DANIEL TEIXEIRA DOS SANTOS

UMA COMPARAÇÃO ENTRE DOIS ALGORITMOS BIOINSPIRADOS PARA A RESOLUÇÃO DO TIMETABLING

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. Dr. Joaquim Quinteiro Uchôa

LAVRAS
MINAS GERAIS – BRASIL
2009

DANIEL TEIXEIRA DOS SANTOS

UMA COMPARAÇÃO ENTRE DOIS ALGORITMOS BIOINSPIRADOS PARA A RESOLUÇÃO DO TIMETABLING

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 27 de Novembro de 2009

_____

Prof. Claudio Fabiano Motta Toledo

_____

Prof. José Monserrat Neto

_____

Prof. Joaquim Quinteiro Uchôa
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2009

AGRADECIMENTOS

Primeiramente agradeço à minha família, pelo apoio, pela confiança depositada e pela paciência, aos meus amigos, em especial ao Fábio, que me deu grande apoio neste trabalho. Agradeço ao meu orientador, Joaquim, por ter me suportado todo esse tempo, e quase sempre de bom humor. Agradeço à Mohana, pelo amor e carinho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A Computação Bioinspirada é a área da computação que emprega conceitos e técnicas baseados em sistemas biológicos naturais. Elas representam opções robustas e facilmente adaptáveis para abordar uma ampla gama de problemas, em especial os de otimização. Neste trabalho será realizada uma abordagem dos principais conceitos em relação aos Algoritmos Genéticos e o algoritmo imune, Clonalg, bem como a implementação de uma aplicação para o problema da geração de grades horárias, objetivando uma comparação entre os dois algoritmos.

Palavras-Chave: Timetabling; Clonalg; Algoritmos Genéticos; Computação Bioinspirada.

ABSTRACT

The Bio-inspired Computing is the area of computing that uses concepts and techniques based on natural biological systems. They represent robust and easily adaptable options to approach a wide range of problems, especially in optimization. In this study, we conducted an analysis of the main concepts in relation to genetic algorithm and the Clonalg immune algorithm, as well an implementation of an application to the problem of generating time tables, aiming to compare the two algorithms.

Key-Words: Timetabling; Clonalg; Genetic Algorithms; Bio-inspired Computing.

1. INTRODUÇÃO

O problema de Timetabling é uma das mais importantes atividades administrativas de todas as instituições acadêmicas. Ele representa um desafio e importante problema de pesquisa nas áreas de Pesquisa Operacional e Inteligência Artificial desde a década de 60 (Qu et al, 2008).

A solução manual do Timetabling pode levar vários dias de trabalho, e em adição a isto, a solução pode ser insatisfatória. O problema de Timetabling é NP-Completo em quase todas as suas variantes, e normalmente uma solução exata só existe em casos com número inferior a dez disciplinas, enquanto casos reais podem envolver centenas de disciplinas (Schaerf, 95).

Em seu trabalho, (Timóteo, 2002) testou diversos parâmetros e operadores em Algoritmos Genéticos, obtendo boas soluções para o Timetabling. O objetivo deste trabalho é investigar o comportamento do Clonalg, realizando diversos testes de seus parâmetros e comparando com as melhores configurações de Algoritmos Genéticos descritas em (Timóteo, 2002).

Para abordar a resolução do Timetabling, este trabalho foi dividido em seis seções, e a estrutura de cada um deles será descrita a seguir:

Capítulo 2: introduz os conceitos relativos ao problema de Timetabling, sua definição formal, restrições e complexidade.

Capítulo 3: introduz os conceitos relativos à Computação Bioinspirada, os Algoritmos Genéticos, o Clonalg, os principais operados e os parâmetros que podem ser configurados.

Capítulo 4: apresenta a modelagem do problema realizada no trabalho e uma descrição da implementação dos operadores das heurísticas implementadas.

Capítulo 5: descreve a aplicação que foi desenvolvida para a realização dos testes.

Capítulo 6: apresenta os resultados obtidos com diversos parâmetros do Clonalg e as melhores configurações de parâmetros do Algoritmo Genético, critérios de avaliação e comparação entre as melhores soluções.

Capítulo 7: apresenta, por fim, uma conclusão para o trabalho.

2. TIMETABLING

O Timetabling é um problema corrente em todo tipo de atividade onde há necessidade de se alocar horários: enfermarias, esportes, transportes, etc. (Schaerf, 95) cita as variações mais comuns, todas equivalentes em termos de complexidade:

- **School Timetabling:** seqüenciamento semanal de aulas de uma escola, evitando que professores e alunos tenham mais de uma aula ao mesmo tempo.
- **Course Timetabling:** seqüenciamento semestral das aulas de um conjunto de cursos de uma universidade, evitando a simultaneidade de cursos com estudantes em comum.
- **Examination Timetabling:** seqüenciamento de um conjunto de cursos em uma universidade, evitando exames simultâneos de cursos com estudantes em comum, e espalhando os exames o máximo possível ao longo do tempo.

Neste trabalho, por questões de simplicidade de implementação do sistema de testes, e para comparação com os resultados de (Timóteo, 2002), o foco será dado no School Timetabling.

Uma definição geral é dada por (Burke et. al., 2004): o Timetabling é um problema com quatro parâmetros:

- **T**, um conjunto finito de tempos - *Times*;
- **R**, um conjunto finito de recursos - *Resources*;
- **M**, um conjunto finito de encontros - *Meetings*;

- **C**, um conjunto finito de restrições - *Constraints.*

Dados estes parâmetros, o Timetabling consiste em arranjar encontros entre os recursos disponíveis e o *slots* de tempo. Um determinado arranjo será considerado uma solução para o problema se absolutamente todas as restrições rígidas forem satisfeitas, e a qualidade da solução será dada pela quantidade de restrições flexíveis satisfeitas. (Qu et. al., 2008) dá uma definição de restrições rígidas e flexíveis:

- Restrições rígidas - não podem ser violadas sob nenhuma circunstância, geralmente representam restrições físicas. Por exemplo, duas disciplinas alocadas no mesmo horário para uma mesma turma de estudantes. Uma instância de Timetabling que atenda a todas as restrições rígidas é dita solucionável**.**
- Restrições flexíveis: são aquelas desejáveis, mas que não são absolutamente críticas. Na prática, é comumente impossível achar uma solução que satisfaça todas as restrições flexíveis. Como exemplo, temos a restrição de se encontrar uma solução de Timetabling em que não existam aulas em bloco. A qualidade de Timetabling normalmente é medida examinando-se o número de restrições flexíveis violadas na geração da solução.

Segundo (Shaerf, 95), quase todas as variações do Timetabling são NP-Completo, incluindo o School Timetabling abordado neste trabalho.

Os problemas da classe NP-Completo são ditos intratáveis, pois não há evidência de que existam métodos exatos que possam resolvê-los em tempo polinomial. Em seu trabalho (Timóteo, 2002) exemplifica usando o caso do Colégio Nossa Senhora de Lourdes:

- Cada turma possui trinta *slots* de tempo (cinco dias X seis aulas por dia).
- O número de professores de cada turma é dado pela tabela 2.1.

Tabela 2.1: Número de professores de cada turma (Timóteo, 2002)

Turma	Número de Professores
5SHP	8
6SHP	9
7SHP	8
8SHP	9
1M	12
2M	13
3M	14

- O número de possibilidades de cada turma é dado pelo arranjo de *n* tomado de *p* a *p*, em que *n* é o número de *slots* da turma e *p* o número de professores da mesma, como ilustrado na Tabela 2.2.

