UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO GERADOR DE GRADES HORÁRIAS PARA INSTITUIÇÕES DE ENSINO

THOMÁS AUGUSTO PREIS

BLUMENAU
2007

2007/2-34

THOMÁS AUGUSTO PREIS

PROTÓTIPO GERADOR DE GRADES HORÁRIAS PARA INSTITUIÇÕES DE ENSINO

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Orientador: Prof. Dr. Mauro Marcelo Mattos

BLUMENAU
2007

2007/2-34

PROTÓTIPO GERADOR DE GRADES HORARIAS PARA INSTITUIÇÕES DE ENSINO

Por

THOMÁS AUGUSTO PREIS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos, Doutor – Orientador, FURB
Membro:	Prof. Maurício Capobianco Lopes, Mestre – FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB

Blumenau, 27 de novembro de 2007

Dedico este trabalho a todos aqueles que ajudaram a fazê-lo acontecer.

AGRADECIMENTOS

Agradeço a Deus por não me deixar na mão e por colocar tantas pessoas especiais ao meu redor.

À minha família, que mesmo tão distante, sempre esteve tão perto de mim. A meu irmão Thiago que me incentivou a não fazer este curso, porém, o casca dura aqui o fez. Ao meu pai Vânio e minha mãe Ana pelos ensinamentos e confiança depositados em mim durante todos estes anos. Pessoas as quais tanto admiro, amo e me espelho.

À minha namorada Fernanda, pelo incentivo, confiança, companheirismo e paciência com esta pessoa teimosa.

Aos meus amigos que sempre estiveram do meu lado mesmo nos momentos mais difíceis ou distantes por conta da produção deste trabalho.

À Dynamix, empresa onde pude aprender muito e colocar em prática os conhecimentos adquiridos em todos estes anos de graduação.

Ao meu orientador Mauro Marcelo Mattos, pessoa que sempre esteve disponível para me auxiliar no desenvolvimento deste trabalho.

Na prática, a teoria é outra.

Thomás Augusto Preis

RESUMO

Problemas na geração de grades horárias são freqüentes em instituições de ensino. Através do uso de algoritmos genéticos é possível solucionar tais problemas. Os algoritmos genéticos utilizam conceitos provenientes da biologia e contemplam uma ampla série de problemas. Robustos, genéricos e facilmente adaptáveis, eles são largamente estudados e utilizados em diversas áreas. O presente trabalho tem como objetivo demonstrar uma aplicação prática desses algoritmos, estando dividido em duas partes: uma abordagem teórica sobre algoritmos genéticos e a criação de uma interface web para a geração de grades horárias com base no modelo de horários utilizado pela Universidade Regional de Blumenau.

Palavras-chave:Algoritmos genéticos . Inteligência artificial. *Timetabling*.

ABSTRACT

Problems related to the generation of timetables are frequent in educational institutions; through the use of genetic algorithms it is possible to solve those problems. The genetic algorithms make use of concepts that come from Biology and approach a wide number of problems. Robust, generic and easily adaptable, they are widely studied and used in many different areas. The present work aims at demonstrating a practical application of those algorithms, being divided in two parts: a theoretical approach on genetic algorithms and the creation of a web interface for the generation of timetables based on the time model used by Regional University of Blumenau (FURB).

Keywords: Genetic algorithms. Artificial inteligence. *Timetabling*.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AG - Algoritmo genético

AGs - Algoritmos genéticos

CCS - Code Charge Studio

CE - Computação Evolucionária

FURB - Fundação Universidade Regional de Blumenau

IA - Inteligência Artificial

RF - Requisitos Funcionais

RNF - Requisitos Não Funcionais

SGBD - Sistema Gerenciador de Banco de Dados

SUMÁRIO

1 INTRODUÇÃO

A construção de grades horárias é um problema enfrentado corriqueiramente por pequenas e principalmente por grandes instituições de ensino. Problema que ocorre devido a alguns fatores, dentre os quais podem ser citados a disponibilidade de horários de cada professor, salas de aula e número de disciplinas. Freqüentemente é uma tarefa difícil juntar todos esses elementos sem que haja conflitos nos horários. Esta tarefa pode tornar-se ainda mais árdua conforme o aumento do número de disciplinas e restrições dos professores.

O problema é potencializado quando o sistema permite que os professores informem suas disponibilidades para ministrarem aulas, pois se torna ainda mais complexo efetuar uma boa distribuição dos horários, de forma que seja interessante a todos, respeitando as restrições de cada um.

Observam-se muitos estudos voltados à análise dessa questão e à busca de soluções no meio computacional. Conforme Alvarenga et al. (2005), problemas na construção de horários acadêmicos são conhecidos na literatura como *timetabling*. Para essa questão, não existe uma solução genérica, a qual sempre possa ser aplicada para alcançar o resultado esperado: cada situação requer uma solução específica.

Michalewicz (1996 apud COSTA; BRUNA, 2003, p. 1) afirma que o *timetabling* pode ser considerado como um dos problemas mais interessantes da pesquisa operacional. A questão tem sido profundamente estudada e vem sendo considerada de difícil solução sob o ponto de vista computacional, isto devido ao grande número de restrições e variáveis de diferentes tipos a serem levadas em consideração e atendidas para obter-se uma solução ótima.

> O problema de alocação de horários há tempos é amplamente discutido e estudado, principalmente para instituições de ensino, onde conciliar, manualmente, disponibilidade entre professores e classes (turmas) é tarefa difícil, que demanda tempo e nem sempre se consegue resolver sem conflitos de disponibilidades das partes envolvidas. Também conhecido como "timetabling", o problema de alocação de horários escolar é assunto abordado por muitos autores que propõem uma variedade de soluções e de restrições. (SCHAEFER, 1999 apud ALVARENGA et al., 2005).

Inúmeros algoritmos podem auxiliar a solucionar problemas de *timetabling.* Conforme Sorroche (2002, p. 84-85), é possível citar: algoritmos genéticos, algoritmo guloso, algoritmo de Bron e Kerbosh, *Simulated Anneling* e *Tabu Search*.

Estudando-se de forma mais aprofundada alguns algoritmos que se propõem a solucionar problemas de *timetabling*, observou-se que não há um algoritmo mais ou menos

apropriado para se chegar ao resultado esperado. Cada algoritmo, em cada caso, apresenta particularidades, dentre as quais: desempenho, complexidade, capacidade de exploração de seu espaço de busca e outros. O presente trabalho explora a aplicação de busca com Algoritmos Genéticos (AG) aplicados na resolução de problemas de*timetabling* .

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma aplicação para elaborar sugestões de grades horárias de disciplinas para a Universidade Regional de Blumenau (FURB), utilizando algoritmos genéticos.

1.1.1 Objetivos específicos

Os objetivos específicos do presente trabalho são:

a) utilizar um algoritmo de alocação para a distribuição de disciplinas na grade de horários[1];
b) disponibilizar uma interface web para a aplicação.

1.2 ESTRUTURA DO TRABALHO

O trabalho está divido em quatro capítulos. No capitulo dois são apresentadas as principais características de problemas de timetabling, também são apresentados os trabalhos correlatos.

No capítulo três é descrita a fundamentação teórica utilizada como embasamento para o presente trabalho. São abordados os seguintes assuntos: história dos algoritmos genéticos, computação evolucionária, bem como são descritas características gerais, componentes e etapas necessárias para a construção de um AG, frisando-se a importância de cada uma dessas

1 Não são levados em consideração os aspectos: distribuição de salas, laboratórios e outros recursos.

partes. Além disso, são apresentadas algumas aplicações nas quais podem ser utilizados AGs.

O capítulo quatro traz a especificação do protótipo. São apresentados os requisitos, diagramas, modelagem de dados, ferramentas e técnicas utilizadas para a construção do protótipo. Também é descrita a forma utilizada para a construção do AG e da interface web. O capitulo é encerrado com uma abordagem a respeito das dificuldades encontradas e soluções adotadas na construção do trabalho.

O capítulo final contém a conclusão do trabalho, juntamente com sugestões para melhoramentos e trabalhos futuros.

2 TIMETABLING

Neste capítulo apresenta-se uma abordagem sobre problemas de *timetabling*. Primeiramente é feita uma introdução e então são apresentadas características dos problemas de *timetabling*, o capítulo é finalizado com a apresentação de alguns trabalhos correlatos.

2.1 INTRODUÇÃO

Problemas na geração de quadros de horários são conhecidos na literatura como problemas de *timetabling*. São problemas complexos de otimização e possuem diversas aplicações práticas, como por exemplo: escalonamento de enfermeiros, horários de aulas em instituições de ensino, planejamento de transporte público e outros (FREITAS et Al., 2006).

O processo de alocação de horário escolar é denominado *timetabling* escolar e constitui um problema de otimização combinatória da classe NP - Difíceis, no qual é mais fácil verificar se o resultado apresentado está correto do que obter analiticamente tal solução em tempo polinomial (BARBOZA, 2003 apud FREITAS et Al., 2006). Este problema depende do tipo de sistema educacional utilizado, cada instituição possui seu próprio padrão para montar as grades horárias, desse modo, não existe um modelo solução universal que possa sempre ser aplicado em qualquer sistema educacional e que resolva os problemas com a grade horária. (ALVARENGA et Al., 2005).

2.2 CARACTERÍSTICAS

Constantino (2003) afirma que problemas de *timetabling* são comuns em todas as instituições de ensino e consistem na construção de horários de aulas para os docentes e alocação de salas de aulas para as disciplinas, atendendo às restrições impostas tanto por professores, quanto por alunos e pela instituição envolvida. Normalmente, as restrições estão relacionadas a conflitos de horários de aula, alocação de salas de aula mais adequadas para cada disciplina e/ou turma, além de preferências de horário dos professores.

Por se tratar de um problema constante em instituições de ensino, com o passar dos anos foram criados diversos modelos de algoritmo que visam facilitar a resolução dessa classe de problemas, alguns dos quais já foram citados anteriormente. Porém a implementação desses métodos é, via de regra, extremamente complexa. Schoeffel (2001, p. 16) diz que com um conjunto razoável de dados de entrada, é impossível explorar todas as soluções possíveis. Com o aumento do número de disciplinas, professores e cursos, torna-se mais difícil e complexa a implementação de uma aplicação que construa uma grade horária sem que haja janelas e colisões nos horários, tanto para os alunos, quanto para os professores (BRAZ, 2000, p. 24-26).

A solução desse tipo de problema consiste em gerar uma tabela de horários que vise a minimizar ou eliminar conflitos, levando em consideração preferências ou prioridades na alocação dos recursos (CONSTANTINO, 2003). Os métodos que auxiliam na resolução de problemas de *timetabling* fazem uso do modelo de árvores de busca e utilizam heurísticas[2] para limitar a busca. Devido ao grande número de variáveis e restrições envolvidas, se não existisse essa limitação, seria praticamente impossível explorar todo o espaço de busca. Deste modo, é praticamente impossível garantir que sempre será obtido o mesmo resultado final e uma solução ótima (CAMPOS, 2002, p. 11).

2.3 TRABALHOS CORRELATOS

Trabalhos correlatos ao desenvolvimento deste projeto são descritos a seguir, os quais estão relacionados a algoritmos genéticos e/ou problemas de timetabling.

Sorroche (2002) desenvolveu em seu TCC na FURB o Sistema de auxílio à matrícula de alunos utilizando Java 2 Enterprise Edition que visa a auxiliar o processo de matrícula de alunos na instituição de ensino. O sistema desenvolvido leva em consideração a opção do aluno, ou em montar uma grade horária atingindo o número mínimo de créditos a pagar, ou em fazer a alocação de horários com o máximo de disciplinas possível. Essa alocação leva em consideração disciplinas relevantes à situação acadêmica do aluno, verificando, por exemplo, pré-requisitos e conflitos nos horários.

Neste trabalho foram adotados dois algoritmos para fazer sugestão de horários, o

2 Heurísticas são métodos exploratórios que baseiam-se em experiências para buscar um melhor resultado final.

algoritmo proposto por Bron e Kerbosh e o algoritmo Guloso. Ambos apresentaram soluções satisfatórias na resolução do problema, porém o algoritmo de Bron e Kerbosh apresentou um tempo de resposta melhor em relação ao Guloso. Em testes realizados com 52 disciplinas, ambos os algoritmos conseguiram montar sugestões de reserva de vaga com o mínimo de créditos em menos de dois segundos.

Campos (2002) criou um sistema chamado de Algoritmo genético na resolução do problema da grade horária, o protótipo serviu como modelo para seu projeto final do curso de Ciência da Computação pela Universidade Federal da Bahia. O protótipo foi desenvolvido em Delphi, possui um cadastro de disciplinas e professores, onde também é possível cadastrar as preferências de horários dos professores. Através destes dados de entrada, o protótipo tenta montar uma grade horária, respeitando as restrições impostas e eliminando conflitos nos horários.

O protótipo possui algumas limitações e problemas, dentre as quais:

a) um professor por disciplina: não é possível cadastrar mais de um professor para uma mesma disciplina;
b) turma única por disciplina: o protótipo não possui opção de cadastrar mais de uma turma para uma mesma disciplina, por exemplo: Programação 1 – Turma A e Programação 1 – Turma B;
c) colisão nos horários: o protótipo não leva em consideração colisões de horários de disciplinas de um mesmo semestre;
d) problemas com carga horária: o protótipo possui alguns problemas relacionados à carga horária das disciplinas. Como não é feita a flexibilização nos horários, pode haver dias da semana com aulas de uma única disciplina.

Apesar de apresentar diversos problemas no protótipo, a abordagem teórica sobre algoritmos genéticos feita por Campos (2002) é extremamente relevante para a construção do presente trabalho.

Braz (2000) desenvolveu o sistema chamado de Otimização de horários em instituições de ensino superior através de algoritmos genéticos. A aplicação serviu de modelo para sua dissertação de mestrado na Universidade Federal de Santa Catarina (UFSC). O protótipo tem por objetivo a resolução de problemas relacionados à geração de grades de horárias. São utilizadas abordagens heurísticas, implementando o modelo de algoritmos genéticos.

O protótipo leva em consideração diversas questões relevantes presentes nos problemas de *timetabling*, tais como: alocação de salas de aulas, laboratórios, preferências de horário dos professores e espaço físico, propondo-se a distribuir os horários de diversos

cursos e campus fisicamente distantes. O protótipo mostrou-se eficaz ao ser submetido a testes reais na geração de grades horárias para instituições de ensino, evidenciando, assim, a aplicabilidade do modelo em casos reais.

No meio comercial, tem-se o Urânia (GEHA, 2007). O Urânia é um software que visa à criação de grades horárias para professores e disciplinas. Utilizado por diversas instituições de ensino do Brasil, o seu desenvolvimento teve início em 1986. Dentre suas principais características pode-se citar:

a) determinar horários: é possível determinar os horários em que cada professor poderá ministrar suas aulas;
b) disposição dos horários: através do software é possível determinar como serão dispostas as turmas durante as semanas, como por exemplo: aulas separadas, germinadas, só uma aula por dia e outros;
c) divisão de classes: a ferramenta possui opções de dividir as turmas para trabalharem com professores diferentes;
d) aulas especiais: possui opções para controlar a utilização de laboratórios, auditórios e outros;
e) controle de sedes: é possível fazer um controle dos horários dos professores respeitando a existência de várias sedes para a instituição, é levado em consideração o tempo de deslocamento dos professores de uma sede para outra;
f) limitar horas-aula: também existe a opção de fazer a limitação de horas-aula por dia, por professor;
g) eliminação de janelas: a ferramenta garante em média uma redução de 30% nas janelas de horários vagos.

Observando-se as principais características dessa ferramenta, pode-se notar que ela oferece uma variedade de recursos interessantes na resolução de problemas de timetabling. Porém, não são especificadas pelo fabricante características como: técnicas utilizadas na resolução dos problemas e desempenho do software.

3 ALGORITMOS GENÉTICOS

Neste capítulo são apresentados alguns aspectos teóricos relacionados com o trabalho, tais como: programação evolucionária, características, componentes e funcionamento de um algoritmo genético. No final do capítulo são apresentadas aplicações que utilizam algoritmos genéticos.

3.1 INTRODUÇÃO

A teoria da evolução das espécies foi proposta em 1859, pelo britânico Charles Darwin. Darwin estudou profundamente o processo de seleção natural, notou que nem todos os seres vivos nascem, sobrevivem e se reproduzem. Indivíduos mais bem dotados ou adaptados têm maiores chances de sobreviver e se reproduzirem. Levando dessa forma, à evolução da espécie, passando para as próximas gerações características herdadas dos pais e aumentando o grau de adaptação com o meio (BARRETO, 2007).

Baseando-se na teoria da evolução das espécies John H. Holland em meados da década de 60 apresentou um algoritmo baseado em um processo evolutivo, semelhante à teoria da evolução das espécies, onde novos conjuntos de soluções são criados até que certas condições sejam atingidas, em outras palavras, buscam soluções mais bem adaptadas. Este processo evolutivo, onde estruturas são manipuladas por modificadores em busca de estruturas mais bem adaptadas é denominada por Holland como *The envoriomento of the system undergoing adaptation* - o ambiente do sistema submetido à adaptação (HOLLAND, 1992 apud PERSONA; CORRÊA; SARAIVA, 2003).

Holland sugeriu a simulação da evolução natural por meio de computadores, através da criação de técnicas semelhantes aos processos de adaptação e seleção na natureza, com o objetivo de resolver determinados problemas. Para isso é preciso ter em mente que cromossomos representam possíveis soluções. As evoluções destes cromossomos auxiliam na busca de uma solução possível para o problema proposto (BARRETO, 2007).

Conforme Barreto (2007), os AGs fazem parte do ramo de computação evolucionária pertencente ao ramo da Inteligência Artificial (IA), estão em um lugar de destaque nos paradigmas da Computação Evolucionária (CE) devido a alguns fatores, dentre os quais

podem-se citar:

a) resultados aceitáveis com melhor custo – beneficio: devido à precisão e recursos utilizados quando comparado a diversos métodos tradicionais os AGs tem um melhor custo – beneficio;
b) altamente flexíveis: é possível de forma fácil alterar a implementação do algoritmo com o objetivo de aumentar a gama de hipóteses interessantes a serem analisadas, afim de obter a convergência para a solução desejável;
c) exigem menor conhecimento especifico do problema: tomando-se por base outros algoritmos da CE, os AGs são os que exigem menor conhecimento especifico do problema em questão para seu funcionamento, o que os torna bem versáteis, no sentido de poderem ser utilizados em diversas áreas.

Os AGs estão entre os paradigmas mais utilizados dentre os paradigmas da CE, juntamente com Lógica de Fuzzy e Redes Neurais Artificiais. Fazem uso de vários conceitos existentes na biologia, tais como: cromossomos, genes, alelos, reprodução, mutação, adaptações e outros. Através da codificação destes conceitos em rotinas computacionais, é possível simular um processo de evolução (BARRETO, 2007).

3.2 CARACTERÍSTICAS GERAIS

AGs são algoritmos de otimização global, baseados em mecanismos da seleção natural e genética. Definem uma busca paralela, estruturada e aleatória com o objetivo de encontrar uma melhor aptidão[3] (CARVALHO, 2004). Apesar de aleatórios, os AGs combinam elementos de busca e trabalham com várias possíveis soluções, ao invés de processarem um único ponto no espaço de busca a cada instante. Isso permite explorar um leque maior de possíveis soluções. (COSTA; BRUNA, 2003).

