UNIVERSIDADE FEDERAL DO PAMPA

CIÊNCIA DA COMPUTAÇÃO

GEANCARLO SALDANHA MAYDANA

GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS PARA O CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIPAMPA

Trabalho de Conclusão de Curso

Alegrete

2011

GEANCARLO SALDANHA MAYDANA

GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS PARA O CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIPAMPA

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Orientador: Prof. Dr. Cleo Zanella Billa

Alegrete

2011

GEANCARLO SALDANHA MAYDANA

GERAÇÃO AUTOMÁTICA DE QUADROS DE HORÁRIOS PARA O CURSO DE CIÊNCIA DA COMPUTAÇÃO DA UNIPAMPA

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Trabalho apresentado e aprovado em: 03 de Janeiro de 2012.
Banca Examinadora:

Prof. Dr. Cleo Zanella Billa
Orientador
Ciência da Computação - UNIPAMPA

Prof. Dr. José Carlos Bins Filho
Ciência da Computação - UNIPAMPA

Prof. Dr. Fabio Natanael Kepler
Ciência da Computação - UNIPAMPA

RESUMO

Este trabalho apresenta uma solução para a geração de horários do curso de Ciência da Computação da Universidade Federal do Pampa. O problema da geração de quadros de horários, conhecido na literatura como *timetabling*, vem sendo estudado pela comunidade acadêmica desde a década de 60. Basicamente, consiste em arranjar encontros entre professores e alunos em um determinado tempo, tipicamente uma semana, em um local, de modo que satisfaça um conjunto amplo de restrições que variam de acordo com o tipo de instituição a qual o problema é aplicado. Vários métodos de IA (Inteligência Artificial) têm sido aplicados para resolver problemas de quadros de horários, dentre os quais podemos citar *Simulated Annealing*, Busca Tabu e Algoritmos Genéticos. Por ser uma metáfora de fácil entendimento e implementação, algoritmos genéticos foi escolhida para ser utilizada este trabalho. Alguns problemas foram encontrados ao implementar os operadores genéticos. Porém, os mesmos foram solucionados e o algoritmo desenvolvido conseguiu encontrar boas soluções e em poucas gerações.

Palavras-chave: *timetabling*, inteligência artificial, algoritmos genéticos.

ABSTRACT

This paper proposes the use of genetic algorithms to find a timetabling solution to the course of Computer Science at UNIPAMPA (Federal University of Pampa). The timetabling problem has been studied by the academic community since the 60's. It consists in scheduling a sequence of lectures in a fixed period of time, satisfying a set of constraints. Several methods of Artificial Intelligence has been used to solve the timetabling problem, like Simulated Annealing, Tabu Search and Genetic Algorithms. For being a metaphor of easy understanding and implementation, we chose to investigate the use of genetic algorithms to solve the timetabling problem.

Keywords: timetabling, artificial intelligence, genetic algorithms.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG – Algoritmos Genéticos

CC – Ciência da Computação

ES – Engenharia de Software

GRASP – Procedimento de busca adaptativa gulosa e randomizada

IA – Inteligência Artificial

SGBD – Sistema de gerenciamento de banco de dados

UML – *Unified Modeling Language*

UNIPAMPA – Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO

O problema da geração de quadros de horários, conhecido na literatura como *timetabling*, em suas várias formulações. Basicamente, consiste em arranjar encontros entre professores e alunos em um determinado tempo, tipicamente uma semana, em um local, de modo que satisfaça um conjunto amplo de restrições que variam de acordo com o tipo de instituição a qual o problema é aplicado.

Um problema de quadro de horários pode ser tratado como um problema de busca. No entanto, também é um problema de otimização, na qual dentre todas as soluções válidas, considera-se a de menor valor da função objetivo (no caso de problema de minimização) ou de maior valor da função objetivo (no caso de problema de maximização), em ambos os casos, para todo o espaço de soluções factíveis.

Em Appleby et al. (1960) foi feita uma das primeiras referências a esse problema. Nesse trabalho, os autores enunciam técnicas para a construção de solução para o problema por meio do emprego de computadores e algoritmos, e delineavam a comparação entre o problema de quadro de horários e outros problemas de agendamento conhecidos na época. Entre esses algoritmos se destacam dois tipos: os métodos exatos e os heurísticos. Os métodos exatos sempre retornam a melhor solução possível, enquanto os heurísticos fazem buscas no conjunto de solução, sem a garantia de uma solução ótima.

Segundo Fogel (1991), existem heurísticas mais gerais que por meio de adaptações podem ser usadas para resolver vários problemas. Elas são chamadas de meta-heurísticas, e entre elas pode-se citar os algoritmos genéticos. Estes métodos não garantem encontrar uma solução ótima para um problema, mas procuram por soluções consideradas de boa qualidade. Segundo Ribeiro (1996), meta-heurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. Contrariamente às heurísticas convencionais, as meta-heurísticas são de caráter geral e têm mais condições de escapar de “ótimos locais”.

Os algoritmos genéticos (AGs) são uma técnica bastante utilizada na resolução de problemas de busca e otimização. Holland (1975) fez um estudo dos processos evolutivos e seleção natural, e apresentou algoritmos genéticos como uma metáfora para os processos evolutivos, de forma que ele pudesse estudar a adaptação e a evolução no mundo real, simulando-a dentro de computadores. Segundo Linden (2006), os algoritmos genéticos são técnicas heurísticas de otimização global. Populações de indivíduos são criadas e submetidas

aos operadores genéticos: seleção, cruzamento (*crossover*) e mutação. Estes operadores utilizam uma caracterização da qualidade de cada indivíduo como solução do problema em questão chamada avaliação e vão gerar um processo de evolução natural destes indivíduos, que eventualmente deverá gerar um indivíduo que caracterizará uma boa solução (talvez até a melhor possível) para o problema.

A formulação de um quadro de horários de qualidade é primordial à administração de qualquer instituição de ensino, seja ela de qualquer nível. O alto número de alunos no ensino superior, por exemplo, torna difícil que o processo seja feito de maneira manual. A solução artesanal a esse tipo de problema também é restringida pelo crescimento das próprias instituições. De fato, formular um quadro de horários demanda tempo e a solução gerada pode não ser satisfatória sob diversos aspectos, como, por exemplo, deixar os alunos ociosos por determinado período do horário letivo e não respeitar as preferências de horários dos docentes para ministrarem suas aulas. Adicionalmente, devido à complexidade do problema e ao amplo espaço de busca, faz-se necessário propor uma solução automatizada para realizar esse processo de geração da grade horária.

O objetivo deste trabalho é desenvolver um algoritmo para gerar quadros de horários baseado na modelagem de *School Timetabling* e usar como estudo de caso o curso de Ciência da Computação da Unipampa. O curso tem aulas de segunda a sábado, quatro períodos por dia. No total, uma turma pode ter no máximo 24 aulas durante a semana. Não necessariamente todas as turmas têm o horário cheio; depende da oferta de disciplinas da turma para o semestre e da carga horária dessas disciplinas.

Muitas técnicas têm sido propostas para a resolução de problemas de *timetabling*. Todavia, nenhuma técnica tem se mostrado melhor ou mais eficiente que outra. Portanto, aplicou-se AG para gerar quadros de horários devido aos resultados satisfatórios da aplicação desta técnica a problemas de otimização e o número de trabalhos relacionados sobre *timetabling* usando algoritmos genéticos.

Este trabalho está disposto da seguinte maneira: o capítulo 2 traz uma revisão da literatura sobre o problema de quadros de horários com suas variantes e formulações, algoritmos genéticos e trabalhos relacionados. No capítulo 3 são descritos a modelagem do problema proposto e a descrição da implementação do AG. No capítulo 4 são apresentados os resultados obtidos com a execução do algoritmo, e finalmente, no capítulo 5 são apresentadas as conclusões.

2 FUNDAMENTOS TEÓRICOS

2.1 Timetabling

O problema da geração de quadros de horários, citado como uma instância do *Timetabling Problem,* consiste em arranjar encontros entre professores e alunos em um determinado tempo, tipicamente uma semana, de modo que satisfaça um conjunto amplo de restrições.

Como é um problema de amplo espectro de aplicação e sua solução é de difícil implementação, desde a década de 60 a comunidade acadêmica tem dado uma atenção considerável ao problema.

Segundo Souza (2000), a solução manual desse problema é árdua e normalmente requer vários dias de trabalho e, mesmo assim, pode não gerar uma boa solução. A elaboração de um quadro de horários por esta via pode demandar semanas de trabalho em uma escola secundária. Em uma universidade, onde temos dezenas de cursos e turmas concomitantes, envolvendo centenas de professores e disciplinas, o problema se acentua ainda mais. Ainda pode haver um caso em que um professor leciona em mais de uma instituição, e em uma solução, pode ter sido alocado em um dia e horário que colide com a outra instituição em que ele trabalha.

Werra (1985) e Schaerf (1999) acreditam que problemas de geração de horários não podem ser completamente automatizados. Há duas justificativas para isso: por um lado, há razões que não podem ser facilmente expressas em um sistema automatizado, que tornam um quadro de horário melhor que o outro. Por outro, uma vez que o espaço de soluções é vasto, a intervenção humana pode conduzir a busca em direção a regiões promissoras, nas quais o sistema, por si só, dificilmente teria condições de chegar.

Um problema de quadro de horários pode ser tratado como um problema de busca. Uma solução para um problema de busca é aquela que satisfaz todas as restrições existentes, ou seja, qualquer solução válida é aceita como solução para o problema. No entanto, o problema de quadro de horários é também um problema de otimização, na qual dentre todas as soluções válidas, considera-se a de menor valor da função objetivo (no caso de problema de minimização) ou de maior valor da função objetivo (no caso de problema de maximização), em ambos os casos, para todo o espaço de soluções factíveis.

Para Werra (1985), o processo da construção de um quadro de horários consiste em duas fases:

- Primeiro: o currículo é definido para cada turma ou para cada grupo de estudantes e cada um tem de especificar os vários recursos (de pessoal ou equipamentos) para as turmas.
- Segundo: quando um acordo sobre a atribuição dos recursos é alcançado, então se tenta achar um horário viável para ser trabalhado, compatível com todos os requisitos previamente definidos.

Muitas variantes desse problema têm sido propostas e diferem umas das outras pelo tipo de Instituição de Ensino envolvida, universidades ou escolas, e pelo tipo de restrições impostas ao problema. Em Schaerf (1999), são citadas três classes de problemas:

- *School Timetabling*: Sequenciamento semanal das aulas de uma escola, evitando que professores e alunos tenham mais de uma aula agendadas para o mesmo período de tempo.
- *Course Timetabling*: Sequenciamento semanal para as aulas de um conjunto de cursos universitários, minimizando as sobreposições de aulas de cursos com estudantes em comum.
- *Examination Timetabling*: Sequenciamento semanal para os exames de um conjunto de cursos universitários, evitando sobreposições de exames de cursos com estudantes em comum.

2.1.1 School Timetabling

Nesta sessão, será descrito em detalhes o *school timetabling*, também conhecido como modelo turma/professor; primeiro uma versão simplificada, a qual pode ser solucionada em tempo polinomial e posteriormente, a formulação básica.

2.1.1.1 Problema simplificado

Seja $c_1, \dots, c_m$ um conjunto de m turmas, $t_1, \dots, t_n$um conjunto de n professores e $1, \dots, p$ um conjunto de p períodos. Dada uma matriz de inteiros não-negativa R_{mxn}chamada matriz de requisitos, onde r_{ij} é o número de aulas dadas pelo professor t_j para a turmac_i, o

problema consiste em arranjar as aulas de forma que nenhum professor ou turma estejam envolvidos em mais de uma aula ao mesmo tempo.

Segue a formulação (SCHAERF, 1999):

Encontrar

$$x_{ijk}(i = 1 \dots m; j = 1 \dots n; k = 1 \dots p)$$

sujeito a

$$\sum_{k=1}^{p} x_{ijk} = r_{ij}(i = 1 \dots m; j = 1 \dots n) \quad (2.1.1)$$

$$\sum_{j=1}^{n} x_{ijk} \leq 1(i = 1 \dots m; k = 1 \dots p) \quad (2.1.2)$$

$$\sum_{i=1}^{m} x_{ijk} \leq 1(j = 1 \dots n; k = 1 \dots p) \quad (2.1.3)$$

$$x_{ijk} = 0 \; ou \; 1 \; (i = 1 \dots m; j = 1 \dots n; k = 1 \dots p) \quad (2.1.4)$$

onde $x_{ijk} = 1$ se a turma c_i e o professor t_j tem um encontro no período *k*, e $x_{ijk} = 0$ caso contrário. A restrição 2.1.1 assegura que cada professor lecione o número correto de aulas para cada turma. As restrições 2.1.2 e 2.1.3 asseguram que cada turma e cada professor estão envolvidos em no máximo uma aula por período. Sempre existe uma solução para este problema, a menos que um professor ou uma turma estejam envolvidos em mais que *p* aulas. Mais precisamente, a solução existe se, e somente se (EVEN et al., 1976):

$$\sum_{i=1}^{m} r_{ij} \leq p(j = 1 \dots n) \quad (2.1.5)$$

$$\sum_{j=1}^{n} r_{ij} \leq p(i = 1 \dots m) \quad (2.1.6)$$

2.1.1.2 Problema de busca básico

O problema anterior não inclui nenhuma restrição a algum possível agendamento das aulas. Em instâncias reais, temos de levar em conta a possibilidade de algum professor (ou turma) estar indisponível em dado momento.

Para isso, podemos formular o *school timetabling* com indisponibilidades de professores e turmas. A formulação a seguir é de Junginger (1986): dadas duas matrizes binárias T_{mxp} e C_{nxp} cada qual $t_{ik} = 1$ (respectivo $c_{jk} = 1$) se o professor t_i (respectiva turma c_j) está disponível no período k, e $t_{ik} = 0$ (respectivo $c_{jk} = 0$) caso contrário. Por conseguinte, trocamos as restrições 2.1.2 e 2.1.3 da formulação anterior pelas restrições 2.1.7 e 2.1.8 abaixo (SCHAERF, 1999):

Encontrar

$$x_{ijk}(i = 1 \dots m; j = 1 \dots n; k = 1 \dots p)$$

Sujeito a

$$\sum_{k=1}^{p} x_{ijk} = r_{ij}(i = 1 \dots m; j = 1 \dots n)$$

$$\sum_{j=1}^{n} x_{ijk} \leq t_{ik}(i = 1 \dots m; k = 1 \dots p) \qquad (2.1.7)$$

$$\sum_{i=1}^{m} x_{ijk} \leq c_{jk}(j = 1 \dots n; k = 1 \dots p) \qquad (2.1.8)$$

$$x_{ijk} = 0 \; ou \; 1 \; (i = 1 \dots m; j = 1 \dots n; k = 1 \dots p)$$

Devem ser consideradas também restrições dadas como pré-associações (WERRA, 1985): uma aula em particular pode ser imposta a ser agendada para determinado período. Essas pré-associações podem ser expressas adicionando um conjunto de restrições da seguinte forma:

$$x_{ijk} \geq p_{ijk}(i = 1 \dots m; j = 1 \dots n; k = 1 \dots p) \qquad (2.1.9)$$

Onde $p_{ijk} = 0$ se não existe uma pré-associação, e $p_{ijk} = 1$ onde uma aula do professor t_j para a turma c_i é pré-associada a ocorrer no período *k*. Este problema formulado acima foi demonstrado como NP-Completo em Even et al. (1976) através de uma redução de 3-SAT (GAREY; JOHNSON, 1979). Even et al. (1976) provou que o problema é polinomial para o caso especial em que as turmas estão sempre disponíveis e cada professor está disponível por exatamente dois períodos.

2.1.1.3 Problema de otimização

O *timetabling* é um problema de busca, onde a solução é qualquer horário viável. Entretanto, em aplicações reais um horário pode ser melhor que outro, e o objetivo é encontrar o ótimo. Essas considerações nos forçam a formular o problema dos horários como um problema de otimização com uma função objetivo a minimizar (ou maximizar).

Podemos adicionar ao problema a seguinte função objetivo (JUNGINGER, 1986):

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} d_{ijk} x_{ijk}$$

onde d_{ijk} é atribuído a períodos *k* a qual uma aula de um professor t_j para a turma c_i é menos desejável.

Em Colorni et al. (1992) pode-se encontrar uma função objetivo mais complexa, que inclui diversos aspectos de um horário. Baseia-se nas seguintes quantizações:

- Custo didático: isto é, espalhar as aulas por toda a semana;
- Custo organizacional: isto é, ter um professor disponível para assumir temporariamente alguma função docente;
- Custo pessoal: isto é, um dia de folga para cada professor.

