PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS: UMA APROXIMAÇÃO POR METAHEURÍSTICAS

Marcone Jamilson Freitas Souza

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. Adilson Elias Xavier, D.Sc.

Prof. Dario José Aloise, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

Prof. Victor Manuel Parada Daza, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2000

SOUZA, MARCONE JAMILSON FREITAS

Programação de horários em escolas: uma aproximação por metaheurísticas [Rio de Janeiro] 2000

XI, 149 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2000)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Programação de horários em escolas.

2. Metaheurísticas.

3. Otimização Combinatória.

I. COPPE/UFRJ II. Título (série).

A Cássia,

Marcus Vinícius e

Pedro Henrique.

Agradecimentos

As pessoas à sua volta são espelhos
que refletem seus pensamentos.

Esses últimos quatro anos foram particularmente penosos para mim. Se, por um lado, eu tinha todos os recursos disponíveis para desenvolver meu trabalho; por outro, eu não contava com a presença diária de minha família. Fiquei distante dos meus filhos, um dos quais nascido no início do doutoramento. Felizmente, encontrei na UFRJ um ambiente acolhedor e tive o privilégio de conhecer pessoas formidáveis, que muito contribuíram para a realização deste trabalho.

Ao Prof. Nelson Maculan, ser extraordinário, de qualidades ímpares, referência para mim tanto como ser humano quanto como pesquisador, pela orientação recebida e pelo apoio, meu sincero reconhecimento.

Ao sempre disponível Prof. Luiz Satoru Ochi, que estimulou e apoiou meu ingresso ao doutorado, pela amizade e pela orientação recebida durante a execução deste trabalho.

À minha esposa, Cássia, meu suporte, que à distância esteve sempre presente, que soube com sabedoria substituir-me na educação de nossos filhos, minha gratidão eterna.

Aos meus filhos, Marcus Vinícius e Pedro Henrique, que pagaram um preço alto pela minha ausência, mas que souberam, apesar da curta idade, entender o sacrifício.

À minha mãe (em sua memória), que nunca mediu esforços para educar-me e que sonhava um dia ver-me graduado.

Ao amigo Lucídio dos Anjos Formiga Cabral, que compartilhou comigo as ansiedades do dia-a-dia de meu trabalho, que deu-me apoio e foi a minha família aqui no Rio, meu eterno agradecimento.

Aos colegas Marcos de Mendonça Passini, Douglas Valiati e Henrique Lima, pela iniciação ao mundo Linux e pela presteza que lhes são peculiares.

Ao Programa de Engenharia de Sistemas e Computação da COPPE, por ter-me dado acesso irrestrito a todos os seus recursos.

À CAPES e ao Departamento de Computação da Universidade Federal de Ouro Preto, pelo investimento em minha formação.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

**PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS:
UMA APROXIMAÇÃO POR METAHEURÍSTICAS**

Marcone Jamilson Freitas Souza

Dezembro/2000

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Este trabalho trata do problema de programação de horários em escolas. Suas características e propriedades básicas são apresentadas. Alguns dos principais algoritmos da literatura são descritos. É desenvolvida uma heurística de busca local, baseada em caminhos mínimos, para melhorar um quadro de horário, ainda que com um certo tipo de inviabilidade. Essa heurística age primeiramente tentando recuperar a viabilidade e, em um segundo momento, se bem sucedida, tentando melhorar os requisitos de qualidade exigidos para o quadro de horário. Várias metaheurísticas conduzindo essa técnica em determinadas fases são desenvolvidas e testadas. É proposta, também, uma modelagem de programação matemática para o problema. Resultados computacionais são apresentados para um conjunto de problemas-teste reais.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SCHOOL TIMETABLING:
AN APPROXIMATION BY METAHEURISTICS

Marcone Jamilson Freitas Souza

December/2000

Advisor: Nelson Maculan Filho

Department: Computing and Systems Engineering

The focus of this work is the school timetabling problem. Its basic characteristics and properties are presented. Some of the main algorithms from the literature are described. An heuristic approach, based on shortest paths, to improve the timetable, even infeasible, is proposed. First, this heuristic try to retrieve the feasibility. If it is successful it will be activated again, now in order to try to improve the compactness of the timetable, as well as other quality requirements. Several metaheuristics using this technique in some phases are developed and tested. A mathematical programming model is proposed too. Computational results are presented for a set of real problems.

Conteúdo

Capítulo 1

Preliminares

Este trabalho está organizado como segue. Neste capítulo introduzimos o problema de programação de horários, classificamo-lo e levantamos as principais questões relativas à sua automação. Apontamos sua intratabilidade, justificando a abordagem por técnicas heurísticas.

No capítulo 2 fazemos um breve descrição das principais técnicas heurísticas referenciadas ao longo do trabalho.

No capítulo 3 fazemos um levantamento do estado da arte com relação ao problema de programação de horários em escolas, detalhando algumas das técnicas apresentadas na literatura.

No capítulo 4 consideramos uma escola brasileira de ensino médio típica e discriminamos suas características para referências futuras.

No capítulo 5 desenvolvemos uma formulação de programação matemática para o problema abordado, de forma a testar a eficiência das heurísticas em problemas de pequeno porte.

No capítulo 6 mostramos como adaptar as metaheurísticas apresentadas no capítulo 2 para tratar do problema de horários em escolas considerado no capítulo 4. Propomos, também, um procedimento para recuperar a viabilidade e melhorar requisitos de qualidade de um quadro de horário, mostrando como inserir esse procedimento nas metaheurísticas.

No capítulo 7 apresentamos os resultados computacionais, comparando o desempenho dos procedimentos propostos.

No capítulo 8 concluímos o trabalho e apontamos algumas idéias para trabalhos futuros.

1.1 Introdução

O problema de programação de horários em uma instituição de ensino, ou simplesmente, o problema de horários, consiste em fixar uma seqüência de encontros entre professores e estudantes em um período prefixado de tempo, em geral uma semana, satisfazendo a um conjunto de restrições de vários tipos.

A solução manual desse problema é uma tarefa árdua e normalmente requer vários dias de trabalho. A elaboração de um quadro de horário por esta via pode demandar duas semanas em uma escola secundária [8, 72] ou até um mês, em uma universidade [19]. Além do mais, a solução obtida pode ser insatisfatória com respeito a diversos aspectos. Por exemplo, um estudante pode ficar impedido de matricular-se em dois cursos que ele gostaria de cursar, se eles forem marcados para um mesmo horário; ou um professor pode ficar descontente se houver muitos *buracos* em sua programação semanal de ensino.

Por estas razões, uma atenção especial vem sendo dedicada à automação deste problema. Os primeiros esforços neste sentido foram feitos na década de 60, com os trabalhos precursores de Csima e Gotlieb [30] e Gotlieb [60]. Desde então, mais de 700 publicações sobre o assunto foram escritas até o ano de 1995, segundo Bardadym [8]. Em 1995 realizou-se a primeira conferência internacional sobre o tema (*Practice and Theory on Automated Timetabling - PATAT*). Duas outras conferências já ocorreram depois desta data. A segunda, no ano de 1997 e a terceira, no ano 2000.

Vale ressaltar, entretanto, que muitos autores, entre os quais Schaefer [94] e de Werra [34], acreditam que problemas de horários não podem ser completamente automatizados. Há duas justificativas para isto: por um lado, há razões, que não podem ser facilmente expressas em um sistema automatizado, que tornam um quadro de horário melhor que outro. Por outro, uma vez que o espaço de soluções é vasto, a intervenção humana pode conduzir a busca em direção a regiões promissoras, nas quais o sistema, por si só, dificilmente teria condições de chegar.

Em virtude disto, a maioria dos sistemas existentes permitem ao usuário pelo menos ajustar manualmente a solução final. Alguns sistemas, no entanto, requerem uma intervenção humana maior, os chamados 'sistemas interativos (ou semi-automáticos) de programação de horários'. Atualmente, há uma tendência para desenvolver sistemas explorando essa característica [8], porque essa técnica une a experiência do especialista na construção de quadros de horário com o poder dos procedimentos de busca local mais sofisticados, tais como Busca Tabu, *Simulated Annealing* etc [36].

Por outro lado, é consenso na comunidade científica que o problema de programação de horários é de difícil generalização. Isto se deve à diversidade de regimes educacionais, os quais variam de região para região, e às características de cada instituição de ensino. Desta forma, os sistemas são comumente desenvolvidos para atender a uma instituição específica. Por este motivo ainda não existe um conjunto de problemas-teste que possa ser usado para avaliar os algoritmos desenvolvidos. No entanto, um esforço nesse sentido está sendo feito pelo grupo WATT (*WorkGroup on Automated TimeTabling*, http://www.asap.cs.nott.ac.uk/asap/watt), de forma a se ter um banco de problemas-teste que inclua, pelo menos, as restrições mais comuns às diversas instituições. Até a presente data, existem de nosso conhecimento apenas dois conjuntos de problemas-teste, todos eles referentes a problemas de programação de horários de exames (vide classificação dos problemas de horário na seção 1.2).

Um levantamento interessante sobre questões relacionadas ao processo de construção de quadros de horário em escolas secundárias é feita por Kwok et al. [72]. Os autores estudaram o problema relativo às escolas de Hong Kong no final do ano de 1995. Segundo eles, cerca de 80% das instituições usam computadores para ajudar a tarefa de alocação. Entretanto, apenas um terço delas usam programas específicos para gerar quadros de horário. Nas demais escolas a alocação é manual e o computador é usado apenas para a verificação da existência de conflito, isto é, para verificar, por exemplo, se em um determinado horário existe alguma turma tendo aula com mais de um professor. Tais programas, no entanto, completam somente dois terços da alocação, em média, deixando o terço remanescente por conta dos elaboradores de quadros de horário das instituições, os quais têm, ainda, que inserir algumas outras restrições não abordadas pelos programas.

Acreditamos que a realidade das instituições de ensino brasileiras não seja muito diferente e que entre aquelas que dispõe de sistemas automatizados, a maioria procure apenas uma solução viável.

1.2 Classificação dos problemas de horários

Schaerf [94] classifica os problemas de programação de horários em 3 categorias principais, de acordo com o tipo de instituição (universidade ou escola) e restrições que aparecem:

Problema de programação de horários em escolas (***School timetabling problem***) Este problema, que será denotado por **PHE**, diz respeito à alocação das aulas de uma instituição com as características de uma escola secundária típica. Basicamente, existe um conjunto de turmas, um conjunto de professores e um conjunto de horários reservados para a realização das aulas. As turmas são conjuntos disjuntos de estudantes que têm um mesmo currículo. Para cada turma há um conjunto de matérias, com suas respectivas cargas horárias, que devem ser cursadas. Para cada professor especifica-se a matéria (ou as matérias), bem como as turmas para as quais o professor lecionará. O objetivo básico é fazer um quadro de horário, em geral semanal, de tal forma que: 1) As cargas horárias de todas as matérias de todas as turmas sejam cumpridas; 2) Cada turma não tenha aula com mais de um professor ao mesmo tempo e 3) Um professor não dê aula para mais de uma turma em um mesmo horário. Uma característica importante dos problemas desta categoria diz respeito à rigidez dos horários que são disponibilizados para a realização das aulas. Em geral, as aulas são realizadas somente em um dos turnos do dia ou, quando o número de horários de um turno não suportar o número de aulas exigidos pelo currículo, disponibiliza-se um turno e mais uma parte de outro turno suficiente para atender ao número requerido. Além do mais, as aulas têm, normalmente, a mesma duração e, eventualmente, é desejável que algumas delas tenham durações diferentes. Outra característica adicional desses problemas é que, como as turmas são conjuntos disjuntos de estudantes, elas recebem suas aulas em uma mesma sala (exceto para as aulas de matérias que exigem salas especializadas). Desta forma, são os professores que se deslocam para lecionar para cada turma.

Problema de programação de horários de cursos (***Course timetabling problem***) Este problema diz respeito à alocação das aulas de uma instituição com as características de uma universidade típica. Basicamente, há um conjunto de cursos (Cálculo I, Cálculo II, Anatomia etc.) e, para cada curso, um certo número de aulas. Há, também, um conjunto de currículos (Engenharia Civil, Engenharia de Produção, Medicina etc.). Cada currículo envolve um conjunto de cursos. Os estudantes matriculam-se em turmas dos cursos de seu currículo. Uma turma de um curso pode ter estudantes de currículos diferentes. Há, também, um conjunto de horários disponibilizados para a realização das aulas e, em cada horário, um número limitado de salas. O problema consiste em alocar as aulas dos cursos aos horários disponibilizados, respeitando as disponibilidades e capacidades das salas existentes, de forma que nenhum estudante tenha duas ou mais aulas simultaneamente. Uma característica dos problemas desta categoria é que, regra geral, ao contrário do problema de horários em escolas, há uma maior flexibilidade com relação aos horários disponibilizados para a realização das aulas. Isto é, a princípio, um curso pode ser alocado a qualquer horário de funcionamento da instituição, o que, em geral, inclui os períodos da manhã, da tarde e da noite. Outra característica diz respeito à configuração das aulas dos cursos. Diferentemente do PHE, as aulas são agrupadas, de uma forma rígida, em sessões. Por exemplo, o curso de Cálculo I, de 5 horas-aula semanais, pode ser ministrado em duas sessões, uma de 3 horas-aula e outra de 2 horas-aula. Como o conceito de turma no problema de programação de horários de cursos é diferente do de escolas, em geral são os estudantes que se deslocam para terem suas aulas.

Problema de programação de horários de exames (*Examination timetabling problem*) Este problema diz respeito à alocação dos exames de uma instituição com as características de uma universidade típica. Há um conjunto de estudantes (os quais estão matriculados em cursos), um conjunto de exames para cada estudante e um conjunto de horários disponibilizados para a realização dos exames. O objetivo primário é alocar cada exame a um horário, de forma que nenhum estudante tenha que fazer dois ou mais exames simultaneamente. Apesar de similar ao problema de programação de horários de cursos, eles se distinguem sobretudo pela natureza das restrições envolvidas. Entre as restrições típicas que aparecem nesta categoria de problemas, destacamos:

nenhum estudante pode fazer mais do que um certo número de exames por dia, exames de certos cursos não podem preceder a exames de determinados outros cursos, alguns exames têm que ser realizados em um mesmo horário, certos exames devem ser realizados em horários consecutivos etc.

Esta classificação, contudo, não é absoluta, no sentido de que existem problemas que não se enquadram de forma precisa nestas categorias. Por exemplo, existem escolas secundárias européias que dão liberdade aos alunos para escolherem, nos dois últimos anos, as matérias a serem cursadas. Neste caso, o problema assemelha-se ao da programação de horários de cursos.

Uma classificação mais abrangente e independente do tipo de instituição é dada por Bardadym [8]. No entanto, a classificação apresentada anteriormente é a mais comumente utilizada na literatura.

O problema de programação de horários em escolas é referenciado, muitas vezes, como o problema professor-turma (*class-teacher*), enquanto o de horários de cursos como o da alocação de cursos (*course scheduling*) ou como o de programação de horários em universidades (*university timetabling*).

1.3 Viabilidade x Otimalidade

Cada problema de programação de horários pode ser, por sua vez, classificado em dois tipos, dependendo de necessitarem otimizar ou não uma função objetivo:

(a) Problema de viabilidade: Neste tipo enquadram-se os problemas nos quais se requer encontrar apenas um quadro de horário, dito *viável*, que satisfaça a todas as restrições impostas.

(b) Problema de otimização: Neste tipo enquadram-se os problemas nos quais se requer encontrar, dentre todos os quadros de horário que satisfazem a um certo conjunto de restrições ditas *essenciais*, aquele(s), chamado(s) *ótimo(s)*, que minimize(m) uma certa função objetivo, a qual incorpora as restrições ditas *não-essenciais*. Como será mostrado no capítulo 3, muitos autores adotam esta formulação para aplicar técnicas de otimização ao problema de viabilidade, uma vez que, neste caso, o problema de encontrar uma solução viável pode ser considerado como um problema de minimizar uma certa *distância de viabilidade*.

1.4 Complexidade

O problema de programação de horários é NP-completo [42], em sua versão de decisão. A intratabilidade do problema pode ser mostrada através da sua redução a um problema de coloração em grafos.

Cooper e Kingston [27] demonstram a NP-completude para uma série de problemas de horários que aparecem comumente na prática. Especificamente, eles mostram que a NP-completude aparece quando estudantes podem escolher suas matérias, quando as aulas têm durações diferentes ou quando há restrições para a escolha dos horários das aulas, como por exemplo, exigir que determinadas aulas sejam ministradas em horários consecutivos ou espalhadas eqüanimamente ao longo da semana.

Entretanto, algumas variantes do problema podem ser resolvidas em tempo polinomial. Este é o caso do problema de programação de horários em escolas, em sua versão básica, quando turmas e professores estão sempre disponíveis. Even et al. [42] apresentam uma solução para este caso baseada em fluxo em grafo. Tal método de solução está descrito na seção 3.2. Outras variantes clássicas desse problema para os quais existem algoritmos polinomiais são as seguintes: 1) Existe uma única turma e todos os professores têm um número arbitrário de horários indisponíveis [34]; 2) Apenas uma turma tem horários indisponíveis, as demais turmas e professores estão sempre disponíveis [37]; 3) Cada professor tem, no máximo, 2 horários disponíveis e as turmas estão sempre disponíveis [42, 34].

Algumas condições necessárias e suficientes para a existência de uma solução viável para o problema básico de programação de horários em escolas, incluindo restrições de indisponibilidade de professores e turmas, são estabelecidas por de Werra [36, 37]. O autor apresenta, também, algumas variantes dessa formulação que podem ser resolvidas em tempo polinomial.

A maioria dos casos resolvíveis são, no entanto, muito especiais e não incluem as restrições mais comuns que aparecem em problemas reais.

Em virtude do exposto, justifica-se a abordagem do problema de programação de horários por técnicas heurísticas, as quais, contudo, não garantem a existência de uma solução viável – mesmo que essa exista – e tampouco sua otimalidade.

Capítulo 2

Técnicas Heurísticas de Otimização

Este capítulo tem como objetivo apresentar, de forma sumária, as principais técnicas heurísticas referenciadas ao longo deste trabalho.

2.1 Métodos de Busca Local

As técnicas de busca local em problemas de otimização constituem uma família de técnicas baseadas na noção de vizinhança. Mais especificamente, seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar. A função N, a qual depende da estrutura do problema tratado, associa a cada solução viável $s \in S$, sua vizinhança $N(S) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s. Denomina-se *movimento* a modificação m que transforma uma solução s em outra, s', que esteja em sua vizinhança. Representa-se essa operação por $s' \leftarrow s \oplus m$.

Em linhas gerais, uma técnica de busca local, começando de uma solução inicial s_0 (a qual pode ser obtida por alguma outra técnica ou gerada de forma aleatória), navega pelo espaço de pesquisa, passando, iterativamente, de uma solução para outra que seja sua vizinha.

2.1.1 Método de Descida

É um método de busca local que se caracteriza por analisar <u>todos</u> os possíveis vizinhos de uma solução s em sua vizinhança $N(s)$, escolhendo, a cada passo, aquele que tem o menor valor para a função objetivo. Nesse método, o vizinho candidato somente é aceito se ele melhorar <u>estritamente</u> o valor da melhor solução até então obtida. Dessa forma, o método pára tão logo um mínimo local seja encontrado.

2.1.2 Método Randômico de Descida

O método de descida requer a exploração de toda a vizinhança. Um método alternativo, que evita essa pesquisa exaustiva é o método randômico de descida. Ele consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente; não o sendo, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é interrompido após um número fixo de iterações sem melhora no valor da melhor solução obtida até então.

Entretanto, o método randômico de descida também fica preso no primeiro mínimo local encontrado.

2.1.3 Método Randômico Não Ascendente (RNA)

O método randômico não ascendente (RNA) é uma variante do método randômico de descida, diferindo dele por aceitar o vizinho gerado aleatoriamente se ele for melhor ou igual à solução corrente. Esse método pára, também, após um número fixo de iterações sem melhora no valor da melhor solução produzida.

Por esse método, entretanto, é possível navegar pelo espaço de pesquisa por *movimentos laterais* [96]. Assim, ele tem condições de percorrer caminhos de descida que passam por regiões planas. Ou seja, se a pesquisa chega em um região dessas, o método tem condições de mover-se nela e sair através de uma solução diferente daquela que a ela chegou.

O método RNA é, portanto, um procedimento que executa a pesquisa no espaço de soluções combinando movimentos de descida com movimentos laterais.

2.2 Metaheurísticas

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico [91].

Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e têm condições de escapar de ótimos locais.

As metaheurísticas, assim como os métodos de busca local tradicionais, diferenciam-se entre si basicamente pelas seguintes características:

a) critério de escolha de uma solução inicial;

b) definição da vizinhança $N(s)$ de uma solução s;

c) critério de seleção de uma solução vizinha dentro de $N(s)$;

d) critério de término;

Apresentamos, a seguir, as principais metaheurísticas referenciadas ao longo deste trabalho.

2.2.1 GRASP (*Greedy Randomized Adaptive Search Procedure*)

GRASP (Procedimento de busca adaptativa gulosa e randomizada) é um método iterativo, proposto em [46], que consiste de duas fases: uma fase de construção, na qual uma solução é gerada, elemento a elemento, e de uma fase de busca local, na qual um ótimo local na vizinhança da solução construída é pesquisado. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado. O pseudo-código descrito pela figura 2.1 ilustra um procedimento GRASP.

procedimento $GRASP(f(.), g(.), N(.), GRASPmax, s)$
$f^\star \leftarrow \infty$;
para $(Iter = 1, 2, \ldots, GRASPmax)$ faça
    $Construcao(g(.), \alpha, s)$;
    $BuscaLocal(f(.), N(.), s)$;
    se $(f(s) < f^\star)$ então
        $s^\star \leftarrow s$;
        $f^\star \leftarrow f(s)$;
    fim-se;
fim-para;
$s \leftarrow s^\star$;
Retorne s;
fim $GRASP$

Figura 2.1: Algoritmo GRASP

Na fase de contrução, uma solução é iterativamente construída, elemento por elemento. A cada iteração dessa fase, os próximos elementos candidatos a serem incluídos na solução são colocados em uma lista C de candidatos, seguindo um critério de ordenação pré-determinado. Esse processo de seleção é baseado em uma função adaptativa gulosa $g : C \mapsto \Re$, que estima o benefício da seleção de cada um dos elementos. A heurística é adaptativa porque os benefícios associados com a escolha de cada elemento são atualizados em cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior. A componente probabilística do procedimento reside no fato de que cada elemento é selecionado de forma aleatória a partir de um subconjunto restrito formado pelos melhores elementos que compõem a lista de candidatos. Este subconjunto recebe o nome de lista de candidatos restrita (LCR). Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP. Seja $\alpha \in [0, 1]$ um dado parâmetro. O pseudo-código representado pela figura 2.2 descreve a fase de construção GRASP.

procedimento $Construcao(g(.), \alpha, s)$;
$s \leftarrow \emptyset$;
Inicialize o conjunto C de candidatos;
enquanto $(C \neq \emptyset)$ faça
    $t_{min} = \min\{g(t) \mid t \in C\}$;
    $t_{max} = \max\{g(t) \mid t \in C\}$;
    $LCR = \{t \in C \mid g(t) \leq t_{min} + \alpha(t_{max} - t_{min})\}$;
    Selecione, aleatoriamente, um elemento $t \in LCR$;
    $s \leftarrow s \cup \{t\}$;
    Atualize o conjunto C de candidatos;
fim-enquanto;
Retorne s;
fim $Construcao$;

Figura 2.2: Fase de construção de um algoritmo GRASP

Observamos que o parâmetro α controla o nível de gulosidade e aleatoriedade do procedimento $Construcao$. Um valor $\alpha = 0$ faz gerar soluções puramente gulosas, enquanto $\alpha = 1$ faz produzir soluções totalmente aleatórias.

Assim como em muitas técnicas determinísticas, as soluções geradas pela fase de construção do GRASP provavelmente não são localmente ótimas com respeito à definição de vizinhança adotada. Daí a importância da fase de busca local, a qual objetiva melhorar a solução construída. A figura 2.3 descreve o pseudo-código de um procedimento básico de busca local com respeito a uma certa vizinhança $N(.)$ de s.

A eficiência da busca local depende da qualidade da solução construída. O procedimento de construção tem então um papel importante na busca local, uma vez

procedimento $BuscaLocal(f(.), N(.), s)$;

$s^\star \leftarrow s$; {Melhor solução encontrada }
$V = \{s' \in N(s) \mid f(s') < f(s)\}$;
enquanto ($|V| > 0$) faça
    Selecione $s \in V$;
    se ($f(s) < f(s^\star)$) então $s^\star \leftarrow s$;
    $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
fim-enquanto;
$s \leftarrow s^\star$;
Retorne s;

fim $BuscaLocal$;

Figura 2.3: Fase de Busca Local de um algoritmo GRASP

que as soluções construídas constituem bons pontos de partida para a busca local, permitindo assim acelerá-la.

O parâmetro α, que determina o tamanho da lista de candidatos restrita, é basicamente o único parâmetro a ser ajustado na implementação de um procedimento GRASP. Em [46] discute-se o efeito do valor de α na qualidade da solução e na diversidade das soluções geradas durante a fase de construção. Valores de α que levam a uma lista de candidatos restrita de tamanho muito limitado (ou seja, valor de α próximo da escolha gulosa) implicam em soluções finais de qualidade muito próxima àquela obtida de forma puramente gulosa, obtidas com um baixo esforço computacional. Em contrapartida, provocam uma baixa diversidade de soluções construídas. Já uma escolha de α próxima da seleção puramente aleatória leva a uma grande diversidade de soluções construídas mas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O procedimento GRASP procura, portanto, conjugar bons aspectos dos algoritmos puramente gulosos, com aqueles dos procedimentos aleatórios de construção de soluções.

Procedimentos GRASP mais sofisticados incluem estratégias adaptativas para o parâmetro α. O ajuste desse parâmetro ao longo das iterações GRASP, por critérios que levam em consideração os resultados obtidos nas iterações anteriores, produz soluções melhores do que aquelas obtidas considerando-o fixo [85, 86, 87].

2.2.2 *Simulated Annealing*

Trata-se de uma técnica de busca local probabilística, proposta originalmente por Kirkpatrick et al. [71], que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos [38].

Esta técnica começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s.

Chamando de Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$, o método aceita o movimento, e a solução vizinha passa a ser a nova solução corrente, se $\Delta < 0$. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método, chamado de *temperatura* e que regula a probabilidade de aceitar soluções de pior custo.

A temperatura T assume, inicialmente, um valor elevado T_0. Após um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_n \leftarrow \alpha \times T_{n-1}$, sendo $0 < \alpha < 1$. Com esse procedimento, dá-se, no início uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora $\left(T \rightarrow 0 \Longrightarrow e^{-\Delta/T} \rightarrow 0\right)$

O procedimento pára quando a temperatura chega a um valor próximo de zero e nenhuma solução que piore o valor da função objetivo é mais aceita, isto é, quando o sistema está estável. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um mínimo local.

Os parâmetros de controle do procedimento são a razão de resfriamento α, o número de iterações para cada temperatura ($SAmax$) e a temperatura inicial T_0.

Apresenta-se, pela figura 2.4, o algoritmo *Simulated Annealing* básico.

Dependendo do processo de resfriamento, pode ser mostrada a convergência do método a uma solução que seja globalmente ótima [38]. Tal resultado, entretanto, é de utilidade prática restrita, uma vez que implica em tempos computacionais exponenciais no tamanho do problema.

```
procedimento SA(f(.), N(.), α, SAmax, T_0, s)
s* ← s;                   {Melhor solução obtida até então}
IterT ← 0;                {Número de iterações na temperatura T}
T ← T_0;                  {Temperatura corrente}
enquanto (T > 0) faça
    enquanto (IterT < SAmax) faça
        IterT ← IterT + 1;
        Gere um vizinho qualquer s' ∈ N(s);
        Δ = f(s') − f(s);
        se (Δ < 0)
            então
                s ← s';
                se (f(s') < f(s*)) então s* ← s';
            senão
                Tome x ∈ [0, 1];
                se (x < e^(−Δ/T)) então s ← s';
        fim-se;
    fim-enquanto;
    T ← α × T;
    IterT ← 0;
fim-enquanto;
s ← s*;
Retorne s;
fim SA;
```

Figura 2.4: Algoritmo *Simulated Annealing*

2.2.3 Busca Tabu

Descrevemos, a seguir, de forma breve, os princípios básicos da Busca Tabu (BT), técnica originada nos trabalhos [52] e [63]. Referenciamos a [52, 53, 54, 58, 56, 57, 35, 66] para um melhor detalhamento do método.

A Busca Tabu é um procedimento adaptativo que utiliza uma estrutura de memória para guiar um método de descida a continuar a exploração do espaço de soluções mesmo na ausência de movimentos de melhora, evitando que haja a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado.

Mais especificamente, começando com uma solução inicial s_0, um algoritmo BT explora, a cada iteração, um subconjunto V da vizinhança $N(s)$ da solução corrente s. O membro s' de V com menor valor nessa região segundo a função $f(.)$ torna-se a nova solução corrente mesmo que s' seja pior que s, isto é, que $f(s') > f(s)$.

O critério de escolha do melhor vizinho é utilizado para escapar de um mínimo local. Esta estratégia, entretanto, pode fazer com que o algoritmo cicle, isto é, que retorne a uma solução já gerada anteriormente.

De forma a evitar que isto ocorra, existe uma lista tabu T, a qual é uma lista de movimentos proibidos. A lista tabu clássica contém os movimentos reversos aos últimos $|T|$ movimentos realizados (onde $|T|$ é um parâmetro do método) e funciona como uma fila de tamanho fixo, isto é, quando um novo movimento é adicionado à lista, o mais antigo sai. Assim, na exploração do subconjunto V da vizinhança $N(s)$ da solução corrente s, ficam excluídos da busca os vizinhos s' que são obtidos de s por movimentos m que constam na lista tabu.

A lista tabu se, por um lado, reduz o risco de ciclagem (uma vez que ela garante o não retorno, por $|T|$ iterações, a uma solução já visitada anteriormente); por outro, também pode proibir movimentos para soluções que ainda não foram visitadas [35]. Assim, existe também uma *função de aspiração*, que é um mecanismo que retira, sob certas circunstâncias, o *status* tabu de um movimento. Mais precisamente, para cada possível valor v da função objetivo existe um nível de aspiração $A(v)$: uma solução s' em V pode ser gerada se $f(s') \leq A(f(s))$, mesmo que o movimento m esteja na lista tabu. A função de aspiração A é tal que, para cada valor v da função objetivo, retorna outro valor $A(v)$, que representa o valor que o algoritmo aspira ao chegar de v. Um exemplo simples de aplicação desta idéia é considerar $A(f(s)) = f(s^\star) - 1$ onde $s^\star$ é a melhor solução encontrada até então. Neste caso, aceita-se um movimento tabu somente se ele conduzir a um vizinho melhor que $s^\star$.

Duas regras são normalmente utilizadas de forma a interromper o procedimento. Pela primeira, pára-se quando é atingido um certo número máximo de iterações sem melhora no valor da melhor solução. Pela segunda, quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele). Esse segundo critério evita a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é julgada suficientemente boa.

Os parâmetros principais de controle do método de Busca Tabu são a cardinalidade $|T|$ da lista tabu, a função de aspiração A, a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração e $BTmax$, o número máximo de iterações sem melhora no valor da melhor solução.

Apresenta-se, pela figura 2.5, o pseudo-código de um algoritmo de Busca Tabu básico. Neste procedimento f_{min} é o valor mínimo conhecido da função f.

```
procedimento BT(f(.), N(.), A(.), |V|, f_min, |T|, BTmax, s)
s* ← s;                    {Melhor solução obtida até então}
Iter ← 0;                  {Contador do número de iterações}
MelhorIter ← 0;            {Iteração mais recente que forneceu s*}
T ← ∅;                     {Lista Tabu}
Inicialize a função de aspiração A;
enquanto (f(s) > f_min  e  Iter − MelhorIter < BTmax) faça
    Iter ← Iter + 1;
    Seja s' ← s ⊕ m o melhor elemento de V ⊂ N(s) tal que
        o movimento m não seja tabu (m ∉ T) ou
        s' atenda a condição de aspiração (f(s') < A(f(s)));
    T ← T − {movimento mais antigo} + {movimento que gerou s'};
    Atualize a função de aspiração A;
    s ← s';
    se (f(s) < f(s*)) então
        s* ← s;
        MelhorIter ← Iter;
    fim-se;
fim-enquanto;
s ← s*;
Retorne s;
fim BT;
```

Figura 2.5: Algoritmo de Busca Tabu

É comum em métodos de Busca Tabu incluir estratégias de intensificação, as quais visam a concentrar a pesquisa em determinadas regiões consideradas promissoras. Uma estratégia típica é retornar a soluções já visitadas para explorar sua vizinhança de forma mais efetiva. Outra estratégia consiste em incorporar atributos das melhores soluções já encontradas durante o progresso da pesquisa e estimular componentes dessas soluções a tornar parte da solução corrente. Nesse caso, são consideradas livres no procedimento de busca local apenas as componentes não associadas às boas soluções, permanecendo as demais componentes fixas. Um critério de término, tal como um número fixo de iterações, é utilizado para encerrar o período de intensificação.

Métodos baseados em Busca Tabu incluem, também, estratégias de diversificação. O objetivo dessas estratégias, que tipicamente utilizam uma memória de longo prazo, é redirecionar a pesquisa para regiões ainda não suficientemente exploradas do espaço de soluções. Estas estratégias procuram, ao contrário das estratégias de intensificação, gerar soluções que têm atributos significativamente diferentes daqueles encontrados nas melhores soluções obtidas. A diversificação, em geral, é utilizada somente em determinadas situações, como, por exemplo, quando dada uma solução s, não existem movimentos m de melhora para ela, indicando que o algoritmo já exauriu a análise naquela região. Para escapar dessa região, a idéia é estabelecer uma penalidade $w(s, m)$ para uso desses movimentos. Um número fixo de iterações sem melhora no valor da solução ótima corrente é, em geral, utilizado para acionar essas estratégias.

Métodos de Busca Tabu incluem, também, listas tabu dinâmicas [31, 93], muitas das quais atualizadas de acordo com o progresso da pesquisa [12, 11, 10].

Recentemente, [55] e [62] provaram a convergência finita de alguns métodos de Busca Tabu baseados em memória por recenticidade e por frequência.

2.2.4 Algoritmos Genéticos

Trata-se de uma metaheurística que se fundamenta em uma analogia com processos naturais de evolução, nos quais, dada uma população, os indivíduos com características genéticas melhores têm maiores chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer. Referenciamos a [59, 89] para um melhor detalhamento do método.

Nos Algoritmos Genéticos (AGs), cada cromossomo (indivíduo da população) está associado normalmente a uma solução do problema e cada gene está associado a uma componente da solução. Um mecanismo de reprodução, baseado em processos evolutivos, é aplicado sobre a população com o objetivo de explorar o espaço de busca e encontrar melhores soluções para o problema.

Mais especificamente, um Algoritmo Genético inicia sua busca com uma população $\{s_1^0, s_2^0, \ldots, s_n^0\}$, aleatoriamente escolhida, a qual é chamada de população no tempo 0.

O procedimento principal é um *loop* que cria uma população $\left\{s_1^{t+1}, s_2^{t+1}, \ldots, s_n^{t+1}\right\}$ no tempo $t + 1$ a partir de uma população do tempo t. Para atingir esse objetivo, os indivíduos da população do tempo t passam por uma fase de reprodução, a qual consiste em selecionar indivíduos para operações de recombinação e/ou mutação.

Na operação de recombinação, os genes de dois cromossomos pais são combinados de forma a gerar um ou dois cromossomos filhos, de sorte que para cada cromossomo filho há um conjunto de genes de cada um dos cromossomos pais. A operação de mutação consiste em alterar aleatoriamente uma parte dos genes de cada cromossomo (componentes da solução). Ambas as operações são realizadas com uma certa probabilidade.

Gerada a nova população do tempo $t + 1$, define-se a população sobrevivente, isto é, as n soluções que integrarão a nova população. Para definir a população sobrevivente, cada solução é avaliada por uma certa função de aptidão. Os critérios comumente usados para escolher os cromossomos sobreviventes são os seguintes: 1) aleatório; 2) roleta (onde a chance de sobrevivência de cada cromossomo é proporcional ao seu nível de aptidão) e 3) misto (isto é, uma combinação dos dois critérios anteriores). Em qualquer um desses critérios admite-se, portanto, a sobrevivência de indivíduos menos aptos. Isto é feito de forma a tentar-se escapar de ótimos locais.

O método termina, em geral, quando um certo número de populações é gerado ou quando a melhor solução encontrada atinge um certo nível de aptidão ou ainda, quando não há melhora após um certo número de iterações.

Os parâmetros principais de controle do método são: o tamanho n da população, a probabilidade da operação *crossover*, a probabilidade de mutação, o número de gerações e o número de iterações sem melhora.

O pseudo código de um Algoritmo Genético básico está descrito na figura 2.6.

procedimento AG
$t \leftarrow 0$;
Gere a população inicial $P(t)$;
Avalie $P(t)$;
enquanto (os critérios de parada não estiverem satisfeitos) faça
    $t \leftarrow t + 1$;
    Gere $P(t)$ a partir de $P(t-1)$;
    Defina a população sobrevivente $(P(t) \leftarrow f(P(t) \cup P(t-1)))$;
fim-enquanto;
fim AG;

Figura 2.6: Algoritmo Genético

2.2.5 *Annealing* Microcanônico

Trata-se de uma variante do algoritmo *Simulated Annealing*, proposta em [9]. Diferentemente de SA, que baseia-se na simulação dos estados de um sistema físico a temperatura constante, o algoritmo *Annealing* Microcanônico simula a variação dos estados a energia constante. Utiliza, para esse propósito, uma versão do Algoritmo de Creutz [29, 75], o qual introduz uma variável, chamada de *demônio*, para modelar as flutuações de energia.

A partir de uma solução inicial arbitrária s, geram-se novas soluções de forma que $E + D =$ constante, sendo D a energia do demônio ($0 \leq D \leq D_{max}$) e E a energia do sistema ($f(s)$). O procedimento principal consiste em um *loop* que gera, randomicamente, em cada iteração, um <u>único</u> vizinho s' da solução corrente.

Chamando de Δ a variação de energia ao mover-se de s para s', o método aceita o movimento, e a solução vizinha s' passa a ser a nova solução corrente, se $\Delta \leq 0$, com a condição de que $D - \Delta \leq D_{max}$, isto é, que a energia liberada não supere a capacidade do demônio. Caso $\Delta > 0$ a solução vizinha s' também pode ser aceita, desde que $\Delta < D$, isto é, que a energia necessária possa ser suprida pelo demônio. Em ambos os casos, se houver aceitação, o demônio D recebe (ou libera, respectivamente) a variação de energia envolvida ao mover-se de s para s', isto é, $D \leftarrow D - \Delta$. Desta forma, a energia total, $E + D$, permanece constante.

O demônio assume, inicialmente, um valor D_{max}. Após um certo número de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico numa dada configuração de energia), esse valor é gradativamente diminuído até anular-se.