Conforme Barreto (2007), não há uma definição rigorosa para caracterizar um AG, porém há alguns elementos em comum que um AG deve possuir, dentre os quais pode-se citar: uma população de cromossomos, operadores de seleção conforme a aptidão dos cromossomos, operadores de cruzamento para gerar novos membros na população e operadores de mutações para os novos membros da população.

---

3 Melhor aptidão pode ser descrito como uma possível solução para o problema proposto.

Conforme Lucas (2002, p. 6-7) as principais características dos AGs são:

a) busca codificada: os AGs não trabalham sobre o domínio do problema e sim sobre representações de seus elementos, o que impõe o uso de restrições para a resolução do problema. É necessário um conjunto de soluções viáveis de forma que essas possam ser codificadas em uma população de indivíduos ou cromossomos;
b) generalidade: os AGs simulam a natureza em um de seus maiores e mais fortes atributos que é a adaptabilidade. As representações de soluções dependem do conhecimento do domínio do problema. Bastam alterações dessas representações para que se representem diferentes soluções. A preocupação do programador não é chegar à solução de qual forma, mas sim estabelecer com o quê ela deve parecer;
c) paralelismo explícito: o paralelismo pode ser facilmente identificado, pois cada indivíduo da população é isolado e avaliado de forma independente;
d) busca estocástica: ao contrário de métodos tradicionais que buscam valores ótimos, os AGs não são determinísticos, porém sua busca não se dá de forma totalmente aleatória pois indivíduos com maior aptidão possuem maiores probabilidades de se reproduzirem;
e) busca cega: um AG ignora o significado das estruturas que manipula e qual é a melhor forma de se trabalhar sobre estas. Isso determina que um AG não necessite de um conhecimento específico do domínio do problema;
f) eficiência mediana: por ser uma busca cega, os AGs tendem a apresentar um desempenho inferior a alguns tipos de buscas heurísticas orientadas ao problema, porém, para solucionar tal desvantagem, é utilizada a tática de hibridização;
g) facilidade no uso de restrições: ao contrário de diversos métodos de busca, os AGs facilitam a implementação de restrições, mesmo que elas apresentem diferentes graus de importância. Nesse caso, se dois indivíduos violam restrições, é considerado mais apto o indivíduo que violou menos restrições ou restrições com menor grau de importância.

Conforme Liden (2006, p. 44), os AGs, como já citado anteriormente, possuem características herdadas da biologia. O Quadro 1 apresenta termos relacionados ao campo das ciências biológicas e seus correspondentes utilizados em computação.

Linguagem natural	Algoritmos genéticos
Cromossomo	Individuo, *string*, cromossomo, árvore
Gen	Característica
Alelo	Valor
Lócus	Posição
Genótipo	Estrutura
Fenótipo	Conjunto de parâmetros

Fonte: Liden (2006, p. 44).

Quadro 1 – Componentes de um algoritmo genético

Na próxima seção são apresentados os passos necessários para a construção de um algoritmo básico, mostrando-se a importância de cada uma das partes.

3.3 REPRESENTAÇÃO

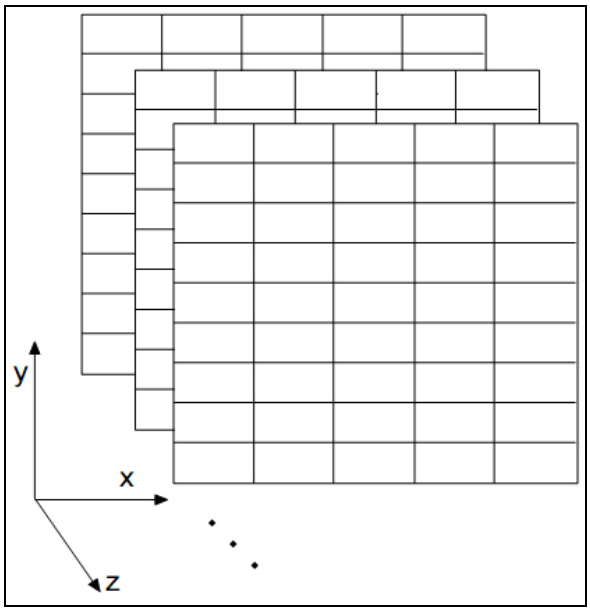
Lucas (2000, p. 40) afirma que a escolha de uma boa representação dos indivíduos é um ponto crucial para os AGs. A representação é um dos fatores mais importantes no seu desenvolvimento e posterior desempenho. Ao optar por uma representação, é necessário avaliar as seguintes características (SERRADA, 1996 apud LUCAS, 2000, p. 40-41):

a) completude: determina se é possível representar todos os fenótipos possíveis e necessários para a resolução do problema;
b) coerência: indica se a partir do esquema de representação é possível gerar um genótipo que represente um fenótipo não pertencente às possíveis soluções do problema;
c) uniformidade: em uma representação uniforme, o número de genótipos correspondentes deve ser o mesmo para todo fenótipo;
d) simplicidade: indica o grau de complexidade dos atos de codificação e decodificação dos cromossomos;
e) localidade: pequenas alterações no genótipo acarretam pequenas alterações em seu fenótipo correspondente.

Considerando as características apresentadas, pode-se fazer uma avaliação de dois possíveis esquemas de representação para solucionar um problema de grade horária (LUCAS, 2000, p. 40).

A representação de *time slots*, é a representação mais intuitiva na qual cada gene representa um *time slot*, por exemplo: disciplina de Inteligência Artificial, segunda-feira, às 10h 30 min, em um determinado semestre. Um genoma seria uma matriz de duas ou três

dimensões, sendo uma dimensão destinada a representar os horários da semana e outra representando o semestre. Apresenta a característica de não permitir, graças a sua estrutura, que duas disciplinas do mesmo semestre ocupem o mesmo horário. Porém, não garante que não haverá colisões de professor (LUCAS, 2000, p. 41). A Figura 5 mostra a representação deste possível modelo de grade horária.

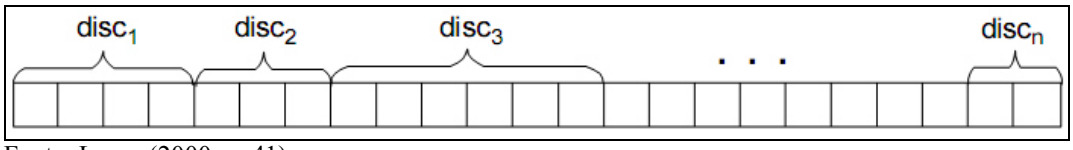


Fonte: Lucas (2000, p. 41).

Figura 1 – Representação de grade *time slots*

A Figura 1 representa uma grade horária onde `x` designa o dia, `y` um determinado horário e `z` o semestre ao qual pertence à grade horária.

No que diz respeito à representação de horas-aula, o genoma representa uma união de todas as horas-aula de todas as disciplinas. Para simplificar a implementação, as horas-aula da mesma disciplina são alocadas em blocos adjacentes, que, por sua vez, são alocados ordenadamente conforme o semestre ao qual pertencem. Apesar dessa representação permitir todos os tipos de colisão em sua codificação, essa forma de representação possui a vantagem considerável de sempre ter como solução horários que respeitam a carga horária de todas as disciplinas (LUCAS, 2000, p. 41). A Figura 2 apresenta o modelo da estrutura de horas-aula.



Fonte: Lucas (2000, p. 41).

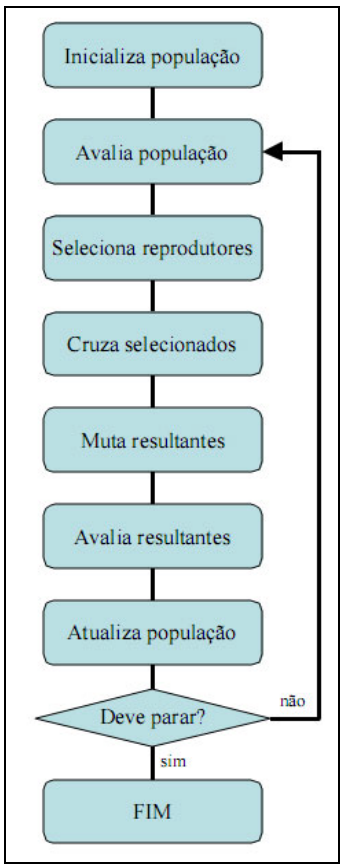
Figura 2 – Representação de horas-aula

A possível solução apresentada na Figura 2, do ponto de vista estrutural, é mais simples do que a grade *time slots*, porém é necessário maior atenção do programador para garantir que não haja nenhuma colisão de horários.

Conforme Braz (2000, p. 2), a representação de um problema de AGs passa a ser definida através de uma cadeia de caracteres ou bits, chamada de cromossomo. A escolha de uma boa representação pode facilitar o trabalho do programador no momento da codificação do AG (LUCAS, 2000, p. 40).

3.4 FUNCIONAMENTO

Nesta seção são apresentadas as etapas necessárias para a construção de um AG. E identificadas a importância e funcionalidade de cada um dos passos. A figura abaixo representa um fluxograma contendo os passos necessários para a construção do AG:



Fonte: Campos (2002, p. 4)
Figura 3 - Fluxograma de um algoritmo genético básico

Conforme pode ser observado na Figura 3, o algoritmo começa com a inicialização da população, o que se resume à criação de uma população de cromossomos sob a qual serão aplicadas as ações subseqüentes. Tradicionalmente fazem uso de funções aleatórias para gerar os indivíduos, este recurso aleatório visa a fornecer uma maior biodiversidade (LUCAS, 2002, p. 11).

Quanto maior for a biodiversidade da população, maiores serão as chances de se chegar a uma solução aceitável, visto que, com isso, o espaço de busca explorado será maior. Criando-se uma população inicial com mais membros, é possível garantir uma boa biodiversidade, porém isso pode ocasionar uma queda significativa no desempenho do algoritmo, fator relevante em diversos casos (LUCAS, 2002, p. 10-11).

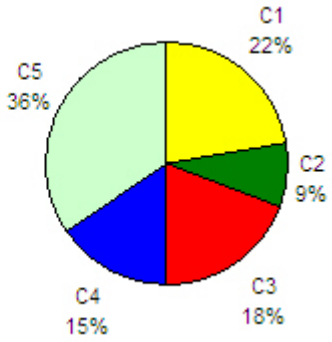
O passo seguinte é a avaliação da população e permite ao algoritmo determinar sua proximidade em relação à solução do problema. Tradicionalmente é criada uma função de avaliação ou objetivo, a qual irá avaliar cada um dos cromossomos da população. A função é aplicada aos cromossomos para se chegar a um valor que represente o grau de adaptabilidade dos indivíduos (MEDEIROS, 2005, p. 35).

Lucas (2002, p. 11) diz que atualmente várias formas de avaliação são utilizadas, tais como: cálculo de distância máxima ou mínima, valor mínimo ou máximo de funções e penalidades. Normalmente em problemas de otimização que possuem muitas restrições são utilizados cálculos de funções através de penalidades, por exemplo, cada vez que um indivíduo infringe uma regra, ele recebe uma penalidade, assim, esse indivíduo é tido como menos apto se comparado a indivíduos que infringiram menos regras.

O passo seguinte consiste na escolha dos cromossomos para fazer um posterior cruzamento. Para isso são utilizados seus graus de adaptação (LIDEN, 2006, p. 52). Os membros selecionados serão utilizados como pais para um posterior cruzamento com o objetivo de gerar novos membros para a população (RODRIGUES, ALMEIDA e GONÇALVES, 2007, p. 5).

Conforme Carvalho (2004), o método de seleção mais utilizado é a roleta, na qual os membros com melhor grau de adaptação têm maiores probabilidades de serem eleitos para um possível cruzamento. Primeiramente é feita a avaliação da população e, em seguida, são calculadas as porcentagens de cada membro na roleta, então, são gerados números aleatórios com os quais serão selecionados os cromossomos para a reprodução. A Figura 4 representa um modelo de uma roleta:

Cromossomos		Aptidão
C1	101001	0,5120
C2	001001	0,2030
C3	011001	0,4230
C4	110101	0,3500
C5	110010	0,8010



Fonte: adaptado de Carvalho, 2004.

Figura 4 - Indivíduos de uma população e sua correspondente roleta de seleção

Observando a Figura 4, é possível notar que os indivíduos com maior grau de aptidão possuem maiores fatias na roleta, tendo assim maiores chances de serem sorteados para um cruzamento. Utilizando-se a roleta, é possível garantir facilmente que indivíduos mais aptos tenham maiores chances de se reproduzirem.

Segundo Lucas (2002, p. 13), dentre os métodos de seleção também se destacam:

a) seleção por ranking: os indivíduos da população são ordenados conforme seu grau de adaptação e, então, sua probabilidade é atribuída conforme a posição que ocupam;
b) seleção por torneio: conjuntos de soluções são escolhidas sucessivamente e as mais aptas dentro de cada uma destas são selecionadas;
c) seleção uniforme: todos os membros da população tem a mesma probabilidade de serem selecionados. Obviamente, este método de seleção possui uma probabilidade bem remota de causar uma boa evolução na população.

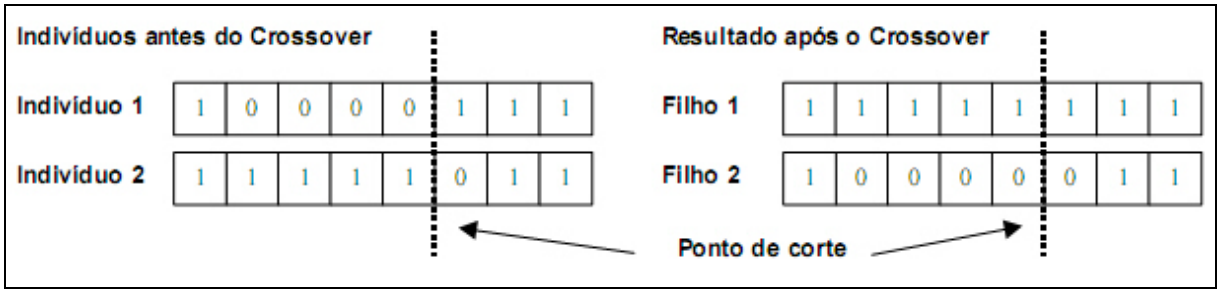
Além dos métodos citados, existem diversas outras formas de se fazer a seleção, cada método com suas particularidades, alguns podem ser mais eficientes ou menos eficientes, dependendo de cada caso. Também é necessário levar em consideração a velocidade da seleção, pois essa etapa se repete diversas vezes durante a execução e isso pode acarretar queda no desempenho do algoritmo.

Após a seleção dos indivíduos, é feita uma das mais importantes etapas do AG, o cruzamento. O cruzamento ou *crossover* é a etapa em que partes dos genes dos membros selecionados são combinados com o objetivo de gerar novos membros para a população (CAMPOS, 2002, p. 7).

Lucas (2000, p. 19) diz que se pode representar os operadores de cruzamento como uma seleção por máscara, onde os elementos são representados por vetores que possam assumir valores binários de comprimento igual ao dos cromossomos a serem combinados. Pode-se citar como os tipos principais de cruzamento: um ponto ou multiponto, segmentado e

uniforme.

Nos cruzamentos de um ponto, são selecionados dois cromossomos, exemplo `i1` e `i2`; sorteia-se um número aleatório `n` que servirá como ponto de corte, então, tem-se n tal que: `0 < n < c`, onde `c` representa o comprimento do vetor do cromossomo. O primeiro filho `f1` receberá os genes de `i1` de `0` até `n` e os genes de `i2` de `n+1` até `c`. Já o segundo filho `f2` será o inverso de `f1` (Campos, 2002, p.7). A Figura 5 representa o cruzamento:

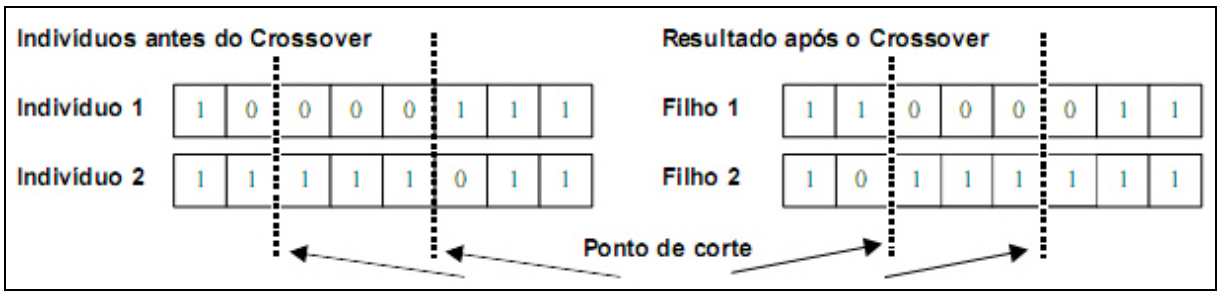


Fonte: adaptado de Medeiros (2005, p. 36).

Figura 5 - Cruzamento de um ponto

A figura 5 mostra de forma clara o cruzamento de um ponto, onde dois indivíduos `i1` e `i2` são eleitos para o cruzamento, tal que `c` é igual a 8 (oito) e `n` é 5 (cinco). A partir disso, são gerados dois indivíduos filhos `f1` e `f2`, e estes possuem partes iguais a `i1` e `i2`.

O cruzamento multiponto é semelhante ao cruzamento de um ponto, porém são feitos dois ou mais cortes, ao invés de um único corte (LUCAS, 2002, p. 15). A figura 6 representa um cruzamento multiponto.



Fonte: adaptado de Medeiros (2005, p. 36).

Figura 6 - Cruzamento multiponto

É possível observar a grande semelhança entre o cruzamento de um ponto e o cruzamento multiponto: a única diferença é o número de cortes feitos nos pais. Entretanto, neste método de cruzamento, os pontos de cortes são fixos, não existem variações nos pontos de corte durante a execução do algoritmo (LUCAS, 2002, p. 15).

O cruzamento segmentado funciona de forma semelhante ao cruzamento multiponto, porém os pontos de corte são sorteados aleatoriamente todas as vezes que é executado o cruzamento (LUCAS, 2002, p. 15).

No cruzamento uniforme são percorridos os genes dos filhos e para cada gene é

sorteado de qual pai será recebido o gene (LUCAS, 2002, p. 16).

O passo seguinte é a mutação. Segundo Campos (2002, p. 9), a mutação opera sobre os membros resultantes dos cruzamentos e através de algum tipo de alteração na estrutura dos indivíduos. O número de mutações é pré-determinado conforme um parâmetro do AG. Os principais tipos de mutação são:

a) mutação aleatória: cada gene que irá sofrer a mutação recebe um valor aleatório dentre os valores possíveis;
b) mutação por troca: são sorteados pares de genes e estes pares de genes trocam de valores entre si;
c) mutação *creep*: um valor é somado ao valor do gene ou subtraído d esse.

A mutação é uma etapa importante nos AGs pois contribui para não ocorrer estagnação da população. Através das mutações nos novos membros da população, são feitas alterações nos indivíduos que auxiliam no direcionamento da pesquisa para a solução desejada (SCIELO, 1999).

