# TCC Sânya

## Capa

HEURÍSTICA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIO: ESTUDO DE CASO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

SANYA CARVALHO DOS SANTOS

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO - UENF
CAMPOS DOS GOYTACAZES - RJ
SETEMBRO - 2013

## Folha em Branco

## Contra capa?

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
CIÊNCIA DA COMPUTAÇÃO

SÂNYA CARVALHO DOS SANTOS

HEURÍSTICA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIO: ESTUDO DE CASO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

CAMPOS DOS GOYTACAZES
2013

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
CIÊNCIA DA COMPUTAÇÃO

SÂNYA CARVALHO DOS SANTOS

HEURÍSTICA PARA O PROBLEMA DA PROGRAMAÇÃO DE HORÁRIO: ESTUDO DE CASO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Trabalho apresentado à Coordenação do Curso de Graduação em Ciência da Computação, da Universidade Estadual do Norte Fluminense como parte das exigências para obtenção do titulo dc Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Ferrnín Alfredo Tang Montané

CAMPOS DOS GOYTACAZES
2013

---

Monografia sob o título Heurística para o Problema da Programação de Caso Ciência da Computação da Universidade Estadual do Norte Fluminense, por Sanya Carvalho dos Santos e aprovada no dia 05/09/2013, em Campos dos Estado do Rio lareiro, pela banca examinadora constituída pelos doutores:

Prof. ry. Fermin Alfredo Tang Montané
Orientador — Universidade Estadual do Norte Fluminense Darcy

Antonio Rivera Escriba
Universidade Estadual do Norte Fluminense Darcy Ribeiro

Prof. Dr. Oscar Alfredo Paz la Torre
Universidade Estadual do Norte Fluminense Darcy Ribeiro

---

Dedico este trabalho a minha mae que me ensinou o amor pelos estudos. Sem o seu apoio e ensinamentos eu não teria a base que precisei para erguer esta conquista.

# Sumário

# 1. Introdução

## 1.1 Considerações iniciais

No inicio de cada período letivo as instituições de ensino desenvolvem sua grade de horários, distribuindo os professores entre as turmas e as turmas entre os horários e salas de aulas disponíveis. Na maioria dessas instituições esse trabalho demanda muito tempo e esforço, geram transtornos e muita das vezes os resultados não são tão satisfatórios, pois a solução pode não atender a todos os requisitos.

O Problema de Programação de Horários (PPH), que na literatura é referenciado como Timetabling Problem (TTP) diz respeito à alocação de aulas na grade de horário de uma instituição de ensino a um conjunto restrito de horários, satisfazendo diversas restrições. Vale ressaltar que o nome Problema de Programação de Horários (PPH) não denota um problema específico, porém, representa a uma família de problemas com diversas variantes que podem ser classificadas em categorias.

Uma categoria deste problema é o Problema de Programação de Horários Professor x Turma com Capacidade (PPTC), neste problema, cada professor deve lecionar um determinado número de aulas para cada turma em um conjunto de períodos [Santos2007]. Outra categoria é o Problema de Alocação de Aulas a Salas (PAAS). Este problema consiste em alocar aulas, com horários de início e término previamente programados, a um número fixo de salas, tal como abordados por Subramanian et al [Subramanian+2011].

O PPH é um problema de dificil generalização, sendo desenvolvido na maioria das vezes, para atender a uma instituição específica, dado que há uma diversidade de regimes que variam de instituição para instituição [Silva+2005]. Sua solução manual geralmente não gera soluções ótimas, demandam muito tempo e esforço, e muita das vezes não atendem aos requisitos essenciais, causando muito transtorno no início do período letivo.

Por ser um problema NP-Dificil a solução ótima para grandes instâncias não é possível de ser encontrada em tempo razoável usando métodos de programação matemática (métodos exatos). O problema é então normalmente tratado através de técnicas heurísticas, que apesar de não enconfrarem soluções ótimas são capazes de retomar soluções de qualidade em um tempo adequado para as necessidades da aplicação.

Entre as técnicas heurísticas destacam-se as chamadas metaheurísticas porque são providas de mecanismos para escapar de ótimos locais. Em geral, no entanto, as metaheurísticas sofrem grande influência das soluções iniciais, isto é, uma solução inicial de boa qualidade induz a um processo de busca mais rápida, com produção de soluções finais melhores [Souza+2002]. Entre as metaheurísticas mais utilizadas para resolver o problema estão: Simulated Annealing, Algoritmos Genéticos, Colônia de Formigas, Busca Tabu, entre outros.

Uma característica particular do problema é a dificuldade de avaliação da solução, pois existem várias soluções viáveis e consideradas de qualidade. Outro aspecto de dificuldade de avaliação do método foi levantado por Santos [Santos2007], ele conclui que apesar das inúmeras heurísticas existentes na literatura, a analise comparativa destas é muitas vezes prejudicada pela metodologia empregada, segundo Santos [Santos2007] as "comparações entre métodos são raras, sendo que a avaliação das soluções obtidas se dá na maioria dos casos por meios subjetivos, através da descrição do sucesso obtido pelo ponto de vista de usuários das aplicações".

## 1.2 Objetivos e justificativas

Existem algumas ferramentas comerciais que prometem a geração automatizada de grades de horários, entretanto, sua utilização é pouco frequente uma vez que este tipo de problema incorre em necessidades específicas de um determinado curso, em detrimento das soluções genéricas existentes [Vieira2011].

O objetivo, portanto, é resolver o problema de alocação de horários de disciplinas para um período letivo qualquer do curso de Ciência da Computação na UENF. Para isso, certas restrições obrigatórias deverão ser atendidas e outras não obrigatórias poderão ser otimizadas. Procura-se produzir um bom quadro de horários de fonna automática.

O PPH por ser um problema NP-Dificil, na medida que os dados de entrada crescem, também cresce sua complexidade, dificultando a obtenção de soluções ótimas. Suas aplicações em diversos contextos, e o grande esforço para encontrar soluções viáveis de alta qualidade vêm atraindo pesquisadores, incluindo uma competição em 2007, a International Timetabling Competition.

Na literatura é possível encontrar diversos trabalhos e abordagens de solução para o PPH, porém, desenvolvidas para instituições especificas. Devido às particularidades de cada instituição não é possível generalizar os requisitos do problema, e fazer um único software generalizado, o que justifica a busca por implementações de métodos de solução automáticos para o curso de Ciência da Computação na UENF , ondejá é encontrada dificuldade pelos professores responsáveis de gerar a grade de todos os períodos em cada semestre,

### 1.3 Estrutura do trabalho

A seguir apresenta-se a estrutura da presente monografia. O Capítulo 2, faz uma introdução aos conceitos de otimização.

Uma revisão bibliográfica referente aos problemas de programação de horários é apresentada no Capítulo 3, onde se destacam os problemas mais estudados.

O Capítulo 4 faz uma descrição geral dos métodos heurísticos.

No Capítulo 5 se encontra a descrição do problema abordado, sua representação e métodos de heurísticas.

O Capítulo 6 apresenta a avaliação dos resultados mediante testes computacionais, que medem tanto a qualidade das soluções quanto o esforço computacional.

Finalmente, no Capítulo 7 apresentam-se as conclusões do trabalho e as propostas de trabalhos futuros.

# 2. Problemas de Otimização Combinatória

## 2.1 Introdução

Otimização, no estudo de um problema, é o processo de encontrar dentre todas as soluções possíveis, ou também chamadas viáveis, a melhor solução, ou a mais satisfatória.

Problemas de otimização podem ser classificados em duas categorias: problemas com variáveis contínuas e com variáveis discretas, também conhecido como Problemas de Otimização Combinatória [Silva2005]. Este capítulo trata do segundo caso.

## 2.2 Métodos de solução exatos

A programação inteira e combinatória ou também conhecida como otimização discreta se caracteriza por possuir variáveis pertencentes a um conjunto discreto.

Existem diferentes problemas de programação matemática que podem ser classificados de acordo com o tipo de variável. Os principais tipos de problemas são os seguintes: Problema de Programação Linear, Problema de Programação Inteira, Problema de Programação Binária e o Problema de Programação Linear Inteira Mista. Estes problemas são descritos a seguir.

Quando todas as variáveis são inteiras, tem-se um problema de programação (linear) inteira (PI),

E se todas as variáveis assumem valores O ou l, tem-se um problema de programação 0-1 ou binária (PB) escrito como:

Já um problema com variáveis inteiras e reais é chamado de programação (linear) inteira mista (PIM) quando possui a seguinte forma:

onde os parâmetros do problema são representados por: A é uma matriz (mxn), D uma matriz (mxp), c um vetor (lxn), d um vetor (lxp) e b um vetor (mxl). Os vetores de variáveis são x e y com dimensões (nxl) e (pxl).

O problema de otimização combinatória (OC) pode ser definido como um conjunto finito N = (1 n}, um conjunto de pesos CJ para cada j e N, e uma família F de subconjuntos factíveis de N, consiste em achar o subconjunto de peso mínimo de F,
isto é:

De modo geral, um problema de OC pode ser formulado como um problema de PI ou de PB. Problemas de programação inteira e combinatória podem ser resolvidos pelos seguintes métodos:

- métodos ótimos (exatos): fornecem soluções ótimas.
- algoritmos aproximados: fornecem soluções subótimas com limite de distância máxima entre elas e o valor da solução ótima.
- métodos heurísticos: fornece uma solução subótima, sem conhecimento da qualidade desta em relação a uma solução ótima.

Dentre os métodos mais bem sucedidos para resolver problemas genéricos de programação inteira estão os baseados nos enfoques de enumeração implícita, ou branch-and-bound, e de planos de corte, presentes nos pacotes comerciais de otimização, tais como CPLEX, XPRESS e LINDO. Diversos problemas de programação de horário podem ser fielmente modelados utilizando a Programação Linear Inteira Mista, e estes podem ser resolvidos através desses softwares [ArenaIes+2007].

