Uma Proposta de Solução para Problemas de Horário Educacional utilizando Busca Dispersa e Reconexão por Caminhos

por

Morgana Spindler

UNIVERSIDADE DO VALE DO RIO DOS SINOS

MORGANA SPINDLER

Uma Proposta de Solução para Problemas de Horário Educacional utilizando Busca Dispersa e Reconexão por Caminhos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Prof. Dr. Leonardo D. Chiwiacowsky
Orientador

São Leopoldo, março de 2010

Aos meus pais.

“A persistência é o caminho do êxito.”
— SIR CHARLES SPENCER CHAPLIN

AGRADECIMENTOS

Gostaria de agradecer a todos que de alguma forma contribuiram para a execução deste trabalho:

- Ao meu orientador, professor Dr. Leonardo não apenas pelo conhecimento compartilhado, mas pela paciência e disponibilidade;
- Às minhas amigas Maira e Ariel pela ajuda nas correções e dicas de Java;
- Aos meus pais pela compreensão em todas as vezes que não me pude fazer presente;
- Ao professor Dr. Arthur e ao banco Santander que, com a ajuda financeira através da bolsa de estudo, tornaram a realização deste mestrado possível.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 8

LISTA DE TABELAS 9

RESUMO 10

ABSTRACT 11

1 INTRODUÇÃO 12
1.1 Motivação e Contribuição 13
1.2 Objetivos 13
1.2.1 Objetivo Principal 13
1.2.2 Objetivos Secundários 13
1.3 Organização do Trabalho 14

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Otimização Combinatória 15
2.1.1 Complexidade Computacional 16
2.2 Horário Educacional 17
2.2.1 Restrições do Problema 18
2.2.2 Viabilidade x Otimalidade 18
2.2.3 Classificações do Problema 19
2.3 Métodos de Solução 21
2.3.1 Heurísticas x Metaheurísticas 21
2.3.2 Intensificação x Diversificação 22
2.3.3 Vizinhanças e Movimentos 22
2.3.4 Heurísticas Construtivas x Heurísticas de Melhoramento 23
2.3.5 Busca em Trajetória x Busca populacional 24
2.3.6 Busca Dispersa 25
2.3.7 Reconexão por Caminhos 27

3 TRABALHOS RELACIONADOS 30
3.1 Formulações do Problema 30
3.2 *Simulated Annealing* 31
3.3 Busca Tabu 31
3.4 Algoritmos Genéticos 32
3.5 Técnicas Híbridas 32

3.6 Busca Dispersa . . . 33
3.7 Reconexão de Caminhos . . . 34
3.8 Métodos de Pesquisa em Vizinhança . . . 34

4 FORMULAÇÃO DO PROBLEMA . . . 36
4.1 Definição do Problema . . . 36
4.1.1 Representação da Solução . . . 37
4.1.2 Custo da Solução . . . 37
4.1.3 Restrições . . . 38
4.1.4 Viabilidade e Otimalidade . . . 42
4.1.5 Função Objetivo . . . 43
4.2 Formulação Matemática . . . 44
4.3 Estudos de caso . . . 47
4.3.1 Estudo de Caso 1 . . . 48
4.3.2 Estudo de Caso 2 . . . 49

5 MÉTODO DE SOLUÇÃO . . . 51
5.1 Aplicação . . . 51
5.2 Algoritmo Implementado . . . 51
5.2.1 Definição de Parâmetros . . . 53
5.2.2 Método de Geração de Soluções Diversas . . . 54
5.2.3 Método de Melhoria . . . 56
5.2.4 Método de Construção do Conjunto de Referência . . . 61
5.2.5 Método de Geração de Subconjuntos . . . 62
5.2.6 Método de Combinação . . . 63
5.2.7 Método de Atualização do Conjunto de Referência . . . 63

6 VALIDAÇÃO E TESTES . . . 65
6.1 Ambiente de Testes . . . 65
6.2 Instâncias de Problema . . . 65
6.2.1 Descrição das Instâncias . . . 65
6.2.2 Pesos Atribuídos às Restrições . . . 66
6.3 Ajuste de Parâmetros . . . 67
6.3.1 Ajuste do Tamanho e Ciclos de Melhoria de P . . . 67
6.3.2 Ajuste do Tamanho de $RefSet$ . . . 68
6.3.3 Ajuste da Melhoria de $Pool$ . . . 69
6.4 Validação do Modelo . . . 70
6.5 Testes de Comparação . . . 72
6.5.1 Testes com a Instância de Competição . . . 72
6.6 Testes com a Instância do Estudo de Caso 2 . . . 75

7 CONCLUSÃO E TRABALHOS FUTUROS . . . 78
7.1 Considerações e trabalhos futuros . . . 79

REFERÊNCIAS BIBLIOGRÁFICAS . . . 81

LISTA DE ABREVIATURAS E SIGLAS

FO - Função objetivo

GRASP - *Greedy Randomized Adaptive Search Procedure*

ITC - *International Timetabling Competition*

NP - Classe de problemas de tempo polinomial não determinístico

P - Classe de problemas de tempo polinomial determinístico

PATAT - *Practice and Theory on Automated Timetabling*

PR - *Path Relinking*

R01 - restrição *Aulas*

R02 - restrição *Conflitos*

R03 - restrição *Ocupação das salas*

R04 - restrição *Disponibilidade*

R05 - restrição *Capacidade das salas*

R06 - restrição *Mínimo de dias de aula*

R07 - restrição *Aulas isoladas (Compactação de currículo 1)*

R08 - restrição *Janelas (Compactação de currículo 2)*

R09 - restrição *Estabilidade de sala*

R10 - restrição *Mínimo e máximo de aulas por dia*

R11 - restrição *Deslocamentos*

R12 - restrição *Adequação de salas*

R13 - restrição *Agrupamento de aulas*

SS - *Scatter Search*

UD2 - Formulação *International Timetabling Competition*

UD3 - Formulação *International Timetabling Competition*

UD4 - Formulação *International Timetabling Competition*

UD5 - Formulação *International Timetabling Competition*

LISTA DE FIGURAS

Figura 2.1: Ótimo global e ótimo local. 16
Figura 2.2: Classes de Complexidade Computacional. 17
Figura 2.3: Estrutura do algoritmo Busca Dispersa canônico. 27
Figura 2.4: Esquema de funcionamento da heurística *Path Relinking*. . . . 28
Figura 2.5: Estrutura do algoritmo Reconexão por Caminhos canônico. . . 29

Figura 4.1: Pesos das restrições do formato UD. 48
Figura 4.2: Problema de horário departamental. 49

Figura 5.1: Fluxo de processos da ferramenta de solução. 51
Figura 5.2: Algoritmo Busca Dispersa (SS) implementado. 52
Figura 5.3: Algoritmo do Método de Geração de Soluções Diversas. 55
Figura 5.4: *Algoritmo Estratégia de Alocação. 56
Figura 5.5: **Algoritmo Definição de Salas Disponíveis. 56
Figura 5.6: Movimento de Inserção de sala I_S. 57
Figura 5.7: Movimento de Realocação de Horário R_H. 57
Figura 5.8: Movimento de Realocação de Horário com Inserção de Sala $R_H I_S$. 58
Figura 5.9: Movimento de Troca de Horários T_H. 58
Figura 5.10: Movimento de Troca de Salas T_S. 58
Figura 5.11: Movimento de Troca de Disciplinas T_D. 59
Figura 5.12: Algoritmo Método de Melhoria. 61
Figura 5.13: Cálculo do índice de diversidade da solução. 62

Figura 6.1: Evolução da melhor solução em $RefSet$. 72
Figura 6.2: Evolução da melhor solução em $RefSet$. 75
Figura 6.3: Evolução da melhor solução em $RefSet$. 77

LISTA DE TABELAS

Tabela 6.1: Pesos das restrições do formato UD (ITC). 67
Tabela 6.2: Ajuste de tamanho e ciclos de melhoria de P. 68
Tabela 6.3: Ajuste do tamanho de $RefSet$. 68
Tabela 6.4: Ajuste da melhoria de $Pool$. 69
Tabela 6.5: Parâmetros utilizados nos testes de validação. 70
Tabela 6.6: Testes TOY_{11}. 70
Tabela 6.7: Testes TOY_{12}. 71
Tabela 6.8: Testes TOY_{13}. 71
Tabela 6.9: Parâmetros dos testes ITC. 73
Tabela 6.10:Testes ITC_{11}. 73
Tabela 6.11:Testes ITC_{12}. 74
Tabela 6.12:Testes ITC_{13}. 74
Tabela 6.13:Parâmetros dos testes $REAL$. 76
Tabela 6.14:Testes $REAL_{11}$. 76
Tabela 6.15:Testes $REAL_{12}$. 76

RESUMO

Este trabalho aborda o uso de uma metaheurística populacional para a solução do problema de otimização conhecido, na Pesquisa Operacional, como Programação de Horário de Cursos Baseada em Currículos. O problema de Programação de Horário de Cursos Baseada em Currículos consiste na construção das grades de horário de cursos em instituição de ensino que indicam em quais períodos semanais cada disciplina destes cursos deverá ocorrer, alocando professores e salas e respeitando um conjunto de requisitos organizacionais, pedagógicos e pessoais. Este trabalho apresenta uma formulação matemática para o problema e especifica um algoritmo de solução baseado na técnica metaheurística Busca Dispersa, combinada com o método de Reconexão por Caminhos. Além disso, é apresentado o registro de testes realizados com instâncias de problemas utilizadas na *International Timetabling Competition* e também em um problema real de uma instituição local de esino superior.

Palavras-chave: Pesquisa Operacional. Otimização Combinatória. Horário Educacional. *Timetabling*. Busca Dispersa. Reconexão por Caminhos. *Scatter Search*. Path Relinking.

ABSTRACT

This paper discusses the use of a populational metaheuristic to solve the optimization problem known in Operational Research, as Curriculum Based Timetabling. The Curriculum Based Timetabling problem is the construction of schedule of courses in educational institutions that indicate which weekly times each subject of these courses should occur, allocating rooms and teachers and a respecting a set of organizational, pedagogical and personal requirements. This paper presents a mathematical formulation for the problem and specify a solution algorithm based on the Scatter Search metaheuristic technique, combined with the method Path Relinking. Furthermore, it is present the record of tests with instances of problems used in the International Timetabling Competition and also a real problem of a local institution.

Keywords: Operation Research, Combinatorial Optimization, Timetabling, Scatter Search, Path Relinking.

1 INTRODUÇÃO

Ao longo dos últimos 40 anos, aproximadamente, a comunidade científica esforça-se para buscar uma solução computacional que auxilie em um antigo, longo e duro processo: a alocação de recursos sob restrições. Um dos problemas típicos desta natureza é o Problema de Programação de Horários Educacionais. O ajuste de interesses entre disponibilidades de salas, professores e alunos ao longo de determinados períodos da semana, é tarefa árdua e exige tempo. Contudo, é essencial e periódica em instituições de ensino, seja semestral ou anualmente, já que todas as atividades são pautadas sobre o período letivo. Esse problema tem sido objeto de pesquisa por parte da comunidade científica, tanto devido ao aumento da sua complexidade, quanto pela busca de melhores caminhos para solucioná-lo.

Segundo Cooper [1], o Problema de Programação de Horários Educacionais pode ser modelado como um problema matemático de Otimização Combinatória de complexidade NP-completo, mundialmente referido como *Timetabling Problem* e de grande relevância na área de Pesquisa Operacional. Por sua vez, Wren [2] define o problema de *Timetabling* como "a alocação, sujeita a restrições, de recursos a objetos colocados no espaço e no tempo, de modo a satisfazer, tanto quanto possível, um conjunto de objetivos desejáveis".

Dentre as diversas variações do Problema de Programação de Horários Educacionais, este trabalho aborda o problema de Programação de Horário de Cursos Baseada em Currículos. Nesta variação do problema de Horário Educacional, deve-se realizar o agendamento de vários cursos em um conjunto de períodos semanal, considerando também a alocação de um número limitado de salas, sendo que os conflitos entre cursos são determinados de acordo com o currículo dos cursos e não de acordo com as matrículas efetuadas. Neste caso, as turmas representam os módulos curriculares.

A dificuldade com os problemas de horário é puramente matemática, uma vez que problemas desse tipo são problemas de associação, de natureza combinatória. Os métodos computacionais exatos geralmente trabalham com enumeração do espaço de soluções, mesmo que implicitamente, limitando severamente o tamanho das instâncias que podem ser resolvidas em tempos computacionais razoáveis.

Nos últimos anos, tem-se constatado avanços consideráveis com a aplicações de heurísticas e metaheurísticas, na tentativa de se determinar soluções aproximadas de problemas complexos de otimização combinatória.

Uma heurística é um procedimento algorítmico desenvolvido através de um modelo cognitivo, usualmente através de regras baseadas na experiência dos desenvolvedores. Como as heurísticas são algoritmos específicos para um caso, é recomendável que se tenha um conhecimento específico do problema que está sendo abordado. As metaheurísticas consistem em um processo iterativo que conduzem e modificam as operações de uma heurística subordinada, de tal forma a produzir, eficientemente, soluções de alta

qualidade. Podem ser aplicadas a um amplo conjunto de problemas, bastando pequenas modificações para adaptá-las a um problema específico [3].

Neste trabalho, um método de solução que combina a metaheurística Busca Dispersa [4], ou *Scatter Search*, com a heurística Reconexão por Caminhos [5] é aplicado para solução do problema de Programação de Curso Baseada em Currículos. O trabalho tem como objetivo mostrar a viabilidade do emprego destas técnicas para a solução do referido problema, já que na literatura, o autor não encontrou trabalhos que reportem resultados utilizando esta abordagem.

1.1 Motivação e Contribuição

Vários trabalhos sobre Problemas de Horários foram publicados nos últimos anos e conferências regulares discutem o tema no meio científico. Um evento de grande relevância na área é o PATAT[1] (*Practice and Theory on Automated Timetabling*), que teve seu início em 1995 e ocorre com periodicidade de dois anos. Da mesma forma, a *International Timetabling Competition* - ITC é organizada anualmente pela *Metaheuristics Network*[2]. Nesta competição, os participantes desenvolvem programas para resolver problemas de Programação de Curso Baseada em Currículos, dentre outras classificações.

Muitos dos trabalhos apresentados nestes eventos avaliam a eficiência de métodos metaheurísticos como Busca Tabu [6], *Simulated Annealing* ou Têmpera Simulada [7] e Algoritmos Genéticos [8], porém, resultados obtidos com o método Busca Dispersa para problemas de Programação de Curso Baseada em Currículos não foram encontrados até a finalização deste texto. Uma das contribuições científicas deste trabalho é, portanto, o registro do emprego desta técnica para solução de problemas deste tipo.

1.2 Objetivos

1.2.1 Objetivo Principal

O objetivo principal deste trabalho é propor um formulação matemática e um algoritmo de solução baseado na técnica Busca Dispersa, combinada com o método de Reconexão por Caminhos, para solução do problema de Programação de Horário de Cursos Baseada em Currículos.

1.2.2 Objetivos Secundários

- Avaliar diferentes estruturas de vizinhança para solução de problemas de Programação de Curso Baseada em Currículos;

- Avaliar a técnica Busca Dispersa e Reconexão por Caminhos com problemas reais de Programação de Curso Baseada em Currículos;

- Avaliar o desempenho da técnica Busca Dispersa e Reconexão por Caminhos levando em consideraçcao diferentes conjuntos de restrições.

[1] `http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml`
[2] `http://www.metaheuristics.net/`

1.3 Organização do Trabalho

O capítulo 2 tem por objetivo apresentar o problema estudado e a técnica de solução abordada, através da introdução de fundamentos teóricos básicos. Inicialmente são apresentados conceitos básicos de Otimização Combinatória e Complexidade Computacional. O problema de Programação de Horário Educacional é descrito, assim como suas classificações e particularidades. A seguir, são apresentadas algumas técnicas de solução para problemas combinatórios. As técnicas são organizadas em métodos exatos, heurísticos e metaheurísticos. A ênfase é dada para as técnicas Busca Dispersa e Reconexão por Caminhos, que serão utilizadas na aplicação.

No capítulo 3 são citados trabalhos encontrados na literatura que abordam o problema de Programação de Horário Educacional, assim como implementação das técnicas de solução, descritas nos capítulos anteriores, para problemas de otimização combinatória relacionados, basicamente, à construção de grades de horário educacional.

O capítulo 4 apresenta a formulação do problema Programação de Horário de Cursos Baseada em Currículos, visando a realização dos testes de desempenho da técnica de solução proposta. Para tanto, foram considerados realizados dois estudos de caso: o primeiro, refere-se ao problema de programação de cursos pós matrícula abordado na *International Timetabling Competition* e, o segundo, considera problemas encontrados no cenário de uma instituição de ensino brasileira, do tipo universidade.

O capítulo 5 detalha a implementação das técnicas Busca Dispersa e Reconexão por Caminhos.

O capítulo 6 apresenta o detalhamento e os resultados dos testes realizados. Este capítulo está organizado em quatro seções: (*i*) instâncias e ambiente de testes; (*ii*) validação; (*iii*) ajuste e sintonia de parâmetros e, por fim, (*iv*) testes de comparação.

O capítulo 7 apresenta as conclusões e propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Segundo Cooper [1], o Problema de Programação de Horário Educacional pode ser modelado como um problema matemático de Otimização Combinatória de complexidade NP-completo, mundialmente referido como *Timetabling Problem* e de grande relevância na área de Pesquisa Operacional.

Este capítulo apresenta uma breve revisão sobre problemas de otimização combinatória e respectivas alternativas de solução, com ênfase em problemas de Programação de Horário Educacional. Além disso, tem como objetivo apresentar a fundamentação teórica para os capítulos 4 e 5 que descrevem o problema abordado e a técnica de solução utilizada na aplicação descrita por esta dissertação, respectivamente.

Inicialmente, são apresentados alguns conceitos básicos sobre Otimização Combinatória e Complexidade Computacional. O problema de horários educacional é descrito, sendo apresentadas algumas de suas classificações e particularidades, bem como algumas técnicas de solução para este problema.

As técnicas são organizadas em métodos exatos, heurísticos e metaheurísticos. Inicialmente, cada um dos métodos é genericamente descrito sendo dada, a seguir, ênfase às técnicas Busca Dispersa e Reconexão por Caminhos, que são utilizadas nesta dissertação.

2.1 Otimização Combinatória

O termo Otimização Combinatória representa um ramo da matemática e da ciência da computação que analisa problemas de otimização em conjuntos. Por otimizar, se entende encontrar um valor ótimo para um determinado problema, usualmente sob determinadas condições, denominadas restrições do problema [9].

Um problema de Otimização Combinatória, em geral, pode ser representado pela tarefa de encontrar, a partir de um conjunto de itens e uma série de regras, alguns elementos (itens) desse conjunto, tipicamente finito, formando conjuntos menores (ou subconjuntos). Este agrupamento de itens específicos, na forma de um subconjunto, é denominado uma possível solução do problema e, normalmente, possui algum custo associado. Desta forma, o objetivo central da otimização combinatória é encontrar um subconjunto cujo custo seja o mínimo possível, ou seja, que represente a solução ótima do problema [9].

A formulação matemática de um problema de otimização combinatória apresenta uma função e um conjunto de requisitos ou restrições, ambos relacionados às variáveis de decisão, que fornecem um valor único a uma solução específica. Em um problema de otimização, essa função, denominada função objetivo (FO), deve ser minimizada ou maximizada [9].

Formalmente, um problema de otimização, para o caso de minimização, pode ser

descrito pela equação (2.1):

$$\text{minimizar } f(s) \quad \text{sujeito a} \quad s \in S, \tag{2.1}$$

onde $f : \Re^n \rightarrow \Re^n$ representa a função objetivo a ser minimizada e $S \subset \Re^n$ representa a região viável ou região de viabilidade, ou seja, o conjunto finito das possíveis soluções viáveis para o problema em questão. Para resolver um problema de otimização combinatória, deve-se encontrar uma solução ótima $s^* \in S$, que atenda à condição (2.2):

$$f(s^*) \leq f(s), \quad \forall s \subset S. \tag{2.2}$$

A resposta para o problema, ou seja, o ótimo global, será o menor (ou maior) valor possível para a função, para o qual o valor atribuído às variáveis não viole nenhuma das restrições, que representam os requisitos do problema. Em alguns casos, partindo de uma solução atual, a exploração dentro de um conjunto de soluções que está próximo, pode não conduzir a resultados melhores, mesmo que esta solução não represente o ótimo global. A essa solução dá-se o nome de ótimo local [9]. A Figura 2.1 apresenta uma representação do ótimo local e ótimo global em um espaço de busca.

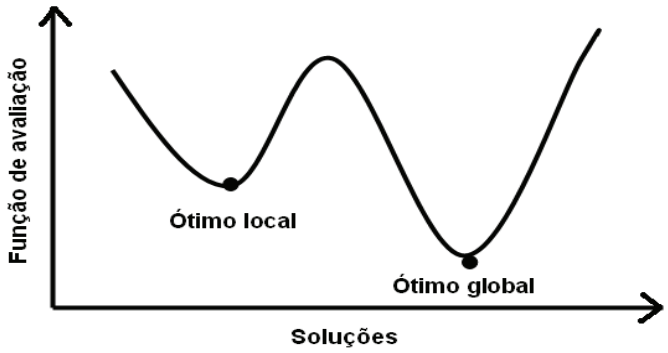

Figura 2.1: Ótimo global e ótimo local.

2.1.1 Complexidade Computacional

Uma forma prática, porém ingênua, de resolver problemas de minimização em Otimização Combinatória, seria enumerar todas as soluções possíveis e selecionar aquela de menor custo. Entretanto, tal estratégia torna-se impraticável quando se trata de problemas reais, visto que em tais aplicações o número de soluções possíveis é muito grande, mesmo para instâncias de tamanho moderado. Apesar da possibilidade de ser empregado um *hardware* com alto poder de processamento, o tempo necessário para obtenção da solução ótima seria inviável, visto que que problemas desta natureza, geralmente, tem sua solução exata obtida através do uso de algoritmos de alta complexidade [9].

A Teoria da Complexidade Computacional é o ramo da teoria da computação dedicado à análise da complexidade de algoritmos por meio de considerações matemáticas. Essa teoria, torna possível a classificação de problemas de otimização combinatória dentro de classes de complexidade, a partir do levantamento dos recursos computacionais necessários para a sua solução, geralmente avaliados em função do tamanho da entrada do algoritmo. Essa classificação é conhecida como teoria de NP-completude para os problemas de Otimização Combinatória. Desta forma, os problemas de otimização combinatória podem ser distribuídos em quatro classes [10]:

- P (*Polinomial Time*): problemas de decisão que podem ser resolvidos por algoritmos polinomiais em função do tamanho da entrada, isto é, representa o conjunto de problemas classificados como tratáveis e passíveis de solução de forma eficiente;
- NP (*NonDeterministic Polinomial Time*): problemas decisão que podem ser resolvidos por algoritmos não-determinísticos polinomiais no tamanho da entrada, ou cuja solução pode ser verificada em tempo polinomial;
- NP-completo: subconjunto de NP, é composto pelos problemas para os quais existe uma redução em tempo polinomial a partir de qualquer problema de NP;
- NP-difícil: composto pelos problemas de otimização que podem ser resolvidos através de um número polinomial de soluções de um problema NP-completo.

A Figura 2.2 mostra a relação existente entre estas classes de problemas:

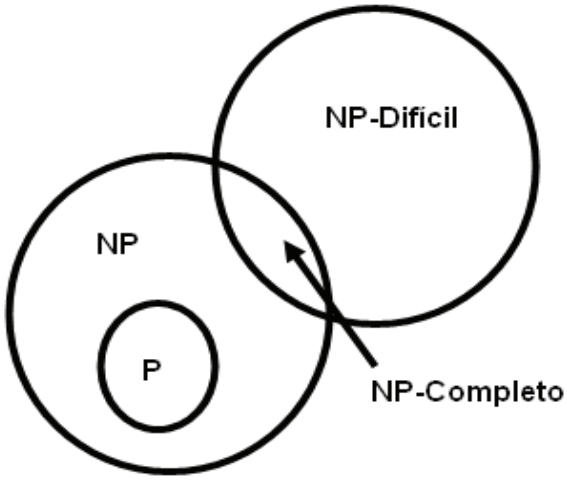

Figura 2.2: Classes de Complexidade Computacional.

Segundo Cooper e Kingston [1], o problema de *Timetabling* é classificado com um problema NP-Completo e de grande relevância para a área de Pesquisa Operacional. Sabe-se que, uma vez que se encontre um algoritmo capaz de resolver tal problema do forma eficiente, i.e. em tempo polinomial, então tal algoritmo poderia ser utilizado para resolver todos os problemas pertencentes à classe NP também de forma eficiente [9].

2.2 Horário Educacional

Uma tarefa comum em todas as instituições de ensino é a construção do quadro de horários das aulas, visando a realização das disciplinas, distribuição dos docentes e alocação de salas, atendendo às restrições, tanto de professores e alunos como de toda a instituição envolvida. Normalmente, as restrições estão relacionadas a conflitos de horários de aula, alocação de salas mais adequadas para cada disciplina/turma, preferências (da disciplina e/ou do professor) por horários, entre outros aspectos. A solução do problema consiste em gerar uma tabela de horários, visando minimizar os conflitos, maximizar preferências, compatibilizar horários de professores e alunos e utilizar de maneira eficiente equipamentos e salas disponíveis.