Uma abordagem diferente é dada em Kaneko (1996). É introduzida uma linguagem de restrições, onde uma penalidade é associada a cada restrição violada. O objetivo é minimizar a penalidade global. As restrições para o problema do quadro de horários podem ser classificadas em restrições “fortes” e “fracas” (CALDEIRA; FERNANDES, 2002). Restrições “fortes” devem ser obrigatoriamente satisfeitas, e violações às mesmas resultam em uma solução inválida. Restrições “fracas” são aquelas desejáveis, mas não obrigatórias. Não

satisfazer uma restrição “fraca” não resulta uma solução inválida, porém afeta a qualidade da solução e pode ser usada como indicador de qualidade da mesma.

Algumas restrições “fortes” são citadas abaixo:

- Professor não pode lecionar em duas turmas ao mesmo tempo;
- Uma turma não pode ter duas aulas simultaneamente.

Como restrições “fracas” podem ser citadas:

- Professor ter um dia de folga durante a semana;
- Respeitar as preferências de horários dos professores.

2.1.1.4 Variações do problema básico

Muitas variantes ao problema básico acima têm sido propostas na literatura de forma a lidar com problemas da vida real. Dentre elas podem ser citadas (SCHAERF, 1999):

- Aulas simultâneas
    - Horários reais usualmente incluem algumas aulas que são ministradas simultaneamente para mais de uma turma. Por exemplo, em algumas escolas, as aulas de educação física geralmente envolvem duas turmas juntas. Obviamente, se uma aula simultânea está agendada em um dado período, todas as turmas envolvidas não podem ser agendadas para outras aulas ao mesmo tempo.
- Salas especiais
    - A disponibilidade das salas não é levada em conta na formulação básica do problema por que é pressuposto que cada turma tenha uma sala dedicada. Entretanto, algumas aulas em particular requerem equipamentos, tais como laboratórios de ciência ou salas de música. O número de salas especiais é obviamente limitado, e, portanto, existe uma restrição adicional não permitindo que mais do que um dado número de aulas que requerem uma sala especial possa ser agendada no mesmo período de tempo.
- Vários professores para a mesma disciplina
    - Algumas disciplinas podem ser ministradas por vários professores, de modo compartilhado, ou seja, cada um se responsabiliza por um conjunto de tópicos inerentes a mesma.

2.1.2 Course Timetabling

O problema de quadros horários de curso consiste em agendar um conjunto de aulas para cada curso dado um número de salas e períodos de tempo. A diferença principal para o *school timetabling* é que cursos universitários podem ter estudantes em comum, já em escolas são conjuntos disjuntos de estudantes. Se dois cursos têm estudantes em comum, então se deve evitar que tenham aulas agendadas para o mesmo período. Além disso, professores de escolas usualmente lecionam para mais de uma turma, enquanto que em universidades, geralmente, um professor leciona apenas uma disciplina.

Adicionalmente, em quadros de horários para universidades, deve-se levar em conta a disponibilidade das salas e seu tamanho.

2.1.2.1 Problema de busca básico

Dados q cursos $K_1, \dots, K_q$, e para cada i, o curso K_i consiste de k_i aulas. Há ainda um conjunto r de currículos $S_1, \dots, S_r$ formado por grupos de cursos com estudantes em comum. Isso significa que cursos em S_l precisam ser agendados em horários diferentes. Tem-se p períodos e l_k é o número máximo de aulas que podem ser agendadas para o período k, ou seja, o número de salas disponíveis para aquele período. Segue-se, então, a seguinte formulação (WERRA, 1985):

Encontrar

$$y_{ik}(i = 1 \dots q; k = 1 \dots p)$$

sujeito a

$$\sum_{k=1}^{p} y_{ik} = k_i(i = 1 \dots q) \tag{2.1.10}$$

$$\sum_{i=1}^{q} y_{ik} \leq l_k(k = 1 \dots p) \tag{2.1.11}$$

$$\sum_{i \in S_t} y_{ik} \leq 1(l = 1 \dots r; k = 1 \dots p) \tag{2.1.12}$$

$$y_{ik} = 0 \; or \; 1 \; (i = 1 \ldots q; k = 1 \ldots p) \qquad (2.1.13)$$

onde $y_{ik} = 1$ se uma aula de um curso K_i está agendada para o período *k*, e $y_{ik} = 0$ caso contrário. A restrição 2.1.11 garante que cada curso tenha um número certo de aulas associadas a ele. A restrição 2.1.12 impede que se tenham mais aulas agendadas para um período *k* do o número de salas disponíveis. A restrição 2.1.13 garante a alocação de uma aula de cada curso do currículo S_i por período *k*.

O problema formulado é NP-Completo, o que pode ser provado através de uma redução ao problema de coloração de grafos. Para mais detalhes, veja Werra (1985).

2.1.2.2 Problema de otimização

Podemos incluir a seguinte função objetivo à formulação básica do *course timetabling* (WERRA, 1985):

$$\max \sum_{i=1}^{q} \sum_{k=1}^{p} d_{ik} y_{ik}$$

onde d_{ik} é a conveniência de haver uma aula do curso K_i no período *k*. Como o problema já é um NP-Completo, pré-associações e indisponibilidades não o tornam mais complexo.

2.1.3 Examination Timetabling

Segundo Schaerf (1999), o problema de quadro de horários para exames consistem em agendar um quadro de exames por curso dentro de um período de tempo. Assim, é semelhante ao problema de quadro de horários de cursos. Existem situações, inclusive, em que ambos se fundem.

Algumas características específicas de um problema de quadro de horários de exames são:

- Há, de modo geral, um exame por curso;
- Em uma sala, pode ocorrer mais de um exame;

- O número de períodos *h* pode variar, ao contrário de um problema de quadro de horários de cursos;
- Trabalham-se restrições, como: a ocorrência de, no máximo, um exame por dia para cada estudante; e/ou não agendar muitos exames consecutivos para o mesmo estudante;
- A um estudante é possível perder uma aula, devido à sobreposição de horários, mas não se pode afirmar o mesmo no que diz respeito a exames.

2.1.3.1 Problema de busca básico

O problema pode ser formulado de maneira similar ao *course timetabling*. Tem-se q cursos $K_1, \dots, K_q$, e um exame para cada curso K_i. Existem r grupos de exames $S_1, \dots, S_r$ formado por grupos de cursos com estudantes em comum. Os cursos pertencentes a S_l não podem ter seus exames agendados para o mesmo período. Tem-se p períodos e l_k é o número máximo de exames que podem ser agendados para o período k. O problema é formulado segundo a expressão a seguir (WERRA, 1985):

Encontrar

$$y_{ik}(i = 1 \dots q; k = 1 \dots p)$$

Sujeito a

$$\sum_{k=1}^{p} y_{ik} = k_i(i = 1 \dots q) \tag{2.1.14}$$

$$\sum_{i=1}^{q} y_{ik} \leq l_k(k = 1 \dots p) \tag{2.1.15}$$

$$\sum_{i \in S_t} y_{ik} \leq 1(l = 1 \dots r; k = 1 \dots p) \tag{2.1.16}$$

$$y_{ik} = 0 \; ou \; 1 \; (i = 1 \dots q; k = 1 \dots p) \tag{2.1.17}$$

onde $y_{ik} = 1$ se o exame do curso K_i está agendada para o período k, e $y_{ik} = 0$ caso contrário.

2.1.3.2 Problema de otimização

Segundo Werra (1985), a restrição mais desejável a um quadro de horários de exames é evitar que um aluno faça dois exames consecutivos. Essa restrição é demonstrada na função a seguir:

$$\sum_{k=1}^{p-1} \sum_{l=1}^{r} \sum_{i,j \in S_t} y_{ik} y_{jk+1}$$

A função acima penaliza os cursos do mesmo grupo Si sempre que tiverem agendados seus exames consecutivamente.

Assim como o *course timetabling*, o problema formulado é NP-Completo, o que pode ser provado através de uma redução ao problema de coloração de grafos. Para mais detalhes, veja Werra (1985).

2.1.4 Técnicas de solução

Na literatura podem ser encontradas muitas técnicas para solucionar problemas de *timetabling*. A maioria se baseia na simulação de métodos humanos para resolver problemas. Soluções assim podem ser tratadas como heurísticas construtivas, e consistem, basicamente em alocar aula a aula, até que todo o quadro de horário esteja completo. A ideia básica dessas abordagens é que os casos "mais restritivos", ou mais importantes, sejam tratados primeiro. A definição para "mais restritivo" dependerá da política de restrições da instituição ou do problema que está sendo modelado.

Em Schaerf (1999) são elencadas algumas técnicas de solução que têm sido propostas para esse problema, como, por exemplo, sua formulação como um problema de programação inteira, como problema de fluxo de redes ou usando coloração de grafos. Mais recentemente, métodos de IA (Inteligência Artificial) começaram a ser aplicados para resolver problemas de quadros de horários, dentre os quais podemos citar *Simulated Annealing*, Busca Tabu, um algoritmo híbrido usando GRASP e Busca Tabu, e Algoritmos Genéticos.

As próximas subseções trazem uma breve descrição do funcionamento de cada método de IA citado acima, e também exemplos de trabalhos que propuseram uma solução para problemas de *school timetabling* usando as mesmas.

2.1.4.1 Simulated Annealing

Simulated annealing é uma técnica de busca local probabilística para encontrar soluções para problemas de otimização. Foi proposta em Gelatt et al. (1983) e o nome da técnica vem de uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos. Conforme Aarts & Korst (1989), a analogia com a otimização (combinatória ou não) é bastante direta. Os estados da matéria são as soluções realizáveis, a quantidade objetiva substitui a energia, os estados metaestáveis da matéria sendo ótimos locais e a estrutura cristalina corresponde ao ótimo global.

O processo inicia com a criação da solução inicial de forma aleatória. O procedimento principal consiste em um loop que gera a cada iteração, de forma randômica, um vizinho da solução atual. Para Dowsland (1990), Δ é a diferença na função objetivo entre a nova solução e a atual. Se $\Delta < 0$, a nova solução é aceita e se torna a solução atual. Se $\Delta \geq \mathbf{0}$, a nova solução é aceita com a probabilidade $\boldsymbol{e^{(-\Delta/T)}}$, onde T é a temperatura e regula a probabilidade de aceitar soluções de pior custo. T decresce gradualmente conforme o progresso do algoritmo, e esse processo é repetido até que T seja tão pequeno que mais nenhum movimento seja aceito. A melhor solução encontrada durante a busca é tomada como uma boa aproximação para a solução ótima. Na figura 2.1, é apresentado, em pseudocódigo, o algoritmo de *simulated annealing*:

```
procedimento SA
1. Seja s0 uma solução inicial, T0 a temperatura inicial, α a taxa de resfriamento e SAmax o
   número máximo de iterações para se atingir o equilíbrio térmico;
2. s ← s0;                {Solução corrente}
3. s' ← s;                {Melhor solução obtida até então}
4. T ← T0;                {Temperatura corrente}
5. IterT ← 0;             {Número de iterações na temperatura T}
6. enquanto (T > 0) faça
7.    enquanto (IterT < SAmax) faça
8.       IterT ← IterT + 1;
9.       Gere um vizinho qualquer s' ∈ N(s);
10.      Δ = f(s') – f(s);
11.      se (Δ < 0)
12.         então
13.            s ← s';
14.            se f(s') < f(s*) então s* ← s';
15.         senão
16.            Tome x ∈ [0,1];
17.            se x < e^(-Δ/T) então s ← s';
18.      fim-se;
19.   fim-enquanto;
20.   T ← α ×T;
21.   IterT ← 0;
22. fim-enquanto;
23. Retorne s*;
fim SA;
```

FIGURA 2.1 – Algoritmo *Simulated Annealing* (SOUZA, 2000).

Para mais detalhes sobre *simulated annealing*, veja Dowsland (1990).

Em Abramson (1991), pode-se encontrar solução para o problema de quadros de horários usando *simulated annealing*. Segundo Abramson (1991), os átomos são representados pelos elementos (professor X turma). A energia do sistema representa o custo do quadro de horários; este custo é usado para refletir a qualidade da solução, assim como a energia reflete a qualidade da substância que está passando pelo processo de recozimento. A temperatura é usada para controlar a probabilidade de um aumento no custo. Podem ser citadas duas condições de parada para o algoritmo (ABRAMSON, 1991): a primeira é quando o custo chega a zero. Nesse caso, um quadro de horários ideal foi encontrado, e não há mais porque o algoritmo continuar sua execução. O segundo critério é se o custo não mudar em um determinado número de iterações.

Em Abramson (1991), o algoritmo de *simulated annealing* foi aplicado a dados fictícios e reais de escolas da Austrália. A função objetivo a ser minimizada é a soma do número dos

conflitos, em cada horário, de turmas e professores. Os resultados foram satisfatórios. Na grande maioria dos casos de teste encontrou-se um quadro de horário com custo final zero, ideal, para o modelo proposto. Há uma comparação entre seis esquemas de resfriamento diferentes no algoritmo de *simulated annealing* aplicado ao *school timetabling*. Um desses esquemas produziu soluções melhores e em tempo computacional menor que os outros. Consiste, basicamente, em uma sequência de reaquecimento, seguido de resfriamento,quando a temperatura atinge certo nível, o que é determinado pelo próprio algoritmo.

Em Khang et al. (2010b), há uma aplicação de *simulated annealing* a casos reais de duas escolas do Vietnã. Foram considerados dois tipos de movimentos:

- Movimento simples, onde um bloco de aulas é movida a uma nova posição, mantendo a viabilidade do quadro de horário;
- Troca, onde ocorre uma troca de posição de dois blocos de aulas;

Os mesmos autores já haviam proposto uma solução usando Busca Tabu em Khanget al. (2010a), mas o algoritmo com *simulated annealing* se mostrou mais eficaz que o anteriormente proposto para todas as instâncias de teste.

Outro trabalho é o encontrado em Colorni et al. (1998). O autor propôs uma modificação no algoritmo de *simulated annealing*. Assim que o quadro de horários ficar congelado, ou seja, quando o algoritmo não for capaz de melhorar o custo atual, a temperatura é ajustada para seu valor inicial. O processo de recozimento começa novamente a partir do quadro de horários previamente congelado. A condição de parada é o tempo de execução. O autor afirma que enquanto *simulated annealing* foi muito eficiente em "recuperar" a viabilidade de quadros de horários inviáveis, já para otimizar um quadro de horários viável não foi muito efetivo. Esta afirmação se mostrou verdadeira nos casos em que os custos de cada inviabilidade eram tanto altos como baixos.

2.1.4.2 Busca Tabu

O método de Busca Tabu, proposto por Glover (1997), é um procedimento iterativo para a solução de problemas de otimização combinatória que aceita movimentos de piora para tentar escapar de ótimos locais distantes de um ótimo global. Na forma clássica, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho s' de um subconjunto V da vizinhança $N(s)$ da solução atual s. Se $f(s')$ for melhor ou pior que $f(s)$, s' sempre será a nova solução atual. Apenas esse mecanismo não é suficiente para escapar de ótimos locais,

uma vez que pode haver retorno a uma solução previamente gerada. Para evitar isso, o algoritmo usa o conceito de lista tabu.

Esta lista define todos os movimentos que têm certo atributo como sendo tabu por um determinado número de iterações, conhecido como tempo tabu. Tais movimentos são proibidos a menos que a solução satisfaça a certo critério de aspiração *A*, em geral que essa solução seja melhor que a melhor solução encontrada até então. Os atributos são escolhidos para prevenir o retorno a soluções visitadas recentemente e são escolhidos por características que são fáceis para detectar.

O algoritmo da Busca Tabu é apresentado, em pseudocódigo, na figura 2.2:

procedimento *BT*
1. Seja s_0 solução inicial;
2. $s^* \leftarrow s$; {Melhor solução obtida até então}
3. Iter $\leftarrow 0$; {Contador do número de iterações}
4. MelhorIter $\leftarrow 0$; {Iteração mais recente que forneceu s^*}
5. Seja BTmax o número máximo de iterações sem melhora em s^*;
6. T $\leftarrow \varnothing$; {Lista Tabu}
7. Inicialize a função de aspiração A;
8. enquanto (Iter – MelhorIter ≤ BTmax) faça
9. Iter $\leftarrow$ Iter + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin T$) ou s' atenda a condição de aspiração ($f(s') < A(f(s))$);
11. T $\leftarrow$ T – {movimento mais antigo} + {movimento que gerou s'};
12. $s \leftarrow s'$;
13. se $f(s) < f(s^*)$ então
14. $s^* \leftarrow s$;
15. MelhorIter $\leftarrow$ Iter ;
16. fim-se;
17. Atualize a função de aspiração A;
18. fim-enquanto;
19. Retorne s^*;

fim *BT*;

FIGURA 2.2 – Algoritmo Busca Tabu (SOUZA, 2000).

