído, um conjunto de currículos (Engenharia de Produção, Engenharia Mecânica e etc), sendo que, cada currículo possui um determinado número de cursos. Os estudantes matriculam-se em turmas dos cursos de seus currículos. Uma turma pode possuir estudantes de currículos diferentes. Além disso, um conjunto de horários, geralmente semanal, são disponibilizados para a realização das aulas, e em cada horário um número limitado de salas. Diferentemente do que ocorre no STP, existe uma flexibilidade com relação aos horários disponibilizados para a realização das aulas, ou seja, um curso pode ser lecionado a qualquer horário de funcionamento da instituição, inclusive aos sábados. Devido às características desse problema, as aulas são agrupadas em sessões, ou seja, podem ser distribuídas da seguinte maneira: uma de três horas-aula, e outra de duas horas-aula totalizando cinco horas-aula semanais. Além disso, a natureza das turmas permite que sejam os alunos a se deslocarem para terem suas aulas [32];

c) ***Examination Timetabling Problem - ETP*** (Problemas de programação de horários de exames): esta categoria se propõe a fazer uma alocação otimizada dos exames de cursos universitários de modo a evitar a sobreposição de suas datas que possuem estudantes em comum, além de, distanciar ao máximo as datas entre exames.

No contexto do CTP, Souza [32] o define como o problema de alocar as aulas dos cursos aos horários disponibilizados, respeitando essas disponibilidades e capacidades das salas existentes, de forma que nenhum estudante tenha duas ou mais aulas simultaneamente. Nesta circunstância, o problema de programação de horários de cursos se subdivide em outras categorias como: *Class-Teacher Timetabling*; *Student Scheduling*; *Teacher Assignment* e *Classroom Assignment*. E é nessa última que se encaixa o PAAS, ou seja, como uma variante do CTP que consiste no agendamento de disciplinas à salas dado um horário já estabelecido.

2.1 *Classroom Assignment* - Problema de Alocação de Aulas em Salas

Conforme descrito na seção anterior o PAAS é uma variação do problema básico de programação de horários de cursos. Desta maneira, segundo Souza [32], este se configura como um conjunto de cursos (ex: Cálculo I, Álgebra Linear, Desenho Básico, etc.), sendo que para cada curso, existe determinado número de encontros, e cada encontro possui uma certa duração em horas-aula. É importante ressaltar que os encontros de um mesmo curso podem ter durações distintas. Por exemplo, pode haver um curso de cálculo I com dois encontros semanais, um de duas horas-aula e um de três horas-aula. Além disso, existe também um conjunto de currículos (Engenharia Metalúrgica, Engenharia Mecânica, Engenharia de Produção etc.). Cada currículo possui um conjunto de cursos que devem ser frequentados pelos estudantes. Estes por sua vez se matriculam em turmas dos cursos de seu currículo. Uma turma de um curso pode ter estudantes de currículos diferentes.

Há, também, um conjunto de horários disponibilizados para a realização dos encontros e, em cada horário, um número limitado de salas. O problema consiste em fazer a alocação dos encontros dos cursos em horários disponibilizados, respeitando as disponibilidades e capacidades das salas existentes, não permitindo que dois ou mais encontros ocorram simultaneamente na mesma sala, em um mesmo horário.

O PAAS pode ser classificado como NP-Difícil [34]. Talbi, em [35], sugere que encontrar a solução ótima em tempo computacional razoável não é uma tarefa simples para problemas desta classe, sendo necessário, em muitos casos, o uso de algoritmos aproximados para encontrar soluções viáveis dentro de um tempo aceitável. Por esse motivo encontra-se muito na literatura o uso de meta-heurísticas no intuito de se obter esse tipo de resultado, visto que, este é um método de resolução geral que fornece tanto uma estrutura quanto diretrizes de estratégia gerais para desenvolver um método heurístico específico, que se ajuste a um tipo de problema específico. Antes de abordar aspectos mais específicos do PAAS, é preciso citar alguns conceitos que serão amplamente utilizados ao longo do texto. Para isso, a Seção 2.2, apresentada a seguir, traz de maneira mais formal alguns conceitos de área de Otimização.

2.2 Problema de Otimização Combinatória e Métodos de Resolução

Um problema de otimização combinatória pode ser definido por um conjunto finito dado por $E = \{1, ...\eta\}$, um conjunto de soluções viáveis $F \subseteq 2^E$ e uma função objetivo formulada como $f : 2^E \longrightarrow \mathbb{R}$, todos definidos para cada problema específico [15].

De acordo com a definição dada por [34] o PAAS é um problema clássico de Otimização Combinatória (OC) pertencente à classe *NP-Difícil*, em que a determinação da solução ótima do problema, em um período de tempo aceitável, não é uma tarefa simples. Nesse tipo de problema, há uma função objetivo, o qual deve-

se otimizar, minimizando-a ou maximizando-a, dependendo da natureza da função objetivo, acompanhada de um conjunto de restrições que estão inter-relacionadas por meio das variáveis de decisão. Estas variáveis podem assumir valores delimitados por restrições impostas sobre cada variável, formando um conjunto discreto de soluções para o problema. Neste caso, dependendo da complexidade da função objetivo e do conjunto de restrições, fica difícil determinar a solução do problema, mesmo com o benefício da utilização de gráficos.

Os problemas de otimização podem ser resolvidos por **Métodos Exatos** ou **Métodos Aproximados**. Os métodos exatos encontram a solução ótima, ou seja, a solução viável que melhor se adéqua à função objetivo. No entanto, para problemas das classes NP-Completa e NP-Difícil, como o problema tratado neste trabalho, algoritmos exatos tendem a utilizar um longo tempo computacional, de ordem exponencial, para encontrar a solução ótima [35].

Já os Métodos Aproximados não buscam a solução ótima, mas sim uma solução de alta qualidade em um tempo razoável para uso prático. É importante notar que alguns destes algoritmos podem encontrar a solução ótima, mas não há garantias de que isto irá ocorrer em todos os casos. A Figura 2.1 mostra um gráfico hierárquico (organograma) com os principais métodos de otimização clássicos, exatos e aproximados, existentes na literatura.

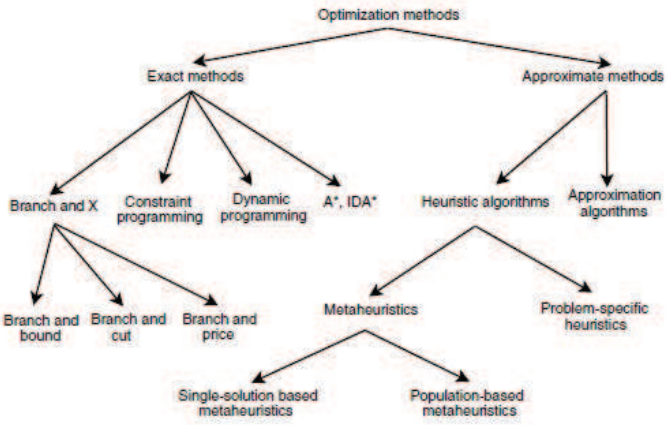


Figura 2.1: Métodos de Otimização Clássicos [35]

Em termos de algoritmos exatos uma abordagem bastante empregada e bem sucedida para se resolver Problemas de OC, segundo [2], consiste em formulá-lo como um problema de Programação Linear Inteiro e empregar algum algoritmo para avaliação de seus limites duais. Este, por sua vez, é inserido em um algoritmo de enumeração inteligente do tipo *Branch-and-bound*, que é exatamente a metodologia utilizada neste trabalho. Para que algoritmos *Branch-and-bound* sejam capazes de resolver instâncias de dimensões de interesse prático em tempos computacionais aceitáveis, é necessário que as formulações de programação inteira empregadas forneçam limites duais fortes, isto é, que as formulações empregadas forneçam uma boa aproximação da envoltória convexa das soluções viáveis do problema. É importante mencionar que determinar qual é a envoltória convexa do problema nem sempre é trivial e, portanto, construir uma formulação matemática eficiente pode ser uma tarefa árdua. A Seção 2.2.1, apresentada a seguir, mostra os preceitos básicos para a construção de uma formulação matemática e alguns modelos gerais.

2.2.1 Métodos Exatos

O rápido aumento no tamanho e na complexidade dos problemas reais tem estimulado o uso de uma abordagem sistemática na resolução desses problemas. Antes de tudo, um sistema pode ser visto como um todo, isso é necessário para entender como os componentes do sistema interagem entre si. Inicialmente, a aplicação da pesquisa operacional pós-guerra no contexto industrial foi principalmente utilizando programação linear e análises estatísticas. Desde aquele tempo, procedimentos eficientes e códigos computacionais tem sido desenvolvido para manusear tais problemas. Nesse sentido, esta seção busca introduzir o problema de programação não linear e discutir situações simples que dão origem a este problema. Além disso, algumas orientações para construção de modelos e formulários serão apresentados.

2.2.1.1 Definições Básicas de Programação Não-Linear

Considere o problema de programação não-linear apresentado pelas Equações (2.1), (2.2), (2.3) e (2.4):

$$Min \quad f(x) \tag{2.1}$$

$$S.a.$$

$$g_i(x) \leq 0, \quad \text{para} \quad i = 1, ..., m, \tag{2.2}$$

$$h_i(x) = 0, \quad \text{para} \quad i = 1, ..., l, \tag{2.3}$$

$$x \in X, \tag{2.4}$$

onde;

f, g_1,..., g_m, h_1, ..., h_l são funções definidas no $\mathbb{R}^n$

X é um subconjunto de $\mathbb{R}^n$

x é um vetor de n componentes $x_1, ..., x_n$

O problema definido por (2.1)-(2.4) deve ser resolvido para os valores das variáveis $x_1, ..., x_n$ que satisfazem as restrições e ao mesmo tempo minimiza a função objetivo, que, neste caso, é representada pela função f.

Cada uma das restrições $g_i \leq 0$ para $i = 1, ..., m$ é chamada de restrição de desigualdade. E cada uma das restrições $h_i = 0$ para $i = 1, ..., l$ é chamada restrição de igualdade.

O conjunto X incluem limites inferior e superior nas variáveis. Alternativamente, este conjunto X pode representar restrições estruturadas que possuem significado especial para ser explorada pela rotina de otimização, ou ele pode re-

presentar certa região de contenção, ou outras restrições complicadas que devem ser tratadas separadamente através de um mecanismo especial.

Um vetor $x \in X$ satisfazendo todas as restrições é chamado uma solução viável para o problema. E a coleção de todas as soluções da região viável.

O problema de programação não-linear, então, é encontrar um ponto viável $\bar{x}$ tal que: $f(x) \geq f(\bar{x})$, para todos ponto x viável.

Este ponto $\bar{x}$ é chamado solução ótima, ou simplesmente uma solução, para o problema. Se existir mais de um ponto $\bar{x}$ ideal, eles são referidos coletivamente como soluções ótimas alternativas.

Um problema de programação não-linear pode ser declarado como um problema de maximização, e a desigualdade das restrições pode ser escrita na forma: $g_i(x) \geq 0$, para $i = 1, ..., m$.

No caso especial onde a função objetivo é linear e quando todas as restrições, incluindo o conjunto X, podem ser representadas equações e/ou inequações lineares, o problema é chamado de Problema de Programação Linear.

Vários problemas práticos podem ser modelados como Problema de Programação Linear. Entre eles: o Problema de Roteamento de Veículos, o Problema de Localização de Facilidades, o Problema de Corte e o Problema de Alocação de Recursos [12]. Devido a relação íntima entre este último e o problema estudado neste trabalho, na sequência apresenta-se uma formulação matemática simples para o Problema de Alocação de Recursos usando os conceitos apresentados.

Considere o problema de programação linear apresentado pelas Equações (2.5), (2.6) e (2.7):

$$Max \quad c^t x \tag{2.5}$$

$$S.a.$$

$$Ax \leq b, \tag{2.6}$$

$$x \geq 0, \tag{2.7}$$

onde c e x são vetores de dimensão n;

b é um vetor de dimensão m;

$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$ é uma matriz de dimensão m x n.

O problema acima pode ser interpretado como um modelo de alocação de recursos da seguinte maneira:

Suponha que existam m recursos, e a quantidade de cada um deles está armazenada no vetor. Assim, b_1, representa a quantidade disponível do recurso um, b_2 a quantidade disponível do recurso dois, e assim sucessivamente.

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

A coluna a_j de A representa uma atividade j,

A variável x_j representa o nível da atividade a ser selecionada,

Para realizar uma atividade j, são necessários a_{kj} unidades de cada recurso k. Portanto, no nível x_j, são necessárias $a_{kj}\ x_j$ unidades de cada recurso k. Consequentemente, considere a restrição apresentada pela Equação (2.8):

$$Ax = \sum_{j=1}^{n} a_j x_j \leq b \tag{2.8}$$

Se o lucro unitário da atividade j é c_j, o lucro total é determinado pela Equação (2.9):

$$\sum_{j=1}^{n} c_j x_j = c^t x \tag{2.9}$$

Objetivo: Encontrar a melhor maneira de alocar o recurso b para as diversas atividades disponíveis, tal que o lucro total seja maximizado.

2.2.2 Métodos Aproximados

Apesar de serem capazes de encontrar as soluções ótimas, os métodos puramente exatos, no caso geral, possuem alta complexidade de resolução, o que exige um alto custo computacional por seu uso, especialmente se o problema contiver variáveis inteiras. Isto implica que o emprego deste método na resolução de problemas pode consumir tempos de ordem exponencial, ainda que o problema tratado fosse de dimensão mediana. Então, são nestes casos que a utilização de métodos exclusivamente exatos, isto é, soluções ótimas computacionais, torna-se praticamente inviável para vários problemas de otimização industrial, fazendo-se necessário o uso de outras técnicas, como alternativa, na obtenção de soluções de qualidade, ou seja, próximas à ótima em tempos computacionais aceitáveis. Os métodos heurísticos são comumente usados para procurar essas soluções, sendo as meta-heurísticas as mais indicadas por serem dotadas de procedimentos de busca local, permitindo que sejam capazes de escapar de Ótimos Locais em busca do Ótimo Global.

Em suma, as meta-heurísticas podem ser definidas como metodologia geral de alto nível que podem ser usadas como um guia estratégico na elaboração de heu-

rísticas adjacentes para resolver problemas específicos de otimização. Porém, diferentemente do que o nome algoritmos aproximados sugere [35], as meta-heurísticas não fornecem uma estimativa da distância entre as soluções obtidas e as soluções ótimas. Apenas garantem a obtenção de "boas" soluções para esses problemas. Uma breve apresentação sobre meta-heurísticas e seus conceitos é apresentada na próxima seção.

2.2.2.1 Meta-Heurísticas

Como mencionado as meta-heurísticas são métodos de otimização classificados como métodos aproximados por fornecerem soluções próximas às ótimas em baixo tempo computacional. Em suma, são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, que consiste na aplicação, em cada passo, de uma heurística subordinada, a qual será modelada para cada problema específico [32].

2.2.2.2 Meta-Heurística baseada em Solução Única

Os problemas de otimização podem se apresentar de diversas maneira. E ainda, são duas as formas de meta-heurísticas que se propõem a resolvê-los. Estas podem ser baseadas em solução única e meta-heurística baseada em população. Devido a natureza do problema tratado neste trabalho, dar-se-á ênfase somente a classe de meta-heurística baseada em solução única.

Este tipo de meta-heurística se propõe a melhorar iterativamente uma solução. Ela pode ser vista como "caminhadores" [19] através da vizinhança ou trajetória de busca ao longo do espaço de busca do problema manuseado. Esses "caminhadores" são manuseados por processos iterativos que se movem de uma solução corrente para outra no espaço de busca.

A iteração da meta-heurística de solução única aplica o procedimento de geração e substituição da solução única corrente (atual). A Figura 2.2 apresenta um

esquema da iteração da meta-heurística.

1. Na **fase de geração**, um conjunto de soluções candidatas é gerado pela solução corrente S. Este conjunto $C(S)$ é geralmente obtido pela transformação local da solução.

2. Na **fase da substituição**, é realizada uma seleção no conjunto de soluções candidatas $C(S)$ para substituir a solução corrente, que é, uma solução $S' \in C(S)$ selecionada para ser a nova solução.

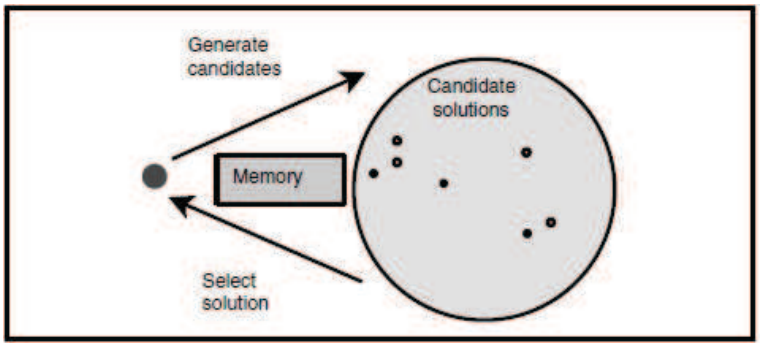


Figura 2.2: Iteração da Meta-Heurística de Solução Única [35]

Este processo de iteração ocorre até que um dado critério de parada seja atingido. A fase de geração e substituição pode ser sem memória (como a meta-heurística SA utilizada neste trabalho). Neste caso, os procedimentos são baseados somente na solução corrente. Caso contrário, o histórico de busca na memória (como ocorre na meta-heurística TS) pode ser usada na geração de uma lista de soluções candidatas e na seleção da nova solução. O Algoritmo 1 mostra como estes passos são executados ao longo do processo de otimização.

Algoritmo 1 *Template* de alto nível de Meta-Heurística Solução Única

1: **Input:** Solução inicial $S0$
2: t=0;
3: **Repeat**
4: /* Geração de soluções candidatas (vizinhança parcial ou completa) de S_t */
5: Geração ($C(S_t)$);
6: /* Seleciona uma solução do conjunto $C(S_t)$ para substituir a solução corrente S_t */
7: S_{t+1}=Seleciona ($C(S_t)$);
8: t=t+1;
9: **Until** o critério de parada ser satisfeito
10: **Output** Melhor solução encontrada

O conceito de busca é comum para todas as meta-heurísticas de solução única, é a definição da **Estrutura de Vizinhança** e a determinação da **Solução Inicial**.

2.2.2.3 Noções de Vizinhança

A definição de vizinhança é um passo comum e necessário para projetar qualquer meta-heurística de solução única. Se a estrutura de vizinhança não for adequada ao problema, qualquer meta-heurística falhará em sua resolução.

Uma função vizinhança N é um mapeamento $N : S \longrightarrow 2^{|S|}$ que atribui para cada solução s de S um conjunto de soluções $N(s) \subset S$ [35].

Uma solução s' na vizinhança de s ($s' \in N(S)$) é chamado um vizinho de s. Um vizinho é gerado pela aplicação de um operador que executa movimento m diante de uma pequena perturbação na solução s, vide Figura 2.3. A propriedade principal que caracteriza uma vizinhança é a localidade. A localidade é o efeito sobre a solução ao executar o movimento (perturbação) na representação. Quando pequenas mudanças são feitas na representação, a solução deve revelar pequenas mudanças. Quando isso ocorre, a vizinhança é dita ter uma forte localidade. A localidade fraca é caracterizada pelo grande efeito na solução quando uma pequena

mudança é feita na representação. Em casos extremos de localização fraca, a processo de otimização tenderá a uma busca aleatória no espaço de soluções.

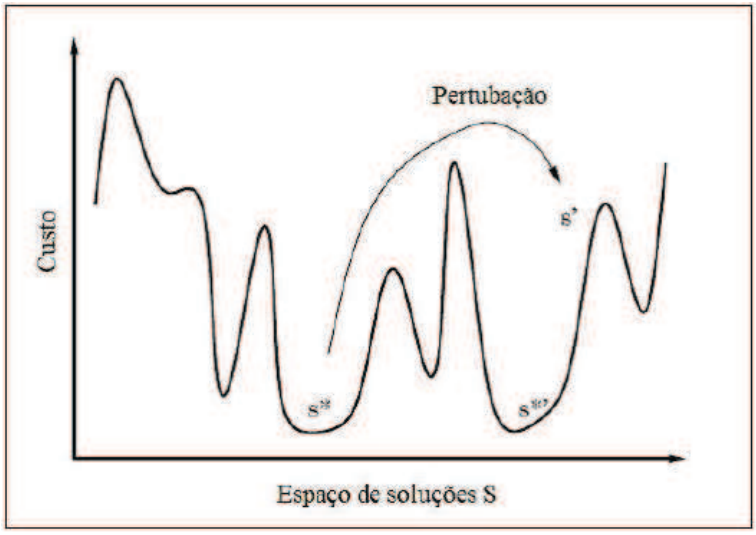


Figura 2.3: Funcionamento da geração de vizinhança. [16]

A estrutura de vizinhança depende do objetivo do problema de otimização. A vizinhança pode ser definida em **Problemas de Otimização Contínua e Discreta**.

A vizinhança $N(s)$ de uma solução s em um espaço contínuo é a esfera com centro s e raio igual a ϵ, com $\epsilon > 0$ [35].

Consequentemente, o conjunto vizinhança será definido por: N(s)$=\{s' \in \mathbb{R}^n \mid ||s'-s|| < \epsilon\}$, sendo $||s'-s||$ a norma Euclidiana, que representa a distância Euclidiana entre s' e s, ilustrado na Figura 2.4. Nesta figura, o círculo representa vizinhança, $N(s)$, de s em um problema contínuo.

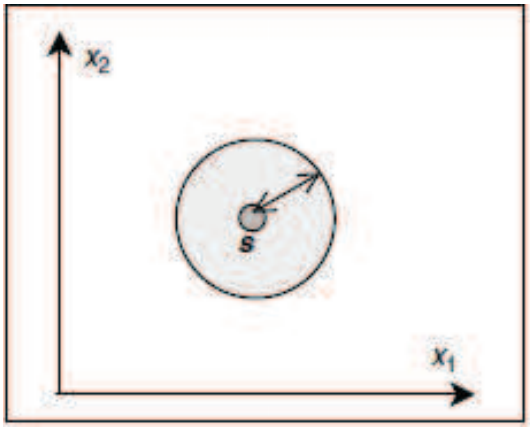


Figura 2.4: Vizinhança em um espaço contínuo.[35]

Em um problema de otimização discreta, a vizinhança $N(s)$ de uma solução s é representada pelo conjunto $(s' \mid d(s', s) \leq \epsilon)$, onde d representa uma dada distância que é relatada pelo movimento do operador [35].

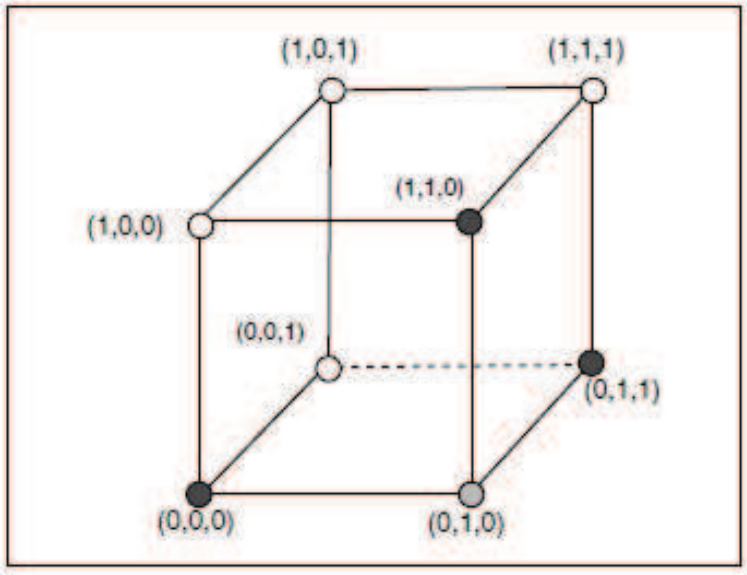


Figura 2.5: Vizinhança para um problema binário de Otimização Discreta [35]

A Figura 2.5 mostra um exemplo de vizinhança discreta para uma solução de representação binária. Na ilustração da vizinhança discreta acima, os nodos do

cubo representam as soluções do problema. Por exemplo, os vizinhos da solução corrente (0,1,0) - Nodo Cinza - são os nodos adjacentes no gráfico - Nodos Pretos.

