Timetabling com algoritmos genéticos: resultados, restrições e exploração do paralelismo

Tiago P. Fucilini, C. S. P., Eli Maruani, Marcelo Trindade Rebonatto

[1]Curso de Ciência da Computação – ICEG - Universidade de Passo Fundo (UPF)
Caixa Postal 611 – 99052-900 – Passo Fundo – RS – Brazil
91081@upf.br, 62389@upf.br, 91049@upf.br, rebonatto@upf.br

Abstract. *This text describes a solution for the problem of timetabling of professors in courses using genetic algorithms. A GA model is detailed, including its implementation and the produced results. Although with satisfactory results, the implemented GA presents some restrictions of cases where it can be applied. Thus, also a new more complex GA is suggested, that will require the use of parallel processing in its execution.*

Resumo. *Este texto descreve uma solução do problema de timetabling de professores em disciplinas usando algoritmos genéticos. Um modelo de AG é detalhado, incluindo sua implementação e os resultados produzidos. Embora com resultados satisfatórios, o AG implementado apresenta algumas restrições de casos onde pode ser aplicado. Assim, também é proposto um novo AG, mais complexo que necessitará do uso de processamento paralelo em sua execução.*

1. Introdução

O problema de designar professores a disciplinas e alocar o conjunto de disciplinas num horário de um curso é um problema complexo, principalmente quando o número de disciplinas e professores é elevado. Tradicionalmente, os professores não possuem todos os horários disponíveis para as disciplinas, seja por que trabalham em outros cursos, na mesma ou outra instituição ou porque assumem outras atividades. Essa restrição de horários dos professores aumenta a complexidade na busca de uma solução. Quando o caso é levado a uma universidade, com dezenas de cursos e turmas em andamento concomitantemente, envolvendo centenas de professores e disciplinas, o problema se acentua ainda mais. A busca por uma solução manual do problema pode demandar o esforço de muitas pessoas durante vários dias e, mesmo assim, não gerar uma boa solução (Pereira, Branco Neto, 2001, p.108).

Segundo Rela (2004), Algoritmos Evolucionários são técnicas de computação inspiradas na teoria da seleção natural de Charles Darwin. A idéia principal destas técnicas é a de manter uma população de soluções candidatas (indivíduos) do problema, e também, a criação de uma função para a avaliação dos indivíduos melhor adaptados. Esta função é utilizada para mensurar a aptidão (*fitness*) de cada indivíduo da população.

A área da ciência da computação que estuda o funcionamento e aplicações de algoritmos evolucionários é a inteligência artificial, uma vez que se concentra na

automação do comportamento inteligente (Lugger, 2004). Os algoritmos evolucionários são uma das metodologias computacionais empregadas pela inteligência artificial, tendo nos algoritmos genéticos (AG) um de seus mais conhecidos modelos (Perone, 2006).

Segundo Pereira e Branco Neto os AG tornaram-se "um campo difundido e aplicável a diversas áreas do conhecimento" (2001). Isso ocorre em virtude de sua simplicidade, de compreensão, funcionamento e implementação, possibilitando buscas eficazes em amplos espaços de procura. Eles estão sendo utilizados em um numeroso e variado grupo de problemas, porém mais especificamente onde ainda não se tem uma solução determinística razoável, que é o caso do problema de *timetabling* (Perone, 2006).

Este texto descreve uma solução para o problema de *timetabling* de professores a disciplinas em horários usando algoritmos genéticos. Ele apresenta a concepção e modelagem do algoritmo bem como sua implementação, resultados e limitações de funcionamento. A fim de suplantar as limitações do algoritmo implementado, um novo AG é descrito, com objetivos maiores e fazendo uso de exploração do paralelismo em sua implementação.

O restante do texto está divido em quatro seções. Na seção 2 o modelo do AG proposto para resolver o *timetabling* é apresentado, sendo sua implementação e resultados discutidos na seção 3. A seção 4 é destinada ao melhoramento do AG, permitindo resolver as restrições do modelo anterior. A seção 5 finaliza o presente texto com as considerações acerca do trabalho e a indicação se sua continuidade.