Uma descrição de um problema clássico de PPH, o Problema de Programação de Horários Professor x Turma (PPT), onde cada professor deve lecionar um determinado números de aulas para cada turma em um conjunto de períodos foi modelado por Santos [Santos 2007]. O modelo é o seguinte:

Onde:

- P: coujunto de professores, com elementos p e P numerados como 1, , IPI;
- T: conjunto de turmas, com elementos t e T numerados como l, ... , ITI
- D: conjunto de dias letivos, com elementos d D numerados como l, , IDI
- H: conjunto de períodos letivos por dia, com elementos h E H numerados como
- IHI;
- RIPIxlTl: matriz de requerimentos, onde Fpt indica quantas aulas o professor p deve lecionar para a turma t;
- IPIxlDlxlHl: matriz de disponibilidade de professores, onde ppdh = 1 indica que o professor p está disponível do dia d no período h, ppdh = 0 caso contrário; e Xptdh é a seguinte variável de decisão:

As restrições (2.5) asseguram o cumprimento da carga horário, (2.7) a não ocorrência de conflitos em horários para turmas e (2.8) o respeito a disponibilidade dos professores e a não existência de conflitos em horários para professores.

Santos [Santos2007] salienta a intratabilidade computacional do PPT, pois apenas casos especiais podem ser resolvidos em tempo polinomial. Simples considerações de restrições de disponibilidade coloca o PPT na classe de problemas combinatórios NP-Completos, sendo essas considerações bastante comuns em problemas reais.

Apesar da importância teórica, 0 objetivo de se resolver problemas desse tipo é sempre para aplicações reais. Necessita-se então encontrar formas viáveis de determinar soluções aceitáveis em tempo hábil. É neste caso que entram as heurísticas e metaheurísticas, já que nos fornecem essa possibilidade. As heurísticas e metaheurísticas são explicadas detalhadamente no capítulo 4.

# 3. Revisão Bibliográfica do Problemas de Programação de Horários

## 3.1 Introdução

O Problema de Programação de Horários (Timetabling Problem) é um problema de grande relevância e amplamente estudado na área de Pesquisa Operacional. Um número significativo de trabalhos sobre esse problema foi publicado nos últimos anos e conferências regulares discutem o tema no meio científico [Splinder2010].

Em geral, os problemas de timetabling são classificados de acordo com os setores de aplicação a que se propõe: área esportiva, hospitalar, educacional, entre outras [Vieira2011]. Em seu trabalho, Splinder [Splinder2010] apresenta uma definição bem genérica para o problema de timetabling como "a alocação, sujeito a restrições, de recursos a objetos colocados no espaço e no tempo, de modo a satisfazer, tanto quanto possível, um conjunto de objetivos desejáveis".

Com relação a área educacional uma definição do problema de programação automatimda de um horário acadêmico consiste em: agendar uma sequência de encontros (aulas, exames, bancas, palestras ou outro tipo de atividade escolar) entre professores e estudantes em um período de tempo prefixado (normalmente uma semana) satisfazendo um conjunto de restrições de vários tipos.

As diversas variantes do Problema de Horário Educacional podem ser classificadas em duas grandes categorias: o Problema de Programação de Horário Escolar (School Timetabling), e o Problema de Programação de Horário de Disciplinas em Universidades (University Course Timetabling).

O Problema de Programação de Horário Escolar pode ser generalizado como o escalonamento semanal das aulas em uma escola sem que professores e alunos tenham mais de uma aula ao mesmo tempo (estudantes são agrupados em turmas com os mesmos planos de aula). Já o Problema de Programação de Horário de Disciplinas em Universidades como o escalonamento semestral das aulas de um conjunto de disciplinas de uma universidade de modo a evitar colisão de horários (estudantes geralmente são considerados individualmente) [Paim+2010].

Como em todo PPH existem diversas variações, e em cada instituição o problema assume particularidades, surgem muitas outras classificações, tratando apenas alguma dessas particularidades como, por exemplo, o Problema de Alocação de Salas, o Problema de Programação Turma com Capacidade, o Problema de Horário Professornurma, etc. Tamanha é a abrangência e o número de variantes que Gomes [Gomes2006] listou as seguintes denominações para os problemas de timetabling: nurse rostering , university timetabling , exam timetabling, sport timetabling, railway timetabling school timetabling course timetabling class-teacher timetabling, student scheduling, teacher assignment, project sheduling, busscheduling, airflight schedules.

Este capítulo apresenta alguns dos casos mais estudados encontrados na literatura, onde podem ser observados diferentes variantes do problema PPH e métodos de resolução adotadas em cada caso.

## 3.2 Casos mais estudados

Para resolver o problema de Programação de Horários em Escolas Souza [Soum2000] desenvolveu uma heurística de Busca Local baseada em caminhos mínimos para melhorar um quadro de horário, ainda que com

certo grau de inviabilidade. A heurística age primeiramente tentando eliminar a inviabilidade, e se bem sucedida tenta melhorar os requisitos de qualidade exigidos para o quadro de horário. Esse procedimento foi inserido em várias metaheurísticas e comprovou-se que sua utilização não somente faz alcançar a viabilidade mais rapidamente, quanto, também, faz gerar soluções finais de melhor qualidade. A partir do desempenho das diversas metaheurfsticas testadas, um algoritmo GRASP com busca local, feito por um algoritmo de Busca Tabu, foi idealizado. Esse algoritmo mostrou-se superior a todos os demais, sendo capaz de produzir soluções melhores mais rapidamente. Foi proposta, também, uma modelagem de programação matemática.

Costa [Costa2003] apresentou uma técnica híbrida com base nas metaheurísticas GRASP e Busca Tabu. Os resultados computacionais obtidos com esse procedimento mostraram que, soluções viáveis de melhor qualidade podem ser encontradas mais rapidamente.

Santos [Santos2007] aplicou em sua pesquisa três metodologias de solução. A primeira, uma nova heurística híbrida, baseada em Busca Tabu, onde os experimentos computacionais demonstraram melhoras dos resultados da literatura, apresentando um desempenho consistentemente superior em todos os problemas teste. A segunda foi usando técnicas de Programação Linear Inteira Mista, no qual permitiu a obtenção de limites inferiores bastante fortes, os quais permitiram a prova da otimalidade para 3 instâncias da literatura. E a terceira foi explorar a sinergia entre heurísticas e métodos exatos. Esses algoritmos híbridos ofereceram os melhores limites superiores disponíveis.

Com o objetivo de comparar três técnicas (método matemático, abordagem heurística e método misto) Góes et al [Góes+2010] desenvolveram um protótipo para a construção da grade horária escolar obtendo a programação de horários de professores/turmas para uma instituição de ensino municipal. O modelo matemático gerou o melhor resultado chegando a solução ótima, porém com um tempo computacional alto em relação aos outros métodos. A abordagem heurística baseada em Algoritmos Genéticos apesar de nem sempre apresentar a melhor solução, retomou um resultado mais rápido do que o modelo matemático. Com o menor tempo computacional o método misto resolve primeiramente o problema através do modelo matemático sem adicionar restrições de preferência (ou não) por aulas geminadas, que juntas compõem mais de 50% das restrições geradas, na sequência, aplica-se urna heurística de melhoramento com relação à preferência de aulas geminadas. Todos os três métodos utilizados apresentaram resultados melhores do que o gerado manualmente para o estudo de caso abordado.

Com relação ao Problema de Programação de Horário em Universidades Vieria [Vieira+201 II apresenta um estudo de caso no Departamento de Computação (DCOMP) da Universidade Federal de Sergipe (UFS), em particular do curso de Sistemas de Informação. A abordagem proposta foi uma metaheurfstica com base em Algoritmos Genéticos. Tais algoritmos realizam um processo de busca inspirado nos princípios da seleção natural e evolução para que a melhor solução seja gerada ao final do processo. A abordagem adotada permite a reprodução da solução de forma a atender restrições de outros cursos e de outras instituições. Os experimentos realizados e as análises feitas deram validade a modelagem apresentada, mostrando que a solução é viável, porém com relação à restrições soft, nem todas foram plenamente atendidas, mesmo na melhor solução encontrada.

Splinder [Splinder2010] propôs uma solução para o Problema de Horário Educacional utilizando um algoritmo de Busca Dispersa (Scatter Search) que utiliza o método de Reconexão por Caminhos como estratégia de intensificação de busca da solução ótima. O autor frisa que não foram encontrados trabalhos que reportem a utilização desta técnica para a solução deste tipo de problema anteriormente ao seu trabalho. Para testar a eficiência da abordagem proposta, foram considerados dois estudos de casos: o primeiro referente ao problema de progamação de cursos pós matrícula, abordado na International Timetabling Competition, e o segundo, referente a problemas encontrados no cenário de uma instituição de ensino brasileira, do tipo universidade. A implementação do modelo e sua execução sobre os dados da International Timetabling Competition proporcionaram a comprovação de sua aplicabilidade, mesmo que com resultados inferiores aos registrados anteriormente por outros métodos de solução. Apesar do tempo computacional elevado que demarxla a aplicação da técnica utilizada, as grades de horário geradas atenderam a todas as restrições consideradas gerando grades de horário de boa qualidade. A forma em que o método foi implementado apresenta uma grande dificuldade de escapar de regiões representando ótimos locais. Outra dificuldade foi a validação da eficiência de estruturas de vizinhança para o problema abordado.

### 3.3 Outras Variantes

Um problema relacionado com o PHH, que é bastante estudado na literatura, é o Problema de Alocação de Salas. Silva [Silva2005] usou instâncias distintas e a linguagem de programação Java para testar a técnica de Simulated Annealing neste problema. O sistema desenvolvido mostrou-se eficiente e produziu bons resultados, além de possuir uma flexibilidade útil ao executar o programa para outras instâncias, bastando atualizar o Banco de Dados.