A definição de vizinhança depende da representação estrutural associada ao problema. Algumas vizinhanças usuais são associadas a codificações tradicionais. A vizinhança natural para representação binária é baseada na **Distância de *Hamming***, que, neste caso, pode ser definida como a quantidade de posições em que as representações das duas soluções diferem. As vizinhanças mais comuns para a representação binária utilizadas na literatura possuem distância de *Hamming* igual a 01 (um), ou seja, os vizinhos de uma solução s diferem desta em apenas uma posição. Assim, um vetor binário de tamanho n terá n vizinhos.

A vizinhança de Hamming para codificação binária pode ser estendida para qualquer representação de vetor discreto usando um dado alfabeto $\sum$. De fato, a substituição pode ser generalizada substituindo o valor discreto de um elemento do vetor por qualquer outro carácter do alfabeto. Se a cardinalidade do alfabeto $\sum$ é K, o tamanho da vizinhança será $(K-1) \times n$ para um vetor discreto de tamanho n.

Porém, para representações baseadas em **permutações**, como a utilizada neste trabalho, uma vizinhança usual é aquela baseada no **Operador *Swap*** ou **Operador de Troca** que consiste em permutar a localização de dois elementos S_i e S_j da permutação. Para uma permutação de tamanho n, o tamanho desta vizinhança é $n \times (n-1)/2$. Este operador pode também ser aplicado em qualquer representação linear. A Figura 2.6 mostra a vizinhança associada a um problema de otimização combinatória usando a codificação permutação. Neste caso, a distância é baseada no movimento do operador *swap*. Ainda nesta Figura 2.6, segundo [24], esta representa um grafo onde os vértices são as soluções (possíveis permutações para um vetor de 3 posições) e os arcos são a relação de vizinhança. Neste sentido, dois vértices são conectados se, e somente se, as soluções representadas por estes vértices são vizinhas, ou seja, se a partir de uma é possível chegar a outra fazendo apenas uma única troca entre duas posições quaisquer do vetor.

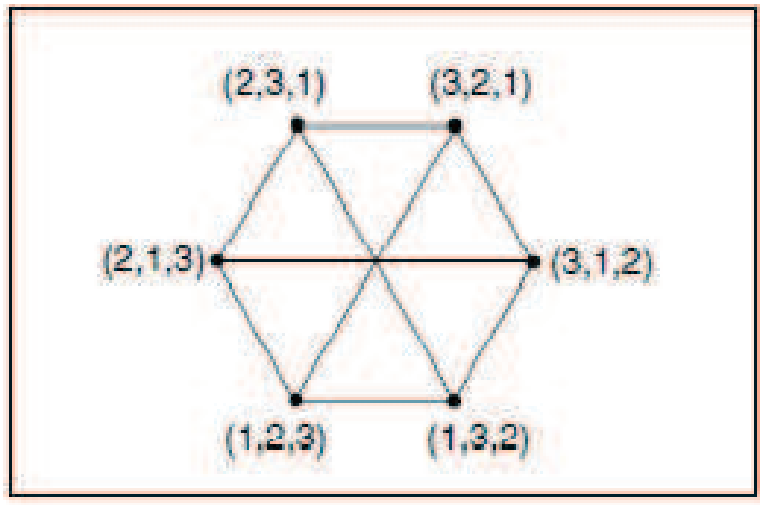


Figura 2.6: Vizinhança associada ao problema OC usando operador *swap* [35]

Ótimo Local: Para uma dada função de vizinhança N, uma solução $s \in S$ é um ótimo local se ele tem uma melhor qualidade que todos seus vizinhos, que é, $f(s) \leq f(s')$, para um problema de minimização, ou ainda, $f(s) \geq f(s')$, para um problema de maximização, para todo $s' \in N(s)$.

Para um mesmo problema de otimização, um ótimo local para uma vizinhança N_1 pode não ser um ótimo local para uma vizinhança N_2 diferente, vide figura abaixo.

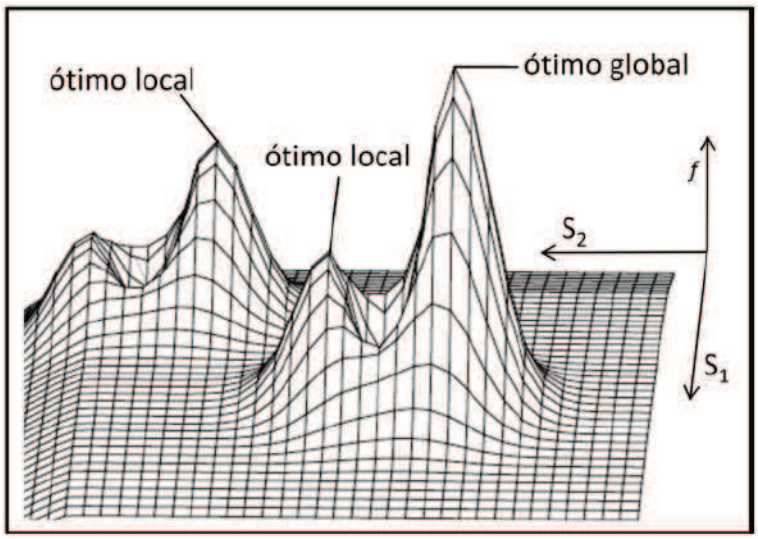


Figura 2.7: Ótimo Local e Global em um Espaço de Busca [25]

A título de melhor compreensão do assunto sobre vizinhança, apresenta-se abaixo, um pequeno exemplo que ilustra como o operador de permuta se comporta quando comparado ao operador de distância da vizinhança.

Exemplo 2.1: k-distância da vizinhança *versus* k-troca da vizinhança

Para este exemplo, será considerado o *Travelling Salesman Problem* (TSP), definido por [24] como um conjunto de cidades que devem ser ordenadas para minimizar a distância total de uma rota que passa por todas as cidades.

Para problemas de permutação (troca ou *swap*), como ocorre no *Travelling Salesman Problem*, o operador de permutação (operador *swap*) pode ser usado como na Figura 2.8. O tamanho da vizinhança é $n \times (n-1)/2$ onde n representa o número de cidades.

Um outro operador muito usado é o operador de permuta-k [35] onde k arestas são removidas da solução e substituídas com outras k arestas. As Figuras 2.8 e 2.9 ilustram a aplicação do operador de permuta-2 e permuta-3, respectivamente, para o PCV. A vizinhança para o operador de permuta-2 é representada por todas

as permutações obtidas pela remoção de 2 arestas.

Ela substitui duas arestas dirigidas (Π_i, Π_{i+1}) e (Π_j, Π_{j+1}) por (Π_i, Π_j) e (Π_{i+1}, Π_{j+1}), onde $i \neq j-1$, j, $j+1$. Formalmente a solução vizinha é definida como:

$$\begin{cases} \Pi'(k) = \Pi(k) \quad \text{para} \quad k \leq i \quad \text{ou} \quad k > j, \\ \Pi'(k) = \Pi(i+j+1-k) \quad \text{para} \quad i < k \leq j. \end{cases}$$

O tamanho da vizinhança para o operador de permuta-2 é $[(n \times (n-1)/2) - n]$, todos os pares de arestas são considerados, menos os pares adjacentes.

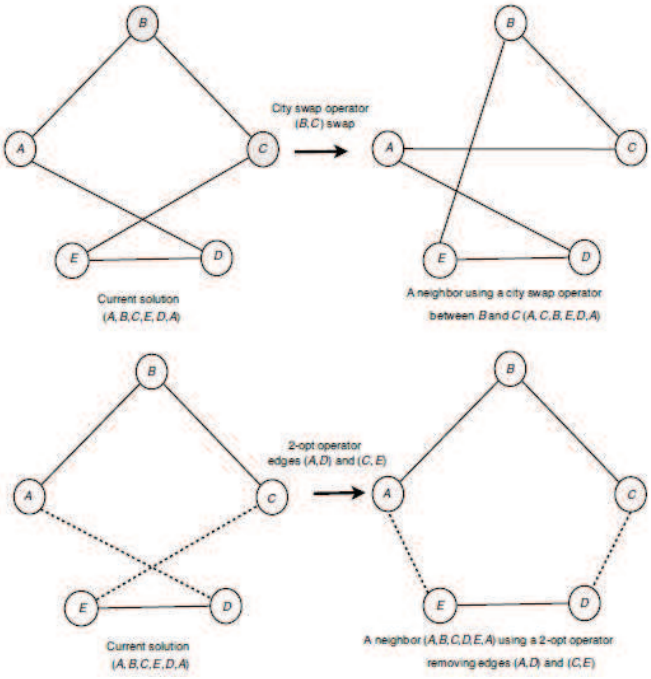


Figura 2.8: Operador de Troca de Cidade e Operador de Permuta-2 para o TSP [35]

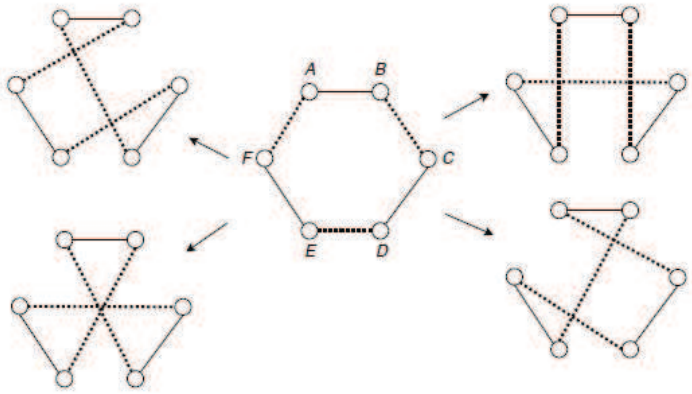


Figura 2.9: Operador de Permuta-3 para o TSP [35]

Para a Figura 2.9 os vizinhos da solução (A,B,C,D,E,F) são (A,B,F,E,C,D), (A,B,D,C.F,E), (A,B,E,F,C,D) e (A,B,E,F,D,C).

Como já mencionado, a eficiência de uma vizinhança não está relacionada somente à representação estrutural, mas também ao tipo de problema a ser resolvido [35]. Por exemplo, em problemas de *scheduling*, a permutação representa uma fila de prioridade. Então, a ordem relativa na sequencia é muito importante. Já no TSP o elemento adjacente ao analisado é importante. Em problemas de agendamento, o operador de permuta-2 irá gerar uma grande variação (localidade fraca), enquanto que para problemas de rotina como TSP, o operador é muito eficiente devido a pequena variação (localidade forte).

Exemplo 2.2: Vizinhança para Problemas de Permutação de Agendamento

Para permutação representando sequenciamento e problemas de agendamento, assunto abordado nesta dissertação, a família de operador de permuta-k não é adequado. Então, o seguinte operador pode ser utilizado:

1) Vizinhança baseada na posição:

A Figura 2.10 ilustra um exemplo de um operador baseado na posição, o ope-

rador de inserção. No operador de inserção, um elemento da sequencia é removido e colocado em outra posição da sequencia. Tanto o elemento removido quanto sua nova posição são selecionados aleatoriamente.

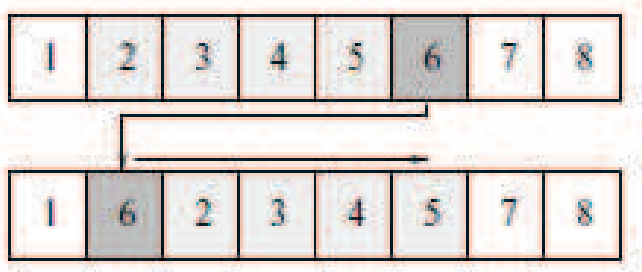


Figura 2.10: Operador de Inserção [35]

2) Vizinhança baseada na ordem:

Existem vários operadores baseados na ordem. Entre eles estão o Operador de Permuta (troca ou *swap*), onde são trocados dois elementos selecionados arbitrariamente (Figura 2.11), e o Operador Inversão, onde a sequencia entre dois elementos selecionados aleatoriamente é invertida (Figura 2.12). Contudo, este último raramente é usado para agendamentos. Seu uso frequentemente cria soluções inviáveis, especialmente para problemas de agendamento com pré-requisitos.

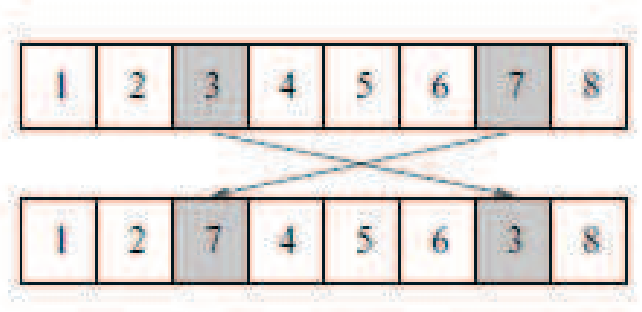


Figura 2.11: Operador de Permuta [35]

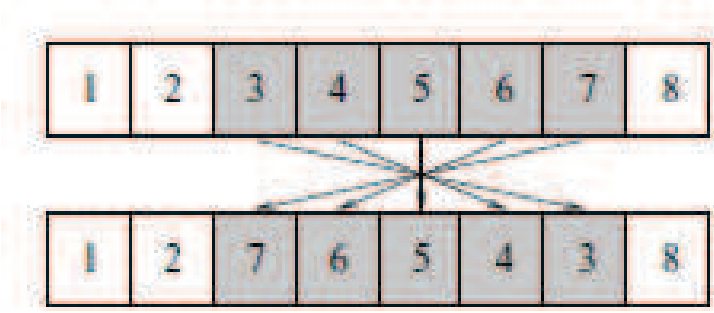


Figura 2.12: Operador de Inversão [35]

Na concepção da meta-heurística de solução única existe um compromisso entre o tamanho (diâmetro) e a qualidade da vizinhança usada e a complexidade computacional para explorá-la. Conceber grandes vizinhanças pode melhorar a qualidade de soluções obtidas desde que mais vizinhos sejam considerados em cada iteração, requerendo assim, um tempo computacional adicional para gerar essa vizinhança, vide Figura 2.13. No entanto, análises detalhadas sobre a relação custo benefício entre o tamanho e qualidade das vizinhanças não faz parte do escopo deste trabalho, sendo este tópico um possível objeto para trabalhos futuros.

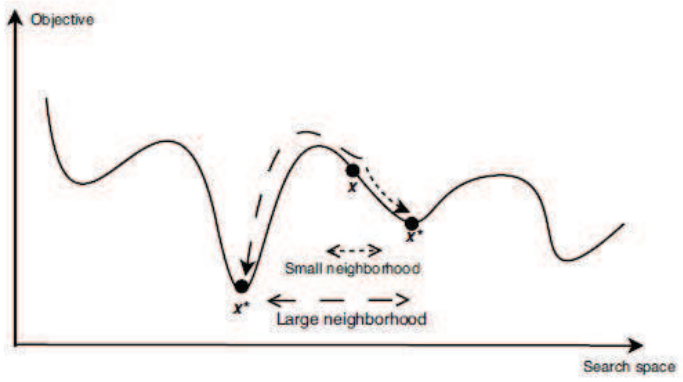


Figura 2.13: Impacto do tamanho da vizinhança em Busca Local. Grande vizinhança melhorando a qualidade da busca despendendo alto tempo computacional [35]

2.2.2.4 Solução Inicial

As principais estratégias para gerar soluções iniciais são: Abordagem Aleatória e Abordagem Gulosa, ou ainda, uma Abordagem Híbrida que associa essas outras duas abordagens na geração de soluções mais aprimoradas para algum problema específico. Há sempre um *trade-off* entre o uso de uma solução inicial aleatória e a solução inicial gulosa em termos de qualidade de soluções e tempo computacional. A melhor resposta para esse dilema dependerá principalmente da eficiência e da eficácia dos algoritmos aleatórios e gulosos manuseados, e as propriedades da meta-heurística de solução única. Por exemplo, quanto maior é a vizinhança, menor é a influência da solução inicial no desempenho dessas meta-heurísticas.

A geração de solução inicial aleatória, método utilizado nesta dissertação, é uma operação rápida, mas, a meta-heurística pode levar maior número de iterações para convergir. Para acelerar a busca, uma heurística gulosa pode ser utilizada. De fato, na maioria dos casos, a combinação de o algoritmo guloso com meta-heuristicas tem uma complexidade de tempo computacional menor. Usando heurísticas gulosas geralmente obtêm-se uma solução inicial de melhor qualidade, ou seja, um ótimo local. Consequentemente, a meta-heurística de solução única irá requerer, no geral, menos iterações para convergir. Algumas aproximações de algoritmos gulosos podem também ser usadas para obter uma garantia do limite para a solução final. Seja como for, isso não significa que usando a melhor solução como solução inicial sempre irá fornecer o melhor resultado.

A experiência obtida com a implementação de soluções iniciais mostra que nem sempre as soluções iniciais gulosas são melhores que as aleatórias [35]. Por exemplo, a heurística de *Clark-Wright* é uma heurística eficiente bem conhecida para o problema do caixeiro viajante e outros problemas de roteamento. As soluções obtidas por essas heurísticas gulosas são muito melhores em qualidade que as soluções aleatórias. No entanto, usando um algoritmo de busca local baseado no movimento do operador de permutação-3, experimentos mostraram que, para o TSP, usar a solução inicial gerada por esta heurística gulosa refinada com a busca

local mencionada, leva a soluções finais piores que as obtidas usando soluções iniciais aleatórias refinadas pela mesma busca local. Além disso, a heurística *Clark-Wright* também pode consumir muito tempo na resolução de grandes instâncias do TSP. Como exemplo, destaca-se que, a estratégia aleatória domina a gulosa em qualidade de solução e tempo de busca.

A estratégia aleatória pode gerar um alto desvio em termos de solução obtida. Para melhorar a robustez, uma estratégia hibrida pode ser usada. Ela consiste em combinações de abordagens: aleatória e gulosa. Por exemplo, um acervo de soluções iniciais pode ser composta de soluções gulosas geradas e soluções aleatória geradas. Além disso, diferentes estratégias gulosas podem ser usadas no processo de inicialização. De fato, para alguns problemas de otimização existem vários algoritmos construtivos gulosos que são fáceis de elaborar e implementar.

Em alguns problemas de otimização restrita, pode ser difícil gerar soluções aleatórias que sejam viáveis. Neste caso, os algoritmos gulosos podem ser alternativas na geração de soluções iniciais viáveis. Para alguns problemas operacionais específico da vida real, a solução inicial pode ser inicializada (parcialmente ou completamente) de acordo com a expertise.

Como alternativa às modelagens puramente matemáticas, têm surgido, ao longo das últimas décadas, soluções através de procedimentos heurísticos e meta-heurísticos. Embora muito progresso tem sido alcançado na tarefa de encontrar tanto soluções ótimas, em problemas de otimização combinatória, quanto em soluções aproximadas via algoritmos aproximativos, muitos problemas de otimização combinatória na prática se beneficiam de métodos heurísticos que produzem soluções de boa qualidade sem necessariamente garantir a otimalidade. Nesse sentido, muitas heurísticas modernas para otimização combinatória são baseadas no *guidelines* de meta-heuristicas tais como: *Tabu Search*, *Simulated Annealing*, Algoritmos Genéticos, *Variable Neighborhood Search*, *Iterated Local Search*, *Ant Colony Optimization*, *Scatter Search*, Reconexão de Caminhos, *Swarm Optimization*, Procedimento de Busca Guloso e Aleatório e adaptado (GRASP) [23].

A meta-heurística SA mencionada acima foi utilizada neste trabalho na resolução do PAAS devido seu êxito na resolução de problemas do gênero em questão. Alfaro e Teran [7], Lopes, Rodrigues e Steiner [23] e Kripka e Kripka [23] obtiveram resultados significativos na resolução do PAAS por meio da meta-heurística *Simulated Annealing*.

2.2.2.5 *Simulated Annealing*

Simulated Annealing aplicado em problemas de otimização iniciou do trabalho de Kirkpatrick et al.[20]. Dentre esses trabalhos, alguns foram pioneiros como aplicação em grafos de particionamento e elaboração de VLSI.

De acordo com Talbi [35] o SA se baseia nos princípios da mecânica estática segundo o qual o processo de recozimento de um metal requer um aquecimento seguido de um lento resfriamento para se obter uma estrutura cristalina forte. A resistência de sua estrutura depende da taxa de resfriamento dos metais. Se a temperatura inicial não é suficientemente elevada ou se aplicar aos metais um arrefecimento brusco, a estrutura do metal irá adquirir imperfeições, fragilizando-o. Sendo assim, uma prática comum é, o cultivo de de cristais fortes a partir de um lento e cuidadoso processo de resfriamento.

É neste contexto que se fundamenta o método. O SA simula as mudanças de energia em um sistema sujeito ao processo de resfriamento até que este atinja seu estado de equilíbrio. Neste contexto, fazendo uma analogia entre o sistema físico e o problema de otimização, a função objetivo é análogo ao estado de energia do sistema. Uma solução do problema de otimização corresponde a um estado do sistema. A variável de decisão associada a uma solução do problema são análogas as posições moleculares. O ótimo global corresponde ao estado fundamental do sistema. Então, encontrar um mínimo local indica que um estado metaestável foi atingido. Como dito anteriormente, o SA é um algoritmo estocástico, que permite sob algumas condições, a degradação de uma solução. O objetivo é escapar de ótimos locais e, assim, evitar uma convergência prematura. Este método funciona

sem memória, ou seja, o algoritmo não utiliza toda informação recolhida durante a pesquisa para escolha do próximo vizinho candidato.

A partir de uma solução inicial, o SA prossegue em várias iterações. Em cada iteração, um vizinho aleatório s' é gerado a partir da solução corrente s. Movimentos que melhoram a função custo são sempre aceitos. Caso contrário, o vizinho é selecionado com uma certa probabilidade de aceitação que depende da temperatura atual e da variação na função objetivo. Essa variação representa a diferença no valor da função objetivo (energia) entre a solução corrente e a solução vizinha gerada. Ao longo do progresso do algoritmo, a probabilidade que tal movimento seja aceito diminui, como mostra a Figura 2.14. Esta probabilidade segue no geral, a distribuição de *Boltzmann*, conforme apresentado pela Equação 2.10:

$$P(\triangle, T) = e^{-\triangle/t} \tag{2.10}$$

onde $\triangle = f(s') - f(s)$.

Este usa como parâmetro de controle, a temperatura, para determinar a probabilidade de aceitação de soluções piores. Em particular, faz-se experimentos para determinar o nível de temperatura. Uma vez que o estado de equilíbrio é alcançado, a temperatura é reduzida gradualmente de acordo com um fator de resfriamento, de tal maneira que, no fim da busca, a chance de soluções piores serem aceitas reduz consideravelmente.

Logo abaixo cabe utilizar uma exemplificação para ilustrar o funcionamento do método meta-heurístico *SA*. Considere a maximização da função dada abaixo:

$$f(x) = x^3 - 60x^2 + 90x + 100 \tag{2.11}$$

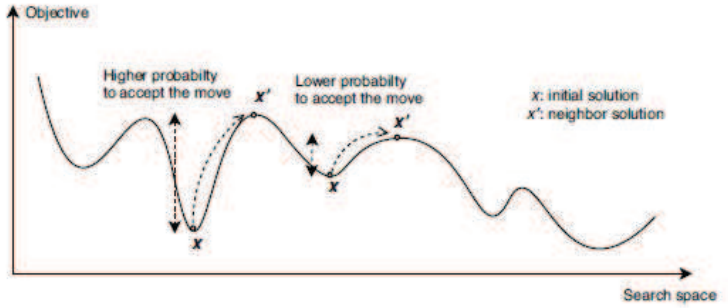


Figura 2.14: *Simulated Annealing* escapando de ótimo local. Quanto mais alta a temperatura, maior a probabilidade de aceitar movimentos de piora. [35]

Uma solução x é representada como um *string* de 5 *bits*. A vizinhança consiste em lançar aleatoriamente um bit. O máximo global desta função é 01010 (x=10, $f(x)$=4100).