1.1 Trabalhos relacionados

Diversos trabalhos envolvendo AG para a resolução de *timetabling* podem ser relacionados. O trabalho de Pereira e Branco Neto (2001) foi descrito como ainda em desenvolvimento e, não puderam ser encontradas implementações posteriores do trabalho com resultados.

Melo (2003) trabalhou na definição, otimização e implementação de uma função para avaliação de AG. Apresentou bons resultados, porém com elevado tempo de execução e sem interface visual para uso do sistema, dificultando sua adoção como solução efetiva. Lenzi (2004) produziu um AG que resolve os objetivos propostos com bom desempenho, porém se limita a um curso com apenas seis (6) semestres de duração com aulas somente à noite. Igualmente a Lenzi, o software carece de uma interface otimizada para uso do software.

Dessa forma, os trabalhos citados não atendem completamente as restrições de um *timetabling* para uma Universidade com vários cursos, sendo os cursos com número diferenciado de semestres de duração e com a possibilidade de aula nos três turnos: manha, tarde e noite.

2. Modelo de algoritmo genético

Nesta seção será apresentado o modelo conceitual do Algoritmo Genético, desde a representação escolhida do cromossomo, como são geradas as gerações iniciais, como trabalham as funções de fitness, crossover, mutação e a base de dados.

2.1 Base de dados

Uma base de dados foi definida para armazenar de forma permanente as informações manipuladas pelo AG. Dentre as informações armazenadas destacam-se: (a) dados dos professores: disponibilidades de horários e as disciplinas que podem lecionar; (b) oferecimento de disciplinas por nível de curso; (c) feriados por dias da semana; (d) horários dos *timeslots* e a distribuição de carga horária gerada; (e) número de laborartórios disponíveis para as aulas práticas por curso.

Os dados manipulados pelo AG devem ser armazenados em memória RAM, a fim de alcançar bom desempenho na execução da função de *fitness*. Ao inicio da execução os dados da base devem ser alocados na memória RAM e ao final os resultados armazenados na base de dados para posterior uso das informações.

2.2 Representação do Cromossomo

A representação cromossômica escolhida para representar os níveis e *slots* de tempo do problema foi a representação vetorial como mostra na Figura 1.

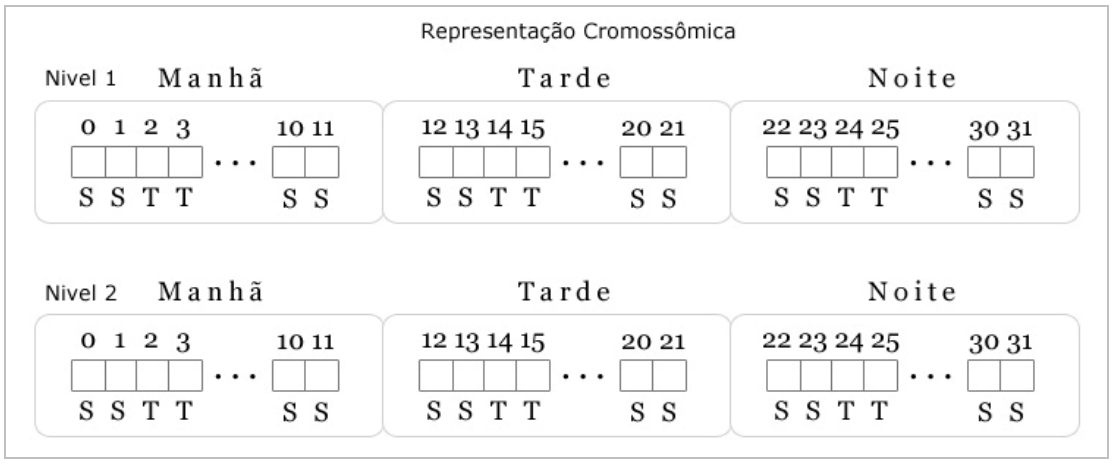

Figura 1 – Representação cromossômica

Como se pode notar na Figura 1, a representação da distribuição dos horários é implementada em um vetor de duas dimensões, onde a primeira dimensão representa os níveis e a segunda dimensão, os *slots* de tempo.

