UNIVERSIDADE CATÓLICA DE PERNAMBUCO

CENTRO DE CIÊNCIAS E TECNOLOGIA

TRABALHO DE CONCLUSÃO DE CURSO

APLICAÇÕES DE ALGORITMOS GENÉTICOS EM OTIMIZAÇÃO DE HORÁRIOS

por

PHILIPPE NORBERT CAVALCANTI AUSSOURD

Recife, Junho de 2015

UNIVERSIDADE CATÓLICA DE PERNAMBUCO

CENTRO DE CIÊNCIAS E TECNOLOGIA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

APLICAÇÕES DE ALGORITMOS GENETICOS EM OTIMIZAÇÃO DE HORÁRIOS

por

PHILIPPE NORBERT CAVALCANTI AUSSOURD

Monografia apresentada ao curso de Ciência da Computação da Universidade Católica de Pernambuco, como parte dos requisitos necessários à obtenção do grau de Bacharel em Ciência da Computação.

ORIENTADOR: CLARISSA DAISY DA COSTA ALBUQUERQUE, Prof.ª Dr.ª

Recife, Junho de 2015.

UNIVERSIDADE CATÓLICA DE PERNAMBUCO

CENTRO DE CIÊNCIAS E TECNOLOGIA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

No dia 09 de Junho de 2015, às 14:00 horas, reuniu-se para deliberar a defesa de Monografia de Trabalho de Conclusão de Curso de Ciência da Computação, do aluno Philippe Norbert Cavalcanti Aussourd, orientado pela professora Clarissa Daisy da Costa Albuquerque, sob título Aplicações de Algoritmos Genéticos em Otimização de Horários, a banca composta pelos professores:

1. Clarissa Daisy da Costa Albuquerque
2. Francisco Madeiro Bernardino Júnior

Após a apresentação da monografia esta foi julgada e **APROVADA**, sendo-lhe atribuída nota _____ .

Recife, 15 de Junho de 2015.

_____

Prof. Msc. Márcio Augusto Silva Bueno
Professor da disciplina de Trabalho de Conclusão de Curso

_____

Prof.ª Dr.ª Clarissa Daisy da Costa Albuquerque
Professor Orientador

_____

Prof. Dr. Francisco Madeiro Bernardino Júnior
Professor Convidado

Dedico esse trabalho as pessoas que me ajudaram como nunca nessa jornada que passou. A meus pais, Christiane Cavalcanti Aussourd e Lionel Robert Marcel Aussourd, à minha irmã Catherine Marie Cavalcanti Aussourd, à minha terapeuta Kátia Sylvana Cruz Monteiro e ao Dr. Leonardo Machado pela força e confiança depositadas nessa fase da minha vida, e a todos os amigos e familiares que me alegraram nos momentos onde mais precisei.

AGRADECIMENTOS

Agradeço principalmente a minha terapeuta e meu médico pelo apoio incondicional.

Agradeço a Universidade Católica de Pernambuco pela oferta do conhecimento científico por meio dos docentes do curso de Ciência da Computação.

Agradeço também a funcionária do Centro de Ciências e Tecnologia da Universidade Católica de Pernambuco, Maria Helena Freire Lopes pela sua contribuição ao trabalho.

A meu amigo e funcionário do Centro de Ciências e Tecnologia da Universidade Católica de Pernambuco, Gilberto Pereira Campos pela companhia, ajuda emocional e também pelos serviços prestados a mim.

Aos meus amigos da turma de Ciência da Computação de 2008.1, pelo compartilhamento de ideias, dificuldades, e também alegrias nesses longos anos de convivência.

A minha orientadora e professora Clarissa Daisy, por mais uma oportunidade de traçar caminho nessa fase da vida acadêmica e abrir portas para oportunidades que virão no futuro e pela supervisão e orientação neste trabalho acadêmico onde recebi toda a ajuda necessária para a conclusão desse trabalho.

E finalmente, em especial a todos os membros da minha família, que estavam ao meu lado desde o início, dando suporte para tudo que precisei até esse momento onde minhas decisões precisavam ser guiadas por mentores para chegar até o momento atual, com muito amor e compreensão.

“Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim. ” (Chico Xavier).

Resumo da Monografia apresentada ao curso de Ciência da Computação da Universidade Católica de Pernambuco.

APLICAÇÕES DE ALGORITMOS GENETICOS EM OTIMIZAÇÃO DE HORÁRIOS

Philippe Norbert Cavalcanti Aussourd

06/2015

Orientador: Clarissa Daisy da Costa Albuquerque, Doutora.
Área de Concentração: Computação Bioinspirada.
Palavras-chave: **Otimização de Horários, Algoritmos Genéticos, Inteligência Artificial.**
Número de Páginas: 94.

A produção de grades horárias (*Timetabling*) escolares e acadêmicas é um problema complexo de alocação de um conjunto de eventos (exames, seminários, projetos ou aulas) e um conjunto de agentes (instrutores, monitores ou professores) em um conjunto finito de horários, sujeito a um amplo conjunto de restrições. O problema deve ser resolvido de forma a maximizar a possibilidade de alocação ou minimizar as violações de restrições. Em muitas instituições de ensino, este trabalho é realizado manualmente, necessitando de vários profissionais, gastando um longo tempo e nem sempre produzindo a melhor alocação de recursos para a instituição. Este trabalho aborda teoria e aplicações de Algoritmos Genéticos (AGs) na resolução de Problemas de *Timetabling* em instituições de ensino fundamental, médio e universitário. Análises realizadas neste trabalho - de diversos artigos, dissertações e monografias publicados na literatura - mostram que os AGs utilizados na resolução do problema em questão, quando testados em diferentes situações, forneceram resultados satisfatórios - tanto referentes a qualidade das soluções obtidas, quanto ao tempo gasto para produzi-las - constituindo-se, portanto em uma ferramenta adequada para resolver problemas de *Timetabling* escolar e acadêmico.

Abstract of Dissertation presented to Catholic University of Pernambuco.

TIMETABLING APPLICATIONS IN GENETIC ALGORITHMS

Philippe Norbert Cavalcanti Aussourd

06/2015

Supervisor: Clarissa Daisy da Costa Albuquerque, PhD.
Area of Concentration: Bioinspired Computation.
Keywords: **Timetabling, Genetic Algorithim, Artificial Intelligence.**
Number of Pages: 94.

Production of school and academic schedules (Timetabling) is a complex problem of allocating a set of events (exams, seminars, projects, or classes) and a set of agents (trainers, instructors and teachers) in a finite number of times subject to a wide range of restrictions. The problem to be solved, aims maximize the possibilities of allocation or minimize constraints violations. In many educational institutions, this work is done manually, requiring several professionals, spending a long time and do not always produce the best allocation of resources for the institution. This work deals with theory and applications of Genetic Algorithms (GAs) in timetabling problem resolution in primary education institutions, secondary and university. Analysis performed in this study - of several articles, dissertations and monographs published in the literature - show that GAs used in solving the problem at hand, when tested in different situations, provided satisfactory results - both regarding the quality of the solutions, as the time spent to produce them - thus constituting themselves in an appropriate tool for solving educational and academic timetabling problems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS / SIGLAS

Termo		Descrição
GRASP	*Greedy Randomized Adaptive Search Procedure*	*Procedimento de Busca Guloso, Aleatório e Adaptativo*
C3	-	*Curso de Ciência da Computação*
UFLA	-	*Universidade Federal de Lavras*
UFPR	-	*Universidade Federal do Paraná*
UFS	-	*Universidade Federal de Sergipe*
UFU	-	*Universidade Federal de Uberlândia*
FURB	-	*Universidade Regional de Blumenau*
AG	-	*Algoritmo Genético*
UNICAP	-	*Universidade Católica de Pernambuco*
PMX	*Partially Matched crossover*	*Cruzamento parcialmente casado*
EEPRA	-	*Escola Estadual Padre Rogério Abdala*
EMAB	-	*Escola Municipal Álvaro Botelho*
FEELT	-	*Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia*
AloGra	-	*Alocador Automático de Grade Horária*
DCOMP	-	*Departamento de Computação da Universidade Federal de Sergipe*
1PX	*One-point crossover*	*Cruzamento em um ponto*
2PX	*Two-point crossover*	*Cruzamento em dois pontos*
MPX	*Multiple-point crossover*	*Cruzamento em múltiplos pontos*
SX	*Segmented crossover*	*Cruzamento segmentado*
FER	*Faculty of Electrical Engineering and Computing in Zagreb*	*Faculdade de Engenharia Elétrica e Computação de Zagreb*
STL	*Standard Template Library*	*Biblioteca Padrão de Gabaritos*
HCS	*Hard Constraint Satisfaction*	*Satisfação da Restrição Rígida*
SCS	*Soft Constraint Satisfaction*	*Satisfação da Restrição Flexível*
SCM	*Sequential Construction Method*	*Método para Construção Sequencial*

SUMÁRIO

1. INTRODUÇÃO

Timetabling é um problema de escalonamento sujeito a restrições, no qual os eventos são considerados como instâncias ou recursos do problema. Os recursos não podem ser solicitados por dois ou mais eventos ao mesmo tempo ou devem existir em uma quantidade suficiente para servir todos os eventos durante todo o tempo de escalonamento (ROSS et al., 1999). Um *timetable* genérico pode ser definido por: (i) um conjunto finito de eventos que inclui atividades diversas como exames, seminários, projetos ou aulas; (ii) um conjunto finito de horários, para realização dos eventos e (iii) um conjunto finito de agentes (instrutores, monitores ou professores), que têm o papel de monitorar eventos particulares (ROSS et al., 2000). *Timetabling* pode ser aplicado a diversos contextos, e para isso basta modificar as variáveis e as restrições envolvidas no problema. Algumas variações de *Timetabling* podem ser descritas como: escalonamento de funcionários em turnos, remanejamento de máquinas em fábrica, tabelas de horários ou recursos (exames, salas de aula, entre outros) para escolas ou para universidades. Assim, o problema é caracterizado por uma natureza fixa e um conjunto de restrições variáveis (BORGES, 2003).

Todos os anos, instituições de ensino enfrentam, principalmente, no início do ano letivo, uma grande dificuldade para gerar suas grades horárias. Para a criação de um calendário letivo de qualidade é necessário levar em conta diversos fatores, tais como: número de matérias, de alunos e professores; salas de aula disponíveis; tempo de duração de aula; número de aulas por semana; respeitando certo número de restrições. Esta dificuldade em gerar grades horárias é conhecida na literatura como problema de *Timetabling*, que consiste em associar horários e recursos a eventos (aulas), tentando satisfazer, da melhor forma possível, diversas restrições (BURKE et.al., 2003 apud VIEIRA, MACEDO, 2011).

O problema de *Timetabling* em *instituições de ensino* envolve à alocação de eventos em um período de tempo definido. Os eventos por sua vez, envolvem combinações de recursos (por exemplo, salas, pessoas e equipamentos), alguns dos quais podem ser especificados pelo problema e alguns dos quais devem ser alocados como parte da solução. *Timetabling* tem sido conhecido por pertencer a classe dos chamados problemas NP-Completo (COOPER; KINGSTON, 1996) (isto é, sem um método de resolvê-lo num tempo polinomial razoável). Existem diversas variações fundamentais sobre este problema. *Timetabling* de instituições de ensino podem ser divididas em duas categorias principais: aulas e exames. As diferenças maiores entre a programação de aulas e exames são: exames

devem ser agendados para que nenhum aluno tenha mais de um exame por vez, mas aulas normalmente devem ser agendadas antes das matrículas estudantis serem conhecidos. Como o espaço é frequentemente um problema, aulas compartilham salas ou são divididas entre elas, mas apenas uma aula pode ser realizada em uma sala a qualquer momento (BURKE, ELLIMAN, WEARE, 1995).

No problema de agendamento, o desafio básico é agendar aulas num período limitado de tempo de modo a evitar conflitos e satisfazer uma série de problemas laterais (CARTER, LAPORTE, CHINNECK, 1994). Os conflitos ocorrem sempre que um agendamento requer que qualquer recurso esteja em dois lugares ao mesmo tempo. Os conflitos laterais variam entre instituições, entre exames e agendamentos de aulas, e a lista é ilimitada. Desafio é uma palavra muito apropriada para ser utilizada com a resolução do problema de *Timetabling*. Dependendo do tamanho do departamento e da diversidade do curso, o tempo necessário para produzir o horário das aulas pode variar de uma tarde até um mês inteiro de trabalho (BLOOMFIELD, MCSHARRY, 1979). O processo de definição dos horários é dificultado pelo fato de que várias pessoas são afetadas pelo seu resultado. De acordo com Romero (1982), são três as principais partes interessadas no agendamento de horários em instituições de ensino, cada uma com seu próprio conjunto de objetivos e desejos:

1. Administração - que estabelece as normas mínimas, com as quais, o calendário deve estar em conformidade. Por exemplo, algumas universidades especificam que nenhum estudante deve ter dois exames em períodos consecutivos.

2. Departamentos – que apresentam as solicitações mais propensas a serem priorizadas no calendário do curso. Podem solicitar que a "agenda esteja em sintonia com o desenvolvimento do tema ensinado", bem como fazer exigências específicas para certas salas de aula ou laboratórios. Em conflito de exames, podem requerer que longos exames sejam colocados mais cedo, visando dar mais tempo para a realização do mesmo.

3. Estudantes – que possuem visão do calendário restrita à parte que os afeta. Devido ao grande número de alunos envolvidos é difícil de obter critérios específicos, como por exemplo, um horário que satisfaça todos os alunos. Muitos alunos preferem não ter uma pausa entre exames consecutivos para não ter aulas no horário noturno da sexta-feira.

Existem muitos estudos voltados à análise de *Timetabling* e à busca de soluções no meio computacional. Para este problema não existe uma solução genérica, a qual sempre possa ser aplicada para alcançar o resultado esperado. Cada situação requer uma solução específica.

Em termos computacionais, não é conveniente o uso de programação determinística para solucionar *Timetabling*. Muitos estudos são realizados na tentativa de propor métodos para auxiliar a descoberta de uma solução ótima, pois, a sua complexidade computacional estimula o contínuo estudo sobre o tema (BURKE et al., 1997). Nas últimas décadas, diferentes abordagens computacionais têm sido propostas para resolver o problema do *Timetabling*, como por exemplo: Busca Tabu (COSTA, 2003), Recozimento Simulado (SILVA, 2005) e *GRASP* (*Greedy Randomized Adaptive Search Procedure*) (COSTA, 2003) e Algoritmos Genéticos (RAGHAVJEE, PILLAY, 2010; MAHIBA et al., 2012), entre outras.

Os Algoritmos Genéticos (AGs) se enquadram em um dos ramos da computação evolutiva que, assim como definido por Goldberg (1989 apud TICONA, 2003, p. 46), foi criado com o intuito de imitar determinados processos observados na evolução natural das espécies. Deste modo, fundamenta-se na teoria de Charles Darwin a respeito da seleção e evolução dos indivíduos na natureza, como também, em outras teorias de genética formuladas, posteriormente, por estudiosos tais como Gregor Mendel.

Entre os principais fatores que fazem do AG uma técnica bem-sucedida, destacam-se (GOLDEBERG, 1989): (i) simplicidade de operação; (ii) facilidade de implementação; (iii) eficácia na busca da região onde, provavelmente, encontra-se o máximo global; (iv) aplicação em situações onde não se conhece o modelo matemático ou quando ele é impreciso. E também em funções lineares e não lineares.

De uma maneira geral, os AGs têm apresentado bons resultados na resolução de problemas de *Timetabling*, sendo muito usados na resolução de problemas de *Timetabling* escolar e acadêmico. Sistemas automatizados para agendamento de horários usando AGs vêm sendo desenvolvidos e/ou utilizados com sucesso, ao longo das duas últimas décadas, em universidades estrangeiras - como a de Edimburgo (CORNE, ROSS, FANG, 1994), de Napier (PAECHTER et al., 1994) e de Nottingham (BURKE, ELLIMAN, WEARE, 1995) – e brasileiras como a UFLA (TIMOTÉO, 2202), UFPR (BORGES, 2003), UFU (HAMAWAKI, 2005), FURB (PREIS, 2007) e UFS (VIEIRA, MACEDO, 2011), entre outras.

Neste contexto, esta proposta de trabalho de conclusão de curso visa contribuir para o maior entendimento de conhecimentos teóricos e práticos envolvidos em aplicações de algoritmos genéticos em otimização de horários de instituições de ensino.

2. O PROBLEMA DA GERAÇÃO DE HORÁRIOS EM INSTITUIÇÕES DE ENSINO

2.1 INTRODUÇÃO

A solução manual do problema do *Timetabling* normalmente requer muitos dias de trabalho de uma pessoa, podendo gerar grades horárias insatisfatórias e violar diversas restrições (FILHO, 2000).

Portanto, o desenvolvimento de sistemas automatizados que otimizem a produção de grades horária é uma meta importante, específica e sempre atual para escolas e universidades.

2.2 CLASSIFICAÇÃO DO PROBLEMA DE *TIMETABLING*

Existem diversas variantes para o problema de *Timetabling* de instituições de ensino. De uma maneira geral, as variantes de problema podem ser enquadradas em uma das seguintes classes:

Course Timetabling: agendamento semestral das aulas de um conjunto de cursos de uma universidade evitando ocorrência de cursos com estudantes em comum (SCHAERF, 1999).

Examination Timetabling: agendamento de exames (provas) de certa quantidade de cursos de uma universidade, evitando exames sobrepostos de cursos com estudantes em comum, e distribuindo os exames o mais longe possível (SCHAERF, 1999).

School Timetabling: agendamento semanal das aulas de uma escola, dificultando que alunos e professores tenham mais de uma aula ao mesmo tempo (SCHAERF, 1999).

University Timetabling: semelhantes ao *School Timetabling* podem existir ainda mais restrições pela quantidade de alunos, salas sendo compartilhadas com diversos cursos e diversos turnos com professores de vários cursos.

2.3 ABORDAGENS DO PROBLEMA

O problema *Timetabling* é NP-Hard (EVEN, ITAI, SHAMIR, 1975) e várias técnicas têm sido desenvolvidas para automação desse problema. O estudo do problema foi iniciado por Gotlieb (1963) e a partir daí vários artigos vêm sido publicados em conferências e vários

sistemas desenvolvidos com sucesso (SCHAERF, 1999, TIMOTÉO, 2002). O número de soluções possíveis para esse problema é enorme.

Um exemplo real, usando uma grade horária da Universidade Católica de Pernambuco (UNICAP) do Curso de Ciência da Computação (C3) localizada em Recife-PE, será apresentado a seguir para ilustrar o grande número de soluções e a variedade de restrições envolvidas na solução do problema. As aulas do curso são realizadas na parte da tarde e à noite, possuindo por dia, três horários disponíveis a tarde e dois horários disponíveis a noite (NO e PQ). Na Tabela 2.1 estão ilustrados os horários disponíveis no horário da noite. Cada horário é considerado um *slot*. Portanto, cada turma do horário noturno tem 10 *slots* disponíveis.