O primeiro cenário começa da solução 10011 ($x = 19, f(x) = 2399$) com uma temperatura inicial T_0 igual a 500. (Tabela 2.1)

T	Movimento	Solução	f	Δf	Movimento?	Nova Solução Vizinha
500	1	00011	2287	112	Sim	00011
450	3	00111	3803	<0	Sim	00111
405	5	00110	3556	247	Sim	00110
364.5	2	01110	3684	<0	Sim	01110
328	4	01100	3998	<0	Sim	01100
295.2	3	01000	3972	16	Sim	01000
265.7	4	01010	**4100**	<0	Sim	01010
239.1	5	01011	4071	29	Sim	01011
215.2	1	11011	343	3728	Não	01011

Tabela 2.1: Cenário 01: T=500 e S_0 = 10011. [35]

O segundo cenário começa com a mesma solução 10011, com uma temperatura inicial T_0 igual a 100. (Tabela 2.2)

T	Movimento	Solução	f	Δf	Movimento?	Nova Solução Vizinha
100	1	00011	2287	112	Não	10011
90	3	10111	1227	1172	Não	10011
81	5	10010	3556	247	Sim	00110
72.9	2	11010	516	2176	Não	10010
65.6	4	10000	3236	<0	Sim	10000
59	3	101000	2100	1136	Sim	10000

Tabela 2.2: Cenário 02: T=100 e S_0 = 10011. [35]

Neste exemplo, a temperatura inicial não é alta o suficiente, e o algoritmo fica preso em um ótimo local. Além da solução atual, a melhor solução encontrada, desde o início da pesquisa, é armazenada. Alguns parâmetros controlam o progresso da busca pelo vizinho, os quais são a temperatura e o número de iterações realizadas a cada temperatura. Essa busca pode ser modelada pela Cadeia de *Markov*, onde o próximo estado depende somente do estado corrente. Neste algoritmo existe uma garantia de convergência em direção a solução ótima.

1) Movimentos Aceitos

O sistema pode escapar de um ótimo local devido a uma probabilidade de aceitação de um vizinho não melhorado. A probabilidade de aceitar um vizinho não melhorado é proporcional a temperatura T e inversamente proporcional a variação da função objetivo $\triangle$. A lei da termodinâmica diz que na temperatura T, a probabilidade de um aumento na energia de magnitude $\triangle$, é dado por:

$$P(\triangle, T) = e^{-\triangle/kt} \tag{2.12}$$

onde k= Constante de Boltzmann.

Então, a probabilidade do movimento não melhorado ser aceito é dado por:

$$P(\triangle, T) = e^{-\triangle/kT} > R \tag{2.13}$$

onde $\triangle$ é a mudança no valor da função, T é a temperatura corrente, e R é um número aleatório uniforme [0,1].

A altas temperaturas, a probabilidade de aceitação de movimentos piores é alta. Se T=∞, todos movimentos seriam aceitos, o que corresponde a percorrer uma trajetória aleatória no espaço de soluções.

Em baixas temperaturas, a probabilidade de aceitação de movimentos piores é baixa. Se t=0, nenhum pior movimento é aceito e a busca é equivalente a um algoritmo guloso, como o Hill Climbing [19]. Além do mais, a probabilidade de aceitar uma grande deterioração na qualidade de solução diminui exponencialmente em direção a zero de acordo com a distribuição de *Boltzmann*.

Para melhor entendimento do procedimento utilizado pela meta-heurística *Simulated Annealing* cabe ilustrar os conceitos básicos em que se fundamenta este método. Neste ponto é importante retomar o conceito de como encontrar o ótimo local e global, mencionado anteriormente. A abordagem utilizada por Hillier e Lieberman em [19] faz uma analogia para determinar qual de uma série de montanhas é a mais alta e depois subir ao topo dessa particular montanha. Infelizmente os processos matemáticos não possuem ferramentas suficientes para avistar uma "montanha" alta a distância. Sendo assim, alguns métodos meta-heurísticos, como o *Tabu Search*, utiliza a metodologia de subir a "montanha" atual na direção mais íngreme até atingir seu topo e depois começar a descer lentamente enquanto procura outra "montanha" para escalar. O único inconveniente é que se leva muito tempo (número de iterações) subindo cada uma, ao invés de, procurar por aquela que seja mais alta.

A metodologia utilizada pela meta-heurística *Simulated Annealing* se concentra na procura da particular "montanha" mais alta. Já que esta pode se localizar em qualquer ponto da região de soluções viáveis, a primeira busca é em caminhar em direções aleatórias (exceto por rejeitar alguns, porém nem todos os passos que levariam para baixo em vez de para cima), visto que, o objetivo principal é de explorar ao máximo essa região de soluções viáveis. Como o objetivo é buscar a montanha mais alta, a grande maioria dos passos aceitos é em direção acima. Isto por sua vez implica que a busca tenderá a se concentrar em regiões do espaço de busca que contenham soluções viáveis onde as montanhas são altas, embora encontrar a montanha mais alta não seja garantido. Dando-se tempo e reduzindo a temperatura de forma adequada, o processo provavelmente atingirá o topo de uma

das "montanhas" mais altas, podendo eventualmente, atingir a mais alta (ótimo global).

Sendo mais específico, cada iteração do SA busca deslocamentos da solução experimental corrente para um vizinho imediato na vizinhança local dessa solução. Considere que:

Z_c = valor da função objetivo para solução experimental atual,

Z_n = valor da função objetivo para o atual candidato a ser próxima solução experimental,

T = Um parâmetro que mede a tendência a aceitar o atual candidato a ser a próxima solução experimental se esse candidato não for uma melhoria em relação à solução experimental atual.

A regra para selecionar qual vizinho imediato será a próxima solução experimental é a seguinte:

Regra de Seleção da Movimentação: Entre todos os vizinhos imediatos à solução experimental atual, selecione um aleatoriamente para se tornar o candidato atual para ser a solução experimental seguinte. Partindo-se do pressuposto de que o objetivo é a maximização de uma função objetivo, aceite ou rejeite esse candidato como próxima solução experimental da seguinte forma:

Se $Z_n \geq Z_c$, sempre aceite esse candidato.

Se $Z_n < Z_c$, aceite o candidato com a probabilidade P, dada por:

$$P = e^x, ondex = \frac{Z_n - Z_c}{T} \tag{2.14}$$

Se, em vez disso, o objetivo for a minimização, inverta os sinais de Z_n e Z_c em (2.14). Se esse candidato for rejeitado, repita esse processo com um novo vizinho imediato à solução experimental atual, selecionado aleatoriamente. Se não restar

mais nenhum vizinho imediato, encerre o algoritmo.

Na análise, se o atual candidato considerado for melhor que a solução experimental atual, ele sempre será aceito como próxima solução experimental. Se for pior, a probabilidade de aceitação depende de quão pior ela é e do valor de T. A regra de seleção da movimentação geralmente vai aceitar um passo que é apenas ligeiramente para baixo, mas raramente aceitará um passo brusco indo para baixo. Partindo de um valor de T, relativamente grande, a probabilidade de aceitação é maior, o que permite que a busca prossiga em quase todas direções aleatórias. A partir disso, diminui-se gradualmente o valor de T à medida que a busca prossegue, reduzindo a probabilidade de aceitação, e aumentando a ênfase na subida.

A experiência com a implementação da meta-heurística SA mostra que a escolha dos valores de T, ao longo do tempo, controla o grau de aleatoriedade no processo permitindo caminhar para baixo. Esse componente aleatório, fornece maior flexibilidade para se movimentar em direção a outra parte da região de soluções viáveis na esperança de encontrar uma montanha mais alta. O método de implementação da regra de seleção da movimentação para estipular se um determinado passo para baixo será aceito é comparar um número aleatório entre 0 e 1 com a probabilidade de aceitação. Um número aleatório deste pode ser imaginado como uma observação aleatória de uma distribuição uniforme entre 0 e 1.

Se o *número aleatório* $<$ Prob{aceitação}, aceitar um passo para baixo.

Caso contrário, rejeite essa movimentação.

O *Simulated Annealing* utiliza uma equação de Prob{aceitação} especificada pela regra de seleção da movimentação, visto que, este método se baseia na analogia com o processo de recozimento físico de metais. Em particular, a probabilidade de aceitação de um aumento quando a temperatura for T tem o mesmo formato de Prob{aceitação} na regra de seleção da movimentação para SA.

A analogia para um problema de otimização na forma de minimização é que o nível de energia da substância no estado atual do sistema corresponde a um valor

da função objetivo na solução viável atual do problema. O objetivo de fazer que a substância atinja um estado estável com um nível de energia que seja o menor possível corresponde a fazer que o problema atinja uma solução viável com um valor da função objetivo que seja o menor possível. Da mesma forma que ocorre com o processo de recozimento físico, a temperatura deve ser programada de forma a ser apropriada para o processo. Em razão da analogia com o processo de recozimento que ocorre no mundo físico, T passará a representar uma temperatura no algoritmo SA. Essa programação precisa especificar quantas movimentações (iterações) devem ser feitas a cada valor de T. Cabe ressaltar que a seleção desses parâmetros para atender ao problema em questão é um fator de extrema importância na eficiência do algoritmo. Normalmente, utiliza-se experimentação preliminar para se orientar na seleção desses parâmetros.

2) Descrição de um Algoritmo *Simulated Annealing*

Inicialização: Comece com uma solução inicial.

Iteração: Utilize a *regra de seleção de movimentação* para selecionar a próxima solução experimental.

Verificar a programação de temperaturas: Quando o número desejado de iterações tiver sido executado no valor atual de T, diminua T para o próximo valor indicado na programação de temperatura e retome as iterações com esse valor seguinte.

Regra de parada: Quando o número de iterações desejado tiver sido executado com o menor valor de T em uma programação de temperaturas, pare. Aceite a melhor solução experimental encontrada em qualquer iteração (inclusive para valores de T maiores) como solução final.

Logo abaixo pode-se verificar o pseudocódigo, denominado Algoritmo 2, que refere-se à descrição apresentada a cerca do funcionamento do método de resolução de problemas de otimização, *Simulated Annealing*.

Algoritmo 2 - *Simulated Annealing*

```
1: Procedimento SA(f(.), N(.), α, SAmax, T0, s)
2: s* ⟵ s;            {Melhor solução obtida até então}
3: IterT ⟵ 0;         {Número de iterações na temperatura T}
4: T ⟵ T0;            {Temperatura corrente}
5: enquanto (T > 0) faça
6:     enquanto (IterT < SAmax) faça
7:         IterT ⟵ IterT+1;
8:         Gere um vizinho qualquer s' ∈ N(s);
9:         △ = f(s') - f(s);
10:        se (△< 0)
11:            então
12:                s ⟵ s';
13:                se (f(s') < f(s*)) então s* ⟵ s';
14:        senão
15:                Tome x ∈ [0,1];
16:                se (x < e^(-△/T)) então s ⟵ s';
17:        fim-se;
18:    fim-enquanto;
19:    T ⟵ α x T;
20:    IterT ⟵ 0;
21: fim-enquanto;
22: s ⟵ s*;
23: Retorne s;
24: fim SA;
```

Capítulo 3

Trabalhos Relacionados

Para que fosse possível a elaboração desta dissertação foi necessário o estudo cuidadoso de diversos materiais com assuntos semelhantes aos abordados neste trabalho, além do acompanhamento da literatura de referência *Tabu Search*, desenvolvida e elaborada por Glover e Laguna [18], pioneiros nesta técnica meta-heurística para resolução de problemas de otimização combinatória. Como mencionado anteriormente, este trabalho tem como objetivo a elaboração de um modelo matemático e aplicação da técnica de otimização SA na resolução do PAAS, em uma unidade da Universidade Federal Fluminense. Não obstante, devido a natureza combinatória do problema de agendamento de recursos, existem hoje várias técnicas de otimização capazes de fornecer soluções viáveis para o problema de horários.

O artigo proposto por [7] trata o PAAS no ITESM, uma universidade mexicana, usando exatamente a mesma técnica heurística SA. Todavia, diferentemente do contexto utilizado no nosso trabalho, e normalmente em outros artigos, os autores utilizam esta técnica fazendo uma alusão à *Cadeia de Markov*, um caso particular de processo estocástico com estados discretos, que utiliza geralmente o tempo como parâmetro, podendo ser este discreto ou contínuo. Um leitor mais atento é capaz de identificar a relação direta entre a aplicação tradicional do SA, que faz analogia ao processo termodinâmico de resfriamento de um conjunto de átomos, e o processo da *Cadeia de Markov*, sequência de variáveis aleatórias onde

estados anteriores são irrelevantes para a predição dos estados seguintes, desde que o estado corrente seja conhecido.

O algoritmo é constituído basicamente por duas etapas:

1) a geração da solução inicial, neste denominado **InitialST**,

2) o refinamento das soluções, geradas pela função objetivo aqui denominada **GenerateST**, por meio do SA.

A função objetivo, neste caso, representa o somatório de cada custo individual de alocar uma única aula à uma sala respeitando alguns critérios de avaliações, obtendo por fim, o custo total de uma sequencia de aulas aceitas em determinadas salas - estados aceitos na *Cadeia de Markov* (atribuição para toda seção de classe respeitando seus critérios de avaliação). Dada a função objetivo composta pelo somatório dos critérios de avaliação multiplicados por seus respectivos pesos, que indicam a importância de cada componente desta função, os autores definiram a estrutura de vizinhança utilizando o método de busca local da seguinte forma:

- **Cenário 01: Movimento de Realocação (*Move*)** - Número de salas disponíveis é superior à quantidade necessária. Diante deste cenário, uma seção de classe já atribuída é selecionada aleatoriamente para ser realocada à uma nova sala, também escolhida aleatoriamente, entre as salas disponíveis.

- **Cenário 02: Movimento de Troca (*Swap*)** - Outra maneira de gerar um novo estado ocorre quando o número de salas disponíveis é exatamente ou pouco maior que a quantidade necessária. Defronte a esta situação, duas seções de classes já atribuídas às salas são selecionadas aleatoriamente para serem trocadas entre si.

A solução proposta para o problema por meio do algoritmo SA forneceu uma tabela com os dez horários mais “disputados” para atribuição dos maiores números

de seção de classe, indicando o horário de início de cada seção de classe, seu tempo de duração e os respectivos dias da semana que serão ministradas, vide Figura 3.1 . Em todo teste, dados os parâmetros de valor de melhor performance para o algoritmo, o custo resultante da atribuição das seções de classes apresentou uma redução de 32,41% no seu tempo de processamento quando comparado a forma manual de fazê-lo.

Meeting Time	*Cost*			
	Initial	*Average*	*Max*	*Min*
10/3 T	44852	11644.8	12001	11360
10/2 M	40286	9412.2	9699	9256
08+/3 T	36359	9545.8	9715	9436
11+/3 T	37849	9365.4	9406	9300
09/2 M	40249	8961.4	9275	8853
11/2 M	41219	7622.2	7814	7353
16/3 T	33288	7106	7253	6926
08/2 M	30784	5338.2	5377	5284
12/2 M	27747	4651.2	4682	4589
18/6 MO	21877	5559.6	6917	5133

Figura 3.1: Resultados: Horários Mais Requisitados [7]

Conforme o material mencionado acima, existem ainda outros métodos meta-heurísticos para resolução deste tipo de problema, e entre esses métodos destaca-se o *Tabu Search* devido sua particularidade de permitir ao processo de busca escapar de um ótimo local através do uso sistemático da memória para restringir alguns movimentos ditos proibido.

E será neste contexto que aborda-se o próximo artigo relacionado ao problema estudado neste trabalho. O seguinte assunto mencionado aborda a aplicação da meta-heurística *TS* também na resolução do PAAS no Centro de Tecnologia da UFPB com o objetivo de obter soluções de alto grau de satisfação e baixo custo

computacional. A escolha da aplicação desta heurística se deve a essa se mostrar bastante adequada e eficiente na resolução de problemas de natureza combinatória. De forma sucinta neste artigo os autores [34] propõem um procedimento construtivo que fornece uma solução inicial. A seguir, esta solução é refinada pela técnica meta-heurística *TS* usando movimentos de realocação e troca de aulas entre salas para explorar o espaço de busca. De acordo com [19] o artifício de proibição de determinados movimentos, característica principal da meta-heurística *TS*, tem a intenção de impedir que a solução retorne ao ponto de ótimo local nas iterações seguintes. Caso não haja essa proibição de determinados movimentos, esse procedimento poderá fazer com que o algoritmo entre em *loop*, inviabilizando o mesmo.

O risco com esse tipo de abordagem é que, após se afastar de um ótimo local, o processo retornará imediatamente para o mesmo ótimo local. Para evitar isso, o *TS* impede temporariamente movimentações que pudessem retornar para (ou quem sabe ir em direção a) uma solução visitada recentemente. Uma lista, chamada lista de tabus, registra essas movimentações proibidas, que são chamadas de movimentos de tabu. A única exceção para proibir uma movimentação dessas é caso se descubra que uma movimentação de tabu, na verdade, é melhor que a melhor das soluções viáveis até agora encontradas.

Segundo [19], além do artifício do uso de memória para orientar a busca, este método também pode incorporar alguns conceitos mais avançados como a **intensificação**, que envolve a exploração de uma parte da região de soluções viáveis de forma mais completa do que a usual, após ter sido identificada como particularmente promissora, por conter soluções excelentes, e a **diversificação**, que envolve forçar a busca em áreas pouco exploradas do espaço de soluções viáveis. Em um artigo complementar, os mesmos autores dão continuidade ao estudo deste problema através de outro método de resolução, o Método Exato [26].

De acordo com as características do problema estudado da UFPB, definiu-se suas restrições e requisitos de qualidade, e atribuiu a estes os pesos devidos

para a análise. Estes requisitos devem ser atendidos sempre que possível, mas, o seu não atendimento não implica em inviabilidade do modelo. O procedimento para gerar a solução inicial forneceu solução satisfatória, visto que, gerou soluções que contemplaram todas as restrições de viabilidade, diferentemente da solução manual que não atendia plenamente aos critérios de viabilidade. Além disso, a solução manual não foi capaz de alocar todas as aulas, deixando aproximadamente 48 horas-aulas sem alocação, o que corresponde a 4,7% do total das aulas.

A fase de refinamento do algoritmo *TS* apresentou melhorias significativas, identificada pela redução de 79% no valor da função objetivo da solução inicial. E finalmente, a solução inicial não apresentou grandes variabilidades gerando soluções satisfatórias com baixo custo computacional. O quadro abaixo, representado pela Figura 3.2, fornece um relatório da execução dos procedimentos computacionais das soluções inicial e final, bem como, da solução manual para os requisitos de qualidade definidos para o problema:

Resultados			
Requisito de Qualidade	**Quantidade de horas-aula em que o requisito não foi atendido**		
	Solução Manual	**Solução Inicial**	**Solução Final**
(a) As aulas devem ser alocadas em blocos pré-determinados de acordo com a turma	340	258	**216**
(b) As aulas das disciplinas, cuja freqüência corresponde a 2 dias semanais, não devem ser alocadas em blocos distintos	**23**	97	120
(c) As aulas das disciplinas, cuja freqüência corresponde a 3 dias semanais, não devem ser alocadas em blocos distintos	**0**	18	24
(d) Aulas a serem ministradas em salas do tipo carteiras não devem ser alocadas nos ateliês.	35	23	**2**
(e) Aulas a serem ministradas em salas do tipo carteiras não devem ser alocadas em mesas	64	**31**	40
(f) Aulas a serem ministradas em salas do tipo carteiras não devem ser alocadas em pranchetas	42	50	**27**
(g) Aulas a serem ministradas em salas do tipo prancheta não devem ser alocadas em mesas	22	**0**	**0**
(h) Todas as aulas devem ser alocadas	48	**0**	**0**

Figura 3.2: Comparação entre as Soluções [34]

Esta baixa dispersão identificada pelo valor do desvio padrão, 4,01 na Figura

3.3, indica que o procedimento *TS* se mostrou consistente. Ainda, esta figura traz os valores da melhora de maior significância e a de menor significância da função objetivo, além do tempo de processamento do algoritmo.

Percentual Médio de Melhora	Desvio Padrão	Melhora de Maior Significância	Melhora de Menor Significância	Tempo Médio de Execução
79%	4,01	82%	69%	30min55s

Figura 3.3: Robustez do Procedimento *Tabu Search* [34]

Além desses dois exemplos de artigos referidos acima, serão apresentados abaixo ainda alguns outros, que servem como materiais de apoio relacionados à este tema, que mencionam outras técnicas heurísticas de resolução de problemas desta natureza.

O próximo trabalho propõe-se resolver o PAAS por meio da meta-heurística Algoritmo Genético Compacto [13], um algoritmo de estimação de distribuição, e por meio de programação inteira. Neste trabalho foi abordado o PAAS modelado segundo os critérios do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). Além do modelo matemático, foi proposto uma abordagem utilizando-se de uma meta-heurística baseada em AGc. A baixa demanda de memória deste algoritmo, por substituir populações por representações estatísticas, é um diferencial em sua utilização em comparação ao genético convencional. Considere ainda que, diferentemente de outras abordagens evolutivas, em que a população inicial determina um viés no espaço de soluções por onde o algoritmo irá trafegar, pois as soluções herdarão as características de seus progenitores, no AGc, no entanto, o espaço inteiro é oferecido para a amostragem no início e o viés estatístico é adicionado, restringindo o espaço de busca.

Algoritmos que utilizam-se de técnicas evolutivas geralmente não são utilizados para este tipo de problema, pois os operadores genéticos tradicionais de *crossover* e mutação com frequência geram soluções infactíveis. Sendo assim, pode ser necessário que se elabore alguma função de reparação para viabilizar, ou seja, factibilizar a solução obtida.

O estudo de caso adotado para o problema foi ICMC-USP, em que somente as disciplinas da graduação são consideradas, conta com 23 salas de aulas distribuídas em 4 blocos didáticos e ministra cerca de 260 aulas para, em média, 140 turmas. Neste estudo cada turma está associada a um ou mais perfis de alunos, que são conjuntos de discentes que dividem o mesmo curso e ano de ingresso, e portanto, estão cursando um conjunto pré estabelecido de disciplinas. Estas turmas podem ter um ou mais encontros semanais, possuem associadas a elas um números de inscritos e necessidades por recursos. As salas possuem recursos disponíveis e capacidades conhecidas. Assim, para que um encontro possa ocorrer em uma sala, ela deve disponibilizar todos os recursos necessários, e ainda, ter capacidade igual ou superior ao número de inscritos. Os encontros estão associados a uma ou mais horas-aula, ou *slots* de horários, que são divisões de tempo dentro da semana.

Devido ao banco de dados específico da instituição estar em fase final de implementação, a obtenção de instâncias reais de dados tomaria muito tempo. Dessa maneira, foi implementado um gerador de instâncias Fictícias, em que o número de *slot* de horários (H), turmas(T), salas(S), recursos(R) e perfis(P) são fornecidos. Os dados de entrada do modelo matemático são amostrados aleatoriamente tomando como base as instâncias reais. Foram geradas por este método 3 instâncias de testes com tamanhos variados. Uma instância real também foi obtida, referente ao primeiro semestre de 2015 do ICMC.

Além disso, foram definidos 3 conjuntos de parâmetros para o modelo, de forma a atender as prioridades da instituição. Para cada instância e cada parametrização o modelo matemático foi executado por 10 minutos, com ênfase na obtenção de soluções factíveis, e a melhor solução factível obtida foi considerada. Vale ressaltar que para a instância Fictícia 3 não foi encontrada nenhuma solução factível durante o tempo de execução para duas parametrizações.

Para a resolução por meio da meta-heurística AGc proposta foram executados 5 testes para cada instância e parametrização, extraindo destes testes a melhor solução, a pior solução e solução mediana. Estes resultados foram comparados com

os fornecidos pelo método exato mostrando que: para instância Fictícia 1 e Real a resolução exata mostrou-se com bons resultados, considerando que para Fictícia 1 conseguiu provar otimalidade em pouco tempo. No entanto, para instâncias maiores a própria construção do modelo levaria muito tempo para ser elaborada e o AGc mostrou-se com melhores resultados, incluindo duas parametrizações da instância Fictícia 3 em que obteve resultados factíveis em todas as execuções contra nenhum resultado encontrado pela resolução do modelo matemático. A principal diferença de performance entre AGc e a resolução do modelo encontra-se pelo tamanho das instâncias.

Por último, o artigo [22] se propõe a resolver o problema operacional de agendamento de horários com intervalos de tempos fixos (OFISP), durante o desenvolvimento de um sistema de suporte à decisão, para o departamento de manutenção da maior companhia aérea holandesa KLM, no aeroporto de *Schiphol*. Para isso propôs-se uma abordagem exata e aproximada na sua resolução do OFISP. O OFISP é caracterizado como o problema de agendamento de horários de um número fixo de tarefas, cada uma com um tempo de início e término fixo, um índice de prioridade e uma classe de tarefas. O objetivo é encontrar uma atribuição de tarefa à máquina com prioridade máxima. A complexidade deste problema se dá devido às seguintes restrições:

1. Cada máquina pode manusear somente uma tarefa de cada vez;
2. Cada máquina pode manusear apenas as tarefas a partir de um subconjunto pré especificado de todas as classes de tarefas;
3. Preempção não é permitido.

