CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Curso de Mestrado em Modelagem Matemática e Computacional

Problema de Alocação de Salas em Cursos Universitários: Um Estudo de Caso

Dissertação de Mestrado, submetida ao Curso de Mestrado em Modelagem Matemática e Computacional, como parte dos requisitos para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Aluno: Alan Souza Prado

Orientador: Prof. Dr. Marcone Jamilson Freitas de Souza (UFOP)

Co-orientador: Prof. Dr. Sérgio Ricardo de Souza (CEFET/MG)

Belo Horizonte, agosto de 2014.

Dedico este trabalho às pessoas que mais amo e foram meus maiores incentivadores e motivadores para a conclusão deste trabalho. Obrigado ao meu pai Saul, minha mãe Dulceneia, minha esposa Gabriela e minha amada filha Marina.

Agradecimentos

Agradeço muito a Deus por me permitir estar sempre saudável e cuidar de meu caminho nestas inúmeras viagens necessárias para a realização deste trabalho.

Os últimos anos foram de muito trabalho e empenho na busca de conhecimento, além de muito árduos pela necessidade de conciliação entre minhas atividades profissionais e docentes. Desta forma agradeço imensamente minha amada esposa Gabriela, pela paciência e apoio neste período em que minhas prioridades e atenção aos estudos refletiam em minha ausência.

A minha filhinha Marina, que nasceu durante o período de desenvolvimento deste trabalho, e hoje é a fonte de inspiração de minha vida.

A todos meus familiares e amigos que entenderam minha ausência durante todo o período e sempre me apoiaram.

Ao meu orientador, Prof. Dr. Marcone Jamilson Freitas de Souza, por me aceitar-me como orientando, por sua paciência, incentivo e pelo conhecimento transmitido constantemente.

Ao meu co-orientador, Prof. Dr. Sérgio Ricardo de Souza, por toda honestidade em seu trato comigo, pelo incessante acompanhamento e incentivo na realização das atividades e pelo auxilio de forma tão veemente na realização deste trabalho.

Aos demais professores e colegas do MMC que contribuíram e apoiaram meu desenvolvimento nesta caminhada e hoje passamos por ter laços de amizade. Muito obrigado pelo apoio.

Ao CEFET-MG por me aceitar em seu programa e permitir desenvolver meu intelecto. Agradeço muito o apoio e incentivo.

Gostaria de agradecer ao grande amigo, inspirador e mentor, Prof. Me. Stéfano Barra Gazzola, que permitiu minha qualificação mediante incentivo e ausência semanal das atividades administrativas, além de me proporcionar fazer parte deste grande sonho que é o Grupo Educacional UNIS. Meus mais sinceros agradecimentos, meu respeito e toda minha admiração.

De forma geral agradeço por todos que, direta ou indiretamente, participaram deste importante momento de crescimento em minha vida.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original"

Albert Einstein

Resumo

Este trabalho apresenta um sistema computacional, baseado na metaheurística *Simulated Annealing*, para gerenciar a solução do Problema de Alocação de Salas de uma instituição de ensino superior. Este problema consiste em alocar turmas de disciplinas, com horários já previamente fixados, a salas de aulas distribuídas em vários prédios, levando-se em conta um conjunto de restrições. O algoritmo desenvolvido gera uma solução inicial de forma gulosa e aplica a metaheurística *Simulated Annealing* para fazer seu refinamento. A exploração do espaço de soluções é feita por meio de cinco movimentos. O algoritmo foi testado usando-se dados reais de alocação de salas de uma instituição de ensino superior.
Palavras-Chave: Problema de Alocação de Salas, *Simulated Annealing*, Metaheurísticas.

Abstract

This work presents a computational system, based on the Simulated Annealing Metaheuristic, in order to solve a Classroom Assignment Problem in university courses. This problem consists in allocating classes, with schedules previously defined, to classrooms distributed in several buildings, respecting a set of constraints. The developed algorithm generates an initial solution in a greedy way and applies the Simulated Annealing to do its refinement. The exploration of the solution space is taken by five movements. The algorithm was tested using real data of an university.

Keywords: Classroom Assignment Problem, Simulated Annealing, Metaheuristics.

Sumário

Lista de Abreviaturas e Siglas

AG Algoritmos Genéticos

AM Algoritmos Meméticos

CF Colônia de Formigas

GRASP *Greedy Randomized Adaptive Search Procedure*

ICEB Instituto de Ciências Exatas e Biológicas da UFOP

IES Instituição de Ensino Superior

ILS Iterated Local Search

PAP Problema de Alocação de Professores

PAS Problema de Alocação de Salas

PH Problema de Horários

PHC Problema de Horário de Cursos

PHE Problema de Horários Educacionais

PHEx O Problema de Horários de Exames

SA Simulated Annealing

SGBD Sistema de Gerenciamento de Banco de Dados

SO Sistema Operacional

TS Tabu Search

UFOP Universidade Federal de Ouro Preto

VND *Variable Neighborhood Descent* - Descida em Vizinhança Variável

VNS *Variable Neighborhood Search* - Busca em Vizinhança Variável

Lista de Figuras

Lista de Algoritmos

Lista de Tabelas

Capítulo 1

Introdução

1.1 Considerações Iniciais

Semestralmente, inúmeras instituições de ensino superior necessitam distribuir as diferentes turmas de disciplinas a salas de aula, obedecendo a uma série de restrições, como, por exemplo, que as aulas de turmas de mesmo curso e período estejam em uma mesma sala.

Normalmente, essas instituições encontram muitas dificuldades nesta tarefa, visto a complexidade envolvida, o que inviabiliza sua solução de forma manual. De fato, dado o número elevado de combinações possíveis de alocações, uma solução manual normalmente não consegue contemplar muitos requisitos exigidos, gerando insatisfação por parte dos professores e alunos neste processo.

A distribuição de aulas previamente definidas com horários estabelecidos, atendo-se a diversas particularidades relacionadas a espaço físico, possibilidade de acesso, infraestrutura e recursos necessários, caracteriza, então, o Problema de Alocação de Salas (PAS) (Schaerf, 1999).

O PAS pode ser considerado como um subproblema do Problema de Programação de Horários de Cursos Universitários (*Course University*) (Bardadym, 1996), problema este que consiste em definir tanto os horários das turmas de disciplinas quanto as salas para essas turmas. Na prática, entretanto, é mais comum que cada departamento da instituição de ensino em tela defina previamente os horários de aula de suas turmas de disciplinas. Assim, a ocorrência do PAS é mais frequente que a do próprio Problema de Programação de Horários de Cursos Universitários.

Em vista do fato de o PAS ser um problema da classe NP-difícil (Even et al., 1976; Carter e Tovey, 1992), ele tem sido normalmente resolvido por meio de técnicas heurísticas, dentre as quais as metaheurísticas Algoritmos Genéticos (Ueda et al., 2001), Busca Tabu (Subramanian et al., 2011) e *Simulated Annealing* (Beyrouthy et al., 2006). Além destes, citam-se algoritmos híbridos, que apresentam a combinação de técnicas, como: Grafo Bipartido mais Algoritmo Húngaro (Politano, 2006) e *Simulated Annealing* mais Busca Tabu (Souza et al., 2002).

Neste trabalho apresenta-se um algoritmo para solução do Problema de Alocação de Salas, que combina uma fase de construção gulosa para gerar uma solução inicial e a metaheurística *Simulated Annealing* para melhorar essa solução na fase de refinamento. Utilizou-se, para fins de validação da solução, um ambiente real, caracterizado pelos ambientes multiprediais da Universidade Federal de Ouro Preto

(UFOP), abrangendo turmas distribuídas em dois prédios.

1.2 Objetivos

Nesta Seção são apresentados os objetivos deste trabalho.

1.2.1 Objetivo Geral

Este trabalho tem por objetivo geral o desenvolvimento de um sistema computacional heurístico para otimizar a alocação de salas do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP).

1.2.2 Objetivos Específicos

Os seguintes objetivos específicos devem ser atingidos para que este objetivo geral seja alcançado:

(i) Vivenciar, *in loco*, o cotidiano dos procedimentos, inerentes ao problema, adotados pelo ICEB.

(ii) Realizar um levantamento de trabalhos da literatura que abordem este mesmo tema e temas relacionados, assim como os métodos utilizados para solucioná-los.

(iii) Modelar o problema a partir dos dados fornecidos pela UFOP.

(iv) Desenvolver e implementar um algoritmo baseado na metaheurística *Simulated Annealing* para resolver o problema.

(v) Apresentar um *software* com interface simples e fácil de usar, que permita solucionar o problema por meio do algoritmo proposto e que permita, também, a interação com o usuário.

(vi) Testar e analisar o algoritmo proposto quanto à qualidade da solução obtida.

1.3 Metologia de Pesquisa

O estudo de conceitos e trabalhos relacionados ao PAS foi realizado em uma revisão da literatura correlata ao tema. O intuito foi aprimorar o entendimento do problema em suas características específicas e avaliar os métodos de solução existentes.

Para o desenvolvimento deste trabalho optou-se por fazer uma pesquisa exploratória, *in loco*, no ICEB da UFOP, objetivando o mapeamento do problema real e a percepção de suas nuances para a associação aos conceitos pré-existentes do PAS.

Após a revisão da literatura e mapeamento realizados foi feito um estudo acerca das metaheurísticas aplicadas à solução do PAS para maior entendimento das definições técnicas e identificação de possíveis aprimoramentos, visando a um melhor desempenho e aderência do modelo aos aspectos específicos do problema em questão.

Encerrado o desenvolvimento do modelo proposto e a definição das técnicas utilizadas, foi implementado o algoritmo heurístico, assim como o sistema computacional com a interface para usá-lo. Esse algoritmo heurístico usa um procedimento para gerar, de forma gulosa, uma solução inicial, e um procedimento metaheurístico para refinar essa solução inicial. Os parâmetros desse algoritmo heurístico foram devidamente ajustados em uma bateria preliminar de testes. O algoritmo heurístico e o sistema computacional resultante foram avaliados e validados utilizando-se um problema-teste real do ICEB.

1.4 Estrutura da Dissertação

A organização do trabalho se dá da seguinte forma.

O capítulo 2 apresenta uma fundamentação teórica acerca da otimização combinatória e uma revisão parcial sobre heurísticas e metaheurísticas utilizadas para a solução de problemas desta natureza.

Uma revisão da literatura direcionada ao Problema de Horário Escolar, características e classificações é apresentada no capítulo 3.

O capítulo 4 caracteriza e modela o problema de alocação de salas (PAS) do Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto.

O algoritmo proposto para a solução do PAS é detalhado no capítulo 5.

A descrição das funcionalidades e características do *software* desenvolvido para tratar o problema estão apresentadas no capítulo 6.

O capítulo 7 apresenta os resultados obtidos.

Por fim, o capítulo 8 traz as conclusões obtidas e trabalhos futuros a partir da realização deste.

Capítulo 2

Fundamentação Teórica

2.1 Considerações Iniciais

Este capítulo apresenta uma fundamentação teórica de temas relacionados à solução do Problema de Alocação de Salas. Tem-se como objetivo estabelecer uma base conceitual sobre otimização combinatória, estruturação de problemas de otimização e técnicas para a solução dessa classe de problemas.

2.2 Otimização Combinatória

Um problema de otimização combinatória pode ser definido por um conjunto finito $E = \{1, ..., n\}$, um conjunto de soluções viáveis $F \subseteq 2^E$ e uma função objetivo $f : 2^E \rightarrow \mathbb{R}$, todos definidos para cada problema específico (Resende et al., 2012).

De acordo com Blum e Roli (2003), um problema de otimização Q pode ser descrito como uma tripla (S, Ω, f), na qual:

1. S é o espaço de busca definido sobre um conjunto finito de variáveis de decisão $X_i, i = 1, ..., n$. Caso essas variáveis tenham domínios discretos, Q é chamado problema de otimização combinatória, e em casos de domínios contínuos, trata-se de um problema de otimização contínua. Ω é um conjunto de restrições entre as variáveis;

2. $f : S \rightarrow \mathbb{R}^+$ é a função objetivo que especifica um valor positivo para cada elemento de S.

Como é sabido, procuram-se soluções para os problemas de otimização combinatória que representem o menor (para problemas de minimização) ou o maior (para problemas de maximização) valor encontrado em uma região factível (que respeite as restrições do problema), alcançando assim o chamado ótimo global. Por outro lado, em modelos práticos, é comum encontrar no espaço de busca, muitos ótimos locais. A presença desses pontos pode atrapalhar o processo de busca, impedindo que a solução ótima global seja encontrada (Barbosa, 2011).

Uma adaptação feita por Chaves (2009), tendo como base o estudo de Weise (2008), pode ser visto na Figura 2.1. Nessa figura é representado um problema de maximização, tendo uma função objetivo f definida em um espaço bidimensional

$S = (S_1, S_2)$. Um ótimo local $\hat{s} \in S$ de f é um elemento com $f(\hat{s}) \geq f(s')$ para todos os vizinhos s' de $\hat{s}$.

Um ótimo global corresponde ao maior valor da função objetivo dentre todos os locais existentes no espaço de busca. A busca de um ótimo global é dificultada pela presença de vários ótimos locais, o que normalmente ocorre em problemas de otimização combinatória.

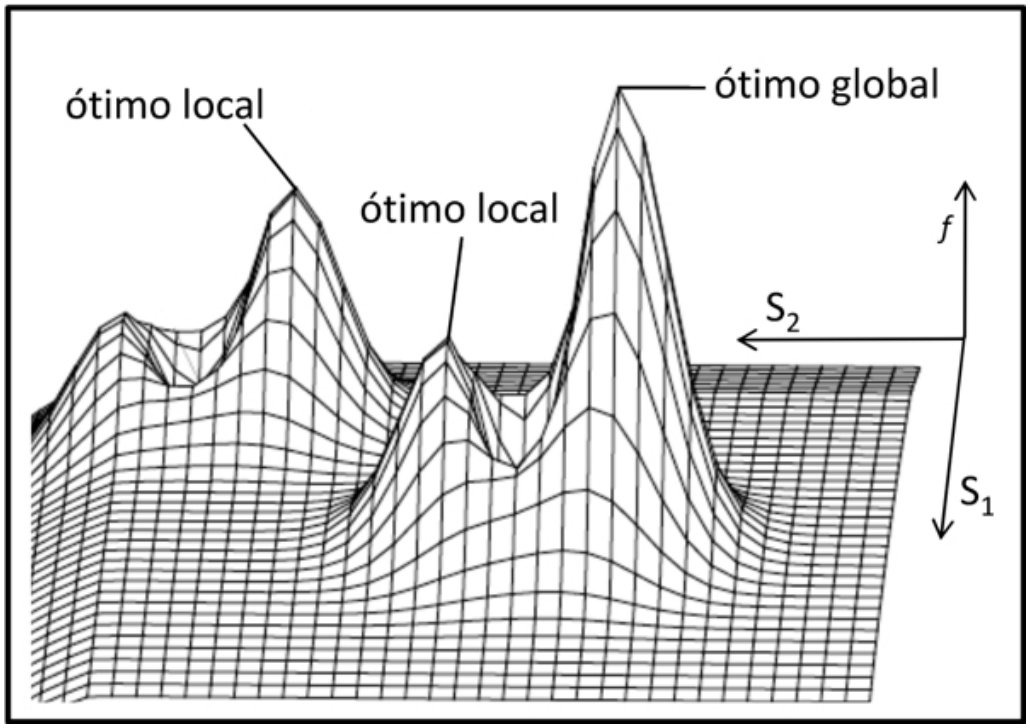


Figura 2.1: Representação de ótimo global e local a partir de uma função bidimencional.(Chaves, 2009).

Para se resolver problemas de otimização combinatória podem ser usados métodos exatos ou métodos heurísticos. Segundo Garey e Johnson (1979), problemas de otimização combinatória são classificados como *NP-difícil*, assim não existe ainda um algoritmo que consiga provar a otimalidade de uma solução em tempo polinomial. Devido a este fato, casos práticos desses problemas são resolvidos normalmente por meio de métodos heurísticos. Segundo Blum et al. (2008), essa escolha ocorre visto que esses métodos produzem, em geral, soluções de boa qualidade em tempo computacional viável, sendo essas soluções consideradas suficientes para a maioria das situações reais.

2.2.1 Vizinhança

Vizinhança de uma solução é o conjunto de todas as soluções factíveis geradas a partir de transformações aplicadas sobre esta solução, transformações estas denominadas movimentos.

Seja S o espaço de busca de um problema de otimização, f a função objetivo a ser minimizada e $s \in S$ uma solução do espaço de soluções do problema. Então, uma vizinhança $N(s)$ é um conjunto de soluções s' formadas pela aplicação de uma

transformação m à solução s. Cada solução $s' \in N(s)$ é chamada de vizinho de s (Souza, 2000).

Segundo Barbosa (2011), a alteração de um atributo, ou conjunto de atributos de uma solução, caracteriza um movimento. Pode-se exemplificar os movimentos como operações de trocas, inserções e/ou remoções de atributos da solução corrente.

2.3 Heurística

O termo heurístico provém do grego *heuriskein* = descobrir, do mesmo radical que deu origem à palavra *heureca*, imortalizada pelo filósofo grego Arquimedes. Uma heurística é um procedimento algorítmico desenvolvido por meio de um modelo cognitivo, usualmente por meio de regras baseadas na experiência dos desenvolvedores. Ao contrário dos métodos exatos, que buscam encontrar uma forma algorítmica de achar uma solução ótima por meio da combinação ou busca de todas as soluções possíveis, as heurísticas normalmente tendem a apresentar um certo grau de conhecimento acerca do comportamento do problema, gerando um número maior de soluções (Cordenonsi, 2008).

As heurísticas seguem modelos intuitivos à procura de uma boa solução para o problema através de tentativa e erro, levando em conta um custo computacional admissível, sem garantias e análises precisas da proximidade da solução ótima. O conhecimento empírico sobre determinado problema é a base para a averiguação da qualidade de uma solução encontrada.

Segundo Noronha (2000), o sucesso de uma heurística está diretamente ligado aos seguintes fatores:

- adaptação a instâncias especiais;
- mecanismos para escapar de ótimos locais;
- estrutura do problema;
- estrutura eficiente de dados;
- pré-processamento;
- boas técnicas para construir soluções iniciais;
- procedimentos de reinicialização;
- melhoria de solução por meio de busca local;
- aleatorização controlada;
- diversificação de busca quando nenhuma melhoria adicional parece possível;
- intensificação da busca em regiões promissoras.

Evans e Minieka (1992) classificam as heurísticas em duas classes:

- heurísticas construtivas: são aquelas que constroem uma solução passo a passo, adicionando um componente dessa solução por vez;

- heurísticas de refinamento: são aquelas que partem de uma solução construída (solução completa) e a melhoram por meio de uma sequência de mudanças.

Uma considerável desvantagem apresentada pelas heurísticas é a dificuldade em se escapar das armadilhas dos ótimos locais (Kelly, 1996).

2.3.1 Heurísticas construtivas

As heurísticas construtivas têm como principal finalidade apresentar uma solução inicial para o problema. Para sua construção parte-se de uma solução vazia e objetiva-se a construção de uma solução completa. A técnica consiste na inserção incremental de elemento por elemento, sendo que, a cada passo um elemento é inserido na solução parcial de acordo com a técnica heurística utilizada e a característica do problema. A escolha do elemento a ser inserido está, geralmente, vinculada a uma determinada função que avalia sua contribuição para a solução.

Nas heurísticas clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa, que estima o benefício da inserção de cada elemento, e somente o "melhor" elemento é inserido a cada passo (Gaspar-Cunha et al., 2012).

Algoritmo 1: Heurística de Construção Gulosa

Entrada: $E = \{$conjunto de todos os candidatos$\}$
Saída: Solução s

```
1  inicio
2      s ← 0;
3      C ← E;
4      enquanto (| C |> 0) faça
5          Selecionar c*;            /* Melhor elemento de C */ ;
6          s ← s ∪ {c*} ;
7          Retirar {c*} de C;
8      fim
9      retorna s;
10 fin
```

O Algoritmo 1 apresenta um procedimento de construção gulosa. Ele recebe como parâmetro de entrada os elementos a serem inseridos na solução inicial. Inicialmente, parte-se de uma solução inicial s vazia e de um conjunto C de candidatos formados pelos elementos que comporão a solução. A cada passo do algoritmo busca-se o elemento com o melhor valor de avaliação. Esse elemento é, então, inserido na solução parcial s e retirado da lista C.

Outra técnica comum utilizada nas heurísticas construtivas é o método aleatório. Nesse método, o elemento que será inserido na solução parcial é escolhido aleatoriamente dentre os elementos que ainda estão fora dela. De acordo com Souza (2014), testes empíricos mostram que as soluções geradas por esse método geralmente são de baixa qualidade e necessitam de um maior esforço na fase de refinamento.

Um algoritmo construtivo aleatório pode ser representado alterando-se a linha 5 do Algoritmo 1. A alteração consiste em selecionar aleatoriamente um elemento da

lista C de candidatos e não o de melhor avaliação.

Em geral, nos problemas considerados difíceis, as heurísticas de construção não produzem boas soluções e normalmente são aplicadas juntamente com heurísticas de refinamento ou metaheurísticas (Barbosa, 2011).

2.3.2 Heurísticas de refinamento

Essas técnicas consistem em melhorar uma solução a partir da busca iterativa por soluções vizinhas que tenham um valor da função de avaliação melhor.

Três métodos heurísticos de refinamento comumente utilizados são o Método de Descida, o Método de Descida Randômica e o *Variable Neighborhood Descent* - VND, os quais são descritos a seguir.

2.3.2.1 Método de Descida

O Método de descida consiste em avaliar todos os possíveis vizinhos a partir de uma solução s, movendo-se sempre para a melhor vizinho encontrado desde que apresente melhora. O método finaliza quando a solução corrente não é mais passível de melhora, caracterizando a solução final como um ótimo local para a vizinhança analisada.

O Algoritmo 2 representa o Método da Descida para um problema de minimização da função de avaliação $f()$, a partir de uma solução inicial s e uma vizinhança $N()$.

Algoritmo 2: Método de Descida

Entrada: Função de Avaliação - $f()$
Entrada: Vizinhança - $N()$
Entrada: Solução inicial - s
Saída: Solução s

```
1 inicio
2     V = {s' ∈ N(s) | f(s') < f(s)};
3     enquanto (| V |> 0) faça
4         Selecione s' ∈ V, onde s' = min{f(s') | s' ∈ V};
5         s ← s';
6         V = {s' ∈ N(s) | f(s') < f(s)};
7     fim
8     retorna s;
9 fin
```

2.3.2.2 Método de Descida Randômica

Este método caracteriza-se basicamente por analisar um vizinho qualquer de uma solução corrente e mover-se para ele se, e somente se, ele representar uma solução melhor que a corrente. Este procedimento é interrompido após um número fixo de iterações sem melhora no valor da melhor solução conhecida até então.

