PATRÍCIA DE SIQUEIRA RAMOS

Sistema Automático de Geração de Horários para a UFLA utilizando Algoritmos Genéticos

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Orientadora
Profa. Olinda Nogueira Paes Cardoso

Lavras
Minas Gerais - Brasil
2002

PATRÍCIA DE SIQUEIRA RAMOS

Sistema Automático de Geração de Horários para a UFLA utilizando Algoritmos Genéticos

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 16 de dezembro de 2002.

_____
Profa. Renata Couto Moreira

_____
Prof. Ricardo Martins de Abreu Silva

_____
Profa. Olinda Nogueira Paes Cardoso
(Orientadora)

LAVRAS
MINAS GERAIS – BRASIL

A Deus.
Aos meus amados pais e irmãs.

Agradecimentos

Agradeço a Deus por me iluminar e por ter-me dado uma família tão especial e pessoas que me incentivam e se orgulham de mim. Também por me ajudar a entender aquelas que não gostam de mim.
Agradeço à minha família que me compreende o tempo todo e me suporta até nos momentos em que eu não consigo tratá-los da forma como eles merecem, além de me amarem incondicionalmente.
Agradeço à minha mãe, que com sua sabedoria, paciência e amor me acalma e me aconchega nos momentos mais difíceis. Basta apenas uma palavra sua para que tudo se torne mais fácil.
Agradeço ao meu pai, que sempre me conduziu e apoiou e, apesar de parecer às vezes frio e exigente, possui um grande coração e me mostra sempre os melhores caminhos a seguir.
Agradeço à Paulinha e à Priscila que torcem por mim e estão sempre prontas a me perdoar pelos meus erros.
Agradeço ao Adô, Vaninha e Xande pelos momentos divertidos e apuros que passamos juntos em nossas reuniões de estudo e trabalhos. Obrigada também pela amizade e companheirismo. Obrigada também à Giselle pelos momentos alegres, apesar de termos tido maior contato apenas no fim do curso.
Agradeço à Vivia pela amizade sincera e desinteressada e por suportar minhas reclamações e lamentações.
Agradeço à Luana pela amizade, pelas risadas e pelo crescimento espiritual que alcancei graças ao seu incentivo.
Agradeço aos professores que me ensinaram o que eu não sabia e a correr atrás do que eu queria. À Iraziet, Toninho e Sérgio pelos conhecimentos adquiridos na minha Iniciação Científica.
Agradeço à Profa. Olinda que nesta etapa final do curso me passou tanta calma e acreditou na minha capacidade.

Resumo

O problema de formação de horários escolares, também conhecido pelo termo *Timetabling*, consiste em arranjar encontros entre professores e alunos em um período de tempo previamente fixado, de modo a satisfazer um conjunto de restrições. Muitas variantes do problema de *Timetabling* têm sido propostas na literatura, e diferem pelo tipo de instituição de ensino envolvida, universidades ou escolas médias, e pelo tipo de restrições impostas ao problema. Para a realização deste trabalho, adotou-se como modelo uma solução obtida para o *Timetabling* de um colégio, para se resolver o mesmo problema da Universidade Federal de Lavras, que conta com um sistema informatizado, porém manual. Os Algoritmos Genéticos constituem uma estratégia para solução de problemas, especialmente de otimização, onde a alta complexidade matemática não permite o uso de métodos exatos. Robustos, genéricos e facilmente adaptáveis, utilizam conceitos do princípio de seleção natural de Darwin e baseiam seu funcionamento num processo de aprendizagem e busca dentro de uma população de soluções. Esta abordagem foi escolhida para o problema da geração de grades horárias porque esta se mostra de difícil modelagem e solução devido ao grande número de restrições e fatores envolvidos, além de que forma alocar os recursos adequadamente obedecendo à estas restrições. A principal dificuldade se encontrou na combinação dos parâmetros para que uma boa solução fosse encontrada. Todos os recursos adicionados e adaptações apresentaram comportamento satisfatório em relação às grades horárias geradas, porém algumas melhoras poderiam ser incorporadas ao sistema para que o aplicativo desenvolvido possa ser usado na UFLA.

Palavras Chave: Algoritmos Genéticos, Gestão Acadêmica, *Timetabling*

Abstract

The problem of formation of school schedules, also known by the term *Timetabling*, consists of arranging encounters between teachers and students in a period of time previously fastened, in way to satisfy a group of restrictions. A lot of variants of the problem of *Timetabling* have been proposed in the literature, and they differ for the type of teaching institution involved, universities or medium schools, and for the type of restrictions imposed to the problem. For the accomplishment of this work, it was adopted as model a solution obtained for *Timetabling* of a school, to solve the same problem of the Universidade Federal de Lavras, that it has a computerized system, however manual. The Genetic Algorithms constitute a strategy for solution of problems, especially of optimization, where the high mathematical complexity doesn't allow the use of exact methods. Robust, generic and easily adaptable, they use concepts of natural selection of Darwin and they base its operation on a learning process and it looks for inside of a population of solutions. This approach was chosen for the problem of the generation of *Timetabling* because this it is shown of difficult modelling and solution due to the great number of restrictions and involved factors, in addition form to allocate the resources appropriately obeying these restrictions. The main difficulty was in the combination of the parameters so that a good solution was found. All of the added resources and adaptations presented satisfactory behavior in relation to the generated hourly grating, however some improvements could be incorporate to the system so that the developed application can be used in UFLA.

Words Key: Genetic algorithms, Academic Administration, *Timetabling*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

O atual quadro das Instituições Públicas brasileiras é de competição por espaço de atuação, manutenção e realização de seu orçamento pelo governo, recurso este cada vez mais limitado. Para se destacar, demonstrando os resultados obtidos e justificar ações futuras, torna-se indispensável que elas disponibilizem informação de maneira organizada, através do aprimoramento dos mecanismos de gestão. Estes mecanismos se mostram mais eficientes com modelos informatizados que auxiliam e agilizam o acompanhamento das atividades, mas o que se observa é que estas são realizadas de forma quase manual e por pessoas não especializadas em administração, na maioria das instituições federais.

No caso da Universidade Federal de Lavras - UFLA, a situação não é diferente, os profissionais envolvidos na gestão acadêmica não são formados na área de Administração e por isso nem sempre conseguem gerenciar todos os recursos e atividades da melhor maneira possível. Suas atividades estão informatizadas, ou seja, os dados são transferidos e armazenados no computador, mas não estão automatizadas. Automação é um sistema automático pelo qual os mecanismos controlam seu funcionamento, dispensando, quase por completo, a interferência do ser humano.

Neste contexto, torna-se necessária a adoção de sistemas especializados que funcionem de forma a resolver os problemas da universidade de forma acurada e padronizada. O enfoque deste trabalho se concentra na atividade de se gerar horário escolar, realizada pela Pró-Reitoria de Graduação (PRG). Uma boa grade horária é um subsídio básico para administração de tempos em uma instituição de ensino. É um problema difícil de ser modelado, o que pode ser verificado em trabalhos citados por [Júnior00], devido à complexidade matemática.

A abordagem a ser utilizada é baseada no trabalho de [Timóteo02], e utiliza os Algoritmos Genéticos como estratégia. Constitui uma atividade de manutenção de *software,* que é definida por [Pressman95] como: "para adaptar ou aperfeiçoar, devemos determinar novos requisitos, reprojetar, gerar código e testar o software existente. Tais tarefas, quando aplicadas a um programa existente, têm sido chamadas manutenção". Pressman ainda declara que a manutenção é responsável por 70% de todo o esforço despendido em uma organização de *software.*

Existem dois tipos básicos de algoritmos, os métodos exatos e as heurísticas. Os métodos exatos garantem a obtenção da melhor solução de um problema (solução ótima), e as heurísticas fazem buscas no conjunto de soluções sem garantir a otimalidade da solução final. Existem heurísticas mais gerais que, por meio de adaptações podem ser usadas para vários problemas – as meta-heurísticas - entre as quais pode-se citar *Simulated Annealing* [Abramson92], Busca Tabu e os Algoritmos Genéticos (AG) [Holland75], [Goldberg89], [Mitchell96].

Os AG são algoritmos probabilísticos que começaram a se popularizar a partir dos anos 80 e utilizam conceitos provenientes do princípio de seleção natural de Darwin para abordar uma série ampla de problemas, em especial de otimização. Fazem parte de um grupo conhecido como Algoritmos Evolutivos, e são baseados num processo coletivo de aprendizagem dentro de uma população de indivíduos, cada um dos quais representando um ponto no espaço de busca de soluções para um dado problema. A população é inicializada e evolui através de gerações com o uso dos operadores de seleção, reprodução e mutação. Durante o procedimento, é feita uma avaliação da qualidade (adaptação) dos indivíduos e essa informação é usada para conduzir o processo evolutivo, favorecendo a seleção de indivíduos bem adaptados para gerar indivíduos também bem adaptados. O mecanismo de recombinação (*crossover*) permite mesclar informações de indivíduos de uma geração e passá-las aos seus descendentes, e um processo de mutação introduz inovação na população. Robustos, genéricos e

facilmente adaptáveis, consistem de uma técnica amplamente estudada e utilizada em diversas áreas.

O grande número de restrições e fatores envolvidos [Burke95] para construir uma grade horária, como a quantidade de disciplinas, a alocação dos professores e o compartilhamento de recursos, torna este problema muito complexo. Como ressalta [Velloso95], o planejamento e alocação de recursos são problemas de solução difícil que precisam ser atacados através da combinação de técnicas de busca e heurísticas, tornando as soluções específicas para os problemas em questão. A dimensão do problema e as características específicas (restrições) da instituição constituem um aspecto importante no que diz respeito à elaboração de um processo de geração de horário, por isso, a necessidade de levantamento de requisitos próprios da PRG da UFLA. Várias são as abordagens para solucionar o problema de *timetabling,* conforme mencionado em [Timóteo02]. Na solução desenvolvida pelo mesmo e que será usada como modelo para o presente trabalho, a geração de horários escolares foi realizada para um colégio de ensino fundamental e médio, Nossa Senhora de Lourdes, de Lavras - MG. Foram desenvolvidos um algoritmo genético e uma aplicação para que os testes pudessem ser executados. O usuário deveria configurar os parâmetros genéticos, e de acordo com os testes realizados por [Timóteo02], observou-se que o método de seleção por torneio, o operador de *crossover* com corte em um ponto e o operador de mutação comum apresentaram os melhores resultados.

No momento da formulação do problema, foi necessário que [Timóteo02] se voltasse às particularidades e restrições de uma escola do Ensino Médio. Uma instituição como esta possui suas próprias necessidades e algumas informações são mais relevantes do que outras, mas que podem se tornar imprescindíveis em uma Instituição de Ensino Superior. O objetivo do trabalho é automatizar o sistema de geração de horários da UFLA, adaptando o sistema original de forma a atender as novas exigências e restrições. O sistema original é um gerador de grades horárias conhecido na literatura como *School Timetablin*g

(seqüenciamento semanal das aulas de uma escola), que foi adaptado e modificado para *Course Timetabling* (seqüenciamento semestral das aulas de um conjunto de cursos de uma universidade). A adaptação deste visa atender os requisitos específicos da PRG da UFLA e alcançar resultados tão positivos quanto os obtidos pelo modelo, satisfazendo as necessidades da Instituição.

Para esclarecer a forma como a adaptação foi realizada, a divisão deste trabalho foi feita em seis capítulos, distribuindo os temas da seguinte forma:

- Capítulo 2: explora as definições dos Algoritmos Genéticos, sua teoria, termos utilizados, conceitos e operadores, além de algumas situações em que eles podem ser empregados.
- Capítulo 3: trata do problema específico de geração de horários e explica a maneira como a solução do problema para uma escola (o modelo adotado como base para este trabalho) foi atingida.
- Capítulo 4: descreve a maneira como foram feitas as alterações, onde e porquê elas ocorreram e de que maneira modificaram o algoritmo e a aplicação originais. As explicações utilizam intensivamente comparações com o modelo. Alguns resultados e grades horárias geradas são mostrados, de acordo com diferentes ajustes dos parâmetros genéticos.
- Capítulo 5: demonstra o que foi atingido com o trabalho de acordo com os objetivos propostos e trabalhos que podem ser realizados no futuro, de forma a melhorar e tornar a aplicação mais específica para a UFLA.

Capítulo 2
Algoritmos Genéticos

[Filho00] descreve como as idéias de evolução natural formaram a base da criação dos Algoritmos Genéticos. Charles Darwin após vários anos de observações e experimentos apresentou em 1858 sua teoria de evolução natural, já o trabalho de Gregor Mendel (desenvolvido em 1865) sobre os princípios básicos da genética, foi estudado por vários cientistas por volta de 1900 e exerceu forte influência sobre os futuros trabalhos relacionados à evolução. A moderna teoria da evolução combina a genética e as idéias de Darwin sobre seleção natural, gerando o princípio básico da Genética Populacional, onde a variabilidade entre os indivíduos em uma população de organismos que se reproduzem sexualmente é produzida pela mutação e recombinação genética. Apenas nos anos 50 e 60, os biólogos começam a desenvolver simulações computacionais de sistemas genéticos. Porém, foi Jonh Holland [Holland75] quem começou, seriamente, a desenvolver as primeiras pesquisas sobre o tema. Após ter melhorado suas idéias, Holland publicou em 1975 o seu livro *Adaptation in Natural and Artificial System*, atualmente considerado como principal referência dos algoritmos genéticos, começando a ser utilizados em várias áreas de estudo.

Os algoritmos genéticos constituem uma técnica bastante utilizada em problemas de otimização, e baseiam a lógica de seu funcionamento nas leis de evolução natural propostas por Charles Darwin, aliadas às idéias sobre genética propostas por Mendel. Estes algoritmos são indicados para se obter soluções de problemas que possuem um espaço de busca muito grande, descobrindo rapidamente sub-regiões de alta qualidade. São versáteis e requerem pouco conhecimento sobre a função a ser otimizada. Para funções desconhecidas, descontínuas e não diferenciáveis, algoritmos genéticos estão entre os mais indicados. A Figura 2.1 mostra um exemplo de AG, extraído de [Filho00].

```
Criar população inicial com M indivíduos
Enquanto critério de parada não for satisfeito
        Avaliar cada indivíduo
        I ← 0
        Enquanto I < M
                Escolher reprodução, cruzamento ou mutação de acordo com
                probabilidades PR, PC, e PM.
                Se clonagem
                        Selecionar indivíduo com base em sua adaptação
                        Copiar indivíduo para nova população
                Se cruzamento
                        Selecionar dois indivíduos com base em sua adaptação
                        Cruzar os indivíduos
                        Inserir os novos indivíduos na nova população
                        I ← I + 1
                Se mutação
                        Selecionar indivíduo com base em sua adaptação
                        Aplicar mutação
                        Inserir indivíduo mutante na nova população
                I ← I + 1
        população ← nova população
Exibir melhor indivíduo
```

Figura 2.1: Exemplo de um algoritmo genético

Os AG são algoritmos de otimização global empregando uma estratégia de busca paralela e estruturada, mas aleatória, voltada em direção da busca de pontos de alta aptidão, ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos). Os AG exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração. O ponto de partida para a utilização de AG, como ferramenta para solução de problemas, é a representação destes problemas de maneira que os AG possam trabalhar adequadamente sobre eles. Tradicionalmente, os indivíduos são representados por vetores binários, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica: o seu genótipo. Os elementos podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo. O princípio básico do funcionamento do AG é que um critério de seleção vai fazer com que,

depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos.

2.1 Termos básicos

Para a adequada compreensão do funcionamento AG, faz-se necessário conhecer sua terminologia biológica básica [Mitchell96]:

- **Cromossomo:** *textitstring* que representa uma determinada característica da solução ou a própria solução;
- **Gene:** característica particular de um cromossomo. O cromossomo é composto por um ou mais genes;
- **Alelo:** valor de determinado gene;
- ***Locus*:** determinada posição do gene no cromossomo;
- **Genótipo:** estrutura que codifica uma solução. Um genótipo pode ser formado por um ou mais cromossomos;
- **Fenótipo:** decodificação ou o significado da estrutura;
- ***Fitness*:** grau de aptidão do indivíduo para determinado ambiente.

2.2 Definição Formal de Algoritmos Genéticos

[Goldberg89] define algoritmo genético como:

> "... um algoritmo de procura baseado nos mecanismos de seleção natural e genética natural. Ele combina a sobrevivência feita por uma função de avaliação entre uma cadeia de caracteres com uma estrutura de informações mudadas aleatoriamente, para formar um algoritmo de procura com algum talento inovador, o mesmo de uma procura de um ser humano. Em toda geração, um novo conjunto de criaturas artificiais (cadeia de caracteres) é criado usando bits e pedaços do teste de avaliação da geração anterior;

ocasionalmente uma parte nova é testada. Enquanto aleatórios, algoritmos genéticos não são nenhum passeio simples sem destino. A procura mais eficiente das informações anteriores para especular os pontos da nova procura resultam em um aumento na sua perfomance."

Nas seções seguintes será realizada a exploração dos conceitos referentes aos algoritmos genéticos, os diversos operadores, os métodos de seleção dos indivíduos e os parâmetros que podem ser configurados.

2.3 Características dos Algoritmos Genéticos

Existem outros mecanismos conhecidos como tradicionais ou exatos, além dos AG. Estes métodos operam com um único ponto e utilizam recursos matemáticos para tentar encontrar uma solução para o problema. Já que são voltados para um modelo matemático tornam-se determinísticos. Como os AG operam sobre uma população de candidatos para a solução do problema e cada indivíduo é avaliado isoladamente, os AG são ditos possuir um paralelismo implícito.