Algumas aeronaves que chegam ao aeroporto necessitam de uma quantidade de tarefas de manutenção. O tempo de processamento, tanto quanto, a ordem que cada uma destas tarefas são realizadas, são orientadas segundo normas de manutenção, e como consequência, essas normas e o tempo de tabela determinam os

intervalos fixos no qual essas tarefas devem ser realizadas, afim de, não atrasarem a partida dos aviões em seus próximos voos. O problema a ser resolvido pelo sistema de suporte de decisão está relacionado a desenvolver um agendamento de manutenção, tal que em princípio, todas as tarefas sejam realizadas. De qualquer maneira, aquelas tarefas com baixas prioridades, as quais não podem ser realizadas dentro do intervalo requerido, podem ser adiadas até a próxima parada das aeronaves no aeroporto.

O problema ocorre no aeroporto de *Schiphol*, onde aviões de diferentes tipos necessitam estacionar em portões durante intervalos de tempos fixos. No entanto, cada portão só pode manusear um conjunto limitado de tipos de aeronaves, devido suas restrições técnicas. Sendo assim, o problema se resume em encontrar uma atribuição de aeronave à portões, onde o número de aviões não atribuídos neste portão seja minimizado. Este trabalho aborda a engenharia de manutenção (portões, salas de aula entre outras) como máquinas, e inspeções (aeronaves, classes entre outras) como tarefas.

Resolução / Artigos	EEIMVR	ITESM	UFBP	ICMC - USP	KLM
Simulated Annealing	X	X			
Tabu Search			X		
Compact Genetic				X	
Dual-Cost Heuristics					X

Tabela 3.1: Meta-Heurística Utilizada em cada Artigo. (Fonte Própria, 2017)

Resolução / Artigos	EEIMVR	ITESM	UFBP	ICMC - USP	KLM
Modelo Matemático	X	-	X	X	X

Tabela 3.2: Modelo Exato Utilizado em Cada Artigo. (Fonte Própria, 2017)

As tabelas 3.1 e 3.2 exibem os métodos de resolução utilizados em cada artigo apresentado nesta seção. As diferenças principais com relação aos métodos se dão segundo a utilização de abordagens exatas, e na heurística escolhida em cada caso mencionado.

Capítulo 4

O Problema de Alocação de Aula em Salas: Um Estudo de Caso

Este capítulo foi desenvolvido com o objetivo de apresentar e resolver o estudo de caso que se refere ao problema de alocação de aulas em salas, de uma instituição de ensino superior, através do método exato, e ainda, por meio da meta-heurística *Simulated Annealing*. Após a obtenção das soluções, realizou-se uma comparação entre aquelas obtidas (exatos e aproximados) e a utilizada atualmente pela instituição (forma manual), e posteriormente, feito as devidas considerações. A Seção 4.1 exibe as características físicas da universidade, além de expor o problema típico enfrentado pela universidade antes do início de todos os semestres letivos, assunto principal deste trabalho. Da Seção 4.2 à Seção 4.4 exibe-se todas as características relevantes para o desenvolvimento dos métodos de resolução escolhidos do problema abordado, com a Seção 4.4.2 apresentando o funcionamento da estrutura de vizinhança que melhor se ajustou ao modelo. E finalmente, na seção 4.5 apresenta-se uma formulação matemática para o problema abordado.

4.1 Caracterização do Problema

A Escola de Engenharia Industrial Metalúrgica de Volta Redonda (EEIMVR/UFF), escolhida para a realização deste estudo, se localiza na região Sul Fluminense [3]. Todas as aula ministradas na EEIMVR são destinadas às turmas dos seguintes cursos: Engenharia Metalúrgica (Eng. Metal), Engenharia Mecânica (Eng. Mec), Engenharia de Produção (Eng. Prod) e Engenharia de Agronegócios (Eng. Agro). No período considerado para este estudo, a instituição contava com uma infraestrutura composta por 20 salas de aula, 07 laboratórios distribuídos ao longo de 03 prédios, 01 auditório e 01 biblioteca, denominados Prédio "Antigo", Prédio "Edil Patury Monteiro" e Prédio "Anexo", onde são ministradas cerca de 790 aulas para uma média de 224 turmas, com um total de aproximadamente 1.580 alunos. Além destes três prédios, existe ainda, um auditório, localizado separadamente dos outros, onde são realizadas formaturas, palestras e outros eventos. Todos os dados aqui apresentados se referem ao ano letivo de 2014.

No prédio "Anexo" as salas encontram-se distribuídas em 04 (quatro) diferentes andares, sendo o 1º e 2º destinados à biblioteca e sala de estudos respectivamente. Além do prédio "Anexo", existem as salas do prédio intitulado como "Edil Patury Monteiro". Neste, há salas de aula, laboratórios e salas de professores. Cabe ressaltar que as salas deste prédio são utilizadas não apenas pelos estudantes de graduação, mais também, pelos discentes de mestrado e doutorado, e ainda, por pesquisadores de pós-doutorado e visitantes. A demanda por aulas de laboratório na unidade é considerada pequena se comparada à demanda por salas de aula. Muitas vezes, os laboratórios são dedicados a um conjunto muito pequeno de disciplinas, o que torna o planejamento de horários destes relativamente simples. Neste sentido, este trabalho não trata a alocação dos laboratórios da unidade, focando exclusivamente na alocação de salas de aula.

Como este estudo se destina em alocar os cursos de todos os períodos de 4 currículos de engenharia, devido os cursos comuns ao ciclo básico da engenharia,

característica mencionada na definição do CTP, no Capítulo 2, algumas das turmas avaliadas são compostas por alunos de diferentes currículos, dado a isto, numa mesma classe pode haver discentes de até 4 currículos de engenharia diferentes, mas, que frequentam o mesmo curso obrigatório. Cada curso possuí um número específico de horas-aula semanal, podendo conter um ou mais encontros semanais com os respectivos docentes dos cursos. Deve-se ressaltar que, em outros trabalhos da literatura, cada encontro solicita um conjunto de requerimentos necessários para sua realização, tais como: quadro negro, projetores, computadores, entre outros. Na EEIMVR, todas as salas de aula estão equipadas com quadro, projetor, *data-show* e computadores, sendo portanto, desnecessário analisar este tipo de requisito neste estudo de caso. Além disso, as salas possuem capacidades conhecidas, então, para que uma aula possa ocorrer a sala deve disponibilizar também capacidade igual ou superior ao número de alunos inscritos.

A Tabela 4.1 mostra como foi feito o mapeamento de horários das aulas por cada dia da semana em *slots* temporais. O número associado ao par ordenado (horário, dia da semana) representa os horários em que as aulas são lecionadas para cada dia da semana. Sendo assim, por exemplo, para os horários 40 e 41 existem aulas que serão lecionadas das 14:00 às 15:00 e das 15:00 às 16:00 do período vespertino de uma quarta-feira, e assim por diante, para todos os outros horários. Com o objetivo de simplificar esse *slot* de horários foi considerado que cada aula possui duração de exatamente 60 minutos (1 hora), os minutos restantes do *slot* são usados para a transição de professores e alunos para outras atividades.

Horários / Dias	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
7:00	1	17	33	49	65	81
8:00	2	18	34	50	66	82
9:00	3	19	35	51	67	83
10:00	4	20	36	52	68	84
11:00	5	21	37	53	69	85
12:00	6	22	38	54	70	86
13:00	7	23	39	55	71	87
14:00	8	24	40	56	72	88
15:00	9	25	41	57	73	89
16:00	10	26	42	58	74	90
17:00	11	27	43	59	75	91
18:00	12	28	44	60	76	92
19:00	13	29	45	61	77	93
20:00	14	30	46	62	78	94
21:00	15	31	47	63	79	95
22:00	16	32	48	64	80	96

Tabela 4.1: Representação dos Horários. (Fonte Própria, 2017)

As aulas são ministradas nos períodos da manhã, tarde e noite, de segunda à sábado, sendo a maior demanda localizada no período vespertino, em que o taxa de ocupação chega a 70%. A Tabela 4.2 ilustra as características de cada sala da EEIMVR, identificadas por suas localidades, denominações e suas respectivas capacidades.

Prédio	Sala	Capacidade
Antigo	B3	57
Antigo	B5	53
Antigo	B6	55
Antigo	B9	60
Antigo	B11	50
Edil Patury Monteiro	D30-A	30
Edil Patury Monteiro	D30-B	30
Edil Patury Monteiro	D42	50
Anexo	N3A	80
Anexo	N3B	55
Anexo	N3C	60
Anexo	N4A	80
Anexo	N4B	65
Anexo	N4C	60
Anexo	N5A	80
Anexo	N5B	60
Anexo	N5C	55
Anexo	N6A	80
Anexo	N6B	55
Anexo	N6C	60

Tabela 4.2: Características das salas da EEIMVR. (Fonte Própria, 2017)

De acordo com os dados fornecidos pela instituição [3], nesta unidade de ensino,

a montagem do quadro de horários, e consequentemente a resolução do problema de alocação das aulas às salas, é desenvolvida por uma comissão com membros de todos os departamentos. Esta comissão é normalmente criada no meio do período letivo para possibilitar a montagem do quadro de horários do semestre subsequente. Para que isso seja possível, a comissão estabelece prioridades para os professores. Normalmente, estas prioridades se dão em função de compromissos particulares e administrativos, visto que, alguns professores também acumulam funções de coordenação e chefia de departamento, ou ainda, compromissos acadêmicos, já que, alguns deles também possuem atividades em pós-graduações. Neste sentido, este comitê parte de uma espécie de escala horária pronta, normalmente referente ao período atual, e faz alterações neste quadro de acordo com as prioridades estabelecidas.

Assim, devido a essas prioridades, os horários são feitos por professores de departamentos diferentes, e suas agendas dificultam as reuniões para esta definição. Então, dadas essas restrições, a comissão tem que tomar decisões rápidas de mudanças de horários, por isso, não pode esperar um longo tempo de *software* para definir as distribuições de salas/encontros (resolução do problema), e ainda assim, este processo fica sujeito a vários erros devido o desgaste interpessoal, fazendo com que os membros mais experientes dessa unidade, não tenham mais interesse em participar dessas atribuições em semestres posteriores.

Segundo Neves [24], este desinteresse em dar continuidade aos trabalhos faz com que a experiência adquirida pelos membros da comissão não seja devidamente compartilhada, o que muitas vezes faz com que as novas comissões realizem os mesmos erros de comissões anteriores, gerando um ciclo vicioso. Neste sentido, a automação de parte das tarefas da comissão pode ajudar a resolver vários destes problemas, reduzindo as chances de erros, o desgaste interpessoal e aproveitando melhor a experiência dos membros, que pode ser incorporada nas ferramentas de automação utilizadas.

4.2 Características do Modelo Computacional

Para elaboração desta seção foram tomadas como referência as restrições utilizadas em [34]. As restrições do problema são listadas abaixo, e devem ser atendidos, sendo que, o seu não atendimento não implica em inviabilidade do modelo:

a) Dois ou mais encontros não podem ocorrer simultaneamente na mesma sala;

b) Encontros de uma mesma turma não podem ser alocados em mais de uma sala num mesmo horário;

c) Encontros de uma determinada turma só devem ser alocados em salas de capacidade maior ou igual à demanda de estudantes desta.

d) Todas as aulas devem ser alocadas.

4.2.1 Função de Avaliação

A análise dos requisitos listados acima, associado à função objetivo, corrobora a citação mencionada, no início desta seção, sobre a inviabilidade do modelo. No entanto, para que fosse possível obter uma solução que retratasse algo mais próximo à realidade do problema, a função objetivo foi calculada visando minimizar o custo de alocar um encontro à um horário e sala disponíveis, visto que, este é obtido a partir do somatório do produto entre as alocações e os seus respectivos pesos, associados a ociosidade e excesso de demanda. Dado isso, os valores atribuídos às penalizações garantem que o número de assentos vazios nas salas será o mínimo possível, e que a função tentará ao máximo alocar os encontros dentro de uma sala compatível, dada a alta penalização atribuída a esse fator. Sendo assim, dada uma solução s avaliada por esta função objetivo, esta é calculada por duas parcelas, uma referente às penalizações relativas à ociosidade da demanda, e a segunda, correspondente às penalizações pelo seu excesso. Para garantir que a

solução do problema fosse viável, as penalizações assumiram os seguintes valores no problema abordado:

- **Peso Ociosidade: 1**
- **Peso Excesso: 1000**

Esses valores estão relacionados a relevância de cada fator no cálculo da função objetivo, e foram parametrizados de acordo com que foi definido experimentalmente.

4.2.2 O Algoritmo *Simulated Annealing* para o Caso EEIMVR

Com base na teoria apresentada até este ponto, e devido a complexidade do problema abordado neste trabalho (NP-Difícil), o uso exclusivamente de métodos de resolução exatos pode não ser uma boa alternativa. Diante disso, adicionalmente, optou-se pelo uso da meta-heurística, pois esta, se modelada corretamente, e com os devidos parâmetros ajustados, é capaz de fornecer uma "boa" solução para problemas de OC.

Entre as diversas meta-heurísticas encontradas na literatura, uma das mais adequada na resolução de problemas de *Scheduling*, como a deste trabalho, é a *Simulated Annealing*, devido seu desempenho satisfatório ao se concentrar na busca da melhor solução, a partir da máxima exploração possível da região de soluções viáveis.

Para um melhor entendimento da meta-heurística desenvolvida para resolução do problema real da EEIMVR, verifica-se abaixo, um diagrama de blocos que descreve o funcionamento do algoritmo SA, apresentado no Algoritmo 2, e adaptado para o problema real. Este traz de forma simplificada os principais passos no funcionamento deste método de resolução. O fluxograma da Figura 4.1 associado ao Algoritmo 2 permite seguir o raciocínio descrito abaixo.

O passo 01 e 02 se referem ao carregamento dos dados necessários no processamento do algoritmo construtivo, responsável pela geração da Solução Inicial que será utilizado como *imput* do algoritmo *SA*, observados nos paralelogramos e retângulo da Figura 4.1.

Em seguida, dos passos 05 a 22, o Algoritmo *SA* é "alimentado" com a Solução Inicial gerada, e inicia-se o processo de refinamento dessa solução, em busca da melhor solução viável, representada pelo algoritmo *Simulated Annealing*, contido no interior do retângulo menor.

Ao término das iterações programadas, e em posse do conjunto de soluções geradas, escolhe-se a melhor solução dentre todas essas do conjunto de soluções, e a retorna como solução da meta-heurística, representada pelo passo 23 e pela figura geométrica da Solução Final.

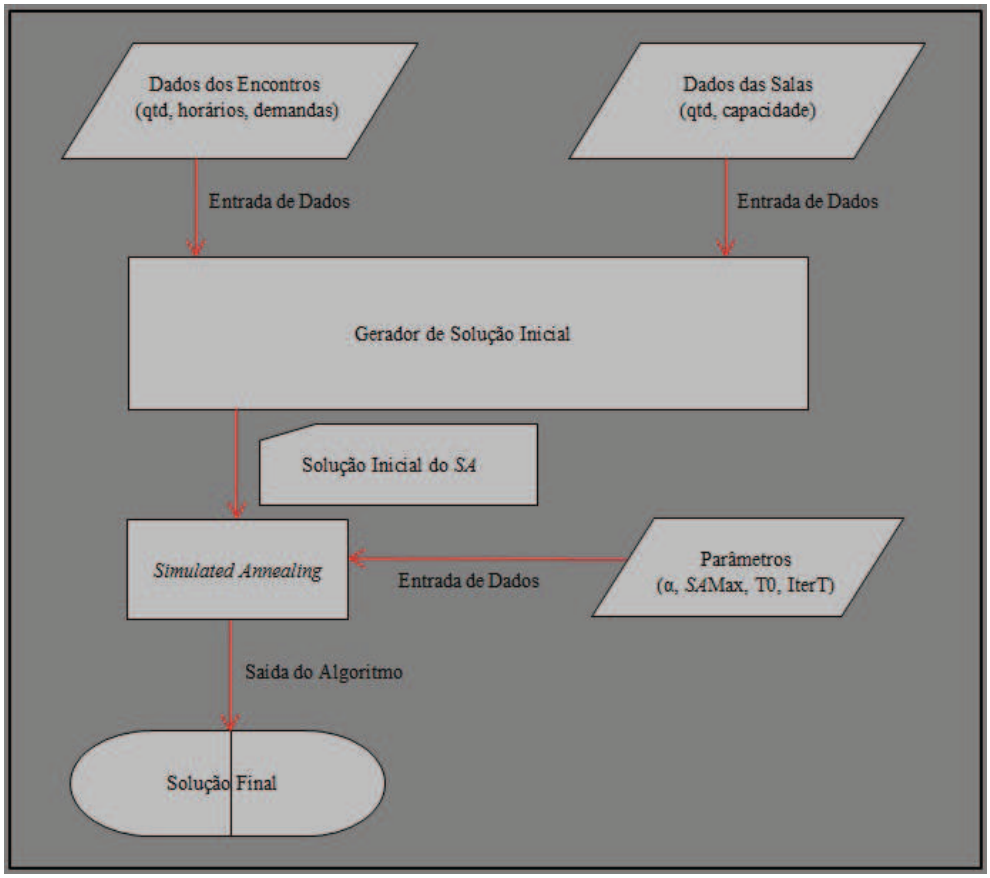


Figura 4.1: Fluxograma de descrição do Método *SA*. (Fonte Própria, 2017)

4.3 O Artifício das Salas Virtuais

Em virtude do tamanho do problema e da disponibilidade de salas da EEIMVR, utilizou-se um artifício, denominado "Salas Virtuais", para que fosse possível realizar a alocação de todos os encontros existentes, e a partir daí, obter uma solução melhor. De forma simplificada, significa dizer que, as aulas que não puderam ser alocadas em Salas Reais, devido suas inviabilidades, foram direcionadas, temporariamente, à estas "Salas Virtuais", que no decorrer do processamento do algoritmo, deixaram de ser utilizadas devido a alta penalidade conferida à estas por causa de sua inviabilidade causada por excesso de estudantes, visto que, suas capacidades são nulas.

O funcionamento das "Salas Virtuais" ocorre da seguinte maneira: Inicialmente, na geração aleatória da solução inicial, todos os encontros são alocados às salas sem se preocupar com a relação demanda/capacidade das salas. Feito isso, a função objetivo faz uma busca por inviabilidades na solução, ou seja, por aulas que foram alocadas em salas cuja as capacidades são menores que a sua demanda, penalizando estes casos em sua avaliação.

Em um segundo momento, o algoritmo *SA* faz uma busca por soluções vizinhas à solução atual. Ao encontrar uma solução de melhor qualidade, ou seja, uma solução com menos penalidades, a solução atual é substituída por essa nova melhor solução, de acordo com os critérios definidos pelo método de resolução. Este processo iterativo continua até que o critério de parada seja atingido e, obviamente, na solução final, o uso destas "Salas Virtuais" é minimizado dada a grande penalidade por seu uso.

A parametrização relativa ao número de "Salas Virtuais" ficou definida, no problema abordado, como a diferença entre o número total de salas e o número de Salas Reais, que no caso estudado, equivalem à 20 salas. Neste sentido, o algoritmo pode utilizar estas salas, em soluções intermediárias, mas, tenta reduzir ao máximo seu uso na solução final.

4.3.1 Representação Utilizada

Para o problema abordado foi utilizado a seguinte representação estrutural: Uma matriz que corresponde a alocação de todos os encontros, sendo, os horários dessas aulas representados pelas linhas da referida matriz, e seus locais de encontro definidos pelas colunas da mesma, representados pelo conjunto somado das salas reais, com as já mencionadas, salas virtuais.

4.4 Procedimento de busca da meta-heurística *Simulated Annealing* no Caso EEIMVR

4.4.1 Geração da Solução Inicial

O método de resolução de problemas escolhido, para o caso de alocação de encontros da EEIMVR, foi o *Simulated Annealing*, Seção 2.2.2.5, e as principais abordagens sobre geração de soluções inciais, contidas na literatura, descritas na Seção 2.2.2.4. Conforme apresentado, esta técnica de resolução inicia sua busca a partir de uma solução inicial qualquer, e em seguida, utiliza-se de refinamentos para obter uma solução viável para o PAAS. Dentre as diversas estratégias na geração de soluções iniciais, optou-se pela abordagem aleatória por sua simplicidade na implementação, por produzir soluções diferentes a cada iteração, e ainda, por fornecer resultados com rapidez. O objetivo da construção da solução inicial é produzir uma tabela de horários válida para que o SA possa iniciar a geração do primeiro vizinho s' da solução.

O pseudocódigo Algoritmo 3, fornece o procedimento de geração de solução inicial. O mecanismo funciona da seguinte maneira. A geração se inicia com uma tabela de horários, denominada $tabelaS0$, vazia. Dado o número de aulas ($nAula$) e encontros ($numEncontros$), caso uma aula i ainda não tenha sido alocada, sorteia-se uma sala ($nSalas$) e verifica-se se esta está vazia no horário ($nHorarios$) daquele encontro. Caso a sala esteja vazia, a alocação é feita. Caso contrário, faz-se um novo sorteio. Esse procedimento é realizado enquanto existir uma aula i com horário de inicio e fim, não alocada. Esse procedimento verifica se existe uma sala vazia com horários de inicio e término disponíveis, caso exista esse conjunto sala e horários possíveis, a aula i é alocada nesse conjunto. Essa busca é controlada por um número máximo de tentativas ($nTentativas$) sem sucesso. E ao atingi-lo, o algoritmo de geração de solução inicial se encerra, e imprime a solução encontrada até aquele exato momento.

Algoritmo 3 - Geração da Solução Inicial

```
1: Procedimento GeracaoSolucaoInicial (nSalas, nHorarios, nTentativas, nu-
   mEncontros)
2: CarregarDados();
3: para (n = 0 até n < nSalas) faça
4:       para (m = 1 até m < nHorarios) faça
5:             tabelaS0 [m][n] ⟵ 0;
6: para (i = 1 até i < nEncontros) faça
7:       para (k = 1 até k < nTentativas) faça
8:        escolha aleatoriamente uma sala n
9:        seja m1 o horario de inicio do encontro i
10:       seja m2 o horario do final do encontro i
11:       Se (a sala n em tabelaS0 está vaga entre m1 e m2) então
12:             Alocar();
13: Retorna tabelaS0;
14: fim GeracaoSolucaoInicial
```

4.4.2 Mecanismo de funcionamento da Estrutura de Vizinhança no problema estudado

A elaboração da estrutura de vizinhança adequada ao problema é um passo extremamente importante ao desenvolver uma meta-heurística [35]. Se a estrutura de vizinhança não for adequada ao problema, as meta-heuristicas que utilizarem esta estrutura de vizinhança podem não fornecer os resultados esperados.

Como mencionado na Seção 2, em um problema de otimização discreta, a vizinhança $N(s)$ de uma solução s é representada pelo conjunto $(s'|d(s',s) \leq \epsilon)$, onde d representa uma dada distância que é relatado pelo movimento do operador. De forma simplificada define-se movimento como uma simples troca de valores distintos que representam as atividades (aulas) exercidas por diferentes docentes, sendo estas indicadas pela matriz $tabelaS[m][n]$, no qual m assume valores que correspondem aos horários de determinadas atividades, e n as salas onde ocorrem estes encontros. Portanto, cada elemento a_{mn} não nulos da matriz $tabelaS[m][n]$, onde $m \in \{1, nhorarios\}$ e $n \in \{0, nsalas\}$, representa uma aula lecionada no horário m na sala n. Em suma, o funcionamento da estrutura de vizinhança

ocorre como se segue:

Em problemas de agendamento, os operadores mais adequados são aqueles baseados em permutações [35] como: Operador de Permuta, Operador de Inserção e o Operador de Inversão, Figuras 2.11, 2.10 e 2.12. Neste trabalho, inicialmente, foi utilizado o operador de permuta para definir o movimento de realocação. Testes computacionais preliminares revelaram que apenas a realocação não era suficiente para atingir os resultados desejados. Então, propôs-se complementar a estrutura de vizinhança com o movimento de troca, que utilizou o mesmo operador de permuta do primeiro movimento, para compor a estrutura de vizinhança utilizada no *Simulated Annealing*. Desta maneira, ficou estabelecido que a estrutura de vizinhança utilizada neste trabalho seria composta pela união da vizinhança gerada pelo movimento de realocação, $\boldsymbol{N^R(s)}$, e pela vizinhança gerada pelo movimento de troca, $\boldsymbol{N^T(s)}$, todos obtidos a partir da pertubação na solução s ocasionada pelo operador de permuta, ou seja, a vizinhança utilizada no SA ficou definida como: $N(s) = N^R(\mathrm{s}) \cup N^T(\mathrm{s})$.