O Algoritmo 3 apresenta o Método de Descida Randômica para um problema de minimização da função de avaliação $f()$, a partir de uma solução inicial s e uma

vizinhança $N()$. IterMax representa o número máximo de iterações sem melhora no valor de $f(s)$.

Algoritmo 3: Método de Descida Randômica

Entrada: Função de Avaliação - $f()$
Entrada: Vizinhança - $N()$
Entrada: Solução inicial - s
Entrada: Máximo de iterações - $IterMax$
Saída: Solução s

```
1  inicio
2      Iter ← 0;
3      enquanto (Iter < IterMax) faça
4          Iter ← Iter + 1;
5          Selecione aleatoriamente s' ∈ N(s);
6          se (f(s') < f(s)) então
7              Iter ← 0;
8              s ← s';
9          fim
10     fim
11     retorna s;
12 fin
```

2.3.2.3 *Variable Neighborhood Descent* - VND

O método de Descida em Vizinhança Variável (VND, das iniciais em inglês *Variable Neighborhood Descent*) foi apresentado por Mladenović e Hansen (1997). Este método de busca local tem como característica a exploração do espaço de busca por meio de trocas sistemáticas da estrutura de vizinhanças. O resultado dessas trocas somente é aceito quando foram geradas soluções melhores que a solução corrente. Neste caso, retorna-se à primeira estrutura; caso contrário, passa-se à próxima vizinhança. O método termina quando não forem encontradas soluções melhores em nenhuma das vizinhanças exploradas, situação que caracteriza um ótimo local com relação a todas as vizinhanças utilizadas. São utilizados três princípios empíricos básicos na concepção do método:

- Um ótimo local com relação a uma vizinhança não é necessariamente um ótimo local relativo à outra vizinhança;
- Um ótimo global é um ótimo local com relação a todas as possíveis vizinhanças;
- Para muitos problemas, ótimos locais com relação a uma ou mais vizinhanças são relativamente próximos.

O Algoritmo 4 mostra o pseudocódigo deste método para um problema de minimização da função f, objetivando refinar uma solução s a partir de um conjunto de r diferentes vizinhanças $N = \{N^{(1)}, N^{(2)}, N^{(3)}, ...N^{(r)}\}$.

Algoritmo 4: Descida em Vizinhança Variável - VND

Entrada: Função de Avaliação - $f()$
Entrada: Vizinhança - $N()$
Entrada: Quantidade de Vizinhanças - r
Entrada: Solução - s
Saída: Solução s

```
1  inicio
2      k ← 1;                                   /* Vizinhança corrente */
3      enquanto (k ≤ r) faça
4          Encontre o melhor vizinho s' ∈ N^(k)(s) ;
5          se (f(s') < f(s)) então
6              s ← s';
7              k ← 1;
8          senão
9              k ← k + 1;
10         fim
11     fim
12     retorna s;
13 fin
```

Souza (2014) afirma que a busca pelo melhor vizinho, representada pela linha 4 do Algoritmo 4, pode gerar um alto custo computacional. Sendo assim, é comum realizar a busca por outros métodos como: Método de Primeira Melhora (*First Improvement Method*), Método de Descida Randômica (*Random Descent*) e ainda outra alternativa que é considerar a exploração apenas em um certo percentual da vizinhança, isto é, procurar o melhor vizinho somente em $PercViz$ % de uma vizinhança $V \subseteq N^{(k)}(s)$, sendo $PercViz$ um parâmetro do método.

2.4 Metaheurísticas

De acordo com (Kelly, 1996), a grande desvantagem das heurísticas está na dificuldade em se escapar das armadilhas dos ótimos locais, o que deu origem ao desenvolvimento das chamadas metaheurísticas. Essas técnicas, ao contrário das heurísticas convencionais, aplicam estratégias de alto nível para explorar o espaço de busca, e apresentam mecanismos para ir além da otimalidade local (Souza et al., 2002).

Diferente das heurísticas clássicas, que eram desenvolvidas para atender problemas bastante particulares, as metaheurísticas são heurísticas de caráter geral. As metaheurísticas se fundamentaram em conceitos relativamente genéricos, como os de vizinhanças de pontos no espaço de soluções discretas, ou de trajetórias conectando dois pontos, ou ainda na organização de informação adquirida a respeito de regiões desse espaço de soluções, permitindo a construção de algoritmos eficientes com características tanto de busca local quanto de busca global (Takahashi e Gaspar-Cunha, 2012).

Glover e Kochenberger (2003) definem metaheurística como sendo métodos de resolução que orquestram uma interação entre procedimentos de melhoria local e diferentes estratégias de busca, para criar um procedimento capaz de escapar de ótimos locais e realizar uma pesquisa mais robusta no espaço de soluções.

A diferenciação entre as metaheurísticas se dá normalmente pela técnica utilizada para explorar o espaço de busca bem como para escapar das armadilhas dos ótimos locais. A maneira pela qual a metaheurística explora o espaço de busca a classifica em duas categorias: busca local ou busca populacional.

- Busca local: Consiste em melhorar uma solução a partir da navegação pela vizinhança dessa solução. Como tal, utiliza movimentos para se explorar o espaço de busca. Estes movimentos são aplicados a cada passo em uma solução corrente para gerar outra solução em sua vizinhança. As características da vizinhança variam de acordo com o problema explorado.

- Busca populacional: Consiste em partir de um conjunto de soluções, e aplicar sobre elas um conjunto de operadores com o intuito de melhorá-las.

De acordo com Talbi (2009), foi considerável o aumento, nos últimos anos, do interesse no desenvolvimento de algoritmos híbridos, assim chamados os algoritmos que combinam metaheurísticas com procedimentos exatos. Blum et al. (2008) afirmam que os algoritmos híbridos podem proporcionar um comportamento mais eficiente e uma maior flexibilidade quando se trata de problemas do mundo real e em larga escala.

2.5 Metaheurísticas Clássicas

Em seguida serão apresentadas, separadamente pela característica exploratória do espaço de busca, algumas metaheurísticas clássicas mais conhecidas.

2.5.1 Métodos baseados em Busca Local

2.5.1.1 *Variable Neighborhood Search* - VNS

Este método foi proposto por Hasen e Mladenovic (1999) e consiste em explorar vizinhanças gradativamente mais distantes da solução atual. Como característica, procura produzir vizinhos de forma aleatória e, somente havendo melhora, move-se para a nova solução. Inclui-se também no método uma busca local a ser aplicada na solução corrente que, originalmente, utiliza o método VND.

O Algoritmo 5 mostra o pseudocódigo do VNS. Nesse algoritmo, s' é um vizinho aleatorio de uma solução s na vizinhança $N^{(k)}(s)$. s' é submetido a uma busca local; caso o ótimo local s'' resultante seja melhor que o da solução corrente s, a busca continua a partir de s'', recomeçando-se a exploração a partir da primeira estrutura de vizinhança $N^{(1)}(s)$. Caso contrário, a busca continua a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$ (Souza, 2014).

Algoritmo 5: *Variable Neighborhood Search* - VNS

Entrada: Critério de Parada - CP
Entrada: Função de Avaliação - $f()$
Entrada: Número de Estruturas de Vizinhança - r
Entrada: Solução Inicial - s_0
Saída: Solução s

```
1  inicio
2      s ← s_0;                                   /* Solução corrente */
3      enquanto (CP não satisfeito) faça
4          k ← 1;                    /* Tipo de Estrutura de Vizinhança */
5          enquanto (k ≤ r) faça
6              Gere um vizinho qualquer s' ∈ N^(k)(s);
7              s'' ← BuscaLocal(s');
8              se (f(s'') < f(s)) então
9                  s ← s'';
10                 k ← 1;
11             senão
12                 k ← k + 1;
13             fim
14         fim
15     fim
16     retorna s;
17 fin
```

2.5.1.2 *Greedy Randomized Adaptive Search Procedure* - GRASP

Greedy Randomized Adaptive Search Procedure - GRASP é uma metaheurística proposta por Feo e Resende (1995), que apresenta as duas fases a seguir, as quais são aplicadas iterativamente:

1. fase de construção, em que uma solução é construída via um procedimento guloso-aleatório;

2. fase de busca local, em que um ótimo local da vizinhança da solução construída é pesquisado.

Ao final a melhor solução encontrada é apresentada como resultado.

O Algoritmo 6 ilustra o pseudocódigo do método GRASP básico para a solução de um problema de minimização.

Algoritmo 6: Pseudocódigo *GRASP* básico

Entrada: Critério de Parada - CP
Entrada: Função de Avaliação - $f()$
Entrada: Função Adaptativa Gulosa - $g()$
Entrada: Estruturas de Vizinhança - $V()$
Entrada: Parâmetro de Aleatoriedade - α
Saída: Solução s^*

```
1  inicio
2      f(s*) ← ∞;
3      enquanto (CP não satisfeito) faça
4          s ← ConstrucaoGulosoAleatorio(g(·), α);
5          s' ← BuscaLocal(f(·), V(·), s);
6          se (f(s') < f(s*)) então
7              s* ← s';
8              f(s*) ← f(s');
9          fim
10     fim
11     retorna s*
12 fin
```

A primeira fase do método *GRASP* tem por objetivo produzir um conjunto diversificado de soluções iniciais de boa qualidade para a busca local. Esse propósito é alcançado pela adição de aleatoriedade à fase de construção. O intuito desta fase é conciliar as qualidades de uma construção gulosa com as qualidades de uma construção aleatória. Os elementos que estão fora da solução são inseridos, um a um, a cada iteração, sendo que sua inclusão é avaliada de acordo com sua contribuição por meio de uma função adaptativa $g(\cdot)$, regulada por um parâmetro α que controla o nível de aleatoriedade do método. Caso esse parâmetro assuma o valor 0(zero), a solução torna-se totalmente gulosa e atribuindo-se o valor 1(um) a solução torna-se totalmente aleatória. A função é dita adaptativa porque os benefícios associados com a escolha de cada elemento são atualizados a cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior (Souza, 2014). Os melhores elementos, segundo esta função, são inseridos em uma lista restrita de candidatos (LRC). Desta lista, aleatoriamente escolhe-se o elemento que será incrementado à solução parcial naquela iteração.

O Algoritmo 7 descreve a fase de construção do *GRASP*.

Algoritmo 7: Construção *GRASP*

Entrada: Função Adaptativa Gulosa - $g()$
Entrada: Parâmetro de Aleatoriedade - α
Saída: Solução s

```
1  inicio
2      s ← 0;
3      Inicialize o conjunto C de candidatos;
4      enquanto (C ≠ 0 ) faça
5          g(t_min) = min{g(t) | t ∈ C};
6          g(t_max) = max{g(t) | t ∈ C};
7          LRC = {t ∈ C | g(t) ≤ g(t_min) + α(g(t_max) − g(t_min))};
8          Escolha aleatoriamente um elemento t ∈ LRC;
9          s ← s ∪ {t};
10         Atualize o conjunto C de candidatos;
11     fim
12     retorna s
13 fin
```

2.5.1.3 *Iterated Local Search* - ILS

O ILS, proposto por Lourenço et al. (2002), é um método com uma estratégia exploratória que realiza uma busca local em uma solução inicial até encontrar um ótimo local. A partir desta solução, perturbações são realizadas e a busca local reiniciada conforme ilustra a Figura 2.2 para um problema de minimização. As perturbações realizadas nos ótimos locais devem apresentar características suficientes para a exploração de novos espaços diferentes do atual; no entanto, deve ser calibrada para evitar reinícios aleatórios e preservar características deste ótimo local.

Quatro componentes são necessários para se especificar o ILS (Gaspar-Cunha et al., 2012):

- Realizar um procedimento *GeraSolucaoInicial()*, que gera uma solução inicial s_0 para o problema;
- Posteriormente um procedimento *BuscaLocal()*, que retorna uma solução possivelmente melhorada s'';
- Um Procedimento *Perturbacao()*, que modifica a solução corrente s guiando a uma solução intermediária s';
- Por fim um procedimento *CriterioAceitacao()*, que decide de qual solução a próxima perturbação será aplicada.

Souza (2014) afirma que o sucesso do ILS está centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha do método de busca local, das perturbações e do critério de aceitação. Em princípio, qualquer método de busca local pode ser usado, mas o desempenho do ILS com respeito à qualidade da solução final e a velocidade de convergência depende fortemente do método escolhido.

O Algoritmo 8 apresenta o pseudocódigo do ILS para um problema de minimização.

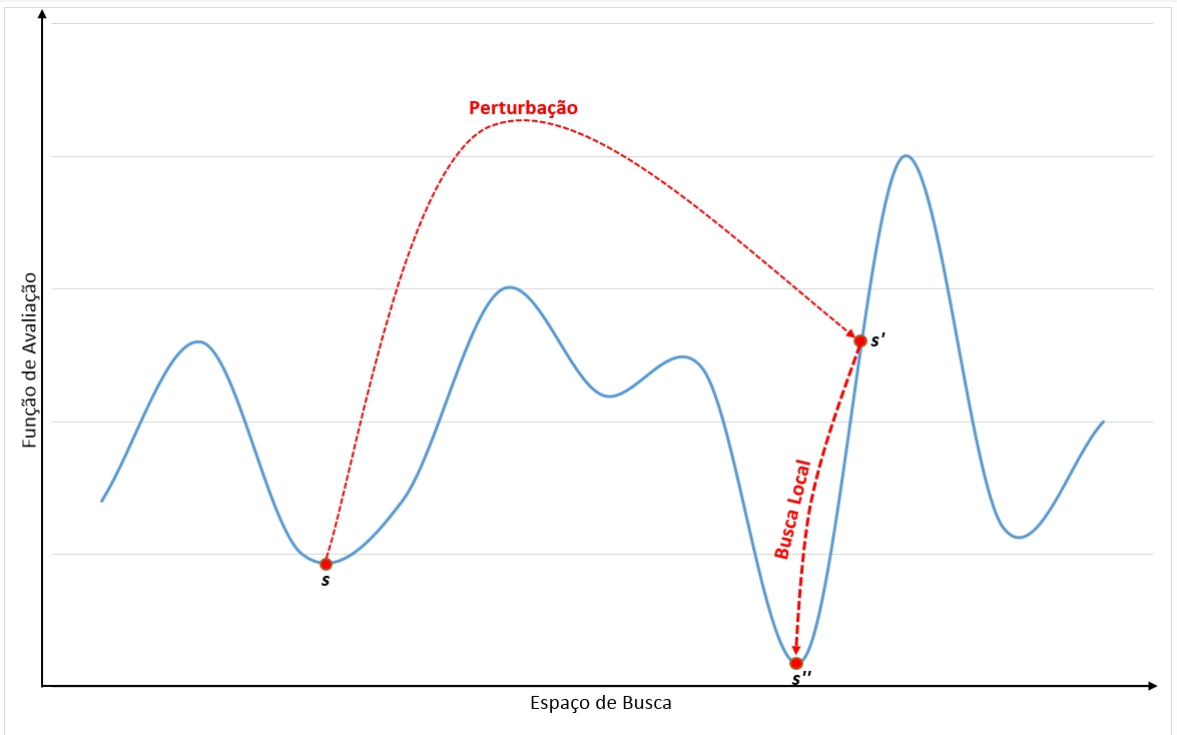


Figura 2.2: Funcionamento do ILS

Algoritmo 8: ILS básico - Problema de Minimização

Entrada: Critério de Parada - CP
Entrada: Função de Avaliação - $f()$
Saída: Solução $s*$

```
1  inicio
2      s_0 ← GeraSolucaoInicial();
3      s ← BuscaLocal(s_0);
4      s* ← s;                                  /* Melhor solução obtida */
5      enquanto (CP não satisfeito) faça
6          s' ← Perturbacao(s);
7          s'' ← BuscaLocal(s');
8          se (f(s'') < f(s*)) então
9              s* ← s'';
10         fim
11         s ← CriterioAceitacao(s, s'');
12     fim
13     retorna s*;
14 fin
```

2.5.1.4 *Tabu Search* - TS

Método proposto por Glover (1986), o *Tabu Search* é uma metaheurística que consiste em mover-se da solução corrente sempre em direção ao melhor vizinho, independentemente do fato de este vizinho ser ou não melhor que a solução corrente.

A Figura 2.3 representa seu funcionamento.

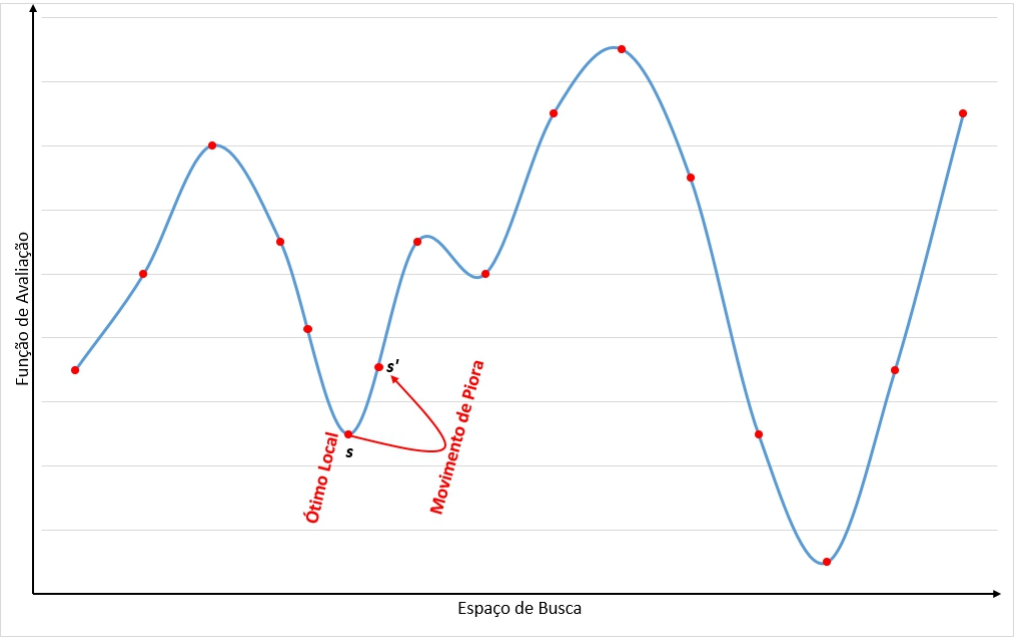


Figura 2.3: Funcionamento do TS

A aceitação de movimentos de piora está direcionada a escapar dos ótimos locais. Esta estratégia ocasionalmente poderia direcionar para o fenômeno conhecido como ciclagem, que consiste na "volta" a uma solução gerada anteriormente, que no caso de um ótimo local, após um movimento de piora, isto sempre ocorreria. Para evitar que a ciclagem aconteça, a técnica mais comumente utilizada é a criação de uma *Lista Tabu*(|L|) composta por movimentos proibidos. Tipicamente, a lista L contém movimentos reversos aos últimos movimentos realizados e funciona como uma fila FIFO (*First-In-First-Out*).

A lista tabu se, por um lado, reduz o risco de ciclagem (uma vez que ela garante o não retorno, por |T| iterações, a uma solução já visitada anteriormente); por outro, também pode proibir movimentos para soluções que ainda não foram visitadas. Assim, existe também uma função de aspiração, que é um mecanismo que retira, sob certas circunstâncias, o status tabu de um movimento (Souza, 2014).

O Algoritmo 9 mostra o pseudocódigo do método para um problema de minimização da função f.

Algoritmo 9: *Tabu Search* básico - Problema de Minimização

Entrada: Função de Avaliação - $f()$
Entrada: Vizinhança - $V()$
Entrada: Função de Aspiração - $A()$
Entrada: Conjunto de soluções vizinhas testadas em cada iteração - SVT
Entrada: Número máximo de iterações - $TSMax$
Entrada: Solução inicial - s
Saída: Solução s^*

```
1  inicio
2      s* ← s;
3      Iter ← 0;
4      L ← 0;                                          /* Lista Tabu */
5      Inicia função de Aspiração A;
6      enquanto (Iter ≤ TSMax) faça
7          Iter ← Iter + 1;
8          Seja s' ← s ⊕ m o melhor elemento de SVT ⊂ V(s) tal que o
           movimento m não é tabu (m ∉ L) ou a condição de aspiração de
           s' (A(s')) é atendida;
9          Atualiza L;
10         s ← s';
11         se (f(s) < f(s*)) então
12             s* ← s;
13             Iter ← 0;
14         fim
15         Atualiza A;
16     fim
17     retorna s*;
18 fin
```

2.5.1.5 *Simulated Annealing* - SA

O método *Simulated Annealing (SA)*, proposto por Kirkpatrick et al. (1983), se inspira no comportamento do processo de recozimento físico (*physical annealing*) de sólidos com fundamentação na termodinâmica. Este processo considera o aquecimento de materiais a altas temperaturas e logo em seguida seu resfriamento de forma lenta. No decorrer deste processo físico o material passa por vários estados possíveis sendo que, se o resfriamento for devidamente realizado obtém-se uma estrutura livre de imperfeições, fenômeno identificado como "estado de mínima energia".

Pereira e Vasconcelos (2012) caracterizam o recozimento simulado como uma metaheurística inspirada no processo físico de recozimento de um sólido para obtenção de estados de baixa energia na área da física da matéria condensada. Esse processo consiste em aquecer o sólido até atingir sua temperatura de fusão, para que a matéria passe do estado sólido para o líquido; posteriormente, a temperatura deve ser lentamente diminuída para evitar estados metaestáveis, objetivando obter a matéria no estado cristalino, ou seja, com energia mínima.

Algoritmo 10: Pseudocódigo do *Simulated Annealing*

Entrada: Solução Inicial - s_0
Entrada: Temperatura Inicial - T_0
Entrada: Número de iterações em T - $SAMax$
Entrada: Taxa de resfriamento - α
Saída: s^*

```
1  inicio
2      s ← s_0;
3      s' ← s;                          /* Melhor solução obtida até então */
4      T ← T_0;
5      IterT ← 0;
6      enquanto (T > 0) faça
7          enquanto (IterT < SAmax) faça
8              IterT ← IterT + 1;
9              Gerar vizinho qualquer s' ∈ N(s);
10             Δ = f(s') − f(s);
11             se (Δ > 0) então
12                 s ← s';
13                 se (f(s' > f(s*)) então
14                     s* ← s';
15                 fim
16             senão
17                 Tome x ∈ [0, 1];
18                 se (x <e^{−Δ/T}) então
19                     s ← s';
20                 fim
21             fim
22         fim
23     fim
24     T ← α × T;
25     IterT ← 0;
26     retorna s*
27 fin
```

Na analogia com um problema de otimização combinatória, o estado de mínima energia é a melhor solução encontrada para o problema, a temperatura é um parâmetro de controle, um estado possível é uma solução viável e a função objetivo é o nível de energia envolvida.