Tabela 2.1: Horários disponíveis no turno noturno do C3 da UNICAP.

Horário	Dia da Semana				
	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
NO 18:30-20:10	-	-	-	-	-
PQ 20:20-22:00	-	-	-	-	-

Na Tabela 2.2 são apresentadas as turmas, o número de professores e o total de possibilidades de grades horárias que podem ser geradas para 7 turmas do horário noturno do Curso de Ciência da Computação da UNICAP.

Tabela 2.2: Turmas e professores para o Curso de Computação da UNICAP.

Turma	NS3	NS4	NS5	NS6	NS7	NS8	NS9
Número de Professores	5	5	5	5	4	5	5
Total de Possibilidades	30240	30240	30240	30240	5040	30240	30240

O total de possibilidades pode ser calculado pelo arranjo combinatório ($A_{n.p}$) do número de *slots* disponíveis (*n=10*) em relação ao número de professores (*p*) professores disponíveis. O produto do total de possibilidades de cada turma fornece o número total de soluções possíveis. Para o exemplo do Curso de Ciência da Computação da UNICAP, o número total de soluções possíveis é igual a $3{,}854 \times 10^{30}$, um número extremamente grande. Se o método da força bruta fosse usado, considerando que a máquina onde o algoritmo será processado consiga gerar uma solução em 1 *ms*, o tempo de processamento seria $3{,}854 \times 10^{30}$ *ms* ou $1{,}22 \times 10^{20}$ *anos*.

Neste caso, usar o método da força bruta é totalmente inviável. O que mostra o trabalho e inteligência dos funcionários que resolvem o problema manualmente e a importância da automatização do processo de produção de grade horárias.

2.4 RESTRIÇÕES E PARTICULARIDADES

O problema de geração de horário de aulas acadêmico é determinar uma sequência de encontros entre estudantes e professores, em um espaço de tempo pré-definido sem que aconteçam conflitos e horários esparsos (“janelas”), levando em conta as restrições de cada professor e a quantidade de aulas semanais de cada disciplina. Algoritmos Genéticos apresentam um grande potencial de uso na área de agendamento de horários na área educacional, particularmente por sua capacidade de considerar e aperfeiçoar uma grande variedade de diferentes restrições que podem ser encontradas em instituições de ensino.

Exemplos detalhados de restrições comuns em escolas podem ser encontradas em Timóteo (2002). A seguir são exemplificados alguns casos de restrições e particularidades que podem ocorrer no Curso de Ciência da Computação da UNICAP.

2.5 COLISÃO POR PROFESSOR

Essa restrição avalia se um professor ministrará mais de uma aula simultaneamente. As Tabelas 2.3 e 2.4 demonstram o ocorrido. Nesse caso ocorreu uma colisão, pois a professora Ana está ministrando aulas no mesmo horário para turmas diferentes.

Tabela 2.3: Horário gerado para a turma NS3.

Horário	Dia da Semana				
	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
NO 18:30-20:10	Circuitos Dig. / Madeiro	Est. Dados I / Ana	-	-	-
PQ 20:20-22:00	-	-	-	-	-

Tabela 2.4: Horário gerado para a turma NS4.

Horário	Dia da Semana				
	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
NO 18:30-20:10	-	Est. Dados II / Ana	-	-	-
PQ 20:20-22:00	Mét. Numéricos / Bertino	-	-	-	-

2.6 HORÁRIOS ESPARSOS

Essa restrição verifica se a grade horária formada possui alguma janela (*slot* vazio), ou seja, se possuem horários vagos num determinado dia. A Tabela 2.5 mostra um dia semanal contendo uma janela.

Tabela 2.5: Exemplo de janela na semana.

NS5	Segunda-feira
NO 18:30-20:10	-
PQ 20:20-22:00	Paradig. Ling de Prog. / TJ

2.7 BLOCOS DE DISCIPLINAS

No curso de Ciência da Computação, até 20 créditos, as disciplinas devem ser alocadas obedecendo a seguinte modulação: 2NO-4NO, 2PQ-5PQ, 3NO-6NO, 3PQ-5NO, 4PQ-6PQ. A Tabela 2.6 apresenta uma grade horária, na qual as disciplinas – INF1701 Computação Gráfica, INF1218 Sistemas Operacionais I, INF1720 Engenharia de Software, INF1224 Banco de Dados II e INF1617 Redes de Computadores II - foram alocadas respeitando rigorosamente a modulação supracitada.

Tabela 2.6: Formação de blocos de disciplinas.

Horário	Dia da Semana				
	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
NO 18:30-20:10	INF1701 Márcio	INF1720 Antônio	INF1701 Márcio	INF1224 Márcio	INF1720 Antônio
PQ 20:20-22:00	INF1218 Mozart	INF1224 Márcio	INF1617 Rubens	INF1218 Mozart	INF1617 Rubens

2.8 PREFERÊNCIAS DOS PROFESSORES

Normalmente os professores trabalham em mais de uma instituição ou trabalham em mais de um curso e tem suas opções de horário restritas a alguns dias e turnos na semana. Por isso, além de avaliar todas as restrições anteriores é preciso observar se um professor está lecionando num horário preferencial. O que se deseja é que se saiba a preferência de horário para cada professor indicando os horários que o professor prefere os que não podem, e os horários que o professor pode lecionar mais não prefere ensinar naquele horário. Na Universidade Católica de Pernambuco é feito semestralmente essa pesquisa para cada professor para melhor adaptação do *Timetabling*.

2.9 CONSIDERAÇÕES FINAIS

As restrições citadas são exemplos algumas das várias que devem ser atendidas durante a produção de grades horárias do turno noturno do Curso de Ciência da Computação da UNICAP. Algumas destas restrições e particularidades também se aplicam a produção de grades horárias do turno da tarde, outras não. Como cada uma dessas instituições possui um regimento interno, particularidades serão impostas além das apresentadas.

O proveito da utilização de um AG é que esses critérios são definidos na função de avaliação. Assim sendo, se houver qualquer modificação em relação aos critérios, altera-se a função de avaliação onde houver a alteração e não o algoritmo completo.

3. ALGORITMOS GENÉTICOS

3.1 COMPUTAÇÃO NATURAL

A nomenclatura chamada de computação natural vem sendo empregada na literatura para descrever todos os sistemas computacionais implementados com inspiração de algum mecanismo natural ou biológico processador de informação (BALLARD, 1999; GRAMß et al., 2001; FLAKE, 1999; PATON et al., 2004; DE CASTRO, VON ZUBEN, 2004).

A computação natural é constituída por abordagens computacionais caracterizadas pela proximidade com a natureza. Dentre os vários objetivos da computação natural, destacam-se:

• Implementar dispositivos (computacionais) que emulam, modelam, simulam fenômenos naturais e descreve sistemas;

• Produzir utilitários matemáticos e computacionais para a solução de problemas complexos em várias áreas do conhecimento;

• Criar artificialmente novas formas de vida, chamadas de vidas artificiais;

• Usar mecanismos naturais, como cadeias de DNA e técnicas de engenharia genética, como novos paradigmas de computação. Estes novos paradigmas vêm complementar os computadores atuais baseados em tecnologia de silício e arquitetura de Von Neumman.

Existem vários exemplos de objetos produzidos com inspiração na natureza, tais como velcro (plantas), sonares (morcegos), coletes a prova de bala (teias de aranha), aviões (pássaros), entre vários outros. Além disso, a atenção da natureza permitiu o desenvolvimento de várias teorias e leis para relatar o seu comportamento.

Como exemplo, podem-se citar, entre outras, algumas leis da física: como as leis de Newton na mecânica e as leis de Maxwell no eletromagnetismo. A computação natural também está fortemente ligada à natureza sob diversas perspectivas e com abordagens

distintas. Como exemplo, o funcionamento do cérebro humano que inspirou o desenvolvimento das redes neurais artificiais (HAYKIN, 1999) e o funcionamento do sistema imunológico dos vertebrados que inspirou os sistemas imunológicos artificiais (DE CASTRO, TIMMIS, 2002).

Deste modo, a computação natural pode ser vista como uma versão computacional dos processos de análise (obtenção de ideias, mecanismos, fenômenos e modelos teóricos) e síntese da natureza para o desenvolvimento de sistemas "artificiais", ou ainda como a utilização de meios e mecanismos naturais para realizar computação. É importante salientar que a palavra "artificial" no contexto de computação natural significa apenas que os sistemas e dispositivos resultantes são desenvolvidos por seres humanos ao invés de serem produtos diretos da evolução das espécies. A área de computação natural pode ser divida em três grandes subáreas (DE CASTRO, VON ZUBEN, 2004):

1. Computação inspirada na natureza: inclui todas as estratégias desenvolvidas a partir de ou inspiradas em algum mecanismo biológico ou natural.

Como exemplos têm-se as redes neurais artificiais (HAYKIN, 1999), a computação evolutiva (BÄCK et al., 2000a, b), a inteligência coletiva (BONABEAU et al., 1999; KENNEDY et al., 2001), e os sistemas imunológicos artificiais (DASGUPTA, 1999; DE CASTRO, TIMMIS, 2002). É importante destacar que os próprios aspectos cognitivos e do raciocínio humano representam mecanismos naturais de fundamental importância. Logo, a inteligência artificial simbólica (RUSSELL, NORVIG, 2004) e os sistemas nebulosos (*fuzzy*) (PEDRYCZ, GOMIDE, F., 1998), essencialmente relacionados a estes temas, também podem inserir-se no escopo da computação natural.

2. Estudos sobre a natureza através da computação: envolve a utilização de mecanismos computacionais para a síntese de comportamentos naturais, padrões e processos biológicos. As linhas de atuação predominantes são os estudos sobre vida e organismos artificiais (ADAMI, 1998, LANGTON, 1988, LEVY, 1992), e a geometria fractal (MANDELBROT, 1983, PEITGEN et al., 1992).

3. Computação com mecanismos naturais: trata-se de um novo paradigma de computação onde mecanismos naturais, como, por exemplo, as cadeias de DNA e os bits quânticos, são

utilizados como estruturas de dados para o desenvolvimento de “computadores naturais”. Os produtos desta subárea podem ser vistos como candidatos a substituir ou complementar os computadores digitais disponíveis atualmente. Seus principais representantes são a computação molecular (PĂUN et al., 1998, GRAMß et al., 2001) e a computação quântica (HIRVENSALO, 2004, NIELSEN, CHUANG, 2000). Estas três subáreas da computação natural têm patrocinado a proposição de poderosas ferramentas computacionais para a solução de problemas complexos de engenharia e têm permitido a geração de novos paradigmas de computação (PATON, 1994; PATON et al., 2004; YOKOMORI, 2000). É importante salientar que se trata de uma linha de pesquisa promissora, não apenas porque resultados concretos e convincentes têm sido produzidos pela aplicação de ferramentas concebidas exclusivamente com base nos princípios da computação natural (ZIMMERMANN, 1999, DASGUPTA; MICHALEWICZ, 1997), mas porque diversos aspectos da biologia podem ser mais bem tratados tendo em mãos ferramentas computacionais apropriadas.

No contexto desse trabalho, abordaremos a técnica de AG, que se fundamenta na área de estudo da computação evolutiva, que utiliza como base as definições de comportamento adaptativo (GOLDBERG; HOLLAND, 1988 apud BITTENCOURT, 2008) definidos por Darwin, em sua Teoria da Evolução Natural.

3.2 DEFINIÇÃO DO ALGORITMO

Os Algoritmos Genéticos constituem uma técnica de busca e otimização inspirada no princípio Darwiniano de seleção natural e reprodução genética (GOLDBERG, 1989). Tendo sido inventado por John Holland nos anos 60 e posteriormente desenvolvido por seus alunos na Universidade de Michigan em meados de 1970, a aplicação dos Algoritmos Genéticos teve como propósito dedicar se ao estudo formal dos fenômenos de evolução, assim como é percebido na natureza (AGUIAR, 1998). De acordo com a teoria de Charles Darwin, o princípio da evolução favorece indivíduos melhores adaptados ao ambiente, proporcionando com isso que estes tenham maior possibilidade de longevidade e reprodução. Com isso, indivíduos mais bem adaptados, tem maior possibilidade de perpetuação do seu código genético nas próximas gerações. Em algoritmos genéticos, um cromossomo é uma das estruturas de dados que representa uma das possíveis soluções do espaço de busca do problema, os cromossomos são então submetidos a um processo que inclui avaliação, seleção e recombinação sexuada (*crossover*) e mutação (PACHECO, 1999). Algoritmos genéticos

têm sido aplicados também a diversos problemas de otimização (MICHALEWICZ, 1996), tais como: otimização de funções matemáticas; otimização combinatorial; otimização de planejamento; problemas como do caixeiro viajante ou de otimização de rota de veículos; otimização de layout de circuitos; otimização de distribuição; otimização em negócios e síntese de circuitos eletrônicos.

3.3 TERMINOLOGIA BIOLÓGICA

De acordo com a biologia, a teoria da evolução diz que o meio ambiente seleciona, em cada geração, os seres vivos mais aptos de uma população. Resultando, que somente os mais aptos onde, conseguem se reproduzir. Durante a reprodução, ocorre, entre outros, fenômenos como cruzamentos e mutações que atuam sobre os genes armazenados nos cromossomos.

Os resultados destes fenômenos levam à variabilidade dos seres vivos da população. Sobre esta população diversificada age a seleção natural, permitindo a sobrevivência apenas dos indivíduos mais adaptados. Os AGs são inspirados nestes fenômenos, o que explica o uso de muitos termos vindos da biologia. A lista apresentada a seguir mostra os principais termos usados na biologia e sua relação com os AGs (CASTRO, 2001, LIMA, 2005; SILVA, 2001):

Cromossomo e Genoma: Genoma é o conjunto completo de genes de um organismo. Um genoma pode ter vários cromossomos. Já nos AGs, os dois representam a estrutura que codifica uma solução para um problema, sendo assim, um cromossomo ou genoma é capaz de se representar dentro do espaço de busca.

Gene: Na biologia é a unidade de hereditariedade que é transmitida pelo cromossomo e que controla as características do organismo. Nos AGs é um elemento codificado no cromossomo.

Indivíduo: Um simples membro da população. Nos AGs, um indivíduo é formado pelo cromossomo e seu valor de aptidão.

Genótipo: Biologicamente, representa a composição genética contida no genoma. Já nos AGs, é usada para representar o que existe contido no cromossomo ou genoma.

Fenótipo: Nos AGs representa o objeto, estrutura ou organismo construído a partir das informações do genótipo. É o cromossomo decodificado. Por exemplo, considerando que o cromossomo codifica parâmetros como as dimensões das vigas em um projeto de construção de um edifício, ou as conexões e pesos de uma Rede Neural. O fenótipo seria o edifício construído.

Alelo: Na biologia é responsável por representar uma das formas de um gene. Já, nos AGs, representa os valores que o gene pode assumir. Por exemplo, um gene que representa o parâmetro cor de um objeto poderia ter o alelo azul, preto, branco etc.

Epistasia: Interação entre genes internos do cromossomo, isto é, quando um valor de gene influência o valor de outro gene. Problemas com alta epistasia são difíceis para um AG solucionar.

População: Conjunto de cromossomos ou soluções.

Pais: Indivíduos que transmitirão suas características para a nova prole.

Filhos: Nova prole, ou indivíduos formados pelos "pais".

População temporária: População intermediária de um AG. Substitui a geração atual parcialmente ou completamente, conforme o modelo de inserção de indivíduos escolhidos.

Geração: É o número de iterações que o algoritmo executa. A cada iteração uma nova população é gerada.

Operações Genéticas: Operações que os AGs realizam sobre os cromossomos.

Espaço de Busca ou Região Viável: Região que possui as soluções possíveis ou viáveis do problema a ser otimizado. Deve ser caracterizado pelas funções de restrição, que definem as soluções viáveis ao problema a ser resolvido.

Função de Avaliação ou Função Objetivo: É a função utilizada para a otimização da população. Nela contém informações respectivas a cada cromossomo. Estão nelas representadas as características do problema que o AG necessita para seu objetivo, sendo expressa normalmente como *F = f(x1, x2, x3, ..., xn),* onde *x1, x2, x3, ..., xn* são os valores que o AG procura determinar para otimizar *F* (Função de aptidão). Essa função é calculada individualmente para cada cromossomo. (RIBEIRO, 2012)

3.4 CÓDIGO DE UM ALGORITMO GENÉTICO GENÉRICO

Um AG, em forma de pseudocódigo, é mostrado na Figura abaixo.

Algoritmo genético genérico
Inicialize a população de cromossomos (geração *i=1*)
Avalie indivíduos na população (função objetivo e sobrevivência)
Repita (evolução)
Selecione indivíduos para reprodução
Aplicar operadores de recombinação e/ou mutação
Avaliar indivíduos gerados na população
Selecione indivíduos para sobreviver (geração *i=i+1*)
Até máximo de gerações ou objetivo final
Fim

Figura 3.1: Algoritmo genético genérico.

Após inicializar a população, a cada geração o algoritmo genético irá selecionar os indivíduos mais aptos, para então, aplicar sobre estes os operadores de recombinação e mutação, gerando com isso a prole que irá servir para compor a geração seguinte. Este processo será repetido iterativamente até que a condição de parada seja alcançada (RIBEIRO, 2012).

3.5 INICIALIZAÇÃO DA POPULAÇÃO

Em um algoritmo genético a população inicial é criada após a execução do operador denominado inicialização. A partir deste operador uma população com *n* indivíduos é criada,

sendo que cada um desses indivíduos serão considerados como cromossomos dentro desta população.

Cada indivíduo deverá apresentar um conjunto de genes (conhecidos também como genótipo) e um conjunto de características observáveis (conhecida como fenótipo do indivíduo). Logo, o fenótipo corresponde a interação do conteúdo genético com o ambiente, sendo que esta interação é representada pelo conjunto de parâmetros do algoritmo genético (RIBEIRO, 2012).

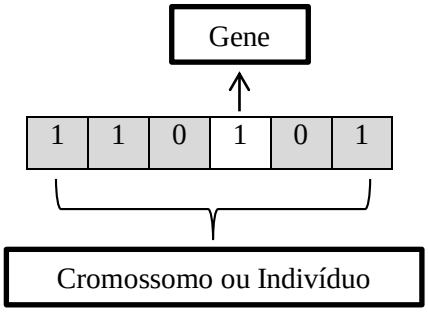

Figura 3.2: Exemplo de gene e cromossomo.

O início da criação pode acontecer de diversas formas, o mais usual é a geração de populações aleatórias. Em certos casos, a adição de heurísticas pode ser um fator de ajuda para a resolução do problema. Nestes casos, costuma-se usar cromossomos com soluções aproximados em meio ao restante da população. Quando usada esta técnica, devemos nos atentar para a possibilidade de que o processamento do algoritmo possa a vir a convergir antes do tempo necessário, fazendo com que em um curto espaço de tempo a população tenha indivíduos muito semelhantes.