Já Oliveira [Oliveira2006] compara duas técnicas heurísticas: Algoritrno Genético e Simulated Annealing. Com implementações pouco modificadas em relação àquela original apresentada na literatura, o Simulated Annealing mostrou-se mais eficiente.

Subramaniam et al [Subramanian+2011] aplica a metaheurística Busca Tabu em um estudo de caso feito em um Centro de Tecnologia (CT) de uma Instituição de Ensino Superior. Esta instituição possui um total de 28 salas de aulas divididas em três tipos — Carteira (17), Mesa (8), Prancheta (3). Através de uma abordagem construtiva foi possível, quase que instantaneamente, soluções que contemplam todas as restrições de viabilidade com adicional qualitativo. Observou-se que os resultados apresentados pelo procedimento desenvolvido foram superiores aos procedimentos manuais, visto que, desta forma, não era possível alocar cerca de 4,7% do total de horas/aula.

Outra variante, na área esportiva, é o Problema na Programação de Jogos. Biajoli em seu trabalho, "Resolução do Problema de Programação de Jogos no Campeonato Brasileiro de Futebol", apresenta contribuições junto à solução de dois problemas de escalonamento de jogos: programação de jogos para competições que apresentam tumo único e programação de jogos para competições que apresentam turno e returno. O objetivo foi construir uma tabela de jogos entre times que participam de uma competição esportiva realizada em vários locais e ao longo de um determinado conjunto de rodadas, minimimndo a distância percorrida pelo time que menos se deslocou e a distância percorrida pelo time que mais se deslocou, satisfazendo a um determinado conjunto de restrições. Duas técnicas metaheurísticas de busca local associadas foram empregadas como estratégia de solução do problema: Simulated Annealing e Busca Tabu. Em todas as soluções, tanto para competições que apresentam turno único quanto para as que apresentam turno e returno, não houve a ocorrência de nenhum tipo de inviabilidade, além da melhora na distância percorrida e na diferença entre a maior e menor distância percorrida pelos times em relação à tabela oficial praticada no ano anterior. Em todos os testes o algoritmo atendeu os requisitos essenciais, obtendo em cada caso, tabelas de jogos viáveis, ou seja, praticáveis.

# 4. Metodologia

## 4.1 Conceitos de Heurística e Metaheurística

O terrno heurística é derivado do grego heuriskein, que significa descobrir ou achar, encontrar [Silva2005]. As heurísticas são métodos aproximados que se preocupam em encontrar soluções próximas da otimalidade em um tempo computacional hábil. Arenales et al [ArenaIes+2007] cita uma definição de Nicholson (1971) que expressa muito bem as características de uma heurística: é um procedimento para resolver problemas por meio de um enfoque "intuitivo", em geral racional, no qual a estrutura do problema possa ser interpretada e explorada inteligentemente para se obter uma solução razoável.

Como se pode perceber, o enfoque de uma heurística é encontrar uma solução aceitável, razoável, que seja útil, mesmo não sendo esta a melhor. Problemas como do PPH são comumente resolvidos através de métodos heurísticos por estes fornecerem soluções satisfatórias para o problema, já que os métodos exatos não o fazem em tempo razoável.

Na seção 4.2, descreve-se três tipos específicos de heurísticas: heurísticas construtivas, heurísticas de busca local e metaheurísticas. Tais heurísticas diferem tanto no esforço computacional realizado quanto na qualidade das soluções que são capazes de produzir, sendo que a qualidade da solução é diretamente proporcional ao esforço realizado. Dentre essas a que se destaca são as metaheurísticas por permitirem escapar de ótimos locais.

## 4.2 Métodos construtivos (Solução Inicial)

Heurísticas consü•utivas constroem uma solução a partir da instância do problema, ou seja, constroem a solução de forma incremental escolhendo passo a passo o melhor candidato de acordo com algum critério inserindo-o na solução até gerar uma solução completa. São geralmente mais simples e fáceis de implementar se comparadas às metaheurísticas. A principal utilidade dos algoritrnos construtivos é ser entrada dos algoritmos iterativos.

A forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, a qual, por sua vez, depende do problema abordado. Dois métodos construtivos comuns na geração de solução inicial são: Construção Aleatória e Construção Gulosa.

Algoritmos de construção aleatória simplesmente geram um posicionamento qualquer, isto é, a cada passo, o elemento a ser inserido na solução é aleatoriamente selecionado dentre o conjunto de elementos candidatos ainda não selecionados. Não há otimização, mas é o mais rápido, além de fácil implementação. A grande desvantagem é a baixa qualidade da solução inicial gerando um maior esforço computacional na fase de refinamento, entretanto, se aplica perfeitamente para algoritmos iterativos que independem da solução inicial.

Nas heurísticas clássicas, os elementos candidatos são geralmente ordenados
segundo uma construção gulosa, que estima o beneficio da inserção de cada elemento, e somente o "melhor" elemento é inserido a cada passo [Dutra2008]. A função de avaliação costuma ser específica para cada problema. A figura 4.1 mostra o pseudocódigo para a geração de uma solução utilizando a construção gulosa.

```
Procedimento Construção Gulosa

1. Inicializar s
2. Enquanto a solução s não esteja completa
3. Identificar a melhor ação possível em relação a um determinado critério
4. Executar a ação modificando s
5. Fim do enquanto
6. Retornar s
```

Figura 4.1- pseudocódigo construção gulosa.

Na linha I o conjunto 8 é inicializdo e está vazio, ou seja, não há candidatos escolhidos. Então, a cada passo, utiliza-se um critério de seleção para detenninar qual o melhor candidato a ser inserido na solução 8 (linhas 3 e 4). Isso deve ser feito enquanto a solução g não esteja completa, ou seja, há candidatos não avaliados que devem ser inseridos (linhas 2 e 5).

As soluções iniciais geradas por um algoritmo de construção gulosa geralmente são melhores que as obtidas de forma aleatória. A desvantagem da construção gulosa é que o número de soluções possíveis é pequeno [Fraga2006].

## 4.3 Métodos de refinamento (Busca Local)

As heurísticas de refinamento em problemas de otimimção, também chamadas de técnicas de busca local, constituem uma família de técnicas baseadas na noção de vizinhança. Sua proposta consiste em buscar um bom resultado para o problema caminhando por ótimos locais, efetuando a troca sistematicamente de vizinhança.

Em linhas gerais, essa classe de heurísticas partem de uma solução inicial qualquer, a qual pode ser obtida por uma heurística construtiva ou então gerada aleatoriamente, e caminha, a cada iteração, de vizinho para vizinho

de acordo com a definição de vizinhança adotada, ao encontrar solução de melhor qualidade em sua vizinhança, esta substitui a solução corrente [Dutra2008].

### 4.3.1 Conceito de Vizinhança

Fraga [Fraga2006] define a vizinhança de uma solução da seguinte forma: seja S o conjunto de todas as soluções possíveis para um determinado problema de otimização e f a função objetivo a minimizar. Uma função g, que varia com a estrutura do problema tratado, associa cada solução s e S a uma vizinhança N(s) ç S. Um movimento m transforma uma solução s em uma outra solução s' , que esteja em sua vizinhança. Logo, s' e N(s) é chamado de vizinho de s. A operação da aplicação do movimento m em s gerando o elemento s' pode ser representada da seguinte forma: s' s m.

A figura 4.2 demonstra o pseudocódigo de um algoritmo de refinamento que recebe uma solução qualquer e retoma o melhor vizinho da mesma, caso exista.

```
procedimento Heurística de Refinamento (s e S)
1. s' = MelhorVizinho(s);
2. Enquanto s' # s
3. s' = MelhorVizinho(s);
5. Fim do enquanto
6. Retornar s
fim Heurística de Refinamento;
```

Figura 4.2 - pseudocódigo de um algoritmo de refinamento.

A definição de vizinhança é crucial em uma heurística de refinamento. De uma solução s do espaço de soluções deve ser sempre possível atingir qualquer outra solução em um número finito de passos, utilizndo um determinado tipo ou tipos de movimentos [Dutra2008]. Mendes [Mendes1999] faz uma abstração de buscas locais partindo de soluções aleatórias conforme a figura 4.3.

Figura 4.3 - Ilustração de buscas locais parindo de soluçaes aleatórias.

### 4.3.2 Método de Descida Clássico

O mais antigo método de procura em vizinhança é o método da descida. Seja f uma frnção que avalia uma solução. O método da descida consiste em 3 passos apresentados na figura 4.4.

```
procedimento Método da Descida
1. Escolher uma solução inicial i.
2. Encontrar a melhor solução j N (i), isto implica que fG) f(k) para
3. qualquer k E N (i).
4. Se f(j) f(G), então pare.
5. Se não je volta para o passo 2.
fim Método da Descida
```

Figura 4.4 - pseudocódigo do método da descida clássico.

O método da descida claramente para em um ótimo local, mas não necessariamente um ótimo global de f. Metaheurísticas são métodos de busca em vizinhança mais elaborados do que o método da descida clássico, elas utilizam estratégias para escapar desses ótimos locais, possibilitando uma maior chance de chegar a um ótimo global.

## 4.4 Metaheurísticas

As metaheurísticas usam técnicas para guiar e modificar as heurísticas de forma a produzir soluções além daquelas geradas por heurísticas de busca local. Resultante dos estudos nas áreas de Otimização Combinatória e Inteligência Artificial, deram aos métodos heurísticos até então existentes, um caráter mais geral, tornando-os mais flexíveis e inteligentes [Silva+2005].

Problemas de otimização combinatória, como já apresentado no capítulo 2, são problemas típicos de minimização ou maximização, nesse tipo de problema "encontra- se um ótimo local quando qualquer movimento a ser feito piore o valor atual da função objetivo. Um ótimo global corresponde ao menor valor da função objetivo, entre todos os ótimos locais existentes no espaço de busca" [Chaves2003]. A Figura 4.5 mostra graficamente um problema de minimização com três ótimos locais e um ótimo global.