A granularidade do intervalo entre um *slot* de tempo (elemento do vetor) e outro são 2 créditos, por exemplo, o professor/disciplina alocado no *slot* 0 da manhã, representa uma disciplina no intervalo das 08:00 até 09:40; um professor/disciplina no *slot* 1 da manhã, representará uma disciplina no horário das 09:55 até 11:35. O turno da manhã é o único com possibilidade de aulas no sábado, por isso o tamanho distinto de 12 elementos em relação aos outros turnos de 10 elementos cada.

2.3 Geração da população inicial

A população inicial deve ser gerada por nível, com um inicializador distinto para cada nível. Este inicializador trabalha de maneira estocástica de modo a gerar uma boa distribuição aleatória no momento de alocar professores/disciplinas em *slots* de tempo.

O mecanismo de alocação estocástica é simples: primeiro cria-se uma lista de posições disponíveis do vetor de determinado nível. Após é escolhido aleatoriamente uma dessas posições e o código professor/disciplina é alocado para tal posição.

A geração inicial pode conter indivíduos idênticos, ou seja, indivíduos com a mesma distribuição, porém durante a evolução ocorrerá a dispersão caso estes indivíduos obtiverem um *score* baixo.

2.4 Constraints e fitness

As *constraints* podem ser separadas em dois tipos principais: *Hard* e *Soft*. Segundo Fang (1994, p. 49), as *constraints* do tipo *hard* não devem ser violadas uma vez que são críticas e afetam de modo significativo a qualidade do resultado. As *constraints* do tipo *soft* são mais sutis, ou seja, devem ser preferencialmente seguidas, porém podem ser violadas caso haja a necessidade.

Foram elaboradas três *hard constraints* e cinco *soft constraints* que farão parte da função de *fitness* e que darão a direção no espaço de procura para o Algoritmo Genético. A Tabela 1 ilustra as *constraints* do modelo.

Tabela 1 – Relação de *constraints*

Hard constraints	*Soft constraints*
Oferecimento: verificação se todas as disciplinas oferecidas para determinado nível estão presentes no cromossomo gerado.	Laboratório: verifica se a quantidade necessária de laboratórios para determinado período não excede o máximo de laboratórios disponíveis.
Disponibilidade: indica quando (quais *timeslots*) os professores estão disponíveis para lecionar.	Preferencial: indica se determinado professor é preferencial para lecionar uma disciplina.
Clash: indica uma situação quando gerada uma distribuição em que um professor está dando aula em níveis diferentes ao mesmo tempo.	Horas: verifica se a quantidade de horas distribuída para um professor não excede a quantidade máxima de horas possíveis para aquele professor.
	Feriado: verifica se a disciplina pode ou não ser alocada em um dia da semana com feriados.
	Disciplinas de 2 créditos: verifica a alocação das disciplinas de 2 créditos em *slots* de tempo adjacentes, ou seja, completando o turno com aula.

A função de *fitness* foi modelada com base na abordagem de penalizações e bonificações. Para tanto, um acumulador para o *score* bruto armazena o somatório de cada uma das *constraints* multiplicado pelo seu respectivo peso.

3.5. Operadores

O operador genético de *crossover* foi projetado para funcionar de maneira distinta das implementações clássicas, fato decorrente da representação utilizada para modelar os níveis e *slots* de tempo. Foi utilizada a técnica *one-point crossover*, com a diferença de que o ponto de corte a partir do qual se terá duas partes distintas do material genético é realizada de modo a permitir a cópia de níveis em sua totalidade, isto é, quando cruzados, os indivíduos farão um intercâmbio dos níveis completos.

A mutação foi desenvolvida visando obter uma boa distribuição estocástica dentro de cada nível do cromossomo. A mutação dos genes acontece em duas etapas: a mutação do professor/disciplina e o *swap* das disciplinas. Na mutação do

professor/disciplina, o algoritmo de mutação procura a disponibilidade de algum outro professor para a mesma disciplina e muta-o para este outro professor/disciplina; na segunda etapa da mutação, o algoritmo realiza a troca da posição do professor/disciplina para outro *slot* disponível.