A fase seguinte consiste na avaliação dos membros resultantes. Nesta fase é feita novamente uma avaliação, mas agora nos indivíduos resultantes dos cruzamentos e mutações. Após a avaliação, esses indivíduos são inseridos na população (LIDEN, 2006, p. 52).

A etapa de inclusão dos novos membros na população faz com que os indivíduos resultantes dos cruzamentos e mutações sejam inseridos na população conforme a política adotada pelo algoritmo genético. Tradicionalmente os algoritmos genéticos mantêm o tamanho da população constante, nestes casos, os pais são removidos na próxima etapa (CAMPOS, 2002, p. 9-10).

O passo seguinte é remover velhos membros. A remoção dos velhos membros não é implementada em todos os algoritmos genéticos. Alguns modelos adicionam os novos membros à população e removem os pais que geraram os filhos, outros AGs utilizam outras funções para remoção dos membros e outras implementações não removem os pais, porém, com o passar das gerações, os recursos computacionais utilizados serão maiores (CAMPOS, 2002, p. 9-10).

Finalmente é verificada a condição de parada, ou seja, é verificado se o algoritmo chegou a uma solução aceitável, ou apenas foi finalizado devido ao estouro de seu tempo de execução, caso contrário, o algoritmo voltará para o passo onde é feita a seleção dos pais até que se chegue à solução ou até que estoure o tempo de execução (LIDEN, 2006, p. 52).

3.5 PARÂMETROS DE CONFIGURAÇÃO

Conforme afirma Catarina (2006, p. 21), é importante analisar de que maneira alguns parâmetros de configuração dos AGs influenciam em seu funcionamento. Cada caso específico possui seus próprios parâmetros de configuração, porém existem alguns mais utilizados, dentre os quais podem-se citar:

a) tamanho da população: determina o número de cromossomos na população e afeta de forma significativa o desempenho global e a eficiência do algoritmo. Com uma população menor o desempenho pode cair, pois a população fornecerá menor biodiversidade e o espaço de busca do AG será menor. Uma população grande geralmente fornece uma boa biodiversidade; com um grande espaço de busca, previnem-se convergências prematuras para soluções locais ao invés de globais. Porém os recursos computacionais necessários serão bem maiores ou o algoritmo terá de ficar mais tempo executando até chegar à solução desejável;
b) taxa de cruzamento: determina a probabilidade com que um cruzamento poderá acontecer. Caso esta taxa seja maior, mais rapidamente novos cromossomos serão adicionados à população. No entanto, se essa taxa for muito alta, grande parte da população de cromossomos será rapidamente substituída e poderá ocorrer perda de cromossomos com alta aptidão. Com um valor baixo, o algoritmo perde muito desempenho;
c) taxa de mutação: representa a probabilidade de ocorrerem mutações. Com uma taxa baixa se previne a ocorrência prematura para um ótimo local, possibilitando ao AG explorar melhor todo seu espaço de busca. Uma taxa muito alta torna o processo de busca muito aleatório, podendo dificultar a convergência para uma solução ótima;
d) intervalo de geração: faz o controle da porcentagem da população que será substituída durante a próxima geração. Com um valor muito alto, a grande maioria da população será substituída, podendo ocorrer a eliminação de estruturas de alta aptidão. Com um valor muito baixo, o algoritmo torna-se lento.

É importante salientar que a influência de cada um dos parâmetros do AG está diretamente ligada à classe de problemas que está sendo tratada (COSTA; BRUNA, 2003).

3.6 APLICAÇÕES

Miranda (2007) diz que os AGs possuem uma larga aplicação em muitas áreas científicas, dentre as quais podem ser destacadas:

a) síntese de circuitos analógicos: no meio eletrônico, para uma certa entrada e uma saída desejada, por exemplo, tensão, o AG gera a topologia, o tipo e o valor dos componentes do circuito;
b) síntese de protocolos: no meio computacional, o AG determina que funções do protocolo devem ser implementadas em hardware e quais devem ser implementadas em software para se alcançar o desempenho desejado;
c) gerenciamento de redes: também no meio computacional, para supervisão de tráfego nos *links* e das filas nos *buffers* de roteadores para se descobrir rotas ótimas e para configurar novamente o tráfego no caso de falha de algum link;
d) computação evolutiva: gera programas que se adaptam a mudanças no sistema ao longo do tempo;
e) problemas de otimização muito complexos: problemas que envolvem muitas variáveis e espaços de soluções de dimensões elevadas, como: problema do caixeiro viajante, gerenciamento de carteiras de fundo de investimentos ou problemas de alocação de horários;
f) ciências biológicas: modela processos biológicos para o entendimento do comportamento de estruturas genéticas.

Na próxima seção é apresentado o desenvolvimento do sistema.

4 DESENVOLVIMENTO DO PROTÓTIPO

Nesta seção é feita a especificação do trabalho através dos diagramas de classes, seqüência e descrição dos principais casos de uso e requisitos. Também é descrito o modelo adotado para a implementação do AG, além da modelagem de dados e interface do sistema.

4.1 REQUISITOS PRINCIPAIS DO TRABALHADO

Para o desenvolvimento de um trabalho que propõe um modelo de um algoritmo genético para gerar grades horárias, fez-se o levantamento de vários requisitos. Foram levantados os requisitos, os quais foram divididos em funcionais e não funcionais.

4.1.1 Requisitos funcionais

Os requisitos funcionais (RF) são:

a) gerar grades de horários, respeitando as restrições impostas pelas disponibilidades dos professores e disciplinas;
b) permitir o cadastro de cursos, semestres, disciplinas e professores para o administrador do sistema;
c) disponibilizar um cadastro de professores disponíveis para cada disciplina;
d) permitir o cadastro de disponibilidades de horários para cada professor, sendo que cada professor poderá efetuar seu próprio cadastro de disponibilidade;
e) permitir a visualização das grades geradas para cada professor.

4.1.2 Requisitos não funcionais

Os requisitos não funcionais (RNF) são:

a) ser implementado para web utilizando a linguagem de programação Java;
b) utilizar o banco de dados MySQL, versão 5.0;

c) ser compatível com os navegadores Internet Explorer 6 ou superior e Mozilla Firefox versão 2 ou superior.

4.2 ESPECIFICAÇÃO

Os tópicos subseqüentes descrevem a especificação do protótipo. A especificação do sistema foi feita através das ferramentas Jude 3.0 (JUDE, 2006) e Enterprise Architect 5.0 (SPARX SYSTEMS, 2005). A modelagem de dados foi feita utilizando a ferramenta Case Studio 2.3 (CHARONWARE, 2006).

4.2.1 Diagrama de atividades

A Figura 7 representa o diagrama de atividades representa o fluxo executado para a geração de grades horárias.

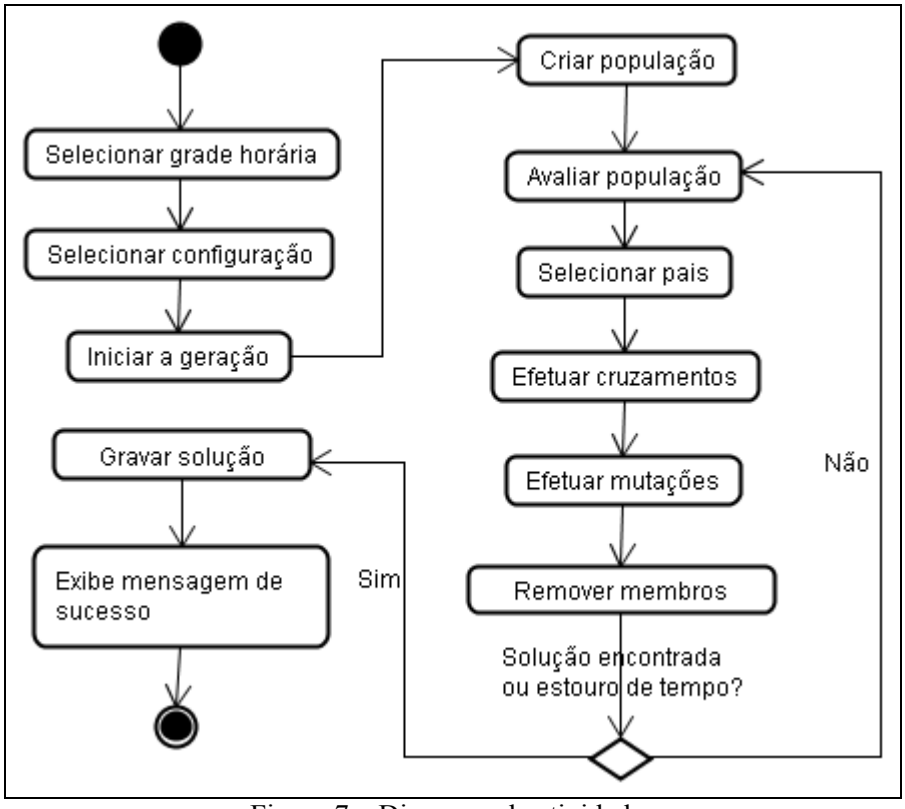


Figura 7 – Diagrama de atividades

Na próxima seção são apresentados os diagramas de pacotes e classes.

4.2.2 Diagrama de pacotes

O diagrama de pacotes é representado na Figura 8, através da qual é possível observar de forma macro os pacotes e relações entre os mesmos.

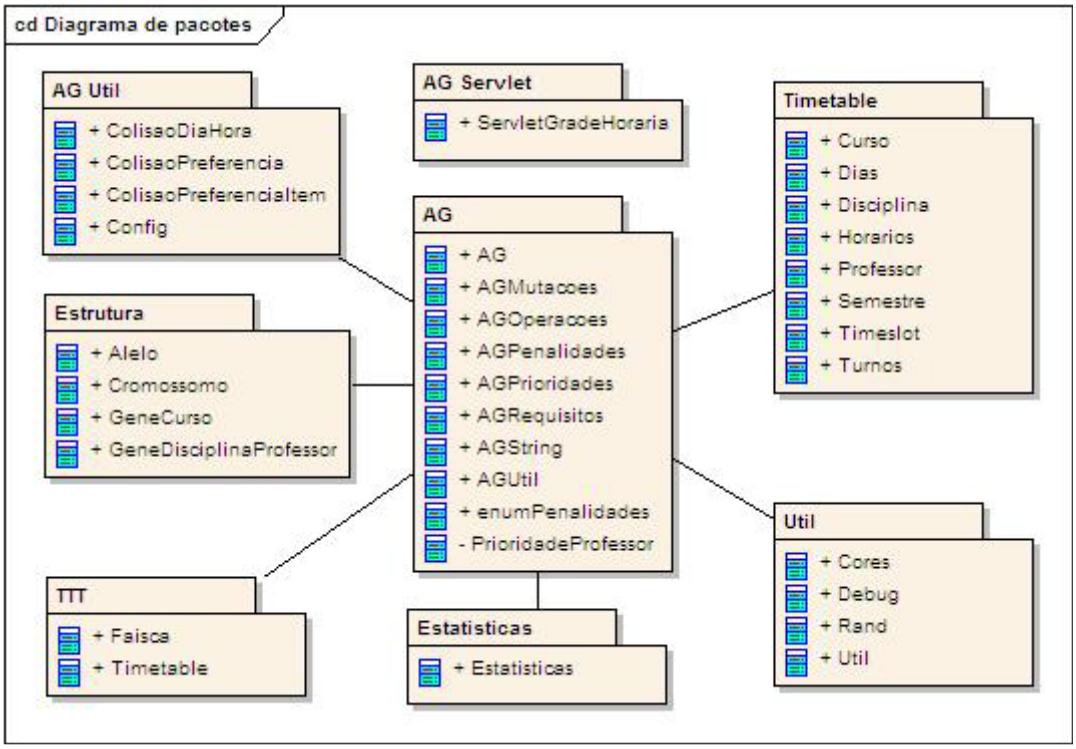


Figura 8 – Diagrama de pacotes

Através da Figura 8 pode-se observar que foram implementados oito pacotes no protótipo. Na próxima seção é feito o detalhamento de cada um deles.

4.2.3 Diagrama de classes

As classes implementadas são representadas nas figuras abaixo, os relacionamentos entre classes de pacotes diferentes foram abstraídos. Na Figura 9 são representadas as classes referentes ao AG.

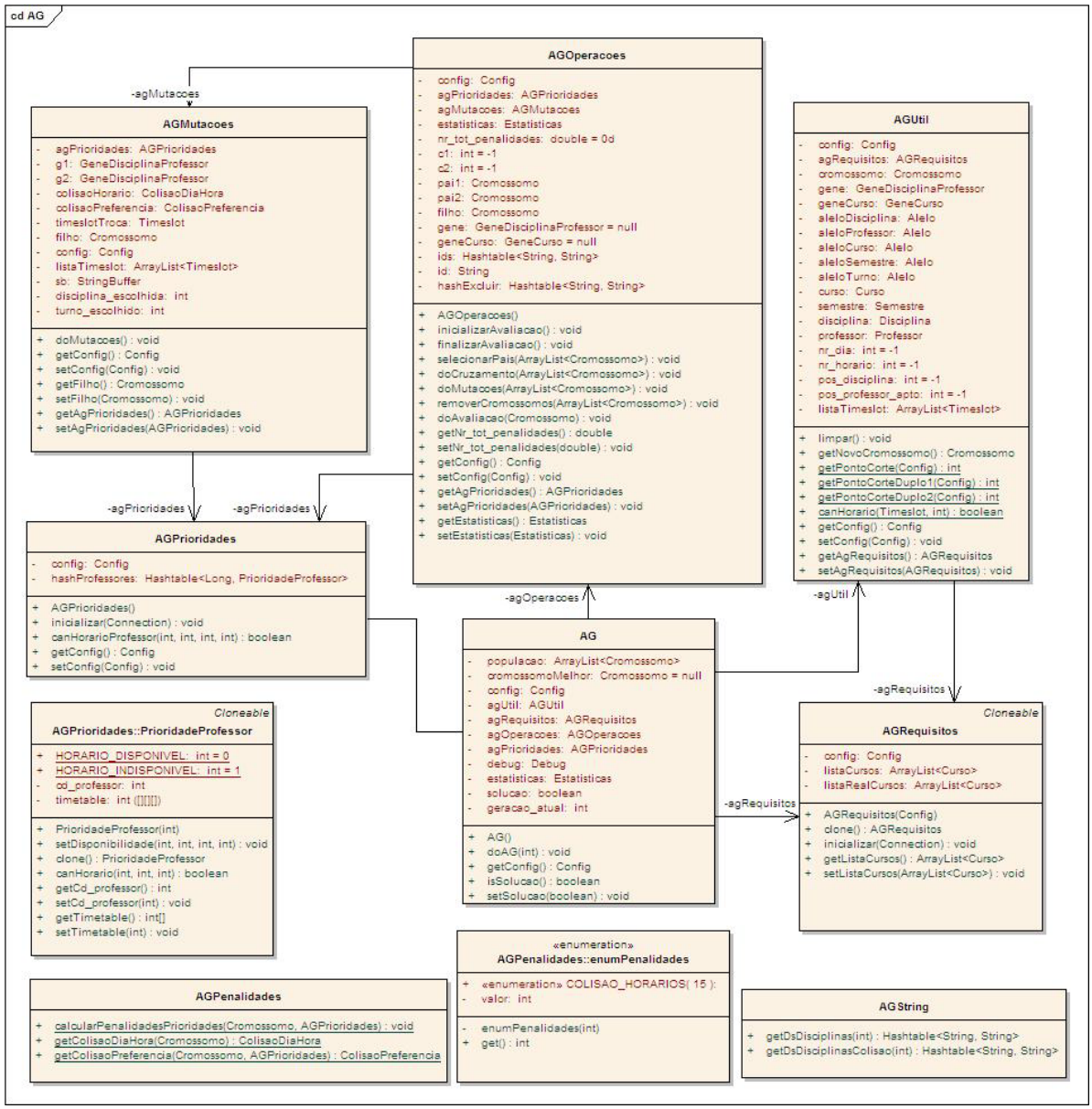


Figura 9 – Diagrama de classes - AG

As classes representadas na Figura 9 possuem as seguintes funcionalidades:

a) `AGMutacoes`: classe responsável por efetuar as operações de mutações nos indivíduos da população;

b) `AGOperacoes`: responsável por diversas funcionalidades, tais como: avaliação da população, seleção dos membros para o cruzamento, operações de cruzamento, remoção dos cromossomos e outros;

c) `AGUtil`: possui como principal funcionalidade a geração dos membros no momento da criação de uma nova população, além de gerar os pontos de cortes;

d) `AGPrioridades`: contém as prioridades de horários dos professores;

e) `AG`: responsável pela execução do algoritmo, através desta classe são feitas as chamadas para as demais classes;

f) `AGRequisitos`: utilizada na inicialização da população, contém as informações referentes aos semestre, disciplinas do semestre e professores aptos para as disciplinas;

g) `AGPenalidades`: efetua o cálculo das penalidades dos cromossomos com o objetivo de gerar o grau de aptidão desses;

h) `enumPenalidades`: contém valores que representam o peso de cada tipo de penalidade;

i) `AGString`: auxiliar para descrever as grades horárias na aplicação web.

A Figura 10 apresenta as classes utilizadas na representação dos cromossomos.

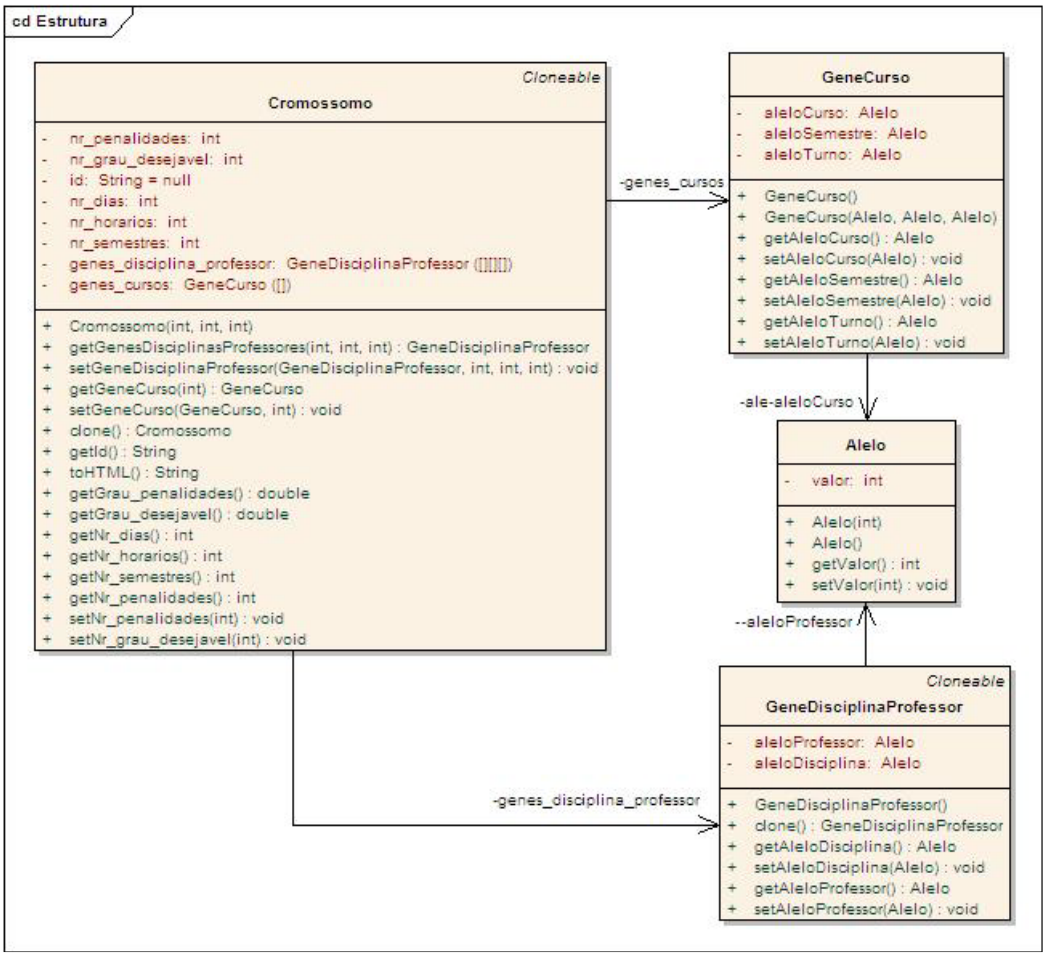


Figura 10 – Diagrama de classes - Estrutura cromossomos

As classes representadas na Figura 10 possuem as seguintes funcionalidades:

a) `Cromossomo`: representa um indivíduo da população;

b) `GeneCurso`: classe que representa o curso, semestre e turno ao qual pertence o curso;

c) `Alelo`: utilizada para representar valores, como, por exemplo, código do curso ou disciplina;

d) `GeneDisciplinaProfessor`: representa os genes onde contêm os valores das disciplinas e professores.