2.2.1 Restrições do Problema

Considerando a modelagem de um problema deste tipo, inicialmente devem ser definidos os requerimentos. Esses requerimentos devem ser expressos em termos de restrições do problema. Santos e Souza [11] dividem os requerimentos de um problema de horário educacional em três classes:

- organizacionais: relativos à instituição de ensino, nesse caso são incluídos os requerimentos que tratam da gestão de recursos, bem como do atendimento da legislação vigente. Um exemplo é a alocação de salas, visto que a realização de determinadas atividades letivas exige salas com uma capacidade específica, adicionalmente, alguns cursos requerem o uso de laboratórios ou ginásios;

- pedagógicos: são importantes para o bom aproveitamento das aulas. Exemplos são a distribuição semanal uniforme das atividades ao invés da concentração de todas as aulas em um único dia, além da existência de alguns dias de intervalo entre as aulas, o que facilita o seu acompanhamento por parte dos alunos;

- pessoais: definidos de acordo com as preferências e necessidades pessoais dos membros do corpo docente. Por exemplo, a preferência do professor por um determinado turno ou disciplina.

2.2.2 Viabilidade x Otimalidade

São várias as formulações encontradas na literatura para o problema de construção da grade de horários, conhecido como Horário Educacional, como pode ser verificado, através dos trabalhos de Werra [12], Schaerf [13] e Junginger [14]. No entanto, tal problema depende fortemente do tipo de instituição de ensino e do sistema educacional no qual a instituição está inserida e, por isso, pode se apresentar de diversas maneiras.

Em alguns casos, o problema de horários consiste em encontrar uma grade de horários que satisfaça todas as restrições. Porém, o atendimento a todas as restrições significa atender aos interesses organizacionais, pedagógicos e pessoais, como disponibilidades de professores, alunos, salas e laboratórios, recursos audiovisuais, dentre outros, o que torna a solução do problema uma tarefa árdua e com grande exigência de tempo [13].

Em outros casos, quando nenhuma solução pode ser encontrada em tempo aceitável, o problema pode ser relaxado, isto é, algumas restrições podem ter sua relevância desconsiderada. A nova configuração do problema pode então ser utilizada para encontrar a solução ótima. Nestes casos, as restrições são agrupadas em dois subconjuntos: fortes, ou restrições cujo cumprimento pela solução encontrada é obrigatório, e fracas, ou restrições cujo cumprimento pela solução encontrada é desejável [13].

Santos e Souza [11] apresentam definições para restrições fortes e fracas e diferenciam o conceito de viabilidade da solução daquele de otimalidade da solução:

- **Restrições fortes** são restrições que devem ser satisfeitas obrigatoriamente, visto que não é possível a implementação de um quadro de horários que não as satisfaça. As restrições fortes determinam o espaço de busca que será considerado, de modo que somente soluções que respeitam todas as restrições desse tipo são consideradas viáveis. As restrições mais comuns desse tipo são a não ocorrência de conflitos físicos, como por exemplo uma turma assistir duas aulas ao mesmo tempo ou a quantidade de alunos de uma turma superar a capacidade da sala;

- **Restrições fracas** são restrições cujo atendimento é desejável, mas caso não seja possível respeitá-las, é possível, ainda assim, implementar o quadro viável de horários. Uma função objetivo, definida pela soma das violações de restrições deste tipo, representará uma medida de qualidade de um quadro de horários, ou seja, o índice de otimalidade da solução. Restrições fracas comuns são, por exemplo, preferências de professores por um turno específico de trabalho ou um conjunto específico de disciplinas.

O conjunto de restrições, bem como a sua divisão em restrições fortes e fracas é bastante dependente da instituição de ensino e do sistema educacional de cada localidade.

2.2.3 Classificações do Problema

Podem ser encontradas na literatura diferentes classificações para o problema de Horário Educacional. Por exemplo, Werra [12] apresenta em seu trabalho alguns tipos básicos, sendo o modelo Turma $\times$ Professor uma denominação para o problema clássico de Horário Educacional. O autor apresenta uma formulação matemática para este problema, configurado neste caso como um problema de busca, com vistas à programação de aulas entre turmas e professores, de forma que professores e turmas estejam envolvidos em apenas um encontro em cada período de tempo.

Sejam t turmas; p professores; h períodos e $R_{t\times p}$ uma matriz de requerimentos não-negativa onde r_{tp} representa o número de aulas envolvendo a turma t e o professor p. Seja $\mathbf{X}_{t\times p\times h}$ a matriz que define a solução do problema (equação (2.3)), onde $x_{tph} = 1$ se o encontro da turma t com o professor p no horário h acontece e $x_{tph} = 0$, caso contrário.

O problema consiste em:

$$\text{Encontrar} \quad \mathbf{X}_{t\times p\times h} \quad (t = 1,\ldots,T;\ p = 1,\ldots,P;\ h = 1,\ldots,H); \tag{2.3}$$

sujeito a

$$\sum_{h=1}^{H} x_{tph} = r_{tp} \quad (t = 1,\ldots,T;\ p = 1,\ldots,P); \tag{2.4}$$

$$\sum_{t=1}^{T} x_{tph} \leq 1 \quad (p = 1,\ldots,P;\ h = 1,\ldots,H); \tag{2.5}$$

$$\sum_{p=1}^{P} x_{tph} \leq 1 \quad (t = 1,\ldots,T;\ h = 1,\ldots,H); \tag{2.6}$$

$$x_{tph} = 0 \quad \text{ou} \quad 1 \quad (t = 1,\ldots,T;\ p = 1,\ldots,P;\ h = 1,\ldots,H); \tag{2.7}$$

A restrição (2.4) assegura que o número de encontros entre o professor p e a classe t acontece um número específico de vezes, as restrições (2.5) e (2.6) garantem que cada professor e cada classe estejam alocados, ao máximo, em um encontro em cada período. A restrição (2.7) garante a não negatividade e a integralidade das variáveis de decisão.

Outra classificação encontrada na literatura é aquela proposta por Shaerf [13], onde os problemas de horário em instituições de ensino são classificados como Horário de Escolas (*School Timetabling*), Horário de Cursos (*Course Timetabling*) e Programação de Exames (*Examination Timetabling*). Esta classificação é adotada pela maioria dos autores. A seguir são apresentadas as três definições:

- **Programação de Horários de Cursos** representa o problema de determinação do calendário semanal de todas as disciplinas para um conjunto de cursos, minimizando sobreposições de horários de disciplinas que tenham estudantes em comum. Este é um problema comumente encontrado em universidades;

- A **Programação de Horários em Escolas** define o calendário semanal para turmas de uma escola, sendo que cada professor atenda apenas uma turma por período e vice-versa. No Brasil, o problema de Horários em Escolas corresponde ao problema de programação de horários encontrado em escolas de educação básica. Segundo Santos e Souza [11], a principal diferença do problema de Horários de Cursos em relação ao problema de Horários em Escolas é que o conceito de turmas neste primeiro é menos relevante, visto que a possibilidade dos alunos personalizarem o currículo é maior e não incomum, inclusive, alunos inscreverem-se em disciplinas em mais de um departamento;

- A **Programação de Exames** representa o problema de formulação do calendário de exames de cursos universitários, minimizando sobreposições de horários de exames que tenham estudantes em comum. Santos e Souza [11] consideram horários de exames como parte do problema Horários de Cursos, e salientam que este neste caso, a agenda é construída após a inscrição ou matrícula.

Em 2002, a *Metaheuristics Network*[1] organizou uma competição de métodos de produção automatizada de quadros de horários, a *International Timetabling Competition* (*ITC*)[2]. Nesta competição, os proponentes deveriam desenvolver programas para resolver uma versão simplificada do problema de programação de cursos em universidades. O programa que produzisse a melhor solução em um tempo estipulado seria o vencedor. Essa atividade foi um marco importante para a pesquisa na área, visto que, tradicionalmente, poucos autores realizavam experimentos computacionais comparando diferentes métodos de solução. Até então, era comum a ocorrência de publicações que comparavam os resultados computacionais com a solução produzida manualmente para o problema, caracterizando um processo impreciso. Essa situação ocorria principalmente devido à diversidade de necessidades das diferentes instituições de ensino, conforme mencionado anteriormente. No caso do modelo considerado na competição, sua qualidade não é ser genérico suficiente para modelar todos os casos existentes, mas sim incorporar algumas características que são comuns à grande maioria dos problemas de programação de cursos encontrados em universidades.

Em 2007, foi lançada uma nova edição da competição *ITC*[3], onde são consideradas três categorias de problemas: Programação de Exames (*Examination Timetabling*), Programação de Cursos Pós-Matrículas (*Post Enrolment based Course Timetabling*) e Programação de Cursos baseada em Currículos (*Curriculum based Course Timetabling*).

No caso da **Programação de Cursos Pós-Matrículas** os alunos matriculam-se nos cursos que gostariam de assistir e a programação de horários deve então ser construída de modo que os estudantes consigam assistir aos cursos que escolheram.

Na **Programação de Cursos Baseada em Currículos** deve-se realizar o agendamento em um conjunto de períodos semanal de vários cursos considerando também a alocação de um número limitado de salas e os conflitos entre cursos são determinados de

[1] `http://www.metaheuristics.net/`
[2] `http://www.idsia.ch/Files/ttcomp2002/`
[3] `http://www.cs.qub.ac.uk/itc2007/`

acordo com o currículo dos cursos e não de acordo com as matrículas efetuadas. Neste caso, as turmas representam os módulos curriculares.

Outra classificação comumente encontrada na literatura, é o **Problema de Alocação de Salas** (*Classroom Assignment Problem*). Nesta classificação, o problema consiste em alocar aulas, com horários de início e término previamente conhecidos, a um número fixo de salas [13]. O Problema de Alocação de Salas pode ser considerado como a alocação de aulas previamente programadas a salas, ou seja, resolve-se o problema de Programação de Cursos (pré-matrícula ou pós-matrícula) desconsiderando-se a alocação de salas e a partir desta solução resolve-se o problema de Alocação de Salas [11].

Esta dissertação aborda o problema de programação de horários de cursos baseada em currículos, e será detalhado no capítulo 4.

2.3 Métodos de Solução

Os métodos de solução para os problemas de otimização combinatória podem ser divididos em dois grandes grupos: (*i*) métodos exatos ou completos; e (*ii*) métodos aproximados.

Os métodos pertencentes ao primeiro grupo garantem a obtenção de uma solução para toda a instância de tamanho finito de um problema de otimização combinatória em um tempo limitado. Para problemas combinatórios classificados como NP-difícil, não são conhecidos algoritmos de solução em tempo polinomial, de modo que o uso de um método exato pode exigir um tempo de computação exponencial, tornando inviável a obtenção de sua solução em um tempo computacional aceitável. Desta forma, é indicado o uso de métodos aproximados, que por um lado comprometem a qualidade da solução encontrada, porém são capazes de fornecer uma boa solução em uma quantidade de tempo razoavelmente pequena [15].

Na prática, muitas aplicações não exigem uma solução exata, tornando aceitável o uso de métodos de solução de problemas que encontrem soluções aproximadas como heurísticas e metaheurísticas.

Nos últimos anos, tem-se constatado avanços consideráveis, tanto no campo teórico como nas aplicações das metaheurísticas, na tentativa de se determinar soluções aproximadas de problemas complexos de otimização combinatória. As metaheurísticas consistem em um processo iterativo que conduzem e modificam as operações de uma heurística subordinada de tal forma a produzir, eficientemente, soluções de alta qualidade. São caracterizadas também por apresentarem estratégias de alto nível que exploram o espaço de busca através da utilização de diferentes métodos [15].

2.3.1 Heurísticas x Metaheurísticas

O termo heurística provém do grego *heuriskein*, do mesmo radical que deu origem à palavra heureca, imortalizada pelo matemático e filósofo grego Arquimedes, que significa descobrir. Uma heurística é um procedimento algorítmico desenvolvido através de um modelo cognitivo, usualmente através de regras baseadas na experiência dos desenvolvedores [3].

Ao contrário dos métodos exatos, que buscam encontrar uma forma algorítmica de obter uma solução ótima através da combinação ou busca de todas as soluções possíveis, as heurísticas normalmente tendem a apresentar certo grau de conhecimento acerca do comportamento do problema, avaliando um número muito menor de soluções. Os métodos heurísticos englobam estratégias, procedimentos e métodos aproximativos, com

o objetivo de encontrar uma boa solução, mesmo que não seja a ótima, em um tempo computacional razoável. Como as heurísticas são algoritmos específicos para um caso, é recomendável que se tenha um conhecimento específico do problema que está sendo abordado [3].

As metaheurísticas, que também pertencem à classe de métodos aproximados, têm grande aplicação em problemas de otimização combinatória, para os quais as heurísticas clássicas não se mostram efetivas nem eficientes. São procedimentos computacionais e matemáticos que foram criados inspirados em outras ciências, como a física e a biologia. Funcionam como uma estratégia mestre que guia e modifica outras heurísticas, com o objetivo de produzir soluções diferentes daquelas que são normalmente geradas por buscas locais. Podem ser aplicadas a um amplo conjunto de problemas, bastando pequenas modificações para adaptá-las a um problema específico.

Técnicas metaheurísticas podem iniciar a busca da solução ótima a partir de uma única solução ou um conjunto de soluções iniciais. Esta solução, ou conjunto de soluções, inicial pode ser construída, através de alguma heurística ou de forma aleatória. Os métodos metaheurísticos irão modificar esta solução inicial através de operações realizadas com suas variáveis, gerando novas soluções, até que um determinado critério de parada seja atingido [3].

2.3.2 Intensificação x Diversificação

Com o objetivo de obter melhores resultados, as metaheurísticas precisam ser adaptadas ao problema ao qual estão sendo aplicadas, seja com relação à função objetivo empregada, seja com relação à melhoria do processo de busca.

Esta adequação pode ser implementada através de estratégias de diversificação e intensificação, a serem utilizadas no processo de busca. Neste âmbito, o termo investigar se refere à capacidade da metaheurística de "saltar" de uma região para outra no espaço de busca. Já o termo exploração reflete a capacidade em explorar, de forma mais intensa, uma mesma região dentro do espaço de busca. Em resumo, o que existe são dois processos distintos de busca, um externo (investigação) e outro interno (exploração). A investigação é comumente chamada de diversificação, enquanto a exploração é denominada de intensificação [15].

O termo vizinhança, se refere aos novos espaços de busca das soluções, que podem ser alcançados por meio de um movimento. Por movimento em um espaço de busca, entende-se a aplicação de uma regra ou função que altere a solução atual, gerando uma nova solução. Movimentos são representados, na prática, por operações como troca, remoção ou inserção de elementos na solução corrente. As heurísticas de melhoria possuem um mecanismo de parada, normalmente, quando nenhum outro movimento melhora o resultado atual, o qual é considerado um ótimo local [15].

Entenda-se por vizinhança, o conjunto de soluções que podem resultar da solução anterior, por alterações realizadas nesta, resultado da aplicação de um movimento. Isto é, uma solução s' diz-se fazer parte da vizinhança da solução s se e só se resultou de uma modificação nesta, de tal forma que continue a fazer parte do conjunto de soluções possíveis. Denomina-se movimento a modificação m que transforma s em s', fazendo com que esta última esteja em sua vizinhança [16].

2.3.3 Vizinhanças e Movimentos

Uma vizinhança da solução s pode ser descrita em termos de mudanças (ou movimentos) aplicadas a s, sendo representada por $N(s)$.

Neste trabalho são aplicados os conceitos de vizinhança composta apresentados por Gaspero e Schaerf [17], onde define-se uma vizinhança composta por soluções geradas por mais de um tipo de movimento. Nesta abordagem, uma vizinhança é formada por soluções obtidas a partir da aplicação de n tipos de movimentos aplicados em seqüência, podendo ser movimentos do mesmo tipo ou não. Tal estratégia de combinação, de dois ou mais movimentos, normalmente é utilizada a fim de aumentar a capacidade de busca de uma estrutura de vizinhança, especialmente quando as vizinhanças combinadas possuirem características complementares [17; 18].

Um movimento é tipicamente composto por um conjunto de atributos ou variáveis que descrevem a mudança de estado. Dado um estado s e um movimento m, a notação $s \oplus m$ representa o estado obtido pela aplicação do movimento m sobre a solução s.

Um dos parâmentros que define um movimento é o tipo do movimento. Dada uma solução s, para atingir uma solução s', onde s' é dito vizinho de s, são usados três diferentes tipos de movimentos básicos:

1. Inserção: na programação de uma aula, o movimento de inserção consiste em substituir um dos recursos que esteja alocado, por outro de mesmo tipo, porém que não se encontre já alocado no quadro de horários. Uma sala é um exemplo de recurso passível de inserção;

2. Realocação: o movimento de realocação consiste em realocar a programação de uma aula, trocando o seu horário original por outro onde as turmas correspondentes não se encontram alocadas;

3. Troca: o movimento de troca consiste na troca dos horários e/ou de recursos do mesmo tipo, alocados na programação de duas aulas.

2.3.4 Heurísticas Construtivas x Heurísticas de Melhoramento

Conforme Cordenonsi [3], os algoritmos heurísticos baseados em construção, ou simplesmente heurísticas construtivas, partem de uma solução vazia, acrescentando elemento a elemento, seguindo um certo critério heurístico, até formar a solução por completo. Algoritmos construtivos não possuem nenhum esquema de *backtracking*, ou seja, após inserir um resultado, não é possível retirá-lo.

As heurísticas construtivas podem fazer uso de dois métodos: (*i*) o método aleatório, sendo considerado como a heurística de construção mais simples; e (*ii*) os métodos de construção gulosa, bastante utilizados e baseados no incremento da solução a cada passo, onde o elemento a incluir é determinado por uma função heurística.

As heurísticas gulosas, em geral, criam soluções de qualidade muito superior à média das soluções aleatórias. No entanto, possuem como desvantagem o fato de que uma decisão tomada na inserção de um elemento não pode ser alterada. Caso ocorra uma decisão equivocada no início do processo, ela não poderá ser corrigida. Em se tratando das soluções finais geradas por esse tipo de heurística, a sua diversidade é muito pequena, ainda que sejam utilizadas regras para conduzir a procura. Essa desvantagem pode ser minimizada com a introdução de um componente aleatório na construção [3].

As heurísticas de melhoramento, também chamadas de heurísticas de melhoria, heurísticas de refinamento ou heurísticas de busca local, iniciam com uma solução inicial e trabalham no melhoramento da solução atual, por meio da realização de passos sucessivos. Estes passos realizam a exclusão e inclusão de novos resultados, de forma a pesquisar a vizinhança da solução em busca de uma melhor qualidade.

Heurísticas de melhoramento exploram o conceito de ótimo local. Algumas vezes, o ótimo local pode ser o ótimo global, ou seja, a melhor solução possível para o problema, mas não há garantias em relação a este fato, como em todos os procedimentos heurísticos. Assim, pode acontecer do algoritmo parar e exibir como resposta um ótimo local, ignorando as possibilidades que poderiam, mais tarde, levar o algoritmo a encontrar um ótimo global [3].

2.3.5 Busca em Trajetória x Busca populacional

Metaheurísticas são heurísticas de busca no espaço de soluções e podem ser divididas em duas classes: (*i*) busca em trajetória; e (*ii*) busca populacional.

A primeira classe compreende os métodos que exploram uma vizinhança a cada iteração ou ciclo, alterando tanto a vizinhança quanto a forma de explorá-la de acordo com suas estratégias, sendo escolhido apenas um elemento dessa vizinhança a cada iteração. Esse tipo de varredura do espaço de soluções gera um caminho ou trajetória de soluções, obtida pela transição de uma solução para outra de acordo com os movimentos permitidos pela metaheurística [15].

Em contraposição aos métodos de busca em trajetória, existem aqueles que exploram uma população de soluções a cada iteração. Esses métodos constituem a segunda classe de metaheurísticas e suas estratégias de busca são capazes de explorar várias regiões do espaço de soluções ao mesmo tempo. Dessa forma, ao longo das iterações não se contrói uma trajetória única de busca, pois novas soluções são obtidas através da combinação de soluções anteriores [15].

Dentro da classe de métodos que implementam a busca em trajetória, podem ser citadas as metaheurísticas Busca Tabu e a Têmpera Simulada:

- **Busca Tabu ou *Tabu Search*** [6] é uma metaheurística que possui a habilidade de guiar o processo de busca através de métodos iterativos de melhoria. Neste contexto, a Busca Tabu fornece meios de explorar o espaço de soluções além dos pontos em que, a heurística que ela está guiando, encontra ótimos locais. O processo no qual a metaheurística busca transcender a otimalidade local se baseia em uma função de avaliação, que escolhe, a cada iteração, o movimento com maior valor de avaliação na vizinhança da solução corrente, em termos de valor da função objetivo, e de restrições de movimentos denominados tabu.

- **Têmpera Simulada ou *Simulated Annealing*** [7] é um algoritmo metaheurístico que busca soluções ótimas globais considerando a forte analogia entre o processo físico de resfriamento lento e controlado de sólidos e a resolução de problemas de otimização combinatória. Em altas temperaturas, o algoritmo aceita uma pior solução, com uma certa probabilidade, como meio de escapar de soluções ótimas locais. Quando a temperatura diminui, a probabilidade de aceitação de soluções piores diminui consideravelmente. Em cada nível de temperatura são aplicadas buscas locais.

Na classe de métodos de busca populacional podem ser citados os Algoritmos Genéticos e a Busca Dispersa.

- **Algoritmos Genéticos ou *Genetic Algorithms*** [8] são algoritmos estocásticos que a cada iteração geram uma nova população, a partir da anterior, através de três operadores genéticos básicos: seleção, combinação e mutação. Inspirado na metáfora

dos mecanismos evolucionários encontrados na natureza, o processo evolutivo se dá pela criação de uma nova população de indivíduos através de um processo de recombinação de dados, gerando novas soluções viáveis. A população é atualizada continuamente pela geração de novos membros da população a partir dos indivíduos existentes, sendo removidos os membros menos aptos de acordo com o valor da sua função de aptidão. Depois de algumas gerações, o melhor indivíduo da população será, potencialmente, a solução ótima, ou próxima da ótima, para o problema analisado.

- **Busca Dispersa ou *Scatter Search*** [6] é um método metaheurístico que realiza uma exploração sistemática sobre uma conjunto de boas soluções e soluções diversas chamado conjunto de referência. Estas soluções são combinadas de modo a criar novas soluções, e assim melhorar o conjunto de soluções original. É, portanto, um método evolutivo. Mesmo assim, a Busca Dispersa difere bastante do método Algoritmos Genéticos, pois usa estratégias determinísticas de exploração de subconjuntos, gerados a partir do conjunto de soluções referência, e usa memória adaptativa para guiar o uso desta estratégia ao longo do processo de busca.

Neste trabalho a metaheurística Busca Dispersa ou *Scatter Search* , combinada com a heurística *Path Relinking*, é aplicada para solução do problema de construção do quadro de horários educacional.

2.3.6 Busca Dispersa

A Busca Dispersa ou *Scatter Search* (SS), é um método que foi formalmente introduzido por Glover [6] em um estudo heurístico para a resolução de problemas de Programação Linear Inteira.

Em comum com outros métodos evolucionários, a Busca Dispersa opera sobre uma população de soluções, ao invés de uma única solução em cada iteração, e emprega procedimentos para combinar essas soluções com o intuito de gerar novas soluções. Por outro lado, diferentemente desses mesmos métodos, viola a premissa de que as abordagens evolucionárias devam ser baseadas em escolhas aleatórias, limitando o uso da randomização aos procedimentos de diversificação utilizados [4].

Assim, ao invés de explorar soluções de forma aleatória, como nos Algoritmos Genéticos, a Busca Dispersa explora extensivamente regiões pré-determinadas em cada iteração. A Busca Dispersa opera sobre um Conjunto de Soluções Referência ($RefSet$) para gerar novas soluções a partir de combinações lineares ponderadas das soluções pertencentes a subconjuntos estruturados do $RefSet$ [6].

O $RefSet$ é composto pelas melhores soluções encontradas. O conceito de melhores neste caso não considera apenas a característica de estar entre os melhores valores da função objetivo (FO), mas leva também em consideração a diversidade da solução em relação às outras pertencentes ao $RefSet$ [4].

Essa diversidade, determinada pela criação de regras de decisão aplicadas à estratégia de combinação, é muito importante para evitar que o algoritmo, ao longo das combinações do $Refset$, encontre em um ótimo local e não consiga mais sair dele. No momento em que o algoritmo encontra uma região deste tipo, deve haver pelo menos uma solução diversificada que garanta a geração de novas soluções diversificadas.

De acordo com Glover [4], o algoritmo consiste basicamente de cinco elementos, descritos a seguir:

1. **Método de geração de soluções diversas**: o método baseia-se em gerar um conjunto P de soluções diversas, com $PSize$ elementos. O objetivo é gerar soluções distantes entre si de forma a garantir a diversidade.

2. **Método de melhoria**: trata-se de um método de busca local utilizado para melhorar as soluções diversas, criadas no passo anterior. No caso de soluções infactíveis, a melhoria primeiramente consiste em torná-las factíveis. Caso a solução já seja factível, aplica-se a heurística de melhoria.

