UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO PARA GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS ESCOLARES

TRABALHO DE GRADUAÇÃO

Árton Dorneles

Santa Maria, RS, Brasil

2010

DESENVOLVIMENTO DE UM APLICATIVO PARA GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS ESCOLARES

por

Árton Dorneles

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Benhur de Oliveira Stein

Co-orientador: Prof. Dr. Olinto César Bassi de Araújo

Trabalho de Graduação N. 300

Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UM APLICATIVO PARA GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS ESCOLARES

elaborado por
Árton Dorneles

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Dr. Benhur de Oliveira Stein
(Presidente/Orientador)

Profª Drª Andrea Schwertner Charão (UFSM)

Profª Drª Giliane Bernardi (UFSM)

Santa Maria, 10 de Dezembro de 2010.

“A melhor maneira de ter uma boa idéia é ter um monte de idéias.”
— LINUS PAULING

“Seu trabalho é descobrir o seu trabalho - e depois - com toda sua energia doar-se a ele.”
— BUDA

“Conversa fiada não cozinha arroz.”
— PROVÉRBIO CHINÊS

AGRADECIMENTOS

Agradeço primeiramente a minha família pelo apoio incondicional.

Um agradecimento especial ao professor Olinto pela amizade, pela colaboração e por ter me livrado do triste destino de me tornar um "micreiro".

Ao professor Haroldo pela amizade e conhecimento compartilhado.

Obrigado aos professores Benhur, Andrea, Giliane pelos seus ensinamentos durante minha vida acadêmica e por fazerem parte deste trabalho.

O meu sincero obrigado ao professor Álvaro e as professoras Camila, Marli, Mari e Dalva que contribuíram gentilmente com a avaliação deste trabalho.

Por fim, não posso deixar de agradecer aos amigos que fiz dentro e fora do curso, a sempre compreensiva Victória e todos aqueles que de alguma forma contribuiram com este trabalho.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM APLICATIVO PARA GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS ESCOLARES

Autor: Árton Dorneles
Orientador: Prof. Dr. Benhur de Oliveira Stein
Co-orientador: Prof. Dr. Olinto César Bassi de Araújo
Local e data da defesa: Santa Maria, 10 de Dezembro de 2010.

A elaboração de horários escolares é um processo de decisão complexo que envolve a resolução de um problema de otimização combinatória com um número expressivo de variáveis. Neste cenário, mesmo especialistas consomem bastante tempo para tomar manualmente uma decisão e, mesmo assim, só conseguem atender as necessidades básicas de suas instituições. Na literatura acadêmica existem muitas soluções computacionais propostas para auxiliar na resolução do problema do horário escolar, porém, além de tratarem apenas um conjunto reduzido de características da realidade, não oferecem uma interface de interação com o usuário final que seja satisfatória. Este trabalho trata do desenvolvimento de uma ferramenta computacional para auxiliar na resolução do problema de geração de horários, composta por uma interface gráfica para entrada de dados amigável e um núcleo resolvedor que faz uso de um algoritmo Busca Tabu. Os resultados comparativos obtidos demonstram que o aplicativo desenvolvido é superior a similares vendidos no mercado.

Palavras-chave: Problemas de otimização combinatória, quadro-de-horários, metaheurísticas, busca tabu.

ABSTRACT

Trabalho de Graduação
Undergraduate Program in Computer Science
Universidade Federal de Santa Maria

DEVELOPMENT OF AN APPLICATION FOR AUTOMATIC GENERATION OF SCHOOL TIMETABLE

Author: Árton Dorneles
Advisor: Prof. Dr. Benhur de Oliveira Stein
Coadvisor: Prof. Dr. Olinto César Bassi de Araújo

Preparation of a high school timetabling is a complex decision process, involving to solve a combinatorial optimization problem with a significant number of variables. In this scenario, even specialists consume a long time to take a decision manually, and yet, they are only able to satisfy the basic needs of their institutions. In the academic literature there are a plenty of methods proposed to solve the timebling high school problem, however, besides handle only a limited set of reality aspects , do not provide a satisfactory interface with the user. This work deals with the development of a computational tool to assist in solving the schedule problem, consisting of a friendly graphical interface for data entry and a core solver based on tabu search algorithm. Experimental results show that the proposed application performs better than the similar one sold on the market.

Keywords: combinatorial optimization problems, timetabling, metaheuristics, tabu search
.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IDE	Integrated Development Environment
UFSM	Universidade Federal de Santa Maria
UML	Unified Modeling Language
GCC	GNU Compiler Collection
TSI	TimeSlotIndex
H.O.P.E.	Horários Otimizados por Programa para Escolas
HOP	Horários Otimizados por Programa
SWT	Standard Widgets Toolkit

SUMÁRIO

1 INTRODUÇÃO

A elaboração de quadros de horários é uma tarefa necessária e inevitável no cotidiano das instituição de ensino. De maneira geral o processo consiste em gerar uma tabela associando professores, turmas e disciplinas em determinados horários do dia. Para que um horário seja considerado viável é preciso respeitar as restrições de disponibilidade dos professores e atender suas preferências. A medida que o número de turmas aumenta e mais variáveis são levadas em consideração, o processo se transforma em um problema difícil de ser resolvido manualmente devido ao seu caráter combinatório. Essa dificuldade motiva a criação de ferramentas computacionais que permitam gerar automaticamente horários de maior qualidade e cada vez mais rápido.

Este problema é conhecido na literatura de forma geral como geração de quadros de horário (*timetabling*, na língua inglesa). Os primeiros trabalhos científicos remontam a década de 60 e, desde então, têm demandado uma crescente atenção (GOTLIEB, 1963). Variações de problemas de *timetabling* têm sido descritas e muitas metodologias diferentes para solução têm sido propostas. Como estes problemas apresentam muitas variações, de acordo com o tipo de ambiente, é comum encontrar na literatura soluções que tratam apenas um conjunto reduzido de características da realidade. Do ponto de vista científico esta abordagem é justificada devido à necessidade de se estabelecer um padrão para se fazer comparações válidas de desempenho entre diferentes técnicas. Porém, do ponto de vista de aplicação, essa generalização faz com que a maioria das soluções existentes tenha pouca utilidade em situações práticas.

Os três principais desafios para se construir um aplicativo de geração de horários são:

1. Implementar novas restrições em adição as pré-existentes na literatura.

2. Obter dados de uma forma mais ágil e com uma representação extensível.

3. Fornecer uma interface com usuário que seja próxima da sua realidade para permitir um *feedback* construtivo.

Com o objetivo de apresentar uma solução para estes desafios iniciou-se um projeto pioneiro em 2006. Uma parceria entre pesquisadores do Colégio Politécnico da UFSM e pesquisadores da UFRJ deu início à construção de um protótipo chamado H.O.P.E. (Horários Otimizados Para Escolas) que foi aplicado em várias escolas e apresentado em Brasília (DORNELES; ARAúJO; SANTOS, 2006). Nesta ocasião, o carácter inovador do projeto chamou atenção de profissionais da educação e foi considerado uma ferramenta bastante necessária às instituições de ensino. O evento motivou a continuidade do projeto com a inclusão de novas funcionalidades e testes até meados de 2009, quando o protótipo foi descontinuado. Ainda em 2009 foram reunidas todas as informações coletadas do protótipo H.O.P.E. para criação de um novo aplicativo. Deu-se início ao desenvolvimento do HOP (Horários Otimizados por Programa), doravante denominado HOP-2009.

Este trabalho dá continuidade ao desenvolvimento do aplicativo HOP-2009 com as seguintes contribuições:

1. Melhorias na usabilidade da interface gráfica.
2. Definição de um formato padrão em XML para representação de instâncias do problema.
3. Criação de um novo *solver* mais rápido, extensível e com mais características.
4. Conclusão da primeira versão completa do aplicativo.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma aplicação capaz de resolver de maneira eficiente o problema do horário escolar de instituições de ensino básico, tanto propedêutico como profissionalizante.

1.1.2 Objetivos Específicos

- Delimitar um conjunto com as principais restrições consideradas na criação de um horário escolar típico para compor uma instância do problema.

- Implementação de uma interface gráfica de usuário que possibilite cadastrar diversas instâncias em formato de documento.
- Coletar instâncias reais de escolas de Santa Maria.
- Definir um formato de representação da instância que seja escalável em relação à inclusão de novas restrições.
- Desenvolver um algoritmo para gerar automaticamente o horário.
- Avaliar o desempenho da aplicação.

1.2 Organização do Trabalho

No capítulo 2 é apresentada a definição do problema do horário escolar e os principais conceitos necessários aos demais capítulos.

No capítulo 3 é apresentada uma breve revisão das principais técnicas de otimização usadas para geração de horários.

No capítulo 4 é apresentada uma visão geral do HOP-2009. São descritas as suas principais funções e limitações apontadas de acordo com avaliações feitas pelos usuários.

No capítulo 5 é apresentada a modelagem da aplicação HOP-2010 englobando soluções para as limitações do HOP-2009 através de padrões de design de interface. Nesta capítulo também é apresentado um novo modelo em XML para representação de instâncias de problemas de horário escolar.

No capítulo 6 é apresentada a implementação de novo resolvedor de horários utilizando a meta-heurística de busca tabu.

No capítulo 7 são apresentados experimentos e resultados do novo resolvedor em comparação com o resolvedor do HOP-2009 e em relação a um resolvedor comercial.

Finalmente, no capítulo 8 é apresentada a conclusão do trabalho e algumas opções para trabalhos futuros.

2 DEFINIÇÃO DO PROBLEMA

De forma resumida, o problema de geração de horários consiste em associar professores, turmas e disciplinas em determinados horários do dia considerando um dado número de restrições. Segundo (SCHAERF, 1999), este problema pode ser dividido em três categorias principais:

Horário de Escolas busca associar os professores às turmas, de maneira que nenhum professor ou turma tenha mais de uma aula ao mesmo tempo.

Horário de Universidades busca escolher os horários para as disciplinas tentando maximizar as matrículas dos alunos.

Horário de Exames visa organizar os exames de cada disciplina de modo que um aluno não tenha que realizar mais de um exame ao mesmo tempo. Além disso, os exames de cada aluno precisam ser tão espaçados na semana quanto possível.

Neste capítulo, o problema do horário escolar é classificado pelo autor em três níveis crescentes de complexidade. No primeiro nível, é apresentado um problema simplificado onde a complexidade é mínima. No segundo nível, o problema é definido com as características comuns da literatura acadêmica. E por último, é apresentado um problema prático que inclui características adicionais, específicas deste trabalho.

Em cada um destes níveis são apresentadas definições em relação as suas principais restrições.

2.1 Conceitos Básicos

Nesta seção apresenta-se os principais conceitos e entidades utilizadas em todo o trabalho.

2.1.1 Restrição

É uma determinada condição que o horário está sujeito a cumprir. Pode ser definida quanto a importância:

Restrição Forte é uma condição que deve ser, obrigatoriamente, satisfeita no horário. Ex.: Um professor **não pode** lecionar na segunda-feira.

Restrição Fraca é uma condição desejável mas não obrigatória para a composição do horário. Ex.: Um professor **não gostaria** de lecionar na sexta-feira.

Tanto as restrições fortes quanto as fracas ainda podem ser classificadas quanto a sua finalidade:

Restrição Pedagógica: influencia na qualidade do ensino.

Restrição Organizacional: referente à instituição de ensino.

Restrição Pessoal: relacionada com a preferência de cada professor.

2.1.2 Instância

É o conjunto específico de entidades e restrições que definem um dado problema de criação de horário. Geralmente as entidades são professores, turmas, disciplinas, salas, quadras, etc.

2.1.3 Contrato

Um contrato é uma especificação das aulas que precisam ser lecionadas e seus componentes. Um contrato simples determina o número de aulas semanais, de uma disciplina, que um professor precisa lecionar em uma turma. Cada professor pode estar presente em mais de um contrato. Na Tabela 2.1 são apresentados alguns exemplos de contratos.

Contrato	Professor(a)	Disciplina	Turma	Número de Aulas na Semana
c_1	Maria	Matemática	A	5
c_2	Rosa	Física	F	2
c_3	Rosa	Física	H	2
c_4	José	Geografia	H	2

Tabela 2.1: Exemplos de contratos

Contratos ainda podem ser compostos de recursos e restrições.

2.1.4 Recursos

Recursos são entidades com uma quantidade limitada e podem ser compartilhadas entre as aulas. Recursos comuns são salas, laboratórios, projetores, quadras de esportes, etc.

2.1.5 Alocação

A alocação é a representação "física"de uma aula do contrato. Ela determina exatamente em que dia, turno e período uma aula acontece. O principal objetivo na criação de um horário é determinar as alocações de cada contrato.

2.1.6 Solução

É o conjunto de todas alocações necessárias em um horário. Uma solução é dita **factível** quando atende todas as restrições fortes e **infactível** quando não atende. Quando uma solução atende todas as restrições impostas para o horário ela é denominada **solução ótima**.

2.1.7 Janela

São períodos livres entre aulas. Na Figura 2.1, a professora Luciana possui duas janelas. Já a professora Magda apenas uma. Os demais professores não apresentam janelas. Períodos livres no ínicio e no fim de um turno não são caracterizados como janelas.

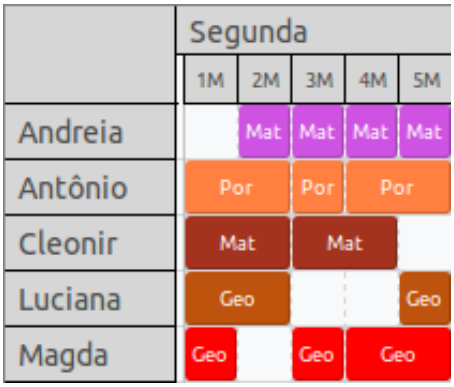

Figura 2.1: Exemplo de Janelas

De maneira análoga, este conceito também é utilizado em relação a dias e turnos, bem como pode ser aplicado em qualquer entidade envolvida no horário.

2.2 Problema Simplificado

O problema de horário escolar apresentado nesta seção não pode ser considerado real, mas a sua definição simplificada é útil para a introdução de alguns conceitos. Os elementos que compõe o problema simplificado são definidos pelos seguintes conjuntos:

Conjunto de professores: P_p onde $p = 1, ..., np$

Conjunto de turmas: T_t onde $t = 1, ..., nt$

Conjunto de períodos: K_k onde $k = 1, ..., nk$. Para facilitar a notação, os períodos são considerados sequencialmente durante os dias na semana.

Matriz de Aulas: $A_{p,t} = c$ onde c é o número de aulas que um professor P_p precisa lecionar na turma T_t.

Matriz Solução: $S_{p,t,k} \in 0, 1$ onde $S_{p,t,k} = 1$ indica a existência de uma aula e $S_{p,t,k} = 0$ onde não existem aulas.

Para encontrar um horário válido basta encontrar valores para a matriz $S_{p,t,k}$ respeitando as seguintes restrições:

Restrição 1 *Um professor não pode ser alocado em mais de uma aula em um dado período. Conforme Equação 2.1*

$$\sum_{p=1}^{np} S_{p,t,k} \leq 1, \forall t \in T_t, \forall k \in K_k \tag{2.1}$$

Restrição 2 *Uma turma não pode ter aula com mais de um professor um dado período. Conforme Equação 2.2*

$$\sum_{t=1}^{nt} S_{p,t,k} \leq 1, \forall p \in P_p, \forall k \in K_k \tag{2.2}$$

Restrição 3 *O professor precisa ser alocado em todas as suas aulas. Conforme Equação 2.3*

$$\sum_{p=1}^{np} S_{p,t,k} = A_{p,t}, \forall t \in T_t, \forall k \in K_k \tag{2.3}$$

Com estas restrições, foi provado que o problema é polinomial (SCHAERF, 1999, apud HOPCROFT e KARP, 1973). Em virtude de todas as restrições serem fortes, qualquer solução factível encontrada é considerada ótima. Na Tabela 2.2, as restrições do problema clássico são classificadas conforme a importância.

Restrição	Tipo
1	Forte
2	Forte
3	Forte

Tabela 2.2: Resumo de Restrições do Problema Clássico

2.3 Problema Acadêmico

Em um problema de horário escolar típico, os professores e turmas estão sujeitos a períodos indisponíveis para as aulas. Um período indisponível pode acontecer por inúmeros motivos: seja por que um professor está ministrando aula em outra escola ou devido a turma ter uma reunião pré-definida em um determinado dia. São definidos a seguir mais dois conjuntos:

Disponibilidade do Professor: $DP_{p,k} = d \ \text{com} \ d = \{0, 1\}$

Disponibilidade da Turma: $DT_{t,k} = d \ \text{com} \ d = \{0, 1\}$

Um período é dito disponível quando $d = 1$ e, caso contrário, indisponível quando $d = 0$. Com a nova notação, para encontrar uma solução $S_{p,t,k}$ é necessário satisfazer as seguintes restrições adicionais:

Restrição 4 (Disponibilidade do Professor) *Um professor não pode ser alocado em um período em que ele não está disponível. Conforme Equação 2.4*

$$\sum_{p=1}^{np} S_{p,t,k} \leq DP_{p,k}, \forall T_t, K_k \tag{2.4}$$

Restrição 5 (Disponibilidade da Turma) *Uma turma não pode ter aula em um período em que ela não está disponível. Conforme Equação 2.5*

$$\sum_{t=1}^{nt} S_{p,t,k} \leq DT_{p,k}, \forall P_p, K_k \tag{2.5}$$

Ao incluir as Restrições 4 e 5, o problema é NP-Completo (SCHAERF, 1999, apud EVEN et al, 1976). Além das restrições disponibilidade, uma série de outras restrições transformam o problema de busca em um problema de otimização interessante para a academia. A seguir são apresentadas as restrições mais comumente consideradas na literatura científica.

Restrição 6 *Um professor não deve ser alocado em um período sinalizado como indesejado.*

Restrição 7 *A turma não deve ser alocada em um período sinalizado como indesejado.*

Restrição 8 *Deve-se evitar a existência de janelas entre períodos no horário.*

Restrição 9 *Deve-se evitar a existência de janelas entre turnos no horário.*

Restrição 10 *Deve-se evitar a existência de janelas entre dias no horário.*

Restrição 11 *Um professor deve ter um número mínimo de dias livres no horário.*

Restrição 12 *O recurso deve ser compartilhado e alocado de maneira que não ultrapasse a sua quantidade disponível em um determinado período.*

Restrição 13 *Uma recurso não pode ser alocado em um período para o qual não está disponível.*

Restrição 14 *Um recurso não deve ser alocado para em um período sinalizado como indesejado.*

Restrição 15 *As aulas de uma disciplina precisam ser alocadas consecutivamente.*

Restrição 16 *Algumas aulas envolvem várias turmas ao mesmo tempo. Esta restrição é uma exceção a Restrição 2.*

Restrição 17 *Algumas aulas envolvem vários professores ao mesmo tempo. Esta restrição é uma exceção a Restrição 1.*

2.4 Problema Prático

Para compor o problema prático foram realizadas entrevistas diretamente com educadores responsáveis pela criação de horários nas seguintes instituições de Santa Maria:

- Colégio Riachuelo;
- Colégio Estadual Manoel Ribas;
- Colégio Técnico Industrial de Santa Maria;
- Escola Estadual Olavo Bilac;
- Escola Municipal de 1º Grau Fontoura Ilha.

Ao analisar todas as restrições consideradas necessárias por cada educador foram reunidas as seguintes restrições:

Restrição 18 *O professor deve ter um limite de turnos de trabalho diário.*

Restrição 19 *O professor deve ter o horário programado de forma que tenha tempo de se deslocar para o almoço ou janta.*

Restrição 20 *Algumas aulas devem acontecer em períodos pré-definidos.*

Restrição 21 *Deve-se evitar a existência de janelas entre disciplinas no horário.*

Restrição 22 *Deve ser possível **sugerir** uma distribuição para as aulas de um contrato.*

Restrição 23 *Deve ser possível **obrigar** uma distribuição para as aulas de um contrato.*

Restrição 24 *Deve ser possível estabelecer um limite diário para as aulas de um contrato.*

Restrição 25 *Aulas geminadas não podem ser separadas pelo sinal de intervalo.*

Restrição 26 *Algumas disciplinas não devem ser alocadas no mesmo dia para uma turma.*

Restrição 27 *Aulas de algumas turmas começam em horários distintos. Ex.: Turma A começa suas aulas às 7:30 e a Turma B começa 7:45.*

3 REVISÃO BIBLIOGRÁFICA

De maneira geral, o problema de geração de horários, assim como outros problemas de otimização combinatória, tem como objetivo minimizar uma função f dentro de um **espaço de soluções** finito S. O espaço de soluções, também chamado de **espaço de busca**, é formado por um conjunto elevado de soluções, sujeitas as restrições do problema. Devido ao número de elementos, geralmente é impossível enumerar S em tempo hábil. Nesta seção são apresentadas as principais técnicas para resolução de horários. Uma ênfase maior é colocada na metaheurística de busca tabu pois os seus conceitos são utilizados na implementação do Capítulo 6.

3.1 Métodos Exatos

A técnica exata utiliza como base um modelo matemático para encontrar a solução do problema considerado respeitando as suas restrições. Estas restrições são descritas de forma semelhante às equações apresentadas na Seção 2.2. Esta técnica sempre encontra a solução ótima (caso ela exista), porém, dependendo da quantidade e da natureza das restrições, a busca pode demandar muito tempo. Esta peculiaridade geralmente inviabiliza a utilização da técnica para a resolução de instâncias de grande porte. Os principais métodos usados para programação de horários são: Programação Linear Inteira Mista (AL-YAKOOB; SHERALI, 2007), Geração de Colunas (BOLAND et al., 2008), Planos de Cortes (MIRHASSANI, 2006) e Relaxação Lagrangeana (RIBEIRO FILHO, 2006).

3.2 Heurísticas e Metaheurísticas

Heurísticas e Metaheurísticas são uma alternativa para resolver problemas de otimização em que os métodos exatos não são aplicáveis. O termo Metaheurísticas, segundo (COLORNI; DORIGO; MANIEZZO, 1998), define uma classe de algoritmos geralmente

inspirados em processos da natureza que atualizam uma ou várias soluções ao longo do tempo. Devido a essa atualização também são denominados *algoritmos evolutivos*. Os principais algoritmos desta classe, usados para geração de horário, são *Simulated annealing* (KIRKPATRICK, 1984), Algoritmos Genéticos] (HOLLAND, 1975) e Busca Tabu (GLOVER, 1986).