Procurar por soluções em um espaço de busca reduzido, pode ocasionar de acordo com Reeves (1993) um risco de não obtenção de respostas satisfatórias. Por outro lado, ao utilizar-se de população grandes demais, o esforço computacional requerido pode ser excessivo para a resolução do problema.

Finalizando, existe uma técnica chamada *seeding*, que consiste em colocar na população inicial, soluções encontradas por outros métodos de otimização podendo ser útil em várias situações. Este método irá garantir que as populações geradas pelos algoritmos sejam, pelo menos, tão boa quanto a gerada por estes métodos (SILVA, 2001).

3.6 AVALIAÇÃO DA POPULAÇÃO

A avaliação da população é realizada pela função de aptidão, esta deve indicar o quão apto um indivíduo é dentro da população, ou seja, o quão o indivíduo está próximo da solução final do problema. De forma parecida ao que ocorre na natureza, em um AG, indivíduos mais adaptados tenderão a multiplicar suas características, enquanto o inverso tenderá a desaparecer.

Durante a avaliação de cada um dos indivíduos tem-se a geração do *fitness* valor esse que será responsável por verificar e ordenar o grau de aptidão do indivíduo para a solução do problema. Em um AG, essa etapa, na maioria das aplicações, é a fase mais crítica do processo, já que deve ser realizada para cada cromossomo de cada população durante todo o processo de evolução (TIMÓTEO, 2002).

3.7 SELEÇÃO

A seleção dos indivíduos para reprodução é definida pelos operadores genéticos de seleção. Esses influenciam na eficácia da resposta, velocidade de convergência e na eficiência do algoritmo (GOMIDE, L. R. et al., 2009). Os cinco operadores genéticos de seleção mais usados são:

a) Elitista - tem como princípio selecionar apenas os melhores indivíduos via *fitness*, os quais são usados na formação de novas gerações (SAKAWA, 2002).

b) Torneio - a seleção é obtida por meio da competição entre *k* indivíduos, sendo escolhido sempre o melhor (maior *fitness*) para compor o grupo a reproduzir (GOMIDE, L. R. et al., 2009; RIBEIRO, 2002).

c) Roleta - a seleção dos indivíduos é feita de acordo com a probabilidade de serem selecionados, segundo os valores de *fitness*. Quanto maior o *fitness*, maior a probabilidade de serem selecionados, a cada nova rodada da roleta (GOLDBERG, 1989).

d) Bi-classista - o método considera a seleção proporcional ao valor ordenado do *fitness* da população, sendo selecionada uma porção para os melhores *fitness* e outra para os piores, até completar a lista de seleção (GOMIDE, L. R. et al., 2009).

e) Mapeamento de Função Objetivo - que pode se valer do Escalonamento Linear e do Método de Ordenamento Linear (TIMÓTEO, 2002).

3.8 OPERADOR DE *CROSSOVER* (CRUZAMENTO)

O operador de *crossover* é utilizado para o cruzamento entre dois cromossomos. Para aplicar este operador é necessário selecionar os cromossomos e o(s) ponto(s) onde o cruzamento deve(m) ser realizado(s).

Para isso, a lista de indivíduos selecionados é embaralhada aleatoriamente, criando-se, uma segunda lista chamada lista de parceiros. Cada indivíduo selecionado é então cruzado como o que ocupa a mesma posição na lista de parceiros. A forma de realizar esse cruzamento encontra-se ilustrada na Figura 3.3. Os cromossomos de cada par de indivíduos a serem cruzados são seccionados em um ponto, chamado de ponto de corte, sendo escolhido aleatoriamente.

Supondo que possua uma taxa de *crossover* de 70%. Sorteando um número aleatório entre 0 a 1 e encontrar um número igual ou maior a 0.7, não será realizado o crossover entre os cromossomos assim permanecendo inalterados. Por outro lado, caso o valor seja menor, deve-se efetuar um "corte de cruzamento" e para isso basta sortear uma posição no cromossomo e, então trocar uma parte de um cromossomo com a outra parte do outro cromossomo.

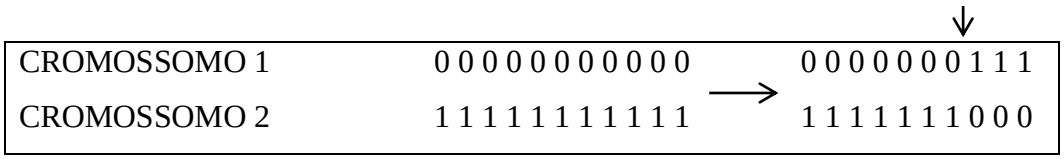

Figura 3.3: Exemplo do processo de crossover.

O exemplo acima é o *crossover* padrão chamado de *crossover* com corte em um ponto (1PX), mas existem outros tipos de crossover que se aplicam a alguns algoritmos, como por exemplo, o *crossover* com corte em dois pontos (2PX), semelhante ao (1PX) - a diferença é o uso de dois pontos de corte ao invés de um.

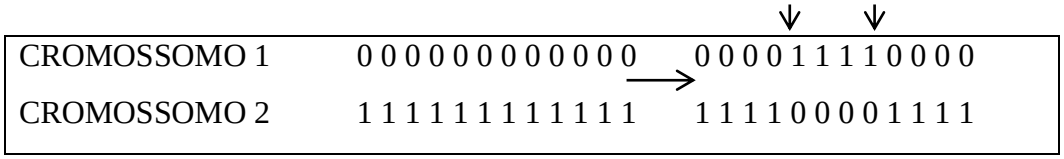

Figura 3.4: Exemplo do processo de crossover com corte em dois pontos (2PX).

Tem-se também o *crossover* com múltiplos cortes (MPX) onde é uma generalização dos operadores apresentados anteriormente. Para este operador, será sorteado um número fixo (*n*) de pontos de corte. Ao invés de um ou dois pontos, serão vários pontos de corte. Na maioria dos casos este operador não funciona de maneira eficiente. Geralmente, os operadores de crossover (1PX) são os que apresentam os melhores resultados. (GOLDBERG, 1989). E finalmente, tem-se o *crossover* segmentado (SX). Esse operador é apenas uma variação do operador de *crossover* (MPX). Ao invés de ser escolhido aletoriamente um número fixo de pontos, será escolhido números variáveis de pontos. Cada vez que o *crossover* for realizado, um novo número de pontos será sorteado. Tanto esse, quanto o operador anterior são complexos de serem implementados e normalmente não apresentam bons resultados (GOLDBERG, 1989).

3.9 MUTAÇÃO

A mutação permite a geração aleatória em um AG. Altera o valor do gene sem escolher especificamente qual será. Usado para que o AG tenha uma maior área de busca, evitando ficar com indivíduos criados apenas pelo cruzamento.

A operação de mutação trabalha sobre os indivíduos que resultam do processo de *crossover* com uma chance pré-definida e com isso efetuando alterações em seu interior. Normalmente a probabilidade é de baixo valor para não fazer da iteração do AG uma busca cega. Alguns dos mais usados operadores de mutação são (GOLDBERG, 1989):

- Mutação *creep*: um valor aleatório é somado ou subtraído do valor do gene.
- Mutação aleatória: cada gene a sofrer mutação recebe um valor aleatório do conjunto válido;
- Mutação por troca: são escolhidos *n* pares de genes, e trocam os valores entre si;

3.10 CRITÉRIO DE SOBREVIVÊNCIA

Nessa etapa, os indivíduos (descendentes) resultantes do processo de cruzamento e mutação irão formar a nova população segundo as regras adotadas pelo AG. As regras mais comuns segundo Goldberg (1989) são: (i) os descendentes sempre substituem os ancestrais;

(ii) os descendentes substituem os ancestrais somente se a média do grau de aptidão dos filhos for maior que a média de aptidão dos pais.

De fato, a grande parte dos algoritmos utiliza o método de sempre substituir os ancestrais pelos descendentes, já que ajuda a manter a diversidade dos indivíduos gerados. (GOLDBERG, 1989).

3.11 CONDIÇÕES PARA TÉRMINO

São as condições que determinam o fim do processo de iteração e podem ocorrer por: tempo, por especificação de um número de gerações e por convergência dos genes. O critério de convergência pode ser arriscado. Podem ocorrer casos, nos quais os indivíduos demorem a convergir. O número de gerações, geralmente, é a melhor escolha para terminar o processamento de um AG, pois não é influenciado pela máquina e torna término mais confiável (FILHO, 2000).

3.12 PARÂMETROS GENÉTICOS

Os seguintes parâmetros influenciam o processamento dos AGs (TIMÓTEO, 2012):

Taxa de cruzamento: quanto mais alto for o valor, mais rapidamente essa taxa gera novas estruturas que serão inseridas na população. Mas, se essa taxa for muito alta, as estruturas de qualidade são quebradas mais frequentemente. Se for o contrário, o algoritmo fica lento.

Taxa de mutação: uma baixa taxa previne que determinada posição do cromossomo fique fixa em um valor. Já o oposto faz com que a busca se torne basicamente aleatória.

Tamanho da População: o tamanho da população afeta o desempenho e a eficiência dos AGs. Uma população formada por poucos indivíduos pode fazer com que o desempenho de busca do AG piore. Isso ocorre pelo fato de fornecer uma cobertura menor da área para o problema. Uma população maior, geralmente gera uma globalização considerável do problema, diminuindo convergências prematuras para soluções locais ao invés de globais. Entretanto, populações muito grandes exigem mais recursos computacionais e dificultam a convergência.

4. APLICAÇÕES DE AGs NA RESOLUÇÃO DE TIMETABLING ESCOLAR E UNIVERSITÁRIO

4.1 INTRODUÇÃO

O desenvolvimento de *timetables* automatizadas facilita a alocação de recursos para tarefas sob restrições pré-definidas, tendo como principal restrição um número fixo de *slots* de tempo. A resolução manual deste problema é repetitiva, demorada, lenta e muitas vezes produzem resultados insatisfatórios. Otimização de agendamento de recursos usando AG - e também outras abordagens meta-heurísticas como a Busca Tabu e *Simulated Annealing* - reduzem o tempo de sua criação, minimizando os erros e retornando soluções cada vez mais próximas a um conjunto de objetivos. Problemas envolvendo geração de grade horária automática para escolas e universidades, funcionários de empresas, eventos esportivos, meios de transportes e hospitais são importantes e desafiadores e tem sido objeto frequente de estudo da Ciência da Computação e da Inteligência Artificial. Nesse capítulo, uma amostra formada por dez trabalhos (artigos, dissertações, monografias e trabalhos publicados em anais de congressos) que abordam a resolução do problema de *Timetabling* usando AGs foi selecionada, entre diversos trabalhos consultados e analisados, quanto ao objetivo, a forma de representação da solução, as restrições, a criação da população inicial, a forma de avaliação da população, os métodos utilizados na seleção dos indivíduos, os operadores aplicados e os resultados obtidos.

4.2 ANÁLISE DE OBJETIVOS E CARACTERÍSTICAS METODOLÓGICAS DE TRABALHOS SOBRE RESOLUÇÃO DE TIMETABLING USANDO AGS

RESOLUÇÃO DE TIMETABLING UTILIZANDO ALGORITMOS GENÉTICOS E EVOLUÇÃO COOPERATIVA

Autor: Suzan Kelly Borges, 2003 – Universidade Federal do Paraná. (UFPR)
Tipo de Trabalho: Dissertação.

- Objetivo do trabalho:

O objetivo geral do trabalho abordado por Borges (2003), foi investigar a aplicação de algoritmos co-evolutivos cooperativos para o caso específico de geração de grades horárias (*Timetabling*) para uma universidade. Os objetivos específicos do trabalho são apresentados a seguir:

- Especificação do problema e de sua modelagem.
- Investigação das várias abordagens, AG, Algoritmos Cooperativos e métodos não evolutivos.
- Implementação de um sistema, para produção automática de grades horarias para o curso de bacharelado em Informática da UFPR (Universidade Federal do Paraná) usando Algoritmos Cooperativos.
- Implementação de um sistema, para produção de grades horárias usando AG clássico.
- Comparar os resultados obtidos usando algoritmos co-evolutivos com os obtidos usando AG clássico.

- Modelagem do problema:

O estudo de caso abordou o curso de Ciência da Computação da UFPR, com duração de 4 anos sendo dividido em 8 períodos por semestre. Cada período ou turma tem uma semana representada por cinco dias de aulas e cada dia da semana distribuído em três horários que precisam ser ocupados com disciplinas. O corpo docente conta com 33 professores, e a grade curricular dos períodos é definida por 50 disciplinas.

- Representação da solução:

A representação por matrizes 2D (bidimensionais) adotada para os indivíduos durante o ciclo evolutivo dos AGs, encontra-se ilustrada na Figura 4.1.

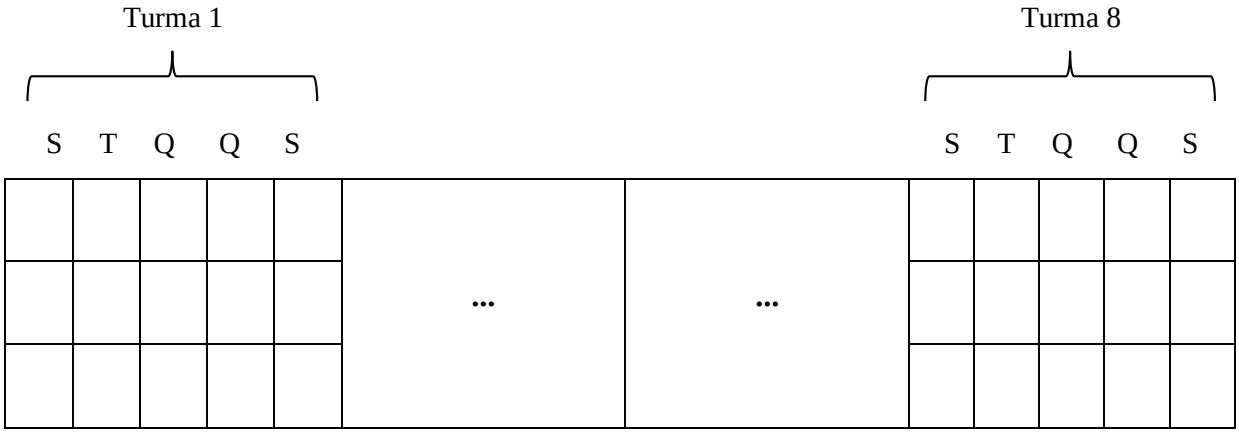

Figura 4.1: Representação de um cromossomo em um AG.
Fonte: (BORGES, 2003)

Os indivíduos são compostos por oito turmas. Cada turma está dividida em 5 dias semanais e 3 horários de aula diários. A população que evolui no ciclo genético é composta por indivíduos que definem grades para os oito períodos do curso.

- Restrições:

  Para o contexto do problema foram definidas as restrições:

  - Disciplinas devam ser ministradas apenas por professores pré-determinados.
  - Cada disciplina tem um número de carga horária por semana que deve ser respeitado.
  - Por período ou turma na universidade, somente é possível ser ministradas disciplinas apropriadas ao cenário da turma.
  - Preferências de horários por parte dos professores foram definidas em números que variam de 1 a 5. Quanto maior o número, menor é a preferência do professor pelo horário.
  - Cada professor tem limite semanal máximo estabelecido em doze aulas e o mínimo em duas.
  - Nenhum professor deve ser alocado em dois horários iguais em turmas diferentes.
  - Não se deve permitir que duas ou mais vezes uma disciplina fosse ministrada no mesmo dia.

- Criação da população inicial:

O processo que gera a população inicial respeita as restrições de aulas por disciplina e carga horária máxima de aula por professor para instanciação dos indivíduos. A não existência de colisão de horários de professores para os diferentes períodos é sempre verificada já que os operadores genéticos modificam as grades horárias constantemente. O algoritmo realiza antes da instanciação de cada gene uma verificação das restrições supracitadas. Se houver violação delas, outro valor é aleatoriamente gerado até que um indivíduo válido seja gerado. O processo continua até que todas as populações estejam de acordo com as restrições.

- Avaliação da população:

De acordo com as restrições do trabalho, a função de avaliação foi definida como:

$$fitness\ do\ indivíduo = \frac{1}{\left(1 + (restrições\ violadas)\right)} \quad (1)$$

O valor de restrições violadas será máximo (valor igual a 1) apenas se a variável restrição violada seja nula. Já a alteração da variável restrição é incrementada em uma unidade toda vez que uma restrição é violada.

- Método de Seleção:

- O método de seleção utilizado foi o método da Roleta viciada, no qual cromossomos com maior *fitness*, possuem maior possibilidade de serem escolhidos.
- O método de elitismo é também utilizado para garantir com que indivíduos com alto valores de *fitness* continuem em gerações posteriores. A taxa de elitismo utilizada foi de 10%.

- Operadores:

  - O de *crossover* foi adotado o *cruzamento parcialmente casado* (PMX), aplicável a problemas onde a ordem dos genes no cromossomo influencia no valor de *fitness* dos indivíduos. Foi utilizada uma taxa de 50%
  - O de mutação é aplicado somente aos indivíduos que são sujeitos a uma taxa já definida. Ocorrendo a mutação aleatória, num mesmo indivíduo dois genes são aleatoriamente selecionados e trocados e recalculado seu *fitness*. Usou-se uma taxa de 20%.

- Critério de parada:

O critério de parada adotado foi o máximo de gerações, ou existir alguma solução que alcance o *fitness* calculado, igual a 1.

- Resultados e Conclusões:

Para a viabilização dos comparativos entre Algoritmos Genéticos e Cooperativos os parâmetros evolutivos, como taxa de *crossover*, taxa de mutação e faixa de elitismo, foram mantidos.

Contudo, para que nenhum dos dois métodos fosse prejudicado quanto ao número de indivíduos a serem tratados no ciclo evolutivo, atribuiu-se ao tamanho da população na abordagem Genética o valor de 8000. Este número foi definido considerando que no Algoritmo Cooperativo são evoluídas 8 populações de 1000 indivíduos cada. Nas amostras analisadas com o Algoritmo Genético apenas duas alcançaram o *fitness* máximo, nas outras foram necessários atingir o máximo de 500 gerações.

O trabalho conclui que AGs clássicos encontram dificuldades no tratamento de problemas de otimização com muitas restrições como o *Timetabling*, diferentemente dos algoritmos co-evolutivos, que forneceram resultados satisfatórios. Borges (2003) deduz que esse resultado foi obtido por efeito de recursos adicionais importantes providos pelo método cooperativo.

ALGORITMOS GENÉTICOS NA OBTENÇÃO DE UMA GRADE DE HORÁRIOS COM MÚLTIPLOS CURSOS PARA UMA INSTITUIÇÃO DE ENSINO

Autores: Alexandre Brasil da Silva, Carlos Michel Betemps, Milton Heinen, 2014 – Universidade Federal do Pampa. (UNIPAMPA)
Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

Aplicar uma técnica heurística (Algoritmos Genéticos) para a resolução do problema da grade horária para múltiplos cursos em uma instituição de ensino. Ou sendo mais específico, percorrer os espaços das soluções usando uma técnica guiada, para que os recursos relativos à grade horária possam ser designados de maneira eficiente e que ofereçam a possibilidade de criação de uma aplicação funcional real com foco na estrutura de períodos letivos da Universidade Federal do Pampa (UNIPAMPA), localizada em Campus Bagé.