Segundo Souza et al. (2002), o procedimento em si consiste em um loop principal que gera a cada iteração um único vizinho (da solução corrente s) s', aleatoriamente. Em análise comparativa, obtendo s' resultado melhor que s, este é aceito e passa a ser considerado a solução mais viável até o momento, assumindo assim o lugar de s. A solução s' apresentando piora quando comparada a s, por uma quantidade Δ, é aceita obedecendo à probabilidade $e^{-\Delta/T}$, sendo T a temperatura corrente. À temperatura T inicial é atribuída um alto valor, de forma a permitir muitas mudanças no estado de energia. No processo iterativo esse valor é decrescido lenta e gradualmente. O critério de parada do processo consiste em atribuir um valor pequeno ao valor de T,

o que resulta na redução drástica, ou mesmo total, dos movimentos de melhora. O resultado final encontrado é considerado como uma aproximação da solução ótima do problema.

O Algoritmo 10 ilustra o pseudocódigo do método (Dowsland, 1993). Neste algoritmo, s_0 é uma solução inicial, T_0 a temperatura inicial, α a taxa de resfriamento e $SAmax$ o número máximo de iterações para se atingir o equilíbrio térmico.

2.5.2 Métodos baseados em Busca Populacional

2.5.2.1 Algoritmos Genéticos - AG

Os Algoritmos Genéticos – AG (Goldberg, 1989) baseia-se na teoria do processo de evolução natural descrito por Charles Darwin. Neste método considera que, dada uma população, os indivíduos com características genéticas melhores têm maiores chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer.

Nos AGs, cada cromossomo (indivíduo da população) está associado normalmente a uma solução do problema e cada gene está associado a uma componente da solução. Um mecanismo de reprodução, baseado em processos evolutivos, é aplicado sobre a população com o objetivo de explorar o espaço de busca e encontrar melhores soluções para o problema. Este mecanismo de reprodução é aplicado a partir da seleção de dois indivíduos para a aplicação das operações de recombinação e/ou mutação. Na operação de recombinação, os genes de dois cromossomos pais são combinados de forma a gerar um ou dois cromossomos filhos, de sorte que para cada cromossomo filho há um conjunto de genes de cada um dos cromossomos pais. A operação de mutação consiste em alterar aleatoriamente uma parte dos genes de cada cromossomo (Souza, 2000).

O Algoritmo 11 descreve seu pseducódigo.

Algoritmo 11: Algoritmo Genético básico. (Souza, 2000)

```
Entrada: Critério de Parada - CP
inicio
    t ← 0;
    Gere a população inicial P(t);
    Avalie P(t);
    enquanto (CP não satisfeito) faça
        t ← t + 1;
        Gere P(t) a partir de P(t − 1);
        Avalie P(t);
        Defina a população sobrevivente;
    fim
fin
```

2.5.2.2 Algoritmos Meméticos - AM

Essa metaheurística, proposta por Moscato (1989), é uma extensão dos Algoritmos Genéticos. Ela consiste em aplicar uma busca local aos indivíduos, como

parte do processo evolutivo, antes de os submeterem às operações de recombinação e mutação (Moscato e Cotta, 2003).

2.5.2.3 *Ant Colony Optimization Metaheuristic* - ACO

A metaheurística *Ant Colony Optimization* (ACO), ou Colônia de Formigas, foi introduzida por Dorigo (1992). Baseado na natureza, ela simula o comportamento das formigas em suas atividades na colônia, ao gerar um caminho entre a fonte de alimentos e o ninho.

O comportamento natural das formigas direciona-as a procurarem, inicialmente, aleatoriamente por uma fonte de alimento. Encontrando essa fonte elas retornam à colônia deixando um rastro de feromônio, que é uma substância química depositada pelas formigas ao longo da caminhada. Quando outras formigas encontram este rastro elas tendem a seguí-lo depositando também seu feromônio pela trilha; assim, quanto maior é o número de formigas pela trilha, mais atrativa ela se torna.

No Algoritmo ACO, os movimentos se tornam mais desejáveis quanto maior for o nível de feromônio da trilha a ser escolhida. Visando a diversificar a busca e explorar outros ótimos locais, a escolha dos caminhos é feita de forma probabilística. O algoritmo ACO considera, ainda, a aplicação de um fator de evaporação do feromônio em cada caminho.

O Algoritmo 12 ilustra o pseudocódigo do ACO básico.

Algoritmo 12: *Ant Colony Optimization* básico

Entrada: Critério de Parada - CP

```
1 inicio
2     enquanto (CP não satisfeito) faça
3         s = {s_1, s_2, ..., s_n} ← ConstroiSolucao(RastroFeromonio);
4         s' ← BuscaLocal(s_i ∈ S);
5         Atualiza Rastro de Feromônio;
6     fim
7 fin
```

Capítulo 3

Revisão Bibliográfica

Este capítulo se inicia com a apresentação de conceitos relativos ao problema de programação de horários. A seção 3.2 mostra uma classificação desta classe de problemas e a seção 3.4 apresenta trabalhos relacionados ao tema.

3.1 Problema de Horários Educacionais

Michalewicz e Schoenauer (1996) consideram os Problemas de Horários (PH), independentemente de sua natureza, como um dos problemas mais interessantes da Pesquisa Operacional.

Em sua essência, o PH consiste na alocação de um número de eventos a um número finito de períodos, respeitadas as restrições associadas (Burke e Newall, 1999).

Segundo Souza (2000), o problema consiste em fixar uma sequência de encontros entre professores e estudantes em um período prefixado de tempo, em geral uma semana, satisfazendo a um conjunto de restrições de várias naturezas.

Qu (2002) direciona seu trabalho para questões inerentes ao Problema de Horários Educacionais (PHE), caracterizando-o como a atividade de se alocar vários eventos, como aulas, cursos, reuniões e exames, em um número limitado de tempo e local, minimizando as penalidades pelo não cumprimento de restrições inerentes ao problema.

A solução apresentada a um problema desta natureza diz respeito a um quadro de horários, que atenda ao máximo as necessidades e as preferências dos envolvidos (professores e estudantes), reduza os conflitos e, da forma mais eficiente, aloque os equipamentos necessários e as salas disponíveis.

O desenvolvimento do quadro de horários em ambientes educacionais é uma atividade periódica adequada ao regime temporal escolar adotado, seja ele anual, semestral, ou qualquer outro período de tempo. O período de tempo adotado pela entidade de ensino impacta diretamente na possibilidade de obtenção de possíveis soluções manuais para estes problemas. Normalmente, a construção de soluções manuais para esta classe de problemas é uma atividade penosa, embasada em técnicas de tentativa e erro e que demandam muito tempo, além de não satisfazerem diversos aspectos inerentes às necessidades dos envolvidos.

Deve-se ter claro que o PHE é um problema de difícil generalização, seja devido à diversidade de regimes educacionais, que variam de região para região, seja devido

às características de cada instituição de ensino. Desta forma, os sistemas computacionais para a solução desse problema são comumente desenvolvidos para atender a uma instituição específica.

Burke e Silva (2004) destacam a característica combinatória do PHE, dificultando sua resolução a partir da exploração de todas as soluções de seu espaço de busca. Este é problema de complexidade NP-Difícil, corforme Even et al. (1976).

Em virtude de suas características, a resolução do PHE pode ocorrer por meio de várias técnicas como métodos exatos, teoria de grafos, heurísticas e metaheurísticas. O grande número de publicações nos últimos tempos acerca da utilização de metaheurísticas para a solução do PHE comprova a eficiência destas técnicas para alcance de boas soluções ao problema.

3.2 Classificação do Problema de Horários Educacionais

A necessidade de adequação do PHE em virtude das características particulares (tipo de instituição, restrições, etc.) de cada ambiente educacional em que é analisado acarreta na percepção e origem de algumas variações quanto a sua classificação.

Esta dissertação apresenta três classes de PHE introduzidas por Schaerf (1999). Apresenta, também, duas classes adicionais, introduzidas por Carter e Laporte (1997). Estas classes mostram a diversidade de questões associadas ao PHE e, além disso, apresentam a modelagem utilizada para o problema objeto desse trabalho.

3.2.1 Problema de Horários de Escolas (*School Timetabling Problem*)

O Problema de Horários Escolares geralmente está associado ao modelo de ensino em regime seriado, que normalmente é adotado por escolas de nível secundário. Trata da elaboração do quadro de horários para uma programação semanal que se repete durante todo o período de oferta (semanal ou anual), mesmo turno (manhã, tarde ou noite) e com característica do regime seriado no qual, para cada classe, existe um conjunto de professores e disciplinas com carga horária semanal para todos os alunos da turma.

De acordo com Souza (2000), o problema delineia-se basicamente pela existência de um conjunto de turmas, um conjunto de professores e um conjunto de horários reservados para a realização das aulas. As turmas são conjuntos disjuntos de estudantes, que têm um mesmo currículo. Para cada turma há um conjunto de matérias, com suas respectivas cargas horárias, que devem ser cursadas. Para cada professor especifica-se a matéria (ou as matérias), bem como as turmas para as quais o professor lecionará. Um detalhe importante é que a divisão e distribuição dos espaços físicos, no caso, as salas de aula, não compõem o escopo do problema, considerando que está ação já está pré-estabelecida e, com exceção das disciplinas que exijam um espaço especial, o deslocamento entre as salas se dá pelo professor e não pelos alunos.

Como objetivo a ser alcançado para a geração de uma solução viável e aplicável, deve-se considerar que as cargas horários de todas as turmas e disciplinas envolvidas

sejam cumpridas, não exista na solução um mesmo professor alocado em turmas diferentes no mesmo período e as turmas não tenham aulas com mais de um professor em um mesmo período.

Outras variações para o Problema de Horários Escolares bem como maiores detalhes podem ser observados em Schaerf (1999).

3.2.2 Problema de Horários de Cursos (*Course Timetabling Problem*)

O Problema de Horário de Cursos (PHC) usualmente aplica-se em ambientes universitários que têm seu modelo de ensino em regime de créditos. Este regime permite aos alunos escolherem, a cada período, quais disciplinas irão cursar, desde que componham a estrutura curricular do curso, possibilitando uma maior flexibilidade curricular.

O problema consiste em alocar as aulas dos cursos aos horários disponibilizados, respeitando as disponibilidades e capacidades das salas existentes, de forma que nenhum estudante tenha duas ou mais aulas simultaneamente (Souza, 2000).

O funcionamento geral se dá a partir da oferta de disciplinas a cada período (normalmente semestralmente), as quais, em muitos casos, compõem não só a estrutura curricular de um curso, mas de vários. A partir deste cenário, determinada disciplina pode apresentar alunos de diversos cursos, sendo que o deslocamento entre as salas se dá pelos alunos e não pelos professores.

Dentre as particularidades do PHC, vale ressaltar que, diferentemente do Problema de Horários Escolares, as alocações podem extrapolar o turno de oferecimento, não necessariamente sendo ofertada somente em um dos três turnos (manhã, tarde e noite).

A necessidade de mobilidade dos alunos faz com que a análise da alocação das salas componha o escopo destes problemas, abrangendo variáveis como disponibilidade, capacidade, recursos e distância entre salas por exemplo.

As particularidades dos ambientes universitários faz com que diversas variações do problema sejam encontradas na literatura. Estas particularidades usualmente estão relacionadas a questões de espaço físico, deslocamento, análise de sobreposições de horários, dentre outros.

3.2.3 Problema Horários de Exame (*Examination Timetabling Problem*)

O Problema de Horários de Exames (PHEx) consiste na necessidade de, em ambientes universitários, se alocar um conjunto de exames, referente a cada disciplina de um currículo, para cada aluno matriculado em um curso, considerando horários e salas disponíveis.

O principal objetivo do PHEx é evitar a sobreposição de horários para a realização de exames pelos alunos de forma simultânea, ou seja, nenhum aluno pode realizar dois ou mais exames simultaneamente.

Souza et al. (2002) salienta que, apesar da similaridade entre o PHC e o PHEx, a distinção entre eles se dá pela natureza de suas restrições. Destacam-se as seguintes restrições ao PHEx:

- Limitar o número de exames por dia;
- Determinadas disciplinas não devem proceder exames determinadas outras;
- Determinadas exames devem ser realizados em um mesmo horário;
- Determinados exames devem ser realizados em consecutivamente.

3.2.4 Problema de Alocação de Professores (*Teacher Assignment Problem*)

Esta classificação apresentada por Carter e Laporte (1997) considera o conhecimento prévio de algumas informações para sua caracterização, sendo elas relacionadas ao professor (disciplinas lecionadas e disponibilidade) e às aulas (distribuição e horários).

Basicamente, o Problema de Alocação de Professores (PAP) consiste em se alocar a cada disciplina presente no quadro de horários um professor.

3.2.5 Problema de Alocação de Salas (*Classroom Assignment Problem*)

O Problema de Alocação de Salas (PAS) consiste em, a partir de um horário pré-estabelecido, realizar a alocação das aulas a um número fixo de salas, respeitando-se uma série de restrições (Schaerf, 1999).

Para Bardadym (1996), o PAS pode ser considerado como um subproblema, ou até mesmo, parte integrante do PHC (*Course Timetabling Problem*).

3.3 Complexidade, Viabilidade e Otimalidade

A complexidade dos PHE está relacionada e sofre interferência direta de dois fatores, no caso, a viabilidade e a otimalidade.

Determinadas situações de aplicação do PHE consideram que todas as restrições envolvidas sejam atendidas para a geração do horário. Para estes casos, considera-se o problema como um problema de viabilidade, limitando seu objetivo à obtenção de uma solução viável. Cooper e Kingston (1996) classificam este problema como NP-Completo, quanto à sua complexidade.

Segundo Souza (2000), problemas nos quais se deseja encontrar, dentre todos os quadros de horário que satisfazem a um certo conjunto de restrições ditas essenciais (fortes), aquele, chamado ótimo, que minimize uma certa função objetivo, a qual incorpora as restrições ditas não-essenciais (fracas), considera-se como problema de otimização. Eikelder e Willemen (2001) classificam este problema como NP-Difícil quanto à sua complexidade. Muitos autores adotam esta formulação para aplicar técnicas de otimização ao problema de viabilidade, uma vez que, neste caso, o problema de encontrar uma solução viável pode ser considerado como um problema de minimizar uma certa distância de viabilidade.

Santos e Souza (2007) apresentam um importante conceito acerca das restrições fortes e fracas:

- restrições fortes: restrições desse tipo devem ser satisfeitas a qualquer custo, visto que não é possível a implementação de um quadro de horários que não as satisfaça. A restrição mais comum desse tipo é a não ocorrência de conflitos, como exemplo, considere a situação impraticável de uma turma assistir a duas aulas ao mesmo tempo; restrições físicas também são desse tipo - a decisão de em qual sala de aula será dada uma aula de uma turma deve considerar que a sala escolhida terá capacidade suficiente para comportar a turma. Desse modo, as restrições fortes determinam o espaço de busca que será considerado. Somente soluções que respeitam todas as restrições desse tipo são consideradas factíveis;

- restrições fracas: restrições desse tipo são aquelas cuja satisfação é desejável, mas caso não seja possível respeitá-las, pode-se, ainda assim, implementar o quadro de horários. Restrições fracas comuns são, por exemplo, ter aulas de uma mesma turma de disciplina em uma mesma sala durante a semana. O atendimento das restrições fracas é a medida de qualidade de um quadro de horários. O problema considerado consiste em encontrar uma solução que minimize a violação das restrições fracas.

A caracterização das restrições fortes e fracas para o problema está diretamente ligada ao conhecimento claro da relação de importância e relevância dentre as restrições, tendo em vista a construção do horário. Para uma aplicação prática, é necessária uma associação de pesos às restrições, de forma que restrições fortes recebem um alto valor de peso associado e restrições fracas um baixo valor de peso associado. Essas penalizações às restrições constituem a função de avaliação de uma solução.

3.4 Resolução do Problema de Horários Educacionais

A solução de problemas caracterizados como de otimização combinatória, como no caso os PHE, pode ser alcançada por dois métodos: os métodos exatos e os métodos heurísticos, conforme Seção 2.2.

O método a se utilizar está diretamente relacionado com o tamanho da instância do problema que está sendo analisado, tendo em vista que, para a solução por métodos exatos, o espaço de busca deve ser limitado, abrangendo um reduzido número de variáveis e restrições, além de considerar uma resolução em tempo polinomial, o que normalmente não representa o PHE em ambiente real.

A caracterização do PHE como NP-Difícil (Even et al., 1976) direciona sua solução para a utilização de métodos heurísticos/metaheurísticos, pois, diferentemente dos métodos exatos que exigiriam tempos computacionais exponenciais, tornando sua aplicação inviável para problemas reais. Já, conforme diversos trabalhos na literatura, as heurísticos/metaheurísticos têm se apresentado com maior eficiência na busca de uma boa solução (não necessariamente a ótima), considerando importantes aspectos, como tempo de execução e qualidade da solução.

O PHE tem sido objeto de vários estudos durante os últimos cinquenta anos. Uma grande quantidade de trabalhos está disponível na literatura, considerando

diferentes abordagens para sua resolução, formulação e classificação. A seguir é feita uma breve descrição de trabalhos relacionados à resolução do PHC e às suas variações.

Hertz (1992) traz adaptação para a técnica busca tabu para os problemas dos horários. Objetiva em seu trabalho reduzir o número de conflitos por cursos que ocorrem simultaneamente. A relação entre seu estudo e o PAS está em, além de avaliar as restrições do PHC, também levar em conta o agrupamento dos alunos, a compacidade e os requisitos de precedência e restrições geográficas, que estão diretamente ligadas ao espaço físico.

Alfaro e Terán (1998) apresentam um trabalho com um modelo do PAS aplicado ao Instituto Tecnológico e de Estudos Superiores de Monterrey - ITESM (Universidade Mexicana). O problema é caracterizado quanto ao ambiente explorado e, para sua solução, foi aplicada a metaheurística *Simulated Annealing*. Uma interessante consideração quanto às restrições, além da abordagem que é feita pelas restrições já conhecidas e mais utilizadas, é a relevância dada à distância da sala de aula para o escritório do professor, fato necessário devido aos vários edifícios da instituição.

Dowsland (1998) apresenta um artigo que examina as técnicas metaheurísticas *Simulated Annealing* e Busca Tabu na utilização para solução de problemas da vida real do problema PH e PHC. Examina questões a partir de um ponto de vista pessoal, e utilizando estudos de caso surgidos nos setores da educação e de hospitais, mostra que é possível projetar soluções robustas baseadas nessas técnicas. Para isso aborda a família de problemas que são frequentemente úteis para essa avaliação.

Burke et al. (2001) apresentam um estudo para o problema de alocação de espaço dentro de universidades do Reino Unido, direciona sua análise sobre a ótica de o problema apresentar comportamento altamente restrito, apresentando múltiplos objetivos, e afirma que esse comportamento varia muito entre as diferentes instituições. Em seu trabalho propõe a discussão entre três métodos conhecidos aplicados para resolver o problema de alocação de espaço: subidas, *Simulated Annealing* e um algoritmo genético. O objetivo foco do trabalho é investigar a aplicação dessas técnicas, comparando vantagens e desvantagens para conseguir melhor compreensão do problema e propor futuras hibridizações destes e outros métodos.

Ueda et al. (2001) apresentam um algoritmo genético dividido em duas fases, denominado *Two-phase Genetic Algorithm* - (TGA). A primeira fase relaciona uma população com a PHC, e a segunda se refere à alocação de espaço (PAS), objeto deste estudo. Descrevem o problema sobre a ótica da evolução independente das populações avaliadas, gerando um valor de custo a cada indivíduo, que é calculado. Em seguida, vários indivíduos com os menores custos são selecionados a partir de cada população, e esses indivíduos são combinados a fim de calcular os valores de fitness. Em análise, o TGA apresentou resultados melhores comparados ao modelo do Algoritmo genético simples, quando a taxa de utilização das salas é alta.

Souza et al. (2002) afirmam que as metaheurísticas *Simulated Annealing* e Busca Tabu são exemplos de métodos que têm sido utilizados para a solução do PAS, apresentando sucesso em sua resolução. Avaliam a qualidade da técnica como apresentando boa solução de melhora da solução inicial através de técnicas de pesquisa em vizinhança. No artigo relatam as experiências utilizando as duas técnicas e apresentam uma técnica híbrida, combinando o método *Simulated Annealing* + Busca Tabu, e combinando as características mais apropriadas, direcionadas à eficácia da solu-

ção. Avaliam a eficiência comparando a aplicação com versões nativas nos métodos envolvidos.

Castro (2003), utilizando a metaheurística *Simulated Annealing* na fase de refinamento do PAS e uma técnica gulosa para a geração da solução inicial, desenvolve uma ferramenta computacional e afirma bom desempenho da proposta, tendo em vista as soluções finais de boa qualidade quanto da aplicação na Universidade Federal de Ouro Preto - UFOP.

Silva (2005) propõe um estudo para o PAS e também relata a experiência com a utilização do método *Simulated Annealing*. Afirma a eficiência de seus resultados nas análises sobre os resultados obtidos em comparação com os resultados existentes no ambiente explorado, no caso, a Universidade Federal de Lavras - UFLA.

Al-Yakoob e Sherali (2006) apresentam modelos de programação matemática para a atribuição a turmas de membros do corpo docente da Universidade do Kuwait. O trabalho tem por objetivo minimizar a insatisfação individual e coletiva dos professores. Os modelos apresentados foram implementados usando-se o pacote CPLEX-MIP (*Simplex Method of Mixed Integer Programming Problem*) versão 7.5.

Beyrouthy et al. (2006) fundamentam seu trabalho afirmando que alguns estudos revelam que existem os espaços de salas de aulas nas instituições subutilizados. Apresentam que a utilização das salas de aulas, muitas vezes, está entre 20% e 40% da capacidade de utilização, levando-se em conta a variável "assento-hora". Mediante esse fato, existe a necessidade de maior eficiência na alocação de horários, o que é dificultado pela característica que remete sempre a um planejamento a ser feito em curto prazo. Assim objetiva em seu trabalho apresentar métodos que proporcionem melhor planejamento de espaço, mesmo porque, mesmo com um planejamento a prazo maior, não se torna viável essa alocação de forma manual. Assim propõem uma modelagem utilizando a metaheurística *Simulated Annealing* para a solução do PAS.