O **Movimento de Realocação**, mostrado pela Figura 4.2, consiste em transferir um determinado encontro, alocado em um horário e sala, para uma outra sala que esteja vazia no mesmo horário. Nesse sentido, a única exigência desse tipo de movimento é que a sala que receberá o encontro, no exemplo a sala 4, esteja disponível (vazia) no horário em que ocorre a aula. A partir da análise dessa figura conclui-se que, o encontro A que originalmente era lecionado na sala 1, no intervalo de horários de 2 à 4, após a realocação, será lecionado na sala 4, no mesmo intervalo de tempo 2 à 4. Para que ocorra um movimento de realocação, existe um número máximo de tentativas que o algoritmo realiza na busca de uma sala disponível, neste caso, a definiu como sendo igual a quantidade de salas do problema. Após atingir esse número de tentativas do algoritmo sem sucesso na busca das salas, este é encerrado. Como em circunstâncias anteriores deste trabalho, a parametrização do número máximo de tentativas ficou definida experimentalmente.

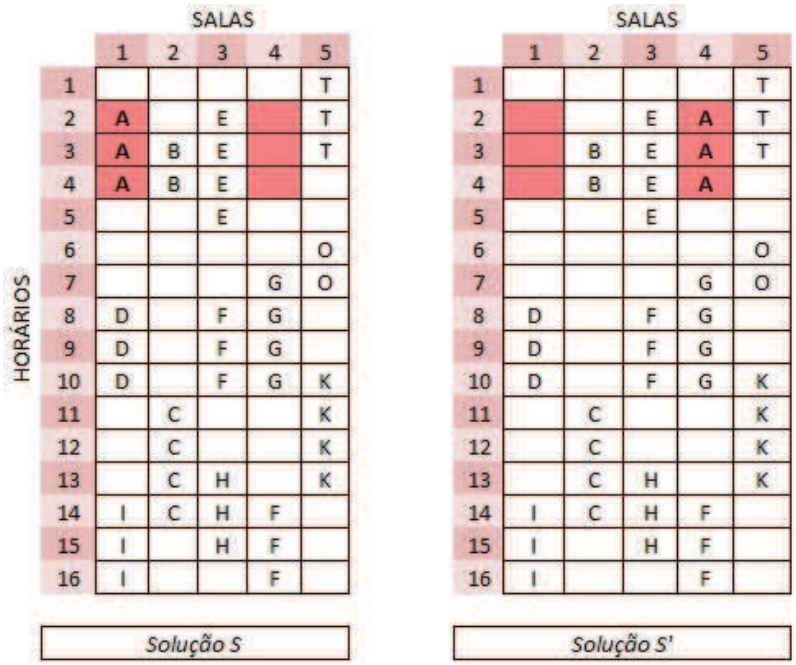


Figura 4.2: Movimento de Realocação. (Fonte Própria, 2017)

O **Movimento de Troca** consiste em permutar a posição entre duas aulas ministradas em salas distintas. Neste caso, verifica-se três tipos de cenários: aqueles em que os horários inicial e final dos dois encontros são coincidentes, como mostrado nas Figuras 4.3 e 4.4, os que possuem seus encontros com horários de início e término deslocados entre si, porém, com o horário de um encontro contido dentro do outro, conforme apresentado nas Figuras 4.5, 4.6, 4.7, 4.8 e 4.9, e ainda, a situação em que os horários de início e fim tem uma interseção, mas um não está contido dentro do outro, condição verificada nas Figuras 4.10 e 4.11. Uma característica importante do movimento de troca é a definição do intervalo em que ocorrem suas trocas. Sendo assim, após a definição desse intervalo de tempo ocasionado pela busca do encontro inicial A, o algoritmo faz uma busca por um encontro G que ocorra em qualquer sala que não a de origem (sala 1 no exemplo) e que esteja dentro do mesmo intervalo de horários definido. A busca por esse encontro é realizada até um número máximo de tentativas estipulado, que para este caso, foi definido como igual ao número de salas disponíveis na EEIMVR. Após o

número máximo de tentativas sem sucesso, o algoritmo encerra-se informando que não foi possível realizar o movimento de troca.

No primeiro caso o movimento descrito pode ser acompanhado por meio da Figura 4.3 e Figura 4.4. Com esse auxílio verifica-se seu funcionamento através da troca entre as aulas A e G, que consiste em alocar, temporariamente, a aula A da sala de origem 1 em um vetor auxiliar, denominado AULA, e transferir a aula G, que inicialmente estava alocada na sala 4, para a sala de origem 1. Feito isso, a aula A é então realocada do vetor para a sala destino 4, finalizando assim, o movimento. Vale ressaltar que tudo isso ocorre somente após estar determinado o intervalo de tempo em que ocorre essas trocas, visto que, neste caso, devem ser coincidentes.

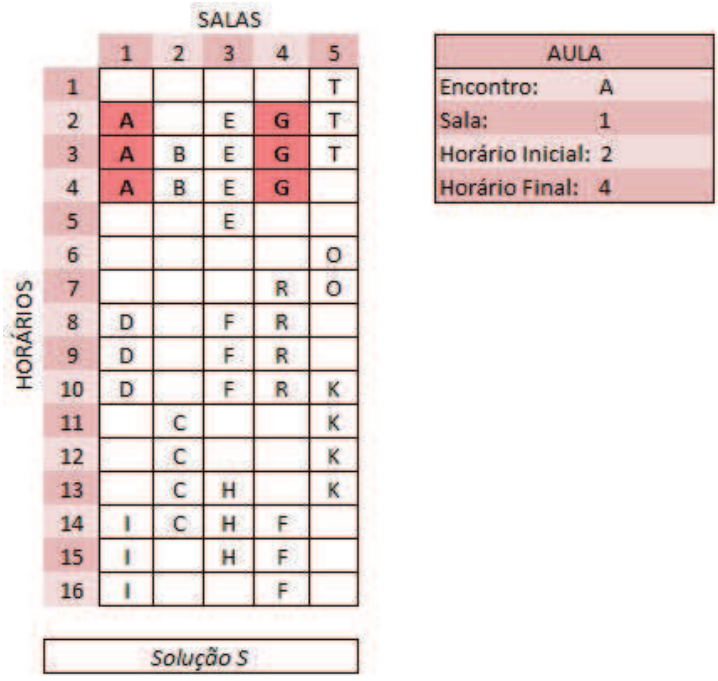


Figura 4.3: Movimento de Troca - Horários Coincidentes. (Fonte Própria, 2017)

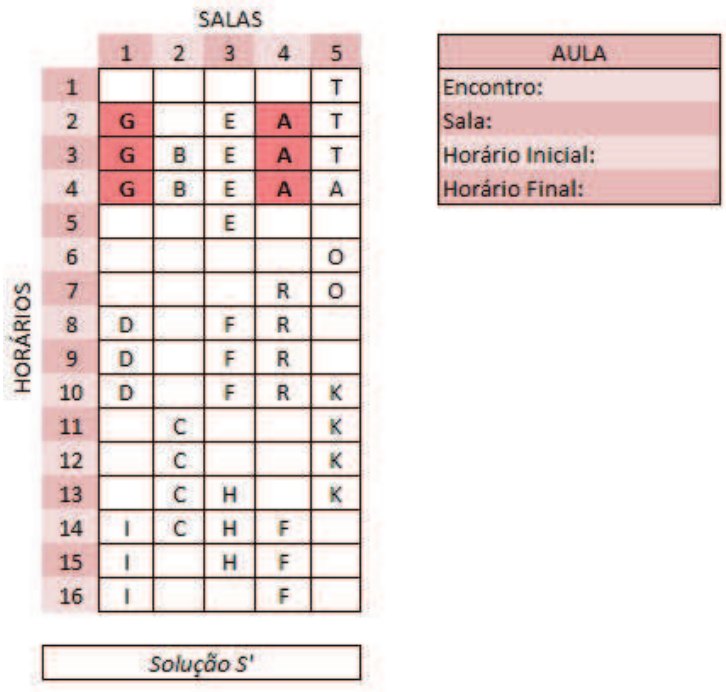


Figura 4.4: Movimento de Troca - Horários Coincidentes. (Fonte Própria, 2017)

O segundo caso pode ser verificado através das Figuras 4.5, 4.6, 4.7, 4.8 e 4.9. Nesta situação, a troca também ocorre entre as aulas A e G, no entanto, devido ao fato dos horários estarem deslocados entre si, utilizou-se um vetor auxiliar, denominado AULA, para que haja a troca sem que ocorra perda de informações. Neste caso, todos os dados relevantes à aula G, como encontro, sala e intervalo de horários, são inseridos na memória externa. A única exigência para que ocorra a troca entre as aulas A e G é a existência de aula única em cada uma dessas salas no intervalo estipulado. Ou seja, caso exista mais de um tipo de aula nessas salas, nos intervalos de horários definidos, não haverá a troca entre as aulas A e G. Na sequencia, a memória da sala 4 é limpa para que receba as informações da aula A sem prejuízo dos dados. Então, a aula A que se encontra na sala 1 é realocada na sala 4. Feito isso, limpa-se a memória da sala 1, e copia-se para esta todo conteúdo da memória externa AULA, aula G, finalizando assim, o movimento. A observação a cerca da determinação dos intervalos de tempo também se aplica para o segundo

e terceiro caso do movimento de troca, no entanto, cada caso trata a determinação desse intervalo de maneira particular.

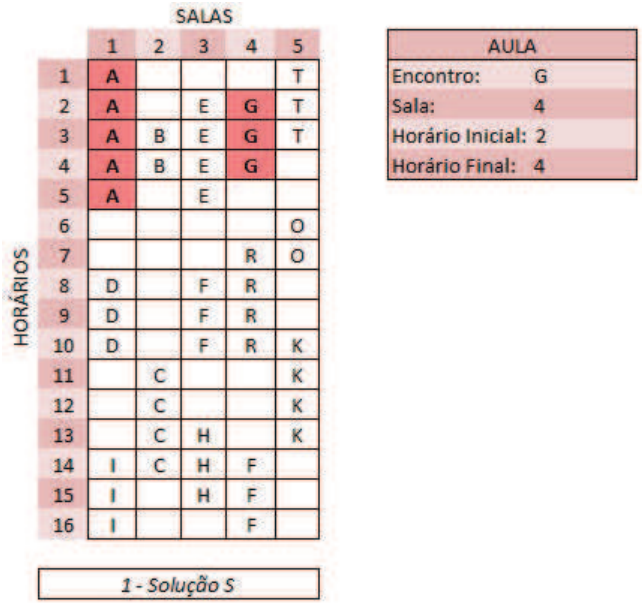


Figura 4.5: Movimento de Troca - Horários Deslocados. (Fonte Própria, 2017)

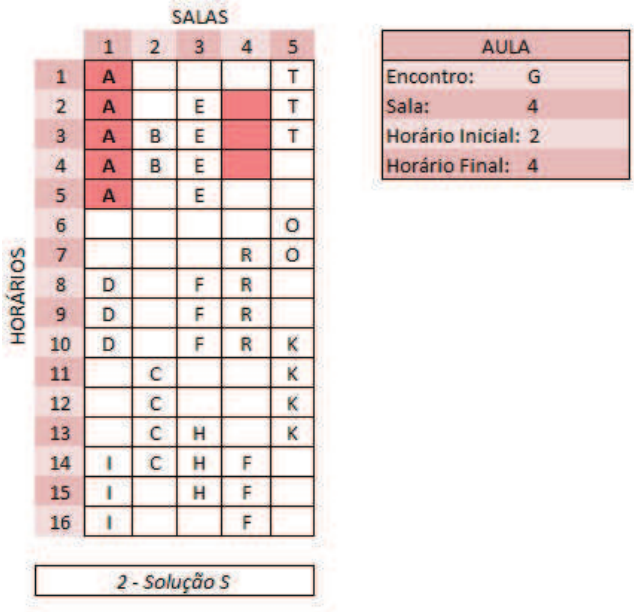


Figura 4.6: Movimento de Troca - Horários Deslocados. (Fonte Própria, 2017)

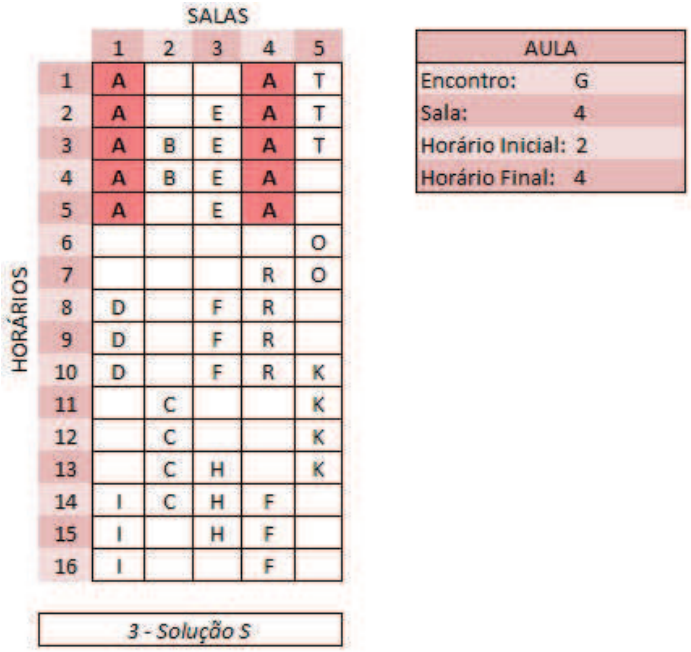


Figura 4.7: Movimento de Troca - Horários Deslocados. (Fonte Própria, 2017)

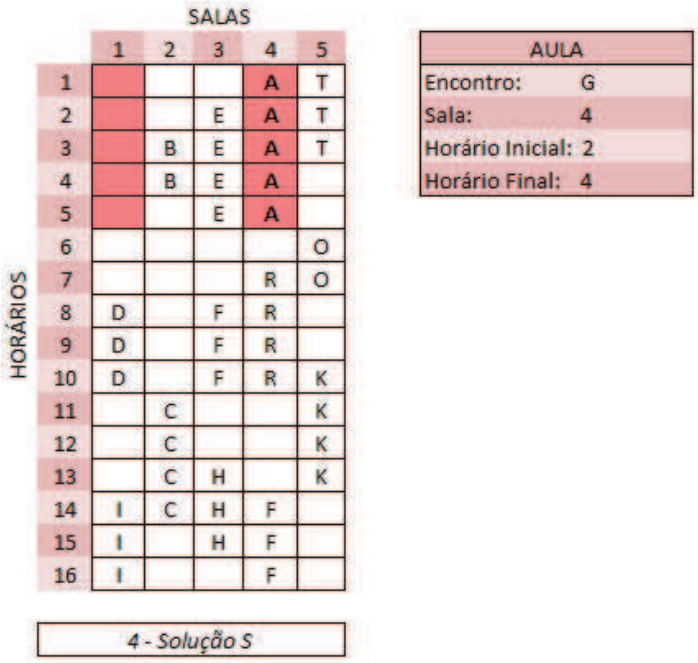


Figura 4.8: Movimento de Troca - Horários Deslocados. (Fonte Própria, 2017)

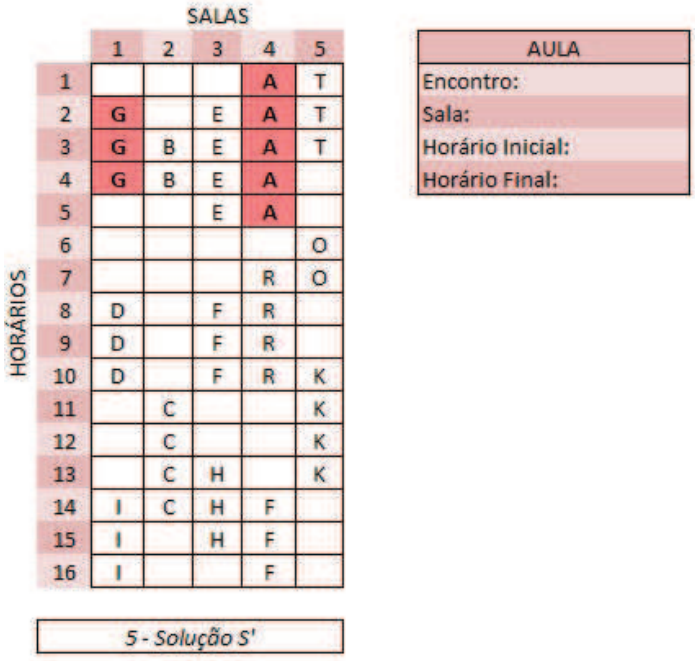


Figura 4.9: Movimento de Troca - Solução Vizinha. (Fonte Própria, 2017)

Por fim, o último caso se configura mediante as Figuras 4.10 e 4.11. Nesta circunstância, inicialmente defini-se um intervalo que envolve os dois encontros A e G, a partir daí verifica-se na sala de cada um dos encontros, nesse intervalo envolvente, a existência de um único encontro. Caso exista somente esse único encontro, a troca entre A e G irá ocorrer da mesma forma como descrito nos casos anteriores, por meio de alocação temporária em memória externa. Porém, na situação em que neste intervalo envolvente existem mais de um único encontro na mesma sala analisada, a troca não dar-se-á, e o processo é reiniciado, procedimento verificado na Figura 4.12. A limitação da ocorrência de uma aula se deve principalmente por este último caso, ou seja, uma alocação não irá ocorrer quando, no intervalo definido, existirem mais de um tipo de encontro, na mesma sala. O que significa dizer que, no exemplo mostrado na Figura 4.12, não pode haver trocar entre A e G, e nem entre B e G, pois, no intervalo definido (H1 até H6) existem dois encontros na sala 1, impossibilitando estas possíveis trocas.

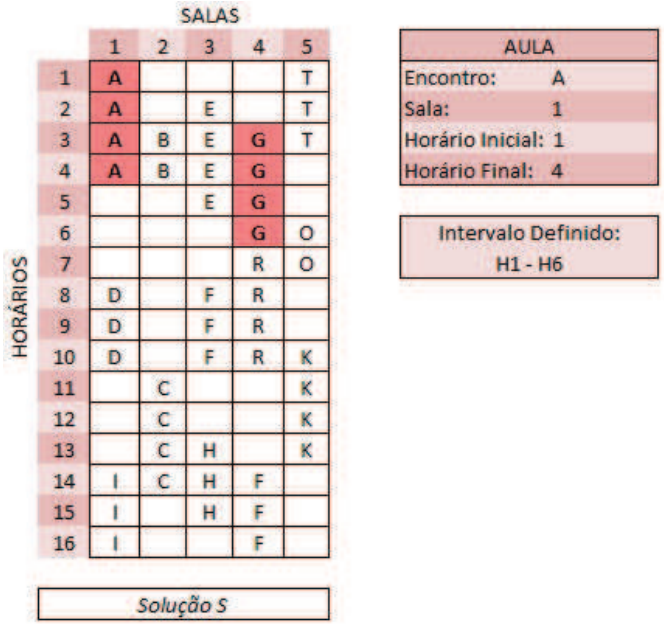


Figura 4.10: Movimento de Troca - Interseção entre Horários. (Fonte Própria, 2017)

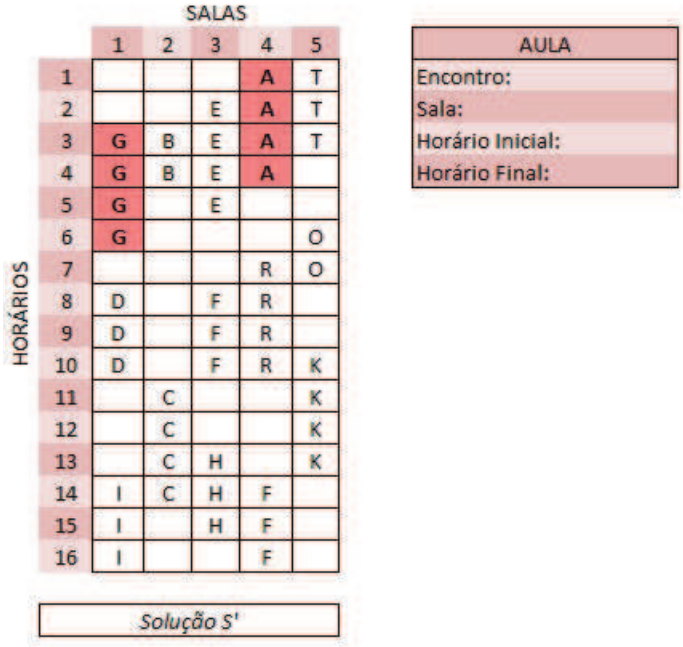


Figura 4.11: Movimento de Troca - Interseção entre Horários. (Fonte Própria, 2017)

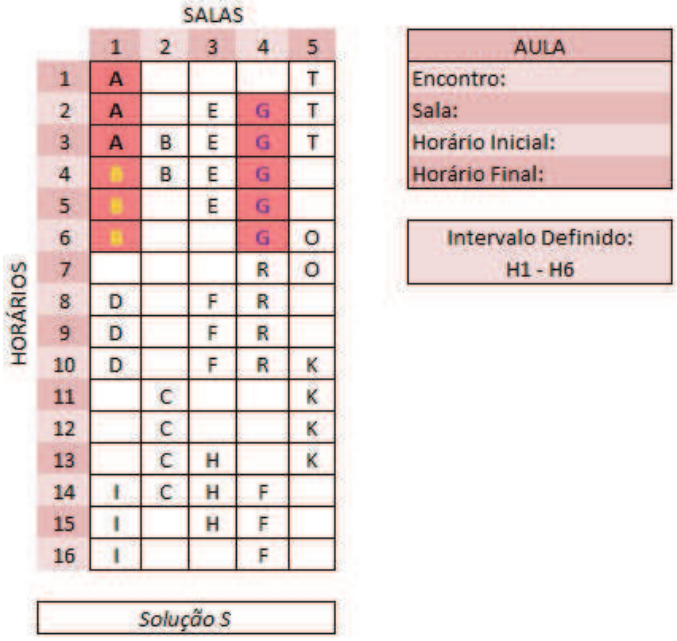


Figura 4.12: Movimento de Troca - Impossibilidades de Trocas. (Fonte Própria, 2017)

4.5 Formulação Matemática

Após testes com as abordagens heurísticas propostas, foi percebido que o uso de mais estruturas de vizinhança poderiam tornar o problema muito complexo. Neste sentido, o uso de uma abordagem exata que, modela o problema através de expressões matemáticas, e que não precisa enumerar possibilidades, se torna uma solução viável. Diante disso, segue abaixo uma formulação matemática proposta, envolvendo programação linear inteira, para a resolução do Problema de Alocação de Aulas em Salas na EEIMVR. Considere as seguintes variáveis binárias:

- $x_{ijs} = \begin{cases} 1, & \text{se o encontro } i \text{ está alocado no horário } j \text{ na sala } s. \\ 0, & \text{caso contrário.} \end{cases}$

- $y_{is} = \begin{cases} 1, & \text{se o encontro } i \text{ utiliza a sala } s \text{ em algum horário.} \\ 0, & \text{caso contrário.} \end{cases}$

Sejam os seguintes conjuntos e dados de entrada:

- E conjunto de encontros.
- S conjunto de salas.
- H conjunto de horários.
- HV_i conjunto de horários válidos para o encontro i.
- n_i número de horas do encontro i.
- d_i demanda do encontro i.
- c_s capacidade da sala s.
- po penalidade por ociosidade na sala.
- pe penalidade por excesso de alunos na sala.
- p_{is} custo de alocar o encontro i na sala s dado por:
  $p_{is} = \begin{cases} po * (c_s - d_i), & \text{se } d_i \leq c_s. \\ pe * (d_i - c_s), & \text{caso contrário.} \end{cases}$

O problema de alocação de encontros pode ser descrito pela seguinte formulação matemática dada pelas Equações (4.1) à (4.9):

$$Min \quad \sum_{i \in E} \sum_{s \in S} p_{is} y_{is} \tag{4.1}$$

$S.a.$

$$\sum_{s \in S} x_{ijs} \leq 1, \forall i \in E, \forall j \in H, \tag{4.2}$$

$$\sum_{i \in E} x_{ijs} \leq 1, \forall s \in S, \forall j \in H, \tag{4.3}$$

$$\sum_{s \in S} \sum_{j \in H} x_{ijs} = n_i, \ \forall i \in E, \tag{4.4}$$

$$x_{ijs} = 0, \forall i \in E, \forall s \in S, \forall j \notin HV_i, \tag{4.5}$$

$$y_{is} \geq x_{ijs}, \forall i \in E, \forall s \in S, \forall j \in H, \tag{4.6}$$

$$\sum_{s \in S} y_{is} = 1, \forall i \in E, \tag{4.7}$$

$$x_{ijs} \in \{0, 1\}, \ \forall i \in E, \forall j \in H, \forall s \in S, \tag{4.8}$$

$$y_{is} \in \{0, 1\}, \ \forall i \in E, \forall s \in S. \tag{4.9}$$

A função objetivo 4.1 minimiza a ociosidade e a sobrecarga das salas. As restrições 4.2 impedem que um encontro seja alocado em mais de uma sala em um mesmo horário. As restrições 4.3 não permitem que múltiplos encontros sejam alocados na mesma sala em um dado horário. As restrições 4.4 afirmam que o número de horas de cada encontro deve ser respeitado. As restrições 4.5 afirmam que um encontro não pode ser alocado fora de seu horário previsto. Já as restrições 4.6 ligam as variáveis x e y, e afirmam que se houve uma alocação de um encontro i em uma sala s em algum horário, então esta sala s foi utilizada por i, e portanto esta alocação deve ser contabilizada na F.O. . A alocação obrigatória de um encontro à uma sala é tratada pelo conjunto de restrições 4.7. As demais restrições tratam da Integralidade e Não-Negatividade.