Detalhes adicionais sobre Busca Tabu podem ser encontrados em Glover (1997).

Em Schaerf (1996) podemos encontrar uma aplicação de Busca Tabu ao problema de quadros de horários. O autor propôs representar a solução usando uma matriz onde as linhas representam os professores e as colunas, os horários. Cada elemento M_{ij} da matriz contém o nome da turma a qual o professor *i* está lecionando no período *j*. O autor também incluiu soluções inválidas no espaço de busca. Para explorar este espaço, dois tipos de movimentos

foram implementados. No primeiro, trocam-se duas aulas distintas de um dado professor. Este tipo de movimento em um quadro de horário viável gera um quadro inviável. Para isso, o segundo tipo de movimento tenta "reparar" essa inviabilidade ou pelo menos uma das inviabilidades geradas.

O proposto em Costa (1994) é muito similar ao proposto em Schaerf (1996). Em Costa (1994) foi implementado outro tipo de movimento. Neste, permite-se somente a mudança de uma única aula para um horário diferente. Nessa representação, um professor pode lecionar mais de uma aula ao mesmo tempo.

Em Hertz (1991) há outra solução para a geração de horários usando Busca Tabu. Uma peculiaridade dessa proposta é a decisão de não levar em conta as restrições fracas, ou seja, qualquer solução sempre satisfará essas, e tentar então diminuir a ocorrência de restrições fortes. O teste do método em duas instituições mostrou que soluções consideravelmente melhores do que as soluções manuais podem ser obtidas.

2.1.4.3 GRASP e Busca Tabu

O GRASP (Procedimento de busca adaptativa gulosa e randomizada) é um método iterativo proposto em Feo & Rezende (1992). Consiste em duas fases:

- Fase de construção: a solução é gerada, elemento a elemento;
- Fase de busca local: um ótimo local na vizinhança da solução construída é pesquisado.

A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado. Na fase de construção do algoritmo, uma solução viável é construída iterativamente, um elemento por vez, até que a solução esteja completa. Os elementos que compõem a solução são ordenados em uma lista, chamada lista de candidatos. Esta lista é ordenada por uma função gulosa que mede o benefício que o mais recente elemento escolhido concede à solução já construída. Um subconjunto denominado lista restrita de candidatos (LRC) é formado pelos melhores elementos que compõem a lista de candidatos, o tamanho da lista restrita de candidatos é controlado por um parâmetro $\alpha \in [0,1]$, onde para $\alpha = 0$ tem-se um comportamento puramente guloso do algoritmo e para $\alpha = 1$ um comportamento aleatório. A componente probabilística do método é devida à

escolha aleatória de um elemento da lista restrita de candidatos. Este procedimento permite que diferentes soluções de boa qualidade sejam geradas.

O pseudocódigo para o algoritmo GRASP é apresentado na figura 2.3:

FIGURA 2.3 – Algoritmo GRASP (SOUZA, 2000).

Para mais detalhes sobre GRASP, veja Feo & Rezende (1992).

O algoritmo híbrido usando GRASP e Busca Tabu procura explorar as características positivas de cada meta-heurística. Pode-se citar Costa et al. (2002).

Em Costa et al. (2002), adotou-se a seguinte estratégia:

- Gerar a solução inicial de forma heurística (não aleatória), através do procedimento GRASP. A geração de soluções aleatórias diversifica o espaço de soluções; no entanto, uma solução aleatória é, em geral, de baixa qualidade, exigindo um tempo de processamento muito elevado para ser melhorada.
- Refinar essa solução usando o procedimento de Busca Tabu.

O pseudocódigo do algoritmo é apresentado na figura 2.4:

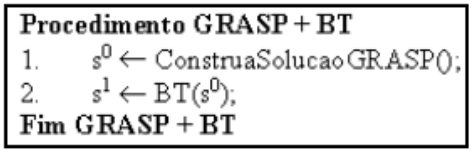

FIGURA 2.4 – Algoritmo híbrido GRASP + Busca Tabu (COSTA et al., 2002)

Segundo Costa et al. (2002), sem o refinamento pela Busca Tabu, o algoritmo tem enorme dificuldade em encontrar soluções viáveis, e as soluções encontradas são de menor qualidade que aquelas que usam o refinamento pela Busca Tabu. O autor destaca que as soluções finais encontradas utilizando este algoritmo híbrido foram de boa qualidade e em tempo computacional viável.

2.1.4.4 Algoritmos Genéticos

Segundo Davis (1991) e Michaelewizs (1994), Algoritmos Genéticos é uma técnica para solucionar problemas de otimização. Diferentemente das técnicas apresentadas anteriormente, AG não é baseado numa busca local. Para Goldberg (1989), Algoritmos Genéticos são algoritmos de busca baseados nos mecanismos da seleção natural e da genética natural. Eles combinam a sobrevivência da estrutura mais adaptada com a troca aleatória das informações genéticas para formar um novo espaço de busca.

Segundo Costa et al. (2002), a geração de horários é considerada um problema de decisão multicritério, porque para determinar a qualidade de uma programação faz-se necessário considerar diferentes objetivos (custos). Com isso, a utilização do Algoritmo Genético traz uma vantagem considerável, uma vez que um conjunto de indivíduos candidatos à solução do problema pode representar em um mesmo instante diversas regiões do espaço de busca.

A solução proposta neste trabalho utiliza Algoritmos Genéticos, devido à adaptabilidade ao problema e o grande número de trabalhos correlatos, e são descritos na seção a seguir.

2.2 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) foram desenvolvidos por John Holland em Holland (1975) ao explorar os processos adaptativos de sistemas naturais e suas possíveis aplicabilidades em projetos de softwares de sistemas artificiais. Holland acreditava que, incorporando os princípios da genética e da seleção natural em um programa de computador, pudesse resolver problemas bastante complexos, assim como a natureza o fazia: produzindo

cegamente organismos complexos para resolver o problema de sua sobrevivência. Segundo Goldberg (1989), Holland (1975) teve duas metas principais:

- Abstrair e rigorosamente explicar os sistemas adaptativos naturais;
- Projetar softwares de sistemas artificiais que assegurassem mecanismos importantes de seleção natural.

Segundo Linden (2006), os algoritmos evolucionários usam modelos computacionais dos processos naturais de evolução como ferramenta para resolver problemas. Apesar de haver vários modelos propostos, todos eles têm em comum o conceito de simulação da evolução das espécies através de seleção, mutação e reprodução, processos estes que dependem do "desempenho" dos indivíduos desta espécie dentro do "ambiente". Estes algoritmos são extremamente dependentes de fatores probabilísticos, tanto na fase de inicialização da população quanto na fase de evolução.

Segundo Tanomaru (1995) Algoritmos Genéticos são métodos computacionais de busca baseados no mecanismo de evolução natural e na genética. Em AG, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.

Segundo Ribeiro Filho (2000), Algoritmos Genéticos são baseados num processo coletivo de aprendizagem dentro de uma população de indivíduos, cada um dos quais representando um ponto no espaço de busca de soluções para um dado problema.

Em Michaelewizs (1994) é descrito de maneira sucinta, em pseudocódigo, um AG básico:

Algoritmo 2.1 Algoritmo Genético Básico. (MICHAELEWIZS, 1994).

Início
    $t \leftarrow 0$;
    inicializa$P(t)$;
    avalia$P(t)$;
    enquanto não alcançar o critério de parada
        $t \leftarrow t + 1$;
        selecione$P(t)$de $P(t-1)$;
        altere$P(t)$;

avalia$P(t)$;

fimEnquanto

Fim

Segundo Barcellos (2000), se tudo ocorrer bem, esta simulação do processo evolutivo irá produzir, à medida que as gerações forem se sucedendo, cromossomos cada vez mais bem adaptados ao problema proposto.

Os AG diferem da maioria dos algoritmos convencionais de otimização, pois estes últimos podem ficar "presos" a máximos locais, sem considerar todo o espaço de busca. Os AG exploram o espaço de busca utilizando diversos pontos em paralelo, diminuindo assim a probabilidade da busca ficar presa a ótimos locais.

Segundo Michaelewizs (1994), um algoritmo genético, ou qualquer algoritmo evolucionário, necessita ter os seguintes cinco componentes:

- Uma representação genética para soluções potenciais ao problema;
- Uma forma de criar uma população inicial de soluções potenciais;
- Uma função de avaliação que mede a adequabilidade daquela solução ao problema;
- Operadores genéticos que modificam a composição dos filhos;
- Valores para os vários parâmetros que o algoritmo genético usa (tamanho da população, probabilidades de aplicar os operadores genéticos).

De acordo com Patnaik & Srinivas (1994), os Algoritmos Genéticos tem se constituído ferramentas poderosas para resolver problemas onde o espaço de busca é muito grande e os métodos convencionais se mostram ineficientes.

Segundo Linden (2006), Algoritmos Genéticos são uma técnica de busca extremamente eficiente no seu objetivo de varrer o espaço de soluções e encontrar soluções próximas da solução ótima, quase sem necessitar interferência humana. Um ponto contra AG é que eles não são tão bons assim em termos de tempo de processamento. Logo, eles são mais adequados em problemas especialmente difíceis, entre os quais estão incluídos os NP-difíceis.

2.2.1 Terminologia básica

Segundo Goldberg (1989), a teoria da evolução prediz que o meio ambiente seleciona, a cada geração, os seres mais aptos a ele. Durante a existência dos indivíduos, fenômenos como mutação e cruzamento atuam sobre o material genético armazenado nos cromossomos. Estes processos promovem a variabilidade genética dos seres vivos na população. Em AG, os processos expressam-se como metáforas a estes.

A tabela 2.1 mostra essa comparação entre os termos usados em biologia e seu respectivo em AG:

TABELA 2.1

Comparação entre biologia e algoritmos genéticos

Termo	Biologia	AGs
Cromossomo	É o conjunto completo de genes de um organismo	Estrutura de dados que codifica uma solução para um problema
Gen ou Gene	Unidade de hereditariedade que é transmitida pelo cromossomo e que controla as características do organismo	Parâmetro codificado no cromossomo, ou seja, um elemento ou variável da estrutura que representa o cromossomo
Fitness	Valor que indica o grau de aptidão do indivíduo	Fornece para cada indivíduo a medida de quão próximo a uma solução considerada satisfatória ele se encontra
Indivíduo	Um membro da população	Um indivíduo é formado pelo cromossomo (conjunto de genes) e por um valor de *fitness*
Genótipo	Composição genética contida no genoma	Informação contida no cromossomo
Fenótipo	Características visíveis, como cor dos olhos, dos cabelos e outras características físicas	Objeto, estrutura ou organismo constituído a partir das informações do genótipo representando a codificação do cromossomo

2.2.2 Características dos AG

Segundo Linden (2006), AG são técnicas probabilísticas, e não técnicas determinísticas. Assim sendo, um AG com a mesma população inicial e os mesmos parâmetros, pode encontrar soluções diferentes a cada execução. AG trabalham com uma população, sendo uma heurística de busca no espaço de soluções.

Além disso, segundo Linden (2006), AG diferenciam-se dos demais esquemas aleatórios por ser uma busca que utiliza informação pertinente ao problema e não trabalham com caminhadas aleatórias (*random walk*) pelo espaço de soluções, mas sim direcionando sua busca através do mecanismo da seleção, equivalente ao processo natural. Apesar de determinar o conjunto de pontos a serem percorridos de forma aleatória, os algoritmos genéticos não podem ser chamados de buscas aleatórias não-direcionadas, pois exploram informações históricas para encontrar novos pontos de busca. Esta exploração é feita através do uso do valor de avaliação de cada cromossomo como guia na escolha dos elementos reprodutores.

Ainda segundo Linden (2006), algoritmos genéticos é uma técnica de busca com as seguintes características positivas:

- Paralela: pois mantém uma população de soluções que são avaliadas simultaneamente;
- Global: algoritmos genéticos não usam apenas informação local, logo, não necessariamente ficam presos em máximos locais;
- Não totalmente aleatória: diferentemente de métodos que usam somente variáveis aleatórias para realizar sua pesquisa, algoritmos genéticos tem componentes aleatórios, mas usam a informação da população corrente para determinar o próximo estado da busca.

2.2.3 Representação dos cromossomos

Segundo Linden (2006), a representação dos cromossomos é fundamental para um algoritmo genético. Consiste em uma maneira de traduzir a informação do problema em uma maneira viável de ser tratada pelo computador. Quanto mais adequada ao problema, maior a qualidade dos resultados obtidos. É interessante que algumas regras sejam seguidas:

- A representação deve ser a mais simples possível;

- Se houver soluções proibidas ao problema, então elas não devem ter uma representação;
- Se o problema impuser condições de algum tipo, estas devem estar implícitas dentro da representação.

Em Holland (1975), foi adotada a representação binária, onde cada cromossomo é uma sequência de *bits* (e cada gene é somente um *bit*). Por ser simples, essa representação é amplamente adotada por pesquisadores da área de algoritmos genéticos.

A figura 2.5 apresenta um exemplo para um cromossomo binário:

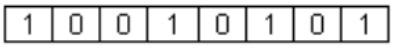

FIGURA 2.5 – Representação de um cromossomo binário.

2.2.4 Função de avaliação

Segundo Barcellos (2000), a função de avaliação tem o papel de ligar o algoritmo genético ao problema propriamente dito. Ela é usada para avaliar o cromossomo para posterior uso pelos operadores de reprodução. Segundo Linden (2006), a função de avaliação mede a qualidade de um cromossomo como solução do problema em questão. É a forma de diferenciar as boas das más soluções para um problema.

A função de avaliação deve embutir todo conhecimento sobre o problema a ser resolvido, tanto as restrições quanto os objetivos de qualidade. No caso do problema do quadro de horários, a função deve avaliar todas as restrições “fortes” e “fracas” da modelagem proposta.

Segundo Linden (2006), a função de avaliação deve refletir os objetivos a serem alcançados na resolução de um problema. Contextualizando com o problema proposto neste trabalho, para cada restrição que não é respeitada, a solução recebe uma penalização na sua função de avaliação. Portanto, uma solução ótima é aquela que tem $avaliação = 0$.

2.2.5 Operadores genéticos

Segundo Ferreira (2003) a reprodução é uma etapa inspirada na natureza e tem por objetivo criar novos indivíduos. São usados operadores similares aos da reprodução humana, como seleção, cruzamento (*crossover*) e mutação.

O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações. Os operadores genéticos são muito importantes para que a população se diversifique e mantenha características da adaptação das gerações anteriores.

Segundo Tanomaru (1995), se não houver o processo de seleção, os AG, além de perderem grande parte do caráter evolutivo, seriam processos ineficientes similares a buscas aleatórias. Sem cruzamentos, haveria somente busca aleatória a partir dos melhores elementos da população. Finalmente, sem mutação, os AG efetuariam busca usando somente a informação contida na população, e não teriam como repor material genético perdido durante o processo.

2.2.5.1 Seleção

Segundo Barcellos (2000), o processo de seleção desempenha o papel da seleção natural na evolução, selecionando para sobreviver e reproduzir os organismos melhor adaptados ao meio, no caso, os cromossomos com melhor valor na função de avaliação. Os cromossomos melhor adaptados terão, necessariamente, uma probabilidade maior de sobrevivência e reprodução que os de baixa função de avaliação.

Em Goldberg (1989) é descrito um modo de realizar a seleção. É chamado de Método da Roleta, onde cada cromossomo recebe um valor, e esse valor é uma proporção entre o seu valor de avaliação e a soma das avaliações da população. Isso faz com que indivíduos com maior valor de avaliação, tenham maior chance de serem escolhidos.

O método da roleta é exemplificado de maneira simples em Linden (2006). O primeiro passo é aplicar a função de avaliação na população. A figura 2.6 mostra um exemplo de uma população com quatro cromossomos e suas respectivas avaliações:

Indivíduo	*Avaliação*
0001	1
0011	9
0100	16
0110	36
Total	*62*

FIGURA 2.6 – Avaliação da população (LINDEN, 2006).