Total de Possibilidades = $A_{n,p}$:

Turma	Número de Possibilidades
5SHP	$2,3 * 10^{11}$
6SHP	$5,2 * 10^{12}$
7SHP	$2,3 * 10^{11}$
8SHP	$5,2 * 10^{12}$
1M	$2,2 * 10^{16}$
2M	$3,9 * 10^{17}$
3M	$6,6 * 10^{18}$

Tabela 2.2: Número de possibilidades por turma

O total de possibilidades do caso é dado pelo produto entre o total de possibilidades de cada turma, o que nos dá cerca de **8,1 x 10^100** possibilidades.

Como já demonstrado, mesmo um caso real de Timetabling de pequena dimensão pode levar a um número de possibilidades difícil de ser explorado usando métodos exatos ou de força bruta. Deve-se então procurar métodos heurísticos que possam encontrar boas soluções para o problema em um intervalo de tempo razoável.

No próximo capítulo serão detalhadas duas heurísticas inspiradas em modelos biológicos e que podem ser aplicadas a este problema.

3. COMPUTAÇÃO BIOINSPIRADA

A Computação Bioinspirada pode ser encarada como a área da computação preocupada em desenvolver ferramentas e algoritmos baseados em processos biológicos naturais.

As técnicas de computação bioinspiradas são meta-heurísticas que podem ser usadas nos casos que obedecem as seguintes condições:

- Problema com grande espaço de busca de soluções;
- Quando é possível estabelecer métricas de comparação entre soluções;
- Problema que não pode ser apropriadamente modelado ou não há um método exato para sua resolução;
- Diversidade de soluções é um fator importante.

Neste trabalho foram escolhidas duas heurísticas bioinspiradas que serão comparadas na resolução do School Timetabling, os Algoritmos Genéticos e o Clonalg.

3.1 Algoritmos Genéticos

Em 1859, no seu livro intitulado “A Origem das espécies”, Charles Darwin divulgou a Teoria da Evolução Natural.

“A teoria da evolução diz que na natureza todos os indivíduos dentro de um ecossistema competem entre si por recursos limitados, tais como comida e água. Aqueles dentre os indivíduos (animais, vegetais, insetos etc.) de uma mesma espécie que não obtêm êxito tendem a ter uma prole menor e esta descendência reduzida faz com que a probabilidade de ter seus genes propagados ao longo de sucessivas gerações seja menor, processo este que é

denominado seleção natural" (Linden, 2006).

Um dos pioneiros na área de computação bioinspirada foi John Holland. Em 1975, Holland publicou seu trabalho *"Adaptation in Natural and Artificial Systems"*, onde o AG é apresentado como uma metáfora para os processos evolutivos do mundo real, de forma que ele se pudesse simular estes processos em um ambiente computacional (Lima, 2008). O modelo proposto em (Holland, 1962), Algoritmo Genético Canônico, se tornou base para uma grande variedade de Meta-Heurísticas aplicadas à otimização.

Os Algoritmos Genéticos são fortemente baseados na Teoria da Evolução Natural, bem como na genética Mendeliana. Como tal, boa parte da nomenclatura dos Algoritmos Genéticos segue a terminologia utilizada na Biologia (Timóteo, 2002):

- **Cromossomo**: 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;
- **Lócus**: 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;
- **Aptidão**: literalmente, o quanto um indivíduo é apto para determinado ambiente.

Dependendo da implementação, os termos cromossomo, gene, alelo e lócus podem ter o mesmo significado, embora na Biologia representem

entidades diferentes. A execução de um Algoritmo Genético dá-se sobre uma população inicial de cromossomos ou indivíduos, onde serão efetuadas as operações de seleção, cruzamento e mutação, que resultarão em uma nova população. O algoritmo para quando é satisfeita uma determinada condição, comumente um número de gerações de indivíduos. O algoritmo está exemplificado no fluxograma da figura 3.1.

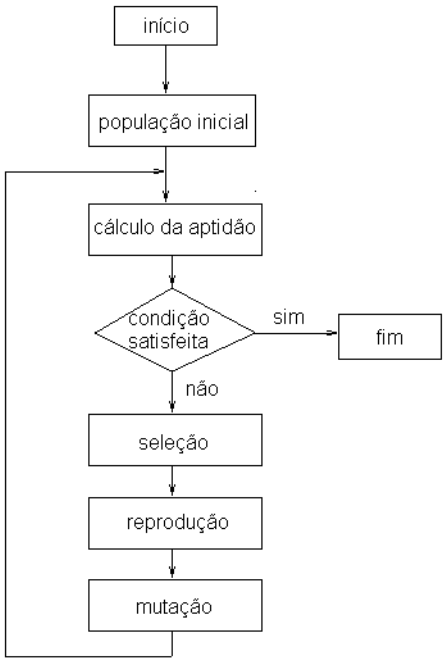

Figura 3.1: Fluxograma do Algoritmo Genético

3.1.2 Representação de Soluções

Um cromossomo pode ter várias implementações. Comumente um

cromossomo é representado como uma cadeia de bits, podendo ser representado também como uma cadeia de caracteres. Cada elemento da cadeia representado um gene.

No entanto, para problemas mais complexos, a representação binária pode se mostrar inadequada. A representação utilizada é extremamente importante para o desenvolvimento e desempenho da solução. Segundo (Koza, 92 *apud* Timóteo, 2002, p. 6), ao optar por uma representação é necessário avaliar características como:

- **Completude**: determina se é possível representar todos os fenótipos possíveis, ou seja, verificar todas as soluções podem ser decodificadas;
- **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.
- **Simplicidade**: representa o grau de complexidade dos atos de codificação e decodificação das soluções;
- **Localidade**: pequenas alterações no genótipo acarretam pequenas alterações em seu fenótipo correspondente.

3.1.3 Geração da População Inicial

Assim como todo algoritmo evolucionário, o Algoritmo Genético tem o seu início com a inicialização de uma população de soluções inicial. Diversos operadores de inicialização de população para geração de melhores populações estão propostos na literatura. (Goldberg, 1989) cita os mais comuns:

- **Inicialização randômica uniforme:** cada gene do indivíduo receberá como valor, um elemento do conjunto de alelos, sorteado de forma

aleatoriamente uniforme;

- **Inicialização randômica não uniforme:** determinados valores a serem armazenados tendem a ser escolhidos com uma freqüência maior do que o restante;
- **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 super-indivíduos tendam a dominar no processo de evolução e causar o problema de convergência prematura;
- **Inicialização heurística:** indivíduos são criados a partir de heurísticas. Essa alternativa apresenta também a possibilidade de convergência prematura.

3.1.4 Avaliação do Indivíduo

Etapa do Algoritmo Genético em que é testada a aptidão de cada indivíduo no ambiente do problema. Para este processo é utilizada uma função de avaliação, ou *fitness*. Em casos como o de otimização de funções, a função de avaliação é a própria função a ser otimizada.

Para problemas de difícil modelagem como o Timetabling, a função de aptidão é baseada em penalidades, descritas pelas restrições impostas pelo problema. Quando menor o número de penalidades, maior a aptidão do indivíduo.

3.1.5 Seleção de Reprodutores

Esta é a etapa do Algoritmo Genético em que são escolhidos os indivíduos que farão cruzamento. A grande maioria dos operadores de seleção

baseia-se no grau de aptidão dos indivíduos da população. (Goldberg, 1989) cita os operadores mais utilizados:

- **Seleção determinística:** a quantidade de descendentes, ou o número de participações no processo de cruzamento, de um determinado indivíduo é proporcional à razão entre a aptidão do indivíduo e o somatório da aptidão dos indivíduos da população.
- **Seleção por Torneio:** *n* pares de indivíduos são sorteados de forma aleatória. O indivíduo de maior aptidão de cada par é selecionado para o cruzamento.
- **Seleção por Roleta Giratória:** simula um sorteio via roleta, em que o tamanho do arco ocupado por cada indivíduo é proporcional à aptidão do indivíduo.
- **Seleção Uniforme:** a seleção é reduzida à uma escolha aleatória, onde cada indivíduo tem a mesma probabilidade de ser selecionado.