Capítulo 5

Resultados Experimentais

Este capítulo se destina na descrição e apresentação dos resultados obtidos através dos experimentos realizados com o modelo exato e o aproximado, para o problema abordado, neste trabalho. Posteriormente, realizou-se uma comparação entre os dois resultados computacionais obtidos e aquele utilizado comumente pela instituição de ensino, com suas devidas considerações.

Os testes computacionais foram realizados em uma máquina com sistema operacional Windows 7 Ultimate e processador Intel Core i5-2450M CPU 2x 2.50GHz. Além disso, o método exato foi desenvolvido utilizando o mecanismo de programação matemática de alto desempenho, solver IBM ILOG CPLEX Optimization Studio [4], Versão 12.6, e o método aproximado, por meio da meta-heurística *Simulated Annealing*, elaborada no CODE::BLOCKS [1] - Ambiente de Desenvolvimento Integrado (IDE), que permitiu escrever todo o código em linguagem de programação C.

No caso do modelo matemático, realizou-se a análise do resultado da função objetivo, seu tempo de processamento e o número de iterações utilizados, pelo modelo, na obtenção de uma solução viável para o problema, bem como, o GAP fornecido pelo software.

Na avaliação do método aproximado, foram utilizadas 12 (doze) instâncias,

definidas segundo a remoção de algumas salas específicas, para que fosse possível analisar o comportamento do algoritmo diante das diferentes situações impostas, e a partir daí, realizar as devidas comparações e conclusões. Para cada instância, foram realizadas 15 (quinze) execuções com diferentes sementes de geração de números aleatórios, escolhidas de acordo com o critério adotado no uso de números primo. Além disso, de todo esse experimento, foi recolhido os melhores resultados, suas médias e desvios padrões. Também realizou-se uma análise das soluções para encontrar a localização do gargalo, dia da semana e horário que inviabilizam o problema analisado, em cada instância apresentada, e da medição dos tempos, fornecendo suas respectivas médias e desvios padrão.

5.1 Parametrização do Problema

Ainda que o problema abordado represente uma situação típica e corriqueira, no universo acadêmico, e apesar de suas características estarem bem fundamentadas na área de otimização, não é de conhecimento dos autores um conjunto de parâmetros preestabelecidos capaz de ser utilizado nos algoritmos desenvolvidos, neste trabalho. Dessa maneira, a parametrização dos algoritmos foi definida segundo um conjunto próprio de dados, determinados de acordo com experimentos previamente realizados, e seus valores apresentados na Figura 5.1 abaixo.

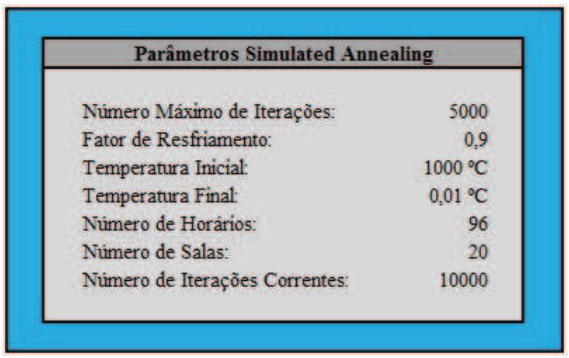


Figura 5.1: Parâmetros utilizados no procedimento SA. (Fonte Própria, 2017)

m	n	α	SAmax	T_0	T	IterT
96	20	0.9	5000	1000	0.01	10000

Tabela 5.1: Notação dos Parâmetros no Código Desenvolvido. (Fonte Própria, 2017)

Sendo cada notação representando:

m, número de horários disponíveis para ocorrência dos encontros;

n, número de salas reais disponíveis;

$\alpha \in [0;1]$, a representação do fator de resfriamento;

SAmax, o número máximo de iterações, a cada temperatura apresentada;

T_0, a temperatura inicial, geralmente um valor muito elevado;

T, a temperatura final;

IterT, o número de iterações corrente.

De acordo com o que foi citado em [31], a escolha adequada dos parâmetros é algo extremamente importante na definição da solução inicial, bem como na execução do algoritmo, e permite que haja um certo grau de diversificação no início da busca, e maior intensificação no final do processo, objetivos esperados neste caso. Ainda segundo os autores, apenas a título de conhecimento, diferentemente do que foi adotado neste trabalho, aqueles afirmam o seguinte:

> "Quando a construção da solução inicial já produz uma solução de boa qualidade, a temperatura inicial não precisa ser muito alta. Segundo essas observações, temperaturas iniciais mais altas não proporcionam melhorias no resultado final, pois, permitem aceitar soluções muito ruins e a melhora é observada somente após várias etapas de resfriamento."

5.2 Resultados da Formulação Heurística

Conforme mencionado acima, foram desenvolvidas 12 (doze) instâncias definidas a partir da remoção de salas específicas, estabelecidas pela Direção da Unidade. Diante disso, utilizou-se, para cada instância, um conjunto de valores, definidos como sementes de geração de números aleatórios, se valendo do critério de números primos, para estabelecer os diferentes cenários analisados neste trabalho. A partir daí, essas informações foram inseridas no algoritmo *Simulated Annealing* desenvolvido, afim de obter dados relevantes à respeito do comportamento deste programa, em diferentes situações. A tabela 5.2 apresenta os dados utilizados no experimento descrito.

Cenários	Salas Removidas
Instância 1	Nenhuma
Instância 2	B11
Instância 3	D30A
Instância 4	D30B
Instância 5	D42
Instância 6	D30A / D30B
Instância 7	D30A / D42
Instância 8	B11 / D42
Instância 9	D30B / B11
Instância 10	D30B / D42
Instância 11	D30A / B11
Instância 12	B11/D30A/D30B/D42

Tabela 5.2: Parâmetros dos Experimentos. (Fonte Própria, 2017)

Em posse dos resultados fornecidos por esse experimento, avaliou-se cada cenário, e extraiu destes, os melhores resultados (menor valor da função objetivo) dado em negrito, suas médias e desvios padrões. De acordo com que foi analisado ao longo desse trabalho, o objetivo do problema é o de minimizar o custo de alocação de um encontro em uma sala e horário disponíveis, sendo assim, esses melhores valores, são verificados no menor valor da função objetivo, visto que, sua finalidade é apresentada neste sentido. Os resultados desses experimentos podem ser verificados, resumidamente, na tabela 5.3 abaixo, e através do gráfico representado pela Figura 5.2.

Cenários	Melhor FO	Melhor Semente Média	Média	Desvio Padrão
Instância 1	**3659**	11	3686	18,6
Instância 2	**34080**	17	54036	22.327,0
Instância 3	**34355**	11	52319	22.706,5
Instância 4	**34342**	23	39385	13.446,5
Instância 5	**34084**	7	49057	21.848,6
Instância 6	**137487**	2	149495	20.501,1
Instância 7	**136721**	41	161650	22.341,1
Instância 8	**136528**	47	156521	22.355,4
Instância 9	**136732**	11	148709	20.490,3
Instância 10	**136690**	7	151688	21.847,6
Instância 11	**136732**	5	144738	16.802,3
Instância 12	**487188**	5	498041	18.229,5

Tabela 5.3: Experimento baseado na Restrição do uso de Salas. (Fonte Própria, 2017)

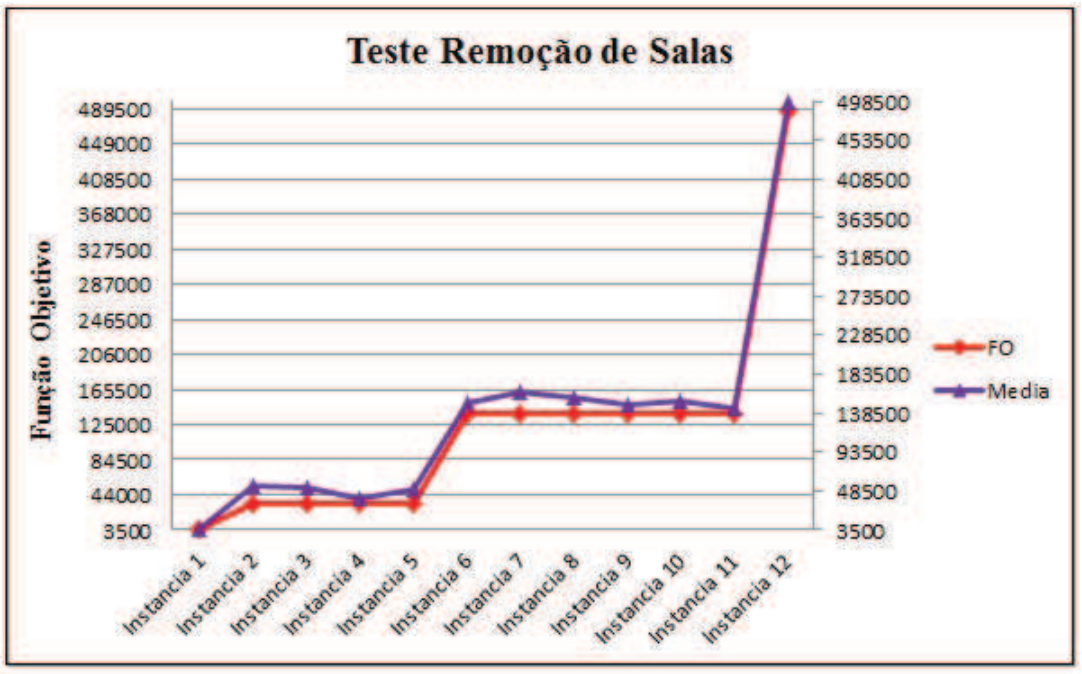


Figura 5.2: Comportamento no Teste de Remoção. (Fonte Própria, 2017)

O experimento foi baseado na remoção de salas e na alteração dos geradores de números aleatórios, para que fosse possível analisar o comportamento da heurística diante dos diferentes cenários apresentados. Sendo assim, para que isso fosse possível, no código, alterou-se as capacidades das salas indicadas para 01 (um) aluno, o que implica na proibição do uso desta sala na alocação. Dessa maneira, o algoritmo poderá, a partir dos valores atribuídos às penalizações na função objetivo, garantir que o número de assentos vazios, nas salas, será o menor possível, e o mais importante, a função objetivo tentará ao máximo alocar os encontro em salas com capacidades compatíveis com o número de alunos inscritos nas disciplinas, dada a alta penalização atribuída a esse fator.

O ideal, neste caso, seria montar um horário sem o uso destas salas, mas, não sendo possível, pelo menos, que sejam utilizadas o menor número possível entre elas. A parametrização escolhida neste caso, se deu, pois, ao optar por atribuir o valor unitário às capacidades dessas salas, assim, seria possível diferenciar estas das salas virtuais, que possuem capacidades nulas. Desta forma, usa-se estas salas em preferencia às salas virtuais devido sua prioridade maior. Por isso, utiliza-se uma capacidade baixa, mas, positiva.

Além disso, o algoritmo foi executado com os dados originais corrigidos (alteração dos geradores de números aleatórios) e com as possíveis combinações de remoção de salas, conforme apresentado na tabela 5.2.

A análise do resultado permitiu concluir que a remoção de uma única sala já é suficiente para tornar o problema inviável, pois, sempre haverá alocação de encontros em salas cuja capacidade seja 01 (um), ou seja, salas removidas, o que indica saturação de recursos (salas). Por último, mas, não menos importante, a verificação dos resultados, associados ao mapeamento das aulas, mostrou que o gargalo se manifesta às quintas-feiras, no horário de 14:00 às 16:00, pois de acordo com as análises, é o único horário envolvido neste intervalo, em que todas as salas são necessárias, vide Figura 5.3, marcação na coloração azul.

Função Objetivo: 34424
Inviabilidades: 2

H-S	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
1																	212			
2				68			26	91				4				66			106	94
3				68	200		26	91				4				66			106	94
4		72	209		200		27	91						44				1	21	67
5		72	209		200		27	91						44				1	21	67
6					200													1		
7									24			32								
8	70	149	41	114	172		98	117	24	122	165	32		186			153		92	136
9	70	149	41	114	172		98	117	24	122	165			186			153		92	136
10	70	144	41	114	134			120	33	129	165			189		127	156	25	92	136
11	70	144	90	114	134			120	33	129	165			189		127	156	25	92	136
12	140	168	90		121		201	176		135	184			56		18			166	130
13	140	168	90		121		201	176		135	184			56	113				166	130
...					...						...									...
...					...						...									...
...					...						...									...
26	174	182	163	95	131		164	97		115	117	33	17	129	25	74	215		206	45
27	174				131			97		115	117	33		129	25		215		206	45
28	161		152	119	180		158	145		155	77	169	84	130		56	191		105	139
29	161	203	152	119	180		158	145		155	77	169	84	130		56	191		105	139
30	161	203	152		180		158	145		210	77	169	84				191		105	139
31		203	152		180		158	145		210	77	169	84				191		105	139
32																				
33																				
34		71			151			12	53	86		5			3		34	54	104	
...					...						...									...
...					...						...									...
...					...						...									...
45	167	148	142	193	176		211	195	111	181	124	199	190	98	155	96	214	125	97	137
46	167	148	142	193	176		211	195	111	181	124	199	190	98		96	214	125	97	137
47	167	148	142		176		211		111	181										137
48																				
49																				
50	106	36	151		107		30	85	3	128	55	5			64				16	
51	106	36	151		107		30	85	3	128	55	5			64	185			16	
52	103	36	60	80	126		31	81	6	209	67	2	133	109		185	157	47	16	71
53	103		60	80	126		31	81	6	209	67	2	133	109		185	157	47	16	71
54												2					157		16	
55		35			102		202	88				8	99						76	
56	141	35	82	110	102	124	202	88	46	174	48	8	99	132	69	61		23	76	37
57	141	35	82	110	102	124	202	88	46	174	48	8	99	132	69	61	194	23	76	37
58	141	131	82	110	39		49	125	22	150	38				63		194	62	183	10
59	141	131	82	110	39		49	125	22	150	38				63		194	62	183	10
60		170	178		192		204	187						138			208	160	183	89
61		170	178		192		204	187						138			208	160		89
62		170	178		192		204	187						138			208	160		89
63		170	178		192		204	187						138			208	160		
64																				
65																				

Figura 5.3: Indicação do gargalo na solução fornecida pelo *SA*. (Fonte Própria, 2017)

Além disso, pode-se perceber que a sala mais relevante, dentre as testadas, é a sala D30A (indicado na tabela de saída pela sala 06), pois, sua remoção provocou maior aumento da função objetivo. Outra forma de avaliar é por meio da análise do tempo de processamento do algoritmo. Seu resultado apresentou a semente 23 como sendo aquela que forneceu o menor tempo na obtenção de uma boa solução (20,73 segundos). Este dado pode ser observado, resumidamente, por meio da Tabela 5.4.

2	3	5	7	11	13	17	19	23	...	Média	Desvio Padrão
20,86	20,77	20,86	20,84	20,86	20,86	20,92	20,75	**20,73**	...	20,86	0,075

Tabela 5.4: Tempo de Processamento do Experimento Heurístico. (Fonte Própria, 2017)

Adicionalmente, com o intuito de visualizar o comportamento da trajetória de busca do método, diante dos vários cenários estudados neste trabalho, selecionou-se a instância 1, e todos os seus dados fornecidos pelo SA, na plotagem do gráfico representado pela Figura 5.4. Devido a elevada quantidade de dados fornecidos pelo método, 110 blocos de temperatura de 5000 iterações cada, optou-se pelo uso da amostragem, utilizando 12 diferentes informações que representam situações particulares na meta-heurística, no desenvolvimento desse gráfico. Os seguintes dados foram utilizados: **Temperatura Atual (T°C)** - Eixo Secundário, **Melhor Valor da Função Objetivo (S^*)** e **Valor da Função Objetivo Atual (S)** - Ordenadas e **Amostra** - Abcissas. Além disso, o modelo ainda fornece o Valor Inicial e Final da Função Objetivo, permitindo uma comparação mais realista.

- Função Objetivo Inicial: 39652838
- Função Objetivo Final (Amostra 12): (S^* = 3760, T = 0,0092 °C)

Partindo do pressuposto que o objetivo é a **minimização**, a análise do gráfico, a partir desse histórico de dados, permite identificar os pontos onde a variação da FO é mais acentuada (S - S^*), observados pelos picos do gráfico na amostra 7, 10 e 11 respectivamente, o que significa dizer que a probabilidade da solução S^* ser escolhida e a solução S atual ser rejeitada é muito grande, visto que, valores que melhoram a função custo são sempre aceitos.

Além disso, o gráfico indica os pontos críticos de Mínimos Locais e Mínimo Global, amostras 3, 4 e 12, respectivamente. Nestas situações, apesar da variação da Função Objetivo ser muito pequena, quando comparada às amostras 7, 10 e 11, a solução S^* ainda é a melhor solução do modelo. Ressalta-se ainda que, para cada amostra apresentada no gráfico, tem-se uma temperatura em que as soluções S^* e

S foram geradas, visto que, esta é utilizada como um dos parâmetros de controle do método SA.

Para o problema real da EEIMVR representado pela Figura 5.4, um procedimento de melhoria local, como métodos heurísticos, partiria de uma solução inicial, e a cada iteração, faria uma busca nas vizinhanças da solução atual até localizar uma solução melhor, e continuaria até que não se conseguisse mais encontrar nenhuma solução melhor nas vizinhanças da solução atual. O inconveniente no uso de heurísticas de melhoria local é que esse procedimento terminaria quando encontrasse um ponto ótimo, que poderia não ser necessariamente um Ótimo Global, visto que, este ponto poderia estar localizado fora da vizinhança da solução atual. A única maneira desse procedimento encontrar o Ótimo Global, seria se, por acaso, este estivesse localizado na vizinhança da solução atual. No caso real estudado, esse procedimento cessaria na amostra 4. São nesses casos que o uso da meta-heurística é aconselhada, devida sua habilidade de escapar de um Ótimo Local e realizar uma busca intensa na região de soluções viáveis até encontrar o Ótimo Global. Sendo assim, a partir da meta-heurística SA, o procedimento de busca ao invés de cessar na amostra 3 e 4, permanece a busca até atingir a amostra 12, melhor solução encontrada.

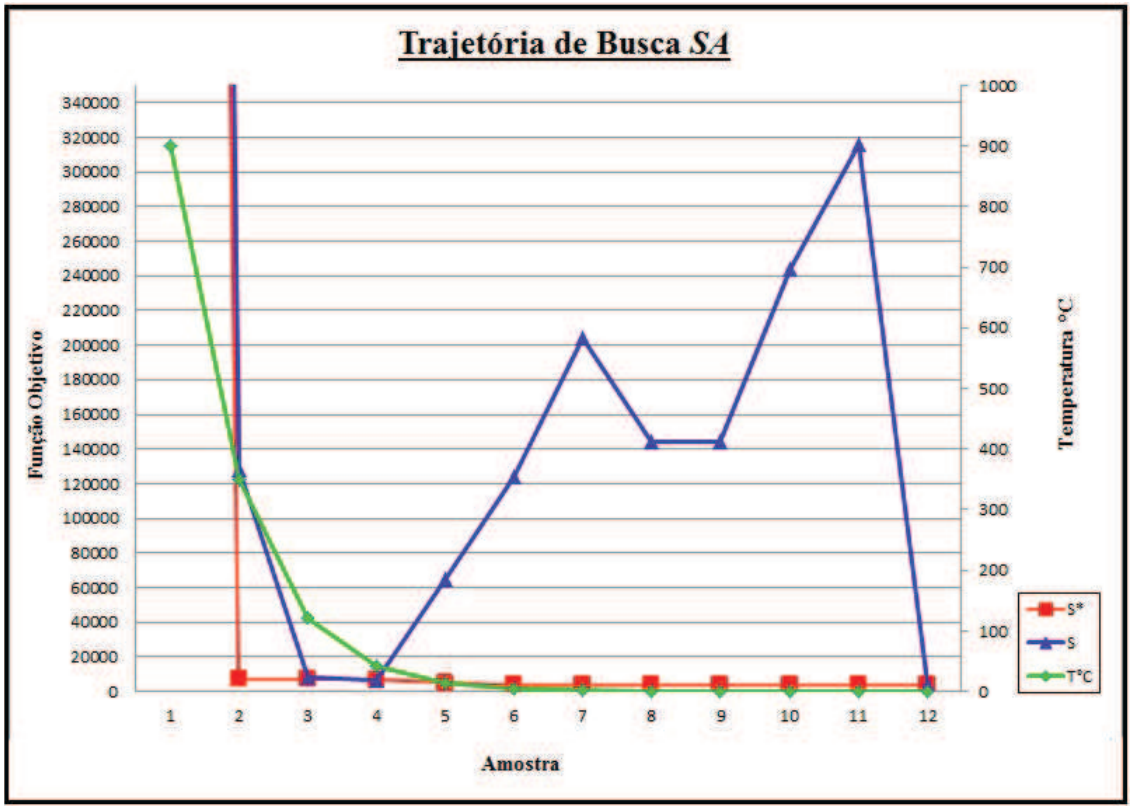


Figura 5.4: Comportamento *SA* no Caso EEIMVR. (Fonte Própria, 2017)

5.3 Resultados da Formulação Exata

Como mencionado anteriormente, foi desenvolvido um modelo exato a partir de formulações matemáticas para que fosse possível realizar um estudo sobre as formas computacionais de resolução do problema de alocação de aulas em salas, e a maneira manual utilizada por algumas instituições de ensino. Para isso, elaborou-se uma função objetivo que minimiza o custo de alocação de aulas em horários e salas disponíveis com suas devidas restrições, e os implementaram no solver de otimização, *CPLEX* - Versão 12.6.

O problema se baseia em Programação Linear Inteira, e fornece os seguintes valores apresentados na tabela 5.5.

Função Objetivo	Tempo Processamento (seg)	Número Iterações	GAP (%)
3625	2,37	564	0,00

Tabela 5.5: Experimento Exato. (Fonte Própria, 2017)

Os resultados obtidos nesta formulação indicam que o modelo matemático possui uma melhor qualidade quando comparado ao método heurístico, visto que, o valor da função objetivo deste é menor que aqueles fornecidos heuristicamente, dado o objetivo do problema, que é o de minimização do custo de alocação. Além disso, uma outra forma de testar o desempenho do modelo matemático, se dá através de seu tempo de processamento. Neste caso, o modelo matemático forneceu um tempo muito menor que o modelo heurístico, o que ratifica sua superioridade sobre este. De modo geral, a instância não levou mais que 2,37 segundos no processamento do algoritmo e encontrou o ótimo em aproximadamente 564 iterações. A análise do GAP é outra forma de avaliá-lo. A tabela 5.5 mostra que a distância entre o resultado fornecido pela função objetivo e o ótimo é nula, o que significa que este resultado é o melhor que poderia ser obtido, ou seja, o ótimo.