Figura 4.5 - Representação gráfica para um problema de minimização.

Dentre as metaheurísticas existentes, destacam-se: Simulated Annealing, Algoritmos Genéticos, Colônia de Formigas e Busca Tabu. Diferentes estratégias são usadas por cada uma das técnicas para explorar o espaço de busca. Observações em fenômenos naturais tem sido também de suma importância, estimulando a construção de vários dentre esses métodos.

### 4.4.1 Algoritmos Genéticos (AG)

É baseado na teoria de Darwin da evolução das espécies, utilizando conceitos da evolução biológica como genes, cromossomos, cruzamento, mutação e seleção na computação [Oliveira2006].

O AG atua sobre uma população fazendo com que esta evolua de acordo com uma função de avaliação, Uma população inicial é gerada, aleatoriamente ou não. A cada iteração novos indivíduos são gerados, ou seja, uma nova população, através de seleção, cruzamento ou mutação. Este processo de evolução continua até que se atinja um critério de parada.

Em analogia a tenninologia biológica o termo cromossomo consiste em uma maneira de traduzir a informação da solução do problema em uma maneira viável de ser tratada pelo computador. Cada pedaço indivisível desta representação é chamado gene, sua representação mais simples é a binária.

### 4.4.2 Colônia de Formigas

Foi inspirado na observação do comportamento das formigas ao saírem de sua colônia (formigueiro) para encontrar comida. As formigas reais são capazes de encontrar o caminho mais curto para uma fonte de alimento do formigueiro sem a utilização de dados visuais.

Enquanto caminham, as formigas depositam no solo uma substância denominada de feromônio e tem seu deslocamento baseado em trilhas de feromônios previamente depositados por outras formigas [Coelho+2004]. Com o passar do tempo, entretanto, as trilhas de feromônio começam a evaporar, reduzindo, assim, sua força atrativa. Trilhas mais longas evaporam mais rápido, consequentemente o caminho mais curto atrairá mais formigas, o que implica aumento na intensidade de feromônio. Portanto, o caminho mais curto sempre será o escolhido, pois a evaporação também possui a vantagem de evitar a convergência para uma solução local ótima. O algoritmo é então utilizado para problemas computacionais que envolvem procura de caminhos em grafos, como, por exemplo, o Problema do Caixeiro Viajante.

### 4.4.3 Busca Tabu

Em sua forma clássica a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho s ' da vizinhança N(s) da solução corrente s. Independentemente def(s') ser melhor ou pior que f(s), s' será sempre a nova solução corrente [Costa2003]. No entanto, esse mecanismo não é suficiente para escapar de ótimos locais, pois pode haver retorno de uma solução previamente gerada. Para que isto não ocorra, o algoritmo utiliza o conceito de lista tabu.

A lista tabu "define todos os movimentos que têm um certo atributo como sendo tabu por um determinado número de iterações, conhecido como tempo tabi/' [Souza+2002]. Esses movimentos são proibidos, a menos que a solução satisfaça um certo critério de aspiração A, como por exemplo que a solução seja melhor do que a melhor solução já encontrada. Os atributos são escolhidos para prevenir o retorno a soluções visitadas recentemente e por possuírem características fáceis de detectar. O procedimento chega ao fim quando alcança um certo critério de parada, geralmente um determinado número de iterações sem melhoras.

## 4.5 Simulated Annealing

Simulated Annealing (SA), um método de otimimção bastante geral, se destaca na literatura por um excelente desempenho em problemas que apresentam um número muito grande de ótimos locais. Inspirado pelo processo fisico proposto por Metropolis et al [Metropolis+1953] que descreve a obtenção de estruturas cristalinas, o nétodo foi estendido de otimização termodinâmica para o problema de otimização combinatorial por a. Kirkpatrick et al [Kirkpatrick+1983] em 1983 no contexto da mecânica estatística.

### 4.5.1 Analogia Física

"Annealing é o processo utilizado para fiadir um metal, onde este é aquecido a uma temperatura elevada e em seguida é resfriado lentamente, de modo que o produto final seja uma massa homogênea" [Haeser+2008].

A ideia fundamental do SA está baseada nesta observação fisica da matéria: a obtenção de estruturas cristalinas é feita através do super aquecimento e resfriamento lento da substância, o que induz o sistema a atingir o nível mais baixo de energia possível. Quanto mais organizada a esü•utura cristalina de um material, menor sua energia.

No princípio fisico descrito, quando há o super aquecimento o cristal é levado a sua temperatura de fusão, neste momento as moléculas estão em um estado muito desordenado e se agitam livremente como descrito por Noronha na figura 4.6.

Figura 4.6 - Estado instável das moléculas da matéria em fusão.

Quando o processo de resfriamento é feito à amostra de forma brusca por um mecanismo análogo aquele da têmpera de um metal, o nível de energia vai baixar rapidamente e as moléculas vão se encontrar em um estado ainda muito desordenado no qual o nível de energia é muito superior ao do cristal perfeito. Este estado, dito amorfo, representado pela figura 4.7, é então distintamente menos estável que o estado ordenado da figura 4.8 [Noronha2000].

Figura 4.7 - Estado das moléculas consecutivo a um resfriamento rápido.

Por outro lado, se o resfriamento da amostra acontecer de maneira progressivamente lenta, as moléculas vão adquirir a estrutura cristalina estável e ordenada, que tem um nível de energia mais fraca possível como na figura 4.8.

Figura 4.8 - Estado das moléculas consecutivo a um resfriamento lento.

### 4.5.2 Descrição do algoritmo

O algoritmo, então, simula este processo substituindo a solução atual por uma solução próxima de acordo com uma função objetivo e com uma variável T, dita temperatura. Inicialmente T admite valores altos, permitindo 0 algoritmo testar soluçoes mais distantes da solução atual e dar mais independência do ponto inicial da pesquisa.

À medida que o algoritmo progride a temperatura vai resfriando, ou seja, o valor de T vai decrescendo, e a probabilidade de aceitação de uma solução pior também, fazendo com que o algoritmo converta para uma solução ótima local. O procedimento é finalizado quando a temperatura chega a um valor próximo de zero e nenhuma solução que piore o valor da melhor solução seja mais aceita, ou seja, quando o sistema estiver estável. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um mínimo local. Na figura 4.9 se encontra o pseudocódigo no algoritmo Simulated Annealing.

```
CÓDIGO
```

Figura 4.9 — Algoritmo Simulated Annealing.

O procedimento começa na linha 1, com a inicialização da melhor solução conhecida s *, calculada previamente por métodos de construção e refinamento. Nas linhas 2 e 3, inicializam-se o contador de iterações para sustentação de temperatura constante, IterT, e a temperatura inicial, T, respectivamente. Enquanto a temperatura corrente T for maior que zero e o contador IterT for menor que o número máximo de iterações, um vizinho de s é gerado, denotado como s' (linha 7). Na linha 8, calcula-se A como a diferença entre os valores das funções objetivo correspondentes às soluções s' e s, solução vizinha e solução atual, respectivamente. Nas linhas de 9 a 15, o valor de A será avaliado para verificação de aceitação da solução vizinha como solução atual. Na linha 9 verifica-se se A<O, a solução vizinha é melhor que a solução atual, neste caso a solução vizinha substitui a solução atual (linha 10). Verifica-se também se essa solução vizinha é melhor que a melhor solução s* (linha I l). Caso A 0, a solução vizinha s' (que é pior que a solução atual s) só será aceita como a solução atual de acordo com um critério de aceitação de piora. Esse critério é avaliado nas linhas 13 e 14. Na linha 13 um número de O a 1 é gerado aleatoriamente, já na linha 14, se este número for menor que uma função conhecida por fator de Boltzrnann, que é dada por e¯Ô/T, a piora da solução será aceita. Nesta função a temperatura Té usada como parâmetro. Inicialmente quando a temperatura está alta a probabilidade de aceitação de piora é maior. Após um número fixo de iterações a temperatura T é gradativamente diminuída por uma taxa de resfriamento a, sendo 0 < a < 1 (linha 17). Conforme a temperatura for caindo a probabilidade de aceitação de piora também será menor. O algoritrno termina quando a temperatura T for suficientemente próxima de zero, retornando a melhor solução s* (linha 21).

Algoritrnos baseados em Simulated Annealing com frequência consideram o princípio de reaquecimento, reinicialização do valor da temperatura T, seguido de um novo processo de resfriamento, utilizado quando a quantidade de movimentos consecutivamente rejeitados é alta. É também comum trabalhar nas temperaturas mais altas com taxa de resfriamento baixa e aumentar essa taxa quando a temperatura diminuir.

# 5. Aplicação da Metodologia ao Problema de Programação de Horários

## 5.1 Definição do Problema de Programação de Horários

A definição básica do problema timetabling, segundo Cooper e Kingston [Cooper1992], é atribuir horários, professores, alunos e salas para uma coleção de aulas e outros encontros de tal forma que a nenhum dos participantes seja necessário estar no mesmo horário em duas aulas ou encontros diferentes.

AS combinações possíveis para resolver o problema de acordo com esta definição já geram uma dificuldade computacional para grandes instâncias. Se tratando de casos reais, a dificuldade pode aumentar significativamente, a ponto de a complexidade ser tão grande que toma o a solução inviável. Isso acontece

devido aos requisitos adicionais considerados, podendo estes ser obrigatórios, ou seja, se não atendidos tornam a solução inviável, ou de qualidade, que devem ser atendidos, mas seu não atendimento não inviabiliza a solução [Souza+2002].