Na Figura 11 é representado o diagrama de classes das configurações do AG e outras classes auxiliares.

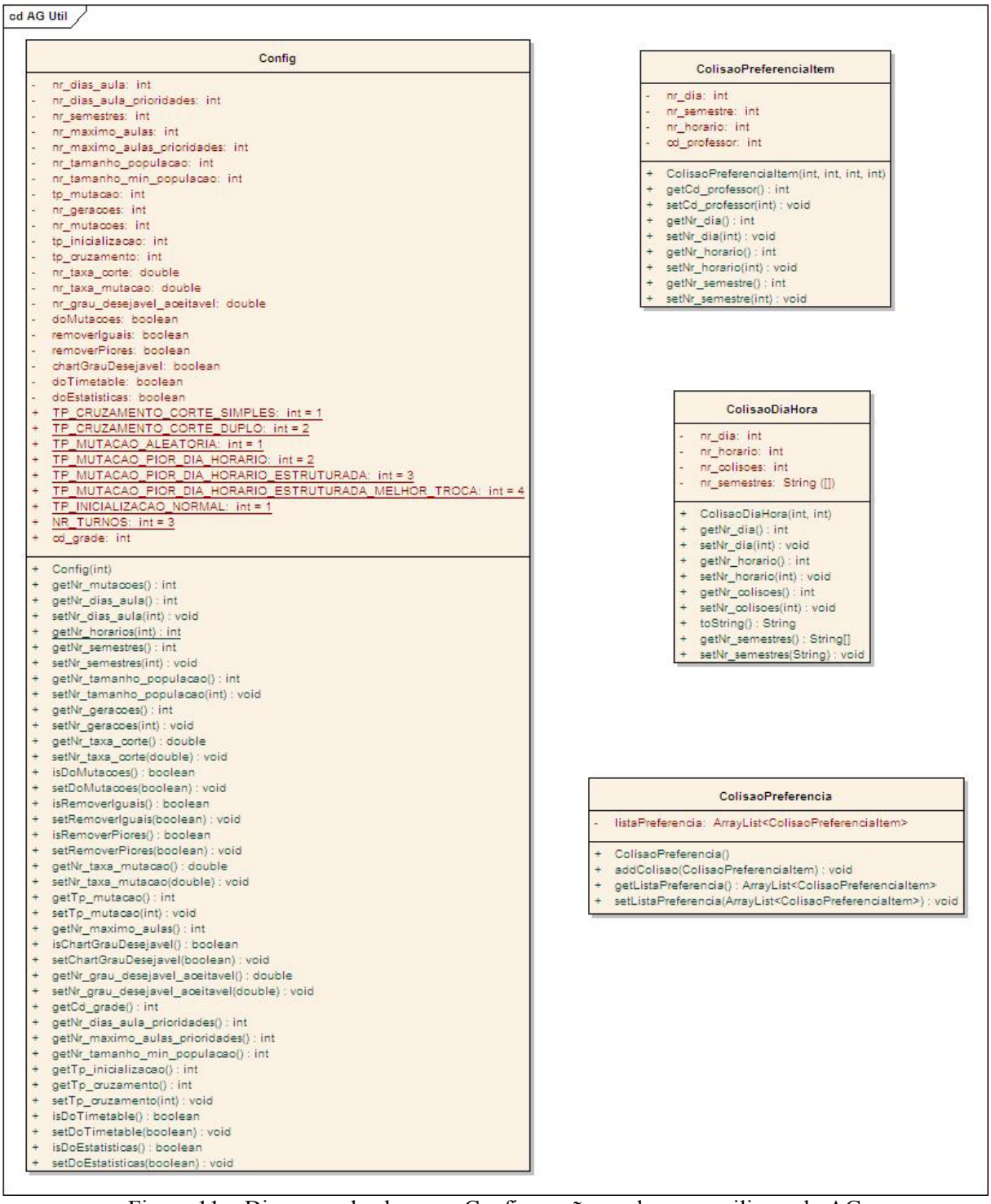


Figura 11 – Diagrama de classes – Configurações e classes auxiliares do AG

As classes representadas no diagrama conforme Figura 11, possuem as seguintes funcionalidades:

a) `Config`: responsável pela configuração do algoritmo, através desta classe é possível configurar parâmetros como, por exemplo, taxa de mutação, tamanho da população e outros;

b) `ColisaoPreferenciaItem`: classe auxiliar utilizada nas mutações dos cromossomos;

c) `ColisaoDiaHora`: utilizada como classe auxiliar no processo de mutação;

d) `ColisaoPreferencia`: também é uma classe auxiliar utilizada nos processos de mutação.

Na Figura 12 são representadas as classes referentes a questões relacionadas ao problema de *timetabling*.

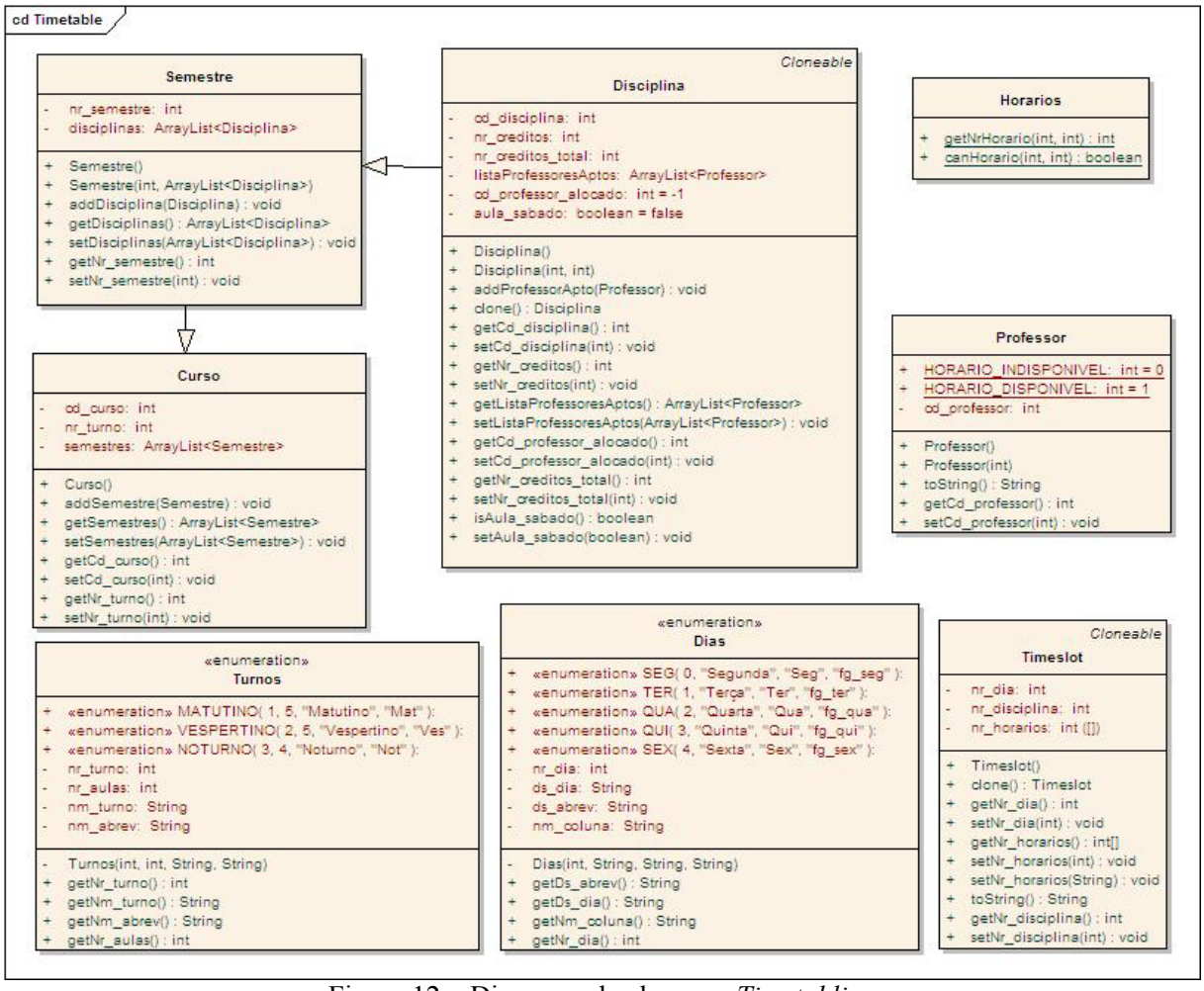


Figura 12 – Diagrama de classes – *Timetabling*

As classes representadas na Figura 12 possuem as seguintes funcionalidades:

a) `Semestre`: classe que representa um semestre, utilizada no momento de inicialização da população;

b) `Disciplina`: representa uma disciplina, também utilizada na inicialização da população;

c) `Horarios`: classe auxiliar utilizada para representar preferências de horários dos professores;

d) `Curso`: representa um curso, usada na criação de novas populações;

e) `Turnos`: representa os turnos - matutino, vespertino e noturno - possuindo a descrição textual deles e números de aulas de cada um;

f) `Dias`: através dessa classe são representados os dias da semana, utilizada em diversas operações como, por exemplo, no momento de verificar disponibilidades nos horários dos professores;

g) `Timeslot`: classe utilizada como auxiliar nas mutações.

Na Figura 13 são apresentadas as demais classes responsáveis por diversas operações, tais como: conexão com o banco de dados, armazenamento de estatísticas, depuração e outras.

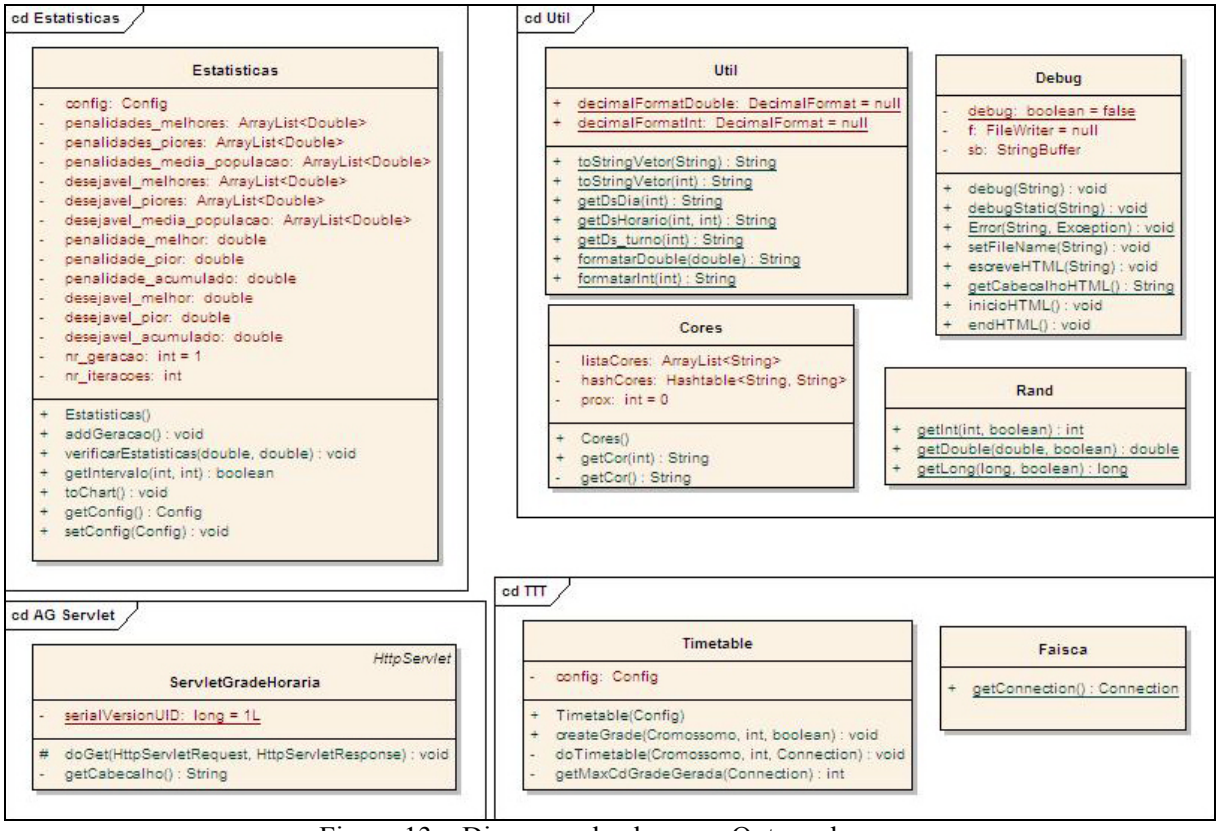


Figura 13 – Diagrama de classes – Outras classes

As classes representadas na Figura 13 possuem as seguintes funcionalidades:

a) `Estatisticas`: classe responsável por armazenar valores relevantes na evolução dos cromossomos com o objetivo de gerar um gráfico representando a evolução da espécie;

b) `Util`: utilizada na formatação de valores para apresentar para os usuários;

c) `Debug`: classe auxiliar para realizar depurações na execução do algoritmo;

d) `Cores`: utilizada na classe Estat sticas para representar as cores do gráfico;

e) `Rand`: classe utilizada para gerar números aleatórios, usada em diversos lugares, tais como: seleção aleatória de cromossomos para mutação, criação aleatória de membros para a população inicial e outros;

f) `ServletGradeHoraria`: tem como objetivo executar o algoritmo genético através do sistema web;

g) `Timetable`: responsável por salvar as grades horárias geradas no banco de dados;

h) `Faisca`: classe responsável por fazer a conexão com o banco de dados.

4.2.4 Casos de uso

Conforme Nogueira (2006) os casos de uso especificam o comportamento do sistema ou partes dele e descrevem a funcionalidade do sistema desempenhada por atores. Pode-se imaginar um caso de uso como um conjunto de cenários em que cada cenário é uma seqüência de passos, a qual descreve uma interação entre um usuário e o sistema. Tradicionalmente os casos de uso são representados por elipses. Na Figura 14 são representados os casos de uso do protótipo.

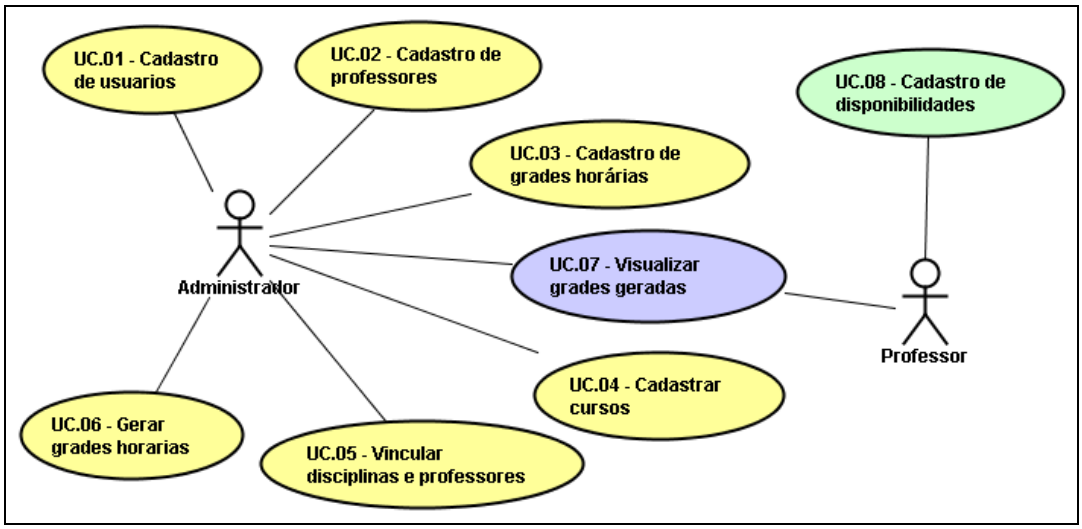


Figura 14 – Casos de uso do protótipo

A seguir são descritas as principais características dos casos de uso:

a) cadastro de usuários: através deste caso de uso é possível efetuar o cadastro e alteração de usuários administradores do sistema;

b) cadastro de professores: neste caso de uso é feito o cadastro e edição de professores e login para os mesmos;

c) cadastro de grades horárias: através deste caso de uso é possível cadastrar e editar grades horárias;
d) cadastro de cursos: essa funcionalidade permite o usuário administrador efetuar o cadastro e edição dos cursos. Também é possível cadastrar e editar as disciplinas dos cursos;
e) vincular disciplinas e professores: através deste caso de uso é possível vincular qual professor poderá ser escolhido para determinada disciplina;
f) gerar grades horárias: neste caso de uso é possível gerar as grades horárias;
g) cadastro de disponibilidades: através dessa funcionalidade os professores podem informar as preferências por horários;
h) visualizar grades horárias: através deste caso de uso podem-se visualizar as grades horárias geradas.

4.2.5 Modelagem de dados

O modelo de dados permite uma visão global do sistema, identificando as entidades e atributos, assim como seus relacionamentos. A Figura 15 representa o Modelo Entidade Relacionamento (MER) do sistema proposto.