Após escolhido o método de seleção, tem-se que escolher os casais para cruzamento. O mais comum é que se faça um sorteio aleatório dos casais, mas além desse método (Goldberg, 1989) cita:

- **Endocruzamento:** indivíduos semelhantes são combinados;
- **Exocruzamento:** indivíduos com diferentes características são combinados;
- **Autofertilização:** um indivíduo pode reproduzir-se de forma assexuada;
- **Propagação Clonal:** réplicas de indivíduos são inseridas na população de descendentes;
- **Acasalamento preferencial:** indivíduos não desejados são combinados.

3.1.6 Cruzamento

O cruzamento é a principal etapa responsável pela variação genética no

cromossomo dos descendentes de uma geração de indivíduos. Após a etapa de seleção, os casais formados sofrem a possibilidade de trocarem parte de sua seqüência genética antes de passá-los aos seus descentes. A troca de genes é orientada por um dos operadores de crossover. Os operadores mais comuns são:

- ***Crossover* uniforme:** para cada gene do descente um valor binário é sorteado. Esse valor indica de qual indivíduo pai o descendente receberá o gene;
- ***Crossover* com corte em um ponto:** uma posição é sorteada no cromossomo. Essa posição é chamada de ponto de corte. A partir do ponto de corte, os genes dos pais serão trocados e passados aos descendentes;
- ***Crossover* com corte em dois pontos:** semelhante ao crossover com corte em um ponto. São sorteados dois pontos de corte, e os genes dos pais entre esses pontos são trocado e passados aos descendentes;
- ***Crossover* com múltiplos pontos de corte:** uma generalização dos outros casos apresentados, onde um número fixo (*n*) de pontos de corte são sorteados;
- ***Crossover* segmentado:** extensão do método de múltiplos pontos de corte, onde o número de pontos de corte é variável para cada execução.

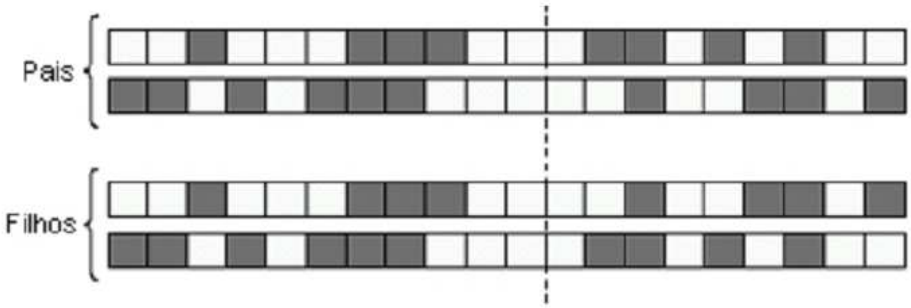

Figura 3.2: Troca genética com um ponto de corte (Timóteo, 2002)

(Goldberg, 1989) atenta para o fato de que os operadores com múltiplos pontos de corte são mais difíceis de implementar, e na maioria dos casos não apresentam bons resultados. Neste trabalho será avaliado apenas o operador de um ponto de crossover (figura 3.2), pois como visto em (Timóteo, 2002) este apresentou o melhor resultado entre os operadores avaliados, além de ser este o operador de mais fácil implementação para o Timetabling.

3.1.7 Mutação

O operador de mutação simula a mutação natural que ocorre nos genes de um indivíduo na natureza. Sua função é trazer aleatoriedade genética nos indivíduos da nova geração de soluções. Essa mudança nos genes ocorre segundo uma constante de probabilidade pré-definida no início do problema, e é testada para cada gene de cada cromossomo da população.

(Goldberg, 1989) cita os principais operadores de mutação:

- **Mutação aleatória:** cada gene a ser mutado recebe um alelo sorteado do alfabeto válido;
- **Mutação por troca:** 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.

3.2 Clonalg

Os sistemas imunológicos artificiais (SIA) são um novo ramo da teoria de sistemas inteligentes que surgiu a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de novas ferramentas computacionais.

(de Castro, 2001) cita as principais características do Sistema Imunológico (SI):

- **Unicidade:** cada animal possui seu próprio sistema imunológico, independente, com suas capacidades e vulnerabilidades particulares;
- **Reconhecimento de padrões internos e externos ao sistema:** as células e moléculas que não pertencem ao organismo são reconhecidas e eliminadas pelo SI;
- **Detecção de anomalia:** o SI pode detectar e reagir a agentes patogênicos (causadores de anomalias) a que o organismo nunca havia sido exposto anteriormente;
- **Detecção imperfeita (tolerância a ruídos):** um reconhecimento perfeito não é necessário para que o SI reaja contra um elemento causador de patologia (patógeno);
- **Diversidade:** existe uma quantidade limitada de células e moléculas no SI que são utilizadas para se obter o reconhecimento de um número praticamente infinito de elementos, incluindo aqueles sintetizados em

laboratório;

- **Aprendizado por reforço:** a cada encontro com o mesmo patógeno, o sistema imunológico melhora a qualidade de sua resposta; e
- **Memória:** os componentes do SI bem-sucedidos no reconhecimento e combate às patologias são armazenados para uma resposta futura mais intensa e efetiva.

Os principais meios de atuação do SI são os mecanismos inatos e adaptativos, sendo conhecidos como SI Inato e SI Adaptativo. Uma definição geral obtida de (Uchôa, 2009):

- **SI Inato:** funciona como uma barreira mecânica à penetração do patógeno, ou por atuação de células que efetuam o reconhecimento de padrões associados a diversos tipos de patógenos (PAMP). Esses padrões são pré-definidos, no sentido em que o sistema inato não tem condições de se modificar para reconhecer padrões não-conservados.
- **SI Adaptativo:** caracterizado pela ativação, multiplicação e diferenciação dos linfócitos. Estas células de defesa, em comparação com os fagócitos do SI Inato, são muito mais específicas, reconhecendo um número muito menor de padrões de patógenos. Essa alta especificidade é compensada pelo amplo repertório, uma vez que há uma grande variação nos receptores antigênicos entre as células dos linfócitos.

3.2.1 Princípio da Seleção Clonal

Quando um anticorpo possui alta afinidade de um determinado antígeno os linfócitos que o produzem se multiplicam com maior velocidade através de

clonagem. Durante esse processo os clones sofrem uma variação genética, com taxa inversamente proporcional à afinidade com o antígeno em questão.

Dentre as novas células geradas, aquelas de maior afinidade são selecionadas, e as demais, suprimidas. Este processo, de expansão clonal e seleção, é conhecido como princípio da seleção clonal (Burnet, 1959).

3.2.2 Hipermutação

Os anticorpos presentes em uma resposta secundária ao antígeno possuem, em média, maiores afinidades do que aqueles das respostas primárias. Este fenômeno é chamado maturação de afinidade, ou hipermutação somática. Essa maturação requer que o fenótipo da ligação ao antígeno apresente diferenças da célula original. Estas diferenças serão inseridas através de mutação nos genes dos cromossomos.

O algoritmo de hipermutação somática é semelhante ao algoritmo de mutação de outras soluções evolutivas, como o algoritmo genético. No entanto a probabilidade de ocorrer uma mutação é calculada dinamicamente seguindo um mecanismo de regulação da hipermutação. Esse mecanismo atua de forma que a mutação ocorra com uma taxa inversamente proporcional à afinidade do anticorpo com o antígeno. O objetivo deste processo é preservar os indivíduos com alta afinidade ao mesmo tempo em que tenta aumentar a afinidade daqueles indivíduos cuja afinidade é baixa. (de Castro, 2002).