Os próximos passos são calcular a proporção de cada indivíduo na roleta, dividindo o seu valor de avaliação pelo valor de avaliação total da população, e rodar a roleta. Montando a roleta para a população da figura 2.6, obtemos os seguintes dados na figura 2.7:

Indivíduo	*Avaliação*	*Pedaço da roleta (%)*	*Pedaço da roleta (°)*
0001	1	1.61	5.8
0011	9	14.51	52.2
0100	16	25.81	92.9
0110	36	58.07	209.1
Total	*62*	*100.00*	*360.0*

FIGURA 2.7 – Avaliação dos indivíduos e suas respectivas proporções na roleta, em porcentagem e em graus. (LINDEN, 2006).

Segundo Linden (2006), o ato de rodar a roleta deve ser aleatório, escolhendo um número entre 0 e 100 (representando a porcentagem de cada indivíduo) ou entre 0 e 360 (representando uma posição na roleta) ou ainda entre 0 e a soma total das avaliações (representando um pedaço do somatório). Para o exemplo da figura 2.8, o indivíduo “0110” recebe um pedaço igual a 58% da roleta, e suas chances de ser sorteado são muito maiores que as do indivíduo “0001”, que recebe 1,6% da roleta. Assim, podemos concluir que os indivíduos mais fortes, ou mais adaptados, têm preferência para a reprodução, mas os indivíduos menos adaptados ainda possuem alguma chance – assim como na natureza.

Outro método de seleção é o proposto em Wetzel (1983). Chama-se de Seleção por Torneio, e funciona de maneira semelhante a um torneio tradicional. Basicamente ocorre um sorteio de *n* indivíduos pertencentes à população e vence o torneio aquele que tiver o melhor

valor de avaliação. Todos os indivíduos vencedores de seus respectivos torneios serão selecionados pelo método.

Tomando como exemplo a população da figura 2.6, se sortearmos os indivíduos “0011” e “0100” e realizarmos um torneio entre os dois, o indivíduo “0100” é o vencedor, pois tem melhor avaliação que o “0011”, e é selecionado pelo método.

Diferentemente do método da roleta, onde os indivíduos mais adaptado têm mais chances de serem selecionados, no método por torneio todos os indivíduos têm a mesma chance de serem sorteados para um torneio.

Como estes métodos são aleatórios, os melhores indivíduos da população podem não ser sorteados na roleta ou para disputar um torneio. Para evitar que isso aconteça, pode-se sempre manter o melhor ou os n melhores indivíduos da geração atual na geração seguinte, garantindo assim, que estes cromossomos não sejam destruídos nas etapas de recombinação e mutação. Este processo é chamado de Elitismo.

2.2.5.2 Cruzamento (*crossover*)

O cruzamento é utilizado após a seleção. Segundo Tanomaru (1995), o cruzamento é um processo que emula o fenômeno de *crossover*, a troca de informações entre pares de cromossomos. Em sua forma mais simples, trata-se de um procedimento aleatório que ocorre com uma probabilidade P_c que deve ser especificada pelo usuário/programador.

As três formas mais comuns são o cruzamento em um ponto, o cruzamento em dois pontos e o cruzamento uniforme, que serão detalhados a seguir.

Cruzamento em um ponto. Segundo Lobo (2005), seleciona-se aleatoriamente um ponto de corte do cromossomo, e a partir desse ponto, a troca de material genético é realizada. Cada um dos dois descendentes recebe a informação genética de cada um dos pais.

A figura 2.8 ilustra esse processo:

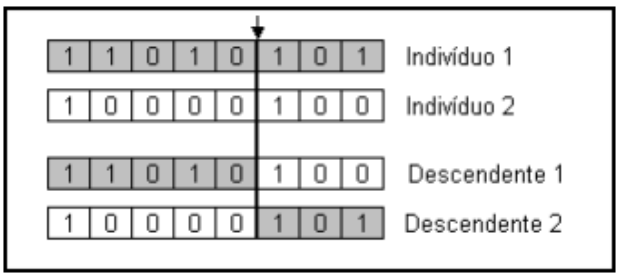

FIGURA 2.8 – Cruzamento em um ponto (LOBO, 2005).

No exemplo da figura 2.8, o ponto de corte se localiza após o quinto gene. Antes do ponto de corte, os descendentes 1 e 2 mantém os genes idênticos aos dos indivíduos 1 e 2. Após o ponto de corte, o descendente 1 recebe os genes do indivíduo 2, e o descendente 2 recebe os genes do indivíduo 1.

Cruzamento em dois pontos. Segundo Lobo (2005), no cruzamento em dois pontos seleciona-se aleatoriamente dois pontos de corte do cromossomo e um dos descendentes fica com a parte central de um dos pais e as externas do outro pai e vice-versa.

A figura 2.9 ilustra o método:

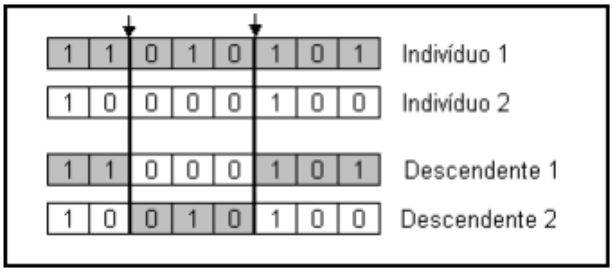

FIGURA 2.9 – Cruzamento em dois pontos (LOBO, 2005).

No exemplo da figura 2.9, o descendente 1 manteve as partes externas do indivíduo 1 e recebeu os genes da parte interna do indivíduo 2. Com o descendente 2, o contrário.

Cruzamento uniforme. Segundo Lobo (2005), o cruzamento uniforme é significantemente diferente dos dois cruzamentos apresentados anteriormente. No cruzamento uniforme, cria-se aleatoriamente uma máscara de cruzamento. A máscara contém valores os 0 ou 1. Se um certo bit da máscara for 1, o gene correspondente será copiado do primeiro pai; se um certo bit da máscara de cruzamento for 0, será copiado do segundo pai.

Conforme ilustrado na figura 2.10, o processo é repetido com os pais trocados para produzir o segundo descendente.

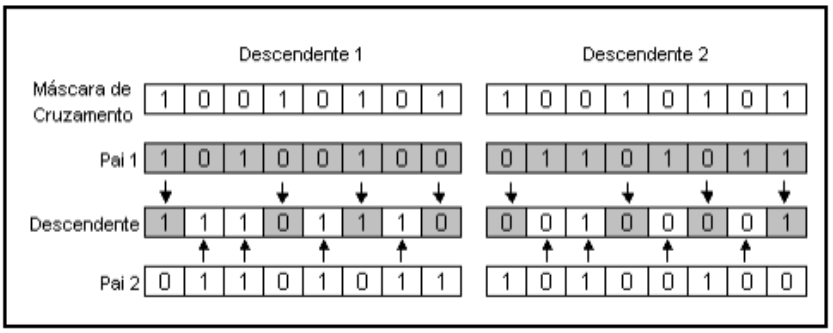

FIGURA 2.10 – Cruzamento uniforme. (LOBO, 2005).

2.2.5.3 Mutação

Segundo Goldberg (1989), a mutação garante a variabilidade em um algoritmo genético. O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade ocorrência de mutação em um gene é denominada taxa de mutação. É extremamente importante por que não deixa o AG restrito aos indivíduos gerados pelo cruzamento genético.

Em Goldberg (1989) são descritos os principais métodos de mutação:

- Mutação aleatória (*Flip Mutation*): cada gene a ser mutado recebe um valor sorteado do alfabeto válido;
- Mutação por troca (*Swap Mutation*): são sorteados *n* pares de genes, e os elementos do par trocam os valores desses genes entre si;
- Mutação *Creep*: um valor aleatório é somado ou subtraído do valor do gene;

2.2.6 Trabalhos relacionados

Em Timóteo (2002) pode-se encontrar um aplicação de AG ao problema de *timetabling*. O trabalho foi baseado em um problema real, o do Colégio Nossa Senhora de Lourdes, de Lavras-MG. Sua implementação não considerou janelas, ou buracos, no horário das turmas, e aulas isoladas. Entretanto, tratou restrições "fracas" como preferência dos professores por determinados horários, média de aulas por dia e eliminação de blocos de disciplinas. O trabalho traz muitas comparações entre as configurações do algoritmo, já que o mesmo implementou vários métodos de criação da população inicial, seleção, mutação e *crossover*.

Ao contrário de Timóteo (2002), o foco de Ciscon (2006) foi a implementação de um AG que eliminasse janelas e aulas isoladas. Algumas restrições como média de aulas por dia e número de aulas por disciplina já são tratadas na geração da população inicial. Para testar o modelo proposto, usaram-se situações de turmas reais da Escola Municipal Álvaro Botelho, de Lavras-MG. Segundo Ciscon (2006), em comparação com horários gerados manualmente em anos anteriores por esta escola, houve uma considerável redução no tempo gasto para gerar o mesmo. Enquanto que manualmente eram necessárias duas pessoas trabalhando durante, no mínimo, cinco dias, com a utilização do algoritmo, foi necessário apenas o processamento de sugestões de horários e uma pessoa responsável pela escolha do melhor horário para a escola.

Em Jacob Junior & Rocha (2005) foi implementada a aplicação AGHORA (Algoritmos Genéticos para geração de Horários de Aula). Fica a critério do usuário decidir qual tipo de seleção será realizada, por torneio ou por roleta, já que ambos estão disponíveis. A modelagem do problema é bastante similar ao do *school timetabling* e não leva em conta a alocação de salas. O algoritmo trata algumas restrições, como preferência dos professores e aulas geminadas. Um ponto a ser citado é que o autor permite que haja indivíduos inviáveis, soluções inválidas, na população. Apesar disso, um quadro de horários viável com $fitness = 0$ foi obtido inúmeras vezes, mas somente configurando o algoritmo para trabalhar com uma população de 250 indivíduos e executar por 1500 gerações, o que aumentou muito o tempo de execução, levando 2 horas e 58 minutos para terminar a execução.

Outro trabalho relacionado que pode ser citado é Barboza et al. (2007). Neste trabalho, a ferramenta Kairós foi desenvolvida, baseada no *school timetabling*, para satisfazer as necessidades da Faculdade Ruy Barbosa (FRB) - Salvador/BA. Foi feito um estudo de caso com os professores da FRB para definir quais as restrições tem maior grau de importância, e os problemas mais citados pelos coordenadores foram as indisponibilidades de determinados docentes em ministrar aulas em alguns períodos, e a priorização de professores em determinadas matérias aumenta as chances de colisões. O trabalho não dá detalhes adicionais sobre a implementação do AG, mas traz resultados obtidos nos testes do algoritmo. Pode-se observar pela tabela dos resultados que o algoritmo aceita uma solução inválida, com choque de horários, como solução final. Em um teste com população inicial de 5000 indivíduos e 1000 gerações o tempo médio de processamento foi de 78 minutos.

Em Martins (2004), o autor apresentou uma ferramenta para auxiliar na geração dos horários do CEFET-Campos/RJ. A solução utiliza AG para resolver o *school timetabling*. O autor gerou as soluções iniciais de forma aleatória, o que permite que se tenham indivíduos

inválidos na população, e não dá detalhes sobre a implementação sobre os procedimentos de seleção, mutação e cruzamento. O algoritmo não leva em consideração buracos nos horários, isto é, espaços vazios entre aulas agendadas, e também as preferências dos professores. Segundo Martins (2004), o algoritmo desenvolvido é capaz de gerar boas soluções, mais ainda se aplicado a instâncias com menos restrições.

Em Costa & Dalla Bruna (2002) também foi aplicado AG para resolver o problema da geração de quadros de horários. Foram levadas em consideração restrições como:

- Uma turma não pode ter um dia da semana sem aulas;
- Disciplinas diferentes, ministradas pelo mesmo professor, não podem ser agendadas para o mesmo horário;
- Cada professor deve ter o máximo de satisfação em relação aos horários agendados;
- Não pode haver aulas geminadas;

Um curso universitário com 8 turmas e 44 disciplinas foi usado como estudo de caso em Costa & Dalla Bruna (2002). Para representação dos indivíduos foi escolhida uma matriz $M_{dias \times horarios}$, onde cada posição da matriz contém uma referência à disciplina e ao professor alocado para aquele horário. Na geração da população inicial, os autores se preocuparam em gerar apenas indivíduos factíveis. Neste trabalho, foi implementado o método de seleção por torneio e também foi usado elitismo. O algoritmo mostrou bom desempenho; à medida que o tamanho da população inicial era maior, o número de gerações para se encontrar uma solução ótima diminuía. Isso se dá pelo fato de se ter um aumento na diversidade dos indivíduos, o espaço de busca é bem mais explorado.

Algoritmos genéticos também foram aplicados para resolver um problema de *school timetabling* em Colorni et al. (1992). Indivíduos inviáveis também foram incluídos no espaço de busca do algoritmo. As soluções são representadas por uma matriz $M_{m \times p}$ onde a i-ésima linhada matriz *M* representa o quadro de horário para o professor t_i. Cada célula m_{ik} damatriz contém o nome da turma a qual o professor está agendado no período *k*. A função de avaliação considera os requisitos didáticos, organizacionais e pessoais citados na seção 2.1.2.3. Para conduzir o algoritmo a encontrar soluções factíveis, a função objetivo atribui às inviabilidades um peso maior que os demais requisitos desejáveis.

3 METODOLOGIA

3.1 Modelagem do problema

O objetivo deste trabalho é desenvolver um algoritmo para gerar quadros de horários usando como estudo de caso o curso de Ciência da Computação da Unipampa.

O curso de Ciência da Computação da Unipampa tem aulas de segunda a sábado, quatro períodos por dia. No total, uma turma pode ter no máximo 24 aulas durante a semana, como mostra a tabela 3.1. Não necessariamente todas as turmas têm o horário cheio; depende da oferta de disciplinas da turma para o semestre e da carga horária dessas disciplinas.

TABELA 3.1

Horário de uma turma do curso de Ciência da Computação

CC1	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	- -	- -	- -	- -	- -	- -
Aula 2	- -	- -	- -	- -	- -	- -
Aula 3	- -	- -	- -	- -	- -	- -
Aula 4	- -	- -	- -	- -	- -	- -

Como base para o trabalho, foi usada a variante de *timetabling* chamada *School Timetabling*, descrita na seção 2.1.2 deste trabalho. Após algumas conversas com coordenadores de curso e responsáveis pela geração do horário acadêmico no campus Alegrete foi necessário fazer algumas adaptações ao modelo clássico de *school timetabling* para tornar o problema mais parecido com um problema real, como por exemplo, impor requisitos desejáveis, ou restrições “fracas” e “fortes” (CALDEIRA & FERNANDES, 2002).

3.1.1 Restrições

3.1.1.1 Restrições “fortes”

As restrições “fortes” são aquelas que tornam uma solução inviável, ou inválida. Para este problema, são as seguintes:

Turma com duas aulas no mesmo período. Uma turma não pode estar agendada para mais de uma aula no mesmo período de tempo. Pela representação das soluções que será proposta, esta restrição nunca será violada.

Professor com duas aulas no mesmo período. Uma professor não pode estar agendado para mais de uma aula no mesmo período de tempo.

As tabelas 3.2 e 3.3 ilustram essa restrição. Para as tabelas geradas, há colisão no horário do professor José, pois ele está agendado para ministrar Algoritmos na turma CC1 e ED II na turma CC3 nas aulas 1 e 2.

TABELA 3.2

Horário da turma ‘CC1’

CC1	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Algoritmos José	-	-	-	-	-
Aula 2	Algoritmos José	-	-	-	-	-
Aula 3	-	-	-	-	-	-
Aula 4	-	-	-	-	-	-

TABELA 3.3

Horário da turma ‘CC3’

CC3	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	ED II José	-	-	-	-	-
Aula 2	ED II José	-	-	-	-	-
Aula 3	-	-	-	-	-	-
Aula 4	-	-	-	-	-	-

3.1.1.2 Restrições “fracas”

As restrições “fracas” não tornam uma solução inviável, ou inválida. Mas podem servir para medir se uma solução é melhor que outra. Para este problema, são as seguintes:

Preferência dos professores. Esta restrição avalia as preferências dos professores em lecionar em determinados períodos. Por exemplo, pode ser que determinado professor esteja indisponível na segunda-feira no primeiro período. Então, não se devem agendar aulas para ele neste período.

A tabela 3.4 exemplifica as preferências de horários do professor José, onde os valores:

- 1 - o professor está disponível para lecionar nesse horário;
- 2 - o professor não deseja lecionar nesse horário;
- 3 - o professor não está disponível para lecionar nesse horário;

TABELA 3.4

Horário do professor José – Tabela de preferência de horários

José	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	1	3	2	2	1	1
Aula 2	1	3	2	2	1	1
Aula 3	1	1	1	3	1	1
Aula 4	1	1	1	3	1	1

Dia de folga. É desejável que os professores tenham um dia de folga durante a semana. A tabela 3.5 mostra o horário semanal do professor José, onde ele tem a sexta-feira livre.