3. Implemantação e resultados

A implementação foi toda desenvolvida na linguagem C++ utilizando o compilador GNU G++ v.4.2.3. Para manuseio permanente das informações foi modelada uma base de dados, implementada com MySQL. Entre os principais componentes de software usados na implementação do AG descrito na seção 3 encontram-se:

- Wrapper para MySQL++ v.3.0.3 da TangentSoft para realizar a interface com o banco de dados MySQL;
- Banco de dados MySQL 5.0.51a para o armazenamento dos dados/estruturas de alimentação do Algoritmo Genético;
- GALib v.2.4.7, Copyright (c) Massachusetts Institute of Technology and Matthew Wall. All rights reserved. Biblioteca em C++ para o desenvolvimento do núcleo do Algoritmo Genético;
- Doxygen , utilizado para geração da documentação do código fonte.
- Qt Open Source Edition 4.3.4v. Biblioteca para o desenvolvimento da interface gráfica.
- Qt4 Designer 4.3.4v, utilizada para a criação do designer da aplicação.
- QDevelop. IDE para o desenvolvimento de aplicações Qt.

Para o computo do *score* dos indivíduos da população, foram definidos os bônus e penalidades descritos na Tabela 2.

Tabela 2 – Bonificações e penalizações utilizadas

Constraint	Bônus/Penalidade	Peso	Multiplicador
Oferecimento	Bonifica	3,5	1
Disponibilidade	Bonifica	2,0	1
Clash	Bonifica	1,5	1
Professor Preferencial*	Bonifica	0,2	1
Laboratório	Bonifica	0,1	1 * cada *slot* de tempo
Horas	Bonifica	0,1	1 * cada professor
Feriado	Penaliza	0,5	quantidade de feriados
Turno completo**	Bonifica	0,8	1

* apenas para oferecimento em turnos corretos

** para cada dupla de *slots* onde houver 2 disciplinas de 2 créditos no mesmo turno

Com o objetivo de verificar os resultados obtidos pela implementação descrita do AG foram realizadas 10 rodadas de testes para a geração da distribuição de cargas horárias. Nestas gerações foram usados os seguintes parâmetros para configurar o Algoritmo Genético:

- Crossover: 0,4%
- Mutação: 0,1%
- Gerações: 500
- Tamanho da população: 1500

O AG implementado foi executado numa maquina com CPU Pentium Dual Core T2330 1.60 GHz, 2 GB de memória RAM com o sistema operacional Ubuntu 8.04. A base de dados era composta de 62 disciplinas, 36 professores, 141 relações de professores/disciplinas e 62 oferecimentos de disciplinas num cursos. A Figura 2 ilustra a interface gráfica para visualização dos resultados produzidos.