3.2.1 Heurísticas Construtivas

Heurísticas construtivas são algoritmos que executam uma série operações com o objetivo de montar uma solução a partir de uma instância do problema. Na maioria das vezes o problema possui uma função g que permite estimar a qualidade de uma operação. Nesta situação é possível montar uma solução s de acordo com um critério guloso, ou seja, cada passo da montagem seleciona a ação p que possui o maior valor em g (GRUBER, 2009). Este processo é apresentado no algoritmo da Figura 3.1

Procedimento *ConstrutivoGuloso* $(g())$
1: $s \leftarrow 0$
2: inicializar um conjunto P com as operações possíveis
3: **enquanto** $|P| \neq 0$ **faça**
4: $\quad p \leftarrow \arg\max_{x \in P} g(x)$
5: $\quad s \leftarrow s + p$
6: $\quad$ retirar p de P
7: **fim enquanto**
8: **retorna** s

Figura 3.1: Algoritmo Construtivo Guloso

Uma outra maneira típica de se construir uma solução consiste em selecionar aleatoriamente uma operação p até que a solução esteja completa. Este método é apresentado no algoritmo da Figura 3.2 e, embora seja simples de implementar, tem como principal desvantagem a construções de soluções de baixa qualidade.

Procedimento *ConstrutivoAleatorio* ()
1: $s \leftarrow 0$
2: inicializar um conjunto P com as operações possíveis
3: **enquanto** $|P| \neq 0$ **faça**
4: selecionar aleatoriamente uma operação $p \in P$
5: $s \leftarrow s + p$
6: retirar p de P
7: **fim enquanto**
8: **retorna** s

Figura 3.2: Algoritmo Construtivo Aleatório

3.2.2 Busca Local

Conforme (GRUBER, 2009), para compreender este método é necessário conhecer os seguintes conceitos básicos:

Movimento m**:** é uma operação que modifica uma solução s e dá origem a uma nova solução s'. Essa modificação é representada pela notação $s + m = s'$. Onde s' é denominada **solução vizinha** de s.

Vizinhança $V(s)$ é um conjunto com todas as soluções vizinhas à s. Cada elemento s'_i de V, é obtido a partir de um movimento m_i sobre a solução s, conforme representado no diagrama da Figura 3.3.

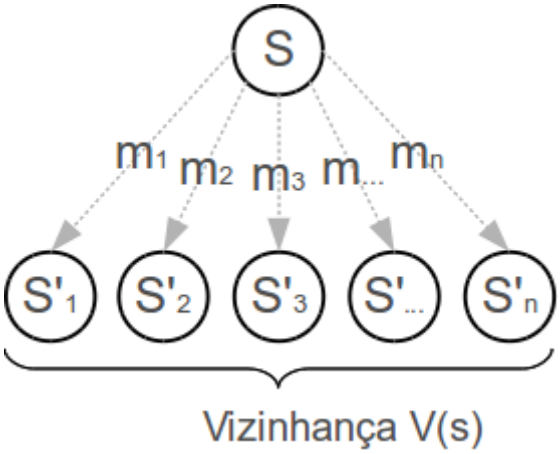

Figura 3.3: Representação de soluções de uma vizinhança

De maneira geral a técnica de busca local consiste em atualizar uma solução s, em repetidas explorações de um vizinhança $V(s)$. Supondo que o objetivo da busca seja

minimizar uma função f e partindo de uma solução inicial s_0 uma busca local genérica é definida pelo algoritmo da Figura 3.4.

Procedimento *BuscaLocal* $(f(), V(), s_0)$
1: $s \leftarrow s_0$
2: **repita**
3: escolher uma solução $s' \in V(s)$
4: **se** $f(s') < f(s)$ **então**
5: $s \leftarrow s'$
6: **fim se**
7: **até que** alguma condição de parada seja satisfeita
8: **retorna** s

Figura 3.4: Algoritmo Genérico de Busca Local

O algoritmo da Figura 3.4 é especializado de acordo com o método de seleção na linha 5. As três especializações mais comuns são:

Seleção do Melhor: Explora toda a vizinhança e seleciona o melhor. $s' \leftarrow \arg\min_{x \in V'} f(x)$

Seleção do Primeiro Melhor: Seleciona a primeira solução $s' \in V(s)$ tal que $f(s') < f(s)$. Esse procedimento é mais barato computacionalmente que a seleção do melhor.

Seleção Aleatória: Seleciona um elemento aleatório da vizinhança. É o método de seleção mais barato computacionalmente, mas com menor chance de satisfazer a condição $f(s') < f(s)$

Independente do tipo de seleção, a busca local termina quando um critério de parada como tempo ou número de iterações é satisfeito ou quando a busca atinge um ótimo local.

- s^* é denominada **ótimo local** quando $f(s^*) \leq f(x), \forall_x \in V(s^*)$.
- s^{**} é denominada **ótimo global** quando $f(s^{**}) \leq f(x), \forall_x \in S$.

Como demonstrado na Figura 3.5, dependendo da solução inicial da busca, diferentes ótimos locais podem ser alcançados no espaço de buscas S.

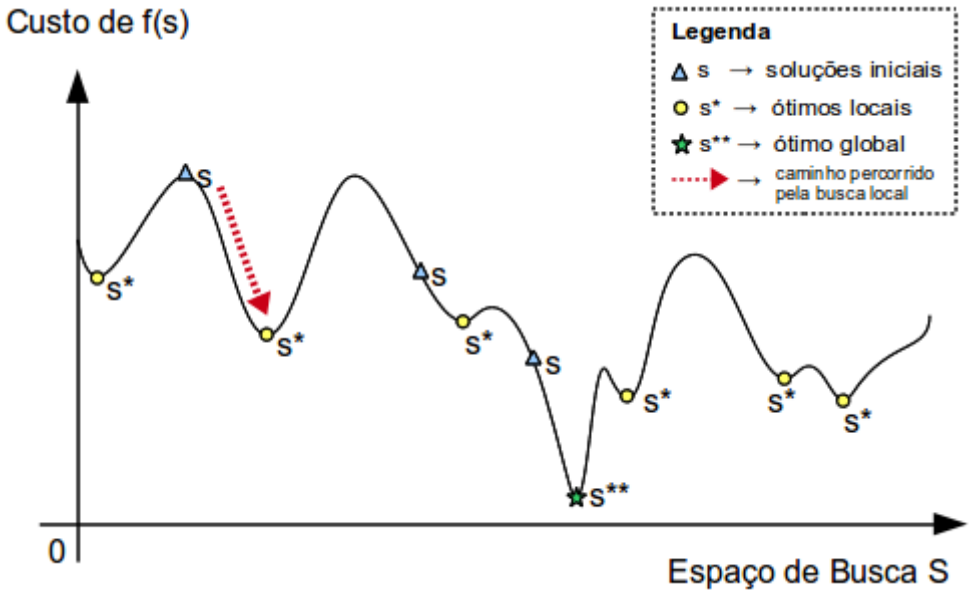

Figura 3.5: Percurso característico de uma busca local

3.2.3 Busca Tabu

A busca tabu surgiu pela primeira vez na literatura nos trabalhos de Fred Glover em 1986 (GLOVER, 1986). De maneira geral, esta técnica supervisiona uma busca local utilizando algumas estruturas de memória de curto e longo prazo para escapar de ótimos locais e explorar o espaço de busca de maneira estratégica (BLUM; ROLI, 2003).

O algoritmo básico parte de uma solução inicial s_0 e utiliza uma busca local para mover-se de uma solução s para a melhor solução s' de uma vizinhança V. Diferentemente da busca local definida na Seção 3.2.2, a busca tabu permite que um movimento s_p de piora seja feito. Como consequência dessa possibilidade, a busca poderia entrar em um ciclo de alternância entre s^* e s. Para evitar este ciclo a técnica define uma estrutura de memória chamada **lista tabu** que é uma lista T onde são armazenados os movimentos proibidos. Desta forma, ao explorar a vizinhança V, as soluções s obtidas através de movimentos m que estão na lista tabu não podem ser escolhidas. A única exceção a esta regra é o **critério de aspiração**, o qual comumente permite que um movimento da lista tabu seja escolhido desde que ele dê origem a uma solução s' com $f(s') < f(s)$.

Na definição clássica, a lista tabu poderia armazenar soluções que não deveriam ser mais visitadas. A principal deficiência dessa abordagem é o uso excessivo de memória necessário para armazenar soluções completas. A abordagem mais comum armazena somente alguns movimentos ou características da solução por um determinado número de

iterações t, onde t é um parâmetro da lista.

Uma representação simplificada da busca tabu é apresentada no algoritmo da Figura 3.6.

Procedimento *BuscaTabu* $(f(), V(), T, s_0)$
1: $s \leftarrow s_0$
2: $L \leftarrow 0$
3: **repita**
4: $s' \leftarrow \arg\min_{x \in V} f(x)$ repeitando T e o critério de aspiração
5: adiciona s' em T
6: **se** $f(s') < f(s)$ **então**
7: $s \leftarrow s'$
8: **fim se**
9: remove os elementos de T que não são mais proibidos
10: **até que** alguma condição de parada seja satisfeita
11: **retorna** s

Figura 3.6: Algoritmo de Busca Tabu

A busca tabu costumam incluir estratégias de diversificação e intensificação. As estratégias de diversificação, cujo objetivo é tentar explorar áreas diferentes do espaço de busca, são geralmente implementadas através de uma **memória de longo prazo**, a qual armazena os atributos da solução que mais se repetem durante a busca.

Quando o procedimento de diversificação é ativado, procura-se gerar soluções com atributos diferentes dos armazenados na memória de longo prazo. Como nem sempre o algoritmo consegue detectar a saturação do seu espaço atual de buscas, é comum ativar a diversificação conforme um intervalo de iterações fixo ou quando um dado número de iterações sem melhora é atingido.

As estratégias de intensificação, ao contrário da diversificação, têm como objetivo focar as buscas em uma região que é potencialmente promissora. O procedimento de intensificação pode ser ativado sempre que uma solução melhor que a atual é encontrada ou quando algum atributo chave do problema é identificado.

Tanto a diversificação como a intensificação podem ser executadas por um número determinado de iterações.

3.3 Algoritmos Híbridos

Conforme (GRUBER, 2009), os métodos exatos e as metaheurísticas podem ser combinados para colaborar de maneira sinérgica. A colaboração pode acontecer de forma paralela, sequencial ou intercalada. Uma outra possibilidade é a colocação de uma técnica para supervisionar e controlar a outra.

4 VISÃO GERAL DO HOP-2009

Nesta seção é apresentada uma visão geral do HOP-2009. Esta versão do aplicativo foi utilizada de maneira experimental por educadores de escolas de Santa Maria para resolverem os horários de suas instituições. O uso foi supervisionado mediante observação em várias interações com o aplicativo. Ao final de cada interação as opiniões do usuário foram coletadas e avaliadas para compor as limitações do HOP-2009 neste capítulo.

4.1 Arquitetura Geral

O HOP-2009 é uma aplicação para geração de quadros de horários projetada com base em dois componentes básicos. O primeiro componente é uma interface gráfica que contém a lógica de armazenamento, manipulação e apresentação dos dados da aplicação. Essa interface se comunica através de um arquivo com um segundo programa, denominado **resolvedor**, que implementa a inteligência computacional responsável por efetivamente resolver o horário.

A Figura 4.1 representa a arquitetura geral do HOP-2009.

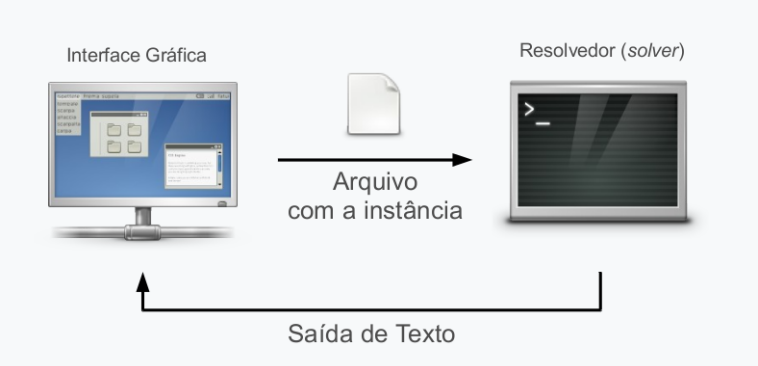

Figura 4.1: Arquitetura do HOP-2009

O arquivo enviado da interface gráfica para o resolvedor contém a representação da

instância do problema que foi cadastrado pelo usuário. Quando o resolvedor recebe o arquivo, este executa um processamento e depois imprime uma saída de texto com a solução. Esta solução é interpretada pela interface e apresentada para o usuário.

4.2 Inteface Gráfica

A interface gráfica foi desenvolvida utilizando a linguagem Java em conjunto com a biblioteca gráfica SWT (IBM, 2010).

Ao iniciar o programa, o usuário se depara com a tela da Figura 4.2. A partir desta tela é possível criar um novo projeto de horário ou abrir um salvo previamente. Conforme observação realizada esta tela não foi considerada intuitiva pelos usuários pois estes não sabiam como proceder para iniciar o cadastro do horário.

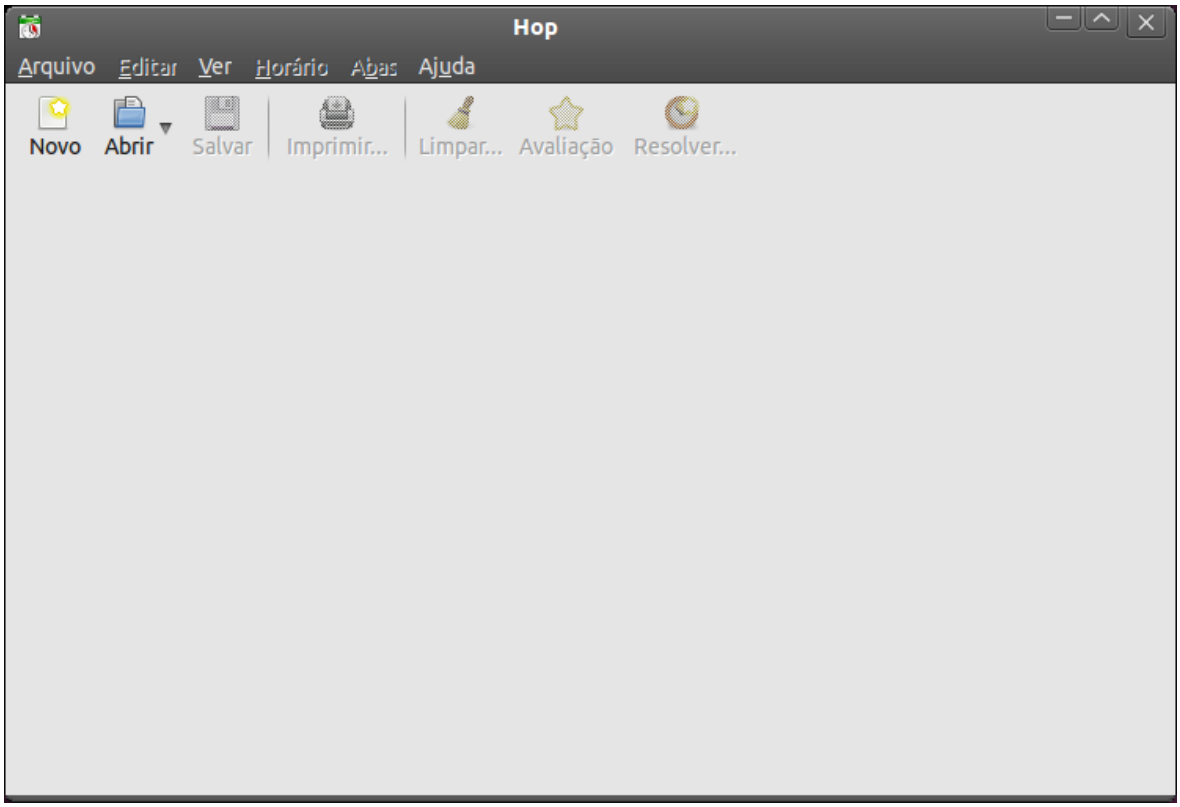

Figura 4.2: Tela inicial do HOP-2009

Na Figura 4.3 tem-se a tela principal do aplicativo com um projeto aberto. Esta interface é composta por seis abas. Cada uma delas dá acesso ao cadastro de uma entidade do projeto.

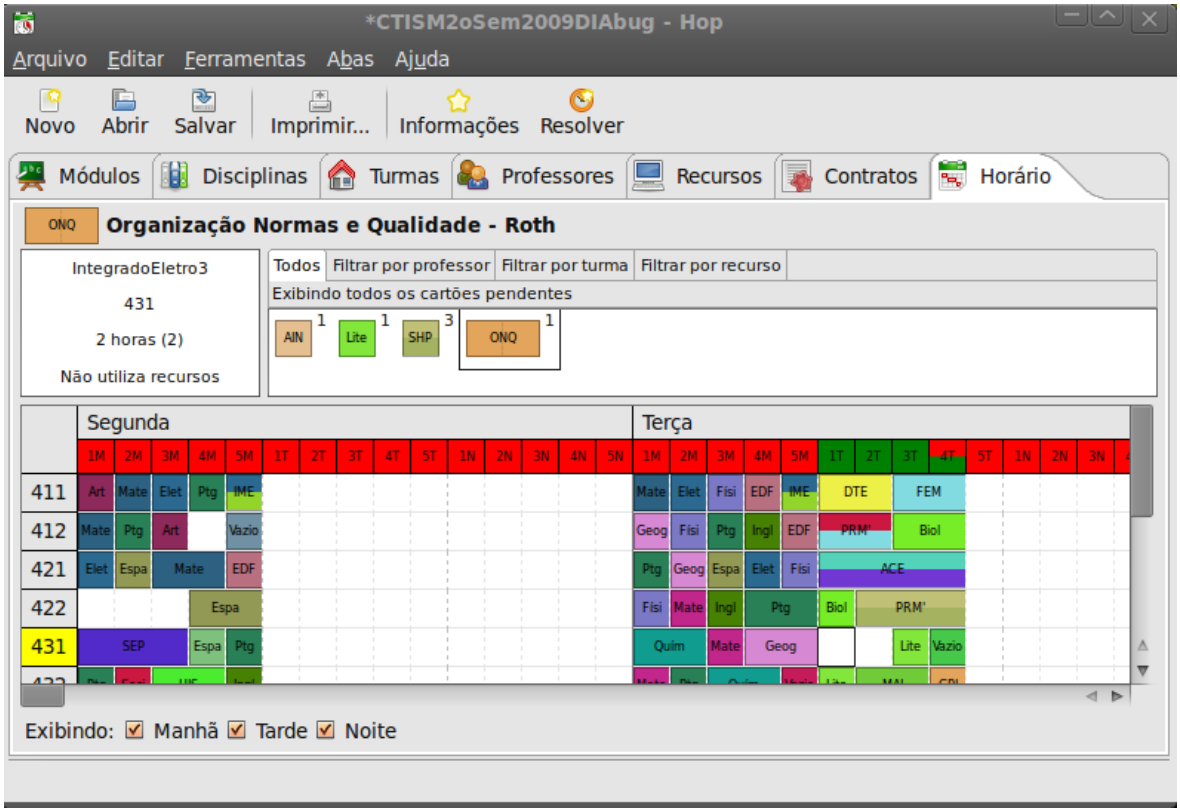

Figura 4.3: Tela principal do HOP-2009

Na Figura 4.4 é possível ver o relacionamento entre as principais entidades de dados. A entidade "Módulo"define a categoria do contrato. Categorias comuns de módulos são séries ou cursos.

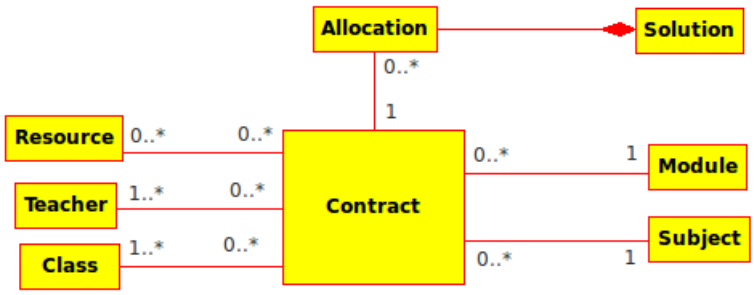

Figura 4.4: Diagrama de classes de dados do HOP-2009

Os principais casos de uso ao manipular um projeto são vistos na Figura 4.5.

Todos eles são bastante simples. Os casos do tipo *Manter* referem-se a operações básicas de cadastro como consultar, alterar, incluir e remover. O Caso *Manter Contratos*, em especial, pode necessitar incluir entidades ainda não cadastradas.

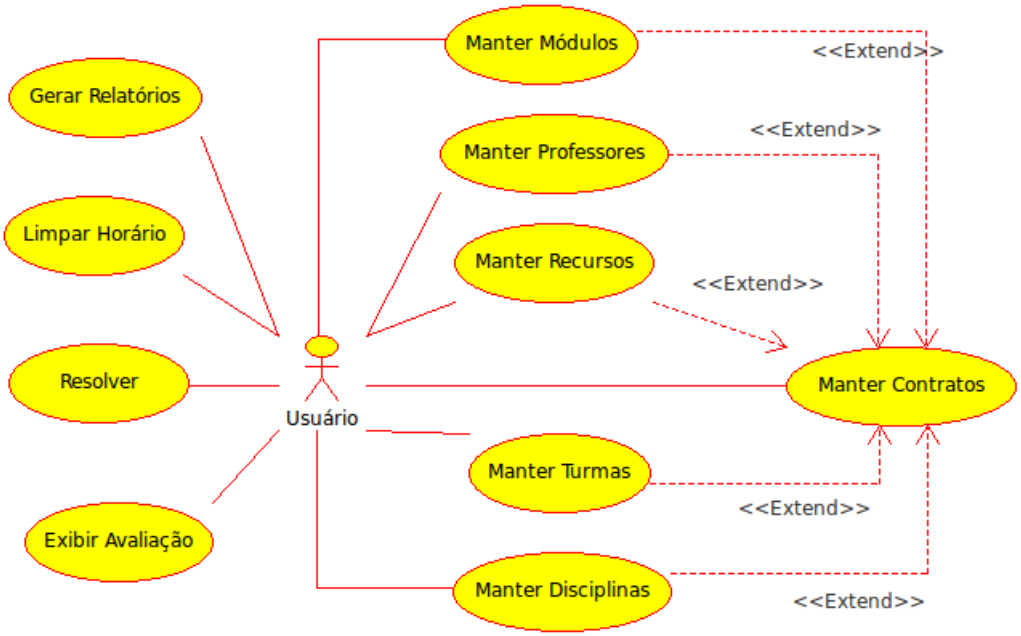

Figura 4.5: Casos de Uso do HOP-2009

O Caso *Resolver* requer que no mínimo um contrato tenha sido cadastrado. Quando o usuário aciona o resolvedor, o programa exibe a janela de progresso da Figura 4.6.