Considerando os dois tipos de requisitos (restrições e requisitos de qualidade), Subramanian et al [Subramanian+2011] modelou uma função de avaliação f, a qual deve ser minimizada, para a avaliar a solução s. Sendo ri(s) as restrições e q(s) os requisitos de qualidade, a expressão da função fé dada por:

$$f(s) = R(s) + Q(s) \text{ (5.1)}$$

sendo

$$R(s) = Mx \sum_{i=R_1}^{R_n} r_i(s) \text{ (5.2)}$$

e

$$Q(s) = \sum_{Q^n}^{i=Q_1} \alpha_j q_j \text{ (5.3)}$$

em que é o números de vezes que a restrição i E {RI ...Rn} , e qj(s) os requisitos de qualidade j c (Ql Qn) não são atendidos na solução s. M é um parâmetro de penalidade associado ao não atendimento das restrições, enquanto aj O é a penalidade associada ao requisito de qualidade qj. A ideia de Subramanian et al [Subramanian+2011], foi permitir ao usuário atribuir valores às penalimções referentes aos requisitos de qualidade diretamente proporcionais às suas respectivas relevâncias, permitindo este gerar diferentes soluçOes, por meio da variação de tais penalidades.

Uma solução viável para este modelo só ocorre quando R(s) assume valor. zero, isto ocorre quando cada termo se anula. Para alcançar este objetivo, o parâmetro de penalidade M deve ter um valor suficientemente elevado na função de
avaliação.

Santos [Santos2007], em seu trabalho, classifica os requisitos em três tipos:

organizacionais: relativo à instituição de ensino; pedagógico: pedidos importantes para o bom aproveitamento das aulas; pessoais: requisitados de acordo com as preferências pessoais dos membros do corpo docente.

Porém a obrigatoriedade ou não dos requisitos não está ligada ao tipo, pois dentro de cada uma dessas classificações podem ter requisitos obrigatórios ou não. Santos [Santos2007] os chamam de restrições fortes, para os obrigatórios, e de restrições fracas, para os de qualidade.

## 5.2 Definição do Estudo de Caso

Como na UENF a tarefa de distribuição de sala não varia muito a cada período, sendo feito separadamente por cada centro e as aulas que necessitam de salas com recursos especiais são geralmente já pré estabelecidas, não há necessidade de automatizar esta tarefa de distribuição de salas. Outra tarefa que no presente cenário do curso de Ciência da Computação não viabiliz algum tipo de automatização é a distribuição de professores, pois além de um número muito pequeno destes, não há muitas alternativas de mudanças de suas respectivas disciplinas.

Desta forma este trabalho visa resolver apenas 0 problema de distribuição de horário para o curso de Ciência da Computação na UENF, onde existem a cada período 5 turmas com em média 7 disciplinas por período, sendo ao todo mais ou menos 35 disciplinas, com geralmente, 4 horas/aulas cada disciplina. O objetivo é criar uma grade de horário que atenda a todos os requisitos dos problema. Estes são divididos em duas categorias: requisitos essenciais e requisitos não essenciais.

Para uma gade de horário ser considerada viável todos os requisitos essenciais devem ser atendidos, ou seja, é obrigatório seu cumprimento. Os não essenciais devem ser satisfeitos sempre que possível. Nao inviabilizam uma solução, contudo, seu atendimento é sempre desejável. Uma grade de horários ideal é aquela que cumpre todos os requisitos, essenciais ou não. Os requisitos são apresentados a seguir.

Requisitos essenciais, ou seja, obrigatórios:

REI - Um professor não pode lecionar aula em duas turmas diferentes no mesmo horário.
RE2 - Uma turma não pode ter aula em duas disciplinas no mesmo horário.

Requisitos não essenciais, de qualidade:

RNEI - O ideal é que existam no máximo duas aulas consecutivas da mesma disciplina.
RNE2 - Não devem haver mais de duas aulas da mesma disciplina em um dia.
RNE3 - Não preencher os horários de 12h às 14h, pois se trata de horário de almoço.
RNE4 - Os professores associados, por terem exclusividade com a instituição, preferem espalhar os horários das aulas dadas, e não acumular todas no mesmo dia.
RNE5 - Os professores contratados, por outro lado, preferem que suas aulas sejam alocadas num mesmo dia, ou no menor número de dias possíveis.

## 5.3 Definição da Representação

O problema da programação de horário abordado considera um conjunto de aulas chamado eventos E e um conjunto de timeslots T onde T é igual ao número de dias vezes a quantidade de horas por dia, sendo portanto 50 timeslots, pois há 10 horários possíveis para 5 dias da semana. Este problema consiste em alocar as aulas do conjunto E aos timeslots do conjunto T, de maneira que todas as aulas sejam alocadas, respeitando todas as restrições fortes além de minimizar ao máximo possívei o núrnero de restrições fracas. Portanto, para cada timeslosts existem no máximo 5 opções de alocação de disciplinas, pois há no máximo 5 turmas por semestre, evitando a violação da restrição forte RE2. Uma solução ideal é aquela que não desrespeita nenhuma restrição. A representação dos timeslosts se encontra na figura 5.1.

Figura 5.1 — Distribuição dos timeslots com relação aos horários e dias da semana.

Uma solução para o problema é encontrada quando cada evento (aula) é alocado em um timeslost respeitando os requisitos essenciais. A qualidade da solução é por sua vez, determinada através do cumprimento dos requisitos não essenciais, ou também chamados requisitos de qualidade.

Com isso, uma solução S para o problema pode ser medida através de uma função objetivo. A função objetivo f associa cada solução S do espaço de soluções a um numero realf(s), e deve ser minimizada.

A criação de soluções neste presente projeto foi divida em 3 fases. A primeira é criação de solução construtiva, utilizando por tanto uma heurística construtiva. A segunda e a terceira fase são, respectivamente, uma heurística de melhoria e uma metaheurística Simulated Annealing, realizando movimentos de melhoria a partir da solução construtiva obtida na primeira fase. Essas fases são detalhadas nas seções a seguir.

## 5.4 Métodos de Construção

Uma heurística construtiva para o PPH consiste em formar uma boa grade de horário considerando a cada iteração somente o próximo passo, ou seja, ela parte de uma solução vazia e através da inserção das disciplinas, uma de cada vez, seguindo algum critério, até atingir a grade de horário completa. Algoritmos construtivos não possuem nenhum esquema de backtracking, ou seja, após uma disciplina inserida na grade não é possível retirá-la de seu horário.

O critério de inserçao para a heurística construtiva neste trabalho consiste em obedecer a uma matriz chamada de matriz de preferência. Esta matriz de tamanho 5x6 representada na figura 5.3, guarda a preferência da alocação do próximo dia para uma disciplina, de forma a espalhar os horários de aulas de uma mesma disciplina. A primeira coluna representa os 5 dias da semana, de segunda a sexta. As próximas colunas representam os próximos dias desejáveis para alocação da disciplina que já está alocada no dia da semana desta primeira coluna. A segunda coluna, portanto, representa o melhor dia para a alocação da disciplina já alocada de acordo com a primeira. Quanto mais se afasta da primeira coluna, pior é considerado o resultado desta alocação, mas mesmo assim viável. A última coluna é igual a primeira, pois não havendo espaço em nenhum outro dia, aloca-se no mesmo dia, fechando todas as opções de dias possível para a alocação. Esta é a pior solução possível.

De acordo com os índices de cada dia da semana mostrados na figura 5.2 e figura 5.3 podemos montar a matriz de preferência.

Figura 5.2- índices correspondentes aos dias da semana.

Figura 5.3- Distribuição dos dias da semana de acordo com a preferência de alocação e a matriz de preferencia.

Através deste método construtivo, onde, seguindo o critério de inserção da matriz de preferência na matriz de timeslot é possível criar uma solução na qual atenda todos os requisitos essenciais, ou seja, após a inserção de todas as disciplinas através da matriz de preferência já existe uma solução viável.

Após a construção dessa solução viável são realizados movimentos de melhoria para atender aos requisitos não essenciais, ou de qualidade. Estes movimentos são explicados na Seção 5.5 e 5.6. Como observado em muitos dos trabalhos encontrados na literatura, a qualidade da solução inicial pode interferir no resultado final após a aplicação de outros métodos. Foi em busca desta qualidade na solução inicial que se pensou neste método de inserção usando a matriz de preferência.

O algoritmo 5.1 Alocar Disciplinas(id) mostra como é feita a alocação de acordo com matriz de preferência.

```
I : inicio
2: Numhoras 4— Disciplina(id).Nhor;
6:
8:
9:
11:
12:
13:
15:
16:
17:
18:
19:
21:
22:
23:
25:
26:
27:
```

```
28:
29:
30:
31:
32:
33:
35:
36:
37:
// índice do timeslot
// índice do timeslot do dia da preferência
// índices iniciais da matriz de preferência
enquanto (Numhoras faça
se Numhoras então
nt<—2; //nt—• total de horas alocadas de uma só vez
senão
encontrare-FaIso;
se então
// sem primeira alocação
enquanto (i<5) e !(encontrar) faça
itd<—matriz_prefência(i,j); //dia da preferência
(id,itd, it.,nt)
se !(encontrar) então
senão
fim-enquanto
senio
// seguinte dia
// seguinte preferência
/.com primeira alocação
enquanto (j<5) e !(encontrar) faça
itd<— matriz_prefência (i,j);
10);
encontrare—BuscaNoDia (id,itd, it,nt)
se !(encontrar) então
// seguinte preferencia
nm—enquanto
fim-se
se (encontrar) então
Numhoras 4— Nwnhoras — nt;
```

```
fim-se
fim-enquanto;
```

O algoriüno, na linha 2 se inicia recebendo o número de horas/aulas da disciplina a ter seus horários alocados na matriz timeslot. Esses horários são alocados dois a dois enquanto possível nas linhas de 7 a 11, ou seja, somente no caso de disciplinas com número de horas/aulas ímpar, que um horário apenas será alocado sozinho, nunca três horários seguidos. A primeira alocação é feita nas linhas 12 a 22. Na linha 15 a variável itd recebe o dia de preferência da primeira alocação de acordo com a matriz de preferência. Na linha 17 a função BuscaNoDia verifica se é viável a alocação desses horários no dia desejado. Se for possível é feita a alocação, caso contrário, um próximo dia de acordo com a matriz de preferência será avaliado. Após a primeira alocação a segunda alocação é feita nas linhas 24 a 31 , seguindo o mesmo procedimento da primeira alocação através da função BuscaNoDia. O que difere nos dois blocos de alocação é maneira como a matriz de preferência é percorrida para obtenção do índice da próxima preferência. Na primeira alocação percorre-se a primeira coluna da matriz. Na segunda alocação percorre-se a linha correspondente ao índice da primeira alocação.