3.2.3 Diversificação

(de Castro, 2002) cita ainda o processo de edição de receptores. A teoria do processo de edição de receptores diz que alguns dos linfócitos auto-reagentes, aqueles que identificam o próprio SI como antígeno, podem apagar seus

receptores de antígenos e gerar novos receptores, através de recombinação gênica. Esse processo não faz parte do modelo original de Seleção Clonal de Burnet, mas ajuda a explicar a diversidade da população de linfócitos ao final do processo de seleção, ajudando a evitar o problema de ótimos locais.

3.2.4 Pseudocódigo

Os dois principais “motores” do Clonalg são o princípio da seleção clonal e a maturação de afinidade. Além destes dois, um outro fator que garante a eficiência é a memória imunológica. É ela quem garante que outros antígenos desconhecidos sejam facilmente reconhecidos pelas células de defesa.

A memória imunológica representa uma população de linfócitos de longa duração que farão parte da resposta imunológica. No caso de aplicações de otimização, a memória imunológica não representa uma estrutura auxiliar, mas a própria população de linfócitos.

No caso de otimização, não há um vetor de antígenos para se apresentar aos linfócitos. O antígeno é representado pela função de avaliação de afinidade, de forma similar à função de aptidão do algoritmo genético, e os anticorpos são representados por parâmetros que codificam elementos do domínio do problema.

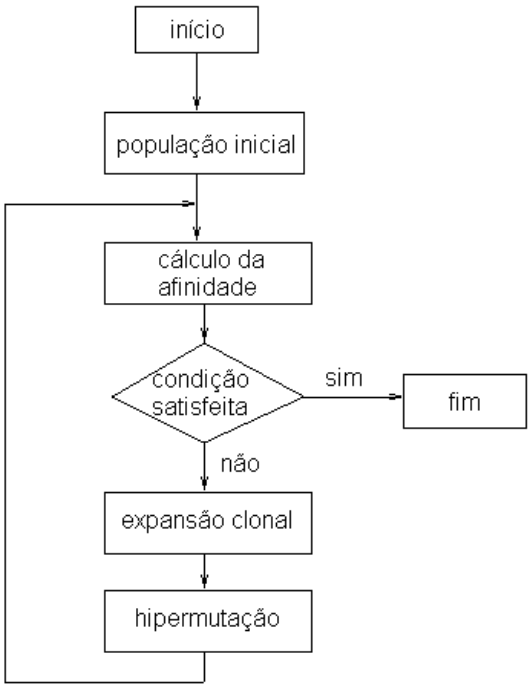

Figura 3.3: Fluxograma do Algoritmo de Seleção Clonal

O fluxograma do algoritmo de seleção clonal está descrito na Figura 3.3.

4. RESOLUÇÃO DO TIMETABLING

4.1 Representação da solução

Para modelar as soluções do problema foi utilizada a seguinte representação, utilizada no Algoritmo Genético:

- **Gene**: um *slot* de tempo, aula de 50 min.;
- **Cromossomo**: cada uma das turmas do colégio, modelado na forma de um vetor de *slots* de tempo;
- **Alelos**: respectivas disciplinas e número de aulas de cada turma;
- **Indivíduo**: um vetor contendo todas as turmas;

Além disso, um objeto do tipo indivíduo armazena o número de dias, o número de aulas por dia, e a lista de possíveis alelos da solução.

A mesma estrutura foi utilizada para representar, de forma análoga, uma solução do Clonalg, onde um indivíduo representa um antígeno do sistema imune.

4.2 Geração da população

Em seu trabalho, (Timóteo, 2002) apresenta dois operadores para se gerar uma população de soluções:

- **Geração aleatória**: para cada *slot* de cada turma, sorteia-se uma disciplina da lista de possíveis disciplinas e seu valor é atribuído àquele slot. No caso do número de aulas total das disciplinas ser menor que o número de *slots* da turma, será sorteado juntamente com a lista de disciplinas o número equivalente de disciplinas "vagas";
- **Geração heurística**: é feita de maneira similar à geração aleatória, mas de forma que sejam sorteados grupos de uma mesma disciplinas e atribuídos a blocos de *slots*;

Como não conhecemos o impacto desses dois operadores sobre o desempenho das duas meta-heurísticas comparadas, e o caso utilizado não dá preferência especial a aulas em bloco, optou-se por utilizar a geração de população aleatória.

4.3 Avaliação dos indivíduos

A avaliação da aptidão ou afinidade das soluções usa as seguintes restrições:

- **Colisão de professores**: número de ocorrências de uma disciplina em *slots* equivalentes de diferentes turmas;
- **Horários vagos esparsos**: número de ocorrências de aulas vagas em *slot* que não seja o último do dia;
- **Blocos de disciplinas**: ausência de blocos de disciplinas em três *slots* consecutivos;
- **Tamanho máximo dos blocos**: número de ocorrências de blocos maiores que dois *slots*;
- **Preferências dos professores**: diferença entre o número de aulas em horário preferido pelo professor e número de aulas em horário rejeitado pelo professor. Valor maior ou igual a zero.

Calcula-se o somatório das penalidades, ponderado pelos pesos definido para cada penalidade. Estes pesos foram decididos à priori e indicam aproximadamente a importância dada pela escola para selecionada penalidade.

Após calculado o somatório das penalidades de uma solução, a avaliação dessa solução será dada pela equação:

$$Avaliação = \frac{1}{1 + penalidades}$$

Através dessa equação podemos ter uma noção da qualidade da solução, ainda que fora de escala. A imagem da função de avaliação varia entre zero (*penalidades* tendendo ao infinito) e um (*penalidades* tendendo a zero).

4.4 Seleção dos indivíduos

Em seu trabalho, (Timóteo, 2002) testou os operadores de seleção por torneio, roleta giratória e roleta giratória com redução, este último operador proposto pelo próprio. Apesar de sugerir que outros parâmetros fossem testados, concluiu em seus testes que o operador melhor sucedido foi o de torneio.

Seguindo estes resultados, optou-se pela implementação do operador de seleção por torneio, e a seleção dos pares foi feita de forma aleatória.

4.5 Cruzamento

Para realizar o cruzamento das soluções foi escolhido o operador com um ponto de corte, avaliado por (Timóteo, 2002) como o operador de cruzamento com o melhor resultado em seus testes.

O cruzamento é realizado de forma independente para cada cromossomo do par de soluções. É importante observar que o operador não pode ser aplicado da forma usual como para cadeias binárias. A simples troca de seqüências genéticas entre os cromossomos poderia gerar soluções incoerentes, onde os cromossomos possuam mais gene do que o permitido para determinado alelo.

De forma a respeitar os requisitos do Timetabling, o cruzamento com um ponto de corte foi implementado tomando como base o cruzamento descrito por (Timóteo, 2002).

A idéia geral desta implementação é trocar a posição apenas dos genes que estejam presentes simultaneamente na seqüência de genes nos dois cromossomos:

Passo um: após decidido o ponto de corte, deve ser sorteado se o cruzamento será realizado antes ou após o corte. Esta medida se torna necessária pelo fato da implementação ter tornado o operador assimétrico em relação ao ponto de corte.

Passo dois: são geradas duas listas auxiliares. Cada nó das listas é composto pelos seguintes itens:

- Disc: representa um gene da seqüência de cruzamento;
- Pgene: posição do gene no cromossomo;
- Pcross: indica a posição de crossover;
- Disponível: indica se o gene está disponível para cruzamento.

ListaDeGenesA fará referência à seqüencia genética de *CromossomoA* e a *ListaDeGenesB* fará referência à seqüencia genética de *CromossomoB.*

Inicialmente cada uma das listas tem seus nós instanciados com o gene e a posição desse gene no cromossomo, o campo Disponível como verdadeiro e o campo Pcross como -1.