3. **Método de atualização do conjunto de referência**: é um método utilizado para criar e atualizar o conjunto de referência. A partir do conjunto de soluções diversas P, extrai-se o conjunto de referência. É dividido em um subconjunto de soluções de alta qualidade e um subconjunto de soluções diversas (intensificação $\times$ diversificação).

   (a) **Criação do conjunto de referência**: o conjunto de referência ($RefSet$), de tamanho $b = b_1 + b_2$, é formado inicialmente pelas b_1 melhores soluções de P, obtidas após o método de melhoria. As b_2 soluções restantes são extraídas de P com o objetivo de maximizar a distância mínima em relação às soluções já presentes no conjunto de referência.

   (b) **Atualização do conjunto de referência**: as soluções resultantes do processo de combinação, no caso de melhoria, substituem as soluções do conjunto de referência. Os tipos de atualizações podem ser: a) estática ou dinâmica, b) por qualidade ou diversidade. O tamanho do conjunto de referência (b), das soluções de alta qualidade (b_1) e das soluções diversas (b_2) permanecem sempre constantes, sendo observado, ao longo da busca, uma melhora do valor médio da qualidade das soluções.

4. **Método de geração de subconjuntos**: este método especifica a forma como são selecionadas as soluções para construção de subconjuntos, para posterior aplicação do método de combinação.

5. **Método de combinação das soluções**: tem por objetivo combinar as soluções do conjunto de referência. Para isso, são considerados os subconjuntos formados no passo 4, e aplica-se o método de combinação. As soluções obtidas desta combinação podem ser imediatamente introduzidas no conjunto de referência (atualização dinâmica) ou armazenadas temporariamente em uma lista até que sejam realizadas todas as combinações, e a partir daí verifica-se quais soluções entrarão no conjunto (atualização estática). O método de melhoria é aplicado às melhores soluções geradas pelas combinações.

A Figura 2.3 apresenta o pseudocódigo do algoritmo Busca Dispersa.

Passo 1. Inicie com $P \leftarrow \emptyset$. Inicie a etapa geradora de soluções iniciais e construa uma solução y. Aplique o método de melhoria a y e encontre a solução melhorada x. Se $x \notin P$ então adicione x a P, caso contrário descarte x. Repita este passo até $|P| = PSize$.
Passo 2. Construa o conjunto de referência $RefSet = x^1, \ldots, x^b$ com as b_1 melhores soluções de P e as b_2 soluções de P com maior diversidade em relação as soluções que estão em $RefSet$. Ordene essas $b = b_1 + b_2$ soluções de forma crescente em relação a função objetivo. Faça o operador lógico $newsolutions \leftarrow TRUE$.
Enquanto ($newsolutions = TRUE$.)
    $newsolutions \leftarrow FALSE$.
    Passo 3. Mediante a geração de subconjuntos gere subconjuntos a partir de $RefSet$, agrupando, por exemplo, pares de soluções deste conjunto.
    Enquanto (existem conjuntos sem examinar)
        Passo 4. Aplique o método de combinação às soluções de cada subconjunto gerado no **Passo 3** e guarde as soluções obtidas em uma lista denominada *Pool*.
        Passo 5. Aplique o método de melhoria a cada solução de *Pool* obtida no **Passo 4**.
    Fim Enquanto
    Passo 6. Caso possível atualize $RefSet$ (b melhores soluções em $RefSet \cup Pool$).
    Reordene $RefSet$.
    Se $RefSet$ atualizado, $newsolutions \leftarrow TRUE$.
Fim Enquanto.

Figura 2.3: Estrutura do algoritmo Busca Dispersa canônico.

Implementações mais recentes da metaheurística Busca Dispersa, ou *Scatter Search*, foram propostas por Glover e colaboradores [4], onde é examinado o seu uso cojunto com a heurística Reconexão por Caminhos (*Path Relinking*) na solução de diferentes problemas de otimização combinatória.

2.3.7 Reconexão por Caminhos

A técnica Reconexão por Caminhos, Religação de Caminhos ou *Path Relinking* (PR), foi proposta por Glover [5] como uma estratégia de intensificação, para explorar trajetórias que conectavam soluções elite obtidas pelos métodos Busca Tabu ou Busca Dispersa.

Esta busca por soluções de melhor qualidade consiste em gerar e explorar caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite. Para tal finalidade, são selecionados movimentos que introduzem atributos das soluções guia na solução corrente. Desse modo, a Reconexão por Caminhos pode ser vista como uma estratégia que tem por objetivo incorporar atributos de soluções de boa qualidade, favorecendo a seleção de movimentos que as contenham.

O método de Reconexão por Caminhos inicia computando a diferença de simetria entre duas soluções s_I e s_G, resultando em um conjunto de movimentos que deve ser

aplicado a uma delas, dita solução inicial, para transformar-se na outra, dita solução guia. A partir da solução inicial, o melhor movimento ainda não executado é aplicado à solução corrente, até que a solução guia seja atingida. A melhor solução encontrada ao longo dessa trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução já encontrada é atualizada. Considere o caminho que liga as soluções s_I e s_G é representado pela seqüência de soluções

$$s_I = s^1, s^2, \ldots, s^r = s_G;$$

tal que a solução $s^{k+1} \in N_{PR}(s^k)$ é obtida pela aplicação de um movimento, onde N_{PR} é a vizinhança considerada conforme mostra a Figura 2.4.

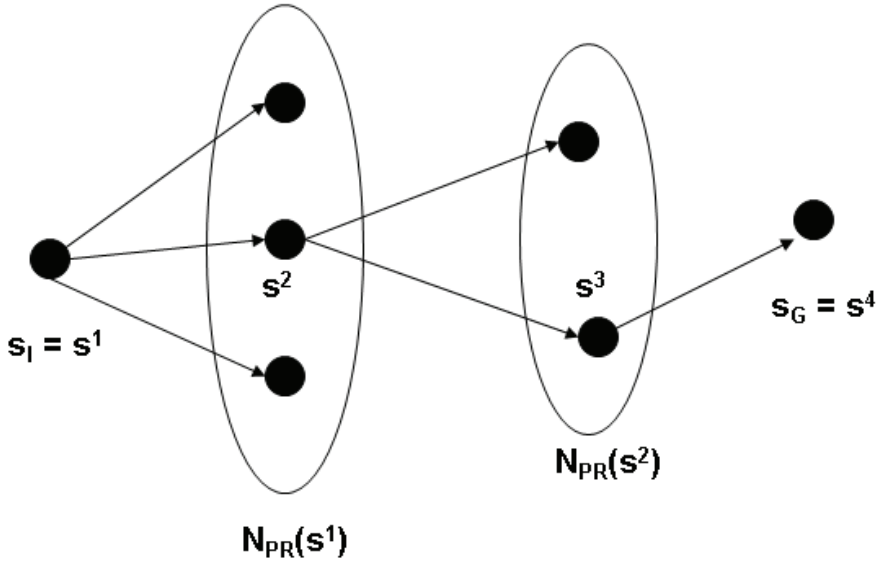

Figura 2.4: Esquema de funcionamento da heurística *Path Relinking*.

Segundo Coelho [19], a heurística Reconexão por Caminhos pode ser aplicada como uma estratégia de pós-otimização entre todos os pares de soluções elite. No modelo implementado nesta dissertação, a Reconexão por Caminhos é aplicada como heurística de combinação das soluções geradas, no método de combinação das soluções.

A Figura 2.5 apresenta o pseudocódigo do algoritmo Reconexão por Caminhos.

Procedimento Reconexão por Caminhos (s_I **;** s_G);
    $s' \leftarrow s_I$;
    Atribuir a s^* a melhor solução entre s_I e s_G;
    Calcular o conjunto de movimentos possíveis entre $\Delta(s_I; s_G)$;
    Enquanto $|\Delta(s_I; s_G)| \neq 0$ **Faça**;
        Atribuir a s'' à melhor solução obtida aplicando o melhor movimento de $\Delta(s_I; s_G)$ a s';
        Excluir de $\Delta(s_I; s_G)$ o movimento aplicado;
        $s' \leftarrow s''$;
        Se $f(s') < f(s^*)$ **Então**;
            $s^* \leftarrow s'$;
        Fim Se;
    Fim Enquanto;
    retorne s^*;
Fim Procedimento;

Figura 2.5: Estrutura do algoritmo Reconexão por Caminhos canônico.

No capítulo 5, é descrita de forma mais detalhada a implementação da metaheurística Busca Dispersa e da heurística Reconexão por Caminhos, quando aplicadas ao problema de construção do quadro de horários, abordado nesta dissertação.

3 TRABALHOS RELACIONADOS

Problemas de Horários em instituições de ensino têm sido bastante estudados nas últimas décadas, inúmeros trabalhos sobre o assunto foram publicados e conferências regulares discutem o tema no meio científico. Um evento de grande relevância na área é a Conferência Internacional de Automação de Problemas de Horário (*Practice and Theory on Automated Timetabling* - PATAT [1]), que teve seu início em 1995 e ocorre a cada dois anos. Da mesma forma, a *International Timetabling Competition* - ITC é organizada, anualmente, pela *Metaheuristics Network* [2]. Nesta competição, os proponentes desenvolvem programas para resolver uma versão simplificada do problema de programação de cursos em universidades.

Neste capítulo são citados trabalhos encontrados na literatura que abordam o problema de programação de horários em instituições de ensino, assim como exemplos de implementação das técnicas de solução descritas nos capítulos anteriores para problemas de implementação de quadro de horários.

Importante frisar que, até o momento, não foram encontrados na literatura trabalhos descrevendo o emprego da metaheurística Busca Dispersa (*Scatter Search*) na solução específica do problema de horário educacional proposto (Programação de Curso Baseada em Currículos). Desta forma, serão referenciados apenas trabalhos que aplicam esta técnica na solução de problemas similares.

3.1 Formulações do Problema

A atenção da comunidade científica foi despertada mais fortemente para o Problema de Horário Educacional a partir do trabalho de Gotlieb [20], que considerou um problema que envolvia o arranjo do número exigido de encontros entre cada classe e cada professor, de tal modo que nenhum professor pudesse lecionar para mais de uma classe no mesmo horário e que nenhuma classe pudesse ter aula com mais de um professor ao mesmo tempo.

São várias as formulações matemáticas que podem ser encontradas na literatura, como pode ser verificado pela grande quantidade de trabalhos publicados sobre o tema como Junginger [14], de Werra [12], Schaerf [13] e Santos e Souza [21].

Junginger [14] descreve a pesquisa na Alemanha sobre o problema de horário em escolas. Em particular, ele descreve os diversos softwares implementados e sua utilização efetiva pelas instituições de ensino. O artigo, também, descreve as abordagens de solução utilizadas, na maioria dos casos, baseadas em heurísticas diretas. No trabalho de Werra

[1] `http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml`
[2] `http://www.metaheuristics.net/`

[12] são apresentados modelos e formulações matemáticas para os problemas clássicos de horário, com ênfase na teoria de grafos.

A pesquisa de Schaerf [13] é um apanhado de trabalhos anteriores organizados por tipo de problema abordado: Horário de Escolas (*School Timetabling*), Horário de Cursos (*Course Timetabling*) e Programação de Exames (*Examination Timetabling*). O autor referencia principalmente trabalhos que utilizam as técnicas *Simulated Annealing*, Busca Tabu, Algoritmos Genéticos e Técnica de Satisfação de Restrições.

Santos e Souza [21] apresentam os principais tipos de Problemas de Programação de Horários encontrados em instituições de ensino. Os autores afirmam que, apesar de cada um desses problemas apresentar características únicas, os mesmos compartilham uma estrutura bastante similar. Desse modo, a experiência adquirida na resolução de cada um, é, geralmente, facilmente generalizável para a resolução de outros. No trabalho são revisadas algumas das principais técnicas de solução como *Simulated Annealing*, Busca Tabu e Algoritmos Genéticos, incluido exemplos de suas aplicações com ênfase no Problema de Programação de Horários em Escolas.

3.2 *Simulated Annealing*

Bai e colaboradores [22] introduzem o conceito de hiper-heurística, sendo definida como *uma heurística para escolher heurísticas*. Nesta abordagem, a metaheurística *Simulated Annealing* gerencia um conjunto de funções de vizinhança e heurísticas para resolver um problema de cursos em universidades.

Nandhini e Kanmani [23] exploram trabalhos publicados nos últimos 10 anos, abordando o uso da técnica *Simulated Annealing* para problemas de programação de horários de cursos em universidades, como referencial para implementação desta técnica na sua instituição.

3.3 Busca Tabu

Subramanian e colaboradores [24] utilizam Busca Tabu para resolver o Problema de Alocação de Salas do Centro de Tecnologia da Universidade Federal da Paraíba. O trabalho procurou identificar as particularidades do problema, para então determinar as restrições de viabilidade e os requisitos de qualidade. Desenvolveu-se ainda um procedimento heurístico para gerar uma solução inicial de boa qualidade. Compararam-se as soluções inicial, final e aquela gerada manualmente e executada no semestre avaliado. O procedimento Busca Tabu demonstrou ser bastante eficiente e robusto, tendo gerado soluções de alta qualidade.

Sousa e colaboradores [25] apresentam um procedimento de Busca Tabu associado a uma Busca Local Aleatória e duas formulações matemáticas para o problema de programação da grade de horários em escolas de ensino fundamental e médio. O procedimento proposto foi experimentado com sucesso em escolas públicas brasileiras. O procedimento de Busca Tabu é composto de uma lista tabu sistematicamente dinâmica e utiliza movimentos simples. O procedimento de Busca Local Aleatória, baseada em movimentos reparadores, foi utilizado para acelerar o processo de busca junto à Busca Tabu. O algoritmo mostrou resultados satisfatórios, tanto com relação ao tempo computacional como com relação à qualidade das soluções obtidas.

3.4 Algoritmos Genéticos

Góes [26] desenvolveu um protótipo utilizando três algoritmos (exato, heurístico e misto) aplicado na construção do horário escolar da Escola Municipal Planalto dos Pinheiros, no município de Araucária-PR. O método exato utilizado é a elaboração da Modelagem Matemática do Problema, abordado como um problema da Programação Linear Inteira Binária, resolvido pelo software LINGO 6.0. O método heurístico utilizado é uma estratégia baseada em Algoritmos Genéticos. O trabalho apresenta uma comparação entre os horários gerados pelo protótipo, o horário gerado pelo software comercial e o horário gerado manualmente na instituição de ensino. Os autores concluem que horários gerados pelos métodos Exato, Heurístico e Misto obtiveram melhor resultado que o gerado manualmente e aquele gerado pelo software comercial, conforme as especificidades da Rede Municipal de Araucária.

Massoodian e Esteki [27] descrevem um Algoritmo Genético baseado em uma abordagem de duas fases para resolver o problema de Programação de Cursos baseada em Currículos. Nesta abordagem, um algoritmo de busca local é aplicado em cada etapa: a primeira fase elimina as violações de restrições fortes, e a segunda fase minimiza as violações às restrições fracas, mantendo o número de violações das restrições fortes em zero.

Lobo [28] apresenta o uso de uma solução multiplataforma, usando a linguagem Java e a arquitetura Corba, para implementar uma heurística de otimização de quadro de horários em ambiente distribuído. É apresentada uma utilização da metodologia de Algoritmos Genéticos para desenvolver uma solução de um quadro de horários do grupo turma–professor. O trabalho utiliza princípios de programação paralela, inerentes ao problema de interesse, e é executado em um cluster de computadores presentes em uma rede local. O objetivo é a diminuição do tempo computacional utilizado na determinação de uma solução viável. O método desenvolvido é aplicado ao estudo do caso do campus Gigante, da Universidade Presidente Antônio Carlos (UNIPAC), situado em Conselheiro Lafaiete, MG. O problema é modelado e as especificidades dessa instituição de ensino, quanto à construção de um quadro de horários, são apresentadas e discutidas.

3.5 Técnicas Híbridas

Souza e colaboradores [29] propõem uma técnica híbrida, envolvendo as metaheurísticas Programação Genética, Busca Tabu e GRASP [30], para solução do problema de programação de horários em escolas. No algoritmo evolutivo utilizado, a população inicial é gerada pelo procedimento construtivo da técnica GRASP. A cada geração deste algoritmo, o melhor indivíduo da população sofre um refinamento através do procedimento de Busca Tabu. O problema considerado para análise é o da programação de horários da Escola Estadual Dom Velloso em Ouro Preto, MG. Esta escola atende o 2º ciclo do ensino fundamental (5º a 8º ano), atendendo 11 turmas no turno matutino e 3 no vespertino. Nos testes preliminares realizados, o procedimento híbrido conseguiu produzir soluções finais de melhor qualidade do que aquele sem a fase de refinamento por Busca Tabu. O algoritmo sem esta fase de refino tem enorme dificuldade para encontrar soluções viáveis.

Rahoul e Saad [31] aplicam um algoritmo híbrido que combina as metaheurísticas Busca Tabu e Algoritmos Genéticos para uma uma instância do problema de Programação de Cursos na *University of Science and Technology Houari Boumediene*, da Argélia. Nos testes realizados para produção de dois horários do departamento de Ciência da Com-

putação desta universidade, o tempo do processo manual, que costumava levar de três a quatro semanas, foi reduzido a apenas uma hora, em média.

Frausto-Solís e Alonso-Pecina [32] apresentam um algoritmo híbrido (*Simulated Annealing* e Busca Tabu), denominado "Fralonso", para solução do problema de Programação de Cursos baseada em Currículos. O algoritmo Fralonso é dividido em duas fases, onde na primeira uma solução viável é encontrada, e na segunda esta solução é melhorada. Em cada fase, a metaheurística Busca Tabu é empregada após o uso do *Simulated Annealing*, com o objetivo de refinar o processo de pesquisa. Além disso, o algoritmo Fralonso usa dois esquemas de vizinhança para que seja preservado um bom nível de diversidade.

3.6 Busca Dispersa

Belfiore e Yoshizaki [33] aplicam a metaheurística Busca Dispersa em um problema real de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. No problema de roteirização de veículos com entregas fracionadas, cada cliente pode ser abastecido por mais de um veículo. O problema é baseado em um único centro de distribuição, a demanda de cada cliente pode ser maior que a capacidade dos veículos e, além das restrições de janelas de tempo, há também as restrições de capacidade dos veículos e acessibilidade (alguns clientes não podem ser atendidos por alguns veículos). O modelo foi implementado em um dos maiores grupos varejistas brasileiros, que abastece 519 clientes distribuídos em 12 estados. Os resultados mostraram melhorias no caso real da empresa, reduzindo em até 8% o custo total da operação.

Belfioree e colaboradores [34] implementam a metaheurística Busca Dispersa para o problema de programação de tarefas em uma única máquina com penalidades de adiantamento e atraso e data de entrega comum. No algoritmo proposto, o método de geração de soluções diversas foi implementado com uma heurística construtiva. Para o método de melhoria foi utilizada busca local, sendo definidos dois tipos de vizinhanças: inserção e troca. Para atualização do conjunto de referência, foi utilizada atualização estática, na qual o conjunto de referência não muda até que todas as combinações de $RefSet$ tenham sido realizadas. O conjunto de referência foi atualizado quando a nova solução gerada pelo método de combinação obteve melhor qualidade que o pior elemento de $RefSet$ (atualização por qualidade).

O trabalho de Lorenzoni e colaboradores [35] apresenta uma abordagem baseada na metaheurística Busca Dispersa para a resolução do problema de alocação de sondas em poços de petróleo. O problema consiste em definir o itinerário de atendimento para cada sonda, uma vez que dedicar exclusivamente uma sonda para cada poço de uma bacia petrolífera torna-se inviável economicamente. Para geração das soluções iniciais, os autores codificam dois métodos. Um primeiro método totalmente aleatório, em que dois valores são gerados randomicamente, sendo um deles associado ao poço que será inserido na solução e o outro que indica qual sonda atenderá o poço. São alocados novos poços continuamente até que todos os poços façam parte da solução. O outro método, denominado guloso, baseia-se na perda de vazão de cada poço para tomar a decisão de qual será o próximo a ser alocado. Desta forma, os poços com maior perda de vazão têm maior probabilidade de serem alocados nas primeiras colunas da matriz solução. Para combinar as soluções, os autores implementaram dois métodos: o primeiro, chamado clássico, consiste na divisão aleatória de duas soluções a partir de uma determinada coluna e a junção das partes complementares dessa divisão com a exclusão de poços repetidos. No

segundo, denominado substituição ajustada, são gerados quatro números aleatórios que definem duas posições, uma em cada solução, que serão substituídas. A quantidade de substituições a serem feitas para gerar uma nova solução basearam-se em 10% do número de poços. Para a atualização do *RefSet*, o algoritmo proposto inicialmente adotou as 30% melhores soluções da população inicial, com o restante sendo escolhido aleatoriamente. A versão que apresentou os melhores resultados foi a que utilizou o método aleatório para a geração das soluções iniciais e o método de substituição ajustada para combinação de soluções.

Mansour e colaboradores [36] propõe um algoritmo baseado na metaheurística Busca Dispersa para o problema de horários de exames. Os resultados gerados pela aplicação da técnica foram comparados com os resultados da programação manual em um problema de horários real de uma universidade. Os autores também fizeram um comparativo destes resultados com os obtidos pela aplicação de três outras metaheurísticas: Algoritmos Genéticos, *Simulated Annealing* e *3-phase Simulated Annealing*. A conclusão foi de que a Busca Dispersa produziu soluções melhores do que aquelas produzidas manualmente ou pelas outras metaheurísticas.

3.7 Reconexão de Caminhos

Esta seção apresenta alguns trabalhos que descrevem o emprego da técnica Reconexão por Caminhos em problemas de horário escolar.

Coelho [19] apresenta um estudo sobre a utilização de metaheurísticas na resolução do Problema de Horário Escolar, também conhecido como Problema Classe-Professor. Os resultados obtidos pelos algoritmos *Simulated Annealing* e *Iterated Local Search* são comparados, verificando a capacidade de cada técnica em encontrar soluções viáveis no menor tempo de execução. Um algoritmo híbrido, baseado em Algoritmos Meméticos, *Iterated Local Search* e Reconexão por Caminhos, é apresentado como proposta para resolver o problema abordado. Os algoritmos foram testados com dados reais do Centro Federal de Educação Tecnológica de Rio Pomba. Os resultados computacionais mostram que o algoritmo híbrido obtém soluções, com sucesso, em todos os problemas-teste.

Moura e colaboradores [37] abordam o problema de programação de horários em escolas. A abordagem é testada sobre uma instância do problema, derivada da realidade brasileira, típica das escolas de ensino fundamental e médio, públicas ou particulares. O trabalho está embasado no uso de metaheurísticas evolutivas (Algoritmos Genéticos) e de busca local (Busca Tabu e GRASP), com a aplicação de técnicas alternativas específicas para obter melhores resultados sobre a instância do problema tratado (Reconexão por Caminhos). O trabalho visa também, através da criação de uma interface amigável para usuários leigos, prover uma ferramenta útil para instituições de ensino brasileiras, que atualmente dispõem de poucas ferramentas para lidar com o problema e despendem grande quantidade de tempo para a montagem manual de seus horários.

3.8 Métodos de Pesquisa em Vizinhança

Esta seção apresenta alguns trabalhos que descrevem o emprego de técnicas de Pesquisa em Vizinhanças em problemas de horário escolar.

Souza e colaboradores [38] aplicam uma técnica recente, baseada em trocas sistemáticas de vizinhança durante a pesquisa, conhecida como Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*), para resolver o problema de alocação de salas.

Uma solução inicial é construída por um procedimento que segue as idéias da fase de construção do método GRASP. Essa solução é então submetida ao método de Pesquisa em Vizinhança Variável, o qual faz uso de três estruturas diferentes de vizinhança. Duas versões da fase de busca local deste método são testadas. Na primeira versão, faz-se uma busca local usando o Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*). Na segunda, procura-se apenas o melhor vizinho na estrutura de vizinhança corrente da solução em análise.

Marcondes [39] propõe a aplicação de três algoritmos heurísticos para o problema de alocação de salas. O primeiro consiste na resolução sucessiva de problemas de designação e o segundo na resolução sucessiva de problemas de designação com gargalo, ambos com três fases cada. O terceiro algoritmo é baseado na metaheurística Busca em Vizinhança Variável. Os testes foram realizados com dados reais de uma universidade e os resultados alcançados pelos três algoritmos foram comparados entre si e com resultados utilizados pela instituição. O primeiro algoritmo, baseado no problema de designação, apresentou os melhores resultados em relação à qualidade da solução e eficiência.

4 FORMULAÇÃO DO PROBLEMA

Este capítulo apresenta a formulação matemática proposta para o problema de Programação de Horário de Cursos Baseada em Currículos.

Conforme comentado na Seção 2.2.2, são várias as formulações encontradas na literatura para o problema de construção da grade de horários, uma vez que esta varia de acordo com o tipo de instituição de ensino e do sistema educacional no qual a instituição está inserida. Assim, a formulação matemática aqui apresentada está baseada na combinação de algumas propostas encontradas na literatura [40], [41], [42], [43], assim como na inserção de novas equações de autoria deste autor.