- Modelagem do problema:

Foi realizada para estudar o caso de alocação de horários do semestre 2014/1 da UNIPAMPA com 10 cursos de graduação, 248 disciplinas e 429 períodos disponíveis, considerando diversos outros parâmetros.

- Representação da solução:

Define o gene na forma de uma tupla composta por Disciplina, Docente, Sala, Semestre e Curso. O indivíduo é formado por um conjunto de genes e implementado na forma de uma lista de matrizes 2D (Figura 4.2). Cada objeto dessa lista se refere a grade horária de um semestre, e a matriz da grade do último semestre de um curso é seguida por uma matriz que possui a grade horária do primeiro semestre do próximo curso.

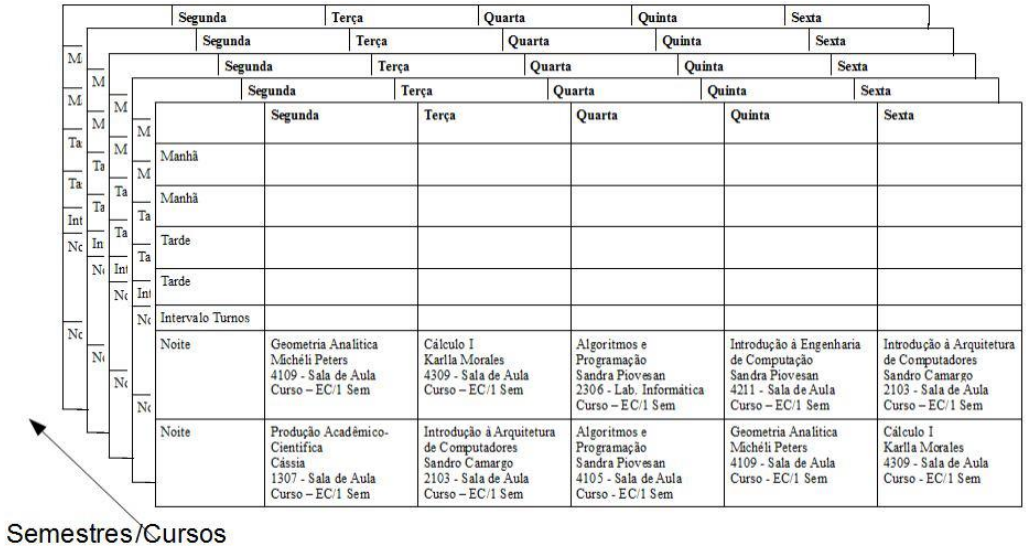

Figura 4.2: Indivíduo representado na forma de lista de matrizes.

Fonte: (SILVA et al., 2014)

- Restrições:

São levadas em consideração algumas restrições que devem ser obrigatoriamente atendidas chamadas de restrições rígidas (*hard constraints*):

- Não ocorrer aulas simultâneas para um mesmo professor em salas diferentes.
- Não haver disciplinas diferentes alocadas em uma mesma sala num mesmo horário.

Existem outras que podem ser aceitas, mas caso não forem, não invalidam a solução obtida que são as restrições flexíveis (*soft constraints*):

- A alocação dos professores conforme suas preferências de horários.
- Controle da carga horária distribuída entre os professores.

- Criação da população inicial:

É criada de forma aleatória, combinando nos genes as informações de disciplinas, professores, salas, semestres e cursos, e realocando-os dentro das matrizes dos indivíduos. O que deve ser observado é não alocar disciplinas à professores que não possuem relação com a mesma, e não alocar disciplinas em semestres diferentes dos cursos/semestres nos quais são ofertadas.

- Avaliação da população:

Leva em consideração as restrições descritas variando seu valor de acordo com a violação, ou não, de cada restrição. Um gene alocado em uma matriz é um bônus para o indivíduo, e também quando um professor é alocado em um período preferencial. Caso ocorram violações das restrições como, por exemplo, alocação em duas salas num mesmo período ou designação simultânea de disciplinas diferentes para uma mesma sala, penalizações são aplicadas.

O desvio padrão na distribuição de carga horária entre os professores é também avaliado. Indivíduos que tem carga horária distribuídas de maneira mais correta tem menor desvio padrão e por isso são menos penalizados em avaliações do que outros indivíduos em que o desvio padrão é mais alto.

A função de aptidão para avaliações dos indivíduos adotada no trabalho por Silva et al. (2014) foi:

$$f(x) = \sum bônus - \sum penalizações \qquad (2)$$

- Método de Seleção:

Os métodos escolhidos foram o Elitismo e a Roleta. O elitismo faz com os *m* melhores indivíduos da geração *t* sejam passados diretamente para a próxima. Os que faltarem para completar o indivíduo (*p* indivíduos) serão selecionados através da Roleta. A roleta seleciona *p* pares de indivíduos e os aplica aos operadores genéticos, que resultam os (*n –m*) indivíduos para compor a população.

O tamanho da população é dado pela equação:

$$n = m + p \qquad (3)$$

- Operadores:

Utiliza-se o *crossover*, baseado no parâmetro de probabilidade de cruzamento, verifica-se é possível o cruzamento.

Aplica-se a mutação aleatória se não for satisfeita a condição de cruzamento.

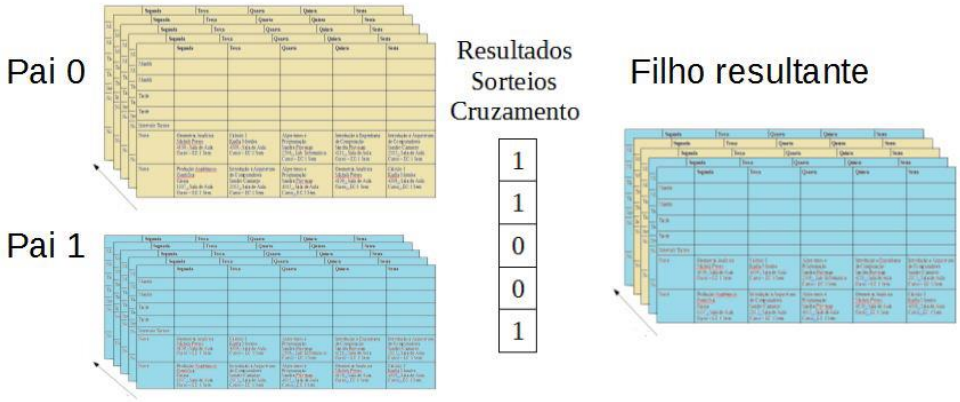

Figura 4.3: Operadores genéticos aplicados a lista de matrizes.
Fonte: (SILVA et al., 2014)

- Critério de parada:

Como parâmetro de parada foi adotado o número máximo de iterações (gerações). Quando satisfeito o indivíduo que for mais apto é tido como resposta para o problema.

- Resultados e Conclusões:

Foram efetuados testes baseados em uma planilha de horários do semestre 2014/1 da universidade aqui abordada com 10 cursos de graduação, 248 disciplinas e 429 períodos disponíveis.

A técnica utilizada se mostrou aplicável por meio de usos de critérios de bonificação e penalização das soluções obtidas, e com isso apresentou resultados adequados para as grades horárias geradas. Os valores dos parâmetros usados no algoritmo são apresentados na Tabela 4.1.

Tabela 4.1: Parâmetros utilizados no AG.

Parâmetro	Valor
Tamanho das populações	1000
Número máximo de gerações	3000
Percentual de Elitismo	10
Probabilidade de Mutação	0.5%
Probabilidade de Crossover	65%
Avaliação do melhor indivíduo	30709
Total de colisões de salas	0
Total de colisões de docentes	0
Total de Disciplinas Alocadas	248
Total de Períodos Alocados	429
Desvio padrão na distribuição de disciplinas por docente	1.91
Total de quebras de restrições de docentes em horários indisponíveis	0
Total de alocações em horários preferenciais de docentes	293
Tempo Total:	9402.4s ~ 2,5h

Fonte: (SILVA et al., 2014)

O artigo conclui que a técnica de exploração de espaço de busca mostrou-se funcional em relação ao problema proposto, com grades horárias geradas com sucesso. Ocorreu a obtenção de grades com colisões de salas e horários de professores, mas apenas com populações pequenas, que limitavam o espaço de busca do AG.

Quanto maior a população, melhor a qualidade da solução obtida. Já por causa dos operadores são executados mais vezes. Encontrou-se uma característica peculiar com populações de 300 indivíduos e 300 gerações que apresentaram soluções nas quais poucos períodos preferenciais foram alocados, sugerindo a necessidade de otimizar o número de gerações usado no AG.

Como trabalho futuro é sugerido alocar e dividir os professores em turmas e paralelizar o algoritmo para obter uma melhora no tempo de execução.

O PROBLEMA DE GERAÇÃO DE HORÁRIOS: UM FOCO NA ELIMINAÇÃO DE JANELAS E AULAS ISOLADAS

Autores: Leonardo Aparecido Ciscon, Adriano César Oliveira, Tatiana Ribeiro Hipólito, Guilherme Bastos Alvarenga, Ana Cristina Roullier, 2005 – Universidade Federal de Lavras (UFLA).

Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

Eliminação de janelas e aulas isoladas, através da busca dos horários mais aceitos pelos docentes. Esta generalização se mostrou necessária para o uso da solução proposta para a geração de horários para duas escolas escolhidas como estudo de caso.

- Modelagem do problema:

Modelagem usando Algoritmo Genético para tratamento do problema de geração de horários escolares (*Timetabling)*, observando a perspectiva de remoção de janelas e aulas isoladas do horário dos professores. Esses têm sido os grandes empecilhos na geração de horários tendo como ponto fundamental a aceitabilidade dos professores envolvidos. Janelas são horários ociosos entre duas aulas de um mesmo turno na grade horária de um professor, e as isoladas são aqueles presentes isoladamente em um dia. Apesar da sua importância, não existem referências a trabalhos que trabalham na minimização destas restrições. Tendo como base para o tratamento, duas escolas Escola Estadual Padre Rogério Abdala (EEPRA) (Monsenhor Paulo / MG) e Escola Municipal Álvaro Botelho (EMAB) (Lavras / MG).

- Representação da solução:

Foi escolhida uma forma intuitiva, onde cada gene representa um *slot*. Um cromossomo seria representado por uma matriz tridimensional (Número de Turmas X Número de Dias X Número de Horários). Esta estrutura não permite que existam duas disciplinas referentes a mesma turma que sejam lecionadas no mesmo horário. Mas ela não garante a não ocorrência de outras colisões.

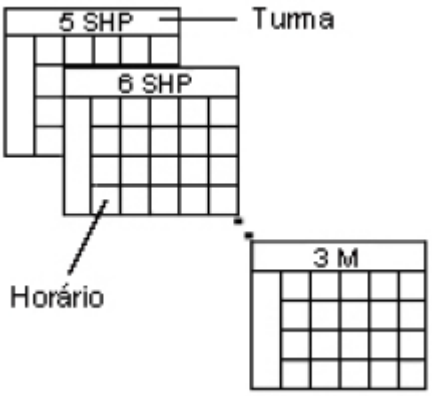

Figura 4.4: Representação do grupo de indivíduos (Solução).
Fonte: (CISCON et al., 2005)

- Restrições:

- Colisão por Professor: Esta restrição avaliará se um professor lecionará mais de uma aula ao mesmo tempo.
- Janelas: avalia se a grade horária formada possui alguma "janela" (horários vagos).
- Aulas Isoladas: verifica as aulas que estão presentes isoladamente na grade horária.
- Blocos de Disciplinas: Didaticamente, é interessante que as disciplinas estejam distribuídas em blocos de duas. Por exemplo, se a carga horária de uma disciplina é de cinco aulas semanais, o ideal é que elas estejam dispostas em dois blocos de duas aulas e uma aula isolada. Lembrando que essa restrição não é aplicável a todas as escolas.
- Vários Blocos por Dia: Baseado na restrição de blocos é preciso evitar que dois blocos estejam alocados num mesmo dia.
- Média de Aulas por Dia: Se uma turma tem vinte e cinco aulas semanais, o ideal é que tenha cinco aulas por dia. Com a avaliação desse critério evita que a turma tenha horários completos em alguns dias e dias com menos em outros.
- Preferência dos Professores: É necessário saber se o professor está lecionando num horário preferível.

- Criação da população inicial:

Neste trabalho, se escolheu a geração de uma população inicial aleatória. Para cada turma, existe uma lista com as disciplinas que devem ser ministradas. Estas são sorteadas e atribuídas em cada *slot* de cada indivíduo. Inicialmente, optou-se por uma população de 100 indivíduos.

- Avaliação da população:

Nesse trabalho, a avaliação de indivíduos adota uma abordagem baseada em penalidades. Para cada tipo de penalidade, um peso será atribuído que induzirá negativamente na seleção do indivíduo para a reprodução.

As restrições de inviabilidade e de qualidade avaliadas são:

- Choque de professores.
- Horário Inviável.
- Janelas no horário dos professores.
- Aulas isoladas.
- Horário indesejável.
- Aulas sequenciadas.
- Média de aulas.

A função de custo de uma solução *s*, a qual deve ser minimizada, pode ser calculada, pela seguinte expressão: *f(s) = g(s) + h(s)*.

A componente *g(s)*, que mensura o nível de inviabilidade (restrições fortes) de uma solução *s*, é avaliada com base na expressão:

$$g(s) = \sum_{l=1}^{K} \alpha_k \mathrm{I}_k \tag{4}$$

Na qual *K* é o número de medidas de inviabilidade, I_k é o valor da *k*-ésima medida de inviabilidade e α_k é o peso associado a essa *k*-ésima medida.

A componente *h(s)*, que mensura a qualidade (restrições fracas) de uma solução *s*, é avaliada com base na seguinte função:

$$h(s) = \sum_{l=1}^{L} \beta_l \varrho_l \tag{5}$$

Na qual *L* representa o número de medidas de qualidade, ϱ_l o valor da *l*-ésima medida de qualidade e β_l o peso associado a essa *l*-ésima medida. Deve ser observado que uma solução *s* é viável se e somente se *g(s)* = 0.

Observa-se que uma solução *s* é viável se e somente se *g(s)* = 0. Nas componentes da função *f(s)* os pesos dados às diversas medidas refletem a importância relativa de cada uma delas e, sendo assim, deve-se $\alpha_k \gg \beta_l \ \forall k, l$, de forma a privilegiar a eliminação das soluções inviáveis.

- Método de Seleção:

Para o processo de seleção foi utilizado o método torneio. Na implementação do método, são formados aleatoriamente grupos que realizam uma disputa entre si, onde o indivíduo mais apto é o vencedor e será selecionado para fase de reprodução. No final do processo, a metade dos indivíduos será selecionada para a fase seguinte.

- Operadores:

O crossover de um ponto de corte é utilizado, porém para esse problema é preciso ressaltar que uma regra deve ser respeitada: o número de aulas que cada disciplina possui deve se manter constante. Para tanto foi proposta uma solução que somente troca as disciplinas quando há relação entre o primeiro e o segundo indivíduo. Para casos que não há correspondência, eles são mantidos na mesma posição. Os indivíduos são selecionados dois a dois, com 60% de probabilidade de efetivamente realizarem o cruzamento. Partes dos indivíduos ainda são mantidas, embora possam ser modificados na fase de mutação.

A mutação utilizada nesse problema é realizada aleatoriamente. Duas disciplinas de uma mesma turma são sorteadas e seus horários trocados. Como o número de possibilidade de inversões de disciplinas é muito elevado (300 para 12 turmas), resolveu-se realizar duas mutações em cada geração.

Antes da realização dos operadores de mutação e cruzamento acima, o algoritmo utiliza o Elitismo para garantir que os dois melhores indivíduos sobrevivam na população seguinte.

- Critério de parada:

Este trabalho faz uso do número de gerações para encerrar o algoritmo. Inicialmente foi escolhido o número máximo de 10.000 gerações.

- Resultados e conclusões:

No trabalho analisado foi apresentada uma solução para um problema de geração de horário considerando a eliminação de janelas e horários isolados. Para uma solução de qualidade em tempo aceitável, foi usado o AG. Foi possível atestar a capacidade do algoritmo na eliminação dos choques e inviabilidade dos horários dos professores através de realizações de testes. Outros requisitos para um horário aceitável também foram tratados. Notou-se uma boa eliminação de janelas e aulas isoladas. Não foi possível comparar o horário gerado com um horário gerado manualmente, entretanto verificou-se também a satisfação da diretoria da EEPRA e dos professores com os resultados alcançados. Anteriormente, eram necessárias duas pessoas trabalhando 5 dias, com o algoritmo foi necessário apenas uma pessoa para a escolha do melhor horário para a escola. Houve redução do período de adaptação do horário gerado e diminuição de reclamações, caindo de 20 solicitações para 4 pedidos sendo que 2 ajustes manuais foram feitos. A utilização e aceitação das soluções encontradas demonstram a importância da generalização e a legitimidade da proposta atual.

Conclui-se que o trabalho fornece uma importante colaboração para a geração de horários escolares.

GERAÇÃO AUTOMÁTICA DE GRADE HORÁRIA USANDO ALGORITMOS GENÉTICOS: O CASO DA FACULDADE DE ENGENHARIA ELÉTRICA DA UFU

Autor: Cristiane Divina Lemes Hamawaki, 2005 – Universidade Federal de Uberlândia (UFU).
Tipo de Trabalho: Dissertação.

- Objetivo do trabalho:

Criação de um aplicativo compreensível para extração de informações importantes com relação à proposta escolhida, possuindo uma representação genética para o problema de geração automática de horário de uma instituição universitária, considerando a limitação dos recursos com base em suas informações, utilizando Algoritmos Genéticos para obter uma solução viável.

- Modelagem do problema:

O problema modelado foi relativo a alocação de horários da Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia (FEELT) - que possui um curso com cinco anos de duração, dividido em dez períodos, com um elenco de 114 disciplinas teóricas e 39 disciplinas práticas, com 45 professores disponíveis com aulas ministradas no período da manhã e tarde.

Um Alocador Automático de Grade Horária (AloGra) composto pelos seguintes componentes foi modelado para a FEELT: algoritmo genético, responsável pela implementação das operações genéticas (mutação, seleção, elitismo, cruzamento e cálculo de aptidão) e pelo processo como um todo; base de dados, local onde é armazenado os fundamentos necessários para o processamento do problema avaliado, lista de disciplinas, docentes e suas relações, restrições de dias e horários para cada aula, números de dias na semana e preferência de dias e horários para cada docente.