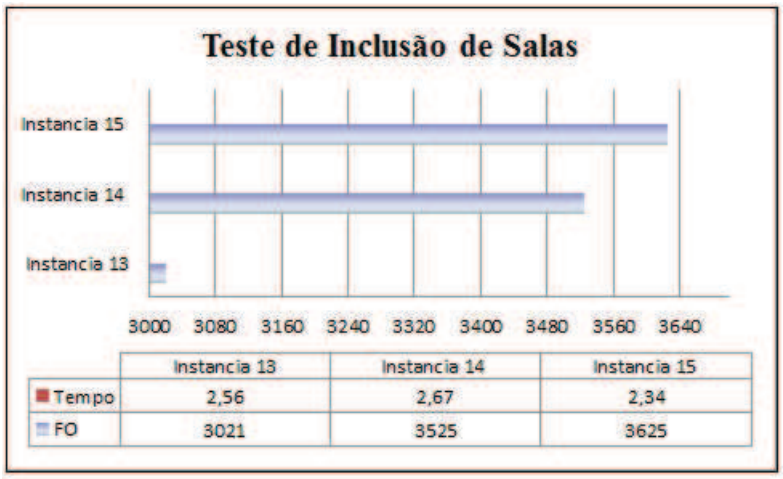
Em geral, a Seção 5.2 descreveu os experimentos realizados, bem como, os resultados obtidos, no **Teste de Remoção de Salas**, o qual, o modelo exato não foi capaz de resolver para as Instância 2 a 12, por serem instâncias inviáveis

devido a quantidade insuficiente de salas para a completa alocação das aulas, como mostra a Figura 5.3. Diante disso, realizou-se outro experimento, agora com base em um **Teste de Inclusão de Salas**, e o resultado indicou o modelo matemático como possível método de resolução, dada sua capacidade de resolvê-lo, apesar do resultado fornecido pelo teste de remoção. Para isso, foram desenvolvidas outras 03 (três) novas instâncias, tomando a Instância 1 como referência, vide Seção 5.2. Sendo assim, temos:

- **Instância 13:** possui todas as salas da Instância 1, e ainda, uma sala com capacidade de 40;
- **Instância 14:** possui todas as salas da Instância 1, e ainda, uma sala com capacidade de 60;
- **Instância 15:** possui todas as salas da Instância 1, e ainda, uma sala com capacidade de 80;

Instâncias	Salas	Função Objetivo	Tempo (seg.)
Instância 13	40	3021	2.56
Instância 14	60	3525	2.67
Instância 15	80	3625	2.34

Tabela 5.6: Resultados do Teste Inclusão de Salas. (Fonte Própria, 2017)



	Instancia 13	Instancia 14	Instancia 15
Tempo	2,56	2,67	2,34
FO	3021	3525	3625

Figura 5.5: Comportamento no Teste de Inclusão. (Fonte Própria, 2017)

Todas as principais características e performance desse experimento podem ser observadas na Tabela 5.6 em associação com o gráfico representado pela Figura 5.5. Sendo assim, se fosse possível incluir uma dessas salas, apresentadas nesse experimento, dentre todas que a escola já utiliza, aquela que traria melhor resultado, em termos da análise em foco neste trabalho, com base nos valores da função objetivo da tabela, seria a sala representada pela Instância 13, ou seja, com capacidade 40.

É importante mencionar que, determinar a melhor distribuição das capacidades das salas é em si um problema de otimização, contudo, a análise deste problema não faz parte do escopo deste trabalho. O objetivo é auxiliar na tomada de decisão para uma possível remoção/inclusão de salas do campus. Portanto, não é possível remover nenhuma sala no contexto atual, uma vez que, a remoção de qualquer uma das salas testadas gerou um problema de otimização sem solução viável, representada pelo aumento da função objetivo.

5.4 Comparação entre as soluções manual, exata e heurística

A tabela 5.7 exibe uma síntese do percentual de utilização das salas para cada dia da semana, e seu respectivo percentual de desuso, gerado a partir da solução atual, comumente utilizada pela instituição de ensino, e o modelo heurístico sugerido, neste trabalho. Os resultados podem ser obtidos através da seguinte expressão matemática representada pelos termos 1 e 2:

A instituição estudada funciona de segunda à sábado, nos três turnos existentes. Sendo assim, considera-se que cada sala da unidade possui um total de 15 horas diárias disponíveis para que ocorram os encontros entre professores e alunos da disciplina oferecida. Logo obtêm-se:

1) Número de salas disponíveis × Total de horas disponíveis para uma dada sala, em um determinado dia da semana;

2) Somatório das horas de todas as salas quando as aulas estão sendo lecionadas, em determinado dia da semana;

O resultado é obtido através da razão do 2ª pelo 1ª termo.

Sendo,

TU - Taxa de Utilização das salas nos respectivos dias da semana;

TD - Taxa de Desuso das salas também em seus respectivos dias da semana.

As Figuras 5.6 e 5.7 referem-se, respectivamente, às taxas de utilização (TU) e de desuso (TD) semanal das salas.

Dia / Turno	Solução Atual	Solução Atual	Solução Proposta	Solução Proposta
	TU	TD	TU	TD
Segunda-Feira	44,33%	55,67%	47%	53%
Terça-Feira	53,66%	46,34%	60%	40%
Quarta-Feira	54%	46%	60,3%	39,7%
Quinta-Feira	56,66%	43,34%	58,3%	41,7%
Sexta-Feira	34,66%	65,34%	32%	68%
Sábado	9,3%	90,7%	5%	95%

Tabela 5.7: Taxa de Utilização Semanal das Salas na EEIMVR. (Fonte Própria, 2017)

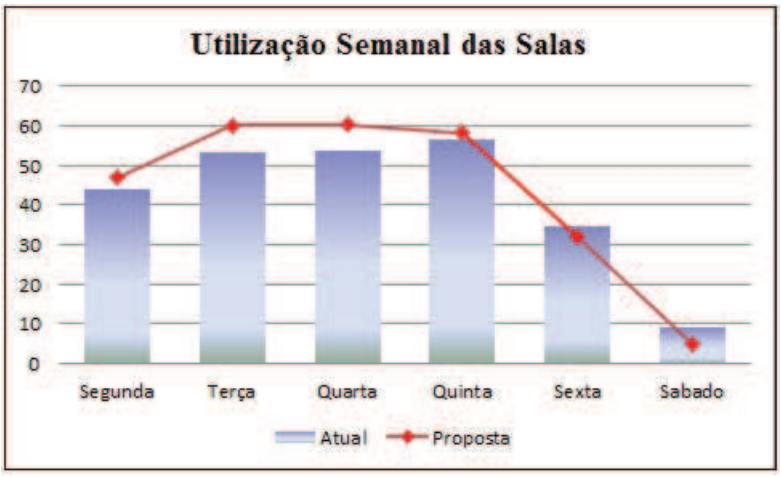


Figura 5.6: Taxa de Utilização Semanal das Salas. (Fonte Própria, 2017)

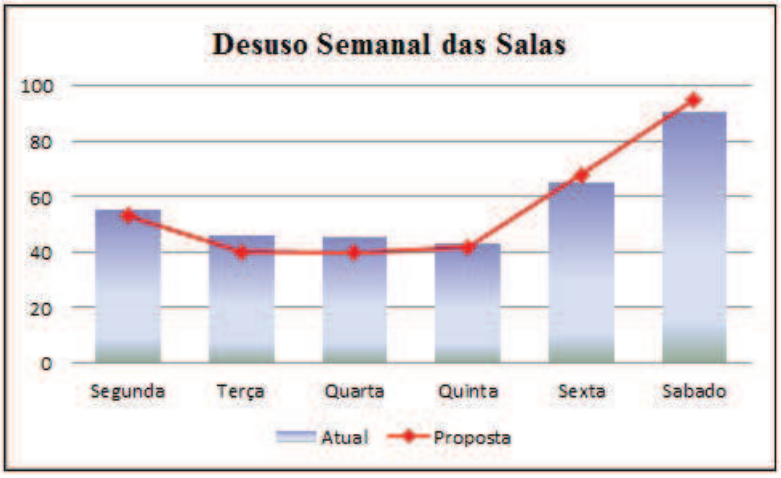


Figura 5.7: Taxa de Desuso Semanal das Salas. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5C	N6A	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	-	13%	-	0%	-	0%	25%	-	-	-	-	-	0%
9:00 - 10:00	-	13%	-	0%	-	0%	25%	100%	-	-	-	-	0%
10:00 - 11:00	0%	13%	-	0%	0%	0%	-	100%	14%	-	-	-	-
11:00 - 12:00	0%	13%	-	0%	0%	0%	-	100%	14%	-	-	-	-
12:00 - 13:00	-	-	-	0%	-	-	-	100%	-	-	-	-	-
13:00 - 14:00	-	-	-	0%	-	-	-	-	-	0%	-	-	-
14:00 - 15:00	0%	13%	9%	0%	20%	0%	-	17%	26%	0%	0%	35%	8%
15:00 - 16:00	0%	13%	9%	0%	20%	0%	-	17%	26%	-	0%	35%	8%
16:00 - 17:00	9%	13%	9%	0%	0%	0%	-	17%	26%	0%	0%	35%	8%
17:00 - 18:00	9%	13%	9%	0%	0%	0%	-	17%	24%	0%	0%	35%	8%
18:00 - 19:00	-	-	20%	31%	60%	22%	-	-	24%	25%	10%	25%	15%
19:00 - 20:00	-	-	-	31%	60%	22%	12%	-	24%	25%	10%	25%	15%
20:00 - 21:00	-	-	-	31%	60%	22%	12%	-	24%	50%	10%	-	15%
21:00 - 22:00	-	-	-	31%	-	22%	12%	-	24%	50%	-	-	15%

Tabela 5.8: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Segunda-Feira. (Fonte Própria, 2017)

Em contra partida, a tabela 5.8 traz a taxa de ociosidade de algumas das salas disponíveis, nos horários de segunda-feira, originalmente utilizada pela EEIMVR (solução manual). É possível verificar as respectivas tabelas, com a taxa de ociosidade de todas as salas disponíveis, para todos os dias da semana, no ApêndiceA deste trabalho. Desta forma definida, cada sala possui 15 horas diárias disponíveis para realização de encontros, mas, como apresentado na tabela de ociosidade, tomando a sala B5 como exemplo, em uma segunda-feira, nos horários de ocorrência dessas aulas, apenas 87% da capacidade da sala é ocupada por alunos, os outros 13% da capacidade restante, se referem à essa ociosidade, identificando possíveis oportunidades de alocações mais eficientes, proposta deste trabalho. A partir desta análise foi possível registrar a ocupação das salas por turno, dia da semana e salas disponíveis, o que permitiu aumentar a eficiência dos responsáveis por essa atividade, com maior aproveitamento do conjunto de salas de aula.

A partir das soluções geradas (manual, exata e computacional) foram montados os quadros de horários 5.9, 5.10 e 5.11 abaixo, correspondentes às disciplinas/horários/salas. Vale ressaltar que, nesses quadros de horários, para os modelos desenvolvidos, as linhas correspondem aos horários e dias da semana em que ocorrem às referidas aulas, as colunas correspondem aos locais (salas) onde ocorrem estes encontros, e o elemento dessa matriz revela a aula que está sendo lecionada

nestas informações exibidas.

Em posse das soluções fornecidas por cada modelo foi possível elaborar e apresentar os quadros de horários 5.9, 5.10 e 5.11, e assim, realizar um estudo sobre cada modelo. Porém, somente os dados apresentados pelas tabelas 5.3 e 5.5 permitem fazer um estudo sobre a viabilidade de cada modelo, e a partir daí, identificar aquele que fornece a melhor solução para o problema apresentado, e dessa maneira, estabelecer a alocação mais adequada.

Devido o tamanho do problema analisado, optou-se por exibir somente parte das alocações feitas a partir da forma manual e das soluções geradas, no modelo exato e heurístico. As tabelas 5.9, 5.10 e 5.11 permitem verificar parte dessa alocação, dentro de um conjunto de 20 (vinte) salas consideradas, no estudo deste trabalho. Na Seção 4.1 pode-se verificar, de forma mais detalhada, os dados pertinentes à estas alocações, tais como: horários considerados, número de salas e aulas lecionadas.

Como forma de direcionar a análise das alocações, considera-se como exemplo de estudo, a alocação dos encontros ocorridos em uma segunda-feira, ao longo dos três turnos de funcionamento da instituição, no 2º Semestre de 2014. Utiliza-se como exemplo nas tabelas 5.9, 5.10 e 5.11 a alocação das aulas na sala 01 (B3), ocorridas entre os horários de 07:00 am às 22:00 pm, de uma segunda-feira. Em ambas alocações, exata e heurística, originalmente as tabelas 5.10 e 5.11, estavam codificadas através de um conjunto de números, definidos em um mapeamento, que indicavam cada aula a ser lecionada na sala, dia e horário específico, mas, para que fosse possível identificar essas aulas, estas foram decifradas e apresentadas por meio de suas denominações.

A tabela 5.12 traz uma síntese dos resultados obtidos nos experimentos realizados, a partir daí, é possível fazer uma breve comparação desses experimentos, o que se conclui que: Como observado, o algoritmo aproximado apesar de fornecer uma "boa" solução, identificado pelo valor da função objetivo, o método exato é capaz de oferecer uma solução ainda melhor, visto que, sua solução apresentada

é a ótima, além de seu tempo de processamento computacional ser muito menor que o da heurística.

H-S	01	02	03	...	15	16	...	20
1	-	-	-	...	-	-	...	-
2	Mon.Calc II	PO I	Sist.Transp	...	C.Q.I	Fab.Usin.	...	-
3	Mon.Calc II	PO I	Sist.Transp	...	C.Q.I	Fab.Usin.	...	-
4	Mon.Calc II	PO I	Sist.Transp	...	C.Q.I	-	...	-
5	Maq.Equ.Ag	PO I	Sist.Transp	...	C.Q.I	-	...	-
6	Maq.Equ.Ag	PO I	Sist.Transp	...	Mon.Mat.	-	...	Mon.GA
7	-	-	Baja	...	Fis.I	-	...	Mon.GA
8	-	-	Baja	...	Fis.I	Anal.Cad.Prod.	...	Mon.Eng.Econ.
9	Emb.	PO I	Fis.Quim.	...	-	Anal.Cad.Prod.	...	Mon.Eng.Econ.
10	Emb.	PO I	Fis.Quim.	...	Fis.I	Proj.Agr.	...	Mon.Eng.Econ.
11	Fisio.	PO I	Vibr.	...	Fis.I	Proj.Agr.	...	Mon.Eng.Econ.
12	Fisio.	PO I	Vibr.	...	Mec.Geral	Gest.Amb.	...	-
13	Top. Mat.	Micro.Estr.II	Int.Eng.	...	Mec.Geral	Gest.Amb.	...	-
14	Top. Mat.	Micro.Estr.II	-	...	-	Gest.Amb.	...	-
15	Top. Mat.	Micro.Estr.II	Mon.Fis.I	...	-	Gest.Amb.	...	-
16	-	-	Mon.Fis.I	...	-	Gest.Amb.	...	-
...	...	...	...	...	...	...	...	...
56	Mec.Flu I	Ens.Mec	-	...	Cal.Vet.	Proc.Mat.	...	Adm.Org.
57	Mec.Flu I	Ens.Mec	Gest.Amb.	...	Cal.Vet.	Proc.Mat.	...	Adm.Org.
58	Mec.Flu I	Ens.Mec	Gest.Amb.	...	Fis.II	Bio.Ger.	...	Adm.Org.
59	Elem.Maq.	Mon.Cal.Vet.	Gest.Amb.	...	Fis.II	Bio.Ger.	...	Adm.Org.
60	Elem.Maq.	Mon.Cal.Vet.	Gest.Amb.	...	Proc.Ind	Sel.Mat.	...	Proc.Desold
61	Plan.Estr.	Top.Metal	Ger.Contr.	...	Proc.Ind	Sel.Mat.	...	Proc.Desold
62	Plan.Estr.	Top.Metal	Ger.Contr.	...	Proc.Ind	Sel.Mat.	...	Proc.Desold
63	Plan.Estr.	Top.Metal	Ger.Contr.	...	-	Sel.Mat.	...	-
64	Plan.Estr.	-	Ger.Contr.	...	-	Sel.Mat.	...	-
65	-	-	-	...	-	-	...	-
66	Quim.Metal	Adm.Prod.	-	...	Int.Met.Num.	Mec.Flu.I	...	-
67	Quim.Metal	Adm.Prod.	-	...	Int.Met.Num.	Mec.Flu.I	...	-
68	Quim.Metal	Adm.Prod.	-	...	-	Mec.Flu.I	...	-
69	Quim.Metal	Adm.Prod.	Int.Eng.Ag.	...	Mec.Flu.I	Mon.Des.	...	-
70	Int.Eng.	Adm.Prod.	-	...	Mec.Flu.I	Mon.Des.	...	-
...	...	...	...	...	...	...	...	...
90	C.Q.II	Mon.EDO	Met.Fis.	...	-	-	...	MBA
91	C.Q.II	-	-	...	-	-	...	MBA
92	-	-	-	...	-	-	...	MBA
93	-	-	-	...	-	-	...	-
94	-	-	-	...	-	-	...	-
95	-	-	-	...	-	-	...	-
96	-	-	-	...	-	-	...	-

Tabela 5.9: Alocação Manual. (Fonte Própria, 2017)

H-S	01	02	03	...	15	16	...	20
1	-	-	-	...	-	-	...	-
2	-	-	-	...	G.A	-	...	Seg.Ind
3	-	-	-	...	G.A	ResMat.I	...	Seg.Ind
4	-	Inst. Ind.I	Máq.Eq.Agr.	...	-	-	...	ResMat.I
5	-	Inst. Ind.I	Máq.Eq.Agr.	...	-	-	...	ResMat.I
6	-	-	-	...	-	-	...	-
7	-	-	-	...	Cal.II	-	...	-
8	Proc. Metal Fab.	Estat.I	Anl.Cad.Prod.	...	Cal.II	Anal.Merc.	...	Adm.Fin.
9	Proc. Metal Fab.	Estat.I	Anl.Cad.Prod	...	Cal.II	Anal.Merc.	...	Adm.Fin.
10	Física I	Estat.I	Proc.Unit.I	...	Fís.I	Proc.Unit.II	...	Adm.Fin.
11	Física I	MKT	Proc.Unit.I	...	Fís.I	Proc.Unit.II	...	Adm.Fin.
12	Arranjo Físico	MKT.	C.Q.II	...	-	-	...	Energ.Bio
13	Arranjo Físico	MKT.	C.Q.II	...	-	-	...	Energ.Bio
14	Arranjo Físico	MKT.	C.Q.II	...	-	-	...	Energ.Bio
15	Arranjo Físico	MKT.	C.Q.II	...	-	-	...	-
16	-	-	-	...	-	-	...	-
...	...	...	...	...	...	...	...	...
56	ETM	Gest.Amb.	-	...	Term.Din.	Planj.Est.Agro	...	Quim.Ger.
57	ETM	Gest.Amb.	Proc.Fus.Vaz.	...	Term.Din.	Planj.Est.Agro	...	Quim.Ger.
58	ETM	Gest.Amb.	Proc.Fus.Vaz.	...	Fís.II	-	...	-
59	ETM	Gest.Amb.	Proc.Fus.Vaz.	...	Fís.II	-	...	-
60	-	Planj.Est.Ind.	-	...	-	-	...	-
61	-	Planj.Est.Ind.	-	...	-	-	...	-
62	-	Planj.Est.Ind.	-	...	-	-	...	-
63	-	Planj.Est.Ind.	-	...	-	-	...	-
64	-	-	-	...	-	-	...	-
65	-	-	-	...	-	-	...	-
66	-	Adm.Prod.	-	...	Mét.Num.	-	...	Mec.Geral
67	-	Adm.Prod.	-	...	Mét.Num.	-	...	Mec.Geral
68	-	Adm.Prod.	-	...	Mét.Num.	Trans.Cal.I	...	-
69	-	Adm.Prod.	Int.Eng.	...	Mét.Num.	Trans.Cal.I	...	-
70	-	-	-	...	-	-	...	-
...	...	...	...	...	...	...	...	...
90	-	-	-	...	-	-	...	-
91	-	-	-	...	-	-	...	-
92	-	-	-	...	-	-	...	-
93	-	-	-	...	-	-	...	-
94	-	-	-	...	-	-	...	-
95	-	-	-	...	-	-	...	-
96	-	-	-	...	-	-	...	-

Tabela 5.10: Alocação Exata. (Fonte Própria, 2017)

H-S	01	02	03	...	15	16	...	20
1	-	-	-	...	-	-	...	-
2	-	-	-	...	G.A	-	...	-
3	-	-	-	...	G.A	ResMat.I	...	-
4	-	Des.I	-	...	-	-	...	ResMat.I
5	-	Des.I	-	...	-	-	...	ResMat.I
6	-	-	-	...	-	-	...	-
7	-	-	-	...	Cal.II	-	...	-
8	Estat.I	Anl.Cad.Prod	POII	...	Cal.II	Proc.Met.Fab.	...	Cont.Qual.I
9	Estat.I	Anl.Cad.Prod	POII	...	Cal.II	Proc.Met.Fab.	...	Cont.Qual.I
10	Estat.I	Proc.Unit.I	POII	...	Calc.II	Proc.Unit.II	...	Cont.Qual.I
11	MKT	Proc.Unit.I	POII	...	Calc.II	Proc.Unit.II	...	Cont.Qual.I
12	MKT	C.Q.II	Arranjo Físico	...	-	ResMat.II	...	Mec.Geral
13	MKT	C.Q.II	Arranjo Físico	...	-	ResMat.II	...	Mec.Geral
14	MKT	C.Q.II	Arranjo Físico	...	-	-	...	-
15	MKT	C.Q.II	Arranjo Físico	...	-	-	...	-
16	-	-	-	...	-	-	...	-
...	...	...	...	...	...	...	...	...
56	ETM	Gest.Amb.	ADM	...	Alg.Lin.	Trans.Cal I	...	Fís.Exp.II
57	ETM	Gest.Amb.	ADM	...	Alg.Lin.	Trans.Cal I	...	Fís.Exp.II
58	ETM	Gest.Amb.	ADM	...	Fís.II	Quim.Geral	...	-
59	ETM	Gest.Amb.	ADM	...	Fís.II	Quim.Geral	...	-
60	-	Planj.Est.Ind.	-	...	-	Cont.Qual.I	...	Fund.Econ.
61	-	Planj.Est.Ind.	-	...	-	Cont.Qual.I	...	Fund.Econ.
62	-	Planj.Est.Ind.	-	...	-	Cont.Qual.I	...	Fund.Econ.
63	-	Planj.Est.Ind.	-	...	-	Cont.Qual.I	...	-
64	-	-	-	...	-	-	...	-
65	-	-	-	...	-	-	...	-
66	-	Adm.Prod.	-	...	Mét.Num.	-	...	-
67	-	Adm.Prod.	-	...	Mét.Num.	-	...	-
68	-	Adm.Prod.	-	...	EDP	-	...	-
69	-	Adm.Prod.	Int.Eng.	...	EDP	-	...	-
70	-	-	-	...	-	-	...	-
...	...	...	...	...	...	...	...	...
90	-	-	-	...	-	-	...	-
91	-	-	-	...	-	-	...	-
92	-	-	-	...	-	-	...	-
93	-	-	-	...	-	-	...	-
94	-	-	-	...	-	-	...	-
95	-	-	-	...	-	-	...	-
96	-	-	-	...	-	-	...	-

Tabela 5.11: Alocação Heurística. (Fonte Própria, 2017)

Técnica / Problema	Função Objetivo	Tempo Processamento	Número Iterações
Manual	12000	02 meses	-
Exata	3625	2,37 seg.	564
Heurística	3659	20,73 seg.	4310

Tabela 5.12: Resultados das Soluções Manual, Exata e Heurística. (Fonte Própria, 2017)

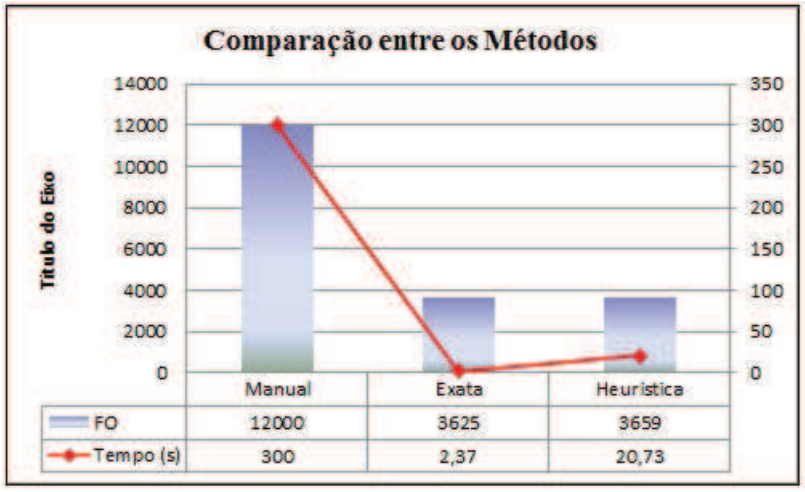


Figura 5.8: Comparativo entre os Métodos de Resolução. (Fonte Própria, 2017)

Distância Referencial da Melhor Solução	0,9379%
Distância Referencial do Valor Médio das Soluções	1,6699%
Distância Referencial do Melhor Tempo	775%
Distância Referencial do Valor Médio dos Tempos	780%

Tabela 5.13: Comparativo entre o método Exato e Heurístico. (Fonte Própria, 2017)

Uma outra maneira de avaliar cada método de forma comparativa, é por meio do cálculo da distância referencial entre as soluções, calculada a partir da seguinte Equação (5.1):

$$Distancia = (FOheuristica - FOexato)/FOexato \tag{5.1}$$

Com este procedimento é possível analisar a distância, em porcentagem, de uma solução exata para a heurística. E assim, a tabela 5.13 foi montada a partir dos

resultados obtidos em termos de solução e tempo, entre os dois métodos estudados. Em resumo, a equação foi calculada para a melhor solução encontrada da heurística e pelo valor médio das soluções, e foram exibidos na tabela 5.13.