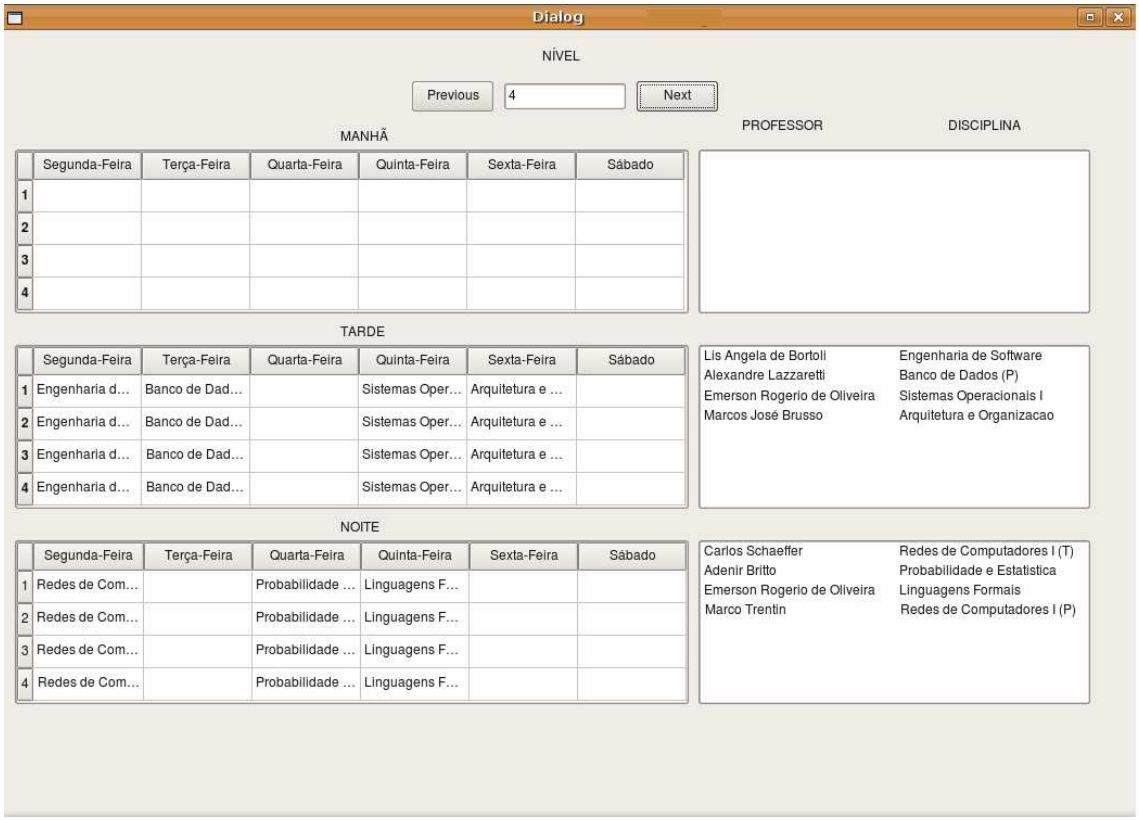

Figura 2 – Interface gráfica para manipulação das informações geradas

Os resultados foram obtidos após 500 gerações, consumindo aproximadamente 12 minutos e 25 segundos, gerando indivíduos para 10 semestres de disciplinas em turno único (somente noite, por exemplo) e turno duplo. A execução não foi encerrada por convergência e sim por número de gerações e obteve um resultado valido, com nehuma *hard constraints* e com um bom score máximo de 182,55.

O AG implementado gera o horário para um curso por vez, porém o modelo de solução pode ser empregado para geração de horários de vários cursos. Para tanto, após uma execução do AG é retirado da disponibilidade de horários dos professores os alocados a uma disciplinas, sendo após o AG novamente executado com dados de oferecimentos do novo curso.

4. Problemas detectados e melhorias

A versão atual do AG possui limitações como apenas a alocação de disciplinas pares, a não alocação de horários concomitantes na mesma turma, e a geração de apenas um curso por execução AG. Somente podem ser geradas disciplinas com número de créditos pares uma vez que a granulariade de horários é sempre de 2 horas/aula. Nos casos de turmas práticas divididas entre diferentes professores no mesmo horário o AG gera uma hard constraint, embora essa constraint não seja válida uma vez que os alunos não são os mesmos. A alocção de mais de uma curso por execução do AG é limitada, principalmente, em função do número variado de semestres que os cursos podem apresentar.

Uma nova versão do modelo proposto foi definida para solucionar os problemas detectados. O novo modelo muda a representação cromossômica e, consequentemente funções de *fitness* e operadores de mutação e *crossover*. No novo modelo de AG o cromossomo pode ser dividido em três partes. A Figura 3 ilustra uma representação gráfica do novo cromossomo.