TABELA 3.5

Horário do professor José – dia de folga

José	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Algoritmos	ED II	ED II	-	-	OAD
	CC1	CC3	CC3	-	-	CC3
Aula 2	Algoritmos	ED II	ED II	-	-	OAD
	CC1	CC3	CC3	-	-	CC3
Aula 3	-	Algoritmos	-	OAD	-	-
	-	CC1	-	CC3	-	-
Aula 4	-	Algoritmos	-	OAD	-	-
	-	CC1	-	CC3	-	-

Aulas consecutivas. Não é desejável que uma turma tenha muitas aulas consecutivas da mesma disciplina ou com o mesmo professor, pois alunos e professores ficariam insatisfeitos com o horário.

As tabelas 3.6 e 3.7 ilustram horários com aulas consecutivas de uma disciplina e de um mesmo professor, respectivamente. Na tabela 3.6, a turma CC3 tem quatro aulas consecutivas de ED II. Na tabela 3.7, a turma CC3 tem não tem aulas da mesma disciplina, mas tem com o mesmo professor. São quatro períodos com o professor Carlos, nas disciplinas de ED II e OAD, dois períodos cada.

TABELA 3.6

Horário da turma ‘CC3’- aulas consecutivas de uma mesma disciplina

CC3	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-
Aula 2	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-
Aula 3	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-
Aula 4	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-

TABELA 3.7

Horário da turma ‘CC3’- aulas consecutivas com o mesmo professor

CC3	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-
Aula 2	-	ED II	-	-	-	-
	-	Carlos	-	-	-	-
Aula 3	-	OAD	-	-	-	-
	-	Carlos	-	-	-	-
Aula 4	-	OAD	-	-	-	-
	-	Carlos	-	-	-	-

Número de dias com aulas. Esta restrição avalia se o número de dias em que as turmas têm aulas está de acordo com o número mínimo de dias necessários para ministrar as mesmas. O objetivo dessa restrição é impedir que uma turma, por exemplo, tenha dois dias na semana com duas aulas apenas. Respeitando essa restrição, as quatro aulas seriam agendadas para o mesmo dia e a turma teria um dia livre.

A tabela 3.8 exemplifica um caso em que a turma CC3 tem dois dias na semana - quarta e sexta - com apenas duas aulas.

TABELA 3.8

Horário da turma ‘CC3’ – número de dias com aulas

CC3	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	OAD	ED II	-	ED II	M. Discreta	-
	Carlos	Carlos	-	Carlos	Cezar	-
Aula 2	OAD	ED II	-	ED II	M. Discreta	-
	Carlos	Carlos	-	Carlos	Cezar	-
Aula 3	Estatística	Estatística	OAD	M. Discreta	-	-
	Matias	Matias	Carlos	Cezar	-	-
Aula 4	Estatística	Estatística	OAD	M. Discreta	-	-
	Matias	Matias	Carlos	Cezar	-	-

Formação de janelas no horário. Este critério avalia se existem janelas no horário das turmas, ou seja, horários vagos. A tabela 3.9 ilustra essa restrição. Na segunda-feira, tem uma janela no horário da turma CC3. A turma tem aulas nos períodos 1, 3 e 4, e fica vaga no período 2.

TABELA 3.9

Horário da turma ‘CC3’ - janelas no horário

CC3	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	OAD	-	-	-	-	-
	Carlos	-	-	-	-	-
Aula 2	-	-	-	-	-	-
	-	-	-	-	-	-
Aula 3	Estatística	-	-	-	-	-
	Matias	-	-	-	-	-
Aula 4	Estatística	-	-	-	-	-
	Matias	-	-	-	-	-

Esta restrição será tratada já na geração da solução inicial. Em vez de termos quatro aulas individuais por dia, teremos dois blocos de duas aulas de uma mesma disciplina.

3.2 Representação dos indivíduos

A primeira tarefa ao se implementar um AG é definir como será a representação das soluções. Para representar as soluções, foi seguido o modelo proposto em Timóteo (2002), onde cada gene representa um *slot* (ex.: segunda-feira às 7:00). Um indivíduo (solução) é uma matriz de três dimensões [número de turmas X (Número de dias X Número de Horários)], conforme mostrado na figura 3.1:

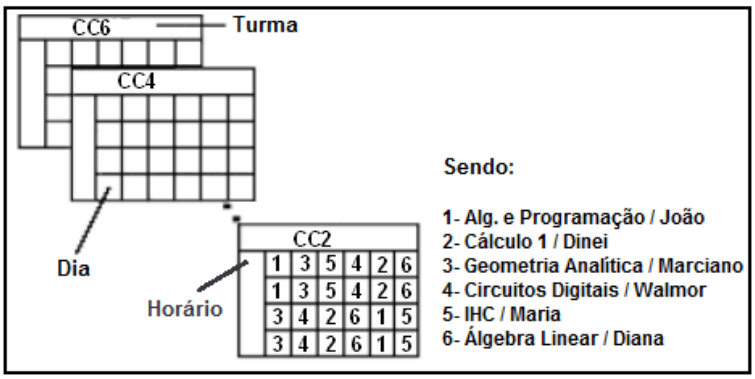

FIGURA 3.1 – Representação dos indivíduos.

Essa representação apresenta a vantagem de não permitir, graças a sua codificação, que duas disciplinas referentes à mesma turma ocupem o mesmo horário. Porém, não garante

a não ocorrência de outras restrições, como por exemplo, um mesmo professor estar agendado para duas aulas ao mesmo período de tempo.

Na figura, por exemplo, o número 1 no *slot* da linha 1, coluna 1, significa que aquele horário da turma CC2 está reservado para a disciplina de Algoritmos e programação, com o professor João. É importante observar que dependendo do número de turmas e *slots* disponíveis, a representação de uma solução pode ser muito grande, requisitando assim maiores recursos computacionais.

Como um AG trabalha com uma população de indivíduos, a representação de uma população seria uma lista encadeada de matrizes tridimensionais semelhantes a da figura acima.

3.3 Geração da população inicial

Para gerar uma solução inicial, foram desenvolvidos dois algoritmos. Ambos os algoritmos carregam a lista de turmas, as disciplinas de cada turma e as preferências dos professores do banco de dados. Conforme a modelagem proposta, em que cada aula é de dois períodos, a carga horária de todas as disciplinas são divididas por dois. Por exemplo, se uma disciplina tem carga horária de seis horas semanais, ela será ministrada em três aulas de dois períodos. O algoritmo abaixo demonstra o processo:

Algoritmo 3.1 Algoritmo de inicialização.

```
LT: Lista de turmas;
Início
    Carrega a lista de turmas LT e as disciplinas referentes a mesma do banco de dados;
    Carrega as preferências dos professores do banco de dados;
    Para cada turma t ∈ LT
        Para cada disciplina d ∈ t
                        n ← d.cargaHoraria/2
            Insere n aulas da disciplina d na lista de aulas da turma t;
        fimPara;
    fimPara;
Fim
```

O primeiro algoritmo para gerar a solução inicial é totalmente aleatório e sempre gera uma solução, independente desta ser válida ou não. Para cada turma, sorteia-se uma aula da lista de aulas de cada turma e ela é alocada em uma posição também randômica da grade horária da turma. O processo repete-se até que todas as aulas da turma estejam alocadas.

O algoritmo é descrito em pseudocódigo abaixo:

Algoritmo 3.2 Algoritmo de geração da solução inicial aleatório

```
LT: Lista de turmas;
Início
    Para cada turma t ∈ LT
        Enquanto a lista de aulas da turma t_l não estiver vazia
            Sorteia uma aula a;
            Aloca a aula a em algum lugar aleatório disponível na grade horária
            da turma t;
            Remove a da lista;
        fimEnquanto;
    fimPara;
Fim
```

O segundo algoritmo, ao contrário do primeiro, não é aleatório. Este sempre gera soluções válidas e consiste em, para cada turma, enquanto a lista de aulas não estiver vazia, sortear uma aula *a* aleatoriamente e alocar a mesma na primeira posição livre do quadro de horários da turma que não afete a factibilidade da solução. Se não tiver mais nenhum lugar disponível na grade que não torne a solução inviável, executa uma troca. Nesta troca, uma aula *b*, já alocada, é movimentada para uma posição livre na grade e a aula *a* é alocada na posição anterior da aula *b*, mantendo a factibilidade da solução.

A seguir, o pseudocódigo do algoritmo:

Algoritmo 3.2 Algoritmo de geração da solução inicial ‘válido’

```
LT: Lista de turmas;
Início
    Para cada turma t ∈ LT
        Enquanto a lista de aulas da turma t_l não estiver vazia
            Sorteia uma aula a;
```

Tenta alocar a aula *a* no primeiro lugar disponível na grade horária da turma *t* a qual não gere uma inviabilidade;
Se não encontrou lugar disponível, executa uma troca comum a aula *b* já alocada, inserindo *b* em algum lugar livre da grade e *a* na posição anterior de *b*;
Remove *a* da lista;
fimEnquanto;
fimPara;
Fim

3.4 Avaliação dos indivíduos

Para avaliar os indivíduos gerados, foram usados os critérios descritos na seção 3.1.1 desta monografia. A cada critério foi escolhido arbitrariamente um valor de penalidade de acordo com a sua importância. A saber, são estes:

- Professor com duas aulas no mesmo período (penalidade = 1000).
- Professor em horário indisponível (penalidade = 100).
- Professor em horário indesejável (penalidade = 60).
- Professor sem dia de folga (penalidade = 50).
- Aulas consecutivas (penalidade = 50).
- Número de dias com aulas (penalidade = 25).

O resultado da função de avaliação é o somatório de todas as ocorrências dessas restrições, multiplicados ao seu respectivo valor de penalidade. Se nenhuma restrição for encontrada, o resultado é zero e o horário é dito ótimo.

3.5 Seleção dos indivíduos

Esta é a etapa onde selecionamos indivíduos para o processo de cruzamento. Dois métodos foram implementados: seleção por roleta e seleção por torneio, descritos na seção 2.2.6.1. Antes de aplicar qualquer um dos métodos, é necessário que os indivíduos passem pela função de avaliação para que recebam a quantificação do seu grau de aptidão ao problema.

Na seleção por roleta, o valor da função de avaliação de cada indivíduo é usado para montar a roleta de acordo com a proporcionalidade de cada um na população. Já na seleção por torneio, se faz uso de outro valor, a *taxa de seleção*, passado por parâmetro ao algoritmo genético. Sorteia-se um valor aleatório, e se este for maior que a taxa de seleção o indivíduo melhor é selecionado, senão o indivíduo menos adaptado é selecionado.

Outro método auxiliar de seleção foi implementado – o elitismo. É passado por parâmetro ao algoritmo um valor chamado *taxa de elitismo* que determina uma porcentagem dos melhores indivíduos da geração atual que devem ser preservados para a próxima geração. Com isso, evita-se perder os indivíduos mais adaptados, já que com os métodos de seleção por torneio e roleta não é certo que eles serão selecionados. O elitismo é executado antes da seleção por torneio ou seleção por roleta.

3.6 Cruzamento

Na fase de cruzamento, dois indivíduos se combinam para criar novos indivíduos carregando a carga genética de ambos. Para isto foram implementados o cruzamento em um ponto, e o cruzamento em dois pontos descritos na seção 2.2.6.2.

Para o problema da geração de quadros de horários os processos de cruzamento não são plenamente aplicáveis. Devem ser adaptados devido a alguns problemas que podem ocorrer se apenas cruzarmos os genes dos pais, como por exemplo, alterar a carga horária das disciplinas e das turmas e consecutivamente afetar a factibilidade. As tabelas 3.10, 3.11, 3.12 e 3.13 exemplificam essa situação, de um cruzamento em um ponto com ponto de corte na quinta-feira - Aula 2, onde antes do corte o filho 1 recebe a carga genética do pai 1 e depois do corte recebe do pai 2, e com o filho 2 acontece o contrário:

TABELA 3.10

Cromossomo Pai 1

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex

TABELA 3.11

Cromossomo Pai 2

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Prob. Estatística Alex	M. Discreta Carlos
Aula 2	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Prob. Estatística Alex	M. Discreta Carlos
Aula 3	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar	M. Discreta Carlos	Prob. Estatística Alex
Aula 4	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar	M. Discreta Carlos	Prob. Estatística Alex

TABELA 3.12

Cromossomo Filho 1

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	M. Discreta Carlos
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	M. Discreta Carlos
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	M. Discreta Carlos	Prob. Estatística Alex
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	M. Discreta Carlos	Prob. Estatística Alex

TABELA 3.13

Cromossomo Filho 2

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Cálculo II Francisco	Prob. Estatística Alex
Aula 2	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Cálculo II Francisco	Prob. Estatística Alex
Aula 3	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex
Aula 4	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex

Como podemos notar nas tabelas 3.12 e 3.13, a carga horária de algumas disciplinas foi alterada. Antes do cruzamento todas as disciplinas eram ministradas em quatro aulas durante a semana. Após o cruzamento o horário apresentou as seguintes alterações:

- Na tabela 3.12:
  - A disciplina M. Discreta passou a ter oito aulas na semana;
  - A disciplina ED I passou a ter apenas duas aulas;
  - A disciplina Cálculo II passou a ter apenas duas aulas;
- Na figura 3.13:
  - A disciplina Cálculo II passou a ter seis aulas;
  - A disciplina ED I passou a ter seis aulas;
  - A disciplina M. Discreta ‘sumiu’ do horário;

Para corrigir isto foi adotada a seguinte estratégia: alocar apenas as aulas que não alterem a carga horária da disciplina e também não inviabilizem a solução, conforme exemplificado pelas figuras 3.14 e 3.15.

TABELA 3.14

Cromossomo Filho 1

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar		Prob. Estatística Alex
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar		Prob. Estatística Alex

TABELA 3.15

Cromossomo Filho 2

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar		Prob. Estatística Alex
Aula 2	ED I Felipe	Circuitos Digitais Alecsandro	Circuitos Digitais Alecsandro	Álgebra Linear Cesar		Prob. Estatística Alex
Aula 3	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar		Prob. Estatística Alex
Aula 4	ED I Felipe	Cálculo II Francisco	Cálculo II Francisco	Álgebra Linear Cesar		Prob. Estatística Alex

As aulas que não puderem ser alocadas diretamente no cruzamento são inseridas em uma lista e alocadas usando o mesmo procedimento da geração de horários válidos, descrito na seção 3.3.

3.7 Mutação

A mutação altera as características de um quadro de horários. Após o cruzamento temos dois novos indivíduos carregando a carga genética dos pais. Algumas boas características podem ter sido perdidas, e durante o processo de mutação tenta-se recuperar estas. Para este problema, foi implementada a mutação por troca aleatória, descrita na seção 2.2.6.3, e outro método que é uma adaptação dessa mutação o qual será chamado de *mutação heurística*.

Na mutação por troca, usou-se uma *taxa de mutação* passada por parâmetro ao algoritmo e para cada turma sorteia-se um valor aleatório. Se esse valor for menor que a *taxa de mutação*, duas aulas são escolhidas aleatoriamente e trocam de lugar na grade da turma, como mostram as tabelas 3.16 e 3.17:

TABELA 3.16

Cromossomo antes da mutação aleatória

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex

TABELA 3.17

Cromossomo após a mutação aleatória

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	M. Discreta Carlos	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	M. Discreta Carlos	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 3	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex
Aula 4	Circuitos Digitais Alecsandro	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex

Na mutação heurística, o processo é similar ao da mutação aleatória. A diferença está no fato de que a troca só acontece se melhorar o valor da função de avaliação da solução. As tabelas 3.18 e 3.19 exemplificam essa troca:

TABELA 3.18

Cromossomo antes da mutação heurística

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Cálculo II Francisco	Prob. Estatística Alex
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Prob. Estatística Alex

TABELA 3.19

Cromossomo após a mutação heurística

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	Prob. Estatística Alex
Aula 2	Circuitos Digitais Alecsandro	ED I Felipe	Circuitos Digitais Alecsandro	M. Discreta Carlos	Prob. Estatística Alex	Prob. Estatística Alex
Aula 3	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Cálculo II Francisco
Aula 4	M. Discreta Carlos	Álgebra Linear Cesar	Cálculo II Francisco	Álgebra Linear Cesar	ED I Felipe	Cálculo II Francisco

Como mostra a tabela 3.18, no sábado a turma tem quatro aulas da disciplina de Prob. Estatística, e segundo a modelagem feita para este problema isso é uma restrição “fraca”, que deve ser evitada. Para isso, ela é deslocada para a sexta-feira, trocando com a disciplina de Cálculo II, e melhorando, assim, o valor de avaliação da solução.