Dammak et al. (2006) mostram uma formulação para o PAS, considerando uma modelagem embasada em programação linear inteira. O trabalho faz referência a artigos anteriores do autor e, segundo eles, serve de complemento para análise do problema. Os trabalhos anteriores estão relacionados com o problema de exames para atribuição de horários na *Faculty of Economics and Management Sciences of Sfax*, usando várias heurísticas baseadas em coloração de grafos. Os trabalhos anteriores abastecem a análise atual como solução inicial para a solução do problema que, segundo os autores, apresenta soluções interessantes comprovadas mediante a aplicação de outras heurísticas a esse problema.

Subramanian et al. (2011) propõem a automatização do processo de alocação de salas do Instituto de Tecnologia da Universidade Federal da Paraíba, recorrendo a estratégias computacionais que proporcionem soluções de qualidade e baixo custo. Eles utilizaram a metaheurística Busca Tabu para refinamento em movimentos de realocação e troca de aulas entre salas para explorar o espaço de busca. Os autores concluíram que o algoritmo se mostrou eficiente, tendo gerado soluções de alta qualidade quando comparado com as soluções manuais.

Outros importantes trabalhos sobre o assunto podem ser vistos com mais detalhes em (Cooper e Kingston (1996); Carter e Laporte (1997); Burke e Varley (1997); Schaerf (1999); Burke e Newall (1999); Souza (2000); Burke et al. (2001); Eikelder e Willemen (2001); Qu (2002); Santos e Souza (2007));

Capítulo 4

Caracterização do Problema

Este capítulo apresenta o estudo de caso abordado. O problema estudado classifica-se como o PAS derivado do PHC e, conforme Seção 3.2.5, considera que a distribuição das aulas e seus horários já foram pré-estabelecidos.

4.1 Problema Abordado

O problema abordado é o do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). Este instituto tem cursos próprios de graduação e pós-graduação, e oferta, também, aulas do ciclo básico de vários outros cursos de graduação da instituição. No instituto também são ofertadas aulas de disciplinas de outros institutos, como por exemplo, dos departamentos de letras e educação, para os cursos de licenciatura do instituto. As aulas ocorrem no próprio prédio do instituto, assim como no Pavilhão Central de Aulas (PCA). Há 16 salas disponíveis no ICEB e 19 no PCA. As aulas ocorrem em três turnos: manhã, tarde e noite, havendo por sala, 16 horários diários de aula.

Estes horários referem-se aos períodos de início e fim de cada aula e estão distribuídos conforme a Tabela 4.1.

Tabela 4.1: Distribuição dos Horários das Aulas

Manhã	Tarde	Noite
De 07:30 às 08:20	De 12:00 às 13:30	De 18:00 às 18:50
De 08:20 às 09:10	De 13:30 às 14:20	De 19:00 às 19:50
De 09:20 às 10:10	De 14:20 às 15:10	De 19:50 às 20:40
De 10:10 às 11:00	De 15:20 às 16:10	De 21:00 às 21:50
De 11:10 às 12:00	De 16:10 às 17:10	De 21:50 às 22:40
-	De 17:10 às 18:00	-

A Tabela 4.2 mostra a quantidade de salas disponíveis e a quantidade de vagas ofertadas em cada sala de cada um dos prédios nos quais as aulas do ICEB são alocadas.

A cada semestre existe a necessidade de alocação de aproximadamente 570 (quinhentos e setenta) turmas de disciplinas, as quais ocupam 2280 (dois mil duzentos e oitenta) horários distribuídos nos três turnos de segunda-feira a sábado.

Tabela 4.2: Características dos Blocos de Salas

Blocos de Salas ICEB

Total de Salas:	16
Nome	Capacidade
Sala 01	70
Sala 02	60
Sala 03	44
Sala 04	55
Sala 05	44
Sala 06	53
Sala 07	70
Sala 08	60
Sala 09	50
Sala 10	50
Sala 14	40
Sala 18	50
Sala 19	50
Sala 21	53
Sala 22	40
Sala 23	50
Total:	839

Bloco de Salas Pavilhão

Total de Salas:	19
Nome	Capacidade
101	60
102	60
103	60
104	60
105	60
106	60
201	60
202	60
203	54
204	60
205	60
206	60
207	60
208	60
209	60
210	60
211	60
213	60
214	60
Total:	1134

O quadro de oferecimento das turmas de disciplinas é apresentado após o final do semestre letivo anterior; entretanto, a matrícula nessas é feita apenas poucos dias antes do início do novo semestre letivo. Isso é um fator complicador, porque só é possível fazer a alocação das salas após conhecer-se o número de alunos matriculados por turma. Como esse número é conhecido muito tardiamente, resta pouco tempo para fazer a alocação das salas. Nesse cenário, a solução manual torna-se uma tarefa inviável, tendo em vista a vasta quantidade de possíveis soluções e o curto período de tempo.

A solução para o problema até então era realizada mediante o sistema desenvolvido por Castro (2003). O aplicativo desenvolvido, no entanto, entre diversas características, considerava a existência de um único prédio, no caso, o próprio instituto. Como as aulas atualmente são ministradas em dois prédios, a solução encontrada para aplicar o sistema era definir *a priori* quais disciplinas devem ficar em um prédio e quais devem ficar em outro prédio. Naturalmente, essa solução traz conflitos, como por exemplo, alunos de um mesmo curso e período podem ter uma aula em um prédio e a aula seguinte em outro prédio.

São os seguintes os requisitos a serem atendidos na alocação das turmas às salas:

1. Todas as aulas devem ser alocadas;

2. Não alocar turmas a salas que não comportem essas turmas;

3. Evitar a alocação de turmas pequenas a salas de maior capacidade;

4. Alocar turmas de disciplinas de mesmo curso e período preferencialmente em uma mesma sala, evitando, assim, o trânsito de alunos entre salas;

5. Preferencialmente, alocar aulas aos prédios requisitados pelos professores das turmas;

6. Alocar uma turma em um prédio específico, em caso de exigência por parte de algum professor;

7. Respeitar as disponibilidades das salas, já que algumas salas (ou horários de algumas salas) são previamente reservadas para cursos de outras unidades acadêmicas.

4.2 Modelagem do Problema Abordado

4.2.1 Representação da Solução

Uma solução x é representada por uma matriz multidimensional de dimensões $|T| \times |P| \times |S| \times |D| \times |H|$, em que T é o conjunto de turmas de disciplinas, P é o conjunto de prédios, no caso, dois (ICEB e PCA), S é o conjunto de salas $s \in S$ de um mesmo prédio, D é o conjunto de dias da semana (segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado) e H é o conjunto dos 16 horários diários disponíveis em cada sala conforme Figura 4.1.

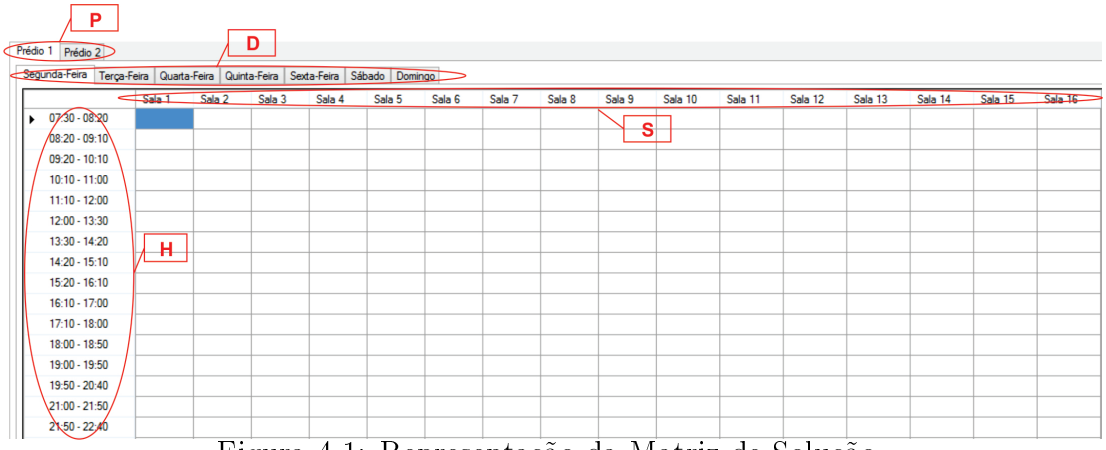


Figura 4.1: Representação da Matriz de Solução.

A Figura 4.2 mostra um fragmento de uma solução x_{tpsdh}, em que t é representação das turmas alocadas, p é o prédio do ICEB, s representa várias salas (Sala 1 a Sala 8), d representa o dia da semana (no exemplo, a segunda-feira) e h os horários disponíveis desse prédio nesse dia, no caso, das 7:30 às 22:40 h.

A capacidade de uma sala s de um prédio p é representada por cap_{ps} e a demanda de turma $t \in T$ por dem_t.

A Figura 4.2 apresenta uma alocação feita pelo Algoritmo 15 apresentado na seção 5.2: Construção Gulosa. Nesta alocação tem-se, por exemplo:

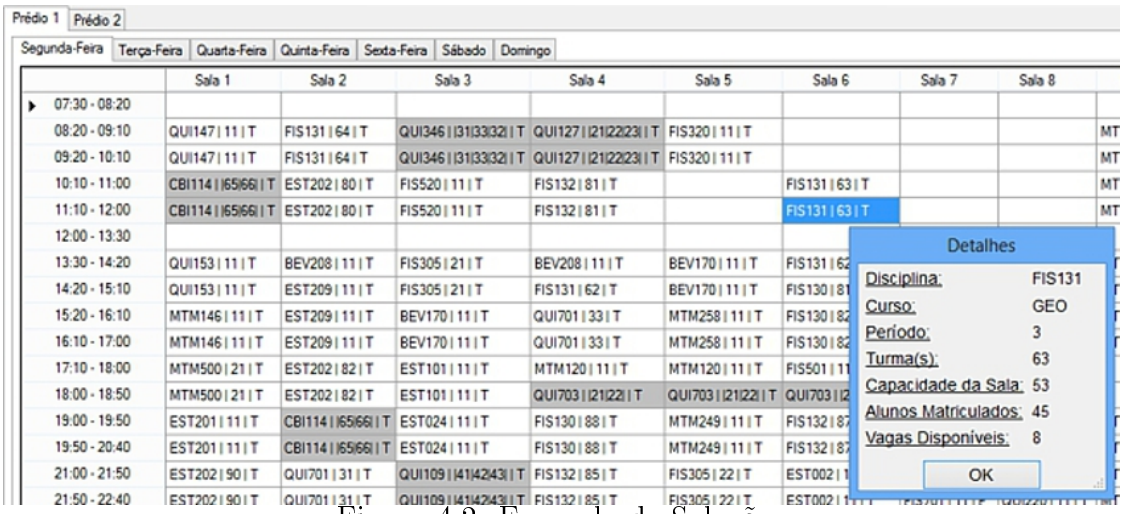


Figura 4.2: Exemplo de Solução.

No Prédio 1, na sala 6, na segunda-feira, nos horários das 10:10 às 11:00 horas e das 11:10 às 12:00 horas, foi alocada:

- Disciplina: Física II, representada pela sigla FIS131
- Curso: Engenharia geológica, representado pela sigla GEO
- Período: 3º
- Turma(s): 63
- Capacidade da Sala: 53
- Demanda: 45 alunos
- Vagas disponíveis: 8

No Prédio 1, na sala 2, na segunda-feira, nos horários das 19:00 às 19:50 horas e das 19:50 às 20:40 horas, foi alocada:

- Disciplina: Anatomia Humana, representada pela sigla CBI114
- Curso: Nutrição, representado pela sigla NUT
- Período: 1º
- Turma(s): junção das turmas 65 e 66
- Capacidade da Sala: 60
- Demanda: 47 alunos
- Vagas disponíveis: 13

A cada horário está disponível um campo para a alocação das disciplinas e suas características. Eventualmente, em uma alocação, quando esses campos não apresentarem dados, ou seja, estiverem vazios, entende-se que não foi alocada nenhuma turma neste prédio, sala, dia da semana e horário. Essa situação pode ser identificada na Figura 4.2, por exemplo, nos horários das 10:10 às 11:00 horas e das 11:10 às 12:00 horas na segunda-feira no prédio 1 nas salas 5, 7 e 8 ($x(0,1,5,2,4), x(0,1,5,2,5), x(0,1,7,2,4), x(0,1,7,2,5), x(0,1,8,2,4)$ e $x(0,1,8,2,5)$ respectivamente).

É importante ressaltar que, para a solução inicial, alocações inviáveis podem acontecer, como por exemplo, a alocação de uma disciplina em uma sala com capacidade inferior ao número de alunos matriculados. Um exemplo é ilustrado na Figura 4.3.

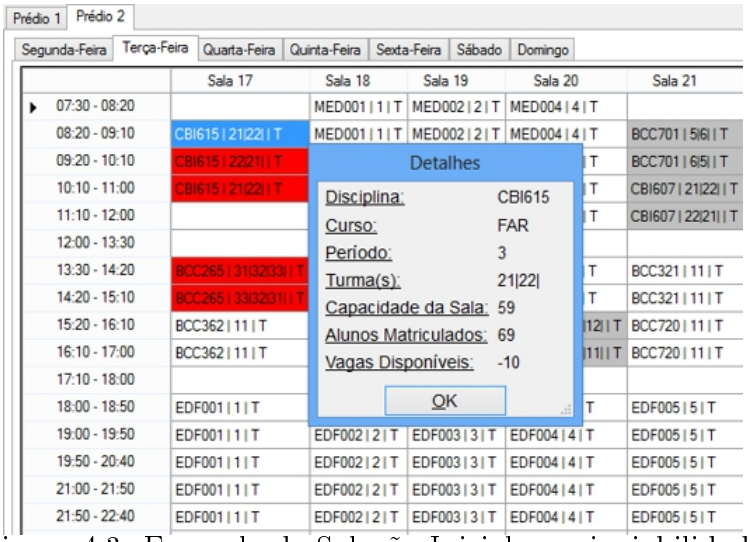


Figura 4.3: Exemplo de Solução Inicial com inviabilidade

A alocação inviável mostrada na Figura 4.3 mostra que no Prédio 2, na sala 17, na terça-feira, nos horários das 08:20 às 9:10 horas e das 9:20 às 10:10 horas, foi alocada:

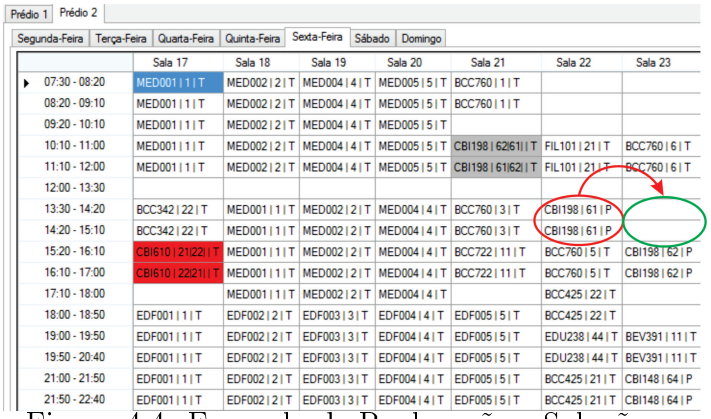
- Disciplina: Bioquímica Celular A, representada pela sigla CBI615
- Curso: Farmácia, representado pela sigla FAR
- Período: 3º
- Turma(s): junção de 21 e 22
- Capacidade da Sala: 59
- Demanda: 69 alunos
- Vagas disponíveis: **-10**

ou seja, o número de alunos matriculados é maior que a capacidade da sala em 9 alunos.

4.2.2 Estrutura de Vizinhanças e Movimentos

Para explorar o espaço de solução do PAS cinco movimentos são utilizados. Cada movimento dá origem a uma vizinhança e o conjunto de todas essas vizinhanças de uma solução $s \in S$ define a vizinhança $N(s)$.

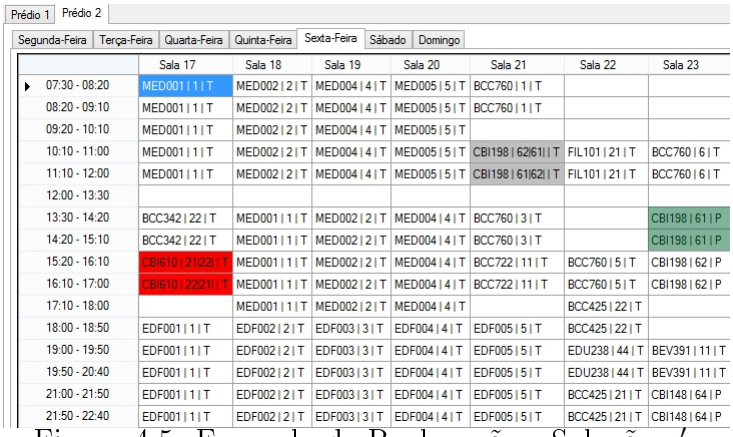
1. Realocação: Consiste em realocar uma turma de disciplina da alocação atual para outra sala que esteja vazia no mesmo prédio, dia da semana e horário.


Prédio 1 Prédio 2

Segunda-Feira | Terça-Feira | Quarta-Feira | Quinta-Feira | Sexta-Feira | Sábado | Domingo

	Sala 17	Sala 18	Sala 19	Sala 20	Sala 21	Sala 22	Sala 23
07:30 - 08:20	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	BCC760 | 1 | T		
08:20 - 09:10	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	BCC760 | 1 | T		
09:20 - 10:10	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T			
10:10 - 11:00	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	CBI198 | 62/61 | T	FIL101 | 21 | T	BCC760 | 6 | T
11:10 - 12:00	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	CBI198 | 61/62 | T	FIL101 | 21 | T	BCC760 | 6 | T
12:00 - 13:30							
13:30 - 14:20	BCC342 | 22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC760 | 3 | T	CBI198 | 61 | P	
14:20 - 15:10	BCC342 | 22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC760 | 3 | T	CBI198 | 61 | P	
15:20 - 16:10	CBI610 | 21/22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC722 | 11 | T	BCC760 | 5 | T	CBI198 | 62 | P
16:10 - 17:00	CBI610 | 22/21 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC722 | 11 | T	BCC760 | 5 | T	CBI198 | 62 | P
17:10 - 18:00		MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T		BCC425 | 22 | T	
18:00 - 18:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 22 | T	
19:00 - 19:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	EDU238 | 44 | T	BEV391 | 11 | T
19:50 - 20:40	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	EDU238 | 44 | T	BEV391 | 11 | T
21:00 - 21:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 21 | T	CBI148 | 64 | P
21:50 - 22:40	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 21 | T	CBI148 | 64 | P

Figura 4.4: Exemplo de Realocação - Solução s.


Prédio 1 Prédio 2

Segunda-Feira | Terça-Feira | Quarta-Feira | Quinta-Feira | Sexta-Feira | Sábado | Domingo

	Sala 17	Sala 18	Sala 19	Sala 20	Sala 21	Sala 22	Sala 23
07:30 - 08:20	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	BCC760 | 1 | T		
08:20 - 09:10	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	BCC760 | 1 | T		
09:20 - 10:10	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T			
10:10 - 11:00	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	CBI198 | 62/61 | T	FIL101 | 21 | T	BCC760 | 6 | T
11:10 - 12:00	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	MED005 | 5 | T	CBI198 | 61/62 | T	FIL101 | 21 | T	BCC760 | 6 | T
12:00 - 13:30							
13:30 - 14:20	BCC342 | 22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC760 | 3 | T		CBI198 | 61 | P
14:20 - 15:10	BCC342 | 22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC760 | 3 | T		CBI198 | 61 | P
15:20 - 16:10	CBI610 | 21/22 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC722 | 11 | T	BCC760 | 5 | T	CBI198 | 62 | P
16:10 - 17:00	CBI610 | 22/21 | T	MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T	BCC722 | 11 | T	BCC760 | 5 | T	CBI198 | 62 | P
17:10 - 18:00		MED001 | 1 | T	MED002 | 2 | T	MED004 | 4 | T		BCC425 | 22 | T	
18:00 - 18:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 22 | T	
19:00 - 19:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	EDU238 | 44 | T	BEV391 | 11 | T
19:50 - 20:40	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	EDU238 | 44 | T	BEV391 | 11 | T
21:00 - 21:50	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 21 | T	CBI148 | 64 | P
21:50 - 22:40	EDF001 | 1 | T	EDF002 | 2 | T	EDF003 | 3 | T	EDF004 | 4 | T	EDF005 | 5 | T	BCC425 | 21 | T	CBI148 | 64 | P

Figura 4.5: Exemplo de Realocação - Solução s'.

As Figuras 4.4 e 4.5 representam a realocação da disciplina Fisiologia II (representada pela sigla CBI198) alocada no prédio do PCA, nos horários das 13:30 às 14:20 horas e das 14:20 às 15:10 horas da sexta-feira na sala 22 para o mesmo prédio, horários e dia da semana na sala 23.

2. Troca: Consiste em trocar uma turma de disciplina por outra já alocada no mesmo prédio, no mesmo dia da semana e horário.

   As Figuras 4.6 e 4.7 representam uma troca entre as disciplinas Fisiologia II (representada pela sigla CBI198) e a disciplina Cálculo Numérico (representada pela sigla CBI198) alocadas no prédio do PCA, nos horários das 15:20 às 16:10 horas e das 16:10 às 17:00 horas da sexta-feira da sala 22 para a 23 no mesmo prédio, horários e dia da semana.

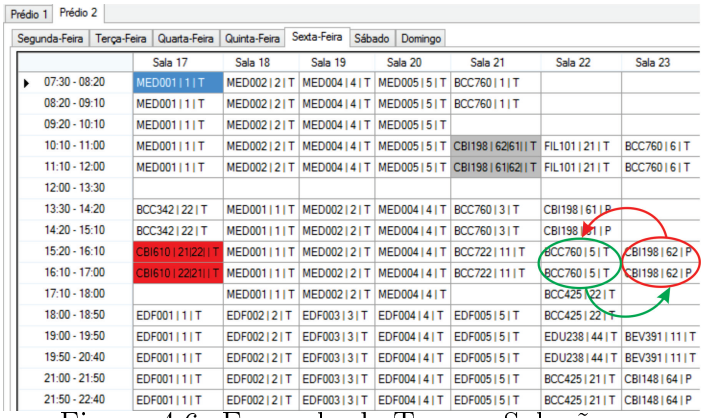


Figura 4.6: Exemplo de Troca - Solução s.