Passo três: para cada nó de *ListaDeGenesA* busca-se um nó de *listaDeGenesB* que faça referência ao mesmo gene de *ListaDeGenesA,* e que esteja disponível. Se o nó é encontrado mudamos o campo Disponível de cada nó para falso. O campo Pcross de cada nó recebe a posição do gene no cromossomo do outro nó.

Passo quatro: os nós de *ListaDeGenesA* são percorridos. Se o campo Pcross do nó for diferente de -1, o gene desse nó será inserido na posição Pcross no *CromossomoA.* O mesmo processo é feito com *ListaDeGenesB e*

CromossomoB.

Ao final do último passo descrito, dá-se por terminado o cruzamento para um par de cromossomos.

4.6 Mutação

O operador de inversão não pode ser utilizado para a representação de cromossomo escolhida nesta implementação. Já o operador de mutação aleatória poderia gerar cromossomos incoerentes com o problema. Por isso foi implementado neste trabalho o operador de mutação por troca.

O operador foi implementado de forma que sejam escolhidas aleatoriamente duas posições contendo genes diferentes do cromossomo, garantindo assim que toda mutação acarrete num cromossomo com carga genética diferente.

4.7 Expansão Clonal

O operador de Expansão Clonal foi implementado como o descrito por (de Castro, 2001). O total de N_c clones gerados será dado pela equação:

$$N_c = \sum_{i=1}^{n} round(\beta \times N)$$

Nesta equação β é um fator multiplicativo, passado como parâmetro do Clonalg, *N* é quantidade de anticorpos da população, *n* é o número de anticorpos selecionados para clonagem, e *i* é a posição do anticorpo numa lista de anticorpos em ordem crescente de afinidade.

Segundo (de Castro, 2001) nos casos de otimização, a reprodução proporcional à afinidade não necessariamente se aplica, pois ela poderia causar a convergência da população para um ótimo local.

Por questões de comparação, o operador de expansão clonal elitista

também foi implementado:

$$N_c = \sum_{i=1}^{n} round(\beta \times \frac{N}{N - i})$$

4.8 Hipermutação

O processo de hipermutação utiliza como base o operador de mutação descrito na seção 4.6, com a diferença que a taxa de probabilidade de mutação é calculada dinamicamente para cada antígeno da população.

Para o cálculo da taxa de probabilidade foi usada a equação descrita em (de Castro, 2001):

$$\alpha = \exp(-\rho D^*)$$

Onde α é a taxa de probabilidade da mutação, ρ é uma constante multiplicativa passada como parâmetro do método e D^* é o valor normalizado da afinidade, dado por $D^* = D + D_{max}$.

4.9 Diversificação

Ao final de cada interação do Clonalg, a fim de garantir a diversificação da população e uma cobertura maior do espaço de soluções são gerados *d* novos anticorpos. Este número *d* é passado como parâmetro do Clonalg no início do processo.

4.10 Critério de Parada

Como critério de parada, tanto do Algoritmo Genético, quanto do Clonalg foi adotado neste trabalho o número de gerações. Além deste poderia ser

usado como critério o tempo de execução dos algoritmos, mas ao invés disso preferiu-se utilizar o tempo como um dos critérios de eficiência para se comparar os dois algoritmos.

5. A APLICAÇÃO DESENVOLVIDA

5.1 Introdução

Após elaborar os passos necessários para desenvolver um Algoritmo Genético e o Clonalg, tornou-se necessário criar uma aplicação para que os testes pudessem ser executados.

A aplicação foi desenvolvida usando a linguagem C# e o ambiente de desenvolvimento Microsoft Visual Studio 2008.

Toda entrada e saída de dados é feita através de arquivos de texto. E embora uma das saídas da aplicação seja a melhor grade horária de cada algoritmo, o foco principal foi dado aos testes dos parâmetros.

5.2 Dados de entrada

Basicamente o usuário precisa fornecer um arquivo de configuração com os seguintes dados:

- O número de turmas;
- O número de dias de aula;
- O número de aulas de cada dia;
- As disciplinas de cada turma;
- As preferências de horário de cada disciplina.

A figura 5.1 mostra um exemplo de arquivo de entrada de dados da escola:

```
2 5 5

Turma 0
M 4
P 4
D 1
R 2
C 4
G 3
H 3
I 2
V 2
fimturma

Turma 1
M 4
P 4
D 1
R 2
C 4
G 3
H 3
I 2
V 2
fimturma

Preferencias
P   3,2,-1   3,5,-1   2,2,-1
C    3,3,-1   4,3,-1   2,3,1
fimpreferencias
```

Figura 5.1: exemplo de arquivo com dados de uma escola

5.3 Parâmetros de configuração

Para realizar os testes deve ser informado um arquivo de configuração

onde estão listados todos os valores testados para cada um dos parâmetros.

Parâmetros Gerais:

- Número de Gerações;
- Tamanho da População do AG;
- Tamanho da População do Clonalg.

Parâmetros do Algoritmo Genético:

- Taxa da probabilidade de mutação;
- Taxa de probabilidade de cruzamento;
- Número de filhos do cruzamento.

Parâmetros do Clonalg:

- Fator de mutação;
- Fator Clonal;
- Fator de diversificação da população.
- Elitismo - se a clonagem será inversamente proporcional à afinidade do anticorpo ou será igual para toda a população.

Um exemplo de arquivo de parâmetro de configuração está mostrado na figura 5.2:

```
parametros_gerais
populacao 180
geracoes_ag 600
geracoes_clonalg 600

parametros_ag
mutacao 0,03
cruzamento 0,7
filhos 0,25

parametros_clonalg
fator_mutacao 2,2
clones 6 8
diversificacao 0,10 0,15
elitismo True
```

Figura 5.2: exemplo de arquivo de entrada do sistema

Após inseridos os arquivos de configuração serão gerados laços de repetição para testar cada uma das variações dos parâmetros, de forma automatizada.

5.4 Dados de saída

Para cada uma das heurísticas será gerado um arquivo contendo os dados estatísticos dos *n* testes realizados:

- Número do teste;
- Parâmetros utilizados;
- Aptidão ou afinidade do melhor indivíduo da geração;
- Aptidão ou afinidade média da geração
- Desvio padrão de cada geração.

Além desses arquivos será gerado também um arquivo contendo a grade da melhor solução encontrada por cada heurística, se a solução resolver todas as restrições rígidas.

6. RESULTADOS E DISCUSSÃO

Para a realização dos testes foram usados dados baseados em um problema real. Os dados utilizados foram os do ensino fundamental da Escola Estadual Pastor Paulo Nobre Nascimento, situada no município de Nanuque-MG.

6.1 Dados da Escola

Os dados de entrada adotados nos testes foram:

- Cadastro de sete turmas;
- Cadastro de nove disciplinas;
- Foram disponibilizadas 25 horários semanais, sendo cinco horários em cada um dos cinco dias da semana.

Os dados como recebidos da diretoria da escola não continham preferências de horários dos professores. Para evitar que a avaliação de preferência fosse descartada, foram geradas preferência de forma sintética, para os testes, segundo a tabela 6.1:

Tabela 6.1: Importância dos critérios de avaliação

Critério	Importância
Aula em Bloco	0.005
Choque de professor	0.5
Aulas vagas esparsas	0.05
Preferências dos Professores	0.2

Como mostrado na tabela, foi dada pouca importância ao critério de

aulas em bloco, visto que a metodologia da escola não dá preferência a este tipo de organização. Além disso, a forma de avaliação das penalidades de aula em bloco levou a uma disparidade entre a qualidade da grade horária e a aptidão da solução, levando a valores não condizentes com a realidade.

As preferências dos professores foram geradas de forma sintética e aleatória, assim foram atribuídas a quatro professores arbitrários de três a quatro horários preferenciais aleatórios.