A vantagem de se usar o algoritmo AM é que pode-se estabelecer, com precisão, se o equilíbrio térmico foi atingido [99]. Da física estatística sabe-se que o valor médio do demônio é igual à temperatura $\bar{D} = T$ e no equilíbrio térmico tem-se $\bar{D}/\sigma(D) = 1$, onde $\sigma(D)$ é o desvio-padrão dos valores do demônio. Entretanto, sob o ponto de vista prático, é custoso avaliar computacionalmente se tal equilíbrio foi atingido. Assim, ao invés de avaliá-lo durante o progresso da pesquisa, prefixa-se um número máximo de iterações em um dado nível de energia, o qual passa a ser um parâmetro de controle do método.

Os parâmetros de controle do procedimento são, pois, a taxa α de diminuição da capacidade do demônio, o número de iterações em um dado nível de energia ($AMmax$), o valor D_i do demônio no início de cada fase e a capacidade inicial do demônio D_{max}.

Apresentamos, pela figura 2.7, o pseudo-código de um algoritmo *Annealing* Microcanônico básico.

```
procedimento AM(f(.), N(.), AMmax, α, D_i, D_max, s)
s* ← s;   {Melhor solução obtida até então}
enquanto (D_max > 0) faça
    D ← D_i;
    para (IterE = 1, 2, ..., AMmax) faça
        Gere um vizinho qualquer s' ∈ N(s);
        Δ = f(s') − f(s);
        se (Δ ≤ 0)
            então
                se (D − Δ ≤ D_max) então
                    s ← s';
                    D ← D − Δ;
                    se (f(s') < f(s*)) então s* ← s';
                fim-se;
            senão
                se (D − Δ ≥ 0) então
                    s ← s';
                    D ← D − Δ;
                fim-se;
        fim-se;
    fim-para;
    D_max ← α × D_max;
fim-enquanto;
s ← s*;
Retorne s;
fim AM;
```

Figura 2.7: Algoritmo *Annealing* Microcanônico

2.2.6 Otimização Microcanônica

Trata-se de uma alternativa ao *Annealing* Microcanônico, proposta originalmente em [100]. Referenciamos a [100, 75, 99] para uma descrição mais detalhada do método.

O algoritmo de Otimização Microcanônica (OM), descrito pela figura 2.8, consiste de dois procedimentos, os quais são aplicados alternadamente: inicialização e amostragem.

```
procedimento OM(f(.), N(.), OMmax, s)
    s* ← s;                       {Melhor solução obtida até então}
    Iter ← 0;                     {Número de iterações}
    MelhorIter ← 0;               {Iteração mais recente que forneceu s*}
    enquanto (Iter − MelhorIter < OMmax) faça
        Iter ← Iter + 1;
        InicializacaoOM(f(.), N(.), Imax, s);
        se (f(s) < f(s*)) então
            s* ← s;
            MelhorIter ← Iter;
        fim-se;
        AmostragemOM(f(.), N(.), D_i, D_max, Amax, s);
    fim-enquanto;
    s ← s*;
    Retorne s;
fim OM;
```

Figura 2.8: Algoritmo de Otimização Microcanônica

Na fase de inicialização, OM realiza somente movimentos de melhora, guiando a uma configuração de mínimo local. Para tentar escapar dessa configuração, executa-se a amostragem, fase na qual um grau extra de liberdade (denominada *demônio*) produz pequenas perturbações na solução corrente. Em cada iteração dessa fase, movimentos randômicos são propostos, sendo aceitos apenas aqueles nos quais o demônio é capaz de absorver ou liberar a diferença de custo envolvida. Após a fase de amostragem, uma nova inicialização é realizada e o algoritmo assim prossegue, alternando entre as duas fases, até que uma condição de parada seja satisfeita.

O demônio é definido por dois parâmetros: sua capacidade D_{max} e seu valor inicial D_i. A fase de amostragem gera uma sequência de estados cuja energia é conservada, exceto para pequenas flutuações, as quais são modeladas pelo demônio. Chamando de E_i a energia (custo) da solução obtida na fase de inicialização, de D e E a energia do demônio e da solução, respectivamente, em um dado instante da fase de amostragem, tem-se $E + D = E_i + D_i =$ constante. Portanto, essa fase gera soluções no intervalo $[E_i - D_{max} + D_i,\ E_i + D_i]$.

Os parâmetros principais de controle do algoritmo OM são o número máximo $OMmax$ de iterações consecutivas sem melhora em OM, o número máximo de iterações consecutivas sem melhora na fase de inicialização $Imax$, o número máximo de iterações de amostragem $Amax$, o valor inicial D_i do demônio e a sua capacidade máxima D_{max}.

Apresentamos, pelas figuras 2.9 e 2.10, os pseudo-códigos das fases de inicialização e amostragem, respectivamente, de um algoritmo de Otimização Microcanônica básico.

```
procedimento InicializacaoOM(f(.), N(.), Imax, s)
s* ← s;                  {Melhor solução obtida até então}
Iter ← 0;                {Número de iterações}
MelhorIter ← 0;          {Iteração mais recente que forneceu s*}
enquanto (Iter − MelhorIter < Imax) faça
    Iter ← Iter + 1;
    Gere um vizinho qualquer s' ∈ N(s);
    Δ = f(s') − f(s);
    se (Δ < 0)
        então
            s ← s';
            se (f(s') < f(s*)) então
                s* ← s';
                MelhorIter ← Iter;
            fim-se;
        senão Ponha Δ na lista dos movs rejeitados;
    fim-se;
fim-enquanto;
s ← s*;
Retorne s;
fim InicializacaoOM;
```

Figura 2.9: Fase de inicialização do algoritmo de Otimização Microcanônica

```
procedimento AmostragemOM(f(.), N(.), D_i, D_max, Amax, s)
Seja s solução advinda da fase de inicialização;
Escolha D_max e D_I da lista de movimentos rejeitados;
D ← D_i;      {Valor corrente do demônio}
para (Iter = 1, 2, ..., Amax) faça
    Gere um vizinho qualquer s' ∈ N(s);
    Δ = f(s') − f(s);
    se (Δ ≤ 0)
        então
            se (D − Δ ≤ D_max) então
                s ← s';
                D ← D − Δ;
            fim-se;
        senão
            se (D − Δ ≥ 0) então
                s ← s';
                D ← D − Δ;
            fim-se;
    fim-se;
fim-para;
Retorne s;
fim AmostragemOM;
```

Figura 2.10: Fase de amostragem do algoritmo de Otimização Microcanônica

Capítulo 3

Estado da Arte

3.1 Introdução

Começando com os trabalhos pioneiros de Csima e Gotlieb [30] e Gotlieb [60] na década de 60, muitos artigos relacionados a problemas de horários encontram-se descritos na literatura e diferentes procedimentos para resolvê-los foram propostos.

A maioria das técnicas antigas (vide Schmidt e Strohlein [95]), é baseada na simulação da forma humana de resolver o problema. Essas técnicas, chamadas de heurísticas construtivas, consistem no preenchimento gradual do quadro de horário. Isto é, um quadro de horário é preenchido inserindo-se uma aula a cada vez. Esse processo de construção continua até que todas as aulas sejam alocadas ou até que um conflito apareça. Nesta última situação, procura-se remanejar algumas das aulas previamente alocadas de forma a eliminar o conflito. A idéia principal por trás desses procedimentos é: aloque primeiro as aulas mais *difíceis*. As heurísticas construtivas diferem entre si apenas no conceito do que se entende por aula *mais difícil*. Um exemplo típico dessa técnica, descrita em Junginger [70], é o sistema SCHOLA, o qual é baseado nas três seguintes estratégias:

a) Aloque a aula *mais urgente* ao horário *mais favorável* para essa aula;

b) Quando um horário puder ser usado apenas por uma única aula, reserve esse horário para essa aula;

c) Mude uma aula já alocada para um horário livre a fim de que seu antigo horário possa ser usado por uma outra aula, a qual está sem alocação. Isto é, suponha que uma dada aula X esteja alocada a um horário H1 e que X possa ser, também, transferida para o horário H2, o qual está livre. Se uma outra aula Y precisa ser alocada, mas no horário livre H2 o professor de Y está indisponível, então transfira a aula X para o horário H2, tornando disponível o horário H1 para a alocação da aula Y, com a condição de que o professor de Y esteja disponível nesse horário.

Nesse sistema, uma aula é *urgente* quando o professor tem pouca disponibilidade e muitas aulas ainda para ministrar. Um horário é *favorável* quando existe um número reduzido de aulas de outros professores que podem ser alocadas naquele horário, em função de suas disponibilidades.

O sistema procura alocar as aulas alternando as estratégias (a) e (b) tanto quanto possível. Quando nenhuma aula pode mais ser alocada dessa forma, usa-se a estratégia (c).

A estratégia (a) é a alma do sistema, e ela é empregada na maioria dos sistemas que usam técnicas construtivas, com diferentes formas de definir urgência de uma aula e horário mais favorável. O uso da estratégia (b) visa a prevenir a estratégia (a) de atingir prematuramente seu final. A estratégia (c) proporciona um mecanismo limitado de se tentar retornar a uma situação anterior de forma a corrigir uma alocação inadequada feita pelo uso da estratégia (a).

Uma outra heurística construtiva é proposta por Papoulias [84]. De forma a privilegiar o espalhamento das aulas de um mesmo professor para uma mesma turma ao longo da semana, ele considera, dentre outros requisitos, que um horário é mais favorável para uma aula se as outras aulas do professor não estão alocadas a dias consecutivos para essa turma.

de Gans [33] descreveu um procedimento heurístico construtivo, aplicado a escolas secundárias da Holanda, que faz uso de regras não determinísticas para selecionar tanto a aula mais urgente quanto o horário mais favorável para essa aula. O autor salienta que, embora a probabilidade de se gerar randomicamente um quadro de horário viável seja muito pequena, sua experiência mostra que o uso de tais mecanismos aleatórios de seleção não são tão ruins quanto parece à primeira vista.

Uma heurística construtiva é também formulada por Mata [77] para tratar um problema de horários em escolas secundárias brasileiras, nas quais um certo número de aulas duplas é exigido para um determinado conjunto de matérias.

Num segundo momento, os pesquisadores começaram a aplicar técnicas gerais para resolver o problema, tais como programação inteira [102, 48], fluxo em grafo

[83] e coloração em grafos [104, 81, 103].

Tripathy [102] aplicou relaxação lagrangeana para tratar de um problema de alocação de cursos que se reduz a um problema professor-turma básico. O autor utiliza uma função objetivo linear que mensura o desejo de uma aula de um determinado curso ser dada em um certo horário. As restrições que impedem os estudantes que têm um mesmo currículo de assistirem a mais de uma aula em um mesmo horário são relaxadas e incorporadas à função objetivo. As variáveis duais relativas a estas restrições são obtidas por um método de otimização por subgradientes combinado com um procedimento *branch-and-bound*.

Pela técnica de coloração em grafos, um PHE é reduzido a um problema de p-coloração de um grafo, interpretando cada cor como um horário diferente. Mais precisamente, dado um problema de horários, constrói-se seu grafo $G = (V, A)$. Cada vértice $v \in V$ corresponde a uma aula e há um arco $(v_1, v_2) \in A$ entre cada par de aulas que não podem ser alocadas simultaneamente. O problema, então, é o de colorir os vértices de G, usando-se apenas p cores, uma para cada horário, de forma que as aulas de um determinado horário estejam alocadas a vértices de uma mesma cor. As pré-alocações são tratadas prefixando-se as cores dos vértices correspondentes. Já as restrições de disponibilidade de professores e turmas são manipuladas restringindo-se as cores que podem ser usadas nos vértices correspondentes.

Čangalović e Schreuder [103] apresentaram um método exato, baseado em coloração de grafos com pesos, para tratar problemas de horários em escolas com aulas de durações diferentes. Cada aula é representada como um vértice com um peso igual à sua duração. O algoritmo consiste em um procedimento de enumeração implícita baseado no princípio *branch-and-bound*.

A NP-completude do problema de programação de horários é mostrada no trabalho de Even, Itai e Shamir [42], a partir da redução ao problema de 3-SATISFABILIDADE de uma versão simplificada do problema básico de programação de horários em escolas, em que aparecem restrições de indisponibilidade de professores. Esses autores apresentaram, também, um procedimento para resolver em tempo polinomial um PHE básico em que turmas e professores estão sempre disponíveis. Esse procedimento consiste em resolver uma seqüência de problemas de fluxo em grafo. Em cada um desses problemas mostra-se que é sempre possível emparelhar os professores às turmas, em vista da inexistência de restrições de indisponibilidade. Esse procedimento é apresentado na seção 3.2.

Baseado na idéia de Even et al. [42], Ostermann e de Werra [83] propuzeram um método heurístico, o qual consiste em reduzir o problema de horários a uma seqüência de problemas de fluxo em grafo. É criado um grafo para cada horário e existe um arco ligando um nó professor a um nó turma, se há uma aula envolvendo o par. O custo de cada arco define a *urgência* da aula. O objetivo, então, é o de encontrar um emparelhamento de peso máximo. De forma semelhante, Chahal e de Werra [23] desenvolveram um sistema, o qual consiste em determinar, a cada horário, um conjunto de caminhos (chamados de *path packing*), representativos das aulas naquele horário. Nesse sistema, diferentemente do problema tradicional, não se fixa um professor a uma matéria. Um detalhamento maior desse último artigo é feito na seção 3.4. Já de Werra [34] apresenta um método semelhante ao de Ostermann e de Werra [83], criando um grafo para cada turma. Uma descrição sucinta desse procedimento é apresentada na seção 3.3.

Mais recentemente, apareceram soluções baseadas em novas técnicas de pesquisa, tais como Simulated Annealing [1, 98, 3], Busca Tabu [65, 28, 93, 5, 106, 94], Algoritmos Genéticos [2, 24, 39, 18, 49], Satisfação de restrições [107, 45] e combinação de métodos diferentes [26].

Fahrion e Dollansky [45] e Yoshikawa et al. [107] tratam a programação de horários como um Problema de Satisfação de Restrições (CSP). Em um CSP procura-se encontrar o valor de cada variável discreta, definida em um certo domínio finito, satisfazendo a um conjunto de restrições. Para resolver um CSP, utiliza-se uma linguagem de programação baseada em restrições, permitindo expressar restrições de vários tipos, as quais são verificadas durante os cálculos.

Abramson [1] aplicou *Simulated Annealing* para resolver um PHE na qual se considera, também, a alocação de salas. A função objetivo a ser minimizada é a soma do número de conflitos, em cada horário, de turmas, professores e salas, isto é, a soma do número de vezes, menos um, que cada turma, professor ou sala aparece em cada horário, se esse número for maior que zero. Assim, se em um determinado horário, uma turma, professor ou sala não aparece nenhuma vez ou apenas uma única vez, o custo associado é nulo. Uma solução é descrita por uma lista de aulas, uma lista para cada horário, sendo cada aula a associação entre um professor, uma turma e uma sala, o que o autor chama de *elemento*. Dada uma solução, a escolha da solução vizinha é feita como segue: seleciona-se, aleatoriamente, um horário e um *elemento* (aula) para ser removido desse horário. Em seguida, escolhe-se, também aleatoriamente, um horário diferente para inserir essa aula. Calcula-se, então, a diferença de custo entre inserir essa aula no novo horário e removê-la do horário antigo. Se esse custo for negativo, isto é, melhorar a solução corrente, o movimento é aceito. Se, no entanto, a solução vizinha for pior, ainda assim ela pode ser aceita, mas com uma certa probabilidade, a qual depende da *temperatura* do processo. O sistema desenvolvido não considera a ocorrência de aulas consecutivas, mas sugere a introdução na função objetivo de um parâmetro de penalidade que leve em consideração o número de aulas consecutivas não atendidas. Pela introdução de componentes de penalidade na função objetivo, o autor sugere contemplar, também, problemas nos quais as turmas tenham estudantes em comum, bem como aulas simultâneas.

Abramson et al. [3] compararam seis esquemas diferentes de resfriamento em um algoritmo *Simulated Annealing* para resolver problemas de horários em escolas. Um desses esquemas, que produziu soluções de melhor qualidade em um menor espaço de tempo, consiste em uma sequência de reaquecimento, seguido de resfriamento, quando a temperatura atinge um determinado nível, o qual é determinado pelo próprio método.

Hertz [65] aplicou Busca Tabu para resolver um problema onde as matérias são divididas em tópicos, os quais devem ser ministrados em uma certa ordem para uma dada turma ao longo de um conjunto de dias. A cada tópico está associado um professor, uma data de início e final e um certo número de aulas, as quais são agrupadas em um certo número de sessões. Cada sessão pode ter uma duração fixa ou não, dependendo do tipo de tópico. Considera-se dois tipos de tópicos: os estáticos, nos quais as sessões têm duração fixa e são ordenadas a priori e os tópicos dinâmicos, nos quais as sessões têm duração arbitrária, mas limitadas por valores mínimos e máximos. Por exemplo, um tópico dinâmico de 12 horas-aula, tendo sessões de curso de 2 ou 3 horários consecutivos, pode ser dividido na seguinte seqüência não ordenada de sessões: (2,2,2,2,2,2), (2,2,2,3,3) ou (3,3,3,3). No problema considerado, portanto, não se conhece, a princípio, o número de sessões de cada tópico dinâmico.

As restrições do problema são divididas em dois grupos, um dos quais é relaxado e considerado na função objetivo, penalizando-se cada restrição desse grupo que for violada. O objetivo é encontrar uma solução que satisfaça a um conjunto de restrições consideradas essenciais e que minimize o número de restrições relaxadas violadas. Um movimento é definido pela mudança de alocação de uma sessão de um tópico a um dia diferente. Para os tópicos dinâmicos são utilizadas várias alternativas diferentes de movimentos.

Costa [28] e Schaerf [93] aplicaram Busca Tabu para resolver um problema de horários em escolas em que aparecem restrições de vários tipos. Schaerf [93] representa uma solução por uma matriz, onde suas linhas representam os professores e as colunas, os horários reservados para as aulas. Cada elemento da matriz, referente a um certo professor e a um dado horário, contém o nome da turma para a qual o professor vai lecionar naquele horário. Um movimento consiste em trocar duas aulas de um dado professor ou mover uma aula para um horário diferente. Costa [28], ao contrário, empregou um tipo de movimento diferente. Mais precisamente, ele permite somente a mudança de uma única aula para um horário diferente. Assim, nessa representação, um professor pode ensinar mais que uma aula ao mesmo tempo. A função objetivo nessas duas abordagens é uma soma de componentes que avaliam o atendimento aos diversos requisitos exigidos para o quadro de horário. A cada componente está associado um peso, que reflete sua importância. Tais artigos encontram-se detalhados nas seções 3.5 e 3.6.

Alvarez-Valdes et al. [5] desenvolveram um algoritmo de três fases para tratar um problema de horários em escolas da Espanha. Na primeira fase uma solução inicial é construída, horário por horário. As aulas que são escolhidas para serem alocadas a cada horário são determinadas a partir da construção de um conjunto independente de cardinalidade máxima em um grafo de recursos. O resultado da primeira fase é uma solução na qual todas as aulas encontram-se alocadas e satisfazendo a todas as restrições estabelecidas, mas cujo número de horários requeridos pode ultrapassar o número de horários reservados semanalmente pela escola. De forma a obter uma solução viável, executa-se a segunda fase do algoritmo, usando-se, para tanto, um procedimento de Busca Tabu. O espaço de soluções considerado é o conjunto de todos os quadros de horário que satisfazem a todas as restrições. Um movimento consiste em uma operação envolvendo uma aula propriamente alocada (isto é, dentro do limite de horários da escola) e uma aula alocada fora dos limites da semana escolar. A função objetivo, a ser minimizada, é o número de aulas não propriamente alocadas. Encontrada uma solução viável, o algoritmo inicia a terceira fase, a qual tem por objetivo melhorar a compacidade do quadro de horário dos professores. Para cumprir esse objetivo os autores criam um grafo de horários, um para cada turma, e procuram ciclos de custo negativo nos mesmos. Um mecanismo com os mesmos princípios encontra-se detalhado e estendido na seção 6.6.

Colorni et al. [24] aplicaram Algoritmos Genéticos ao PHE. Quadros inviáveis de horário também são incluídos no espaço de busca do algoritmo. A função objetivo considera requisitos didáticos, organizacionais e pessoais e incorpora, também, o número de inviabilidades. De forma a conduzir a busca em direção a quadros viáveis de horário, a função objetivo é hierarquizada, atribuindo-se às inviabilidades um peso maior do que os dos demais requisitos. Esse artigo encontra-se detalhado na seção 3.7.

Uma técnica recente de otimização, conhecida como *Algoritmos Meméticos* (os quais podem ser vistos como uma extensão dos Algoritmos Genéticos, onde ao invés de genes tem-se os *memes*, e cada indivíduo passa por uma operação de busca local antes de se submeter às operações de recombinação e mutação) está sendo desenvolvida para resolver o problema de horários em escolas (vide Moscato e Schaerf [79]). Esta técnica já tem aplicações bem sucedidas em problemas de horários de exames e de cursos [17, 88, 16]. Uma introdução ao assunto pode ser encontrado em Moscato [78].

Um modelo de programação linear inteira é usado por Birbas [13] para tratar de problemas de horários em escolas gregas. Professores têm que assumir pelo menos uma aula em cada dia de trabalho. Alguns requisitos de qualidade também são tratados como restrições. Esse é o caso do número de aulas (em horas-aula) que devem ser alocadas diariamente a cada turma. Para que essa quantidade seja equilibrada ao longo da semana, impõe-se que esse número diário não exceda a média em mais do que uma unidade. Determinadas matérias devem ser ensinadas preferencialmente nos primeiros horários. Para atender a esse requisito, considera-se uma função objetivo linear com um custo que penaliza as aulas dessas matérias que ocorram nos horários finais do turno e associa-se, também, um custo mais elevado para as aulas das outras matérias que venham a ocorrer nos horários iniciais. Com esse mecanismo, o modelo é capaz de gerar, indiretamente, quadros de horário de professores com um número reduzido de *buracos*. Os problemas, os quais têm dimensões reduzidas, são resolvidos sem decomposição por uma técnica *branch-and-bound*.

Wood e Whitaker [105] tratam um problema de programação de horários em escolas secundárias da Nova Zelândia onde os estudantes têm liberdade para escolher as matérias. Apresenta-se uma formulação de programação matemática com algumas restrições não lineares. Essa formulação não tem uma função objetivo e a meta é encontrar uma solução viável. Como usualmente se faz, divide-se o conjunto das restrições em dois grupos. O grupo das restrições *não essenciais* é relaxado e variáveis objetivo são inseridas em cada uma delas. Isto é, uma variável não-negativa é introduzida em cada restrição relaxada, bem como na função objetivo. A cada variável objetivo está associado um custo na função objetivo, de forma a distinguir a importância da restrição frente às demais. Entre as restrições consideradas não essenciais incluem-se aquelas referentes à escolha das matérias pelos estudantes. O modelo resultante é então decomposto e resolvido heurísticamente em três estágios, combinando as técnicas *Simulated Annealing*, algoritmo húngaro e algoritmo exato de enumeração. O modelo não leva em consideração restrições de compacidade nem de duração de cada aula.

Evenborn e Ronnqvist [43, 44] consideram um PHE em que as aulas não têm um padrão de duração e podem, além disso, começar em qualquer intervalo de tempo. Assim, uma turma pode ter uma aula de 35 minutos às 8:20 e outra de 50 minutos às 9:05. No modelo considerado, cada turma deve ter pelo menos uma aula de cada matéria por dia e, por uma questão de disponibilidade de espaço, há um limite no número de alunos que podem estar livres, ao mesmo tempo, para o horário das refeições. Requisitos de qualidade incluem, dentre outros: 1) turmas com carga horária de duração uniforme ao longo da semana; 2) espalhamento de aulas de uma mesma matéria para uma mesma turma e 3) intervalos entre aulas de uma mesma turma (ou de um mesmo professor) tão pequenos quanto possível. O problema é decomposto em 3 subproblemas, os quais são resolvidos sequencialmente. No primeiro, decide-se qual dia cada aula deve ser ministrada. No segundo, estipulase tanto a sala quanto o horário de início das aulas simultâneas. No terceiro e último passo, faz-se a alocação das salas e dos horários de início das aulas remanescentes. O objetivo do primeiro passo é o de disponibilizar o maior número possível de soluções viáveis para os passos seguintes, o que é alcançado maximizando as folgas de cada recurso (turma, professor e sala). Um procedimento baseado na formulação de particionamento de conjuntos, combinado com geração de colunas para tratar o terceiro passo, é utilizado para resolver o problema.

Faz-se, a seguir, uma descrição mais detalhada dos principais procedimentos heurísticos encontrados na literatura para resolver o problema de programação de horários em escolas em diversos contextos. Outros artigos, os quais envolvem procedimentos para resolver, principalmente, problemas de horários de cursos e exames, encontram-se listados nas referências bibliográficas.

3.2 Problema Básico de Programação de Horários em Escolas

Descreve-se, a seguir, através de um modelo utilizado por Even et al. [42], o problema básico de programação de horários em escolas.

Os autores mostram que o problema geral de programação de horários é NP-completo, a partir da redução de um caso particular do problema básico de horários em escolas (envolvendo 3 horários e com restrição de disponibilidade de professores) a um problema de 3-SATISFABILIDADE (satisfabilidade de uma forma conjuntiva normal com 3 literais por cláusula), que foi demonstrado ser NP-completo por Cook [25] em 1971.

Os autores mostram, também, dois casos particulares que podem ser resolvidos por algoritmos polinomiais. No primeiro caso, considera-se a situação em que turmas e professores estão sempre disponíveis e a carga horária de cada professor e turma está limitada ao número de horários disponibilizados para as aulas. O procedimento apresentado baseia-se no emparelhamento de turmas e professores em grafos bipartidos. O outro caso particular apresentado considera que cada professor só está disponível em 2 horários.

3.2.1 Formulação do Problema

Sejam os seguintes dados:

(1) um conjunto finito H de horários semanais;

(2) um conjunto de m professores, aos quais estão associados conjuntos $T_i \subseteq H$, representativos dos horários durante os quais o professor i está disponível para ensino;

(3) um conjunto de n turmas e um conjunto $C_j \subseteq H$, onde C_j é o conjunto de horários durante os quais a turma j está disponível;

(4) uma matriz $R = (r_{ij})_{m\times n}$ de números inteiros não-negativos, os quais representam o número de aulas que cada professor i tem que ministrar para cada turma j;

Definição 3.2.1 *O problema básico de programação de horários em escolas, denotado por PBHE, consiste em determinar a existência de uma função encontro,* $f(i,j,k) : \{1,\ldots,m\} \times \{1,\ldots,n\} \times H \mapsto \{0,1\}$, *(onde* $f(i,j,k) = 1$ *se e somente se o professor* i *ensina para a turma* j *no horário* k*), satisfazendo:*

(a) $f(i,j,k) = 1 \Rightarrow k \in (T_i \cap C_j)$;

(b) $\sum_{k\in H} f(i,j,k) = r_{ij} \quad \forall i = 1,\ldots,m \;\; e \;\; \forall j = 1,\ldots,n$;

(c) $\sum_{i=1}^{m} f(i,j,k) \leq 1 \quad \forall j = 1,\ldots,n \;\; e \;\; k \in H$;

(d) $\sum_{j=1}^{n} f(i,j,k) \leq 1 \quad \forall i = 1,\ldots,m \;\; e \;\; k \in H$

A restrição (a) assegura que existe um encontro entre um professor i e uma turma j somente se ambos estiverem disponíveis no horário k. As restrições (b) garantem que o número total de encontros durante a semana entre o professor i e a turma j é exatamente o número requerido r_{ij}. As restrições (c) impedem que alguma turma tenha mais do que um professor em um dado horário e o conjunto de restrições (d), por sua vez, assegura que nenhum professor ensina para mais de uma turma simultaneamente.

3.2.2 A NP-completude do Problema

Even et al. [42] demostram que a instância do problema de programação de horários definida a seguir pertence à classe dos problemas NP-completos e que, portanto, o problema geral de programação de horários também o é.

Teorema 3.2.1 *Pertence à classe NP-completo o problema básico de programação de horários em escolas (PBHE) que tem as seguintes características:*

(1) $|H| = 3$*, isto é, existem apenas 3 horários de aula para cada turma;*

(2) $C_j = H \quad \forall j = 1, \ldots, n$*, isto é, as turmas estão sempre disponíveis;*

(3) $|T_i| \in \{2, 3\}$*, isto é, cada professor tem dois ou três horários disponíveis para ensino;*

(4) $r_{ij} \in \{0, 1\}$*, isto é, cada professor i ministra, no máximo, uma aula para cada turma* j*.*

3.2.3 Um Problema de Horários Polinomial

Descreveremos nesta seção um problema de programação de horários que pode ser resolvido por um algoritmo polinomial. Antes de enunciá-lo, porém, apresentaremos alguns conceitos necessários à prova desse fato.

Seja o grafo bipartido $G = (V, E)$, onde $V = X \cup Y$ é o conjunto de vértices, E é o conjunto de arestas (onde cada aresta é um par não ordenado de vértices distintos com extremidades em X e Y). O conjunto dos vértices adjacentes a um dado vértice $v \in V$ é chamado de *vizinhança* de v em G e será denotado por $N(v)$. Chamaremos de $N(S)$ a *vizinhança* de um subconjunto $S \subseteq X$ de vértices no grafo G, isto é, o conjunto de vértices de G adjacentes a algum vértice de S. Definimos o grau de um vértice como sendo o número de arestas à ele incidentes.

Definição 3.2.2 *Seja* $G = (V, E)$ *um grafo bipartido com partição* $V = X \cup Y$*. Seja* $M \subseteq E$ *um emparelhamento de* G*, isto é, um conjunto de arestas* (x_i, y_i) *tal que nenhum par de arestas em* M *tem uma extremidade comum. Dizemos que* M *é um* ***emparelhamento perfeito*** *se todo vértice* $v \in V$ *é incidente a alguma aresta de* M*.*

Definição 3.2.3 *Seja* M *um emparelhamento de* $G = (V, E)$*. Dizemos que* M *satura um vértice* $v \in V$ *se existe alguma aresta* $e \in M$ *que é incidente a* v*.*

É interessante observar que em um grafo bipartido com emparelhamento perfeito as partições X e Y têm o mesmo número de vértices, há exatamente $|X| = |Y|$ arestas e todo vértice é saturado.

O teorema 3.2.2, a seguir, conhecido como Teorema de Hall [61], cuja demonstração pode ser encontrada em Figueiredo e Szwarcfiter [50], descreve condições necessárias e suficientes para a existência de um emparelhamento que satura todo vértices de X.

Teorema 3.2.2 (Teorema de Hall) *Seja G um grafo bipartido com bipartição X, Y. Então G admite emparelhamento que satura todo vértice de X se e somente se $|N(S)| \geq |S|$ para todo $S \subseteq X$.*

O corolário 3.2.1 segue imediatamente do Teorema 3.2.2 e caracteriza a existência de emparelhamentos perfeitos em grafos bipartidos.

Corolário 3.2.1 *Seja G um grafo bipartido com bipartição X, Y. Então G admite emparelhamento perfeito se e somente se $|X| = |Y|$ e $|N(S)| \geq |S|$, para todo $S \subseteq X$.*

Um grafo bipartido *regular*, isto é, em que todos os vértices têm o mesmo grau, sempre admite emparelhamento perfeito. Este resultado pode ser obtido como consequência do Teorema de Hall.

Corolário 3.2.2 *Se G é um grafo bipartido p-regular, com $p > 0$, então G tem um emparelhamento perfeito.*

Prova: Seja $G = (V, E)$ um grafo bipartido p-regular com bipartição X, Y. Como G é p-regular, o número de arestas de G é $|E| = p|X| = p|Y|$ e como $p > 0$, segue que $|X| = |Y|$. Agora seja $S \subseteq X$ e sejam E_1 as arestas incidentes a S e E_2 as arestas incidentes a $N(S)$. Por definição de $N(S)$, toda aresta incidente a S é também incidente a $N(S)$. Além disso, $E_1 \subseteq E_2$. Portanto, $p|S| = |E_1| \leq |E_2| = p|N(S)|$, o que implica $|S| \leq |N(S)|$ e garante, pelo Teorema de Hall, que G tem um emparelhamento que satura todo vértice em X. Como $|X| = |Y|$, este emparelhamento é perfeito. ■

Enunciaremos, a seguir, um problema cuja solução pode ser obtida por um algoritmo polinomial.

Definição 3.2.4 *Seja (PBHER) o problema (PBHE), definido à pág. 33, satisfazendo as seguintes condições adicionais:*

(1) $|T_i| = |C_j| = H \quad \forall i = 1, \ldots, m$ *e* $\forall j = 1, \ldots, n$, *isto é, professores e turmas estão disponíveis em todos os horários;*

(2) $\sum_{i=1}^{m} r_{ij} \leq |H| = p \quad \forall j = 1, \ldots, n$, *isto é, cada turma tem, no máximo,* p *aulas semanais;*

(3) $\sum_{j=1}^{n} r_{ij} \leq |H| = p \quad \forall i = 1, \ldots, m$, *isto é, cada professor tem, no máximo,* p *aulas semanais.*

As condições (2) e (3) são claramente necessárias para a existência de uma solução para o problema, mas não são suficientes.

A demonstração do teorema 3.2.3 a seguir apresenta uma forma construtiva de se obter uma solução para o problema (PBHER). Mostra-se que este problema pode ser resolvido reduzindo-o a uma seqüência de problemas de emparelhamento perfeito em multigrafos bipartidos, para os quais existem algoritmos polinomiais como, por exemplo, o de Hopcroft e Karp [67] .

Teorema 3.2.3 *O problema (PBHER) tem uma função encontro.*

Prova: Inicialmente, sejam:

$$q = \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij},\ p = |H|,\ \nu = n - \lfloor q/p \rfloor \text{ e } \mu = m - \lfloor q/p \rfloor$$

Agora, seja t_i um vértice representando um professor, c_j um vértice representando uma turma e o multigrafo bipartido $G = (X \cup Y, E)$ onde:

$$X = \{t_1, t_2, \ldots, t_m\} \cup \{\xi_1, \xi_2, \ldots, \xi_\nu\}$$
$$Y = \{c_1, c_2, \ldots, c_n\} \cup \{\eta_1, \eta_2, \ldots, \eta_\mu\}$$

E é o conjunto das arestas que ligam os vértices de X e Y, construídos como segue. Para cada $i = 1, \ldots, m$ e $j = 1, \ldots, n$ colocam-se r_{ij} arestas paralelas entre os vértices t_i e c_j. A seguir, para cada $i = 1, \ldots, m$, completa-se o grau do vértice t_i até que ele atinja exatamente p. Isto é feito inserindo-se $p - \sum_{j=1}^{n} r_{ij}$ arestas entre t_i e qualquer um dos vértices em $\{\eta_1, \eta_2, \ldots, \eta_\mu\}$. Igualmente, para cada $j = 1, \ldots, n$

completa-se o grau do vértice c_j inserindo-se $p - \sum_{i=1}^{m} r_{ij}$ arestas entre c_j e qualquer um dos vértices em $\{\xi_1, \xi_2, \xi_\nu\}$ até que se atinja p. Finalmente, conectando-se ξ_l a ν_k, completa-se o grau de cada um dos vértices em $\{\xi_1, \xi_2, \xi_\nu\}$ e $\{\eta_1, \eta_2, \ldots, \eta_\mu\}$ até que cada um também atinja o grau p.

Mostra-se, a seguir, que é sempre possível construir o multigrafo pelo procedimento anterior.

Com efeito, o número de arestas necessárias para completar o grau dos vértices $t_1, t_2, \ldots, t_m$ é $m \times p - q$. Pode-se fazer esta operação uma vez que μ satisfaz a desigualdade $\mu \times p \geq m \times p - q$. Analogamente, o número de arestas necessárias para completar o grau dos vértices $c_1, c_2, \ldots, c_n$ é $n \times p - q$ e esta operação pode ser feita, haja visto que ν satisfaz a relação $\nu \times p \geq n \times p - q$. Finalmente, o número de arestas requeridas para completar o grau dos vértices $\xi_1, \xi_2, \ldots, \xi_\nu$ é $\nu \times p - (n \times p - q)$, parcela esta representativa do resto da divisão de p por q, isto é, $q - \lfloor q/p \rfloor \times p$. Igualmente, o número de arestas requeridas para completar o grau dos vértices $\eta_1, \eta_2, \ldots, \eta_\mu$ é o mesmo. Do exposto, conclui-se que é possível construir o multigrafo segundo o procedimento anterior.

Será mostrado, a seguir, que as restrições (a), (b), (c) e (d) do problema básico do horário (PBHE), explicitadas na seção 3.2.1, são satisfeitas.

De fato, tendo em vista que todos os vértices de G têm, por construção, o mesmo grau p, e que, portanto, G é p-regular, então, pelo corolário 3.2.2, G tem um emparelhamento perfeito. Portanto, existe um conjunto de $m + \nu (= n + \mu)$ arcos, onde qualquer par de arcos não tem um vértice em comum. Esse conjunto M de arcos será utilizado para encontrar a solução (função encontro) para o primeiro horário $h_1 \in H$. Assim, se $(x_i, y_j) \in M$ fazemos $f(i, j, h_1) = 1$; caso contrário $f(i, j, h_1) = 0$. Claramente, as condições (c) e (d) de (PBHE), explicitadas na seção 3.2.1, são satisfeitas no horário h_1. A seguir, remove-se M de E. O novo grafo G' tem, agora, grau $p - 1$ em todos os seus vértices. Este fato garante, em vista do corolário 3.2.2, a existência de um outro emparelhamento completo M' de X e Y, o qual pode ser usado para encontrar a solução para o segundo horário $h_2 \in H$. Esse procedimento é repetido até que na p-ésima iteração todas as $|E|$ arestas tenham sido usadas, o que assegura que a condição (b) seja satisfeita. Fica, assim, completa a prova do teorema 3.2.4. ∎

3.2.4 Outro Problema Polinomial

Teorema 3.2.4 *O (PBHE) com* $|T_i| = 2$ *pode ser resolvido em* $O(m^2)$.

Os autores apresentam um procedimento simples de busca em árvore que resolve, em tempo polinomial, um problema (PBHE) onde cada professor ministra exatamente 2 horários de aula semanais. O algoritmo proposto retorna um quadro de horário viável ou mostra que é impossível encontrar um.

3.3 Procedimentos Baseados em Fluxo em Grafos

Descreve-se, a seguir, sucintamente, dois procedimentos heurísticos apresentados por de Werra [34] para resolver um problema básico de horários (PBHE), formulado na seção 3.2.1, no qual aparecerem restrições de indisponibilidade de professores e turmas.

Os algoritmos propostos visam tão somente obter uma solução viável. O primeiro deles, descrito na seção 3.3.1 consiste em construir o quadro de horário de uma turma por vez, a partir da obtenção do fluxo máximo em um grafo associado ao problema. Para melhor compreensão desse procedimento, apresenta-se um exemplo na seção 3.3.2.

Na seção 3.3.3 indicamos como funciona o segundo procedimento, o qual consiste em fixar os horários, ao invés das turmas.