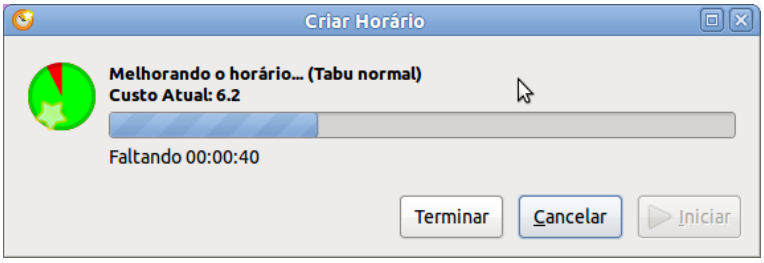

Figura 4.6: Tela de progresso da resolução do HOP-2009

O Caso de uso *Exibir Avaliação* apresenta a tela com as restrições da Figura 4.7.

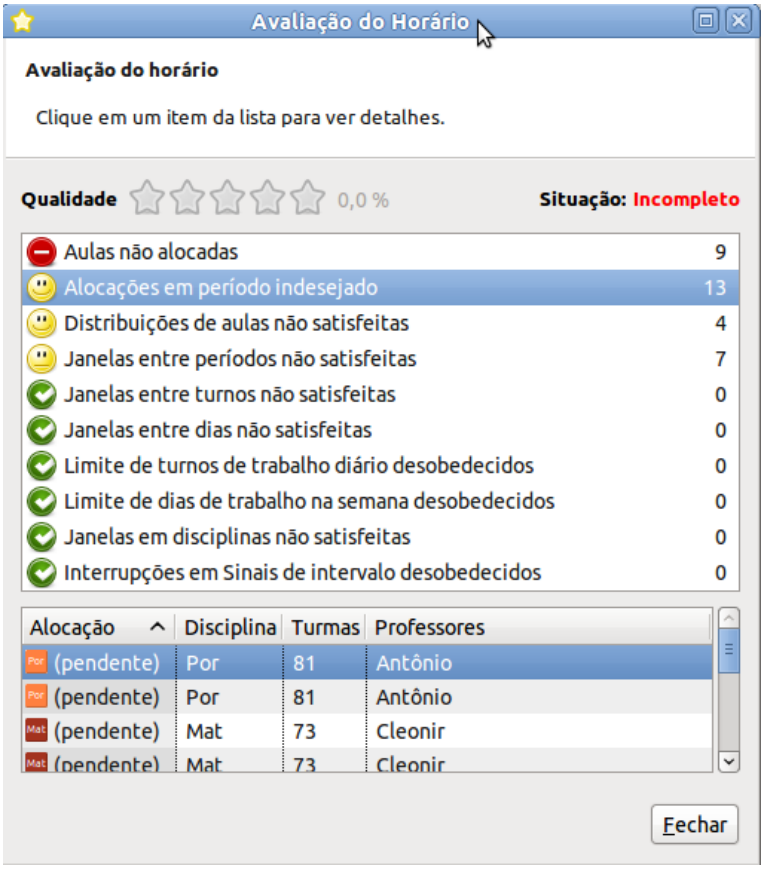

Figura 4.7: Tela de Avaliação de Horário do HOP-2009

O Caso de uso *Gerar Relatórios* apresenta a tela da Figura 4.8.

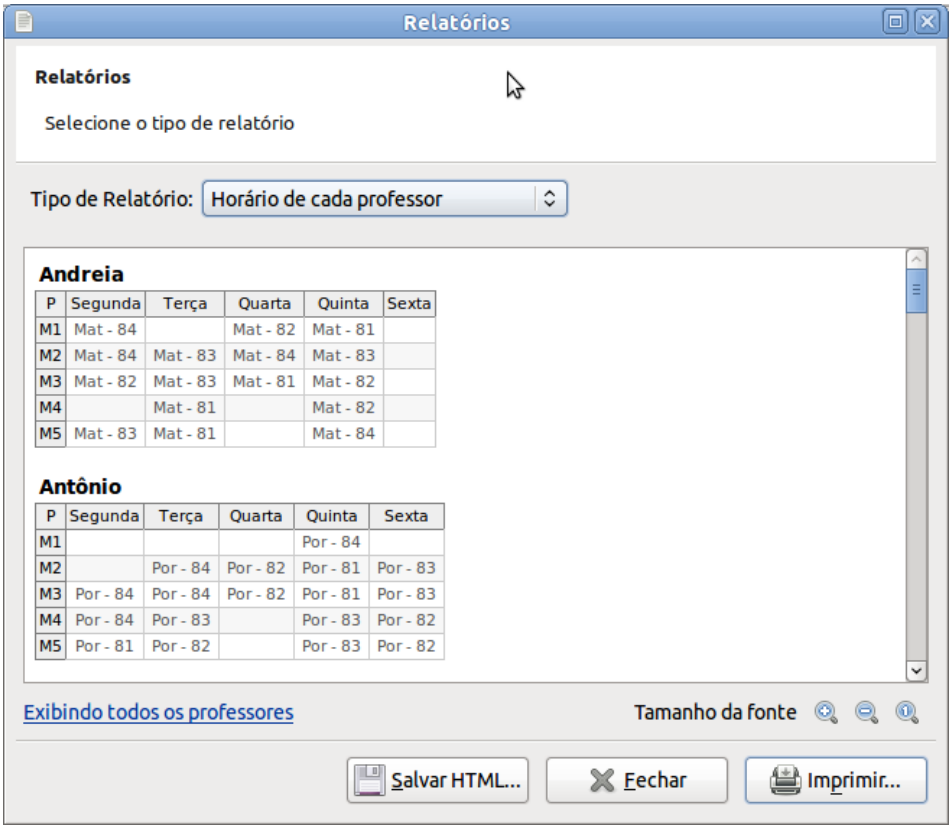

Figura 4.8: Tela de Exibição de Relatórios do HOP-2009

O Caso de uso *Manter Professores* inclui as operações de cadastro de professor apresentado na Figura 4.9, a edição da sua disponibilidade na tela da Figura 4.10 e a configuração de suas restrições na Figura 4.11.

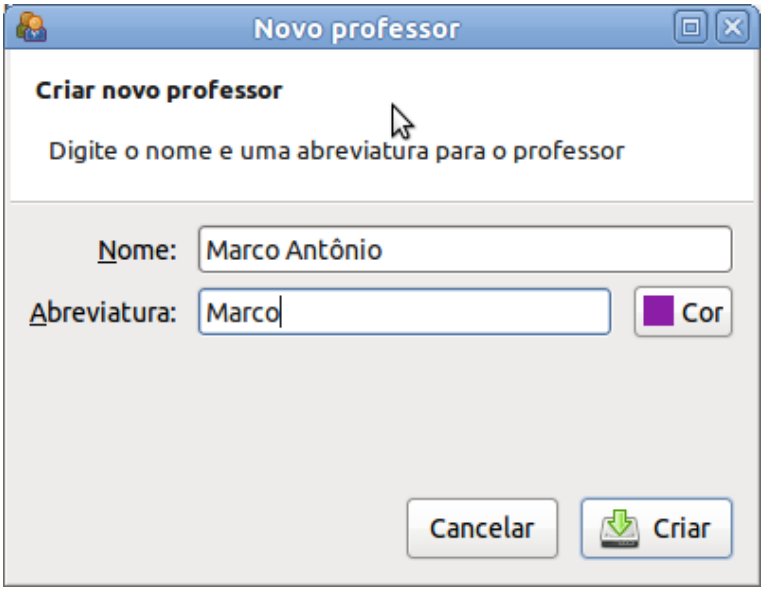

Figura 4.9: Tela de Cadastro de professor do HOP-2009

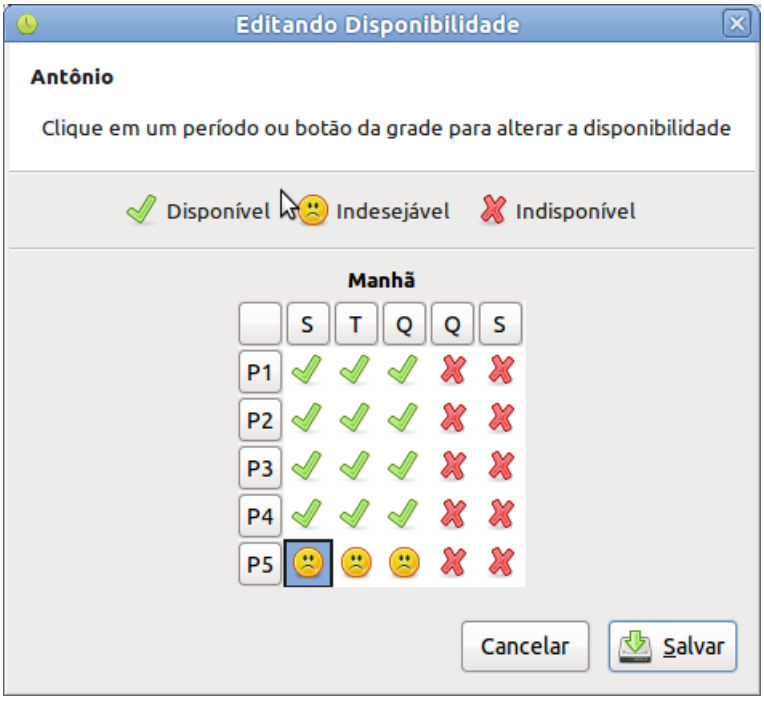

Figura 4.10: Tela de disponibilidade do professor do HOP-2009

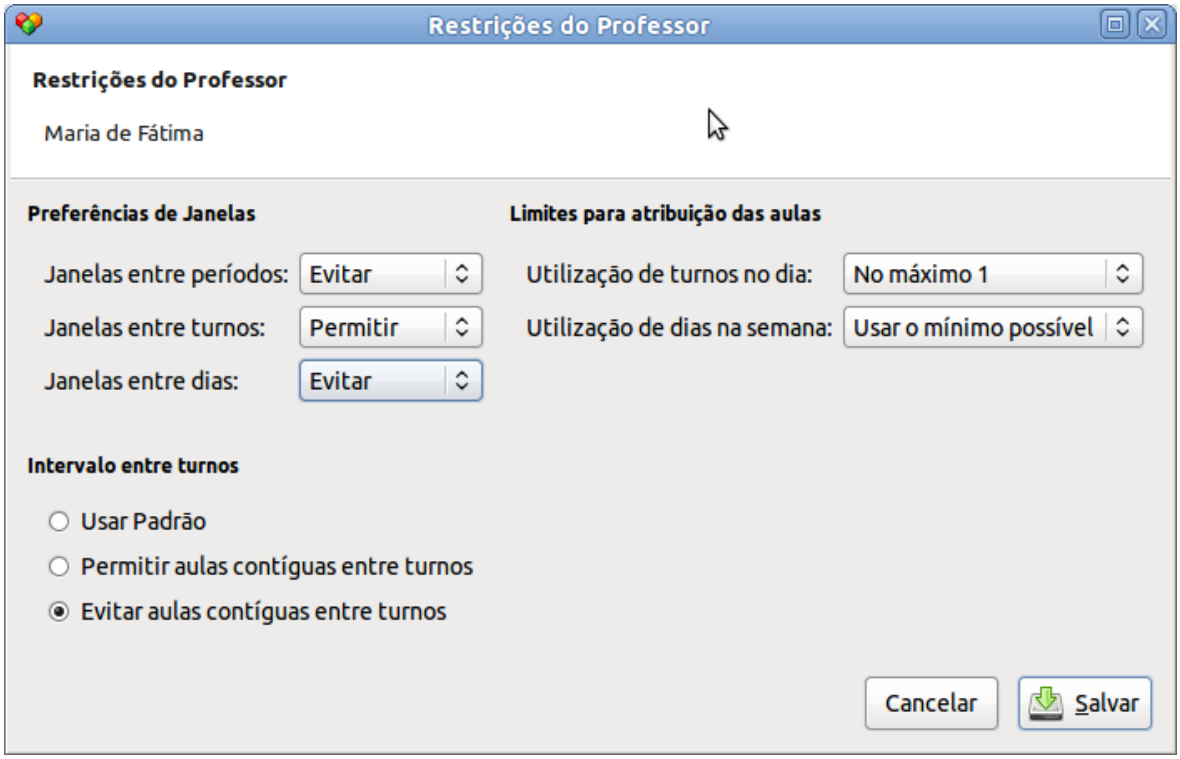

Figura 4.11: Tela de restrições do professor do HOP-2009

O caso de uso *Manter Contratos* reúne operações de inclusão, edição e exclusão. Destas operações a inclusão é mais a mais representativa e é realizada conforme os passos

abaixo:

1. Selecionar o módulo
2. Selecionar a disciplina
3. Selecionar professores
4. Selecionar turmas
5. Configurar distribuição das aulas
6. Selecionar recursos (Opcional)

Cada passo é indicado na Figura 4.12.

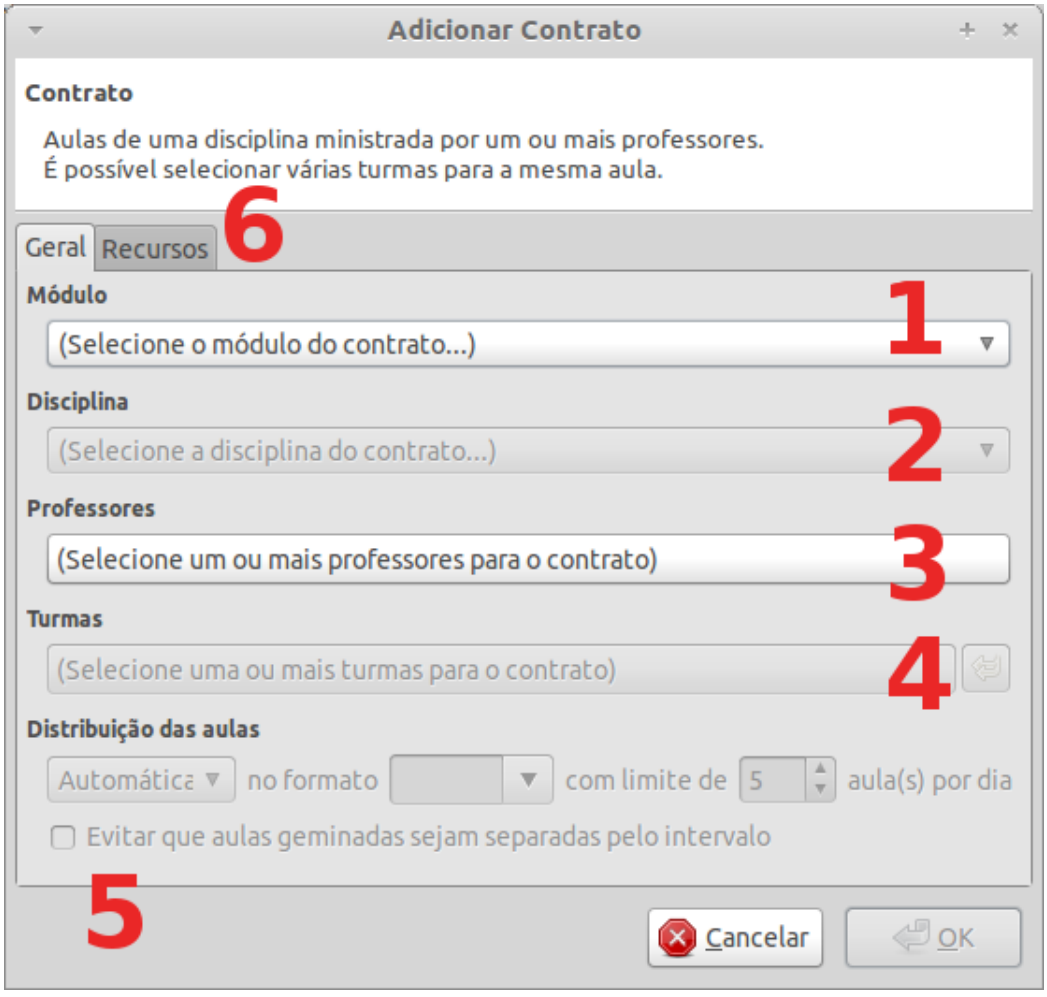

Figura 4.12: Passos para inclusão de um contrato no HOP-2009

Os passos de 1 a 4 ainda envolvem percorrer uma lista com um grande número de elementos. Conforme constatado via observação, um usuário mediano gasta aproximadamente 40 segundos para incluir cada contrato. Na Tabela 4.1 é apresentada uma estimativa

do tempo total utilizado no cadastro das instâncias do Apêndice E. Conforme avaliação dos usuários, o cadastro de contratos é considerado um dos principais problemas do HOP-2009, pois consome muito tempo para ser realizado.

Instância	Contratos	Tempo total em minutos
E.1	210	140
E.2	72	48
E.3	82	54

Tabela 4.1: Estimativa do tempo necessário para cadastrar contratos no HOP-2009

Embora a aplicação não defina uma ordem exata para as tarefas, é possível incluí-las em uma fase de cadastro e uma fase de resolução. A seguir um roteiro típico para utilização do aplicativo:

1. Cadastro dos dados
   (a) Cadastrar as entidades e suas restrições
   (b) Cadastrar contratos e suas restrições
2. Resolução do horário
   (a) Resolver
   (b) Exibir avaliação do horário
   (c) Gerar relatórios e imprimir

Ao observar a utilização do aplicativo foi constatado que os usuários apresentaram dificuldades para realizar esse roteiro sem solicitar ajuda.

4.3 Resolvedor

O resolvedor do HOP-2009 é o mesmo utilizado na última versão do protótipo H.O.P.E. Esta versão, denominada Har2008, tem suporte a todas as restrições definidas no Capítulo 2, com exceção as Restrições 19, 23, 25, 27.

Um limitação bastante relevante deste resolvedor é a ausência do seu código-fonte, que impede novas modificações. Outra limitação importante é referente ao formato de representação da instância através de arquivos texto. Cada restrição da instância é representada em um arquivo texto diferente, o que dificulta a extensão das informações e inclusão de novas restrições.

5 MODELAGEM DA APLICAÇÃO

Neste capítulo é apresentada a modelagem da nova aplicação desenvolvida, doravante denominada HOP-2010. Com com base nas limitações avaliadas pelos usuários do HOP-2009 no Capítulo 4, foram identificados pontos de modelagem no que se refere a interface gráfica, ao arquivo de representação da instância e por fim, em relação ao resolvedor.

Para realizar as modificações na interface gráfica foram levados em consideração padrões de design de interfaces para o aumento da usabilidade. Estes padrões são apresentados na seção 5.1 e utilizados para compor as soluções propostas para interface na seção 5.2.

Como o resolvedor do HOP-2009 não é passível de modificações devido a ausência do código-fonte, ficou evidente a necessidade da implementação de um novo resolvedor que tenha suporte a todas as restrições definidas no Capítulo 2. Esta nova implementação que é descrita no Capítulo 6 é projetada para trabalhar com o formato de representação da instância descrito na seção 5.3.

5.1 Padrões para Design de Interface

Nesta seção são apresentados os padrões de design de interface que serviram como base para as modificações da interface do HOP-2010.

5.1.1 *Clear Entry Points*

Este padrão é indicado para ser usado em telas iniciais ou quando o usuário acessa a aplicação pela primeira vez. Consiste em apresentar algumas poucas opções descritivas que conduzam o usuário a iniciar suas atividades (TIDWELL, 2005).

5.1.2 *Wizard*

O padrão *Wizard* é bem indicado em situações em que o usuário precisa realizar uma atividade longa e tomar várias decisões. A implementação deste padrão consiste em dividir a tarefa principal em pequenas tarefas e apresentá-las passo a passo em uma ordem pré-determinada (TIDWELL, 2005).

5.1.3 *Center Stage*

Este padrão determina que a parte mais importante da interface deve ser colocada no centro da janela. Todas as ferramentas secundárias devem ficar distribuídas ao redor da janela. Esta disposição permite que o usuário foque sua atenção nas nas informações mais importantes.(TIDWELL, 2005).

5.1.4 Áreas de Prioridade na Tela

Conforme (BRANDãO, 2006), existe um padrão para a visualização no qual o usuário dá mais atenção para determinadas áreas da tela. Na Figura 5.1 são apresentadas as áreas de prioridade, sendo a prioridade 1 a mais alta.

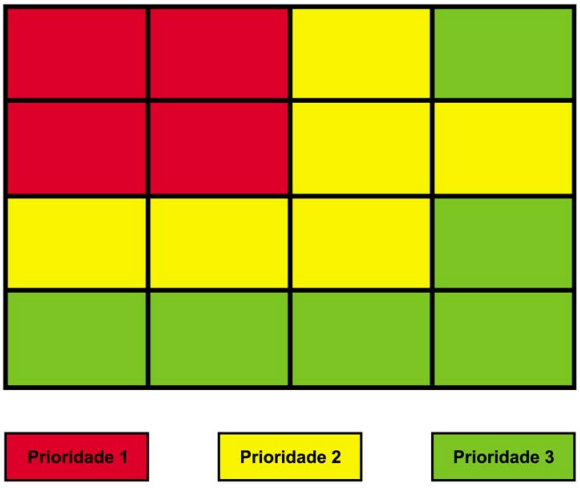

Figura 5.1: Áreas de Prioridade na Tela

5.2 Soluções propostas para a interface

Nesta seção são apresentadas as soluções desenvolvidas para a interface gráfica. Os detalhes técnicos da implementação são omitidos por não fazerem parte do escopo deste trabalho.

5.2.1 Tela inicial

Para melhorar a intuitividade da tela inicial foi utilizado o padrão *Clear Entry Points* com dois pontos de entrada: um para **criar um novo projeto** e outro para **abrir um projeto recente** a partir de uma lista. A nova versão da tela inicial é apresentada na Figura 5.2.