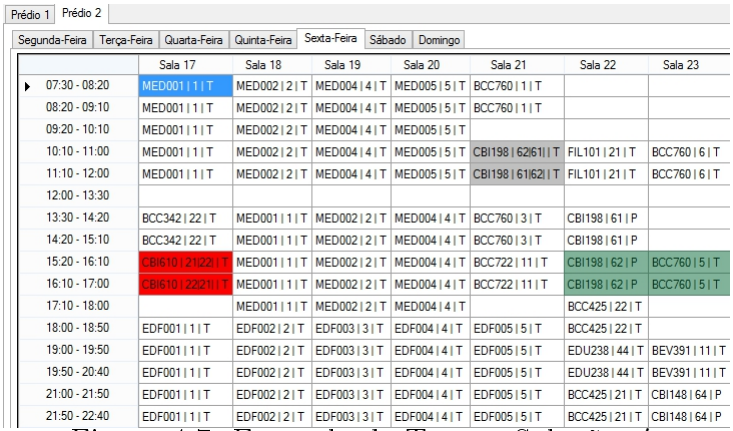


Figura 4.7: Exemplo de Troca - Solução s'.

3. Realocação entre Prédios: Consiste em realocar uma turma de disciplina da alocação atual para outra sala vazia em prédio diferente, de um mesmo dia da semana e horário.

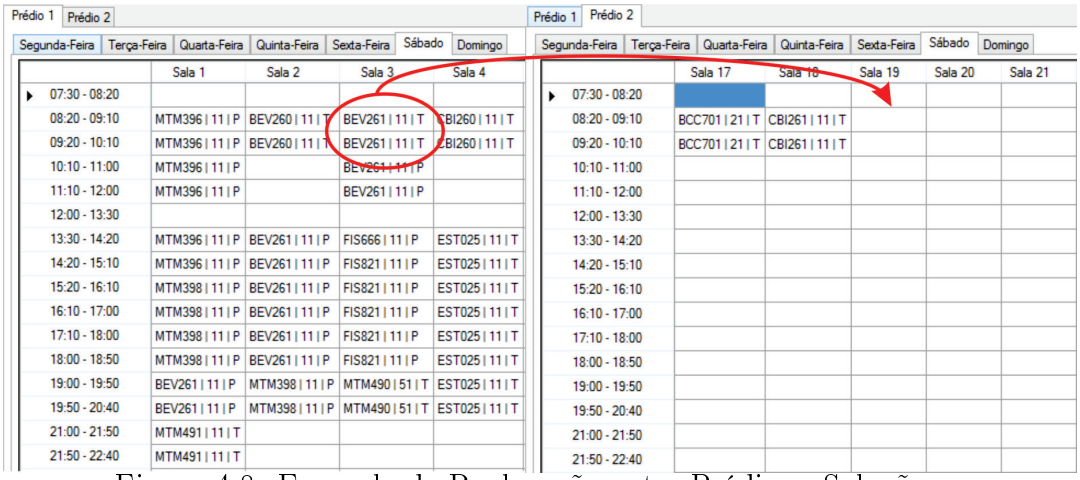


Figura 4.8: Exemplo de Realocação entre Prédios - Solução s.

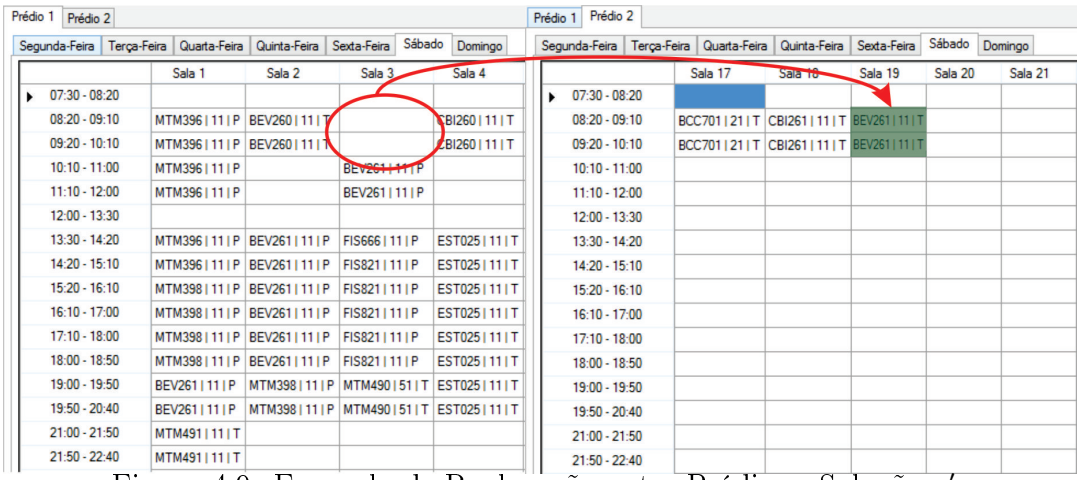


Figura 4.9: Exemplo de Realocação entre Prédios - Solução s'.

As Figuras 4.8 e 4.9 ilustram a realocação entre prédios da turma da disciplina Seminário de Pesquisa e Monografia II (representada pela sigla BEV261) alocada no prédio do ICEB, nos horários das 08:20 às 09:10 horas e das 09:20 às 11:10 horas do sábado na sala 3 para a sala 19 do prédio do PCA, nos mesmos horários e dia da semana.

4. Troca entre Prédios: Consiste em trocar uma turma de disciplina por outra já alocada em prédio diferente, para o mesmo dia da semana e horário.

   As Figuras 4.10 e 4.11 mostram uma troca entre prédios das disciplinas Seminário de Pesquisa e Monografia I (representada pela sigla BEV260) e a disciplina Seminário de Pesquisa e Monografia II (representada pela sigla CBI261) alocadas nos prédios do ICEB e PCA, salas 2 e 18, respectivamente, nos horários 08:20 às 09:10 horas e das 09:20 às 11:10 horas do sábado.

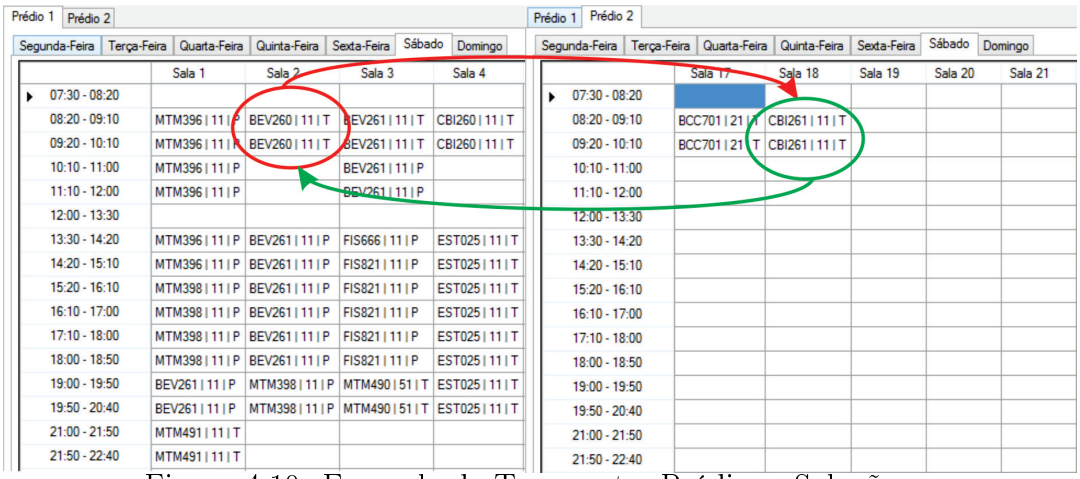


Figura 4.10: Exemplo de Troca entre Prédios - Solução s.

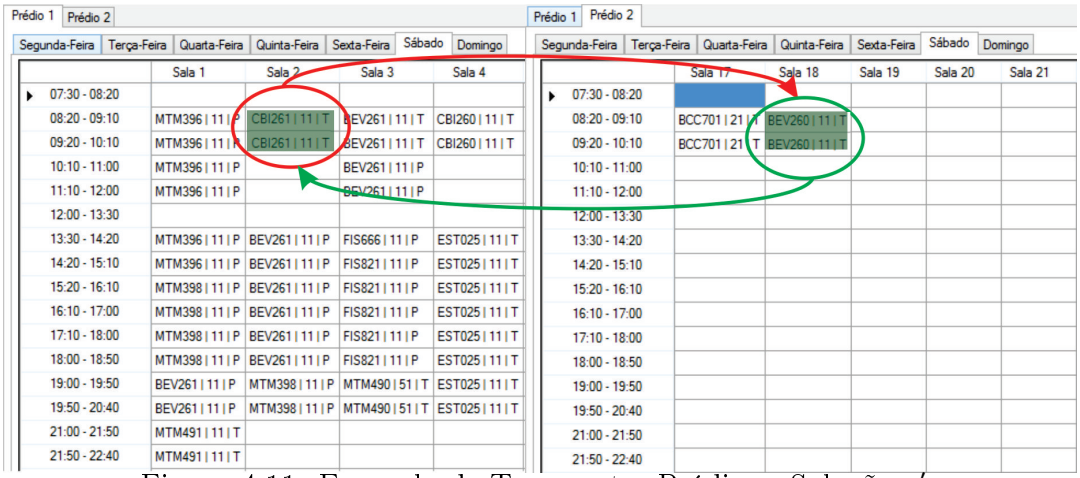


Figura 4.11: Exemplo de Troca entre Prédios - Solução s'.

5. Disciplina/Turno/Curso/Período para mesma sala: Consiste em efetuar realocações ou trocas, independentemente do prédio, visando, desde que em mesmo turno (manhã, tarde ou noite), que todas as disciplinas de um mesmo curso e período fiquem em uma mesma sala. Mais especificamente, dada uma turma (para clareza, seja A essa turma) de disciplina de um dado período de curso (por exemplo, segundo período), se sua aula for ministrada por exemplo na parte da manhã, então procura-se outra turma do segundo período do mesmo curso que seja dada também na parte da manhã. Encontrando uma turma com essas características (seja B essa turma), então é feita a realocação dessa turma para a mesma sala, ou seja, as turmas A e B ficarão na mesma sala. Se não for possível fazer a realocação da turma B para a sala da turma A porque no horário seguinte ou anterior, a sala da turma A está ocupada com outra turma (digamos, turma C), então é feita a troca da aula da turma C pela aula da turma B. Não existindo outra turma de mesmo período no mesmo turno, nenhuma ação é realizada.

4.2.3 Avaliação de uma Solução

Uma solução é avaliada por uma função baseada em penalidades pelo não atendimento às restrições do problema. Para tal, cada restrição i é contabilizada quantas vezes ela não é respeitada. Essa quantidade de vezes é, então, multiplicada por um peso α_i. Valores maiores para α_i indicam uma importância maior da respectiva restrição. Essa função deve ser minimizada, o que significa que menores valores para ela indicam um maior atendimento às restrições.

A formulação da função de avaliação se dá da seguinte forma:

1. Excesso de Alunos em sala: Determinam-se todas as salas onde a demanda de alunos é superior à capacidade da sala. Em seguida, encontra-se, para cada uma dessas salas, a diferença entre a demanda de alunos e a capacidade da sala. Essa diferença é somada até que todas as salas com excesso de alunos sejam verificadas. Ao final, multiplica-se a quantidade de alunos em excesso em todas as salas pelo peso α_1.

2. Ociosidade em sala: Encontram-se todas as salas onde a demanda de alunos é inferior à capacidade da sala. Posteriormente, encontra-se, para cada uma dessas salas, a diferença entre a capacidade da sala e a demanda de alunos. Esta diferença é somada até que todas as salas com ociosidade sejam verificadas. Ao final, multiplica-se a quantidade de carteiras ociosas em todas as salas pelo peso α_2.

3. Turmas de disciplinas de mesmo curso e período em salas diferentes: Determinam-se todas as turmas de disciplinas de mesmo curso e período que estejam alocadas em salas diferentes. Soma-se a quantidade de turmas nessas condições e, ao final, multiplica-se pelo peso α_3.

4. Turmas de disciplinas em prédios diferentes: Encontram-se todas as turmas de disciplinas que, inicialmente, tinham como preferência a alocação em determinado prédio e, posteriormente, foi alocada em outro. Somam-se todas as turmas nessa condição e, ao final, multiplica-se o resultado pelo peso α_4.

5. Turmas de disciplinas alocadas em salas virtuais: Encontram-se todas as turmas de disciplinas que foram alocadas em salas virtuais. Soma-se a quantidade de turmas nessa condição, multiplicando o resultado pelo peso α_5.

O valor da função de avaliação é dado pelo somatório da quantidade de não atendimentos às restrições multiplicado pelo respectivo peso. O Algoritmo 13 mostra o pseudocódigo dessas operações.

Algoritmo 13: Cálculo da Função de Avaliação

Entrada: $S = \{conjunto\ de\ solucoes\ alocadas\}$
Saída: Valor de FO

inicio
  $\alpha_1 \leftarrow Penalizacao\ p/\ excesso$;
  $\alpha_2 \leftarrow Penalizacao\ p/\ ociosidade$;
  $\alpha_3 \leftarrow Penalizacao\ p/\ salas\ diferentes$;
  $\alpha_4 \leftarrow Penalizacao\ p/\ predios\ diferentes$;
  $\alpha_5 \leftarrow Penalizacao\ p/\ alocacao\ em\ salas\ virtuaus$;
  $S \leftarrow \{conjunto\ de\ solucoes\ alocadas\}$;
  $AlunosExcesso \leftarrow$
  $\{Alocacoes\ com\ a\ demanda\ de\ alunos\ maior\ que\ a\ capacidade\ da\ sala\}$;
  enquanto $|\ AlunosExcesso\ | > 0$ **faça**
    $ValorExcesso \leftarrow ValorExcesso + DemandaAlunos - CapacidadeSala$;
  fim
  $ValorExcesso \leftarrow \alpha_1 * ValorExcesso$;
  $OciosidadeSala \leftarrow$
  $\{Alocacoes\ com\ a\ capacidade\ da\ sala\ maior\ que\ a\ demanda\ de\ alunos\}$;
  enquanto $|\ OciosidadeSala\ | > 0$ **faça**
    $ValorOciosidade \leftarrow$
    $ValorOciosidade + CapacidadeSala - DemandaAlunos$;
  fim
  $ValorOciosidade \leftarrow \alpha_2 * ValorOciosidade$;
  $SalasDiferentes \leftarrow$
  $\{Alocacoes\ de\ disciplinas\ semelhantes\ em\ salas\ diferentes\}$;
  enquanto $|\ SalasDiferentes\ | > 0$ **faça**
    $ValorSalasDiferentes \leftarrow ValorSalasDiferentes + 1$;
  fim
  $ValorSalasDiferentes \leftarrow \alpha_3 * ValorSalasDiferentes$;
  $PrediosDiferentes \leftarrow$
  $\{Alocacoes\ de\ disciplinas\ semelhantes\ em\ predios\ diferentes\}$;
  enquanto $|\ PrediosDiferentes\ | > 0$ **faça**
    $ValorPrediosDiferentes \leftarrow ValorPrediosDiferentes + 1$;
  fim
  $ValorPrediosDiferentes \leftarrow \alpha_4 * ValorPrediosDiferentes$;
  $SalasVirtuais \leftarrow$
  $\{Alocacoes\ de\ disciplinas\ em\ predios\ diferentes\ dos\ de\ origem\}$;
  enquanto $|\ SalasVirtuais\ | > 0$ **faça**
    $ValorSalasVirtuais \leftarrow ValorSalasVirtuais + 1$;
  fim
  $ValorSalasVirtuais \leftarrow \alpha_5 * ValorSalasVirtuais$;
  $FO \leftarrow ValorExcesso + ValorOciosidade + ValorSalasDiferentes +$
  $ValorPrediosDiferentes + ValorSalasVirtuais$;
  retorna FO;
fin

Capítulo 5

Descrição do Algoritmo Proposto

5.1 Considerações Iniciais

Nesta seção serão apresentadas as técnicas e a descrição dos algoritmos desenvolvidos para a solução do problema abordado.

A metodologia proposta para a solução do problema, considera uma aplicação composta por duas fases. A primeira fase é disposta por um algoritmo guloso para a geração de uma solução inicial (Algoritmo 15), possuindo uma pequena particularidade, visando maior diversificação na geração de possíveis soluções iniciais, foi a inserido um parâmetro de aleatoriedade (γ), o qual permite, ao se gerar uma solução, escolher o "grau" de aleatoriedade da solução desejada.

Para a fase subsequente, de refinamento, optou-se pela utilização do *Simulated Annealing*, pois, pela natureza combinatória, com complexidade *NP-Difícil* (Even et al. (1976); Carter e Tovey (1992)), do PAS, algumas importantes características desta metodologia, como por exemplo a possível aceitação de soluções intermediárias de piora para se escapar dos ótimos locais, contribuem para a geração de uma solução viável e aplicável ao ambiente estudado em um tempo computacional aceitável. Além disso, a literatura dispões de diversos trabalhos deste tipo que alcançaram bons resultados a partir desta metaheurística, dentre os quais destacam-se: Alfaro e Terán (1998); Antunes e Peeters (2001); Souza et al. (2002); Castro (2003); Silva (2005); Kripka e Kripka (2010).

O Algoritmo 14 representa, de forma geral, a proposta apresentada.

Algoritmo 14: Construção Gulosa + *SA*

Entrada: $E = \{$conjunto de todos horários a se alocar$\}$
Saída: Solução $s*$

```
1 inicio
2 |  s ← ConstruoGulosa(E);          /* Algoritmo 15 */
3 |  s* ← SA(s);                     /* Algoritmo 16 */
4 fin
5 retorna s*;
```

5.2 Construção da Solução Inicial

A solução inicial é gerada de forma que inicialmente, as aulas das turmas são ordenadas pelo tamanho, de sorte que as turmas maiores são as primeiras da lista e as menores, as últimas. A seguir, começa-se a alocação, de acordo com o coeficiente de gulosidade (γ), caso seja zero, inicia com a turma de maior demanda. Em seguida, a partir desta turma, procura-se na lista das turmas, a(s) turma(s) de mesmo curso, período e turno em que a turma. Elas são alocadas à maior sala disponível no horário em que são ofertadas, independentemente de a turma caber ou não nesta sala. Se não existir sala nessas condições, então é criada uma sala fictícia, dita virtual, e essa(s) turma(s) são então alocadas nesta sala virtual. A partir de então passa-se para a alocação da segunda maior turma. Para esta turma, procura-se, então, a maior sala disponível. Este procedimento é repetido até que todas as aulas das turmas sejam alocadas a alguma sala, real ou virtual.

Eventualmente pode-se utilizar uma solução inicial com um certo grau de aleatoridade, para isso, informa-se a porcentagem de aleatoriedade desejada (entre zero e cem) mediante o coeficiente γ. Um exemplo que se tenham dez horários a se alocar e deseja-se 50% (cinquenta por cento) de aleatoriedade, após a ordenação das turmas pelo tamanho, a lista a se alocar é divida pelo coeficiente e, no caso, os cinco primeiros horários de maior turma serão alocados de acordo com o expresso no parágrafo anterior, no entanto, a escolha dentre estes cinto é feita de forma aleatória. Após este procedimento a lista restante de horários será novamente dividida pelo coeficiente γ, os escolhidos alocados e assim sucessivamente até que não existam mais horários a se alocar.

O pseudocódigo da construção de uma solução inicial para o problema é apresentado pelo Algoritmo 15.

Algoritmo 15: Construção Gulosa

Entrada: $E = \{$conjunto de todos horários a se alocar$\}$
Entrada: $\gamma = \{$Coeficiente de aleatoriedade$\}$
Saída: Solução S

```
1  inicio
2      S ← 0;
3      C ← E        /* ordenado de forma decrescente pelo tamanho das
       turmas */;
4      g ← 0 /* armazena o valor das inserções na solução inicial */;
5      enquanto | C |> 0 faça
6          se (γ == 0) então
7              Seleciona-se c*;      /* Turma de maior demanda em C */ ;
8          senão
9              Seleciona-se c*;      /* De acordo com coeficiente γ */ ;
10         fim
11         L(c) ← {Turma(s) de mesmo curso, período e turno que c*};
12         Verifica a existência de sala disponível para alocar L(c);
13         se (Existe Sala Disponível) então
14             Alocar L(c) à sala disponível;
15             Atualizar a solução parcial: S ← S ∪ {L(c)} ;
16         senão
17             Criar Sala Virtual;
18             Alocar L(c) à sala virtual criada;
19             Atualizar a solução parcial: S ← S ∪ {L(c)} ;
20         fim
21         g ← g + o valor do custo da inserção de L(c) em S;
22         Retirar L(c) de C;
23     fim
24     retorna S;
25 fin
```

Como geralmente uma heurística de construção não produz uma solução final de boa qualidade (Barbosa, 2011), é aplicada sobre ela uma heurística de refinamento, conforme seção 5.3.

5.3 Algoritmo da Fase de Refinamento

O refinamento da solução gerada na seção 5.2 é feita pela metaheurística *Simulated Annealing* – SA Kirkpatrick et al. (1983). A este algoritmo apenas uma pequena alteração foi realizada quanto a inserção de um novo parâmetro de parada, no caso, o tempo, ou seja, a interrupção da execução será realizada mediante o "resfriamento" total ou um determinado período de tempo ser alcançado. As modificações para tal podem ser verificadas nas linhas 2 e 7 do Algoritmo 16.

Algoritmo 16: *Simulated Annealing* Aplicado

Entrada: Solução Inicial - s_0
Entrada: Temperatura Inicial - T_0
Entrada: Número de iterações em T - $SAMax$
Entrada: Taxa de resfriamento - α
Entrada: Tempo máximo de execução - $TEMPMax$
Saída: s^*

```
1  inicio
2      Iniciar contagem de tempo em TEMPCorrente;
3      s ← s_0;
4      s' ← s;                          /* Melhor solução obtida até então */
5      T ← T_0;
6      IterT ← 0;
7      enquanto (T > 0)ou(TEMPCorrente < TEMPMax) faça
8          enquanto (IterT < SAmax) faça
9              IterT ← IterT + 1;
10             Gerar vizinho qualquer s' ∈ N(s);
11             Δ = f(s') − f(s);
12             se (Δ < 0) então
13                 s ← s';
14                 se (f(s') < f(s*)) então
15                     s* ← s';
16                 fim
17             senão
18                 Tome x ∈ [0, 1];
19                 se (x <e^{−Δ/T}) então
20                     s ← s';
21                 fim
22             fim
23         fim
24     fim
25     T ← α × T;
26     IterT ← 0;
27     retorna s*
28 fin
```

No Algoritmo 16 linha 10, cada uma das cinco vizinhanças implementadas (vide Subseção 4.2.2) é escolhida aleatoriamente a cada iteração. Por padrão, a probabilidade de escolha da vizinhança é a mesma para todas as vizinhanças. No entanto, ao usuário é dada a oportunidade de estabelecer uma diferenciação entre essas probabilidades, conforme mostra a Figura 6.15. Isso pode ser interessante se um movimento for considerado melhor que outro na exploração do espaço de busca.