4.1 Definição do Problema

O problema de Programação de Horário de Cursos Baseada em Currículos consiste em programar as aulas previstas para um período letivo, realizando o agendamento em um conjunto de períodos semanal de vários cursos e alocando um número limitado de professores e salas. A programação das aulas deve atender ao conjunto de requisitos e respeitar as limitações de recursos disponíveis que representam as restrições do problema. Neste caso, os conflitos entre cursos são determinados de acordo com os módulos curriculares dos cursos, representados pelas turmas de alunos, e não de acordo com as matrículas efetuadas.

O problema consiste dos seguintes itens básicos:

- **Dias, Horários e Períodos**: Dado um número de dias úteis na semana (normalmente cinco ou seis), cada dia é dividido em um número fixo de períodos, que é igual para todos os dias. Um horário é representado pelo par composto por um dia e um período. O número total de horários de programação é o produto do número de dias pelo número de períodos;

- **Turmas**: Uma turma representa um grupo de alunos que assiste a um módulo curricular de um curso. Um módulo curricular, por sua vez, é composto por um grupo de disciplinas, de maneira que qualquer par de disciplinas deste grupo pode apresentar estudantes em comum;

- **Disciplinas e Professores**: Cada disciplina é composta por um número fixo de aulas (agendadas em períodos distintos), sendo frequentada por um determinado número de alunos e ministrado por um professor. Para cada disciplina, existe um número mínimo de dias nos quais as aulas da disciplina devem estar distribuídas, além disso, existem alguns períodos em que a disciplina não pode ser agendada, normalmente, em virtude da indisponibilidade do professor ou do turno de ocorrência do curso;

- **Salas**: Cada sala ou espaço tem uma capacidade (expressa em termos do número de alunos que pode abrigar) e uma localização (expressa por um valor inteiro que representa a numeração de um prédio dentro do *Campus*). Algumas salas podem não ser adequadas para alguns cursos (ausência de recursos).

Na formulação apresentada neste trabalho, uma *aula* é representada pela alocação de uma *turma*, *professor* e *sala* para a ocorrência de uma *disciplina* em um *horário* (*dia* da semana e *período* do dia).

A carga horária das disciplinas que devem ser agendadas, assim como as turmas que frequentarão as aulas, são dados de entrada do problema. Da mesma forma, as salas disponíveis com seus respectivos recursos também são relacionadas. O professor que deve ministrar cada disciplina é previamente determinado pela instituição. As indisponibilidades de turmas e professores, assim como a exigência de recursos em sala para cada disciplina, é uma informação pré-definida. O número total de aulas que devem ser programadas será definido pelo número de disciplinas $\times$ carga horária de cada disciplina.

4.1.1 Representação da Solução

O conjunto de aulas programadas por uma solução é representado por uma matriz binária definida na equação (4.1). Esta matriz representa os períodos de ocorrência das disciplinas quando alocadas a uma respectíva sala. Desta forma, sejam:

- C: conjunto de disciplinas com elementos $c \in C$ e identificados por $c = 1, \ldots, \bar{c}$;
- R: conjunto de salas com elementos $r \in R$ e identificados por $r = 1, \ldots, \bar{r}$;
- P: conjunto de períodos alocáveis com elementos $p \in P$ e identificados por $p = 1, \ldots, \bar{p}$.

Assim, a solução é dada pela matriz

$$\mathbf{X}_{c \times r \times p}\,, \tag{4.1}$$

onde $X_{crp} = 1$ indica que a disciplina c tem uma aula na sala r no período p, e $X_{crp} = 0$, caso contrário.

4.1.2 Custo da Solução

A importância do cumprimento ou não de uma restrição representa a relevância desta restrição na formulação do problema. A soma dos pesos associados a cada restrição violada por uma solução gerada é considerado o associado àquela solução. Desta forma, sejam

- V: conjunto de restrições do problema com elementos $v \in V$ e identificados por $v = 1, \ldots, \bar{v}$;
- w_v: peso da restrição de avaliação $f_v(\mathbf{X})$ na FO;
- $f_v(\mathbf{X})$: função de avaliação da restrição R_v.

A função objetivo, equação (4.2), minimiza o custo da solução encontrada quando o problema departamental é formulado como um problema de otimização e submetido ao algoritmo de solução:

$$\text{Minimizar} \quad \sum_{v=1}^{\bar{v}} \left[w_v \times f_v(\mathbf{X}) \right] , \tag{4.2}$$

sujeito às restrições definidas a seguir. Já o detalhamento de cada uma das funções de avaliação é apresentado na Seção 4.1.5.

4.1.3 Restrições

Para definição da formulação matemática das restrições empregadas no problema, foi realizado um estudo bibliográfico onde verificou-se a existência de diferentes formulações. Assim, a formulação matemática aqui apresentada está baseada na combinação de algumas propostas encontradas na literatura [40], [41], [42], [43].

Vale salientar que nesta Seção registra-se uma das contribuições deste trabalho. A formulação das restrições aqui apresentada é aplicável ao problema de Programação de Cursos baseada em Currículos da *International Timetabling Competition* e este autor não encontrou nenhum trabalho que apresentasse a formulação matemática completa das restrições consideradas neste problema.

Inicialmente, serão apresentadas a notação e a definição dos parâmetros utilizados na formulação matemática das restrições, apresentada em seguida. Portanto, sejam:

- C: conjunto de disciplinas com elementos $c \in C$ e identificados por $c = 1, \ldots, \bar{c}$;
- R: conjunto de salas com elementos $r \in R$ e identificados por $r = 1, \ldots, \bar{r}$;
- P: conjunto de períodos alocáveis com elementos $p \in P$ e identificados por $p = 1, \ldots, \bar{p}$;
- D: conjunto de dias da semana com elementos $d \in D$ e identificados por $d = 1, \ldots, \bar{d}$;
- G: conjunto de turmas com elementos $g \in G$ e identificados por $g = 1, \ldots, \bar{g}$;
- V: conjunto de restrições do problema com elementos $v \in V$ e identificados por $v = 1, \ldots, \bar{v}$;
- $\mathbf{DL}_{\bar{c}}$: vetor que define se uma disciplina exige aulas agrupadas ou não, onde $DL_c = 1$ se a disciplina c exige aulas agrupadas, e $DL_c = 0$ caso contrário;
- $\mathbf{CM}_{\bar{c} \times \bar{c}}$: matriz de turmas com disciplinas comuns, onde $CM_{c_1 c_2} = 1$ se existe uma turma que inclui as disciplinas c_1 e c_2, e $CM_{c_1 c_2} = 0$ caso contrário;
- $\mathbf{TM}_{\bar{c} \times \bar{c}}$: matriz de disciplinas com professores comuns, onde $TM_{c_1 c_2} = 1$ se existe um professor que leciona as disciplinas c_1 e c_2, e $TM_{c_1 c_2} = 0$ caso contrário;
- $\mathbf{A}_{\bar{c} \times \bar{p}}$: matriz de disponibilidades, onde $A_{cp} = 1$ se as aulas da disciplina c podem ser agendadas no período p, e $A_{cp} = 0$ caso contrário;
- $\mathbf{RS}_{\bar{c} \times \bar{r}}$: matriz de adequação das salas, onde $RS_{cr} = 1$ se as aulas da disciplina c podem ser agendadas na sala r, e $RS_{cr} = 0$ caso contrário;

- C_g: conjunto de disciplinas da turma g;
- B_r: prédio onde a sala r está localizada;
- P_d: conjunto de períodos do dia d;
- l_c: carga horária da disciplina c em número de aulas;
- md_c: mínimo de dias de ocorrência da disciplina c;
- s_c: máximo de alunos matriculados na disciplina c;
- min: mínimo de aulas por dia para cada turma;
- max: máximo de aulas por dia para cada turma;
- cap_r: capacidade da sala r;
- w_v: peso da restrição de avaliação $f_v(\mathbf{X})$ na FO;
- $f_v(\mathbf{X})$: função de avaliação da restrição R_v.

A seguir, são apresentadas as definiçõe de cada uma das restrições.

4.1.3.1 *Restrição* Aulas *[R01]*

Esta restrição determina que todas as aulas de uma disciplina devem ser agendadas e atribuídas a períodos distintos.

$$\sum_{r \in R} \sum_{p \in P} X_{crp} = l_c, \quad \forall\, c \in C. \tag{4.3}$$

A equação (4.3) é apresentada no trabalho de Avella e Vasil'ev [42].

4.1.3.2 *Restrição* Conflitos *[R02]*

Esta restrição determina que aulas de disciplinas do mesmo currículo (ou turma) devem ser agendadas em horários diferentes, assim como aulas ministradas pelo mesmo professor, devem ser agendadas em horários diferentes.

$$(X_{c_1rp} + X_{c_2rp})\, CM_{c_1c_2} \leq CM_{c_1c_2}, \quad \forall\, c_1,\, c_2 \in C, \quad \forall\, p \in P; \tag{4.4}$$

$$(X_{c_1rp} + X_{c_2rp})\, TM_{c_1c_2} \leq TM_{c_1c_2}, \quad \forall\, c_1,\, c_2 \in C, \quad \forall\, p \in P. \tag{4.5}$$

As equações (4.4) e (4.5) são apresentadas no trabalho de Avella e Vasil'ev [42].

4.1.3.3 *Restrição* Ocupação das salas *[R03]*

Esta restrição determina que duas aulas não podem ser agendadas na mesma sala e no mesmo horário.

$$\sum_{c \in C} X_{crp} \leq 1, \quad \forall\, r \in R, \quad \forall\, p \in P. \tag{4.6}$$

A equação (4.6) é apresentada no trabalho de Avella e Vasil'ev [42].

4.1.3.4 *Restrição* Disponibilidade *[R04]*

Esta restrição determina que se o professor da disciplina não está disponível para um determinado horário, nenhuma aula da disciplina pode ser agendada nesse período. Da mesma forma, existem horários que determinadas disciplinas não podem ser agendadas, por exemplo, em função do turno de ocorrência do curso.

$$X_{crp} \leq A_{cp}, \quad \forall\, c \in C, \quad \forall\, r \in R, \quad \forall\, p \in P. \tag{4.7}$$

A equação (4.7) é baseada no trabalho de Gaspero e Schaerf [41].

4.1.3.5 *Restrição* Capacidade das salas *[R05]*

Esta restrição determina que para cada disciplina, o número de alunos que frequentam as aulas deve ser menor, ou igual, ao número de lugares existentes na sala alocada.

$$X_{crp} \times s_c \leq cap_r, \quad \forall\, c \in C, \quad \forall\, r \in R, \quad \forall\, p \in P. \tag{4.8}$$

A equação (4.8) é baseada no trabalho de Gaspero e Schaerf [41].

4.1.3.6 *Restrição* Mínimo de dias de aula *[R06]*

Esta restrição determina as aulas de cada disciplina devem ser distribuídas em um determinado número mínimo de dias.

$$\sum_{d \in D} \left[\mathscr{H}\left(\sum_{r \in R} \sum_{p \in P_d} X_{crp} \right) \right] \geq md_c, \quad \forall\, c \in C; \tag{4.9}$$

onde a notação $\mathscr{H}(\cdot)$, utilizada na formulação desta restrição mas também das restrições [R07] e [R09], indica a clássica função de *Heaviside* discreta [44].

A equação (4.9) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.7 *Restrição* Aulas isoladas *[R07]*

Esta restrição (compactação de currículo 1) determina que as aulas pertencentes a um currículo (turma) devem ser adjacentes, ou seja, em períodos consecutivos de um mesmo dia.

$$\sum_{c \in C_g} \sum_{p-1 \in P_d}^{p+1 \in P_d} \sum_{r \in R} X_{crp}\, \mathscr{H}\left(\sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crp} \right) \geq 2\, \mathscr{H}\left(\sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crp} \right), \quad \forall\, d \in D, \forall\, g \in G. \tag{4.10}$$

A equação (4.10) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.8 *Restrição* Janelas *[R08]*

Esta restrição (compactação de currículo 2) determina que as aulas pertencentes a um currículo (turma) não devem ter janelas de tempo entre elas no mesmo dia, isto é, períodos sem agendamentos.

$$(X_{c''r''p''} \times p'') - (X_{c'r'p'} \times\ p') < \sum_{c \in C_g} \sum_{p=p'}^{p''} \sum_{r \in R} X_{crp},$$

$$\begin{aligned} &\forall\, c', c'' \in C_g, \quad \forall\, r', r'' \in R, \quad \forall\, p', p'' \in P_d, \quad \forall\, X_{c'r'p'} = 1, \\ &\forall\, X_{c''r''p''} = 1, \quad \forall\, p \geq p', \quad \forall\, p'' > p', \quad \forall\, d \in D, \quad \forall\, g \in G. \end{aligned} \tag{4.11}$$

A equação (4.11) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.9 Restrição Manutenção de sala *[R09]*

Esta restrição determina que todas as aulas de ums disciplina devem ocorrer na mesma sala.

$$\sum_{p \in P} X_{crp} \left[\mathscr{H}\left(\sum_{p \in P} X_{crp} \right) \right] = l_c\ \mathscr{H}\left(\sum_{p \in P} X_{crp} \right), \quad \forall\, c \in C, \quad \forall\, r \in R. \tag{4.12}$$

A equação (4.12) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.10 Restrição Mínimo e máximo de aulas por dia *[R10]*

Esta restrição determina que, para cada turma, o número de aulas diárias deve estar dentro de um determinado intervalo, indicando o número mínimo e máximo de aulas.

$$min \leq \sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crj} \leq max, \quad \forall\, g \in G, \quad \forall\, d \in D. \tag{4.13}$$

A equação (4.13) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.11 Restrição Deslocamentos *[R11]*

Esta restrição determina que os alunos devem ter tempo para se deslocar de um prédio para outro entre duas aulas. Duas aulas da mesmo turma, em períodos adjacentes no mesmo dia, devem ocorrer em salas localizadas no mesmo edifício.

$$(X_{c'r'p'} \times p') - (X_{crp} \times p) > 1,$$

$$\begin{aligned} &\forall\, c, c' \in C_g, \quad \forall\, r, r' \in R, \quad \forall\, p, p' \in P_d, \quad \forall\, X_{crp} = 1, \\ &\forall\, X_{c'r'p'} = 1, \quad \forall\, p' > p, \quad \forall B_r \neq B_{r'}, \quad \forall\, d \in D, \quad \forall\, g \in G. \end{aligned} \tag{4.14}$$

A equação (4.14) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.12 *Restrição* Adequação de salas *[R12]*

Esta restrição garante a adequação da sala, visto que algumas salas podem não ser adequadas para uma determinada disciplina por causa da ausência de recursos necessários (ginásio de esportes, projetor, computadores, etc).

$$X_{crp} \leq RS_{cr}, \quad \forall\, c \in C, \quad \forall\, p \in P, \quad \forall\, r \in R. \tag{4.15}$$

A equação (4.15) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.3.13 *Restrição* Agrupamento de aulas *[R13]*

Esta restrição determina que algumas disciplinas exigem que as aulas agendadas no mesmo dia sejam agrupadas. Duas aulas estão agrupadas se são realizadas na mesma sala e seus horários de realização são adjacentes.

$$\left[(X_{cr'p''} \times p'') - (X_{cr'p'} \times p')\right]\, DL_c < \sum_{r \in R} \sum_{p=p'}^{p''} X_{crp}\, DL_c,$$

$$\begin{aligned} &\forall\, c \in C_h, \quad \forall\, r' \in R, \quad \forall\, p', p'' \in P_d, \quad \forall\, X_{crp} = 1, \\ &\forall\, X_{crp'} = 1, \quad \forall\, p'' > p', \quad \forall\, p \geq p', \forall\, d \in D, \quad \forall\, g \in G. \end{aligned} \tag{4.16}$$

A equação (4.16) é de autoria deste autor e está baseada em equações de restrições similares encontradas nos trabalhos de Alvarez-Valdes e colaboradores [43], Gaspero e Schaerf [41], Aladag e Hocaougl [40] e Avella e Vasil'ev [42].

4.1.4 Viabilidade e Otimalidade

Todos os requisitos obrigatórios devem ser atendidos para que uma solução seja considerada viável, sendo desta forma denominados *restrições fortes*. Soluções que violem qualquer uma destas restrições serão consideradas fora do espaço de viabilidade.

Violações às demais restrições, contribuem para a perda de qualidade da solução, mas quadros de horário que possuam violações a estes requisitos ainda podem ser aceitos, caso apresentem boa qualidade. Estes requisitos são denominados *restrições fracas*.

A cada uma das restrições é atribuído um peso, que influenciará no grau de relevância deste requisito para a qualidade da solução final. Com isso, a *função objetivo* (FO) representa o somatório dos produtos entre o número de violações e os pesos de cada restrição, definindo desta forma o custo total da solução. Caso o peso atribuído à restrição seja zero, a restrição não é incluída no problema. Esta característica permite que sejam concebidas diferentes formulações, fazendo com que este modelo seja aplicável a diversas situações.

Uma vez que as *restrições fortes* devem ser atendidas a qualquer custo, soluções que apresentem violações a estas restrições são consideradas inviáveis. Sendo assim, em soluções viáveis, onde o custo correspondente às *restrições fortes* é zero, violações das *restrições fracas* alteram para qualidade da solução. Portanto, o custo de uma solução representa o grau de otimalidade desta solução.

4.1.5 Função Objetivo

Em função das restrições definidas na Seção 4.1.3 e das informações ali contidas, o problema de *Programação de Curso Baseada em Currículos* poder ser assim definido:

$$\begin{aligned} \text{Minimizar FO}(\mathbf{X}) &= w_1 f_1(\mathbf{X}) + w_2 f_2(\mathbf{X}) + w_3 f_3(\mathbf{X}) + w_4 f_4(\mathbf{X}) + w_5 f_5(\mathbf{X}) + \\ &+ w_6 f_6(\mathbf{X}) + w_7 f_7(\mathbf{X}) + w_8 f_8(\mathbf{X}) + w_9 f_9(\mathbf{X}) + w_{10} f_{10}(\mathbf{X}) + \\ &+ w_{11} f_{11}(\mathbf{X}) + w_{12} f_{12}(\mathbf{X}) + w_{13} f_{13}(\mathbf{X}); \end{aligned} \tag{4.17}$$

sujeito às restrições:
[R01], [R02], [R03], [R04], [R05], [R06], [R07], [R08], [R09], [R10], [R11], [R12] e [R13], definidas pelas equações (4.3)–(4.16), respectivamente, tal que:

- A função $f_1(\mathbf{X})$ avalia a restrição *Aulas* (R01). Todas as aulas de uma disciplina devem ser agendadas, e devem ser atribuídas a períodos distintos. Uma violação ocorre se uma aula não é agendada ou dois encontros são agendados no mesmo horário;

- A função $f_2(\mathbf{X})$ avalia a restrição *Conflitos* (R02). Aulas de disciplinas da mesma turma, ou ministradas pelo mesmo professor, devem ser todas agendadas em horários diferentes. Duas aulas no mesmo horário representam uma violação. Três aulas no mesmo horário contam como três violações, uma para cada par;

- A função $f_3(\mathbf{X})$ avalia a restrição *Ocupação das salas* (R03). Duas aulas não podem ser agendadas na mesma sala no mesmo horário. Duas aulas na mesma sala no mesmo horário, representam uma violação. Qualquer aula extra, no mesmo horário conta como mais uma violação;

- A função $f_4(\mathbf{X})$ avalia a restrição *Disponibilidade* (R04). Se o professor da disciplina não está disponível em um determinado horário, nenhuma aula da disciplina pode ser agendada nesse período. Cada aula em um horário de indisponibilidade do professor da diciplina é uma violação. Da mesma forma, existem horários que determinadas disciplinas não podem ser agendadas, em função do turno de ocorrência do curso, por exemplo;

- A função $f_5(\mathbf{X})$ avalia a restrição *Capacidade das salas* (R05). Para cada aula, o número de alunos que freqüentam a disciplina deve ser menor ou igual ao número de lugares da sala alocada. Cada aluno acima da capacidade da sala conta como uma violação;

- A função $f_6(\mathbf{X})$ avalia a restrição *Mínimo de dias de aula* (R06). As aulas de cada disciplina devem ser distribuídas em um determinado número mínimo de dias. Em uma distribuição inferior ao número mínimo de dias, cada dia a menos conta como uma violação;

- A função $f_7(\mathbf{X})$ avalia a restrição *Aulas isoladas* (R07). As aulas pertencentes a um currículo (turma) devem ser adjacentes, ou seja, em períodos consecutivos. Cada vez que uma aula de um currículo é não adjacente a qualquer outra aula deste currículo no mesmo dia, ela é cosiderada uma aula isolada. Cada aula isolada conta como uma violação;

- A função $f_8(\mathbf{X})$ avalia a restrição *Janelas* (R08). Aulas pertencentes a um currículo (turma) não devem ter janelas de tempo entre elas no mesmo dia, isto é, períodos sem agendamentos. Cada vez que há uma janela de tempo entre duas aulas do mesmo currículo no mesmo dia, o número de violações é igual ao tamanho da janela, em períodos;

- A função $f_9(\mathbf{X})$ avalia a restrição *Estabilidade de sala* (R09). Todas as aulas de uma disciplina devem ocorrer na mesma sala. Cada sala distinta da primeira alocada para as aulas de uma disciplina, conta como uma violação;

- A função $f_{10}(\mathbf{X})$ avalia a restrição *Mínimo e máximo de aulas por dia* (R10). Para cada turma o número de aulas diárias deve estar dentro de um determinado intervalo de número mínimo e máximo de aulas. Cada aula abaixo do mínimo ou acima do máximo representa uma violação;

- A função $f_{11}(\mathbf{X})$ avalia a restrição *Deslocamentos* (R11). Os alunos devem ter tempo para se deslocar de um prédio para outro entre duas aulas. É considerada uma violação toda vez que há um movimento instantâneo, isto é, quando duas aulas da mesma turma estão agendas em salas localizadas em edifícios diferentes em dois períodos adjacentes no mesmo dia;

- A função $f_{12}(\mathbf{X})$ avalia a restrição *Adequação de salas* (R12). Algumas salas podem não ser adequadas para uma determinada disciplina por causa da ausência de recursos necessários (ginásio de esportes, projetor, computadores, etc.). Cada aula de uma disciplina alocada em sala inadequada conta como uma violação;

- A função $f_{13}(\mathbf{X})$ avalia a restrição *Agrupamento de aulas* (R13). Algumas disciplinas exigem que as aulas agendadas no mesmo dia sejam agrupadas. Duas aulas estão agrupadas se o horário de ambas é adjacente e na mesma sala. Para uma disciplina que exige agrupamento de aulas, cada vez que há mais de uma aula em um mesmo dia, cada aula não agrupada conta como uma violação.

Algumas formulações de funções objetivo similares podem ser encontradas nos trabalhos encontrados na literatura, como os de Aladag e Hocaoglu [40], Gaspero e Schaerf [41], Avella e Vasil'ev [42] e Alvarez-Valdes e colaboradores [43].

4.2 Formulação Matemática

Com o objetivo de unificar toda a informação apresentada até o momento, a seguir é apresentada a formulação matemática completa do problema proposto. Os esclarecimentos de cada uma das equações forma já apresentados nas Seções 4.1.3 e 4.1.5.