De acordo com [Goldberg89], as principais características que diferenciam os AG de métodos tradicionais são:

- **Parâmetros:** trabalham com a codificação dos parâmetros e não com os próprios;
- **Número de soluções:** trabalham com uma população de indivíduos (conjunto de soluções) e não com um único ponto (uma única solução);
- **Avaliação das soluções:** utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
- **Regras:** utilizam regras probabilísticas e não determinísticas.

E algumas das vantagens de se utilizar algoritmos genéticos são:

- Podem resolver problemas complexos de forma rápida e confiável;
- A construção de algoritmos genéticos e modelos existentes é geralmente simples;
- São extensíveis;
- São fáceis de combinar com outros métodos.

Os AG constituem uma estratégia gerar-e-testar, realizando os testes baseados nos parâmetros da evolução biológica. São capazes de encontrar boas soluções ou até mesmo soluções ótimas, mas este fato não é garantido pelo algoritmo. A área biológica mais proximamente ligada aos Algoritmos Genéticos é a Genética Populacional. Uma desvantagem é a variação dos parâmetros genéticos do algoritmo em cada problema. Assim, para resolução de determinado problema torna-se necessário um estudo particular a respeito do mesmo.

2.4 Funcionamento

Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada: para cada indivíduo é dada uma nota (índice), refletindo sua habilidade de adaptação a determinado ambiente. Uma porcentagem dos mais adaptados é mantida, enquanto os outros são descartados (darwinismo). Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais através de mutações e cruzamento (*crossover*) ou recombinação genética gerando descendentes para a próxima geração. Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada.

Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativa poderosos e robustos. O processo iterativo de um AG e sua estrutura

de funcionamento estão ilustrados na Figura 2.2 [Lucas00]. Várias maneiras para se executar os passos podem ser adotadas e variam de problema para problema.

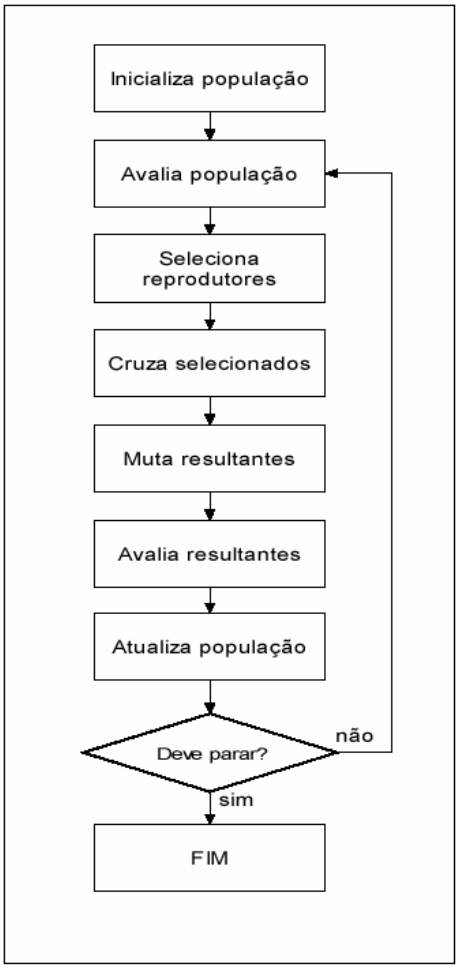

Figura 2.2: Funcionamento de um AG tradicional

2.5 Representação de soluções

Representar de forma eficiente as soluções é o primeiro passo ao se implementar AG. Para [Holland75], um AG trabalha com uma representação binária (0 e 1) para associar uma solução ou componentes de uma solução do problema abordado. Para muitos problemas, tal representação mostrou-se eficiente, mas à medida que foram crescendo as aplicações de AG, esta se foi

tornando menos adequada. Assim foram surgindo formas alternativas, como a representação por um vetor de números inteiros para o caixeiro viajante, onde cada inteiro representa uma cidade e o vetor representa a rota a ser tomada. Mas é importante salientar o efeito de simplificação e diminuição de complexidade obtidas ao se utilizar a representação binária, podendo-se realizar operações booleanas (AND, OR e NOT) em vários métodos.

Características devem ser notadas para se optar por alguma representação, como cita [Koza92], apud [Timóteo02]:

a) completude: determina se é possível representar todos os fenótipos possíveis, verifica se todas as soluções podem ser decodificadas;

b) coerência: indica se a partir do esquema de representação é possível gerar um genótipo que codifique um fenótipo não pertencente ao espectro de soluções do problema. No caso do caixeiro viajante seria a geração de uma rota inexistente;

c) simplicidade: representa o grau de complexidade dos atos de codificação e decodificação das soluções;

d) localidade: pequenas alterações no genótipo acarretam pequenas alterações em seu fenótipo correspondente.

2.6 Inicialização

Sobre uma população inicial de cromossomos serão aplicadas as ações dos passos posteriores do algoritmo. Geralmente, são usadas funções aleatórias para gerar os indivíduos que visa a fornecer maior biodiversidade (indivíduos bons, médios e ruins), fundamental para garantir uma boa abrangência do espaço de busca. Os operadores de inicialização mais comuns são [Goldberg89]:

a) Inicialização heurística: indivíduos são criados a partir de heurísticas. Pode-se gerar indivíduos semelhantes prejudicando a biodiversidade. Por isso, adotam-se métodos Randômicos em conjunto com as heurísticas.

b) Inicialização randômica não uniforme: determinados valores a serem armazenados no gene tendem a ser escolhidos com uma freqüência maior do que o restante;

c) Inicialização randômica com "dope": indivíduos otimizados são inseridos em meio à população aleatoriamente gerada. Esta alternativa apresenta o risco de fazer com que um ou mais superindivíduos tendam a dominar no processo de evolução e causar o problema de convergência prematura[1];

d) Inicialização randômica uniforme: cada gene do indivíduo receberá como valor, um elemento do conjunto de alelos[2] sorteado de forma aleatoriamente uniforme;

2.7 Avaliação

Nesta etapa, o primeiro passo da seleção em si é dado: o universo da população sofre, indivíduo a indivíduo, um processo de avaliação, que visa a representar seu grau de adaptação. Na verdade, este é, em conjunto com a escolha da representação, o ponto do algoritmo mais dependente do problema em si – é necessário aqui que o AG seja capaz de responder sobre quão boa uma resposta é para o problema proposto. Em problemas com muitas restrições, funções baseadas em penalidades são mais comuns.

2.8 Seleção dos Reprodutores

Nesta etapa, os indivíduos são escolhidos para posterior cruzamento. Serão selecionados os indivíduos com maior probabilidade de se reproduzirem. Calcula-se a partir da aplicação da função de avaliação em cada indivíduo,

---

1 - Convergência Prematura: problema recorrente na técnica de algoritmos genéticos: ela ocorre quando a diversidade de uma população cai de tal forma que o processo de reprodução gera a cada geração filhos muito semelhantes aos pais e retarda (ou até mesmo estanca por completo) a evolução.

2 - Alelo: os alelos de um gene correspondem ao conjunto de valores que ele pode assumir. Normalmente, o conjunto de alelos é o mesmo para todos genes de um genoma.

obtendo-se o quão apto ele está para a reprodução em relação à população a que pertence. Este é o passo mais importante de um AG, pois uma má escolha dos reprodutores pode acarretar resultados indesejáveis. Em [Goldberg89], são citados os seguintes métodos de seleção:

- Seleção Determinística [Brindle79];
- Seleção por Roleta Giratória [Holland75];
- Seleção por Torneio [Wetzel83];
- Seleção Uniforme [Backer85].

Um método de seleção alternativo foi proposto no trabalho de [Timóteo02]. Trata-se do método de seleção por Roleta Giratória com Redução, a ser detalhado em seções posteriores.

2.8.1 Seleção Determinística

Indivíduos são selecionados calculando-se a expectativa do número de descendentes. Este cálculo é feito através do grau de aptidão de cada indivíduo:

$$ND(x) = \frac{f(x)}{\sum_{x=1}^{n} f(x)} * n$$

Onde:

ND(*x*): nº de descendentes do indivíduo x.

f(*x*): função que retorna o grau de aptidão do indivíduo x.

n: nº de indivíduos da população.

Na Tabela 2.1 [Timóteo02], segue um exemplo com uma população de quatro indivíduos. Os indivíduos selecionados ficam armazenados temporariamente em uma estrutura de dados. O número de descendentes que cada indivíduo possui está relacionado na Tabela 2.1 [Timóteo02]. Este número é obtido pelo arredondamento de *ND*(*x*). Para o exemplo, uma possível estrutura está ilustrada na Figura 2.3 [Timóteo02].

Tabela 2.1: Tabela de Aptidões dos Indivíduos

Indivíduo	$f(x)$	$ND(x)$	Descendentes
1	169	0.58	1
2	576	1.97	2
3	64	0.22	0
4	361	1.23	1

2	2	1	4

Figura 2.3: Estrutura para armazenar indivíduos selecionados

O indivíduo 2, recebeu duas cópias na estrutura e os indivíduos 1 e 4 receberam uma. O indivíduo 3 por ter um grau de aptidão muito baixo e número de descendentes igual a zero não participará do processo de cruzamento. Os casais são selecionados preferencialmente de forma aleatória. Uma possível escolha poderia ser:

Casal 1: Indivíduo 2 X Indivíduo 4
Casal 2: Indivíduo 1 X Indivíduo 2

Mas há outros métodos para a escolha de casais além do aleatório e um AG pode combinar esses métodos num mesmo algoritmo. O método de escolha aleatória é o mais utilizado e também o mais eficiente. [Goldberg89] cita os seguintes:

- ***In Breeding*** - indivíduos semelhantes são combinados;
- ***Out Breeding*** - indivíduos com diferentes características são combinados;
- ***Self-Fertilization*** - um indivíduo isolado é capaz de se reproduzir;
- ***Clonal Propagation*** - réplicas de determinados indivíduos são inseridas na população de descendentes;
- ***Positive-assortive*** - bons indivíduos são combinados com outros bons indivíduos;
- ***Negative-assortive*** - indivíduos não desejados são combinados.

2.8.2 Seleção por Roleta Giratória

"*Roulette Wheel*" escolhe uma string de acordo com a porção ocupada por esta em uma roleta, isto é, de acordo com o grau de aptidão relativo a esta roleta (em porcentagem). Quanto maior o grau de aptidão, maior a porção na roleta e conseqüentemente maiores as chances de seleção. A Figura 2.4 é apenas uma ilustração do processo e que não corresponde ao exemplo seguinte.

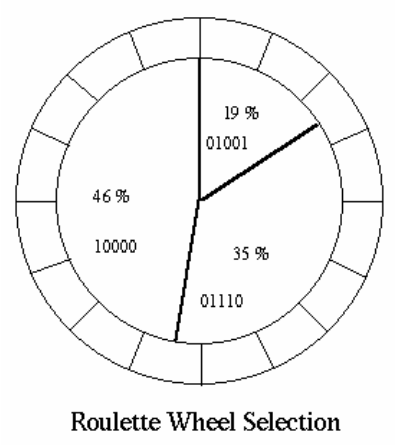

Figura 2.4: Ilustração da roleta

Este algoritmo pode ser descrito da sequinte maneira:

1. Calcule a soma do grau de aptidão de todos os indivíduos.;
2. Sorteie um número s entre 1 e a soma das aptidões;
3. Selecione o Indivíduo x tal que $s \in [(\sum_{x=1}^{n-1} f(x)) + 1, \sum_{x=1}^{n} f(x)]$.

Intervalo de Seleção para Roleta Giratória

Indivíduo	$f(x)$	Intervalo de Seleção
1	169	1 - 169
2	576	170 - 745
3	64	746 - 809
4	361	810 - 1171

O intervalo de seleção é dado através dos somatórios. O primeiro valor é obtido por $(\sum_{x=1}^{n-1} f(x)) + 1$ e o segundo por $\sum_{x=1}^{n} f(x)$, sendo que *f(x)* é a função que avalia o grau de aptidão do indivíduo *x*. É necessário realizar o sorteio de um número a cada giro. Suponha que esse número fosse 712. Nesse caso, o indivíduo selecionado seria o indivíduo 2, pois 712 é um valor que se encontra entre o intervalo de seleção desse indivíduo. Como a população possui quatro indivíduos, seria necessária a realização de quatro sorteios e através destes seriam formados os casais para a reprodução.

A desvantagem deste método é o surgimento do superindivíduo, que possui um intervalo de seleção bem maior que os outros. Neste caso, como as chances desse indivíduo ser selecionado são bem grandes, a tendência é que este indivíduo tenha muitos descendentes nas próximas gerações, o que poderia levar o algoritmo para uma convergência prematura. Para resolver este problema, [Goldberg89] propôs um método para escala do grau de aptidão, chamado de *linear scalin*g. O objetivo desse método é reduzir a probabilidade de superindivíduos e fazer com que as chances dos indivíduos medianos da população serem sorteados aumentem. Para realizar esta tarefa, o método faz com que o valor retornado pela função objetivo sofra uma alteração. Essa alteração é provocada por uma função f_a que recebe como parâmetro o valor retornado pela função de avaliação *f(x)*. Segundo [Goldberg89] essa função seria

$$fa(f(x)) = a * f(x) + b$$

onde:

$$a = \frac{(C_{mult} - 1.0) * U_{avg}}{U_{max} - U_{avg}}$$

$$b = \frac{U_{avg} * (U_{max} - C_{mult} * U_{avg})}{U_{max} - U_{avg}}$$

Sendo que:

U_{avg} - média do grau de aptidão dos indivíduos

U_{max} - maior grau de aptidão entre os indivíduos

C_{mult} - *fitness* multiple

Após a realização de vários experimentos, [Goldberg89] recomenda que o valor da constante C_{mult} varie de 1.2 a 2.0 para os problemas onde o número de indivíduos presentes na população varia de 50 a 100. Um outro detalhe é que a função $f_a(x)$ não pode receber como parâmetro valores negativos, pois isso geraria resultados incoerentes.

2.8.3 Seleção por Torneio

Proposto por [Wetzel83], este método funciona similarmente a um torneio tradicional. Ocorre um sorteio de dois indivíduos de uma população e vence o torneio aquele que tiver o maior grau de aptidão. Assim, todos os indivíduos que venceram seus torneios serão selecionados pelo método.

Tabela 2.2: Grau de aptidão para o método de seleção por torneio

Indivíduo	$f(x)$
1	169
2	576
3	64
4	361

Baseado na Tabela 2.2 de [Timóteo02] e nos exemplos descritos pelo autor, suponha que os indivíduos sorteados foram 1 e 2. Quem vence o torneio é o indivíduo 2, pois seu grau de aptidão é o maior. Assim, este indivíduo é selecionado para cruzamento. Esse método possui a vantagem de não gerar super-indivíduos. A chance do indivíduo com maior grau de aptidão ser selecionado é a mesma, independentemente do quão grande é seu grau de aptidão. A chance do indivíduo 2 ser selecionado é 1/4, pois se for sorteado, independentemente de quem seja seu rival, ele vencerá o torneio. Se o grau de aptidão fosse 3000 ao invés de 576, as chances de seleção continuariam as

mesmas neste método. Já no método da roleta, o intervalo de seleção iria aumentar muito e por isso a chance do indivíduo ser selecionado também iria ser bem maior.

2.8.4 Seleção Uniforme

Reduz a seleção a uma escolha aleatória. Cada indivíduo tem a mesma chance de ser selecionado, independentemente do grau de aptidão. Se utilizado sozinho este método reduz o AG a uma busca cega. Pode complementar outros métodos de seleção. Um AG pode utilizar o método de seleção por roleta e em alguns momentos utilizar a seleção uniforme para diversificar a população, ou seja, selecionar também indivíduos ruins que seriam dificilmente selecionados pela roleta.

2.8.5 Seleção por Roleta com Redução

Este método foi proposto e utilizado no trabalho de [Timóteo02], e funciona de maneira semelhante ao método da roleta giratória. A única diferença é que toda vez que um indivíduo for selecionado, o seu grau de aptidão diminui em $x\%$, fazendo com que seu intervalo de seleção também diminua.

Tabela 2.3: Intervalo de Seleção para Roleta com Redução

Indivíduo	$f(x)$	Intervalo de Seleção
1	169	1 - 169
2	576	170 - 745
3	64	746 - 809
4	361	810 - 1171

O exemplo: sorteia-se um número entre 1 e a soma das aptidões dos indivíduos. Suponha que o número sorteado fosse o 1090. Neste caso, o indivíduo 4 será selecionado devido ao seu intervalo de seleção. A diferença desse método para o da roleta giratória é que no próximo sorteio o indivíduo 4

terá seu intervalo de seleção reduzido. Se o fator de redução utilizado fosse 2%, o grau de aptidão e o intervalo de seleção do indivíduo 4 teria que ser reajustado de acordo com este percentual.

Tabela 2.4: Reajuste do intervalo de seleção e grau de aptidão do indivíduo selecionado

Indivíduo	$f(x)$	Intervalo de Seleção
1	169	1 - 169
2	576	170 - 745
3	64	746 - 809
4	354^*	$810 - 1163^*$

É importante observar que tanto o grau de aptidão quanto o intervalo de seleção do indivíduo 4 diminuíram. Com isto, as chances desse indivíduo ser selecionado diminuem e talvez as chances de convergência prematura também, porém esse fato não é garantido.

2.9 Operadores Genéticos

O operador genético representa o núcleo de um AG. O objetivo básico de um operador genético é produzir novos cromossomos que possuam propriedades genéticas superiores às encontradas nos pais. Em AG convencionais, os operadores genéticos clássicos são: a mutação e o *crossover*. Seu princípio básico é transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores.

