Timetabling com algoritmos genéticos paralelos

Tiago A. P. Fucilini[1], Matheus L. Lodi[1], Marcelo Trindade Rebonatto[1]

[1]Curso de Ciência da Computação – ICEG – Universidade de Passo Fundo (UPF)
Caixa Postal 611 – 99052-900 – Passo Fundo – RS – Brasil

tiagof182@gmail.com, 97568@upf.br, rebonatto@upf.br

1. Introdução

O problema de designar professores a disciplinas e alocar o conjunto de disciplinas num horário de um curso é um problema de difícil solução manual, principalmente quando o número de disciplinas e professores é elevado. Na maior parte dos casos, ocorre uma restrição em relação aos horários disponíveis dos em virtude de suas demais atividades. 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.

Em muitos casos, o uso de algoritmos evolucionários, em especial o uso de algoritmos genéticos (AG), representam soluções de moderado consumo computacional. Porém, em casos onde o espaço de busca dos algoritmos é elevado, pode ocorrer um aumento considerável da capacidade computacional necessária para sua resolução. Esse aumento pode ocorrer em virtude do consumo de memória ou em relação à necessidade de processamento. Para os casos onde os algoritmos genéticos consomem grande quantidade de recursos computacionais, demandando com isso elevado tempo de execução, o uso de técnicas para a paralelização dos AGs é uma solução que deve ser investigada.

O uso do paralelismo pode proporcionar ao AG condições de explorar melhor e de forma mais aprofundada o espaço de busca. Além disso, caso a paralelização seja realizada em ambientes com diferente espaço de endereçamento de memória (multicomputadores), pode-se aplicar operados genéticos com diferentes parâmetros em cada AG sendo executado, aumentando ainda mais a diversidade das soluções encontradas.

Este texto descreve o projeto de uma solução para o problema de *timetabling* de professores a disciplinas em horários usando algoritmos genéticos. Ele é fortemente centrado no trabalho descrito em [Fucilini et al, 2008], pois pode representar sua continuidade. Ele começa com as principais deficiências do AG descrito e complementa

com o modelo do novo AG, suas estruturas de memória e descreve algumas das possibilidades para a exploração do paralelismo.

2. Resultados e limitações do AG existente

O modelo de algoritmo genético foi implementado e teve seus principais resultados descritos em [Fucilini et al, 2008]. 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 (tarde e noite). A execução não foi encerrada por convergência e sim por número de gerações e obteve indivíduos com resultados validos, ou seja, com nenhuma *hard constraint* 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 disciplina, sendo após o AG novamente executado com dados de oferecimentos do novo curso.

A referida versão 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 granularidade de horários é sempre de 2 horas/aula. Nos casos de turmas práticas divididas entre diferentes professores no mesmo horário (isso ocorre em virtude de divisão de turmas em cursos com apenas uma turno de oferecimento) o AG gera uma hard *constraint*, embora essa *constraint* não seja válida uma vez que os alunos não são os mesmos. A alocação de mais de um curso por execução do AG é limitada, principalmente, em função do número variado de semestres que os cursos podem apresentar.

Esses problemas ensejam a definição de um novo modelo de AG, que suplante essas limitações. Para tanto, estima-se que o tempo de execução seja majorado, possivelmente de forma exponencial. Assim, a implementação no novo modelo de AG terá de fazer uso de paralelismo na execução.

3. Novo Modelo do Algoritmo Genético

Uma nova versão do modelo proposto foi definida para solucionar os problemas detectados. O novo modelo altera a representação cromossômica e, conseqüentemente, funções de fitness e operadores genéticos de mutação e *crossover*.

3.1 Representação do Cromossomo

Grande parte das mudanças e melhorias ocorreu em virtude da nova representação cromossômica da solução. Diferentemente da antiga abordagem em um vetor de duas dimensões, o novo modelo pode ser dividido em três partes, como mostra a figura 1.

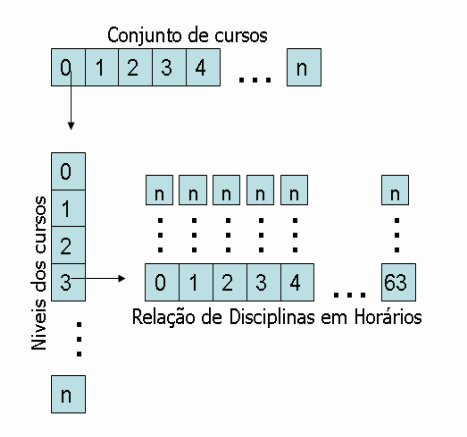

Figura 1 - Representação do novo modelo cromossômico

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 um crédito e poderá ter mais de uma disciplina referenciada nele (Relação de disciplinas em Horários).

3.2 Geração da população inicial

A população inicial é gerada por nível, com um inicializador distinto para cada nível. Este inicializador trabalha percorrendo as disciplinas oferecidas no nível, para cada uma ele encontra todas as relações professor/disciplina referente a ela. Depois de encontrada todas as relações, aleatoriamente uma é escolhida. O mesmo método é utilizado para escolher o *slot* onde a relação será alocada, encontram-se todos os *slots* livres, e aleatoriamente um é escolhido.

3.3 Constraints e fitness

Devido ao tamanho do novo cromossomo essa será a seção critica do algoritmo, onde o maior custo computacional será empregado. As *constraints* podem ser separadas em dois tipos principais: *Hard* e *Soft*. Segundo Fang (1994, p. 49), as 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*: (a) todas as disciplinas oferecidas foram alocadas, (b) disponibilidade do professor no *slot*, (c) professor dando aula no mesmo *slot* em níveis ou cursos diferentes. E é na verificação delas, e principalmente na *constraint* "a", onde boa parte do processamento ocorrerá. As *soft constraints*, ainda estão sendo elaboradas, visando aprimorar avaliação de cada um dos cromossomos. A verificação de se todas as disciplinas oferecidas foram alocadas, consiste em cada nível de todos os cursos, comparar o *timeslot* gerado, como o oferecimento que foi exigido.

Dependendo da quantidade de cursos que está sendo utilizado, além do tamanho do cromossomo, o tamanho da lista de oferecimentos será grande. Para limitar o espaço de busca e comparação, a lista de oferecimentos será gerada em um vetor bidimensional, onde cada uma das linhas representará um curso.

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.

3.4 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. Serão utilizadas duas formas distintas para realizar o *crossover* em ambas foi utilizada a técnica *one-point crossover*. Na primeira 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 segunda vai agir da mesma forma, porém possibilitando a troca dos cursos em sua totalidade. 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.

3.4 Paralelismo

A partir da nova representação cromossômica, mais complexa para manuseio das informações e com elevado consumo de memória, a implementação necessita de um grande esforço computacional para ser executado. Para resolver esse problema à nova implementação está sendo desenvolvida para usar 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 ou máquina

References

Fucilini, Tiago P. et. al, Timetabling com algoritmos genéticos: resultados, restrições e exploração do paralelismo. Hífen, Uruguaiana, RS, v. 32, n. 62, p. 156-163, 2008.

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

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.

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