- Representação da solução:

Na implementação deste algoritmo, resolveu-se codificar o cromossomo como uma *structure* (estrutura) composta por *arrays* (vetores) e variáveis simples. Com isso, obteve-se um controle mais preciso dos dados, que se encontram estrategicamente dispostos. A *structure* é formada por quatro matrizes de tamanho 10 x 30, sendo duas correspondentes às disciplinas teóricas e duas às disciplinas práticas. Nessas matrizes são armazenados inteiros de 0 a 29 que são os *slots* alocados. Completando a codificação do cromossomo existem também as máscaras de rótulos. Estas máscaras são permanentes durante a execução do algoritmo e por isso não está ligada a estrutura cromossômica. As linhas da matriz correspondem aos períodos e suas colunas à posição das matrizes do cromossomo. Em seus elementos estão armazenados em inteiros respectivos ao código das disciplinas a serem alocadas.

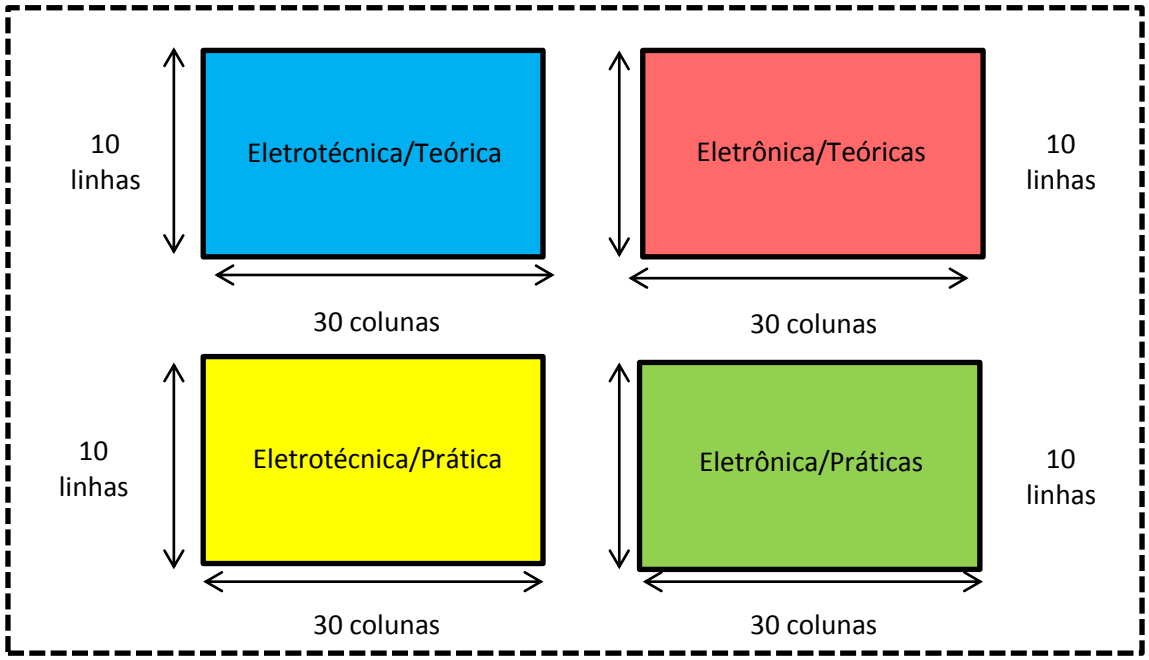

Figura 4.5: Codificação do cromossomo.

Fonte: (HAMAWAKI, 2005)

Para tornar mais clara a codificação do cromossomo, é ilustrada a seguir uma linha de matriz do cromossomo com uma linha de matriz de máscara de rótulo. Decodificando essa grade horária verifica-se que a disciplina representada pelo inteiro 35 ocupará os *slots* 2 e 3 e a disciplina 40 os *slots* 36, 37 e 38.

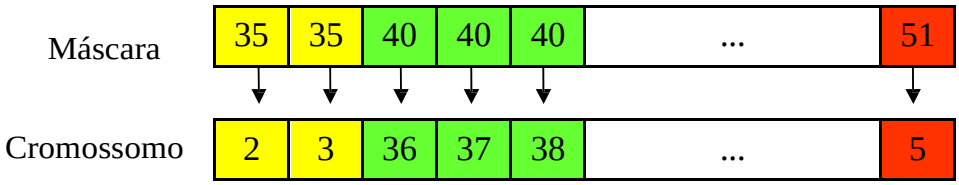

Figura 4.6: Equivalência máscara/cromossomo.

Fonte: (HAMAWAKI, 2005)

- Restrições:

- Leves: Ligadas as preferências dos professores de acordo com uma ficha preenchida pelos professores classificando suas preferências de horário. As penalizações associadas são baixas, obtidas diretamente da pontuação das preferências.

- Médias: Mais complexas que as leves, buscam soluções com a distribuição uniforme das aulas, evitar situações críticas, como o exemplo do professor que dá cinco aulas por semana e mesmo podendo fazer num dia só, é obrigado a ir a instituição todos os dias; ou alunos saindo de um lugar para outro, pois o tempo para troca de sala é pouco; aulas vazias ou "janelas" durante o período. As penalizações envolvidas são de peso médio ou alto.

- Severas: Estas que verificam a viabilidade de uma solução, avaliando o cumprimento de regras importantes como a verificação de carga horária e as disponibilidades dos professores. As penalizações são as mais altas, servindo como agente filtrador, descartando soluções não factíveis. As restrições abaixo são severas e devem ser atendidas em qualquer sistema de otimização de Grade Horária.

  - Cada professor e classe devem estar presentes na grade horária em um número pré-definido de horas.
  - Não deve existir mais que um docente nas mesmas classes em um mesmo horário.
  - Nenhum docente pode estar em duas classes no mesmo horário.
  - Não devem existir "horas abertas", horas em que nenhum professor foi elencado para uma classe.

Para o algoritmo analisado, foram consideradas as seguintes restrições:

- Cada professor deverá ter a melhor satisfação com os horários alocados a ele.
- O número de aulas por semana de cada disciplina deverá ser respeitado.
- Não deve haver aulas seguidas da mesma disciplina.
- O surgimento de “janelas” entre horários deverá ser evitado ao máximo.
- Diferentes disciplinas, ministradas pelo mesmo professor, não podem concorrer.

Também foram levantadas as seguintes limitações para o sistema:

- Método de matricula dos alunos: por disciplina a cada período.
- Turma: os alunos que cursam juntos uma mesma disciplina em um mesmo campus, curso e semestre.
- Não é importante a matricula de alunos por período letivo.
- Modelar apenas a matricula por disciplina.
- Um aluno pode se matricular em disciplinas de semestres diferentes, atendendo os pré-requisitos e número máximo de disciplinas semestrais.

Observações importantes sobre a implementação do sistema:

- A geração de grades horarias leva em conta vários fatores, entre eles, horários de trabalho dos professores. As disciplinas disponíveis no período são mais importantes.
- Um conjunto de disciplinas é oferecido a cada período (semestre).
- Para cada disciplina no semestre tem um horário agregado, fazendo com que cada aluno possua durante a semana, uma ou mais disciplinas.
- Dependendo da procura de matrículas para uma disciplina, podem ser abertas turmas a mais.

- Criação da população inicial:

A inicialização da população é feita atribuindo-se valores aleatórios aos elementos das matrizes correspondentes aos *slots* que as disciplinas irão ocupar. Apesar de a inicialização ser aleatória ela não gera soluções inválidas, o que comprometeria toda a evolução do algoritmo. O algoritmo não permite que disciplinas diferentes de uma mesma ênfase e período não ocupem o mesmo *slot*.

- Avaliação da população:

É definida por várias restrições cada uma com sua formula de penalidade, por exemplo, Disponibilidade do Professor, Janelas, Ocorrência de disciplinas por dia além do permitido, Ocorrência de um mesmo professor lecionando disciplinas diferentes em mesmo horário (por ser muito severa faz com que zere a aptidão).

$$apitidão = 2 * \sum_{i=0}^{n} disponibilidade(slot) - \left(\sum 10 * N_{janelas} 1000 * N_{agrupamentos}\right) \quad (6)$$

- Método de Seleção:

Nesse algoritmo é utilizado o método da Roleta e o Elitismo.

- Operadores:

O *crossover* utilizado pode ser entendido como uma fusão do *crossover* de um ponto de corte com um multiponto.

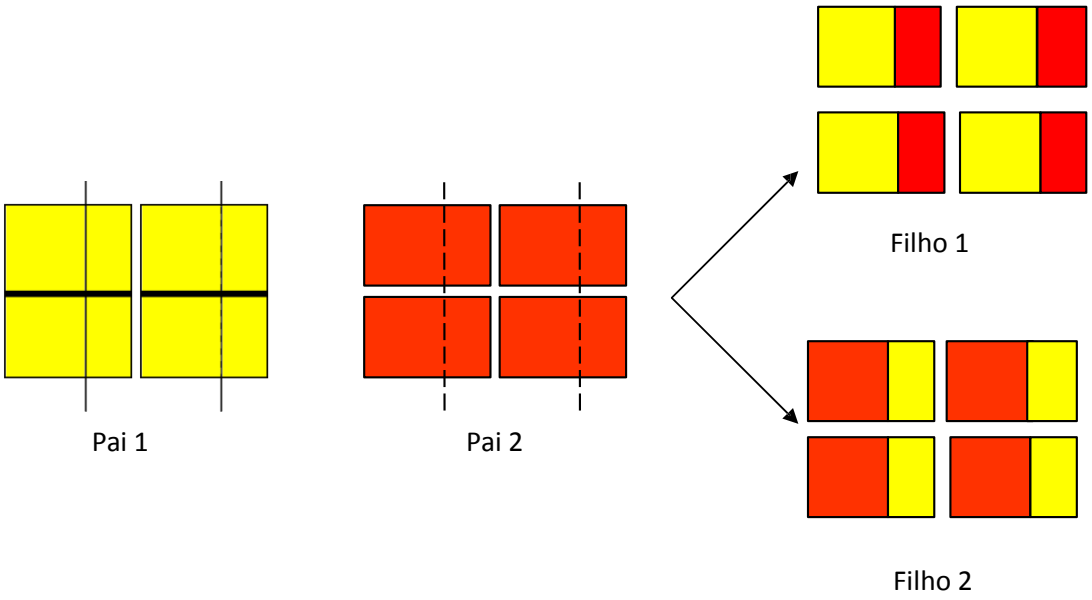

Figura 4.7: Exemplo de cruzamento.

Fonte: (HAMAWAKI, 2005)

Implementou-se a mutação trocando um bloco de gene por outro que não esteja reservado por aulas. O bloco sofre mutação aleatoriamente com base em parâmetros.

- Critério de parada:

  É utilizado o número de gerações.

- Resultados e Conclusões:

O aplicativo foi desenvolvido para uma instituição de ensino superior que possui restrições e problemas a serem resolvidos em relação à grade horária. A geração das grades horárias na instituição era realizada manualmente, sendo extremamente trabalhosa e custosa. A execução do aplicativo foi realizada em uma máquina com processador Pentium 4 de 1.5 GHz e 256MB de memória com duração de aproximadamente 3 horas para cada execução. Para tais execuções foram utilizados os seguintes parâmetros:

- Número de indivíduos: 100
- Número de gerações: 500
- Probabilidade de cruzamento: 60%
- Probabilidade de mutação: 1%

A ferramenta desenvolvida, além de reduzir o tempo para produzir o horário, tenta diminuir o esforço por parte da secretaria. A partir da análise dos resultados obtidos pode-se concluir que as técnicas que foram implementadas obtiveram uma solução satisfatória e eficiente para o problema.

Como principal contribuição desse trabalho pode ser mencionada o estudo e aplicação de Algoritmos Genéticos em conjunto com o problema de restrições, vindo como solução inovadora para o problema proposto. Tratou-se de um caso especifico da instituição FEELT – Faculdade de Engenharia Elétrica da UFU – Universidade Federal de Uberlândia – MG.

Como trabalho futuro sugeriu-se uma pesquisa entre as diversas faculdades de ensino superior da UFU de forma a definir o claramente o escopo do problema e os aspectos requeridos de uma grade horaria válida. Pode-se também sugerir a elaboração de novos

operadores genéticos que possam adaptar-se melhor à cada contexto e a verificação de desempenho com novas restrições.

PROTÓTIPO GERADOR DE GRADES HORÁRIAS PARA INSTITUIÇÕES DE ENSINO

Autor: Thomás Augusto Preis, 2007 - Universidade Regional de Blumenau (FURB).
Tipo de Trabalho: Monografia.

- Objetivo do trabalho:

Desenvolver uma aplicação com interface Web para gerar soluções de grades horárias de disciplinas para a Universidade Regional de Blumenau (FURB), utilizando Algoritmos Genéticos.

- Modelagem do problema:

A modelagem do problema abrangeu o desenvolvimento de uma aplicação para implementar uma estrutura com Algoritmos Genéticos visando a otimização de grades horárias, com interface WEB. As ferramentas computacionais usadas na modelagem foram software para construção automatizada de websites Code Charge Studio 3.2 (YES SOFTWARE, 2007), a linguagem de programação Java Servlets e o banco de dados MySQL 5.0. Como base de dados para o algoritmo foram usadas as grades originais da FURB referentes aos cursos de Ciência da Computação, noturno e matutino, Licenciatura em Computação, noturno e Sistemas de Informação, noturno do ano 2007/1.

- Representação da solução:

Foi definida uma estrutura tridimensional para o cromossomo (Figura 4.8). O eixo *z* representa os semestres, o eixo *x* representa os dias semanais e o eixo *y* os horários de cada aula. Cada elemento da matriz é um gene, que possui dois alelos, um para indicar qual é o professor alocado e o outro para indicar a disciplina.

O esquema de matriz 3D (tridimensional) garante que não haja colisão de horários para um mesmo professor, num semestre. Entretanto, não se pode garantir que não ocorram colisões para um professor em semestres diferentes.

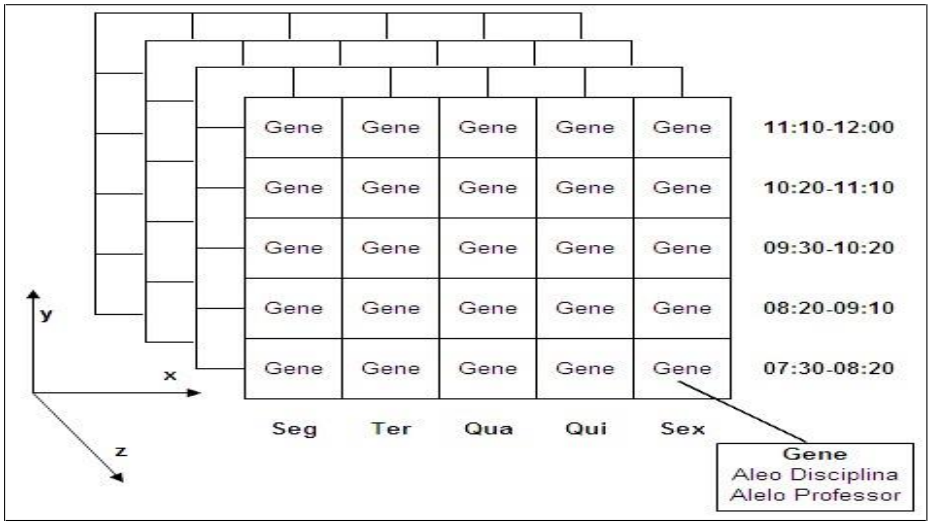

Figura 4.8: Representação do cromossomo escolhida.
Fonte: (PREIS, 2007)

Para representar todas as soluções que envolvem o problema de alocação de horário presente foram construídas, em Java, as classes Cromossomo, Gene e Alelo.

Na Figura 4.9 é apresentado o código em Java referente a classe Cromossomo:

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

Figura 4.9: Representação do cromossomo na linguagem Java.
Fonte: (PREIS, 2007)

Os genes possuem dois alelos cada. A representação em linguagem Java da classe Gene é apresentada a seguir:

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

Figura 4.10: Representação do gene na linguagem Java.
Fonte: (PREIS, 2007)

O código em Java referente a classe Alelo é apresentado na Figura 4.11. Na classe Alelo ficam armazenados os valores disciplina e o professor de cada um dos genes do cromossomo.

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

Figura 4.11: Representação do alelo na linguagem Java.
Fonte: (PREIS, 2007)

- Restrições:

  Algumas restrições foram citadas como base para criação da população inicial:

  - Professor e Disciplinas
  - Grade Horária Completa
  - Disciplinas repetidas num mesmo dia

- Criação da população inicial:

A criação da população não é totalmente aleatória foram respeitadas algumas restrições antes de alocar os genes e alelos na população de cromossomos. Restrições citadas anteriormente.

- Professor e disciplinas: cada novo cromossomo receberá apenas genes e alelos de disciplinas e professores que estejam aptos a ministrarem as disciplinas. Garantindo que não haja inconsistências, como exemplo, um professor de álgebra linear alocado na disciplina de compiladores.
- Grade Horária Completa: através da restrição da grade horária completa é garantido que cada um dos semestres possua todas as disciplinas especificas para cada semestre. Com isso, garante que não existam semestres que possuam todas as disciplinas e/ou cargas horárias incompletas.
- Disciplinas Repetidas num Mesmo Dia: são criados cromossomos que garantem que todas as aulas de uma mesma disciplina não sejam fixadas em um dia da semana. Essa condição pode ser perdida devido a cruzamento e mutações.

- Avaliação da população:

A função aptidão usada para avaliar os cromossomos foi a seguinte:

$$f(x) = \frac{1}{1 + Número\ de\ Penalidades} \tag{7}$$

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

- Método de Seleção:

O método escolhido para seleção foi o método da Roleta.

- Operadores:

Como *crossover* foi utilizado dois métodos para efetuar os cruzamentos: O cruzamento de um ponto e o de dois pontos.

A mutação foi aleatória e aplicada em um único indivíduo.

- Critério de parada:

O processo de varredura em todos os membros da população, a procura de um cromossomo ótimo é repetida até que o algoritmo encontre um indivíduo ótimo ou atinja o máximo de gerações limitadas previamente Caso não sejam satisfeitas as condições necessárias para finalizar a execução, o algoritmo reiniciará o processo a partir da etapa de avaliação da população.

- Resultados e conclusões:

O algoritmo implementado se mostrou funcional na geração da solução sem ocorrência de colisão nos horários. O tempo de execução e a qualidade da solução varia com alguns fatores:

- Tamanho da População: quanto maior o tamanho da população, maior o espaço de busca e maior o tempo de processamento do algoritmo. Com uma população menor, o espaço de busca é reduzido e algoritmo será bem mais rápido. Foi verificado através de testes, que com uma população em torno de 250 membros, o algoritmo mostrou-se mais eficiente.

- Coeficiente de Mutação: A configuração do parâmetro de mutação influenciou o desempenho e qualidade das soluções. Foi verificado que muitas mutações provocam uma queda no desempenho do AG e geram filhos que não se assemelham tanto aos pais, o que pode comprometer a procura da busca do indivíduo ótimo. Já com poucas mutações, o algoritmo não consegue evitar as colisões de horário. De acordo com testes realizados, deve-se usar um valor mediano para a taxa de mutação.