2.9.1 *Crossover*

Estando definido o grupo dos genes selecionados, o próximo operador do arsenal dos algoritmos genéticos é o *crossover*. Em termos biológicos, significa misturar os cromossomos dos indivíduos progenitores para gerar um

novo cromossomo do indivíduo filho. A mesma idéia aplica-se ao *crossover* dos AG. Entre os vários operadores de *crossover*, [Filho00] cita:

- *Crossover* com corte em 1 ponto (1PX);
- *Crossover* com corte em 2 pontos (2PX);
- *Crossover* com múltiplos pontos (MPX);
- *Crossover* segmentado (SX);
- *Crossover* uniforme.

2.9.1.1 *Crossover* com corte em um ponto (1PX)

Um ponto de cruzamento é escolhido e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores à este ponto no outro pai. Na Figura 2.5 [Timóteo02], cada bloco representa um gene e a linha tracejada representa o ponto de corte. Os indivíduos descendentes (filhos) serão a combinação dos seus ancestrais.

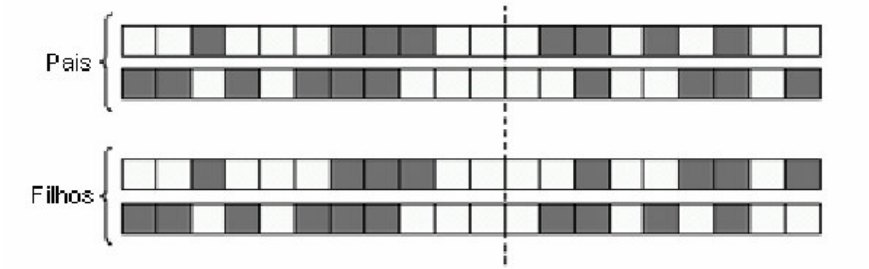

Figura 2.5: Troca genética entre os indivíduos

2.9.1.2 *Crossover* com corte em dois pontos (2PX)

Este método utiliza dois pontos de corte ao invés de um e funciona de maneira equivalente ao método anterior. Na Figura 2.6 [Timóteo02], os indivíduos descendentes (filhos) serão criados da combinação genética dos ancestrais (pais) na região situada entre os pontos de corte.

2.9.1.3 *Crossover* com múltiplos cortes (MPX)

É uma generalização desta idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados. O sorteio de um número fixo (n) de pontos de corte é efetuado. Ao invés de um ou dois pontos, serão vários pontos de corte. Freqüentemente, este operador não funciona de maneira eficiente, como frisa [Goldberg89], os operadores de *crossover* 1PX são os que apresentam melhores resultados.

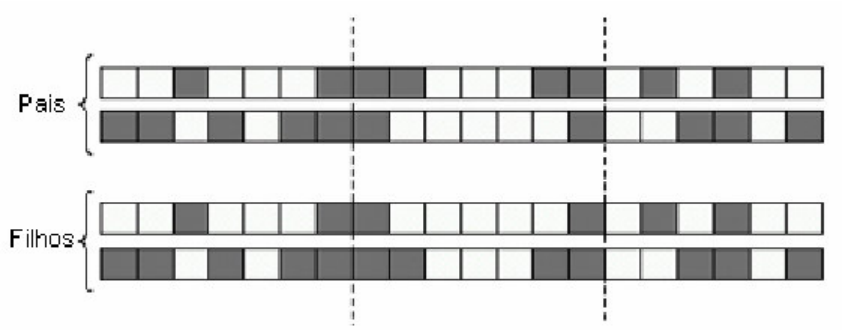

Figura 2.6: Troca genética entre os dois pontos de corte do *Crossover 2PX*

2.9.1.4 *Crossover* segmentado (SX)

É uma variação do operador de *crossover* MPX. Ao invés de ser sorteado um número fixo de pontos, serão sorteados números variáveis de pontos. [Goldberg89] afirma que este método e o MPX são de implementação mais complexa e não apresentam bons resultados.

2.9.2 Mutação

Os operadores de mutação são necessários para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, fornecendo assim, meios para a introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca

será zero. A mutação opera sobre os indivíduos resultantes do processo de cruzamento com uma probabilidade pré-determinada efetuando algum tipo de alteração em sua estrutura. Geralmente a probabilidade é bem pequena para não fazer do processo uma busca cega. Os principais operadores de mutação são descritos a seguir [Goldberg89]:

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

2.10 Parâmetros Genéticos

É importante também, analisar de que maneira alguns parâmetros influem no comportamento dos Algoritmos Genéticos, para que se possa estabelecê-los conforme as necessidades do problema e dos recursos disponíveis.

a) **Tamanho da População:** o tamanho da população afeta o desempenho global e a eficiência dos AG. Uma população muito pequena oferece uma pequena cobertura do espaço de busca, causando uma queda no desempenho. Uma população suficientemente grande fornece uma melhor cobertura do domínio do problema e previne a convergência prematura para soluções locais.
b) **Taxa de Cruzamento**: quanto maior for esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Entretanto, isto pode gerar um efeito indesejado, pois a maior parte da população será substituída, ocorrendo assim perda de variedade genética.
c) **Taxa de Mutação**: mesmo uma baixa taxa de mutação previne que uma dada posição fique estagnada em um valor, mas uma taxa de mutação muito alta pode tornar a busca essencialmente aleatória.

d) **Intervalo de Geração**: controla a porcentagem da população que será substituída durante a próxima geração.

2.11 Finalização

A finalização envolve um teste que dá fim ao processo de evolução caso o AG tenha chegado a algum ponto pré-estabelecido de parada. Os critérios para a parada podem ser vários, entre os quais:

- Tempo;
- Número de Gerações;
- Convergência.

Adotar o critério de convergência pode ser arriscado em alguns casos, [Filho00] fala em 95% dos genes iguais entre os indivíduos. Podem ocorrer situações em que os indivíduos demorem um tempo relativamente grande para convergirem. O tempo pode ser abstrato, pois no mesmo intervalo de tempo máquinas diferentes podem realizar diferentes processamentos. O número de gerações, na maioria dos casos é a melhor escolha para terminar o processamento de um AG, pois independe da máquina e torna o critério de término mais realístico.

2.12 Exemplo de AG

Após vistos todos os componentes de um AG, é sugerido um exemplo de código resumindo-se os passos já conhecidos. O primeiro passo é criar uma população inicial e logo em seguida calcular o grau de aptidão dos indivíduos gerados. Após isso, se inicia o processo iterativo até a condição de término ser atingida. Após a seleção dos casais, é necessário realizar um *crossover* (cruzamento) com probabilidade *Pc* (segundo [Mitchell96], $Pc = 0.7$ na maioria dos casos). O resultado do *crossover* são dois novos descendentes resultantes da troca genética entre os ancestrais.

```
Início
   Crie uma população inicial
   Calcule o grau de aptidão dos indivíduos
   Enquanto não atingiu a condição de término
      Selecione os indivíduos e forme os casais
      Realize crossover com probabilidade Pc
   Para cada gene dos descendentes
         Realize uma mutação com probabilidade Pm
      Se utiliza critério de sobrevivência
         Verifique os sobreviventes
      Calcule grau de aptidão dos sobreviventes
   Fim do Enquanto
Fim
```

A mutação deve ser aplicada com probabilidade *Pm* em cada gene dos descendentes. Ainda segundo [Mitchell96], geralmente *Pm* = 0.001. Se o algoritmo levar em conta critérios de sobrevivência, é necessário avaliar e decidir quem formará a população para a próxima geração: descendentes ou ancestrais. Se o critério de sobrevivência não for utilizado, os descendentes formarão a nova população independente de qualquer fator.

2.13 Aplicações

As utilizações dos AG são muitas, todas relacionadas de uma forma ou de outra a uma análise multidimensional, onde se busca conseguir uma solução global. Geralmente o material genético é utilizado para codificar os valores dos vários parâmetros que definem o espaço de resultados admissíveis, e procura-se encontrar o valor destes parâmetros (valor dos cromossomos) que soluciona um certo problema de otimização. Os Algoritmos Genéticos, desde os conceitos básicos realizados por Holland, vêm sendo utilizados em várias áreas de pesquisa e em situações do mundo real com bons resultados.

- *Multiprocessor Scheduling*: aplicação de AG no problema de associação ótima de processos e processadores
- Biologia Molecular e Físicoquímica
- Engenharia em Construções: otimização discreta de estruturas

- Busca em Base de Dados
- Geofísica: *Seismic Waveform Inversion*
- Redes Neurais
- Compressão de Dados: compressão de dados em geral, e a compressão de imagens sólidas em particular
- Melhorias nas Telecomunicações: cita-se o caso da *US West*, uma companhia regional de telecomunicações do estado do Colorado, que vem usando um sistema baseado em AG para projetar, em duas horas, redes óticas especializadas, trabalho que levaria seis meses utilizando especialistas humanos
- Otimização de plantão médico-hospitalar: no Hospital universitário da UFSC, em Florianópolis, os Algoritmos Genéticos foram utilizados para auxiliar na elaboração de uma escala de trabalho dos médicos plantonistas neonatalogistas da maternidade
- Aprendizagem de máquina: pesos para redes neurais e sensores para robôs
- Ecologia: modelar fenômenos ecológicos como competição entre nichos ecológicos, co-evolução parasita-hospedeiro, simbiose e fluxo de recursos
- Genética: utilizados para estudar questões da genética de populações, tais como a viabilidade das condições de um gene em termos evolucionários
- Sistemas sociais: estudar aspectos evolucionários de sistemas sociais, tais como a evolução do comportamento social em colônias de insetos, e mais genericamente, a evolução da cooperação e comunicação em sistemas multi-agentes.

Capítulo 3
Problema de geração de horários

O problema da geração de horários escolares, conhecido na literatura como *Timetablin*g, consiste em arranjar encontros entre professores e alunos em um período de tempo previamente fixado (tipicamente uma semana) de modo a satisfazer um conjunto de restrições. É um problema clássico. A solução manual do problema além de ser demorada, pode ocasionar grades horárias não ideais, como por exemplo, uma grade onde um professor lecione duas aulas ao mesmo tempo. Vários métodos de otimização foram tentados, mas a maioria encontra problemas ao lidar com o grande número de restrições e a alta variedade de seu grau de importância [Filho00]. Estas características, aliadas ao conjunto de restrições impostas, atingem tal complexidade que o estabelecimento do horário de aulas torna-se uma operação de difícil solução, mesmo com o auxílio de computador. O modelo que se desenvolve, além de reduzir o tempo requerido para a confecção do horário, pode ser empregado como instrumento de decisão para uma melhor utilização dos recursos físicos e humanos da instituição.

Horários são uma necessidade de toda instituição de ensino, incluindo escolas primárias, secundárias e universidades. São vários os problemas encontrados na elaboração de grades horárias, tais como alocação das aulas para os professores, alocação de salas, laboratórios, etc. Além disso, é necessário considerar as combinações de algumas destas restrições. Na elaboração de grades horárias para uma instituição de ensino superior, é necessário atender todos os cursos, e muitas vezes vários *campi*, o que não é o caso da UFLA. Este é um fator a mais na complexidade da construção das grades horárias, já que muitas vezes é necessário distribuir a carga horária dos professores de um curso entre os diferentes *campi* da instituição. Como explica [Júnior00], as aulas de uma instituição são o conjunto de reuniões entre professores e alunos num conjunto de períodos de tempo. Nestes períodos de tempo, os recursos disponíveis são limitados, e várias outras restrições podem ser adicionadas.

Portanto, o problema de construção de grades horárias pode ser visto como relações entre conjuntos de professores, disciplinas, alunos, períodos de tempo e recursos. Estas relações estão sujeitas ainda a uma série de restrições, cujas combinações aumentam a complexidade do problema. Algoritmos Genéticos são uma técnica de busca heurística que faz uma analogia com as teorias de evolução natural e genética.

Esta técnica foi utilizada em diversos trabalhos e o enfoque do presente trabalho se baseia na implementação feita por [Timóteo02], para encontrar uma solução viável para o problema da geração de grades horárias, dentro de uma série de combinações e restrições existentes. O processo de busca de uma solução é feito mediante operadores genéticos que fazem o papel da evolução natural, seguindo limites definidos (os parâmetros genéticos). As restrições do problema são transformadas em funções que avaliam cada geração de indivíduos, possibilitando ou não a evolução dos mesmos.

3.1 Abordagem AG

Os objetos específicos ao utilizar AG consistem em:

- Representação do problema;
- Definir os operadores genéticos.
- Definir as formas de avaliação do cromossomo;
- Especificar e testar os parâmetros genéticos;
- Validar o modelo proposto.

3.2 Conceitos

A maioria dos estabelecimentos de ensino superior apresenta termos padronizados. Turma: são os alunos que cursam juntos uma mesma disciplina em um mesmo *campus*, curso e semestre curricular. Com isso, cada aluno tem,

possivelmente, aulas com diferentes colegas em cada disciplina em que estiver inscrito naquele semestre letivo.

3.3 Formulações do Problema

O problema de construção de um horário acadêmico é conhecido na literatura de forma geral como *timetabling*. Uma grande variedade de problemas de *timetabling* têm sido descrita e muitos caminhos diferentes para a sua solução têm sido propostos. Estes problemas dependem do tipo de escola e de seu sistema educacional. Desta forma, não existe um modelo universal que pode ser aplicado sempre. [Schaerf99] cita três classes para esse problema:

- ***School Timetabling***: seqüenciamento semanal das aulas de uma escola, evitando que professores e alunos tenham mais de uma aula ao mesmo tempo [Timóteo02];
- ***Course Timetabling***: seqüenciamento semestral das aulas de um conjunto de cursos de uma universidade, evitando a simultaneidade de cursos com estudantes em comum [Júnior00]. O trabalho realizado constitui um *Course Timetabling*;
- ***Examination Timetabling***: seqüenciamento de exames de um conjunto de cursos em uma universidade, evitando exames simultâneos de cursos com estudantes em comum, e espalhando os exames o máximo possível [Burke94].

Quanto aos caminhos propostos para a solução, cada um deles é desenvolvido para atender certos aspectos de um problema de *timetabling*. Se o tamanho do problema for pequeno, um modelo baseado em programação linear inteira pode ser utilizado. Conforme cresce a dimensão do problema, é freqüente o uso de técnicas de decomposição. Outras formulações têm obtido sucesso considerável: algoritmos genéticos ou evolucionários; métodos heurísticos de busca, como por exemplo, a bem conhecida pesquisa tabu; a programação de lógica de restrições; entre outros.

3.4 Particularidades e restrições

O enfoque principal do trabalho de [Timóteo02] foi a variante citada como *School Timetablin*g, já que ela se adapta de uma forma coerente à realidade da maioria das escolas de ensino fundamental e médio. Tipicamente, essas escolas possuem um número de turmas e alunos determinado pelas suas limitações físicas. Em geral, existem mais turmas do que salas, pois trabalham em períodos matutinos, vespertinos e às vezes em períodos noturnos. Cada turma possui uma lista de disciplinas obrigatórias que variam de acordo com o currículo escolar. Geralmente, as aulas de determinada turma são em apenas um período (matutino, vespertino ou noturno). As aulas são ministradas por um conjunto de professores que trabalham na escola e até em outras também. Cada professor tem seu próprio número de aulas.

Existem várias restrições que devem ser consideradas na tentativa de encontrar uma solução para o problema. Elas variam de acordo com necessidades específicas das escolas. Cada restrição tem uma determinada importância, que deve ser quantificada pela instituição de ensino. Para que o algoritmo seja mais dinâmico é necessário permitir a alteração dos pesos de cada critério.

A UFLA oferece sete cursos de graduação: Administração, Agronomia, Ciência da Computação, Engenharia Agrícola, Engenharia Florestal, Medicina Veterinária e Zootecnia. Foi levantado, junto à Pró-Reitoria de Graduação da UFLA (PRG), um conjunto de particularidades da instituição, sendo que algumas são utilizadas no atual sistema de geração manual de horários, outras não. Algumas destas características são:

- Há apenas um professor responsável por disciplina, sendo que, muitas vezes, há mais de um professor que a leciona, ficando a cargo do colegiado[3] de cada curso resolver este problema;

3 - COLEGIADO DE CURSO: Cada curso de graduação possui um Colegiado responsável pela coordenação, planejamento, controle e avaliação das atividades de ensino.

- O atual sistema acusa choques de locais, horários, o que deve ser analisado e corrigido manualmente (tais choques ocorrem pela limitação de apenas um professor ser responsável por disciplina e porque o sistema é manual);
- Muitos professores possuem restrições de horários, não estando disponíveis para ministrar aulas em qualquer momento;
- Cada disciplina necessita de um conjunto de recursos para ser ministrada. A grande maioria necessita apenas de uma sala de aula com capacidade que suporte as vagas oferecidas. Outras disciplinas requerem recursos adicionais, tais como laboratórios de informática, salas de desenho, laboratórios, etc. As disciplinas dos cursos são lecionadas pelos professores do quadro da instituição;
- A cada semestre, cada período possui aulas nos horários matutino ou vespertino, porém em alguns cursos, há aulas adicionais nos dois horários, e também há os casos onde alunos pedem adiantamento de disciplina e possuem aulas nos dois períodos, mas isto é tratado pela DRCA;
- Pode-se compor dois turnos distintos com os horários destes períodos, a fim de atender os horários das turmas. Também se pode agrupar horários de turnos;
- As aulas são ministradas de segunda a sexta-feira nos dois períodos (matutino e vespertino). As aulas são compostas de tempos de 50 minutos;
- Outras particularidades da UFLA não serão tratadas, como aquelas referentes à matrícula, que também ficam sob a responsabilidade da Diretoria de Registro e Controle Acadêmico (DRCA).