6.2 Testes

Para a execução dos algoritmos foi utilizado um computador Core 2 Duo® T6500 @ 2.1GHz, 3GB de memória RAM DDR2 @ 800MHz.

Os testes foram realizados de forma gradual, com problemas de menor tamanho, para calibragem dos parâmetros do Clonalg. Em seguida os resultados do Clonalg foram comparados com os resultados gerados pelo Algoritmo Genético, configurado com os parâmetros que geraram os melhores resultados descritos por (Timóteo, 2002). Todos os testes foram realizados usando 400 gerações.

6.3 Dados Sintéticos

Os testes foram iniciados com os dados sintéticos configurados da seguinte forma:

- Cinco turmas;
- 25 horários, onde são cinco aulas por dia em cinco dias da semana;
- Oito disciplinas para as cinco turmas;
- Duas aulas vagas;

- Dois professores com três horários preferenciais cada.

Os resultados do primeiro teste dos parâmetros do Clonag são mostrados na Tabela 6.2.

Tabela 6.2: Resultados do primeiro teste do Clonalg

Param.	Pop.	ρ	β	Divers.	Elit.	Tempo	Máximo	Média	Desvio
Teste_0	20	1.5	0.6	0.1	True	00:08	.784	0.736	0.164
Teste_1	20	1.5	0.6	0.1	False	00:35	.820	0.782	0.170
Teste_2	20	1.5	1	0.1	True	00:12	.781	0.745	0.164
Teste_3	20	1.5	1	0.1	False	00:56	.784	0.749	0.158
Teste_4	20	2	0.6	0.1	True	00:08	.766	0.731	0.159
Teste_5	20	2	0.6	0.1	False	00:34	.803	0.765	0.169
Teste_6	20	2	1	0.1	True	00:12	.781	0.745	0.160
Teste_7	20	2	1	0.1	False	01:00	.781	0.745	0.163
Teste_8	40	1.5	0.6	0.1	True	00:21	.766	0.749	0.111
Teste_9	40	1.5	0.6	0.1	False	02:27	.794	0.775	0.116
Teste_10	40	1.5	1	0.1	True	00:32	.797	0.778	0.116
Teste_11	40	1.5	1	0.1	False	03:50	.787	0.769	0.117
Teste_12	40	2	0.6	0.1	True	00:20	.787	0.769	0.118
Teste_13	40	2	0.6	0.1	False	02:16	.763	0.746	0.112
Teste_14	40	2	1	0.1	True	00:02	.769	0.751	0.113
Teste_15	40	2	1	0.1	False	03:42	.810	0.791	0.118

Na Tabela 6.2 o parâmetros ρ representa o fator de hipermutação, onde valores maiores apresentam taxas de mutação menores para cada indivíduo da população. O parâmetro β é o fator de clonagem, e determina o número máximo de clones para cada indivíduo.

Ainda na Tabela 6.2, na coluna *Parâmetros*, os *Testes* de 0 a 15 referem aos Testes dos parâmetros de configuração do Clonalg. Neste trabalho *teste* será usado como referência à tentativa de resolução de uma determinada instância do Timetabling. As três últimas colunas, *Máximo*, *Média* e *Desvio* referem-se aos valores da aptidão da população na última geração do teste.

Como mostrado na tabela, para todos os parâmetros, os resultados foram muito próximos. Detalhe para as configuração onde não há uso de *elitismo*, configurações estas que apresentaram resultados superiores às configurações em que foi usado *elitismo*, porém com variações insignificantes, e tendo como desvantagem o tempo de execução, que chegou a ser quase 100 vezes maior que o tempo de execução das configurações elitistas.

Tabela 6.3: Resultados do primeiro teste do AG

Parametros	Pop.	Mut.	Cruzament.	Filhos	Tempo	Máximo	Média	Desvio
Teste_0	120	0.03	0.7	0.25	00:19	0.781	0.741	0.092

Em comparação com os resultados do Clonalg, pode-se ver que ambos obtiveram resultados muito próximos, observando apenas que o Desvio Padrão do AG foi menor que o Desvio Padrão nos resultados do Clonalg.

O Desvio Padrão é uma das métricas usadas para se medir a diversidade das soluções de cada geração. Um Desvio Padrão muito baixo indica que o algoritmo está convergindo para um ótimo local no espaço de soluções.

Para um melhor entendimento dos resultados estão os gráficos com os dados históricos de duas das melhores configurações do Clonalg e a configuração do AG para este teste.

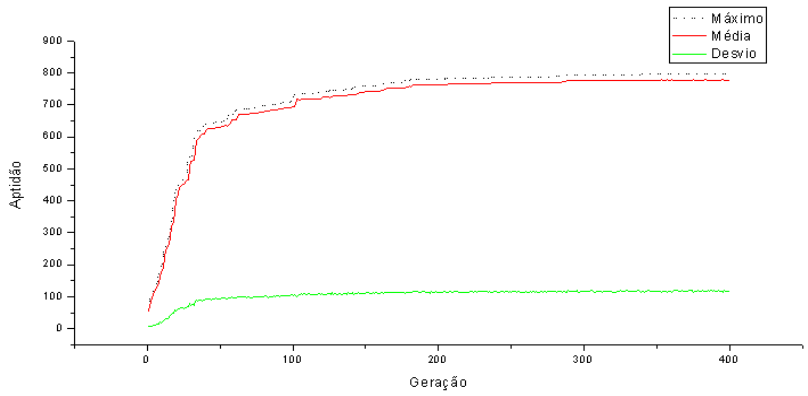

Figura 6.1: Geração X Aptidão da Configuração 10 do primeiro teste do Clonalg

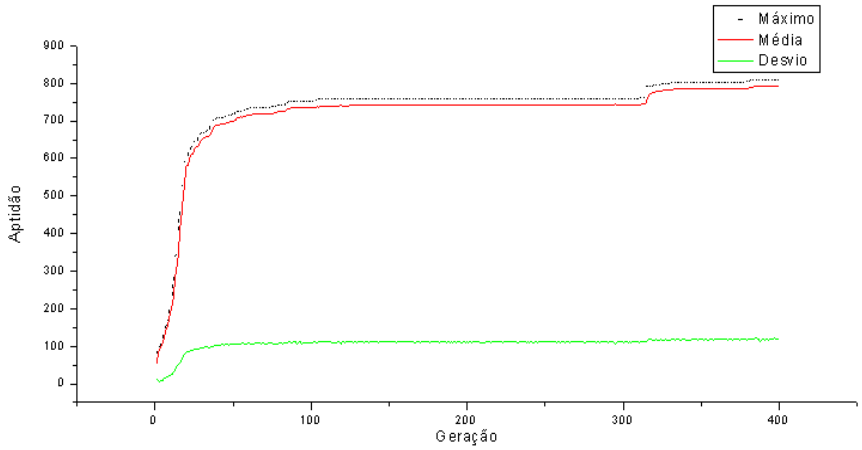

Figura 6.2: Geração X Aptidão da Configuração 15 do primeiro teste do Clonalg

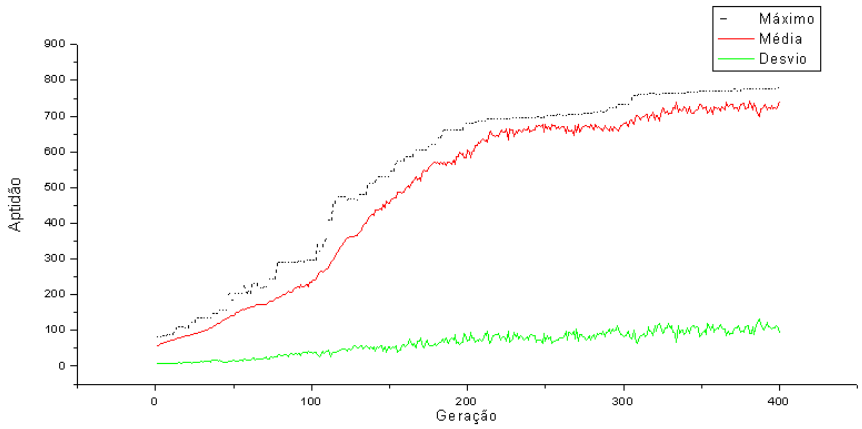

Figura 6.3: Geração X Aptidão do primeiro teste do AG

Uma primeira análise dos gráficos das Figuras 6.1 e 6.2 leva a pensar se a população do Clonalg não estaria sofrendo uma convergência muito rápida para um ótimo local, em comparação com a população do AG. Provavelmente os valores muito altos de clonagem levaram a esta convergência prematura. Por esse motivo, no seguinte teste foram usados valores mais baixos para esses parâmetros.