- Numero de gerações: Determina qual o número máximo de iterações que o algoritmo irá executar. Quanto maior o número de iterações, maior o número de populações geradas,

maior o espaço de busca, maior o tempo de execução do algoritmo e maior a possibilidade de alcançar a solução do problema.

Para os testes foram utilizados quatro cursos, 118 disciplinas e 53 professores. Tendo normalmente um professor vinculado a cada disciplina. Foram avaliados testes com número de gerações entre 10.000 e 30.000 e o número de população variando de 100 a 500 indivíduos.

O algoritmo genético mostrou ser extremamente eficiente na geração de grades horárias sem ocorrência de colisões, mesmo quando sujeitado a testes reais. Porém não mostrou ser igualmente eficiente na geração de soluções que levam em conta as restrições de horárias impostas pelos professores. Também ocorreram problemas relacionados às mutações, algumas aulas de uma mesma disciplina, num mesmo dia, não ficaram agrupadas.

Como trabalhos futuros, foram sugeridos pelo autor:

(i) a correção do problema gerado pelas preferências de horários dos professores, alterando o código para a utilização de apenas um coeficiente no grau de aptidão;

(ii) o aperfeiçoamento das funções de mutações para solucionar o problema de agrupamento de horários;

(iii) a inclusão de novas funcionalidades para fazer a distribuição de salas de aulas, laboratórios e equipamentos, através de alterações nas estruturas dos cromossomos, adicionando mais alelos às características.

SISTEMA DE ALOCAÇÃO DE HORÁRIOS DE CURSOS UNIVERSITÁRIOS: UM ESTUDO DE CASO NO DEPARTAMENTO DE COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DE SERGIPE

Autor: F. Vieira, H. Macedo, 2011 – Universidade Federal de Sergipe (UFS).
Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

Descrever o sistema de geração automática de grade de horários desenvolvido para atender as necessidades do Departamento de Computação (DCOMP) da Universidade Federal de Sergipe (UFS), especificamente do curso de Sistemas de Informação. O problema foi definido como uma busca local em espaço de estados e utilizou a metáfora da seleção natural e evolucionária, representada através de Algoritmos Genéticos para que a melhor solução fosse encontrada no final do processo. A abordagem adotada permite a reprodução da solução para atender restrições de outros cursos e de outras instituições.

- Modelagem do problema:

O DCOMP é composto por três cursos: Ciência da Computação, Sistemas de Informação e Engenharia da Computação. O departamento possui 28 professores, entre permanentes e substitutos, que ministram disciplinas nesses 3 cursos. Atualmente o DCOMP conta com 650 alunos, que distribuídos entre os três turnos.

A modelagem através de Algoritmos Genéticos foi usada para facilitar o processo de análise e verificação da validade de uma grade horária de um período específico pela coordenação do DCOMP da UFS, evitando transtornos na criação de turmas extras, e horários conflitantes, entre outros problemas.

- Representação da solução:

O modelo cromossômico de grade horária adotado no trabalho está ilustrado na Figura 4.12. Cada cromossomo é composto por todas as turmas que deverão ser ofertadas no período em vigência. Por si só, esta definição evita que seja violada a restrição de que *todas as disciplinas obrigatórias do período vigentes devem ser ofertadas*. Cada gene representado por uma coluna, armazena informações sobre uma turma: disciplina, professor responsável, horários de aula e número de vagas. O fato dos horários já estarem presente no gene faz com que seja atendida a restrição de que *todas as disciplinas ofertadas devem preencher o número de horas estabelecidas pela estrutura curricular*.

Disciplina	Disciplina	Disciplina	Disciplina	Disciplina	Disciplina		Disciplina
Professor	Professor	Professor	Professor	Professor	Professor	...	Professor
Horário[ ]	Horário[ ]	Horário[ ]	Horário[ ]	Horário[ ]	Horário[ ]		Horário[ ]
Vagas[ ]	Vagas[ ]	Vagas[ ]	Vagas[ ]	Vagas[ ]	Vagas[ ]		Vagas[ ]

Figura 4.12: Modelo cromossômico de uma grade horária.
Fonte: (VIEIRA, MACEDO, 2011)

- Restrições:

  Foram representados dois tipos de restrições, a *hard* e a *soft*.

  Restrições *hard*:

  - Todas as disciplinas ofertadas devem preencher o número de horas por semana estabelecidas pelo currículo.
  - O professor não pode lecionar em duas turmas diferentes num mesmo dia e horário.
  - Aulas de uma mesma turma não devem acontecer em um mesmo dia e horário.
  - Todas as disciplinas obrigatórias do período atual devem ser ofertadas.
  - A oferta das disciplinas deve obedecer ao turno dos cursos.

  Restrições *soft*:

  - Todas as aulas de uma turma devem ser ofertadas, preferencialmente num mesmo horário durante a semana.
  - Aulas de uma mesma turma não devem ser ofertadas em dias seguidos, nem em um único dia.
  - A preferência do professor em optar por lecionar apenas disciplinas de seu ‘ interesse deve ser respeitada.

- Criação da população inicial:

Os indivíduos da população inicial foram criados com base na lista de todas as turmas que devem ser ofertadas no período vigente. Informações sobre professores e horários da

disciplina são geradas aleatoriamente em cada indivíduo de forma a criar variabilidade genética na população. O professor é adicionado através da lista oficial de docentes do departamento e o vetor de horários é preenchido através de valores inteiros pré-definidos.

Tabela 4.2: Representação inteira dos horários semanais.

Horário	Segunda	Terça	Quarta	Quinta	Sexta
07:00 às 09:00	0	1	2	3	4
09:00 às 11:00	5	6	7	8	9
11:00 às 13:00	10	11	12	13	14
13:00 às 15:00	15	16	17	18	19
15:00 às 17:00	20	21	22	23	24
17:00 às 19:00	25	26	27	28	29
19:00 às 21:00	30	31	32	33	34
21:00 às 23:00	35	36	37	38	39

Fonte: (VIEIRA, MACEDO, 2011)

- Avaliação da população:

Os indivíduos foram avaliados através da Equação (8), que calcula o somatório de um conjunto de valores de bonificação e penalização.

$$f(x) = \sum bônus - \sum penalizações \qquad (8)$$

Os valores dos pesos dos bônus e penalizações, especificados na Tabela 4.3, foram definidos considerando sua importância em conversas com especialistas.

Tabela 4.3: Pesos utilizados para bonificações ou penalizações.

Restrição	**Bônus/Penalidade**	**Peso**	**Multiplicador**
Disciplina atende horário do curso	Bônus	3	1 * aula no horário
Evitar choque entre aulas de mesma turma	Bônus	5	1 * turma sem choque

Professor dar aula em dois lugares ao mesmo tempo	Penalidade	3	1 * choques encontrados
Aulas de uma turma devem ser ofertadasno mesmo horário	Bônus	2	1 * horários queatendem a
Aulas de uma turma não devem ser ofertadas em dias consecutivos, nem nomesmo dia.	Penalidade	2	1 * horários que não atendem a restrição
Professor optar por disciplinas de seu interesse	Bônus	5	1 * disciplinas do interesse do professor que ele leciona

Fonte: (VIEIRA, MACEDO, 2011)

- Método de Seleção:

  Método da Roleta Russa.

- Operadores:

  É utilizado o *crossover* de um ponto de corte.

  Utiliza mutação simples.

- Critério de parada:

  Por número máximo de gerações.

- Resultados e conclusões:

Realizou-se o teste com o sistema através de dois grupos de testes. Um com uma lista com 6 professores, juntamente com as disciplinas que mais os interessavam, 12 turmas para serem ofertadas entre os 3 cursos, focando as turmas do primeiro período de Sistemas de Informação. O outro grupo, tem 8 professores ofertando 38 turmas do departamento de

computação, dando maior ênfase às disciplinas de Ciência da Computação que é o curso que possui maior demanda por vagas.

Os parâmetros utilizados foram os seguintes:

- População de 750 indivíduos;
- 200 Gerações;
- Todos os indivíduos estão disponíveis para seleção;
- Taxa de *crossover* de 95%;
- Taxa de mutação de 5%;
- Método de seleção escolhido foi o da roleta;
- *Crossover* com apenas um ponto de corte;
- Não existe pré-processamento para eliminar indivíduos anômalos.

Os resultados obtidos nos testes mostraram o potencial da solução proposta. O software baseado em AG será usado no DCOMP, apenas durante alguns períodos, como auxilio ao método manual, já em funcionamento. Após a comprovação definitiva de sua eficácia, poderá vir a ser utilizado de forma oficial no DCOMP.

Em trabalhos futuros é interessante a utilização de outras técnicas de otimização, comparando-as aos Algoritmos Genéticos. É possível a realização de análise em outros departamentos, de forma a generalizar uma solução integral para a instituição. Outra linha de trabalho é a criação de um módulo para análise da retenção de alunos, servindo como dados de entrada para o sistema.

DESENVOLVIMENTO DE UM ALGORITMO GENÉTICO PARA A RESOLUÇÃO DO TIMETABLING

Autor: Guilherme Tadeu Silva Timóteo, 2002 – Universidade Federal de Lavras (UFLA).
Tipo de Trabalho: Monografia.

- Objetivo do trabalho:

Implementar um Algoritmo Genético para resolução do problema de *Timetabling*, e analisar seu comportamento, realizando testes com diversos parâmetros.

- Modelagem do problema:

Um modelo usando Algoritmos Genéticos foi desenvolvido para resolver o problema de alocação de grade horária do Colégio Nossa Senhora de Lourdes, uma escola de ensino fundamental e médio localizada em Lavras-MG.

- Representação da solução:

Cada cromossomo é representado por uma matriz de 3 dimensões (Número de Turmas X [Número de Dias X Número de Horários]), no qual cada gene representa um *slot*. A representação adotada tem a vantagem de não permitir que duas disciplinas referentes a mesma turma ocupem o mesmo horário. Mas não garante por si só a ocorrência de colisões de professores e outras restrições. As Figuras 4.13 e 4.14 ilustram respectivamente as representações de 1 e de *N* individuo(s) ou cromossomo(s).

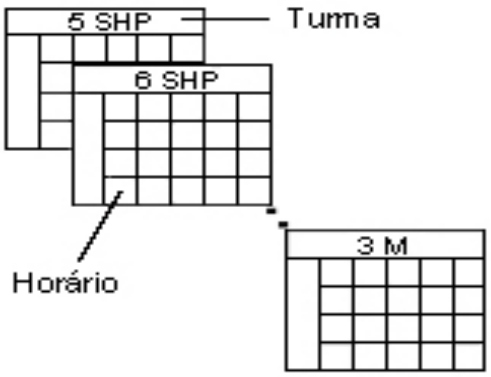

Figura 4.13: Representação da solução.
Fonte: (TIMÓTEO, 2002)

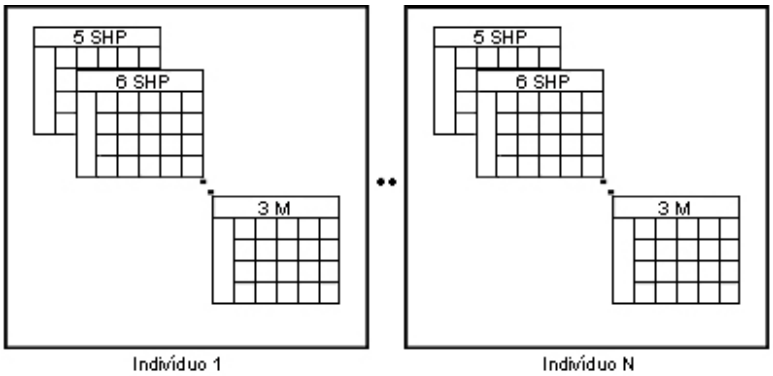

Figura 4.14: Representação da população de soluções.

Fonte: (TIMÓTEO, 2002)

- Restrições:

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

- Criação da população inicial:

  Neste trabalho foram utilizados dois operadores para geração:

  - Geração Aleatória: Percorre uma lista de uma determinada turma e para cada disciplina sorteia um *slot* na grade horaria. O processo continua até que as listas de disciplinas de todas as turmas sejam percorridas

  - Geração Heurística: De forma semelhante, seleciona blocos de duas disciplinas ao invés de uma. Essa geração é efetuada com auxílio de listas. A lista de disciplinas de uma determinada turma é dividida em duas outras listas: a lista de disciplinas em blocos disponíveis par sorteio e alista de disciplinas isoladas. Primeiro são sorteados os blocos de disciplinas. A seguir são sorteadas as disciplinas isoladas, que ocuparão cada uma um *slot*.

- Avaliação da população:

A aptidão dos indivíduos da população é avaliada pela seguinte equação:

$$f(x) = \frac{1}{1 + penalidades} \tag{9}$$

Ocorrendo violações de restrições, o valor da variável *penalidades* pode ser incrementado, diminuindo o valor da função de aptidão do indivíduo,

- Método de Seleção:

Foram implementados os métodos da roleta giratória, o método da seleção por torneio e o método da roleta com redução.

- Operadores:

Implementaram-se três tipos de *crossover* o com corte em um ponto (1PX), o com corte em dois pontos (2PX), e o heurístico com corte em um ponto.

Foram implementados dois tipos de mutação: (i) a comum, onde dois genes são trocados aleatoriamente, mas a troca só ocorre num indivíduo- ou seja, são selecionadas duas posições dentro do indivíduo e efetua-se a troca e (ii) a heurística, onde é feito sorteio e a troca de dois blocos dentro de um indivíduo. Caso não existam blocos disponíveis, efetuar a mutação simples. Dependendo do nível de importância é preferível quebrar um bloco a entrar em conflito com a preferência de um professor ou outro critério qualquer.

- Critério de parada:

Foi escolhido como critério para finalização do algoritmo genético o número de gerações.

- Resultados e conclusões:

A implementação dos métodos de seleção, dos operadores de *crossover* e mutação e 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, é necessário efetuar parametrizações no algoritmo, configurando o mesmo de acordo com o problema.

Para validar o sistema desenvolvido, foram realizados testes de geração de horários escolares para uma instituição de ensino de ensino fundamental e médio, chamada Colégio Nossa Senhora de Lourdes, localizada em Lavras – MG. O interesse principal consistiu em gerar boas grades horárias de acordo com os parâmetros estabelecidos pela instituição.

Nos testes foram usados cadastros de 7 turmas, de 19 professores, suas preferências de horário, suas disciplinas e também as turmas onde cada professor leciona determinada disciplina e disponibilizados 30 horários semanais em 5 dias da semana, sendo 6 em cada dia.

Foram avaliados testes para todos os métodos de seleção explicados na monografia, alterando seus parâmetros diversas vezes para tentar conseguir um resultado ótimo, com o limite máximo de 15000 gerações por teste. Os testes também envolveram os operadores de *crossover* 1PX, *crossover* 2PX, *crossover* heurístico, mutação comum e mutação heurística. Os parâmetros de cada operador e o tamanho da população foram alterados de teste para teste.

- Seleção por Torneio (população variando de 80 a 120).
- Seleção da Roleta Giratória (população variando de 80 a 160)
- Método da Roleta com Redução (população variando de 120 a 160)

Os testes foram realizados baseados em apenas um problema. Para que o nível de confiança dos parâmetros aumente é preciso abordar vários problemas e realizar uma comparação entre eles. O problema abordado pode estar favorecendo um método de seleção ou crossover ou outro parâmetro. Por outro lado, se o algoritmo for bem "calibrado" ele pode gerar ótimos resultados. Nos testes realizados nesse trabalho observou-se que o método de seleção por torneio, o operador de crossover com corte em um ponto e operador de mutação comum foram os que apresentaram os melhores resultados.

RESOLVENDO PROBLEMAS DE ALOCAÇÃO DE TIMETABLING USANDO ALGORITMOS GENÉTICOS

(*Solving Timetable Scheduling Problem by Using Genetic Algorithms*)

Autores: Branimir Sigl, Marin Golub, Vedran Mornar, 2003 - University of Zagreb.
Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

  Resolver o problema de alocação de *timetables* usando Algoritmos Genéticos.

- Modelagem do problema:

Um modelo baseado em Algoritmo Genéticos foi desenvolvido pelos autores para resolver o problema de *Timetabling* e testado na Faculdade de Engenharia Elétrica e Computação (FER) em Zagreb. A interface do programa foi desenvolvida em C#. Já o núcleo do AG foi desenvolvido em C++ com o apoio da biblioteca STL (*Standard Template Library*).

- Representação da solução:

A alocação de horários de uma *timetable* também pode ser representada como uma classe especial de problemas de corte em 3D. A *timetable* poderia ser apresentada como uma estrutura 3D. As dimensões dos horários 3D são: dias (eixo x), *slots* de tempo (eixo y), e salas (eixo z). As classes são apresentadas como cubos os quais devem ser colocados numa estrutura *timetable* (Figura 4.15).

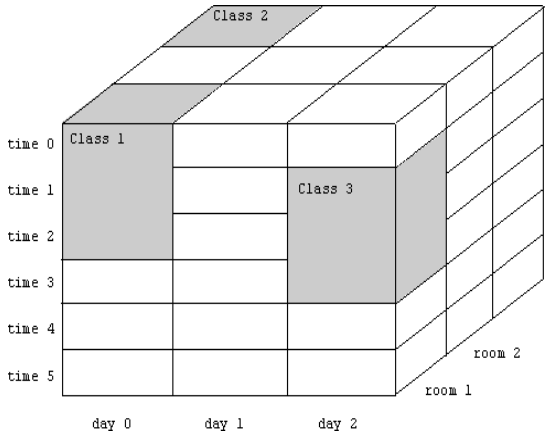

Figura 4.15: Estrutura 3D apresentada de uma *timetable*.
Fonte: (SIGL et al., 2003)

A alocação de horários é um processo de colocar esses cubos numa *timetable*, de forma que não haja restrições como, por exemplo, classes conflitantes (que repartiriam o mesmo recurso, um grupo de estudantes ou um instrutor) são colocadas num mesmo *slot* de tempo. O processo de alocação de horário numa *timetable* pode ser formalmente definido com variáveis binárias x_{cdtrgi} que tem o valor de 1 se e somente se um instrutor *i* ministra a classe *c* no dia *d* no tempo *t*, para um grupo *g* na sala *r*.

- Restrições:

A *timetable* deve satisfazer as seguintes condições:

- O grupo *g* pode participar apenas de uma classe de cada vez.
- Instrutor *i* pode ensinar apenas uma classe em um determinado tempo.
- Na sala *r* apenas uma classe pode ter aula por vez.
- Todas as aulas devem ser mantidas exatamente uma vez.

Na aplicação, a interface gráfica facilita a entrada de dados para cada classe definindo:

- Dias e horários em que a classe pode ser colocada.
- Salas onde a classe pode ser colocada.
- Número de salas ocupadas por uma classe simultaneamente.
- Grupos de alunos que frequentam a classe.

- Instrutores que ensinam a classe.