No processo de ensino de uma disciplina, são utilizados diferentes bens físicos da IES, denominados de recursos. Estes recursos só podem ser alocados para uma turma em um determinado horário, sendo que uma turma pode ter de um ou mais recursos diferentes alocados durante um semestre.

3.5 Gerador de grade horária

Define-se um gerador de grade horária como um procedimento sistemático e automático que procura compatibilizar em uma tabela de horários disponíveis na semana, todas as disciplinas que são oferecidas em um semestre de um curso, satisfazendo as disponibilidades de recursos. É importante ressaltar que a dimensão do problema e as características específicas da instituição constituem um aspecto extremamente importante no que diz respeito à elaboração de um processo de geração de horário. É possível especificar a disciplina e o horário como parte do problema. A solução deverá encontrar um professor capacitado a ministrá-la, com base nas informações sobre os professores disponíveis e seus horários. Segundo [Burke95], as restrições de horários são muitas e variadas, sendo que alguns dos tipos mais comuns são relacionadas a seguir:

- **Alocação de Recursos:** Um recurso deve ser associado para um outro recurso de tipo diferente ou para uma turma. Por exemplo um professor pode preferir lecionar algumas aulas de uma disciplina numa sala normal, e outras num laboratório.
- **Tempo alocado**: Uma aula ou um recurso pode ser associado a um professor, implicando que neste período o professor está indisponível para outras aulas. É possível também associar previamente um horário a uma reunião em particular.
- **Restrições de tempo entre reuniões**: Um exemplo desta classe de restrições é uma aula em particular ter que acontecer antes de outra. Um caso comum são as aulas teóricas que devem anteceder as aulas práticas em laboratório.
- **Distribuição de aulas**: Aulas devem ser distribuídas uniformemente durante o período. Por exemplo, aulas teóricas devem ser intercaladas com aula práticas, mas isto não ocorre no caso da UFLA.

- **Coerência das aulas:** Esta restrição foi projetada para produzir horários mais organizados e convenientes, vindo em oposição à restrição de distribuição de aulas. Por exemplo, uma disciplina que poderia ter suas aulas todas num único dia, deve ser alocada em dois dias, para não sobrecarregar os alunos.
- **Capacidades das salas**: O número de estudantes em uma sala não pode exceder a sua capacidade.
- **Continuidade**: Qualquer restrição cujo propósito principal é assegurar certas características dos horários dos estudantes, sendo ela constante ou previsível. Por exemplo, aulas para um mesmo curso devem ser alocadas em uma mesma sala ou em um mesmo período.

Ainda em relação às restrições, [Burke95] divide-as em duas categorias: Rígidas *(hard*) e Flexíveis *(soft*):

- **Restrições Rígidas:** Um horário que quebra uma restrição rígida não pode ser considerado parte da solução, e deve ser reparado ou rejeitado pelo algoritmo do horário.
- **Restrições Flexíveis:** Restrições flexíveis nem sempre são menos importantes que restrições rígidas, e dificilmente levam um horário a ser rejeitado. São aplicadas a qualquer método de horário, geralmente avaliadas por uma função que penaliza o horário, calculando até que ponto este quebrou sua restrição.

Uma vantagem da utilização de um algoritmo genético é que esses critérios serão definidos na função de avaliação. Desta forma, se houver qualquer modificação em relação aos critérios, somente a função de avaliação será modificada e não o algoritmo inteiro.

3.6 Otimização de Planejamento de Horários por AG

Segundo [Velloso95], o planejamento e a alocação de recursos são problemas de solução difícil com um histórico longo e variado nas áreas da

Pesquisa Operacional e Inteligência Artificial. Tais problemas precisam, em geral, ser atacados através de uma combinação de técnicas de busca e heurísticas, o que torna as soluções específicas para os problemas em questão. O planejamento é difícil por duas razões: primeiro, é um problema computacionalmente complexo, descrito como NP-*hard*; segundo, problemas de planejamento são freqüentemente complicados pelos detalhes e restrições associados às tarefas e recursos.

[Timóteo02] produziu um algoritmo genético para tratar especificamente a variante citada como *School Timetabling* (contendo algumas características particulares). Os passos para se gerar o algoritmo desenvolvido serão descritos.

3.6.1 Representação da solução

Foi escolhida uma forma intuitiva, onde cada gene representa um *slot* (ex: segunda-feira às 7:00). Um cromossomo seria então uma matriz de 3 dimensões (Número de turmas X [Número de Dias X Número de Horários]). Essa representação apresenta a vantagem de não permitir, graças à sua codificação, que duas disciplinas referentes à mesma turma ocupem o mesmo horário. Porém, não garante por si só a não ocorrência de colisões de professores e outras restrições. Cada indivíduo é formado por uma matriz de três dimensões (Figura 3.1 [Timóteo02]). Dependendo do número de turmas e *slots* disponíveis, a representação de uma solução pode ser muito grande, requisitando assim maiores recursos computacionais.

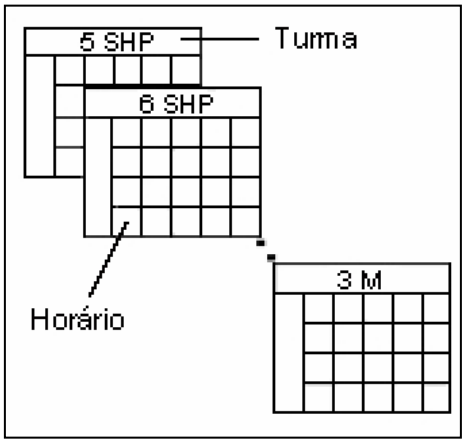

Figura 3.1: Representação da solução para o *Timetabling*

Como uma população ideal é de 50 a 100 indivíduos segundo [Goldberg89], pode-se afirmar que esta representação de soluções para o *Timetabling* é grande. Entretanto, existem outras maneiras de representar a solução: cada dia é descrito como uma lista encadeada (representando o cromossomo) e o genótipo é o conjunto das listas de todos os dias. Há a representação por matriz, porém cada turma é representada por um cromossomo. O conjunto da representação de todas as turmas formam o genótipo.

3.6.2 Criação da População Inicial

[Timóteo02] desenvolveu dois operadores com este propósito:

a) Geração aleatória: este operador percorre uma lista disciplinas de determinada turma e para cada uma das disciplinas, sorteia um *slot* disponível na grade horária.

b) Geração Heurística: funciona de maneira semelhante ao operador anterior. A única diferença é que ao invés de sortear uma disciplina isolada, serão sorteados blocos de duas disciplinas. A lista de disciplinas de uma determinada turma é dividida em duas outras listas: *LBlocos* e *LIsolada*s. A lista *LBlocos* contém os blocos disponíveis para sorteio e a

lista *LIsoladas* contém as disciplinas que não puderam formar blocos e desta forma precisam ser sorteadas isoladamente.

A população gerada pelo método heurístico geralmente é mais evoluída que a gerada pelo método aleatório, pois preserva blocos ideais de disciplinas.

3.6.3 Avaliação dos Indivíduos

Os critérios de avaliação utilizados foram:

- Colisão por professor;
- Horários esparsos;
- Blocos de disciplinas;
- Vários blocos por dia;
- Média de aulas por dia;
- Preferências dos professores.

Uma função de avaliação faz o uso do cálculo de penalidades associadas a um indivíduo. Cada vez que a função de avaliação encontra uma colisão por professor, um horário esparso ou qualquer outra restrição, uma penalidade é atribuída e seu valor varia de acordo com a sua importância. A função de avaliação retorna um valor numérico de acordo com a fórmula a seguir:

$$f(x) = \frac{1}{1 + penalidades}$$

Através dessa fórmula é possível perceber que o valor de f(x) varia de 0 a 1. O indivíduo ótimo é aquele que não possui nenhuma penalidade (*f*(*x*) se torna igual a 1).

3.6.4 Seleção dos Indivíduos

Os métodos de seleção implementados foram o da roleta giratória, método de seleção por torneio e por fim o método da roleta com redução. Antes da seleção, os indivíduos são avaliados separadamente e o resultado é a

quantificação do seu grau de aptidão. Um método de seleção é uma função que recebe como parâmetro o grau de aptidão de cada indivíduo e retorna apenas os indivíduos selecionados. O casamento ocorre entre os indivíduos selecionados pelo método de seleção.

3.6.5 *Crossover*

O *crossover* conforme descrito é o cruzamento genético entre os indivíduos selecionados. Em geral, os operadores de *crossover* buscam trocar determinados genes e a partir daí gerar novos indivíduos. Os operadores de *crossover* implementados por [Timóteo02] foram:

1. *Crossover* com corte em um ponto;
2. *Crossover* com corte em dois pontos;
3. *Crossover* heurístico com corte em um ponto.

Os operadores de *crossover* com corte em um ponto e o de *crossover* com corte em dois pontos foram implementados por serem os mais tradicionais entre os operadores. O operador de *crossover* heurístico foi uma tentativa de realizar a troca genética com alguma heurística envolvida. No caso, a heurística utilizada foi a preservação de blocos de duas disciplinas.

3.6.5.1 *Crossover* com corte em um ponto (1PX)

Existem alguns cuidados que devem ser tomados para que os novos indivíduos gerados não sejam incoerentes. Para resolver este problema, basta que haja troca apenas entre genes compatíveis, ou seja, somente genes que possuam como valor a mesma disciplina.

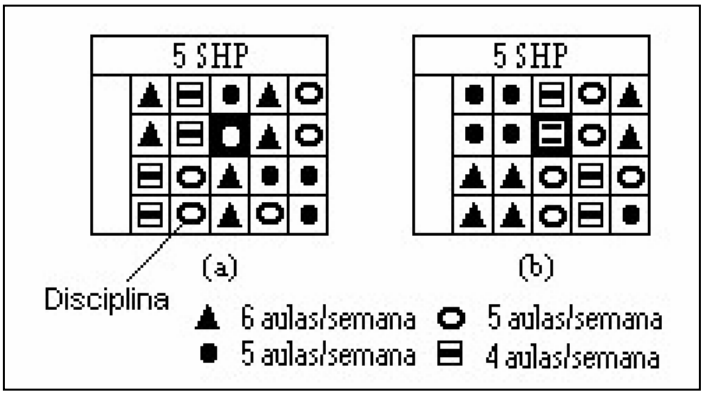

Figura 3.2: Indivíduos representando a grade horária

Como primeiro passo da realização do *crossover*, criou-se duas listas com os genes dos indivíduos a partir do ponto de corte. A Figura 3.3 apresenta dois indivíduos (a) e (b) e duas listas representadas por (c) e (d). A lista (c) representa os genes do indivíduo (a) que possivelmente irão realizar *crossover* e a lista (d) representa os genes do indivíduo (b). Cada nó dessas listas são compostos pelos campos:

- ***Disc*:** representando a disciplina;
- ***Pgene*:** posição do gene no cromossomo;
- ***Pcross*:** indica a posição de *crossover*;
- ***Disponível*:** indica se o gene está disponível para *crossover*.

Inicialmente as listas são preenchidas com a disciplina e a posição absoluta dessa disciplina no cromossomo (*slot* referente na grade horária), setando assim o campo *Disc* e *Pgene* respectivamente. O campo relativo á posição de *crossover* é iniciado como -1 e o campo *Disponivel* configurado para VERDADEIRO. Neste pseudocódigo as listas (c) e (d) serão referenciadas como ListaC e ListaD respectivamente.

```
Para cada gene i da ListaC faça
      posgene:= ProcuraGene(ListaC[i].Disc,ListaD)
      Se posgene <> -1 então
      ListaC[i].Pcross:= ListaD[posgene].Pgene
      ListaD[posgene].Pcross:= ListaC[i].Pgene
```

```
        ListaC[i].Disponivel:= FALSO
        ListaD[posgene].Disponivel:= FALSO
        Fim do Se
Fim do Para
```

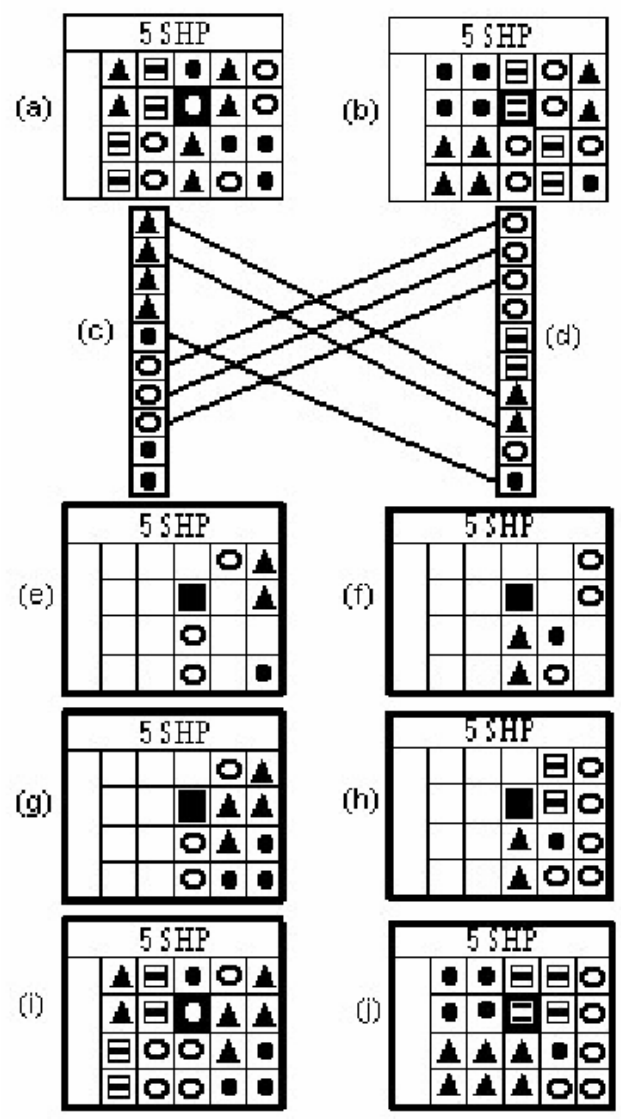

Figura 3.3: *Crossover*

A função *ProcuraGene* descrita no pseudocódigo, recebe como parâmetro uma disciplina e uma lista, retornando a posição do gene que tem

como valor essa disciplina. Um detalhe é que essa função retorna -1 quando não encontra um gene compatível na lista. Também é importante ressaltar que a função só retorna o gene compatível se ele estiver disponível *(Disponivel* = VERDADEIRO), pois se não preocupasse com isso, dois ou mais genes poderiam ser relacionados a uma mesma posição de *crossover*. Ainda é necessário realizar a troca dos genes compatíveis nos indivíduos Neste caso, é necessário percorrer a lista de genes e para cada gene com posição de *crossover* diferente de -1 realizar a troca. Na Figura 3.3, os indivíduos (e) e (f) são os resultantes da troca de genes compatíveis. Houve uma troca na posição das disciplinas em relação aos seus ancestrais. A troca é possível justamente pelo campo *Pcross* das listas. Esse campo guarda a posição de um gene compatível no cromossomo do outro indivíduo participante do *crossover*, fazendo com que haja troca da posição onde as disciplinas estão alocadas. Os genes não compatíveis são aqueles sem relação nas listas de genes. A posição de *crossover* desses é configurada para -1. Como não há possibilidade de troca, eles são alocados de forma aleatória nos slots vagos restantes (a partir do ponto de corte) da grade horária. Na Figura 3.3, os indivíduos indicados como (g) e (h) mostram uma maneira pela qual os genes não compatíveis foram alocados na grade horária. A partir do ponto de corte os genes já foram trocados, restando os genes anteriores a esse ponto, copiando-os. O indivíduo indicado na Figura 3.3(i) recebeu os genes que antecedem o ponto de corte relativo ao indivíduo (a) da mesma figura e o indivíduo indicado como (j) recebeu os genes que antecedem o ponto de corte relativos ao indivíduo (b).

Após essa operação dois novos descendentes foram criados: (i) e (j). Os novos indivíduos, além de diferentes dos seus ancestrais, são coerentes em relação as cargas horárias das disciplinas.

Foram também efetuados por [Timóteo02] o *Crossover* com corte em dois pontos (2PX) e *Crossover* heurístico com corte em um ponto.

3.6.6 Mutação heurística

Mutação é útil quando o critério de alocação de disciplinas em bloco é importante. Uma proposta para um operador de mutação foi elaborada. A idéia básica é ter o conjunto de disciplinas pertencentes a uma determinada turma como um conjunto de blocos formados. Ao invés de sortear dois genes isolados, sorteiam-se dois blocos e em seguida é feita a inversão de posição. Uma desvantagem notável é que dependendo do nível de importância, é preferível quebrar um bloco do que entrar em conflito com uma preferência de um professor ou outro critério qualquer.

3.6.7 Verificação do critério de sobrevivência

Verifica-se os critérios de substituição da população ancestral pela população descendente, ou seja, avaliar quando a substituição é interessante para o desempenho do algoritmo. Foram abordadas por [Timóteo02] duas formas para substituição da população:

1. Sempre substituir os ancestrais pelos descendentes;
2. Substituir os ancestrais pelos descendentes somente se a média dos descendentes for maior ou igual a média dos ancestrais.

3.6.8 Critério de Parada

O critério para finalizar o algoritmo genético adotado por [Timóteo02] é o número de gerações. O critério de tempo também poderia ser utilizado, mas nesse caso o número de gerações processadas iria ser determinado pela velocidade do *hardware* utilizado.