Capítulo 6

Sistema Desenvolvido

6.1 Considerações Iniciais

O sistema foi desenvolvido sob a linguagem C# (*C Sharp*), no ambiente IDE Microsoft Visual Studio 2013 e utiliza o banco de dados Microsoft SQL Server 2012. O desenvolvimento considera os preceitos da Orientação a Objetos e da estrutura de desenvolvimento em camadas tendo em vista suas vantagens quanto a segurança, manutenção, reusabilidade e escalabilidade da codificação.

A linguagem C# foi escolhida devido ao fato de ser derivada do C e C++, sua flexibilidade de utilização em ambiente *Windows*, possibilitando a utilização de recursos nativos deste Sistema Operacional (SO), a facilidade na construção de ambientes ergonômicos e de fácil usabilidade, a grande compatibilidade com o Sistema de Gerenciamento de Banco de Dados (SGBD) SQL Server (utilizado pelo seu desempenho), dentre outros.

O armazenamento de dados foi feito pela ferramenta Microsoft SQL Server 2012 e foi desenvolvido a partir do conceito abstrato de Banco de Dados Relacional, facilitando o acesso, normalizando e garantindo mais fortemente a integridade dos dados.

6.2 Estrutura de Armazenamento de Dados

Para expor o Modelo Relacional de Dados do *software* desenvolvido para a solução do problema proposto são expressas as entidades e atributos do banco de dados criado. Uma divisão relacionada à característica de utilização, pelo sistema, das entidades e atributos, é proposta visando a facilitar o entendimento.

A divisão proposta se dá em duas partes: entidades de parametrização e entidades de aplicação.

6.2.1 Entidades de Parametrização

Estas entidades foram criadas para armazenar informações de configurações dos parâmetros necessários para a execução do algoritmo desenvolvido. Elas estão diretamente relacionadas às características do ambiente explorado, que servem de base para a geração das possíveis soluções.

Com a alteração dos registros (ou tuplas) de algumas destas entidades é possível caracterizar e, assim, aplicar o algoritmo proposto a outras IES com características e necessidades semelhantes.

As entidades apresentadas na Figura 6.1 estão diretamente relacionadas às características da estrutura física da IES disponível para a alocação, assim como detalhes sobre o formato acadêmico de oferecimento das disciplinas que compõem os currículos dos cursos.

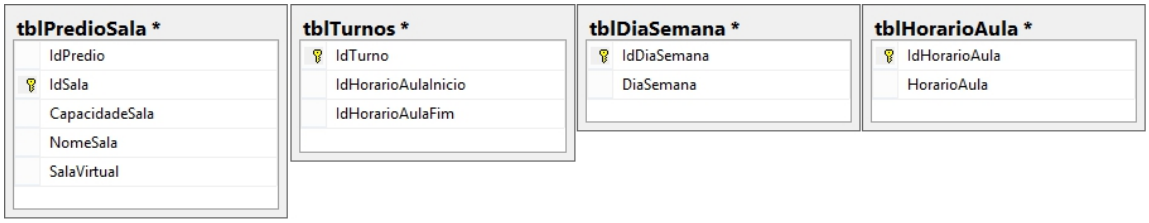


Figura 6.1: Entidades de Parametrização - IES.

A Figura 6.2 traz as entidades relacionadas às características dos parâmetros inerentes ao algoritmo proposto, como os pesos α_i das penalizações apresentadas na Subseção 4.2.3, os parâmetros de entrada da metaheurística *Simulated Annealing* presentes no Algoritmo 16 e as probabilidades de escolhas dos movimentos para alteração da vizinhança conforme o segundo parágrafo da seção 5.3.

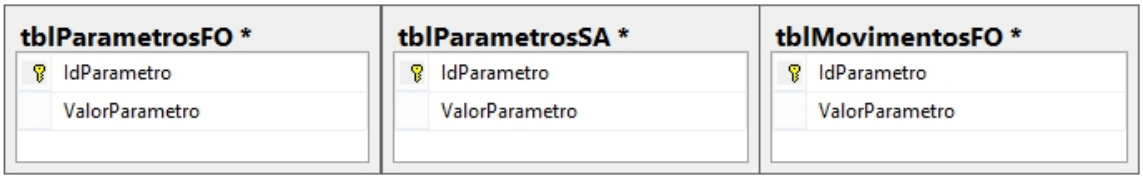


Figura 6.2: Entidades de Parametrização - Algoritmo.

6.2.2 Entidades de Aplicação

Estas entidades estão relacionadas com os dados referentes ao quadro de horários pré-estabelecido, a disponibilidade das salas conforme requisito exposto no item 7 da seção 4.1, dados inerentes à execução do algoritmo, os detalhes sobre a "agregação de turmas" (mais detalhes na seção 6.4.4) e o armazenamento dos dados referentes aos resultados obtidos após a execução do algoritmo.

As entidades apresentadas na Figura 6.3 referem-se às informações advindas do quadro de horários pré-estabelecido. Um importante detalhe a se destacar é o atributo "ObrigaPredio" da entidade "tblDisciplina", que garante o item 6 da seção 4.1.

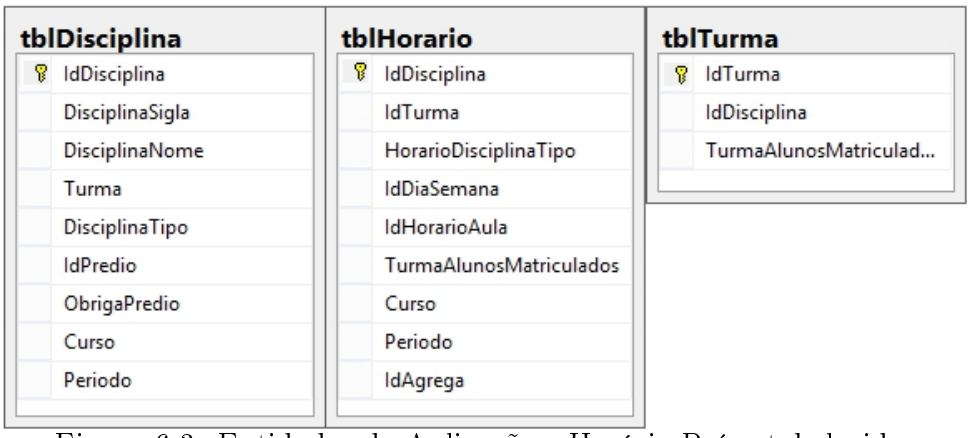


Figura 6.3: Entidades de Aplicação - Horário Pré-estabelecido.

A Figura 6.3 apresenta os atributos necessários para o bloqueio de salas pré-agendadas e a agregação de turmas.

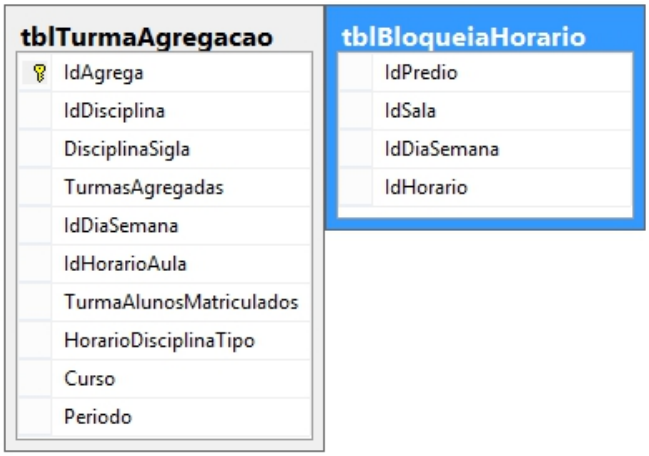


Figura 6.4: Entidades de Aplicação - Agregação e Bloqueio.

A solução final, encontrada como resultado da execução do algoritmo, bem como o valor final da função de avaliação referente a esta melhor solução encontrada, estão armazenados nas entidades da Figura 6.5.

6.3 Entrada de Dados

A entrada de dados do sistema tem como ponto principal os dados referentes às Entidades de Parametrização - IES (vide Figura 6.1), pela relação direta com o espaço físico estudado e as Entidades de Aplicação - Horário Pré-estabelecido (vide Figura 6.3), que servem de base para toda a aplicação, pois representam o quadro de horários informado pela IES, bem como os dados relacionados a este horário. O abastecimento destes dados para o sistema é realizado mediante importação direta,

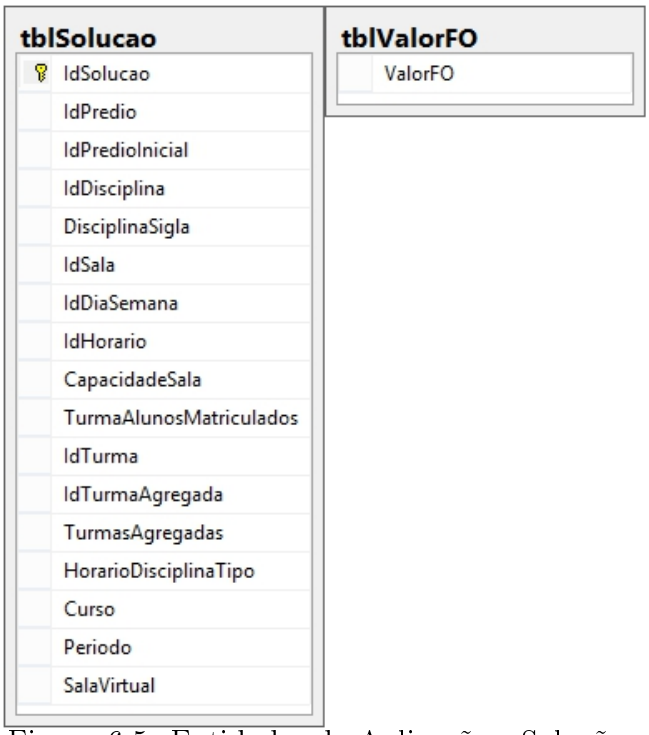


Figura 6.5: Entidades de Aplicação - Solução.

pela IDE Microsoft SQL Server Management, de arquivo texto (.txt) ou *Comma-separated values* (.csv) que seguem um formato padrão.

Os atributos referentes a cada uma das quatro Entidades de Parametrização - IES e seus formatos padrão estabelecidos para a geração de um arquivo (.txt ou .csv) para importação estão expressos nas Tabelas 6.1, 6.2, 6.3 e 6.4.

Tabela 6.1: Entidades de Parametrização - IES - **tblPredioSala**

Atributo	Descrição	Tipo de Dado	Observação
IdPredio	Codificação numérica para representação de um prédio	int	≠ NULO
IdSala	Codificação numérica para representação de uma sala	int	≠ NULO
CapacidadeSala	Valor inteiro que representa capacidade de uma determinada sala	int	≠ NULO
NomeSala	Valor alfanumérico do nome de uma sala	varchar(50)	
SalaVirtual	Valor booleano que caracteriza uma sala como virtual ou não (0=não virtual; 1=virtual)	bit	≠ NULO

Os atributos referentes a cada uma das três Entidades de Aplicação - Horário Pré-estabelecido assim como o formato padrão estabelecido para a geração de um arquivo (.txt ou .csv) para importação estão expressos nas Tabelas 6.5, 6.6 e 6.7.

Tabela 6.2: Entidades de Parametrização - IES - **tblTurnos**

Atributo	Descrição	Tipo de Dado	Observação
IdTurno	Codificação numérica para representação de um turno (1=Manhã; 2=Tarde; 3=Noite)	int	≠ NULO
IdHorarioAulaInicio	Valor numérico que identifica um horário de aula de acordo com a entidade "tblHorarioAula" que define o início de um turno	int	≠ NULO
IdHorarioAulaFim	Valor numérico que identifica um horário de aula de acordo com a entidade "tblHorarioAula" que define o fim de um turno	int	≠ NULO

Tabela 6.3: Entidades de Parametrização - IES - **tblDiaSemana**

Atributo	Descrição	Tipo de Dado	Observação
IdDiaSemana	Codificação numérica para representação dos dias da semana (Ex.: 1=domingo; 2=segunda-feira; 3=terça-feira; 4=quarta-feira; 5=quinta-feira; 6=sexta-feira; 7=sábado)	int	≠ NULO
DiaSemana	Valor alfanumérico do nome do dia da semana	varchar(50)	≠ NULO

Tabela 6.4: Entidades de Parametrização - IES - **tblHorarioAula**

Atributo	Descrição	Tipo de Dado	Observação
IdHorarioAula	Codificação numérica para representação dos horários de aulas (Ex.: 1=07:30 - 08:20; 2=08:20 - 09-10; 3=09:20 - 10:10; 4=10:10 - 11:00; 5=11:10 - 12:00; 6=12:00 - 13:30; etc.)	int	≠ NULO
HorarioAula	Valor alfanumérico do nome do dia da semana	varchar(50)	≠ NULO

Tabela 6.5: Entidades de Aplicação - Horário Pré-estabelecido - **tblDisciplina**

Atributo	Descrição	Tipo de Dado	Observação
IdDisciplina	Codificação numérica para representação de uma disciplina	int	≠ NULO
DisciplinaSigla	Valor alfanumérico para identificação resumida de uma disiplina	varchar(10)	≠ NULO
DisciplinaNome	Valor alfanumérico do nome da disciplina	varchar(200)	≠ NULO
Turma	Valor inteiro que representa a turma de uma disciplina	int	≠ NULO
DisciplinaTipo	Valor inteiro que representa um tipo de disciplina (1=obrigatória; 2=optativa; 3=eletiva)	int	
IdPredio	Codificação numérica que representa uma possível preferência de prédio para oferecimento de uma disciplina	int	
ObrigaPredio	Valor booleano que representa a obrigatoriedade de uma disciplina ser oferecida em um determinado prédio (0=não obrigatoriedade; 1=obrigatoriedade)	bit	
Curso	Valor alfanumérico do nome do curso de uma disciplina	varchar(50)	≠ NULO
Periodo	Codificação numérica que representa o período de oferecimento de uma disciplina	int	≠ NULO

As entidades representadas pelas Figuras 6.4, 6.5 e 6.2, estão vinculadas a características, configurações e resultados utilizados/gerados/obtidos no decorrer da

Tabela 6.6: Entidades de Aplicação - Horário Pré-estabelecido - **tblHorario**

Atributo	Descrição	Tipo de Dado	Observação
IdDisciplina	Codificação numérica para representação de uma disciplina	int	≠ NULO
IdTurma	Codificação numérica para representação de uma turma	int	≠ NULO
HorarioDisciplinaTipo	Valor alfanumérico que representa o tipo da aula (T=teórica; P=prática)	varchar(5)	≠ NULO
IdDiaSemana	Codificação numérica para representação dos dias da semana de acordo com a entidade "tblDiaSemana" (Ex.: 1=domingo; 2=segunda-feira; 3=terça-feira; 4=quarta-feira; 5=quinta-feira; 6=sexta-feira; 7=sábado)	int	≠ NULO
IdHorarioAula	Codificação numérica para representação dos horários de aulas de acordo com a entidade "tblHorarioAula" (Ex.: 1=07:30 - 08:20; 2=08:20 - 09-10; 3=09:20 - 10:10; 4=10:10 - 11:00; 5=11:10 - 12:00; 6=12:00 - 13:30; etc.)	int	≠ NULO
TurmaAlunosMatriculados	Valor inteiro que representa a quantidade de alunos matriculados em uma turma	int	≠ NULO
Curso	Valor alfanumérico do nome do curso de uma disciplina	varchar(50)	≠ NULO
Periodo	Codificação numérica que representa o período de oferecimento de uma disciplina	int	
IdAgrega	Codificação numérica que representa a existência ou não de uma agregação para determinada turma	int	

Tabela 6.7: Entidades de Aplicação - Horário Pré-estabelecido - **tblHorario**

Atributo	Descrição	Tipo de Dado	Observação
IdTurma	Codificação numérica para representação de uma turma	int	≠ NULO
IdDisciplina	Codificação numérica para representação de uma disciplina	int	≠ NULO
TurmaAlunosMatriculados	Valor inteiro que representa a quantidade de alunos matriculados em uma turma	int	≠ NULO

execução do *software*, assim não caracterizam entradas de dados e são melhor entendidas nas subseções 6.4.4, 6.4.8, 6.4.5 e 6.4.7.

6.4 Funcionalidades do Sistema

O sistema apresenta algumas funcionalidades que permitem, além de solucionar o PAS abordado, alterar parâmetros a fim de realizar diversos testes que auxiliem na sua calibragem. Estas alterações estão dispostas de forma simples e acessível na tela inicial de execução do sistema (Figura 6.6). Com a finalidade de expor estas funcionalidades as ferramentas estão descritas a seguir.

6.4.1 Tela Inicial do Sistema

A tela inicial do sistema foi desenvolvida com foco ergonômico objetivando o acesso simples e sempre rápido aos dados, ferramentas do sistema e informações visuais.

A "navegação", por exemplo, entre prédios, dias da semana, informações sobre as disciplinas alocadas estão disponíveis a qualquer momento, bastando o clique do

mouse sobre as referências desejadas para ter acesso às informações desejadas (botão esquerdo para indicar qual prédio e dia da semana e botão direito sobre uma célula para ter acesso a diversas informações)(vide Figura 6.7).

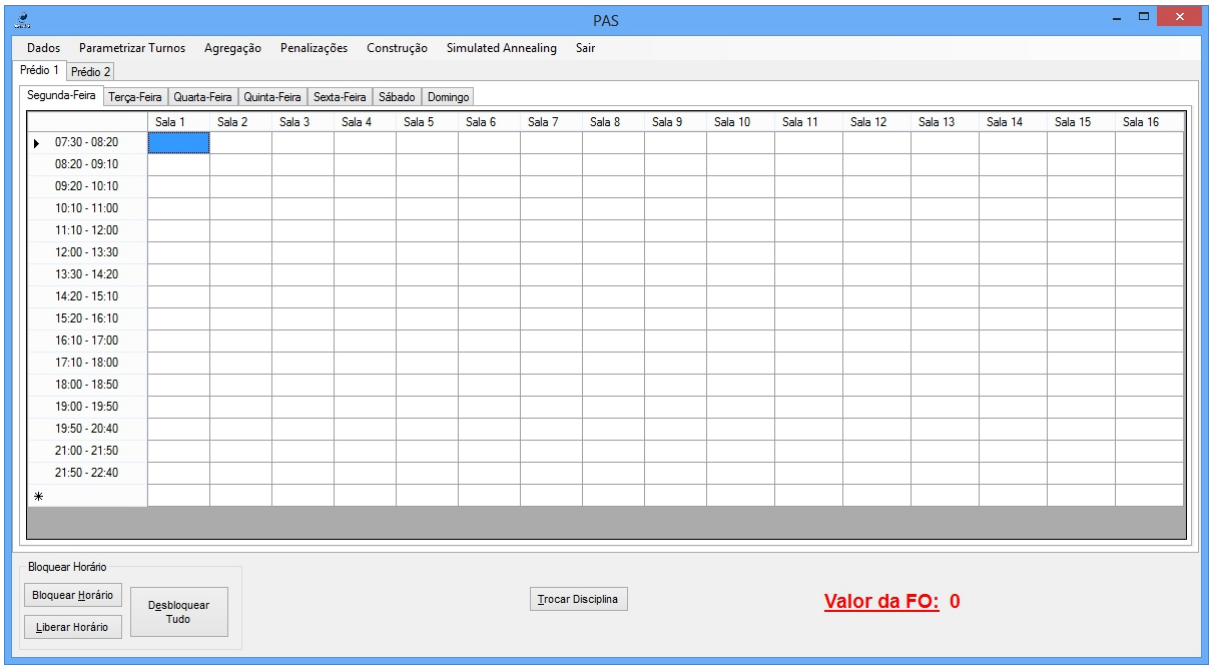


Figura 6.6: Tela inicial do sistema sem alocação.

As cores das células também trazem importantes informações, como por exemplo, caso uma aula tenha sido alocada em uma sala com capacidade inferior ao número de alunos matriculados na turma ela ficará da cor vermelha (conforme a alocação do segundo horário da terça-feira na sala 18 do prédio 2 na Figura 6.7) e se a aula alocada em uma sala seja oriunda de uma disciplina que possua agregação de turmas, e não tenha extrapolado sua capacidade, ela ficará na cor cinza (conforme a alocação do terceiro horário da terça-feira na sala 17 do prédio 2 na Figura 6.7).

Outra importante informação visual disponível é, caso após uma alocação realizada o sistema tenha criado salas virtuais, toda a coluna que representa os horários desta sala estarão em outra tonalidade de vermelho, como no caso das salas 36, 37 e 41 da Figura 6.8.

As demais parametrizações e funcionalidades estão disponíveis através do menu superior ou por botões presentes na tela inicial.

6.4.2 Limpar Dados

Esta funcionalidade está disponível através da opção "Dados" no menu superior e executa a limpeza total dos dados referentes à alocação e agregações realizadas, voltando o sistema para o estado inicial conforme Figura 6.6.

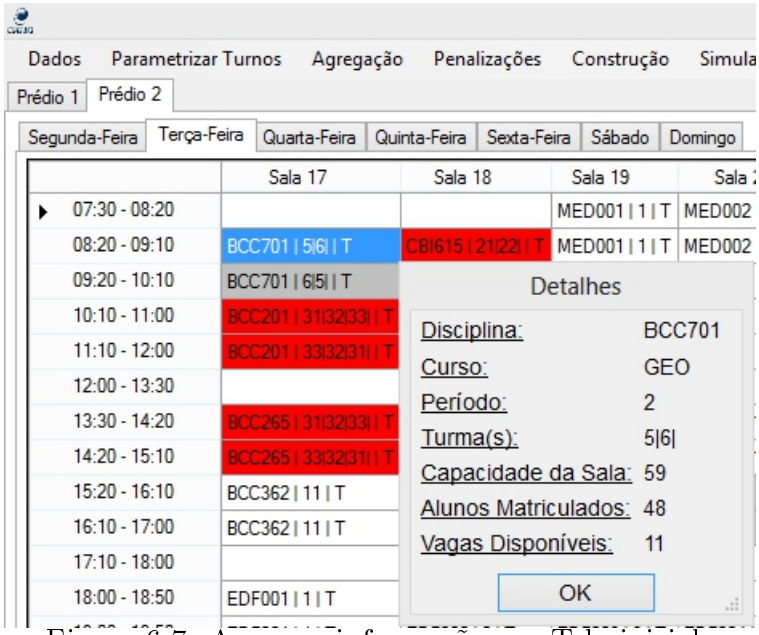


Figura 6.7: Acesso a informações na Tela inicial.