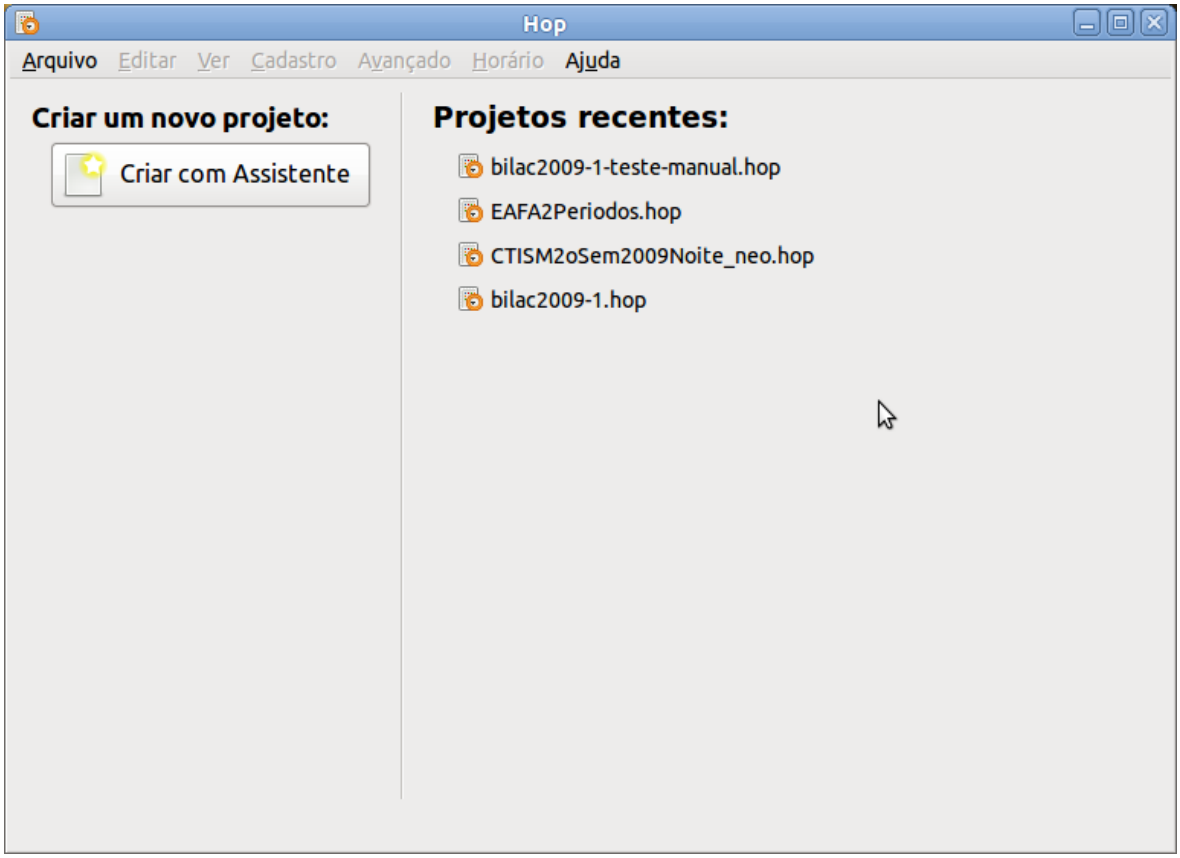

Figura 5.2: Tela Inicial do HOP-2010

5.2.2 Cadastro de Contratos

Para solucionar o problema da demora no cadastro de contratos foram feitas experiências com alguns modelos de cadastro. Como solução final aprovada pelos usuários, foi adotado um modelo de grade para a inclusão dos contratos como o da Figura 5.3.

A grade de contratos permite selecionar um professor na lista a esquerda e simplesmente clicar na posição correspondente a disciplina e turma desejada para criar um contrato. Caso seja necessário adicionar um recurso ou personalizar a distribuição do contrato, o usuário pode clicar em "Restrições".

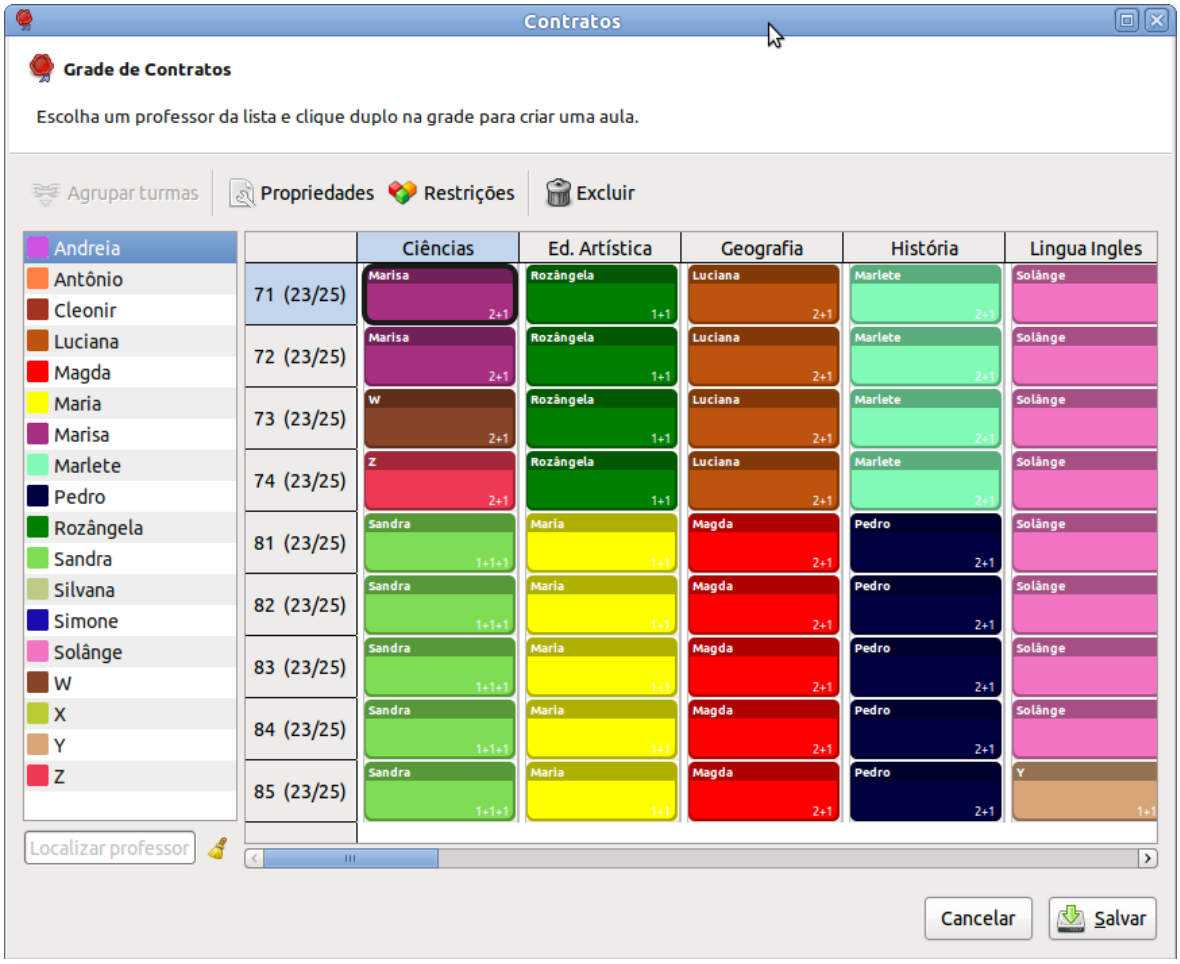

Figura 5.3: Tela de edição de contratos em grade do HOP-2010

Conforme novos testes realizados com este novo modelo de cadastro, um usuário mediano leva apenas 7 segundos para cadastrar um contrato. Na Tabela 5.1 é apresentada uma estimativa do tempo total utilizado no cadastro das instâncias do Apêndice E. Em relação as estimativas da Tabela 4.1 a velocidade de cadastro ficou aproximadamente 5,5 vezes mais rápida.

Instância	Contratos	Tempo total em minutos
E.1	210	25
E.2	72	8
E.3	82	10

Tabela 5.1: Estimativa do tempo necessário para cadastrar contratos no HOP-2010

5.2.3 Iteração Geral com a interface

Ao usar o HOP-2009 os usuários relataram dificuldade para memorizar o roteiro de uso da aplicação. Para solucionar essa dificuldade foram adotas duas estratégias: A pri-

meira consiste em um assistente utilizando o padrão *Wizard* para facilitar o cadastro dos dados. A segunda estratégia visa melhorar a apresentação da tela principal. O padrão *Center Stage* foi implementado através da remoção das abas de entidades do HOP-2009 e com centralização do horário na janela. Conforme as áreas de prioridade, as principais ações utilizadas pelo usuário foram dispostas no canto superior direito da barra de ferramentas.

A tela do assistente de criação pode ser vista na Figura 5.4 e a nova tela principal na Figura 5.5.

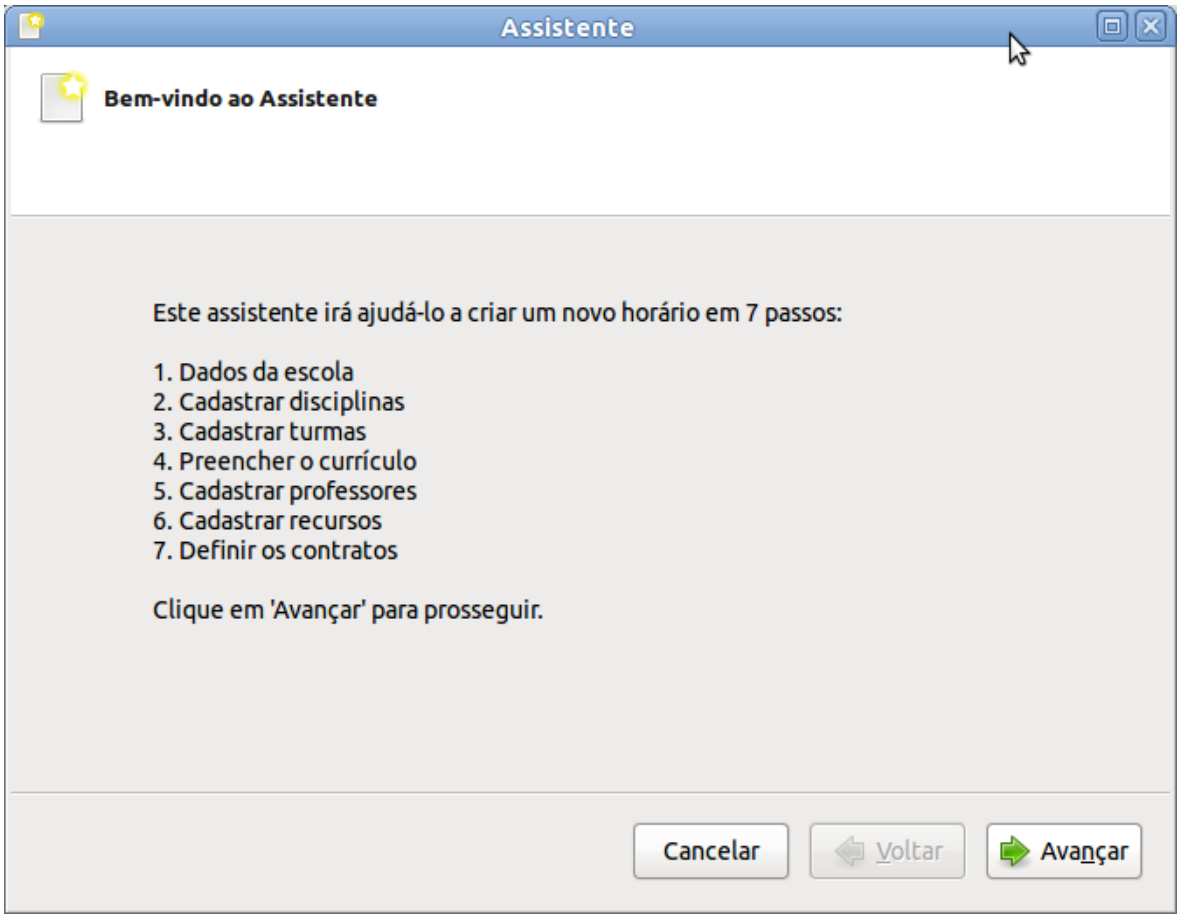

Figura 5.4: Tela do Assistente de Criação do HOP-2010

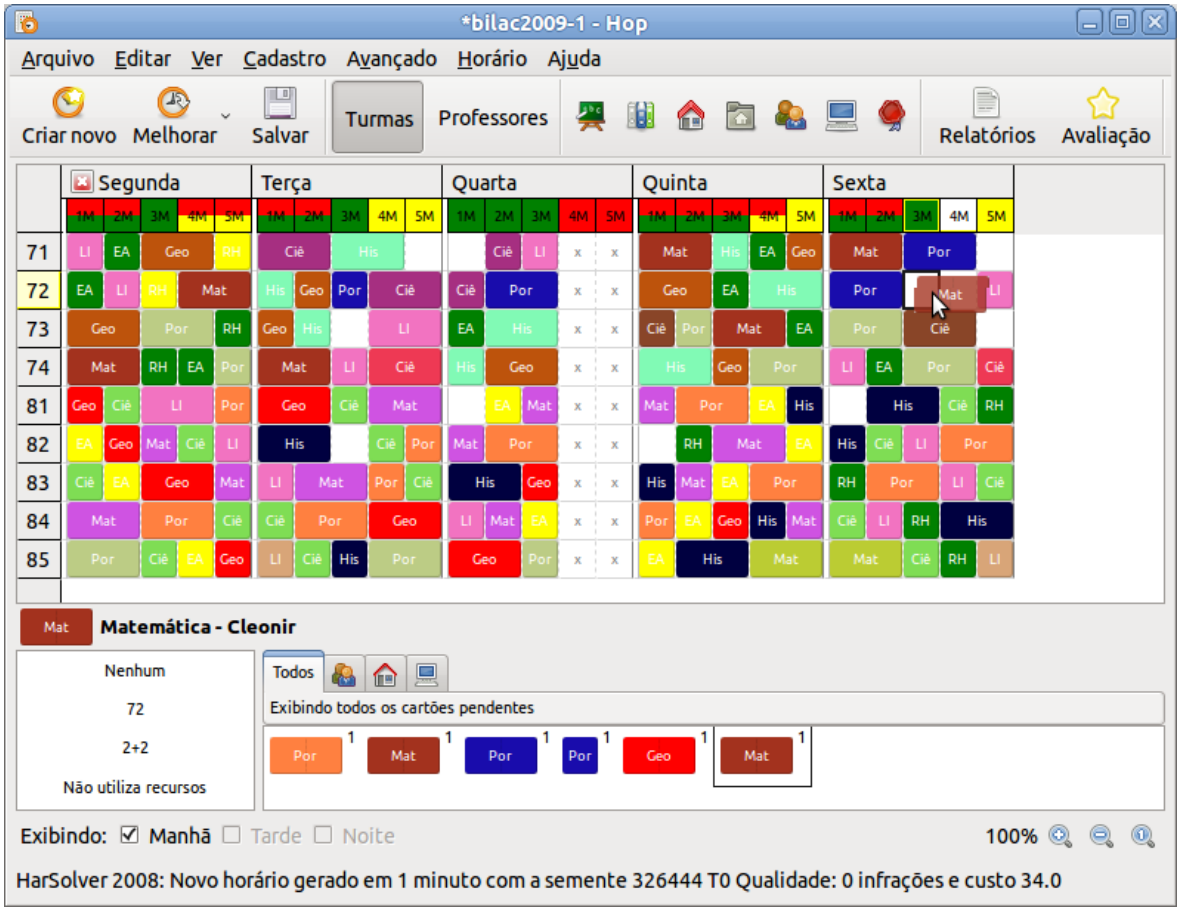

Figura 5.5: Tela Principal do HOP-2010

5.3 Arquivo de Representação da Instância

Ao invés de utilizar vários arquivos de texto para representar a instância de forma segmentada como no HOP-2009, optou-se por utilizar um único arquivo em XML para representar toda a instância.

Este XML é gerado pela interface do HOP-2010 e carregado pelo novo resolvedor implementado no Capítulo 6.

Na instância de exemplo a seguir, os comentários iniciados por // foram incluídos para explicar os elementos relevantes da representação, mas não fazem parte do formato. Comentários do tipo **//...** indicam que o elemento anterior, com a mesma identação, pode se repetir conforme a cardinalidade indicada no seu elemento pai.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance>
  <days>5</days> //Número de dias do horário
  <shifts>1</shifts> //Número de turnos do horário
  <periods>5</periods> //Número de períodos por turno do horário
```

```
<teachers> //0..N
  <teacher>
    <id>0</id> //Identificador único
    <name>Adriane</name> //Nome
    <abrName>Adriane</abrName> //Abreviatura
    <freeDays>2</freeDays> //Restrição 11
    <lessonsBetweenShifts>false</lessonsBetweenShifts> //Restrição 19
    <maxPeriodHoles>0</maxPeriodHoles> //Restrição 8
    <maxShiftHoles>1</maxShiftHoles> //Restrição 9
    <maxDayHoles>5</maxDayHoles> //Restrição 10
    <workShiftsByDay>1</workShiftsByDay> //Restrição 18
    <availability> //Disponibilidade do professor (Restrições 4 e 6)
      <periodAvailability> //0..N (Indica a disponibilidade de um período)
        <day>0</day>
        <shift>0</shift>
        <period>0</period>
        <status>0</status> //0=Disponível, 1=Indesejado, 2=Indisponível
      </periodAvailability>
      //...
    </availability>
  </teacher>
  //...
</teachers>

<classes> //0..N
  <class>
    <id>0</id> //Identificador único da turma
    <name>71</name> //Nome da turma
    <syncGroups> //0..N (Restrição 27 - não considerada neste trabalho)
      <syncGroup>
        <group>0</group>
        <day>0</day>
      </syncGroup>
      //...
    </syncGroups>
    <intervals> //0..N (Sinais de intervalo da turma)
      <interval>
        <shift>0</shift> //Turno do intervalo
        <period>2</period> //Período anterior ao intervalo
      </interval>
      //...
    </intervals>
    <availability> //Disponibilidade da turma (Restrições 5 e 7)
      //idem ao formato da disponibilidade de professores
    </availability>
  </class>
  //...
</classes>

<subjects> //0..N
  <subject>
    <id>0</id> //Identificador único da disciplina
    <name>Ciências</name> //Nome
    <abrName>Ciê</abrName> //Abreviatura
    <group>0</group> //Restrição 26
       //(Cada número identifica um grupo. Disciplinas do mesmo grupo não
       // serão colocadas no mesmo dia para uma turma)
  </subject>
  //...
</subjects>

<resources> //0..N
  <resource>
    <id>0</id> //Identificado único do recurso
    <name>Laboratório de Química</name> //Nome
    <quantity>2</quantity> //Quantidade disponível
    <availability> //Disponibilidade do Recurso (Restrições 13 e 14)
      //idem ao formato da disponibilidade de professores
    </availability>
  </resource>
  //...
</resources>
```

```
  <contracts> //0..N
    <contract>
      <id>0</id> //Identificador único do contrato
      <teachers> //Professores do contrato
        <teacher>
          <id>10</id> //Identificador único do professor
        </teacher>
        //...
      </teachers>
      <classes> //Turmas do contrato
        <class>
          <id>0</id> //Identificador único da turma
        </class>
        //...
      </classes>
      <resources> //Recursos utilizados pelo contrato
        <resource>
           <id>0</id> //Identificado único do recurso
           <usage>1</usage> //Quantidade utilizada pelo contrato
        </resource>
        //...
      </resources>
      <lessons>3</lessons> //Número de aulas do contrato
      <maxLessonsByDay>2</maxLessonsByDay> //Restrição 24
      <distributionBlocks exact="false"> //Se exact == false representa
                      // a Restrição 22  senão a 23
        <distributionBlock> //0..N
          <size>2</size> //Tamanho de cada aula
        </distributionBlock>
        //...
      </distributionBlocks>
      <avoidIntervalsBetweenLessons> //Restrição 25
          false
      </avoidIntervalsBetweenLessons>
      <fixedAllocations> //Restrição 20
         <fixedAllocation> //Define o dia, turno e período que uma aula deve acontecer
            <day>1</day><shift>0</shift><period>0</period>
         </fixedAllocation>
         //...
      </fixedAllocations>
      <flexibleAllocations> //Representa as aulas atualmente no horário que podem
                            //ser modificadas de posição pelo resolvedor
        <flexibleAllocation>
          <day>1</day><shift>0</shift><period>0</period>
        </flexibleAllocation>
        //...
      </flexibleAllocations>
      <subjectId>0</subjectId> //Identificador único da disciplina do contrato
    </contract>
    //...
  </contracts>
</instance>
```

6 IMPLEMENTAÇÃO DO RESOLVEDOR

Devido a extensão do código, neste capítulo é apresentada uma visão em alto nível da implementação. A cada seção são apresentados mais detalhes em relação aos pontos mais relevantes do ponto de vista da otimização.

O resolvedor implementado neste capítulo, doravante denominado **NeoSolver**, foi programado na linguagem C++ e compilado com o compilador GCC na versão 4.4.X. As convenções adotadas na implementação são disponibilizadas no Apêndice A. No Apêndice B são apresentadas opções do binário.

6.1 Visão Geral

O resolvedor executa sequencialmente as seguintes etapas de alto nível:

1. Leitura dos parâmetros do programa;
2. Leitura da instância do problema a partir do XML gerado pela interface;
3. Construção de uma solução inicial com uma heurística construtiva;
4. Refinamento da solução inicial utilizando uma busca tabu.

As principais classes do resolvedor são apresentadas na Figura 6.1 em um diagrama simplificado. As classes sublinhadas identificam classes abstratas.

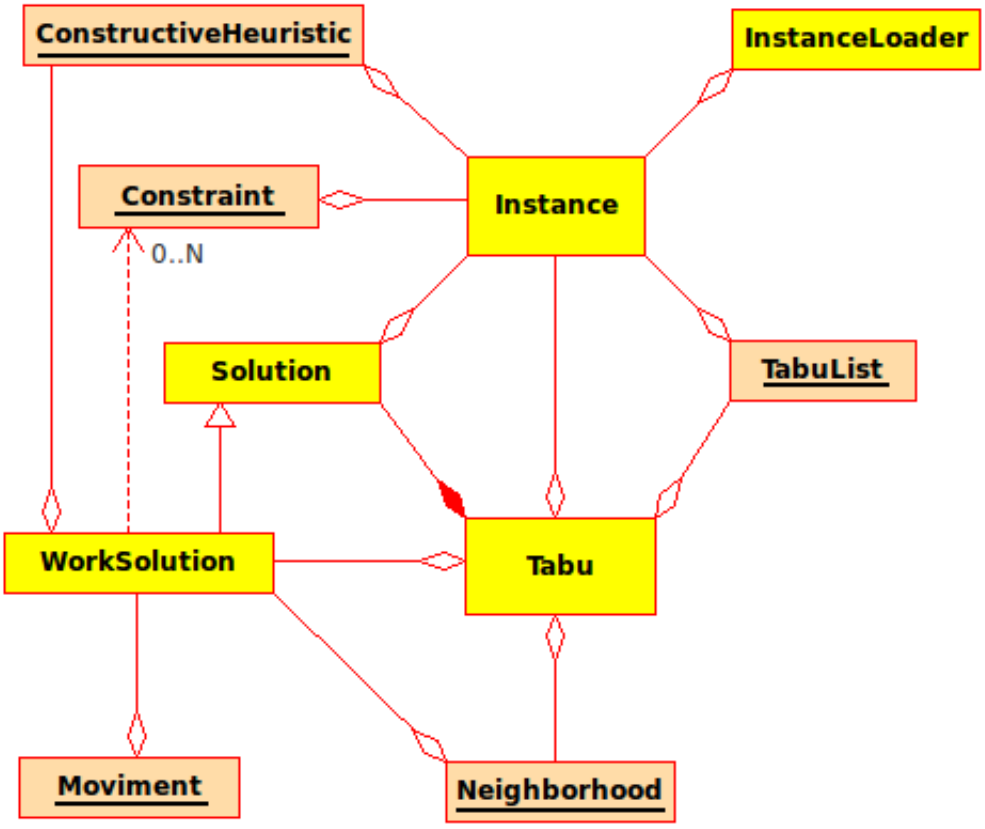

Figura 6.1: Arquitetura simplificada do resolvedor

Um breve resumo de cada classe:

Tabu: Implementa uma busca tabu com estratégias de diversificação e intensificação.

Instance: Contém todas as entidades e especificações do problema.

InstanceLoader: Responsável por ler o arquivo XML e criar o objeto Instance.

ConstructiveHeuristic: Implementa um algoritmo construtivo para criar a solução Inicial do problema.

Solution: Representação básica da solução. Contém apenas os dados para representar um horário.

WorkSolution: Em adição a Solution possui estruturas adicionais para avaliar o horário.

Moviment: Representa uma modificação que pode ser feita ou desfeita na solução.

TabuList: Implementa a lista tabu utilizada pela classe TabuList.

Constraint: Representa uma das várias restrições do horário.

Neighborhood: Implementa a lógica de exploração da vizinhança.

Cada uma destas classes serão descritas nas seções seguintes de acordo com o seu comportamento genérico. As classes abstratas terão suas especializações citadas sempre que os detalhes forem considerados relevantes.

Na Figura 6.2 é apresentada uma versão simplificada da função $main()$. Ao observar a função é possível perceber todas as etapas de preparação realizadas antes da busca começar. Além disso, o relacionamento entre as classes e seus principais parâmetros são exibidos.

main.cpp (~/neo/simplificado) - gedit

Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda

main.cpp