3.6.9 Observações

A implementação dos métodos de seleção, dos operadores de *crossover* e mutação, da geração da população inicial e outros passos de um algoritmo genético, não garante por si só, a geração de boas soluções. Após a etapa de implementação, torna-se necessário calibrar o algoritmo, configurando os parâmetros genéticos de acordo com as necessidades do problema. Depois de elaborar todos os passos necessários para desenvolver um algoritmo genético específico para a resolução do problema das grades escolares, tornou-se necessário criar uma aplicação para que os testes pudessem ser executados. As modificações e adaptações do presente trabalho foram feitas no algoritmo e na aplicação originais.

Capítulo 4
A Solução Modificada

O primeiro passo para a realização do presente trabalho foi compreender de forma clara a maneira como o Algoritmo Genético foi implementado por [Timóteo02] para resolver o problema de geração de horários de uma instituição de Ensino Médio.

Foi necessário um estudo minucioso sobre como os métodos e funções resolviam cada parte do problema para gerar grades horárias satisfatórias. Foram selecionadas quais as porções do programa poderiam ser mantidas (ou sofrer poucas modificações), quais sofreriam grandes ajustes e o que deveria ser adicionado para resolver o problema de uma instituição de Ensino Superior como a UFLA. Todas estas tarefas constituem a etapa denominada manutenção de *software* (fase da Engenharia de *Software*), que [Pressman95] divide em quatro atividades que são levadas a efeito depois que um *software* é liberado para o uso:

- Corretiva – fase de diagnóstico e correção de um ou mais erros
- Adaptativa – modifica o *software* para que ele tenha uma interface adequada com o ambiente (*hardware*, sistemas operacionais) mutante
- Perfectiva – ampliações, modificações em funções existentes, atendimento a pedidos de aumento na capacidade são realizados nesta atividade
- Preventiva. – atividade que modifica o *software* para melhorar a confiabilidade ou a manutenibilidade futura

O enfoque esteve mais na fase perfectiva, mas sempre se atentando às outras fases, para se obter um aplicativo mais fácil de ser mantido e/ou adaptado posteriormente.

A explicação sobre as modificações será feita ao longo das próximas seções deste capítulo, dividindo-se entre as mudanças no AG e as mudanças no aplicativo em si. Os dados referentes aos departamentos, salas, disciplinas,

códigos, enfim, as informações sobre a Instituição foram retiradas do Manual Acadêmico 2002 [Manua02] e junto aos membros da PRG.

4.1 Modificações na Aplicação

Após o estudo da implementação do programa original, seguiu-se um aprofundamento de técnicas de programação de Banco de Dados em *Delphi* (que foi a ferramenta utilizada pelo modelo, bem como o Banco de Dados *Paradox*), de vital importância para uma maior compreensão dos métodos utilizados, que fazem uso intenso de consultas, além de inserções e modificações nos dados gravados sobre as turmas cadastradas, professores, horários, enfim, todas as informações necessárias para gerar os horários.

Como o objetivo do trabalho é adaptar o programa para uma instituição de ensino superior, algumas variáveis e recursos adicionais deveriam ser incorporados e levados em consideração, dependendo de sua importância no problema. Assim, foram feitos o levantamento e elaboração de novos métodos que conseguissem lidar com estes recursos, alocando-os de maneira satisfatória para gerarem bons resultados.

Durante todo esta Seção, será apresentada a aplicação desenvolvida após as modificações. Como muitos aspectos e partes da aplicação original implementada por [Timóteo02] foram mantidos, as explicações serão feitas acerca do sistema, apenas destacando o que se refere às adaptações do presente trabalho.

4.1.1 Dados de entrada

Os dados que deveriam ser inseridos para executar o algoritmo implementado por Guilherme eram:

- Os horários disponíveis (de acordo com a duração das aulas)
- As turmas

- As disciplinas
- Os professores e o número de aulas semanais
- As preferências de horário para cada professor.

Após as alterações no algoritmo, mais dados de entrada se fizeram necessários, além dos já mencionados:

- As salas disponíveis (teóricas e práticas)
- Especificar salas para disciplinas que necessitem de salas especiais
- Cursos e número de módulos para conclusão destes (assim, cada curso possui o número de turmas correspondente ao número de módulos)
- As preferências de horário da tarde e da manhã para cada professor
- Esquema vigente do semestre (aulas de manhã para módulos ímpares ou vice-versa).

A seguir, serão apresentados estes recursos acrescentados, um de cada vez, assim como a maneira com que estes afetaram o código original e seus resultados.

4.1.2 Salas

Uma das primeiras diferenças distinguidas foi a necessidade de distribuição deste recurso durante os dias da semana para as disciplinas. Na UFLA, como em muitas Instituições de Ensino Superior, há salas destinadas às aulas teóricas e práticas. Porém, há disciplinas que necessitam de salas de capacidade adequada para abrigar um maior número de alunos, pois são destinadas a disciplinas lecionadas a mais de uma turma. E, além disso, é necessário que tais disciplinas e seus locais sejam idênticos em todas as turmas envolvidas, pois ocorrerão ao mesmo tempo.

A tabela do Banco de Dados que armazena as salas é da forma como se apresenta na Tabela 4.1. Os campos código da sala, onde se localiza dentro da universidade e quantos alunos comporta foram criados conforme estão no

Manual Acadêmico da UFLA 2002 [Manua02]. Já o campo que indica se a sala pode ser utilizada por qualquer turma (como as salas dos pavilhões) ou se é destinada a alguma disciplina em especial foi criado para facilitar no momento de alocar as salas para as disciplinas específicas. Alguns de seus registros são (a tabela na realidade contém um grande número de locais cadastrados):

Tabela 4.1: Algumas salas da UFLA [Manua02]

Código	Local	Capacidade
0101	ANFITEATRO DE ZOOTECNIA	160
0102	ANFITEATRO DE QUIMICA	120
0225	LABORATORIO DE METALOGRAFIA	25
0226	LABORATORIO FOTOINTERPRETACAO	30
0227	LABORATORIO DE TOPOGRAFIA	25
0279	SALA DE NECROPSIA	25
0333	SALA 8 DA ENGENHARIA	50
0443	SALA 3 DA AGRICULTURA	30
0481	SALA 1 DO PAVILHAO II	55
0482	SALA 2 DO PAVILHAO II	55
0571	SALA 1 DO PAVILHAO I	25
0572	SALA 2 DO PAVILHAO I	55
0573	SALA 3 DO PAVILHAO I	25
0583	SALA 13 DO PAVILHÃO I	80
0640	LABORATÓRIO DE INFORMÁTICA I	30
0641	LABORATORIO DE COMPUTAÇÃO I	25
0642	LABORATÓRIO DE COMPUTAÇÃO II	25

O aplicativo desenvolvido é capaz de realizar a inserção de novas salas (com todos os seus campos devidamente preenchidos) e exclusão dentre as cadastradas. A Figura 4.1 mostra o formulário responsável pela inserção e a Figura 4.2 a exclusão:

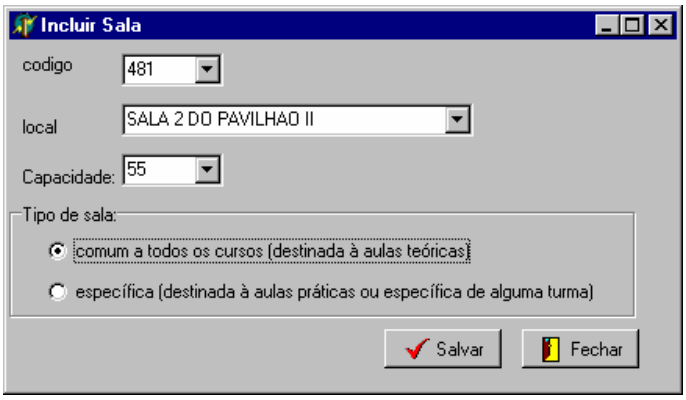

Figura 4.1: Formulário de inclusão de salas

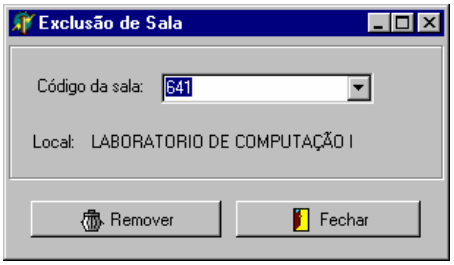

Figura 4.2: Formulário de exclusão de salas

Desta forma, todas as salas, juntamente com as informações correspondentes, ficam armazenadas no Banco de Dados para serem utilizadas pelo algoritmo.

4.1.3 Disciplinas

A Tabela 4.2 mostra todos os departamentos existentes na UFLA, juntamente com seus códigos e os códigos das disciplinas associadas. Os dados referentes às disciplinas que devem ser preenchidos pelo usuário nesta nova versão do programa são conforme ilustrado na Figura 4.3:

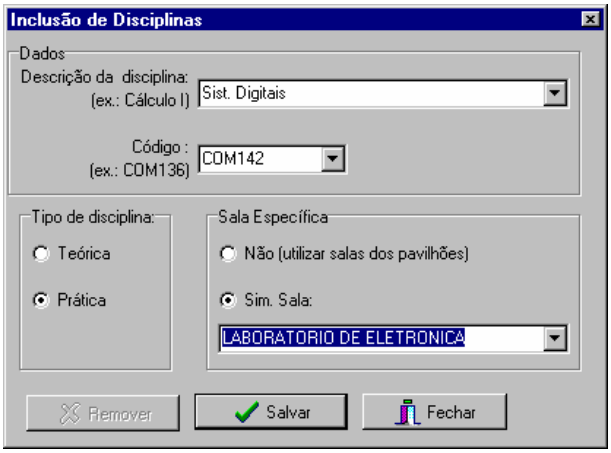

Figura 4.3: Controle das disciplinas

- Código da disciplina
- Descrição (o nome da disciplina)
- Tipo (teórica ou prática)

- Sala utilizada (pode-se definir se a disciplina será lecionada em salas de uso comum ou em algum local específico – informações sobre as salas já devem ter sido armazenadas).

Tabela 4.2: Departamentos, seus códigos e códigos de disciplinas

DEPARTAMENTO	*CÓDIGO*	*COD.DISC.*
Administração e Economia	DAE	EAS
Agricultura	DAG	FIT
Biologia	DBI	BIO
Ciência da Computação	DCC	COM
Ciência dos Alimentos	DCA	ALI
Ciências do Solo	DCS	CSO
Ciências Exatas	DEX	CEX
Ciências Florestais	DCF	CIF
Educação	DED	EDU
Educação Física	DEF	EFD
Engenharia	DEG	ENG
Entomologia	DEN	ENT
Fitopatologia	DFP	FIP
Medicina Veterinária	DMV	VET
Química	DQI	QUI
Zootecnia	DZO	ZOO

Com estes dados já inseridos, o cadastro de professores poderá ser feito juntamente com as disciplinas ministradas por eles.

4.1.4 Turmas

Ao se cadastrar as turmas, ao invés de manter a possibilidade de apenas armazenar os nomes destas, como no programa original, incluiu-se o campo módulo. É muito útil se distinguir qual o semestre a turma está cursando para que se aloque o período matutino ou vespertino para a mesma (a UFLA não oferece aulas no período noturno). O sistema adotado pela Universidade é que, durante um intervalo de dois anos, os módulos ímpares ou os pares tenham aulas no período da manhã, e decorrido este tempo, o esquema se inverta, é o chamado "tombo".

As informações necessárias para preenchimento da tabela de turmas se apresentam como no formulário da Figura 4.4 (a) e (b). (a) diz respeito ao gerenciamento dos cursos, de acordo com o número de módulos ou semestres referentes a este (tempo médio para conclusão), turmas são criadas. Por exemplo, o curso de Ciência da Computação leva em média quatro anos ou oito semestres para ser concluído, o que gera a existência de oito turmas, que gerarão oito grades horárias especialmente para este curso. A Figura 4.4 (b) mostra o gerenciamento de turmas separadamente, se for necessário excluir apenas uma turma de um curso ou inserir pode-se utilizar este formulário.

Figura 4.4 (a) e (b): Formulário de gerenciamento de cursos e turmas

E ainda há a possibilidade de se excluir um curso em (a), o que acaba removendo todas as turmas do mesmo. É importante observar que o número de turmas é realmente grande em relação ao Colégio Lourdes, que possui apenas sete turmas (a UFLA oferece sete cursos, cujos quais possuem de oito a dez semestres, ou seja, oito a dez turmas para cada curso). Isto gera uma maior complexidade no algoritmo que lidará com muito mais dados e restrições a fim de gerar o resultado final.

4.1.5 Horários

Após a inserção dos módulos das turmas (juntamente com os outros dados a respeito destas), foi necessário criar uma maneira de se utilizar esta informação para se gerar os horários com os períodos de aulas de acordo com o esquema vigente no semestre.

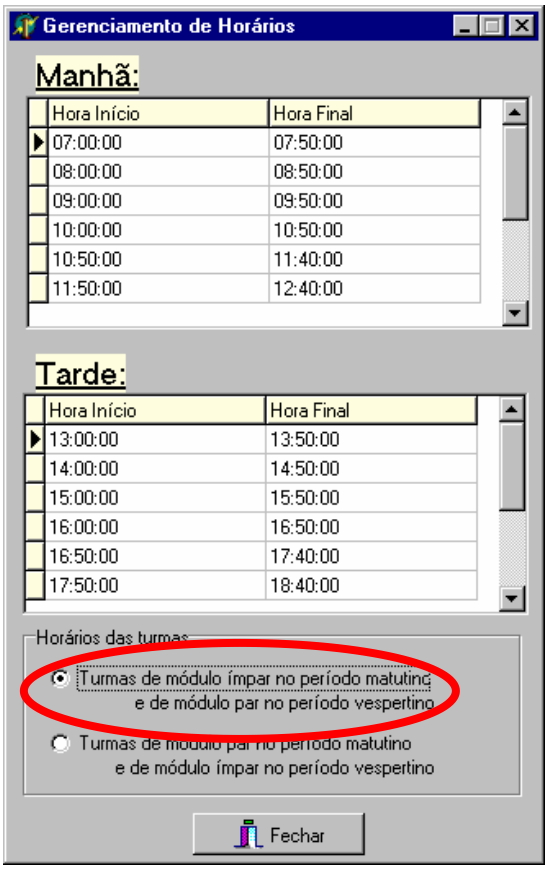

Figura 4.5: Formulário de visualização dos horários disponíveis

Os períodos de aulas matutino e vespertino são visualizados em um único formulário (no programa original só havia os horários da manhã), onde foi incluída a opção com as duas combinações possíveis (módulos pares/manhã e ímpares/tarde e vice-versa), como mostrado na Figura 4.5 e que não era necessária no original. Armazenada esta informação, o sistema poderá alocar as turmas de forma a satisfazer a condição do semestre corrente.

Esta se mostrou uma diferença notável em relação ao aplicativo de [Timóteo02], onde as turmas freqüentavam o colégio apenas no período da manhã e não havia preocupação com módulo.

4.1.6 Departamentos

A divisão da Instituição em departamentos também é uma inovação em relação ao Colégio. Ao inserir os departamentos (nomes, códigos e códigos das disciplinas) obtém-se dados adicionais para se trabalhar com as informações da Universidade, utilizando melhor as categorias e trabalhando de forma mais organizada.

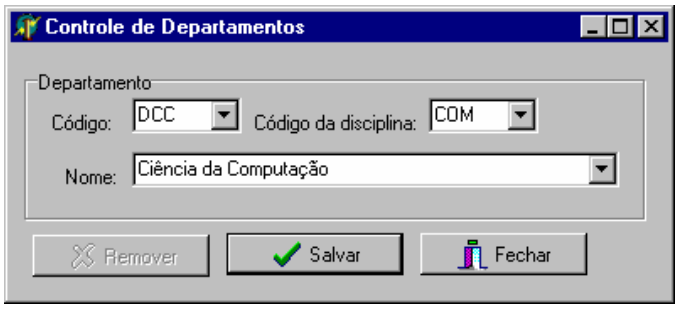

Figura 4.6: Formulário de controle dos departamentos

As conexões do cadastro da Figura 4.6 com as outras partes do sistema são que cada professor é vinculado a um departamento, bem como cada disciplina (campo 'código' da Figura 4.3).

4.1.7 Professores

O cadastro de professores consiste de três etapas: cadastro do nome do professor mais o departamento ao qual pertence (Figura 4.7), a informação sobre as disciplinas lecionadas por ele e o número de aulas semanais para uma turma específica (Figura 4.9) e o cadastro de preferências de horários (Figura 4.8).