6.4 Teste de um caso real de Timetabling

Neste segundo teste foram usados os dados da Escola Pastor Paulo, configurados da seguinte forma:

- Sete turmas;
- Quatro turmas com oito disciplinas;
- Três turmas com nove disciplinas;
- Nenhuma aula vaga;
- Nenhuma preferência de professor.

Como comentado no teste anterior, os parâmetros de clonagem muito altos podem ter levado a uma convergência prematura da população para um ótimo local. Por esse motivo, neste teste foram usados valores mais baixos para estes parâmetros.

Para compensar a redução nos valores do parâmetro de clonagem, foram feitos testes com valores reduzidos do parâmetro de diversidade, para garantir que uma perturbação excessiva não fosse gerada em cada nova geração da população, e foi descartado o elitismo, para aumentar ao máximo possível o espaço de busca das soluções.

Tabela 6.4: Resultados do segundo teste do Clonalg.

Clonalg	POP.	ρ	β	Divers.	Elitis.	Tempo	Máximo	Média	Desvio
Teste_0	120	2	0,01	0,1	False	04:29	0,117	0,091	0,01
Teste_1	120	2	0,01	0,05	False	04:34	0,11	0,088	0,009
Teste_2	120	2	0,05	0,1	False	03:16	0,169	0,167	0,013
Teste_3	120	2	0,05	0,05	False	03:16	0,169	0,168	0,013
Teste_4	120	2	0,1	0,1	False	05:53	0,169	0,168	0,013
Teste_5	120	2	0,1	0,05	False	05:56	0,171	0,17	0,013
Teste_6	120	1	0,01	0,1	False	06:06	0,116	0,084	0,011
Teste_7	120	1	0,01	0,05	False	06:20	0,13	0,088	0,015
Teste_8	120	1	0,05	0,1	False	05:54	0,156	0,138	0,013
Teste_9	120	1	0,05	0,05	False	05:47	0,156	0,136	0,014
Teste_10	120	1	0,1	0,1	False	08:12	0,164	0,156	0,013
Teste_11	120	1	0,1	0,05	False	07:45	0,163	0,154	0,012

Os resultados dessas configurações estão na Tabela 6.4. Como visto nessa tabela, nenhuma configuração foi eficaz na resolução dessa instância do Timetabling. Além disso, nenhuma das configurações foi capaz de resolver todas as restrições rígidas do problema.

Em comparação temos o resultado do teste do AG na Tabela 6.5.

Tabela 6.5: Resultados do segundo teste do AG.

AG	Pop.	Mut.	Cruzament.	Filhos	Tempo	Máximo	Média	Desvio
Teste_0	160	0.03	0.7	0.25	00:19	0.332	0.270.	0.025

Se comparado com os resultados do primeiro teste, o AG também não teve um resultado muito animador. No entanto, o AG conseguiu resolver todas as restrições rígidas, e num tempo muito pequeno, se comparado com o Clonalg.

Para tentar resolver essa instância do Timetabling optou-se por retornar às configurações que geraram as melhores soluções no primeiro teste. O resultado pode ser visto na Tabela 6.6:

Tabela 6.6: Resultados do segundo teste do Clonalg usando melhores parâmetros do primeiro teste

Clonalg	POP.	ρ	β	Diversidade	Elitis.	Tempo	Máximo	Média	Desvio
Teste_12	50	1	0.6	0.1	True	00:44	0.138	0.123	0.016
Teste_13	50	1	1	0.1	True	01:07	0.166	0.146	0.019
Teste_15	50	2	0.6	0.1	True	00:43	0.168	0.164	0.020
Teste_16	50	2	1	0.1	True	01:06	0.168	0.165	0.019

Os resultados descritos na Tabela 6.6 não foram sobremaneira diferentes dos resultados na Tabela 6.4. Da mesma forma, as configurações utilizadas no Clonalg não se mostraram capazes de resolver esta instância do problema.

O gráfico da Figura 6.5, mostra o comportamento do Clonalg utilizando a melhor configuração neste teste. A população convergiu para um máximo local, não conseguindo então, evoluir além deste ponto.

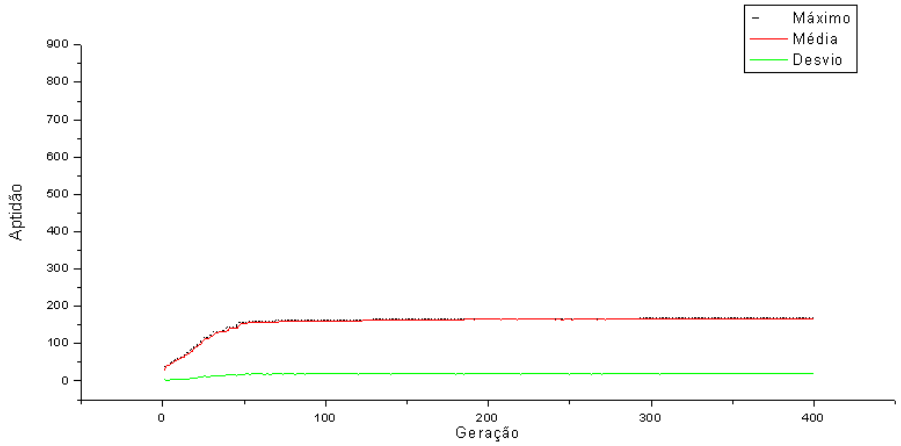

Figura 6.4: Geração X Aptidão da Configuração 16, no segundo teste do Clonalg

Em contrapartida, uma analisa do resultado do AG (Figura 6.5) mostra a população em evolução constante, indicando que resultados ainda melhores poderiam ser encontrados usando mais gerações.

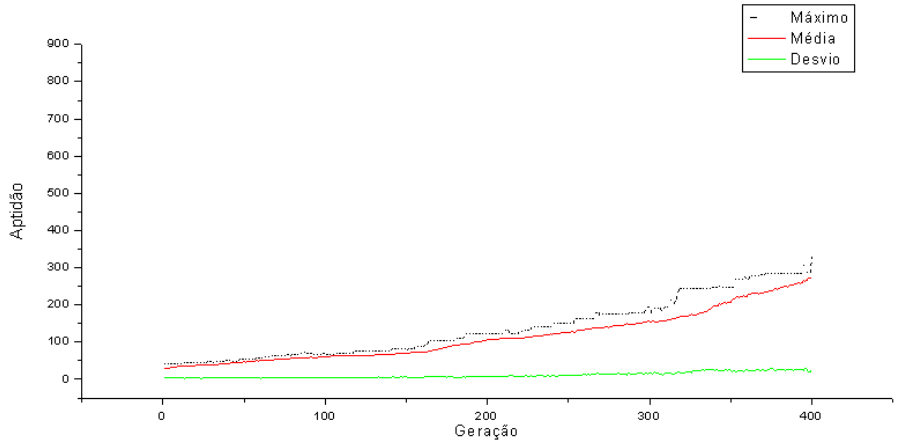

Figura 6.5: Geração X Aptidão do primeiro teste do AG

6.5 Dados baseados no caso real

Neste último teste, foram usados os dados do caso real da Escola Pastor Paulo. As configurações são as mesmas do teste anterior, porém simulando um problema com apenas cinco turmas, e dois professores tendo preferências de horários. Além desta configuração, foi testada outra com seis turmas, sem preferências de professores, mas obtendo um resultado similar ao desta configuração de teste, optou-se por não incluí-la neste trabalho.