```
56 int main(int argc, char**argv) {
57    //Carrega a instância a partir do xml
58    InstanceLoader loader("/home/arton/bilac_2010.xml");
59    Instance* inst = loader.load();
60
61    //Configuração da solução com as restrições e peso das penalidades
62    Solution solu;
63    WorkSolution workSolu(solu);
64    workSolu.addConstraint(new TeacherClash(10)); //Conflito de professor
65    workSolu.addConstraint(new ClassClash(10)); //Conflito de turma
66    //workSolu.addConstraint(Várias outras restrições...);
67    workSolu.loadConstraints();
68
69    //Constrói a solução inicial utilizando um algoritmo guloso
70    ConstructiveHeuristic* builder = SmartConstructor();
71    builder->build(workSolu);
72
73    //Configuração da Lista Tabu com tenure variando de 10 a 30
74    TabuList *tabuList = TLBlockContractPeriod(10,30);
75
76    //Cria a lógica usada para explorar a vizinhança
77    Neighborhood *neighborhood = NBlockChange(&workSolu,tabuList);
78
79    //Configuração da busca tabu
80    Tabu tabu(&workSolu,tabuList,neighborhood);
81    tabu.setMaxTime(60); //60 segundos
82    tabu.setMaxIteration(INFINITO);
83
84    //Quando disparar a diversificação?
85    tabu.setDiversificationInterval(1000); //A cada 1000 iterações sem melhora
86    tabu.setDiversificationDuration(200);  //Diversifica por 200 iterações
87    tabu.setDiversificationIntensity(0.1); //Cada iteração diversifica 10%
88
89    //Inicia a busca e imprime melhor solução quando ela terminar.
90    tabu.search();
91    solu.print();
92
93    //Liberação do que foi alocado
94    delete tabuList; delete neighborhood; delete builder; delete inst;
95    return 0;
96 }
```

Figura 6.2: Versão simplificada da função principal do resolvedor

6.2 Classe Tabu

Esta é a classe central do resolvedor que implementa o algoritmo da busca tabu de maneira semelhante ao apresentado na seção 3.2.3. Esta implementação é feita no método *search* da classe **Tabu** e apresentada na Figura 6.3.

Tabu.cpp (~/neo) - gedit

Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda

Tabu.cpp

```
34 void Tabu::search(){
35
36    Crono crono;
37    Moviment *bestMove;
38
39    m_incumbent=*m_workSolu; // solucao incumbente recebe a solucao inicial
40
41    EventInfo::onStartSolver(&m_incumbent,&crono);
42
43    crono.run();
44    while (crono.getTime() < m_maxTime && m_iteration < m_maxIteration &&
45           m_incumbent.getValue()>0){
46
47       bestMove = m_neighborhood->search(&m_incumbent);
48
49       if (!bestMove->isReseted()){
50          bestMove->apply();
51       }  else {
52          cout << "Nenhum movimento disponível para aplicar." << endl;
53       }
54
55       m_tabuList->add(m_workSolu->getChanges());
56
57       m_noImprovement++;
58       if ((*m_workSolu)<m_incumbent){  // atualiza solucao incumbente
59          m_incumbent = *m_workSolu;
60          EventInfo::onNewIncumbentFound(m_iteration);
61          m_noImprovement = 0;
62       }
63
64       EventInfo::onIteration(m_workSolu,m_iteration);
65
66       diversification();
67       intensification();
68
69       m_iteration++;
70       m_tabuList->nextIteration();
71    }
72    m_workSolu->loadFrom(&m_incumbent);
73    EventInfo::onEndSolver(m_iteration);
74 }
```

Figura 6.3: Método de busca da classe Tabu

Nesta implementação são incluídos procedimentos de diversificação (linha 66) e intensificação (linha 67).

6.2.1 Diversificação

A diversificação é ativada sempre que um número `m_noImprovement` de iterações sem melhora é alcançado. Nesse momento são selecionados os contratos que menos se movimentaram. Cada um destes contratos têm seu fator de movimentação aumentado, de forma que sejam preferencialmente escolhidos pela vizinhança. A influência do fator de movimentação pode ser vista na linha 32 do algoritmo da Figura 6.13. Na Figura 6.4 é apresentada a implementação do procedimento de diversificação.

Tabu.cpp (~/neo) - gedit

Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda

Tabu.cpp

```
91 void Tabu::diversification(){
92    //Adiciona as alterações na lista de frequencia
93    m_freqList.add(m_workSolu.getChanges();
94
95    //Ativa a diversificação
96    if( m_noImprovement > m_diversificationInterval && !m_isDiversificating){
97       Contracts** list = m_freqList.getLowFreqContracts();
98       for(/*cada contrato c em list*/){
99          c->setMoveFactor(m_diversificationIntensity);
100      }
101      m_maxDiverstificatonIter = m_iteration + m_diversificationDuration;
102      clearAll();
103      m_isDiversificating = true;
104   }
105
106   //Finaliza a diversificação
107   if( m_iteration == m_maxDiverstificatonIter ){
108      m_noImprovement = 0;
109      for (/*cada contrato c da instancia*/) {
110         c->setMoveFactor(1);
111      }
112      clearAll();
113      m_isDiversificating = false;
114   }
115 }
116
117 void Tabu::clearAll() { //Limpa todas as estrutura auxiliares
118    m_neighborhood->reset();
119    m_tabuList->clear();
120    m_freqList.clear();
121 }
```

Figura 6.4: Representação em pseudocódigo/C++ do procedimento de diversificação

6.2.2 Intensificação

A intensificação é feita de maneira análoga a diversificação, porém, aumenta o fator de movimentação dos contratos que mais se movimentaram.

6.3 Classe Solution

A classe solução é uma representação simples de um horário. Seu uso no programa é feito exclusivamente para armazenar a melhor solução encontrada durante o procedimento

de busca. Ou seja, toda a estrutura de dados que armazena o horário precisa ser copiada repetidamente e de maneira eficiente. Na Figura 6.5 é apresentado o diagrama de classe Solution.

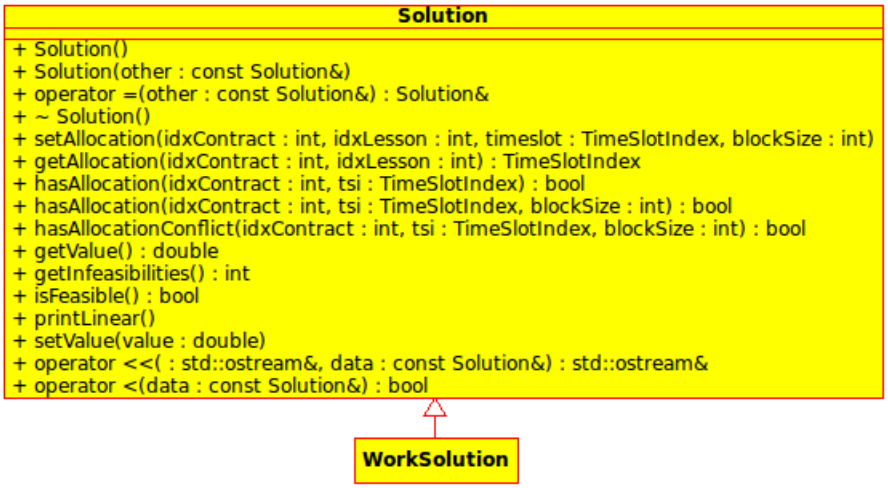

Figura 6.5: Classe Solution

Para o problema de geração de horários, a solução necessita armazenar, para cada aula de um contrato, o dia, o turno e o período em que esta aula ocorre. Ao invés de se armazenar todos estes valores optou-se pela representação indexada da Tabela 6.1. Nessa representação, uma aula no último período da tarde de quinta é representada simplesmente pelo seu índice de valor 27. Este índice numérico é denominado **TimeSlotIndex** ou de maneira abreviada TSI.

Turno/Período	Seg	Ter	Qua	Qui	Sex	Sáb	Dom
Manhã/1	0	8	16	24	32	40	48
Manhã/2	1	9	17	25	33	41	49
Manhã/3	2	10	18	26	34	42	50
Manhã/4	3	11	19	27	35	43	51
Tarde/1	4	12	20	28	36	44	52
Tarde/2	5	13	21	29	37	45	53
Tarde/3	6	14	22	30	38	46	54
Tarde/4	7	15	23	31	39	47	55

Tabela 6.1: Representação dos períodos através de índices

Com o uso da representação indexada, todo o horário é representado em único vetor de inteiros. As aulas de cada contrato são identificadas conforme a posição que ocupam no vetor. Na Tabela 6.2, por exemplo, o contrato c_0 é representado nas posição 0 e 1 do vetor solução. E seus valores correspondem aos TSIs 4 e 5, respectivamente. Então, conforme a Tabela 6.1 é possível saber que o contrato c_0 tem uma aula na segunda de manhã no primeiro período e outra aula na terça de tarde no segundo período.

c_0		c_1				c_2		
4	13	33	46	47	4	29	37	0
0	1	2	3	4	5	6	7	8

Tabela 6.2: Vetor solução. Cada índice é um *TimeSlotIndex*

Para modificar ou obter os dados do vetor solução a classe fornece, respectivamente, os métodos *setAllocation* e *getAllocation*. No vetor da Tabela 6.1, uma chamada a *getAllocation(1,2)* retorna o TSI 47.

Outros métodos importantes desta classe são *getValue* e *getInfeasibilities*. O método *getValue* corresponde a função f definida no Capítulo 3. Quanto menor o seu valor, melhor a solução. O método *getInfeasibilities* serve para retornar o número de infactibilidades da solução. Para comparar a qualidade entre duas soluções,s_1 e s_2, considera-se qualitativamente melhor aquela que tem menos infactibilidades, ou seja, aquela solução que tem um número maior de restrições fortes satisfeitas.

Na Tabela 6.3 são apresentados alguns valores de exemplo para três soluções.

Solução	Valor	Infactibilidades
s_1	23.4	4
s_2	31.2	3
s_3	12.1.	3

Tabela 6.3: Valores exemplo para soluções s_1, s_2, s_3

De acordo com a sobrecarga do operador **<** da classe *Solution* são válidos os resultados da Tabela 6.4.

Comparação	Resultado
$s_1 < s_2$	Não
$s_1 < s_3$	Não
$s_2 < s_1$	Sim
$s_2 < s_3$	Não

Tabela 6.4: Exemplo da comparação de qualidade entre as soluções da Tabela 6.3

6.4 Classe Instance

Esta classe representa a instância do problema com todas as suas especificações. Através do *InstanceLoader* são carregados professores, turmas, disciplinas, recursos e contratos. Todas estas entidades são indexadas com base 0. Um atributo importante da instância é a classe *Dimensions* que contém informações relativas ao número de dias, períodos e turnos do horário. Esta classe também permite fazer a conversão de um *TimeSlotIndex* para sua representação em dia, turno e período. Esta representação é fornecida pela classe ***TimeSlot***.

Como visto na Figura 6.1 os dados da instância são necessários por praticamente todas as classes do programa. Este acesso é feito a partir do método estático *getInstance()*. De maneira análoga, este acesso também é disponibilizado na classe *Dimensions*.

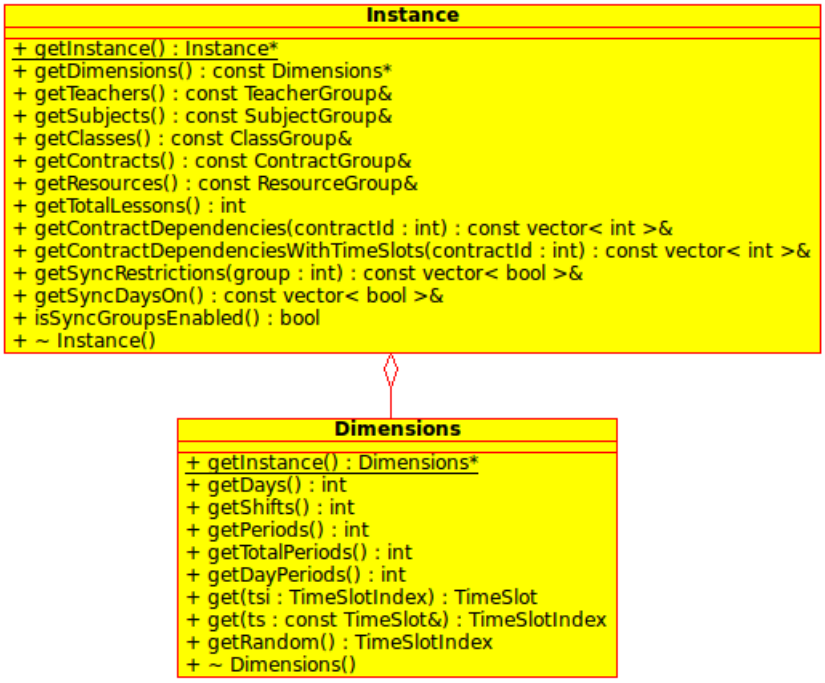

Figura 6.6: Classe Instance e Dimensions

6.5 Classe InstanceLoader

Esta classe, representada no diagrama da Figura 6.7, possui apenas o método *load* que carrega os dados a partir de um arquivo XML para um objeto *Instance*.

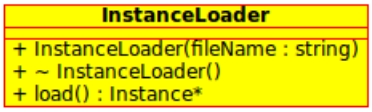

Figura 6.7: Classe InstanceLoader

6.6 Classe ConstructiveHeuristic

O principal método da classe *ConstructiveHeuristic* é o *build*, cujo objetivo é construir uma solução inicial. Esta classe foi especializada de duas formas: através da classe *RandConstructor*, que implementa o *build* aleatoriamente, e através da classe *SmartConstructor* que utiliza um algoritmo guloso. Embora seja possível escolher um destes métodos de construção, a heurística *SmartConstructor* proporciona melhores soluções iniciais. O diagrama da classe ConstructiveHeuristic e suas especializações é apresentado na Figura 6.8.

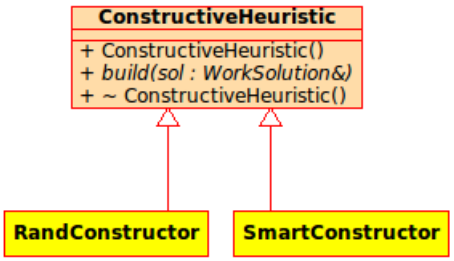

Figura 6.8: Classe ConstructiveHeuristic e especializações

6.7 Classe Constraint

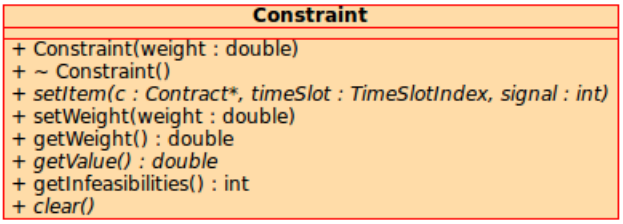

Figura 6.9: Classe Constraint

A classe *Constraint*, conforme representada no diagrama da Figura 6.9, serve de base para as implementações de todas as restrições apresentadas no Capítulo 2. Os métodos *getValue* e *getInfeasibilities* tem a mesma função dos métodos correspondentes na classe *Solution*, porém avaliam a restrição isoladamente de acordo com um peso.

Na Figura 6.10, são apresentadas as especializações de *Constraint*. A medida que movimentos são realizados na solução, as restrições são informadas através do método *setItem* e seus valores são atualizados.

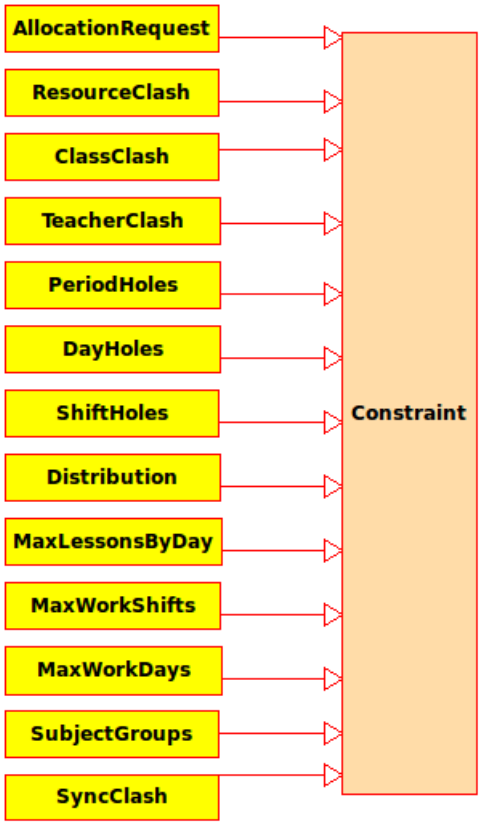

Figura 6.10: Classe Constraint e especializações

6.8 Classe WorkSolution

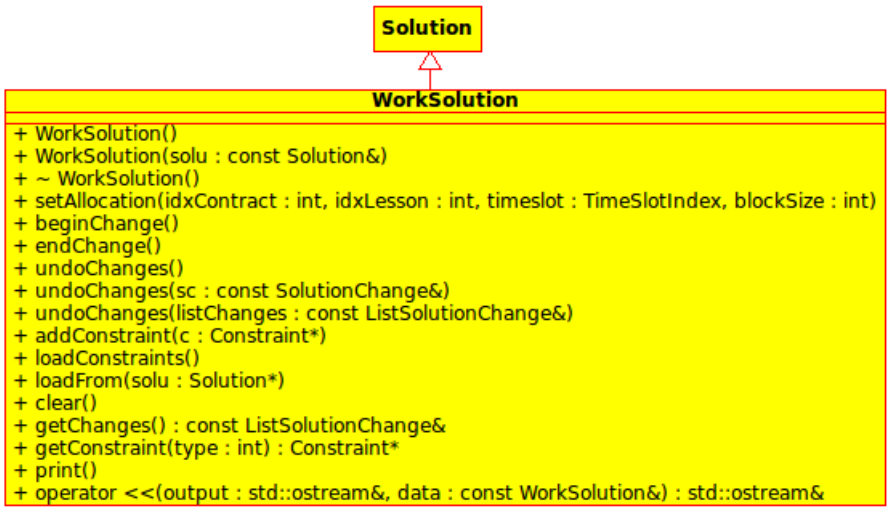

Figura 6.11: Classe WorkSolution

Esta classe, representada na Figura 6.11, é responsável pelas seguintes responsabilidades:

- Informar as modificações da solução para as restrições atualizarem seus valores.

- Armazenar as alterações para que possam ser desfeitas pelo método *undoChanges*.
- Avaliar a qualidade da solução.

Os valores dos métodos *getValue* e *getInfeasibilities* correspondem ao somatório dos valores retornados, pelos métodos análogos das restrições a classe gerencia.

6.9 Classe Moviment

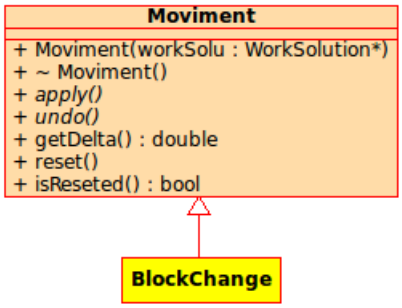

Figura 6.12: Classe Moviment e sua especialização

Esta classe é responsável por efetuar as modificações na solução com os métodos *apply* e *undo*, conforme representados na Figura 6.12. Para obter o valor da modificação causada na solução, utiliza-se o método *getDelta*. Quanto menor o valor retornado, melhor é o movimento.

Moviment é especializada pela classe *BlockChange* e implementada conforme a Figura 6.13.

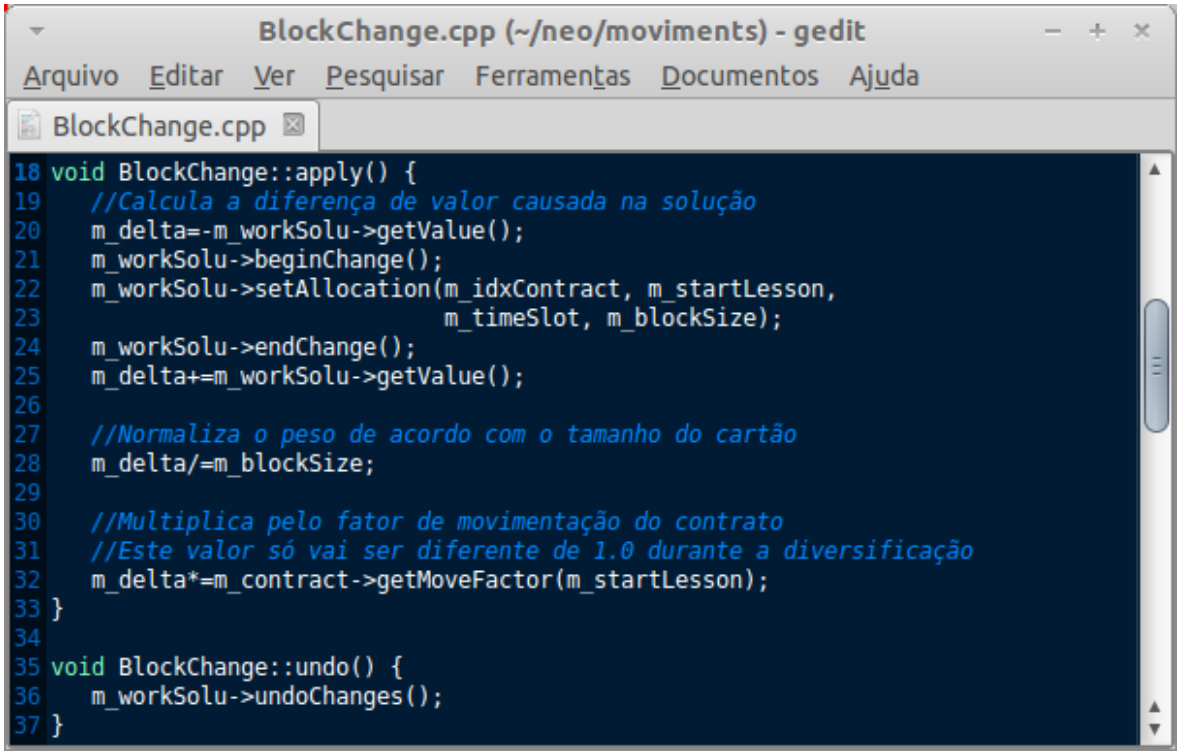

Figura 6.13: Métodos para aplicar e desfazer um movimento

6.10 Classe TabuList

Esta classe define a lista tabu genérica usada pelo programa para bloquear determinados movimentos por um número de iterações. Quando instanciada, o seu construtor recebe os parâmetros *minTenure* e *maxTenure*. Este parâmetro define o intervalo de iterações que uma determinada característica fica proibida quando adicionada na lista. A decisão relativa a quais características serão consideradas proibidas é feita nas classes *TLContracPeriod* e *TLBlockContractPeriod*.

A representação da classe TabuList e suas especializações são apresentadas na Figura 6.14.

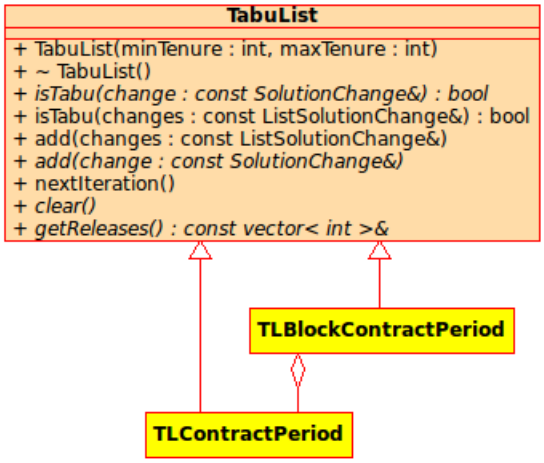

Figura 6.14: Classe TabuList e especializações

A classe *TLContractPeriod* considera que uma aula não pode voltar para o lugar de onde se moveu por um determinado número de iterações. Ou seja, ao mover uma aula de um contrato c do TSI X para o TSI Y e adicionar essa alteração na lista tabu, o contrato c fica proibido de efetuar um movimento em que o destino seja TSI X. Essa proibição dura t iterações, onde t é um número aleatório entre *minTenure* e *maxTenure*.

A classe *TLBlockContractPeriod* utiliza a mesma proibição de *TLContractPeriod*, porém discrima por tamanho do bloco do movimento. Ou seja, ao mover uma aula de um contrato c do TSI X para o TSI Y com bloco de tamanho N, ao adicionar essa alteração na lista tabu, o contrato c fica proibido de efetuar um movimento em que o destino seja TSI X e o bloco seja de tamanho N. Caso o tamanho do bloco seja diferente de N é permitido que este se mova para TSI X.

Ambas as implementações utilizam uma fila de tamanho *maxTenure* e uma matrix bidimensional indexada por contrato e TSI, conforme é apresentado na Figura 6.15. Os testes realizados com ambas as proibições evidenciam desempenhos diferentes conforme a instância, mas não foi possível identificar a situação em que é melhor aplicar uma ou outra implementação.

TLContractPeriod.cpp (~/neo) - gedit

Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda

TLContractPeriod.cpp