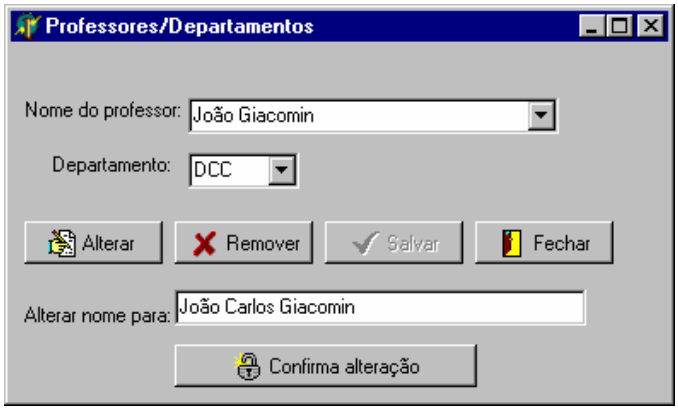

Figura 4.7: Formulário de cadastro de professores/departamentos

No momento da inserção dos nomes dos professores, a escolha do departamento deverá ser feita dentre as opções retornadas pela consulta à tabela de departamentos, ou seja, é necessário que esta já esteja preenchida. Adicionalmente, é possível realizar alteração no nome do professor, no departamento ao qual pertence ou ambos. Neste momento, foi implementado o preenchimento automático das preferências do professor, com a finalidade de facilitar o cadastro, pois o número de professores de uma Universidade é bem grande e poderia se tornar cansativo preencher o formulário para cada um dos dias da semana. O preenchimento é feito atribuindo-se todos os horários (de manhã e à tarde) e dias da semana como preferenciais (ou 'verde'), menos o sábado, que é um dia um pouco incomum para ocorrerem aulas e que é atribuído como 'vermelho'. Um exemplo de um formulário já preenchido pelo algoritmo diretamente está na Figura 4.8. Mas para as disciplinas que deverão ser lecionadas neste dia, basta-se inverter as preferências.

E para que se obtenha um resultado satisfatório em relação às preferências de horários (e que disciplinas não sejam alocadas em dias indesejados), é importante que o grau de importância do quesito "preferência dos professores" esteja bem elevado e, adicionalmente, foi atribuído no próprio algoritmo uma penalidade maior (chamada de penalidade rígida) para quando ocorresse o choque de preferência, o que simula uma restrição rígida (conceito

definido na Seção 3.5 e pretende descartar o horário onde o choque de preferência aparece. A intenção ao realizar este procedimento era a de eliminar qualquer possibilidade de haver aulas no sábado (a não ser aquelas que deveriam ocorrer neste dia), porém, isto privilegia o critério de 'preferências dos professores' e acaba não avaliando adequadamente os outros critérios, independente dos valores de suas prioridades. Por exemplo, se é atribuída uma prioridade altíssima (valor 15 de penalidade) a um horário esparso que ocorre em um indivíduo, pretende-se descartar o horário que apresente esta característica, entretanto, se no momento da seleção, o método tiver que escolher entre este horário (de *fitness* baixo devido à penalidade do 'horário esparso') e um outro que apresentou um choque de preferência de professor (cujo valor de penalidade rígida é de 1000), ele optará obviamente pelo primeiro que possui um valor de *fitness* maior (este é calculado pela fórmula (*1/1+penalidades*)). Ou seja, praticamente só serão descartados horários que apresentem choques de preferências de professores.

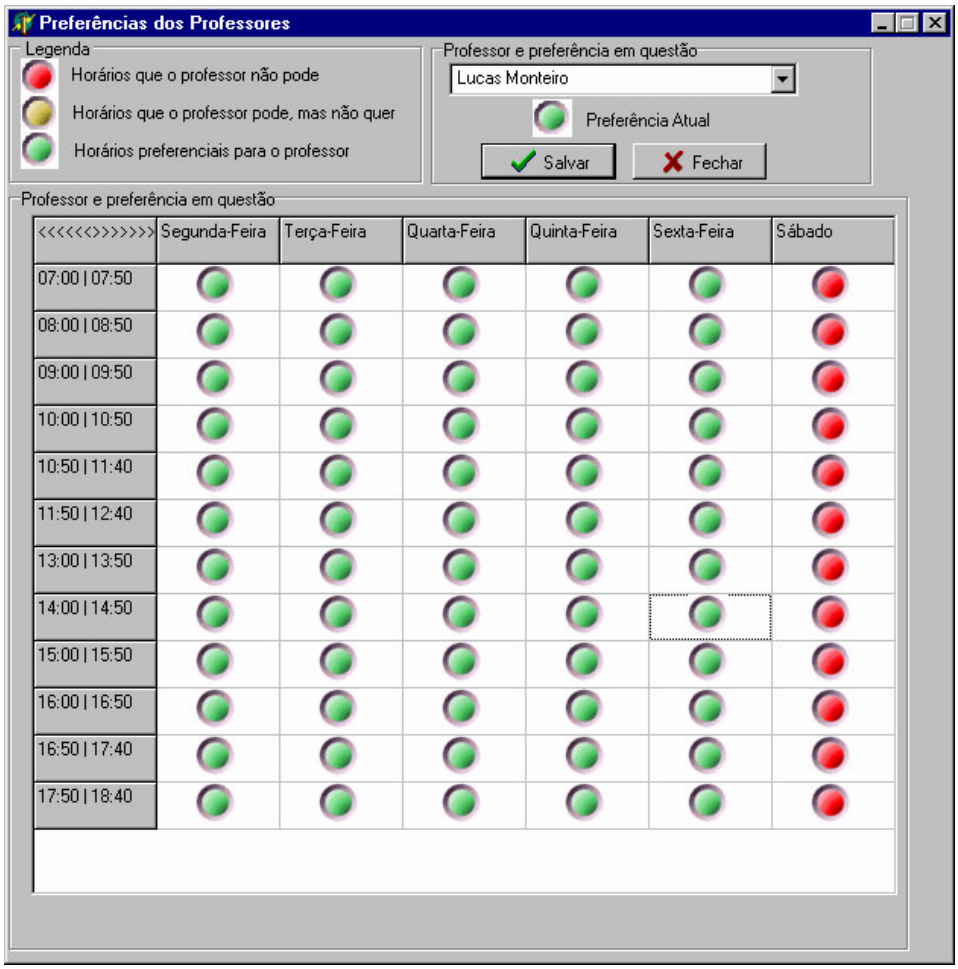

Figura 4.8: Exemplo de preferências preenchidas pelo algoritmo no momento da inserção do nome do professor

Finalmente, uma das mais importantes informações acerca dos professores: as disciplinas lecionadas por eles, para quais turmas e quantas aulas devem haver por semana. Além disso, existe o campo de número de alunos matriculados que, juntamente com a capacidade das salas (Figura 4.1) poderia resultar na alocação mais coerente das salas, de acordo com a quantidade de alunos. O comportamento do formulário de disciplinas/professores está ilustrado na Figura 4.9:

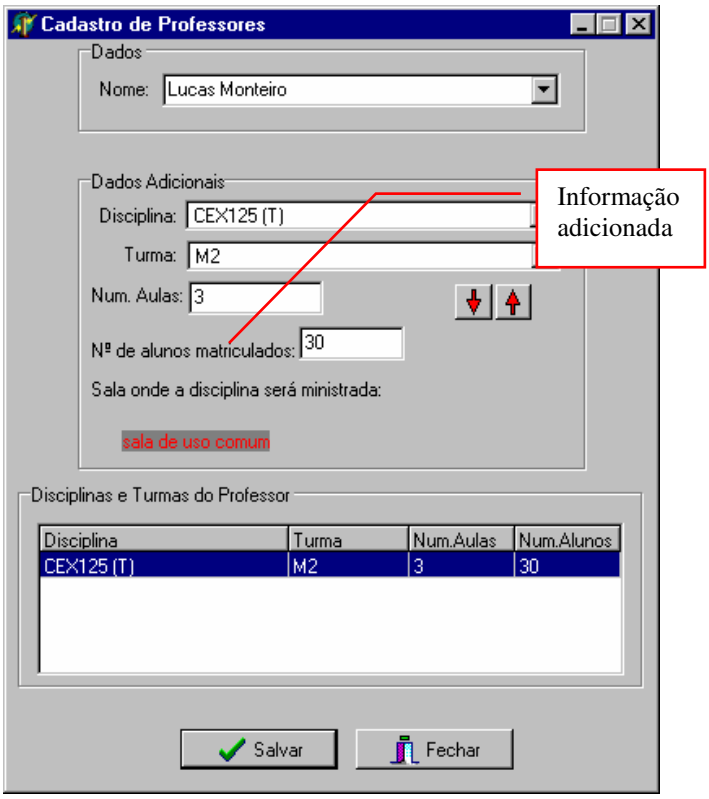

Figura 4.9: Cadastro de disciplinas lecionadas por professor

4.1.8 Ajuste dos parâmetros Genéticos

O usuário deve definir alguns parâmetros do algoritmo genético para que este funcione de forma a encontrar uma solução mais próxima da desejada. Alguns destes parâmetros são: taxa de *crossover*, taxa de mutação, método de seleção de indivíduos, entre outros. Por isso, a aplicação desenvolvida manteve o formulário responsável pela configuração destes parâmetros, conforme a Figura 4.10. O *frame* 'Avaliação dos Indivíduos' da Figura 4.10 se refere aos critérios de avaliação dos indivíduos. Este é um dos aspectos mais específicos do problema tratado, juntamente com a representação deste. Avaliando-se e atribuindo as penalidades adequadamente, consegue-se selecionar os indivíduos mais aptos para o cruzamento. O critério de colisão por sala foi criado para esta aplicação, não existindo no modelo.

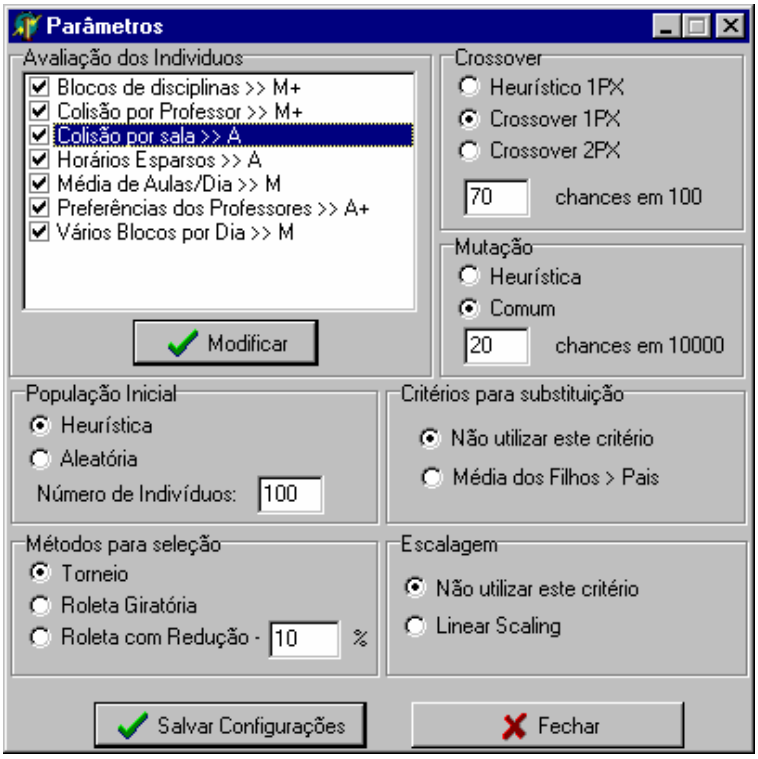

Figura 4.10: Configuração dos parâmetros genéticos

A descrição destes critérios, e as modificações destes são:

1. Blocos de Disciplinas - avalia se as disciplinas encontram-se agrupadas em blocos de duas

2. Colisão por Professor - avalia se um professor está lecionando mais de uma aula ao mesmo tempo, agora se levando em conta o módulo da turma. Ou seja, este critério passa a avaliar apenas as turmas que possuem aulas no mesmo período do dia

3. Colisão por sala – foi criado para que, similarmente à colisão por professor, avalie se uma mesma sala foi alocada para turmas diferentes no mesmo horário e no mesmo período do dia

4. Horários Esparsos - avalia a existência de janelas na grade horária formada

5. Médias Aulas/Dia - avalia se o número de aulas por dia está de acordo com a média de aulas para a turma

6. Preferências dos Professores - avalia se uma determinada disciplina está alocada de acordo com as preferências do professor, que passaram a ser divididas em preferências da manhã e da tarde

7. Vários Blocos por Dia - avalia se em um dia existem mais de 2 aulas de uma mesma disciplina.

E ao lado da descrição do critério aparece o grau de importância para o mesmo, e que varia de acordo com o caso, podendo o usuário alterar cada importância de acordo com sua necessidade. Os graus de importância existentes são:

A+ - nível de prioridade altíssima

A - nível de prioridade alta

M+ - nível de prioridade variando entre média e alta

M - nível de prioridade média

B+ - nível de prioridade baixa

B - nível de prioridade baixíssima

O *frame* 'População Inicial' da Figura 4.10 diz respeito à formação da população inicial, que pode ser obtida a partir de uma forma heurística (levando em conta o critério de alocação de disciplinas em blocos), ou de forma aleatória. Os métodos de seleção também podem ser configurados através da aplicação, bem como as taxas e os tipos de *crossover*, mutação, escalagem e método de substituição.

4.1.9 Dados de Saída

O passo seguinte após o cadastro de todos os dados já mencionados nas seções anteriores, o algoritmo está apto a resolver o problema de alocação de horários. Quando o processamento do algoritmo se encerrar, o usuário pode realizar as seguintes tarefas:

- Salvar a grade horária gerada como resultado
- Verificar as colisões, ou seja, a avaliação daquela solução

- E gerar um gráfico como o da Figura 4.11, que demonstra a evolução da população no decorrer das gerações.

O gráfico gerado fornece informações completas que ajudam a entender o comportamento global do algoritmo. De acordo com a Figura 4.11:

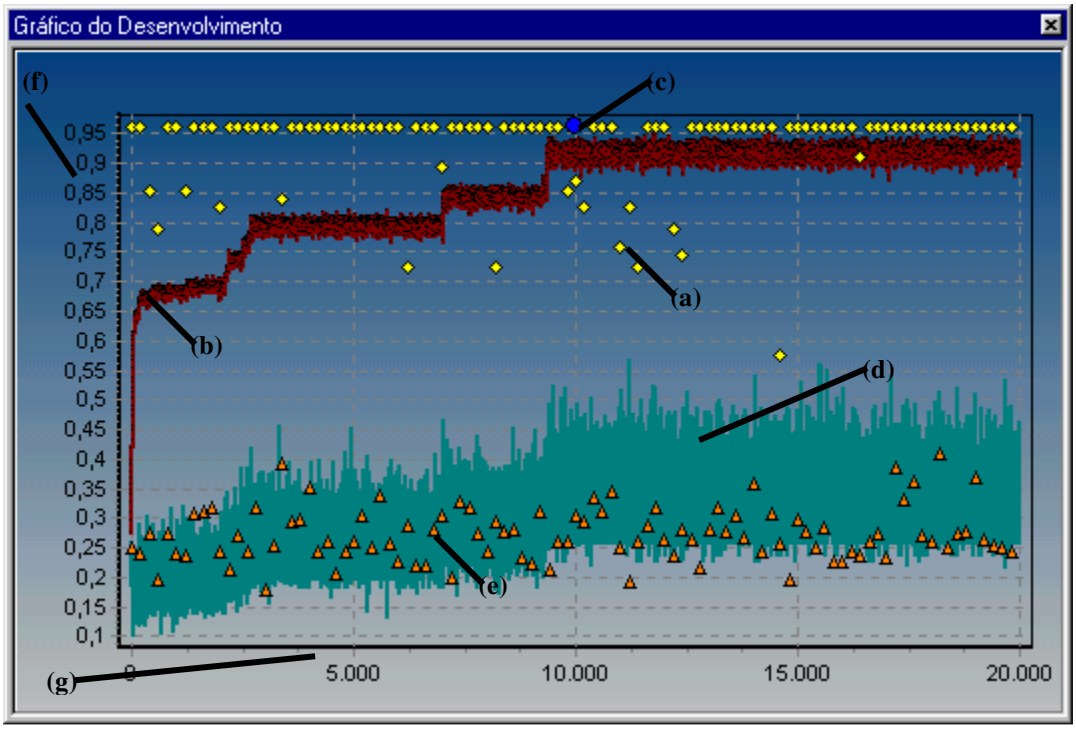

Figura 4.11: Gráfico exemplo

(a) significa o grau de aptidão de cada indivíduo constituinte da população final. Existem vários desses pontos na parte superior do gráfico.

(b) significa a evolução da média do grau de aptidão da população ao longo das gerações.

(c) indica o indivíduo com melhor grau de aptidão encontrado durante todo o processamento.

(d) indica uma noção aproximada da diversidade biológica existente entre os indivíduos constituintes de uma determinada geração.

(e) indica o grau de aptidão da população inicial gerada. É importante observar que neste caso, ao final do processo, a população final (a) está bem mais evoluída que a população inicial (e).
(f) indica o grau de aptidão dos indivíduos que varia de 0 a 1.
(g) indica o número de gerações.

4.2 Modificações na Solução

As modificações descritas na Seção 4.1 dizem respeito à aplicação, como inserir as novas informações e de que forma estas novas inserções afetariam o código. Nesta Seção, o enfoque será dado ao algoritmo em si, demonstrando as modificações em cada etapa do AG. Antes de iniciar este detalhamento, é preciso que alguns pontos sejam esclarecidos:

- Ao realizar as comparações para se encontrar blocos de disciplinas, assumiu-se apenas as disciplinas como parâmetro e não o *slot* disciplina/professor. Isto porque foi verificado, junto à PRG, que somente um professor é responsável por disciplina. Fica a critério do Colegiado de cada curso distribuir as aulas das disciplinas presentes no horário já concluído pelos professores disponíveis
- Cada grade horária corresponde a uma turma e preenche os horários do período matutino ou vespertino. Esta foi a maneira como Guilherme implementou e que foi mantida para este trabalho, ou seja, cada turma só tem aulas no período da manhã ou da tarde.