3.8 Critério de parada

Os critérios de parada adotados para este problema são:

- Número de gerações, passado por parâmetro ao algoritmo;
- Não haver diferença entre o valor de avaliação da melhor e da pior solução daquela geração;
- Encontrar uma solução “ótima”, com valor de avaliação $= 0$.

3.9 Execução do algoritmo

Antes de executar o algoritmo, devem ser passados como parâmetro ao mesmo os seguintes dados:

- Tamanho da população;
- Número máximo de gerações;
- Taxa de seleção
    - A taxa de seleção é utilizada somente se o método de seleção escolhido for o método por torneio.
- Taxa de mutação;
- Método de geração da população;
- Método de seleção;
- Método de cruzamento;
- Método de Mutação;

3.10 Implementação

Para implementar o algoritmo genético e os métodos descritos acima, foi escolhido o paradigma de programação orientado a objetos, por ser um paradigma de fácil entendimento e aplicação. A linguagem usada foi Java. Java foi escolhida por ser uma linguagem robusta, livre e multi-plataforma, isto é, pode ser usada tanto em Windows, quanto Linux e outros sistemas operacionais.

A utilização de um SGBD se fez necessária neste trabalho, pois os cadastros (turmas, aulas, professores, disciplinas e indisponibilidades) precisavam ser armazenados. O SGBD escolhido foi o PostgreSQL, versão 1.8.4, por ser livre e ser um SGBD relacional. Os diagramas de classes UML, bem como do banco de dados relacional, se encontram na seção de apêndices ao final desta monografia.

Afim de que o algoritmo pudesse ser usado pelos coordenadores de curso, foi desenvolvida uma aplicação que usa o algoritmo genético implementado neste trabalho. Essa aplicação utiliza os componentes gráficos da linguagem Java e possibilita manipular os dados referentes a aulas, professores, turmas e disciplinas, bem como executar o algoritmo e ver o resultado da sua execução. As figuras 3.2, 3.3 e 3.4 mostram algumas telas do aplicativo:

FIGURA 3.2 – Tela inicial da aplicação.

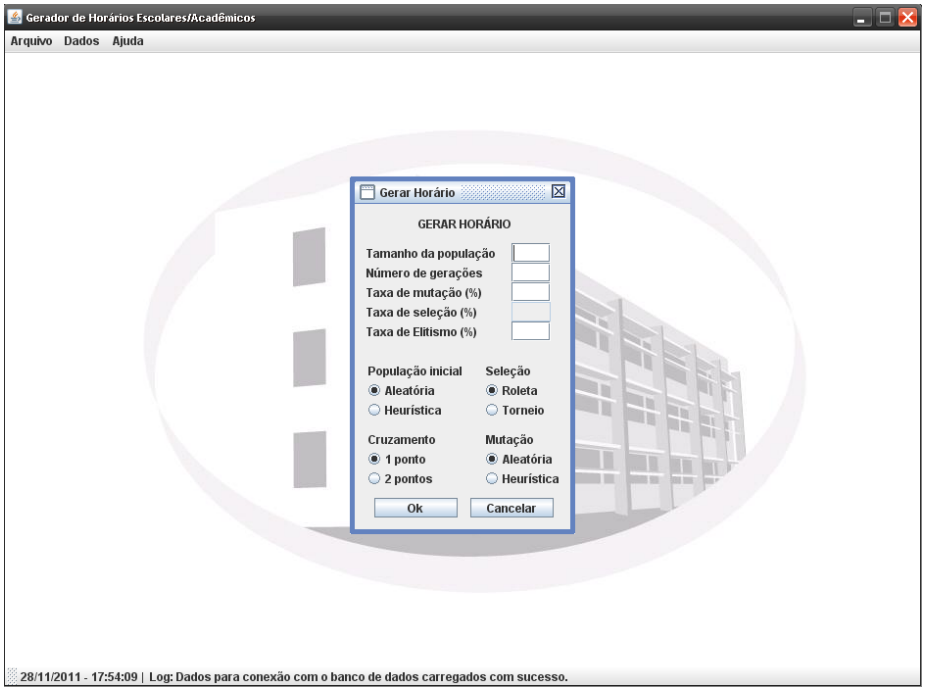

FIGURA 3.3 – Tela de inicialização do AG.

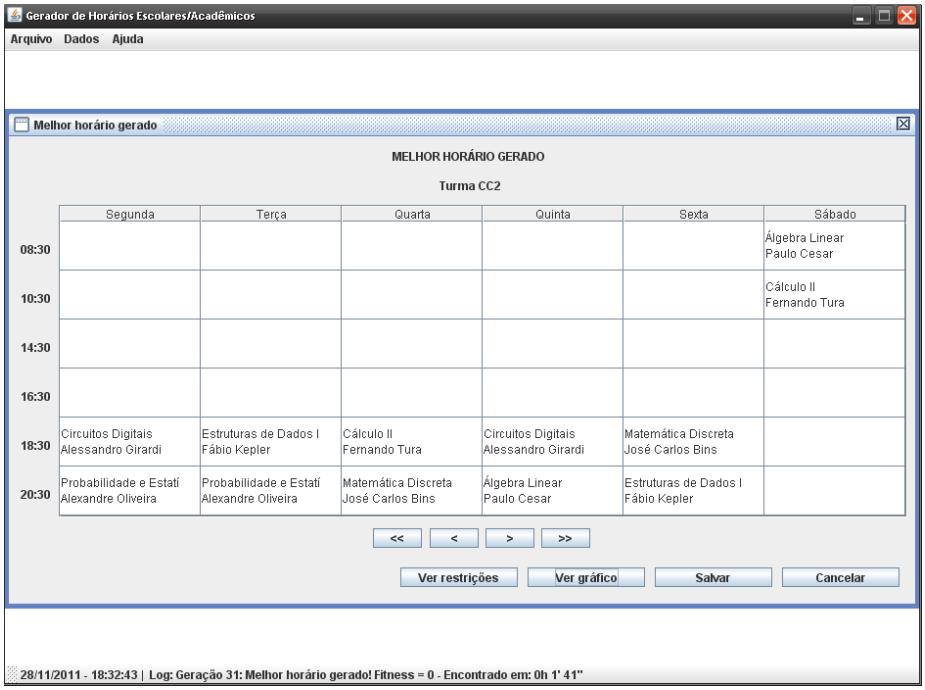

FIGURA 3.4 – Tela de representação da melhor solução encontrada.

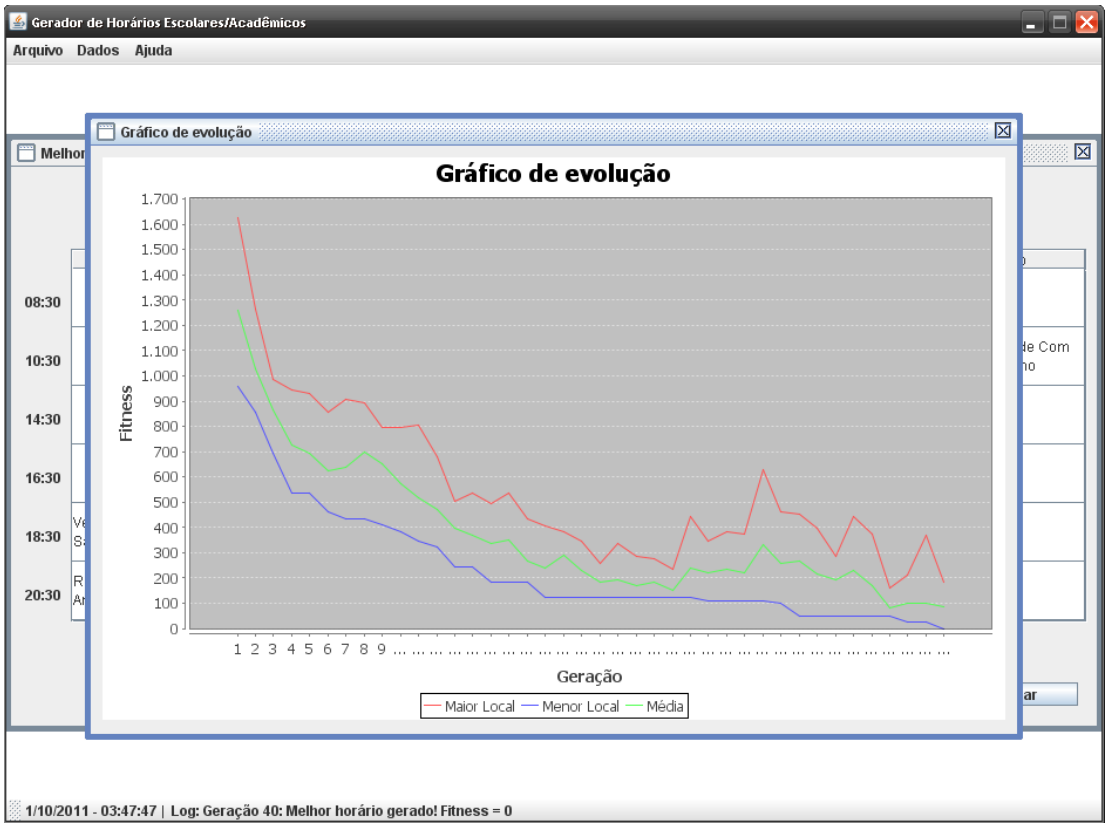

FIGURA 3.5 – Tela de representação da evolução da população a cada geração do algoritmo.

4 RESULTADOS OBTIDOS

Foram realizados vários testes no algoritmo desenvolvido. Para realização dos mesmos foi utilizado um notebook Dual-Core 2.1 Ghz com 3Gb de memória e rodando sobre o sistema operacional Windows XP SP2.

Para testar o algoritmo, duas instâncias foram utilizadas. A instância um é uma instância real, e a instância dois é fictícia. A tabela 4.1 traz alguns dados das instâncias de teste:

TABELA 4.1

Dados gerais das instâncias de teste

Instância	Turmas	Disciplinas	Professores
1	6	30	19
2	4	22	6

A instância um diz respeito às turmas dos cursos de Ciência da Computação e Engenharia de Software do semestre 2011/2 da Unipampa (os nomes dos professores foram trocados). As turmas da Engenharia de Software foram incluídas nos testes devido ao fato das aulas do curso serem no mesmo turno das aulas da Ciência da Computação e também por que os dois cursos têm professores em comum.

Segue abaixo os dados referentes às turmas (CC2, CC4, CC6, CC8, ES2, ES4), disciplinas e carga horária, e professor que ministra as mesmas, da instância um:

- CC2
    - Circuitos Digitais / 4 aulas / Prof. Alecsandro Souza
    - M. Discreta / 4 aulas / Prof. João Silva
    - Estruturas de Dados I / 4 aulas / Prof. Fabiano Santos
    - Álgebra Linear / 4 aulas / Prof. Carlos Alberto Torres
    - Prob. e Estatística / 4 aulas / Prof. Alex Raphael
    - Cálculo II / 4 aulas / Prof.ª Fernanda Soares
- CC4
    - Programação OO / 4 aulas / Prof. Daniel Freitas
    - Banco de Dados I / 4 aulas / Prof. Willian Lucas

    - Arq. de Computadores II / 4 aulas / Prof.ª Gabriele Buchert
    - Programação p/Web / 4 aulas / Prof. Cássio Fernandes
    - Proj. e Análise de Algoritmos / 4 aulas / Prof. Fabiano Santos
- CC6
    - Compiladores / 4 aulas / Prof. João Silva
    - Redes de Computadores / 4 aulas / Prof.ª Gabriele Buchert
    - Sist. de Tempo Real / 4 aulas / Prof. Elias Figueroa
    - Engenharia de Software II / 4 aulas / Prof. Eder Pinheiro
    - Metod. de Pesq. Científica / 4 aulas / Prof. Cristian Melo
    - Proc. de Linguagem Natural / 4 aulas / Prof. Fabiano Santos
- CC8
    - Tec. em Contexto Social / 4 aulas / Prof.ª Mara Drebes
    - Sistemas de Informação / 4 aulas / Prof. Cristian Melo
- ES2
    - Prob. e Estatística / 4 aulas / Prof. Rodrigo Ruas
    - Teste e Depuração de Código / 2 aulas / Prof. Daniel Freitas
    - Resolução de Problemas II / 2 aulas / Prof. Daniel Freitas
    - Resolução de Problemas II / 2 aulas / Prof. Paulo Ribeiro
    - Resolução de Problemas II / 4 aulas / Prof. Eder Pinheiro
    - Programação OO / 4 aulas / Prof. Cássio Fernandes
    - Tipos Abstratos de Dados / 4 aulas / Prof. Ronaldo Fetter
    - Teoria dos Grafos / 2 aulas / Prof. Paulo Ribeiro
- ES4
    - Análise de Software / 2 aulas / Prof.ª Mara Drebes
    - Intro. a Análise de Algoritmos / 2 aulas / Prof. Cássio Fernandes
    - Intro. a Sistemas de Computação/ 4 aulas / Prof. Emerson Neves
    - Resolução de Problemas IV / 2 aulas / Prof.ª Mara Drebes
    - Resolução de Problemas IV / 6 aulas / Prof. Jean Gonçalves
    - Práticas em Banco de Dados / 4 aulas/ Prof. Willian Lucas
    - Verificação e Validação de Software / 4 aulas / Prof. Jean Gonçalves

A tabela 4.2 faz uma relação professor/turma. As preferências de horários dos professores foram atribuídas de maneira aleatória. Todos têm horários em que estão indisponíveis e também que não gostariam de lecionar.

TABELA 4.2

Relação professor/turma – Instância um

Professor	CC2	CC4	CC6	CC8	ES2	ES4
Alecsandro Souza	X					
Alex Raphael	X					
Cássio Fernandes		X			X	X
Carlos Alberto Torres	X					
Cristian Melo			X	X		
Daniel Freitas		X			X	
Eder Pinheiro			X		X	
Elias Figueroa			X			
Ewerson Carvalho						X
Fabiano Santos	X	X	X			
Fernanda Soares	X					
Gabriele Buchert		X	X			
Jean Gonçalves						X
João Silva	X		X			
Mara Drebes				X		X
Rodrigo Ruas					X	
Ronaldo Fetter					X	
Paulo Ribeiro		X			X	
Willian Lucas		X				X

O algoritmo foi executado dez vezes para a instância um usando a seguinte configuração:

- População: 50;
- Número de Gerações: 100;
- Taxa de Mutação: 10%;
- Taxa de Elitismo: 10%;
- Geração somente de indivíduos válidos na solução inicial;
- Seleção por roleta;
- Cruzamento em um ponto;
- Mutação heurística;

O resultado da execução do algoritmo para a instância um pode ser visto na tabela 4.3. Esta tabela contempla as dez execuções do algoritmo e em cada linha traz os dados referentes à geração na qual o algoritmo parou a execução, o tempo que o mesmo levou para executar, o valor de avaliação da melhor solução encontrada e a quantidade de violações a cada restrição descrita na seção 3.1.1. O algoritmo teve um ótimo desempenho, conseguindo encontrar uma solução ótima em todas as execuções. Em média essa solução foi encontrada na 18ª geração, bem antes do critério de parada passado ao algoritmo que era de 100 gerações.

TABELA 4.3

Resultado da execução do algoritmo - Instância um

#	Geração	*Fitness*	Tempo	Conflitos	Indesejáveis	Indisponíveis	Folga	Aulas geminadas	Aulas p/dia
1	18	0	2’ 36”	0	0	0	0	0	0
2	19	0	2’ 56”	0	0	0	0	0	0
3	16	0	2’ 32”	0	0	0	0	0	0
4	21	0	3’ 12”	0	0	0	0	0	0
5	13	0	1’ 59”	0	0	0	0	0	0
6	17	0	2’ 30”	0	0	0	0	0	0
7	20	0	2’ 57”	0	0	0	0	0	0
8	30	0	4’ 15”	0	0	0	0	0	0
9	18	0	2’ 32”	0	0	0	0	0	0
10	17	0	2’ 20”	0	0	0	0	0	0

A instância dois é uma instância fictícia. Foram utilizadas as turmas da Ciência da Computação do semestre 2011/1. Os professores foram alterados para que pudéssemos ter um problema mais complexo de ser resolvido. Isso se dá pelo fato de que quanto maior o número de disciplinas e menor o número de professores se torna mais difícil encontrar uma grade onde não há nenhum conflito de professor lecionando no mesmo horário em mais de uma turma. Se tivermos um professor para cada disciplina, o problema se torna trivial e não há a possibilidade de haver esse tipo de conflito.