```
4  TLContractPeriod::TLContractPeriod(int minTenure, int maxTenure):
5     TabuList(minTenure,maxTenure){
6     int contractSize = m_inst->getContracts().size();
7     m_contractPeriod = new int*[contractSize];
8     for(int i=0; i<contractSize; i++){
9        m_contractPeriod[i]= new int[m_inst->getDimensions()->getTotalPeriods()];
10    }
11    m_queueRelease = new vector<int>[m_maxTenure];
12    clear(); //preenche m_queuRelease com -1, m_topQueue=0 e m_firstAdd=1
13 }
14
15 bool TLContractPeriod::isTabu(const SolutionChange& change){
16    int cId = change.getContractId();
17    TimeSlotIndex tsi = change.getNewTimeSlot();
18    if( m_iteration < m_contractPeriod[cId][tsi]) {
19       return true;
20    }
21    return false;
22 }
23
24 void TLContractPeriod::add(const SolutionChange& change){
25    int tabuTenure = m_minTenure + rand()%(1+m_maxTenure-m_minTenure);
26    int releaseIteration = tabuTenure + m_iteration;
27    int queueTenure = ( m_topQueue + tabuTenure) % m_maxTenure;
28
29    if (m_isFirstAdd) {
30       m_queueRelease[m_topQueue].clear();
31       m_isFirstAdd = false;
32    }
33
34    int cId = change.getContractId();
35    TimeSlotIndex tsi = change.getOldTimeSlot();
36
37    m_contractPeriod[cId][tsi] = releaseIteration;
38    m_queueRelease[queueTenure].push_back(cId);
39
40 }
41
```

Figura 6.15: Implementação da lista tabu

6.11 Classe Neighborhood

A classe Neighborhood, representada no diagrama da Figura 6.16, é implementada a lógica de exploração dos movimentos. O método *search* realiza a busca e retorna o melhor movimento que não seja tabu. Para esta classe foram feitas duas especializações: *NBlockChangeNoCache* e *NBlockChange*

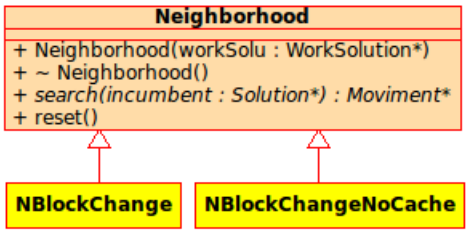

Figura 6.16: Classe Neighborhood e especializações

A vizinhança *NBlockChangeNoCache* é uma busca local típica que seleciona o melhor movimento. Desta forma, avalia todos os movimentos possíveis e retorna o melhor que for encontrado respeitando a lista tabu. Esta implementação, embora simples, é cara devido ao custo de avaliação dos movimentos.

A vizinhança *NBlockChange* utiliza um algoritmo mais eficiente demonstrado na Figura 6.17. Este algoritmo mantém uma memória dos melhores movimentos de cada contrato. A medida que alterações são feitas na solução atual (geralmente poucas) somente os movimentos relacionados com estas alterações são marcados como "resetados", ou seja, precisam ser reavaliados.

NBlockChange.cpp (~/neo/simplificado) - gedit

Arquivo Editar Ver Pesquisar Ferramentas Documentos Ajuda

NBlockChange.cpp

```
13 Moviment* NBlockChange::search(Solution *incumbent){
14
15    bestMove.reset();
16
17    for (/*cada alteração s que ocorreu na solução*/) {
18       int c = s.getContractId();
19       const vector<int>& deps = m_inst->getContractDependencies(c);
20       for (/*cada contrato d em deps*/) {
21          m_memory[d].reset();
22       }
23    }
24
25    for (/*cada contrato c da instância*/) do
26       if(m_memory[c].isReseted()){
27          for (/*cada aula j do contrato c*/) {
28             for (/*cada TSI it que o contrato possa se mover/*/){
29                BlockChange m(m_workSolu,c,currentLesson,it,block(c,j));
30                m.apply();
31                if (/*m não é tabu*/ && m.getDelta()<m_memory[c].getDelta()){
32                   m_memory[c]=move;
33                }
34                move.undo();
35             }
36          }
37       }
38       //Atualiza o melhor movimento
39       if (m_memory[c].getDelta()<m_bestMove.getDelta()){
40          m_bestMove=m_memory[c];
41       }
42    }
43    return &m_bestMove;
44 }
```

Figura 6.17: Representação em peseudocódigo/C++ da vizinhança *NBlockChange*

7 RESULTADOS

Embora no projeto neste trabalho tenha sido prevista uma avaliação dos resultados da implementação em comparação com os resultados elaborados manualmente pelos educadores das escolas, não foi possível obter dados precisos para esta comparação. Devido à natureza combinatória, obviamente os resultados de um processo computacional geram soluções melhores e em um tempo menor do que um especialista humano. Por esse motivo a avaliação manual foi substituída por uma comparação de desempenho com o resolvedor do HOP-2009 e com o resolvedor comercial Urânia (GEHA, 2010).

Para a realização dos testes foram utilizadas três instâncias obtidas em escolas de Santa Maria que estão disponíveis no Apêndice E.

7.1 Comparação com o Urânia

O Urânia é um *software* proprietário para geração de quadro de horários escolares que é comercializado sob uma licença de uso. É bastante conhecido nacionalmente e tem uma vasta gama de opções. A sua principal diferença em relação ao HOP-2010 está no modo como ele define as distribuições das aulas. As variações de distribuição do HOP-2010 são um *subset* das variações do Urânia. Na execução dos testes, a tabela 7.1 foi utilizada para representar a mesma restrição de horário do HOP-2010 no Urânia. Detalhes sobre o significado das distribuições do Urânia podem ser vistos no Apêndice C. No Apêndice D são apresentadas as distribuições do HOP-2010.

HOP	Urânia
^1	A
^2	G
^3	Q
^4	U
^5	U
^6	U
(1 1)^2	E
(2 1)^2	C
(2 2)^2	G
(3 2)^3	Q
(1 1 1)^2	E
(2 2 1)^2	G
(2 2 2)^2	G
2	N
3	O
4	P
5	Z
6	$
1+1	A
2+1	K
2+2	N
1+1+1	A
2+1+1	K
2+2+1	N
2+2+2	N

Tabela 7.1: Tabela de equivalência de distribuições

7.2 Experimentos

Nos testes são considerados o resolvedor Har2008 (Har) que acompanha o HOP 2009, o NeoSolver (Neo), desenvolvido nesse trabalho, e o Urânia 2010 (Ura). A máquina usada tem as seguintes especificações:

Processador Intel Core 2 Duo T6600 2.2GHz

Memória RAM 4GB 800 MHz

Cada teste consiste na análise de qualidade em 1, 5 e 30 minutos de execução para cada instância. Cada resolvedor foi executado 5 vezes e o resultado tabulado a partir da mediana em relação a restrição mais importante da instância.

7.2.1 Instância 1: Maneco

Nesta instância a principal restrição é o número de janelas entre períodos do professor. O número de aulas dispersas na turma é a segunda restrição mais importante.

Restrição	N	H	U
Aulas não alocadas	**0**	3	**0**
Janelas entre Períodos do Professor	13	14	**9**
Aulas Dispersas na Turma	34	**3**	12
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	**0**	**0**
Distribuições sugeridas não atendidas	**0**	3	**0**

Tabela 7.2: Resultados para 1 minuto de execução na Instância 1

Restrição	N	H	U
Aulas não alocadas	**0**	1	**0**
Janelas entre Períodos do Professor	11	13	**8**
Aulas Dispersas na Turma	13	**0**	9
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	**0**	**0**
Distribuições sugeridas não atendidas	**0**	1	**0**

Tabela 7.3: Resultados para 5 minutos de execução na Instância 1

Restrição	N	H	U
Aulas não alocadas	**0**	**0**	**0**
Janelas entre Períodos do Professor	9	15	**8**
Aulas Dispersas na Turma	4	**0**	9
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	**0**	**0**
Distribuições sugeridas não atendidas	**0**	**0**	**0**

Tabela 7.4: Resultados para 30 minutos de execução na Instância 1

7.2.2 Instância 2: Bilac

Nesta instância a principal restrição é o número de dias de trabalho em excesso. O número de aulas janelas entre períodos do professor é a segunda restrição mais importante.

Restrição	N	H	U
Aulas não alocadas	1	7	**0**
Janelas entre Períodos do Professor	10	9	**2**
Aulas Dispersas na Turma	4	**0**	5
Dias de Trabalho em Excesso	**6**	10	11
Aulas em Período Indesejado	**1**	3	2
Contratos com Separação imprópria pelo Intervalo	2	**1**	**1**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	3	**0**
Distribuições sugeridas não atendidas	2	**1**	4

Tabela 7.5: Resultados para 1 minuto de execução na Instância 2

Restrição	N	H	U
Aulas não alocadas	**0**	2	**0**
Janelas entre Períodos do Professor	7	10	**2**
Aulas Dispersas na Turma	8	**0**	6
Dias de Trabalho em Excesso	**6**	8	13
Aulas em Período Indesejado	**2**	3	3
Contratos com Separação imprópria pelo Intervalo	1	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	1	**0**
Distribuições sugeridas não atendidas	6	**0**	**0**

Tabela 7.6: Resultados para 5 minutos de execução na Instância 2

Restrição	N	H	U
Aulas não alocadas	**0**	**0**	2
Janelas entre Períodos do Professor	7	7	**2**
Aulas Dispersas na Turma	**2**	**2**	5
Dias de Trabalho em Excesso	**5**	6	11
Aulas em Período Indesejado	**1**	**1**	3
Contratos com Separação imprópria pelo Intervalo	2	1	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	**0**	1
Distribuições sugeridas não atendidas	2	2	**1**

Tabela 7.7: Resultados para 30 minutos de execução na Instância 2

7.2.3 Instância 3: CTISM

Nesta instância as principais restrições são, em ordem, as distribuições obrigatórias, aulas dispersas na turma e número de janelas entre períodos do professor.

Restrição	N	H	U
Aulas não alocadas	**0**	56	22
Janelas entre Períodos do Professor	**0**	4	11
Aulas Dispersas na Turma	**0**	**0**	6
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	15	5
Distribuições sugeridas não atendidas	**0**	7	13

Tabela 7.8: Resultados para 1 minuto de execução na Instância 3

Restrição	N	H	U
Aulas não alocadas	**0**	32	13
Janelas entre Períodos do Professor	**0**	**0**	8
Aulas Dispersas na Turma	**0**	**0**	6
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	8	3
Distribuições sugeridas não atendidas	**0**	5	11

Tabela 7.9: Resultados para 5 minutos de execução na Instância 3

Restrição	N	H	U
Aulas não alocadas	**0**	15	2
Janelas entre Períodos do Professor	**0**	**0**	12
Aulas Dispersas na Turma	**0**	**0**	8
Dias de Trabalho em Excesso	**0**	**0**	**0**
Aulas em Período Indesejado	**0**	**0**	**0**
Contratos com Separação imprópria pelo Intervalo	**0**	**0**	**0**
Distribuições com aulas excedendo o limite diário	**0**	**0**	**0**
Distribuições obrigatórias não atendidas	**0**	4	**0**
Distribuições sugeridas não atendidas	**0**	**0**	15

Tabela 7.10: Resultados para 30 minutos de execução na Instância 3

7.2.4 Análise dos Resultados

Na instância 1 o Neosolver apresentou resultados ligeiramente inferiores aos outros resolvedores nas principais restrições. Porém, é possível notar que o NeoSolver continua melhorando o horário com o passar do tempo. Já os demais resolvedores permanecem quase estagnados. Isto permite inferir que se fosse fornecido um tempo maior que 30 minutos de resolução, o NeoSolver provavelmente obteria resultados melhores que o HarSolver e o Urânia.

Na instância 2, considerando que a principal restrição é o número de dias de trabalho em excesso, o Urânia apresentou o pior desempenho. O NeoSolver apresentou desempenho ligeiramente superior ao HarSolver. Caso mais tempo fosse disponibilizado, é provável que o HarSolver e o NeoSolver atingiriam desempenhos similares. Já o Urânia provavelmente ficaria estagnado.

Na instância 3, onde são utilizados vários blocos grandes de aula o resolvedor desenvolvido conseguiu sempre chegar na solução ótima com apenas 1 minuto de resolução. Enquanto os demais resolvedores obtiveram soluções substancialmente inferiores. Esses resultados estão diretamente relacionados ao movimento em bloco implementado e que provavelmente não é adotado pelos outros resolvedores.

8 CONSIDERAÇÕES FINAIS

8.1 Conclusão

Foi possível constatar que o resolvedor desenvolvido gera horários de maior qualidade em relação ao processo manual, mesmo com a dificuldade de coletar horários confeccionados manualmente para fins de comparação. Esta afirmação é embasada pelo fato dos usuários, após utilizarem o resolvedor, demonstrarem resistência em voltar à confecção manual dos horários, uma vez que testemunharam o ganho em tempo e qualidade proporcionado pelo auxílio do computador.

Conforme os resultados obtidos, ficou evidente que o resolvedor proposto e implementado possui desempenho similar ou superior a longo prazo em relação aos demais considerados nos testes comparativos. Em comparação com o Urânia a implementação demonstra-se competitiva a nível de produto.

Com base nos conceitos de usabilidade levados em consideração, conclui-se que a interface é amigável ao usuário final.

8.2 Trabalhos Futuros

Ao final da implementação, o resolvedor ficou com muitos parâmetros. Desta forma, é relativamente difícil garantir que os conjunto de parâmetros sirva para qualquer instância. Isto dá margem para um trabalho futuro sobre *Tuning* de Parâmetros.

Novas vizinhanças, restrições e movimentos podem ser desenvolvidos e incorporados facilmente ao resolvedor para análise de desempenho.

O problema real requer duas funcionalidades que não puderam ser implementadas nesse trabalho, mas podem ser incluídas no futuro: Substituição de professores e Dicas.

REFERÊNCIAS

AL-YAKOOB, S. M.; SHERALI, H. D. A mixed-integer programming approach to a class timetabling problem: a case study with gender policies and traffic considerations. **European Journal of Operational Research**, [S.l.], v.180, n.3, p.1028 – 1044, 2007.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. **ACM Comput. Surv.**, New York, NY, USA, v.35, n.3, p.268–308, 2003.

BOLAND, N.; HUGHES, B. D.; MERLOT, L. T.; STUCKEY, P. J. New integer linear programming approaches for course timetabling. **Computers & Operations Research**, [S.l.], v.35, n.7, p.2209 – 2233, 2008. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

BRANDãO, E. **Publicidade on-line, ergonomia e usabilidade**: o efeito de seis tipos de banner no processo humano de visualização do formato do anúncio na tela do computador e de lembrança da sua mensagem. [S.l.]: Pontifícia Universidade Católica do Rio de Janeiro, 2006.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. Metaheuristics for High School Timetabling. **Computational Optimization and Applications**, [S.l.], v.9, p.275–298, 1998. 10.1023/A:1018354324992.

DORNELES Árton; ARAúJO, O. C. B. de; SANTOS, H. G. H.O.P.E. - Horários Otimizados para Escolas. In: I JORNADA NACIONAL DA PRODUçãO CIENTíFICA EM EDUCAçãO PROFISSIONAL E TECNOLóGICA, 2006, Brasília, DF. **Anais...** SETEC/MEC, 2006.

GEHA. **Urânia**. 2010.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers & Operations Research**, [S.l.], v.13, n.5, p.533 – 549, 1986.

GOTLIEB, C. The construction of class-teacher timetables. In: IFIP, 1963, Amsterdam. **Anais...** North-Holland, 1963. p.73–77.

GRUBER, M. **Exact and Heuristic Approaches for Solving the Bounded Diameter Minimum Spanning Tree Problem**. 2009. Tese (Doutorado) — University of Vienna.

Adaptation in natural and artificial systems. [S.l.]: University of Michigan Press, 1975.

IBM. **Standard Widget Toolkit**. 2010.

KIRKPATRICK, S. Optimization by simulated annealing: quantitative studies. **Journal of Statistical Physics**, [S.l.], v.34, p.975–986, 1984. 10.1007/BF01009452.

MIRHASSANI, S. A computational approach to enhancing course timetabling with integer programming. **Applied Mathematics and Computation**, [S.l.], v.175, n.1, p.814 – 822, 2006.

RIBEIRO FILHO, L. L. **An Integer Programming Model for the School Timetabling Problem**. 2006.

SCHAERF, A. A Survey of Automated Timetabling. **Artificial Intelligence Review**, [S.l.], v.13, p.87–127, 1999. 10.1023/A:1006576209967.

Designing Interfaces: patterns for effective interaction design. [S.l.]: O'Reilly, 2005.

APÊNDICE A CONVENÇÃO DE DESENVOLVIMENTO EM C++

- Codificação UTF-8 com Identação de 3 espaços sem utilizar TAB
- Nomes de classes utilizam a convenção *CamelCase* com a primeira letra em maiúscula.
- Nomes de variáveis, métodos e diretórios utilizam a convenção *CamelCase* com a primeira letra em minúscula.
- Documentação no formato do *Doxygen* na declaração de classes, métodos e variáveis.
- Constantes em maiúsculas. Nomes compostos são separados por *underline*. Ex.: `MAX_COUNT`
- Definições (`#define`) seguem a mesma regra de constantes
- A abertura da primeira chave ({) de qualquer bloco de comando nunca é solitária.
- Nomes de variáveis membro da classe são prefixadas por m_. Ex.: `m_id`
- Não são colocados códigos dentro do .h que não sejam de definição de interface.
- Todo arquivo .h é protegido por guardiões. Ex.: Uma classe chamada *Convert* tem um guardião chamado `__CONVERT_H__`
- Todo arquivo possui um .h e um .cpp (Exceto para classes abstratas sem implementação)
- O nome do .h é sempre igual o do .cpp
- O nomes dos arquivos são iguais aos nomes das classes/*namespaces* que correspondem.
- O arquivo com a função main se chama main.cpp

APÊNDICE B OPÇÕES DO NEOSOLVER