4.2.1 Representação da solução

Cada indivíduo continua sendo formado por uma matriz de três dimensões como mostrado na Figura 3.1. E como o AG trabalha com uma

população de indivíduos, a representação desta solução é um vetor de matrizes, conforme a Figura 4.12:

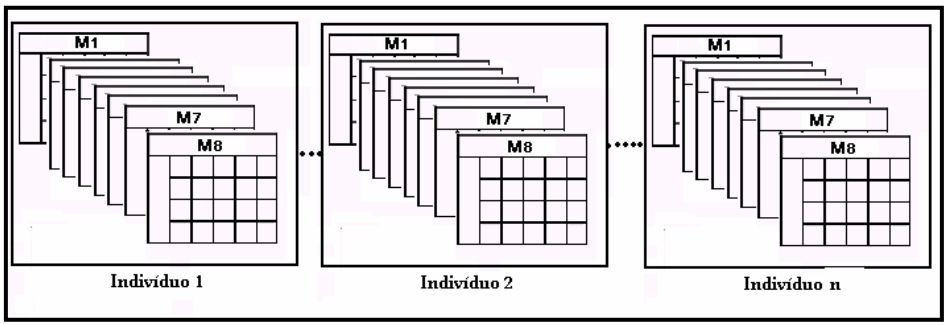

Figura 4.12: Representação da população de soluções

Como já citado, uma população ideal é de 50 a 100 indivíduos segundo [Goldberg89], pode-se afirmar que esta representação de soluções é realmente grande e há outras maneiras de representar a solução (lista encadeada [Abramson92], cada turma representando um cromossomo [Burke94]).

4.2.2 Criação da População Inicial

A dimensão dos cromossomos se manteve a mesma com a adição das salas (turmas x dias x horários). No método aleatório, após o sorteio do horário e do dia a se inserir o *slot* disciplina/professor, fez-se necessária a inclusão do atributo sala em cada cromossomo (além do professor e disciplina já definidos). Para as disciplinas que necessitam de salas especiais para serem ministradas, tais locais já são unidos às disciplinas e atribuídos ao *slot* diretamente, porém, para as que utilizam salas de uso comum, um sorteio é efetuado. As salas comuns formam uma lista, dentro da qual uma será sorteada para cada disciplina que não definiu sua sala anteriormente, o que resulta naquela que será utilizada no horário escolhido. Porém, antes de se alocar este atributo, é verificado se no dia corrente, um bloco de disciplinas foi formado, se isto ocorrer, a sala utilizada é repetida, senão, sorteia-se uma nova. Devido ao grande número de vezes que tal teste é feito ao longo de todas as etapas do AG, foram implementadas duas

funções especialmente para isso, que testam o horário anterior àquele em questão e o posterior, retornando a sala a ser usada.

Para o método heurístico, ao invés de se criar apenas a lista de disciplinas isoladas, uma outra lista de blocos de disciplinas é criada para facilitar no momento de se criar horários que priorizem formação de blocos. Para a alocação de salas, bastou-se utilizar a mesma sala para o horário subseqüente da disciplina, quando estão sendo sorteadas salas para os blocos. Se, no momento da alocação do bloco para dois horários subseqüentes na grade, um bloco maior de três ou mais disciplinas é formado no mesmo dia, repete-se a sala para todos os horários. No momento de sortear salas para as disciplinas isoladas, também se incorporou o teste de blocos, procedendo-se da mesma forma que a explicada no método aleatório.

4.2.3 Avaliação dos Indivíduos

Após a geração da população inicial (que somente é realizada uma vez, não participa das iterações de acordo com o critério de parada – gera apenas o número de indivíduos definidos no aplicativo), um critério foi acrescentado aos outros já existentes (colisão por professor, horários esparsos, blocos de disciplinas, vários blocos por dia, média de aulas por dia e preferências dos professores) para se retornar o grau de aptidão do indivíduo: colisão por sala. Cada vez que uma sala é utilizada no mesmo horário por outra turma, uma penalidade é indicada e um valor atribuído, de acordo com o que foi escolhido pelo usuário (A+, A, M+, M, B+ e B). Esta função avalia os indivíduos de acordo com as penalidades calculadas de cada critério e posteriormente somadas e retorna uma quantificação do grau de aptidão de cada indivíduo.

A avaliação de todos os critérios sofreu uma alteração: como as turmas estão divididas em turmas da manhã e turmas da tarde, os quesitos colisão por professor e preferências dos professores passaram a ser analisados após um teste. Para se verificar se professores estavam definidos em horários iguais para

turmas diferentes, dever-se-ia analisar se também o período de aulas era o mesmo, uma das utilidades de se saber o módulo da turma. Como as preferências do período da tarde para os professores também foram criadas, para se analisar choques nestas preferências, (se os horários formados não as seguiam) também se fez necessário saber o período (matutino ou vespertino) que estava sendo analisado.

4.2.4 Seleção dos Indivíduos

Esta fase do AG se manteve a mesma, já que ela não lida diretamente com os novos recursos alocados. A função de seleção (roleta giratória, método de seleção por torneio ou da roleta com redução) recebe como parâmetro o grau de aptidão devolvido pela avaliação e retorna somente indivíduos selecionados para o passo seguinte – o *crossover.*

4.2.5 *Crossover*

Para esta importante etapa do AG, onde há o cruzamento dos cromossomos, para a alocação das salas, torna-se necessário um teste. Tanto para o *crossover* com corte em um ponto (1PX) com para *crossover* com corte em dois pontos (2PX), procede-se da seguinte maneira: ao se escolher o ponto de corte e inverter os *slots* a partir deste ponto, verifica-se a possibilidade de, ao alocar estes *slots*, formar-se um novo bloco de disciplinas (disciplinas seguidas) e ter que alocar a sala de forma a utilizar a mesma para o bloco todo. Este procedimento evita a situação de se ter aulas seguidas em salas diferentes. Para o *crossover* heurístico com corte em um ponto, ao sortear uma turma, dia e horário para o corte, é preciso verificar se há a quebra de um bloco, ou seja, se o ponto é a 2ª disciplina do bloco. Este teste é realizado para os dois indivíduos envolvidos no *crossover*, o mesmo ponto é escolhido para os dois, dando preferência para aquele que preservou bloco.

4.2.6 Mutação

Ao realizar a troca de um *slot* disciplina/professor, a sala não deve também ser trocada diretamente. Antes, torna-se necessário um teste para descobrir se a disciplina que ocupará um outro local forma um bloco com as disciplinas imediatamente anteriores e posteriores. Na Figura 13 (b), os slots indicados pelas setas sofrerão mutação, tendo suas posições invertidas. O que aconteceria se o teste de blocos de salas não fosse feito: no momento de alocar a disciplina da seta vermelha na posição de seta verde, aquela forma um bloco de três com os slots já alocados e estes já possuem uma sala associada; sem o teste, a disciplina da seta vermelha ocorreria em outra sala, o que não faria sentido, já que as aulas são seguidas.

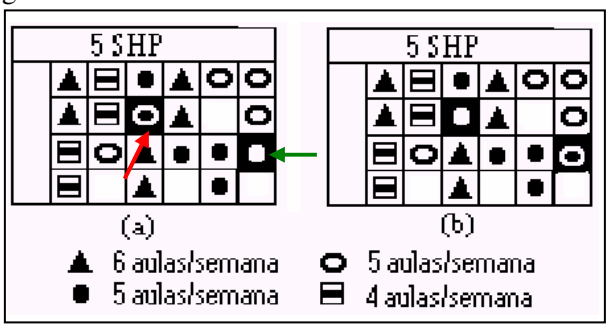

Figura 4.13: Ilustração da mutação onde há a geração de um novo bloco de três disciplinas

4.2.6.1 Mutação Heurística

Este tipo de mutação já preserva os blocos formados, mas é preciso verificar a possibilidade de blocos maiores de disciplinas (três ou mais delas), efetuando o teste como o da mutação simples.

4.2.7 Verificação do critério de sobrevivência

Este passo avalia quando a substituição é interessante para o desempenho do algoritmo. Neste trabalho se mantiveram as duas formas

utilizadas por Guilherme para substituição da população, conforme explicado na Seção 3.6.7:

1. Sempre substituir os ancestrais pelos descendentes
2. Substituir os ancestrais pelos descendentes somente se a média dos descendentes for maior ou igual à média dos ancestrais.

4.2.8 Critério de Parada

Adotou-se para este trabalho o mesmo critério de parada adotado pelo programa original, que foi o número de gerações.

4.2.9 Observações

Aqui, serão relatadas algumas observações acerca dos testes realizados e que constituem uma forma de avaliar a solução encontrada. Os testes foram realizados com 8 turmas normais do curso de Ciência da Computação, 6 turmas especiais, 21 professores, 3 salas especiais (e que foram alocadas para algumas das disciplinas) e 7 salas comuns. No trabalho de [Timóteo02] foi concluído que os métodos de seleção por torneio, o operador de *crossover* com corte em um ponto e o operador de mutação comum apresentaram os melhores resultados, ou seja, o desempenho do algoritmo já foi testado. Por isso, estes foram os mais utilizados para os testes. Os outros operadores também foram testados, porém como seus resultados não foram melhores do que aqueles obtidos com os citados, não foram mostrados. Além disso, o tempo para gerar cada solução foi de 35 min a 1h e 30 min, não se devendo perder tempo com testes que não trariam boas grades. E o interesse ao realizar os testes está em avaliar se as inclusões dos novos recursos realmente foram utilizadas para criar e avaliar os indivíduos. Os próximos tópicos mostram como os diferentes ajustes resultaram em diversas soluções. Os gráficos foram usados para verificar a diversidade da população e a taxa de aptidão dos indivíduos, descritos na Seção 4.1.8.

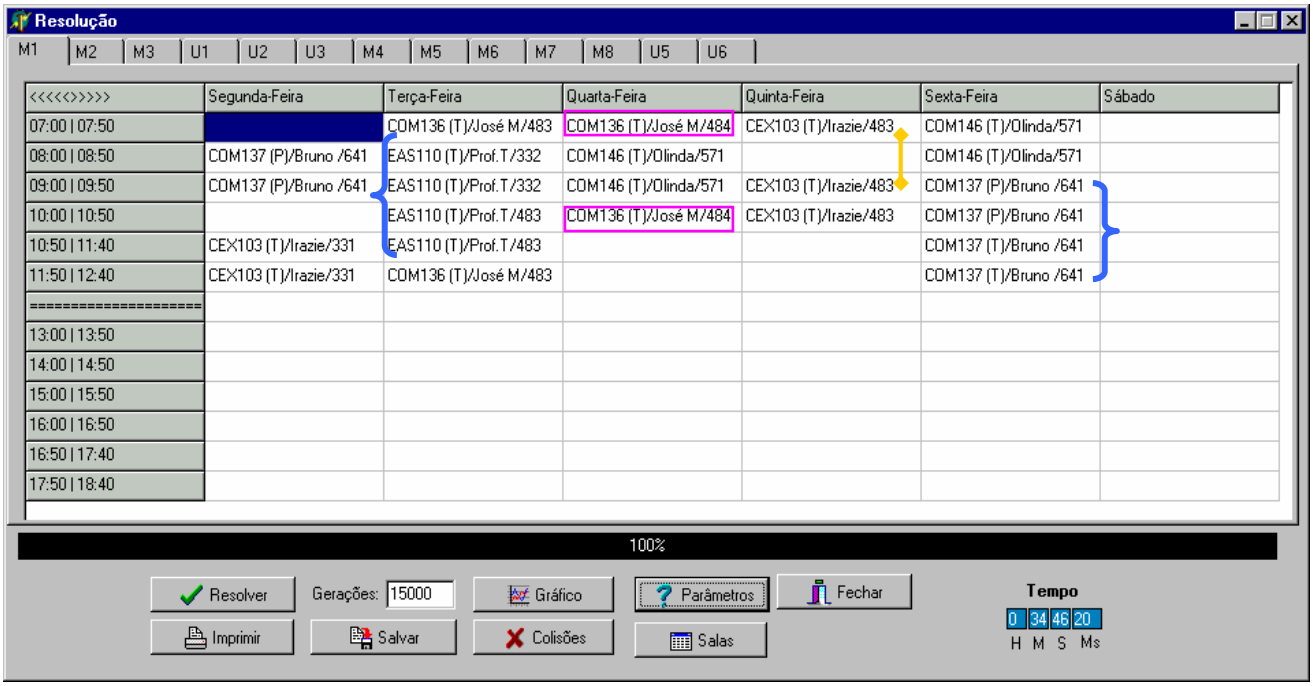

Figura 4.14: Grade horária para a turmas M, 1º módulo, do curso de Ciência da Computação para 15000 gerações e horário no sábado

Antes de mostrar os resultados obtidos com os ajustes dos diferentes parâmetros genéticos, é preciso fazer algumas observações. Na Figura 4.14, há uma grade gerada com 15000 gerações e prioridade altíssima para critérios de avaliação como 'horários esparsos', 'blocos de disciplinas' (quadrado rosa), 'vários blocos/dia' (chave azul) e 'preferências dos professores' e se pode perceber que nenhum deles foi atendido nesta melhor solução, apenas o último, devido à atribuição de penalidade altíssima para ele (para que não houvesse aula no sábado). A cada vez que um choque de preferência ocorria, um valor muito alto de penalidade era atribuído (a chamada penalidade rígida), o que não retornava resultados satisfatórios. Uma observação a ser feita é em relação à presença de janelas (ou 'horários esparsos'). Na solução original, o número de aulas semanais praticamente preenchia todos os horários das turmas e o número de janelas era bem reduzido, tornando-se simples descartar horários onde isto ocorria. No presente trabalho, o número de aulas semanais é bem menor do que

o número de horários para todas as turmas, tornando-se quase impossível descartar os horários que apresentam janelas.

- Teste 1: Com a mutação heurística atentando para blocos de salas seguidas observou-se que três vezes salas diferentes foram utilizadas para blocos de disciplinas. Isto demonstra que não basta apenas a mutação cuidar das salas para as disciplinas, era necessário que no *crossover* isto também fosse realizado. Outro detalhe é a diminuição da biodiversidade (Figura 4.15) utilizando a mutação heurística, conforme verificado em [Timóteo02].

Nº de gerações	5000
População Inicial	Aleatória
Seleção	Torneio
Crossover	1PX
Mutação	Heurística

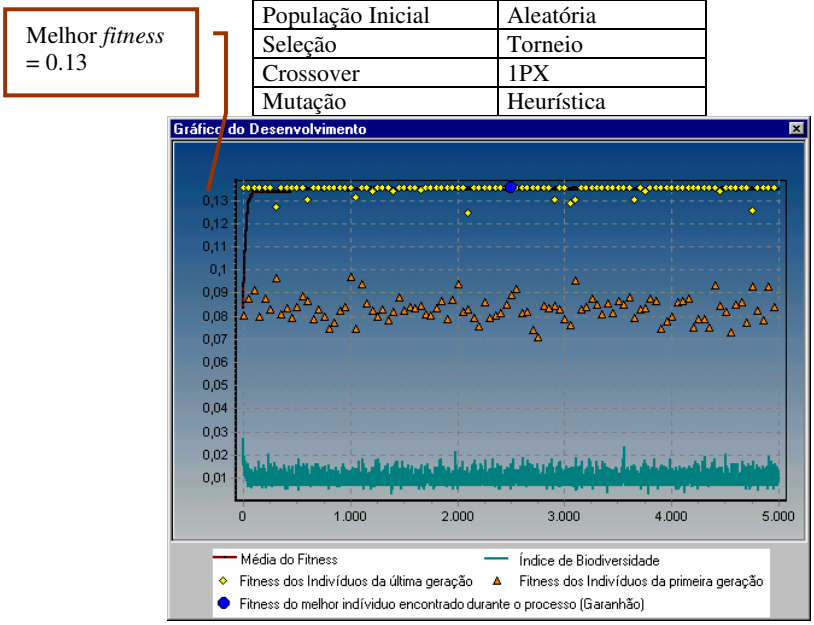

Figura 4.15: Resultado para 5000 gerações

- Teste 2: A mutação comum cuidando de blocos de salas seguidas forneceu uma biodiversidade maior (Figura 4.16).

Nº de gerações	15000
População Inicial	Aleatória
Seleção	Torneio
Crossover	1PX
Mutação	Comum

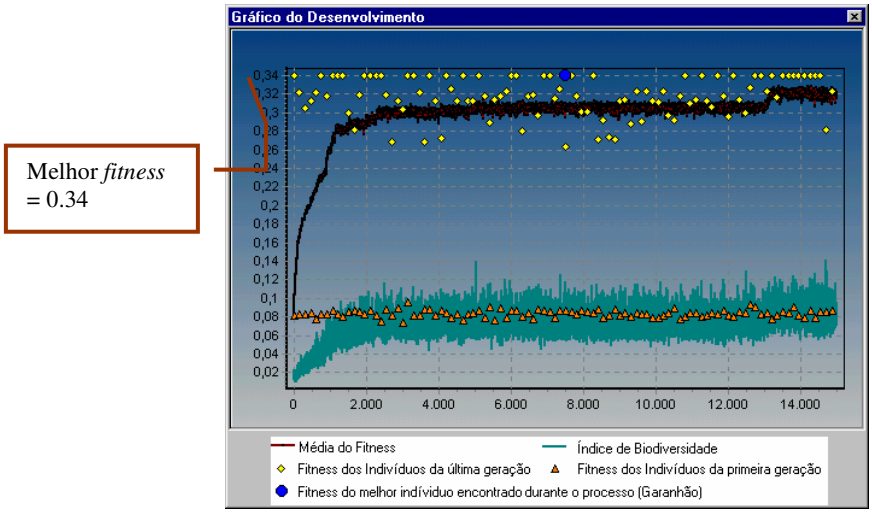

Figura 4.16: Resultado para 15000 gerações

- Teste 3: Apesar da prioridade ser altíssima para o critério de 'horários esparsos', tal fato ocorreu, como pode ser percebido na Figura 4.17, e que já foi explicado conforme a Figura 4.14. Blocos de disciplinas não foram mantidos. O único critério seguido foi o de 'preferências dos professores', pois no sábado não ocorreu nenhuma aula, e o maior *fitness* retornado foi de 0.1, provavelmente devido à alta penalidade atribuída a este último critério.