Esta função Alocar Disciplina é chamado através do procedimento Alocação Inicial (algoritmo 5.2). No algoritmo 5.2, o primeiro passo para se iniciar a construção de uma solução é carregar todos os dados para o programa. Para este projeto os dados a serem inseridos são os professores e as disciplinas. Como não é feita a alocação de professor, cada disciplina já é inserida com seu respectivo professor, assim como também suas horas/aulas e o período no qual pertence.

No curso de Ciência da Computação da UENF existem matérias fornecidas por outros Centros ou outros Cursos, em que seus horários são pré-estabelecidos, ou seja, não podem sofrer alterações, há necessidade de entrar com essa informação também. Portanto, antes da inserção das disciplinas de acordo com a matriz de preferência, é inserido todas as disciplinas com horário já pré-estabelecidos através da matriz de disciplinas pré-fixadas. Nesta matriz na primeira coluna se encontra a disciplina que tem os horários pré-estabelecidos, e nas próximas colunas o timeslot correspondente a esta disciplina.

Quando inserido os dados de todas as disciplinas deve-se ter, portanto, a informação se esta já foi alocada ou não. Desta forma há também um índice em cada disciplina no qual diz se ela é pré-fixada e já foi alocada, ou se é uma disciplina que vai ser alocada através da inserção da matriz de preferência. Sendo assim as informações associadas a cada disciplina é: o seu respectivo professor, seu número de horas/aulas, O índice que informa se esta já tem horário pré-fixado ou não, e o período a qual pertence.

```
I: Inicio
Inicial
3: T e— Alocar Disciplinas Prefixadas();
4: id O; I/ índice da disciplina
5: enquanto (id NumDisciplinas) faça l/ NumDisciplinas é o número total de disciplinas
se Disciplina(id) não estiver alocada então
6:
T 4— AlocarDiscipIina(id);
fim-se
8:
id 4— id+l ;
9:
10: fim-enquanto;
```

```
11: retorne T;
12: Fim
```

Após a aplicação da heurística construtiva num exemplo com apenas II disciplinas temos um resultado como mostrado na figura 5.4.

Figura 5.4 - Resultado da heurística construtiva do exemplo.

## 5.5 Descrição das Vizinhanças

No presente projeto, cada ponto do espaço de soluções é, naturalmente uma grade de horário completa e válida, embora não necessariamente ótima. Esta solução é adquirida através da inserção de disciplinas obedecendo a matriz de preferência, como explicado na Seção 5.4. Para criar novas soluções que atendam aos requisitos não essenciais, melhorando a sua qualidade, define-se o critério de soluções vizinhas.

Uma solução vizinha é adquirida através do movimento de troca de uma disciplina que se encontra em um timeslot para outro timeslot. Porém só é válido a realização deste movimento se a nova solução também seja uma solução válida.

A utilimção desses movimentos foi dividida em duas etapas, que correspondem a segunda e terceira fase do processo de construção da solução. Na segunda fase este movimento é utilimdo para resolver apenas um dos requisitos não essenciais, o não preencher os horários de 12h às 14h, pois se trata de horário de almoço.

Esta etapa é simples, e consiste em apenas varrer a matriz de timeslot com a solução já obtida na primeira etapa buscando os timeslots dos horários de almoço (timeslots 4, 5, 14, 15, 24, 25, 34, 35, 44, 45) no caso destes estarem ocupados com uma disciplina sem horário prefixado, transferi-la para outro timeslost viável. Essa transferência é feita da seguinte forma: tira a disciplina do horário do almoço e varre a matriz timeslot de forma sequencial a procura de um novo timeslot viável para esta disciplina. Um passo da heurística de melhoria, segunda fase da implementação para se chegar a uma solução final, é ilustrada na figura 5.5 através de uma troca realimda na solução do exemplo anterior da figura 5.4.

Figura 5.5 - Resultado da heurística de melhoria do exemplo.

Uma observação quanto ao exemplo, é que não necessariamente as disciplinas retiradas do horário do almoço serão alocadas sequencialmente como mostra na figura, pois elas são alocadas individualmente e buscam espaços individuais. Neste exemplo coincidiu de ficarem novamente em sequência.

A terceira e última fase na implementação da solução para o problema é realizada através da heurística Simulated Annealing. Realizando movimentos aleatoriamente e um critério de parada, chaga-se a solução final. Esse processo é descrito na próxima seção 5.5.

## 5.6 Metaheurística Simulated Annealing

Partindo de uma solução ao aplicarmos a metaheurística Simulated Annealing, o algoritmo substitui a solução atual por uma solução próxima, (i.e., na sua vizinhança no espaço de soluções) que neste projeto é obtida através do movimento de troca de uma disciplina que se encontra em um timeslot para outro timeslot. Porém esta troca só é aceita e escolhida como nova solução de acordo com a função objetivo e com uma variável T (dita Temperatura por analogia ao processo explicado na seção 4.4.1). Quanto maior for T, maior a componente aleatória que será incluída na próxima solução escolhida. À medida que o algoritmo progride, o valor de Té decrementado, começando o algoritmo a convergir para uma soluçao ótima, necessariamente local. Uma das principais vantagens deste algoritmo é permitir testar soluções mais distantes da solução atual e dar mais independência do ponto inicial da pesquisa.

A principal dificuldade neste trabalho é a definição da função objetivo, pois esta quantifica a qualidade de uma solução de forma que possamos comparar uma solução a outra e dizer qual é melhor. Para o problema da distribuição de horário afirmar que uma grade de horários está melhor que a outra depende do quanto cada grade desrespeita as restrições, podendo ocorrer por tanto muitas grades diferentes que não desrespeitam nenhuma restrição, esta por sua vez não temos como comparar sem criar novos critérios.

Portanto, a função objetivo criada para os teste desta monografia é baseada no excesso de aulas de uma mesma disciplina em um dia; no número de disciplinas que uma turma tem em um determinado dia; no número de dias que um professor contratado vai à universidade; e na disüibuição dos horários dos professores associados. Há penalização na função objetivo quando:

i. há mais de 3 aulas de uma mesma disciplina no mesmo dia;
ii. há duas aulas no mesmo dia, porém não consecutivas;
iii. quando uma mesma turma tem muitas aulas consecutivas, independente da disciplina.
iv. quando um professor contratado precisa ir mais dias à universidade do que deveria ir;
v. quando um professor associado leciona mais de 4 aulas em um dia e há dia da semana que ele não leciona nenhuma disciplina.

As penalizações são calculadas para cada dia, onde para cada requisito quebrado soma-se o valor 1. Os valores de cada dia, portanto, são somados, gerando o valorf(s). Desta forma a função objetivo consegue quantificar os requisitos de qualidade, o que nos leva a conclusão de que os requisitos essenciais não estão sendo avaliados. Isto acontece porque mesmo aplicando 0 algoritmo SA que permite piora da solução, não foi permitido pioras que quebrem os requisitos obrigatórios, ou seja, só é aceita uma solução vizinha válida.

Como o requisito não essencial RNE3, que diz que nao é permitido preencher os horários de 12h às 14h, pois se trata de horário de almoço, já foi atendido através da heurística melhoria, não foi permitida também este tipo de piora. Assim não é permitida uma solução vizinha que quebre esta regra, por isso este requisito não foi avaliado na função objetivo também.

O pseudocódigo do algoritmo Simulated Annealing utilizado neste projeto é apresentado no algoritrno 5.3. Os identificadores utiliados são:

- So —+ Solução Inicial;
- Si —Y Solução da Iteração i;
- S —+ Configuração Final;
- To —4 Temperatura Inicial;
- Ti —+ Temperatura na Iteração i;
- M Número máximo de iterações;
- P Número máximo de Perturbações por iteração;
- L —9 Número máximo de sucessos por iteração;
- a —+ Fator de redução da temperatura;
- (Si) —+ Valor da função objetivo correspondente á Solução Si;
- nSucesso —+ Contador de sucesso em uma iteração;
- i ej Variáveis de controle de Loops.
- Perturba(S) —+ Função que realiza uma perturbação na Solução S;
- Randomiza() Função que gera um número aleatório no intervalo [0,1];

Como na literatura as metodologias adotadas são muito distintas para cada SA os parâmetros variam muito, portanto, os valores utilizados para nos parâmetros M, P, L, To e a foram obtidos através de testes computacionais, como será explicado na próxima seção.

```
I: Inicio
2:
```

```
Ler (So, M, P, L, To, a); // Entrada do Algoritmo
l* Inicialiaçao das variáveis*/
3:
Se—So;
5: 6: Té-To•,
7: Loop principal — verifica se foram atendidas as condições de término do
8: algoritrno*/
9: Repita
II:
Loop Interno — Realização de perturbação em uma iteraçao
12:
Repita
13:
Teste de aceitação de urna soluçao 8/
16:
se (AFi O) ou (exp(-AfifD > Randomiza()) entao
17:
SeSi,•
nSucesso 4— nSucesso + 1 ;
19:
fim-se
21:
Até (nSucesso L) ou (i > P)
22:
Atualização da Temperatura t/
23:
j «3+1 ; P Atualização do Contador de iterações
Até (nSucesso=O) ou )
25:
26: Retorne(S)
27: Fim
```

# 6. Resultados computacionais

## 6.1 Descrição do Estudo de Caso

O problema de programação de horários para o curso de Ciência de Computação da UENF foi dividido em dois subproblemas:

1. Problema de Programaçao de Horários dos Períodos ímpares (PPH-I) e
2. Problema Programação de Horários dos períodos Pares (PPH-P).