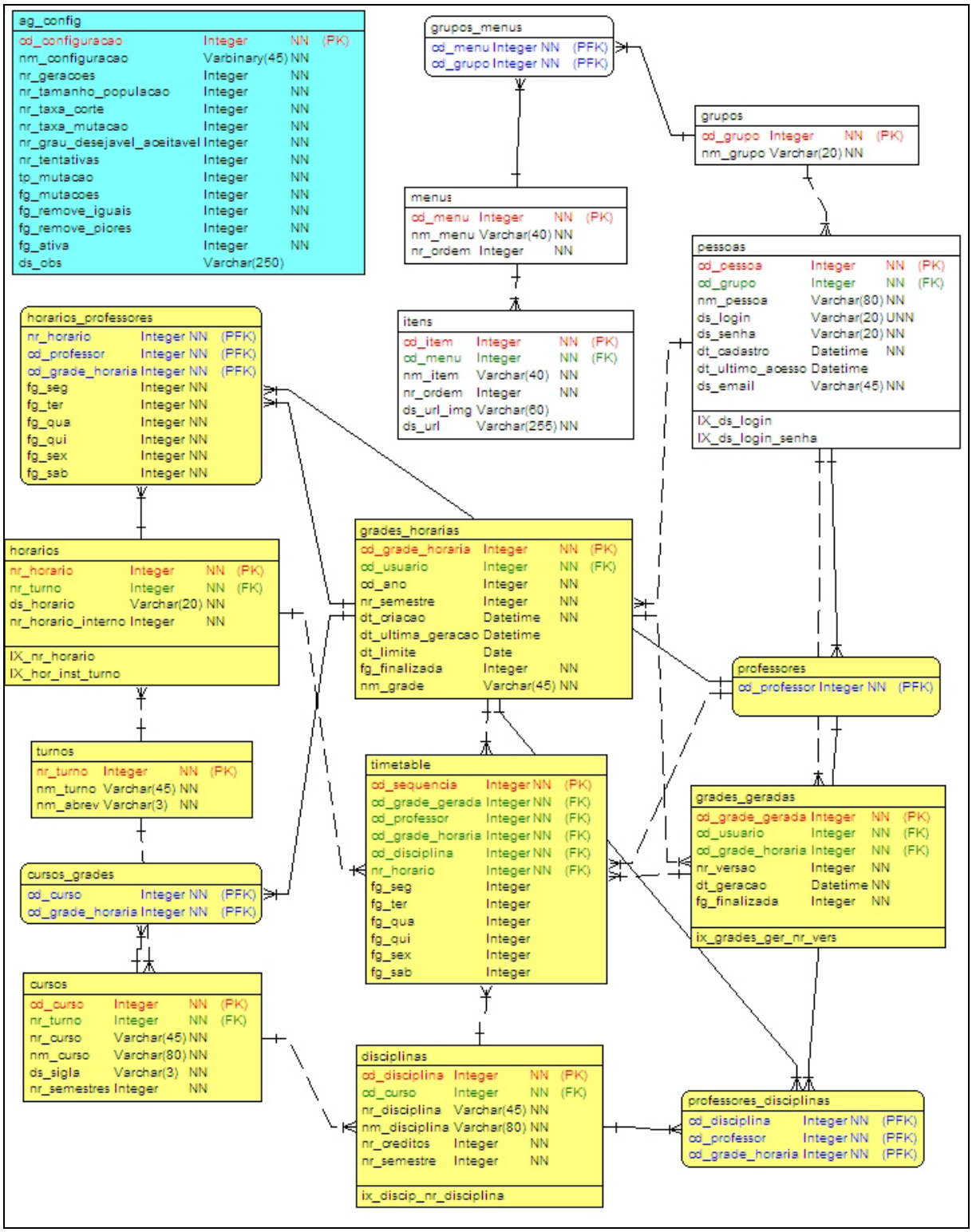


Figura 15 – MER do sistema

No Quadro 2 a seguir é descrito o dicionário de dados das tabelas do protótipo:

Nome da tabela	Descrição
ag_config	Parâmetros de configuração do AG
horarios_professores	Utilizada para armazenar as preferências de horários dos

	professores
horarios	Armazena os horários de aula
turnos	Armazena os turnos existentes
cursos_grades	Utilizada para fazer a vinculação do curso com as grades horárias
cursos	Serve para armazenar os cursos
grupos_menus	Utilizada para armazenar os grupos de acessos dos menus
menus	Armazena os menus do sistema
itens	Utilizada para armazenar os itens dos menus
grades_horarias	Utilizada para armazenar as grades horárias
timetable	Tabela utilizada para armazenar os horários de aulas dos professores em uma determinada grade horária
disciplinas	Armazenar as disciplinas
grupos	Utilizada para armazenar os grupos
pessoas	Usada para o cadastro de pessoas
professores	Armazena o código das pessoas que são professores
grades_geradas	Utilizada para armazenar as grades horárias geradas
professores_disciplinas	Armazena qual professor é apto para ministrar qual disciplina em uma determinada grade horária

Quadro 2 – Dicionário de dados

4.3 IMPLEMENTAÇÃO

Nesta seção são relatados aspectos relacionados à implementação do protótipo final. São também mostrados detalhes técnicos da construção da aplicação, fazendo-se um detalhamento em relação aos passos necessários para a construção do algoritmo genético.

4.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do protótipo foram utilizadas várias ferramentas e linguagens em conjunto. A programação do AG foi feita utilizando a linguagem Java e a ferramenta

Eclipse 3.2 (ECLIPSE, 2007). A modelagem de dados foi feita com a ferramenta Case Studio 2.3 (CHARONWARE, 2006) e o Sistema Gerenciador de Banco de Dados (SGBD) adotado foi o MySQL 5.0 (MYSQL, 2007).

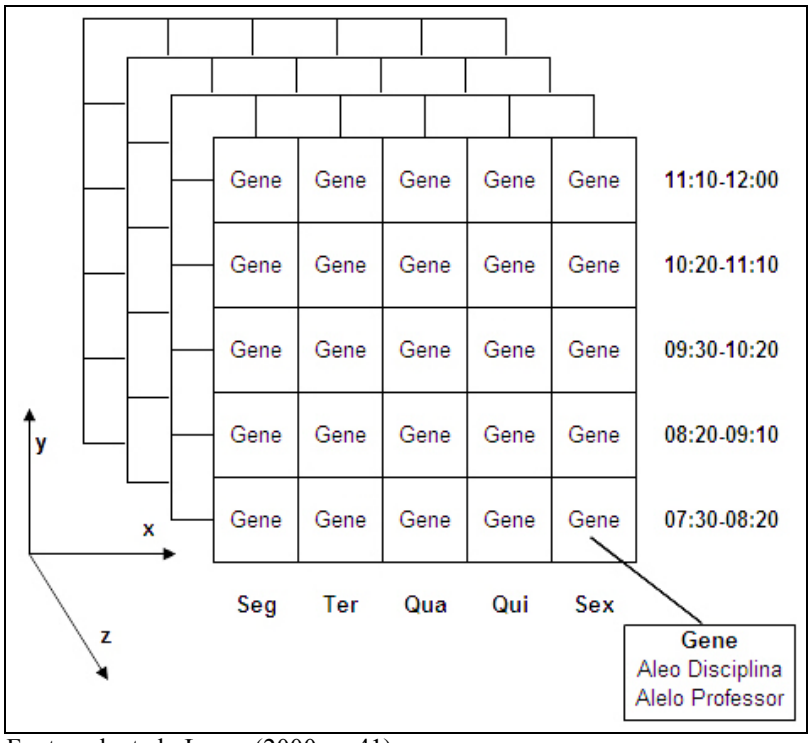
A parte web foi feita através do *framework* Code Charge Studio 3.2 (YES SOFTWARE, 2007). A linguagem adotada no *framework* foi Java Servlets e o servidor de aplicações Tomcat 5.0 (APACHE, 2007).

4.3.2 Implementação do AG

Nesta seção serão mostrados detalhes da implementação do AG. Primeiramente são apresentados detalhes com relação à modelagem do problema, então é apresentado um trecho de código que resume o funcionamento do AG, em seguida é feito um detalhamento sobre as etapas mais importantes do AG para, então, finalizar com os problemas encontrados e conclusões.

4.3.2.1 Representação do problema

Uma das grandes dificuldades encontradas na implementação de AGs é a representação do problema, questão já descrita anteriormente. É importante modelar ao solução do problema de forma que seja possível representar todas as soluções interessantes na sua resolução, prezando, também, pela simplicidade da estrutura de dados escolhida. Dessa forma, facilita-se o processo de manipulação dos cromossomos durante a execução do AG. A Figura 16 representa o modelo de cromossomo adotado no presente trabalho.



Fonte: adaptado Lucas (2000, p. 41).

Figura 16 – Representação escolhida

Na Figura 16 é representada a estrutura do cromossomo adotado: o eixo `z` representa os semestres, o eixo `x` representa os dias da semana e o eixo `y` os horários das aulas. Cada posição na matriz possui um gene, que por sua vez possui dois alelos, um para indicar qual é a disciplina e outro para indicar qual é o professor alocado. No Quadro 3, é representado um trecho de código simplificado da classe `Cromossomo` em Java.

```
4  public class Cromossomo {
5      private Gene[][][] genes;
6
7      public Cromossomo( int nr_dias, int nr_aulas, int nr_semestres ) {
8          genes = new Gene[ nr_dias ][ nr_aulas ][ nr_semestres ];
9      }
10
```

Quadro 3 – Classe `Cromossomo`

É possível observar a simplicidade da estrutura adotada, a qual se resume a uma matriz de três dimensões contendo genes. Os genes, a seu turno, possuem dois alelos cada, no Quadro 4 é possível observar os atributos e o construtor da classe `Gene`.

```
4  public class Gene {
5      private Alelo aleloProfessor;
6      private Alelo aleloDisciplina;
7
8      public Gene( Alelo professor, Alelo disciplina ) {
9          aleloProfessor = professor;
10         aleloDisciplina = disciplina;
11     }
```

Quadro 4 – Classe `Gene`

A classe `Alelo` é representada no Quadro 5. Essa classe é extremamente simples, porém nela ficam armazenados os valores que representam a disciplina e o professor de cada um dos genes do cromossomo.

```
3  public class Alelo {
4      private int valor;
5
6      public Alelo( int vlr ) {
7          valor = vlr;
8      }
9
```

Quadro 5 – Classe `Alelo`

Através dessas três classes é possível representar todas as soluções que envolvem o problema de alocação de horários relevantes no presente trabalho. Nesse contexto, o esquema de matriz de três dimensões garante que não haja colisão de horários para um mesmo professor, em um determinado semestre. Porém, não é possível garantir que não haja colisões de horários para um professor em semestres diferentes.

4.3.2.2 Funcionamento do AG

O funcionamento do AG é baseado no modelo proposto por Campos (2002, p. 4), em que, primeiramente, é feita a inicialização da população; então, é feito um laço de repetição, para, a seguir, ser feita a avaliação dos membros. Na seqüência é feita a seleção dos pais, então, são feitos os cruzamentos. O próximo passo é a mutação, depois são removidos os velhos membros, para, em seqüência, verificar-se a condição de parada, caso não seja satisfeita, a execução volta para a avaliação dos membros; caso seja satisfeita o algoritmo termina sua execução. O Quadro 6 mostra de forma resumida o trecho de código que faz o AG.

```
6  public class AG {
7      private Config config;
8      private ArrayList<Cromossomo> populacao;
9
10     public void doAG( int cd_grade ) {
11         inicializarPopulacao();
12         for( int geracao = 0 ; geracao < config.getNr_geracoes() ; geracao++ ) {
13             avaliar( geracao );
14             selecionarPais();
15             doOperadoresCruzamento();
16             doMutacoes();
17             removerCromossomos();
18             if( canStop() ) {
19                 break;
20             }
21         }
22         finalizar();
23     }
```

Quadro 6 – Classe AG

A classe AG, dentre outros atributos, possui um atributo com os parâmetros de configuração do AG. Nesse caso, utiliza o número de gerações que serão feitas até finalizar a execução, podendo finalizá-la antes caso sejam satisfeitas as condições de parada. Nas próximas seções, é descrito o funcionamento de cada um dos passos da classe AG.

4.3.2.3 Inicialização da população

A inicialização da população consiste em criar um conjunto de indivíduos iniciais. A população inicial, via de regra, não possui indivíduos ótimos, pois a probabilidade de existir alguma colisão nos horários de professores é extremamente grande, devido ao grande número de restrições envolvidas e inicialização aleatória da população.

A criação da população não é totalmente aleatória, foram respeitadas algumas condições antes de povoar os genes e alelos na população de cromossomos. Dentre as condições na criação da população inicial podem-se citar:

a) restrições de professor e disciplinas: cada novo cromossomo recebe apenas genes e alelos de disciplinas e professores que sejam aptos a ministrarem as disciplinas. Isso garante que não haja inconsistências de professores com disciplinas, como, por exemplo: um professor de álgebra linear alocado na disciplina de compiladores;
b) grade horária completa: através da restrição de grade horária completa é possível garantir que cada um dos semestres possua todas as disciplinas necessárias. Através disso, é garantido que não existam semestres que possuam todas as disciplinas e/ou carga horária incompletas;

c) disciplinas repetidas num mesmo dia: são criados cromossomos que possibilitam que todas as aulas de uma mesma disciplina não sejam centralizadas em um dado dia da semana. Essa condição pode ser quebrada posteriormente devido a cruzamentos e/ou mutações.

Através da criação dessas condições iniciais foi possível reduzir o tempo de execução do algoritmo. Caso não existissem essas restrições , em cada uma das iterações do algoritmo teriam de ser verificados todos os cromossomos com o objetivo de garantir que não haja professores alocados em disciplinas errôneas e de verificar se todos os semestres possuem a grade horária e de disciplinas completas.

4.3.2.4 Funções de avaliação

As funções da avaliação criadas no protótipo são baseadas em penalidades. Os indivíduos com maior número de penalidades são menos aptos do que indivíduos com menos penalidades. É caracterizada uma penalidade quando ocorre uma ou mais colisões de horários de professores. No Quadro 7 é representada uma grade horária com colisões.

Semestre 1						Semestre 2					
D2-P5	D4-P6	D2-P5	D1-P1	D4-P6	H4	D6-P2	D8-P7	D9-P3	D8-P7	D7-P5	H4
D2-P5	D4-P6	D2-P5	D1-P1	D4-P6	H3	D6-P2	D8-P7	D9-P3	D8-P7	D7-P5	H3
D1-P1	D3-P3	D4-P6	D3-P3	D2-P5	H2	D5-P1	D7-P5	D6-P2	D5-P1	D9-P3	H2
D1-P1	D3-P3	D4-P6	D3-P3	D2-P5	H1	D5-P1	D7-P5	D6-P2	D5-P1	D9-P3	H1
Seg	Ter	Qua	Qui	Sex		Seg	Ter	Qua	Qui	Sex	

Quadro 7 – Exemplo de grade horária com colisão

No Quadro 7 é representado um cromossomo de uma grade horária com dois semestres, onde `D1` a `D9` representam as disciplinas e `P1` a `P8` representam os professores. É possível notar a colisão de horários para o professor `P1` na segunda-feira: horários `H1` e `H2`.

Para calcular o grau de aptidão dos cromossomos é utilizada a fórmula representada no Quadro 8.

$$G = \frac{1}{(1 + Np)} \quad \text{tal que:} \quad \begin{matrix} G = \text{Grau de aptidão} \\ Np = \text{Número de penalidades} \end{matrix}$$

Quadro 8 – Formula para calcular o grau de aptidão

Os valores de aptidão variam entre 0 e 1. Um cromossomo sem nenhuma penalidade terá o grau de aptidão igual a 1 e será uma solução ótima. Já os cromossomos que possuem valores de aptidão diferentes de 1, são cromossomos inaptos para a solução do problema.

Aplicando-se a fórmula descrita no Quadro 8 e utilizando-se as penalidades do cromossomo representadas no Quadro 7, tem-se um grau de aptidão representado no Quadro 9.

$$G = \frac{1}{(1 + Np)} \quad \text{onde:} \quad Np = 4$$

$$G = \frac{1}{(1 + 4)} \quad \text{então:} \quad G = 0,2$$

Quadro 9 – Calculo do grau de aptidão

Com a utilização dessa fórmula é possível garantir que indivíduos com mais penalidades possuam grau de aptidão menor, apresentando, desse modo, menos chances de serem escolhidos para os possíveis cruzamentos.

Na implementação do protótipo foram utilizados dois graus de aptidão para os cromossomos. O primeiro grau representa as colisões nos horários de professores, o segundo grau representa as disponibilidades de horários dos professores. Cada vez que uma disponibilidade de horário de um professor é violada, é acrescentado um valor no número de penalidades. O funcionamento é igual ao citado nesta seção.

4.3.2.5 Seleção dos membros para reprodução

O método de seleção utilizado no protótipo é o modelo de roleta, baseado no modelo proposto por Carvalho (2004). Os cromossomos com maior grau de aptidão recebem fatias maiores da roleta. Primeiramente é feita a soma dos graus de aptidão de todos os cromossomos da população, então, é gerado um número aleatório variando de zero até o valor correspondente à soma dos graus, para, em seguida, verificar-se qual foi o cromossomo escolhido.

Nesta etapa são selecionados dois membros para o cruzamento, sendo que não há diferenciação sexual dos cromossomos. Caso seja sorteado o mesmo membro duas vezes, na mesma iteração, será feita a seleção novamente, até que sejam selecionados dois membros diferentes. Através disso, é possível garantir que o filho que será gerado seja geneticamente diferente de seus pais.

4.3.2.6 Operadores de cruzamento

Operadores de cruzamento, como já citado anteriormente, são responsáveis por gerar novos membros através do cruzamento de seus pais. Neste trabalho foram implementados dois métodos para efetuar os cruzamentos: cruzamento de um ponto e de dois pontos.

Inicialmente foi implementado o cruzamento de um ponto representado na Figura 17. Neste método são utilizados os dois pais e é feito um corte: o cromossomo resultante recebe parte dos semestres de um dos pais e o resto do outro pai.

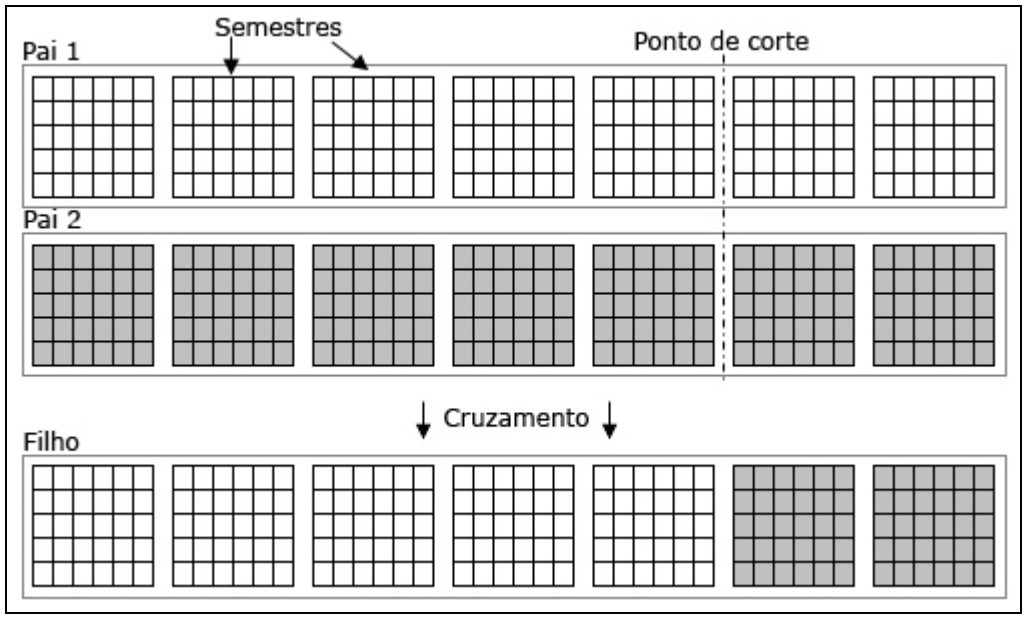


Figura 17 – Cruzamento de um ponto implementado

O valor do ponto de corte utilizado neste cruzamento é um parâmetro de configuração do AG. É utilizado um valor de 1 a 99 que representa uma porcentagem que é utilizada como ponto de corte. Exemplo: uma população de cromossomos com dez semestres e ponto de corte igual a vinte - no momento do cruzamento, o cromossomo resultante receberá oito semestres de um dos pais e dois semestres do outro pai.