Sejam:

- C: conjunto de disciplinas com elementos $c \in C$ e identificados por $c = 1, \ldots, \bar{c}$;

- R: conjunto de salas com elementos $r \in R$ e identificados por $r = 1, \ldots, \bar{r}$;

- P: conjunto de períodos alocáveis com elementos $p \in P$ e identificados por $p = 1, \ldots, \bar{p}$;

- D: conjunto de dias da semana com elementos $d \in D$ e identificados por $d = 1, \ldots, \bar{d}$;

- G: conjunto de turmas com elementos $g \in G$ e identificados por $g = 1, \ldots, \bar{g}$;
- V: conjunto de restrições do problema com elementos $v \in V$ e identificados por $v = 1, \ldots, \bar{v}$;
- $\mathbf{DL}_{\bar{c}}$: vetor que define se uma disciplina exige aulas agrupadas ou não, onde $DL_c = 1$ se a disciplina c exige aulas agrupadas, e $DL_c = 0$ caso contrário;
- $\mathbf{CM}_{\bar{c}\times\bar{c}}$: matriz de turmas com disciplinas comuns, onde $CM_{c_1c_2} = 1$ se existe uma turma que inclui as disciplinas c_1 e c_2, e $CM_{c_1c_2} = 0$ caso contrário;
- $\mathbf{TM}_{\bar{c}\times\bar{c}}$: matriz de disciplinas com professores comuns, onde $TM_{c_1c_2} = 1$ se existe um professor que leciona as disciplinas c_1 e c_2, e $TM_{c_1c_2} = 0$ caso contrário;
- $\mathbf{A}_{\bar{c}\times\bar{p}}$: matriz de disponibilidades, onde $A_{cp} = 1$ se as aulas da disciplina c podem ser agendadas no período p, e $A_{cp} = 0$ caso contrário;
- $\mathbf{RS}_{\bar{c}\times\bar{r}}$: matriz de adequação das salas, onde $RS_{cr} = 1$ se as aulas da disciplina c podem ser agendadas na sala r, e $RS_{cr} = 0$ caso contrário;
- C_g: conjunto de disciplinas da turma g;
- B_r: prédio onde a sala r está localizada;
- P_d: conjunto de períodos do dia d;
- l_c: carga horária da disciplina c em número de aulas;
- md_c: mínimo de dias de ocorrência da disciplina c;
- s_c: máximo de alunos matriculados na disciplina c;
- min: mínimo de aulas por dia para cada turma;
- max: máximo de aulas por dia para cada turma;
- cap_r: capacidade da sala r;
- w_1: peso da restrição de avaliação $f_1(\mathbf{X})$ na FO;
- w_2: peso da restrição de avaliação $f_2(\mathbf{X})$ na FO;
- w_3: peso da restrição de avaliação $f_3(\mathbf{X})$ na FO;
- w_4: peso da restrição de avaliação $f_4(\mathbf{X})$ na FO;
- w_5: peso da restrição de avaliação $f_5(\mathbf{X})$ na FO;
- w_6: peso da restrição de avaliação $f_6(\mathbf{X})$ na FO;
- w_7: peso da restrição de avaliação $f_7(\mathbf{X})$ na FO;
- w_8: peso da restrição de avaliação $f_8(\mathbf{X})$ na FO;
- w_9: peso da restrição de avaliação $f_9(\mathbf{X})$ na FO;

- w_{10}: peso da restrição de avaliação $f_{10}(\mathbf{X})$ na FO;
- w_{11}: peso da restrição de avaliação $f_{11}(\mathbf{X})$ na FO;
- w_{12}: peso da restrição de avaliação $f_{12}(\mathbf{X})$ na FO;
- w_{13}: peso da restrição de avaliação $f_{13}(\mathbf{X})$ na FO;
- $f_1(\mathbf{X})$: função de avaliação da restrição [R01];
- $f_2(\mathbf{X})$: função de avaliação da restrição [R02];
- $f_3(\mathbf{X})$: função de avaliação da restrição [R03];
- $f_4(\mathbf{X})$: função de avaliação da restrição [R04];
- $f_5(\mathbf{X})$: função de avaliação da restrição [R05];
- $f_6(\mathbf{X})$: função de avaliação da restrição [R06];
- $f_7(\mathbf{X})$: função de avaliação da restrição [R07];
- $f_8(\mathbf{X})$: função de avaliação da restrição [R08];
- $f_9(\mathbf{X})$: função de avaliação da restrição [R09];
- $f_{10}(\mathbf{X})$: função de avaliação da restrição [R010];
- $f_{11}(\mathbf{X})$: função de avaliação da restrição [R011];
- $f_{12}(\mathbf{X})$: função de avaliação da restrição [R012];
- $f_{13}(\mathbf{X})$: função de avaliação da restrição [R013].

$$\begin{aligned}\text{Minimizar FO}(\mathbf{X}) &= w_1 f_1(\mathbf{X}) + w_2 f_2(\mathbf{X}) + w_3 f_3(\mathbf{X}) + w_4 f_4(\mathbf{X}) + w_5 f_5(\mathbf{X}) + \\ &+ w_6 f_6(\mathbf{X}) + w_7 f_7(\mathbf{X}) + w_8 f_8(\mathbf{X}) + w_9 f_9(\mathbf{X}) + w_{10} f_{10}(\mathbf{X}) + \\ &+ w_{11} f_{11}(\mathbf{X}) + w_{12} f_{12}(\mathbf{X}) + w_{13} f_{13}(\mathbf{X}); \qquad (4.18)\end{aligned}$$

Sujeito a:

$$\sum_{r \in R} \sum_{p \in P} X_{crp} = l_c, \quad \forall\, c \in C; \qquad (4.19)$$

$$(X_{c_1 rp} + X_{c_2 rp})\, CM_{c_1 c_2} \leq CM_{c_1 c_2}, \quad \forall\, c_1,\, c_2 \in C, \quad \forall\, p \in P; \qquad (4.20)$$

$$(X_{c_1 rp} + X_{c_2 rp})\, TM_{c_1 c_2} \leq TM_{c_1 c_2}, \quad \forall\, c_1,\, c_2 \in C, \quad \forall\, p \in P; \qquad (4.21)$$

$$\sum_{c \in C} X_{crp} \leq 1, \quad \forall\, r \in R, \quad \forall\, p \in P; \qquad (4.22)$$

$$X_{crp} \leq A_{cp}, \quad \forall\, c \in C, \quad \forall\, r \in R, \quad \forall\, p \in P; \qquad (4.23)$$

$$X_{crp} \times s_c \leq cap_r, \quad \forall\, c \in C, \quad \forall\, r \in R, \quad \forall\, p \in P; \tag{4.24}$$

$$\sum_{d \in D} \left[\mathscr{H}\left(\sum_{r \in R} \sum_{p \in P_d} X_{crp} \right) \right] \geq md_c, \quad \forall\, c \in C; \tag{4.25}$$

$$\sum_{c \in C_g} \sum_{p-1 \in P_d}^{p+1 \in P_d} \sum_{r \in R} X_{crp}\, \mathscr{H}\left(\sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crp} \right) \geq 2\, \mathscr{H}\left(\sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crp} \right), \quad \forall\, d \in D, \forall\, g \in G; \tag{4.26}$$

$$(X_{c''r''p''} \times p'') - (X_{c'r'p'} \times\ p') < \sum_{c \in C_g} \sum_{p=p'}^{p''} \sum_{r \in R} X_{crp},$$

$$\begin{aligned} &\forall\, c', c'' \in C_g, \quad \forall\, r', r'' \in R, \quad \forall\, p', p'' \in P_d, \quad \forall\, X_{c'r'p'} = 1, \\ &\forall\, X_{c''r''p''} = 1, \quad \forall\, p \geq p', \quad \forall\, p'' > p', \quad \forall\, d \in D, \quad \forall\, g \in G; \end{aligned} \tag{4.27}$$

$$\sum_{p \in P} X_{crp} \left[\mathscr{H}\left(\sum_{p \in P} X_{crp} \right) \right] = l_c\, \mathscr{H}\left(\sum_{p \in P} X_{crp} \right), \quad \forall\, c \in C, \quad \forall\, r \in R; \tag{4.28}$$

$$min \leq \sum_{c \in C_g} \sum_{p \in P_d} \sum_{r \in R} X_{crj} \leq max, \quad \forall\, g \in G, \quad \forall\, d \in D; \tag{4.29}$$

$$(X_{c'r'p'} \times p') - (X_{crp} \times p) > 1,$$

$$\begin{aligned} &\forall\, c, c' \in C_g, \quad \forall\, r, r' \in R, \quad \forall\, p, p' \in P_d, \quad \forall\, X_{crp} = 1, \\ &\forall\, X_{c'r'p'} = 1, \quad \forall\, p' > p, \quad \forall B_r \neq B_{r'}, \quad \forall\, d \in D, \quad \forall\, g \in G; \end{aligned} \tag{4.30}$$

$$X_{crp} \leq RS_{cr}, \quad \forall\, c \in C, \quad \forall\, p \in P, \quad \forall\, r \in R; \tag{4.31}$$

$$\left[(X_{cr'p''} \times p'') - (X_{cr'p'} \times p') \right]\, DL_c < \sum_{r \in R} \sum_{p=p'}^{p''} X_{crp}\, DL_c,$$

$$\begin{aligned} &\forall\, c \in C_h, \quad \forall\, r' \in R, \quad \forall\, p', p'' \in P_d, \quad \forall\, X_{crp} = 1, \\ &\forall\, X_{crp'} = 1, \quad \forall\, p'' > p', \quad \forall\, p \geq p', \forall\, d \in D, \quad \forall\, g \in G. \end{aligned} \tag{4.32}$$

4.3 Estudos de caso

Com vistas à definição da formulação matemática a ser adotada, foram analisados dois estudos de caso: o primeiro refere-se ao problema de programação de cursos pós-matrícula abordado na *International Timetabling Competition*, e o segundo considera problemas encontrados no cenário de uma instituição de ensino superior brasileira. Para definição do problema abordado, os requisitos observados nos dois casos estudados foram combinados em uma única formulação aplicável a ambos os casos, cujo detalhamento foi apresentado nas Seções anteriores.

4.3.1 Estudo de Caso 1

O primeiro estudo de caso foi realizado considerando os problemas de Programação de Horário de Cursos Baseada em Currículos propostos pela *International Timetabling Competition* (ITC).

Na *International Timetabling Competition* (ITC), o problema de Programação de Horário de Cursos Baseada em Currículos consiste na programação semanal das aulas para diversos cursos universitários, dado um determinado número de salas e períodos de tempo, onde os conflitos entre os cursos são definidos de acordo com os currículos estabelecidos pela Universidade e não com base nos dados de matrícula de alunos.

O problema consiste dos seguintes itens básicos:

- Dias, Horários e Períodos;
- Cursos e Professores (onde um curso corresponde a uma disciplina no problema proposto na Seção 4.1);
- Salas;
- Currículos (onde um currículo corresponde a uma disciplina no problema proposto na Seção 4.1).

A solução do problema será representada por uma atribuição de um horário (dia e período) e de uma sala para todas as aulas de cada curso.

As restrições do problema são divididas em dois grupos: os básicos (restrições [R01]–[R05], apresentadas na Seção 4.1.3), que pertencem a todas as formulações e constituem o núcleo dos problemas de competição propostos pela ITC, e os opcionais (restrições [R06]–[R13], apresentadas na Seção 4.1.3), que são consideradas apenas em algumas formulações.

Na *International Timetabling Competition*, os pesos correspondentes a cada restrição são definidos por diferentes formulações, onde cada formulação prioriza um tipo ou um conjunto de restrições. A Figura 4.1 [45] apresenta os pesos referentes às formulações propostas na competição. O símbolo H indica a referida restrição como sendo do tipo forte, já os valores numéricos indicam a restrição como sendo do tipo fraca e representa o peso desta restrição a ser utilizado na função custo, equação (4.17).

Problem Formulation: Cost Component	UD1	UD2	UD3	UD4	UD5
Lectures	H	H	H	H	H
Conflicts	H	H	H	H	H
RoomOccupancy	H	H	H	H	H
Availability	H	H	H	H	H
RoomCapacity	1	1	1	1	1
MinWorkingDays	5	5	—	1	5
IsolatedLectures	1	2	—	—	1
Windows	—	—	4	1	2
RoomStability	—	1	—	—	—
StudentMinMaxLoad	—	—	2	1	2
TravelDistance	—	—	—	—	2
RoomSuitability	—	—	3	H	—
DoubleLectures	—	—	—	1	—

Figura 4.1: Pesos das restrições do formato UD.

A formulação UD1 foi introduzida pelos autores Gaspero e Schaerf e considera um número reduzido de restrições [17]. A formulação UD2 foi utilizada na edição 2007 da competição [46]. As formulações UD3, UD4 e UD5 [45] priorizam, respectivamente, carga horária dos estudantes, agrupamento de aulas e deslocamento entre aulas.

4.3.2 Estudo de Caso 2

O segundo estudo de caso foi realizado considerando os problemas de Programação de Horário de Cursos Baseada em Currículos de uma instituição de ensino superior brasileira, que oferece educação básica (ensino fundamental, médio, educação de jovens e adultos e ensino profissionalizante), cursos de graduação e pós-graduação e cursos de qualificação.

Nesta instituição, o problema geral de horários é composto de subproblemas por nível de ensino. Estes por sua vez, são compostos de subproblemas departamentais que representam a programação de ocorrência das aulas de um ou mais cursos durante um período letivo.

Como objetivo de pesquisa, foi selecionado um problema departamental. O problema representa a programação das grades de horário dos cursos profissionalizantes. Nos cursos técnicos, a construção das grades de horários é tarefa da coordenação, porém, os problemas de curso podem ser combinados, já que recursos, como professores e laboratórios, são de uso comum. A Figura 4.2 representa a combinação de problemas departamentais.

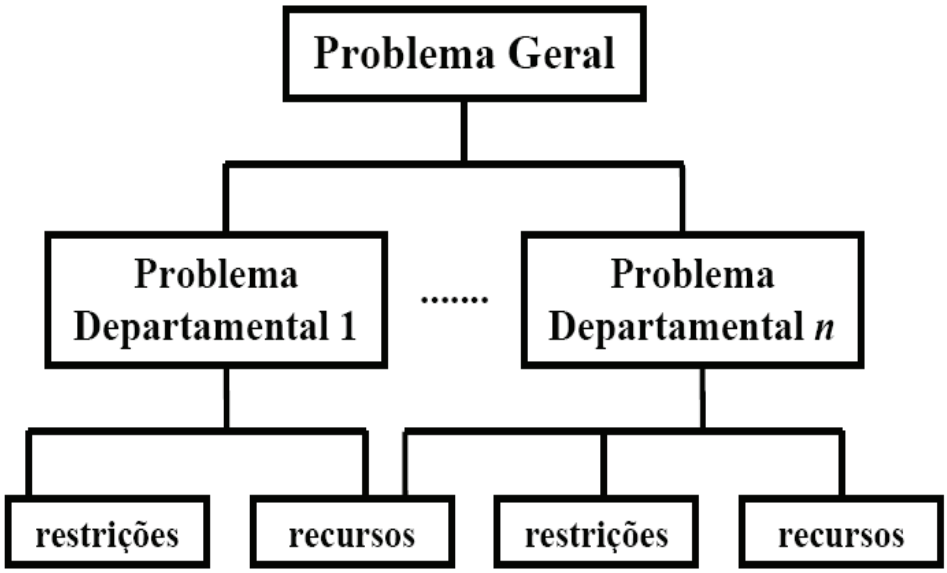

Figura 4.2: Problema de horário departamental.

No caso colocado anteriormente, o processo de construção das grades de horários ocorre em um processo pré-matrícula, quando então são definidos os períodos semanais em que cada componente curricular, ou disciplina de cada módulo dos cursos, irá ocorrer. Este problema corresponde ao problema de Programação de Horário de Cursos Baseada em Currículos apresentado na Seção 4.3.1.

Nos cursos técnicos, cada curso possui uma grade curricular associada. A grade curricular do curso representa a carga horária, isto é, o número de aulas que cada *disciplina* deverá ocorrer em cada módulo do curso, ou seja, o número de períodos semanais que cada *disciplina* deverá ocorrer em um período letivo. Uma *disciplina* equivale a um curso do cenário ITC, detalhado anteriormente.

O período semanal, ou simplesmente *período*, representa o índice sequencial para uma unidade de tempo (p_1, p_2, ..., p_n) de um dia da semana (segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira ou sábado). A combinação dia $\times$ período representa um *horário* em que uma aula pode ser agendada.

Como a programação das grades de horário ocorre em um processo pré-matrícula, o conceito de *turma* associa-se ao conceito de módulo. Sendo assim, um curso, cuja grade curricular prevê a conclusão em quatro semestres, terá que, a cada período letivo, programar as aulas para quatro turmas, cada uma relacionada aos alunos que irão matricular-se em cada módulo. Em comparação ao problema do ITC, uma *turma* equivale a um *currículo*.

Com base nos conceitos apresentados, programar uma *aula* significa determinar os períodos semanais em que ela ocorrerá, alocando os recursos necessários (*turma*, *professor* e *sala*) e respeitando um conjunto de requisitos físicos, pedagógicos e pessoais.

5 MÉTODO DE SOLUÇÃO

Para solução do problema apresentado no capítulo 4, foi desenvolvida uma ferramenta baseada no uso da metaheurística Busca Dispersa com uso da heurística Reconexão por Caminhos, apresentadas nas seções 2.3.6 e 2.3.7. Neste capítulo é descrita a forma na qual a metaheurística Busca Dispersa (*Scatter Search*) foi implementada, detalhando cada uma de suas etapas, bem como as estruturas de vizinhanças utilizadas.

5.1 Aplicação

Para definir um problema de horários, inicialmente devem ser relacionadas os recursos disponíveis e as restrições aplicáveis. Os dados relativos aos recursos disponíveis, por exemplo professores e salas, assim como as especificações dos currículos, utilizados para definir as aulas que deverão ser programadas para cada turma, são importados para a aplicação através de um processo denominado definição do problema. Neste processo, também são definidos os parâmetros utilizados pela metaheurística (ver Seção 5.2.1) e os pesos relacionados a cada uma das restrições aplicáveis. Aquelas restrições que tiverem seu peso definido como nulo, serão desconsideradas pela aplicação (ver Seção 2.2.2).

Uma vez definido o problema, tem início o processo de otimização, cujo algoritmo é descrito na Seção 5.2. A solução do problema será a melhor programação de horários fornecida pelo método ao fim do processo de otimização. A Figura 5.1 resume os processos executados pela ferramenta de solução.

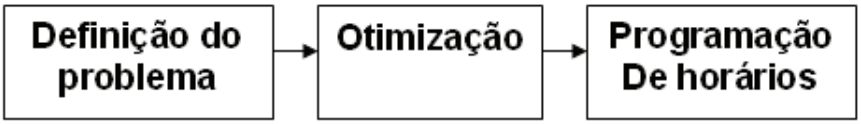

Figura 5.1: Fluxo de processos da ferramenta de solução.

5.2 Algoritmo Implementado

O processo de otimização é implementado por um algoritmo baseado na metaheurística Busca Dispersa, apresentada na Seção 2.3.6. A Figura 5.2 apresenta um diagrama de blocos com o funcionamento deste algoritmo.

O processo de otimização da metaheurística Busca Dispersa inicia com o emprego do método de geração de soluções diversas, que tem por objetivo gerar uma população P de soluções diversas e com tamanho $PSize$. Tal método tem sua aplicação considerando

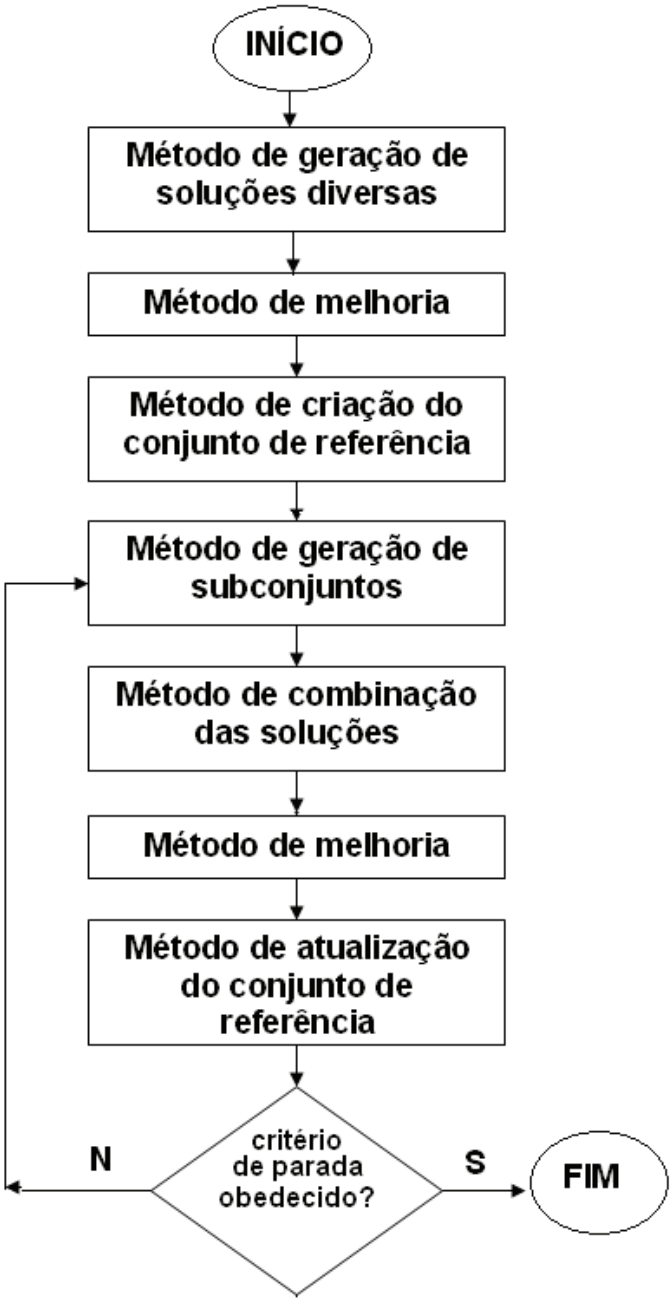

Figura 5.2: Algoritmo Busca Dispersa (SS) implementado.

apenas a restrição de que todas as aulas sejam programadas, desconsiderando violações às demais restrições do problema.

Uma vez geradas, as soluções são submetidas ao método de melhoria, que visa eliminar violações às restrições fortes, as quais inviabilizam uma solução, e também diminuir as violações às restrições fracas, que influenciam na otimalidade da solução.

A partir daí, através do emprego do método de criação do conjunto de referência, um conjunto menor, de tamanho b (geralmente com tamanho igual a 10% do tamanho de P) é gerado com as b_1 melhores soluções (aquelas com menor custo) e as b_2 soluções mais diversas (aquelas com um maior grau de diversidade em relação às melhores) de P. Este conjunto de soluções de referência é denotado por $RefSet$.

O conceito de melhores soluções está relacionado ao custo destas soluções. Uma

solução é melhor que outra se o seu custo é menor. O conceito de diversidade está relacionado ao número de diferenças entre uma solução e outra, ou em relação a um conjunto de soluções.

Após esta etapa, inicia-se um processo iterativo onde, a cada iteração, as soluções presentes no $RefSet$ são combinadas e melhoradas, até que um critério de parada seja obedecido. O critério de parada consiste no atendimento a uma das três hipóteses seguintes: (i) quando for encontrada uma solução viável de custo zero; (ii) quando o processo iterativo demonstrar que nenhuma nova solução, de melhor qualidade (ou diferente e de mesma qualidade), pode ser incorporada ao $RefSet$, quando oriunda da combinação e/ou melhoria das soluções anteriores e; (iii) quando um número máximo de iterações é atingido, evitando que a aplicação fique indefinidamente em execução.

Fazendo parte do processo iterativo, o método de geração de subconjuntos e o método de combinação das soluções são responsáveis pela criação do conjunto *Pool*, formado pelas soluções combinadas do $RefSet$. Antes de substituirem as piores e menos diversas soluções de $RefSet$, as soluções presentes em *Pool* são submentidas ao método de melhoria e posteriormente avaliadas. Esta etapa é realizada pelo método de atualização do conjunto de referência, que tem como objetivo melhorar a qualidade das soluções, através da eliminação das violações às restrições fortes e diminuição das violações às restrições fracas, tendo como resultado soluções viáveis de melhor qualidade.

A avaliação de uma solução significa contabilizar o número de violações a cada tipo de restrição (ver Seção 4.1.3) que esta solução apresenta e, assim, determinar o seu custo, conforme a equação (4.17). A solução final do problema será a melhor solução de $RefSet$, ou seja, aquela solução que apresentar o menor custo após o encerramento do processo iterativo.

5.2.1 Definição de Parâmetros

Para aplicação da metaheurística Busca Dispersa, alguns parâmetros devem ser definidos:

- *PSize*: tamanho do conjunto de soluções iniciais;
- b_1: número de melhores soluções que farão parte do conjunto de referência;
- b_2: número de soluções com maior diversidade que farão parte do conjunto de referência;
- $b = b_1 + b_2$: número total de soluções presentes no conjunto de referência;
- máximo de iterações: número máximo de iterações realizado sobre o conjunto de referência;
- ciclos de melhoria de P: número máximo de chamadas ao método de melhoria na etapa de melhoria das soluções de P;
- ciclos de melhoria de *Pool*: número máximo de chamadas ao método de melhoria na etapa de melhoria das soluções de *Pool*;
- máximo de vizinhaças: número máximo de vizinhaças exploradas por cada ciclo do método de melhoria.

No capítulo 6 são apresentados os testes realizados visando a sintonia destes parâmetros. Nas próximas seções deste capítulo serão detalhados os métodos utilizados no algoritmo de solução: geração de soluções diversas, melhoria (inicial, a cada iteração e final), atualização do conjunto de referência, geração de subconjuntos e combinação das soluções.

5.2.2 Método de Geração de Soluções Diversas

O Método de Geração de Soluções Diversas tem como objetivo gerar o conjunto de soluções iniciais. Este conjunto, denominado *P*, é composto por *Psize* soluções diferentes. A partir do problema definido, o método realiza o agendamento de cada aula prevista (número de disciplinas $\times$ carga horária de cada disciplina), alocando a esta aula uma sala e um horário. A ordem de programação das aulas, ou seja, das disciplinas, é aleatória para cada nova solução gerada.

Inicialmente, é feita uma escolha aleatória a partir do conjunto de salas disponíveis que satisfaçam o critério de Capacidade da sala (R05). Cada vez que uma sala atinge o seu valor máximo de alocação (valor presumido que representa os horários de disponibilidade de uma sala no problema), ela é eliminada do conjunto de salas disponíveis. Se nenhuma sala com capacidade igual ou maior a necessária (definida pelo número de alunos da disciplina) é encontrada, outra sala é escolhida, mesmo que o critério capacidade da sala não seja atendido. A restrição adequação da sala (R12) não é avaliada para geração de soluções iniciais.

O horário em que a aula irá ocorrer é escolhido de um conjunto contendo todos os períodos em que a turma está disponível. Baseado neste conjunto, inicialmente o método tenta encontrar um horário que não viole nenhuma das restrições fortes Conflitos (R02), Ocupação das salas (R03) e Disponibilidades (R04). No caso da restrição Conflitos, apenas conflitos de turma são considerados, deixando os conflitos relativos a alocação dos professores para serem resolvidos posteriormente. Se nenhum horário que não viole as restrições Ocupação das salas e Disponibilidades é encontrado, estas restrições são desconsideradas e qualquer horário que não viole a restrição de conflito de alocação de turma é escolhido.