3.3.1 Fixando as Turmas

Dada uma turma c_j, construa um grafo como segue:

(1) Introduza um vértice para cada horário h_k e para cada um dos professores t_i que dão aula para a turma;

(2) Ligue cada nó h_k a cada nó t_i somente se o professor t_i estiver disponível no horário h_k;

(3) Introduza um vértice fonte s com arcos (s, h_k) para todos os horários h_k e um vértice destino t com arcos (t_i, t) para todos os professores t_i;

(4) Utilize como capacidade u dos arcos (t_i, t) o número de aulas do professor t_i para a turma c_j, isto é, $u(t_i, t) = r_{ij}$;

(5) Fixe a capacidade dos demais arcos a uma unidade, isto é, $u(s, h_k) = 1 \ \ \forall k$ e $u(h_k, t_i) \ = \ 1 \ \forall k, i$.

A seguir, calcule o fluxo máximo de s a t. A solução de fluxo máximo desse grafo fornece um quadro de horário de aulas para a turma c_j, uma vez que à cada componente da solução de fluxo máximo, cujo valor é sempre inteiro (no caso, igual a 1), está associado um arco ligando um professor a um horário.

A construção do grafo é repetida para todas as demais turmas, de forma a se tentar obter um quadro de horário completo. Evidentemente, uma turma terá um quadro de horário totalmente preenchido se o fluxo máximo associado a seu grafo for igual ao número de horários de aula para essa turma. Caso contrário, ou não existe solução ou esse procedimento fez uma alocação inadequada de professores a horários em alguma outra turma em uma iteração precedente, inviabilizando a solução para a turma considerada. Essa última situação pode ocorrer em vista do fato de que não há operação de retorno aos grafos das turmas que já têm seus horários prontos, não se garantindo, assim, que uma solução seja encontrada, mesmo que ela exista.

De forma a incluir outras restrições que comumente aparecem em casos reais, tais como aulas duplas e aulas simultâneas, o autor sugere tratar esses casos com pré-alocação, caso a frequência de tais requisitos seja pequena. Não o sendo, esse modelo não é indicado para tais situações.

3.3.2 Um Exemplo

A figura 3.1 ilustra o grafo de uma turma c_j para um conjunto de 5 horários $(h_1, h_2, \ldots, h_5)$, 3 professores (t_1, t_2 e t_3) e uma matriz de requisitos $R = (r_{ij})$, onde a j-ésima coluna é dada pela tabela 3.1.

Tabela 3.1: Matriz de requisitos

prof/turma	...	c_j	...
t_1	...	2	...
t_2	...	2	...
t_3	...	1	...

É dada, também, uma matriz $T = (t_{ik})$, onde $t_{ik} = 1$ se o professor t_i está disponível no horário h_k e $t_{ik} = 0$, caso contrário.

$$\mathrm{T} \;=\; \begin{array}{c} \\ \\ \\ \\ \end{array} \begin{array}{ccccc} h_1 & h_2 & h_3 & h_4 & h_5 \\ \end{array}$$

$$\mathrm{T} = \begin{array}{|ccccc|c} 1 & 0 & 0 & 1 & 1 & t_1 \\ 0 & 1 & 1 & 0 & 1 & t_2 \\ 0 & 0 & 0 & 1 & 1 & t_3 \end{array}$$

Pela tabela 3.1 vê-se que os professores t_1 e t_2 devem ministrar 2 aulas para a turma, enquanto o professor t_3, apenas uma aula.

O grafo da turma c_j está ilustrado na figura 3.1. Há um arco (h_k, t_i), de capacidade unitária, para cada professor t_i que esteja disponível no horário h_k. A capacidade de cada arco (t_i, t) representa o número de aulas que o professor t_i deve ministrar para a turma c_j. Há um arco (s, h_k), de capacidade unitária, para cada horário h_k no qual a turma deve ter aula.

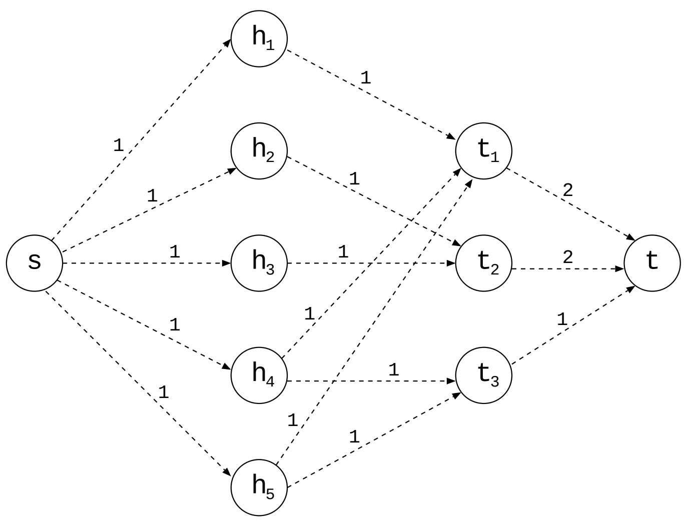

Figura 3.1: Grafo da turma c_j

Na figura 3.2 as linhas cheias representam arcos saturados, representativos da solução de fluxo máximo no grafo da turma c_j, de valor $v = 5$.

Assim, o quadro de horário correspondente da turma c_j é:

$$[\; t_1 \quad t_2 \quad t_2 \quad t_1 \quad t_3 \;]$$

isto é, a turma c_j tem aula com o professor t_1 no primeiro e no quarto horário, com o professor t_2 no segundo e terceiro horário e, finalmente, com o professor t_3 no quinto horário.

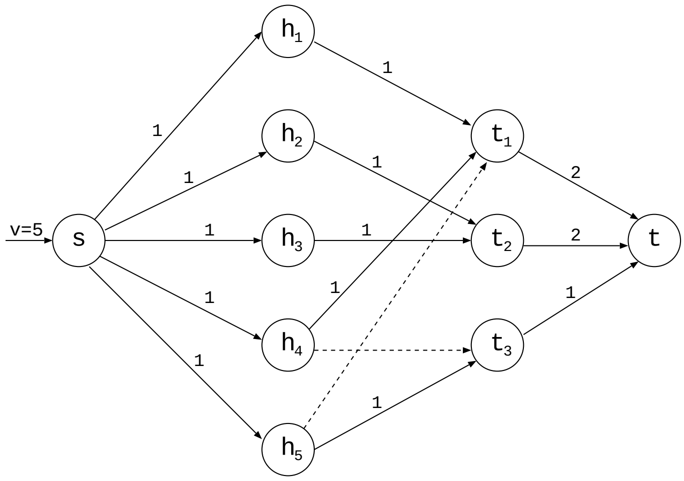

Figura 3.2: Solução de fluxo máximo de s a t

3.3.3 Fixando os Horários

Um procedimento análogo ao anterior consiste em considerar um horário por vez [83]. Cria-se um grafo para cada horário e existe um arco ligando um nó professor a um nó turma, se há uma aula envolvendo o par. O custo de cada arco define a *urgência* da aula. O objetivo, então, é o de encontrar um emparelhamento de peso máximo.

Trata-se, também, de um procedimento heurístico, uma vez que não há garantia de que fazendo-se as alocações de um dado horário, o problema remanescente ainda admitirá solução.

3.4 O Modelo de Chahal e de Werra

Chahal e de Werra [23] aplicaram uma estratégia semelhante à de Even et al. [42] para resolver um problema de programação de horários em uma escola de treinamento de adultos.

O problema abordado consiste de um conjunto de professores, um conjunto de turmas e também um conjunto de matérias a serem ministradas semanalmente. Para cada turma é especificado seu currículo, isto é, a relação das matérias a serem cursadas, com suas respectivas cargas horárias semanais. Para cada professor são listadas as matérias que ele deve lecionar (isto é, seu currículo), bem como suas disponibilidades de horário. Os horários do dia são classificados em duas categorias: (1) horários de prioridade 1, que são os horários normais de funcionamento da escola e (2) horários de prioridade 2, que são os horários nos quais a escola não estaria funcionando regularmente. Os horários de prioridade 2 são indesejáveis e somente devem ser usados em casos extremos. Algumas matérias necessitam, também, de salas especializadas, e existem apenas uma delas para cada tipo de matéria.

No modelo considerado, não se fixa uma matéria a um professor específico, ao contrário do modelo tradicional. Assim, permite-se que o conjunto de aulas de uma dada matéria seja lecionado para uma certa turma por professores diferentes. No entanto, cada um dos professores tem que cumprir a carga horária semanal dessa matéria prevista em suas programações individuais, isto é, o número certo de aulas da matéria em questão constante em seus currículos.

O objetivo do problema é encontrar um conjunto de horários em uma semana tal que:

(a) Cada turma tenha todas as matérias de seu currículo com a freqüência apropriada;

(b) Cada professor ministre todas as matérias de sua competência, cumprindo a carga horária semanal prevista;

(c) Um professor i não seja alocado a um horário no qual não esteja disponível;

(d) O quadro de horário das turmas e professores seja tão compacto quanto possível, isto é, haja o menor número possível de horários vagos entre dois horários de aula;

(e) Horários de prioridade 1 sejam usados tanto quanto possível;

(f) Matérias que requeiram uma mesma sala especializada não sejam alocadas a um mesmo horário;

(g) Um professor não ministre uma mesma matéria por mais de dois horários consecutivos.

Apresenta-se, a seguir, um procedimento heurístico, baseado em fluxo em grafos, para resolver o problema.

3.4.1 O Modelo

Seja um grafo G consistindo de:

(a) um nó c_j para cada turma;

(b) um nó s_u para cada matéria;

(c) um nó t_i para cada professor;

(d) um nó fonte s;

(e) um nó destino t;

(f) um conjunto de arcos descritos conforme a tabela 3.2

Tabela 3.2: Arcos do grafo G e seus parâmetros

Par (x, y)	Multiplicidade $r(x, y)$	Condição
(s, c_j)	Número de aulas no currículo da turma c_j	Para cada turma c_j
(c_j, s_u)	Carga horária semanal da matéria s_u para a turma c_j	Se s_u está no currículo de t_i
(s_u, t_i)	Carga horária semanal da matéria s_u para o prof. t_i	Se s_u está no currículo de t_i
(t_i, t)	Número de aulas no currículo do professor t_i	Para cada professor t_i

Cada par de nós x e y é conectado por uma família de r arcos paralelos orientados, com origem em x e destino em y. $r(x, y)$ representa a multiplicidade (número de arcos) do par x, y.

Para facilidade de entendimento, será eliminado, a princípio, as restrições relativas às matérias que exigem salas especializadas.

Seja G' o grafo obtido pela remoção dos nós s e t, bem como de seus arcos adjacentes em G.

Define-se sobre G', a seguir, um conjunto P de caminhos da forma $\{c_j, s_u, t_i\}$ satisfazendo as seguintes condições:

(a) não existem dois caminhos em P começando no mesmo nó c_j;

(b) não existem dois caminhos em P terminando no mesmo nó t_i.

Tal conjunto P é chamado de *path packing* em G'.

Assim, o problema de construir um quadro de horário usando p horários é e-qüivalente ao de particionar os arcos do grafo G' em *path packing*'s $P_1, P_2, \ldots, P_p$. Em vista das hipóteses consideradas tem-se que, para cada matéria s_u, o número de arcos chegando a t_i é igual ao número de arcos saindo de t_i. Por causa da restrição de indisponibilidade dos professores, os *path packing*'s $P_1, P_2, \ldots, P_p$ devem satisfazer:

(c) se o professor t_i não estiver disponível no horário k, então nenhum caminho em P_k termina no nó t_i.

Conforme relatado anteriormente (vide Even et al. [42]), se nenhuma turma (ou professor) tem mais do que p aulas em seu currículo, então um quadro de horário usando p horários pode ser encontrado, com a condição de que não haja restrições de disponibilidade (Problema básico de programação de horários, o qual não inclui o atendimento aos requisitos (f) e (g), relacionados na introdução da seção 3.4). Em tal situação, a construção de uma partição dos arcos de G' em *path packing*'s pode ser realizada construindo-se, consecutivamente, um conjunto de fluxos compatíveis em G (o qual contém os nós s e t), onde cada caminho $\{c_j, s_u, t_i\}$ torna-se um caminho da forma $\{s, c_j, s_u, t_i, t\}$.

3.4.2 O Algoritmo

No caso geral, em que há restrições de indisponibilidade, a construção de um quadro de horário segundo o modelo anterior é feita usando-se um procedimento heurístico.

Primeiramente, os horários são divididos em duas categorias: horários com alta prioridade e horários com baixa prioridade. A seguir, os horários são ordenados considerando-se primeiro os horários de prioridade mais alta. Isto é, eles são ordenados de acordo com o número de professores disponíveis naquele horário, de forma que o primeiro horário da lista é aquele com o menor número de professores disponíveis. A seguir, consideram-se os horários de mais baixa prioridade, ordenando-os da mesma forma. Para ambas as categorias, em caso de empate, considera-se um horário qualquer. Então, usando esta ordem, constrõem-se quadros de horário, horário por horário, determinando-se um *path packing* P_k para cada horário k, consecutivamente. Removem-se, a seguir, os arcos da forma c_j, s_u e s_u, t_i que foram usados e repete-se a construção para o próximo horário.

A alocação das aulas a um horário k é determinada como segue. Para cada horário k, o grafo $G'(k)$ para ser considerado é detalhado na tabela 3.3.

Tabela 3.3: Grafo $G'(k)$ usado para o horário k

Arco (x, y)	Capacidade		Custo	
(s, c_j)	1	Se todas as aulas da turma c_j ainda não foram alocadas	$\alpha \times L_{c_j} - q_j$	Se a turma c_j está disponível no horário k
	0	caso contrário	∞	caso contrário
(c_j, s_u)	1	Se todas as aulas da matéria s_u para turma c_j ainda não foram alocadas	∞	Se s_u foi alocada à turma c_j nos dois horários anteriores
	0	caso contrário	0	caso contrário
(s_u, t_i)	1	Se todas as aulas da matéria s_u do professor t_i ainda não foram alocadas	0	
	0	caso contrário		
(t_i, t)	1	Se todas as aulas do professor t_i ainda não foram alocadas	$\beta \times L_{t_i} - l_i$	Se o professor t_i está disponível no horário k
	0	Caso contrário	∞	caso contrário

$G'(k)$ tem os mesmos nós de G. Para facilidade de entendimento, eliminamos o índice k para referenciar os elementos de $G'(k)$.

Um *path packing* P é construído procurando-se o fluxo máximo, com custo mínimo, de s a t. O custo dos arcos (s, c_j) e (t_i, t) é função dos graus de liberdade L_{c_j} e L_{t_i}, respectivamente, definidos conforme abaixo:

$$L_{t_i} = QHD(t_i) - QAR(t_i)$$

onde $QHD(t_i)$ é a quantidade de horários disponíveis do professor t_i e $QAR(t_i)$ é a quantidade de aulas que ainda faltam para serem alocadas ao professor t_i.

Analogamente define-se o grau de liberdade da turma c_j. No contexto considerado pelos autores, as turmas estão sempre disponíveis.

Esses custos visam priorizar turmas e professores aos quais estão associados pequenos graus de liberdade. α e β são parâmetros do programa e seus valores padrão são $\alpha = \beta = 1$.

Para atender aos requisitos de compacidade do quadro de horário de professores e turmas procede-se como segue.

Na alocação de aulas ao horário k dá-se uma prioridade mais alta a um professor que já tenha se envolvido em uma aula no horário $k+1$ (ou no horário $k-1$). Para cumprir este objetivo, a solução adotada pelos autores é a de reduzir o custo dos arcos (t_i, t) por uma constante l_i. Uma operação semelhante envolvendo a constante q_j é aplicada para a turma c_j.

Obviamente, o fluxo máximo de s a t em $G'(k)$ corresponde ao maior conjunto possível de alocações de professores t_i a matérias s_u e a turmas c_j no horário k.

Após a obtenção de tais fluxos nos horários de mais alta prioridade, o procedimento é repetido para os horários remanescentes.

A aplicação deste procedimento, que é heurístico, devido à indisponibilidade dos professores e dependente da ordem na qual são feitas as alocações, resultará em um quadro de horário aceitável ou então, em um quadro de horário incompleto, com poucas aulas não alocadas. Quando esta última situação ocorre, faz-se necessário um ajuste manual no quadro de horário, o qual é feito utilizando-se de facilidades oferecidas pelo sistema desenvolvido.

Para tratar de restrições envolvendo matérias que requerem salas especializadas, introduz-se em $G'(k)$ arcos de capacidade unitária para cada uma dessas matérias s_u. Mais especificamente, divide-se cada nó s_u em dois nós s'_u e s''_u, ligando-os por um arco (s'_u, s''_u) de capacidade unitária, já que, conforme foi especificado, existe apenas uma sala especializada por matéria que demande esse tipo de recurso.

3.5 O Modelo de Schaerf

Descreve-se, a seguir, o modelo tratado por Schaerf [93], o qual é aplicado a uma escola italiana de nível médio.

O algoritmo apresentado é uma adaptação do método de Busca Tabu [56] e pode ser usado de forma interativa ou não. Isto é, permite-se que, durante a fase de busca, o usuário modifique manualmente o quadro de horário bem como as restrições do problema.

3.5.1 Descrição do Problema

Existem n turmas $c_1, \ldots, c_n$, m professores $t_1, \ldots, t_m$ e p horários $1, 2, \ldots, p$. É conhecida a *matriz de requisitos* $R_{m \times n}$, de elementos inteiros não-negativos r_{ij}, os quais representam o número de aulas que o professor t_i precisa ministrar para a turma c_j.

Para tratar da indisponibilidade de professores e turmas são introduzidas duas matrizes binárias $T_{m \times p}$ e $C_{n \times p}$ tais que $t_{ik} = 1$ ($c_{jk} = 1$) se o professor t_i (turma c_j) está disponível no horário k e $t_{ik} = 0$ ($c_{jk} = 0$), caso contrário.

A variável de decisão é x_{ijk}, a qual assume o valor 1 se o professor t_i é alocado à turma c_j no horário k e 0, caso contrário.

A formulação matemática do problema considerado é a seguinte:

$$\begin{aligned}
\text{(PBHE)} \qquad & \text{Encontre } x_{ijk} && (3.1)\\
\text{s.a:} \qquad & \sum_{i=1}^{m} x_{ijk} \leq c_{jk} \quad \forall j, k && (3.2)\\
& \sum_{j=1}^{n} x_{ijk} \leq t_{ik} \quad \forall i, k && (3.3)\\
& \sum_{i=1}^{p} x_{ijk} = r_{ij} \quad \forall i, j && (3.4)\\
& x_{ijk} \in \{0, 1\} \quad \forall i, j, k && (3.5)
\end{aligned}$$

De forma a tratar de instâncias reais do problema, o autor considerou, também, as restrições adicionais especificadas a seguir.

Nas escolas italianas de nível médio abordadas, as aulas são realizadas em, no máximo, 6 horários consecutivos diários, de segunda a sábado, totalizando, portanto, 36 horários semanais. Cada turma tem cerca de 27 a 32 aulas semanais e as escolas requerem que as turmas tenham aulas nos horários iniciais de cada dia. Por outro lado, de acordo com a escola, as turmas podem estar indisponíveis no último horário de cada dia ou no último horário do sábado. Assim, considera-se a restrição abaixo, a qual modela a situação na qual cada turma, para um dado conjunto de horários, precisa necessariamente estar envolvida em pelo menos uma aula:

$$\sum_{i=1}^{m} x_{ijk} \geq d_{jk} \quad \forall j, k \tag{3.6}$$

onde $D_{n \times p}$ é uma matriz binária tal que $d_{jk} = 1$ se a turma c_j precisa ter aula no horário k e $d_{jk} = 0$, caso contrário.

Uma outra restrição considerada pelo autor, que aparece em casos reais, é a seguinte: alguns pares de aulas requerem ser alocados simultaneamente. Mais especificamente, existe uma quádrupla de índices $\langle i_1, i_2, j_1, j_2 \rangle$ tais que todas as aulas do professor t_{i_1} para a turma c_{j_1} precisam ser simultâneas às aulas do professor t_{i_2} para a turma c_{j_2}. Chamando de S tal quádrupla de índices, a restrição anterior pode ser modelada como segue:

$$x_{i_1 j_1 k} - x_{i_2 j_2 k} = 0 \quad \forall \langle i_1, j_1, i_2, j_2 \rangle \in S, \ \ \forall k = 1, \ldots, p \tag{3.7}$$

Estas restrições aparecem para tratar, por exemplo, de turmas bilíngües, isto é, turmas que cursam duas línguas estrangeiras. Essas matérias exigem laboratórios específicos, os quais não comportam toda a turma. Assim, a solução é dividir a turma em dois grupos. Enquanto um dos grupos de estudantes cursa uma língua estrangeira com um professor, o outro grupo tem que ter aula, no mesmo horário, da outra língua estrangeira com outro professor.

3.5.2 A Função Objetivo

A função objetivo é avaliada com base em uma série de indicadores relativos a professores e turmas. Para cada professor especifica-se o número máximo e mínimo de aulas por dia, os horários de ensino não desejados, bem como seu *status* (Trata-se de uma medida de prestígio entre professores. Por exemplo, um professor mais renomado ou um que trabalha há mais tempo na escola tem mais prestígio que um outro que seja menos renomado ou que tenha menos tempo de casa). Para cada turma estabelece-se o prédio onde ocorrerão as aulas, bem como a sala na qual a maioria das aulas será ministrada. Para cada par professor-turma define-se o número máximo de aulas por dia, a duração de cada aula (isto é, o número de aulas dadas consecutivamente em um dia por um mesmo professor para uma mesma turma, pelo menos uma vez na semana).

Mais especificamente, a função objetivo é um somatório, sobre todos os professores, das componentes, com seus respectivos pesos, listadas na tabela 3.4, as quais referem-se à alocação de um único professor. A coluna referente ao valor das penalidades, refere-se ao peso padrão dos parâmetros de penalidade.

Tabela 3.4: Componentes da função objetivo, com seus respectivos pesos padrão

Componentes	**Valor w_i da penalidade**	**Descrição**
Buracos	1	Número de horários ociosos entre dois horários de aula na agenda diária do professor
Quebras	6	Número de vezes em que se ministrou duas aulas não consecutivas para uma mesma turma em um mesmo dia, tendo-se, nesse período ocioso, ministrado pelo menos uma aula para outras turmas
Abaixo-limite-diário	4	Número de vezes em que a carga horária diária do professor ficou abaixo da quantidade mínima especificada para ele
Acima-limite-diário	3	Número de vezes em que a carga horária diária do professor superou a quantidade máxima especificada para ele
Excesso-aulas	6	Número de vezes em que se tem mais aulas para a mesma turma no mesmo dia que o máximo especificado pelo par professor-turma
Indesejabilidades	3	Quantidade de aulas em horários indesejados. A penalidade para esta característica é normalizada com base no número de solicitações de cada professor e seu *status*
Aulas-consecutivas	10	Número de vezes em que não é possível, para o par professor-turma, pelo menos uma vez na semana, ter aulas consecutivas, quando requerido
Deslocamentos	5	Número de vezes em que o professor tem que mudar de prédio entre 2 horários consecutivos de aula

De forma a calcular corretamente algumas das componentes anteriores da função objetivo (isto é, buracos, quebras, etc.), as indisponibilidades dos professores são divididas em duas espécies: as *fora da escola* e as *dentro da escola*. As indisponibilidades do tipo *fora da escola* são aquelas nas quais o professor não pode ensinar porque está ausente da escola; enquanto a do tipo *dentro da escola* representa a situação na qual o professor está na escola, porém está indisponível para ensinar (como, por exemplo, envolvido em uma atividade administrativa).

A indisponibilidade do tipo *dentro da escola* também pode ser usada (em conjunto com outras indisponibilidades) para modelar pré-alocações de aulas, as quais não estão explicitamente incluídas no modelo considerado.

3.5.3 Representação do Problema

Um quadro de horário é representado como uma matriz $Q_{m \times p}$ de valores inteiros, onde em cada linha i de Q representa-se a alocação semanal do professor t_i. Cada elemento q_{ik} contém o nome da turma que o professor t_i dá aula no horário k. Valores maiores que o número n de turmas representam atividades especiais, tais como estar à disposição para atividades de ensino esporádicas, encontro com pais de alunos e períodos de assistência aos alunos.

Segundo o autor, esta representação foi escolhida porque ela permite a definição de tipos simples e naturais de movimentos (conforme especificado na seção 3.5.3.2), possibilitando uma navegação mais eficaz pelo espaço de pesquisa. Além disso, essa representação torna mais fácil o levantamento de várias características da função objetivo, a qual está mais baseada na alocação de professores. Além de tudo, esta é a representação com a qual trabalham as pessoas que elaboram manualmente quadros de horários e, portanto, ela é indicada para a elaboração interativa das tabelas de horário.

A tabela 3.5 mostra um fragmento de uma tabela real de horário. Para facilidade de entendimento, na representação externa os números das turmas são convertidos para seus nomes ('1A', '2A', etc.). O símbolo '$\langle\rangle$' representa horários nos quais o professor está à disposição para possíveis substituições em atividades de ensino. A tabela 3.5 também mostra a indisponibilidade de professores, sendo que '—' representa a indisponibilidade do professor por estar ausente da escola e '**' por estar na escola envolvido em outras atividades.

Os nomes das turmas em minúsculo representam turmas fictícias, as quais são usadas para modelar alocações simultâneas (restrições 3.7). Por exemplo, na tabela 3.5, 4ª feira, segundo horário, o professor 13 é um professor de inglês que ensina a primeira língua estrangeira para a turma bilíngüe '2E', sendo assistido pelo professor de francês (professor número 14), o qual cuida dos estudantes que têm francês como primeira língua estrangeira. Portanto, quando o professor 13 ensina para a turma '2E', o professor 14 ensina para a turma fictícia '2e'. Inversamente, a segunda língua estrangeira é ensinada para a turma '2E' pelo professor 17, de francês, o qual é assistido pelo professor 13, de inglês, para os estudantes que tiveram francês como primeira língua estrangeira (e inglês, como segunda).

Tabela 3.5: Um fragmento de um quadro de horário

Dia	**2ª f.**						**3ª f.**						**4ª f.**						...
Hor.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
Prof.																			
3			3A	3A	⟨⟩	2A					⟨⟩	2A	—	—	—	—	—	—	...
4	1A	1A	4A	5A	5A	—	1A	4A	4A	⟨⟩		—	⟨⟩	1A				—	...
5	2B	2B	2B	⟨⟩	1B		2B	3B						2B	⟨⟩	⟨⟩	3B	1B	...
6	1B	1B	4B	4B			⟨⟩	1B	**	4B	5B		—	—	—	—	—	—	...
7	1C	⟨⟩	3C	2C					⟨⟩	3C	3C	2C	3C	⟨⟩	2C	2C	2C		...
8	5C	4C	4C	⟨⟩	1C					5C	1C	1C	4C	4C	⟨⟩	5C			...
9	1D	⟨⟩	⟨⟩	2D	2D						2D	2D	1D	1D					...
10	—	—	—	—	—	—	2E	2E	3D	3D	5D		5D	5D	2E	3D			...
11	3A	2A						5A	3A	2A	1A	4A	1A	⟨⟩	2A				...
12	—	—	—	—	—	—		2C	3C	2B	1B	3B	1C	3C		3B	1B	2B	...
13		2d	2D	1D	3D	2e				1D	3D		3D	2E	2d	2e			...
14	—	—	—	—	—	—					3A	4a		2e	⟨⟩	2A	1A	3A	...
15				2B	4B					1B	3B	5B	2B	5B	4B				...
16	⟨⟩	⟨⟩	5C	1C	4C		—	—	—	—	—	—	2C	1C	3C				...
17	—	2D	2d	3D	1D	2E	—	—	—	—	—	—	—		2D	2E	5D		...

3.5.3.1 Introduzindo Inviabilidades na Função Objetivo

Quadros inviáveis de horário também são incluídos no espaço de pesquisa do algoritmo. A função descrita na seção 3.5.2 é acrescida de um outra componente, a qual leva em consideração o número de inviabilidades. Em particular, contam-se: (1) o número de vezes que dois professores ensinam para a mesma turma em um mesmo horário e/ou o número de vezes que uma turma está a descoberto, isto é, sem nenhum professor alocado; (2) o número de vezes que uma alocação simultânea deixa de ocorrer e (3) o número de vezes que um professor ensina (ou uma turma é ensinada) quando está indisponível.

A possibilidade de que um professor ensine simultaneamente para duas ou mais turmas é rejeitada automaticamente pela representação escolhida.

O peso ω dado para as inviabilidades ($\omega = 20$) é fixado em um valor superior ao de todas as outras componentes. Entretanto, conforme será relatado na seção 3.5.4.1, de forma a assegurar uma melhor navegação no espaço de pesquisa, permite-se a variação de tal peso durante a fase de busca.

3.5.3.2 Tipos de Movimentos e Estrutura da Vizinhança

Considera-se dois tipos de movimento. O primeiro, chamado de *movimento elementar*, consiste na simples troca de dois valores distintos de uma dada linha. Isto é, as aulas de um professor t_i em dois diferentes horários p_1 e p_2 são permutadas entre si ou, no caso em que um valor é zero, uma aula é transferida para um horário diferente. Tal tipo de movimento é identificado pela tripla $\langle t_i, p_1, p_2 \rangle$. Assume-se, sem prejuízo de generalidade, que $p_1 > p_2$, o que faz com que um movimento seja o único movimento reverso de si mesmo.

Movimentos elementares aplicados a quadros viáveis de horário geralmente criam inviabilidades ao alocar dois professores a uma mesma turma. Por esta razão, considera-se, também, outro tipo mais complexo de movimento: o chamado *movimento duplo*. O movimento duplo consiste de um par de movimentos elementares e é aplicado quando o primeiro movimento provoca uma inviabilidade. Neste caso, executa-se um segundo movimento, o qual visa a restaurar a viabilidade (no todo ou em parte) perdida pela execução do primeiro movimento. Esse segundo movimento consiste na troca da aula que está em conflito com aquela que foi inserida, em virtude do primeiro movimento. Se o primeiro movimento elementar não criar inviabilidade, então não há um segundo movimento e o movimento duplo se reduz a um movimento elementar.

Os movimentos elementares e duplos estabelecem uma relação de vizinhança, permitindo a navegação pelo espaço de pesquisa. Através dela é possível, a partir de um certo quadro de horário, atingir um outro quadro de horário em uma quantidade finita de movimentos. Pode-se provar que esse número de movimentos necessários para ir de um quadro a outro está limitado ao número total de aulas (isto é, à soma de todos os elementos da matriz R de requisitos).

3.5.4 Aplicação da Busca Tabu

O algoritmo proposto é basicamente um algoritmo de Busca Tabu, com a vizinhança gerada por movimentos elementares, sendo intercalado com uma fase do Método RNA (vide seção 2.1.3), usando movimentos duplos. O autor utiliza a idéia de alternar tipos diferentes de movimentos, conforme sugerido por Glover et al. [58], como um melhoramento tático para a Busca Tabu.

Detalhando, a solução inicial é obtida alocando-se, aleatoriamente, as aulas de cada professor, respeitando a matriz de requisitos (podendo ser obtida, também, como resultado da aplicação do algoritmo, no caso de elaboração interativa do quadro de horário).

Inicia-se, a seguir, uma fase do algoritmo RNA, a qual cessa quando nenhum melhoramento ocorre em um certo número, $RNAmax$, de iterações. Neste ponto, o algoritmo de Busca Tabu entra em ação, sendo interrompido quando um certo número, $BTmax$, de iterações sem melhora no valor da função objetivo é atingido.

A seguir, todo o processo (BT + RNA) é repetido novamente considerando, agora, como ponto de partida a melhor solução obtida. Esse procedimento é repetido até que não ocorra nenhum melhoramento em um certo número, $CICLOS$, de vezes.

O RNA é usado com dois diferentes propósitos: 1^o, para gerar a solução inicial para o Algoritmo BT. Na verdade, segundo o autor, o uso de BT começando de uma solução randômica consome muito tempo de processamento e o RNA, ao invés disso, representa um método rápido para gerar uma boa solução inicial (a qual geralmente ainda contém inviabilidades). Em 2^o lugar, após o BT não ter produzido nenhum melhoramento no valor da função objetivo em um certo número de iterações, ele chama a execução do RNA sobre a melhor solução encontrada. A razão para isto é dupla: por um lado, o RNA (usando movimentos duplos) pode encontrar melhoramentos que o BT não está habilitado a encontrar naquele estágio. Por outro, o RNA com movimentos duplos, mesmo não conseguindo melhorar a solução (o que ocorre com freqüência), faz movimentos laterais importantes. Portanto, ele *embaralha* a solução antes que o algoritmo BT comece novamente a tentar melhorá-la. A idéia fundamental por trás de tal procedimento é que após BT ter trabalhado sem sucesso por um dado número de iterações, o RNA é útil para modificar a solução de forma que o algoritmo BT possa começar em uma direção diferente.

Uma descrição sumária do algoritmo, onde *delta* representa a variação no valor da função objetivo, é apresentada na figura 3.3. Discute-se nas subseções seguintes algumas de suas características.

```
procedimento RNA+BT;
Considere um quadro de horário inicial;
Inicialize as variáveis de pesquisa;
para (c = 1, ..., CICLOS) faça
    SemMelhora ← 0;
    enquanto (SemMelhora < RNAmax) faça                {Executa a fase RNA}
        SemMelhora ← SemMelhora + 1;
        MovimentoDuplo ← MovimentoDuploRandomico;
        se (delta(MovimentoDuplo) ≤ 0) então
            AtualizeQuadroCorrente(MovimentoDuplo);
            se (delta(MovimentoDuplo) < 0) então SemMelhora ← 0 ;
        fim-se;
    fim-enquanto;
    SemMelhora ← 0;
    enquanto (SemMelhora < BTmax) faça                 {Executa a fase BT}
        SemMelhora ← SemMelhora + 1;
        MelhorMovimento ← MovimentoElementarRandomico;
        MovimentoDelta ← delta(MelhorMovimento, PesoDinamico);
        para (todo movimento na VizinhancaLegal(QuadroCorrente)) faça
            se (delta(Movimento, PesoDinamico) < MelhorDelta
                e (não Tabu(Movimento) ou Aspira(Movimento))
                e ((delta(Movimento, PesoDinamico) < 0 ou
                    não Semi–ilegal(Movimento)) então
                MelhorMovimento ← Movimento;
                MelhorDelta ← delta(Movimento, PesoDinamico);
            fim-se;
        fim-para;
        AtualizeQuadroCorrente(MelhorMovimento);
        AtualizeListaTabu(MelhorMovimento);
        se (Melhor(QuadroCorrente)) então
            MelhorQuadro ← QuadroCorrente;
            SemMelhora ← 0;
        fim-se;
        se (AtualizeIteracao) então AtualizePesoDinamico;
    fim-enquanto;
    QuadroCorrente ← MelhorQuadro;
fim-para;
Retorne MelhorQuadro;
fim RNA+BT;
```

Figura 3.3: Algoritmo de Busca Tabu proposto por Schaefer

3.5.4.1 Relaxação Adaptativa

Durante a fase RNA o peso da componente de inviabilidade é fixado no valor $\omega = 20$, o qual é superior a todos os demais pesos envolvidos na função objetivo. Entretanto, durante o estágio BT, tal peso é ajustado dinamicamente da seguinte forma (tal como proposto em Gendreau et al. [51]): Para cada uma das 3 fontes de inviabilidade (vide seção 3.5.3.1), a saber (1) sobreposição de professores e/ou turmas descobertas, (2) simultaneidade de aulas e (3) indisponibilidades, multiplica-se ω por um número real $\alpha_i \;\; \forall i = 1, 2, 3$, o qual varia de acordo com o seguinte esquema:

1. No início da pesquisa fixa-se $\alpha_i = 1 \quad \forall i = 1, 2, 3$
2. A cada q movimentos ($q = 10$ no experimento realizado):
   - 2.1 Se todas as q soluções visitadas são viáveis então $\alpha_i \leftarrow \alpha_i/\gamma$
   - 2.2 Se todas as q soluções visitadas são inviáveis então $\alpha_i \leftarrow \alpha_i \times \gamma$
   - 2.3 Se nem todas as soluções são viáveis então α_i permanece inalterado

Ao contrário de Gendreau et al. [51], no qual o parâmetro real γ é fixado determinísticamente em 2, o autor seleciona γ randomicamente no intervalo $[1.8, 2.2]$ à cada iteração da Busca Tabu. O autor preferiu randomizá-lo para evitar que razões determinísticas entre componentes diferentes da função objetivo pudessem conduzir a busca.

O valor de cada α_i é, no entanto, limitado por duas constantes, a saber, $\alpha_{i_{min}}$ e $\alpha_{i_{max}}$. Assim, se α_i superar $\alpha_{i_{max}}$ então faz-se $\alpha_i \leftarrow \alpha_{i_{max}}$. De forma similar, se ele atingir um valor abaixo de $\alpha_{i_{min}}$, ele é fixado em $\alpha_{i_{min}}$. O valor de α_i é limitado de forma a preveni-lo de atingir um valor muito alto (ou, respectivamente, muito baixo) após uma sequência longa de soluções inviáveis (respectivamente, viáveis). Se tal atitude não fosse tomada, α_i poderia demorar a voltar a assumir valores comparáveis com os outros termos da função objetivo. No experimento realizado pelo autor, as três constantes $\alpha_{i_{max}}$ são fixadas em valores diferentes (no caso, $\alpha_{1_{max}} = 1$, $\alpha_{2_{max}} = 10$, $\alpha_{3_{max}} = 3$ e $\alpha_{i_{min}} = 0.01 \;\; \forall i = 1, 2, 3$) para que não haja situação estável na qual fontes diferentes de inviabilidade coexistam.

3.5.4.2 O Gerenciamento da Lista Tabu

Emprega-se uma lista tabu de tamanho variável. Quando um movimento é realizado, ele é inserido na lista tabu juntamente com o número da iteração I na qual ele está entrando na lista. O número I é randomicamente selecionado entre dois parâmetros dados I_{min} e I_{max}, sendo $I_{min} \leq I_{max}$. À cada vez que um movimento é inserido na lista, o valor I de todos os movimentos da lista é atualizado (no caso, diminuído) e quando ele atinge o valor zero, o movimento é removido da lista.

Forçam-se dois mecanismos tabu diferentes, embora faça-se uso de uma única lista tabu. O primeiro mecanismo estabelece que um movimento $m = \langle t_i, p_1, p_2 \rangle$ é tabu se a tripla exata $\langle t_i, p_1, p_2 \rangle$ aparece na lista tabu. O segundo mecanismo, o qual tem o efeito de uma memória de curto prazo, estabelece que m é tabu se ou (t_i, p_1) ou (t_i, p_2) aparecem entre os últimos I_{st} elementos inseridos (com $I_{st} \leq I_{min}$) da lista tabu. Assim, as atividades do professor t_i nos horários p_1 e p_2 não podem ser permutadas por iterações, assim como também não podem ser permutadas separadamente por outras atividades durante I_{st} iterações.

3.5.4.3 Função de Aspiração

Devido ao uso da relaxação adaptativa, o valor de uma dada solução depende dos valores correntes dos α's. De forma a ter um critério de aspiração com significado, a função de aspiração foi definida baseando-se nos valores da função objetivo não relaxada, isto é, aquela na qual $\alpha_i = 1 \;\; \forall i = 1, 2, 3$.

Um movimento tabu é aceito somente se ele melhorar o valor da solução ótima corrente. No caso, faz-se $A(f(s)) = f(s^\star) - 1$ para todas as soluções s, onde $s^\star$ é a solução ótima corrente.