- Criação da população inicial:

Cada indivíduo na população representa uma *timetable*. O algoritmo inicia de uma *timetable* inviável e tenta obter uma timetable *viável*. Toda classe deve ser colocada apenas uma vez na estrutura de *timetable* 3D. Isto pode ser assegurado pela criação de uma nova restrição para cada classe que deve ser agendada. Como toda classe pode ter apenas uma variável definida em 1, indivíduos podem ser gerados de modo que todo gene em um indivíduo represente uma classe. O valor do gene será um ordinal de uma variável binária pertencente aquela classe.

- Avaliação da população:

O valor de *fitness* de um indivíduo é calculado através da equação:

$$fitness(individual) = (Número\ de\ conflitos) * K + Qualidade \quad (10)$$

Na qual K é uma constante.

- Métodos de seleção:

O método da seleção por Roleta foi utilizado para os pais. Os valores cumulativos de *fitness* individual são calculados pela equação:

$$q_k = \sum_{i=1}^{k} fitness(individual_i), \qquad D = max(q_k) \quad (11)$$

Na qual k = 1, 2, ... TAMANHO_DA_POPULAÇÃO.

O algoritmo gera um número randômico *r* do intervalo (0, *D*) e seleciona um indivíduo que satisfaz a condição:

$$\max(q_k) \leq r \quad (12)$$

- Operadores:

Nesse caso utilizou-se o *crossover* uniforme que testa todos os genes de ambos os pais. Se os pais têm valores iguais de um gene, este valor é escrito para o filho. Se os valores dos genes pais são diferentes, então o algoritmo escolhe aleatoriamente um pai como dominante e usa seu gene. O método da roleta é usado para selecionar o pai.

Também se utilizou a mutação aleatória, com uma alteração onde o algoritmo itera por todo o indivíduo da população. Para cada gene um número aleatório no intervalo (0, 1) é gerado. Se o valor gerado é menor do que a probabilidade de mutação especificada, o gene muda o valor para um valor randômico que denota um valor de dia-hora diferente ou uma combinação de sala.

Tamanhos de população - pequeno (clássico) e grande (melhorado) - e operadores melhorados para ambos os tamanhos foram testados.

- Critério de parada:

Foi utilizado o conceito da convergência para finalizar os testes. O algoritmo com operadores melhorados apresentou resultados muitos melhores que o algoritmo com operadores básicos. Convergência mais rápida e poucos conflitos foram obtidos em um tempo menor de processamento.

- Resultados:

O algoritmo foi testado em problemas de alocação em grades horárias de pequeno e grande porte. O principal problema enfrentado foi o tamanho da grade horária completa da FER para o semestre de outono. Uma grade horária de pequeno porte foi obtida a partir da grade horária de grande porte com exclusão de 70% das classes do processo de agendamento. O problema da grade horária de pequeno porte foi resolvido sem conflitos. Ao resolver o problema da grade horária de grande porte, o algoritmo básico parou em cerca de 95 conflitos. O desempenho do algoritmo foi significativamente melhorado com a modificação dos

operadores genéticos básicos, que coíbem a criação de novos conflitos no indivíduo. Com operadores inteligentes, o algoritmo atingiu cerca de 20 conflitos.

Tabela 4.4: Tamanho do problema.

	MENOR	**MAIOR**
Classes / Tamanho do Indivíduo	227	770
Salas	27	41
Grupos	55	114
Instrutores	46	157
Número de variáveis binárias	16103	35026
Número de fronteiras	4345	10898
Tamanho da população (clássica)	256	256
Tamanho da população (aprimorada)	5120	5120

Fonte: (SIGL et al., 2003)

O problema inicial de agendamento com um número grande de variáveis binárias foi reduzido para um tamanho aceitável eliminando certas dimensões do problema e incorporando essas dimensões em restrições. O agrupamento de diversas variáveis binárias em um valor de um único gene reduziu significativamente o tamanho do indivíduo. Agora é possível tentar resolver o problema de tamanho completo (Problema de toda agenda da FER) com uma abordagem com AG. Tal representação do problema de agendamento atinge a velocidade do algoritmo aceitável, assim problemas de pequeno porte são resolvidos em dezenas de segundos. Melhorias significativas foram conseguidas através de operadores inteligentes. O algoritmo inteligente converge muito mais rápido do que o algoritmo básico e representa um bom ponto de partida para resolver o problema completo.

Para resolver completamente o problema da escala completa, melhorias no algoritmo terão que ser feitas. Ao gerar restrições ela pode ser útil para registrar cada um, de forma que nenhuma uma restrição será definida (e verificada) duas vezes. Os indivíduos devem ser gerados de tal maneira que as classes que são mais difíceis de agendar ocupem os genes da frente de um indivíduo, enquanto as classes mais fácies de agendar ocupariam a parte traseira. Isso seria útil para um operador de *crossover* inteligente que define e verifica os conflitos de frente para trás de um indivíduo. Além disso, a abordagem de computação paralela poderia ser tentada, portanto, a verificação de espaço do problema poderia ser ampliada. Cada *thread* poderia começar com populações iniciais diferentes e a quantidade de soluções deverá ser maior.

ALGORITMO GENÉTICO COM ESTRATÉGIAS DE BUSCA EM BANCO PARA O PROBLEMA DO TIMETABLING EM CURSOS DE UNIVERSIDADE

(*Genetic Algorithm with Search Bank Strategies for University Course Timetabling Problem*)

Autor: A. Araisa Mahiba, C. Anand Deva Durai, 2012 - Karunya University.
Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

Desenvolver um novo método baseado em Algoritmo Genético com estratégias de busca em banco, local, guiada e busca tabu para resolver problemas de Timetabling em Cursos de Universidade.

- Modelagem do problema:

O problema de Timetabling em cursos de Universidade foi modelado através de um novo método de algoritmo genético com estratégia de pesquisa em banco, usando busca tabu, local e guiada. Busca local foi usada para aumentar a prole ou soluções. Busca guiada foi usada para limitar as soluções pelo uso de estrutura de dados de eventos. Busca tabu foi utilizado para remover as soluções usadas.

- Representação da solução:

É representada como um vetor com genes específicos a situação do problema da universidade.

Funcionário	Matérias	Estudantes	*Timeslot*	Sala de Aula

Figura 4.16: Representação da solução de um indivíduo.
Fonte: (MAHIBA, DURAI, 2012)

- Restrições:

  Restrições *hard:*

  - Um funcionário não será atribuído a dois ou mais classes de estudantes no mesmo de intervalo de tempo.
  - Dois ou mais funcionários não devem ser atribuídos a uma classe de estudante no mesmo *timeslot*.
  - Dois ou mais salas não devem ser atribuídos a uma classe de alunos no mesmo *timeslot*.

  Restrições *soft:*

  - Funcionários não devem ter três aulas consecutivas de *timeslot*, exceto seções de laboratório.
  - Para *timeslot* consecutivos, uma turma de alunos não deve mudar suas salas de aula.
  - Todos os *timeslots* devem ser ocupados.

- Criação da população inicial:

O cromossomo é dividido em 5 genes nomeados de Funcionário, Matérias, Estudantes, *Timeslot* e Sala de Aula (Figura 4.16). Cada valor de um gene é um inteiro, que pode ser mapeado para qualquer um deles. O primeiro gene do cromossomo representa o Funcionário, o segundo representa Matéria e o quinto representa Sala de Aula. Cada gene terá diferentes limites. Isto é limite inferior e superior. Por exemplo, o gene Funcionário tem limite inferior 0 e superior 30. Os limites precisam ser configurados antes de gerar a população. Uma vez que toda a configuração é feita, a população inicial é criada. A população inicial será criada aleatoriamente. Ou seja, cada gene em cada cromossomo será atribuído um valor aleatório entre os limites antes configurados.

- Avaliação da população:

A função de *fitness* é calculada de acordo com a seguinte equação:

$$f(s) = \sum HCS + \sum SCS \quad (12)$$

Em duas etapas:

- Cálculo do peso de satisfação de restrições *hard* (HCS).
- Cálculo do peso de satisfação de restrições *soft* (SCS).

Em primeiro lugar todas as soluções potenciais são avaliadas. As soluções que satisfazem as restrições *hard*, são chamadas de soluções viáveis. As soluções viáveis que satisfazem a restrição *soft* são consideradas soluções aptas para serem selecionadas para as próximas gerações.

- Método de Seleção:

A seleção é feita através da Busca Guiada.

Através do método de Busca Local ao procurar um cromossomo que tenha um valor de aptidão baixo, ele é selecionado e um gene aleatório sofre mutação dentro dos valores limites definidos para cada tipo de gene.

- Operadores:

Crossover e a mutação são feitos através de busca local. Crossover é chamado de busca local porque é usado para aumentar o número de novos cromossomos.

No *crossover* dois tipos de movimentos de vizinhança são utilizados. Em primeiro lugar estão os cromossomos pobres, dois cromossomos são selecionados aleatoriamente para passar por *crossover* escolhendo aleatoriamente um gene para trocar com outro gene entre dois cromossomos. Assim dois novos cromossomos são gerados. A próxima vizinhança será selecionada três cromossomos aleatoriamente. Aqui também um gene vai ser escolhido aleatoriamente e esse valor vai ser trocado entre os três cromossomos produzindo assim três

novos cromossomos. Se os cromossomos estão aptos, serão em seguida selecionados para a próxima geração para produzir um bom resultado de uma *timetable*. Caso contrário, sofrerá mutação que é outro meio para gerar novos cromossomos. Mutação é chamada de busca local porque ela produz novas soluções. Através do método de Busca Local ao procurar um cromossomo que tenha um valor de aptidão baixo, ele é selecionado e um gene aleatório sofre mutação dentro dos valores limites definidos para cada tipo de gene.

- Critério de parada:

Para finalização do algoritmo um tempo máximo para cada tipo de problema é definido. O de tamanho pequeno ficou com uma margem de 10-12 segundos. O médio de 40-45 segundos e o grande com 60-70 segundos.

- Resultados:

Esse novo método faz uso de Busca Local, Guiada e Tabu para fazê-lo mais eficiente. A Busca Local é utilizada para produzir o máximo de novas possíveis soluções usando mutação e crossover. Busca guiada é usada para direcionar a procura usando a estrutura de dados Eventos para melhor solução. Busca tabu é utilizada para remover soluções usadas e evitar o mínimo local. Todos esses métodos foram implementados para entradas de larga escala e alcançaram resultados promissores. Três tipos de teste, de pequeno, médio e grande porte foram realizados com parâmetros variados. Testes com a população variando de 120 a 250, apresentaram melhores resultados para problemas de grande porte. Os trabalhos futuros podem trabalhar no desenvolvimento de novos operadores genéticos e na adição de restrições *hard* e *soft*, tanto quanto possível, de acordo com as normas e necessidades da universidade.

USANDO ALGORITMOS GENÉTICOS PARA RESOLVER O PROBLEMA DE TIMETABLING DAS ESCOLAS SUL-AFRICANAS
(*Using Genetic Algorithms to Solve the South African School Timetabling Problem)*

Autores: Rushil Raghavjee, Nelishia Pillay, 2010 - University of KwaZulu-Natal.
Tipo de Trabalho: Artigo.

- Objetivo do trabalho:

Aplicar um AG para resolver o problema de *School Timetabling* de escolas primária e de ensino médio sul-africanas.

- Modelagem do problema:

O algoritmo genético usado para resolver o problema de *timetabling* de escolas sul-africanas modela o problema em duas fases. Na primeira fase, a modelagem se concentra na produção de horários viáveis, enquanto na segunda melhora a qualidade das *timetables* encontrados durante a primeira fase.

- Representação da solução:

Cada indivíduo na população é um *timetable*, que é representado como uma matriz 2D, com linhas correspondendo aos períodos e colunas correspondendo as classes.

A intersecção de cada linha e coluna contém o professor que vai lecionar para a classe durante este período particular.

	C1	C2	...
P1	T2	T1	...
P2	T7, T4	T7, 74	...
P3	T2	T3	...
...	...	...	...

Figura 4.17: Um exemplo de um elemento da população.
Fonte: (RAGHAVJEE, PILLAY, 2010)

- Restrições:

Existem restrições para as duas escolas. A para escola primária são restrições *hard* que deverão ser satisfeitas:

- Todos os requisitos devem ser atendidos, ou seja, todos os encontros classe-professor devem ser alocados.
- Não devem existir conflitos por professor.
- Não devem existir conflitos de classes.
- Exclusividade período-matéria.
- Matemática deve ser ensinada no período da manhã em certas séries.
- As reuniões de classes devem ser espalhadas durante a semana por cada classe.

Para escola de ensino médio existem restrições *hard* e restrições *soft*:

Restrições *hard*:

- Todas as reuniões classe-professor devem ser agendadas.
- Não deve haver conflitos de professor.
- Não deve haver conflitos de classe.
- Requerimentos de períodos divididos devem ser atendidos e reagrupados para uma aula em um período. Mais de um professor para ensinar cada grupo pode ser necessário.

Restrições *soft*:

- Restrição de classe-período.
- Restrição de professor-período.
- Restrição dividida/combinada.

- Criação da população inicial:

Para construção não pode haver qualquer tipo de confrontos de classe, ou seja, uma classe não pode ser alocada mais de uma vez num período. No exemplo, ilustrado na Figura 4.17, no período 1, T1 Ensina para classe C2. No período 2, as classes C1 e C2 são divididas e então reagrupadas. Assim existem dois professores alocados para o período 2 das duas classes. O Professor T7 ensina um grupo e o T4 o outro grupo. Isso é um exemplo de classe

dividida e de reagrupamento como no caso de ensino de línguas diferentes. Este exemplo será referido como uma classe múltipla. Classes padrão que não serão divididas serão chamadas de classes individuais.

O método usado para construção sequencial (SCM) é usado para criar a população do Algoritmo Genético na primeira fase. O SCM cria uma população de *m* indivíduos que são copiados para a população inicial do AG. O SCM é implementado para criar cada elemento da população inicial. O valor de *m* é dependente do problema e é, portanto, um parâmetro genético. Durante o SCM cada *timetable* é construído através da atribuição de cada tupla classe-professor na lista de requisitos, como exemplo, reunião de classe e professores para um período específico. As seguintes heurísticas de baixo nível e combinação de heurísticas foram testadas para a escola primária e de ensino médio.

- Heurística da classe dividida.
- Grau de período consecutivo.
- Grau de saturação.
- Grau de divisão de professores.
- Prioridade de aulas.
- Grau da classe-professor.

Cada tupla é alocada para período de pena mínima. O período de pena mínima é identificado através da triagem de todos os períodos viáveis disponíveis para a tupla de acordo com o custo resultante da restrição *soft* de atribuição da tupla para a versão atual da *timetable*. O período de pena mínima é o período com o menor custo. Se houver mais de um período viável com o menor custo de restrição *soft*, o período é escolhido aleatoriamente entre eles. Se o problema não tem restrições *soft*, a tupla é atribuída a um período viável escolhido aleatoriamente. A população inicial da primeira fase é iterativamente refinada para um determinado conjunto de número *g* de gerações. Esse valor é um parâmetro genético. Experimentos indicaram que todos os elementos da última geração da primeira fase formam a população inicial da segunda fase.

- Avaliação da população:

A resolução do problema de *Timetabling* de escolas sul-africanas foi modelado em duas fases, usando dois algoritmos genéticos. Cada algoritmo genético segue o modelo de controle de gerações com distintas gerações. Uma população inicial é criada na primeira geração. Durante cada geração a população é avaliada, os pais são selecionados e o operador de mutação é aplicado aos pais escolhidos para criar a prole de cada geração. Durante a primeira fase, a aptidão de um indivíduo é uma restrição de custo *hard* (*Hard Constraint Cost)* do *timetable.* Na segunda fase, a aptidão de um indivíduo é uma restrição de custo *soft* (*Soft Constraint Cost)* do *timetable*. Os algoritmos genéticos de ambas as fases objetivam minimizar a função de aptidão, isto é, o número de restrições, evitando novos conflitos.

- Método de Seleção:

Os pais são escolhidos usando uma variação de seleção por torneio - que mostrou ser mais eficaz que a seleção por torneio padrão - na qual o indivíduo mais apto pode ser devolvido ao torneio.

- Operadores:

Operadores de *crossover* não foram utilizados na resolução do problema proposto. Para criação de novas proles a mutação e o método de seleção modificado foram aplicados. Foram utilizados dois tipos de mutação. O operador de mutação simples, para troca de classes e o operador de subida da montanha (*hill-climber),* para troca de professores. Para o problema da escola primária o operador de mutação simples foi suficiente para o problema. Mas para o problema das escolas de ensino médio foi necessária uma combinação dos operadores de mutação simples e de subida da montanha.

- Critério de parada:

Foi limitado pelo número de gerações.

- Resultados e conclusões:

Este estudo usou AGs para resolver o problema dos horários de uma escola primária e uma de ensino médio Sul-Africanas. Os AGs analisados encontraram soluções viáveis tanto

para o problema da escola primária, quanto para o da escola secundária. O melhor horário gerado pelo AG para o problema de definição dos horários da escola primária reuniu todas as restrições duras. O calendário usado atualmente pela escola é gerado por um sistema de definição dos horários online. O calendário gerado normalmente não satisfaz todas as restrições, e é alterado manualmente para atender as restrições, se possível. O algoritmo genético apresentado neste estudo produziu os calendários para o ensino médio com restrições de custo hard e soft ótimas. Trabalhos futuros irão aplicar AG a problemas de *Timetabling* de escolas adicionais.

4.3 CONSIDERAÇÕES FINAIS

Diversos trabalhos sobre aplicações de AGs para resolução do Problema de *Timetabling* foram analisados neste capítulo. Na Tabela 4.5 encontram-se resumidas informações qualitativas e quantitativas sobre: (i) a forma de representação, (ii) o tamanho da população, (iii) a função de aptidão, (iv) os operadores e (v) o critério de parada adotados nos AGs implementados nos referidos trabalhos.

A forma de representação de solução mais utilizada pelos trabalhos analisados foi a matriz 3D. Possui vantagens, devido ao seu tipo de implementação, como a de não permitir que algumas restrições possam ocorrer, como por exemplo, duas disciplinas associadas a mesma turma ocupando o mesmo horário, outras formas de representação mais compactas foram utilizadas, como é o caso do Vetor 2D, para implementar problema com menor complexidade. Soluções também foram representadas por *structure* (estrutura) de vetores composta por diversas informações para facilitar o acesso durante a execução do algoritmo.

Tabela 4.5: Tabela de características mais comuns entre os trabalhos analisados.