Contudo, o método de cruzamento de um ponto, após exaustivos testes na implementação, tornou-se ineficiente em diversos casos. Com o passar das gerações, esse tipo de corte produzia diversos indivíduos geneticamente iguais, pois o corte sempre é feito no mesmo ponto, ocasionando uma redução significativa na biodiversidade da população. A probabilidade de ocorrerem convergências prematuras é elevada, podendo levar o AG a tornar-se extremamente ineficiente em alguns casos.

O cruzamento de dois pontos é semelhante ao de ponto único, porém são feitos dois cortes nos pais. Na Figura 18 é representado este tipo de cruzamento.

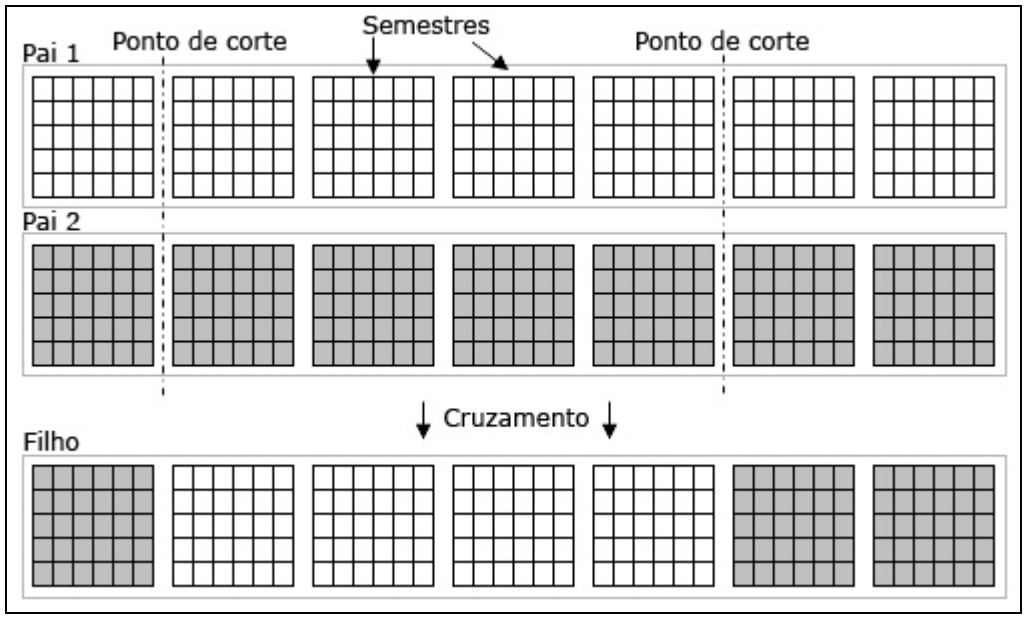


Figura 18 – Cruzamento de dois pontos implementado

Neste cruzamento além de serem feitos os dois cortes, os pontos de corte são aleatórios. O primeiro ponto de corte pode variar entre o segundo semestre até o número total de semestres dividido por dois, já o segundo corte pode variar entre o total de semestres dividido por dois até o total de semestres menos um. Na Figura 19, é representado um exemplo deste tipo de corte.

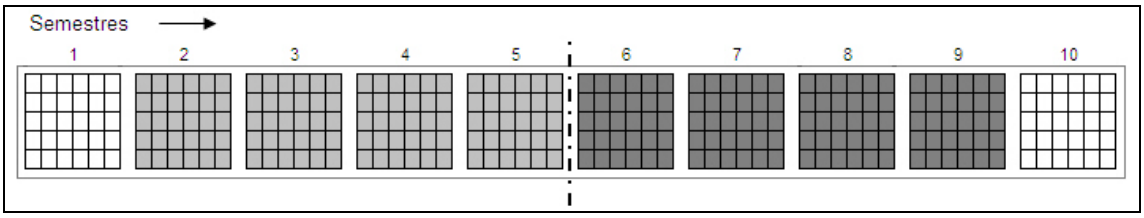


Figura 19 – Cruzamento de dois pontos aleatórios

Conforme exemplo de cromossomo apresentado na Figura 18, o primeiro ponto de corte pode variar entre os semestres dois e cinco, semestres destacados em cinza claro. O segundo corte pode variar entre os valores seis e nove, semestres destacados em cinza escuro. Através da utilização desta forma de cruzamento, foi possível garantir uma melhor biodiversidade da população. A probabilidade de surgirem indivíduos geneticamente idênticos é consideravelmente reduzida, com isso, foi possível explorar melhor o espaço de busca do algoritmo, retardando a ocorrência de convergências prematuras.

4.3.2.7 Mutação, inclusão e remoção de cromossomos na população

A etapa das mutações é uma etapa importante do algoritmo, pois através dela são feitas

trocas de posição entre os genes, com o objetivo de garantir uma boa convergência e biodiversidade da população. Como os operadores de cruzamentos operam fazendo cortes nos semestres para gerar novos cromossomos, as características de cada um dos semestres após o cruzamento permanecem as mesmas. Porém, com as mutações são feitas trocas de posições nos genes dentro de um mesmo semestre, o que pode auxiliar na busca de uma solução ótima. Na Figura 20, é representado um exemplo de mutação, sendo possível observar trocas de genes.

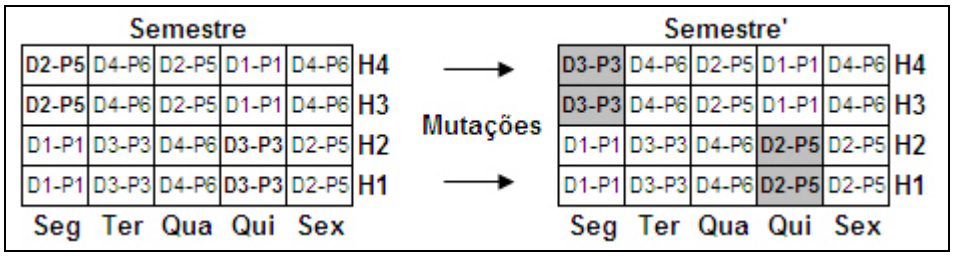


Figura 20 – Exemplo de mutação

As mutações ocorrem apenas em indivíduos resultantes de cruzamentos, sendo o número de mutações que o cromossomo filho sofre é configurada através de um parâmetro do AG. A utilização de um valor elevado de mutações pode ocasionar problemas com a convergência da população, pois os cromossomos filhos poderão não se assemelhar aos cromossomos pais.

Após o processo de mutação, o membro resultante é incluso na população, porém os pais, ao contrário de alguns AGs, permanecem na população. Dessa forma, a população nunca pára de crescer, levando a uma queda de desempenho da aplicação. Para solucionar este problema foi criada mais uma etapa no AG, na qual é feita a remoção do membro menos apto da população. Isso garante um tamanho fixo para a população e também um desempenho mais uniforme para o algoritmo.

4.3.2.8 Verificação de parada

Nesta etapa é feita uma varredura em todos os membros da população, em busca de um cromossomo ótimo. É caracterizado um cromossomo ótimo quando o mesmo não possui penalidades de colisões de horários e seu grau de disponibilidades de horários dos professores for totalmente satisfeito ou maior do que o grau configurado no AG. Caso não sejam satisfeitas as condições necessárias para finalizar a execução, o algoritmo reiniciará o processo a partir da etapa de avaliação da população. Este processo se repete até que o

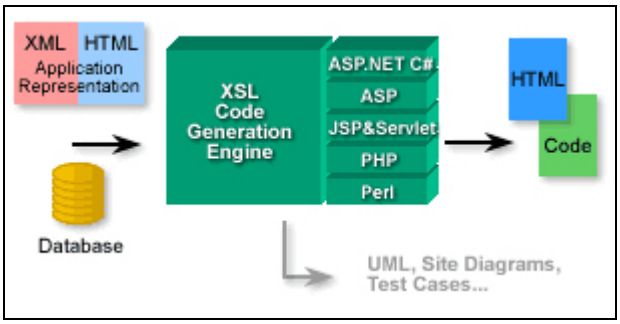
algoritmo encontre um individuo ótimo ou atinja o limite máximo de gerações.

4.3.3 Implementação da interface web

A implementação da interface web foi feita utilizando o *framework* Code Charge Studio 3.2 (YES SOFTWARE, 2007). O Code Charge Studio (CCS) é uma ferramenta que automatiza a construção de web sites. Basicamente, ele utiliza um banco de dados como base para a criação de páginas web. Possui, dentre as diversas funcionalidades: opções que permitem a criação de telas para cadastro, pesquisa e listagens de forma rápida e visual, dispensando a programação de algumas partes, como a persistência de dados.

O CCS é compatível com diversos SGBDs, como, por exemplo: MySQL, Postgree, Oracle, SQL Server, MS Access e outros. Também dá suporte a várias linguagens de programação tais como: ASP, .NET, C#, JSP & Servlets, PHP e Perl. A ferramenta utiliza *templates* e gera os códigos conforme a linguagem e banco de dados escolhidos na criação do projeto. Para a implementação da interface web foi utilizado o banco de dados MySQL 5.0 e a linguagem de programação Java Servlets. Primeiramente foi feita a modelagem de dados utilizando o Case Studio 2.3 (CHARONWARE, 2006); a seguir, foi criada a base de dados; então, foram criadas as telas através do CCS e, por fim, a biblioteca contendo o AG foi adicionada ao projeto do CCS. A biblioteca foi feita em Java através da ferramenta Eclipse 3.2

O funcionamento do CCS é simples, com pouco esforço do programador é possível criar aplicações complexas com acesso à base de dados. O CCS funciona com base em eventos. A interação do usuário com a página web dispara eventos, para os quais o programador pode utilizar ações existentes no próprio CCS ou criar suas próprias ações, como por exemplo, a chamada de uma classe Java externa. Através da criação destes eventos com chamada de classes externas, foi possível fazer a integração da interface web com a biblioteca criada para o AG. Na Figura 21, é representado o esquema de funcionamento básico do CCS.



Fonte: YES SOFTWARE (2007).

Figura 21 – Funcionamento do CCS

A ferramenta faz uso de uma base de dados para a criação de suas páginas, na qual são criados dois arquivos principais: um arquivo XML para as configurações dos campos e validações e um outro arquivo HTML, o qual contém comentários específicos para controle interno Também são criados alguns códigos-fonte. Então, a ferramenta gera o código conforme a linguagem escolhida; os fontes compilados podem ser acrescentados no servidor de aplicação escolhido, permitindo, dessa forma, a execução da aplicação

4.4 OPERACIONALIDADE

Nesta seção são apresentadas as principais telas do sistema. Na Figura 22, é apresentado o cadastro de professores.

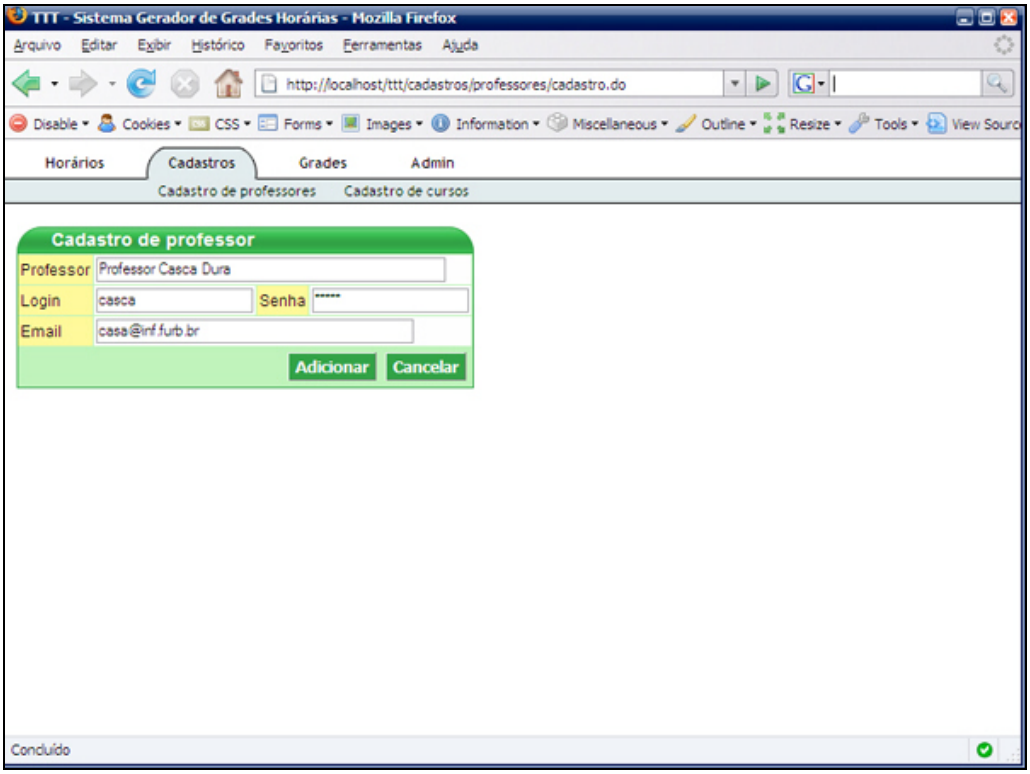


Figura 22 – Tela: cadastro de professores

Através da tela representada na Figura 22 é possível cadastrar um professor, sendo utilizado no AG para efetuar a geração das grades horárias. Nesta tela, também é cadastrado o usuário e a senha para o professor, que poderá acessar o sistema para informar suas disponibilidades de horários. Na Figura 23, é apresentada a tela onde o professor pode informar suas disponibilidades de horários.

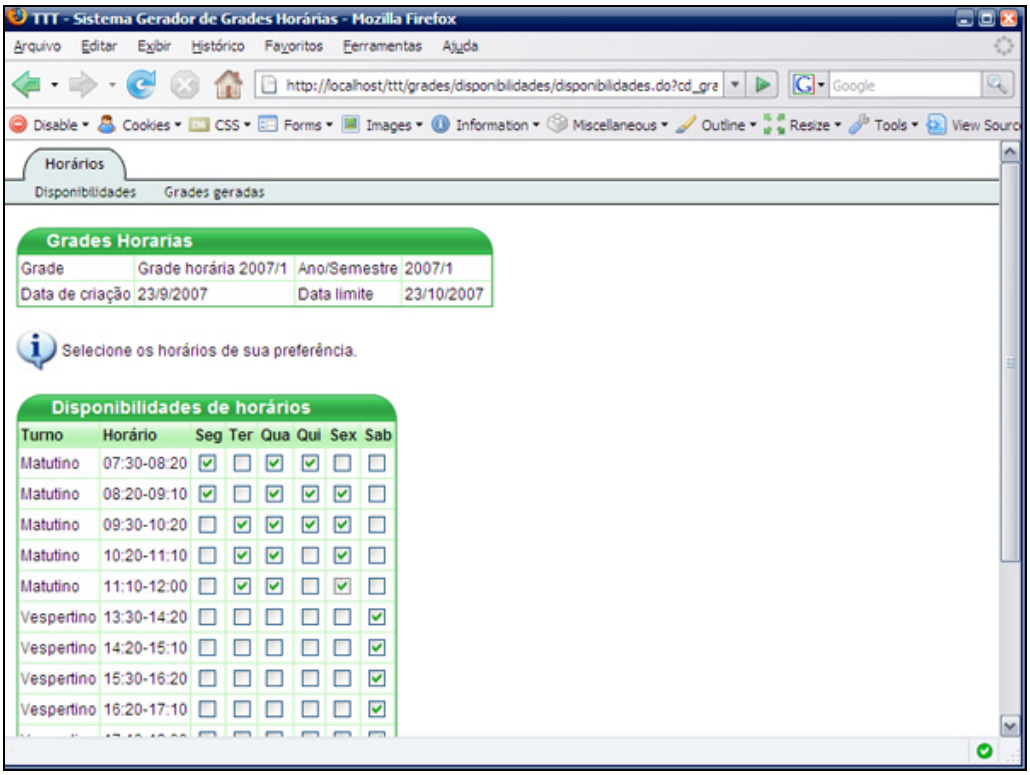


Figura 23 – Tela: disponibilidades de horários

A tela representada na Figura 23 é exclusiva para os professores, na qual o usuário seleciona seus horários preferenciais. Na Figura 24, é representada a visualização da grade horária gerada.

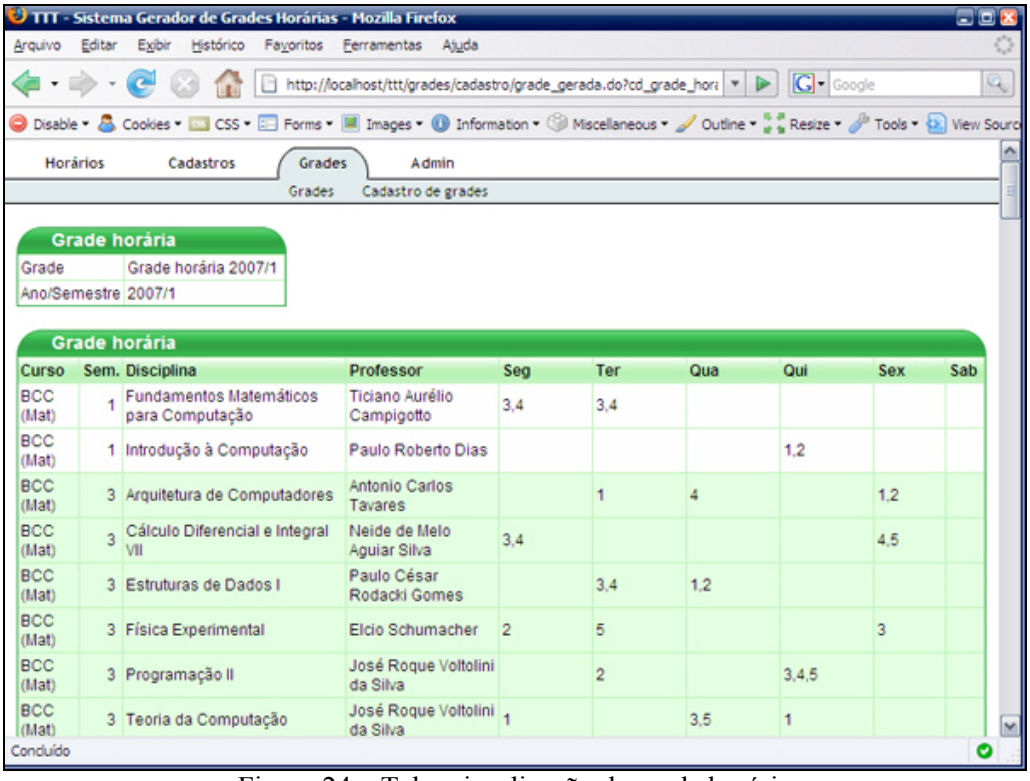


Figura 24 – Tela: visualização da grade horária

A tela de visualização de grades horárias geradas é representada na Figura 24, tanto usuários administradores como professores possuem acesso a esta tela. Finalmente tem-se a

tela, representada na Figura 25, onde é feita a seleção da configuração da grade horária para a geração.