3.5.4.4 Exploração da Vizinhança

O tamanho da vizinhança de uma solução s pode ser calculado com base na seguinte expressão:

$$|N(s)| = \frac{m \times p \times (p-1)}{2}$$

a qual é o número de professores vezes o número de pares desordenados de horários distintos. Entretanto, movimentos que permutam duas aulas idênticas não alteram o quadro de horário. Desta forma, pode-se considerar este tipo de movimento como *ilegal*, excluindo-o da vizinhança. Este fato diminui o número de movimentos legais por um fator que depende do número n de turmas e da matriz R de requisitos (em casos práticos, segundo o autor, esse fator é de aproximadamente 30%).

Devido à relaxação adaptativa, é muito difícil encontrar uma forma de predizer o movimento mais promissor. Por esta razão, a vizinhança inteira é analisada a cada iteração. Contudo, há certos movimentos, como por exemplo, permutar um horário sem aula com um horário em que o professor está à disposição para possível substituição ("$\langle\rangle$") que, em muitas situações, não afetam realmente a estrutura da solução. Por outro lado, tais movimentos podem produzir um pequeno melhoramento na função objetivo (como, por exemplo, preenchimento dos buracos das agendas dos professores). Pela razão anterior, o autor introduz os chamados movimentos *semi-ilegais*. Consideram-se movimentos semi-ilegais aqueles que permutam dois valores representativos de turmas não reais (isto é, quando representar uma situação na qual o professor está sem aula ou prestando assistência ou à disposição para possível substituição) e aqueles que movem turmas entre dois horários nos quais os professores não estão requisitados a estar na escola. A restrição que impõe-se aos movimentos semi-ilegais é que eles podem ser feitos somente se melhorarem estritamente o valor da solução ótima corrente. Em casos práticos, segundo o autor, o número de movimentos semi-ilegais é cerca de 10 a 15% do número de movimentos legais.

3.6 O Modelo de Costa

Descreve-se, a seguir, o modelo tratado por Costa [28], o qual procura englobar o maior número possível de restrições que aparecem nas mais diferentes escolas secundárias.

No modelo considerado assume-se que as aulas têm duração previamente definida, podendo ser simples ou duplas. Desta forma, o problema pode ser enquadrado como o de programação de horários de cursos.

3.6.1 Descrição do Problema

Consideram-se os seguintes elementos para definir o problema:

(a) Um conjunto $T = \{t_1, t_2, \ldots, t_m\}$ de m professores;

(b) Um conjunto $C = \{c_1, c_2, \ldots, c_n\}$ de n turmas, onde considera-se que uma turma é um grupo de estudantes que têm o mesmo currículo;

(c) Um conjunto $S = \{s_1, s_2, \ldots, s_{ns}\}$ de ns matérias;

(d) Um conjunto $R = \{r_1, r_2, \ldots, r_{nr}\}$ de nr salas especializadas;

(e) Um subconjunto $SS = \{ss_1, ss_2, \ldots, ss_{nss}\}$ de S, o qual contém todas as matérias que requerem salas especializadas. Tais matérias são chamadas de *matérias especializadas*. Cada matéria especializada ss_j está associada a um certo tipo de sala especializada na qual ss_j deve ser ensinada;

(f) Um conjunto $B = \{b_1, b_2, \ldots, b_{nb}\}$ de nb prédios distantes. Considera-se que dois prédios são distantes se não há tempo suficiente para mover-se de um prédio a outro entre dois horários consecutivos;

(g) Um conjunto $CO = \{co_1, co_2, \ldots, co_{nco}\}$ de nco sessões de cursos. Cada sessão de curso co_j é definida por:

- um conjunto de professores;

- um conjunto de turmas;

- um conjunto de matérias;

- um número de aulas simples;

- um número de aulas duplas;

- um prédio onde o curso se realiza;

- o número de salas especializadas necessárias;

- a informação se há a necessidade de que alguma sala especializada precisa estar preparada antes de cada aula do curso (ou arrumada após o final da aula);

Um curso é denominado simples se as turmas a ele vinculadas não se misturam para ter aula em conjunto. Caso contrário, ele é denominado curso agrupado. Um curso agrupado corresponde à reunião de estudantes de diferentes turmas, os quais são reagrupados de forma a ter, ao mesmo tempo, aulas diferentes dadas por professores diferentes. Isto ocorre, por exemplo, com as aulas de educação física: algumas turmas são agrupadas em certos horários, subdivididas a seguir e cada grupo faz uma modalidade diferente de educação física.

Ao conjunto CO está associado o conjunto $A = \{a_1, a_2, \ldots, a_{na}\}$ de todas as aulas (simples e/ou duplas) a serem ministradas.

(h) Um conjunto $H = \{h_1, h_2, \ldots, h_p\}$ de p horários nos quais as aulas precisam ser ministradas. Considera-se que $p = nd \times hd$, onde nd é o número total de dias da semana em que há atividade de ensino e hd é o número de horários diários;

(i) Um conjunto $\{H_{a_1}, \ldots, H_{a_{na}}\}$, onde H_{a_k} é o conjunto dos horários durante os quais a aula a_k está pré-alocada. $|H_{a_k}| = 1$ significa que a k-ésima aula está pré-alocada apenas no horário dado. Quando não há a necessidade de pré-alocar a aula a_k, considera-se que $H_{a_k} = H$;

(j) Um conjunto $\{HT_{t_1}, \ldots, HT_{t_m}\}$, onde HT_{t_i} representa o conjunto dos horários durante os quais o i-ésimo professor não está disponível para ensinar ($HT_{t_i} \subset H$);

(k) Um conjunto $\{HC_{c_1}, \ldots, HC_{c_n}\}$, onde HC_{c_j} representa o conjunto dos horários durante os quais a j-ésima turma não está disponível para estudar ($HC_{c_j} \subset H$);

(l) Um conjunto $\{HS_{s_1}, \ldots, HS_{s_{ns}}\}$, onde HS_{s_j} representa o conjunto dos horários durante os quais o ensino da j-ésima matéria não é desejado (por alguma razão pedagógica) ($HS_{s_j} \subset H$);

(m) Uma matriz $MR_{nr \times p}$ de números inteiros não-negativos, onde cada elemento mr_{ik} representa o número de salas especializadas do tipo i disponíveis no horário k.

As seguintes hipóteses são assumidas:

(H1) As diferentes aulas simultâneas, ministradas aos cursos agrupados, são lecionadas em um mesmo prédio;

(H2) Em cada horário, o número de salas comuns disponíveis é ilimitado;

(H3) Para cada curso, o número de aulas, bem como suas durações (aulas simples e/ou duplas) é conhecido a priori;

(H4) Não há a necessidade de repetir alguns cursos durante a semana se eles envolverem um número elevado de estudantes.

Baseado nas definições e hipóteses consideradas anteriormente, o problema de horário considerado consiste em alocar cada uma das aulas de cada curso a um horário, de forma que as seguintes restrições sejam satisfeitas:

(1) *Sobreposição de professores*: Um professor não pode lecionar mais do que uma aula em um dado horário;

(2) *Sobreposição de turmas*: Uma turma não pode ter mais do que uma aula em um dado horário;

(3) *Sobreposição de salas especializadas*: Em um dado horário, o número de salas especializadas requeridas não pode ser maior que o número de salas especializadas disponíveis;

(4) *Restrições de pré-alocação*: Certas aulas precisam ser pré-alocadas a um conjunto específico de horários;

(5) *Indisponibilidade de professores*: Uma aula não pode ser alocada a um horário durante o qual um professor não esteja disponível;

(6) *Indisponibilidade de turmas*: Uma aula não pode ser alocada a um horário durante o qual uma turma não esteja disponível;

(7) *Indisponibilidade de matérias*: Uma aula de uma matéria não deve ser ministrada em um horário durante o qual o ensino dela não é permitido. Por exemplo, os elaboradores de tabelas de horário procuram não marcar aulas de matérias 'importantes' nos últimos horários diários;

(8) *Restrições geográficas*: É necessário que haja tempo suficiente para deslocar de um prédio a outro. Assim, duas aulas ministradas em dois locais distantes e envolvendo um mesmo professor ou uma mesma turma não podem ser alocadas em horários consecutivos;

(9) *Restrições de compacidade*: Cada professor deseja uma agenda de aulas com o menor número possível de *buracos* e aulas isoladas. O quadro de horário das turmas é, a princípio, compacto, uma vez que o número de horários disponibilizados para cada turma é, em geral, exatamente igual ao número de aulas constantes em seu currículo (gerando a inexistência de 'buracos' na tabela de horário de aulas das turmas);

(10) *Restrições de distribuição*: As várias aulas de uma mesma matéria para uma dada turma precisam ser espalhadas tão uniformemente quanto possível ao longo da semana;

(11) *Gerenciamento de aulas duplas*: Os dois horários de uma aula dupla precisam ser alocados em dois horários consecutivos;

(12) *Restrições de precedência*: O quadro de horário de uma turma não pode conter algumas seqüências específicas de matérias. Na prática, normalmente evita-se que uma turma tenha aulas seguidas de duas matérias consideradas 'difíceis';

(13) *Restrições de preparação de salas especializadas*: Uma sala especializada não está disponível quando ela está sendo preparada para uma aula específica;

(14) *Intervalo variável de refeição*: Em certas instituições, é necessário planejar o horário para a refeição em dois horários consecutivos, sendo que cada turma pode usar apenas um dentre esses dois horários (chamados de primeiro e segundo intervalos). Assim, requer-se que professores e turmas não possam ter mais do que uma aula simples durante os dois horários considerados;

(15) *Intervalo balanceado de refeição*: O número de turmas que usa do primeiro intervalo de refeição não deve ser muito diferente do número de turmas que usa do segundo intervalo. Este requisito é normalmente considerado nas instituições aonde o espaço destinado à alimentação não é suficiente para abrigar todos os alunos ao mesmo tempo.

Algumas das restrições formuladas anteriormente não ocorrem freqüentemente em problemas de horários. Contudo, o autor considerou-as para tornar a formulação mais abrangente.

Na prática, uma tabela de horário satisfazendo a todos esses requisitos pode não existir. As restrições (9) e (10), por exemplo, são contraditórias. Por um lado, deseja-se compactar a agenda dos professores e, por outro, deseja-se espalhar as aulas de uma mesma matéria ao longo da semana. Entretanto, faz-se necessário encontrar um compromisso conveniente entre professor e turma para gerar uma tabela de horário tão boa quanto possível, mesmo na presença de alguns conflitos. O número desses conflitos tem que ser minimizado no sentido que a seguir será explicado.

3.6.2 Formulação Matemática

Apresenta-se, a seguir, algumas definições necessárias à formulação matemática do problema considerado.

Seja uma partição (Y_e, Y_r) do conjunto Y de todas as restrições descritas na seção 3.6.1. As restrições incluídas na categoria Y_e são chamadas de *restrições essenciais*, enquanto aquelas incluídas na categoria Y_r são chamadas de *restrições relaxadas*.

Uma tabela de horários se diz *aceitável*, se ela satisfaz a todas as restrições do conjunto Y.

Uma tabela de horários se diz *viável*, se ela satisfaz as restrições consideradas essenciais. Assim, de acordo com essa definição, uma tabela de horários se diz aceitável, se for viável e satisfazer, além disso, as restrições relaxadas.

O problema de programação de horários descrito na seção anterior será, agora, enquadrado em um problema de otimização combinatória. Seja a função $f(Q)$, a qual mede a inaceitabilidade de um quadro de horário. O princípio do procedimento proposto consiste em procurar pelo quadro de horário $Q^\star$ que minimiza o valor de f sobre o conjunto X de quadros viáveis de horário. Em outras palavras, o problema a ser resolvido é da forma:

$$(PH) \qquad \begin{array}{rl} \min & f(Q) \\ \text{s.a:} & Q \in X \end{array}$$

Na seção seguinte será definido um procedimento iterativo que começa de uma solução viável inicial e tenta encontrar um ótimo deste problema, movendo-se sucessivamente de uma solução a outra, que seja sua vizinha. De forma a aplicar tal princípio para resolver o problema (PH), é necessário definir adequadamente o conjunto X, o qual servirá para encontrar um quadro aceitável de horário a partir de um quadro inicial viável. Necessita-se encontrar um compromisso entre a extensão do conjunto X e a complexidade da função objetivo.

O autor incluiu as restrições (4)-(6), (10)-(12) e (14) na categoria Y_e e relaxou todas as demais, conforme mostra a tabela 3.6.

Embora as restrições (1)-(3) sejam fundamentais, elas também foram relaxadas. Segundo o autor, tal atitude foi tomada visando à dar um maior grau de liberdade aos movimentos durante a fase de busca no espaço de soluções, uma vez que a introdução de tais restrições no conjunto Y_e poderia reduzir significativamente o

Tabela 3.6: Particionamento do conjunto Y

Restrições essenciais Y_e	**Restrições relaxadas** Y_r
Pré-alocações (4)	Sobreposição de professores (1)
Indisponibilidade de professores (5)	Sobreposição de turmas (2)
Indisponibilidade de turmas (6)	Sobreposição de salas (3)
Restrições de distribuição (10)	Restrições de preparação de salas (13)
Gerenciamento de aulas duplas (11)	Indisponibilidade de matérias (7)
Restrições de precedência (12)	Restrições geográficas (8)
Intervalo variável de refeição (14)	Restrições de compacidade (9)
	Intervalo balanceado de refeição (15)

tamanho de X.

As restrições (10), (12) e (14) também poderiam ser relaxadas. No entanto, experimentos feitos pelo autor nesse sentido apontaram que esta forma de manusear as restrições não afetava significativamente a qualidade dos resultados. Assim, decidiu-se não relaxá-las, garantindo-se, portanto, que as mesmas sejam satisfeitas na solução final.

As restrições de distribuição foram modeladas considerando-se que não mais do que $u(j)$ aulas do curso co_j podem ser alocadas durante um mesmo dia. O valor $u(j)$ depende do número de aulas do curso co_j e é fixado pelo elaborador do quadro de horários.

Define-se, a seguir, a função objetivo a qual mede a relação entre um quadro viável de horário e as necessidades efetivas da escola:

$$f(Q) = \sum_{i=1}^{7} \omega_i f_i(Q)$$

Esta função é estruturada hierarquicamente: ela é definida usando-se um conjunto de pesos ω_i, os quais conferem uma importância relativa às restrições relaxadas. A componente $f_i(Q)$ calcula o grau de violação à i-ésima restrição relaxada da tabela de horários.

Antes de explicitar as componentes de $f(Q)$, serão introduzidas algumas notações complementares à da seção 3.6.1.

Sejam:

A_k, o conjunto de aulas lecionadas no horário k;

$tcomum(x, y)$, o número de professores comuns às aulas x e y. Se essas aulas não tem professores em comum, então $tcomum(x, y) \in \{0, 1\}$;

$ccomum(x, y)$, o número de turmas comuns às aulas x e y;

$RecDisp(t, k)$, o número de recursos especiais (salas especializadas) do tipo t disponíveis no horário k;

$RecReq(t, k)$, o número de recursos especiais (salas especializadas) do tipo t requeridas no horário k;

HS_s, o conjunto de horários durante os quais a matéria s não deve ser ensinada;

$AulasAloc(s, k)$, o número de aulas da matéria s alocadas ao horário k;

$cheios(i, turno)$, a soma do número de seqüências de aula ministradas pelo professor i durante uma das metades do dia ($turno$). Por exemplo, um professor que tenha, em um turno de 5 horários, aulas no primeiro, terceiro e quarto horários, tem 2 seqüências de aula e, portanto, $cheios(i, turno) = 2$;

$livres(i, turno)$, o número de horários livres na agenda do professor i durante uma das metades do dia ($turno$);

$refeicao(d, i)$, o número de turmas que têm uma aula no i-ésimo intervalo para refeição do dia d ($i \in \{1, 2\}$);

Especifica-se, a seguir, todas as 7 componentes da função objetivo $f(Q)$.

(1) $f_1(Q)$ avalia quantos professores estão envolvidos simultaneamente em 2 aulas:

$$f_1(Q) = \sum_{k=1}^{p} \sum_{x<y \in A_k} tcomum(x, y)$$

(2) $f_2(Q)$ avalia quantas turmas têm, simultaneamente, 2 aulas:

$$f_2(Q) = \sum_{k=1}^{p} \sum_{x<y \in A_k} ccomum(x, y)$$

(3) $f_3(Q)$ avalia quantas vezes a disponibilidade de salas especializadas não é respeitada:

$$f_3(Q) = \sum_{t=1}^{nr} \sum_{k=1}^{p} \max \{RecReq(t, k) - RecDisp(t, k),\ 0\}$$

(4) $f_4(Q)$ avalia quantas aulas são alocadas em um horário durante o qual o ensino de uma matéria não é desejado:

$$f_4(Q) = \sum_{s=1}^{ns} \sum_{k \in HS_s} AulasAloc(s, k)$$

(5) $f_5(Q)$ avalia o número de professores e turmas que participam de duas aulas alocadas consecutivamente durante o mesmo dia, em dois prédios distantes. Seja $\delta(x, y) = 1$ se as aulas x e y são ministradas em lugares distantes e $\delta(x, y) = 0$, caso contrário. Então, tem-se:

$$f_5(Q) = \sum_{d=1}^{nd} \sum_{k=nh\times(d-1)+1}^{nh\times d-1} \sum_{\substack{x\in A_k \\ y\in A_{k+1}}} [\delta(x, y) \times (ccomum(x, y) + tcomum(x, y))]$$

(6) $f_6(Q)$ mede o grau de compacidade do quadro de horário:

$$f_6(Q) = \sum_{i=1}^{m} \sum_{turno=1}^{2\times nd} (cheios(i, turno) \times livres(i, turno))$$

(7) $f_7(Q)$ soma, a cada dia, a diferença, em valor absoluto, entre o número de turmas que têm uma aula no primeiro intervalo e o número de turmas que têm aula no segundo intervalo:

$$f_7(Q) = \sum_{d=1}^{nd} |refeicao(d, 1) - refeicao(d, 2)|$$

Para exemplificar, considere a tabela 3.7, a qual apresenta a agenda de um professor durante 3 dias de 10 horários cada. O intervalo para refeição é fixado no sexto horário de cada dia. Em cada célula dessa agenda encontra-se o número de aulas dadas pelo professor no horário considerado. Seja $f_{6,d}(Q)$ a parcela de $f_6(Q)$ referente à agenda do professor no dia d.

Tabela 3.7: Agenda de um professor durante 3 dias de 10 horários

Dia\Hor.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	1	0	2	1	0	**int**	0	0	1	0	$f_{6,1}(Q) = 2 \times 2 + 1 \times 3 = 7$
2	0	0	0	0	0	**int**	1	1	1	1	$f_{6,2}(Q) = 0 \times 5 + 1 \times 0 = 0$
3	1	1	1	1	0	**int**	0	0	0	0	$f_{6,3}(Q) = 1 \times 1 + 0 \times 4 = 1$

Evidentemente, esse quadro de horário é inviável, uma vez que o professor em questão está envolvido em duas aulas no 3º horário do primeiro dia. Na agenda do primeiro dia, observa-se que, pela manhã, o professor ministra duas seqüências de aula e tem dois horários livres e que, pela tarde, ministra apenas uma aula e tem três horários livres. Portanto, $f_{6,1}(Q) = 2 \times 2 + 1 \times 3 = 7$. Quatro aulas consecutivas são alocadas no turno da tarde do segundo dia, bem como na manhã do terceiro dia. Entretanto, ao contrário de $f_{6,2}(Q)$, $f_{6,3}(Q)$ é diferente de zero. Isto é devido ao fato de que há um horário livre na manhã do último dia. Assim, faz-se necessário melhorar a compacidade dessa agenda, alocando-se uma aula no 5º horário do terceiro dia.

Exceto para $f_6(Q)$, o limite inferior de cada componente da função objetivo é zero. O valor mínimo de $f_6(Q)$ é estritamente positivo se a agenda de pelo menos um professor inclui um número de horários que não seja uma combinação linear do número de horários em um dia e em uma metade do dia.

3.6.3 Adaptação de Busca Tabu

Descreve-se nesta seção uma adaptação do algoritmo de Busca Tabu para o problema considerado.

3.6.3.1 Geração de um quadro de horário inicial

Para gerar um quadro inicial, as aulas são ordenadas de acordo com as suas disponibilidades (isto é, de acordo com o número de horários durante os quais ela pode alocada, levando-se em conta as pré-alocações e a indisponibilidade de professores e turmas). Essa ordenação é feita considerando-se, primeiramente, aquelas com o menor número de horários disponíveis.

Para alocar uma aula a_i a um horário k (operação denotada por $a_i \rightarrow k$), leva-se em consideração o mais alto grau possível dentre os seis seguintes graus de sobreposição:

grau 5: Sem qualquer tipo de sobreposição;

grau 4: Sem sobreposição de professores e turmas;

grau 3: Sem sobreposição de turmas;

grau 2: Sem sobreposição de professores;

grau 1: Sem sobreposição de salas;

grau 0: Com qualquer tipo de sobreposição;

O princípio usado para produzir um quadro de horário Q inicial é detalhado na figura 3.4.

procedimento *SolucaoInicial*

Ordene as aulas de acordo com as suas disponibilidades;
$Q \leftarrow \emptyset$;
para $(i = 1, \ldots, na)$ faça
    grau $\leftarrow 5$;
    enquanto (a aula a_i não estiver alocada) faça
        Gere um horário k qualquer relativamente ao grau de
            sobreposição, tal que $Q \cup \{a_i \rightarrow k\}$ seja viável;
        se (k existe) então $Q \leftarrow Q \cup \{a_i \rightarrow k\}$ senão $grau \leftarrow grau - 1$;
    fim-enquanto;
fim-para;
Retorne Q;

fim *SolucaoInicial*;

Figura 3.4: Algoritmo de geração de uma solução inicial

3.6.3.2 Definição da Vizinhança de um Quadro de Horário

A vizinhança $N(Q)$ de um quadro de horário Q consiste de todos os quadros de horário que podem ser obtidos de Q mudando-se o horário de uma única aula.

De acordo com essa definição, dois quadros de horário são vizinhos se todas as aulas, exceto uma, são alocadas nos mesmos lugares nos dois quadros.

3.6.3.3 Definição das Listas Tabu

De forma a prevenir ciclagem, duas listas tabu, T_1 e T_2 são utilizadas. Sempre que uma aula a_i move-se de um horário h_1 a um horário h_2 ($h_1 \neq h_2$), introduz-se a_i na lista T_1 e o par (a_i, h_1) em T_2. Assim, durante $|T_1|$ iterações do algoritmo, a aula a_i fica impedida de mover-se e durante $|T_2|$ passos, a_i não pode ser alocada novamente ao horário h_1. Evidentemente, a segunda lista tabu tem sentido se e somente se $|T_1| < |T_2|$.

3.6.3.4 Definição das Funções de Aspiração

Consideram-se duas funções de aspiração, g_1 e g_2, definidas a seguir, as quais são partes da função objetivo.

$$g_1(Q) = \sum_{i=1}^{3} q_i \times f_i(Q) \quad \text{e} \quad g_2(Q) = \sum_{i=1}^{7} \omega_i \times f_i(Q)$$

onde $q_i = \dfrac{\omega_i}{\text{mdc}(\omega_1, \omega_2, \omega_3)}$

O *status* tabu de um movimento de Q a Q' é retirado quando ocorrer pelo menos uma das seguintes condições:

(i) $g_1(Q')$ é menor que o menor valor de g_1 encontrado quando é feito um movimento a partir de um quadro Q'' tal que $g_1(Q'') = g_1(Q)$;

(ii) $g_1(Q')$ é igual ao menor valor de g_1 encontrado quando é feito um movimento a partir de um quadro Q'' tal que $g_1(Q'') = g_1(Q)$ e $g_2(Q')$ é menor que o melhor valor de g_2 encontrado durante tal movimento.

Duas funções de aspiração, $A_1(z)$ e $A_2(z)$ são desenvolvidas. Inicialmente, faz-se $A_1(g_1(Q)) = g_1(Q)$ e $A_2(g_1(Q)) = \infty$ para cada valor de g_1. A figura 3.5 mostra como as funções de aspiração são atualizadas sempre que um movimento é feito de um quadro Q ao melhor quadro Q' da vizinhança de Q. O movimento reverso de Q' a Q é considerado nessa atualização, muito embora ele não tenha sido feito explicitamente.

Em resumo, o *status* tabu de um movimento de Q a Q' é cancelado se o seguinte critério é satisfeito:

$$g_1(Q') < A_1(g_1(Q)) \quad \text{ou} \quad \{g_1(Q') = A_1(g_1(Q)) \;\text{ e }\; g_2(Q') < A_2(g_1(Q))\}$$

procedimento *AtualizaFuncaoAspiracao*;
$Q_{1,1} \leftarrow Q$; $Q_{1,2} \leftarrow Q'$;
$Q_{2,1} \leftarrow Q'$; $Q_{2,2} \leftarrow Q$;
<u>para</u> $(k = 1, 2)$ <u>faça</u>
    <u>se</u> $(g_1(Q_{k,2}) \leq A_1(g_1(Q_{k,1})))$ <u>então</u>
        <u>se</u> $(g_1(Q_{k,2}) < A_1(g_1(Q_{k,1})))$
            <u>então</u>
                $A_1(g_1(Q_{k,1})) \leftarrow g_1(Q_{k,2})$;
                $A_2(g_1(Q_{k,1})) \leftarrow g_2(Q_{k,2})$;
            <u>senão</u>
                $A_2(g_1(Q_{k,1})) \leftarrow \min\{A_2(g_1(Q_{k,1})), g_2(Q_{k,2})\}$;
        <u>fim-se</u>;
    <u>fim-se</u>;
<u>fim-para</u>;
fim *AtualizaFuncaoAspiracao*;

Figura 3.5: Atualização das funções de aspiração

3.6.3.5 Geração de uma Solução vizinha

Considera-se que uma aula esteja em *conflito* em um quadro de horário se ela viola pelo menos uma das restrições relaxadas. Do ponto de vista das restrições de compacidade, entende-se que uma aula a_i está em conflito se ela for uma aula isolada (isto é, não se há uma aula no horário anterior nem no horário seguinte a a_i) na agenda de pelo menos um dos professores que a ministre. Considera-se que uma aula pré-alocada (cuja disponibilidade está reduzida a um único horário) nunca está em conflito.

Durante todo o algoritmo, somente aulas conflitantes sofrem movimentos. Parte-se do pressuposto de que a movimentação de aulas não conflitantes diminui muito pouco o valor da função objetivo. Na verdade, apenas a componente f_6, a qual mede a compacidade do quadro de horário, pode ser reduzida.

Nem toda aula conflitante, no entanto, pode movimentar-se livremente. Se a melhor solução encontrada não satisfaz as restrições de sobreposição então somente são permitidos movimentos de aulas que violam pelo menos uma das restrições de sobreposição. Esta estratégia visa a permitir encontrar rapidamente boas regiões do espaço de soluções. Entretanto, tão logo uma solução sem nenhum tipo de restrição de sobreposição é encontrada, qualquer aula conflitante pode voltar a movimentar-se novamente.

Quando o número de aulas conflitantes torna-se muito pequeno, a primeira lista tabu definida na seção anterior pode proibir todos os movimentos que guiam a vizinhos viáveis da solução corrente. Neste caso, esquece-se o *status* tabu de um movimento se ele for devido à primeira lista.

A cada passo do algoritmo, procura-se gerar *nvizinhos* diferentes pares (a_i, d_j), onde a_i é uma aula conflitante no quadro corrente e d_j é um dia da semana. Seja *nConfSemana* o número de aulas conflitantes vezes o número de dias da semana. Se *nConfSemana* for menor que *nvizinhos*, então somente *nConfSemana* pares (a_i, d_j) são considerados. Determina-se, a seguir, para cada aula a_i, o melhor horário k do dia d_j. Evidentemente, se a aula já encontra-se alocada ao dia d_j, então k precisa ser diferente do horário corrente de a_i. O melhor dos movimentos $a_i \rightarrow k$ gerados é, então, armazenado visando à construção do próximo quadro de horário.

Uma vez que o valor mínimo da função objetivo é desconhecido, o algoritmo pára quando nenhum melhoramento na melhor solução for encontrado em $BTmax$ iterações.

3.6.3.6 Estratégia de Diversificação

No início do algoritmo os pesos relativos às restrições de sobreposição $(\omega_1, \omega_2, \omega_3)$ são maiores do que todos os demais. Isto força o algoritmo a visitar soluções que são relativamente boas do ponto de vista dessas restrições, isto é, a gerar soluções com um menor número de sobreposições.

Para forçar o algoritmo a pesquisar regiões promissoras ainda não exploradas, faz-se, então, a redução drástica de tais pesos $(\omega_i \leftarrow \omega_i' \ \ \forall i = 1, 2, 3)$. Assim, ao reduzí-los, a busca concentra-se em restrições relaxadas menos importantes, tendo-se, consequentemente, um grau maior de mobilidade sem, contudo, abandonar completamente a região do espaço de soluções em que se está.

Esta estratégia de diversificação é realizada durante $ndiv$ iterações após ter-se visitado um quadro de horário sem restrições de sobreposição e executado um certo número de iterações sem melhora no valor da melhor solução encontrada até então.

3.7 O Modelo de Colorni, Dorigo e Maniezzo

Descreve-se, a seguir, o modelo tratado por Colorni et al. [24], desenvolvido para uma escola italiana de nível médio, em que as turmas têm 5 horários diários de aula, dadas em um único turno, durante 6 dias por semana (de segunda a sábado).

O algoritmo proposto é um procedimento baseado em Algoritmos Genéticos. Como o algoritmo gera freqüentemente soluções inviáveis, em virtude da aplicação dos operadores de reprodução (mutação e *crossover*), os autores desenvolveram um filtro, o qual tem por objetivo reparar a inviabilidade criada pelos operadores. As inviabilidades também são tratadas pela função objetivo, a qual inclui uma penalidade para cada restrição violada.

A metodologia apresentada a seguir considera, sem prejuízo de generalidade, que existem 10 turmas.

3.7.1 Conceito de Viabilidade de um Quadro de Horário

Considera-se que um quadro de horário é viável se ele satisfaz as seguintes restrições:

(a) Cada professor cumpre sua carga horária semanal de ensino;

(b) Cada turma tem todas as suas aulas semanais;

(c) Nenhum professor está alocado a mais de uma turma ao mesmo tempo;

(d) Em cada horário não há mais do que um professor lecionando para uma mesma turma;

(e) As turmas não têm horários descobertos, isto é, horários sem aula.

3.7.2 Representação do Problema

Um quadro de horário é representado como uma matriz $Q_{m \times p}$ de valores inteiros, onde em cada linha i de Q representa-se a alocação semanal do professor t_i e em cada coluna k de Q, um horário. Cada posição q_{ik} representa uma atividade exercida pelo professor t_i no horário k. A cada posição q_{ik} está associado um alfabeto A, representativo de todas as tarefas (aulas e outras atividades) que os professores têm que realizar.

O alfabeto utilizado é $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0, D, P, S, ., -\}$, onde cada elemento tem o seguinte significado:

(i) Os caracteres "1, 2, 3, ..., 9, 0" representam as 10 turmas para as quais os professores têm que ensinar;

(ii) O caracter "D" indica os horários em que os professores estão à disposição para possível substituição;

(iii) O caracter "P" representa os horários reservados para o desenvolvimento profissional do professor;

(iv) O caracter "S" indica os horários durante os quais as aulas são fixas. Esses horários são fixados na fase de inicialização manualmente ou por rodada prévia do algoritmo e são chamados de *horários fixos*. Esses horários são utilizados para tratar as aulas simultâneas;

(v) O caracter "." é usado para representar os horários nos quais o professor está sem atividade;

(vi) O caracter "———" indica dia de folga do professor.

A tabela 3.8 mostra um exemplo de um quadro de horário.

Tabela 3.8: Exemplo de um quadro de horário

Professor/ Matéria	**Seg.**	**Ter.**	**Qua.**	**Qui.**	**Sex.**	**Sab.**
Literatura 1	1100	0010 .	000 . .	00 . . .	000D1	———
Literatura 2	. . 1D1	555D5	. 1511	55 . . .	———	5511.
Literatura 3	6656.	66D6 .	666 . .	———	65D . .	. . D55
Literatura 4	———	44 . D4	744 . .	7747 .	77 . . .	7D4D7
Literatura 5	. . . 33	. . 322	———	2D2 . .	232D3	22D33
Literatura 6	. D889	. 9D88	9D899	———	. . . 8D	. . 899
Inglês	———	. . 441	2333 .	. . . 22	. 1100	40D41
Alemão	———	776D9	. 9955	66788	. . . . .	6958 .
História/Filos 1	D4242	332 . .	4D2 . .	———	3443 .	342 . .
História/Filos 2	877 . .	8D . 9 .	———	88D99	. . 979	. 7978
Mat/Física 1	. . 324	———	. 2D44	. 4334	. . 342	. 3322
Mat/Física 2	99 . 97	. . 877	———	998D7	898 . .	887 . .
Matemática 1	———	1101 .	. . . 66	1D660	1D6 . .	06D00
Matemática 2	55S5 .	DSS . .	5SS . .	DSSS5	DSS . .	———
Arte	30 . 78	. 8956	. 7102	43 . 41	. 2596	———
Ciências Nat.	4261 .	9273 .	38 . 87	. 2913	. 8764	———
Física Exp.	0SSS.	SSSS0	1SSS0	———	5SS55	11666
Ginástica 1	234D .	———	. . D23	. 100 .	4DD1 .	. . . . .
Ginástica 2	789D .	———	. . D78	. D556	96D . .	. . . . .
Religião	 6	———	. 57 . .	3 . 1 . .	. . . 28	9 . 0 . 4

3.7.3 Estrutura Hierárquica da Função Objetivo

A função objetivo considerada mede a diferença entre um quadro corrente de horário e um quadro ideal de horário, isto é, um quadro de horário que satisfaz a todas as restrições e requisitos didáticos, organizacionais e pessoais, conforme definido a seguir.

A função objetivo $z(Q)$ de um quadro Q é calculada conforme abaixo:

$$z(Q) = \alpha \times Inv + \beta_1 \times Did + \beta_2 \times Org + \beta_3 \times Pes$$

onde:

Inv é o número de inviabilidades no quadro Q;

Did mede a razão de insatisfação ao atendimento dos requisitos didáticos do quadro Q. Mais precisamente, $Did = \sum_{i=1}^{4} \delta_i \times d_i$, onde δ_i é o peso associado a cada requisito didático, conforme detalhado mais adiante, e d_i é o número de vezes que o i-ésimo requisito didático não foi alcançado no quadro Q;

Org mede a razão de insatisfação ao atendimento dos requisitos organizacionais do quadro Q. Esta componente é calculada através da expressão $Org = \sum_{i=1}^{3} \omega_i \times o_i$, onde ω_i é o peso associado a cada um dos requisitos organizacionais, conforme detalhado a seguir, e o_i é o número de vezes que o i-ésimo requisito organizacional não foi alcançado no quadro Q;

Pes mede a razão de insatisfação ao atendimento dos requisitos pessoais do quadro Q. É calculada como $Pes = \sum_{i=1}^{m} p_i \times \gamma_i$, onde γ_i, mede a insatisfação do professor i no quadro por causa de desejos pessoais não alcançados e p_i é o peso associado a cada professor;

$\alpha, \beta_1, \beta_2, \beta_3$ são pesos que refletem a importância relativa de cada uma das componentes da função objetivo. Os autores os escolheram satisfazendo à condição: $\alpha \gg \beta_1 \approx \beta_2 \approx \beta_3$, de forma a gerar uma estrutura hierárquica na função objetivo. Assim, três níveis hierárquicos são considerados:

Nível 1: condições de viabilidade (Inv);

Nível 2: condições de gerenciamento (Did, Org);

Nível 3: : condições relativas somente a professores (Pes);

No nível 1, são avaliadas as possíveis sobreposições de professores, (isto é, 2 ou mais professores ensinando para uma mesma turma em um mesmo horário) e os horários descobertos das turmas (isto é, horários em que uma turma está sem aula, quando deveriam estar em atividade).

No nível 2 os seguintes requisitos são avaliados:

a) Requisitos didáticos:

1) No máximo 4 horários de aula por dia por professor (peso associado: δ_1);

2) Nunca o mesmo professor todo dia no último horário (peso associado: δ_2);

   3) Distribuição uniforme de horários da mesma matéria ao longo da semana (peso associado: δ_3);
   4) Aulas duplas para os professores que as requererem (peso associado: δ_4);

b) Requisitos organizacionais:

   1) Concentração no mesmo dia (tanto quanto possível) de horários para encontro dos professores com pais de alunos (peso associado: ω_1);
   2) Nunca menos do que 2 horários de aula por dia para cada professor (peso associado: ω_2);
   3) Agenda do professor sem *buracos* (peso associado: ω_3);

δ_i e ω_i são parâmetros definidos pelos usuários e relativizam a importância dos diferentes requisitos didáticos e organizacionais.

No nível 3 são avaliadas as preferências de cada professor pelo seu quadro específico. Cada professor fixa e valora seus requisitos pessoais, tais como o dia de folga desejado, não trabalhar no primeiro ou no último horário, etc. Esses valores são então normalizados, considerando os requisitos de todo o conjunto de professores. O número de requisitos pessoais não atingidos multiplicado pelo peso normalizado associado a cada requisito constituem os termos p_i na função objetivo. Além disso, cada professor é ordenado utilizando-se um peso γ_i, que leva em consideração sua importância.

3.7.4 Adaptação de Algoritmos Genéticos

Os autores consideram como cromossomo, um quadro Q. Assim, cada célula de Q é um gene pertencente ao alfabeto A.

A função de aptidão *ff* é calculada como segue. A cada geração, calcula-se o valor da função objetivo $z(Q)$ de cada um dos indivíduos da população, determinando-se seus valores mínimo e máximo ($z_{\min}$ e $z_{\max}$, na figura 3.6). Esses valores definem um intervalo no eixo da função objetivo $z(Q)$, o qual é linearmente mapeado em um intervalo no eixo da função de aptidão *ff*, limitado por duas constantes do sistema, a saber, *MAXFIT* e *MINFIT*, tal que $z_{\min}$ corresponde a *MAXFIT* e $z_{\max}$ corresponde a *MINFIT*. Segundo os autores, esse procedimento atinge dois objetivos: primeiro, ele minimiza a função objetivo ao mesmo tempo que maximiza a função de aptidão; segundo, ele discrimina soluções pertencentes a populações cujos indivíduos têm função objetivo com valores praticamente iguais, situação que é freqüente nos últimos estágios do processo de busca, quando $z_{\min}$ e $z_{\max}$ são valores próximos.

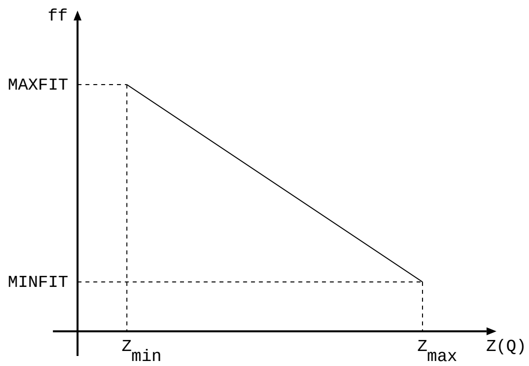

Figura 3.6: Gráfico da função de aptidão *ff*

O algoritmo usado está esquematizado na figura 3.7. $LOCAL$ é uma variável booleana que tem valor verdadeiro se um procedimento de busca local deve ser ativado e falso, caso contrário.