Segue abaixo os dados referentes às turmas, disciplinas e carga horária, e professor que ministra as mesmas, da instância dois:

- CC1
    - Algoritmos e Programação / 4 aulas / Prof. Cláudio
    - Cálculo I / 4 aulas / Prof. Marcos

    - Geometria Analítica / 4 aulas / Prof. Luiz
    - Lógica Matemática / 4 aulas / Prof. Simão
    - Introdução a Ciência e Tecnologia / 4 aulas / Prof.ª Julia
    - Eletrotécnica / 4 aulas / Prof. Ronaldo
- CC3
    - Comunicação de Dados / 4 aulas / Prof. Cláudio
    - Arquitetura e Organização de Computadores I / 4 aulas / Prof. Marcos
    - Organização de Arquivos e Dados / 4 aulas / Prof. Luiz
    - Estruturas de Dados II / 4 aulas / Prof. Simão
    - Acessibilidade e Inclusão Digital / 4 aulas / Prof.ª Julia
- CC5
    - Sistemas Operacionais / 4 aulas / Prof. Cláudio
    - Linguagens Formais / 4 aulas / Prof. Marcos
    - Engenharia de Software I / 4 aulas / Prof. Luiz
    - Proj. de Linguagens de Programação / 4 aulas / Prof. Simão
    - Inteligência Artificial / 4 aulas / Prof.ªJulia
    - Desafios de Programação / 4 aulas / Prof. Ronaldo
- CC7
    - Bancos de Dados II / 4 aulas / Prof. Cláudio
    - Sistemas Distribuídos / 4 aulas / Prof. Marcos
    - Intro. ao Processamento de Imagens Digitais / 4 aulas / Prof. Luiz
    - Compiladores / 4 aulas / Prof. Simão
    - Administração e Empreendedorismo / 4 aulas / Prof.ª Julia

A tabela 4.6 mostra a relação professor/turma da instância dois. Assim como na outra instância de teste, as preferências de horários foram atribuídas a cada professor aleatoriamente. Todos têm horários em que não desejam ou não podem lecionar, ou ambos.

TABELA 4.6

Relação professor/turma – Instância dois

Professor	CC1	CC3	CC5	CC7
Cláudio	X	X	X	X
Julia	X	X	X	X
Luiz	X	X	X	X
Marcos	X	X	X	X
Ronaldo	X		X	
Simão	X	X	X	X

O algoritmo foi executado dez vezes utilizando a seguinte configuração:

- População: 100;
- Número de Gerações: 100;
- Taxa de Mutação: 10%;
- Taxa de Elitismo: 10%;
- Geração somente de indivíduos válidos na solução inicial;
- Seleção por roleta;
- Cruzamento em um ponto;
- Mutação heurística;

Como o problema da instância dois é bem restritivo, a população foi aumentada em relação à instância um. Com uma população maior, a exploração do espaço de busca é maior, e aumentam as chances de encontrar uma boa solução. A tabela 4.7 mostra os resultados da execução do algoritmo para a instância dois.

TABELA 4.7

Resultado da execução do algoritmo - Instância dois

#	Geração	*Fitness*	Tempo	Conflitos	Indesejáveis	Indisponíveis	Folga	Aulas geminadas	Aulas p/dia
1	71	110	3’ 01”	0	1	0	0	0	2
2	57	145	2’ 23”	0	2	0	0	0	1
3	53	145	2’ 21”	0	2	0	0	0	1
4	59	60	2’ 39”	0	1	0	0	0	0
5	37	135	1’ 35”	0	1	0	1	0	1
6	39	85	1’ 43”	0	1	0	0	0	1
7	43	85	1’ 51”	0	1	0	0	0	1
8	48	60	1’ 58”	0	1	0	0	0	0
9	52	85	2’ 12”	0	1	0	0	0	1
10	88	85	3’ 19”	0	1	0	0	0	1

Em todas as execuções o algoritmo parou porque não havia mais diferença entre o valor de avaliação da melhor e o da pior solução naquela geração. As execuções nas linhas 1 e 2, apesar de terem violado o mesmo número de restrições, geraram soluções com valor de avaliação diferentes. Isso se deve ao tipo de restrição que cada uma violou. Por exemplo, na linha 1 a solução encontrada tem um ‘horário indisponível’ somando 60 ao valor de avaliação e duas violações de ‘aulas por dia’, somando mais 50 ao valor de avaliação, totalizando 110. Já na linha 2 acontece o contrário; duas violações de ‘horário indisponível’ e uma de ‘aulas por dia’, somando 145.

Dado o número de professores, seis para vinte e duas disciplinas, o algoritmo teve bom desempenho, sempre chegando perto de uma solução “ótima”. Em duas execuções, inclusive, encontrou uma solução que infringia apenas uma restrição “fraca”.

Os melhores horários gerados para ambas as instâncias, bem como as tabelas de disponibilidades dos professores, se encontram na seção de apêndices ao final desta monografia.

5 CONCLUSÕES

Neste trabalho foi apresentada uma solução para o problema da geração de grades horária utilizando algoritmos genéticos. O algoritmo desenvolvido, a partir de uma população inicial de baixa qualidade, transforma esta mesma população utilizando os operadores genéticos até encontrar uma solução aceitável.

O algoritmo não pôde ser comparado ao de outros autores devido ao fato de que a modelagem e as restrições são diferentes, e também por não se ter disponíveis os dados sobre turmas, disciplinas, e disponibilidades dos professores dos trabalhos relacionados.

Conforme observado nos resultados descritos no capítulo 4, o algoritmo teve bom desempenho, gerando soluções "ótimas", e quando não foi possível, gerando soluções que violavam de uma a três restrições. Também se pode notar que o algoritmo convergiu em poucas gerações, o que foi bastante interessante já que em alguns trabalhos relacionados os algoritmos implementados demoraram para convergir e executavam por 15000 gerações como no caso de Ciscon (2006). Em comparação com a confecção manual de um quadro de horários do tamanho dos usados nos testes, o algoritmo mostrou eficiência com relação ao tempo e a qualidade, gerando um bom quadro de horários em menos de 10 minutos.

Encontrar a melhor configuração para o algoritmo é um desafio. Por exemplo: utilizar uma população maior permite explorar melhor o espaço de busca. Quanto maior for a taxa de mutação, em menos gerações o algoritmo pode encontrar uma boa solução, porém o tempo de execução tende a aumentar devido ao número de trocas que o algoritmo faz. Dependendo do número de turmas o tempo de execução tende a aumentar, já que a representação das soluções proposta na seção 3.2 é por matrizes, maior é o número de operações envolvendo as mesmas. Assim, a configuração ideal depende de um estudo aprofundado do problema em questão.

Um problema encontrado foi o operador de cruzamento. Ao aplicá-lo observou-se que as cargas horárias das disciplinas eram alteradas e também que conflitos nos horários dos professores eram gerados, inviabilizando assim as soluções. Este problema foi contornado usando a estratégia descrita na seção 3.6. Lobo (2005) sugere a utilização de *backtracking* para eliminar este problema. Deve haver outras formas para tratar esse problema. Portanto, este operador merece um estudo mais aprofundado para encontrar a melhor forma de implementá-lo.

Conclui-se, portanto, que o presente trabalho é uma importante colaboração para a geração de horários da Unipampa e também pode servir como incentivo para a pesquisa científica na área do trabalho. Há a expectativa de que a aplicação desenvolvida seja útil aos

coordenadores acadêmicos já no próximo semestre. Apesar de o algoritmo não contemplar todas as restrições e peculiaridades da instituição, pode servir como um ponto inicial na confecção dos horários.

5.1 Trabalhos futuros

Como sugestão para trabalhos futuros, pode-se citar a melhoria do algoritmo genético adaptando-o para execução em paralelo. Assim, a geração dos horários seria de forma mais rápida.

Outra sugestão é a adaptação do algoritmo proposto ao modelo da Unipampa, para que a aplicação possa ser utilizada pela universidade como um todo, e não apenas pelos cursos de Ciência da Computação e Engenharia de Software do Campus Alegrete.

REFERÊNCIAS

AARTS, E. H. L.; KORST, J. **Simulated Annealing and Boltzmann Machines**. John Wiley & Sons, 1989.

ABRAMSON, D. **Constructing schools timetables using simulated annealing: sequential and parallel algorithms**. Management Science, 1(37):98-113, 1991.

ABRAMSON, D.; KRISHNAMOORTHY, M.; DANG, H. **Simulated annealing cooling schedules for the school timetabling problem**. Asia-Pacific Journal of Operational Research, 16:1-22, 1999.

APPLEBY, J. S.; BLACK, D. V.; NEWMAN, E. A. **Techniques for producting school timetables on a computer and their application to other scheduling problems**. The Computer Journal, 3:237-245, 1960.

BARBOZA, F. J. R.; FREITAS, C. C.; GUIMARÃES, P. R. B.; NETO, M. C. M. **Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar**. Sétima Escola Regional de Computação Bahia-Sergipe, 2007.

BARCELLOS, J. C. H. **Algoritmos genéticos adaptativos: Um estudo comparativo**. Escola Politécnica da Universidade de São Paulo, 2000.

CALDEIRA, J.; FERNANDES, C. **Infected genes evolutionary algorithm for school timetabling**. WSES, 2002.

CISCON, L. A. **O problema de geração de horários: um foco na eliminação de janelas e aulas isoladas**. Universidade Federal de Lavras, Lavras-MG, 2006.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. **Metaheuristics for high school timetabling**. Computational Optimization and Applications, 9:275-298, 1998.

CORLONI, A.; DORIGO, M.; MANIEZZO, V. **A genetic algorithm to solve the timetable problem**. Technical report, Politecnico di Miliano, Italy, 1992.

COSTA, D. **A tabu search algorithm for computing an operational timetable**. European Journal of Operational Research, 76:98-110, 1994.

COSTA, E. O.; DALLA BRUNA, M. **Resolução de timetabling utilizando evolução cooperativa**. Universidade Federal do Paraná, 2002.

COSTA, F. P.; GUIMARÃES, I. F. G.; SOUZA, M. J. F. **Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas**. XXII Encontro Nacional de Engenharia de Produção, 2002.

DAVIS, L. **Handbook of GeneticAlgorithms**. Van Nostrand Reinhold, 1991.

DOWSLAND, K. A. **Simulated annealing**. Blackwell Scientific Publications, pages 20-69, 1990.

EVEN, S.; ITAI, A.; SHAMIR, A. **On the complexity of timetabling and multi-commodity flow problems**. SIAM Journal of Computation, 5(4):691-703, 1976.

FEO, T. A.; REZENDE, M. G. C. **Greedy randomized adaptive search procedures**. Journal of Global Optimization, 6:109-133, 1992.

FERRREIRA, A. C. P. L. **Computação evolutiva**. Sistemas inteligentes - Fundamentos e Aplicações, 2003.

FOGEL, D. B. **System identification through simulated evolution: A machine learning approach to modeling**. Ginn Press, 1991.

GAREY, M. R.; JONHSON, D. S. **Computers and Intractability - A guide toNP-completeness**. W.H Freeman and Company, 1979.

GELATT JR, C. D.; KIRKPATRICK, S.; VECCHI, M. P. **Optimization by simulated annealing.** Science, 220:671-680, 1983.

GLOVER, F. **Tabu search**. ORSA Journal of Computing, 1:190-206, 1997.

GOLDBERG, D. E. **Genetic algorithms in search, optimization and machine learning**. The University of Alabama, 1989.

HERTZ, A. **Tabu search for large scale timetabling problems**. European Journal of Operational Research, 54(1):39-47, 1991.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. The University of Michigan Press, Ann Arbor, 1975.

JACOB JUNIOR, A. F. L; ROCHA, C. A. J. **Aghora: Algoritmos genéticos para geração de horários de aula**. II CTIC - Congresso de Tecnologia da Informação e Telecomunicações, 2005.

JUNGINGER, W. **Timetabling in germany - a survey**. Interfaces, 16:66-74, 1986.

KANEKO, K.; YAMANOUCHI, T.; YOSHIKAWA, M. **A constraint-based high school scheduling system**. IEEE Expert, 11(1):63-72, 1996.

KHANG, N.; KHON, T.; NGUYEN, D.; NUONG, T. **Using tabu search for solving a high school timetabling problem**. Studies in Computational Intelligence, 283:305-313, 2010.

KHANG, N.; NGA, L.; NGUYEN, D.; NUONG, T.; TUNG, P. **Simulated annealing-based algorithm for a real-world high school timetabling problem**. Second International Conference on Knowledge and Systems Engineering, pages 125-130, 2010.

LINDEN, R. **Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional**. Brasport, 2006.

LOBO, E. L. M. **Uma solução do problema de horário escolar via algoritmo genético paralelo**. CEFET-MG, 2005.

MARTINS, S. V. **Melhor horário para o sistema CEFET Campos: Um aplicativo para automatizar a elaboração de horários das aulas**. Vértices, 6:9-26, 2004.

MICHAELEWIZS, Z. **Genetic Algorithms + Data Structures = Evolution Programs**. Springer-Verlag, second, extended edition, 1994.

PATNAIK, L. M.; SRINIVAS, M. **Genetic algorithmics: A survey**. IEEE Computer Society, 27:17-26, 1994.

RIBEIRO, C. C. **Meta-heuristics and applications**. Advanced School on Artificial Intelligence. Estoril, Portugal, 1996.

RIBEIRO FILHO, G. **Melhoramentos do Algoritmo Genético Construtivo e Novas Aplicações em Problemas de Agrupamento**. INPE, São José dos Campos, 2000.

SCHAERF, A. **Tabu search techniques for large high school timetabling problems**. Proceedings of the 30th National Conference on Artificial Intelligence, 1:363-368,1996.

SCHAERF, A. **A survey of automated timetabling**. Artificial Intelligence Review,13:87-123, 1999.

SOUZA, M. J. F. **Programação de horários em escolas: uma aproximação por meta-heurísticas**. Programa de Engenharia de Sistemas e Computação, UFRJ, 2000.

TANOMARU, J. **Motivação, fundamentos e aplicações de algoritmos genéticos**. Anais do II Congresso Brasileiro de Redes Neurais, 1995.

TIMÓTEO, G. T. S. **Desenvolvimento de um algoritmo genético para a resolução do timetabling**. Universidade Federal de Lavras, Lavras-MG, 2002.

WETZEL, A. **Evaluation of the effectiveness of genetic algorithms in combinatorial optimization**. University of Pittsburg, 1983.

WERRA, D. **An introduction to timetabling**. European Journal of Operational Research, 19:151-162, 1985.

APÊNDICE

APÊNDICE A – Diagrama de classes e diagrama entidade-relacionamento do banco de dados

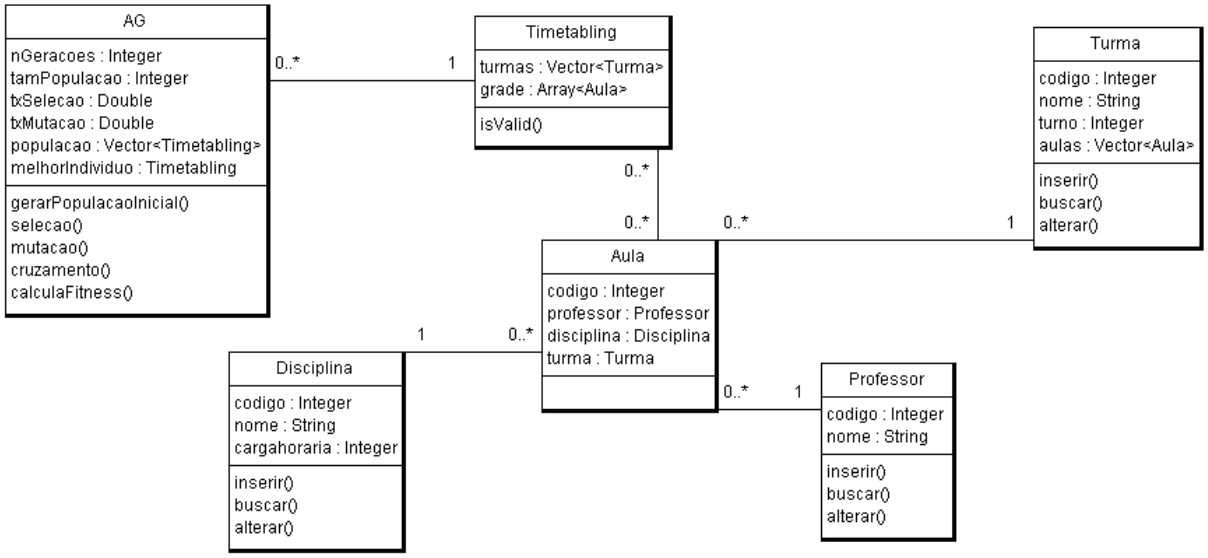

FIGURA 7.1 – Diagrama de classes UML.