Os resultados do Clonalg são mostrados a seguir (Tabela 6.6):

Tabela 6.7: Resultados do terceiro teste do Clonalg

Clonalg	POP.	ρ	β	Diversidade	Elitis.	Tempo	Máximo	Média	Desvio
Teste_0	35	2	0.6	0.1	True	00:21	0.295	0.288	0.041
Teste_1	35	2	0.6	0.1	False	02:12	0.299	0.291	0.042
Teste_2	35	2	1	0.1	True	00:32	0.296	0.289	0.042
Teste_3	35	2	1	0.1	False	03:35	0.297	0.290	0.039

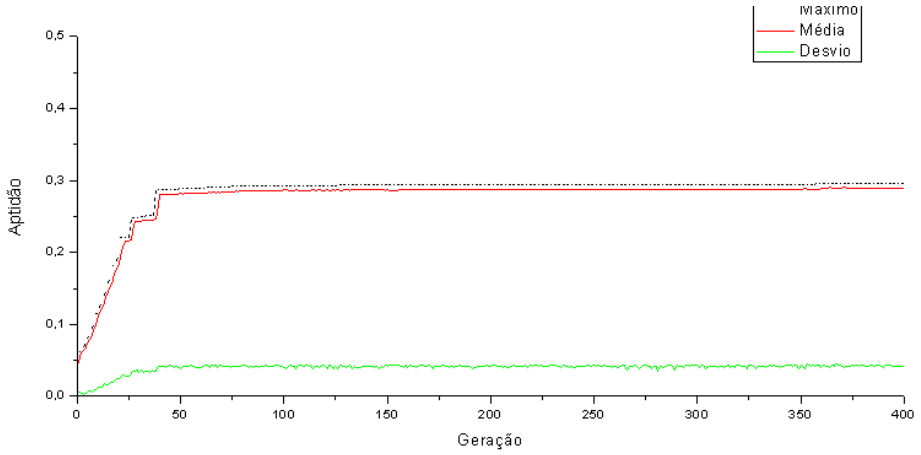

Figura 6.6: Geração X Aptidão da Configuração 3, no terceiro teste do Clonalg

Observando o gráfico da Figura 6.6 é possível observar que embora a população tenha conseguido um bom resultado na solução, ela parou de convergir antes da geração 50, não havendo alterações observáveis na população após este período.

Em comparação, temos os dados do teste usando o Algoritmo Genético(Tabela 6.8):

Tabela 6.8: Resultado do teste terceiro teste do AG

AG	Pop.	Mut.	Cruzament.	Filhos	Tempo	Máximo	Média	Desvio
Teste_0	120	0.03	0.7	0.25	00:23	0.883	0.770.	0.100

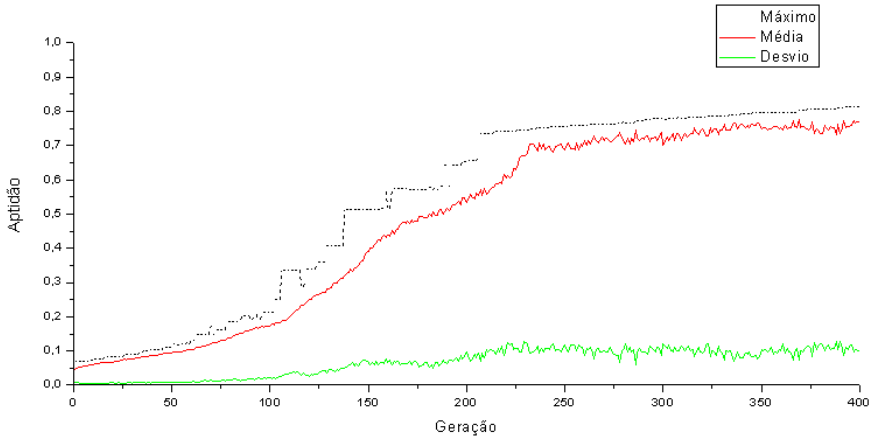

Figura 6.7: Geração X Aptidão da Configuração 3, no terceiro teste do AG

Com base no gráfico da Figura 6.7, é possível perceber que o Algoritmo Genético conseguiu uma boa diversidade de soluções, ao passo que ao fim da geração 400 a população continua em evolução.

6.6 Discussão

Os testes foram realizados em apenas três instâncias do problema de Timetabling, uma sintética, e duas baseadas em caso real. O ideal seria testar várias instâncias e realizar comparações entre elas, aumentando assim a confiança nas configurações de parâmetros testados.

Em duas das instâncias testadas, o Algoritmo Genético atingiu um resultado em muito superior ao atingido pelo Clonalg. Aparentemente a implementação utilizada do Clonalg não conseguiu lidar bem com instâncias do School Timetabling com mais de cinco turmas, ou talvez a representação do problema não tenha sido a mais conveniente para utilização exclusiva com o operador de hipermutação.

7. CONCLUSÃO

Foi realizado um exame dos principais conceitos relativos ao problema do Timetabling, sua definição formal e complexidade.

Foram examinados dois métodos Bioinspirados para a resolução do Timetabling, os Algoritmos Genéticos e o Clonalg, onde uma aplicação foi implementada para comparar a geração de soluções de ambos.

Nos testes, para as configurações de parâmetros utilizadas, o Clonalg não se mostrou capaz de gerar boas soluções para instâncias do problema de School Timetabling com mais de cinco turmas, ao passo que a implementação dos Algoritmos Genéticos conseguiu gerar boas soluções para todas as instâncias testadas.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Burke, E. K., Kingston, J., De Werra. D., **Handbook of Graph Theory**, Chapman Hall/CRC Press, 2004.

Burnet, M., **The Clonal Theory Of Acquired Immunity**, University Press – Cambridge, 1959.

de Castro. L. N., **Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais**, Tese de Doutorado, Unicamp, 2001.

de Werra. D., **An introduction to timetabling**, European Journal of Operationl Research, 1985.

Goldberg, D.E., **Genetic Algorithms in Search, Optimization And Machine Learning**, The University of Alabama, 1989.

HOLLAND, J. H. **Outline for a logical theory of adaptative systems**, J. Assoc. Comput. Mach.,vol.3, 1962

Lima, E. O., **Algoritmo Genético Híbrido aplicado à otimização de funções**, Trabalho de Conclusão de Curso, DCC/UFLA, 2008.

Linden, R., **Algoritmos Genéticos**, Brasport, 2006.

Qu, R., Burke, E. K.,McCollum, B. L., Merlot, T. G., Lee. S. Y., **A survey of search methodologies and automated system development for examination timetabling**, Springer Science + Business Media, 2008

Schaerf. A., **A suvery of automated timetabling**, *CW,* 1995.

Timóteo. G. T. S., **Desenvolvimento de um Algoritmo Genético para a**

Resolução do Timetabling, Trabalho de Conclusão de Curso, DCC/UFLA, 2002.

Uchôa, J. Q., **Algoritmos Imunoinspirados Aplicados em Segurança Computacional: Utilização de Algoritmos Inspirados no Sistema Imune para Detecção de Intrusos em Redes de Computadores**, Teste de Dourado, UFMG, 2009.