Detalha-se, a seguir, o algoritmo proposto.

reproducao é um operador que tem por objetivo selecionar os indivíduos que têm os maiores valores para a função de aptidão *ff*. A cada indivíduo Q aplica-se uma probabilidade de reprodução $p_{reproducao}(Q)$, que é igual ao valor de sua função de aptidão dividido pela soma dos valores das aptidões de todos os indivíduos da população.

crossover é um operador que visa à geração de filhos mais aptos que os pais, sendo os filhos obtidos através da mistura de genes dos pais. Considerando *lff* a função de aptidão local, isto é, a parte da função de aptidão *ff* que diz respeito somente às características específicas de cada professor, descreve-se, a seguir, como age o operador *crossover*. Dados dois indivíduos, Q_1 e Q_2, da população, as linhas de Q_1 são ordenadas em ordem decrescente dos valores de *lff*. O primeiro filho é gerado combinando-se as melhores l_1 linhas de Q_1 com as $m - l_1$ linhas remanescentes de Q_2, sendo m o número de professores. O segundo filho é obtido combinando-se as linhas não utilizadas dos indivíduos Q_1 e Q_2. O valor de l_1 é determinado pelo algoritmo

```
procedimento AG Colorni
Crie uma população de N indivíduos randômicos,
    satisfazendo cada indivíduo as seguintes restrições:
    - Cada professor cumpre sua carga horária semanal;
    - Alguns horários são fixos e não podem ser modificados;
Iter ← 0;    {Contador do número de iterações}
AGmax ← Número máximo permitido de iterações;
NumMaxInv ← Número máximo permitido de inviabilidades;
enquanto (Iter < AGmax) faça
    Iter ← Iter + 1;
    Aplique reproducao;
    Aplique crossover;
    para (i = 1, ..., N) faça
        Aplique mutacao de ordem k;
        Aplique mutacao diaria;
        se (LOCAL) então aplique PesquisaLocal;
        se (NumInv > NumMaxInv) então aplique filtro;
    fim-para;
fim-enquanto;
fim AG Colorni;
```

Figura 3.7: Algoritmo Genético proposto por Colorni

com base na função de aptidão local dos pais (os autores não especificaram como isso é feito). O operador *crossover* é aplicado com uma probabilidade $p_{crossover}$, a qual é um parâmetro do algoritmo, para cada par de pais em potencial.

A figura 3.8 mostra como o operador *crossover* é implementado.

mutacao de ordem k é um operador que considera k genes contíguos e permuta-os com outros k diferentes genes contíguos de uma mesma linha. A mutação de ordem 1 é um caso especial deste operador. Ele não pode ser aplicado quando, dentre os genes mutáveis, há o caracter S, representativo dos horários fixos. Este operador é aplicado para cada linha de cada indivíduo com uma certa probabilidade $p_{mutacaok}$, a qual é um parâmetro do sistema.

mutacao diaria é um operador que considera um dia e permuta todos os genes desse dia com os genes de um outro dia pertencentes à uma mesma linha. O i-ésimo dia em um quadro de horário de um professor é um subconjunto de caracteres do alfabeto A contendo genes que codificam 5 horários contíguos, do primeiro ao quinto de um mesmo dia. Ele é, portanto, um caso especial de mutação de ordem $k = 5$ e foi introduzido por razões de eficiência, especialmente no que se diz respeito à

procedimento *crossover*
1 Emparelhe aleatoriamente os indivíduos da população
2 para (cada par de indivíduos e com probabilidade $p_{crossover}$) faça
3 Calcule a função de aptidão local *lff* das linhas dos dois indivíduos;
4 Ordene, por valores decrescentes, os valores das *lff* dos dois indivíduos;
5 Crie dois filhos misturando os genes dos dois indivíduos, conforme se segue: O primeiro filho é gerado tomando-se as melhores l_i linhas do melhor pai e as remanescentes linhas do pior pai; o segundo filho é gerado usando as linhas não usadas dos dois pais;
6 fim-para;
fim *crossover*;

Figura 3.8: Operação *crossover*

alocação do dia de folga de cada professor. Este operador é aplicado com uma certa probabilidade $p_{mutacaod}$, a qual é parâmetro do sistema.

filtro é um operador que recebe como entrada uma solução inviável e retorna uma solução viável. Ele é usado para assegurar viabilidade global a um quadro de horário e é baseado na observação de que, em cada coluna (isto é, horário) do quadro de horário, cada turma precisa estar presente uma e somente uma única vez. Se a turma estiver presente 2 ou mais vezes, há sobreposição de professores. Se a turma não estiver presente, ela está a descoberto, isto é, sem aulas naquele horário. O filtro é baseado em quatro procedimentos, sendo que os dois primeiros identificam trocas de horários de um único professor como uma forma de eliminar as inviabilidades. Os outros dois procedimentos modificam os horários de mais de um professor e, por terem complexidade maiores, somente são aplicados se os dois primeiros falharem.

Capítulo 4

O Problema de Programação de Horários em Escolas Abordado

Este capítulo tem como objetivo apresentar o problema de programação de horários em escolas abordado. A seção 4.1 introduz o problema, relatando as características da escola considerada. Na seção 4.2 discriminamos os requisitos exigidos para um quadro de horário. Finalmente, a seção 4.3 exemplifica um quadro de horário de professores dessa escola com o respectivo quadro de horário das turmas.

4.1 Introdução

A instituição de ensino considerada para análise é a Escola Dom Silvério, situada à av. Manoel Leandro Correia s/n, em Mariana, Minas Gerais, pertencente à rede pública de ensino do Estado de Minas Gerais.

Trata-se de uma típica escola pública de segundo grau brasileira, que funciona em três turnos: manhã, tarde e noite. Em cada dia são reservados 15 horários para a realização das aulas, sendo 5 para cada turno. Cada turma tem 25 horas-aula, distribuídas uniformemente em 5 dias, de segunda a sexta-feira, e realizadas em um único turno.

Cada hora-aula é de 50 minutos. Entre o terceiro e o quarto horário diário há um intervalo de 20 minutos. Para o turno da noite esse intervalo é de 10 minutos.

A instituição conta com um quadro de 34 professores, em média, entre efetivos (22) e contratados. Esse número varia de ano para ano, dependendo do número de turmas que a escola oferece. No ano de 2000 a escola trabalhou com 30 turmas, sendo 12 no turno matutino, 6 no vespertino e 14 no noturno.

A cada professor efetivo está associado um 'cargo', isto é, um regime de trabalho no qual o professor deve lecionar, no mínimo, 18 horas-aula. Na rede estadual, um professor pode assumir, no máximo, dois cargos. A cada cargo está associada uma matéria, para a qual o professor foi contratado. Para cumprir seu regime de trabalho, cada professor deve lecionar a(s) matéria(s) de sua competência, procurando totalizar 18 horas-aula. Caso a carga horária da(s) matéria(s) para a(s) qual(is) foi contratado, para todas as turmas, em todos os turnos, não atinja o limite estipulado pelo cargo, então, obrigatoriamente, o professor deve completar sua carga horária contratual com outras matérias afins.

Em geral, os professores procuram concentrar suas atividades em um único turno. Assim, escolhem primeiramente as turmas de um mesmo turno relacionadas à sua matéria. Não se completando a carga horária contratual, passam, então, para um outro turno. Para a grande parte dos professores, um turno, ou dois, no máximo, são suficientes para completar sua carga de trabalho. Alguns poucos, como por exemplo, os que ministram a disciplina de Artes, necessitam trabalhar três turnos.

A distribuição dos professores às turmas é feita respeitando-se as preferências dos professores efetivos com maior tempo de serviço. Isto é, o professor com maior tempo de serviço escolhe primeiro as turmas para as quais vai lecionar. A seguir, é a vez do segundo professor; e assim sucessivamente.

Cada professor efetivo tem direito a um dia de folga. O número de professores efetivos de folga por dia é o quociente da divisão do número de professores efetivos pelo número de dias da semana. Os professores efetivos com maior tempo de serviço escolhem, primeiro, seu dia de folga. Os efetivos que não quiserem gozar um dia de folga têm direito a outras facilidades, como, por exemplo, ministrar aulas apenas nos quatro primeiros horários de cada turno.

Para complementar seus salários, a maioria dos professores efetivos que não possuem dois cargos, têm outras atividades. Em geral, lecionam em outras instituições de ensino, ficando, assim, indisponíveis em determinados horários ou com dedicação restrita à escola. Para esses professores é fundamental que a escola dê-lhes uma agenda de trabalho compacta, com tantos turnos livres quanto possível, de forma que possam assumir essas suas atividades extras nos dias (turnos) em que não estiverem envolvidos em aula na escola. Os professores contratados também não têm disponibilidade integral e requerem o mesmo tratamento.

Por outro lado, o horário das aulas sofre pequenas modificações ao longo do ano. Isso decorre, sobretudo, pelo fato de existirem professores que cursam a Universidade, cujo período letivo não coincide com o da escola, ficando eles dependentes dos horários das disciplinas em que podem se matricular.

Cada professor especifica a configuração desejada da carga horária da(s) matéria(s) sob sua responsabilidade e o planejador de horários da escola procura atender sempre que possível. Entretanto, por questões pedagógicas, uma turma não pode ter mais do que duas horas-aula por dia de uma mesma matéria.

As aulas de educação física podem ser realizadas em qualquer horário, mas são limitadas a uma hora-aula diária.

As turmas têm um intervalo de descanso entre o terceiro e o quarto horários, mas esse período não influi no conceito de aula dupla, isto é, considera-se que também é uma aula dupla uma aula iniciada imediatamente antes do intervalo e retomada imediatamente após o mesmo.

Algumas matérias, como por exemplo, física, biologia e química, exigem algumas poucas aulas de laboratório. No entanto, essas aulas não têm freqüência semanal. Assim, é sempre possível negociar entre os professores responsáveis por essas matérias a melhor data e horário para oferecê-las, de forma a não haver sobreposição de turmas.

Não há exigências de aulas simultâneas.

Por todas estas características mencionadas, o planejador de horários da escola decompõe o problema de alocação dos professores às turmas em três subproblemas independentes. Cada subproblema diz respeito a um único turno. Portanto, em cada um deles, apenas as turmas do turno e um subconjunto dos professores estão envolvidos. O objetivo é concentrar, tanto quanto possível, as atividades dos professores envolvidos, procurando atender, secundariamente e se possível, ao pedido de configuração da carga horária de cada matéria.

4.2 Formulação do Problema

O Problema de programação de horários em escolas (PHE) abordado neste trabalho consiste em um conjunto de m professores, n turmas, s matérias e p horários semanais reservados para a realização das aulas. Os p horários semanais são distribuídos em d dias da semana de h horários diários realizados em um único turno, isto é, $p = d \times h$. As turmas, as quais estão sempre disponíveis, são conjuntos disjuntos de estudantes que cursam as mesmas matérias. A cada matéria de uma turma está associado um único professor, previamente fixado. Além disso, a carga horária de cada turma é exatamente p.

Os seguintes requisitos, relativos ao quadro de horário dos professores, devem ser satisfeitos:

(a) *Sobreposição de turmas*:

Um professor não pode ministrar aula para mais de uma turma ao mesmo tempo;

(b) *Sobreposição de professores*:

Uma turma não pode ter aula com mais de um professor em um mesmo horário;

(c) *Carga horária*:

Todo professor tem que cumprir sua carga horária semanal de ensino estabelecida para cada uma das turmas;

(d) *Indisponibilidade*:

Um professor não pode ser alocado a um horário no qual não esteja disponível;

(e) *Número máximo de aulas diárias por turma*:

Uma turma não pode ter mais do que 2 horários diários de aula de uma mesma matéria. No caso de educação física, limita-se a uma aula diária;

(f) *Número de aulas duplas*:

Para um dado conjunto de matérias, os professores requerem que, se possível, as aulas obedeçam a uma certa configuração semanal. Por exemplo, em uma matéria de 3 horas-aula semanais, a configuração solicitada pode ser 2-1, isto é, uma aula dupla (ministrada em dois horários consecutivos) e outra ministrada em um único horário, não necessariamente nessa ordem;

(g) *Compacidade do quadro de horário*:

A agenda dos professores deve ser tão compacta quanto possível.

Um quadro de horário de professores que viola pelo menos um dos requisitos (a), ..., (e) é considerado **inviável** e não pode ser praticado pela escola.

4.3 Um exemplo

A tabela 4.1 relaciona as diversas matérias, e suas respectivas cargas horárias semanais, a serem cursadas pelas turmas em cada ano do segundo grau.

Tabela 4.1: Carga horária semanal das matérias

Turno	Manhã/Tarde			Noite		
Matéria\Ano	1^0 Ano	2^0 Ano	3^0 Ano	1^0 Ano	2^0 Ano	3^0 Ano
História	2	2	2	2	2	2
Física	3	3	3	3	3	3
Geografia	2	2	2	2	2	2
Literatura	1	1	-	2	2	1
Biologia	3	2	2	3	3	3
Inglês	2	2	2	2	2	2
Português	3	4	4	4	4	4
Matemática	3	4	4	3	4	4
Química	3	3	2	3	3	2
Ed. Física	2	2	2	-	-	-
Filosofia	1	-	1	1	-	1
Artes	-	-	1	-	-	1
Total	25	25	25	25	25	25

A tabela 4.2, a seguir, mostra o horário de aula do turno da tarde da escola Dom Silvério, praticado no início do primeiro semestre letivo do ano 2000. Os números sobrescritos representam o número do professor responsável pela matéria. Por exemplo, a turma '1G' tem aula no último horário da terça-feira (horário 10 da semana) com o professor '13'.

A tabela 4.3 mostra o correspondente quadro Q de horário dos professores do turno da tarde. Cada célula q_{ik} desta tabela contém o nome da turma para a qual o professor i deve lecionar no horário k. O caracter '$--$' indica indisponibilidade do professor e um caracter em branco indica que o professor está sem atividade naquele horário. Ao lado do número de cada professor consta o nome da matéria que ele leciona. Por exemplo, o professor 13, que ministra educação física, tem um encontro marcado com a turma '1G' às terças-feiras, no último horário.

Tabela 4.2: Quadro de horário das turmas do turno da tarde

Hor.\Turma		**1º E**	**1º F**	**1º G**	**1º H**	**1º I**	**2º E**
2^a Feira	1	Física$^{(2)}$	Química$^{(12)}$	Matemática$^{(10)}$	Biologia$^{(8)}$	Geografia$^{(3)}$	Geografia$^{(4)}$
	2	Ed. Física$^{(13)}$	Física$^{(2)}$	História$^{(1)}$	Matemática$^{(10)}$	Geografia$^{(3)}$	Biologia$^{(8)}$
	3	História$^{(1)}$	Ed. Física$^{(13)}$	Química$^{(12)}$	Física$^{(2)}$	Português$^{(6)}$	Matemática$^{(10)}$
	4	Física$^{(2)}$	Química$^{(12)}$	História$^{(1)}$	Português$^{(6)}$	Matemática$^{(10)}$	Ed. Física$^{(13)}$
	5	História$^{(1)}$	Física$^{(2)}$	Ed. Física$^{(13)}$	Química$^{(12)}$	Português$^{(6)}$	Matemática$^{(10)}$
3^a Feira	6	Química$^{(12)}$	História$^{(1)}$	Português$^{(5)}$	Biologia$^{(8)}$	Inglês$^{(9)}$	Matemática$^{(10)}$
	7	Literatura$^{(5)}$	História$^{(1)}$	Biologia$^{(8)}$	Ed. Física$^{(13)}$	Matemática$^{(10)}$	Química$^{(12)}$
	8	Português$^{(5)}$	Ed. Física$^{(13)}$	Química$^{(12)}$	História$^{(1)}$	Biologia$^{(8)}$	Matemática$^{(10)}$
	9	Química$^{(12)}$	Português$^{(5)}$	Inglês$^{(9)}$	História$^{(1)}$	Ed. Física$^{(13)}$	Biologia$^{(8)}$
	10	Biologia$^{(8)}$	Português$^{(5)}$	Ed. Física$^{(13)}$	Inglês$^{(9)}$	Matemática$^{(10)}$	Química$^{(12)}$
4^a Feira	11	Inglês$^{(9)}$	Literatura$^{(6)}$	Português$^{(5)}$	Matemática$^{(10)}$	Química$^{(12)}$	Física$^{(2)}$
	12	Ed. Física$^{(13)}$	Matemática$^{(10)}$	Português$^{(5)}$	Química$^{(12)}$	Português$^{(6)}$	Inglês$^{(9)}$
	13	Inglês$^{(9)}$	Matemática$^{(10)}$	Física$^{(2)}$	Química$^{(12)}$	Ed. Física$^{(13)}$	Português$^{(6)}$
	14	Matemática$^{(10)}$	Inglês$^{(9)}$	Física$^{(2)}$	Ed. Física$^{(13)}$	Química$^{(12)}$	Português$^{(6)}$
	15	Matemática$^{(10)}$	Química$^{(12)}$	Inglês$^{(9)}$	Português$^{(6)}$	Física$^{(2)}$	Ed. Física$^{(13)}$
5^a Feira	16	Português$^{(5)}$	Matemática$^{(10)}$	Geografia$^{(3)}$	Física$^{(2)}$	Biologia$^{(8)}$	Química$^{(12)}$
	17	Biologia$^{(8)}$	Português$^{(5)}$	Geografia$^{(3)}$	Matemática$^{(10)}$	Física$^{(2)}$	História$^{(1)}$
	18	Português$^{(5)}$	Biologia$^{(8)}$	Matemática$^{(10)}$	Português$^{(6)}$	Química$^{(12)}$	História$^{(1)}$
	19	Matemática$^{(10)}$	Biologia$^{(8)}$	Química$^{(12)}$	Física$^{(2)}$	História$^{(1)}$	Português$^{(6)}$
	20	Química$^{(12)}$	Física$^{(2)}$	Matemática$^{(10)}$	Biologia$^{(8)}$	História$^{(1)}$	Português$^{(6)}$
6^a Feira	21	Filosofia$^{(14)}$	Geografia$^{(3)}$	Biologia$^{(8)}$	Inglês$^{(9)}$	Física$^{(2)}$	Geografia$^{(4)}$
	22	Geografia$^{(3)}$	Filosofia$^{(14)}$	Literatura$^{(7)}$	Geografia$^{(4)}$	Biologia$^{(8)}$	Física$^{(2)}$
	23	Biologia$^{(8)}$	Inglês$^{(9)}$	Filosofia$^{(14)}$	Geografia$^{(4)}$	Literatura$^{(7)}$	Física$^{(2)}$
	24	Geografia$^{(3)}$	Biologia$^{(8)}$	Física$^{(2)}$	Filosofia$^{(14)}$	Inglês$^{(9)}$	Literatura$^{(7)}$
	25	Física$^{(2)}$	Geografia$^{(3)}$	Biologia$^{(8)}$	Literatura$^{(7)}$	Filosofia$^{(14)}$	Inglês$^{(9)}$

Tabela 4.3: Quadro de horário dos professores

Dia	2^a Feira					3^a Feira					4^a Feira					5^a Feira					6^a Feira				
Prof.\Hor.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1 (História)		1G	1E	1G	1E	1F	1F	1H	1H								2E	2E	1I	1I					
2 (Física)	1E	1F	1H	1E	1F						2E		1G	1G	1I	1H	1I		1H	1F	1I	2E	2E	1G	1E
3 (Geografia)	1I	1I														1G	1G				1F	1E		1E	1F
4 (Geografia)	2E																				2E	1H	1H		
5 (Port/Lit)						1G	1E	1E	1F	1F	1G	1G				1E	1F	1E							
6 (Port/Lit)			1I	1H	1I						1F	1I	2E	2E	1H			1H	2E	2E	——	——	——	——	——
7 (Lit.)																						1G	1I	2E	1H
8 (Biologia)	1H	2E				1H	1G	1I	2E	1E						1I	1E	1F	1F	1H	1G	1I	1E	1F	1G
9 (Inglês)						1I			1G	1H	1E	2E	1E	1F	1G						1H		1F	1I	2E
10 (Mat.)	1G	1H									1H	1F	1F	1E	1E	1F	1H	1G	1E	1G	——	——	——	——	——
11 (Mat.)			2E	1I	2E	2E	1I	2E		1I															
12 (Química)	1F		1G	1F	1H	1E	2E	1G	1E	2E	1I	1H	1H	1I	1F	2E		1I	1G	1E					
13 (Ed.Fís.)		1E	1F	2E	1G		1H	1F	1I	1G		1E	1I	1H	2E										
14 (Filosofia)	——	——	——	——	——						——	——	——	——	——	——	——	——	——	——	1E	1F	1G	1H	1I

Capítulo 5

Um Modelo de Programação Matemática para o PHE

O objetivo do presente capítulo é o de apresentar um modelo de programação matemática para o problema de horários em escolas abordado no capítulo 4. No final deste capítulo, estendemos o referido modelo, de forma a contemplar outras restrições que aparecem em escolas secundárias brasileiras.

5.1 Introdução

Existem, na literatura, poucas formulações matemáticas para o PHE. Isto se deve, em parte, ao fato de o PHE ser classificado como NP-difícil [42], o que limita o uso exclusivo de técnicas exatas para abordá-lo. Observamos, também, que a maioria dos modelos de programação matemática existentes é uma versão bastante simplificada do caso real, restringindo (ou até mesmo invalidando), por esta ótica, seu uso como ferramenta para se gerar quadros de horário.

A modelagem de um problema de programação de horários em escolas não é uma tarefa simples. Entre as questões importantes a serem observadas, está a de estabelecer uma função objetivo adequada. O PHE não possui uma função objetivo tão natural quanto às de muitos outros problemas de otimização. Em geral, o valor ótimo da função objetivo obtido na prática, de quadros de horário feitos manualmente, exprime o resultado de uma negociação entre professores, turmas e a própria instituição de ensino, iteradas por várias tentativas de conciliar interesses legítimos, mas muitas vezes mutuamente excludentes. É, portanto, uma tarefa complexa construir uma função objetivo que exprima fidedignamente a interação entre os inúmeros aspectos a considerar. A solução que parece mais adequada e é amplamente utilizada consiste em dividir as restrições do problema em dois subconjuntos: um, contendo as restrições que, em hipótese alguma, podem ser violadas e outro, cujas restrições, apesar de o atendimento a elas serem desejáveis, podem ser violadas caso necessário. O atendimento a essas últimas restrições é, então, avaliado por uma função objetivo linear, baseada em penalidade. Utilizamos esse mecanismo para mensurar o atendimento aos requisitos estabelecidos na seção 4.2.

Assim, apresentamos, a seguir, um modelo de programação matemática, baseado em penalidade, para o problema abordado.

5.2 O Modelo

Na escola Dom Silvério, cada turma recebe h horários diários de aula durante d dias da semana (de segunda a sexta-feira), totalizando $p = d \times h$ aulas semanais. As aulas são realizadas em horários consecutivos, no caso, em um único turno (manhã, tarde ou noite). As turmas têm aula em todos os p horários.

Nessa escola, um quadro de horário que não satisfaz a pelo menos um dos requisitos (a), ..., (e) estabelecidos na seção 4.2 é considerado inviável e não pode ser colocado em prática. Entretanto, o não atendimento aos requisitos (f) e (g), apesar de criar insatisfação entre os professores, não torna o quadro de horário inaceitável. Assim, justifica-se considerar o atendimento a esses dois requisitos como objetivos a serem alcançados. Para tanto, consideramos uma função objetivo que leva em consideração esses dois requisitos, aos quais associamos pesos, de forma a expressar a importância relativa de cada um deles.

De forma a facilitar a modelagem dos diversos requisitos estabelecidos na seção 4.2, utilizaremos, diferentemente da notação comumente usada na literatura, variáveis tetradimensionais. Para não carregar a notação utilizada admitiremos, sem prejuízo de generalidade, que todos os professores requerem aulas duplas, assim como todos desejam uma agenda compacta.

Sejam, pois, m professores, n turmas, d dias da semana, h horários diários e $p = d \times h$ horários semanais de aula concentrados em um único turno.

Considere as variáveis de decisão x_{ijkl} e V_{ik} definidas por:

$$x_{ijkl} = \begin{cases} 1 & \text{se o prof. } i \text{ é alocado à turma } j \text{ no dia } k \text{ e horário diário } l \\ 0 & \text{caso contrário} \end{cases}$$

$$V_{ik} = \begin{cases} 1 & \text{se o prof. } i \text{ é alocado à alguma turma no dia } k \\ 0 & \text{caso contrário} \end{cases}$$

Seja B_{ik} o número de *buracos* (horários ociosos entre dois horários de aula de um mesmo turno e dia) na agenda do professor i no dia k e D_{ij} a diferença, não negativa, entre o número de aulas duplas requeridas pelo professor i para a turma j e o número efetivamente atendido.

A função objetivo, a ser minimizada, que avalia um quadro Q de horário, é:

$$f(Q) = \sum_{i=1}^{m} \sum_{k=1}^{d} \alpha_i B_{ik} + \sum_{i=1}^{m} \sum_{k=1}^{d} \beta_i V_{ik} + \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_i D_{ij} \tag{5.1}$$

As duas primeiras componentes mensuram a compacidade de um quadro de horário (requisito (g)) e a última o atendimento ao requisito (f). α_i, β_i e γ_i são pesos que refletem, na agenda de cada professor, a importância relativa de cada uma das componentes associadas.

Mostraremos, a seguir, como modelar cada um dos requisitos estabelecidos na seção 4.2, bem como avaliar cada uma das componentes da função objetivo.

(a) *Sobreposição de turmas*: Em um dia k e horário l, uma turma j tem aula no máximo com um único professor:

$$\sum_{i=1}^{m} x_{ijkl} \leq 1 \quad \forall j, k, l \tag{5.2}$$

(b) *Sobreposição de professores*: Em um dado dia k e horário l, um professor i leciona no máximo para uma única turma:

$$\sum_{j=1}^{n} x_{ijkl} \leq t_{ikl} \quad \forall i, k, l \tag{5.3}$$

onde $t_{ikl} = \begin{cases} 1 & \text{se o prof. } i \text{ está disponível no dia } k\text{, horário } l \\ 0 & \text{caso contrário} \end{cases}$

(c) *Carga horária*: Cada professor deve ministrar exatamente um certo número de aulas semanais para uma dada turma:

$$\sum_{k=1}^{d} \sum_{l=1}^{h} x_{ijkl} = r_{ij} \quad \forall i, j \tag{5.4}$$

onde r_{ij} é a carga horária semanal do professor i para a turma j.

(d) *Respeito às indisponibilidades dos professores*: Estas restrições encontram-se contempladas ao utilizarmos a matriz t_{ikl} nas restrições 5.3.

(e) *Limite diário de aulas por matéria*: Uma turma não pode ter mais do que um certo número de aulas diárias de uma mesma matéria:

$$\sum_{l=1}^{h} x_{ijkl} \leq u_{ij} \quad \forall i, j, k \tag{5.5}$$

onde u_{ij} é o número máximo diário de aulas do professor i (responsável pela matéria) para a turma j.

(g.1) *Presença em um dado dia*: Para verificar se um professor i encontra-se ou não em atividade de ensino em um dado dia k, basta avaliarmos as desigualdades:

$$V_{ik} \geq x_{ijkl} \quad \forall i, j, k, l \tag{5.6}$$

(g.2) *Número de buracos*: Para calcular o número de horários ociosos entre dois horários de aula de um mesmo turno de um professor, introduziremos duas novas variáveis P_{ik} e U_{ik}, assim definidas:

$$U_{ik} = \begin{cases} l & \text{se } l \in \{1, 2, \ldots, h\} \text{ é o horário da última aula do prof. } i \text{ no dia } k \\ 0 & \text{se o professor } i \text{ não é alocado a alguma turma no dia } k \end{cases}$$

$$P_{ik} = \begin{cases} l & \text{se } l \in \{1, 2, \ldots, h\} \text{ é o horário da primeira aula do prof. } i \text{ no dia } k \\ 0 & \text{se o professor } i \text{ não é alocado a alguma turma no dia } k \end{cases}$$

O número de buracos diários B_{ik} na agenda de cada professor pode, então, ser avaliado de acordo com a expressão:

$$B_{ik} \geq U_{ik} - P_{ik} + V_{ik} - \sum_{j=1}^{n} \sum_{l=1}^{n} x_{ijkl} \quad \forall i, k \tag{5.7}$$

Observamos que o maior número de buracos ocorre quando há somente duas aulas, sendo a última lecionada no último horário e a primeira aula, no primeiro horário. Desta forma, as variáveis B_{ik} em 5.7 são limitadas superiormente a $h - 2$.

Avaliação de U_{ik}: A última aula diária de cada professor i pode ser determinada impondo-se a restrição:

$$U_{ik} \geq l \sum_{j=1}^{n} x_{ijkl} \quad \forall i, k, l \tag{5.8}$$

Observamos que não é necessário definir $U_{ik} \in \{0, 1, 2, \ldots, h\}$ como variável inteira, pois sendo positivo seu coeficiente na equação 5.7 e estando B_{ik} na função objetivo de minimização também com coeficiente positivo, o valor de U_{ik} será o mínimo do segundo membro da expressão 5.8, constituído por operações de soma e multiplicação entre inteiros.

Avaliação de P_{ik}: A primeira aula diária de cada professor i pode ser determinada impondo-se a restrição:

$$P_{ik} \leq (h+1)V_{ik} - (h+1-l)\sum_{j=1}^{n} x_{ijkl} \quad \forall i, k, l \tag{5.9}$$

Assim como U_{ik}, também não é necessário definir P_{ik} como variável inteira.

A primeira parcela do segundo membro dessa equação é um artifício para modelar a situação em que o professor i não tem aula no dia k.

(f) *Atendimento ao pedido de aulas duplas*: Seja s_{ij} o número mínimo de aulas duplas requeridas pelo professor i para a turma j. Então, o pleito do professor i para a turma j, medido pela diferença D_{ij} não-negativa entre o número requerido e o efetivamente existente no quadro corrente, pode ser avaliado com base na expressão:

$$D_{ij} \geq s_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{h-1} x_{ijkl}x_{ijk,l+1} \quad \forall i, j \tag{5.10}$$

a qual é válida desde que o número de aulas diárias não supere 2 unidades, isto é, que $u_{ij} \leq 2$.

Dado que o segundo membro da equação 5.10 realiza somente operações de soma e multiplicação entre inteiros, não é necessário definir D_{ij} como variável inteira.

Observamos, também, que as variáveis D_{ij} são limitadas superiormente ao quociente da divisão inteira de r_{ij} por 2 subtraído do número mínimo de aulas duplas requeridas, isto é, $D_{ij} \leq \lfloor r_{ij}/2 \rfloor - s_{ij}$.

Mostraremos, a seguir, um procedimento para linearizar as restrições 5.10, o qual consiste em, para cada produto de variáveis binárias, criar uma variável real não negativa e três restrições lineares.

Tendo em vista que existe uma restrição desse tipo para cada par professor-turma que deseja ter aulas duplas, que uma aula dupla refere-se a uma aula ministrada em um mesmo dia e que, portanto, apenas o último índice l da variável x_{ijkl} se altera em um dado dia, consideraremos, para clareza de entendimento, que essas variáveis binárias são unidimensionais. Seja, então, a restrição a seguir, onde as variáveis x_l são binárias e c e s são inteiros:

$$c \geq s - x_1 x_2$$

Fazendo-se $x_1 x_2 = y_{12}$ e impondo-se $x_1 + x_2 + y_{12} \leq 1$, $y_{12} \leq x_1$, $y_{12} \leq x_2$, com $y_{12} \geq 0$, a restrição não linear anterior é equivalente [76] ao seguinte conjunto de restrições lineares:

$$\begin{gathered} x_1 + x_2 + y_{12} \leq 1 \\ y_{12} \leq x_1 \\ y_{12} \leq x_2 \\ c \geq s - y_{12} \\ x_1, x_2 \in \{0, 1\} \\ y_{12} \geq 0 \end{gathered}$$

Sendo assim, fazendo $x_{ijkl} x_{ijk,l+1} = y_{ijkl,l+1}$, onde $y_{ijkl,l+1} \geq 0$, a restrição 5.10 pode ser linearizada substituindo-a pelo seguinte conjunto de restrições:

$$x_{ijkl} + x_{ijk,l+1} - y_{ijkl,l+1} \leq 1 \quad \forall i, j, k \text{ e } \forall l = 1, \ldots, h-1 \tag{5.11}$$

$$x_{ijkl} - y_{ijkl,l+1} \geq 0 \quad \forall i, j, k \text{ e } \forall l = 1, \ldots, h-1 \tag{5.12}$$

$$x_{ijk,l+1} - y_{ijkl,l+1} \geq 0 \quad \forall i, j, k \text{ e } \forall l = 1, \ldots, h-1 \tag{5.13}$$

$$x_{ijkl} + x_{ijk,l+1} - y_{ijkl,l+1} \leq 1 \quad \forall i, j, k \text{ e } \forall l = 1, \ldots, h-1 \tag{5.14}$$

$$D_{ij} \geq s_{ij} - \sum_{k=1}^{d} \sum_{l=1}^{h-1} y_{ijkl,l+1} \quad \forall i, j \tag{5.15}$$

$$y_{ijkl,l+1} \geq 0 \quad \forall i, j, k \text{ e } \forall l = 1, \ldots, h-1 \tag{5.16}$$

Em resumo, o problema de horário em escolas abordado pode ser formulado como o seguinte problema de programação linear inteira mista:

$$\min f(Q) = \sum_{i=1}^{m}\sum_{k=1}^{d} \alpha_i B_{ik} + \sum_{i=1}^{m}\sum_{k=1}^{d} \beta_i V_{ik} + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_i D_{ij} \tag{5.1}$$

s.a:

$$\sum_{i=1}^{m} x_{ijkl} \leq 1 \quad \forall j,k,l \tag{5.2}$$

$$\sum_{j=1}^{n} x_{ijkl} \leq t_{ikl} \quad \forall i,k,l \tag{5.3}$$

$$\sum_{k=1}^{d}\sum_{l=1}^{h} x_{ijkl} = r_{ij} \quad \forall i,j \tag{5.4}$$

$$\sum_{l=1}^{h} x_{ijkl} \leq u_{ij} \quad \forall i,j,k \tag{5.5}$$

$$V_{ik} \geq x_{ijkl} \quad \forall i,j,k,l \tag{5.6}$$

$$B_{ik} \geq U_{ik} - P_{ik} + V_{ik} - \sum_{j=1}^{n}\sum_{l=1}^{n} x_{ijkl} \quad \forall i,k \tag{5.7}$$

$$U_{ik} \geq l \sum_{j=1}^{n} x_{ijkl} \quad \forall i,k,l \tag{5.8}$$

$$P_{ik} \leq (h+1)V_{ik} - (h+1-l) \sum_{j=1}^{n} x_{ijkl} \quad \forall i,k,l \tag{5.9}$$

$$x_{ijkl} + x_{ijk,l+1} - y_{ijkl,l+1} \leq 1 \quad \forall i,j,k \text{ e } \forall l = 1,\ldots,h-1 \tag{5.11}$$

$$x_{ijkl} - y_{ijkl,l+1} \geq 0 \quad \forall i,j,k \text{ e } \forall l = 1,\ldots,h-1 \tag{5.12}$$

$$x_{ijk,l+1} - y_{ijkl,l+1} \geq 0 \quad \forall i,j,k \text{ e } \forall l = 1,\ldots,h-1 \tag{5.13}$$

$$x_{ijkl} + x_{ijk,l+1} - y_{ijkl,l+1} \leq 1 \quad \forall i,j,k \text{ e } \forall l = 1,\ldots,h-1 \tag{5.14}$$

$$D_{ij} \geq s_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{h-1} y_{ijkl,l+1} \quad \forall i,j \tag{5.15}$$

$$y_{ijkl,l+1} \geq 0 \quad \forall i,j,k \text{ e } \forall l = 1,\ldots,h-1 \tag{5.16}$$

$$V_{ik} \in \{0,1\} \quad \forall i,k \tag{5.17}$$

$$x_{ijkl} \in \{0,1\} \quad \forall i,j,k,l \tag{5.18}$$

$$0 \leq B_{ik} \leq (h-2) \quad \forall i,k \tag{5.19}$$

$$0 \leq D_{ij} \leq \lfloor r_{ij}/2 \rfloor - s_{ij} \quad \forall i,j \tag{5.20}$$

$$0 \leq P_{ik} \leq h \quad \forall i,k \tag{5.21}$$

$$0 \leq U_{ik} \leq h \quad \forall i,k \tag{5.22}$$

5.3 Pré-processamento

Evidentemente, nem todos os professores requerem aulas duplas. Para contornar essa situação, basta criar variáveis D_{ij} e acionar as restrições associadas (5.11, 5.12, 5.13, 5.14, 5.15 e 5.16), apenas quando o número de aulas duplas requeridas por um professor (s_{ij}) for diferente de um determinado *flag*, como por exemplo, quando $s_{ij} \neq -1$.

A mesma regra anterior se aplica quando o professor não se importar com o número de dias em que deve comparecer à escola, assim como com o número de buracos em sua agenda. Assim, as variáveis V_{ik}, B_{ik}, P_{ik} e U_{ik} somente são criadas quando necessário.

Quando o número de aulas semanais de um professor i superar $(d-1) \times h$, isto é, $\sum_{j=1}^{n} r_{ij} \geq (d-1) \times h + 1$, deve-se prefixar V_{ik} em 1 $\forall k$, o que significa dizer que para cumprir sua carga horária contratual, não há como o professor i deixar de comparecer à instituição todos os dias da semana.

Quando a carga horária do professor i for igual ao número de horários reservados para a realização das aulas, isto é, $\sum_{j=1}^{n} r_{ij} = p$, deve-se prefixar B_{ik} em 0 $\forall k$ (não criando-se, evidentemente, as variáveis U_{ik} e P_{ik} envolvidas em seu cálculo), haja visto que o professor i está com agenda cheia.

Dois testes simples podem ser aplicados para verificar a existência, a priori, de inviabilidade. O primeiro verifica se o número de professores disponíveis em um dado dia k e horário l é inferior ao número n de turmas, isto é, se $\sum_{i=1}^{m} t_{ikl} < n$ para algum k, l, o problema não tem solução. O outro teste verifica se para cada horário da semana, todos os professores que dão aula para uma dada turma estão indisponíveis. Ou seja, se I_j é o conjunto dos professores que lecionam para a turma j, então não existe um quadro viável de horário se $\sum_{i \in I_j} t_{ikl} = 0$ para algum j, k, l.

5.4 Cortes

Apresentamos, a seguir, alguns cortes que podem ser inseridos no modelo apresentado de forma a melhorar o desempenho da técnica *branch-and-bound*.