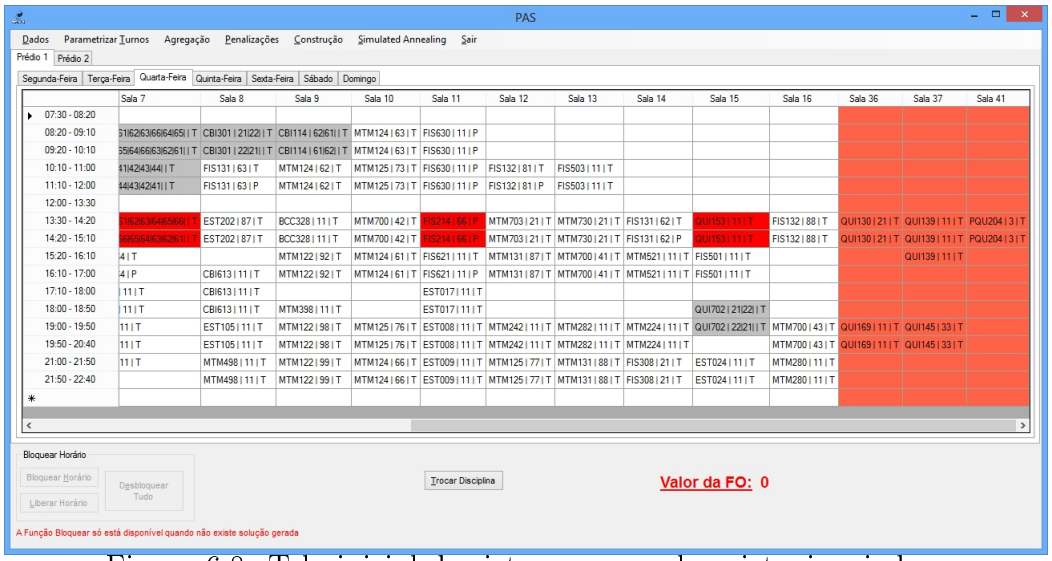


Figura 6.8: Tela inicial do sistema com salas virtuais criadas.

Os dados importados referentes ao quadro de horários e parametrizações não são afetados na utilização desta funcionalidade.

6.4.3 Parametrização de Turnos

Esta parametrização pode ser acessada através do menu superior na opção "Parametrizar Turnos" e diz respeito às configurações necessárias para caracterizar o turno de oferecimento das disciplinas (manhã, tarde, noite).

Estes parâmetros são editáveis e podem ser alterados a qualquer momento, sendo necessária uma nova execução posterior para contemplar os novos parâmetros.

Um importante detalhe a ser descrito é que devido à necessidade de, eventualmente, efetuar a alocação de turmas de outros institutos, os turnos podem apresentar início e fim específicos, e os movimentos previstos no Algoritmo 16, consideram os turnos para sua realização. A fim de solucionar tal situação, a ferramenta permite a sobreposição de início e término de período, conforme ilustrado nos turnos da tarde e noite presente na Figura 6.9. Percebe-se que o término do turno da tarde (18:00 - 18:50) sobrepõe o início do período noturno (17:10 - 18:00); assim, caso existam disciplinas alocadas entre estes períodos, os movimentos poderão ser realizados não prejudicando o espaço de busca.

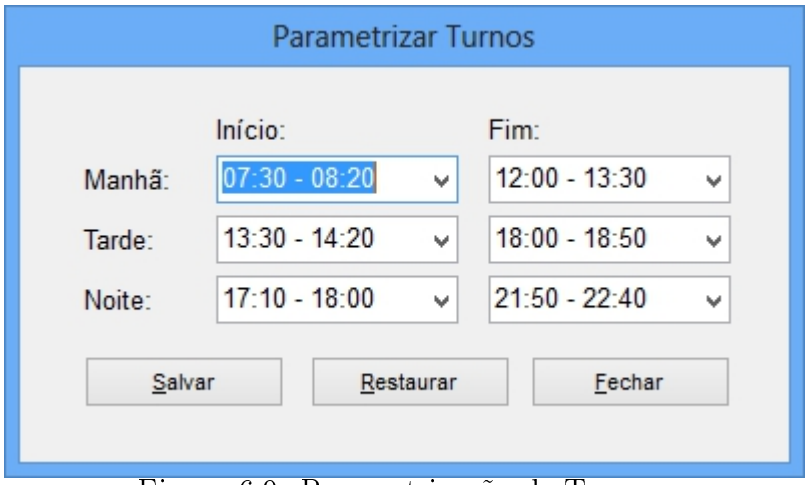


Figura 6.9: Parametrização de Turnos.

6.4.4 Agregação de Turmas

Esta é uma importante funcionalidade do sistema que permite a agregação de turmas, isto é, que duas ou mais turmas de disciplinas que sejam ministradas em um mesmo dia e horário sejam consideradas uma única turma.

Tal situação é necessária tendo em vista que, devido ao regime por créditos da IES (vide seção 4.1), disciplinas comuns que compõem currículos diferentes, frequentemente são ministradas por um mesmo professor em horários concomitantes para turmas diferentes, tendo assim a necessidade de serem lecionadas na mesma sala.

Além da agregação, esta funcionalidade também permite que seja feita a desagregação de uma turma que tenha sido agregada equivocadamente.

O sistema traz algumas opções que permitem resolver tais situações de forma mais ágil:

AutoAgregar Faz-se uma busca de cada disciplina a se alocar procurando-se outras turmas da mesma disciplina que sejam ministradas no mesmo horário e, caso alguma seja encontrada (uma ou mais), faz-se a agregação automática dessas turmas.

Desagregar Tudo Desagrega todas as turmas de disciplinas agregadas.

Agregar Para esta opção é necessária a seleção (tal seleção deve ser realizada clicando-se em uma turma de disciplina desejada. A seguir, segura-se a tecla CTRL e clica-se nas outras turmas da mesma disciplina ofertadas em mesmo horário e dia da semana. Em seguida, clica-se nesta opção e a agregação será, então, realizada.

Desagregar Caso alguma(s) agregação(ões) tenha(m) sido realizada(s) de forma indevida, ou não tenha a necessidade de agregação, basta clicar sobre ela(s) e posteriormente clicar nesta opção, assim ela será desagregada.

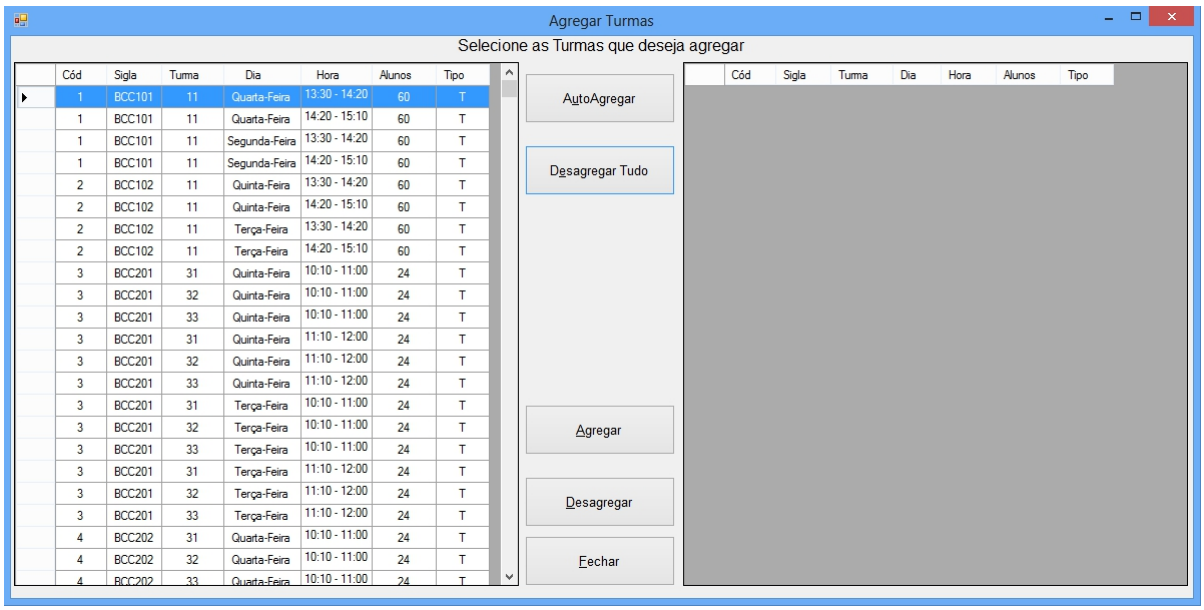


Figura 6.10: Agregação de Turmas.

Para acessar está funcionalidade basta clicar na opção “Agregação” no menu superior e a janela (conforme Figura 6.10) será apresentada.

6.4.5 Penalizações

Esta funcionalidade está relacionada com o descrito na seção 4.2.3 e permite que os pesos associados às penalizações desejadas pelo não atendimento das restrições do problema sejam valoradas.

Quanto melhor for a calibragem destes parâmetros, relacionadas às reais necessidades de atendimento as restrições do problema, melhor será a solução gerada para uma aplicação real da alocação. A alteração destes parâmetros tem reflexo na qualidade da solução final.

A alteração dos pesos pode ser realizada a qualquer momento, mas para que se tenha reflexo na solução o algoritmo deve ser executado novamente.

Para efetuar a alteração destes parâmetros, basta acessar, no menu superior, a opção "Penalizações" e uma janela (conforme Figura 6.11) será exibida.

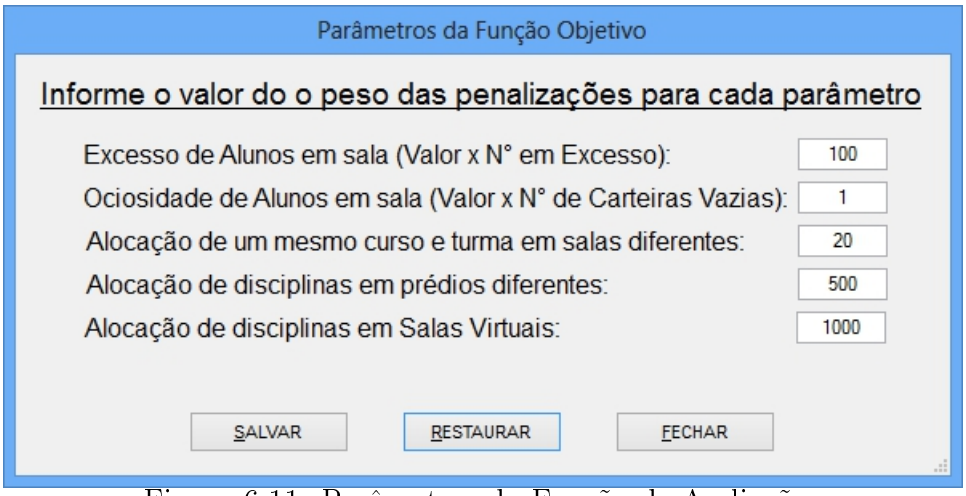


Figura 6.11: Parâmetros da Função de Avaliação.

6.4.6 Construção

A aplicação do algoritmo descrito na seção 5.2 ao horário pré-existente ocorre mediante a execução desta funcionalidade (que pode ser acessada através da opção "Construção" do menu superior).

Para o Algoritmo 15 uma das entradas é o coeficiente de aleatoriedade que pode ser informado para o sistema a partir da janela representada pela Figura 6.12.

A partir desta execução uma solução inicial, respeitando a aleatoriedade escolhida, será apresentada respeitando os recursos da Tela Inicial (vide seção 6.4.1).

A Figura 6.13 apresenta uma solução inicial, totalmente gulosa, ou seja, foi passado o valor zero para o coeficiente de aleatoriedade, gerada a partir de uma instância real e após a realização de uma "AutoAgregação".

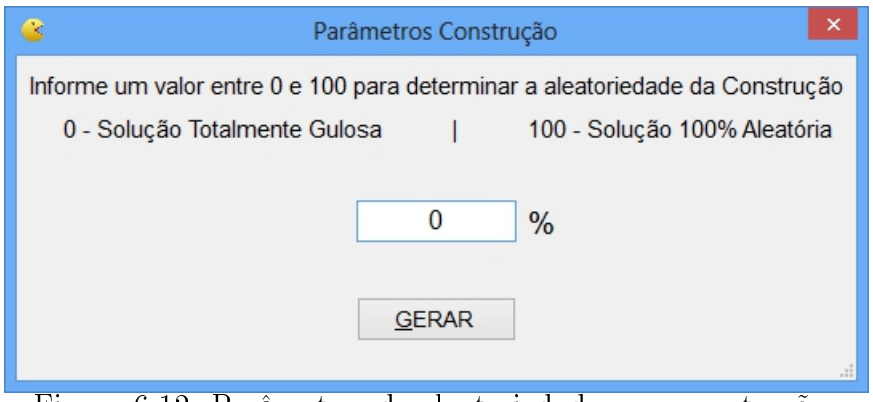


Figura 6.12: Parâmetros de aleatoriedade para construção.

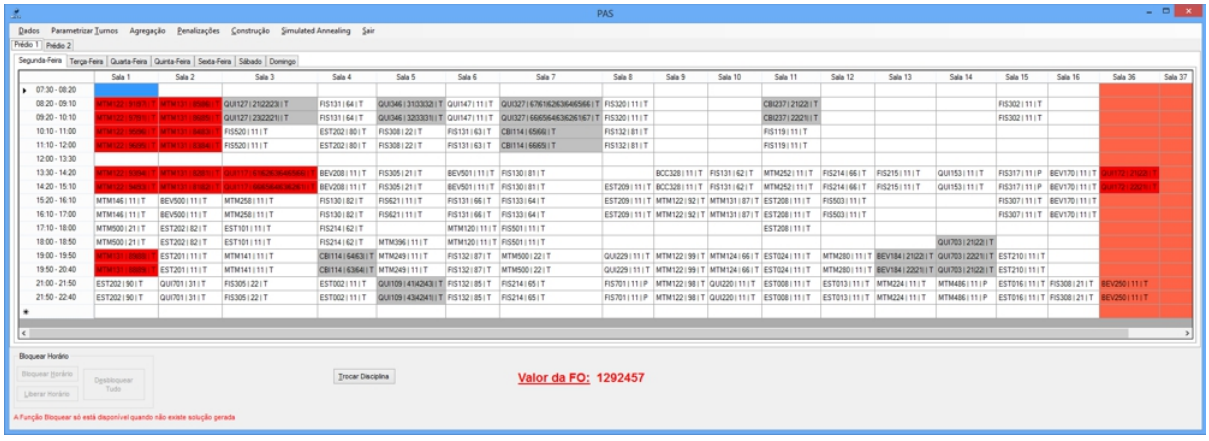


Figura 6.13: Solução Inicial Gerada - Totalmente Gulosa.

6.4.7 *Simulated Annealing*

Diferentemente das demais, esta funcionalidade está dividida em três submenus para atender: as entradas descritas no Algoritmo 16, as parametrizações de probabilidade de escolha da vizinhança conforme descrito no último parágrafo da seção 5.3 e por fim executar o refinamento da solução inicial encontrada.

O submenu "Parâmetros SA" acessado após a o clique na opção "Simulated Annealing" do menu superior é apresentado conforme Figura 6.14 para a parametrização das entradas do Algoritmo 16.

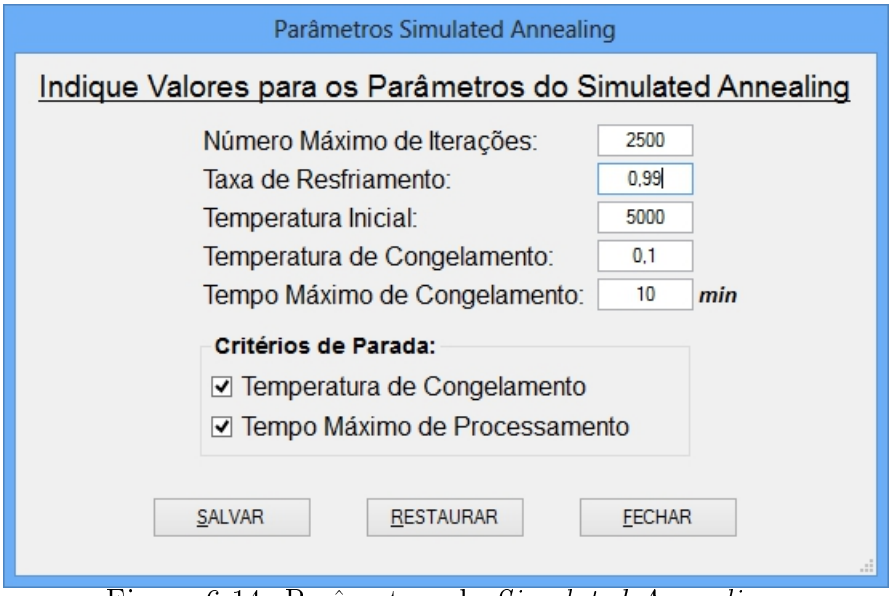


Figura 6.14: Parâmetros do *Simulated Annealing*.

A probabilidade da escolha dos movimentos para escolha das vizinhanças pode ser informada pelo submenu "Movimentos" acessado após a o clique na opção "Simulated Annealing" do menu superior e é apresentado conforme Figura 6.15.

A janela que apresenta o refinamento pela metaheurística *simulated annealing* é acessada mediante o clique na opção "Simulated Annealing" no menu superior e posteriormente no submenu "Executar SA". Importante e relevante ressaltar que algumas informações são apresentadas no momento da execução desta opção como: a contagem instantânea do tempo decorrido, o valor corrente da Função de Avaliação, a quantidade de soluções de melhora apresentadas até o momento, além de um contador de movimentos realizados dividido por tipo de movimento realizado conforme Figura 6.16.

6.4.8 Bloquear Horários

Com a necessidade de atender a situações específicas relacionadas ao atendimento de possíveis "reservas" prévias de salas demandadas por outros cursos (conforme

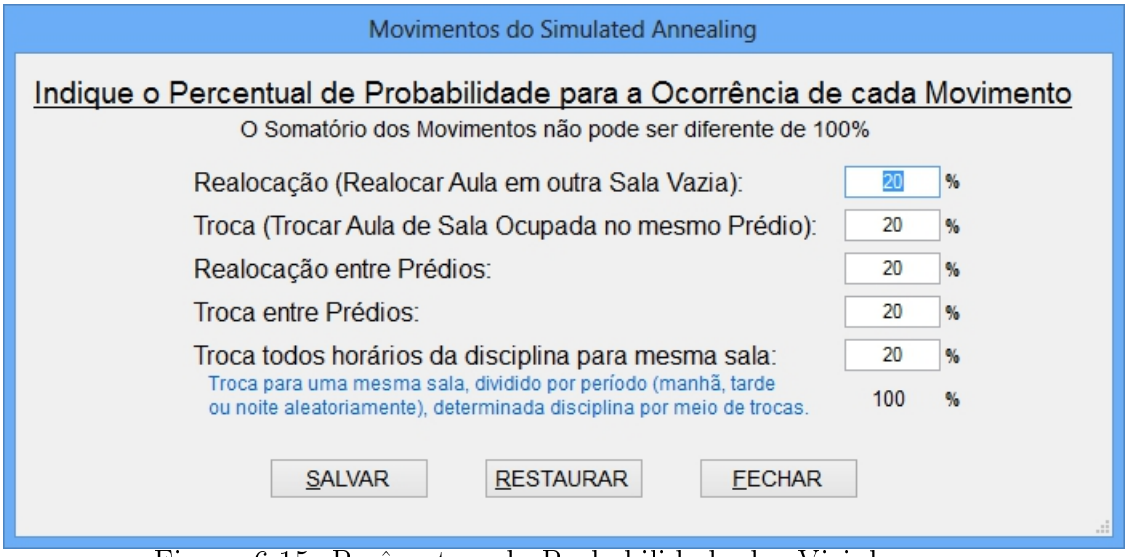


Figura 6.15: Parâmetros da Probabilidade das Vizinhanças.

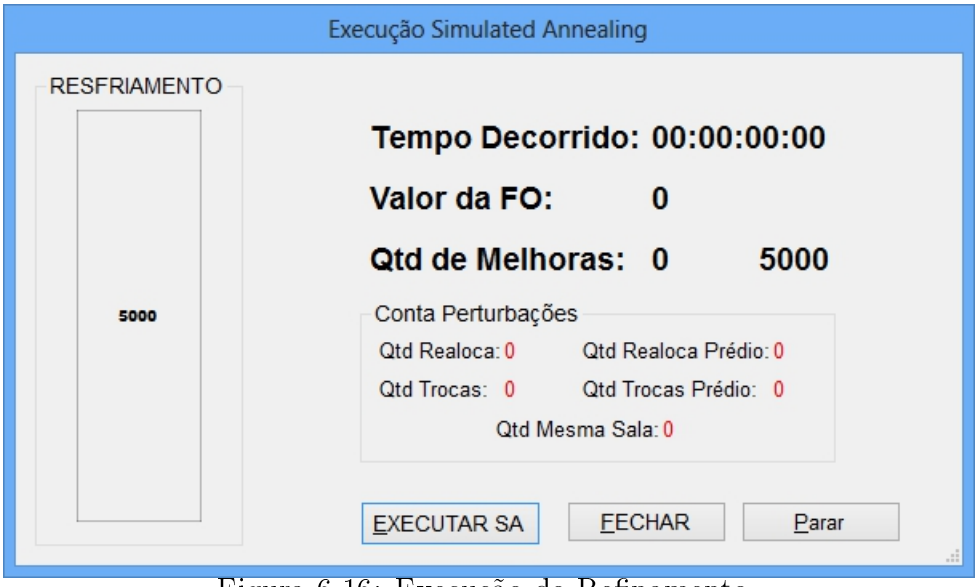


Figura 6.16: Execução do Refinamento.

item 7 da seção 4.1), criou-se esta ferramenta que permite "bloquear" determinados horários em deteminados prédios e dias da semana.