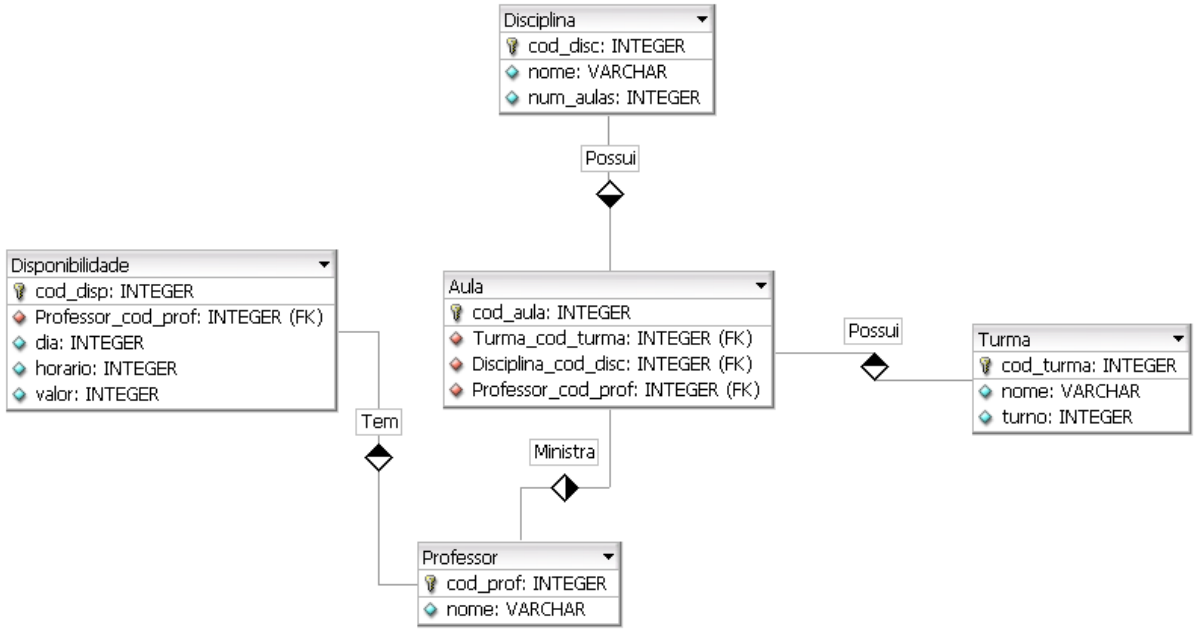

FIGURA 7.2 – Diagrama entidade-relacionamento do banco de dados implementado

APÊNDICE B – Preferências dos professores e melhor horário gerado na instância um

TABELA 7.1

Preferências dos professores – Instância um

Alecsandro	**S**	**T**	**Q**	**Q**	**S**	**S**	**Alex**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	1	2	3	**Aula 1**	1	1	1	1	3	2
Aula 2	1	1	1	1	2	3	**Aula 2**	1	1	1	1	3	2
Aula 3	1	1	1	1	2	3	**Aula 3**	1	1	1	1	3	2
Aula 4	1	1	1	1	2	3	**Aula 4**	1	1	1	1	3	2
Carlos Alberto	**S**	**T**	**Q**	**Q**	**S**	**S**	**Cássio**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	3	1	1	1	**Aula 1**	1	1	1	2	3	1
Aula 2	1	1	3	1	1	1	**Aula 2**	1	1	1	2	3	1
Aula 3	1	1	3	1	1	1	**Aula 3**	1	1	1	1	3	1
Aula 4	1	1	3	1	1	1	**Aula 4**	1	1	1	1	3	1
Cristian	**S**	**T**	**Q**	**Q**	**S**	**S**	**Daniel**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	1	2	1	**Aula 1**	1	1	3	1	1	1
Aula 2	1	1	1	1	2	1	**Aula 2**	1	1	3	1	1	1
Aula 3	1	1	1	1	2	1	**Aula 3**	1	1	3	1	1	1
Aula 4	1	1	1	1	2	1	**Aula 4**	1	1	3	1	1	1
Eder	**S**	**T**	**Q**	**Q**	**S**	**S**	**Emerson**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	3	1	2	**Aula 1**	1	3	1	1	1	1
Aula 2	1	1	1	3	1	2	**Aula 2**	1	3	1	1	1	1
Aula 3	1	1	1	3	1	2	**Aula 3**	1	3	1	1	1	1
Aula 4	1	1	1	3	1	2	**Aula 4**	1	3	1	1	1	1
Fabiano	**S**	**T**	**Q**	**Q**	**S**	**S**	**Fernanda**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	1	1	1	**Aula 1**	2	2	1	1	1	1
Aula 2	1	1	1	1	1	1	**Aula 2**	2	2	1	1	1	1
Aula 3	1	3	2	1	1	1	**Aula 3**	2	2	1	1	1	1
Aula 4	1	3	2	1	1	1	**Aula 4**	2	2	1	1	1	1
Figueroa	**S**	**T**	**Q**	**Q**	**S**	**S**	**Gabriele**	**S**	**T**	**Q**	**Q**	**S**	**S**

Aula 1	3	1	1	1	1	2	**Aula 1**	1	2	2	2	1	1
Aula 2	3	1	1	1	1	2	**Aula 2**	1	2	2	2	1	1
Aula 3	3	1	1	1	1	2	**Aula 3**	1	1	1	1	3	1
Aula 4	3	1	1	1	1	2	**Aula 4**	1	1	1	1	3	1
Jean	**S**	**T**	**Q**	**Q**	**S**	**S**	**João**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	1	2	1	**Aula 1**	3	1	1	1	1	3
Aula 2	1	1	1	1	2	1	**Aula 2**	3	1	1	1	1	3
Aula 3	1	1	1	1	2	1	**Aula 3**	3	1	1	1	1	3
Aula 4	1	1	1	1	2	1	**Aula 4**	3	1	1	1	1	3
Mara	**S**	**T**	**Q**	**Q**	**S**	**S**	**Paulo**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	1	1	1	**Aula 1**	1	1	1	1	2	1
Aula 2	1	1	1	1	1	1	**Aula 2**	1	1	1	1	2	1
Aula 3	1	3	3	1	1	1	**Aula 3**	1	1	1	1	2	1
Aula 4	1	3	3	1	1	1	**Aula 4**	1	1	1	1	2	1
Rodrigo	**S**	**T**	**Q**	**Q**	**S**	**S**	**Ronaldo**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	2	1	1	1	1	1	**Aula 1**	3	2	1	1	1	1
Aula 2	2	1	1	1	1	1	**Aula 2**	3	2	1	1	1	1
Aula 3	2	1	3	3	1	1	**Aula 3**	3	2	1	1	1	1
Aula 4	2	1	3	3	1	1	**Aula 4**	3	2	1	1	1	1
Willian	**S**	**T**	**Q**	**Q**	**S**	**S**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Aula 1	2	1	2	2	1	1	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Aula 2	2	1	2	2	1	1	**-**	-	-	-	-	-	-
Aula 3	2	1	1	1	1	1	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Aula 4	2	1	1	1	1	1	**-**	-	-	-	-	-	-

TABELA 7.2

Horário ‘ótimo’ gerado – Instância um

CC2	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Circ. Digitais Alecsandro	Est. de Dados I Fabiano	Prob. e Estatíst. Alex	Circ. Digitais Alecsandro	Cálculo II Fernanda	Álgebra Linear Carlos Alberto
Aula 2	Circ. Digitais Alecsandro	Est. de Dados I Fabiano	Prob. e Estatíst. Alex	Circ. Digitais Alecsandro	Cálculo II Fernanda	Álgebra Linear Carlos Alberto
Aula 3	Est. de Dados I Fabiano	Mat. Discreta João	Mat. Discreta João	Prob. e Estatíst. Alex	Álgebra Linear Carlos Alberto	Cálculo II Fernanda
Aula 4	Est. de Dados I Fabiano	Mat. Discreta João	Mat. Discreta João	Prob. e Estatíst. Alex	Álgebra Linear Carlos Alberto	Cálculo II Fernanda
CC4	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Prog. OO Daniel	B. de Dados I Willian	Proj. e Análise Fabiano	Prog. OO Daniel	Arq de Comp. I Gabriele	- -
Aula 2	Prog. OO Daniel	B. de Dados I Willian	Proj. e Análise Fabiano	Prog. OO Daniel	Arq de Comp. I Gabriele	- -
Aula 3	Prog. p/Web Cássio	Prog. p/Web Cássio	B. de Dados I Willian	Arq de Comp. I Gabriele	Proj. e Análise Fabiano	- -
Aula 4	Prog. p/Web Cássio	Prog. p/Web Cássio	B. de Dados I Willian	Arq de Comp. I Gabriele	Proj. e Análise Fabiano	- -
CC6	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Proc. de Ling. Fabiano	Eng. Software Eder	Sist. Tempo Real Figueroa	Compiladores João	Compiladores João	Proc. de Ling. Fabiano
Aula 2	Proc. de Ling. Fabiano	Eng. Software Eder	Sist. Tempo Real Figueroa	Compiladores João	Compiladores João	Proc. de Ling. Fabiano
Aula 3	Redes Gabriele	Sist. Tempo Real Figueroa	Redes Gabriele	Met. Pesq. Cient. Cristian	Eng. Software Eder	Met. Pesq. Cient. Cristian
Aula 4	Redes Gabriele	Sist. Tempo Real Figueroa	Redes Gabriele	Met. Pesq. Cient. Cristian	Eng. Software Eder	Met. Pesq. Cient. Cristian
CC8	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Sist. Informação Cristian	Tec. Cont. Social Mara	- -	- -	- -	- -
Aula 2	Sist. Informação Cristian	Tec. Cont. Social Mara	- -	- -	- -	- -
Aula 3	Tec. Cont. Social Mara	Sist. Informação Cristian	- -	- -	- -	- -

Aula 4	Tec. Cont. Social Mara	Sist. Informação Cristian	- -	- -	- -	- -
ES2	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	R. Problemas II Eder	R. Problemas II Daniel	Prog. OO Cássio	Prob. e Estatíst. Rodrigo	T. A. de Dados Ronaldo	Prog. OO Cássio
Aula 2	R. Problemas II Eder	R. Problemas II Daniel	Prog. OO Cássio	Prob. e Estatíst. Rodrigo	T. A. de Dados Ronaldo	Prog. OO Cássio
Aula 3	Teste/Depuração Daniel	T. dos Grafos Paulo	R. Problemas II Eder	R. Problemas II Paulo	Prob. e Estatíst. Rodrigo	T. A. de Dados Ronaldo
Aula 4	Teste/Depuração Daniel	T. dos Grafos Paulo	R. Problemas II Eder	R. Problemas II Paulo	Prob. e Estatíst. Rodrigo	T. A. de Dados Ronaldo
ES4	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Análise de SW. Mara	Int. Análise Algo Cássio	R. Problemas IV Jean	R. Problemas IV Mara	Int. Sist. Comp. Emerson	Práticas em BD Willian
Aula 2	Análise de SW. Mara	Int. Análise Algo Cássio	R. Problemas IV Jean	R. Problemas IV Mara	Int. Sist. Comp. Emerson	Práticas em BD Willian
Aula 3	R. Problemas IV Jean	V. e Validação Jean	Int. Sist. Comp. Emerson	V. e Validação Jean	Práticas em BD Willian	R. Problemas IV Jean
Aula 4	R. Problemas IV Jean	V. e Validação Jean	Int. Sist. Comp. Emerson	V. e Validação Jean	Práticas em BD Willian	R. Problemas IV Jean

APÊNDICE C – Preferências dos professores e horário gerado na instância dois

TABELA 7.3

Preferências dos professores – Instância dois

Cláudio	**S**	**T**	**Q**	**Q**	**S**	**S**	**Julia**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	3	1	2	1	1	1	**Aula 1**	1	1	1	2	2	1
Aula 2	3	1	2	1	1	1	**Aula 2**	1	1	1	2	2	1
Aula 3	3	1	2	1	1	1	**Aula 3**	3	1	1	1	1	1
Aula 4	3	1	2	1	1	1	**Aula 4**	3	1	1	1	1	1
Luiz	**S**	**T**	**Q**	**Q**	**S**	**S**	**Marcos**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	2	1	1	1	**Aula 1**	1	1	1	1	2	2
Aula 2	1	1	2	1	1	1	**Aula 2**	1	1	1	1	2	2
Aula 3	1	1	2	1	1	1	**Aula 3**	1	1	1	1	1	2
Aula 4	1	1	2	1	1	1	**Aula 4**	1	1	1	1	1	2
Ronaldo	**S**	**T**	**Q**	**Q**	**S**	**S**	**Simão**	**S**	**T**	**Q**	**Q**	**S**	**S**
Aula 1	1	1	1	3	1	1	**Aula 1**	1	1	2	1	1	3
Aula 2	1	1	1	3	1	1	**Aula 2**	1	1	2	1	1	3
Aula 3	1	1	1	3	1	1	**Aula 3**	1	1	2	1	1	3
Aula 4	1	1	1	3	1	1	**Aula 4**	1	1	2	1	1	3

TABELA 7.4

Horário gerado com valor de valor de avaliação = 110 – Instância dois

CC1	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
Aula 1	Cálculo I Marcos	Int. Ciência e T. Julia	Cálculo I Marcos	G. Analítica Luiz	Eletrotécnica Ronaldo	Eletrotécnica Ronaldo
Aula 2	Cálculo I Marcos	Int. Ciência e T. Julia	Cálculo I Marcos	G. Analítica Luiz	Eletrotécnica Ronaldo	Eletrotécnica Ronaldo
Aula 3	G. Analítica Luiz	Lógica Mat. Simão	Algoritmos Cláudio	Lógica Mat. Simão	Algoritmos Cláudio	Int. Ciência e T. Julia
Aula 4	G. Analítica Luiz	Lógica Mat. Simão	Algoritmos Cláudio	Lógica Mat. Simão	Algoritmos Cláudio	Int. Ciência e T. Julia
CC3	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Acessibilidade Julia	OAD Luiz	- -	Estr. de Dados II Simão	Estr. de Dados II Simão	Acessibilidade Julia
Aula 2	Acessibilidade Julia	OAD Luiz	- -	Estr. de Dados II Simão	Estr. de Dados II Simão	Acessibilidade Julia
Aula 3	Arq. e Org. de C. Marcos	Com. de Dados Cláudio	Arq. e Org. de C. Marcos	Com. de Dados Cláudio	OAD Luiz	- -
Aula 4	Arq. e Org. de C. Marcos	Com. de Dados Cláudio	Arq. e Org. de C. Marcos	Com. de Dados Cláudio	OAD Luiz	- -
CC5	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	PLP Simão	Ling. Formais Marcos	Int. Artificial Julia	Sist. Operac. Claudio	Eng. de SW Luiz	Sist. Operac. Claudio
Aula 2	PLP Simão	Ling. Formais Marcos	Int. Artificial Julia	Sist. Operac. Claudio	Eng. de SW Luiz	Sist. Operac. Claudio
Aula 3	Desaf. de Progr. Ronaldo	Int. Artificial Julia	Desaf. de Progr. Ronaldo	Ling. Formais Marcos	PLP Simão	Eng. de SW Luiz
Aula 4	Desaf. de Progr. Ronaldo	Int. Artificial Julia	Desaf. de Progr. Ronaldo	Ling. Formais Marcos	PLP Simão	Eng. de SW Luiz
CC7	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**
Aula 1	Int. ao Proc. Img Luiz	Compiladores Simão	- -	Sist. Distrib. Marcos	B. de Dados II Cláudio	Int. ao Proc. Img Luiz
Aula 2	Int. ao Proc. Img Luiz	Compiladores Simão	- -	Sist. Distrib. Marcos	B. de Dados II Cláudio	Int. ao Proc. Img Luiz
Aula 3	Compiladores Simão	Sist. Distrib. Marcos	Adm. e Empre. Julia	Adm. e Empre. Julia	- -	B. de Dados II Cláudio
Aula 4	Compiladores Simão	Sist. Distrib. Marcos	Adm. e Empre. Julia	Adm. e Empre. Julia	- -	B. de Dados II Cláudio