O número mínimo de dias de aula que cada professor i deve visitar semanalmente cada turma j pode ser calculado com base na expressão:

$$\sum_{k=1}^{d} V_{ik} \geq \left\lceil \frac{r_{ij}}{u_{ij}} \right\rceil \quad \forall i, j$$

Entretanto, tendo em vista todas as turmas e de forma a diminuir o número de restrições, podemos substituí-la, por um pré-processamento, por:

$$\sum_{k=1}^{d} V_{ik} \geq \max_{j=1,\ldots,n} \left\lceil \frac{r_{ij}}{u_{ij}} \right\rceil \quad \forall i$$

Por outro lado, o número de dias de aula de um professor não é inferior ao quociente entre sua carga horária total e o número de horários diários, isto é:

$$\sum_{k=1}^{d} V_{ik} \geq \left\lceil \left(\sum_{j=1}^{n} r_{ij} \right) / h \right\rceil \quad \forall i$$

Reunindo os dois resultados anteriores, podemos estabelecer o seguinte limite inferior para o número de dias de aula de um professor:

$$\sum_{k=1}^{d} V_{ik} \geq \max \left\{ \left\lceil \left(\sum_{j=1}^{n} r_{ij} \right) / h \right\rceil, \max_{j=1,\ldots,n} \left\lceil \frac{r_{ij}}{u_{ij}} \right\rceil \right\} \quad \forall i \tag{5.23}$$

Já o limite superior para o número de dias de aula de um professor é dado por:

$$\sum_{k=1}^{d} V_{ik} \leq d \quad \forall i \tag{5.24}$$

5.5 Redução do Número de Restrições

As restrições 5.6 podem ser substituídas pelo seguinte conjunto reduzido de restrições:

$$V_{ik} \geq (1/h) \sum_{j=1}^{n} \sum_{l=1}^{h} x_{ijkl} \ \ \forall i, k \tag{5.25}$$

as quais, segundo [69], contudo, produzem relaxações de programação linear mais fracas.

5.6 Extensões do Modelo

Mostraremos, a seguir, como modelar alguns outros requisitos que podem ser exigidos em outras escolas secundárias brasileiras.

5.6.1 Simultaneidade de Aulas

Alguns pares de aulas requerem ser ministrados simultaneamente. Esta restrição é usada para que turmas se juntem seja para terem uma atividade comum, seja para se subdividirem em dois outros grupos diferentes, de forma a terem atividades distintas. É o caso das aulas de educação física de algumas escolas: juntam-se duas turmas em um mesmo horário para subdividí-las em dois grupos, um de mulheres, outro, de homens tendo, cada grupo, aula com um professor diferente.

Para modelá-la, seja IJ_s o conjunto dos pares de aulas que precisam ser simultâneas, isto é:

$$IJ_s = \{\langle(i_1, j_1), (i_2, j_2)\rangle : i_1, i_2 \in I_s \text{ e } j_1, j_2 \in J_s\}$$

onde I_s é o conjunto dos professores que requerem aulas simultâneas e J_s o conjunto das turmas associadas aos professores de I_s.

Então, esta restrição pode ser modelada como:

$$x_{i_1 j_1 kl} - x_{i_2 j_2 kl} = 0 \quad \forall k, l \text{ e } \forall \langle(i_1, j_1), (i_2, j_2)\rangle \in IJ_s \tag{5.26}$$

5.6.2 Número de Recursos Especiais

Em um dado horário, apenas um número limitado de recursos especiais (laboratórios, projetores, etc.) encontra-se disponível para os professores que os requerem:

$$\sum_{i \in I_R} \sum_{j \in J_R(i)} x_{ijkl} \leq nrd_{tkl} \quad \forall t, k, l \tag{5.27}$$

onde nrd_{tkl} é o número de recursos do tipo t disponíveis no dia k horário l, I_R é o conjunto dos professores que requerem o recurso do tipo t e $J_R(i)$ é o conjunto das turmas associadas aos professores i que requerem o recurso do tipo t.

5.6.3 Pré-alocações

Algumas aulas precisam ser pré-alocadas a determinados horários:

$$x_{ijkl} \geq a_{ijkl} \quad \forall i, j, k, l$$

onde $a_{ijkl} = 1$ se há pré-alocação do professor i à turma j no dia k, horário l e $a_{ijkl} = 0$ caso contrário.

A restrição anterior, entretanto, pode ser implicitamente contemplada no modelo apresentado na seção 5.2, bastando, para tanto, definir a variável x_{ijkl} da seguinte forma: $x_{ijkl} = 1$ se o professor i está disponível para encontrar a turma j no dia k, horário l para uma aula que não está pré-alocada e $x_{ijkl} = 0$, caso contrário. Neste caso, evidentemente, o segundo membro da equação 5.4 deve ser atualizado, isto é:

$$r_{ij} = r_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{h} a_{ijkl}$$

Atualização idêntica deve ser feita para corrigir o termo u_{ij} na desigualdade 5.5.

Na restrição 5.3, devemos considerar $t_{ikl} = 1$ se o professor i está disponível e não pré-alocado a nenhuma turma no dia k, horário l e $t_{ikl} = 0$, caso contrário.

As restrições envolvidas no cálculo da primeira e última aulas diárias, bem como aquelas relativas ao cálculo da freqüência do professor à instituição devem, também, evidentemente, incorporar o fato de o professor já se encontrar pré-alocado a determinados horários. Assim, as equações 5.6, 5.8 e 5.9 devem ser substituídas, respectivamente, pelas seguintes:

$$V_{ik} \geq x_{ijkl} + a_{ijkl} \quad \forall i, j, k, l \tag{5.28}$$

$$U_{ik} \geq l \sum_{j=1}^{n} (x_{ijkl} + a_{ijkl}) \quad \forall i, k, l \tag{5.29}$$

$$P_{ik} \leq (h+1)V_{ik} - (h+1-l) \sum_{j=1}^{n} (x_{ijkl} + a_{ijkl}) \quad \forall i, k, l \tag{5.30}$$

5.6.4 Indisponibilidade de Turmas

Algumas turmas podem estar indisponíveis em determinados horários. Assim, tomando-se $c_{jkl} = 1$ se a turma j está disponível no dia k, horário l e $c_{jkl} = 0$, caso contrário, a equação 5.3 deve ser substituída por:

$$\sum_{i=1}^{m} x_{ijkl} \leq c_{jkl} \quad \forall j, k, l \tag{5.31}$$

Esta restrição se aplica, evidentemente, somente quando a carga horária das turmas que têm alguma indisponibilidade for estritamente inferior ao número de horários reservados para as aulas, isto é, $\sum_{i=1}^{m} r_{ij} < p \ \ \forall j$. Caso contrário, o problema é inviável.

Capítulo 6

Heurísticas aplicadas ao PHE

Este capítulo tem como objetivo aplicar as metaheurísticas descritas no capítulo 2 ao problema de horários em escolas abordado no capítulo 4.

Na seção 6.1 justificamos o uso de métodos de busca local para resolver problemas de programação de horários. Para clareza de entendimento e para tornar este capítulo, de certa forma, independente dos demais, reapresentamos, na seção 6.2, a formulação do problema, assim como introduzimos a notação utilizada ao longo do capítulo. A seção 6.3 mostra como um quadro de horário de professores é codificado. Na seção 6.4 descrevemos o tipo de vizinhança considerado, enquanto na seção 6.5 discriminamos a função objetivo usada para avaliar um quadro de horário. A seção 6.6 descreve um procedimento, baseado em caminhos mínimos, usado para fazer uma busca local em um quadro de horário. As seções 6.8 a 6.12 mostram como as metaheurísticas descritas no capítulo 2 são adaptadas para tratar do problema do horário de escolas, mostrando, também, como inserir esse procedimento nessas técnicas. Na seção 6.13 mostramos como tratar outros requisitos, além daqueles relacionados ao problema abordado. Na última seção indicamos como estender o procedimento baseado em caminhos mínimos para que o mesmo possa ser utilizado no caso de existência de turmas com cargas horárias diferentes. Mostramos, também, como utilizar esse procedimento no caso de serem exigidas aulas simultâneas.

6.1 Introdução

A utilização de técnicas heurísticas para tratar problemas de horário é justificada pela intratabilidade do problema [42], o qual pode ser resolvido exatamente somente para pequenas instâncias.

As técnicas de busca local, por sua vez, são adequadas tanto para construir (inclusive interativamente) um quadro de horário, quanto para mantê-lo. De fato, de posse de um quadro de horário gerado por um procedimento qualquer, que contemple um certo conjunto de restrições, podemos usá-lo como ponto de partida para uma nova pesquisa que inclua outras restrições. A manutenção de um quadro de horário é, também, facilitada. Por exemplo, se um professor for substituído por outro, que tenha disponibilidades diferentes, basta substituir as indisponibilidades do professor antigo pelas do novo professor e recomeçar a busca pelo quadro de horário anterior.

Essa capacidade de se poder construir um quadro de horário de forma interativa é largamente reconhecida pela comunidade científica como crucial para o desenvolvimento de sistemas automatizados de construção de quadros de horário [93]. A esse respeito, os métodos de busca local superam outros métodos, tais como os métodos construtivos e as técnicas exatas.

6.2 Formulação do Problema

No (PHE) abordado, os seguintes requisitos, relativos ao quadro de horário dos professores, devem ser satisfeitos:

(a) um professor não pode ministrar aula para mais de uma turma ao mesmo tempo;

(b) uma turma não pode ter aula com mais de um professor em um mesmo horário;

(c) cada professor i tem que cumprir sua carga horária semanal r_{ij} para cada turma j;

(d) um professor não pode ser alocado a um horário no qual não esteja disponível;

(e) uma turma não pode ter mais do que u_{ij} horários diários de aula de uma mesma matéria;

(f) atendimento ao maior número possível de aulas duplas para os professores que as requererem;

(g) a agenda dos professores deve ser tão compacta quanto possível.

Definição 6.2.1 *Um quadro de horário de professores que não satisfaz a pelo menos um dos requisitos (a), ..., (e) é considerado* ***inviável****.*

Definição 6.2.2 *Um quadro Q de horário de professores se diz* ***inviável do tipo 1*** *se, em pelo um horário, uma das condições a seguir for satisfeita:*

i) Existe turma tendo aula com mais de um professor (restrição (b) não é verificada);

ii) Existe turma sem aula;

Definição 6.2.3 *Um quadro Q de horário de professores se diz* ***inviável do tipo 2*** *se a restrição (e) não for satisfeita para algum professor.*

6.3 Representação do Problema

Um quadro de horário de professores é representado como uma matriz $Q_{m \times p}$ de valores inteiros, onde em cada linha i de Q representa-se a alocação semanal do professor i. Cada elemento $q_{ik} \in \{-1, 0, 1, 2, \ldots, n\}$ indica a atividade do professor i no horário k. Valores negativos indicam que o professor está indisponível, enquanto valores nulos indicam inatividade no horário.

6.4 Estrutura da Vizinhança

Um movimento consiste na simples troca de dois valores distintos e não negativos de uma dada linha de Q. Tal movimento é identificado pela tripla $< i,\ k,\ \bar{k} >$, onde k e $\bar{k}$ representam os horários nos quais as atividades q_{ik} e $q_{i\bar{k}}$ do professor i serão permutadas, sendo $q_{ik} \neq q_{i\bar{k}} \neq -1$.

Observamos que esse tipo de movimento pode produzir inviabilidade dos tipos 1 e/ou 2, isto é, ao permutar os horários de duas atividades de um mesmo professor, permite-se que sejam geradas as seguintes inviabilidades: a) o professor seja alocado a uma turma que já tem um outro professor previamente alocado; b) uma turma fique sem aula (no caso de uma das atividades representar uma turma que tem aula em um dos horários somente com o professor em questão) e c) o professor ultrapasse o limite diário de aulas da matéria sob sua responsabilidade.

Entretanto, a possibilidade de que um professor ensine simultaneamente para mais de uma turma (violação à restrição (a)) é automaticamente rejeitada pela representação escolhida.

6.5 A Função Objetivo

A função objetivo é construída da forma usual. Isto é, o conjunto de requisitos listados na seção 6.2 é dividido em subconjuntos, cada qual representado em uma função objetivo linear de penalidade, assegurando uma importância maior às restrições consideradas *essenciais*. Assim, a função objetivo busca medir a diferença entre um quadro corrente e um quadro ideal de horário, isto é, um quadro de horário que satisfaz a todos os requisitos definidos na seção 6.2. Mais especificamente, um quadro Q é avaliado com base na seguinte função objetivo, a qual deve ser minimizada:

$$f(Q) = \omega * f_1(Q) + \delta * f_2(Q) + \rho * f_3(Q) \tag{6.1}$$

onde as duas primeiras componentes mensuram o nível de inviabilidade dos tipos 1 e 2, respectivamente, e a terceira, o nível de satisfação dos professores com relação ao atendimento de seus requisitos pessoais.

ω, δ e ρ são pesos que refletem a importância relativa de cada uma das componentes de f. De forma a gerar uma estrutura hierárquica, tais pesos são escolhidos satisfazendo a condição: $\omega > \delta \gg \rho$.

6.5.1 Avaliação da inviabilidade do tipo 1

O nível de inviabilidade do tipo 1 de Q, $f_1(Q)$, é mensurado somando-se, para cada horário k: (a) o número de vezes l_k que uma turma está sem atividade em k; (b) o número de vezes s_k que mais de um professor dá aula para uma mesma turma no horário k.

$$f_1(Q) = \sum_{k=1}^{p} (l_k + s_k) \tag{6.2}$$

6.5.2 Avaliação da inviabilidade do tipo 2

Com relação ao nível de inviabilidade do tipo 2, o quadro Q é avaliado somando-se o número de vezes e_i que a restrição (e) não é atendida para cada professor i, isto é:

$$f_2(Q) = \sum_{i=1}^{m} e_i \tag{6.3}$$

6.5.3 Avaliação dos requisitos pessoais

A satisfação ao atendimento dos requisitos pessoais dos professores é medida com relação à compacidade do quadro de horários (restrição (g)), bem como ao atendimento do número de aulas duplas requeridas (restrição (f)). Mais precisamente,

$$f_3(Q) = \sum_{i=1}^{m} (\alpha_i * B_i + \beta_i * V_i + \gamma_i * D_i) \tag{6.4}$$

onde α_i, β_i e γ_i são pesos e refletem, respectivamente, a importância relativa do número de buracos B_i (horários ociosos entre dois horários de aula de um mesmo turno), do número de dias V_i na semana que cada professor está envolvido em alguma atividade de ensino em um mesmo turno e da diferença não-negativa D_i entre o número mínimo requerido de aulas duplas e o efetivamente existente na agenda corrente de cada professor i. $\left(D_i = \max\left\{0,\ \text{duplas}(Q_i^{requerido}) - \text{duplas}(Q_i^{corrente})\right\}\right)$

6.6 Procedimento Intraturmas-Interturmas

Intraturmas-Interturmas (II) é um procedimento baseado em caminhos mínimos que é acionado quando uma solução sem inviabilidade do tipo 1 está disponível. Primeiramente, ele procura restaurar a viabilidade do quadro de horário, isto é, zerar a parcela f_2 da função objetivo 6.1 definida à pp. 99. Se bem sucedido, ele tenta, numa segunda etapa, melhorar os requisitos de qualidade exigidos para o quadro de horário, navegando no espaço das soluções viáveis, isto é, ele tenta diminuir o valor da componente f_3 respeitando as restrições (a),...,(e).

Alvarez-Valdes et al. [5] usam um procedimento semelhante, mas apenas para melhorar um quadro de horário viável. O tipo de movimento considerado nesse trabalho é, também, mais restrito do que o de II. Mais precisamente, o procedimento proposto não permite que as aulas duplas possam mudar de horário, ao contrário de II. Ademais, várias características desse procedimento são aqui melhor exploradas.

Como a forma de atuação de II tanto para restaurar a viabilidade quanto para melhorar um quadro de horário viável é semelhante, mostraremos seu princípio de funcionamento apenas para o segundo caso.

6.6.1 Procedimento Intraturmas

Suponhamos, então, disponível uma solução sem inviabilidade.

Assim, dado um quadro Q de horário de professores nessas condições ($f_1(Q) = f_2(Q) = 0$), definimos o grafo da turma j por $G_j = (V_j, A_j)$, onde V_j é o conjunto dos horários reservados para a turma j e A_j é o conjunto dos arcos orientados definido conforme a seguir:

$A_j = \Big\{(k, \bar{k}) :$ a aula da turma j lecionada no horário k pode ser lecionada também no horário $\bar{k}$ sem violar os requisitos (a), ..., (e) da seção 6.2$\}$

Da construção do grafo G_j deduz-se que para um arco orientado $(k, \bar{k})$ pertencer ao conjunto A_j as seguintes condições precisam ser atendidas:

(i) o professor do horário k precisa estar disponível no horário $\bar{k}$;

(ii) as restrições relativas à aula do horário k precisam estar, também, satisfeitas no horário $\bar{k}$.

A cada arco $(k, \bar{k}) \in G_j$ associamos um custo $\Delta f_i(k, \bar{k})$, o qual representa a variação do custo de se transferir o professor i do horário k para o horário $\bar{k}$, tendo em vista a componente f_3 da função objetivo 6.1. Desta forma, o custo é obtido calculando-se a diferença entre os valores da função objetivo, relativa ao professor, nas configurações nova e antiga, isto é:

$$\Delta f_i(k, \bar{k}) = f_i(\bar{k}) - f_i(k) \tag{6.5}$$

sendo $f(.) = (\rho * f_3)(.)$

A tabela 6.1 mostra um fragmento de um quadro Q_1 de horário de professores. Cada linha i representa um professor ($i = P1, P2, P3, P4$) e cada coluna k um horário ($k = H1, H2, H3, H4, H5$) de um mesmo dia. Cada elemento q_{ik} desta tabela representa a atividade do professor i no horário k. A, B, C e D são turmas.

Um traço (–) significa que o professor está indisponível, enquanto uma célula vazia indica que não há atividade no horário. A coluna f_i indica o valor da função objetivo de cada professor, calculada conforme 6.1 e 6.4, com $\rho = 1$, $\alpha_i = 1$ e $\beta_i = \gamma_i = 0\ \forall i$. Para esse quadro de horário, tem-se $f(Q_1) = f_{P1}+f_{P2}+f_{P3}+f_{P4} = 1+1+0+0 = 2$.

Tabela 6.1: Quadro Q_1

	H1	H2	H3	H4	H5	f_i
P1	A		B	B		1
P2	B	C		A	A	1
P3		B	A	C	B	0
P4	C	A	C	D	–	0

A figura 6.1 ilustra G_A, o grafo de horários da turma A. Cada horário é representado por um vértice, ao qual está associado um professor. O arco $(H1, H5)$ de custo -1 indica, por exemplo, que se o professor $P1$ mudar sua aula do horário $H1$ para o horário $H5$, haverá uma diminuição no valor da sua função objetivo de 1 unidade ($\Delta f_{P1}(H1, H5) = f_{P1}(H5) - f_{P1}(H1) = 0 - 1 = -1$).

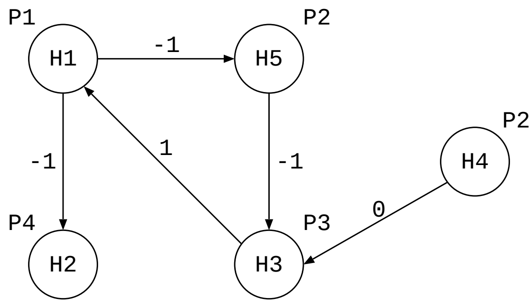

Figura 6.1: G_A, Grafo da Turma A

Para encontrar um quadro de horário com um valor menor para a função objetivo 6.1, é suficiente procurar um ciclo de custo negativo em G_j. Observa-se que a existência de um ciclo garante a re-alocação de todas as aulas nele envolvidas, preservando o atendimento aos requisitos (a), ..., (d). O atendimento ao requisito (e) será discutido mais adiante.

No exemplo em questão, a sequência de arcos $\{(H1, H5), (H5, H3), (H3, H1)\}$ forma um ciclo de custo total -1 ($= -1 + (-1) + 1$). Tal sequência, por envolver aulas de uma mesma turma, define um conjunto de movimentos ditos *intraturmas*.

A tabela 6.2 mostra Q'_1, o quadro de horário dos professores após os movimentos e a figura 6.2 ilustra G'_A, o novo grafo de horários da turma A. Em função dos movimentos promovidos em G_A, o novo quadro Q'_1 tem função objetivo dada por $f(Q'_1) = f_{P1} + f_{P2} + f_{P3} + f_{P4} = 0 + 0 + 1 + 0 = 1$.

Tabela 6.2: Quadro Q'_1

	H1	H2	H3	H4	H5	f_i
P1			B	B	A	0
P2	B	C	A	A		0
P3	A	B		C	B	1
P4	C	A	C	D	-	0

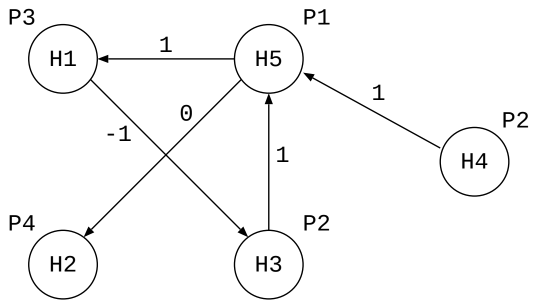

Figura 6.2: G'_A, Grafo da Turma A após os movimentos

Após a atualização do quadro de horário e do grafo correspondente da turma A, procuramos pela existência de novos ciclos de custo negativo.

Como pode ser visto pela figura 6.2, não existem mais tais ciclos no grafo G'_A do exemplo considerado. A idéia, então, é repetir o procedimento para uma outra turma e assim sucessivamente até que não seja mais possível melhorar o quadro de horário dos professores através de movimentos intraturmas.

A existência de um ciclo de custo negativo pode, todavia, não garantir melhora no valor da função objetivo, assim como pode gerar soluções inviáveis do tipo 2 (vide [97]), conforme exemplifica-se a seguir.

A tabela 6.3 representa um fragmento de um quadro Q_2 de horário de professores, de 2 dias com 5 horários cada. A função objetivo f é avaliada conforme 6.1 e 6.4, com $\rho = 1$, $\alpha_i = 1$, $\beta_i = 2$ e $\gamma_i = 0$ $\forall i$ (Lembre-se que estamos partindo do pressuposto que um quadro viável está disponível, isto é, que $f_1(Q_2) = f_2(Q_2) = 0$). A este quadro de horário está associado uma função objetivo com valor $f(Q_2) = f_{P1} + f_{P2} + f_{P3} + f_{P4} = 5 + 4 + 4 + 2 = 15$.

A figura 6.3, que ilustra o grafo de horários da turma A relativo a este quadro, mostra que existe um ciclo de custo -1 ($= -1 + 0 + 0 + 0$).

Tabela 6.3: Quadro Q_2

	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	f_i
P1	A		A			A					5
P2		A						B	B	A	4
P3				A	A		A				4
P4								A	A		2

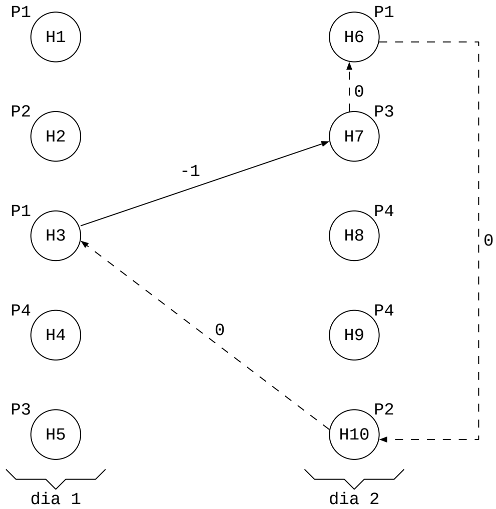

Figura 6.3: Exemplo de ciclo de custo negativo que não representa melhora no valor da função objetivo

O quadro Q'_2 resultante (tabela 6.4) mostra que, na realidade, o valor da função objetivo aumentou em 1 unidade ($f(Q'_2) = f_{P1}+f_{P2}+f_{P3}+f_{P4} = 6+4+4+2 = 16$), ao invés de diminuir de uma unidade, conforme se esperava.

Outra situação que pode ocorrer está exemplificada pela figura 6.4. Supondo que cada professor possa ministrar no máximo dois horários de aula por dia para cada turma, são possíveis os movimentos das aulas dos horários $H6$ e $H11$ para o dia 1, tomados individualmente. No entanto, se os arcos correspondentes a esses movimentos fizerem parte de um mesmo ciclo de custo negativo, tal como o mostrado na figura 6.4, a viabilidade não estará garantida.

Essas situações, entretanto, só ocorrem quando um mesmo professor participa

Tabela 6.4: Quadro Q'_2

	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	f_i
P1	A						A			A	6
P2		A	A					B	B		4
P3				A	A	A					4
P4								A	A		2

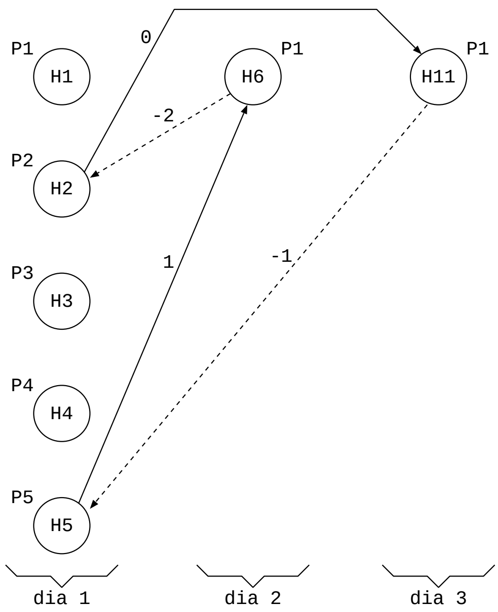

Figura 6.4: Exemplo de ciclo de custo negativo que produz inviabilidade do tipo 2.

em mais de um arco no ciclo. Assim, faz-se necessário checar a viabilidade e o valor da função objetivo após os movimentos candidatos. De forma a encontrar outros ciclos de custo negativo em um grafo nessas condições, procedemos como segue: Escolhemos um arco qualquer do ciclo, $(k, \overline{k}) \in G_j$, e o inserimos em uma lista L de movimentos proibidos. A seguir, atualizamos o grafo da turma sob avaliação, excluindo de G_j os arcos pertencentes à L, e procuramos outro ciclo de custo negativo. Quando não mais for possível encontrar ciclos de custo negativo, passamos para uma outra turma e zeramos a lista L.

Propomos, portanto, um procedimento iterativo que analisa, a cada vez, uma turma. Para cada turma j construímos seu grafo G_j e aplicamos um algoritmo para detectar ciclos de custo negativo. Enquanto houver ciclo de custo negativo que melhore o valor da função objetivo e não produza inviabilidade, realizamos os movimentos, atualizamos o quadro de horário e o grafo G_j. Inexistindo mais tais ciclos no grafo da turma j partimos para uma nova turma. Esse procedimento é encerrado quando nenhum movimento de melhora for possível para todas as turmas. O procedimento Intraturmas, descrito na figura 6.5, formaliza esse mecanismo.

```
procedimento Intraturmas(Q, f(.))
j ← 1 ;    {Número da turma}
novarodada ← false ;
enquanto ( j ≤ n ) faça
    enquanto ( houver ciclo de custo negativo em G_j
                que melhore a função objetivo
                e não viole a viabilidade ) faça
        Atualize G_j e o Quadro de Horário da turma j ;
        novarodada ← true ;
    fim-enquanto;
    se ( novarodada = true  e  j = n )
        então
            j ← 1 ;
            novarodada ← false ;
        senão
            j ← j + 1 ;
    fim-se;
fim-enquanto;
fim Intraturmas;
```

Figura 6.5: Procedimento Intraturmas

6.6.2 Procedimento Interturmas

Ao final do procedimento Intraturmas podem restar, ainda, arcos de custo negativo nos grafos das turmas. A figura 6.6, que considera os grafos de duas turmas, j e $\bar{j}$, ilustra tal situação.

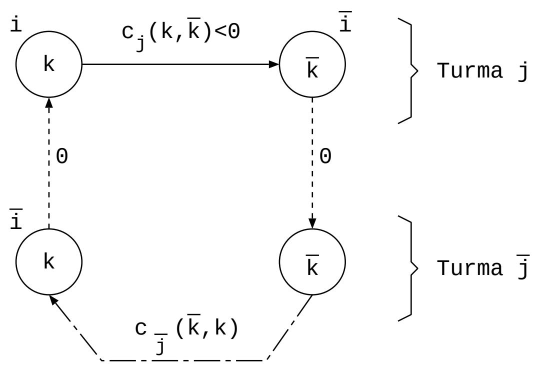

Figura 6.6: Grafos das turmas j e $\bar{j}$ resultantes do procedimento Intraturmas

Nesta figura, há um arco de custo negativo na turma j, de k a $\bar{k}$, a saber $c_j(k, \bar{k})$, sinalizando que poderá haver uma melhora no valor da função objetivo se o professor i tiver sua aula do horário k transferida para o horário $\bar{k}$.

Esta transferência, entretanto, não pode ocorrer porque o professor $\bar{\imath}$, do horário $\bar{k}$, não está disponível no horário k (Neste horário, ele está lecionando para a turma $\bar{\jmath}$).

A idéia, então, é trocar o horário dessas aulas do professor $\bar{\imath}$, de forma a permitir a procura de um ciclo de custo negativo envolvendo os grafos das duas turmas, conectados nos horários k e $\bar{k}$.

Seja $c_{\bar{\jmath}}(\bar{k}, k)$ o custo do caminho mínimo de $\bar{k}$ a k em $G_{\bar{\jmath}}$. A existência de um ciclo de custo negativo envolvendo as duas turmas pode ser observada verificando se a condição $c_j(k, \bar{k}) + c_{\bar{\jmath}}(\bar{k}, k) < 0$ é satisfeita ao transferir o professor $\bar{\imath}$ da turma j para a turma $\bar{\jmath}$ no horário $\bar{k}$ e da turma $\bar{\jmath}$ para a turma j no horário k (arcos de custo nulo da figura 6.6, já que não há custos envolvidos na transferência de um professor de uma turma para outra em um mesmo horário).

Desta forma, para cada arco $(k, \bar{k}) \in G_j$, de custo negativo, pesquisamos a existência de um ciclo de custo negativo envolvendo esse arco de G_j, o grafo $G_{\bar{\jmath}}$ e os arcos de custo nulo que os conectam. Esta sequência de arcos, por envolverem turmas diferentes, define os chamados movimentos *interturmas*. Assim como em Intraturmas, faz-se necessário checar a viabilidade e o valor da função objetivo antes e depois dos movimentos candidatos.

O procedimento Interturmas, descrito na figura 6.7, formaliza essa idéia.

Observamos que toda vez que o Interturmas produz uma solução melhor, o procedimento Intraturmas é novamente acionado (linha 12 da figura 6.7). Isso se deve ao fato de que o quadro de horário dos professores poderá ser melhorado com movimentos intraturmas, em função das alterações promovidas nos quadros de horário das turmas j e $\bar{\jmath}$.

6.6.3 Procedimento II

O procedimento II consta de dois procedimentos, os quais são usados em conjunto. O primeiro, Intraturmas, realiza somente movimentos intraturmas e sua aplicação resulta em n grafos G_j, $\forall j = 1, ..., n$, com, possivelmente, arcos de custo negativo. O segundo, Interturmas, realiza movimentos interturmas envolvendo esses arcos de custo negativo e aciona Intraturmas sempre que há uma melhora no valor da função objetivo.

```
procedimento Interturmas(Q, f(.))
  j ← 1 ;    {Número da turma}
  novarodada ← false ;
  enquanto ( j ≤ n ) faça
      para ( todo arco (k, k̄) ∈ G_j de custo negativo ) faça
          Seja ī o professor do horario k̄ ;
          Seja j̄, j̄ ≠ j, a turma na qual ī está em atividade no horário k ;
          Seja c_j(k, k̄) o custo de se transferir, na turma j,
              a aula do horário k para o horário k̄ ;
          Seja c_j̄(k̄, k) o custo, na turma j̄, do caminho mínimo
              entre k̄ e k ;
          se ( c_j(k, k̄) + c_j̄(k̄, k) < 0
              e função objetivo melhorar
              e viabilidade não for violada )  então
              Atualize G_j e o Quadro de Horário da turma j ;
              Atualize G_j̄ e o Quadro de Horário da turma j̄ ;
              Aplique o Algoritmo Intraturmas ;
              novarodada ← true ;
          fim-se;
      fim-para;
      se ( novarodada = true  e  j = n )
          então
              j ← 1;
              novarodada ← false ;
          senão
              j ← j + 1;
      fim-se;
  fim-enquanto;
fim Interturmas;
```

Figura 6.7: Procedimento Interturmas

Conforme mencionado no início da seção 6.6, o procedimento II é acionado tão logo haja uma solução sem inviabilidade do tipo 1. II atua em duas etapas: primeiramente, ele tenta recuperar a viabilidade. Sendo bem sucedido, ele tenta, em uma segunda etapa, melhorar os requisitos de qualidade do quadro de horário. Na primeira etapa II usa somente a componente f_2 da função objetivo como função avaliadora dos custos dos movimentos, isto é, a expressão 6.5 é avaliada tomando-se unicamente $f(.) = f_2(.)$. Na etapa seguinte a expressão 6.5 volta a ser avaliada usando-se a função 6.1, no caso, tomando-se $f(.) = (\rho * f_3)(.)$, uma vez que $f_1(Q) = f_2(Q) = 0$.

O algoritmo II encontra-se descrito pela figura 6.8.

procedimento $II(Q, f(.))$
Seja Q solução inicial satisfazendo $f_1(Q) = 0$;
Seja Q^* a melhor solução resultante de II;
se $(f_2(Q) \neq 0)$ então
    $Q \leftarrow Intraturmas(Q, f_2)$;
    se $(f_2(Q) \neq 0)$ então $Q \leftarrow Interturmas(Q, f_2)$;
fim-se;
se $(f_2(Q) = 0)$ então
    $Q \leftarrow Intraturmas(Q, f_3)$;
    $Q \leftarrow Interturmas(Q, f_3)$;
fim-se;
$Q^* \leftarrow Q$;
Retorne Q^*;
fim II;

Figura 6.8: Procedimento II

6.7 Geração de uma Solução Inicial

A solução inicial é gerada de forma construtiva, via uma procedimento parcialmente guloso, conforme a seguir se descreve.

Inicialmente, determinamos os horários mais críticos, isto é, aqueles que têm o menor número de professores disponíveis. A seguir, enumeramos e ordenamos todas as aulas, priorizando aquelas mais difíceis de serem alocadas, ou seja, aquelas que têm os professores com maior carga horária e maior número de horários indisponíveis. Formamos, então, uma lista restrita de candidatos dessas aulas, de tamanho $|LCR|$, e selecionamos, aleatoriamente, uma delas. A seguir, procuramos alocá-la (usando a ordem de horários críticos) de forma que não haja, a princípio, nenhum tipo de inviabilidade (no caso, violação às restrições (b) e (e) da seção 6.2). Não sendo possível, admitimos, agora, violação somente à restrição (e). Se ainda assim a alocação não for possível, a aula é alocada admitindo-se violação também à restrição (b). Toda vez que uma aula é alocada, atualizamos os horários críticos, bem como as aulas mais difíceis remanescentes.

Observamos que, como todas as aulas são alocadas, ainda que com algum tipo de inviabilidade, a restrição (c) da seção 6.2 é automaticamente verificada.

A figura 6.9 apresenta o pseudo-código correspondente.

Observamos, também, que o tamanho $|LCR|$ da lista de candidatos restrita e o

procedimento *ConstruaSolucaoInicial*

Seja $I(h_k)$ o número de professores indisponíveis no horário h_k;

Seja HC o conjunto dos p horários h_k ordenados segundo sua criticidade:

$$HC \leftarrow \{h_1, h_2, \ldots, h_p\}$$

$h_k \in HC$ satisfaz $I(h_k) \geq I(h_{k+1}) \ \ \forall k = 1, \ldots p-1$;

Seja $CH(a_j)$ a carga horária do prof. da j-ésima aula a_j somada ao número de seus horários indisponíveis;

Seja AD o conjunto das q aulas a_j ordenadas segundo sua dificuldade de alocação:

$$AD \leftarrow \{a_1, a_2, \ldots, a_q\}$$

$a_j \in AD$ satisfaz $CH(a_j) \geq CH(a_{j+1}) \ \ \forall j = 1, \ldots q-1$;

Seja $Q^0 \leftarrow \emptyset$ o quadro de horário dos professores;

Seja $TipoAloc = \begin{cases} 3 & \text{se a aula é alocada sem inviabilidade} \\ 2 & \text{se a aula é alocada com inviabilidade do tipo 2} \\ 1 & \text{se a aula é alocada com inviabilidade do tipo 1} \end{cases}$

enquanto $(AD \neq \emptyset)$ faça
    Seja a_j uma das $|LCR|$ aulas mais difíceis;
    $t = 3$;
    repita
        $k = 1$;
        repita
            se (a_j pode ser alocada ao horário h_k com *TipoAloc* t)
                então $Q^0 \leftarrow Q^0 \cup \{a_j \rightarrow h_k\}$
                senão $k \leftarrow k + 1$;
        até ($k > p$ ou a_j seja alocada com *TipoAloc* t);
        Se (a_j não foi alocada com *TipoAloc* t)
            então $t \leftarrow t - 1$;
    até (que a_j seja alocada);
    $AD \leftarrow AD - \{a_j\}$;
    Atualize e reordene os conjuntos AD e HC;
fim-enquanto;
Retorne Q^0;

fim *ConstruaSolucaoInicial*

Figura 6.9: Geração de uma solução inicial valor inicial t permitido para o tipo de alocação *TipoAloc* são os únicos parâmetros do procedimento *ConstruaSolucaoInicial*. Se tivermos $|LCR| = 1$ e $t = 3$, gera-se uma solução gulosa; se $|LCR|$ for igual ao número de aulas ainda a alocar e $t = 1$, será gerada uma solução aleatória.

6.8 Algoritmo de Busca Tabu

Partindo de uma solução inicial, gerada pelo procedimento construtivo, a meta-heurística de Busca Tabu (BT) segue, iterativamente, explorando toda a vizinhança $N(Q)$ da solução corrente Q, através de movimentos definidos na seção 6.4 e guiados pela função objetivo expressa em 6.1. O algoritmo segue, então, para o vizinho Q' que produzir o menor valor $f(Q')$, independentemente de ele ser pior que o valor $f(Q)$ corrente.

De forma a prevenir a ocorrência de ciclagem, a cada vez que um movimento é realizado, ele é armazenado em uma lista tabu T. Essa lista tabu contém os $|T|$ movimentos mais recentes e reduz o risco de se revisitar uma das $|T| - 1$ últimas soluções visitadas anteriormente.

Como a lista tabu pode ser muito restritiva [57], o algoritmo BT usa também uma função de aspiração $A(f(Q))$, onde $A(f(Q)) = f(Q^\star) - 1$. Dessa forma, um movimento perde seus *status* tabu se produzir uma solução Q' cujo valor seja inferior ao da melhor solução, $Q^\star$, obtida até então.

Sempre que uma solução sem sobreposições é gerada (isto é, $f_1(Q) = 0$), acionamos o procedimento Intraturmas-Interturmas, com a condição de que uma solução de mesmo valor ainda não tenha sido submetida a II. Evidentemente, soluções com um mesmo valor não são necessariamente iguais. Entretanto, esta condição foi imposta por dois motivos: (1) pela eficiência do procedimento II, mostrada nos testes realizados e (2) para não elevar desnecessariamente o tempo de processamento do algoritmo.