A partir deste ponto, caso o conjunto de horários possíveis seja maior que o número de períodos a serem alocados, duas estratégias para a escolha do horário podem ser utilizadas:

1. Menor alocação: neste método, a rotina escolhe o horário com a menor alocação de turmas até o momento. Este método mostrou uma maior eficiência nos casos de problemas com poucas disponibilidades de sala, no que diz respeito à viabilidade das soluções iniciais;

2. Aleatório: neste método, a escolha do horário é feita de forma aleatória. Este método mostrou uma maior eficiência na criação de soluções com maior diversidade.

Uma vez determinado o método de escolha de horários para a alocação da primeira aula, este também será o método utilizado para programação de todos os horários da solução.

O método gera novas soluções até que o conjunto *P* seja composto de *Psize* soluções. As soluções geradas sempre respeitam a restrição forte Aulas (R01), ou seja, todas as aulas definidas pelo problema devem ser programadas. Caso um solução gerada pelo método não respeite esta restrição, esta solução não é adicionada a *P*. As outras restrições

do problema não são determinantes para a incorporação de uma solução a *P*, mesmo que algumas delas sejam consideradas durante o processo de geração de soluções.

A Figura 5.3 apresenta o pseudo-código do algoritmo utilizado como Método de Geração de Soluções Diversas.

```
Enquanto (existem aulas a alocar);
    Sorteia Estratégia de Alocação (menor alocação ou aleatória)*;
    Define Salas Disponíveis**;
    Sorteia sala;
    períodosDisponíveis = 0;
    Para cada período
       Se (turma = disponível) E (professor = disponível) E (sala = disponível)
          períodosDisponíveis++;
       Fim se;
    Fim para;
    Se (períodosDisponíveis < períodosNecessários)
       períodosDisponíveis = 0;
       Para cada período
          Se (turma = disponível) E (sala = disponível)
             períodosDisponíveis++;
          Fim se;
       Fim para;
    Fim se;
    Se (períodosDisponíveis < períodosNecessários)
       períodosDisponíveis = 0;
       Para cada período
          Se (turma = disponível) E (professor = disponível)
             períodosDisponíveis++;
          Fim se;
       Fim para;
    Fim se;
    Se (períodosDisponíveis < períodosNecessários)
       períodosDisponíveis = 0;
       Para cada período
          Se (turma = disponível)
             periodosDisponíveis++;
          Fim se;
       Fim para;
    Fim se;
Fim se;
```

Figura 5.3: Algoritmo do Método de Geração de Soluções Diversas.

As Figuras 5.4 e 5.5 apresentam, respectivamente, o pseudo-código dos algoritmos de Estratégia de Alocação (identificado por * na Figura 5.3) e de Definição de Salas Disponíveis (identificado por ** na Figura 5.3), utilizados internamente no algoritmo de Geração de Soluções Diversas.

```
Se (estratégia = alocação aleatória)
    Enquanto (existem aulas a alocar);
        sorteia período de PeríodosDisponíveis;
        aloca aula, professor, sala;
    Fim enquanto;
Fim se ;
Se (estratégia = menor alocação)
    Enquanto (existem aulas a alocar);
        sorteia período de PeríodosDisponíveis;
        aloca aula, professor, sala;
    Fim enquanto;
Fim se ;
```

Figura 5.4: *Algoritmo Estratégia de Alocação.

```
salasDisponíveis = 0;
Para cada sala
    Se (capacidadeSala ≥ alunosTurma)
        salasDisponíveis++;
    Fim se;
Fim para;
Se (salasDisponíveis = 0)
    salasDisponíveis ← todas as salas;
Fim se;
```

Figura 5.5: **Algoritmo Definição de Salas Disponíveis.

5.2.3 Método de Melhoria

O objetivo do método de melhoria é, através de um processo de busca em vizinhança, obter uma solução de melhor qualidade a partir da solução original. O método de melhoria é aplicado em dois momentos, após a criação dos conjuntos *P* e *Pool*.

Após a criação do conjunto *P*, o método de melhoria é chamado *nt* vezes para cada solução, onde *nt* é definido pelo parâmetro ciclos de melhoria de *P*. O número de vizinhanças *nv* (ver Seção 2.3.3) que serão exploradas é definido pelo parâmetro máximo de vizinhanças. A cada ciclo, é realizado o sorteio de um número entre 1 e *nv*, definindo assim a vizinhança que será explorada. Basicamente, *nv* define o número de movimentos em sequência, antes da avaliação, que serão aplicados à solução original para gerar uma solução vizinha. Com relação ao conjunto *Pool*, o processo é idêntico, porém é aplicado após a criação de cada novo conjunto, sendo, inicialmente, as soluções ordenadas em ordem descrescente de custo.

5.2.3.1 Tipos de Movimentos

A partir dos movimentos básicos, definidos na Seção 2.3.3, seis variações são constituídas a partir de suas combinações. Estas variações são apresentadas a seguir.

Inserção de sala (I_S): o movimento de inserção consiste em substituir, na programação de uma aula, a sala alocada por outra não alocada no mesmo horário. A Figura 5.6 representa um movimento deste tipo.

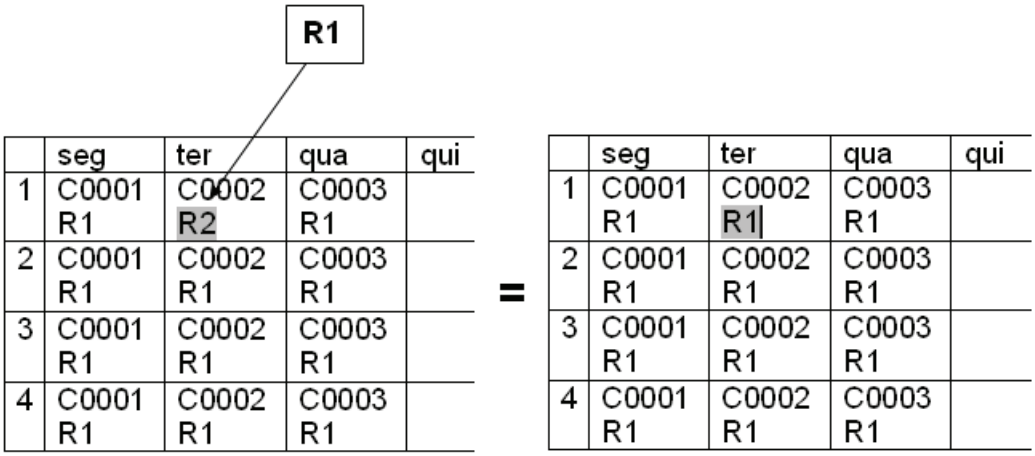

Figura 5.6: Movimento de Inserção de sala I_S.

Realocação de horário (R_H): o movimento de realocação de horário consiste realocar a programação de uma aula do horário original para outro, onde as turmas correspondentes não se encontram alocadas e a sala encontra-se disponível. A Figura 5.7 representa um movimento deste tipo.

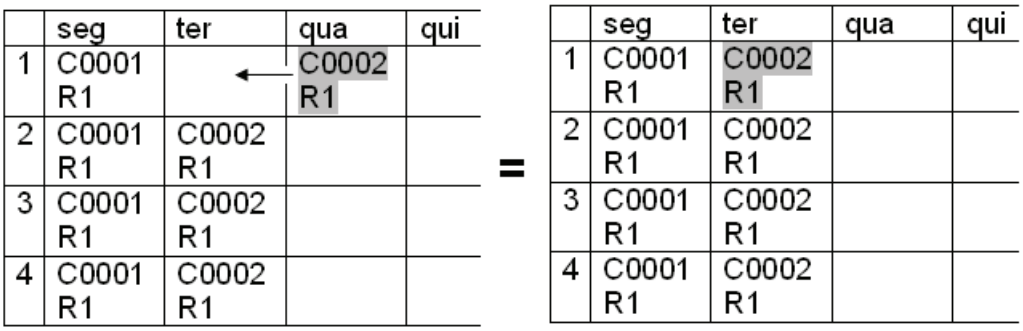

Figura 5.7: Movimento de Realocação de Horário R_H.

Realocação de horário com inserção de sala ($R_H I_S$): o movimento de realocação de horário com inserção de sala consiste realocar a programação de uma aula do horário original para outro onde as turmas correspondentes não se encontram alocadas e inserindo-se uma nova sala (considerando-se que a nova sala encontra-se disponível). A Figura 5.8 representa um movimento deste tipo.

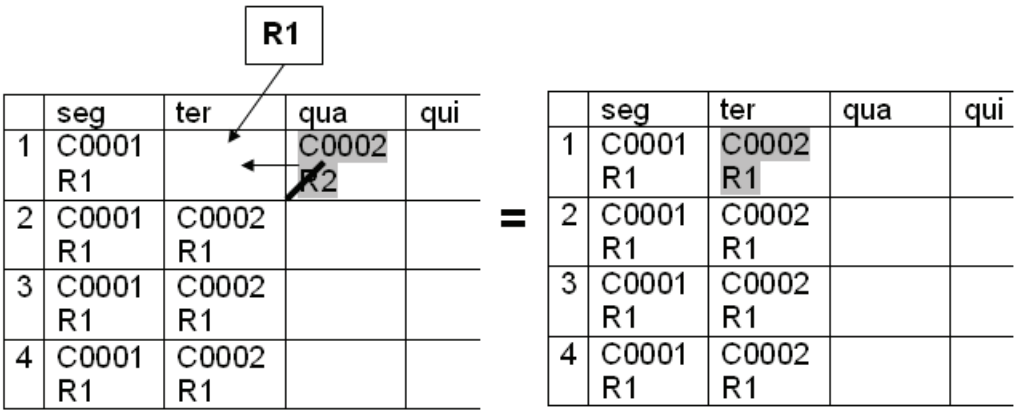

Figura 5.8: Movimento de Realocação de Horário com Inserção de Sala $R_H I_S$.

Troca de horários (T_H): o movimento de troca de horários consiste na troca dos horários de programação de duas aulas, de forma que a disciplina programada no horário de origem seja programada na mesma sala no horário de destino e vice-versa. A Figura 5.9 representa um movimento deste tipo.

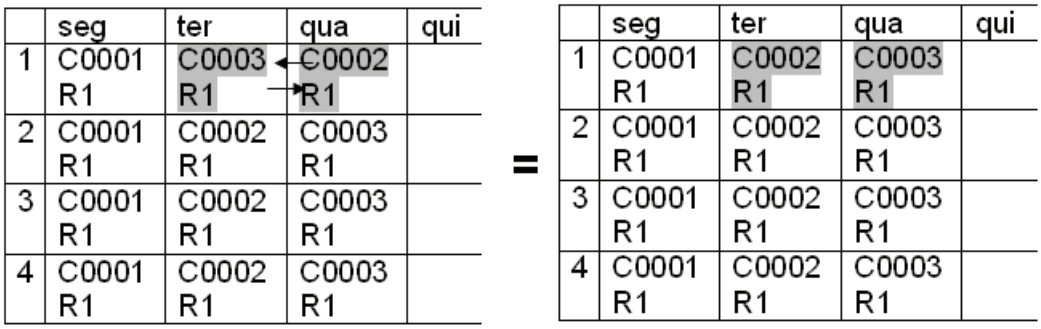

Figura 5.9: Movimento de Troca de Horários T_H.

Troca de salas (T_S): o movimento de troca de salas consiste na troca das salas alocadas na programação de duas aulas, de forma que a disciplina programada no horário de origem aconteça na sala programada no horário de destino e vice-versa. A Figura 5.10 representa um movimento deste tipo.

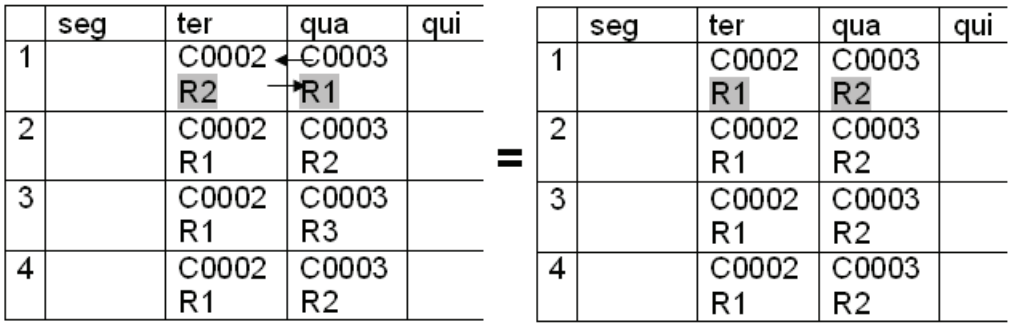

Figura 5.10: Movimento de Troca de Salas T_S.

Troca de disciplinas (T_D): o movimento de troca de disciplinas consiste na troca das disciplinas da programação de duas aulas, de forma que a disciplina programada no horário de origem aconteça no horário e sala de destino e vice-versa. A Figura 5.11 representa um movimento deste tipo.

	seg	ter	qua	qui
1		C0003 R1	C0002 R2	
2		C0002 R1	C0003 R2	
3		C0002 R1	C0003 R3	
4		C0002 R1	C0003 R2	

=

	seg	ter	qua	qui
1		C0002 R1	C0003 R2	
2		C0002 R1	C0003 R2	
3		C0002 R1	C0003 R2	
4		C0002 R1	C0003 R2	

Figura 5.11: Movimento de Troca de Disciplinas T_D.

Os trabalhos encontrados na literatura onde eram apresentadas alternativas de movimentos tratavam de problemas semelhantes, mas que não incluem todas as restrições previstas na formulação do problema abordado. As estruturas de vizinhança utilizadas na aplicação foram oriundas da combinação de estruturas estudadas, mas projetadas especificamente para atender ao conjunto de restrições previstas na formulação apresentada nesta dissertação.

5.2.3.2 Alvo do Movimento

Além do atributo tipo, a aplicação de um movimento é sempre realizada em um componente da solução. Sendo assim, cada tipo de movimento é realizado através da definição de parâmetros próprios. A seguir estes parâmetros são definidos:

- Inserção de sala: neste movimento os parâmetros são a turma, cuja programação de horários sofrerá a mudança, o horário em que a inserção de sala irá ocorrer, a sala que será inserida e o tamanho do bloco. O tamanho do bloco representa o número de períodos consecutivos de um movimento;

- Realocação de horário: neste movimento os parâmetros são a turma, cuja programação de horários sofrerá a mudança, o horário de origem, o horário de destino e o tamanho do bloco. Neste caso, o horário de destino deve ser diferente do horário de origem;

- Realocação de horário com inserção de sala: neste movimento os parâmetros são a turma, cuja programação de horários sofrerá a mudança, o horário de origem, o horário de destino, a sala que será inserida no horário de destino, e o tamanho do bloco. Neste caso, o horário de destino deve ser diferente do horário de origem;

- Troca de horários: neste movimento os parâmetros são a turma, cuja programação de horários sofrerá a mudança, o horário de origem, o horário de destino e o tamanho do bloco que serão permutados. Neste caso, o horário de destino deve ser diferente do horário de origem;

- Troca de salas: neste movimento os parâmetros são a turma de origem, o horário de origem, a turma de destino, o horário de destino, cuja sala será permutada (visto que salas podem ser permutadas entre turmas), e o tamanho do bloco. Caso as turmas sejam iguais, o horário de destino deve ser diferente do horário de origem. Da mesma forma, se as turmas forem diferentes, o horário de destino pode ser igual ao horário de origem;

- Troca de disciplinas: neste movimento os parâmetros são a turma, cuja programação de horários sofrerá a mudança, o horário de origem e o horário de destino, cuja disciplina será permutada, e o tamanho do bloco. Neste caso, o horário de destino deve ser diferente do horário de origem.

5.2.3.3 Avaliação de Custo

Para avaliação do custo de uma solução, ela deve ser submetida a uma rotina de avaliação, onde o total de violações a cada uma das restrições é computado. O custo final da solução é a soma dos produtos dos números de violações de cada restrição pelo seu respectivo peso na formulação avaliada. Se uma solução apresenta custo igual a uma outra, o número total de violações é avaliado. A solução que apresentar um número de violações menor ou igual é considerada a melhor.

5.2.3.4 Funcionamento do Método em P e $Pool$

A geração de um novo vizinho, a cada aplicação do método de melhoria, seja em P ou em $Pool$, acontece cumprindo as seguintes etapas:

1. O custo da solução s é computado. Os parâmetros número de vizinhanças (limitado ao número máximo de vizinhaças), turma de origem, turma de destino, horário de origem, horário de destino e tamanho do bloco são sorteados aleatoriamente;

2. Uma lista denominada tipos de movimentos, contendo um item para cada tipo de movimento (ver Seção 5.2.3.1) é criada;

3. Um item da lista tipos de movimentos é sorteado, correspondendo ao tipo de movimento que será aplicado. Uma vez que os parâmetros turma de origem, turma de destino, horário de origem e horário de destino já foram definidos, parâmetros adicionais (como sala) são sorteados aleatoriamente, se necessários;

4. Efetua-se uma tentativa de aplicação do movimento m à solução s:

   (a) Algumas vezes, a aplicação do movimento não é possível. Por exemplo, o tipo de movimento selecionado é $\oplus I_S$, mas não existem salas disponíveis no horário. Neste caso, se o tamanho da lista tipos de movimentos é maior que zero, retorna-se ao passo 3. Se o tamanho da lista tipos de movimentos é igual a zero, executa-se o passo 5;

   (b) Caso a tentativa de aplicação do movimento seja bem sucedida, isto é, se o número de vizinhanças exploradas (ou movimentos aplicados) seja menor que o parâmetro número de níveis de vizinhanças, decrementa-se 1 deste contador e retorna-se ao passo 3; caso contrário executa-se o passo 5;

5. Calcula-se o custo da solução, conforme descrito na Seção 5.2.3.3, e encerra.

```
calcula custoInicial;
Enquanto (ciclosDeMelhoria > 0)
    sorteia númeroDeVizinhanças*;
    listaDeMovimentos ← todos;
    Enquanto ((númeroDeVizinhanças > 0) E (listaDeMovimentos > 0) )
        sorteia tipoDeMovimento (de listaDeMovimentos);
        sorteia parâmetros**;
        resultado ← aplica movimento;
        Se (resultado = TRUE)
            númeroDeVizinhanças - -;
            listaDeMovimentos ← todos;
        Senão
            remove tipoDeMovimento da listaDeMovimentos;
        Fim se;
    Fim enquanto;
    calcula custoFinal;
    Se (custoFinal < custoInicial)
        solução ← novaSolução;
    Fim se;
Fim enquanto;
```

Figura 5.12: Algoritmo Método de Melhoria.

A Figura 5.12 apresenta o pseudo-algoritmo utilizado como Método de Melhoria. O parâmetro número de vizinhanças (identificado por * na Figura 5.12) é limitado ao número máximo de vizinhanças e define quantos movimentos serão aplicados.

Os parâmetros turma de origem, turma de destino, horário de origem, horário de destino e tamanho do bloco (identificados por ** na Figura 5.12) são sorteados aleatoriamente. Uma vez que os parâmetros turma de origem, turma de destino, horário de origem e horário de destino já foram definidos, parâmetros adicionais (como sala, por exemplo) são sorteados aleatoriamente, se necessários.

5.2.4 Método de Construção do Conjunto de Referência

O Método de Construção do Conjunto de Referência consiste em uma rotina que seleciona, a partir da população de soluções inicias P, um conjunto de soluções de referência denominado $RefSet$. O $RefSet$ é criado selecionando as melhores (menor custo) e as mais diversas soluções da população inicial, sendo o seu tamanho (b) relativamente menor que o tamanho da população inicial ($PSize$) [36].

O $RefSet$ é composto por dois subconjuntos: o $HQRefSet$, composto pelas b_1 soluções de menor custo, e o $DivRefSet$ composto pelas b_2 soluções com maior índice de diversidade, tal que $RefSet = HQRefSet \cup DivRefSet$, portanto $b = b_1 + b_2$.

Para determinar quais soluções farão parte de $HQRefSet$, as soluções em P, após a aplicação do método de melhoria, são ordenadas em ordem crescente de custo. As soluções de menor custo são selecionadas e removidas de P. O parâmetro $Psize$ é atualizado com o novo tamanho de P. Para determinar quais soluções farão parte de $DivRefSet$, as soluções em P são ordenadas em ordem descrescente de índice de diversidade, a ser

definido a seguir. As soluções mais diversas em relação a $HQRefSet$ são adicionadas a $DivRefSet$ e removidas de P. O parâmetro $Psize$ é atualizado com o novo tamanho de P.

5.2.4.1 Avaliação de Custo

Para avaliação do custo de uma solução, a solução é submetida a uma rotina específica que soma o número de violações a cada uma das restrições. Conforme já descrito anteriormente, o custo final da solução será a soma dos produtos dos números de violações de cada restrição pelo seu respectivo peso na formulação problema.

5.2.4.2 Avaliação de Diversidade

O índice de diversidade de uma solução representa o número de diferenças existentes entre uma solução e as demais. Uma diferença é interpretada como a disparidade de valor entre um termo específico das matrizes de duas soluções, isto é, sempre quando $T^a_{ikj} \neq T^b_{ikj}$ soma-se uma unidade ao índice de diversidade.

Por exemplo, na Figura 5.13, a aula da disciplina C0003 que ocorre na quinta-feira, primeiro período na solução de $Pool$, ocorre no segundo período na solução 1 de $RefSet$. Isto significa que, na comparação destas duas soluções, existem duas ocorrências nas quais os seus valores T_{ikj} estão em desacordo.

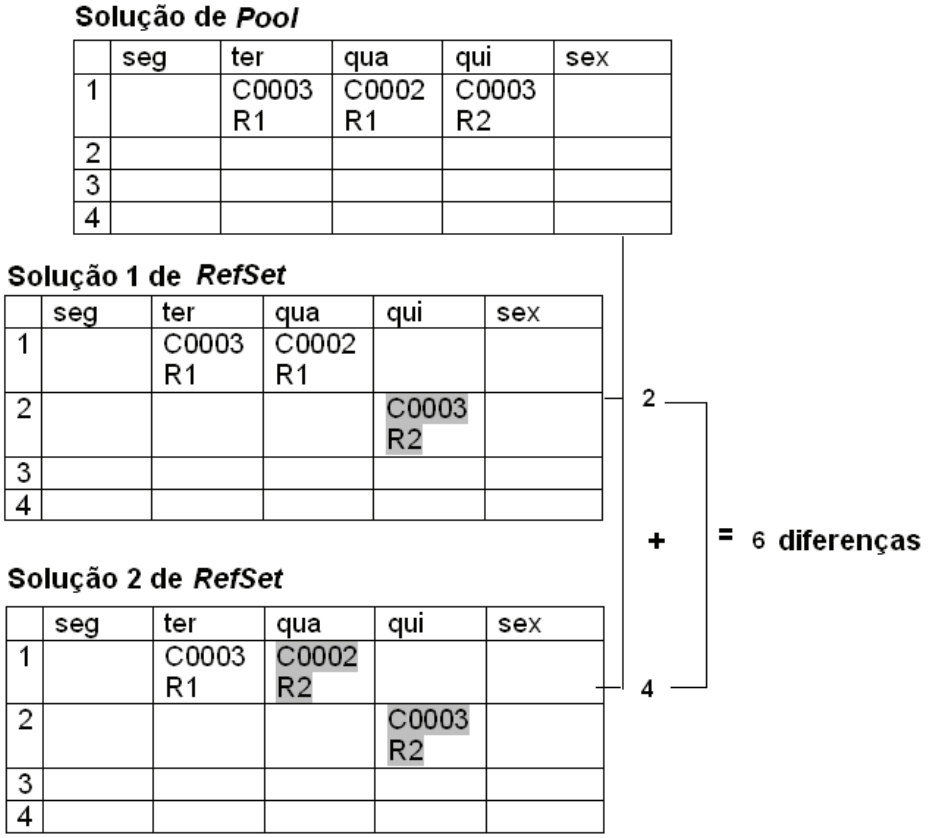

Figura 5.13: Cálculo do índice de diversidade da solução.

5.2.5 Método de Geração de Subconjuntos

O Método de Geração de Subconjuntos organiza subconjuntos derivados da combinação de todos os possíveis pares de soluções de $RefSet$ $((s_i, s_j), i = 1, \ldots, b$ e $j = 1, \ldots, b)$.

A partir daí, cada subconjunto, ou cada par de solução, é combinado através do Método de Combinação.

5.2.6 Método de Combinação

O Método de Combinação consiste em combinar as soluções presentes nos subconjuntos gerados pelo Método de Geração de Subconjuntos, adicionando-as em seguida a um novo conjunto, denominado *Pool*, que será avaliado para a atualização do $RefSet$.