```
Usage: neosolver [options]
To invoke the menu, type:
neosolver --menu
The options are:
 -i, --instance       <STRING>        Arquivo xml com a instancia
 -s, --seed           <INT>           Semente do resolvedor (Padrão: 0=Aleatória)
 -T, --maxTime        <INT>           Limite de tempo em segundos (Padrão: Infinito)
 -I, --maxIterations  <INT>           Limite de iterações (Padrão: Infinito)
 -c, --cache          <BOOL>          Habilita o uso de cache no Tabu (Padrão: Habilitado)
 -v, --verbose        <INT>           Ajusta o nível de exibição de mensagens (Padrão: desabilitado)
 -t, --tenure         <INT ARRAY>     Tabu Tenure (min,max) (Padrão: 10,30)
     --diInterval     <INT>           Intervalo de diversificação (Padrão: 4000 iterações)
     --diDuration     <INT>           Duração da diversificação (Padrão: 200 iterações)
     --diIntensity    <DOUBLE>        Intensidade de uma modificação feita pela diversificação
                                      (Padrão: 0.1 )
     --neighborhood   <STRING>        Tipo de vizinhança. BlockChange. (Padrão: BlockChange)
     --builder        <STRING>        Tipo de algoritmo construtivo. Aceita Rand ou Smart.
                                      (Padrão: Smart)
     --tabuList       <STRING>        Tipo de Lista Tabu. Aceita ContractPeriod ou BlockContractPeriod
                                      (Padrão: BlockContractPeriod)
     --constrTeacherClash <DOUBLE>
                                      Peso dos conflitos de professor(0=Desabilitada)
     --constrClassClash <DOUBLE>
                                      Peso dos conflitos de turma (0=Desabilitada)
     --constrResourceClash <DOUBLE>
                                      Peso dos conflitos de recurso (0=Desabilitada)
     --constrPeriodHoles <DOUBLE>
                                      Peso das janelas entre períodos do professor (0=Desabilitada)
     --constrAntiHoles <DOUBLE>
                                      Peso das aulas contíguas entre turnos
                                      do professor (0=Desabilitada)
     --constrShiftHoles <DOUBLE>
                                      Peso das janelas entre turnos do professor (0=Desabilitada)
     --constrDayHoles <DOUBLE>        Peso das janelas entre dias do professor (0=Desabilitada)
     --constrDistributionApproximate <DOUBLE>
                                      Peso da distribuição aproximada  (0=Desabilitada)
     --constrDistributionExact <DOUBLE>
                                      Peso da distribuição exata (0=Desabilitada)
     --constrDistributionHole <DOUBLE>
                                      Peso das janelas entre contratos (0=Desabilitada)
     --constrAllocationRequest <DOUBLE>
                                      Peso das aulas indesejadas (0=Desabilitada)
     --constrMaxWorkShiftsByDay <DOUBLE>
                                      Peso do limite de turnos por dia (0=Desabilitada)
     --constrMaxLessonsByDay <DOUBLE>
                                      Peso do limite de aulas por dia (0=Desabilitada)
     --constrMaxWorkDays <DOUBLE>
                                      Peso do limite de dias por semana (0=Desabilitada)
     --constrSubjectGroups <DOUBLE>
                                      Peso das aulas diárias de disciplina com
                                      mesmo grupo (0=Desabilitada)
     --constrDistributionHoleSoft <INT>
                                      Indica se as janelas entre disciplinas deve ser consideradas
                                      soft (0=Desabilitada)
     --constrSyncClash <DOUBLE>       Peso dos conflitos de sincronia de  horários (0=Desabilitada)
```

APÊNDICE C DISTRIBUIÇÕES DO URÂNIA

Grupo 1:
Cada turma só poderá ter, no máximo,
uma aula por dia com o professor...

TIPO A : ... (sem mais restrições)
TIPO B : ... além de um intervalo mínimo de um dia entre duas aulas
TIPO @: ... sendo que uma das aulas terá um dia de intervalo das demais aulas

Grupo 2: Cada turma poderá ter, no máximo,
um dia da semana com duas aulas do professor, sendo que, se isso acontecer...

TIPO C : ... tentar geminar essas aulas
TIPO D : ... geminar essas aulas, obrigatoriamente
TIPO E : ... tentar separar essas aulas
TIPO F : ... separar essas aulas, obrigatoriamente

Grupo 3 : Cada turma poderá ter até duas aulas por dia com o professor,
sendo que, se isso acontecer ...

TIPO G : ... tentar geminar essas aulas
TIPO H : ... geminar essas aulas, obrigatoriamente
TIPO I : ... tentar separar essas aulas
TIPO J : ... separar essas aulas, obrigatoriamente

Grupo 4 : Cada turma terá obrigatoriamente um par de
aulas geminadas com o professor. Nos demais dias...

TIPO K : ... terá no máximo uma aula
TIPO L : ... se caírem duas aulas, tentar geminá-las
TIPO M : ... se caírem duas aulas, geminá-las obrigatoriamente

Grupo 5 : Cada turma terá suas aulas, com o professor, agrupadas ...

TIPO N : ... em pares de aulas, geminadas
TIPO Y: ... em pares de aulas, geminadas, com intervalos mínimos de um dia
TIPO O: ... em trios de aulas, geminadas
TIPO P: ... em quartetos de aulas, geminadas
TIPO Z: ... em quintetos de aulas, geminadas
TIPO $: ... em sextetos de aulas, geminadas

Grupo 6 : Cada turma poderá ter até três aulas por dia,
com o professor, sendo que, se isso acontecer ...

TIPO Q : ... tentar geminar essas aulas
TIPO R : ... geminar essas aulas, obrigatoriamente
TIPO # : ... geminar duas aulas, e separar a terceira aula
TIPO S: ... tentar separar essas aulas
TIPO T : ... separar essas aulas, obrigatoriamente

Grupo 7 : Cada turma poderá ter várias aulas por dia,
com o professor, sendo que, se isso acontecer ...

TIPO U : ... tentar geminar essas aulas
TIPO V : ... geminar essas aulas, obrigatoriamente
TIPO W : ... tentar separar essas aulas
TIPO X : ... separar essas aulas, obrigatoriamente

APÊNDICE D DISTRIBUIÇÕES DO HOP-2010

O HOP-2010 possui quatro tipos de distribuições:

Distribuição com Limite diário: define o número máximo de aulas que um contrato pode ter podia. Por exemplo, para um contrato com 3 aulas é possível estabelecer as seguintes distribuições: **^1** (No máximo 1 aula por dia), **^2** (No máximo 2 aulas por dia) ou **^3** (No máximo 3 aulas por dia)

Distribuição Sugerida: define uma distribuição que **preferencialmente** deve ser atendida pelo resolvedor. Para um contrato com 3 aulas tem-se os seguintes formatos de distribuição: **(1 1 1)** (uma aula em cada dia), **(2 1)** (uma aula dupla e outra simples), **(3)** (uma única aula tripla).

Distribuição Obrigatória: define uma distribuição que **obrigatoriamente** deve ser atendida pelo resolvedor. O formato é semelhante ao da distribuição sugerida. Alguns exemplos: **(1+1+1)** (uma aula em cada dia), **(2+1)** (uma aula dupla e outra simples), **(3)** (uma única aula tripla).

Distribuição Sugerida com Limite diário: define uma distribuição sugerida com um limite de aulas por dia. Exemplo: **(1 1 1)^2** (preferencialmente uma aula em cada dia mas obrigatoriamente respeitando o limite diário de 2 aulas).

APÊNDICE E INSTÂNCIAS

E.1 Maneco

Turnos: 5
Turnos: 1
Períodos por turno: 5

Id	Nome	JP	JT	JD	MT	MD	In	Disponibilidade
0	ALAIDES	E	P	P	1	5	P	(xxxxx) (....x) (.....) (.....) (.....)
1	ANA MARIA	E	P	P	1	3	P	(.....) (xxxxx) (xxxxx) (.....) (.....)
2	ANA MARLI	E	P	P	1	3	P	(.....) (.....) (xxxxx) (.....) (xxxxx)
3	ANA PAULA	E	P	P	1	3	P	(.....) (.....) (.....) (xxxxx) (xxxxx)
4	BATISTELA	E	P	P	1	1	P	(.....) (xxxxx) (xxxxx) (xxxxx) (xxxxx)
5	CARMEM	E	P	P	1	4	P	(xxxxx) (.....) (.....) (.....) (.....)
6	EDERVAL	E	P	P	1	3	P	(xxxxx) (xxxxx) (.....) (.....) (.....)
7	ELISA	E	P	P	1	3	P	(.....) (xxxxx) (.....) (.....) (xxxxx)
8	ELIZETE	E	P	P	1	5	P	(.....) (.....) (.....) (.....) (.....)
9	ELVIO	E	P	P	1	3	P	(.....) (xxxxx) (.....) (.....) (xxxxx)
10	GIANA	E	P	P	1	2	P	(xxxxx) (xxxxx) (xxxxx) (.....) (.....)
11	GILMAR	E	P	P	1	2	P	(xxxxx) (xxxxx) (xxxxx) (.....) (.....)
12	GISELE	E	P	P	1	3	P	(.....) (.....) (.....) (xxxxx) (xxxxx)
13	JOSELAINE	E	P	P	1	3	P	(xxxxx) (.....) (.....) (xxxxx) (.....)
14	KATIA	E	P	P	1	3	P	(.....) (xxxxx) (xxxxx) (.....) (.....)
15	LIBIA	E	P	P	1	4	P	(xxxxx) (.....) (.....) (.....) (.....)
16	Lucia	E	P	P	1	3	P	(xxxxx) (.....) (.....) (.....) (xxxxx)
17	Lucimara	E	P	P	1	2	P	(.....) (.....) (xxxxx) (xxxxx) (xxxxx)
18	MARIA HELENA	E	P	P	1	4	P	(xxxxx) (.....) (.....) (.....) (.....)
19	MARIA INEZ	E	P	P	1	5	P	(.....) (.....) (.....) (.....) (.xxxx)
20	MARIA LEDA	E	P	P	1	3	P	(xxxxx) (xxxxx) (.....) (.....) (.....)
21	MARION	E	P	P	1	4	P	(xxxxx) (.....) (.....) (.....) (.....)
22	MARLENE	E	P	P	1	3	P	(.....) (.....) (xxxxx) (.....) (xxxxx)
23	MIRIAN	E	P	P	1	3	P	(xxxxx) (.....) (xxxxx) (.....) (.....)
24	PAULO J	E	P	P	1	4	P	(xxxxx) (.....) (.....) (.....) (.....)
25	PAULO REIS	E	P	P	1	3	P	(.....) (.....) (.....) (xxxxx) (xxxxx)
26	REGINA	E	P	P	1	3	P	(xxxxx) (xxxxx) (.....) (.....) (.....)
27	RODRIGO	E	P	P	1	4	P	(.....) (.....) (.....) (xxxxx) (.....)
28	ROGERIA	E	P	P	1	2	P	(.....) (xxxxx) (xxxxx) (xxxxx) (.....)
29	RONE	E	P	P	1	4	P	(.....) (.....) (.....) (.....) (xxxxx)
30	ROSANGELA	E	P	P	1	2	P	(xxxxx) (.....) (xxxxx) (.....) (xxxxx)
31	SABRINA	E	P	P	1	3	P	(xxxxx) (xxxxx) (....x) (....x) (....x)
32	SUELI	E	P	P	1	4	P	(.....) (xxx..) (xxxxx) (.....) (.....)
33	SUZANA	E	P	P	1	3	P	(.....) (.....) (.....) (xxxxx) (xxxxx)
34	TALES	E	P	P	1	4	P	(.....) (.....) (.....) (xxxxx) (.....)
35	TATIANA	E	P	P	1	3	P	(.....) (xxxxx) (....x) (xxxxx) (.....)
36	TAVARES	E	P	P	1	2	P	(.....) (.....) (xxxxx) (xxxxx) (xxxxx)
37	VIVIANE	E	P	P	1	2	P	(.....) (.....) (xxxxx) (xxxxx) (xxxxx)

Tabela E.1: Professores(38)

Id	Nome	Abreviatura	G	Disponibilidade
0	BIO	BIO		(.....) (.....) (.....) (.....) (.....)
1	FIL	FIL		(.....) (.....) (.....) (.....) (.....)
2	FIS	FIS		(.....) (.....) (.....) (.....) (.....)
3	GEO	GEO		(.....) (.....) (.....) (.....) (.....)
4	HIS	HIS		(.....) (.....) (.....) (.....) (.....)
5	LEM	LEM		(.....) (.....) (.....) (.....) (.....)
6	LIT	LIT		(.....) (.....) (.....) (.....) (.....)
7	MATEMÁTICA	MAT		(.....) (.....) (.....) (.....) (.....)
8	POR	POR		(.....) (.....) (.....) (.....) (.....)
9	QUI	QUI		(.....) (.....) (.....) (.....) (.....)

Tabela E.2: Disciplinas(10)

Id	Nome	Sinal de intervalo	Disponibilidade
0	2A	Após 3M	(.....) (.....) (.....) (.....) (.....)
1	2B	Após 3M	(.....) (.....) (.....) (.....) (.....)
2	2C	Após 3M	(.....) (.....) (.....) (.....) (.....)
3	2D	Após 3M	(.....) (.....) (.....) (.....) (.....)
4	2E	Após 3M	(.....) (.....) (.....) (.....) (.....)
5	2F	Após 3M	(.....) (.....) (.....) (.....) (.....)
6	2G	Após 3M	(.....) (.....) (.....) (.....) (.....)
7	2H	Após 3M	(.....) (.....) (.....) (.....) (.....)
8	2I	Após 3M	(.....) (.....) (.....) (.....) (.....)
9	2J	Após 3M	(.....) (.....) (.....) (.....) (.....)
10	2K	Após 3M	(.....) (.....) (.....) (.....) (.....)
11	2L	Após 3M	(.....) (.....) (.....) (.....) (.....)
12	3A	Após 3M	(.....) (.....) (.....) (.....) (.....)
13	3B	Após 3M	(.....) (.....) (.....) (.....) (.....)
14	3C	Após 3M	(.....) (.....) (.....) (.....) (.....)
15	3D	Após 3M	(.....) (.....) (.....) (.....) (.....)
16	3E	Após 3M	(.....) (.....) (.....) (.....) (.....)
17	3F	Após 3M	(.....) (.....) (.....) (.....) (.....)
18	3G	Após 3M	(.....) (.....) (.....) (.....) (.....)
19	3H	Após 3M	(.....) (.....) (.....) (.....) (.....)
20	3I	Após 3M	(.....) (.....) (.....) (.....) (.....)

Tabela E.3: Turmas(21)

Id	Di	Pr	Tu	AS	Dt	In
0	0	34	0	3	^2	P
1	0	34	2	3	^2	P
2	0	34	3	3	^2	P
3	0	34	4	3	^2	P
4	0	34	5	3	^2	P
5	0	34	1	3	^2	P
6	0	22	13	3	^2	P
7	0	22	14	3	^2	P
8	0	22	15	3	^2	P
9	0	22	12	3	^2	P
10	0	29	7	3	^2	P
11	0	29	8	3	^2	P
12	0	29	9	3	^2	P
13	0	29	10	3	^2	P
14	0	29	11	3	^2	P
15	0	29	6	3	^2	P
16	0	33	17	3	^2	P
17	0	33	18	3	^2	P
18	0	33	19	3	^2	P
19	0	33	20	3	^2	P
20	0	33	16	3	^2	P
21	1	4	20	2	^2	P
22	1	4	19	2	^2	P
23	1	11	10	2	^2	P
24	1	11	11	2	^2	P
25	1	11	9	2	^2	P
26	1	14	13	2	^2	P
27	1	14	14	2	^2	P
28	1	14	15	2	^2	P
29	1	14	16	2	^2	P
30	1	14	17	2	^2	P
31	1	14	18	2	^2	P
32	1	14	12	2	^2	P
33	5	10	18	2	^2	P
34	5	10	19	2	^2	P
35	5	10	20	2	^2	P
36	5	10	17	2	^2	P
37	6	10	20	2	^2	P
38	1	27	1	2	^2	P
39	1	27	2	2	^2	P
40	1	27	3	2	^2	P
41	1	27	4	2	^2	P
42	1	27	5	2	^2	P
43	1	27	6	2	^2	P
44	1	27	7	2	^2	P
45	1	27	8	2	^2	P
46	1	27	0	2	^2	P
47	2	0	13	3	^2	P
48	2	0	14	3	^2	P
49	2	0	15	3	^2	P
50	2	0	16	3	^2	P
51	2	0	17	3	^2	P
52	2	0	12	3	^2	P
53	2	2	1	3	^2	P
54	2	2	2	3	^2	P
55	2	2	3	3	^2	P
56	2	2	4	3	^2	P
57	2	2	0	3	^2	P
58	2	8	6	3	^2	P
59	2	8	7	3	^2	P
60	2	8	8	3	^2	P
61	2	8	9	3	^2	P
62	2	8	10	3	^2	P
63	2	8	11	3	^2	P
64	2	8	5	3	^2	P
65	2	36	19	3	^2	P
66	2	36	20	3	^2	P
67	2	36	18	3	^2	P
68	3	6	8	2	^2	P
69	3	6	9	2	^2	P
70	3	6	10	2	^2	P
71	3	6	11	2	^2	P
72	3	6	7	2	^2	P
73	3	6	20	2	^2	P
74	3	6	19	2	^2	P
75	3	9	1	2	^2	P
76	3	9	2	2	^2	P
77	3	9	3	2	^2	P
78	3	9	4	2	^2	P
79	3	9	5	2	^2	P
80	3	9	6	2	^2	P
81	3	9	0	2	^2	P
82	3	20	13	2	^2	P
83	3	20	14	2	^2	P
84	3	20	15	2	^2	P
85	3	20	16	2	^2	P
86	3	20	17	2	^2	P
87	3	20	18	2	^2	P
88	3	20	12	2	^2	P
89	4	5	13	2	^2	P
90	4	5	14	2	^2	P
91	4	5	15	2	^2	P
92	4	5	16	2	^2	P
93	4	5	17	2	^2	P
94	4	5	18	2	^2	P
95	4	5	19	2	^2	P
96	4	5	12	2	^2	P
97	4	18	1	2	^2	P
98	4	18	2	2	^2	P
99	4	18	3	2	^2	P
100	4	18	4	2	^2	P
101	4	18	5	2	^2	P
102	4	18	6	2	^2	P
103	4	18	7	2	^2	P
104	4	18	0	2	^2	P
105	4	37	9	2	^2	P
106	4	37	10	2	^2	P
107	4	37	11	2	^2	P
108	4	37	8	2	^2	P
109	4	37	20	2	^2	P
110	5	7	1	2	^2	P
111	5	7	2	2	^2	P
112	5	7	3	2	^2	P
113	5	7	4	2	^2	P
114	5	7	5	2	^2	P
115	5	7	0	2	^2	P
116	5	16	7	2	^2	P
117	5	16	8	2	^2	P
118	5	16	9	2	^2	P
119	5	16	10	2	^2	P
120	5	16	11	2	^2	P
121	5	16	6	2	^2	P
122	5	30	13	2	^2	P
123	5	30	14	2	^2	P
124	5	30	15	2	^2	P
125	5	30	16	2	^2	P
126	5	30	12	2	^2	P
127	8	3	1	3	^2	P
128	8	3	2	3	^2	P
129	8	3	3	3	^2	P
130	8	3	4	3	^2	P
131	8	3	0	3	^2	P
132	8	12	13	3	^2	P
133	8	12	14	3	^2	P
134	8	12	15	3	^2	P
135	8	12	16	3	^2	P
136	8	12	12	3	^2	P
137	8	26	6	3	^2	P
138	8	26	7	3	^2	P
139	8	26	8	3	^2	P
140	8	26	5	3	^2	P
141	8	31	18	3	^2	P
142	8	31	19	3	^2	P
143	8	31	20	3	^2	P
144	8	31	17	3	^2	P
145	8	32	10	3	^2	P
146	8	32	11	3	^2	P
147	8	32	9	3	^2	P
148	6	15	1	2	^2	P
149	6	15	2	2	^2	P
150	6	15	3	2	^2	P
151	6	15	4	2	^2	P
152	6	15	5	2	^2	P
153	6	15	6	2	^2	P
154	6	15	7	2	^2	P
155	6	15	0	2	^2	P
156	6	24	13	2	^2	P
157	6	24	14	2	^2	P
158	6	24	15	2	^2	P
159	6	24	16	2	^2	P
160	6	24	17	2	^2	P
161	6	24	18	2	^2	P
162	6	24	19	2	^2	P
163	6	24	12	2	^2	P
164	6	32	9	2	^2	P
165	6	32	10	2	^2	P
166	6	32	11	2	^2	P
167	6	32	8	2	^2	P
168	7	13	1	3	^2	P
169	7	13	2	3	^2	P
170	7	13	3	3	^2	P
171	7	13	4	3	^2	P
172	7	13	0	3	^2	P
173	7	19	6	3	^2	P
174	7	19	7	3	^2	P
175	7	19	8	3	^2	P
176	7	19	9	3	^2	P
177	7	19	10	3	^2	P
178	7	19	11	3	^2	P
179	7	19	5	3	^2	P
180	7	25	17	3	^2	P
181	7	25	18	3	^2	P
182	7	25	19	3	^2	P