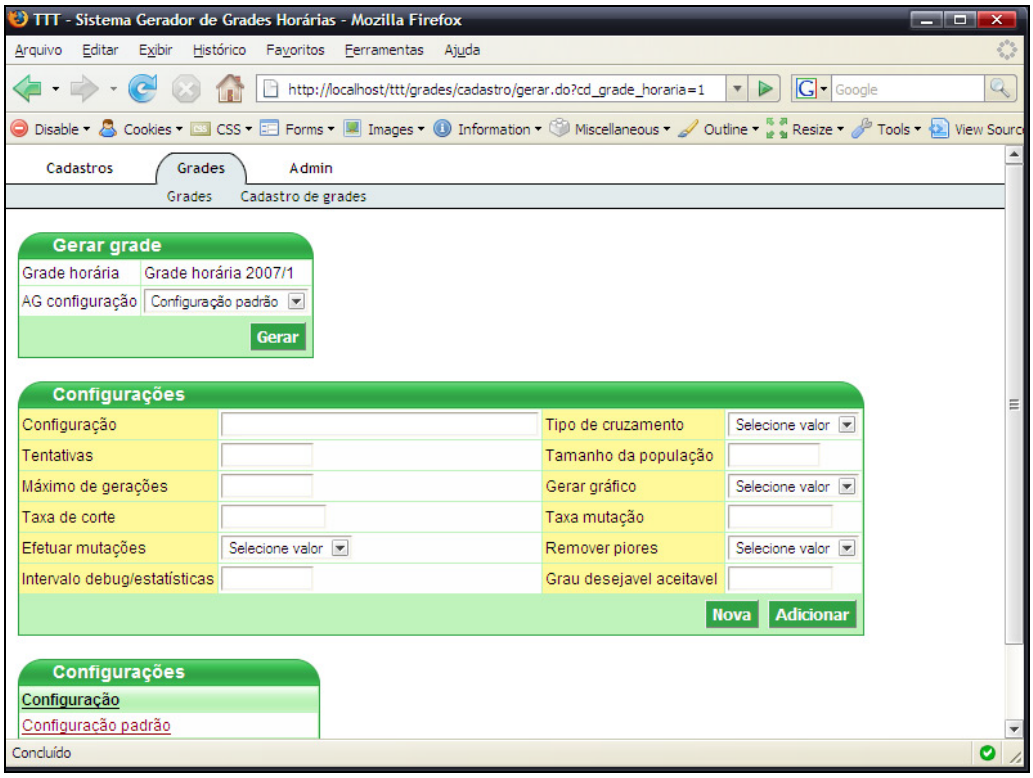


Figura 25 – Tela: gerar grade horária

Através da tela acima o usuário seleciona a configuração para o AG e pode gerar a grade horária desejada. Também é possível efetuar o cadastro de configurações do AG. A Figura 26 representa a tela onde é gerada a grade horária.

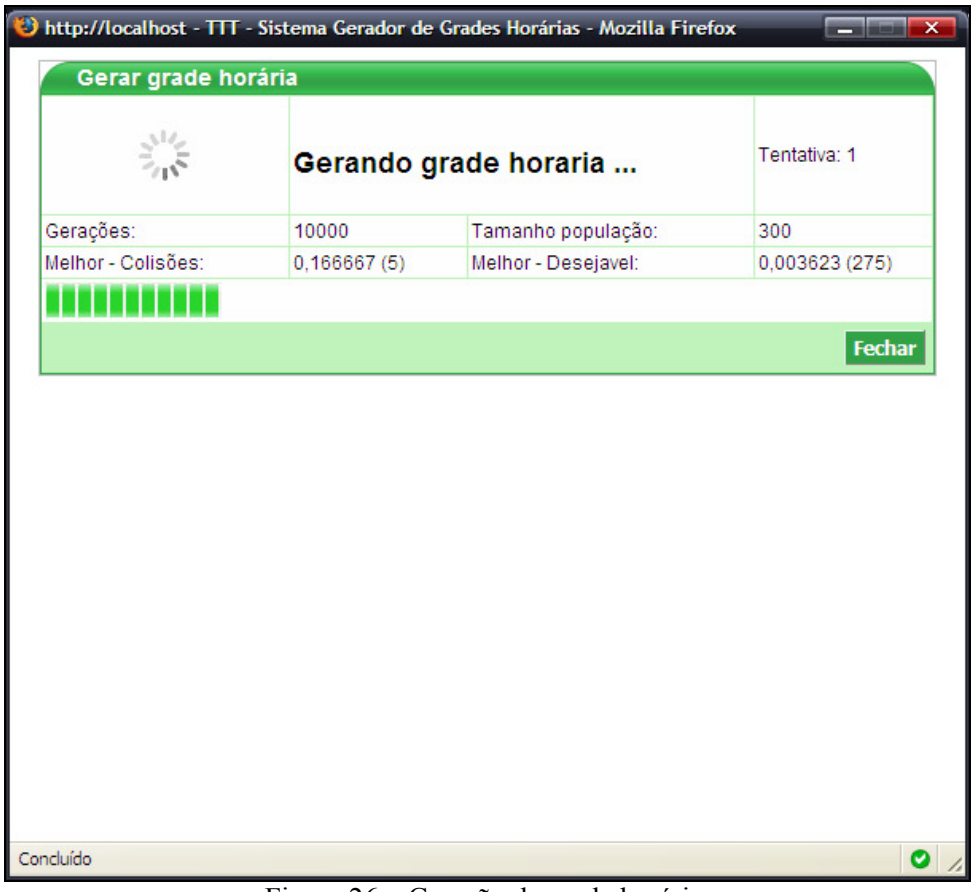


Figura 26 – Geração de grade horária.

A tela representada na Figura 26 possui alguns campos nos quais se pode acompanhar a evolução do algoritmo. Tais como: coeficiente do melhor indivíduo, tamanho da população e progresso em relação ao máximo de gerações.

4.5 RESULTADOS

O algoritmo genético implementado se mostrou eficaz na geração da solução sem que haja colisão nos horários. O tempo de execução e qualidade da solução do algoritmo varia conforme alguns fatores, tais como:

a) tamanho da população: com uma população maior o algoritmo torna-se mais lento, porém o espaço de busca a ser explorado é maior. Já com uma população menor, o algoritmo é bem mais ágil, mas o espaço de busca fica comprometido. Após diversos testes observou-se que com uma população de tamanho médio, em torno de 250 membros, o algoritmo é mais eficiente;

b) coeficiente de mutação: através da configuração do parâmetro que representa o número de mutações do AG, o desempenho e qualidade das soluções são influenciados. Com muitas mutações notou-se uma queda no desempenho do algoritmo e os filhos gerados não se assemelhavam tanto aos pais, o que pode comprometer a busca da solução ótima. Já com poucas mutações, em muitos casos, o AG não consegue eliminar as colisões de horário. Também, como item anterior, deve-se utilizar um valor mediano nesta configuração;

c) número de gerações: com este parâmetro pode-se determinar qual o máximo de gerações que o algoritmo irá executar em busca da solução. Com um valor maior, o algoritmo ficará muito tempo executando, porém, terá maiores possibilidades de chegar à solução, com um valor baixo, talvez não consiga explorar de forma satisfatória o espaço de busca.

No Quadro 9, são apresentados alguns testes efetuados com o protótipo. Foi levado em consideração apenas o grau indicando as colisões nos horários Através do quadro é possível fazer um comparativo entre os parâmetros de configuração utilizados para as mutações, tamanho da população e número máximo de gerações. As linhas em negrito são soluções ótimas, sem colisão noshorários.

Gerações		Taxa	Tamanho	Melhor	Colisões	Tempo
Máximo	Realizadas	mutações	população	solução		
10000	**5064**	**0,0015**	**100**	**1,000000**	**0**	**00:19**
10000	10000	0,0015	300	0,200000	4	00:34
10000	10000	0,0015	500	0,083333	11	00:38
20000	20000	0,0015	100	0,500000	1	00:59
20000	20000	0,0015	300	0,500000	1	00:58
20000	**19316**	**0,0015**	**500**	**1,000000**	**0**	**01:02**
30000	30000	0,0015	100	0,500000	1	01:40
30000	**15645**	**0,0015**	**300**	**1,000000**	**0**	**00:50**
30000	**20311**	**0,0015**	**500**	**1,000000**	**0**	**01:03**
10000	**5619**	**0,0500**	**100**	**1,000000**	**0**	**00:20**
10000	10000	0,0500	300	0,166667	5	00:29
10000	10000	0,0500	500	0,142857	6	00:35
20000	**4149**	**0,0500**	**100**	**1,000000**	**0**	**00:15**
20000	**14967**	**0,0500**	**300**	**1,000000**	**0**	**00:56**
20000	20000	0,0500	500	0,500000	**1**	01:11
30000	**4680**	**0,0500**	**100**	**1,000000**	**0**	**00:18**
30000	**7791**	**0,0500**	**300**	**1,000000**	**0**	**00:27**
30000	30000	0,0500	500	0,500000	1	01:32
10000	10000	0,1500	100	0,500000	1	00:50
10000	10000	0,1500	300	0,166667	5	00:51
10000	10000	0,1500	500	0,090909	10	00:52
20000	20000	0,1500	100	0,500000	1	01:50
20000	20000	0,1500	300	0,333333	2	01:44
20000	20000	0,1500	500	0,142857	6	01:53

30000	**19168**	**0,1500**	**100**	**1,000000**	**0**	**01:48**
30000	30000	0,1500	300	0,500000	1	02:40
30000	30000	0,1500	500	0,500000	1	03:13

Quadro 10 – Testes na execução do AG

Com a série de testes apresentada no Quadro 10 pode-se notar que não há uma melhor ou pior configuração para o AG, pois este problema é combinatório, dependendo da população inicial gerada ou das mutações, o algoritmo pode não encontrar uma solução adequada. Porém, podem-se notar algumas características como: com o aumento no número de mutações o algoritmo tornou-se menos eficiente. Dentre nove testes feitos utilizando um coeficiente de mutações maior, em apenas uma das tentativas o algoritmo chegou à solução sem nenhuma colisão nos horários. Já com a utilização de um coeficiente mais baixo, dentre nove tentativas, foram encontradas quatro soluções; contudo utilizando-se um valor mediano, o AG mostrou-se mais eficiente: dentre nove tentativas, foram obtidas cinco soluções. No Apêndice A, são representadas grades horárias originais da FURB e grades geradas através do protótipo.

Para os testes foram utilizados quatro cursos, cento e dezoito disciplinas e cinqüenta e três professores. Tendo, em média, apenas um professor vinculado a cada disciplina. A máquina utilizada nos testes foi um Intel Centrino Core 2 Duo, 1,66 GHz, 2 megabytes de memória, cache com 1 gigabyte de memória, ram padrão DDR2.

5 CONCLUSÕES

No desenvolvimento deste trabalho foram atingidos os objetivos propostos Primeiramente foi feita uma abordagem teórica sobre algoritmos genéticos e problemas de *timetabling*; então, foi desenvolvido um algoritmo genético para a geração de grades horárias, conforme estrutura adotada pela FURB; por fim foi feita uma interface web para a aplicação.

O algoritmo genético desenvolvido mostrou-se extremamente eficiente na geração de grades horárias sem que haja colisões, mesmo quando submetido a testes reais com diversos cursos e disciplinas. Porém, não se mostrou tão eficiente na geração de soluções que levam em consideração as restrições de horário impostas pelos professores. Na grande maioria dos casos, o algoritmo consegue gerar soluções ótimas no quesito de não possuir colisões de horários, entretanto, as restrições de horários de professores não são totalmente satisfeitas. Também ocorreram alguns problemas relacionados às mutações, algumas aulas de uma mesma disciplina, em um mesmo dia, não ficam agrupadas.

Os problemas relacionados às disponibilidades dos professores ocorrem devido ao fato dos cromossomos possuírem dois coeficientes para representar seu grau de aptidão, sendo um para as colisões de horário e outro para as preferências de horário. Em virtude disso, a seleção dos membros mais aptos para reprodução é afetada, pois o grau de penalidades tem peso maior nos critérios de seleção. Também o surgimento de super-indivíduos pode ocasionar convergências da população para um no qual não é possível satisfazer as restrições dos professores.

Por fim, identificou-se a grande importância na escolha de uma boa representação para os cromossomos, quando representados de forma mais simples, esses facilitam as demais etapas da implementação do AG. As ferramentas utilizadas na construção do protótipo supriram de forma satisfatória as necessidades encontradas e as referências utilizadas serviram como sustentação para este trabalho e vieram a enriquecer ainda mais o projeto.

5.1 EXTENSÕES

Como extensão do presente trabalho, primeiramente fica a correção no problema que ocorre com as preferências nos horários dos professores. Através da alteração dos códigos-

fonte para a utilização de apenas um coeficiente no grau de aptidão certamente virá a suprir estas dificuldades. Também é necessário efetuar alterações nas funções de mutações para solucionar o problema do agrupamento de horários.

Sugere-se ainda, para trabalhos futuros, a inclusão de novas funcionalidades para fazer a distribuição de salas de aula, laboratórios e equipamentos. Para isso podem ser feitas alterações nas estruturas dos cromossomos, adicionando mais alguns alelos às características, como, por exemplo, as salas de aula.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVARENGA, F. V. et Al. Algoritmo heurístico para construção de tabela de horário escolar. In: CONGRESSO CIÊNTIFICO, 4., 2005, Palmas. **Anais**... Palmas, TO: Universidade Luterana do Brasil, 2005. Não paginado. Disponível em: <http://www.ulbra-to.br/eventos/congresso2005/doc/artigo.aspx?aid=163>. Acesso em: 15 maio 2007.

APACHE. **Tomcat.** [S. l.]. 2007. Disponível em: <http://tomcat.apache.org/download-55.cgi>. Acesso em: 22 out. 2007.

BARRETO, G. A. **Introdução aos algoritmos genéticos**. Fortaleza, 2007. Disponível em: <http://www.deti.ufc.br/~guilherme/TI016/AG_parte1.pdf>. Acesso em: 04 out. 2007.

BRAZ, O. O. J. **Otimização de horários em instituições de ensino superior através de algoritmos genéticos**. 2000. 144 f. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://teses.eps.ufsc.br/defesa/pdf/3436.pdf>. Acesso em: 29 maio 2007.

CAMPOS, F. N. **Algoritmo genético na resolução do problema de grade horária.** 2002. 21 f. Projeto final (Curso de Ciência da Computação) – Departamento de Ciência da Computação, Universidade federal da Bahia, Salvador. Disponível em: <http://www.dcc.ufba.br/~frieda/mat057/trabalhos/flaviopdf.zip>. Acesso em: 29 maio 2007.

CARVALHO, A. P. L. F. **Algoritmos genéticos**. São Paulo, [2004]. Disponível em: <http://www.icmc.usp.br/~andre/research/genetic>. Acesso em: 10 out. 2007.

CATARINA, A. S. **Um algoritmo genético com representação explicita de relacionamentos espaciais para modelagem sócio-ambiental**. 2006. 53 f. Tese (Doutorado em Computação) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

CHARONWARE. **Case studio**: database design, modeling tool. Ostrava, República Checa, 2006. Disponível em: <http://www.casestudio.com>. Acesso em: 21 out. 2007.

CONSTANTINO, A. A. **Otimização combinatória.** Maringá, 2003. Disponível em: <http://www.din.uem.br/~ademir/Otimizacao.html>. Acesso em: 22 maio 2007.

COSTA, E. O.; BRUNA, M. D. Resolução do "timetabling" utilizando evolução cooperativa. **Revista Eletrônica de Iniciação Científica**, Curitiba, v. 3, n. 1, 2003. Não paginado. Disponível em: <http://www.sbc.org.br/reic/edicoes/2003e1/>. Acesso em: 15 maio 2007.

ECLIPSE. **An open development platform**. [S. l.], 2007. Disponível em: <http://www.eclipse.org>. Acesso em: 22 out. 2007.

FREITAS, C. C. **Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horários escolar**. Salvador, 2006. Disponível em: <http://manoelnetom.googlepages.com/ArtigoERBASE_Full.Apresentado.pdf>. Acesso em: 30 out. 2007.

GEHA. **Urânia**Software para elaboração de quadros horários de aula de professores. Curitiba, 2007. Disponível em: <http://www.horario.com.br/horario_esc.php>. Acesso em: 30 out. 2007.

LIDEN, R. **Algoritmos genéticos**: uma importante ferramenta da inteligência computacional. Rio de Janeiro: Brasport, 2006.

LUCAS, D. C. **Algoritmos genéticos**: uma introdução. Porto Alegre, 2002. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Acesso em: 23 maio 2007.

JUDE. **System design tool**. Tóquio, Japão, 2006. Disponível em: <https://jude.change-vision.com>. Acesso em: 21 out. 2007.

NOGUEIRA, A. **Casos de uso: cenários**. [S. l.], 2006. Disponível em: <http://www.imasters.com.br/artigo/3811/uml/casos_de_uso_cenarios/>. Acesso em: 30 out. 2007.

MEDEIROS, S. C. D. **Inversão de parâmetros em dados sísmicos por algoritmos genéticos**. 2005. 89 f. Dissertação (Mestrado em engenharia elétrica) – Curso de Pós-Graduação em Engenharia Elétrica, Pontifica Universidade Católica, Rio de Janeiro.

MIRANDA, M. N. **Algoritmos genéticos**: fundamentos e aplicações. Rio de Janeiro, [2007]. Disponível em: <http://www.gta.ufrj.br/~marcio/genetic.html>. Acesso em: 21 out. 2007.

MYSQL. **The world's most popular open source database**. Estados Unidos, [2007]. Disponível em: <http://dev.mysql.com/downloads/mysql/5.0.html>. Acesso em: 22 out. 2007.

PERSONA, L.; CORRÊA, P. L. P.; SARAIVA, A. M. Algoritmo genético GARP para modelagem ambiental. In: CONGRESSO BRASILEIRO DA SOCIEDADE BRASILEIRA APLICADA À AGROPECUÁRIA E À AGROINDÚSTRIA, 2003, Porto Seguro, **Anais...** Não paginado. Disponível em: <http://www.lucaspersona.com.br/files/PaperSBIAgro.pdf>. Acesso em: 11 out. 2007.

RODRIGUES, N. M.; ALMEIDA, C. P.; GONÇALVES, R. A. Um algoritmo genético para o problema do despacho econômico ambiental. In: ESCOLA REGIONAL DE INFORMATICA DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO – PARANÁ, 14., 2007, Guarapuava. **Anais...** Guarapuava: SBC, 2007. Não paginado, 12 f. Disponível em: <http://bibliotecadigital.sbc.org.br/download.php ?paper=684>. Acesso em: 14 out. 2007.