O método utiliza uma rotina baseada na técnica Reconexão por Caminhos (Seção 2.3.7) para que, a partir do subconjunto (s_1, s_2), seja encontrada a solução s de maior qualidade no caminho entre s_1, dita solução inicial, e s_2 , dita solução guia. O detalhamento da rotina é apresentado a seguir:

1. Considere s_0 a solução inicial e s_g a solução guia. A rotina cria um nova solução s_e (eleita) idêntica a s_0;

2. Crie uma lista das turmas programadas na solução;

3. Se a lista tiver tamanho maior que zero, uma turma desta lista é selecionada aleatoriamente. Se o tamanho da lista de turmas for igual a zero, encerra-se;

4. Para cada aula programada em s_0, tenta-se aplicar um movimento (Seção 5.2.3.1) que transforme s_0 em s_g. Se possível, uma nova solução s_k é gerada, onde $k = 1,\ldots,n$, é um indice para a solução;

5. A rotina avalia as soluções s_k e seleciona a de menor custo. No primeira iteração (primeira turma sorteada), a solução s_k selecionada substitui s_e. Nas demais iterações, a solução s_k selecionada substitui s_e se $f(s_k) < f(s_e)$. A rotina retorna ao passo 3.

Sendo assim, se qualquer movimento é possível de ser realizado, o conjunto *Pool* recebe um nova solução s diferente e diferente das originárias s_1 e s_2.

5.2.7 Método de Atualização do Conjunto de Referência

O Método de Atualização do Conjunto de Referência acontece logo após a criação e melhoria do conjunto *Pool*. O método tem como objetivo atualizar o conjunto $RefSet$ a partir das novas soluções geradas.

Inicialmente, o conjunto *Pool* é comparado com $RefSet$. Todas as soluções idênticas são eliminadas de *Pool*. Em seguida, as soluções de *Pool* são ordenadas em ordem decrescente de custo. São mantidas as b melhores soluções, enquando as demais são eliminadas do conjunto. Esta estratégia foi utilizada para evitar um tempo excessivo de processamento, desconsiderando a melhora de soluções que, possivelmente, não serão incluídas em $RefSet$ posteriormente.

São utilizadas duas formas de atualização. A primeira é baseada em qualidade, sendo utilizada quando o conjunto *Pool* tem pelo menos uma solução de custo inferior à melhor solução de $RefSet$. A segunda forma é baseada em qualidade e diversidade, sendo utilizada quando, após duas iterações, o conjunto *Pool* não apresentar nenhuma solução de custo inferior à melhor solução de $RefSet$.

A cada novo conjunto *Pool*, isto é, a cada nova iteração, os custos das soluções geradas são comparados com a melhor solução de $RefSet$. Se *Pool* contém um solução de custo inferior, o parâmetro iterações sem melhoria é reinicializado com zero. Se a melhor

solução de *Pool* é maior que a melhor solução em $RefSet$, o parâmetro iterações sem melhoria sofre um incremento. Se iterações sem melhoria for igual ou superior a 2, e *PSize* for maior que b_2, então é realizada a atualização baseada em qualidade. Se iterações sem melhoria for menor que 2, ou *Pzise* for menor que b_2, então é realizada a atualização baseada em qualidade e diversidade. A seguir, ambas estratégias são apresentadas em detalhe.

5.2.7.1 Atualização por Qualidade

- O novo $HQRefSet$ é composto pelas b_1 melhores soluções de $RefSet$. O conjunto é ordenado em ordem decrescente de custo. Se o custo da melhor solução do conjunto *Pool* for menor do que o custo da pior solução em $HQRefSet$, a solução de *Pool* substitui a solução de $RefSet$, sendo posteriormente removida *Pool*. Se o custo da melhor solução em *Pool* é igual a pior solução em $RefSet$, o número total de violações é avaliado. A solução que apresentar o menor número de violações é considerada melhor. $HQRefSet$ é novamente ordenado e uma nova comparação é efetuada até que todas as soluções de *Pool* sejam testadas.

- O novo $DivRefSet$ é composto pelas b_2 mais diversas soluções de $RefSet$. O conjunto é ordenado em ordem crescente de diversidade, utilizando os mesmo critérios do método de criação do conjunto de referência descrito na Seção 5.2.4.

  Se o índice de diversidade da solução mais diversa em *Pool* for maior do que o índice de diversidade da solução menos diversa em $DivRefSet$, a solução de *Pool* substitui a solução de $DivRefSet$, sendo posteriormente removida de *Pool*. Se o índice de diversidade da solução mais diversa em *Pool* for igual ao índice de diversidade da solução menos diversa em $DivRefSet$, é escolhida a solução de menor custo. Se os custos também forem iguais, é escolhida a solução com o menor número de violações. $DivRefSet$ é novamente ordenado e uma nova comparação é executada até que todas as soluções de *Pool* sejam testadas.

- O novo conjunto $RefSet$ é composto por $HQRefSet \cup DivRefSet$, portanto ainda mantém-se $b = b1 + b2$.

5.2.7.2 Atualização por Qualidade e Diversidade

- O novo $HQRefSet$ será composto pelas b_1 melhores soluções de $RefSet$. O conjunto é ordenado em ordem decrescente de custo. Se o custo da melhor solução do conjunto *Pool* for menor que o custo da pior solução em $RefSet$, a solução de *Pool* substitui a solução de $RefSet$, sendo posteriormente removida de *Pool*. Se o custo da melhor solução em *Pool* é igual a pior solução em $RefSet$, o número total de violações é avaliado. A solução que apresentar o menor número de violações é considerada melhor. O conjunto $RefSet$ é novamente ordenado, sendo efetuada uma nova comparação até que todas as soluções de *Pool* sejam avaliadas.

- O novo $DivRefSet$ é composto pelas b_2 melhores soluções de P, sendo posteriormente removidas de P. O parâmetro *Psize* é atualizado com o novo tamanho de P.

6 VALIDAÇÃO E TESTES

Este capítulo apresenta a descrição e os resultados dos testes realizados com o modelo apresentado anteriormente nos capítulos 4 e 5. O capítulo está organizado em quatro seções:

1. Ambiente de testes: define o ambiente computacional de testes;
2. Instâncias de problema: define quais as instâncias utilizadas para a validação e testes;
3. Ajuste de parâmetros: apresenta os testes referentes ao ajuste dos parâmetros utilizados na ferramenta de solução.
4. Validação: apresenta os resultados de validação do modelo e do método de solução.
5. Testes de comparação: apresentam os resultados comparativos dos testes realizados.

6.1 Ambiente de Testes

O programa foi implementado em linguagem Java (*Version* 6), ambiente Windows 32 bits. Todos os testes foram realizados em um computador com processador Pentium Dual-Core 2.5GHz e memória RAM de 2GB.

6.2 Instâncias de Problema

Para definição do problema, são necessários dois conjuntos de informação: (*i*) a instância que define as disponibilidades de recursos e; (*ii*) os pesos correspondentes a cada restrição do problema.

6.2.1 Descrição das Instâncias

As instâncias utilizadas representam problemas dos dois estudos de caso descritos nas seções 4.3.1 e 4.3.2, correspondentes, respectivamente, à *International Timetabling Competition* e a uma universidade local.

As instâncias da *International Timetabling Competition*, utilizadas nos testes, são denominadas *toy* e comp01 [1]:

[toy]: 4 disciplinas, 3 salas, 5 dias da semana, 4 períodos por dia, 2 turmas e 8 restrições de indisponibilidade.

[1] `http://tabu.diegm.uniud.it/ctt/index.php?page=instances`

[comp01]: 30 disciplinas, 6 salas, 5 dias da semana, 6 períodos por dia, 14 turmas, no mínimo 2 aulas e no máximo 5 aulas por dia, 53 restrições de indisponibilidade e 23 restrições de sala.

A instância denominada real01, corresponde a um problema real de programação de aulas de cursos em uma universidade local:

[real01]: 17 disciplinas, 9 salas, 5 dias da semana, 4 períodos por dia, 14 turmas, 4 aulas como mínimo e máximo por dia e 68 restrições de indisponibilidade.

Os dados referentes a instância que define o problema são importados de um arquivo texto no formato `ectt` [2], o mesmo utilizado na *International Timetabling Competition*.

Neste formato, cada problema é definido em um arquivo único, contendo um cabeçalho e cinco seções: disciplinas, salas, turmas, restrições de indisponibilidade e restrições de adequação de sala. O cabeçalho fornece todos os valores escalares necessários: número de disciplinas, número de salas, número de dias da semana, número de períodos por dia, turmas, mínimo e máximo de aulas por dia, número de restrições de indisponibilidade e número de restrições de adequação de salas. Cada seção fornece as matrizes do problema. As seções são disciplinas, salas, turmas, restrições de indisponibilidades e restrições de indisponibilidade de sala e são descritas a seguir:

- Disciplinas: ID da disciplina, Professor, Número de aulas, Mínimo espalhamento de dias, Número de alunos, Aulas agrupadas;
- Salas: ID da sala, Capacidade da sala, Prédio;
- Turmas: ID da turma, Número de disciplinas, ID da disciplina;
- Restrições de indisponibilidade: ID da disciplina, Dia da semana, Período do dia;
- Restrições de sala: ID da disciplina, ID da sala.

Todos os IDs são *strings* sem espaços e iniciados por uma letra. A contagem de Dias e Períodos começa em 0. Por exemplo, a restrição de indisponibilidade TecCos 3 2 representa que a disciplina TecCos não pode ser agendada no terceiro (2) período de quinta-feira (3). No mesmo exemplo, a restrição de sala TecCos R1 representa que a disciplina TecCos não pode ser agendada na sala R1.

6.2.2 Pesos Atribuídos às Restrições

A definição dos parâmetros e dos pesos que serão atribuídos a cada restrição também são definidos por um arquivo texto correspondente a cada teste realizado. O peso atribuído a uma restrição define a importância desta restrição no problema avaliado.

Para os testes com as instâncias de competição, foram selecionadas as formulações UD2, UD3 e UD4 (ver Seção 4.3.1). A tabela 6.2.2 apresenta os pesos para estas formulações. Nesta tabela, o símbolo F indica uma restrição forte, enquanto que o valor numérico corresponde ao peso desta restrição no problema.

[2] `http://tabu.diegm.uniud.it/ctt/index.php?page=format#`

Tabela 6.1: Pesos das restrições do formato UD (ITC).

restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
UD2	F (*)	F (*)	F (*)	F (*)	1	5	2	1	0	0	0	0	0	-
UD3	F (*)	F (*)	F (*)	F (*)	1	0	0	4	0	2	0	3	0	-
UD4	F (*)	F (*)	F (*)	F (*)	1	1	0	1	0	1	0	F (*)	1	-

* Restrição forte

Na instância correspondente à universidade local, todas as restrições tem o mesmo nível de importância no problema, portanto, no teste inicial, todas as restrições recebem peso 1. Foi realizado um teste adicional, atribuindo-se um peso maior à restrição R10 (mínimo e máximo de aulas por dia) por ser aquela com maior dificuldade de solução.

6.3 Ajuste de Parâmetros

O objetivos dos testes de ajuste de parâmetros é a definição da melhor configuração de parâmetros para realização dos testes de resultado. Os testes de ajuste, ou sintonia, de parâmetros foram realizados em três etapas:

1. ajuste do tamanho do conjunto P (conjunto das soluções diversas) definido por *PSize* e ajuste do número de ciclos de melhoria de P, parâmetro ciclos de melhoria de P;

2. ajuste do tamanho do conjunto $RefSet$ (conjunto de referência), definido pelos parâmetros b_1 e b_2;

3. ajuste da melhoria do conjunto *Pool*, parâmetros ciclos de melhoria de P e máximo de vizinhanças.

Para fins de comparação, todos os testes de ajuste de parâmetros foram realizados com a instância comp01, instância de competição da *International Timetabling Competition*. Os pesos das restrições fortes do problema foram fixados em 100 e os pesos das restrições fracas seguem as definições da formulação UD2 da *International Timetabling Competition* de 2007.

Em todos os testes, o critério de parada, ou seja, número máximo de laços de iterações sobre o conjunto de referência, foi fixado em 200.

6.3.1 Ajuste do Tamanho e Ciclos de Melhoria de P

Esta rodada de testes executou apenas os métodos de criação das soluções diversas e de melhoria destas soluções.

O objetivo destes testes foi verificar o melhor tamanho do conjunto P, definido por *PSize*, e o melhor parâmetro ciclos de melhoria de P, que define o número de ciclos de melhoria neste conjunto. Foram realizados quatro testes: AP_{01}, AP_{02}, AP_{03} e AP_{04}, cujos resultados são apresentados na tabela 6.2.

Tabela 6.2: Ajuste de tamanho e ciclos de melhoria de P.

nome	*Psize*	ciclos de melhoria	custo de s^P (*) antes da melhoria	custo de s^P (*) após melhoria
AP_{01}	100	10	2166	2166
AP_{02}	100	30	2260	2260
AP_{03}	300	10	2040	2040
AP_{04}	300	30	2168	2160

* s^P = melhor solução de P

Na tabela 6.2, o símbolo s^P indica o valor assumido pela função objetivo na solução de menor custo do conjunto P. Este valor foi computado antes e depois da melhoria para fins de avaliação.

Os testes indicaram que, quanto maior o valor de *PSize*, menor o custo da melhor solução do conjunto inicial. Além disso, mostrou também que o valor utilizado para o parâmetro ciclos de melhoria de P não influencia significativamente na qualidade da melhor solução inicial. Desta forma, a melhor configuração é aquela com $PSize = 300$ e ciclos de melhoria de $P = 10$.

6.3.2 Ajuste do Tamanho de $RefSet$

O objetivo destes testes é verificar qual o melhor valor para os parâmetros b_1 e b_2 em termos do percentual de melhora em relação à melhor solução inicial gerada.

Nesta rodada de testes, o parâmetro b (tamanho do conjunto de referência) foi fixado em 10% de *PSize* (tamanho do conjunto P), com *PSize* = 100, 150 e 200. O valor de b_1 (melhores soluções do conjunto de referência) é assumido igual a $b/2$, se b for um número par, e $(b-1)/2+1$ se b for um número impar. O valor de b_2 (mais diversas soluções do conjunto de referência) é definido por $b - b_1$.

Os valores para os parâmetros referentes aos ciclos de melhoria de P (conjunto de soluções diversas) e de *Pool* (conjunto de soluções combinadas), representados por ciclos de melhoria de P e ciclos de melhoria de Pool, respectivamente, foram fixados em 1. O valor para o parâmetro máximo de vizinhaças também foi fixado em 1.

Foram realizados três testes: AP_{11}, AP_{12} e AP_{13}, cujos resultados são apresentados na tabela 6.3.

Tabela 6.3: Ajuste do tamanho de $RefSet$.

nome	*Psize*	b	$b1$	$b2$	custo de s^P (*)	custo de s^{RefSet} (†)	iteração de s^{RefSet} (†)	% (‡)
AP_{11}	100	10	5	5	2253	172	98	92,36
AP_{12}	150	15	8	7	2158	132	71	93,88
AP_{13}	200	20	10	10	2165	175	99	91,91

* s^P = melhor solução de P após melhoria
† s^{RefSet} = melhor solução encontrada
‡ % de melhora da solução s^{RefSet} em relação a s^P

Na tabela 6.3, o símbolo s^P indica o valor assumido pela função objetivo na solução

de menor custo do conjunto P após a melhoria inicial. O parâmetro s^{RefSet} indica o valor assumido pela função objetivo na solução de menor custo do conjunto $RefSet$ após a otimização, ou seja, a solução final. A tabela também apresenta em qual ciclo de iteração esta solução foi encontrada. A última coluna indica a porcentagem de melhora de s^{RefSet} em relação a s^P, definida pela expressão $(s^P - s^{RefSet})/s^P$.

Os testes indicaram que existe uma melhora pouco significativa de qualidade quando o tamanho de $RefSet$ cresce de $b = 10$ para $b = 15$, mas este aumento pouco significativo de qualidade representa um aumento significativo (aproximadamente uma hora) de tempo de processamento. Incrementos no tamanho do conjunto de referência, levando a valores maiores que $b = 15$, não representam aumento na qualidade das soluções geradas. Portanto, a melhor configuração de tamanho para $RefSet$ é $b = 10$.

6.3.3 Ajuste da Melhoria de *Pool*

O objetivo destes testes é (*i*) verificar o melhor valor para o número de ciclos de melhoria do conjunto *Pool* (conjunto de soluções combinadas), representados por ciclos de melhoria de Pool, e (*ii*) verificar o melhor valor para o parâmetro máximo de vizinhanças exploradas.

Foram fixados os valores dos seguintes parâmetros: $b = 10$, $b_1 = 5$ e $b_2 = 5$.

O parâmetro ciclos de melhoria de P foi fixado em 10. Com estes parâmetros determinados, foram realizados quatro diferentes testes: AP_{21}, AP_{22}, AP_{23} e AP_{24}, cujos resultados são apresentados na tabela 6.4.

Tabela 6.4: Ajuste da melhoria de *Pool*.

nome	ciclos de melhoria de *Pool*	máximo de vizinhanças	custo de s^P (*)	custo de s^{RefSet}	iteração de s^{RefSet} (†)	% (‡)
AP_{21}	10	4	2058	196	98	90,47
AP_{22}	10	6	1781	131	96	92,64
AP_{23}	30	6	2114	131	98	93,80
AP_{24}	100	6	2084	67	100	96,78

* s^P = melhor solução de P após melhoria
† s^{RefSet} = melhor solução encontrada
‡ % de melhora da solução s^{RefSet} em relação a s^P

Na tabela 6.4, o símbolo s^P indica o valor assumido pela função objetivo na solução de menor custo do conjunto P após a melhoria inicial. O parâmetro s^{RefSet} indica o valor assumido pela função objetivo na solução de menor custo do conjunto $RefSet$ após a otimização, ou seja, a solução final. A tabela também apresenta em qual ciclo de iteração esta solução foi encontrada. A última coluna indica a porcentagem de melhora de s^{RefSet} em relação a s^P, conforme expressão definida no caso de ajuste anterior.

Os testes indicaram que quanto maior o valor para o parâmetro ciclos de melhoria de *Pool*, melhor a qualidade das soluções geradas, mas este parâmetro influencia diretamente no tempo de processamento. Desta forma, assumiu-se ciclos de melhoria de *Pool* = 100 e o parâmetro máximo de vizinhanças = 6.

6.4 Validação do Modelo

O programa utilizado gera como saída um arquivo texto com a melhor programação de horários encontrada. Outro arquivo texto, auxiliar, descreve o número de violações de cada restrição e o custo total correspondente a cada solução.

O arquivo de programação de horários, no caso das instâncias da *International Timetabling Competition*, pode ser submetido a um validador da competição [3]. Portanto, para validação do modelo, foi selecionada a instância *toy* da *International Timetabling Competition*.

Foram realizados três testes com a instância *toy*, conforme as definições de pesos utilizadas nas formulações UD2 e UD3 e UD4. Estes testes foram denominados TOY_{11}, TOY_{12} e TOY_{13}, respectivamente, e seus resultados são apresentados nas tabelas 6.4–6.4.

Para realização dos testes, foram assumidos os parâmetros apresentados na tabela 6.5.

Tabela 6.5: Parâmetros utilizados nos testes de validação.

Psize	*b*1	*b*2	ciclos de melhoria de *P*	ciclos de melhoria de *Pool*	máximo de vizinhanças	critério de parada (número de ciclos)
100	5	5	10	100	6	200

Na tabela 6.6, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, tais restrições recebem um peso significativamente superior àqueles atribuídos às restrições R05, R06, R07 e R08, que representam as restrições fracas do problema.

O custo total da solução, composto pelo somatório de custos das restrições fortes e fracas (como definido na função objetivo apresentada na Seção 4.1.5), é representado na coluna custo.

Tabela 6.6: Testes TOY_{11}.

nome: TOY_{11} instância: toy pesos: UD2														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	5	2	1	0	0	0	0	0	-
ótimo	0	0	0	0	0	0	0	0	-	-	-	-	-	0
melhor SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
média	0	0	0	0	0	0	0	0	0	0	0	0	0	0
desvio padrão	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A linha melhorSS representa o custo da melhor solução obtida na rodada de testes correspondente às violações a cada restrição (indicada pelas respectivas colunas).

A linha média representa a média de violações a cada uma das restrições, considerando todos os testes realizados.

[3] `http://tabu.diegm.uniud.it/ctt/index.php?page=valid`

A linha desvio padrão representa o desvio padrão referente as violações a cada uma das restrições, considerando todos os testes realizados.

Em todos os testes, a função objetivo atingiu o custo zero, ou seja, todas as restrições foram atendidas completamente.

Na tabela 6.7, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R08, R09 e R10, que representam as restrições fracas do problema.

Tabela 6.7: Testes TOY_{12}.

nome: TOY_{12} instância: toy pesos: UD3														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	0	0	4	3	2	0	0	0	-
ótimo	0	0	0	0	0	0	0	0	-	-	-	-	-	0
melhor SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
média	0	0	0	0	0	0	0	0	0	0	0	0	0	0
desvio padrão	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Em todo os testes, a função objetivo atingiu o custo zero, ou seja, todas as restrições foram atendidas completamente. A utilização de pesos diferenciados para o teste TOY_{12} em relação ao teste TOY_{11} não influenciou a capacidade do método de encontrar a solução ótima.

Na tabela 6.8, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03, R04 e R05 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R06, R08, R10 e R13, que representam as restrições fracas do problema.

Tabela 6.8: Testes TOY_{13}.

nome: TOY_{13} instância: toy pesos: UD4														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	1	0	1	0	1	0	100	1	-
ótimo	0	0	0	0	0	0	0	0	-	-	-	-	-	0
melhor SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
média	0	0	0	0	0	0	0	0	0	0	0	0	0	0
desvio padrão	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Em todo os testes, a função objetivo atingiu o custo zero, ou seja, todas as restrições foram atendidas completamente. A utilização de pesos diferenciados para o teste TOY_{13} em relação aos testes TOY_{11} e TOY_{12} não influenciou a capacidade do método de encontrar a solução ótima.

Em ambos os testes o método foi capaz de encontrar o ótimo global, em média no décimo oitavo ciclo de iteração. A média e o desvio padrão foram calculados com base em cinco simulações, e o tempo médio de processamento de cada simulação foi de 1 minuto. A Figura 6.1, correspondente ao melhor resultado para o teste TOY_{11}, representa a evolução do custo da melhor solução encontrada pelo algoritmo implementado.

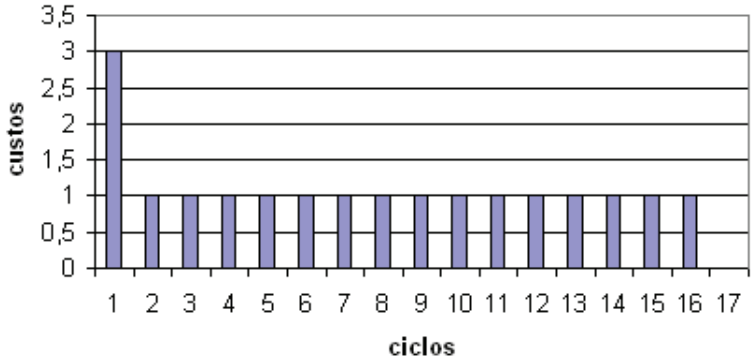

Figura 6.1: Evolução da melhor solução em $RefSet$.

Vale salientar que os testes realizados com a instância Comp01 da *International Timetabling Competition* apresentados na Seção 6.5.1 também são considerados testes de validação já que os resultados obtidos nestes testes podem ser comparados com os resultados obtidos na competição.

6.5 Testes de Comparação

Os testes de comparação foram realizados com duas instâncias, a primeira referente ao Estudo de caso 1, conforme apresentado na Seção 4.3.1, e a segunda referente ao Estudo de caso 2, apresentado na Seção 4.3.2.

6.5.1 Testes com a Instância de Competição

Para os testes foi utilizada a instância comp01 do banco de instâncias da *International Timetabling Competition*. Os resultados foram comparados com o melhor resultado apresentado no *ranking* da competição e disponível no sítio da ITC [4].

Para realização dos testes, foram fixados os valores para os parâmetros envolvidos na ferramenta de solução e apresentados na tabela 6.5.1. Estes valores foram definidos de acordo com os resultados dos testes de ajuste e sintonia de parâmetros descritos na Seção 6.3.

Foram realizados três testes com a instância comp01, conforme as definições de pesos das formulações UD2, UD3 e UD4 apresentadas na tabela 6.2.2. Os testes foram denominados ITC_{11}, ITC_{12} e ITC_{13}, respectivamente, e seus resultados são apresentados nas tabelas 6.10–6.12.

Os valores das médias e os respectivos desvios padrão, para cada caso de testes, foram calculados com base em cinco simulações, sendo o tempo médio de processamento de cada simulação de aproximadamente duas horas.

[4] `http://tabu.diegm.uniud.it/ctt/index.php?page=rankings&see=best`

Tabela 6.9: Parâmetros dos testes ITC.

Psize	$b1$	$b2$	ciclos de melhoria de P	ciclos de melhoria de *Pool*	máximo de vizinhanças	critério de parada (número de ciclos)
500	5	5	10	100	6	200

Tabela 6.10: Testes ITC_{11}.

nome: ITC_{11} instância: comp01 pesos: UD2														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	5	2	0	1	0	0	0	0	-
melhor ITC	0	0	0	0	4	0	0	-	1	-	-	-	-	5
melhor SS	0	0	0	0	31	0	3	24	12	2	35	21	13	49
média	0	0	0	0	37,6	0	6,2	25,2	16,8	6,8	50,4	22	29,6	66,8
desvio padrão	0	0	0	0	10,96	0	1,78	3,03	3,27	2,94	11,65	4,84	5,41	15,49

Na tabela 6.10, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R06, R07 e R09, que representam as restrições fracas do problema.