Parte-se da hipótese de que não há necessidade de criar uma SOIUÇÃO conjunta para os períodos pares e ímpares uma vez que, como regra geral, a UENF não oferece matrículas referentes a disciplinas de períodos consecutivos no mesmo semestre. Caso isso aconteça, é tratado como um caso excepcional, Os dados para os testes computacionais foram Obtidos através da secretaria do referido curso. Pelos motivos descritos, esses dados foram organizados em dois grupos: períodos impares e pares.

A Tabela I mostra os dados das disciplinas utilizadas para a programação de horários no caso dos períodos ímpares. Esta tabela apresenta 29 disciplinas. A primeira coluna contém o código de identificação da disciplina, A segunda coluna apresenta o nome da disciplina. A terceira coluna mostra um digito binário que indica se a disciplina tem horário fixo (valor I) ou caso contrário (valor O). Somente as disciplinas com digito igual a zero precisaram ser incluídas no problema de alocação de horários. Este dígito é importante pois há sempre a necessidade de verificar durante as mudanças se a disciplina pode ou não sofrer transferência de horário. O número de horas aula da disciplina é mostrado na quarta coluna. A quinta coluna corresponde ao período ao qual a disciplina pertence. A sexta e sétima coluna contém o código de identificação do professor associado à disciplina e 0 seu respectivo nome,

Tabela 6.1 — Dados das disdpllnas de períodos ímpares.

Os dados das disciplinas utilizadas para a programação de horários no caso dos períodos pares são apresentados na tabela 6.2. Esta tabela apresenta 26 disciplinas e possui estrutura idêntica à tabela 6.1

Tabela 6.2 — Dados das disciplinas de períodos pares.

Além dos dados das disciplinas descritos nas tabelas 6.1 e 62. foram utilizadas mais três matrizes de dados complementares: i) matriz de preferência ii) matriz de horários préfixados e iii) matriz de categoria de professor.

A matriz de preferência foi descrita na seção 5.4. A matriz de horários pré- fixados contém as informações das disciplinas que tem horário fixo. Cada linha da matriz contém o código de identificação da disciplina, na primeira coluna e nas colunas subsequentes a identificação dos timeslots pré-fixados a essa disciplina (ver Figura 6.1).

Figura 6.1 — RepresentaçSo da matriz de horários pré-fixados.

A matriz de categoria de professor guarda os índices dos professores e um dígito binário que será igual a um se o professor for contratado e zero se for associados. (ver Figura 6.2).

Figura 6.2 — Representação da matriz de categoria de

Todos os dados de entrada descritos anteriormente foram armazenados em arquivos de texto e lidos pelos algoritmos heurísticos.

## 6.2 Representação da Solução

A estrutura de dados utilizda para representar a solução do problema foi uma matriz de tamanho 50x5, chamada mostrada na figura 6.3. O índice i está associado aos horários disponíveis durante a semana (numerados de 0 a 49). Cada horário possível é chamado de timeslot. Já, o índice j está associado a quantidade de disciplinas que podem ser alocadas em um mesmo timeslot.

Cada timeslot pode ser entendido como um tempo de aula, no presente trabalho este tempo é de uma hora. Vale observar, que um timeslot oferece a informação referente ao dia da semana e hora em que será feita a alocação de uma disciplina. Por exemplo, o timeslot 6 corresponde ao horário de 14h às 15h de segunda-feira. O timeslot 16 corresponde o mesmo horário, porém da terça-feira. O tamanho de j pode variar de acordo com a quantidade de disciplinas alocadas a um mesmo timeslot. Como 0 máximo de disciplinas alocadas a um timeslot no atual estudo de caso é igual ao número de períodos ímpares (ou pares), não é necessário que j seja maior do que 5.

Soluções inviáveis são proibidas, não pode haver mais de uma disciplina de uma mesma turma no mesmo horário, e só existem 5 turmas por período.

Figura 6.3 — Representação dos timeslots.

## 6.3 Resultados Computacionais

Foram implementadas três heurísticas para resolver o problema de programação de horários do curso da Ciência da Computação da UENF. As heurísticas foram codificadas na linguagem de programação C, no ambiente DevC4+ 4.9.92 com compilador GCC e executadas em um computador Core 2 Duo, com processador de 2,2GHz, 4 Gb de memória RAM, com sistema operacional Windows 7, 64 bits.

As heurísticas implementadas foram: i) Heurística construtiva; ii) Heurística de melhoria e iii) Metaheurística Simulated Annealing.

Todas as heurísticas forneceram como resultado uma grade de horários viável, onde todos os requisitos obrigatórios são atendidos.

Na primeira heurística, destinada à construção da solução inicial, as disciplinas foram inseridas obedecendo a matriz de preferência, fornecendo uma solução construtiva. Embora utilize um método de construção bom para uma solução inicial o valor da função objetivo é alto, já que esta heurística só se preocupa com os requisitos essenciais.

Após a obtenção da primeira solução, esta se torna um dado de entrada para a heurística de melhoria, que trabalha com a modificação da solução construtiva através do movimento de troca. A heurística de melhoria realiza o movimento de troca dos horários do almoço a fim de atender ao requisito não essencial RNE3. Após a heurística de melhoria não houve nenhuma melhora da função objetivo. A terceira aplica o algoritmo Simulated Anneling fornecendo uma solução final. É nesta etapa que há melhoria de qualidade da solução, já que o algoritmo usa como critério de iteração a minimização do valor da função objetivo.

Foram necessários testes computacionais para se chegar aos melhores valores para os parâmetros M, P, L, TO e a. A tabela 6.3 mostra os resultados dos testes. Os valores escolhidos foram determinados observando-se a média e o desvio padrão do valor da função objetivo, considerando-se 10 execuções da heurística para certas combinações dos parâmetros. Os testes foram feitos com os dados dos períodos ímpares, porém foram usados como padrão para qualquer teste, incluindo para os períodos pares. De acordo com os testes os valores escolhidos para foram:

- M = 5000
- P = 100
- L- 1000
- a = 0.2
- T0 =200

Tabela 6.3 — Dados das disciplinas de períodos pares.

De acordo com os códigos das disciplinas para os períodos ímpares apresentados na tabela 6.1, o resultado da solução construtiva é apresentado na figura 6.4. A figura 6.5 por sua vez, apresenta a distribuição das disciplinas de acordo com a heurística de melhoria. Para ambas as soluções o valor da função objetivo foi de 206, ou seja, perrnaneceu o mesmo, já que a função objetivo não penalizou horários de almoço.

Figura 6.4 — Solução construtiva: Caso perfodos ímpares.
Figura 6.5 — Solução da Heurística de mehoria: Caso períodos ímpares.

Na figura 6.6 temos o resultado da solução da metaheurística simulated annealing. Após várias execuções o menor valor encontrado da função objetivo foi 10. Não se chegou em momento algum a uma função objetivo de

valor zero. Nenhum caso de piora da função objetivo também foi objetivo, porém, muita das vezes, o valor mesmo diminuindo permanece alto.

Figura 6.6 — Solução da metaheurfstica Simulated Annealing: Caso períodos ímpares.

O resultado da solução construtiva com os códigos das disciplinas dos períodos pares mostrado na tabela 6.2 é mostrado na figura 6.7. A figura 6.8 mostra a solução da heurística de melhoria. Como aconteceu com os períodos ímpares, para ambas as soluções o valor da função objetivo pennaneceu o mesmo, o valor foi 120.

Figura 6.7 — Soluçao construtiva: Caso perfodos pares.
Figura 6.8 — Solução da Heurística de melhoria: Caso períodos pares.

A figura 6.9, é apresentada a solução para o melhor resultado da função objetivo encontrado para os dados dos períodos pares após a aplicação da metaheurística Simulated Anneling. Assim como no exemplo anterior, não houve registro de piora da frnção objetivo e nem de seu valor ter chegado a zero. O valor menor encontrado foi 15.

Figura 6.9 — Solução da metaheurística Simulated Annealing: Caso períodos pares.

Como os resultados apresentados estão com códigos de disciplina, e vários períodos ao mesmo tempo, a compreensão dos mesmos fica prejudicada. Para melhor visualização e análise dos resultados os Apêndices trazem os resultados de cada heurística separados por períodos e com 0 nome da disciplina, assim como seus respectivos professores.

A fim de automatimr o processo de verificação das soluções obtidas em cada heurística, pois é um trabalho longo e propenso a erros se fazê-lo manualmente, uma função auxiliar foi implementada verificando se a solução realmente respeitava todos os requisitos obrigatórios.

### 6.4 Análise da qualidade dos resultados

Após a análise dos resultados da solução construtiva podemos perceber que o resultado se encontra dentro do esperado. As disciplinas foram distribuídas duas a duas sequencialmente, e em dias diferentes e com um intervalo de um dia pelo menos entre elas. Isso era de se esperar pelas sequencias de preferências da matriz de preferência. Como também era de se esperar, as disciplinas se acumularam na segunda e quarta, já que a forrna de inserção do algoritmo é sequencial por período e sempre começa pela segunda-feira. A pesar disso, ainda sim os resultados foram satisfatórios por se tratar da solução inicial.

A heurística de melhoria obteve o resultado desejado, também como se esperava, já que seu objetivo era apenas a retirada das disciplinas que estavam no horário do almoço, e este propósito foi atingido. Contudo, a melhoria na solução objetivo não ocorreu devido a não avaliação do requisito RNE3.