SCHOEFFEL, P. **Protótipo de um sistema de controle de reservas de sala de aula da Universidade Regional de Blumenau.** 2001. 65 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SCIELO. **Química nova – genetic algorithm in chemeitry**: algoritmo genético em química. São Paulo, 1998. Disponível em: <http://www.scielo.br/scielo.php?pid=S0100-40421999000300019&script=sci_arttext>. Acesso em: 17 out. 2007.

SORROCHE, R. **Sistema de auxílio à matrícula de alunos utilizando Java 2 Enterprise Edition**. 2002. 108 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SPARX SYSTEMS. **Enterprise Architect**. Victoria, Australia, 2005. Disponível em: <http://www.sparxsystems.com.au/products/ea.html>. Acesso em: 21 out. 2007.

YES SOFTWARE. **Code Charge Studio 3.2**. Las Vegas, 2007. Disponível em: <http://www.yessoftware.com/products/product.php?product_id=1>. Acesso em: 21 out. 2007.

APÊNDICE A – Comparativos das grades horárias

Nesta seção são demonstradas as grades horárias originais da FURB do ano 2007/1, dos cursos de Ciência da Computação noturno e matutino, Licenciatura em Computação noturno e Sistemas de Informação noturno. Também são demonstradas as mesmas grades geradas através do protótipo. No Quadro 11 é apresentada a grade horária do curso de Sistemas de informação original e uma grade gerada pelo protótipo.

FASE	DISCIPLINAS	Grade horária - Original						Grade horária - Protótipo					
		Seg	Ter	Qua	Qui	Sex	PROFESSOR	Seg	Ter	Qua	Qui	Sex	PROFESSOR
PRIMEIRA	Funções Empresariais I		14/15			14/15	Terezinha Vicenti		12/15				Terezinha Vicenti
	Programação de Computadores	14/15		12/15			Wilson Pedro Carli Maurício Capobianco Lopes				12 15	12/15	Wilson Pedro Carli
	Arquitetura de Computadores I	12/13				12/13	Miguel Alexandre Wisintainer			12/13	13/14		Miguel Alexandre Wisintainer
	Fundamentos Matemáticos para Sistemas de Informação		12/13		12/15		Evandro Felin Londero	12/15		14/15			Evandro Felin Londero
SEGUNDA	Funções Empresariais II		12/13			12/13	Terezinha Vicenti				12/15		Terezinha Vicenti
	Estatística Aplicada à Informática		14/15	12/13			Henriette Damm Friske	14/15	14/15				Henriette Damm Friske
	Programação Orientada a Objetos			14/15	12/13		Mauro Marcelo Mattos		12/13	12/13			Mauro Marcelo Mattos
	Teoria Geral de Sistemas	14/15				14/15	Ricardo Alencar de Azambuja	12/13				12/13	Ricardo Alencar de Azambuja
TERCEIRA	Funções Empresariais III	14/15				14/15	Edson Luis Francês	12/13			14/15		Edson Luis Francês
	Análise Econômica de Empreendimentos		12/13			12/13	Wagner Alfredo D'Ávila				12/13	12/13	Wagner Alfredo D Ávila
	Universidade, Ciência e Pesquisa	12/13			14/15		Maria da Conceição Lima de Andrade			12/13		14/15	Maria da Conceição Lima de Andrade
	Estruturas de Dados			12/13	12/13		Antonio Carlos Tavares	14/15		14/15			Antonio Carlos Tavares
QUARTA	Banco de Dados		12/13		14/15		Alexander Roberto Valdameri		14/15			14/15	Alexander Roberto Valdameri
	Programação Web		14/15			14/15	Adilson Vahldick	12/13		12/13			Adilson Vahldick
	Projeto de Software I	12/13		12/13			Fabiane Barreto Vavassori Benitti				12/13	12/13	Fabiane Barreto Vavassori Benitti
	Fundamentos de Sistemas de Informação				12/13	12/13	Ricardo Alencar de Azambuja		12/13	14/15			Ricardo Alencar de Azambuja
	Desafios Sociais Contemporâneos	14/15		14/15			Gilberto Friedenreich dos Santos	14/15			14/15		Gilberto Friedenreich dos Santos
QUINTA	Pesquisa Operacional	12/13			14/15		Osmar Leonardo Kuhnen					12/15	Osmar Leonardo Kuhnen
	Prática em Banco de Dados			14/15	12/13		Adilson Vahldick	14/15	12/13				Adilson Vahldick

	Comunicação de Dados e Redes de Computadores	14/15				14/15	Francisco Adell Péricas				12/15		Francisco Adell Péricas
	Fundamentos de Sistemas Operacionais		12/13			12/13	Antonio Carlos Tavares		14/15	12/13			Antonio Carlos Tavares
SEXTA	Gestão de Pequenos Negócios		12/14				Jerusa Betina Schroeder			14/15		12/12	Jerusa Betina Schroeder
	Inteligência Artificial I			12/15			Paulo de Tarso Mendes Luna				12/13	14/15	Paulo de Tarso Mendes Luna
	Legislação em Informática					14/15	Alejandro Knaesel Arrabal				14/15		Alejandro Knaesel Arrabal
	Análise e Projetos de Sistemas I				12/15		Everaldo Artur Grahl	12/15					Everaldo Artur Grahl
	Gerência de Projetos de Informática		15/15			12/13	Evaristo Baptista		12/13			13/13	Evaristo Baptista
	Tóp. em Sist. de Informações II - Optativa	12/15					Paulo Roberto Dias		14/15	12/13			Paulo Roberto Dias
SÉTIMA	Tóp. em Desenvolvimento de Sistemas - Opt.	14/15					Adilson Vahldick				14/15		Adilson Vahldick
	Qualidade de Software			12/13		14/15	Carlos Eduardo Negrão Bizzotto	14/15		12/13			Carlos Eduardo Negrão Bizzotto
	Multimídia				12/13	12/13	Carlos Eduardo Negrão Bizzotto	12/13				14/15	Carlos Eduardo Negrão Bizzotto
	Trabalho de Conclusão de Curso I			14/15	14/15		Dalton Solano dos Reis		12/13	14/15			Dalton Solano dos Reis
	Computador e Sociedade		14/15				Paulo Roberto Dias				12/13		Paulo Roberto Dias
OITAVA	Trabalho de Conclusão de Curso	12/13			12/13		Wilson Pedro Carli	14/15		14/15			Wilson Pedro Carli

Quadro 11 – Grade horária: Sistemas de informação

No Quadro 12 são apresentadas as grades horárias do curso de Ciência da computação noturno.

FASE	DISCIPLINAS	Grade horária - Original							Grade horária - Protótipo					
		Seg	Ter	Qua	Qui	Sex	Sáb	PROFESSOR	Seg	Ter	Qua	Qui	Sex	PROFESSOR
PRIMEIRA	Programação de Computadores		12/15		14/15			Maurício Capobianco Lopes Mauro Marcelo Mattos Carlos Henrique Correia			12/13		14/15	Maurício Capobianco Lopes
	Introdução à Computação	12/13						Wilson Pedro Carli Mauro Marcelo Mattos				14/15		Mauro Marcelo Mattos
	Computação Digital			12/13		14/15		Miguel Alexandre Wisintainer	14/15				12/13	Miguel Alexandre Wisintainer
	Universidade, Ciência e Pesquisa	14/15			12/13			Maria da Conceição Lima de Andrade		12/15				Maria da Conceição Lima de Andrade
	Fundamentos Matemáticos para Computação			14/15		12/13		Luiz Heinzen Ticiano Aurélio Campigotto	12/13			12/13		Luiz Heinzen

SEGUNDA	Fundamentos em Computação Digital		14/15		12/13			Miguel Alexandre Wisintainer	12/13	14/15				Miguel Alexandre Wisintainer
	Lógica para Computação			12/13		12/13		Jomi Fred Hübner	14/15		14/15			Jomi Fred Hubner
	Programação I		12/13		14/15			Marcel Hugo Carlos Henrique Correia			12/13		12/13	Carlos Henrique Correia
	Álgebra Linear e Geometria Analítica	12/15		14/15				Evandro Felin Londero		12/13		12/13	14/15	Evandro Felin Londero
TERCEIRA	Programação II		12/13		14/15			Adilson Vahldick		14/15	14/15			Adilson Vahldick
	Arquitetura de Computadores	14/15		14/15				Antônio Carlos Tavares		12/13			14/15	Antonio Carlos Tavares
	Teoria da Computação	12/13			12/13			José Roque Voltolini da Silva	12/13				12/13	José Roque Voltolini da Silva
	Cálculo Diferencial e Integral VII			12/13		14/15		Neide de Melo Aguiar Silva			12/13	14/15		Neide de Melo Aguiar Silva
	Estrutura de Dados I		14/15			12/13		Paulo César Rodacki Gomes	14/15			12/13		Paulo César Rodacki Gomes
QUARTA	Linguagens Formais		14/15					Joyce Martins				14/15		Joyce Martins
	Tópicos em Programação I			12/13				Mauro Marcelo Mattos	15/15			12/13		Mauro Marcelo Mattos
	Prática em Arquitetura de Computadores	14/15		14/14				Miguel Alexandre Wisintainer		12/13			14/14	Miguel Alexandre Wisintainer
	Cálculo Diferencial e Integral VIII	12/13				12/13		Neide de Melo Aguiar Silva	12/12		14/15		15/15	Neide de Melo Aguiar Silva
	Estatística IV				14/15	14/15		Arthur Alexandre Hackbarth Neto	13/14				12/13	Arthur Alexandre Hackbarth Neto
	Estrutura de Dados II		12/13		12/13			Paulo César Rodacki Gomes		14/15	12/13			Paulo César Rodacki Gomes
QUINTA	Compiladores	12/13		12/13				Joyce Martins	14/14				12/14	Joyce Martins
	Sistemas Operacionais		15/15		14/15	14/15		Antônio Carlos Tavares	12/13			13/15		Antonio Carlos Tavares
	Linguagens de Programação	14/15				12/13		José Roque Voltolini da Silva		14/15		12/12	15/15	José Roque Voltolini da Silva
	Física Experimental		12/13 14/14					Élcio Schumacher		12/13	12/12			Elcio Schumacher
	Banco de Dados I			14/15	12/13			Alexander Roberto Valdameri	15/15		13/15			Alexander Roberto Valdameri
SEXTA	Inteligência Artificial		12/13	14/15				Jomi Fred Hübner	12/13	14/15				Mauro Marcelo Mattos
	Transmissão de Dados	14/15				14/15		Sérgio Stringari	14/15		14/15			Sérgio Stringari
	Cálculo Numérico	12/13			12/13			Andresa Pescador			12/13		14/15	Andresa Pescador
	Relações Humanas		14/15		14/15			Vivane Giombelli		12/13			12/13	Vivane Giombelli
	Banco de Dados II			12/13		12/13		Adilson Vahldick				12/15		Alexander Roberto Valdameri
SÉTIMA	Administração III	12/13		14/15				André Buzzi	13/14		12/13			André Buzzi
	Redes de Computadores		14/15			12/13		Francisco Adell Péricas	12/15				12/13	Francisco Adell Péricas
	Computação Gráfica			12/13	14/15			Paulo César Rodacki Gomes			14/15		14/15	Paulo César Rodacki Gomes
	Automação e Controle				12/13	14/15		Denísio Lourenço Lobo		12/13		14/15		Denísio Lourenço Lobo

	Engenharia de Software	14/15	12/13					Everaldo Artur Grahl		14/15		12/13		Everaldo Artur Grahl
OITAVA	Sistemas Multimídia				12/13	12/13		Dalton Solano dos Reis	14/15			13/13	12/12	Dalton Solano dos Reis
	Filosofia V		13/15					Leonir Alba		15/15	12/14			Leonir Alba
	Requisitos de Software	14/15		14/15				Fabiane Barreto Vavassori Benitti		12/13		14/15		Fabiane Barreto Vavassori Benitti
	Tópicos Especiais I - Optativa	12/13	12/12					Francisco Adell Péricas		14/14	13/15			Fracisco Adell Péricas
	Tópicos Especiais II - Optativa			12/13			2/2	Paulo Fernando da Silva	12/13			12/12		Paulo Fernando da Silva
	Tópicos Especiais III - Optativa				14/15			Jomi Fred Hubner					13/14	Jomi Fred Hubner
NONA	Trabalho do Conclusão de Curso |			12/13		14/15		Dalton Solano dos Reis	12/13			14/15		Dalton Solano dos Reis
	Sistemas Distribuídos			14/15	14/15			Paulo Fernando da Silva		14/15	14/15			Paulo Fernando da Silva
	Projeto de Software	12/13	14/15					Everaldo Artur Grahl			12/13		12/13	Everaldo Artur Grahl
	Tópicos em Desenvolvimento de Sistemas II - Optativa	14/15						Adilson Vahldick				13/13	14/15	Adilson Vahldick
	Computador e Sociedade				12/13			Paulo Roberto Dias		12/13				Paulo Roberto Dias
Décima	Trabalho de Conclusão de Curso II			12/15				José Roque Voltolini da Silva	14/15			14/15		José Roque Voltolini da Silva

Quadro 12 – Grade horária: Ciência da computação noturno

No Quadro 13 são representadas as grades horárias do curso de Licenciatura em computação, grade horária original e grade gerada através do protótipo.

FASE	DISCIPLINAS	Grade horária - Original						Grade horária - Protótipo					
		Seg	Ter	Qua	Qui	Sex	PROFESSOR	Seg	Ter	Qua	Qui	Sex	PROFESSOR
TERCEIRA	Sistemas de Informação I				12/13	12/13	Ricardo Alencar de Azambja			12/13		14/15	Ricardo Alencar de Azambuja
	Programação II		12/13		14/15		José Roque Voltolini da Silva Marcel Hugo Maurício Capobianco Lopes Mauro Marcelo Mattos	14/15			12/13		Maurício Capobianco Lopes
	Currículo e Didática - EAL	12/13		14/15			Maria Luci Bittencourt	12/13	14/15				Maria Luci Bittencourt
	Psicologia da Educação - EAL	14/15		12/13			Leonida Pinto		12/13		14/15		Leonida Pinto
QUARTA	Metodologia do Ensino de Informática			14/15		12/13	Carlos Eduardo Negrão Bizzotto Paulo Roberto Dias		14/15		12/13		Carlos Eduardo Negrão Bizzotto
	Legislação em Informática					14/15	Alejandro Knaesel Arrabal	14/15					Alejandro Knaesel Arrabal
	Humanidade, Educação e Cidadania - EAL		12/13		14/15		Tarcísio Alfonso Wickert			12/13		12/13	Tarcísio Alfonso Wickert
	Estágio Curricular Supervisionado I	12/13	14/15				Tania Baier			14/15	14/15		Tania Baier
	Estrutura de Dados			12/13	12/13		Paulo César Rodacki Gomes	12/13	12/13				Paulo César Rodacki Gomes

							Antonio Carlos Tavares							

Quadro 13 – Grade horária: Licenciatura em computação

No Quadro 14 é representada a grade horária do curso de Ciência da computação matutino.

FASE	DISCIPLINAS	Grade horária - Original						Grade horária - Protótipo					
		Seg	Ter	Qua	Qui	Sex	PROFESSOR	Seg	Ter	Qua	Qui	Sex	PROFESSOR
PRIMEIRA	Introdução à Computação			3/4			Mauro Marcelo Mattos					4/5	Paulo Roberto Dias
	Programação de Computadores		3/4		1/4		Maurício Capobianco Lopes		3/5			1/3	Maurício Capobianco Lopes
	Computação Digital	3/4				1/2	Miguel Alexandre Wisintainer			1/2	1/2		Miguel Alexandre Wisintainer
	Universidade, Ciência e Pesquisa		1/2			3/4	Aparecida Beduschi Schwab		1/2		3/4		Aparecida Beduschi Schwab
	Fundamentos Matemáticos para Computação	1/2		1/2			Ticiano Aurélio Campigotto	3/4		4/5			Ticiano Aurélio Campigotto
TERCEIRA	Programação II			3/4		1/2	José Roque Voltolini da Silva				1/2 4/5		José Roque Voltolini da Silva
	Arquitetura de Computadores	4/5		1/2			Antonio Carlos Tavares			1/2		3/4	Antonio Carlos Tavares
	Teoria da Computação		1/3			3/3	José Roque Voltolini da Silva	3/5			3/3		José Roque Voltolini da Silva
	Física Experimental	1/3					Elcio Schumacher		2/3	3/3			Elcio Schumacher
	Cálculo Diferencial e Integral VII		4/5		1/2		Neide de Melo Aguiar Silva		1/1 4/4	4/5			Neide de Melo Aguiar Silva
	Estruturas de Dados I				3/4	4/5	Paulo César Rodacki Gomes	1/2				1/2	Paulo César Rodacki Gomes
QUINTA	Inteligência Artificial			1/2		4/5	Jomi Fred Hübner		4/5		1/2		Jomi Fred Hubner
	Transmissão de Dados	4/5				1/2	Sérgio Stringari	3/4			4/5		Sérgio Stringari
	Compiladores		1/2		1/2		Joyce Martins			2/2 5/5	3/3	3/3	Joyce Martins
	Cálculo Numérico	1/3		3/3			Nelson Hein			3/4		1/2	Nelson Hein
	Relações Humanas		3/5		3/3		Lucienne da Silva	1/2 5/5	1/1				Lucienne da Silva
	Banco de Dados I			4/5	4/5		Alexander Roberto Valdameri		2/3			4/5	Alexander Roberto Valdameri
SÉTIMA	Sistemas Multimídia				4/5	4/5	Dalton Solano dos Reis	4/5				4/5	Dalton Solano dos Reis
	Filosofia V				1/3		Leonir Alba		1/2		5/5		Leonir Alba
	Empreendedor em Informática			3/5			Oscar Dalfovo	3/3			3/4		Oscar Dalfovo
	Requisitos de Software	1/2		1/2			Fabiane Barreto Vavassori Benitti		3/4	3/4			Fabiane Barreto Vavassori Benitti
	Tópicos Especiais I - Optativa		1/3				Francisco Adell Péricas					1/3	Francisco Adell Péricas
	Tópicos Especiais II - Optativa					1/3	Jomi Fred Hübner	1/2		1/1			Jomi Fred Hubner
OITAVA	Trabalho de Conclusão de Curso I		3/4		3/4		Joyce Martins					1/2 4/5	Joyce Martins
	Projeto de Software	1/2		1/2			Everaldo Artur Grahl	1/2		3/4			Everaldo Artur Grahl
	Tópicos em Desenvolvimento de Sistemas I - Optativa	3/5					Fábio Rafael Segundo		3/3	1/2			Fábio Rafael Segundo
	Tópicos Especiais III - Optativa			3/5			Miguel Alexandre Wisintainer		1/2 5				Miguel Alexandre Wisintainer
	Computador e Sociedade					4/5	Paulo Roberto Dias		4/4	5/5			Paulo Roberto Dias

NONA	Trabalho de Conclusão de Curso II			1/4			José Roque Voltolini da Silva			1/4			José Roque Voltolini da Silva

Quadro 14 – Grade horária: Ciência da computação matutino

Através dos quadros acima é possível verificar que não houve colisão nos horários nas grades horárias geradas pelo protótipo. Porém, em alguns casos, ficaram representados os problemas descritos anteriormente relacionados às mutações e algumas aulas ficaram sem agrupamento.