O estudo da tabela 5.13 permite concluir que a distância referencial entre as soluções heurísticas e as exatas é menor que 1%, o que mostra que ambas as soluções possuem uma diferença pouco significativa, quando comparadas. No entanto, a análise do tempo de processamento dos dois métodos não retrata o que foi descrito acima, visto que, a distância entre eles é de aproximadamente 775%, identificando, dessa maneira, o método exato como a opção mais adequada em detrimento ao aproximando.

Capítulo 6

Conclusões e Considerações Finais

6.1 Conclusões

Foi realizado um estudo sobre os métodos de otimização em problemas de Otimização Combinatória, e aplicado dois deles, um método exato e um método heurístico, para o problema real da instituição EEIMVR.

Os resultados mostraram que é possível obter soluções viáveis e de qualidade, para o problema estudado, em menos de 30 segundos. O que é aceitável para os padrões de exigência estabelecidos.

Quando comparados entre si, o método exato se mostrou a técnica mais eficiente no caso do problema se mostrar viável, ou seja, quando o uso de todas as salas for permitido. Diferentemente do trabalho apresentado por [13], no qual, mesmo ambos os problemas possuindo dimensões semelhantes, este provou por meio da metodologia de resolução evolutiva, AGc, aplicada à instâncias fictícias devido a complexidade no uso de instâncias reais utilizado no método exato, que em todos os cenários desenvolvidos para este experimento, foi o método evolutivo aquele responsável por fornecer soluções de maior qualidade quando comparado ao modelo matemático.

Porém, no caso em que o uso de todas as salas não for permitido, o modelo

matemático não consegue fornecer uma solução apropriada, e neste caso, o modelo heurístico ainda é a melhor opção.

Como apresentado, o experimento foi baseado na remoção de salas e na alteração dos geradores de números aleatórios, representados através da atribuição de valor unitário à capacidade das salas mencionadas. Sendo assim, a partir dos valores atribuídos às penalizações na função objetivo, é possível garantir que, a função objetivo tentará alocar os encontros em salas com capacidades compatíveis aos números de alunos matriculados na referida disciplina, dada a penalização atribuída à este fator. No entanto, os experimentos revelaram que mesmo após a remoção de salas, ainda existem algumas que são utilizadas mesmo com suas capacidades reduzidas à 01 (um). Então, é possível concluir que, a proibição do uso de determinada sala, mostrou que esses recursos, neste caso, apresentam saturações, inviabilizando o problema, visto que, mesmo após a redução de sua capacidade, esta ainda se apresenta como possível local de ocorrência de aulas.

Como forma de complementar o estudo sobre restrição do uso de salas, foi realizado outro teste, agora com base na inclusão de salas, visto que, o primeiro identificou o modelo exato como sendo incapaz de resolver o problema com a remoção de algumas salas específicas, dada a quantidade insuficiente de salas, para a total alocação. Para o teste de inclusão, foram desenvolvidas 03 (três) novas instâncias, além daquelas já apresentadas inicialmente, com inclusão de salas com capacidades de 40, 60 e 80. O experimento permitiu identificar a sala com capacidade 40, como aquela que seria escolhida, caso fosse possível incluir uma dessas, visto que, esta traria melhor resultado na função objetivo.

Além disso, o resultado apresentado na tabela 5.13 mostrou que a distância entre as melhores soluções da heurística e da exata é menor que 1%, o que indica que ambas soluções possuem uma diferença muito pequena, quando comparadas. Porém, ao analisar o resultado apresentado na tabela de tempo de processamento, a distância entre eles é de aproximadamente 775%, identificando, dessa maneira, o método exato como a opção mais adequada.

Finalmente, o experimento permitiu identificar o gargalo do problema analisado. Sendo este apresentado às quintas-feiras, no horário de 14:00 às 16:00, visto que, nestes horários, todas as salas são requeridas. Adicionalmente, a sala D30A é aquela que possui maior relevância, já que, sua remoção apresentou maior aumento da função objetivo.

6.2 Trabalhos Futuros

A partir da aferição do tempo de processamento das abordagens propostas, foi possível concluir que o modelo exato, neste caso, forneceu a solução ótima e foi mais rápido que o *Simulated Annealing*. Baseado nisto, sugere-se como trabalho futuro melhorar o desempenho de execução do método heurístico, fazendo com que este chegue a uma solução mais rapidamente que o método exato. Para isso, uma possível mudança nas estruturas de dados utilizadas podem ser objetos de estudos futuros.

Além disso, outras estratégias de busca podem ser utilizadas, tais como, *Tabu Search* e os Algoritmos Genéticos, apesar deste último, segundo a comunidade acadêmica, ser pouco utilizado na resolução de problemas como o abordado neste trabalho. No entanto, isso não impede que se faça uso destes métodos de resolução citados, de forma eficiente, em outros casos, que não este estudado. Algoritmos evolutivos, como os genéticos, por exemplo, possuem conhecida eficiência na resolução de problemas de otimização combinatória complexos, que envolvam muitas variáveis e um espaço de soluções de dimensão elevada [27]. Também, é possível o emprego de outros algoritmos de geração de solução inicial, sejam eles aleatórios, gulosos ou híbridos como apresentados em [28] e [29].

Uma outra sugestão de trabalho futuro é a inclusão da análise dos módulos nas abordagens propostas. A análise dos módulos, nada mais é, do que a maneira como os diferentes currículos são distribuídos pelas turmas disponíveis. Ou seja, deve-se definir a quantidade de alunos de cada curso de graduação presente em

cada uma das turmas.

Para que fosse possível a interpretação dos resultados fornecidos pelos métodos de resolução empregados, foi necessário um mapeamento dos dados (capacidade, demanda, disciplina, horário e solução) utilizados no processamento de cada caso. No entanto, apesar do uso deste artifício como um fator simplificador, a manipulação dos diferentes dados, bem como, do algoritmo principal, exige um certo nível de conhecimento à respeito de modelagem computacional, uma condição necessária, mas, não condizente com a realidade dos profissionais que normalmente se deparam com este problema. Diante disso, recomenda-se a criação de um mecanismo que seja capaz de atuar na interface entre os *output* da máquina e o usuário final, ou seja, que possa realizar essa "comunicação" entre os métodos e os responsáveis por essa atividade.

Referências

[1] Code::Blocks. Disponível em: <http://www.codeblocks.org. Acessado em 11/2017.

[2] Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. Disponível em: <http://www.dcc.ufmg.br/dcc/?q=en/node/825>. Acessado em 09/2016.

[3] Escola de Engenharia Industrial Metalúrgica de Volta Redonda. Disponível em: <http://www.engenhariavr.uff.br. Acessado em 07/2017.

[4] ILOG S.A., CPLEX 12.6 user's manual, 2012.

[5] Ahuja, R. K., Magnanti, T. L., Orlin, J. *Network Flows:Theory, Algorithms, and Applications*, 1 ed. Prentice Hall, 1993.

[6] Aiex, R., Resende, M., Ribeiro, C. Probability distribution of solution time in GRASP: An experimentak investigations. *Journal of Heuristics 8* (2002), 343–373.

[7] Alfaro, H. M., Terán, G. F. **Solving the Classroom Assignment Problem with *Simutated Annealing.*** *Proceedings of Center Artificial Intelligence – ITESM, Monterrey, México.* (1998).

[8] Barr, R. S., Helgason, R. V., Kennington, J. L. ***Interfaces in Computer Science and Operations Research**: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*, 1 ed. Springer Science and Business Media, Texas, 1997.

[9] Bazarra, M. S., Jarvis, J. J. ***Linear Programming and Network Flows.***, 2 ed. Wiley Interscience, School of Industrial and Systems Engineering and Georgia Institute of Technology, Atlanta, Georgia, 1977.

[10] Bazarra, M. S., Sherali, H. D., Shetty, C. M. ***Nonlinear Programming**: Theory and Algorithms.*, 3 ed. Wiley Interscience, School of Industrial and Systems Engineering, Atlanta, Georgia, 2006.

[11] Celes, W., Cerqueira, R., Rangel, J. L. ***Introdução a Estrutura de Dados:*** *Com técnicas de programação em C*, 2 ed. Editora Campus Grupo Elsevier, Rio de Janeiro, RJ, 2004.

[12] Chaves, A. A. **Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória**. *PhD thesis, Instituto Nacional de Pesquisas Espaciais, INPE, São José dos Campos, São Paulo, Brasil* (2009).

[13] Cirino, R. B. Z., Santos, M. O., Delbem, A. C. B. Aplicação da meta-heurística AGc para o problema de alocação de aulas à salas. Em *Proceedings of Simpósio Brasileiro de Pesquisa Operacional, Porto de Galinhas, PE: SBPO.* (2015), p. 1–10.

[14] Cormen, T. H., Leiserson, C. E., Rivest, R. R. ***Introduction to Algorithms.***, 4 ed. McGraw-Hill Book Company, Massachusetts Institute of Technology - MIT, Cambrigde, 1990.

[15] Cunha, A. G., Takahashi, R., Antunes, C. H. ***Manual de Computação Evolutiva e Meta-heurística***, 1 ed. Editora da Universidade Federal de Minas Gerais, Universidade Federal de Minas Gerais, Minas Gerais/Brasil, 2012.

[16] de Almeida, M. H. R. Heurísticas BRKGA para o problema da cadeia de caracteres mais próxima alocação de salas em cursos universitários: Um estudo de caso. Dissertação de Mestrado, CEFET - MG, Brasil, 2013.

[17] Gendreau, M., Potvin, J.-Y. ***Handbook of Metaheuristics.***, 2 ed. Springer Science, Montreal, Canadá, 2010.

[18] Glover, F., Laguna, M. ***Tabu Search***, 7 ed. Kluwer Academic Publishers, University of Colorado at Boulder, Boston/Dordrecht/London, 1997.

[19] Hillier, F. S., Lieberman, G. J. ***Introduction to Operations Research***, 9 ed. McGrawHill, Stanford University, 2010.

[20] Kirkpatrick, S., Gellat, D. C., Vecchi, M. P. Optimization by simulated annealing. *Science 220* (1983), 671–680.

[21] Kripka, R. M. L., Kripka, M., da Silva, M. C. **Formulação para o Problema de Alocação de Salas de Aula com Minimização de Deslocamentos**. *Proceedings of Simpósio Brasileiro de Pesquisa Operacional, Ubatuba, SP: SBPO.* (2011).

[22] Kroon, L. G., Salomon, M., Wassenhove, L. N. V. Exact and approximation algorithms for the operational fixed interval scheduling problem. *Journal of Operational Research* (1993).

[23] Lopes, H. S., de Abreu Rodrigues, L. C., Steiner, M. T. A. ***Meta-Heurística em Pesquisa Operacional***. Omnipax Editora Ltda, Curitiba, PR, 2013.

[24] Neves, T. A. **Resolução do problema de Roteamento de Veículos com Frota Heterogênea e Janelas de Tempo**. *Monografia submetida ao Departamento de Ciência da Computação da UFOP* (2004).

[25] Prado, A. S. Problema de alocação de salas em cursos universitários: Um estudo de caso. Dissertação de Mestrado, CEFET - MG, Brasil, 2014.

[26] Queiroga, E. V., Júnior, T. L. B., Cabral, L. A. F., da Costa, L. C. A., Subramanian, A. **Problema de Alocação de Aulas**: O caso da central de aulas da UFPB. *Proceedings of Simpósio Brasileiro de Pesquisa Operacional, Porto de Galinhas, PE: SBPO.* (2015).

[27] Redusino, A. C. E. Aplicações de Algoritmos Genéticos. *Journal of Heuristics* (2016), 1–6.

[28] Resende, M., Ribeiro, C. GREEDY Randomized Adaptive Search Procedures: Advances and Extensions. *Journal of Heuristics* (2017), 1–7.

[29] Rizzi, M. M., Chaves, A. A., Senne, E. L. F., Lorena, L. A. N. Metaheurística Híbrida Aplicada ao Problema de Job Shop. *Journal of Heuristics* (2015), 1–9.

[30] Schaerf, A. A survey of automated timetabling. *Artificial Intelligence Review 13* (1999), 87–127.

[31] Segatto, E. A., Boeres, M. C. S., Kampke, E. H. **Um Algoritmo GRASP com Cadeia de Kempe Aplicado ao Problema de Tabela-Horário para Universidades.** *Proceedings of Simpósio Brasileiro de Pesquisa Operacional, Porto de Galinhas, PE: SBPO.* (2015).

[32] Souza, M. J. F. **Programação de horários em escolas**: Uma aproximação por meta-heurística. *Tese (Doutorado), Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil* (2000).

[33] Souza, M. J. F., Martins, A. X., Araújo, C. R. Experiências com simulated annealing e busca tabu na resolução do problema de alocação de

salas. Em *Proceedings of Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro, RJ: SBPO.* (2002).

[34] Subramanian, A., Medeiros, J. M. F., dos Anjos Formiga, L., Souza, M. J. F. **Aplicação da Meta-heurística Busca Tabu ao Problema de Alocação de Aulas a Salas em uma Instituição Universitária.** *In: Proceedings of Revista Produção Online: ABEPRO. v. 11* (2011), 54–75.

[35] Talbi, E.-G. ***Metaheuristics**: From Design to Implementation*, 1 ed. Wiley Interscience, University of Lille and France Research Centre - INRIA, França, 2009.

APÊNDICE A

Segue abaixo, algumas tabelas com informações completas sobre as alocações estudadas nesta dissertação:

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	-	13%	-	-	0%	-	0%	25%	-	-	-	0%	-	-	-	0%
9:00 - 10:00	-	13%	-	-	0%	-	0%	25%	100%	-	-	0%	-	-	-	0%
10:00 - 11:00	0%	13%	-	-	0%	0%	0%	-	100%	14%	-	0%	-	-	-	-
11:00 - 12:00	0%	13%	-	-	0%	0%	0%	-	100%	14%	-	0%	-	-	-	-
12:00 - 13:00	-	-	-	-	0%	-	-	-	100%	-	-	-	-	-	-	-
13:00 - 14:00	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-	-	-
14:00 - 15:00	0%	13%	9%	-	0%	20%	0%	-	17%	26%	0%	0%	0%	35%	0%	8%
15:00 - 16:00	0%	13%	9%	-	0%	20%	0%	-	17%	26%	-	0%	0%	35%	0%	8%
16:00 - 17:00	9%	13%	9%	-	0%	0%	0%	-	17%	26%	0%	0%	0%	35%	0%	8%
17:00 - 18:00	9%	13%	9%	-	0%	0%	0%	-	17%	24%	0%	0%	0%	35%	0%	8%
18:00 - 19:00	-	-	20%	-	31%	60%	22%	-	-	24%	25%	0%	10%	25%	0%	15%
19:00 - 20:00	-	-	-	-	31%	60%	22%	12%	-	24%	25%	0%	10%	25%	0%	15%
20:00 - 21:00	-	-	-	-	31%	60%	22%	12%	-	24%	50%	0%	10%	-	0%	15%
21:00 - 22:00	-	-	-	-	31%	-	22%	12%	-	24%	50%	0%	-	-	0%	15%

Tabela A.1: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Segunda-Feira. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	0%	15%	-	-	-	-	0%	25%	-	-	0%	-	-	0%	10%	0%
9:00 - 10:00	0%	15%	-	-	0%	-	0%	25%	83%	-	0%	-	-	0%	10%	0%
10:00 - 11:00	0%	15%	27%	-	-	0%	-	35%	83%	-	-	0%	-	63%	10%	13%
11:00 - 12:00	0%	15%	27%	-	0%	0%	-	35%	83%	-	-	0%	-	63%	10%	13%
12:00 - 13:00	-	62%	-	-	0%	-	-	35%	-	-	-	-	-	-	-	-
13:00 - 14:00	0%	62%	-	-	0%	-	0%	-	15%	9%	0%	-	-	-	-	-
14:00 - 15:00	0%	62%	82%	-	0%	0%	0%	25%	15%	9%	0%	0%	-	25%	0%	8%
15:00 - 16:00	0%	-	82%	20%	0%	0%	0%	25%	15%	9%	-	0%	-	25%	0%	8%
16:00 - 17:00	0%	6%	82%	20%	0%	0%	-	25%	15%	27%	0%	0%	-	25%	0%	8%
17:00 - 18:00	-	6%	-	20%	0%	0%	-	-	-	27%	0%	0%	-	25%	0%	-
18:00 - 19:00	0%	25%	27%	-	19%	0%	0%	6%	-	29%	25%	0%	10%	25%	10%	0%
19:00 - 20:00	0%	25%	27%	-	19%	0%	0%	6%	67%	29%	25%	0%	10%	25%	10%	0%
20:00 - 21:00	0%	25%	-	-	19%	0%	0%	6%	67%	29%	50%	0%	10%	-	10%	0%
21:00 - 22:00	0%	25%	-	-	19%	-	0%	6%	67%	29%	50%	-	10%	-	10%	0%

Tabela A.2: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Terça-Feira. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	0%	-	9%	-	0%	-	-	-	-	-	0%	-	0%	0%	-	20%
9:00 - 10:00	0%	4%	9%	-	0%	-	-	-	-	-	0%	-	0%	0%	-	20%
10:00 - 11:00	-	4%	-	-	0%	0%	-	-	-	-	0%	-	0%	-	0%	8%
11:00 - 12:00	-	4%	-	-	0%	0%	-	-	-	-	0%	-	0%	-	0%	8%
12:00 - 13:00	-	-	-	-	0%	0%	-	-	-	-	-	-	-	-	-	-
13:00 - 14:00	-	25%	-	-	-	-	-	-	-	-	-	-	-	-	20%	-
14:00 - 15:00	9%	25%	9%	-	0%	0%	-	-	-	-	0%	-	0%	-	20%	15%
15:00 - 16:00	9%	25%	9%	-	0%	0%	-	-	-	-	0%	-	0%	-	20%	15%
16:00 - 17:00	0%	-	9%	-	0%	20%	-	-	-	-	0%	20%	0%	25%	-	-
17:00 - 18:00	0%	15%	9%	-	0%	20%	-	-	-	-	0%	20%	0%	25%	-	-
18:00 - 19:00	0%	15%	27%	0%	25%	20%	-	-	-	-	80%	20%	0%	80%	20%	8%
19:00 - 20:00	0%	15%	27%	0%	25%	20%	-	-	-	-	80%	20%	-	80%	20%	8%
20:00 - 21:00	0%	15%	-	0%	25%	20%	-	-	-	-	80%	20%	-	80%	20%	8%
21:00 - 22:00	0%	15%	-	0%	25%	-	-	-	-	-	-	-	-	-	-	8%

Tabela A.3: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Quarta-Feira. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	0%	-	9%	-	0%	0%	-	0%	-	14%	38%	0%	0%	-	10%	-
9:00 - 10:00	0%	17%	9%	-	0%	0%	-	0%	-	14%	38%	0%	0%	-	10%	-
10:00 - 11:00	0%	17%	27%	-	0%	0%	0%	0%	20%	14%	35%	0%	0%	0%	10%	-
11:00 - 12:00	0%	17%	27%	-	0%	0%	0%	0%	20%	14%	35%	0%	0%	0%	10%	-
12:00 - 13:00	-	-	27%	-	-	-	-	0%	-	-	-	-	0%	-	-	-
13:00 - 14:00	0%	6%	-	60%	-	-	-	0%	-	-	40%	4%	-	20%	-	-
14:00 - 15:00	0%	6%	5%	60%	0%	0%	0%	0%	5%	14%	40%	4%	-	20%	0%	8%
15:00 - 16:00	0%	6%	5%	60%	0%	0%	0%	0%	5%	14%	40%	4%	20%	20%	0%	8%
16:00 - 17:00	0%	25%	5%	-	0%	-	0%	0%	5%	-	50%	20%	20%	25%	0%	8%
17:00 - 18:00	0%	25%	5%	-	0%	-	0%	0%	5%	-	50%	20%	20%	25%	0%	8%
18:00 - 19:00	0%	25%	0%	-	0%	20%	-	25%	50%	14%	38%	0%	-	-	-	-
19:00 - 20:00	0%	-	0%	-	0%	20%	-	25%	50%	14%	38%	0%	-	-	-	-
20:00 - 21:00	0%	-	0%	-	0%	20%	-	25%	50%	14%	38%	0%	-	-	-	-
21:00 - 22:00	0%	-	0%	-	0%	20%	-	-	50%	14%	38%	0%	-	-	-	-

Tabela A.4: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Quinta-Feira. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	-	0%	-	-	0%	-	-	-	-	14%	0%	-	0%	-	-	20%
9:00 - 10:00	-	0%	-	-	0%	-	-	-	-	14%	0%	-	0%	-	-	20%
10:00 - 11:00	-	0%	-	-	-	-	-	-	20%	14%	0%	-	0%	0%	-	20%
11:00 - 12:00	0%	0%	20%	-	-	-	-	-	20%	14%	0%	-	0%	0%	-	-
12:00 - 13:00	-	-	-	-	-	-	-	-	20%	-	-	-	0%	-	-	-
13:00 - 14:00	0%	-	-	-	-	20%	17%	-	-	-	-	-	-	-	-	13%
14:00 - 15:00	0%	15%	0%	-	-	20%	17%	-	-	14%	0%	10%	20%	0%	0%	13%
15:00 - 16:00	-	15%	0%	-	-	20%	17%	-	-	14%	0%	10%	20%	0%	0%	13%
16:00 - 17:00	0%	15%	0%	-	-	-	-	0%	25%	-	0%	10%	20%	0%	-	-
17:00 - 18:00	0%	15%	0%	-	-	-	-	0%	25%	-	0%	10%	20%	-	-	-
18:00 - 19:00	0%	43%	-	-	-	0%	-	-	-	-	-	-	20%	-	-	67%
19:00 - 20:00	0%	43%	-	-	-	0%	-	-	-	-	-	-	-	-	-	67%
20:00 - 21:00	-	43%	-	-	-	0%	-	-	-	-	-	-	-	-	-	67%
21:00 - 22:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	67%

Tabela A.5: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Sexta-Feira. (Fonte Própria, 2017)

Horário / Sala	B3	B5	B6	B11	N3A	N3B	N3C	N4A	N4B	N4C	N5A	N5B	N5C	N6A	N6B	N6C
7:00 - 8:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8:00 - 9:00	-	-	-	-	-	-	-	-	-	-	-	-	0%	-	-	-
9:00 - 10:00	-	-	-	-	-	-	-	13%	-	-	-	-	0%	-	-	-
10:00 - 11:00	-	-	-	-	-	-	-	13%	-	-	-	-	0%	-	-	-
11:00 - 12:00	-	-	-	-	-	-	-	13%	-	-	-	-	0%	-	-	-
12:00 - 13:00	-	-	-	-	-	-	-	13%	-	-	-	-	-	-	-	-
13:00 - 14:00	-	-	-	-	50%	-	-	-	-	-	-	-	-	-	-	-
14:00 - 15:00	-	-	-	-	50%	-	-	0%	-	-	-	-	-	-	-	-
15:00 - 16:00	-	-	-	-	50%	-	-	0%	-	-	-	-	-	-	-	-
16:00 - 17:00	-	-	-	-	50%	-	-	0%	-	-	-	-	-	-	-	-
17:00 - 18:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
18:00 - 19:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19:00 - 20:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
20:00 - 21:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
21:00 - 22:00	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Tabela A.6: **Solução Manual:** Taxa de Ociosidade das Salas nos Horários de Sábado. (Fonte Própria, 2017)