Característica Analisada	**Especificação**	**Frequência de Uso**
Forma de representação	Vetor de Matrizes 2D	2
	Matriz 2D	1
	Vetor 2D	2
	Matriz 3D	4
	Estrutura de vetores composta por variáveis simples	1
Tamanho da população	Entre 80 a 160 Indivíduos	1
	100 Indivíduos	2
	Entre 120 a 250 Indivíduos	1
	Entre 100 a 500 Indivíduos	1

	Entre 200 a 750 Indivíduos	1
	750 Indivíduos	1
	1000 Indivíduos	2
	Entre 256 a 5120 Indivíduos	1
Função Aptidão	$f(x) = \frac{1}{(1+(restrições\ violadas))}$	3
	$f(x) = \sum bônus - \sum penalizações$	2
	$f(x) = \sum_{l=1}^{K} \alpha_k \mathrm{I}_k + \sum_{l=1}^{L} \beta_l \varrho_l$	1
	$f(x) = 2 * \sum_{i=0}^{n} disponibilidade(slot) - \left(\sum 10 * N_{janelas} 1000 * N_{agrupamentos}\right)$	1
	$fitness(indivíduo) = (Número\ de\ conflitos) * K + Qualidade$ $fitness(cumulativo) = \sum_{i=1}^{k} fitness(indivíduo_i)$	1
	Primeira fase do AG: $f(s)^1 = \sum HCC$ Segunda fase do AG: $f(s)^2 = \sum SCC$	1
	$f(s) = \sum HCS + \sum SCS$	1
Operadores	Crossover	9
	Mutação	10
Critério de parada	Convergência	1
	Tempo Máximo de Execução	1
	200 iterações	1
	Entre 300 a 1000 iterações	1
	500 iterações	2
	3000 iterações	1
	10000 iterações	1
	15000 iterações	1
	Entre 10000 e 30000 iterações	1

O tamanho da população dos AGs variou entre 80 e 5120 indivíduos, mas na maioria dos trabalhos não passou dos 1000 indivíduos. Em problemas mais simples não foi necessária uma população de grande porte para testar os AGs. O tamanho da população inicial dos AGs aumentou bastante, quando foram realizados testes usando dados reais de instituições de ensino.

As funções de aptidão adotadas foram bastante específicas para cada trabalho, sendo em geral, desenvolvidas de forma a evitar soluções não viáveis, incorporando através de pesos, penalidades para soluções que violam restrições (*hard* e/ou *soft*).

A maioria dos trabalhos utilizou os operadores *crossover* e mutação para geração de novas populações. Em um dos trabalhos o operador de crossover não foi utilizado, sendo utilizados dois tipos de operadores de mutação. O operador de mutação simples, para troca de

classes e o operador de subida da montanha (*hill-climber),* para troca de professores. No referido trabalho, o operador de mutação simples foi suficiente para gerar novas populações em escolas primárias mantendo a diversidade. Mas para o problema das escolas de ensino médio foi necessária uma combinação dos operadores de mutação simples e de subida da montanha.

Foram identificadas ocorrências de três tipos de condições de término para os AGs: por tempo, por número de gerações e por convergência dos genes. Em 80% dos trabalhos, o critério de parada foi por número de iterações. Critério de parada por restrição de tempo foi usado para verificação de desempenho. Critério de parada por número de gerações foi usado tanto para teste de desempenho especificando um número reduzido de gerações, quanto para busca de soluções viáveis para o problema usando um número alto de iterações, independentemente do tempo.

De uma maneira geral, as soluções propostas neste capítulo para o problema do *Timetabling* escolar e acadêmico apresentaram as seguintes características em comum: (i) matriz 3D como forma de representação da solução; (ii) população variando entre visando de 100 a 1000 indivíduos; (iii) função de aptidão baseada em restrições calculada através de equações do tipo $f = 1/((1 + (restrições\ violadas))$), (iv) geração de novas populações por meio dos operadores de *crossover* e mutação e (v) critério de parada por número de geração. Para concluir estas considerações finais, deve ser ressaltado que os AGs analisados foram testados em diferentes situações, fornecendo resultados satisfatórios - tanto referentes a qualidade das soluções obtidas, quanto ao tempo gasto para produzi-las - constituindo-se, portanto em uma ferramenta adequada para resolução de problema de *Timetabling* em escolas e universidades.

5. CONCLUSÕES E SUGESTÕES DE TRABALHOS FUTUROS

O planejamento de grades horárias de cursos de instituições de ensino é uma tarefa difícil porque nem sempre existem sistemas automatizados e algoritmos eficientes para sua execução em tempo de processamento. Apesar de atualmente existir uma grande variedade de sistemas automatizados desenvolvidos especificamente para geração de horário escolar e acadêmico, existe ainda uma grande dificuldade em escolas e universidades de encontrar um sistema que satisfaça completamente suas necessidades. Diversos estudos têm sido desenvolvidos na tentativa de resolver este problema usando AGs e outras técnicas. Os resultados obtidos têm mostrado que não existe uma única técnica ou sistemas automatizados totalmente adequados para resolução do problema de *Timetabling*, que deve ser abordado segundo as especificidades de cada instituição de ensino. Cada instituição de ensino, portanto, deve buscar construir e se não for possível tentar adequar, sistemas automatizados para atender suas necessidades específicas.

Sistemas automatizados para resolução de problema do *Timetabling* escolar e acadêmico baseados em AGs foram selecionados entre diversos consultados na literatura e analisados com respeito a uma série de características metodológicas. Apesar de todos os sistemas terem sido desenvolvidos para diferentes instituições de ensino, todos usaram uma metodologia de desenvolvimento similar. De uma maneira geral, os sistemas automatizados baseados em AGs desenvolvidos para resolver o problema em questão apresentaram as seguintes características: (i) matriz 3D como forma de representação da solução; (ii) população variando entre visando de 100 a 1000 indivíduos; (iii) função de aptidão baseada em restrições calculada através de equações do tipo $f = 1/((1 + (restrições\ violadas))$), (iv) geração de novas populações por meio dos operadores de *crossover* e mutação e (v) critério de parada por número de geração. Os AGs analisados foram testados em diferentes situações, forneceram resultados satisfatórios - tanto referentes a qualidade das soluções obtidas, quanto ao tempo gasto para produzi-las - constituindo-se, portanto, em uma ferramenta adequada para resolução de problema de *Timetabling* em escolas e universidades.

Como trabalhos futuros sugere-se que aplicativos de *Timetabling* sejam desenvolvidos para os diversos cursos da UNICAP, iniciando pelo o Curso de Ciência da Computação. A construção de um sistema automatizado baseado em AGs para produção de grades horárias

para o Curso de Ciência da Computação deve se fundamentar, a princípio, tanto na experiência dos funcionários responsáveis pela organização dos horários, quanto nas informações fornecidas por esta monografia. Gerar uma solução que é próxima da ideal facilitará o trabalho dos funcionários que não terão que passar muito tempo analisando a solução gerada para garantir um horário aceitável.

6. REFERÊNCIAS BIBLOGRÁFICAS

ADAMI, C. **An Introduction to Artificial Life**, 1. ed. New York: Springer-Verlag, 1998. 376 p.

AGUIAR, M. S. **Análise Formal da Complexidade de Algoritmos Genéticos**. 75 f. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática da Universidade Federal do Rio Grande do Sul, Universidade Federal do Rio Grande do Sul, Porto Alegre. 1998.

BÄCK, T., FOGEL, D. B.; MICHALEWICZ, Z. **Evolutionary Computation 1:** Basic Algorithms and Operators, 1. ed. Boca Raton: CRC Press, 2000. 378 p.

BÄCK, T.; FOGEL, D. B; MICHALEWICZ, Z. 2000. **Evolutionary Computation 2:** Advanced Algorithms and Operators, 1. ed. Boca Raton: CRC Press, 2000. 308 p.

BALLARD, D. **An Introduction to Natural Computation**, 1. ed. Massachusetts: MIT Press, 1999. 336 p.

BITTENCOURT, G. **Inteligência Computacional**. Santa Catarina: Universidade Federal de Santa Catarina, 2008. Apostila.

BLOOMFIELD, S. D.; MCSHARRY, M. M. Preferential Course Scheduling, **Interfaces**, v. 9, n. 4, p. 24-31, 1979.

BONABEAU, E.; DORIGO, M.; THERAULAZ, T. **Swarm Intelligence**: From Natural to Artificial Systems, 2. ed. New York: Oxford University Press. 1999. 320 p.

BORGES, S. K. **Resolução de Timetabling utilizando Algoritmos Genéticos e Evolução Cooperativa**. 2003. 104 f. Dissertação (Mestrado em Informática) — Departamento de Informática, Setor de Ciências Exatas, Universidade Federal do Paraná, Curitiba. 2003.

BURKE, E. K.; ELLIMAN, D. G.; WEARE, R. F. The Automation of the Timetabling Process in Higher Education, **Journal of Educational Technology Systems**, v. 23, n. 4, p. 257-266, 1995.

BURKE, E.; JACKSON, K.; KINGSTON, J.; WEARE, R. Automated University Timetabling: The State of the Art. **The Computer Journal**, v. 40, n. 9. p. 565-571, 1997.

CARTER, M. W.; LAPORTE, G.; CHINNECK, W. A. General Examination Scheduling System. **Interfaces**, v. 24, n. 3, p. 109-120, 1994.

CASTRO, R. **Otimização de Estruturas com Multiobjetivos Via Algoritmos Genéticos de Pareto**. 224 f. Tese (Doutorado em Ciência da Computação) — Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro. 2001.

CISCON, L. A.; ALVARENGA, G. O Problema de Geração de Horários: um Foco na Eliminação de Janelas e Aulas Isoladas. In: **XXXVII Brazilian Symposium of Operational Research**, 2005. p. 1725-1735.

COOPER, T. B.; KINGSTON, J. H. The Complexity of Timetable Construction Problems, in the Practice and Theory of Automated Timetabling, In: BURKE, E. K.; ROSS, P. (Eds.), **Lecture Notes in Computer Science**, [S.I]: Springer-Verlag, 1996. p. 283-295.

CORNE, D.; ROSS, P.; FANG, H. L. Fast Practical Evolutionary Timetabling, Lecture Notes in Computer Science, In: FOGARTY, C. T. **Evolutionary Computing:** AISB Workshop Leeds, U.K., April 11–13, 1994 Selected Papers. Berlin: Springer-Verlag, 1994. p. 251-263.

COSTA, F. F. **Programação de Horários em Escolas via GRASP e Busca Tabu**. 39 f. Monografia (Graduação em Engenharia de Produção) — Departamento de Engenharia de Produção, Universidade Federal de Ouro Preto, Ouro Preto. 2003.

DA SILVA, A. B.; BETEMPS, C. M.; HEINEN, M. Algoritmos Genéticos na obtenção de uma Grade de Horários com Múltiplos Cursos para uma Instituição de Ensino. In: ENCONTRO ANUAL DE TECNOLOGIA DA INFORMAÇÃO E SEMANA ACADÊMICA DE TECNOLOGIA DA INFORMAÇÃO, 4., 2014, Frederico Westphalen. **Anais do EATI — Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação**. Frederico Westphalen: [s.n], 2014. 1 CD-ROM.

DASGUPTA, D. **Artificial Immune Systems and Their Applications**, 1. ed. Berlin: Springer-Verlag, 1999. 306 p.

DASGUPTA, D.; MICHALEWICZ, Z. **Evolutionary Algorithms in Engineering Applications**, 1. ed. Berlin: Springer-Verlag, 1997. 558 p.

DE CASTRO, L. N.; TIMMIS, J. I. **Artificial Immune Systems: A New Computational Intelligence Approach**, 1. ed. London: Springer-Verlag, 2002. 380 p.

DE CASTRO, L. N.; VON ZUBEN, F. J. From Biologically Inspired Computing to Natural Computing. In: DE CASTRO, L. N.; VON ZUBEN, F. J. (Eds.). **Recent Developments in Biologically Inspired Computing**. Hershey: IGI Global, 2004. p. 1-8.

EVEN, S.; ITAI, A.; SHAMIR, A. On the Complexity of Time Table and Multi-Commodity flow Problems. In: **Foundations of Computer Science, 1975., 16th Annual Symposium on**. IEEE, 1975. p. 184-193.

FILHO, G. R. **Melhoramentos do Algoritmo Genético Construtivo e Novas Aplicações em Problemas de Agrupamento**. 129 f. Tese (Doutorado em Computação Aplicada) — Instituto Nacional De Pesquisas Espaciais, São José dos Campos. 2000.

FLAKE, G. W. **The Computational Beauty of Nature**, 2. ed. Massachusetts: MIT Press, 1999. 493 p.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**, 1. ed. Boston: Addison-Wesley, 1989. 432 p.

GOMIDE, L. R.; ARCE, J. E.; DA SILVA, A. C. L. Uso do Algoritmo Genético no Planejamento Florestal considerando seus Operadores de Seleção. **Cerne**, v. 15, n. 4, p. 460–467, 2009.

GOTLIEB, C. The Construction of Class-Teacher Timetables. In: **Proc. IFIP Congress**, 62., 1963. p. 73–77.

GRAMß, T.; BORNHOLDT, S.; GROß, M.; MITCHELL, M.; PELLIZZARI, T. **Non-Standard Computation**: Molecular Computation - Cellular Automata - Evolutionary Algorithms - Quantum Computers**,** 1. ed. Weinheim: Wiley-VCH, 2001. 240 p.

HAMAWAKI, C. D. L. **Geração Automática de Grade Horária usando Algoritmos Genéticos**: o Caso da Faculdade de Engenharia Elétrica da UFU. 104 f. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia. 2011.

HAYKIN, S. **Neural Networks**: A Comprehensive Foundation, 2. ed. Upper Saddle River: Prentice Hall, 1999. 842 p.

HIRVENSALO, M. **Quantum Computing**, 2. ed. Berlin: Springer-Verlag, 2004. 214 p.

HOLLAND J. H. **Adaptation in Natural and Artificial System**. 1. ed. Michigan: University of Michigan Press, 1975. 183 p.

KENNEDY, J.; EBERHART, R.; SHI Y. **Swarm Intelligence**, 1. ed. San Francisco: Morgan Kaufmann Publishers, 2001. 512 p.

LANGTON, C. Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, In: LANGTON, C. (Ed.), **Artificial Life**: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, Boston: Addison-Wesley, 1988. p. 1-47.

LEVY, S. **Artificial Life**: a Report from the Frontier where Computers meet Biology, 1. ed. New York: Vintage Books, 1992. 390 p.

LIMA, C. M. V. **Otimização de Trânsito uma Abordagem Utilizando Algoritmos Genéticos**. 80 f. Monografia (Graduação em Ciência da Computação) — Centro de Informática, Universidade Federal de Pernambuco, Recife. 2005.

MAHIBA, A. A.; DURAI, C. A. D. Genetic Algorithm with Search Bank Strategies for University Course Timetabling Problem. **Procedia Engineering**, v. 38, p. 253-263, 2012.

MANDELBROT, B. **The Fractal Geometry of Nature**, 2. ed. New York: W. H. Freeman and Company, 1983. 468 p.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs**. 3. ed. Berlin: Springer-Verlag, 1996. 387 p.

MITCHELL, M. **An Introduction to Genetic Algorithms**. 1. ed. Massachusetts: MIT Press, 1996. 221 p.

NIELSEN, M. A.; CHUANG, I. L. **Quantum Computation and Quantum Information**, 1. ed. Cambridge: Cambridge University Press, 2000. 700 p.

PACHECO, M. A. **Algoritmos Genéticos: Princípios e Aplicações**. Rio de Janeiro, Universidade Pontifícia do Rio de Janeiro, 1999. Apostila.

PAECHTER, B. et al. Two Solutions to the General Timetable Problem using Evolutionary Methods. In: **Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on**. IEEE, 1994. p. 300-305.

PATON, R. **Computing with Biological Metaphors**, 1. ed. London: Chapman & Hall, 1994. 446 p.

PATON, R.; BOLOURI, H.; HOLCOMBE, M. **Computing in Cells and Tissues: Perspectives and Tools of Thought**, 1. ed. Berlin: Springer-Verlag, 2004. 345 p.

PĂUN, G.; ROZENBERG, G.; SALOMA, A. **DNA Computing**. 1. ed. Berlin: Springer-Verlag, 1998. 400 p.

PEDRYCZ, W.; GOMIDE, F. **An Introduction to Fuzzy Sets**. 1. ed. Cambridge: The MIT Press, 1998. 469 p.

PEITGEN, H.-O.; JÜRGENS, H.; SAUPE, D. **Chaos and Fractals: New Frontiers of Science**. 2. ed. New York: Springer-Verlag, 1992. 864p.

PREIS, T. A. **Protótipo Gerador de Grades Horárias para Instituições de Ensino**. 72 f. Monografia (Graduação em Ciência da Computação) — Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. 2007.

RAGHAVJEE, R.; PILLAY, N. Using Genetic Algorithms to solve the South African School Timetabling Problem. In: **Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on**, IEEE, 2010. p. 286-292.

REEVES, C. **Modern Heuristic Techniques for Combinatorial Problems**. 1. ed. New Jersey: John Wiley & Sons, 1993. 320 p.

RIBEIRO, M. D. R. **Um Estudo sobre Algoritmos Genéticos e Culturais e Métodos de Descoberta de Conhecimento, Perfis e Reputação em Redes Sociais.** 79 f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas. 2012.

ROMERO, B. P. Examination Scheduling in a Large Engineering School: A Computer-Assisted Participative Procedure, **Interfaces**, v. 12, n. 2, p. 17-24, 1982.

ROSS, P.; FANG, H.; CORNE, D. **Genetic Algorithms for Timetabling and Scheduling**. Departamento de IA Universidade de Edinburgh, 80 South Bridge, Edinburgh EH 11HN, Scotland, 1999.

ROSS, P.; FANG, H.; CORNE, D. **Fast Pratical Evolutionary Timetabling**. Departamento de IA Universidade de Edinburgh, 80 South Bridge, Edinburgh EH 11HN, Scotland, 2000.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Editora Campus, 2004. 1020p.

SAKAWA, M. **Genetic Algorithms and Fuzzy Multiobjective Optimization**. Netherlands: Springer, 2002. 306 p.

SIGL, B.; GOLUB, M.; MORNAR, V. Solving Timetable Scheduling Problem using Genetic Algorithms, In: **Proc. of the 25th International Conference on Information Technology Interfaces**. 2003. p. 519-524.

SILVA, E. **Otimização de Estruturas de Concreto Armado utilizando Algoritmos Genéticos**. 194 f. Dissertação (Mestrado em Ciência da Computação) — Escola Politécnica, Universidade de São Paulo, São Paulo. 2001.

SILVA, A. S. N. **Estudo e Implementação, mediante Recozimento Simulado, do Problema de Alocação de Salas**. 120 f. Monografia (Graduação em Ciência da Computação) — Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras. 2005.

TICONA, W. G. C. **Aplicação de Algoritmos Genéticos Multi-Objetivo para Alinhamento de Sequências Biológicas**. 129 f. Dissertação (Mestrado em Ciências Matemáticas e de Computação) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2003.

TIMÓTEO, G. T. S. **Desenvolvimento de um Algoritmo Genético para a Resolução do Timetabling**. 86 f. Monografia (Graduação em Ciência da Computação) — Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras. 2002.

VIEIRA, F.; MACEDO, H. Sistema de Alocação de Horários de Cursos Universitários: Um Estudo de Caso no Departamento de Computação da Universidade Federal de Sergipe. **Scientia Plena**, v. 7, n. 3, p. 1-12, 2011.

ZIMMERMANN, H.-J. **Practical Applications of Fuzzy Technologies**. 1. ed. [S.I]: Springer Science & Business Media, 1999. 667 p.