Neste ponto, consideramos duas variantes, *BT-IIm* e *BT-II*, conforme a seguir descreveremos.

Na primeira, *BT-IIm*, o procedimento de Busca Tabu continua sua busca a partir da solução produzida por II. Neste caso, tendo em vista que II faz mudar a região pesquisada, incluímos na lista tabu T os movimentos reversos àqueles realizados por

II caso, evidentemente, a busca tenha logrado êxito, isto é, os movimentos tenham sido de melhora relativamente ao valor da solução corrente e nenhuma inviabilidade tenha sido gerada.

A figura 6.10 apresenta o pseudo-código da variante *BT-IIm* do algoritmo básico de Busca Tabu usando o procedimento II.

procedimento *BT-IIm(Q)*
Obtenha uma solução inicial Q;
$Q^\star \leftarrow Q$; {Melhor solução obtida até então}
$Iter \leftarrow 0$; {Contador do número de iterações}
$MelhorIter \leftarrow 0$;{Iteração mais recente que forneceu Q^*}
$T \leftarrow \emptyset$; {Lista Tabu}
$BTmax \leftarrow$ Número máximo permitido de iterações consecutivas sem melhora em $f(Q^\star)$;
Inicialize a função A de aspiração;
<u>enquanto</u> $(Iter - MelhorIter < BTmax)$ <u>faça</u>
  $Iter \leftarrow Iter + 1$;
  Seja $Q' \leftarrow Q \oplus m$ o melhor elemento de $N(Q)$ tal que
    o movimento m não seja tabu $(m \notin T)$ <u>ou</u>
    Q' atenda a condição de aspiração $(f(Q') < A(f(Q)))$;
  Atualize a lista tabu T;
  Atualize a função de aspiração A;
  $Q \leftarrow Q'$;
  <u>se</u> $(f_1(Q') = 0)$ <u>então</u>
    $Q \leftarrow$ Intraturmas-Interturmas(Q', f);
    Atualize a lista tabu T com os movimentos reversos de II;
  <u>fim-se</u>;
  <u>se</u> $(f(Q) < f(Q^\star))$ <u>então</u>
    $Q^\star \leftarrow Q$;
    $MelhorIter \leftarrow Iter$;
  <u>fim-se</u>;
<u>fim-enquanto</u>;
Retorne $Q^\star$;
fim *BT-IIm*;

Figura 6.10: Algoritmo BT-IIm

Esta variante é uma sofisticação do algoritmo BT, idealizada em vista do bom desempenho de II no algoritmo básico de Busca Tabu, e é usada em um procedimento GRASP [46], conforme descrito na seção 6.9.

Diferentemente de *BT-IIm*, na outra variante, *BT-II*, o procedimento de Busca Tabu segue seu curso normalmente, isto é, prossegue sua busca a partir do último vizinho gerado, com II servindo apenas para fazer uma busca local nas soluções que estão em sua trajetória e satisfazem a condição de acionamento do procedimento Intraturmas-Interturmas. Para alcançar este objetivo, sem que II interferisse na trajetória do algoritmo básico, trabalhamos com uma solução ótima virtual, que não leva em conta a influência de II e é utilizada para determinar o valor corrente da função de aspiração.

Tal como em *BT-IIm*, o procedimento II somente é acionado uma única vez para cada possível valor que a solução corrente possa assumir.

A figura 6.11 apresenta o pseudo-código da variante *BT-II* do algoritmo básico de Busca Tabu usando o procedimento II.

procedimento *BT-II*(Q)
Obtenha uma solução inicial Q;
$Q^\star_{virtual} \leftarrow Q$; $\{$Melhor solução obtida por BT até então$\}$
$Q^\star \leftarrow Q$; $\{$Melhor solução obtida por BT-II até então$\}$
$Iter \leftarrow 0$; $\{$Número de iterações de BT-II$\}$
$T \leftarrow \emptyset$; $\{$Lista Tabu$\}$
Inicialize a função A de aspiração $(A(f(Q) = f(Q^\star_{virtual}) - 1)$;
enquanto (*condição de parada não satisfeita*) faça
    $Iter \leftarrow Iter + 1$;
    Seja $Q' \leftarrow Q \oplus m$ o melhor elemento de $N(Q)$ tal que
        o movimento m não seja tabu $(m \notin T)$ ou
        Q' atenda a condição de aspiração $(f(Q') < A(f(Q)))$;
    Atualize a lista tabu T;
    Atualize a função de aspiração A;
    $Q \leftarrow Q'$;
    se $(f_1(Q) = 0)$ então
        $Q' \leftarrow$ Intraturmas-Interturmas(Q, f);
        se $(f(Q') < f(Q^\star))$ então $Q^\star \leftarrow Q'$;
    fim-se;
    se $(f(Q) < f(Q^\star_{virtual}))$ então $Q^\star_{virtual} \leftarrow Q$;
fim-enquanto;
se $(f(Q^\star_{virtual}) < f(Q^\star))$ então $Q^\star \leftarrow Q^\star_{virtual}$;
Retorne Q^*;
fim *BT-IIm*;

Figura 6.11: Algoritmo BT-II

Esta variante foi considerada para estabelecer um critério mais justo de comparação com as demais metaheurísticas, já que foi dessa forma que II foi nelas incluída.

6.9 Algoritmo GBT-II

O Algoritmo GBT-II é um procedimento GRASP [46, 90], onde uma solução inicial é gerada por um procedimento construtivo parcialmente guloso (conforme seção 6.7) e o refinamento é feito através de um método de Busca Tabu (seção 6.8, figura 6.10). Quando uma solução sem inviabilidade do tipo 1 é gerada, o método *BT-IIm* aciona o procedimento Intraturmas-Interturmas (descrito na seção 6.6.3). Esta sequência de construção e refinamento é repetida até que uma determinada condição de parada seja satisfeita.

A figura 6.12 apresenta o pseudo-código do algoritmo GRASP-Busca Tabu usando o procedimento II.

procedimento *GBT-II(Q)*
Seja Q^* a melhor solução e f^* o valor associado;
$f^* \leftarrow \infty$;
enquanto (*condição de parada não satisfeita*) faça
    $Q^0 \leftarrow ConstruaSolucaoInicial$;
    $Q \leftarrow BT\text{-}IIm(Q^0)$;
    se $(f(Q) < f^*)$ então
        $Q^* \leftarrow Q$;
        $f^* \leftarrow f(Q)$;
    fim-se;
fim-para;
Retorne Q^*;
fim *GBT-II*;

Figura 6.12: Algoritmo GBT-II

6.10 Algoritmo SA-II

O algoritmo SA-II é um algoritmo baseado em *Simulated Annealing* que ativa o procedimento II sempre que uma solução sem inviabilidade do tipo 1 é gerada (isto é, quando $f_1(Q) = 0$), com a condição de que uma solução de igual valor ainda não tenha sido submetida a ele.

A justificativa para não ativar II quando uma solução de igual valor já foi analisada por II é a mesma apresentada para *BT-IIm*, qual seja: 1) eficiência de II e 2) para não elevar o tempo de processamento do algoritmo.

Apresentamos pela figura 6.13 o pseudo-código do algoritmo *SA-II*.

```
procedimento SA-II(Q)
Obtenha uma solução inicial Q;
Q* ← Q;             {Melhor solução obtida até então}
IterT ← 0;          {Número de iterações na temperatura T}
α ← α0;             {Razão de resfriamento 0 < α0 < 1}
T ← T0;             {Temperatura inicial}
SAmax ← Número máximo permitido de iterações
        na temperatura T;
enquanto (condição de parada não satisfeita) faça
    enquanto (IterT < SAMax) faça
        IterT ← IterT + 1;
        Gere um vizinho qualquer Q' ∈ N(Q);
        Δ = f(Q') − f(Q);
        se (Δ < 0)
            então
                Q ← Q';
                se (f1(Q) = 0) então
                    Q' ← Intraturmas-Interturmas(Q, f);
                se (f(Q') < f(Q*)) então Q* ← Q';
            senão
                Tome x ∈ [0, 1];
                se (x < e^(−Δ/T)) então Q ← Q';
        fim-se;
    fim-enquanto;
    T ← α × T;
    IterT ← 0;
fim-enquanto;
Retorne Q*;
fim SA-II;
```

Figura 6.13: Procedimento SA-II

6.11 Algoritmo AM-II

O algoritmo AM-II é um algoritmo baseado em *Annealing* Microcanônico, que aciona o procedimento II sempre que uma solução sem inviabilidade do tipo 1 é gerada, com a condição de que uma solução de mesmo valor não tenha sido anteriormente analisada por ele. O primeiro critério foi adotado de forma a não alterar a essência do método, o qual, em uma dada configuração E_i de energia, tendo D_i como valor inicial do demônio e D_{max} como sua capacidade máxima, só admite soluções no intervalo $[E_i - D_{max} + D_i,\ E_i + D_i]$, mesmo que uma solução de custo menor seja encontrada. O segundo critério leva em consideração a eficiência de II. A adoção desses critérios possibilita, também, uma comparação mais justa desse algoritmo com os demais.

A figura 6.14 apresenta o pseudo-código do algoritmo *Annealing* Microcanônico usando o procedimento II.

procedimento *AM-II(Q)*
Obtenha uma solução inicial Q;
$Q^* \leftarrow Q$; {Melhor solução obtida até então}
$IterE \leftarrow 0$; {Número de iterações na energia E}
$D_{max} \leftarrow D^0_{max}$; {Capacidade máxima do demônio na energia E}
$D_i \leftarrow$ Valor inicial do demônio na energia E;
$AMmax \leftarrow$ Número máximo permitido de iterações na energia E;
enquanto (*condição de parada não satisfeita*) faça
    $D \leftarrow D_i$;
    para ($IterE = 1, 2, \ldots, AMmax$) faça
        Gere um vizinho qualquer $Q' \in N(Q)$;
        $\Delta = f(Q') - f(Q)$;
        se ($\Delta \leq 0$)
            então
                se ($D - \Delta \leq D_{max}$) então
                    $Q \leftarrow Q'$;
                    $D \leftarrow D - \Delta$;
                    se ($f_1(Q) = 0$) então
                        $Q' \leftarrow$ Intraturmas-Interturmas(Q, f);
                    se ($f(Q') < f(Q^\star)$) então $Q^\star \leftarrow Q'$;
                fim-se;
            senão
                se ($D - \Delta \geq 0$) então
                    $Q \leftarrow Q'$;
                    $D \leftarrow D - \Delta$;
                fim-se;
        fim-se;
    fim-para;
    $D_{max} \leftarrow \alpha \times D_{max}$;
fim-enquanto;
Retorne $Q^\star$;
fim *AM-II*;

Figura 6.14: Procedimento AM-II

6.12 Algoritmo OM-II

O algoritmo *OM-II* é um algoritmo baseado em Otimização Microcanônica, que utiliza o procedimento II para fazer uma busca local em uma solução sem sobreposições gerada durante a fase de inicialização do algoritmo, com a condição de que uma solução de igual valor não tenha sido previamente analisada.

A figura 6.15 apresenta o pseudo-código do algoritmo Otimização Microcanônica usando o procedimento II.

procedimento *OM-II*(Q)
Obtenha uma solução inicial Q;
$Q^* \leftarrow Q$; $\{$Melhor solução obtida até então$\}$
$Iter \leftarrow 0$; $\{$Número de iterações de $OM\}$
enquanto (*condição de parada não satisfeita*) faça
    $Iter \leftarrow Iter + 1$;
    $Q' \leftarrow$ *InicializacaoOM-II*(Q)
    se $(f(Q') < f(Q^\star))$ então $Q^\star \leftarrow Q'$;
    $Q \leftarrow AmostragemOM(Q')$
fim-enquanto;
Retorne Q^*;
fim *OM-II*;

Figura 6.15: Algoritmo de Otimização Microcanônica

O procedimento *InicializacaoOM-II* encontra-se descrito pela figura 6.16. Já *AmostragemOM* é o mesmo procedimento apresentado à pp. 23, com a exceção de que não apresentamos a lista completa dos parâmetros de entrada.

6.13 Estendendo os Métodos Heurísticos

Mostramos, a seguir, como incorporar às heurísticas apresentadas alguns outros requisitos não considerados no problema abordado. Evidentemente, muitos outros requisitos poderiam ser modelados. No entanto, o mecanismo utilizado é o mesmo, tornando desnecessária sua apresentação. Enquadram-se nesse contexto requisitos tais como: limite diário de aulas por professor, preferências de professores por certos horários, limite diário de aulas de um determinado grupo de matérias, etc.

```
procedimento InicializacaoOM-II(Q);
Seja Q solução inicial ou proveniente da fase de amostragem;
Q* ← Q;                {Melhor solução obtida até então}
Iter ← 0;              {Número de iterações}
MelhorIter ← 0;        {Iteração mais recente que forneceu Q*}
Imax ← Número máximo permitido de iterações
       consecutivas sem melhora em f(Q*);
enquanto (Iter − MelhorIter < Imax) faça
    Iter ← Iter + 1;
    Gere um vizinho qualquer Q' ∈ N(Q);
    Δ = f(Q') − f(Q);
    se (Δ < 0)
        então
            Q ← Q';
            se (f1(Q) = 0) então Q' ← Intraturmas-Interturmas(Q, f);
            se (f(Q') < f(Q*)) então
                Q* ← Q';
                MelhorIter ← Iter;
            fim-se;
        senão Ponha Δ na lista dos movs rejeitados;
    fim-se;
fim-enquanto;
Retorne Q*;
fim InicializacaoOM-II;
```

Figura 6.16: Fase de Inicialização do Algoritmo de Otimização Microcanônica

6.13.1 Avaliação do Número de Recursos Disponíveis

Em certos horários, apenas um número limitado de recursos (laboratórios, quadras de esportes, etc) estão disponíveis.

Para modelar esta situação, seja nrd_{tk} o número de recursos do tipo t disponíveis no horário k e nrr_{tk} a quantidade requerida desses recursos no horário k do quadro corrente. A função f_r abaixo mensura quantas vezes a disponibilidade de recursos não é respeitada em um quadro Q:

$$f_r(Q) = \sum_{t=1}^{nr} \sum_{k=1}^{p} \max\{0, nrr_{tk} - nrd_{tk}\} \tag{6.6}$$

onde nr representa a quantidade de recursos diferentes.

A função f_r é uma medida de inviabilidade semelhante à do tipo 2, definida à pp. 98. Para manter a mesma notação, a componente f_2 da função objetivo 6.1 deve incluir f_r e, portanto, ser avaliada, agora, com base na expressão:

$$f_2(Q) = \sum_{i=1}^{m} e_i + \sum_{t=1}^{nr} \sum_{k=1}^{p} \max\{0, nrr_{tk} - nrd_{tk}\} \tag{6.7}$$

ao invés da expressão dada por 6.3.

O procedimento II pode ser aplicado normalmente para tratar esta nova situação, inclusive para tentar recuperar a viabilidade do tipo 2.

6.13.2 Espalhamento de Aulas

Requer-se, em geral, por razões pedagógicas, que as aulas de certas matérias sejam suficientemente espaçadas uma das outras ao longo da semana, para possibilitar, por exemplo, que os alunos tenham mais tempo para sedimentar o conhecimento adquirido na aula anterior.

Para modelar esta situação, seja $espr_{ij}$ o espaçamento mínimo requerido, em número de dias, entre pelo menos duas aulas do professor i para a turma j e $espe_{ij}$ o espaçamento efetivamente existente no quadro de horário corrente.

Então a função f_e, a seguir, mensura o atendimento a esse requisito em um quadro Q:

$$f_e(Q) = \sum_{i=1}^{m} \sum_{j=1}^{n} \max\{0, espr_{ij} - espe_{ij}\} \tag{6.8}$$

Como f_e é uma medida de qualidade, ela pode ser incorporada à componente f_3 da função objetivo 6.1 com um certo peso λ_i associado, isto é:

$$f_3(Q) = \sum_{i=1}^{m} (\alpha_i B_i + \beta_i V_i + \gamma_i D_i + \lambda_i \sum_{j=1}^{n} \max\{0, espr_{ij} - espe_{ij}\}) \tag{6.9}$$

6.13.3 Aulas Simultâneas

Para tratar uma aula simultânea em qualquer uma das metaheurísticas apresentadas, basta adicionar à função objetivo 6.1 (pp. 99) mais um termo que conte o número de vezes que uma alocação simultânea deixa de ocorrer. Esse termo deve ser considerado como inviabilidade do tipo 1, de forma a respeitar a classificação que adotamos para usar o procedimento II.

6.14 Estendendo o Procedimento II

Mostraremos, a seguir, como aplicar o procedimento II em duas outras situações que aparecem em algumas escolas brasileiras.

6.14.1 Turmas com Cargas Horárias Diferentes

Em algumas escolas, principalmente aquelas que trabalham com o primeiro e o segundo graus, as turmas do segundo grau têm um número maior de aulas do que as do primeiro grau. O comum nessas escolas, é o segundo grau ter 30 horas-aula semanais e o primeiro grau 25, com as aulas sendo realizadas em um único turno. Como há professores comuns a essas turmas, o planejador de horários dessas instituições reserva 30 horários, de mesma duração, para a realização das aulas. O problema que ocorre, agora, é que as turmas com carga horária menor podem ter *buracos* em suas agendas, o que é altamente indesejável.

Mostremos, então, como aplicar o procedimento II quando a carga horária de alguma turma é inferior ao número de horários disponibilizados para a realização das aulas, ou seja, se:

$$\sum_{i=1}^{m} r_{ij} < p \quad \text{para alguma turma } j$$

Para modelar essa situação, basta considerar um professor fictício $i = dummy$ para cada turma j nessas condições e atribuir-lhe $p - \sum_{i=1}^{m} r_{ij}$ horas-aula semanais.

Exemplificando, se em uma escola tem-se $p = 30$, $d = 5$, $h = 6$ e uma certa turma necessita de somente 25 horas-aula semanais, então criamos, para essa turma, um professor fictício $i = dummy$, atribuindo-lhe 5 $(= 30 - 25)$ aulas. Desta forma, a turma passará a ter carga horária completa p.

Para contornar a possibilidade de existência de *buracos* nos quadros de horário de turmas com carga horária menor, o planejador de horários normalmente fixa os horários vagos dessas turmas aos primeiros ou aos últimos horários diários. Assim, para aplicar o procedimento II preservando a compacidade do quadro de horário dessas turmas, devemos tornar indisponíveis, para esses professores fictícios, os horários não fixados. Esse mecanismo impedirá a existência de ciclos de custo negativo com arcos envolvendo professores fictícios.

No exemplo considerado, como o professor fictício teria 5 aulas e dispomos de 5 dias, poderíamos fixar uma aula diária ao professor fictício no último horário. Assim, a turma em questão teria 5 aulas por dia, realizadas nos primeiros 5 horários. Mesmo que outros professores estivessem disponíveis no sexto horário, a compacidade do quadro de horário da turma ficaria preservada, dado que inexistiria no grafo da turma qualquer arco, envolvendo professores fictícios, ligando o vértice relativo ao sexto horário com qualquer outro vértice.

6.14.2 Aulas Simultâneas

O procedimento II não opera com aulas simultâneas, mas pode considerá-las. Para preservar a simultaneidade das aulas entre as turmas envolvidas, basta impedir que elas possam mudar de horário. A única possibilidade de melhora de alocação de aulas simultâneas ficará, no caso, a cargo da metaheurística empregada.

Capítulo 7

Resultados Computacionais

Apresentamos, neste capítulo, resultados computacionais obtidos pela aplicação das heurísticas propostas no capítulo 6 a um conjunto de problemas-teste. Alguns resultados aplicando-se a formulação de programação matemática proposta no capítulo 5 também são relatados.

7.1 Problemas-teste

Apesar de o PHE ser um problema clássico da área de otimização, não existe, de nosso conhecimento, até a presente data, um conjunto consagrado de problemas-teste que possa ser usado para avaliar os algoritmos desenvolvidos no capítulo 6. As razões para este fato foram mencionadas anteriormente e dizem respeito à diversidade de regimes de ensino e características peculiares das instituições envolvidas. Existe um grupo de pesquisadores, entretanto, que está trabalhando visando à padronização do problema. Esse grupo, chamado WATT (*WorkGroup on Automated TimeTabling*) pode ser encontrado em http://www.asap.cs.nott.ac.uk/asap/watt.

Assim sendo, procedemos como em todos os trabalhos existentes na literatura sobre o assunto, isto é, consideramos um conjunto próprio de dados, relativo a uma aplicação específica. Utilizamos 7 problemas-teste, sendo o primeiro e o terceiro extraídos de [77]. O primeiro problema, na realidade, é fictício e serviu para testar, mais rapidamente, diversos parâmetros dos algoritmos. O problema 3 refere-se a dados de uma escola de ensino médio de Brasília (DF), relativos ao ano de 1989. Nesse problema exigia-se a realização de algumas aulas duplas, o que podia ser obtido considerando-se este requisito como essencial e, portanto, tratando-o como inviabilidade semelhante à do tipo 2, conforme nossa classificação à pp. 98. No

Tabela 7.1: Características dos problemas testados

Prob.	m	n	#aulas	#duplas	*dens*	#Variáveis		
						#0-1	#reais	#Total
1	8	3	75	21	0.57	452	354	806
2	14	6	150	29	0.50	1525	776	2301
3	16	8	200	4	0.70	1444	260	1704
4	23	12	300	66	0.82	2543	1105	3648
5	31	13	325	71	0.42	3115	1782	4897
6	30	14	350	63	0.48	3609	1770	5379
7	33	20	500	84	0.61	5290	2259	7549

entanto, relaxamos essa condição (mesmo sendo fácil atendê-la), de modo a tratar de forma igual todos os problemas. Os problemas 4, 2 e 6 referem-se ao planejamento de aulas dos turnos da manhã, tarde e noite, respectivamente, da escola abordada no capítulo 4, relativo ao ano de 2000. O problema 2 é o da página 84. O problema 5 refere-se a dados do ano de 1998 dessa escola, turno da manhã. O problema 7 é uma simulação de uma situação na qual essa escola teria que oferecer um número maior de turmas concentradas em um único turno. Neste último problema considera-se que os professores estão disponíveis em todos os horários e que todos os professores com mais de duas horas-aula semanais para uma mesma turma gostariam de ter, pelo menos, uma aula dupla.

Para todos os problemas considerados são disponibilizados $p = 25$ horários semanais para cada turma.

Apresentamos, pela tabela 7.1, algumas características dos problemas-teste analisados.

As colunas m, n e #aulas representam, respectivamente, o número de: professores, turmas e aulas a serem alocadas. A coluna *dens* indica a densidade do quadro de horário de professores, isto é, a razão entre a carga horária dos professores (somada ao número de horários indisponíveis) e o número de horários reservados para a realização das aulas de todas as turmas. A coluna #duplas mostra o número de aulas duplas requeridas pelos professores em cada problema. As colunas #0-1 e #reais indicam, respectivamente, o número de variáveis binárias e reais envolvidas na resolução do problema usando-se a formulação apresentada no capítulo 5, enquanto a coluna #Total totaliza a soma dessas quantidades. Estes números dizem respeito às variáveis originais da referida formulação e levam em conta o pré-processamento da

Tabela 7.2: Resultados da formulação de programação matemática

Prob	Melhor Solução*	1^a Solução viável Valor†	Tempo‡	Observações
1	210	248	26	Formulação pp. 91, c/restr. 5.25, pp. 93
2	367	367	6786	Formulação pp. 91, c/restr. 5.25, pp. 93
3	453	489	5483	Formulação pp. 91, c/restr. 5.6, pp. 88
6	1021	1021	38936	Formulação pp. 91, c/restr. 5.25, pp. 93

* Valor da melhor solução encontrada após 12 horas de processamento

† Valor da primeira solução viável encontrada

‡ Tempo gasto pelo modelador e otimizador, em segundos de CPU, para encontrar a primeira solução viável.

seção 5.3, isto é, não estão especificadas as variáveis previamente fixadas e nem as variáveis de folga e artificiais exigidas pelos métodos exatos.

7.2 Resultados da Formulação Exata

Para resolver os problemas-teste utilizando a formulação de programação matemática proposta no capítulo 5, utilizamos o modelador e otimizador XPRESS/MP [32], versão 11, com a opção da estratégia de corte automático ativada.

Os testes foram realizados em microcomputador Pentium-III, 600 MHz, de 256 MB de RAM, com sistema operacional linux.

Foram testadas duas variantes da formulação proposta à pp. 91, uma usando as restrições 5.6 à pp. 88 e outra, usando as restrições 5.25 à pp. 93.

A tabela 7.2 mostra os resultados obtidos após 12 horas de processamento, situação na qual o processo de busca foi interrompido. Nela constam apenas os resultados da formulação que apresentou a melhor solução final. Os pesos dados às componentes da função objetivo 5.1 (pp. 87) foram os seguintes: $\alpha_i = 3$, $\beta_i = 9$ e $\gamma_i = 1 \ \forall \ i = 1, ..., m$.

Nos problemas 4, 5 e 7 o otimizador não conseguiu encontrar nenhuma solução viável em 12 horas de processamento. O processo foi interrompido nesse tempo porque o otimizador gerava arquivos de *swap* com tamanhos cada vez mais proibitivos para o espaço que dispúnhamos.

Os resultados ilustrados na tabela 7.2 mostram a dificuldade de usar métodos exatos para o problema de horário enfocado neste trabalho.

Esta limitação já esperada incentivou o desenvolvimento de métodos aproxima-

Tabela 7.3: Plataformas usadas

Problemas	**Plataforma**
1, 2 e 3	Intel Celerom, 400 MHz, 64 MB de RAM
4 e 5	AMD K6-II, 450 MHz, 64 MB de RAM
6	Pentium II, 450 MHz, 128 MB de RAM
7	Pentium III, 600 MHz, 256 MB de RAM

dos ou heurísticos, cujo desempenho é mostrado a seguir.

7.3 Influência do Procedimento II nos Algoritmos

Analisamos a influência do procedimento II (descrito à pp. 107) nos algoritmos em duas situações. Na primeira, com relação à qualidade das soluções finais produzidas pelos algoritmos. Na segunda, com respeito à capacidade de cada algoritmo para gerar uma solução viável, bem como com relação ao tempo gasto nesse objetivo.

Os resultados de todos os testes são frutos da média entre 10 execuções de cada algoritmo, cada qual partindo de uma semente diferente de números aleatórios. O valor da função objetivo foi sempre arredondado.

Os algoritmos foram codificados na linguagem C e testados em microcomputadores especificados pela tabela 7.3, todos com sistema operacional Linux. Para determinar os ciclos de custo negativo em um grafo implementamos o algoritmo de *Floyd* [4], o qual detecta esta situação quando um elemento diagonal da matriz de distâncias entre todos os pares de vértices torna-se negativo.

As soluções iniciais dos algoritmos foram geradas aleatoriamente. Para tanto, utilizamos o procedimento *ConstruaSolucaoInicial*, apresentado na seção 6.7 à pp. 109, com a lista de candidatos restrita de tamanho $|LCR|$ igual ao número de aulas ainda não alocadas e permitindo-se qualquer tipo de alocação, inclusive com sobreposição de professores. Como foi mencionado na seção 2.2.1 à pp. 10 este parâmetro regula o nível de gulosidade e aleatoriedade do método.

Os pesos dados às componentes da função objetivo 6.1 (pp. 99) foram os seguintes: $\omega = 100$, $\delta = 30$, $\rho = 1$, $\alpha_i = 3$, $\beta_i = 9$ e $\gamma_i = 1 \; \forall \; i = 1, ..., m$. Privilegiamos, assim, a obtenção de soluções viáveis, já que o peso dado às componentes de inviabilidade (ω e δ) é maior que os demais. Por outro lado, dentre as soluções viáveis, valorizamos a obtenção de soluções que compactam a agenda

Tabela 7.4: Parâmetros usados em cada algoritmo

Algoritmos	**Parâmetros**
SA, SA-II	$T_i = \lfloor 0.5 \times ProspeccaoCompleta(Q)/\ln 2 \rfloor$
	$SAmax = 2 \times p \times (p-1) \times m$
	$\alpha = 0.95$
AM, AM-II	$D^0_{max} = \lfloor 2 \times ProspeccaoCompleta(Q)/\ln 2 \rfloor$
	$AMmax = 2 \times p \times (p-1) \times m$
	$\alpha = 0.95$, $D_i = 0$
OM, OM-II	$Imax = p \times (p-1) \times m$, $Amax = \lfloor 2\sqrt{p \times m \times n} \rfloor$
	D_{max} = maior Δ rejeitado na fase de inicialização
	$D_i \in [2\omega, D_{max}]$
BT, BT-II	$\lvert T \rvert = 25$, $A(f(s)) = f(s^\star) - 1$, $\lvert V \rvert = \lvert N(s) \rvert$
GBT-II	$BTmax = 250$, $\lvert T \rvert = 25$
	$A(f(s)) = f(s^\star) - 1$, $\lvert V \rvert = \lvert N(s) \rvert$
	$\lvert LCR \rvert = \max\{1, \lfloor \#\text{aulas não alocadas}/10 \rfloor\}$

dos professores, haja visto que o peso dessas componentes (β_i e α_i) é maior do que aquele dado à componente que mede a satisfação ao atendimento do pedido de aulas duplas dos professores (γ_i).

Uma fase preliminar de experimentos computacionais foi efetuada para calibrar os diversos parâmetros usados nos algoritmos, os quais estão relacionados na tabela 7.4.

Nesta tabela, para os algoritmos baseados em *Simulated Annealing* (SA e SA-II), T_i é a temperatura inicial, $SAmax$ é o número máximo de iterações em uma dada temperatura, α é a taxa de resfriamento e $ProspeccaoCompleta(Q)$ retorna o custo do pior movimento a partir de uma solução inicial Q. Para os algoritmos baseados em *Annealing* Microcanônico (AM e AM-II), D^0_{max} é a capacidade inicial do demônio, $AMmax$ é o número de iterações em um dado nível de energia, D_i é o valor do demônio no início de cada fase e α é a taxa de diminuição da capacidade do demônio. Para os algoritmos baseados em Otimização Microcanônica (OM e OM-II), $Imax$ é o número máximo de iterações na fase de inicialização, D_{max} é o custo do pior movimento dessa fase, D_i é um número aleatoriamente escolhido no intervalo $[2\omega, D_{max}]$ e representa o valor inicial do demônio na fase de amostragem, enquanto $Amax$ representa o número máximo de iterações nessa fase. Para os algoritmos baseados em Busca Tabu (BT e BT-II), $|T|$ é o tamanho da lista tabu, a qual é fixa e do tipo FIFO (*First In First Out*), $A(.)$ é a função de aspiração e $|V| = |N(s)|$ é a cardinalidade da vizinhança da solução corrente. Finalmente, para o algoritmo GBT-II, $BTmax$ é o número máximo de iterações sem melhora, o qual é usado para acionar o mecanismo de diversificação e $|LCR|$ é o tamanho da lista de candidatos restrita, usado para gerar uma solução parcialmente gulosa. Para os métodos baseados em Busca Tabu (BT, BT-II e GBT-II), dada uma solução s, pesquisamos seu melhor vizinho s' na vizinhança inteira de s.

A qualidade das soluções finais geradas pelos algoritmos foi comparada como segue. Para cada problema, considerando os parâmetros do algoritmo básico de *Simulated Annealing* especificados na tabela 7.4, determinamos o tempo gasto para que o mesmo se estabilizasse, situação na qual nenhum movimento de piora é mais aceito. Em algoritmos baseados em *Simulated Annealing* esta situação ocorre quando a temperatura está próxima de zero. Utilizamos o valor 0.1 como a temperatura na qual o método estabiliza-se. Esse tempo foi o utilizado como critério de parada para todos os demais algoritmos. No caso dos algoritmos baseados em *Annealing* Microcanônico, se a variação máxima de energia permitida (D_{max}) atingisse um valor inferior a uma unidade antes de o tempo de execução extinguir-se, eles eram novamente acionados a partir da melhor solução encontrada.

A tabela 7.5 mostra a melhor solução produzida por cada um dos algoritmos.

Tabela 7.5: Melhor solução final*

Alg.**Prob.**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
SA	268	491	601	762	1076	1100	1511
SA-II	214	361	478	717	829	841	1115
OM	259	479	606	772	1088	1109	1520
OM-II	212	357	476	721	819	831	1106
AM	296	538	697	835	1214	1218	1684
AM-II	231	364	488	782	828	851	1125
BT	216	360	500	721	837	851	1146
BT-II	212	356	479	708	818	828	1101
GBT-II	207	353	469	705	808	812	1090

* Melhor solução obtida após os seguintes tempos de execução de cada algoritmo, em segundos de CPU: (1) 90 s. (2) 280 s. (3) 380 s. (4) 870 s. (5) 1930 s. (6) 1650 s. (7) 2650 s.
** Algoritmo básico x Algoritmo com procedimento II

Como se pode observar, o procedimento II conseguiu melhorar a solução final de todos os algoritmos básicos em todos os problemas-teste. Para mensurar a influência de II, mostramos pela tabela 7.6, o percentual de melhora produzida por II no algoritmo básico que o conduz. A coluna #Média indica o percentual médio de melhora considerando todos os problemas.

Tabela 7.6: Percentual de melhora em cada problema usando II

Alg\Prob	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**#Média**
SA	26,0%	26,5%	20,5%	5,9%	23,0%	23,5%	26,2%	21,7%
OM	18,1%	25,5%	21,5%	6,6%	24,7%	25,1%	27,2%	21,2%
AM	22,0%	32,3%	30,0%	6,3%	31,8%	30,1%	33,2%	26,5%
BT	1,9%	1,1%	4,2%	1,8%	2,3%	2,7%	3,8%	2,5%

A influência típica de II é também ilustrada pelas figuras 7.1 e 7.2. A figura da esquerda mostra o comportamento dos algoritmos básicos em uma execução do problema 6, com relação à evolução da melhor solução, enquanto a da direita mostra a influência de II nesses algoritmos.

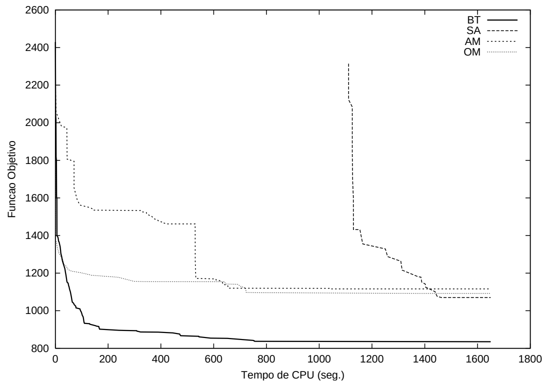

Figura 7.1: Algoritmos básicos

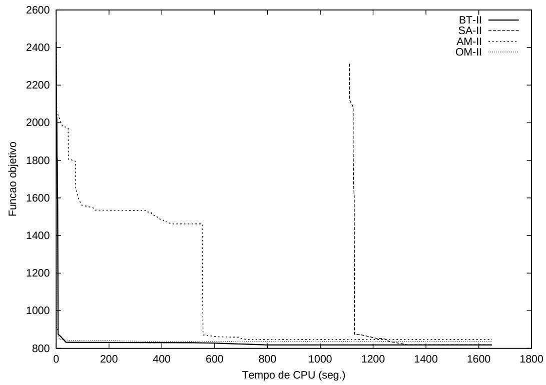

Figura 7.2: Algoritmos com II

Dentre os algoritmos básicos, observa-se que no tocante à qualidade da solução final obtida, o algoritmo BT foi o que teve melhor desempenho em todos os problemas-teste, sendo pouco influenciado por II com relação à solução final. Ele foi, inclusive, competitivo com os algoritmos que fazem uso do procedimento II. A figura 7.3, no entanto, apesar de referir-se a uma rodada de um problema específico (no caso, problema 6), revela que o comportamento do algoritmo BT é bem diferente do de BT-II.

Mostra-se, nessa figura, a evolução da melhor solução produzida pelos algoritmos BT e BT-II nos primeiros segundos de suas execuções. Apesar de as soluções finais produzidas por esses algoritmos serem relativas próximas uma da outra (ao final

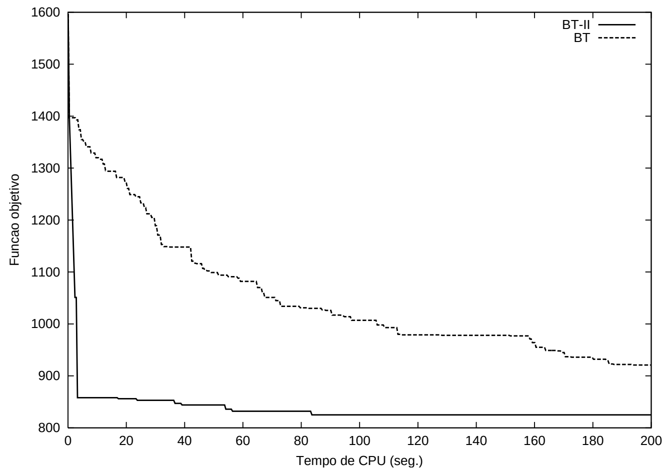

Figura 7.3: Comportamento típico dos algoritmos BT e BT-II

de 1650 segundos elas diferem entre si em cerca de 2,7%), a trajetória percorrida é bem diferente. Enquanto BT produz melhores soluções gradativamente, de forma suave, BT-II, ao contrário, gera boas soluções em um curto espaço de tempo e depois estabiliza-se até o final da busca, sem conseguir produzir soluções melhores. Se o critério de parada adotado fosse um tempo mais reduzido, a ação de II mostrar-se-ia também evidente.

Com base nessas observações, foi idealizado o algoritmo GBT-II. Mais precisamente, como a solução final (ou uma solução com valor próximo a ela) era gerada rapidamente e nenhuma melhora significativa era produzida depois de um certo número de iterações sem melhora (BTmax), a idéia foi promover uma diversificação quando esse número fosse atingido. A partir da análise desse comportamento de BT e BT-II em diversas execuções dos problemas-teste, observamos que o número BTmax = 250 mostrou-se adequado para acionar a diversificação no algoritmo GBT-II.

A figura 7.4 mostra o comportamento dos algoritmos BT, BT-II e GBT-II em uma execução do problema 3, a partir do momento em que uma solução sem sobreposições é gerada.

Observa-se que BT-II e GBT-II rapidamente encontram soluções melhores que BT, mas este, depois de 100 segundos, consegue produzir soluções competitivas, confirmando as observações feitas anteriormente. Entretanto, depois disso, nenhuma solução melhor é produzida pelos algoritmos BT e BT-II. Já o algoritmo GBT-II, ao contrário, devido ao mecanismo de diversificação, que tem por objetivo a exploração

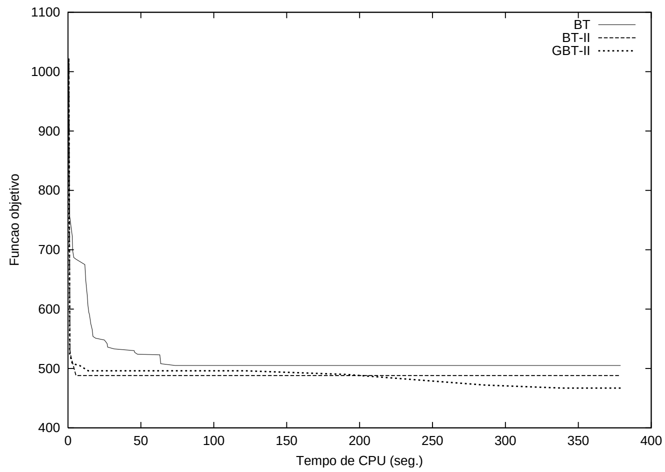

Figura 7.4: Comportamento típico dos algoritmos BT, BT-II e GBT-II

de regiões ainda não suficientemente pesquisadas, é capaz de encontrar soluções ainda melhores. A figura 7.5 ilustra a forma típica de atuação de GBT-II. A linha mais forte mostra o valor da melhor solução produzida ao longo do tempo, enquanto a linha pontilhada mostra a evolução da melhor solução em cada fase de diversificação.