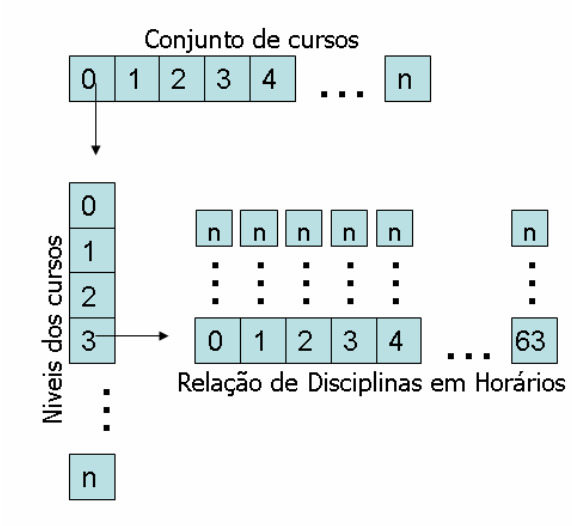

Figura 3 – Representação do novo modelo cromossômico

A primeira parte representa os cursos, armazenados num conjunto de dados de mesmo tipo, como, por exemplo, um vetor (Conjunto de Cursos). A segunda parte do cromossomo possuirá as informações dos níveis de um curso, armazenada em estrutura de dados de tamanho variável, como por exemplo, uma lista (Níveis dos cursos). A terceira parte será um vetor de 63 posições que serão os *slots* de horários de um nível, cada *slot* será 1 crédito e poderá ter mais de uma disciplina referenciada nele (Relação de disciplinas em Horários).

A partir da nova representação cromossômica, mais complexa para manuseio das informações e com elevado consumo de memória, a implementação de um novo AG necessita de um grande esforço computacional para ser executado. Para resolver esse problema à nova implementação sendo desenvolvida usa recursos de exploração explícita do paralelismo, com uso da biblioteca *Messaging Passing Interface* (MPI) para as funções necessárias ao controle e execução paralela. O modelo de AG paralelo sendo estudado é o *coarse-grained* uma vez que o ambiente de execução será um cluster.

Considerações finais e trabalhos em andamento

Este texto, apresentou o modelo conceitual e implementação de um protótipo funcional para a solução do problema de distribuição de cargas horárias a professores em cursos. A implementação do modelo e sua execução sobre dados reais de cursos proporcionaram a comprovação de sua aplicabilidade e eficiência na resolução do problema proposto. Mesmo funcional, a versão atual do modelo apresenta algumas restrições em relação ao funcionamento, principalmente a geração de um curso por vez, o que pode ocasionar soluções pouco eficientes para os útlimos cursos a terem seus horários gerados.

Um novo modelo, com representação cromossômica descrita nesse texto, corrige as deficiências do antigo. Os *slots* de tempo que antes representavam 2 créditos agora representam 1, proporcionando disciplinas com número de créditos impares. O problema de cursos com variado número de níveis é corrigido quando cada curso poderá possuir sua própria representação de níveis e oferecimento. Disciplinas com mais de uma turma do mesmo nível e horário também serão possíveis. A geração de vários cursos numa única execução do algoritmo também será suportada, propiciando condições semelhantes de geração de horários a todos os cursos gerados.

O refinamento e implementação do novo modelo está em andamento. Ele terá um protótipo para validação de modelo desenvolvido sequencialmente e sua implementação final usará recursos de processamento paralelo e distribuído, usando a biblioteca MPI para suporte a execução paralela.

Referências

FANG, Hsiao-Lan. Genetic Algorithms in Timetabling and Scheduling. 1994. Thesis (Ph.D.) - University of Edinburgh, Scotland.

Lenzi A. L. Desenvolvimento de grades horárias por Algoritmos Genéticos. 2003. Trabalho de conclusão de curso (Graduação). Uniplac, Lages, 2004.

Luger, G. F. Inteligência Artificial estruturas e estratégias para a solução de problemas complexos, Trad. Paulo Engel. 4 ed., BOOKMAN, Porto Alegre. 2004.

Melo R. M. Definição e implementação de uma função de avaliação para um sistema de geração de grade horária que utiliza como método de busca Algoritmos Genéticos. 2003. Trabalho de conclusão de curso (Graduação). Uniplac, Criciúma, 2003.

Pereira, Jean Fretta; BRANCO NETO, Wilson Castello. Uma Abordagem Evolucionária para a Geração de Quadros de Horários. Revista do Ccei Centro de Ciências da Economia e Informática da Universidade da Região da Campanha Urcamp, Bagé RS, v. 5, n. 8, p. 103-111, 2001.

Perone, Christian Samuel. Algoritmo genético para alocação de carga horária. Trabalho de conclusão de curso (Graduação).UPF, Passo Fundo, 2006.

Rela, Leo. Evolutionary computing in search-based software engineering. 2004. Master's thesis – Lappeenranta University of Technology, Departament of Information Technology, Finland.