Diferentemente das demais apresentadas até então, esta funcionalidade está disponível em formato de botões, na parte inferior esquerda da tela inicial (vide Figura 6.6). São disponibilizadas três opções:

Bloquear Horário Utilizada para bloquear um ou diversos horários, bastando apenas selecionar os horários desejados e clicar neste botão. Após esta ação, este(s) horário(s) nesta(s) sala(s) não serão considerados na alocação e as células serão caracterizadas como bloqueadas tendo a cor do fundo alterada para cinza e contendo a palavra "BLOQUEADO" (Figura 6.17).

Liberar Horário Utilizada para desbloquear eventuais horários em salas que tenha sido bloqueados de forma equivocada. Para tal basta selecionar a(s) célula(s) desejadas e clicar nesta opção.

Desbloquear Tudo Esta opção realizará a liberação de todas as salas e horários bloqueados.

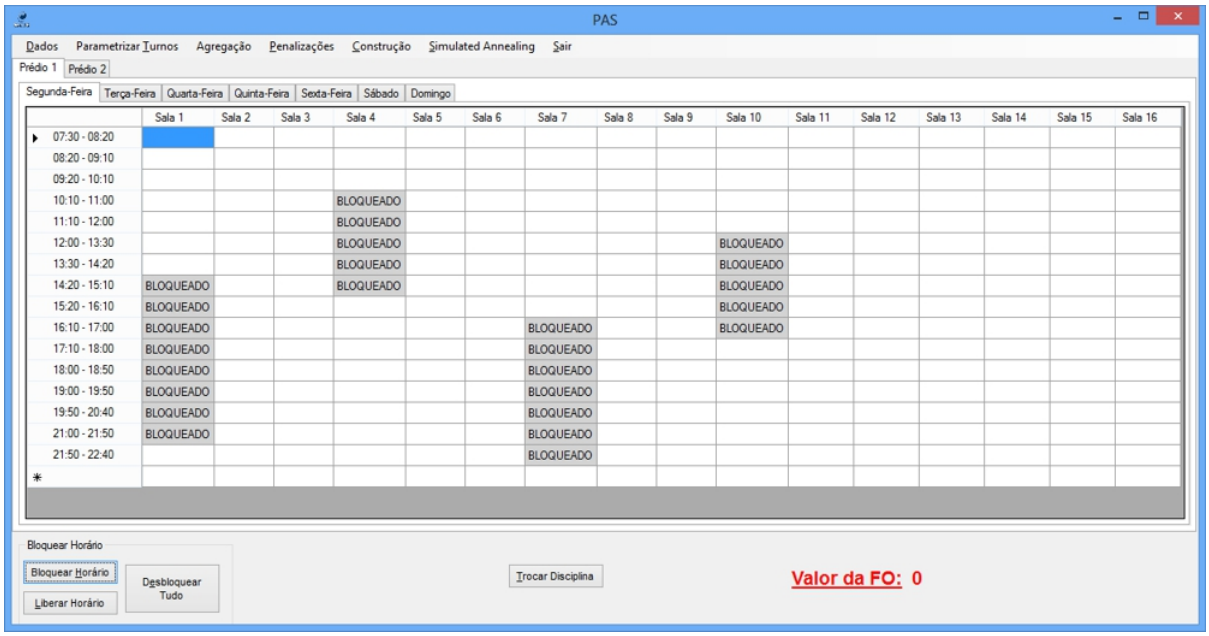


Figura 6.17: Exemplo de bloqueio de horários em salas específicas.

Uma importante particularidade é que esta opção só está disponível (isto é, habilitada) caso nenhuma solução tenha sido gerada. Assim, para utilizá-la o sistema deve estar no seu estado inicial.

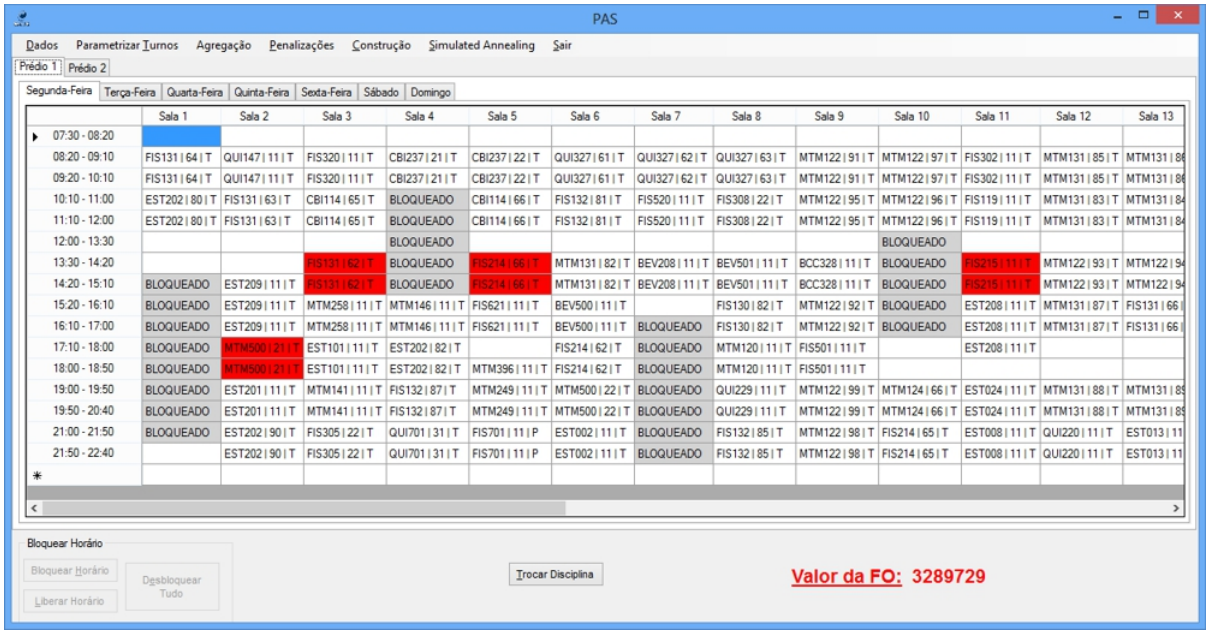


Figura 6.18: Exemplo de solução inicial com horários bloqueados.

6.4.9 Trocar Disciplina

Esta ferramenta está disponível também em formato de botão na tela inicial (vide Figura 6.6) e após uma solução refinada ter sido encontrada, permite a realização dos movimentos (com exceção do quinto movimento) apresentados na subseção 4.2.2 de forma manual e não automática.

Sua utilização refere-se à troca de turmas de disciplina entre salas e está sujeita a duas validações:

- A troca só pode ocorrer, independentemente dos prédios, para disciplinas ministradas nos mesmos horários e dias da semana. No caso de uma turma de disciplina ter um número de aulas maior que o da outra turma de disciplina a ser trocada, é necessário que haja horário vago na sala que receberá as aulas dessa turma maior, de forma a viabilizar a troca.

- A realocação de uma turma de disciplina para outra sala também só é permitida se houver, nos mesmos horários, disponibilidade nessa outra sala.

Para efetuar os movimentos basta selecionar as disciplinas/turmas das salas diferentes e clicar no botão.

Capítulo 7

Resultados Obtidos

7.1 Considerações Iniciais

Os testes foram realizados em um microcomputador com processador Intel CORE I5-2450M CPU 2,50GHz 2,49GHz, com 4 GB de memória RAM, sob o sistema operacional Windows 8 64bits. Utilizou-se, como teste, a instância real fornecida pela UFOP, caracterizada conforme Capítulo 4.

Dado o caráter estocástico do Algoritmo, ele foi executado trinta vezes nessa instância. Os pesos para a função de avaliação foram fixados nos valores especificados na Tabela 7.1.

Tabela 7.1: Penalizações da Função Objetivo

Penalizações	Valor
Excesso de Alunos	100
Ociosidade de Carteiras	1
Mesma Turma/Curso em Salas Diferentes	20
Turma/Curso em Prédios Diferentes	500
Turma/Curso em Salas Virtuais	200

7.2 Resultados

A solução inicial foi gerada de forma totalmente gulosa. Assim, o valor inicial para a função objetivo foi sempre 767902. O movimento aplicado a cada iteração do *Simulated Annealing* foi escolhido dando-se a mesma chance a cada um dos cinco movimentos possíveis. Os valores dos parâmetros do *Simulated Annealing* foram fixados conforme Tabela 7.2.

A Tabela 7.3 apresenta algumas características das soluções geradas. Como pode ser observado, a solução inicial foi melhorada em 15%, em média, com a aplicação da fase de refinamento.

A ocorrência de turmas alocadas em salas virtuais, apesar de gerar uma solução inviável, é comum em problemas de alocação de salas. Este fato acontece porque

Tabela 7.2: Parametrizações do *Simulated Annealing*

Parâmetros	Valor
Iterações por Temperatura	2500
Taxa de Resfriamento	0,98
Temperatura Inicial	5000
Temperatura de Congelamento	0,1

Tabela 7.3: Resultados das Execuções

Item	Valor
Valor inicial da função de avaliação	767902
Tempo Médio de Execução	98 min
Valor médio da função de avaliação	649738
Melhor valor da função de avaliação	643570
Salas virtuais criadas no Pavilhão	3
Salas virtuais criadas no ICEB	3

há concentração de aulas em poucos dias, normalmente às terças, quartas e quintas-feiras. Na solução apresentada, essas são as menores turmas. A solução encontrada mostra que a diretoria do ICEB tem a tarefa de encontrar salas em outras unidades acadêmicas, ou, caso isso não seja possível, alocar essas turmas pequenas a outros espaços do Instituto, em geral, em laboratórios.

Capítulo 8

Conclusões Gerais e Trabalhos Futuros

8.1 Conclusões Gerais

Este trabalho apresentou um sistema computacional destinado a resolver o Problema de Alocação de Salas do Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto (UFOP). Para cumprir esse objetivo o sistema aplica um algoritmo de otimização que parte de uma solução inicial construída de forma gulosa e a refina pela metaheurística *Simulated Annealing*.

A fase de refinamento utiliza cinco movimentos para explorar o espaço de soluções, a saber: Realocação de turmas, Troca de turmas entre salas, Realocação de turmas entre Prédios, Troca de turmas entre Prédios e Disciplina/Turno/Curso/Período para mesma sala.

O sistema desenvolvido incluiu duas novas funcionalidades ao sistema previamente existente para resolver esse problema. Assim, turmas de disciplinas de mesmo período e curso procuram ser alocadas a uma mesma sala. A outra funcionalidade incorporada é a possibilidade de se trabalhar com mais de um prédio, no caso, com o Pavilhão Central de Aulas (PCA). Assim, uma turma de disciplina pode ser alocada tanto no prédio principal do Instituto, quanto no PCA.

Os resultados mostraram que a fase de refinamento foi capaz de melhorar a solução inicial em até 15%. Além disso, o sistema desenvolvido inclui facilidades ao usuário para manipular a solução, evitando a ocorrência de erros caso a solução fosse gerada manualmente.

8.2 Trabalhos Futuros

Sugerem-se os seguintes trabalhos futuros:

(i) Implementação de novas heurísticas para construção de soluções iniciais;

(ii) Implementação de outras metaheurísticas para refinamento a fim de comparar o desempenho das mesmas na solução do problema;

(iii) Inclusão no algoritmo de métodos de intensificação e diversificação;

(iv) Inclusão da possibilidade de gerenciar o uso das restrições, permitindo a inclusão e exclusão de acordo com a necessidade do ambiente estudado;

(vi) Aprimoramento da ferramenta no que se diz respeito à entrada de dados, permitindo a administração destes de forma mais "amigável".

Referências Bibliográficas

Al-Yakoob, Salem M. e Sherali, Hanif D. (2006). Mathematical programming models and algorithms for a class-faculty assignment problem. *European Journal of Operational Research*, v. 173, p. 488–507.

Alfaro, Horacio Martinez e Terán, Gerard Flores. (1998). Solving the classroom assignment problem with simulated annealing. *IEEE International Conference on Systems, Man, and Cybernetics*, v. 4, p. 3703 – 3708.

Antunes, António Pais e Peeters, Dominique. (2001). On solving complex multi-period location models using simulated annealing. *European Journal of Operational Research*, v. 130, n. 1, p. 190–201. URL `http://dblp.uni-trier.de/db/journals/eor/eor130.html`.

Barbosa, S. H. D. (2011). Resolução do problema de programação de cursos universitários baseada em currículos via meta-heurísticas. Dissertação de mestrado, Programa de Pós-Graduação em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG, Belo Horizonte.

Bardadym, V. A. (1996). Computer-aided school and university timetabling: The new wave. volume 1153 of *Lecture Notes in Computer Science*, p. 22–45. Springer-Verlag, Berlin.

Beyrouthy, Camille; Burke, Edmund K.; Landa-Silva, J. Dario; McCollum, Barry; McMullan, Paul e Parkes, Andrew J. (2006). Towards improving the utilisation of university teaching space. *PATAT*, v. , p. 103–122.

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, v. 35, n. 3, p. 268–308.

Blum, Christian; Aguilera, Maria J. B.; Roli, Andrea e Sampels, Michael. (2008). *Hybrid metaheuristics: studies in computational intelligence*, Capítulo Hybrid metaheuristics: an introduction, p. 1–30. Springer, Berlin/Heidelberg.

Burke, E. K.; Cowling, P.; Silva, J.D. Landa; Silva, A e McCollum, Barry. (2001). Three methods to automate the space allocation process in uk universities. *Lecture Notes in Computer Science*, v. 2079, p. 254–276.

Burke, E. K. e Silva, A. (2004). The design of memetic algorithms for scheduling and timetabling problems. *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, p. 289–312. Springer, (2004).

Burke, E. K. e Varley, D. B. (1997). Space allocation: An analysis of higher education requirements. *Lecture Notes in Computer Science*, v. 1408, p. 20–36.

Burke, Edmund K. e Newall, James P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE Trans. Evolutionary Computation*, v. 3, n. 1, p. 63–74. URL `http://dblp.uni-trier.de/db/journals/tec/tec3.html`.

Carter, Michael W. e Laporte, Gilbert. (1997). Recent developments in practical course timetabling. Burke, Edmund K. e Carter, Michael W., editors, *PATAT*, volume 1408 of *Lecture Notes in Computer Science*, p. 3–19. Springer, (1997). ISBN 3-540-64979-4. URL `http://dblp.uni-trier.de/db/conf/patat/patat1997.html`.

Carter, Michael W. e Tovey, Craig A. (1992). When is the classroom assignment problem hard? *Operations Research*, v. 40, p. 28–39.

Castro, Otávio Mello. (2003). Resolução do problema de alocação de salas de aula via simulated annealing. Monografia. (Ciência da Computação) - Departamento de Ciência da Computação, Universidade Federal de Ouro Preto.

Chaves, Antonio Augusto. *Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória*. PhD thesis, Instituto Nacional de Pesquisas Espaciais (INPE), (2009).

Cooper, Tim B. e Kingston, Jeffrey H. (1996). *The complexity of timetable construction problems*, volume 1153 of *Lecture Notes in Computer Science*, p. 281–295. Springer Berlin Heidelberg.

Cordenonsi, Andre Zanki. (2008). Ambientes, objetos e dialogicidade : uma estratégia de ensino superior em heurísticas e metaheurísticas. Tese de doutorado, Programa de Pós-Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul - UFRGS, Porto Alegre.

Dammak, Abdelaziz; Elloumi, Abdelkarim e Kamoun, Hichem. (2006). Classroom assignment for exam timetabling. *Advances in Engineering Software*, v. 37, p. 659–666.

Dorigo, Marco. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, (1992).

Dowsland, Kathryn A. (1993). Modern heuristic techniques for combinatorial problems. p. 20–69. John Wiley & Sons, Inc., New York, NY, USA. ISBN 0-470-22079-1. URL `http://dl.acm.org/citation.cfm?id=166648.166653`.

Dowsland, Kathryn A. (1998). Off-the-peg or made-to-measure? timetabling and scheduling with sa and ts. *Lecture Notes in Computer Science*, v. 1408, p. 37–52.

Eikelder, H. M. M. ten e Willemen, R. J. (2001). Some complexity aspects of secondary school timetabling problems. *Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, PATAT '00, p. 18–27, London, UK, UK. Springer-Verlag. ISBN 3-540-42421-0. URL `http://dl.acm.org/citation.cfm?id=646431.692906`.

Evans, James e Minieka, Edward. (1992). *Optimization Algorithms for Networks and Graphs*. CRC Press, USA.

Even, S.; Itai, A. e Shamir, A. (1976). On the complexity of time table and multi-commodity flow problems. *SIAM Journal of Computation*, v. 5, p. 691–703.

Feo, Thomas A. e Resende, Maurício G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133.

Garey, Michael R. e Johnson, David S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.

Gaspar-Cunha, Antônio; Takahashi, Ricardo e Antunes, Carlos Henggeler. (2012). *Manual de Computação Evolutiva e Metaheurística*. UFMG, Belo Horizonte.

Glover, F. e Kochenberger, G.A. (2003). *Handbook of Metaheuristics*. International series in operations research & management science. Springer. ISBN 9781402072635.

Glover, Fred. may(1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, v. 13, n. 5, p. 533–549. ISSN 0305-0548. doi: 10.1016/0305-0548(86)90048-1. URL `http://dx.doi.org/10.1016/0305-0548(86)90048-1`.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edição. ISBN 0201157675.

Hasen, P. e Mladenovic, N. (1999). An introduction to variable neighborhood search. S. Voß, I. OsmanS. Martello e Roucairol, C., editors, *Metaheuristics: Advances and trends in local search paradigms for optimization*, p. 433–438. Kluwer Academic Publishers, Norwell, MA, EUA.

Hertz, A. (1992). Tabu search for large scale timetabling problems. *European Journal of Operational Researsh*, v. 54, p. 39–47.

Kelly, James P. (1996). *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA. ISBN 0792397002.

Kirkpatrick, S.; Gelatt, C. D. e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, v. 220, p. 671–680.

Kripka, Rosana Maria Luvezute e Kripka, Moacir. (2010). Simulated annealing aplicado na otimização da alocação de salas em instituição de ensino superior. *Mecánica Computacional*, v. XXIX, n. 95, p. 9317–9325. URL `http://www.cimec.org.ar/ojs/index.php/mc/article/viewFile/3671/3583`.

Lourenço, H. R.; Martin, O. e Stützle, T. (2002). Iterated local search. Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, p. 321–353. Kluwer Academic Publishers, Norwell, MA.

Michalewicz, Zbigniew e Schoenauer, Marc. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, v. 4, p. 1–32.

Mladenović, N. e Hansen, P. November(1997). Variable neighborhood search. *Comput. Oper. Res.*, v. 24, n. 11, p. 1097–1100. ISSN 0305-0548. doi: 10.1016/S0305-0548(97)00031-2. URL `http://dx.doi.org/10.1016/S0305-0548(97)00031-2`.

Moscato, Pablo. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Relatório Técnico C3P 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, CA.

Moscato, Pablo e Cotta, Carlos. oct(2003). Una introducción a los algoritmos meméticos. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, , n. 19, p. 131–147. ISSN 1137-3601.

Noronha, Thiago Ferreira. (2000). Uma abordagem sobre estratégias metaheurísticas. Projeto. (Informática de Matemática) - Departamento Informática de Matemática de Aplicada, Universidade Federal do Rio Grande do Norte.

Pereira, Marconi A. e Vasconcelos, Joao A. (2012). *Manual de Computação Evolutiva e Metaheurística*, Capítulo 8, *Recozimento Simulado*, p. 163–176. UFMG, Belo Horizonte.

Politano, Gustavo Lico Cunha. (2006). Modelo e algoritmo para alocação de espaço físico. Monografia - (Ciência da Computação) Faculdade de Ciência da Computação, Universidade Federal de Maringá.

Qu, Rong. *Case-Based Reasoning for Course Timetabling Problems*. phd, August(2002). URL `http://www.cs.nott.ac.uk/ rxq/files/PhDThesis.pdf`.

Resende, Maurício G. C.; Mateus, Geraldo R. e Silva, Ricardo M. A. (2012). *Manual de Computação Evolutiva e Metaheurística*, Capítulo 10, *GRASP: Busca Gulosa Aleatorizada e Adaptativa*, p. 203–2013. In , Gaspar-Cunha et al. (2012).

Santos, Haroldo Gambini e Souza, Marcone Jamilson Freitas. (2007). Programação de horários em instituições educacionais: Formulações e algoritmos. *XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO*, v. 39, p. 2827–2882.

Schaerf, A. (1999). A survey of automated timetabling. *Lecture Notes in Computer Science*, v. 13, p. 87–127.

Silva, Amanda Sávio Nascimento. (2005). Estudo e implementação, mediante recozimento simulado, do problema de alocação de salas. Monografia. (Ciência da Computação) - Departamento de Ciência da Computação, Universidade Federal de Lavras.

Souza, Marcone Jamilson Freitas. (2000). Programação de horários em escolas: uma aproximação por metaheurísticas. Tese de doutorado, Programa de Pós-Graduação em Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro - UFRJ, Rio de Janeiro.

Souza, Marcone Jamilson Freitas. (2014). *Inteligência Computacional para Otimização - Notas de aula*. URL `http://www.iceb.ufop.br/decom/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm`. Acesso em: 20 jan. 2008.

Souza, Marcone Jamilson Freitas; Martins, Alexandre Xavier e de Araujo, Cássio Roberto. (2002). Experiências com simulated annealing e busca tabu na resolução do problema de alocação de salas. *XXXIV Simpósio Brasileiro de Pesquisa Operacional*, v. 34, p. 1100–1110.

Subramanian, Anand; Medeiros, José Maurício Fernandes; Cabral, Lucídio Formiga e Souza, Marcone Jamilson Freitas. (2011). Aplicação da metaheurística busca tabu ao problema de alocação de aulas a salas em uma instituição universitária. *Revista Produção Online*, v. 11, p. 54–75.

Takahashi, Ricardo H. C. e Gaspar-Cunha, Antônio. (2012). *Manual de Computação Evolutiva e Metaheurística*, Capítulo 1, *Introdução*, p. 1–21. In , Gaspar-Cunha et al. (2012).

Talbi, El-Ghazali. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing. ISBN 0470278587, 9780470278581.

Ueda, Hiroaki; Ouchi, Daisuke; Takahashi, Kenichi e Miyahara, Tetsuhiro. (2001). A co-evolving timeslot/room assignment genetic algorithm technique for university timetabling. *Lecture Notes in Computer Science*, v. 2079, p. 48–63.

Weise, Thomas. (2008). *Global Optimization Algorithms - Theory and Application*. it-weise.de (self-published): Germany. URL `http://www.it-weise.de/projects/book.pdf`.