183	7	25	20	3	^2	P
184	7	25	16	3	^2	P
185	7	35	13	3	^2	P
186	7	35	14	3	^2	P
187	7	35	15	3	^2	P
188	7	35	12	3	^2	P
189	9	1	1	3	^2	P
190	9	1	2	3	^2	P
191	9	1	3	3	^2	P
192	9	1	0	3	^2	P
193	9	17	13	3	^2	P
194	9	17	14	3	^2	P
195	9	17	12	3	^2	P
196	9	21	16	3	^2	P
197	9	21	17	3	^2	P
198	9	21	18	3	^2	P
199	9	21	19	3	^2	P
200	9	21	20	3	^2	P
201	9	21	15	3	^2	P
202	9	23	5	3	^2	P
203	9	23	6	3	^2	P
204	9	23	7	3	^2	P
205	9	23	8	3	^2	P
206	9	23	4	3	^2	P
207	9	28	10	3	^2	P
208	9	28	11	3	^2	P
209	9	28	9	3	^2	P

Tabela E.4: Contratos (210)

E.2 Bilac

Turnos: 5
Turnos: 1
Períodos por turno: 5

Id	Nome	JP	JT	JD	MT	MD	In	Disponibilidade
0	Adriane	E	P	P	1	0	P	(.....) (.....) (.....) (.....) (...xx)
1	Andreia	E	P	P	1	0	P	(.....) (.....) (.....) (.....) (xxxxx)
2	Antônio	E	P	P	1	0	P	(.....) (.....) (.....) (.....) (.....)
3	Carla	E	P	P	1	0	P	(xxxxx) (..xxx) (xxxxx) (xxxxx) (xxxxx)
4	Elisângela	E	P	P	1	0	P	(.....) (.....) (.....) (.....) (.....)
5	Horto	E	P	P	1	0	P	(.....) (xxxxx) (.....) (xxxxx) (xxxxx)
6	Kátia	E	P	P	1	0	P	(.....) (.....) (xxxxx) (.....) (xxxxx)
7	Laura	E	P	P	1	0	P	(xxxx.) (xxx..) (xxxxx) (xxxxx) (xxxxx)
8	Luciana	E	P	P	1	0	P	(xxxxx) (i....) (i....) (i....) (i....)
9	Maria de Fátima	E	P	P	1	0	P	(x....) (xxxxx) (.....) (.....) (xxxxx)
10	Marisa	P	P	P	1	0	P	(xxxxx) (..x..) (..x..) (xxxxx) (xxxxx)
11	Marlete	E	P	P	1	0	P	(.....) (xxxxx) (.....) (.....) (xxxxx)
12	Pedro	E	P	P	1	0	P	(xxxxx) (.....) (.....) (.....) (.....)
13	Rozângela	E	P	P	1	0	P	(.....) (xxxxx) (.....) (.....) (.....)
14	Sandra	E	P	P	1	0	P	(.....) (.....) (.....) (xxxxx) (.....)
15	Silvana	E	P	P	1	0	P	(xxxxx) (.....) (.....) (.....) (.....)
16	Simone	E	P	P	1	0	P	(xxxxx) (.....) (.....) (.....) (.....)
17	Simone2	E	P	P	1	0	P	(.....) (.....) (xxxxx) (xxxxx) (xxxxx)
18	Solânge	E	P	P	1	0	P	(.....) (.....) (.....) (xxxxx) (.....)

Tabela E.5: Professores(19)

Id	Nome	Abreviatura	G	Disponibilidade
0	Ciências	Ciê		(.....) (.....) (.....) (.....) (.....)
1	Ed. Artística	EA		(.....) (.....) (.....) (.....) (.....)
2	Geografia	Geo		(.....) (.....) (.....) (.....) (.....)
3	História	His		(.....) (.....) (.....) (.....) (.....)
4	Lingua Ingles	LI		(.....) (.....) (.....) (.....) (.....)
5	Matemática	Mat		(.....) (.....) (.....) (.....) (.....)
6	Português	Por		(.....) (.....) (.....) (.....) (.....)
7	RH	RH		(.....) (.....) (.....) (.....) (.....)

Tabela E.6: Disciplinas(8)

Id	Nome	Sinal de intervalo	Disponibilidade
0	71	Após 3M	(.....) (.....) (...xx) (.....) (.....)
1	72	Após 3M	(.....) (.....) (...xx) (.....) (.....)
2	73	Após 3M	(.....) (.....) (...xx) (.....) (.....)
3	74	Após 3M	(.....) (.....) (...xx) (.....) (.....)
4	81	Após 3M	(.....) (.....) (...xx) (.....) (.....)
5	82	Após 3M	(.....) (.....) (...xx) (.....) (.....)
6	83	Após 3M	(.....) (.....) (...xx) (.....) (.....)
7	84	Após 3M	(.....) (.....) (...xx) (.....) (.....)
8	85	Após 3M	(.....) (.....) (...xx) (.....) (.....)

Tabela E.7: Turmas(9)

Id	Di	Pr	Tu	Re	AS	Dist	In
0	0	10	0		3	(2 1)^2	P
1	0	10	1		3	(2 1)^2	P
2	0	17	2		3	(2 1)^2	P
3	0	7	3		3	(2 1)^2	P
4	0	14	4		3	(1 1 1)^2	P
5	0	14	5		3	(1 1 1)^2	P
6	0	14	6		3	(1 1 1)^2	P
7	0	14	7		3	(1 1 1)^2	P
8	0	14	8		3	(1 1 1)^2	P
9	1	13	0		2	(1 1)^2	P
10	1	13	1		2	(1 1)^2	P
11	1	13	2		2	(1 1)^2	P
12	1	13	3		2	(1 1)^2	P
13	1	9	4		2	(1 1)^2	P
14	1	9	5		2	(1 1)^2	P
15	1	9	6		2	(1 1)^2	P
16	1	9	7		2	(1 1)^2	P
17	1	9	8		2	(1 1)^2	P
18	2	8	1		3	1+1+1	P
19	2	8	2		3	1+1+1	P
20	2	8	3		3	1+1+1	P
21	2	8	0		3	1+1+1	P
22	3	11	0		3	2+1	E
23	3	11	1		3	2+1	E
24	3	11	2		3	2+1	E
25	3	11	3		3	2+1	E
26	3	12	4		3	2+1	E
27	3	12	5		3	2+1	E
28	3	12	6		3	2+1	E
29	3	12	7		3	2+1	E
30	3	12	8		3	2+1	E
31	4	18	1		2	(1 1)^2	P
32	4	18	2		2	(1 1)^2	P
33	4	18	0		2	(1 1)^2	P
34	4	18	3		2	(1 1)^2	P
35	4	18	4		2	(1 1)^2	P
36	4	18	5		2	(1 1)^2	P
37	4	18	6		2	(1 1)^2	P
38	4	18	7		2	(1 1)^2	P
39	4	3	8		2	^2	P
40	5	1	4		4	2+1+1	P
41	5	1	5		4	2+1+1	P
42	5	1	6		4	2+1+1	P
43	5	1	7		4	2+1+1	P
44	5	4	8		4	(2 2)^2	P
45	6	16	0		5	(2 2 1)^2	P
46	6	16	1		5	(2 2 1)^2	P
47	6	15	2		5	2+2+1	P
48	6	15	3		5	2+2+1	P
49	6	2	4		5	2+2+1	E
50	6	2	5		5	2+2+1	E
51	6	2	6		5	2+2+1	E
52	6	2	7		5	2+2+1	E
53	6	5	8		5	^3	P
54	7	9	0		1	^2	P
55	7	9	1		1	^2	P
56	7	13	2		1	^2	P
57	7	13	3		1	^2	P
58	7	13	4		1	^2	P
59	7	13	5		1	^2	P
60	7	13	6		1	^2	P
61	7	13	7		1	^2	P
62	7	13	8		1	^2	P
63	2	0	4		3	^2	P
64	2	0	5		3	^2	P
65	2	0	6		3	^2	P
66	2	0	7		3	^2	P
67	2	0	8		3	^2	P
68	5	4	0		4	^2	P
69	5	4	1		4	^2	P
70	5	6	2		4	2+2	P
71	5	6	3		4	2+2	P

Tabela E.8: Contratos (72)

E.3 CTISM

Turnos: 5
Turnos: 1
Períodos por turno: 4

Id	Nome	JP	JT	JD	MT	MD	In	Disponibilidade
0	Abílio	E	P	P	1	5	P	(....) (....) (....) (....) (....)
1	Adriane	E	P	P	1	5	P	(....) (....) (xxxx) (....) (xxxx)
2	Aita	E	P	P	1	5	P	(....) (....) (....) (....) (xxxx)
3	Alessandro	E	P	P	1	5	P	(xxxx) (....) (....) (....) (xxxx)
4	Alysson	E	P	P	1	5	P	(....) (....) (....) (xxxx) (....)
5	Augusto	E	P	P	1	5	P	(xxxx) (xxxx) (xxxx) (..xx) (.xxx)
6	Bólico	E	P	P	1	5	P	(....) (....) (....) (....) (....)
7	Cauduro	E	P	P	1	5	P	(....) (....) (....) (....) (....)
8	Cláudio	E	P	P	1	5	P	(xxxx) (....) (..xx) (xx..) (....)
9	Colusso	E	P	P	1	5	P	(....) (....) (....) (....) (....)
10	Eduardo	E	P	P	1	5	P	(....) (....) (....) (....) (xxxx)
11	Elida	E	P	P	1	5	P	(....) (....) (....) (....) (....)
12	Erika	E	P	P	1	5	P	(....) (....) (....) (....) (....)
13	Fernando	E	P	P	1	5	P	(....) (....) (....) (....) (....)
14	Iran	E	P	P	1	5	P	(....) (....) (....) (....) (....)
15	Ivan	E	P	P	1	5	P	(xxxx) (....) (....) (....) (xxxx)
16	Josiane	E	P	P	1	5	P	(....) (....) (xxxx) (....) (xxxx)
17	Juzelia	E	P	P	1	5	P	(..xx) (xxxx) (..xx) (xxxx) (xxxx)
18	Losekan	E	P	P	1	5	P	(....) (....) (....) (....) (....)
19	Luciano	E	P	P	1	5	P	(....) (....) (....) (....) (....)
20	Marco	E	P	P	1	5	P	(....) (....) (....) (....) (....)
21	Mauricio	E	P	P	1	5	P	(....) (....) (....) (xxxx) (....)
22	Mauro	E	P	P	1	5	P	(....) (....) (....) (....) (....)
23	Moacir	E	P	P	1	5	P	(....) (....) (....) (....) (....)
24	Nara	E	P	P	1	5	P	(xxxx) (xxxx) (xxxx) (....) (xxxx)
25	Neverton	E	P	P	1	5	P	(....) (....) (....) (xxxx) (xxxx)
26	Nirvan	E	P	P	1	5	P	(....) (....) (....) (....) (xxxx)
27	Olinto	E	P	P	1	5	P	(....) (....) (....) (....) (....)
28	Paula	E	P	P	1	5	P	(....) (....) (....) (....) (....)
29	Paulo	E	P	P	1	5	P	(xxxx) (....) (....) (....) (xxxx)
30	Renor	E	P	P	1	5	P	(xxxx) (..xx) (xxxx) (xxxx) (xxxx)
31	Retzlaz	E	P	P	1	5	P	(....) (....) (....) (....) (....)
32	Rodrigo	E	P	P	1	5	P	(xxxx) (xxxx) (....) (....) (xxxx)
33	Rosi	E	P	P	1	5	P	(....) (....) (....) (xxxx) (xxxx)
34	Roth	E	P	P	1	5	P	(....) (....) (....) (....) (....)
35	Saigon	E	P	P	1	5	P	(....) (....) (xxxx) (xxxx) (xxxx)
36	Saul	E	P	P	1	5	P	(....) (....) (....) (xxxx) (....)
37	Segurança	E	P	P	1	5	P	(....) (....) (....) (....) (....)
38	Susana	E	P	P	1	5	P	(...x) (...x) (...x) (...x) (...x)
39	Suzete	E	P	P	1	5	P	(..xx) (..xx) (..xx) (..xx) (..xx)
40	Tânia	E	P	P	1	5	P	(xxxx) (xxxx) (....) (....) (xxxx)
41	Vizzotto	E	P	P	1	5	P	(....) (....) (....) (....) (xxxx)
42	Zancan	E	P	P	1	5	P	(....) (....) (....) (....) (xxxx)

Tabela E.9: Professores(43)

Id	Nome	Abreviatura	G	Disponibilidade
0	Acionamentos de Circuitos Elétricos	ACE		(....) (....) (....) (....) (....)
1	Acionamentos Hidráulicos e Pneumáticos	AHP		(....) (....) (....) (....) (....)
2	Administração e Organização do Trabalho	AOT		(....) (....) (....) (....) (....)
3	Automação Aplicada	AAP		(....) (....) (....) (....) (....)
4	Automação de Processos	APR		(....) (....) (....) (....) (....)
5	Automação e Controle	AUT		(....) (....) (....) (....) (....)
6	Automação Industrial	AIN		(....) (....) (....) (....) (....)
7	Automação Industrial e Ajustagem	AINeAJU		(....) (....) (....) (....) (....)
8	Automação Industrial e Processos de Usinagem	AINePRU		(....) (....) (....) (....) (....)
9	Biologia	Biol		(....) (....) (....) (....) (....)
10	CNC e Robótica	CRO		(....) (....) (....) (....) (....)
11	Comandos Numéricos Computadorizados	CNC		(....) (....) (....) (....) (....)
12	Controle e Programação Industrial e Soldagem	CPIeSD		(....) (....) (....) (....) (....)
13	Desenho Assistido por Computador	DAC		(....) (....) (....) (....) (....)
14	Desenho Técnico	DT		(....) (....) (....) (....) (....)
15	Desenho Técnico Mecânico	DTM		(....) (....) (....) (....) (....)
16	Educação Artística	EDA		(....) (....) (....) (....) (....)
17	Educação Física	EDF		(....) (....) (....) (....) (....)
18	Elementos de Máquinas	EM		(....) (....) (....) (....) (....)
19	Eletrotécnica	ELET		(....) (....) (....) (....) (....)
20	Eletrônica Industrial	ELI		(....) (....) (....) (....) (....)
21	Ensaios de Materiais	ENM		(....) (....) (....) (....) (....)
22	Espanhol	Espa		(....) (....) (....) (....) (....)
23	Ferramentas e Elementos de Máquinas	FEM		(....) (....) (....) (....) (....)
24	Filosofia	Filo		(....) (....) (....) (....) (....)
25	Física	Físi		(....) (....) (....) (....) (....)
26	Geradores e Motores Elétricos	GME		(....) (....) (....) (....) (....)
27	Gestão Ambiental	GEA		(....) (....) (....) (....) (....)
28	Gestão da Qualidade	GQU		(....) (....) (....) (....) (....)
29	Gestão e Qualidade Industrial	GQI		(....) (....) (....) (....) (....)
30	Hidráulica e Pneumática	HP		(....) (....) (....) (....) (....)
31	Higiene Ocupacional	HOC		(....) (....) (....) (....) (....)
32	História	HIST		(....) (....) (....) (....) (....)
33	Informática e Segurança do Trabalho	IST		(....) (....) (....) (....) (....)
34	Inglês	ING		(....) (....) (....) (....) (....)
35	Instalacoes Eletricas eProcessos de Fabricação	IELePFA		(....) (....) (....) (....) (....)
36	Instalações e Manutenção Elétrica	IME		(....) (....) (....) (....) (....)
37	Instalações Elétricas	IEL		(....) (....) (....) (....) (....)
38	Lingua Portuguesa	Ptg		(....) (....) (....) (....) (....)
39	Literatura	Lite		(....) (....) (....) (....) (....)
40	Manutenção Eletromecânica	MAN		(....) (....) (....) (....) (....)
41	Matemática	MAT		(....) (....) (....) (....) (....)
42	Medicina do Trabalho	MET		(....) (....) (....) (....) (....)
43	Metrologia	MEF		(....) (....) (....) (....) (....)
44	Máquinas Elétricas e Transformadores	MTR		(....) (....) (....) (....) (....)
45	Máquinas Térmicas	MTE		(....) (....) (....) (....) (....)
46	Organizações e Normas	ON		(....) (....) (....) (....) (....)
47	Planejamento Industrial	PIN		(....) (....) (....) (....) (....)
48	Pneumática	PNM		(....) (....) (....) (....) (....)
49	Prevenção e Combate a Sinistros	PCS		(....) (....) (....) (....) (....)
50	Produção Mecânica	PRM		(....) (....) (....) (....) (....)
51	Projetos e Instalações Elétricas	PIE		(....) (....) (....) (....) (....)
52	Química	Quím		(....) (....) (....) (....) (....)
53	Redacao Técnica	RT		(....) (....) (....) (....) (....)
54	Redes Industriais	RIN		(....) (....) (....) (....) (....)
55	Relações Humanas	RHU		(....) (....) (....) (....) (....)
56	Resistência dos Materiais	RMA		(....) (....) (....) (....) (....)
57	Segurança do Trabalho	STR		(....) (....) (....) (....) (....)
58	Sistemas de Usinagem	SUS		(....) (....) (....) (....) (....)
59	Sistemas Hidráulicos e Pneumáticos	SHP		(....) (....) (....) (....) (....)
60	Soldagem e Ajustagem	SLDeAJU		(....) (....) (....) (....) (....)
61	Tecnologia Mecânica	TEM		(....) (....) (....) (....) (....)
62	Tecnologias e Processos Industriais	TPI		(....) (....) (....) (....) (....)
63	Termodinâmica Técnica	TTE		(....) (....) (....) (....) (....)
64	Transformadores	TRA		(....) (....) (....) (....) (....)
65	Tubulações Industriais	TUB		(....) (....) (....) (....) (....)
66	Técnicas de Promoção e Divulgação	TPD		(....) (....) (....) (....) (....)
67	Ténicas de Manutenção Aplicada	TMA		(....) (....) (....) (....) (....)
68	Têcnicas e Planejamento da Manutenção	TPM		(....) (....) (....) (....) (....)

Tabela E.10: Disciplinas(69)

Id	Nome	Sinal de intervalo	Disponibilidade
0	311	Nenhum	(....) (....) (....) (....) (....)
1	318	Nenhum	(....) (....) (....) (....) (....)
2	321	Nenhum	(....) (....) (....) (....) (....)
3	322	Nenhum	(....) (....) (....) (....) (....)
4	325	Nenhum	(....) (....) (....) (....) (....)
5	327	Nenhum	(....) (....) (....) (....) (....)
6	328	Nenhum	(....) (....) (....) (....) (....)
7	342	Nenhum	(....) (....) (....) (....) (....)
8	346	Nenhum	(....) (....) (....) (....) (....)
9	347	Nenhum	(....) (....) (....) (....) (....)

Tabela E.11: Turmas(10)

Id	Nome	Q	Disponibilidade
0	Lab Acionamentos	1	(....) (....) (....) (....) (....)
1	Lab Eletronica	1	(....) (....) (....) (....) (....)
2	Lab Instalações Elétricas	1	(....) (....) (....) (....) (....)
3	Lab de Controles, Ensaios e Máquinas Elétricas	1	(....) (....) (....) (....) (....)
4	Lab de Máquinas e Ferramentas	1	(....) (....) (....) (....) (....)
5	Lab Sistemas Pneumáticos	2	(....) (....) (....) (....) (....)
6	Lab CNC	1	(....) (....) (....) (....) (....)
7	Lab Soldagem	1	(....) (....) (....) (....) (....)
8	Lab de Ajustagem	1	(....) (....) (....) (....) (....)

Tabela E.12: Recursos(9)

Id	Di	Pr	Tu	Re	AS	Dist	In
0	55	39	0		1	^2	P
1	47	34	9		2	2	P
2	20	8	2	1[1]	3	2+1	P
3	42	30	4		2	^2	P
4	26	1	2	3[1]	2	2	P
5	48	13	9	5[1]	2	2	P
6	60	15,22	1	7[1], 8[1]	3	3	P
7	30	13	2		3	3	P
8	17	5	1		1	^1	P
9	51	0	2		2	(2)^2	P
10	49	2	4		2	^2	P
11	15	12	3		2	(2)^2	P
12	64	10	9		2	(2)^2	P
13	43	2	3		2	(2)^2	P
14	45	25	3		2	(2)^2	P
15	18	3	1		1	^1	P
16	61	3	3		2	(2)^2	P
17	62	37	4		2	^2	P
18	28	29	7		2	(2)^2	P
19	10	20	8	6[1]	2	2	P
20	55	39	7		2	^2	P
21	19	10	5		4	4	P
22	34	28	1		2	^2	P
23	29	21	8		2	(2)^2	P
24	67	26	7		2	(2)^2	P
25	66	37	4		2	^2	P
26	13	12	5		2	(2)^2	P
27	54	36	8	1[1]	2	2	P
28	25	27	1		2	(2)^2	P
29	21	6	5		2	(2)^2	P
30	58	23	8	4[1]	2	2	P
31	14	21	1		2	2	P
32	9	16	1		2	(2)^2	P
33	56	6	5		2	(2)^2	P
34	2	34,37	4		2	(2)^2	P
35	65	26	5		2	(2)^2	P
36	16	17	6		2	^2	P
37	22	40	6		2	^2	P
38	24	15	6		1	^1	P
39	32	14	6		2	^2	P
40	38	35	6		1	^1	P
41	39	33	6		1	^1	P
42	41	27	6		1	^1	P
43	44	10	6		2	(2)^2	P
44	52	11	6		2	^2	P
45	19	4	0		7	4+3	P
46	5	42	9	0[1]	4	4	P
47	36	0,9	0	2[1]	4	4	P
48	31	25	4		4	4	P
49	40	26,36	9	3[1]	4	4	P
50	0	4	2	0[1]	4	4	P
51	6	32	2		4	4	P
52	63	25	9		4	4	P
53	1	13	8	5[1]	4	4	P
54	57	29	4		4	4	P
55	50	2,22	3	7[1]	4	4	P
56	3	36	8	1[1]	4	4	P
57	59	31	3	5[1]	4	4	P
58	4	42	8		4	4	P
59	60	2,22	5	7[1], 8[1]	4	4	P
60	8	7,41	7	4[1], 5[1]	4	4	P
61	35	34,42	6	2[1], 4[1]	4	4	P
62	7	19,41	7		4	4	P
63	12	18,22	7	7[1]	4	4	P
64	14	12	0		3	3	P
65	68	26	5		2	2	P
66	27	24	4		2	2	P
67	23	18	0		2	2	P
68	46	0	0		2	2	P
69	11	23	9	6[1]	2	2	P
70	53	38	0		1	(1)^1	P
71	33	12,29	1		2	2	P
72	64	10	2		2	(2)^2	P
73	18	3	3		2	(2)^2	P
74	38	35	1		2	^2	P
75	56	18	3		2	(2)^2	P
76	65	26	7		2	(2)^2	P
77	37	21	5	2[1]	2	2	P
78	39	33	1		1	^1	P
79	41	27	1		2	(2)^2	P
80	17	5	6		1	^1	P
81	25	27	6		1	^1	P

Tabela E.13: Contratos (82)