Nº de gerações	15000	Blocos de disciplinas: A Colisão por Professor: M+ Colisão por sala: M+ Horários Esparsos: A+ Média de Aulas/Dia: M Preferências dos Professores: A+ Vários Blocos por Dia: A+
População Inicial	Aleatória	
Seleção	Torneio	
Crossover	1PX	
Mutação	Comum	

<<<<<>>>>>	Segunda-Feira	Terça-Feira	Quarta-Feira	Quinta-Feira	Sexta-Feira	Sábado
13:00 | 13:50	EFD102 (P)/P. Ed./290		COM139 (T)/Bruno /484	EFD102 (P)/P. Ed./290	COM107 (T)/Joaqui/483	
14:00 | 14:50				COM107 (T)/Joaqui/483		
15:00 | 15:50				COM139 (T)/Bruno /484	COM139 (T)/Bruno /484	
16:00 | 16:50				COM107 (T)/Joaqui/483		
16:50 | 17:40	COM107 (T)/Joaqui/483			EAS103 (T)/Jovino/481		
17:50 | 18:40	COM139 (T)/Bruno /484			EAS103 (T)/Jovino/481		

Figura 4.17: Resultado para uma das turmas do Teste 3

- Teste 4: Para este teste, foi retirado o dia de sábado do horário, bem como a prioridade rígida para 'preferências dos professores', pois isto estava

afetando a solução, como mostrado no Teste 4. Ocorreram as colisões demonstradas na Tabela 4.3.

Nº de gerações	20000	Blocos de disciplinas: A+ Colisão por Professor: M+ Colisão por sala: M+ Horários Esparsos: A+ Média de Aulas/Dia: A+ Preferências dos Professores: M+ Vários Blocos por Dia: A+
População Inicial	Aleatória	
Seleção	Torneio	
Crossover	1PX	
Mutação	Comum	

Tabela 4.3: Colisões retornadas pelo Teste 4

Professores	2
Preferências	10
Blocos	14
Nblocos	0
Janelas	19
Média Aulas/Dia	6
Salas	8

- Teste 5: Aqui a tentativa foi diminuir as colisões e não considerar os horários esparsos. As colisões diminuíram, mas como a prioridade de média de aulas/dia foi diminuída, isto ocorreu mais vezes. Colisões na Tabela 4.4.

Nº de gerações	20000	Blocos de disciplinas: A+ Colisão por Professor: A Colisão por sala: A Horários Esparsos: NÃO CONSIDERAR Média de Aulas/Dia: B Preferências dos Professores: M+ Vários Blocos por Dia: A+
População Inicial	Aleatória	
Seleção	Torneio	
Crossover	1PX	
Mutação	Comum	

Tabela 4.4: Colisões retornadas pelo Teste 5

Professores	2
Preferências	3
Blocos	8
Nblocos	0
Janelas	0
Média Aulas/Dia	24
Salas	2

- Teste 6: Todos os parâmetros equivalentes ao teste 6, porém todos os critérios com prioridade A+ e horários esparsos não considerados. O *fitness* do melhor indivíduo foi 0.17 (devido às altas prioridades).

Tabela 4.5: Colisões retornadas pelo Teste 6

Professores	0
Preferências	1
Blocos	28
Nblocos	0
Janelas	0
Média Aulas/Dia	0
Salas	0

<<<<<>>>>>	Segunda-Feira	Terça-Feira	Quarta-Feira	Quinta-Feira	Sexta-Feira
13:00 | 13:50			COM136 (T)/José M/482	CEX103 (T)/Irazie/584	EAS110 (T)/Prof. /571
14:00 | 14:50	COM146 (T)/Olinda/482	CEX103 (T)/Irazie/584	COM136 (T)/José M/482	CEX103 (T)/Irazie/584	EAS110 (T)/Prof. /571
15:00 | 15:50	CEX103 (T)/Irazie/584	CEX103 (T)/Irazie/584	COM137 (P)/Bruno /641	COM137 (P)/Bruno /641	COM137 (T)/Bruno /641
16:00 | 16:50	EAS110 (T)/Prof. /584		COM146 (T)/Olinda/482	COM136 (T)/José M/331	COM137 (P)/Bruno /641
16:50 | 17:40	EAS110 (T)/Prof. /584	COM137 (P)/Bruno /641	COM146 (T)/Olinda/482	COM136 (T)/José M/331	COM146 (T)/Olinda/482
17:50 | 18:40		COM137 (T)/Bruno /641			

Figura 4.18: Resultado para uma das turmas do Teste 6

- Teste 7: Como nos resultados anteriores foi observado que o indivíduo de maior *fitness* não ultrapassava 0.5 (e o número atingido é de 0 a 1), foram realizados testes especiais, onde somente foram utilizadas 3 turmas, nas quais foram criadas aulas que preenchessem os horários de forma a deixar poucas janelas nos horários. Isto permitiu a criação de indivíduos cujos valores de *fitness* foram mais elevados (maior *fitness* = 0.95).

Nº de gerações	15000	Blocos de disciplinas: B Colisão por Professor: A+ Colisão por sala: A Horários Esparsos: A Média de Aulas/Dia: M Preferências dos Professores: B Vários Blocos por Dia: A+
População Inicial	Aleatória	
Seleção	Torneio	
Crossover	1PX	
Mutação	Comum	

Tabela 4.6: Colisões retornadas pelo Teste 7

Professores	0
Preferências	1
Blocos	3
Nblocos	0
Janelas	0
Média Aulas/Dia	0
Salas	0

- Teste 8: teste especial, como o teste 7, apenas modificando prioridades na tentativa de melhorar as colisões, o maior *fitness* encontrado foi de 0.9 (próximo do indivíduo ótimo: *fitness* = 1). Nos testes 7 e 8, obviamente não houve colisões de salas e professores devido ao pequeno número de turmas.

Nº de gerações	20000	Blocos de disciplinas: A+ Colisão por Professor: M+ Colisão por sala: A Horários Esparsos: A Média de Aulas/Dia: M Preferências dos Professores: B Vários Blocos por Dia: A+
População Inicial	Aleatória	
Seleção	Torneio	
Crossover	1PX	
Mutação	Comum	

Tabela 4.7: Colisões retornadas pelo Teste 8

Professores	0
Preferências	1
Blocos	2
Nblocos	0
Janelas	0
Média Aulas/Dia	1
Salas	0

Capítulo 5
Conclusões

A principal motivação deste trabalho foi de passar o atual sistema de geração de horários utilizados pela PRG da UFLA de informatizado para automático. Como foi adotada uma solução já existente para se modificar e adaptar para novas necessidades, tal tarefa se caracteriza como manutenção de *software*, definida e caracterizada por [Pressman95] como uma atividade que demanda 70% do esforço de criação e utilização de um *software.* A facilidade de se ter uma implementação em mãos é ter uma diretiva a seguir, porém, requer muitas vezes um esforço maior do que implementar uma nova solução. Algumas dificuldades foram encontradas, demonstrando bem o que Pressman afirmou, pois freqüentemente não é tão simples apenas inserir uma nova restrição no problema todo, pois isto acarreta alterações de uma forma global, porque o algoritmo já está implementado para funcionar com os recursos presentes desde o início da modelagem. Tal fato ocorreu diversas vezes durante toda a execução do trabalho, onde uma pequena alteração ocasionou o aparecimento de erros e exceções, devido às complexas ligações entre os componentes do programa.

Todos os testes foram feitos com as turmas, professores e disciplinas do curso de Ciência da Computação, de acordo com o segundo semestre de 2002. O sistema que gerencia o banco de dados utilizados permite a inserção dos outros cursos, juntamente com todos os seus dados, para gerar os horários para a UFLA. O que se esperava com este trabalho era conseguir lidar com as novas restrições, portanto os métodos e etapas do AG foram projetados para este fim.

A dificuldade se encontrou justamente na combinação dos parâmetros para que uma boa solução fosse encontrada. Nos testes realizados, observou-se que muitas vezes critérios que haviam sido atribuídos com alta prioridade não foram levados em conta porque houve a tentativa de não se alocar aulas no sábado, fazendo com que o critério 'preferências dos professores' se sobressaísse na avaliação das soluções. Quando se retirou a possibilidade de

haver aulas neste dia, os resultados corresponderam mais aos ajustes das prioridades, porém, assim mesmo os resultados variaram muito. Para os testes, foram mais utilizados os operadores de *crossover*, mutação, geração de população inicial que [Timóteo02] concluiu serem os candidatos à calibragem ideal do algoritmo (citados na Seção 4.2.10), já que sua essência continuou a mesma, apesar das alterações. O interesse maior estava em observar se as inclusões de recursos se incorporariam ao sistema: salas que deveriam ser alocadas para determinadas disciplinas, sorteio de salas para aquelas que eram de uso comum, se as preferências de horários dos professores de manhã e à tarde seriam seguidas, entre outras inclusões citadas ao longo do Capítulo 4 e nos próximos tópicos. Os testes de desempenho do algoritmo já foram efetuados no trabalho original, aqui a tentativa foi manter os bons resultados, todavia para uma Instituição de Ensino Superior.

Um AG bem ajustado fornece bons resultados, sem (ou com poucas) colisão de salas, professores ou preferências, com recursos alocados de forma satisfatória, bastando associar a estes critérios prioridades adequadas. Mostrou-se bem adaptável. A escolha pela abordagem de AG se deu devido à quantidade de recursos que envolvem o problema *Timetabling*, especialmente o *Course Timetabling*.

A seguir serão enumeradas as restrições que deveriam ser acrescentadas, aquelas que foram incluídas na solução e as que poderiam ser futuramente:

- Inclusão do novo recurso sala para ser distribuída dentre as disciplinas.
- Teste constante, em qualquer alocação de um professor+disciplina+sala (*slot*), se blocos de disciplinas possuem realmente salas iguais para eles, não fazendo sentido ter duas ou mais aulas seguidas em salas diferentes. Isto foi realizado nos passos de geração da população inicial, *crossover* e mutação.
- Além das penalidades anteriormente implementadas, criou-se a penalidade para colisão de salas, para evitar alocação da mesma sala para turmas diferentes.

- Inserção dos novos dados de entrada, como descrito na Seção 4.1.1.
- Conseguiu-se tratar aspectos novos referentes a uma instituição superior e não existentes em um colégio:
    1. salas (armazenamento de nome, local e tipo de aula lecionada para se sortear dentre aquelas que não são especificas de uma disciplina);
    2. adição de mais um dia letivo, o sábado. Na UFLA, este dia é pouco usado para aulas, por isso, torna-se necessário avaliar as preferências dos professores para que aulas que não devam estar neste dia tenham a preferência como proibida. Porém, ao se atribuir a penalidade como rígida para a solução que apresentasse aula no sábado, acabou praticamente descartando apenas horários com esta característica. Isto não é aceitável, já que há outros critérios que têm prioridades associadas e que devem ser levados em conta.
    3. possibilidade da mesma disciplina poder ser de dois tipos: teórica e/ou prática;
    4. disciplina armazenada com o professor + turma + número_de_aulas_semanais + número_de_alunos_matriculados + se é necessária alguma sala específica para ocorrer a aula;
    5. criar as turmas de acordo com o curso e seu número de módulos;
    6. antes de gerar os horários, especificar qual o esquema de aulas do semestre corrente (se turmas de módulos impares/manhã e pares/tarde ou vice-versa) para se distribuir adequadamente as turmas nos dois períodos;
    7. especificar os módulos das turmas como pares e impares, tornando esta informação imprescindível em conjunto com a anterior para se gerar os horários de acordo com o esquema de aulas vigente no semestre;

8. e juntamente com a informação do módulo de cada turma, poder verificar choques de salas, professores, preferências, para as turmas que possuem aulas no mesmo período.

Todos estes aspectos apresentaram resultados favoráveis, porém para que o aplicativo desenvolvido possa ser usado de maneira satisfatória pela UFLA, alguns passos ainda serão necessários e valem como propostas para trabalhos futuros:

- Ao invés de considerar que há um professor por disciplina, cadastrar mais de um, se houver, e distribuir as aulas entre eles. Não se implementou tal possibilidade porque foi levantado junto à PRG que é inserido apenas um professor por disciplina;
- A possibilidade de pré-definir horários para determinadas disciplinas. Ex: dizer que a disciplina de Genética deve ser na terça-feira e na quinta-feira às 14 h. Foi realizada a simulação desta situação para determinados dias, especificamente para aulas no sábado, que não ocorrem normalmente.
- Foi previamente assumido que todos os professores no momento do cadastro não poderiam dar aulas no sábado. Mas para uma disciplina que devesse ocorrer neste dia, era preciso atribuir os outros dias como proibidos, atribuir penalidade altíssima para choques de 'preferências de professores' e isto retornou horários satisfatórios apenas no que diz respeito a este último critério, mas não levou muito em conta outros critérios que possuíam alta prioridade. Para que todos os critérios fossem analisados de forma justa, dever-se-ia adotar uma outra estratégia para alocar aulas no sábado, como considerá-lo um dia isolado em que apenas as disciplinas específicas para este dia fossem alocadas.
- Aulas teóricas antes de aulas práticas de uma disciplina. Também foi pesquisada esta exigência junto à PRG, e foi afirmado que já se tentou levar isto em conta no atual sistema informatizado e que não se consolidou;

- Foi coletada a informação do número de alunos matriculados em cada disciplina, mas que não foi utilizada como critério para escolher a sala, mas que seria uma boa maneira de se selecionar os locais para ministração de aulas;
- Sabe-se que algumas turmas possuem aulas em dois períodos do dia, e a maneira como o programa foi implementado inicialmente, necessitaria de consideráveis modificações a fim de acrescentar esta possibilidade, pois ele cria os horários para um período do dia para cada turma;
- Considerar que a mesma disciplina pode ser dada para várias turmas, e considerar que esta deve estar alocada no mesmo local e horário para as turmas envolvidas, não permitindo que o programa considere isto como uma colisão de professor ou sala.

Finalmente, uma outra sugestão de melhoria para a solução, para que este sistema automático de geração de horários seja utilizado satisfatoriamente na UFLA, seria primeiramente, acrescentar os pontos descritos anteriormente e realizar testes de desempenho utilizando-se os dados de todos os cursos, o que aumentaria a quantidade de dados a serem tratados e poder-se-ia fazer uma análise mais realista do programa desenvolvido.

Capítulo 6
Referências Bibliográficas

[Abramson92] D. Abramsom. *A parallel genetic algorithm for solving the school timetabling problem,* Australian Computer Science Conference, Hobart, 1992.

[Backer85] J. E. Backer. *Adaptive selection methods for genetic algorithm,* Internacional Conference on Genetic Algorithms and Their Applications, 1985.

[Brindle79] A. Brindle. *Genetic algorithms for function optimization*, University of Alberta, 1979.

[Burke94] Edmund Burke, David Elliman and Rupert Weare. *A Genetic Algorithm Based University Timetabling System*, Department of Computer Science, University of Nottingham, Nottingham, NG7 2RD, 1994.

[Burke95] BURKE, E., ELLIMAN, D. G. and WEARE, R. F. *The automation of the timetabling process in higher education*. Journal Educational Technol Systems, v. 23, p. 257-266, 1995.

[Filho00] Geraldo Ribeiro Filho. Melhoramentos do Algoritmo Genético Construtivo e Novas Aplicações em Problemas de Agrupamento. *PhD thesis*, INPE, São José dos Campos, SP, Setembro de 2000.

[Goldberg89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. The University of Alabama, 1989.

[Holland75] J. H. Holland. *Adaptation in Natural and Artificial System*. University of Michigan Press, 1975.

[Júnior00] Osmar de Oliveira Braz Júnior. Otimização de horários em Instituições de Ensino Superior através de Algoritmos Genéticos. (Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção), Universidade Federal de Santa Catarina, Florianópolis, 2000.

[Koza92] J. R. Koza. *Genetic Programming*. MIT Press, 1992.

[Lucas00] Diogo Correa Lucas. Algoritmos Genéticos: um estudo de seus conceitos fundamentais e aplicação no problema de grade horária. Universidade Federal de Pelotas, Dezembro, 2000.

[Manua02] Manual Acadêmico da UFLA . 16. ed. Lavras, MG, 2002. 68 p.

[Mitchell96] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Massachusetts Institute of Tecnology, 1996.

[Pressman95] Roger S. Pressman. Engenharia de Software. São Paulo: *Makron Books*, p. 876 – 915, 1995.

[Schaerf99] A. Schaerf. *A survey of automated timetabling*. Artificial Intelligence Review, (13), 1999.

[Timóteo02] Guilherme Tadeu Silva Timóteo. Desenvolvimento de um Algoritmo Genético para a Resolução do *Timetabling*. (Monografia de Graduação apresentada ao Departamento de Ciência da Computação), Universidade Federal de Lavras, Lavras, Minas Gerais, 2002, 76 p.

[Velloso95] VELLOSO, M. F.; MENDONÇA J. M. ; PACHECO M. A. ; VELLASCO M. M. B. R. *Otimização de Planejamento de Horários por Algoritmos Genéticos*. Congresso Brasileiro de Redes Neurais, Curitiba, 1995.

[Wetzel83] A. Wetzel. *Evaluation of the effectiveness of genetic algorithms in combinatorial optimization*, University of Pittsburg, 1983.