O custo total da solução é representado pelo somatório de custos das restrições fortes e fracas (como definido na função objetivo apresentada na Seção 4.1.5). As demais restrições, que recebem peso zero, não são consideradas no cálculo do custo total da solução, mas ainda assim são avaliadas e seus valores são apresentados na tabela.

A linha melhor ITC representa os valores de comparação, ou seja, o custo relativo a restrição (indicada pelas respectivas colunas) do melhor resultado para a formulação UD2 registrado na competição [5].

No teste de melhor resultado, a função objetivo atingiu o custo 49. A média de custo, considerando-se todos os testes foi de 66,8 com desvio padrão de 15,49. Vale salientar que, apesar do custo final devido à restrição R05 nesta solução ser 31, isto representa apenas 7 salas com alocação superior à capacidade permitida. Da mesma forma, o custo final 6 para a restrição R07 representa apenas 3 aulas isoladas. Portanto a restrição R09, correspondente a ocorrência de todas as aulas de uma disciplina na mesma sala, foi a restrição que apresentou maior número de violações, ou seja, 12.

Os resultados do teste ITC_{11} demonstram que o método foi capaz de melhorar 96,80%, em média, o custo da solução final em relação à solução original, com desvio padrão de 0,88%. Em média, a solução final é obtida no ciclo 125, com desvio padrão de 25,45 ciclos.

Na tabela 6.11, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do

[5] http://tabu.diegm.uniud.it/ctt/index.php?page=rankings&see=best

Tabela 6.11: Testes ITC_{12}.

nome: ITC_{12} instância: comp01 pesos: UD3														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	0	0	4	3	2	0	0	0	-
melhor ITC	0	0	0	0	4	-	-	0	0	4	-	-	-	8
melhor SS	0	0	0	0	29	16	3	4	13	5	56	29	18	94
média violações	0	0	0	0	40,4	12	5,8	5,2	14,4	6,4	43,2	22,6	22,6	117,2
desvio padrão	0	0	0	0	13,83	2,34	1,64	1,30	1,67	1,34	8,72	3,97	3,84	17,76

problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R08, R09 e R10, que representam as restrições fracas do problema.

A linha melhor ITC representa os valores de comparação, ou seja, o custo relativo a restrição (indicada pelas respectivas colunas) do melhor resultado para a formulação UD3 registrado na competição [6].

No teste de melhor resultado, a função objetivo atingiu o custo 94. A média de custo, considerando-se todos os testes foi de 117,2 com desvio padrão de 17,76.

Os resultados do teste ITC_{12} demonstram que o método foi capaz de melhorar 94,70%, em média, o custo da solução final em relação à solução original, com desvio padrão de 0,74%. Em média, a solução final é obtida no ciclo 145, com desvio padrão de 20,42 ciclos.

Tabela 6.12: Testes ITC_{13}.

nome: ITC_{13} instância: comp01 pesos: UD4														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	1	0	1	0	1	0	100	1	-
melhor ITC	0	0	0	0	4	1	-	0	-	1	-	0	0	6
melhor SS	0	0	0	0	52	2	34	14	14	7	48	0	14	83
média violações	0	0	0	0	55,6	2	34,4	14	14,6	7,2	49,8	0	16,8	95,6
desvio padrão	0	0	0	0	7,30	0,70	3,28	0,70	1,14	0,44	2,94	0	3,27	9,34

Na tabela 6.12, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03, R04 e R12 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R06, R08, R10 e R13, que representam as restrições fracas do problema.

[6] http://tabu.diegm.uniud.it/ctt/index.php?page=rankings&see=best

A linha melhor ITC representa os valores de comparação, ou seja, o custo relativo a restrição (indicada pelas respectivas colunas) do melhor resultado para a formulação UD4 registrado na competição [7].

No teste de melhor resultado, a função objetivo atingiu o custo 83. A média de custo, considerando-se todos os testes foi de 95,6 com desvio padrão de 9,34.

Os resultados do teste ITC_{13} demonstram que o método foi capaz de melhorar 95,69%, em média, o custo da solução final em relação à solução original, com desvio padrão de 0,66%. Em média, a solução final é obtida no ciclo 147, com desvio padrão de 22,12 ciclos.

Em todas as simulações, o método obtém melhoras consecutivas no custo das soluções, em média, até o ciclo 30. A partir deste ponto do processo de solução, as melhoras ocorrem, em média, a cada 8 ciclos, até que o ótimo local seja encontrado.

A Figura 6.2, correspondente ao teste de melhor resultado (ITC_{11}), apresenta esta evolução. O gráfico mostra uma melhora acentuada no custo das soluções até o ciclo 25. Do ciclo 25 ao 96, as melhoras ocorrem, na média, a cada 8 ciclos. A solução final é obtida no ciclo 96. O custo da solução final é 97,24% melhor do que a melhor solução inicial.

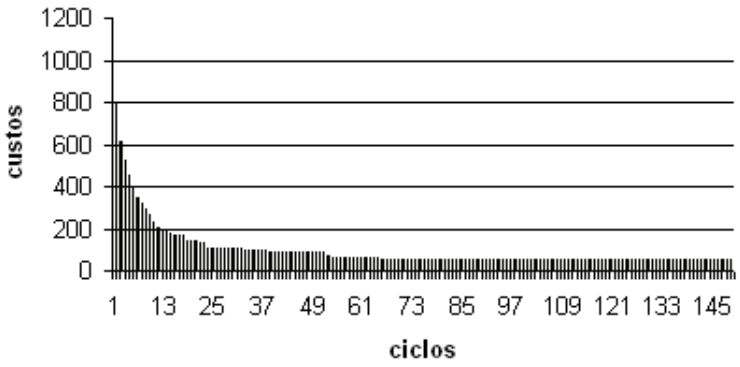

Figura 6.2: Evolução da melhor solução em $RefSet$.

A implementação do modelo e a sua execução sobre dados da *International Timetabling Competition* proporcionaram a comprovação de sua aplicabilidade, porém, em nenhuma das simulações, o método foi capaz de encontrar o ótimo global, ficando preso em soluções que representam ótimos locais.

6.6 Testes com a Instância do Estudo de Caso 2

Para a realização destes testes foi utilizada a instância real01 do Estudo de caso 2, sendo os valores para os parâmetros utilizados na ferramenta computacional apresentados na tabela 6.13. Estes valores foram definidos conforme resultados dos testes de ajuste de parâmetros realizados e apresentados anteriormente.

Foram realizados dois testes com a instância real01, sendo utilizadas diferentes definições de pesos, aqui denominadas $REAL_{11}$ e $REAL_{12}$. Os valores dos pesos para cada restrição, bem como os respectivos resultados, são apresentados nas tabelas 6.14 e 6.15.

[7] http://tabu.diegm.uniud.it/ctt/index.php?page=rankings&see=best

Tabela 6.13: Parâmetros dos testes *REAL*.

Psize	$b1$	$b2$	ciclos de melhoria de P	ciclos de melhoria de *Pool*	máximo de vizinhanças	critério de parada (número de ciclos)
500	5	5	10	100	6	200

Tabela 6.14: Testes $REAL_{11}$.

nome: $REAL_{11}$ instância: real01														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	1	1	1	1	1	1	1	1	-
objetivo	0	0	0	0	0	0	0	0	0	0	0	0	0	0
melhor SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
média violações	0	0	0	0	0	0	0	0	0	4,66	0	0	0	4,66
desvio padrão	0	0	0	0	0	0	0	0	0	5,31	0	0	0	5,31

Na tabela 6.14, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R06, R07, R08, R09 , R10 , R11, R12 e R13, que representam as restrições fracas do problema.

No teste de melhor resultado, a função objetivo atingiu o custo zero, ou seja, todas as restrições foram atendidas completamente. A média de custo, considerando-se todos os testes foi de 4,66 com desvio padrão de 5,31. Em todos os testes em que a solução de custo zero não foi obtida, a restrição R10 revelou-se a mais difícil de ser atendida.

Tabela 6.15: Testes $REAL_{12}$.

nome: $REAL_{12}$ instância: real01														
restrições	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	custo
pesos	100	100	100	100	1	1	1	1	1	100	1	1	2	-
objetivo	0	0	0	0	0	0	0	0	0	0	0	0	0	0
melhor SS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
média violações	0	0	0	0	0	0	0	0	0	333,33	0	0	0	333,33
desvio padrão	0	0	0	0	0	0	0	0	0	531,66	0	0	0	531,66

Na tabela 6.15, a linha pesos representa os pesos atribuídos às restrições consideradas no problema. As restrições R01, R02, R03 e R04 representam as restrições fortes do problema e devem ser atendidas obrigatoriamente. Portanto, recebem um peso significativamente superior aos atribuídos às restrições R05, R06, R07, R08, R09 , R10 , R11, R12 e R13, que representam as restrições fracas do problema. Um peso superior, idêntico ao de uma restrição forte foi atribuído à restrição R10, com maior dificuldade de satisfação.

No teste de melhor resultado, a função objetivo atingiu o custo zero, ou seja, todas as

restrições foram atendidas completamente. A média de custo, considerando-se todos os testes foi de 4,66 com desvio padrão de 5,31. Em todos os testes em que a solução de custo zero não foi obtida, a restrição R10 revelou-se a mais difícil de ser atendida.

A média e o desvio padrão foram calculados com base em cinco simulações e o tempo médio de processamento de cada simulação foi de 2 horas.

A Figura 6.3, correspondente ao teste de melhor resultado ($REAL_{11}$), apresenta a evolução de $RefSet$. O gráfico mostra uma melhora acentuada no custo das soluções até o ciclo 7. Do ciclo 7 ao 27 ocorre um período de estabilidade, onde soluções melhores não são encontradas. A solução final é obtida no ciclo 33.

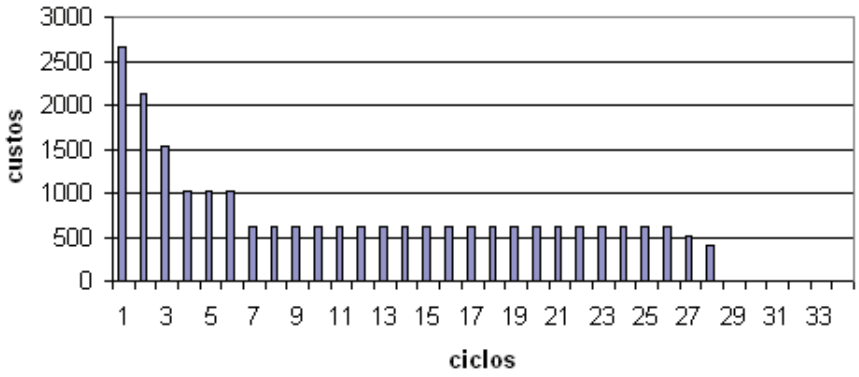

Figura 6.3: Evolução da melhor solução em $RefSet$.

Todos os testes com a instância de caso real apresentaram resultados satisfatórios. A implementação do modelo, e a sua execução sobre dados reais de cursos, proporcionaram a comprovação de sua aplicabilidade e eficiência na resolução do problema proposto. Os resultados finais alcançados atenderam a todas as restrições em um tempo computacional aceitável, isto é, um tempo inferior a duas horas.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou um modelo de solução para o problema de construção das grades de horário de cursos em instituições de ensino, conhecido no meio científico como Programação de Horário de Cursos Baseada em Currículos. Este problema consiste na construção das grades de horários que indicam em quais períodos semanais cada disciplina destes cursos deverá ocorrer, respeitando as especificações dos currículos e alocando professores e salas.

A técnica de solução utilizada é um algoritmo baseado na metaheurística Busca Dispersa (*Scatter Search*) que utiliza a heurística Reconexão por Caminhos (*Path Relinking*) como estratégia de intensificação de busca da solução ótima. Vale frisar, novamente, que não foram encontrados, até o momento, trabalhos que reportem a utilização desta técnica para solução deste tipo de problema, revelando assim a validade deste trabalho.

Para formulação do problema, foram considerados dois estudos de caso: o primeiro, referente ao problema de programação de cursos pós matrícula, abordado na *International Timetabling Competition*, e o segundo, referente a problemas encontrados no cenário de uma instituição de ensino brasileira, do tipo universidade.

O objetivo deste trabalho foi registrar o desempenho desta técnica na solução de instâncias de problemas encontradas nos dois estudos de caso, e com isso avaliar o desempenho da metodologia de solução adotada, baseada na metaheurística Busca Dispersa.

Uma das principais dificuldades encontrada no desenvolvimento desta pesquisa foi a falta de referências na literatura para a implementação dos métodos Busca Dispersa e Reconexão por Caminhos para o problema Programação de Horário Cursos Baseada em Currículos. Este autor encontrou apenas um trabalho atestando a aplicação da metaheurística Busca Dispersa para o subproblema da classe de Problemas de Horários denominado Problema de Programação de Exames [36]. Portanto, o registro de implementação desta técnica de solução é uma das contribuições deste trabalho.

A formulação das restrições, apresentada no capítulo 4, é aplicável ao problema de Programação Horário de Cursos Baseada em Currículos da *International Timetabling Competition* e este autor não encontrou nenhum trabalho que apresentasse a formulação matemática completa das restrições consideradas neste problema. Registra-se neste fato outra contribuição desta dissertação.

A escolha das instâncias de problemas propostos na *International Timetabling Competition* deveu-se à necessidade de comparação dos resultados obtidos e, consequentemente, validação da eficácia do método para o tipo de problema abordado. A implementação do modelo e sua execução sobre dados da *International Timetabling Competition* proporcionaram a comprovação de sua aplicabilidade, mesmo que com resultados inferiores aos registrados anteriormente por outros métodos de solução.

A justificativa para este comportamento consiste em que o método, na forma em que

foi implementado, apresenta uma grande dificuldade para escapar de regiões representando ótimos locais. Por exemplo, nos testes com a instância comp01 segundo o modelo UD2, a maior dificuldade deu-se no atendimento à restrição capacidade de salas, já que esta instância possui um número bastante limitado de soluções possíveis capazes de satisfazer esta restrição.

Dada esta colocação, outra dificuldade foi a validação da eficiência de estruturas de vizinhança para o problema abordado, já que as referências bibliográficas citadas nesta dissertação tratam de problemas semelhantes, mas que não incluem todas as restrições previstas na formulação do problema abordado. As estruturas de vizinhança utilizadas na aplicação foram oriundas da combinação de estruturas estudadas, mas projetadas especificamente para atender ao conjunto de restrições previstas na formulação apresentada nesta dissertação. Um estudo mais aprofundado da relação de eficiência de cada uma destas estruturas, com sua capacidade de resolver violações a cada tipo de restrição, será considerado como trabalho futuro deste autor a fim de melhorar o desempenho da técnica implementada.

Da mesma forma, a implementação dos métodos de combinação de soluções, utilizados na técnica Busca Dispersa, foi realizada especificamente para o problema formulado, já que estratégias de combinação mais comuns como, por exemplo, cortes na matriz de solução, poderiam inviabilizar soluções já viáveis.

Outra consideração importante refere-se ao tempo computacional elevado que demanda a aplicação da técnica utilizada. Apesar de aceitável para a aplicação proposta, (os testes com todas as instâncias da *Internationa Timetabling Competition* foram concluídos em um tempo inferior a duas horas), pequenas modificações de parâmetros, como por exemplo o aumento do tamanho do conjunto de referência, incrementam significativamente o tempo de execução da aplicação. Um dos fatores que contribui para este fato é a utilização da técnica Reconexão por Caminhos como método de combinação, que apesar de demonstrar-se mais eficiente para obtenção de soluções melhores, exige um grande esforço computacional e conseqüente aumento do tempo de processamento.

Já a implementação do modelo e sua execução sobre dados da universidade local, proporcionaram a comprovação de sua aplicabilidade e eficiência na resolução do problema. As grades de horário geradas atenderam a todas as restrições consideradas gerando grades de horário de boa qualidade.

7.1 Considerações e trabalhos futuros

Embora tenha sido gerada uma versão funcional da ferramenta de otimização baseada na metaheurística Busca Dispersa, a sua forma atual apresenta algumas restrições em relação ao funcionamento, a seguir individuadas:

- o tempo de processamento pode ser reduzido com a escolha de outro ambiente de programação, como C++, por exemplo. A máquina virtual Java apresenta restrições quanto ao uso de memória em ambientes 32 bits;

- a dificuldade em superar regiões que representam ótimos locais pode estar relacionada a escolha das estruturas de vizinhança utilizadas durante o método de melhoria. Em um trabalho futuro, pretende-se verificar a eficiência de cada uma destas estruturas de vizinhança, para que seja possível o emprego de um mecanismo estocástico que guie a escolha de cada uma delas de acordo com a sua eficiência em resolver violações de restrições consideradas difíceis;

- explorar, de diferentes formas, a influência do uso de níveis de vizinhança nos resultados do método de melhoria, ou seja, testar vizinhanças de forma única ou conjuntamente, visando a melhora de desempenho do método.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] COOPER, T. B.; KINGSTON, J. H. The complexity of timetable construction problems. *In Burke and Ross*, Springer-Verlag, p. 283–295, 1995.

[2] WREN, A. Scheduling, timetabling and rostering - a special relationship? *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, Springer-Verlag, p. 46–75, 1996.

[3] CORDENONSI, A. Z. *Ambientes, objetos e dialogicidade : uma estratégia de ensino superior em heurísticas e metaheurísticas*. Dissertação (Mestrado) — Curso de Doutorado em Informática na Educação da Universidade Federal do Rio Grande do Sul, 2008.

[4] GLOVER, F. et al. Scatter search. Springer-Verlag Inc., New York, NY, USA, p. 519–537, 2003.

[5] GLOVER, F. Scatter search and path relinking. McGraw-Hill Ltd., Maidenhead, UK, England, p. 297–316, 1999.

[6] GLOVER, F. A template for scatter search and path relinking. In: *Artificial Evolution '97: Selected Papers from the Third European Conference on Artificial Evolution*. London, UK: Springer-Verlag, 1998. p. 13–54. ISBN 3-540-64169-6.

[7] KIRKPATRICK, S. et al. Optimization by simulated annealing. *Science*, v. 220, p. 671–680, 1983.

[8] HOLLAND, J. Genetic algorithms. *Scientific American*, v. 267, p. 44–50, 1992.

[9] PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982.

[10] GAREY, M.; JOHNSON, D. *Computers and intractability : a guide to the theory of NP-completeness*. New York: W.H. Freeman and Company, 1999. 338 p.

[11] SANTOS, H.; SOUZA, M. Programação de horários em instituições educacionais: Formulações e algoritmos. In: *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO, Fortaleza*. [S.l.: s.n.], 2007. p. 2827–2882.

[12] WERRA, D. de. An introduction to timetabling. *European Journal of Operational Research*, v. 19, p. 151–162, 1985.

[13] SCHAERF, A. A survey of automated timetabling. *Artif. Intell. Rev.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 13, n. 2, p. 87–127, 1999. ISSN 0269-2821.

[14] JUNGINGER, W. Timetabling in germany - a survey. *Interfaces*, v. 16, p. 66–74, 1986.

[15] BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 2003.

[16] SOUZA, M. et al. Métodos de pesquisa em vizinhança variável aplicados ao problema de alocação de salas. In: *XXII Encontro Nacional de Engenharia de Produção - ENEGEP, Outubro 2002, Curitiba, Brasil*. [S.l.: s.n.], 2002.

[17] GASPERO, L. D.; SCHAERF, A. Neighborhood portfolio approach for local search applied to timetablingproblems. *Journal of Mathematical Modeling and Algorithms*, Springer Science+Business Media, v. 5, n. 1, p. 65–89, 2006.

[18] LÜ, Z. et al. *Neighborhood analysis: A case study on curriculum-based course timetabling*. Angers, France, 2009.

[19] COELHO, A. M. *Uma Abordagem Via Algoritmos Meméticos para a Solução do Problema de Horário Escolar*. Dissertação (Mestrado) — Curso de Mestrado em Modelagem Matemática e Computacional do Centro federal de Educação Técnológica de Minas Gerais, 2006.

[20] GOTLIEB, C. C. The construction of class-teacher timetables. In: *IFIP congress 62*. North-Holland: [s.n.], 1962. p. 73–77.

[21] SANTOS, H.; SOUZA, M. Programação de horários em instituições educacionais: Formulações e algoritmos. In: *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO, Fortaleza*. [S.l.: s.n.], 2007. p. 2827–2882.

[22] BAI, R. et al. A simulated annealing hyper-heuristic for university course timetabling. In: *Anais Fortaleza: ABEPRO, 2006*. [S.l.: s.n.], 2006.

[23] NANDHINI, M.; KANMANI, S. A survey of simulated annealing methodology for university course timetabling. 2009.

[24] SUBRAMANIAN, A. et al. Aplicação da metaheurística busca tabu na resolução do problema de alocação de salas do centro de tecnologia da UFPB. In: *XXVI ENEGEP - Fortaleza, CE, Brasil, 9 a 11 de Outubro de 2006*. [S.l.: s.n.], 2006.

[25] SOUSA, V. N. de et al. Programação da grade de horário em escolas de ensino fundamental e médio. *Pesquisa Operacional*, 2008.

[26] GÓES, A. R. T. *Otimização Na Distribuição da Carga Horária de Professores Método Exato, Método Heurístico, Método Misto e Interface*. Dissertação (Mestrado) — Programa de Pós-Graduação em Métodos Numéricos em Engenharia – Programação Matemática, Setores de Tecnologia e de Ciências Exatas da Universidade Federal do Paraná, 2005.

[27] MASSOODIAN, S.; ESTEKI, A. A hybrid genetic algorithm for curriculum based course timetabling. In: *PATAT 2007*. [S.l.: s.n.], 2007.

[28] LOBO, E. L. M. *Uma Solução Do Problema de Horário Escolar Via Algoritmo Genético Paralelo*. Dissertação (Mestrado) — Curso de Mestrado em Modelagem Matemática e Computacional do CEFET-MG, 2005.

[29] SOUZA, M. et al. Experiências com a utilização de simulated annealing e busca tabu na resolução do problema de alocação de salas. In: *XXXIV Simpósio Brasileiro de Pesquisa Operacional, 2002, Rio de Janeiro*. [S.l.: s.n.], 2002. p. 1100–1110.

[30] RESENDE, M. G. C.; RIBEIRO, C. C. *Greedy Randomized Adaptive Search Procedures*. Florham Park, USA, August 2002. 29 p.

[31] RAHOUAL, M.; SAAD, R. Solving timetabling problems by hybridizing genetic algorithms and tabu search. In: BURKE, E.; RUDOVÁ, H. (Ed.). *VI International Conference on the Practice and Theory of Automated Timetabling - PATAT*. Brno, Czech Republic, 2006. p. 467–472.

[32] FRAUSTO-SOLÍS, J.; ALONSO-PECINA, F. A hybrid simulated annealing-tabu search algorithm for post enrolment course timetabling. In: *PATAT 2007*. [S.l.: s.n.], 2007.

[33] BELFIORE, P. P.; YOSHIZAKI, H. T. Y. Scatter search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. *Pesquisa Operacional*, 2006.

[34] BELFIORE, P. P. et al. Scatter search para programação de tarefas em uma única máquina com penalidades de adiantamento e atraso e data de entrega comum. In: . [S.l.: s.n.], 2006.

[35] LORENZONI, L. L. et al. *Scatter Search* aplicado ao problema de otimização da alocação de sondas de produção em poços de petróleo. In: . [S.l.: s.n.], 2007.

[36] MANSOUR, N. et al. Scatter search technique for exam timetabling. In: . [S.l.]: Springer, 2009. p. 13–54.

[37] MOURA, A. et al. Técnicas metaheurísticas aplicadas à construção de grades horárias escolares. In: *Anais: XXXVI SBPO*. [S.l.: s.n.], 2004.

[38] SOUZA, M. J. F. et al. Método de pesquisa em vizinhança variável aplicado ao problema de alocação de salas. In: *Anais do XXII ENEGEP*. [S.l.: s.n.], 2002.

[39] MARCONDES, W. *Desenvolvimento e Aplicação de Algoritmos Heurísticos ao Problema de Alocação de Espaço Físico em Universidade*. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, 2008.

[40] ALADAG, C. H.; HOCAOGLU, G. A tabu search algorithm to solve a course timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, v. 36, n. 1, p. 53 – 64, 2007.

[41] GASPERO, L. D.; SCHAERF, A. Multi-neighbourhood local search with application to course timetabling. In: *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2002), number 2740 in Lecture Notes in Computer Science*. [S.l.]: Springer-Verlag, 2003. p. 262–275.

[42] AVELLA, P.; VASIL'EV, I. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Heuristics*, Springer Netherlands, v. 8, n. 6, p. 497–514, 2005. ISSN 1099-1425.

[43] ALVAREZ-VALDES, R. et al. Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, Elsevier, v. 137, n. 3, p. 512–523, 2002.

[44] ABRAMOWITZ, M.; STEGUN, I. A. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. New York, USA: US. Nat. Bureau Stand., 1964. (Dover books on intermediate advanced mathematics).

[45] CESCO, F. et al. *Benchmarking Curriculum-Based Course Timetabling: Formulations, Data Formats, Instances, Validation, and Results*. Undine, Italy, 2009.

[46] MCCOLLUM, B. et al. *The Second International Timetabling Competition ITC – 2007 Curriculum-based Course Timetabling Track*3. Undine, Italy, 2007.