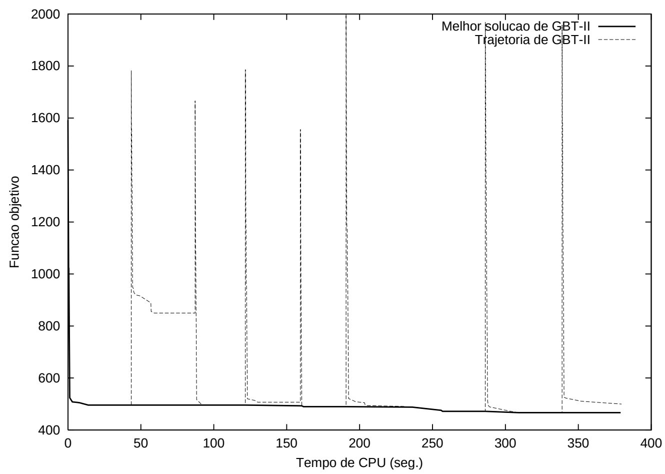

Figura 7.5: Comportamento típico do algoritmo GBT-II

Os resultados da tabela 7.5 mostram, ainda, que o algoritmo GBT-II suplantou todos os demais, tendo produzido soluções de melhor qualidade em todos os problemas-teste. O segundo melhor desempenho é o do algoritmo BT-II. O pior desempenho é claramente o do algoritmo AM, seguido de perto de OM. Para os demais algoritmos, no entanto, a superioridade de um sobre o outro não é tão evidente. Assim, estabelecemos o seguinte critério para comparar os algoritmos entre si, com relação à qualidade da solução final produzida: Para cada algoritmo atribuímos uma nota variando de 1 a 9, dependendo de sua ordem de classificação em cada problema. Mais precisamente, em cada problema, o algoritmo que produziu a melhor solução recebeu nota 9, o segundo melhor, nota 8, e assim sucessivamente. Em caso de empate entre y algoritmos, os envolvidos receberam uma mesma nota N e aqueles que os sucederam, notas começando com N - y. No gráfico de barras da figura 7.6, a altura de cada barra representa a média dessas notas.

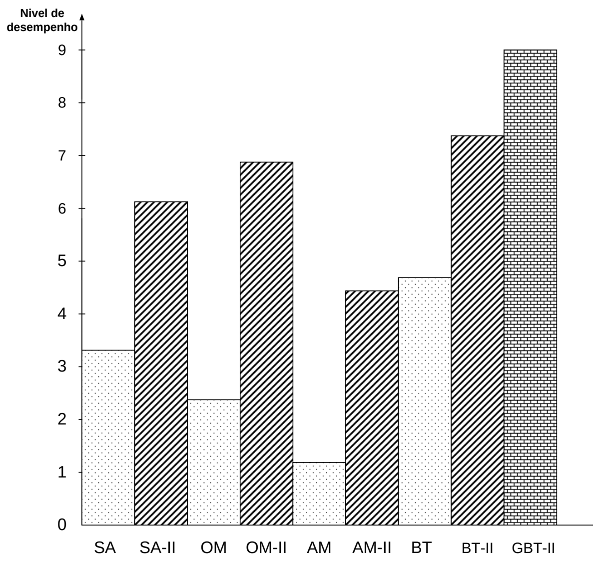

Figura 7.6: Classificação dos algoritmos

Por este critério, vê-se que, ao menos nos testes realizados, os algoritmos podem ser classificados na seguinte ordem: 1) GBT-II, 2) BT-II, 3) OM-II, 4) SA-II, 5) BT, 6) AM-II, 7) SA, 8) OM e 9) AM.

Para fazer o segundo teste, partimos também de soluções aleatórias geradas pelo procedimento *ConstruaSolucaoInicial*, definido à pp. 110, com a condição de que elas fossem inviáveis. Isto é, qualquer solução viável gerada pelo procedimento supracitado era descartada.

Inicialmente testamos cada algoritmo com relação à sua capacidade para encontrar uma solução viável.

A tabela 7.7 mostra o número de vezes em que cada algoritmo conseguiu encontrar pelo menos uma solução viável, partindo de 10 soluções iniciais inviáveis. Por ela, observa-se que o procedimento II, quando inserido no algoritmo básico correspondente, consegue ampliar o número de soluções viáveis obtidas. Esse é o caso dos problemas 2 (algoritmos AM x AM-II), 3 (algoritmos SA x SA-II e AM x AM-II),

Tabela 7.7: Capacidade de se encontrar soluções viáveis

Alg.\Prob.(*)	1	2	3	4	5	6	7
SA	10	10	7	10	9	10	10
SA-II	10	10	10	10	10	10	10
OM	10	10	10	8	10	10	10
OM-II	10	10	10	9	10	10	10
AM	9	8	4	8	9	9	7
AM-II	9	10	10	8	10	10	10
BT	10	10	10	10	10	10	10
BT-II	10	10	10	10	10	10	10
GBT-II	10	10	10	10	10	10	10

* Número de vezes em que cada algoritmo conseguiu encontrar pelo menos uma solução viável partindo de 10 soluções inviáveis, sendo a busca limitada aos seguintes tempos de execução para cada problema (em segundos de CPU): (1) 90 s. (2) 280 s. (3) 380 s. (4) 870 s. (5) 1930 s. (6) 1650 s. (7) 2650 s.

4 (algoritmos OM x OM-II), 5 (algoritmos SA x SA-II e AM x AM-II), 6 (algoritmos AM x AM-II) e 7 (algoritmos AM x AM-II). Nos problemas 2, 3, 5, 6 e 7, a utilização de II fez com que os algoritmos básicos que não obtiveram sucesso a partir de determinadas soluções iniciais, conseguissem alcançar a viabilidade em todas as 10 tentativas. Observa-se, também, que os algoritmos baseados em Busca Tabu (BT, BT-II e GBT-II) e o algoritmo SA-II foram os que tiveram melhor desempenho, tendo sido capazes de alcançar a viabilidade em todos os casos. O pior desempenho ficou por conta dos algoritmos AM e AM-II.

Apenas em duas situações II não conseguiu fazer com que o algoritmo que o conduzia alcançasse a viabilidade. Essas situações ocorreram no problema 4 com os algoritmos baseados em Otimização Microcanônica e *Annealing* Microcanônico. Analisando mais atentamente o comportamento desses algoritmos nesse problema, verificamos que os mesmos ou não conseguiram gerar soluções sem sobreposições (condição necessária para acionar II) ou geraram poucas soluções desse tipo, não dando a II chance suficiente para êxito. Mais precisamente, das 10 soluções iniciais consideradas, em 2 delas o algoritmo baseado em *Annealing* Microcanônico sequer encontrou uma solução sem sobreposições. Assim, o algoritmo AM-II comportou-se como o algoritmo AM, já que II não foi acionado. No outro caso, dentre as 10

Tabela 7.8: Tempo gasto (em segundos) para se encontrar a primeira solução viável

Alg.\Prob.$^{(\star)}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
SA	43.84	157.54	†	619.5	†	1132.95	1481.63
SA-II	40.75	157.91	228.96	597.8	1391.80	1126.28	1406.06
OM	1.45	0.75	†	†	85.68	34.77	46.12
OM-II	0.12	0.49	5.31	†	20.28	11.19	15.37
AM	†	†	†	†	†	†	†
AM-II	†	4.47	103.46	†	411.29	335.55	1011.08
BT	0.19	0.29	13.34	131.20	16.89	6.14	18.78
BT-II	0.10	0.24	10.59	128.66	15.02	5.85	14.51
GBT-II	0.11	0.22	8.88	127.54	22.58	5.85	14.62

† Não foi encontrada uma solução viável em todas as 10 tentativas, sendo a busca limitada aos seguintes tempos de execução de cada algoritmo para cada problema (em segundos de CPU): (1) 90 s. (2) 280 s. (3) 280 s. (4) 870 s. (5) 1930 s. (6) 1650 s. (7) 2650 s.

soluções iniciais, em uma o algoritmo OM-II não foi capaz de gerar uma solução viável. Foram geradas apenas 2 soluções sem sobreposições, mas o procedimento II não logrou êxito em nenhuma delas, muito embora tenha conseguido diminuir sensivelmente o nível de inviabilidade.

A tabela 7.8 compara os algoritmos com relação ao tempo gasto para encontrar a primeira solução viável. Foram comparados apenas os algoritmos que conseguiram alcançar a viabilidade em todos os testes.

Por esta tabela, verifica-se que o procedimento II acelerou a obtenção de uma solução viável em praticamente todos os algoritmos básicos. Em alguns casos, como no problema 5, a viabilidade é alcançada pelo algoritmo OM-II em um tempo quatro vezes menor do que seu algoritmo básico correspondente. Apenas em um caso, dentre 17, a inserção de II não conseguiu gerar soluções viáveis antes dos algoritmos básicos correspondentes. Esse é o caso do problema 2, em que o algoritmo SA-II chegou à viabilidade em 157.91 segundos, contra 157.54 de SA. A diferença de tempo envolvida, que é de aproximadamente 0.4 segundos, é, no entanto, insignificante frente ao benefício advindo da utilização de II, com respeito à solução final obtida. Conforme mostra a tabela 7.5, SA produz neste exemplo solução final de valor 491, enquanto SA-II, 361.

Ainda com relação à tabela 7.8, verifica-se que os algoritmos baseados em Busca Tabu demoraram mais tempo para encontrar uma solução viável no problema 4 do que nos outros problemas de dimensões maiores. A justificativa para isso reside no fato de que o problema 4 é o que tem maior maior densidade (cerca de 82%, conforme tabela 7.1) tendo, provavelmente, um número reduzido de soluções viáveis.

A partir destas análises podemos inferir que o procedimento II faz alcançar a viabilidade mais rapidamente, assim como consegue produzir soluções finais melhores do que aquelas obtidas nos algoritmos básicos correspondentes.

7.4 Comparação entre as soluções manual, exata e heurística

A tabela 7.9 compara as soluções finais produzidas pelo algoritmo heurístico GBT-II com aquelas obtidas com a formulação de programação matemática, bem como com as obtidas manualmente pelo programador de horários da escola Dom Silvério.

Tabela 7.9: Resultados das soluções manual, exata e heurística

Técnica\Problema	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Manual	†	396	†	747	952	942	†
Exata[a]	210	367	453	‡	‡	1021	‡
Heurística (GBT-II)[b]	207	353	469	705	808	812	1090
Melhora Heurística/Manual	-	10.9%	-	5.6%	15.1%	13.8%	-
Melhora Heurística/Exata	1.4%	3.8%	-3.4%	-	-	20.5%	-

† Não disponível

‡ Nenhuma solução inteira obtida após 12 horas de processamento

[a] Valor da melhor solução obtida após 12 horas de processamento

[b] Valor da melhor solução encontrada após os seguintes tempos de execução de GBT-II em cada problema (em segundos de CPU): (1) 90 s. (2) 280 s. (3) 280 s. (4) 870 s. (5) 1930 s. (6) 1650 s. (7) 2650 s.

Como se vê, o algoritmo GBT-II produziu soluções melhores que o procedimento manual em todos os casos, com o percentual de melhora variando de 5.6% a 15.1%.

O algoritmo GBT-II também gerou soluções melhores do que aquelas obtidas pela formulação de programação matemática em 12 horas de processamento. Apenas no problema 3 a heurística produziu uma solução pior, diferindo daquela em 3.4%. Entretanto, deve ser ressaltado que a solução da formulação exata foi alcançada após 12 horas de processamento e a da heurística apenas em 870 segundos.

Com relação ao tempo gasto para se encontrar a primeira solução viável, observa-se, pelas tabelas 7.2 e 7.8, que o algoritmo GBT-II alcança a viabilidade em um tempo muito menor do que aquele da formulação de programação matemática.

Capítulo 8

Conclusões e Perspectivas

Estudamos o problema de programação de horários em escolas, descrevemos suas características básicas e apontamos sua intratabilidade. Apresentamos o estado da arte com relação ao tema, detalhando alguns dos algoritmos heurísticos existentes na literatura. Descrevemos um problema de horário de uma típica escola secundária brasileira. Para esse problema, como contribuição deste trabalho, desenvolvemos uma formulação de programação matemática, além de várias metaheurísticas para tratá-lo. Contudo, a contribuição de maior impacto é um procedimento heurístico, baseado em caminhos mínimos, com capacidade para recuperar a viabilidade de um quadro de horário de professores. Esse procedimento, denotado por II, também é usado para melhorar requisitos de qualidade exigidos para o quadro de horário. Mostramos como inserir esses procedimentos II em várias metaheurísticas e comprovamos que a sua utilização não somente faz alcançar a viabilidade mais rapidamente, quanto, também, faz gerar soluções finais de melhor qualidade. A partir do desempenho das diversas metaheurísticas testadas, um algoritmo GRASP, com busca local feita por um algoritmo de Busca Tabu, foi idealizado. Esse algoritmo mostrou-se superior a todos os demais, sendo capaz de produzir soluções melhores mais rapidamente.

Mostramos, além disso, como incluir no métodos abordados algumas outras restrições que podem aparecer em escolas secundárias brasileiras.

Como trabalho futuro, apontamos a necessidade de tornar mais abrangente os métodos propostos, incluindo mais restrições, de forma que o mesmo possa ser aplicado a um maior número de instituições brasileiras de ensino.

O algoritmo GRASP apresentado também pode ser melhorado, incorporando a ele, por exemplo, uma memória de uma iteração para outra. Isto é, o tamanho da lista de candidatos restrita, ao invés de fixo, pode ser ajustado de acordo com os resultados obtidos nas iterações anteriores. Um dos objetivos deste mecanismo é o de evitar que soluções iniciais diferentes guiem a soluções finais de mesmo valor. Há experiências bem sucedidas de utilização desta tática, conhecida como GRASP reativo, em várias aplicações [85, 86, 87].

A utilização de uma estratégia adaptativa para ajustar o tamanho da lista tabu e acionar a diversificação também pode melhorar o desempenho do algoritmo GRASP proposto. O procedimento Busca Tabu Reativo [12, 10], que faz uso desta tática, guarda uma trilha das soluções geradas e aciona a diversificação sempre que uma mesma solução é frequentemente visitada, situação na qual o tamanho da lista é aumentada. Quando a busca guia a soluções ainda não exploradas ou pouco exploradas, a lista tabu tem seu tamanho reduzido.

Outra proposta futura é a paralelização dos métodos heurísticos aqui propostos que nas simulações obtiveram os melhores resultados, tais como BT e GBT, com e sem o procedimento II. A idéia de paralelizar tem como meta não só reduzir os tempos computacionais, mas também melhorar a qualidade da solução final obtida utilizando estratégias que incorporam comunicação entre os processadores.

Vislumbramos, também, a possibilidade de tratar o problema por decomposição, ainda que de forma heurística, usando técnicas de geração de colunas, relaxação lagrangeana, etc. Vários otimizadores já dispõem de facilidades para trabalhar com técnicas exatas. Além disso, a quantidade de memória requerida para tratar os problemas pode não ser um impecilho. Isto é, muitos problemas que hoje são intratáveis computacionalmente, amanhã podem não ser. Assim, há um grande espaço para a utilização de técnicas exatas diretamente ou dentro de procedimentos heurísticos.

Apêndice A

Glossário

Aula: atividade de ensino realizada em um ou mais horários consecutivos.

Aula dupla (ou geminada): aula cuja duração é de dois horários consecutivos.

Aula simples: aula cuja duração é de um horário.

Aulas disjuntas (ou separadas): aulas de uma mesma matéria, para uma mesma turma, realizadas em um mesmo dia, em dois ou mais horários não consecutivos.

Aulas simultâneas: aulas ministradas por diferentes professores para diferentes turmas e que ocorrem ao mesmo tempo;

Carga horária semanal de uma matéria é o número de horas-aula da matéria que deve ser ministrado semanalmente;

Configuração da carga horária semanal de uma matéria é cada um dos possíveis desdobramentos da carga horária semanal em uma ou mais aulas, tendo cada aula uma duração nominal especificada. Exemplo: para uma carga horária semanal de 4 horas-aula, as configurações possíveis são 4, 3-1, 2-2, 2-1-1, e 1-1-1-1;

Conflito de turmas: Duas ou mais turmas de um mesmo turno se dizem estar em conflito se exigem um mesmo recurso (professor ou sala) para a realização de suas aulas. Os quadros de horário das turmas conflitantes deve ser disjunto;

Corpo discente de um curso (Aplicável, em geral, às universidades) é o conjunto de todos os alunos inscritos no curso em um dado período letivo;

Currículo de um professor é o conjunto das matérias (ou cursos) que o professor tem que ministrar.

Currículo de uma turma (de uma escola) é o conjunto das matérias que a turma tem que cursar.

Curso (ou disciplina) é o desdobramento do conteúdo de uma matéria. Exemplo: A matéria Cálculo Diferencial e Integral pode ser fracionada em cursos de Cálculo I, Cálculo II, etc.

Disponibilidade de uma turma é o conjunto de horários em que a turma está livre para a realização de aulas;

Disponibilidade de um professor é o conjunto de horários em que o professor está livre para lecionar;

Duração de uma aula: é o número de horários consecutivos necessários para realizar a aula.

Horário é uma unidade de tempo. Em geral, um horário é de 50 minutos. Pode significar, também, uma tabela onde se especifica, no tempo, as aulas que uma ou mais turmas (ou professores) devem receber (ministrar).

Hora-aula: é a unidade de tempo de uma aula. Uma hora-aula é, em geral, de 50 minutos.

Matriz de conflito de um problema de programação de horários, envolvendo as turmas $c_1, c_2, \ldots, c_n$, é a matriz quadrada $M = (m_{ij})_{n \times n}$ tal que $m_{ij} = 1$ se, e somente se, as turmas c_i e c_j requerem um mesmo recurso (professor ou sala) e $m_{ij} = 0$, caso contrário;

Quadro de Horário (ou simplesmente, horário) de uma turma é uma tabela onde se consta todas as aulas da turma durante a semana, com seus respectivos horários de início e término;

Quadro de Horário (ou agenda ou horário) de um professor é uma tabela onde se consta todas as aulas que o professor tem que ministrar ao longo da semana, com seus respectivos horários de início e término;

Sessão de um curso é cada um dos conjuntos de aulas nos quais o curso foi configurado semanalmente. Exemplo: Um curso de Cálculo I de 5 horas-aula pode ser configurado na forma 2-2-1, isto é, dividido em 3 três sessões, a primeira e a segunda, com aulas duplas e a terceira tendo uma aula simples;

Turma (em uma escola) é o conjunto dos estudantes que fazem as mesmas matérias e estudam juntos, constituindo o menor grupo indivisível de alunos para efeito da construção do horário de aulas;

Turma de um curso (em uma universidade) é um subconjunto do corpo discente do curso, constituído por todos os alunos que participam do mesmo curso e estudam juntos, constituindo o menor grupo indivisível de alunos para efeito da construção do horário de aulas;

Apêndice B

Publicações

Listamos, a seguir, os trabalhos oriundos desta pesquisa que foram publicados e/ou apresentados em eventos científicos.

Trabalhos publicados em periódicos

Título: Melhorando Quadros de Horário de Escolas através de Caminhos Mínimos

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Periódico: Tendências em Matemática Aplicada e Computacional

Editores: E.X.L de Andrade, G.N. Silva e A. Sri Ranga

Editora: Sociedade Brasileira de Matemática Aplicada e Computacional

ISBN: 85-86883-02-6

Volume: 1 **Número**: 1 **Páginas**: 515-524 **Ano**: 2000

Trabalhos apresentados em eventos internacionais

Título: *Solving School Timetabling Problems by Microcanonical Optimization*

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Evento: *Third International Conference on Practice and Theory of Automated Timetabling* (PATAT 2000)

Editora: Springer-Verlag

Local: Konstanz, Germany

Número do trabalho: PATAT36

Período: 16 a 18 de Agosto de 2000

Título: GTS-II: Uma heurística para o problema do horário de escolas

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Evento: X Congreso Latino-Ibero-Americano de Investigación de Operaciones y Sistemas (CLAIO)

Local: Ciudad de Mexico

Número do trabalho: A179

Período: 04 a 08 de Setembro de 2000

Trabalhos apresentados em eventos nacionais

Título: Heurísticas para o problema do horário de escolas

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Evento: XXIII Congresso Nacional de Matemática Aplicada e Computacional

Local: Santos (SP)

Período: 11 a 15 de Setembro de 2000

Título: Um algoritmo de busca local para o problema do horário de escolas

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Evento: XXXI Simpósio Brasileiro de Pesquisa Operacional (SBPO)

Local: Juiz de Fora (MG)

Período: 20 a 22 de Outubro de 1999

Título: Melhorando Quadros de Horário de Escolas através de Caminhos Mínimos

Co-autores: Nelson Maculan e Luiz Satoru Ochi

Evento: XXII Congresso Nacional de Matemática Aplicada e Computacional

Local: Santos (SP)

Período: 13 a 17 de Setembro de 1999

Bibliografia

[1] Abramson, D. "Constructing school timetables using simulated annealing: sequential and parallel algorithms". *Management Science*, v. 37, pp. 98–113, 1991.

[2] Abramson, D., Abela, J. "A Parallel Genetic Algorithm for Solving the School Timetabling Problem". In *Proceedings of the 15th Australian Computer Science Conference*, Hobart, Australia, 1992.

[3] Abramson, D., Krishnamoorthy, M., Dang, H. "Simulated annealing cooling schedules for the school timetabling problem". *Asia-Pacific Journal of Operational Research*, v. 16, pp. 1–22, 1999.

[4] Ahuja, R.K., Magnanti, T.L., Orlin, J.B. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, New Jersey, 1993.

[5] Alvarez-Valdes, R., Martin, G., Tamarit, J.M. "Constructing Good Solutions for the Spanish School Timetabling Problem". *Journal of the Operational Research Society*, v. 47, pp. 1203–1215, 1996.

[6] Aubin, J., Ferland, J.A. "A large scale timetabling problem". *Computers and Operations Research*, v. 16, pp. 67–77, 1989.

[7] Aust, R.J. "An improvement algorithm for school timetabling". *The Computer Journal*, v. 19, pp. 339–343, 1976.

[8] Bardadym, V.A. "Computer-Aided School and University Timetabling: The New Wave". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 22–45. Springer-Verlag, Berlin, 1996.

[9] Barnard, S. "Stereo matching by hierarchical microcanonical annealing". In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987.

[10] Bastos, M.P., Ribeiro, C.C. "Reactive Tabu Search with Path Relinking for the Steiner Problem in Graphs". In *Proceedings of the Third Metaheuristics International Conference*, pp. 31–36, Angra dos Reis, Brazil, 1999.

[11] Battiti, R. "Reactive Search: Toward Self-Tuning Heuristics". In Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. (eds), *Modern Heuristic Search Methods*, chapter 4, pp. 61–83. John Wiley & Sons, New York, 1996.

[12] Battiti, R., Tecchiolli, G. "The Reactive Tabu Search". *ORSA Journal of Computing*, v. 6, pp. 126–140, 1994.

[13] Birbas, T., Daskalaki, S., Housos, E. "Timetabling for Greek high schools". *Journal of the Operational Research Society*, v. 48, pp. 1191–1200, 1997.

[14] Boufflet, J.P., Nègre, S. "Three Methods Used to Solve an Examination Timetable Problem". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 327–344. Springer-Verlag, Berlin, 1996.

[15] Brittan, J.N.G., Farley, F.J.M. "College timetable construction by computer". *The Computer Journal*, v. 14, pp. 361–365, 1971.

[16] Burke, E.K., Jackson, K., Kingston, J.H., Weare, R. "Automated University Timetabling: The State of the Art". *The Computer Journal*, v. 40, pp. 565–571, 1997.

[17] Burke, E.K., Newall, J.P., Weare, R.F. "A Memetic Algorithm for University Exam Timetabling". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 241–250. Springer-Verlag, Berlin, 1996.

[18] Caldeira, J.P., Rosa, A.C. "School Timetabling using Genetic Search". In *Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling*, pp. 115–122, 1997.

[19] Carter, M.W. "A survey of practical applications of examination timetabling algorithms". *Operations Research*, v. 34, pp. 193–202, 1986.

[20] Carter, M.W., Laporte, G. "Recent Developments in Practical Examination Timetabling". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 3–21. Springer-Verlag, Berlin, 1996.

[21] Carter, M.W., Laporte, G., Lee, S.Y. "Examination Timetabling: Algorithm Strategies and Applications". *Journal of the Operational Research Society*, v. 47, pp. 373–383, 1996.

[22] Carter, M.W., Tovey, C.A. "When is the classroom assignment problem hard?". *Operations Research Supplement 1*, v. 40, pp. 28–39, 1992.

[23] Chahal, N., de Werra, D. "An interactive system for constructing timetables on a PC". *European Journal of Operational Research Society*, v. 40, pp. 32–37, 1989.

[24] Colorni, A., Dorigo, M., Maniezzo, V. "Metaheuristics for High School Timetabling". *Computational Optimization and Applications*, v. 9, pp. 275–298, 1998.

[25] Cook, S. "An Overview of Computational Complexity". *Communications of the ACM*, v. 26, pp. 401–403, 1983.

[26] Cooper, T.B., Kingston, J.H. "The solution of real instances of the timetabling problem". *The Computer Journal*, v. 36, pp. 645–653, 1993.

[27] Cooper, T.B., Kingston, J.H. "The Complexity of Timetable Construction Problems". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 283–295. Springer-Verlag, Berlin, 1996.

[28] Costa, D. "A tabu search algorithm for computing an operational timetable". *European Journal of Operational Research Society*, v. 76, pp. 98–110, 1994.

[29] Creutz, M. "Microcanonical Monte-Carlo Simulation". *Physical Review Letters*, v. 50, pp. 1411–1414, 1983.

[30] Csima, J., Gotlieb, C.C. "Tests on a computer method for construction of school timetables". *Communications of the ACM*, v. 7, pp. 160–163, 1961.

[31] Dammeyer, F., Voß, S. "Dynamic tabu list management using the reverse elimination method". In Hammer, P.L. (editor), *Tabu Search*, v. 41, *Annals of Operations Research*, pp. 31–46. Baltzer Science Publishers, Amsterdan, 1993.

[32] Dash Optimization Co., http://www.dashopt.com. *XPRESS-MP - Release 11*, 1999.

[33] de Gans, O.B. "A computer timetabling system for secondary schools in the Netherlands". *European Journal of Operational Research Society*, v. 7, pp. 175–182, 1981.

[34] de Werra, D. "An introduction to timetabling". *European Journal of Operational Research Society*, v. 19, pp. 151–162, 1985.

[35] de Werra, D. "Tabu Search Techniques: A Tutorial and an Application to Neural Networks". *OR Spektrum*, v. 11, pp. 131–141, 1989.

[36] de Werra, D. "Some Combinatorial Models for Course Scheduling". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 296–308. Springer-Verlag, Berlin, 1996.

[37] de Werra, D. "The combinatorics of timetabling". *European Journal of Operational Research Society*, v. 96, pp. 504–513, 1997.

[38] Dowsland, K.A. "Simulated Annealing". In Reeves, C.R. (editor), *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series, chapter 2, pp. 20–69. Blackwell Scientific Publications, London, 1993.

[39] Drexl, A., Salewski, F. "Distribution requirements and compactness constraints in school timetabling". *European Journal of Operational Research Society*, v. 102, pp. 193–214, 1997.

[40] Eiselt, H.A., Laporte, G. "Combinatorial optimization problems with soft and hard requirements". *Journal of the Operational Research Society*, v. 38, pp. 785–795, 1987.

[41] Erben, W., Keppler, J. "A Genetic Algorithm Solving a Weekly Course-Timetabling Problem". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 198–211. Springer-Verlag, Berlin, 1996.

[42] Even, S., Itai, A., Shamir, A. "On the complexity of timetabling and multicommodity flow problems". *SIAM Journal of Computation*, v. 5, pp. 691–703, 1976.

[43] Evenborn, P. "A Column-generation Approach for School Timetabling Problems". In *Proceedings of the 15th Triennial Conference of the IFORS*, Beijing, China, 1999.

[44] Evenborn, P., Ronnqvist, M. "A Sequential Approach to Solve Hard School Timetabling Problems Using Column Generation". In *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Konstanz, Germany, 2000. Springer-Verlag.

[45] Fahrion, R., Dollansky, G. "Construction of university faculty timetables using logic programming techniques". *Discrete Applied Mathematics*, v. 35, pp. 221–236, 1992.

[46] Feo, T.A., Resende, M.G.C. "Greedy randomized adaptive search procedures". *Journal of Global Optimization*, v. 6, pp. 109–133, 1995.

[47] Ferland, J.A., Hertz, A., Lavoie, A. "An object-oriented methodology for solving assignment-type problems with neighborhood search techniques". *Operations Research*, v. 44, pp. 347–359, 1996.

[48] Ferland, J.A., Roy, S. "Timetabling problem for university as assignment of activity to resources". *Computers and Operations Research*, v. 12, pp. 207–218, 1985.

[49] Fernandes, C., Caldeira, J.P., Melicio, F., Rosa, A. "An Evolutionary Algorithm to Solve the High-School and University Timetabling Problem". In *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, Konstanz, Germany, 2000.

[50] Figueiredo, C.M.H., Szwarcfiter, J.L. "Emparelhamento em Grafos: Algoritmos e Complexidade". In *Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação*, pp. 127–161, 1996.

[51] Gendreau, M., Hertz, A., Laporte, G. "A tabu search heuristic for the vehicle routing problem". *Management Science*, v. 40, pp. 1276–1290, 1994.

[52] Glover, F. "Future paths for Integer Programming and links to Artificial Intelligence". *Computers and Operations Research*, v. 5, pp. 553–549, 1986.

[53] Glover, F. "Tabu Search: Part I". *ORSA Journal of Computing*, v. 1, pp. 190–206, 1989.

[54] Glover, F. "Tabu Search: Part II". *ORSA Journal of Computing*, v. 2, pp. 4–32, 1990.

[55] Glover, F., Hanafi, S. *Tabu Search and Finite Convergence.* To appear in *Discrete Applied Mathematics.*

[56] Glover, F., Laguna, M. "Tabu Search". In Reeves, C.R. (editor), *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series, chapter 3, pp. 70–150. Blackwell Scientific Publications, London, 1993.

[57] Glover, F., Laguna, M. *Tabu Search.* Kluwer Academic Publishers, Boston, 1997.

[58] Glover, F., Taillard, E., de Werra, D. "A user's guide to tabu search". In Hammer, P.L. (editor), *Tabu Search*, v. 41, *Annals of Operations Research*, pp. 3–28. Baltzer Science Publishers, Amsterdan, 1993.

[59] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Berkeley, 1989.

[60] Gotlieb, C.C. "The construction of class-teacher timetables". In *Proceedings of IFIP Congress*, pp. 73–77, Amsterdan, 1963.

[61] Hall, M. "An algorithm for distinct representatives". *American Math. Monthly*, v. 63, pp. 716–717, 1956.

[62] Hanafi, S. *On the Convergence of Tabu Search.* To appear in *Journal of Heuristics.*

[63] Hansen, P. "The steepest ascent mildest descent heuristic for combinatorial programming". In *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[64] Hertz, A. "Tabu search for large scale timetabling problems". *European Journal of Operational Research Society*, v. 54, pp. 39–47, 1991.

[65] Hertz, A. "Finding a feasible course schedule using tabu search". *Discrete Applied Mathematics*, v. 35, pp. 255–270, 1992.

[66] Hertz, A., de Werra, D. "The tabu search metaheuristic: how we used it". *Annals of Mathematics and Artificial Intelligence*, v. 1, pp. 111–121, 1990.

[67] Hopcroft, J.E., Karp, R.M. "A $n^{5/2}$ algorithm for maximum matching in bipartite graphs". *SIAM Journal of Computation*, v. 2, pp. 225–231, 1973.

[68] Johnson, D.S. "Experimental Analysis of Heuristics: The Good, the Bad, and the Ugly". In *Proceedings of the Third Metaheuristics International Conference*, pp. 287–289, Angra dos Reis, Brazil, 1999.

[69] Johnson, E.L., Nenmhauser, G.L., Savelsbergh, M.W.P. "Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition". *INFORMS Journal on Computing*, v. 12, pp. 2–23, 2000.

[70] Junginger, W. "Timetabling in Germany - a survey". *Interfaces*, v. 16, pp. 66–74, 1986.

[71] Kirkpatrick, S., Gellat, D.C., Vecchi, M.P. "Optimization by Simulated Annealing". *Science*, v. 220, pp. 671–680, 1983.

[72] Kwok, L.F., Kong, S.C., Kam, Y.Y. "Timetabling in Hong Kong secondary schools". *Computers & Education*, v. 28, pp. 173–183, 1997.

[73] Laporte, G., Desroches, S. "The problem of assigning students to course sections in a large engineering school". *Computers and Operations Research*, v. 13, pp. 387–394, 1986.

[74] Lawrie, N.L. "An integer linear programming model for a school timetabling problem". *The Computer Journal*, v. 12, pp. 307–316, 1969.

[75] Linhares, A., Torreão, J.R.A. "Microcanonical Optimization Applied to the Traveling Salesman Problem". *International Journal of Modern Physics C*, v. 9, pp. 133–146, 1998.

[76] Maculan, N., Lucena, A. *Otimização Linear e Inteira*. Notas de aula, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, 2000.

[77] Mata, S.S. *O problema do horário na escola de segundo grau: modelagem e implementação*. Dissertação de mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, 1989.

[78] Moscato, P. *Memetic Algorithms: A short introduction*. Technical report, UNICAMP, 1998.

[79] Moscato, P., Schaefer, A. *Local Search Techniques for Scheduling Problems*. Comunicação pessoal, 1998.

[80] Mulvey, J.M. "A classroom/time assignment model". *European Journal of Operational Research Society*, v. 9, pp. 64–70, 1982.

[81] Neufeld, G.A., Tartar, J. "Generalized Graph Colorations". *SIAM Journal on Applied Mathemathics*, v. 29, pp. 91–98, 1974.

[82] Ochi, L.S., Mendes, A., Marques, E. "A Time Table System for a Brazilian University using Genetic Algorithm". In *Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling*, Toronto, Canada, 1995.

[83] Ostermann, R., de Werra, D. "Some experiments with a timetabling system". *OR Spektrum*, v. 3, pp. 199–204, 1983.

[84] Papoulias, D.B. "The assignment-to-days problem in a school time-table, a heuristic approach". *European Journal of Operational Research Society*, v. 4, pp. 31–41, 1980.

[85] Prais, M., Ribeiro, C.C. "Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment". *INFORMS Journal on Computing*, 1998.

[86] Prais, M., Ribeiro, C.C. "Parameter Variation in GRASP Implementations". In *Proceedings of the Third Metaheuristics International Conference*, pp. 375–380, Angra dos Reis, Brazil, 1999.

[87] Prais, M., Ribeiro, C.C. "Variação de Parâmetros em Procedimentos GRASP". *Investigación Operativa*, 1999.

[88] Rankin, R.C. "Automatic Timetabling in Practice". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 266–279. Springer-Verlag, Berlin, 1996.

[89] Reeves, C.R. "Genetic Algorithms". In Reeves, C.R. (editor), *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series, chapter 4, pp. 151–196. Blackwell Scientific Publications, 1993.

[90] Resende, M.G.C. *Greedy Randomized Adaptive Search Procedures (GRASP)*. Technical report, AT&T Labs Research, 1998.

[91] Ribeiro, C.C. "Metaheuristics and Applications". In *Advanced School on Artificial Intelligence*, Estoril, Portugal, 1996.

[92] Robert, V., Hertz, A. "How to Decompose Constrained Course Scheduling Problems Into Easier Assignment Type Subproblems". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 364–373. Springer-Verlag, Berlin, 1996.

[93] Schaefer, A. "Tabu search techniques for large high-school timetabling problems". In *Proceedings of the 30th National Conference on Artificial Intelligence*, pp. 363–368, 1996.

[94] Schaefer, A. "A survey of automated timetabling". *Artificial Intelligence Review*, v. 13, pp. 87–127, 1999.

[95] Schmidt, G., Ströhlein, T. "Timetable construction - an annotated bibliography". *The Computer Journal*, v. 23, pp. 307–316, 1979.

[96] Selman, B., Levesque, H., Mitchell, D. "A new method for solving hard satisfiability problems". In *Proceedings of the 10th National Conference on Artificial Intelligence*, pp. 440–446, 1992.

[97] Souza, M.J.F., Maculan, N., Ochi, L.S. "Melhorando Quadros de Horário de Escolas através de Caminhos Mínimos". *Tendências em Matemática Aplicada e Computacional*, v. 1, pp. 515–524, 2000.

[98] Thompson, J., Dowsland, K.A. "General Cooling Schedules for a Simulated Annealing Based Timetabling System". In Burke, E.K., Ross, P. (eds), *Practice and Theory of Automated Timetabling*, v. 1153, *Lecture Notes in Computer Science*, pp. 345–363. Springer-Verlag, Berlin, 1996.

[99] Torreão, J.R.A. *Inteligência Computacional.* Notas de aula, Universidade Federal Fluminense, Niterói, 1998.

[100] Torreão, J.R.A., Roe, E. "Microcanonical Optimization Applied to Visual Processing". *Physics Letters A*, v. 122, pp. 377–382, 1980.

[101] Tripathy, A. "A Lagragean Relaxation Approach to Course Timetabling". *Journal of the Operational Research Society*, v. 31, pp. 599–603, 1980.

[102] Tripathy, A. "School timetabling - A case in large binary integer linear programming". *Management Science*, v. 30, pp. 1473–1489, 1984.

[103] Čangalović, M., Schreuder, A.M. "Exact Colouring Algorithm for Weighted Graphs applied to Timetabling Problems with Lectures of different lenghts". *European Journal of Operational Research Society*, v. 51, pp. 248–258, 1991.

[104] Wood, D.C. "A technique for colouring a graph applicable to large scale timetabling problems". *The Computer Journal*, v. 12, pp. 317–319, 1969.

[105] Wood, J., Whitaker, D. "Student centred school timetabling". *Journal of the Operational Research Society*, v. 49, pp. 1146–1152, 1998.

[106] Wright, M. "School Timetabling Using Heuristic Search". *Journal of the Operational Research Society*, v. 47, pp. 347–357, 1996.

[107] Yoshihawa, M., Kaneko, K., Nomura, Y. "A constraint-based approach to high school timetabling problems: a case study". In *Proceedings of the 12th National Conference on Artificial Intelligence*, pp. 1111–1116, 1994.