Em relação a metaheurística Simulated Annealing o resultado esperado de minimizar para zero a função objetivo não foi atingido. Muita das vezes a função não foi minimizada para um valor considerado bom. Apesar disso, os resultados foram considerados satisfatórios, já que ela atingiu para a melhor solução encontrada um valor da função objetivo baixo. O que permite esta conclusão são os tempos computacionais atingidos. Metaheurísticas como Simulated Ampaling geralmente atingem tempos computacionais altos, o que não ocorreu neste problema, O maior valor registrado foi de 332 milisegundos. Portanto, mesmo valores altos sendo encontrados aleatoriamente é totalmente viável a repetição da execução da metaheuristicas até que se atinja um valor baixo da função objetivo.

# 7. Considerações Finais

### 7.1 Conclusões

Com resultados dentro do esperado conclui-se que as heurísticas atenderam aos seus propósitos. Porém devido a função objetivo não permitir piora desobedecendo aos requisitos obrigatórios não se chegou em momento algum ao valor zero, que seria em princípio a melhor solução possível. Vale observar no entanto, que dada a natureza do problema, não é possível saber se de fato uma solução com valor zero seria viável.

A heurística de melhoria apesar de cumprir seu propósito não melhorou em nada o valor da função objetivo, já que a fünção objetivo não é penalizada quando uma disciplina é alocada nos horários de 12h às 14h. Contudo isto não influenciou o resultado final. Uma altemativa a forma sequencial de troca seria um modelo que frocasse de forma aleatória.

A heurística construtiva trouxe grande contribuição para resoluções de problemas de Programação de Horários. Esta atendeu a todos os requisitos obrigatórios e distribuiu as disciplinas duas a duas sequencialmente, e em dias diferentes com um intervalo de pelo menos um dia entre elas.

### 7.2 Trabalhos Futuros

O Problema da Programação de Horário é um problema complexo, assim, depois desse estudo muitas questões ficaram em aberto. Apesar da abordagem proposta neste trabalho, o estudo pode ser melhorado. Muitas alternativas de criações de soluções foram expostas após a avaliação dos resultados. A seguir, algumas questões importantes as quais poderiam ser pesquisadas como continuação do trabalho:

- Realisar mais testes com novos dados fictícios, com maiores restrições, como por exemplo mais disciplinas, ou menos professores;
- Para o presente projeto foi realizado apenas um tipo de movimento, que troca uma disciplina de horário. Dois novos tipos de movimentos poderiam ser criados e testados como: a permutação de duas disciplinas, ou a permutação de três disciplinas ;
- Avaliar por completo todos os requisitos obrigatórios e não obrigatórios na função objetivo. E acrescentar novos requisitos, com por exemplo acrescentar o requisito de se evitar aulas sexta a tarde, pois a maioria dos alunos da UENF moram em oufra cidade, ou de ser obrigatório haver pelo menos uma hora de almoço entre os horários de 10h às 14h.
- Na metaheurística Simulated Annealing permitir pioras do tipo que desrespeitem requisitos essenciais.
- Executar a heurística de melhoria após a metaheurítica Simulated Annelaing e verificar se resultados melhores são obtidos.
- Criar um mecanismo que execute o Simulated Annelaing quantas vezes necessárias até se obter uma solução com valor de função objetivo aceitável. Para isto usar o método de reaquecimento.

# Referência Bibliográfica


1. [Arenales2007] Arenales, Marcos; Armentano, Vinicius; Morabito, Reinaldo e Yanasse, Horacio (2007). Pesquisa Operacional para Cursos de Engenharia. Editora Elsevier /Campus-Abepro, la Edição.
2. [Chaves2003] Augusto, Antônio. Modelagem Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios, Monografia para obtenção de Bacharel em Ciência da Computação, Universidade Federal de Ouro Preto, 2003.
3. [Coelho+2004] Santos, Leandro; Neto, Roberto. Colônia de Formigas: Uma Abordagem Promissora para Aplicações de Atribuição Quadrática e Projeto de Layout, XXIV Encontro Nacional de Engenharia de Produção, Florianópolis — SC, de 3 a 5 de novembro de 2004 (ENEGEP 2004).
4. [Cooper+1992] Cooper, Tim; Kingston, Jefrrey. The Solution of Real Instances of Timetabling Problem. Basser Department of Computer Science, The University of Sydney 2006, Australia, 1992.
5. [Costa2003] Pereira, Felipe. Programação de Horários em Escolas via GRASP e Busca Tabu, Monografia para obtenção do Grau de Engenheiro de Produção, Universidade Federal de Ouro Preto, 2003.
6. [Dutra2008] Dutra, Vagner Gonçalves. AI goritmo Genético Aplicado Ao Problema De P-Medianas Capacitado, Monog•afia para obtenção do Grau de Engenheiro de Produção, Universidade Federal de Ouro Preto, 2008.

7. [Fraga2006] Fraga, Marcelo C. Pimentel. Uma metodologia híbrida Colônia de Formigas — Busca Tabu — Reconexão por Caminhos para resolução do Problema de Roteamento de Veículos com Janelas de Tempo, Dissertação de Mestrado para a obtenção do título de Mestre em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica em Minas Gerais, CEFET-MG, 2006.
8. [Góes+2010] Teixeira, Anderson; Costa, Deise; Steiner, Maria. Otimização na programação de horário de professores/turmas; Modelos Matemático, Abordagem Heurística e Método Misto, SISTEMAS & GESTÃO, v. 5, n. 1, 2010, p. 50-66.
9. [Haeser+2008] Haeser, G.; Gomes—Ruggiero, M. Aspectos Teóricos de Simulated Annealing e um Algoritmo duas Fases em Otimização Global, TEMA Tend. Mat. Apl. Comput., 9, No. 3, 2008, pp 395-404.
10. [Kirkpatrick+1983] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by Simulated Annealing, Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.
11. [Lob02005] Lobo, Eduardo. Uma Solução do Problema de Horário Escolar Via Algoritmo Genético Paralelo, Dissertação para obtenção do título de Mestre em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica de Minas Gerias, CEFET-MG, 2005.
12. [Mendes1999] Mendes, Alexandre. Dissertação para obtenção do título de Mestre em Engenharia Elétrica, Universidade Estadual de Campinas, 1999.
13. [Metropolis+1953] Metropolis, Nicholas; Rosenbluth, Arianna W.; Rusenbluts Marshall N. e Teller, Augusta H. Equations of state calculations by fast computing machines, Journal of Chemical Physics, 1953.
14. [Noronha2000] Noronha, T.F. Uma Abordagem sobre Estratégias Metaheurísticas, Projeto Orientado, Universidade Federal do Rio Grande do Norte. Rio Grande do Norte, 2000.
15. [Oliveira2006] César, Adriano. Uso do Algoritmo Genético e Recozimento Simulado para o Problema de Alocação de Salas, Monografia para obtenção de Bacharel em Ciência da Computação, Universidade Federal de Lavras, 2006.
16. [Santos2007] Gambini, Haroldo. Formulações e Algoritmos para o Problema de Programação de Horários em Escolas, Tese para a obtenção do título de Doutor submetida ao Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, 2007.
17. [Silva2005] Silva, Amanda. Estudo e implementação, mediante recozimento simulado, do problema de alocação de salas, Monografia para obtenção de Bacharel em Ciência da Computação, Universidade Federal de Lavras, 2005.
18. [Silva+2005] Silva, Amanda; Sampaio, Rudini e Alvarenga, Guilherme. Uma Aplicação Simmulated Annealing para Problema de Alocação de Salas. NFO COMP (Journal of Computer Science), 2005. 17
19. [Souza2000] Freitas, Marcone. Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas, Tese para obtenção do título de Doutor em Ciências em Engenharia de Sistemas da Computação submetida ao Programa de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, 2000.
20. [Souz+2002] Freitas, Marcone; Costa, Felippe e Guimarães, Irce. Um Algoritmo Evolutivo Híbrido para o Problema de Programação de Horários em Escolas. XXII Encontro Nacional de Engenharia de Produção, Curitiba — PR, 23 a 25 de outubro de 2002 (ENEGEP 2002), pp 1-8.
21. [Souza+2002] Freitas, Marcone; Martins, Alexandre e Araújo, Cássio. Experiências com Simulated Annealing e Busca Tabu na Resolução do Problema de Alocação de Salas, XXXIV Simpósio Brasileiro De Pesquisa Operacional, Rio de Janeiro — RJ, 8 a 11 de novembro de 2002 (SBPO 2002).
22. [Spindler2010] Spindler, Morgana. Uma Proposta de Solução para Problemas de Horário Educacional utilizando Busca Dispersa e Reconexão por Caminhos, Dissertação para a obtenção do grau de Mestre em Computação Aplicada, Universidade do Vale do Rio dos Sinos, 2010.
23. [Subramanian+2011] Subramanian, Anand; Medeiros, José M.; Formiga, Lucídio e Souza, Marcone. Aplicação da Methaurística Busca Tabu ao Problema de Alocação de Aulas a Salas em uma Instituição Universitária. Associação Brasileira de Engenharia de Produção (ABEPRO), Revista Produção Online, v.11, n.1, mar. 2011, pp 54-75.

# Apêndice A - Solução Construtiva

Horários das disciplinas dos períodos ímpares



# Apêndice B - Solução Construtiva

Horários das disciplinas dos períodos pares



# Apêndice C - Heurística de Melhoria

Horários das disciplinas dos períodos ímpares



# Apêndice D - Heurística de Melhoria

Horários das disciplinas dos períodos pares



# Apêndice E - Metaheurística Simulated Annealing

Horários das disciplinas dos períodos ímpares



# Apêndice F - Metaheurística Simulated Annealing

Horários das disciplinas dos períodos pares