# Um Algoritmo Evolutivo Híbrido para o Problema de Programação de Horários em Escolas[1]

**Prof. Dr. Marcone Jamilson Freitas Souza**
Departamento de Computação, Universidade Federal de Ouro Preto,
354000.000, Ouro Preto, MG, Brasil, e-mail: marcone@iceb.ufop.br

**Felippe Pereira da Costa**
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto, Campus Universitário,
CEP 35400-000, Ouro Preto, MG, Brasil, e-mail: felippepereiradacosta@yahoo.com.br

**Irce Fernandes Gomes Guimarães**
Graduanda em Engenharia de Produção, Universidade Federal de Ouro Preto, Campus Universitário,
CEP 35400-000, Ouro Preto, MG, Brasil, e-mail: irceguima@bol.com.br

**ABSTRACT**
*It is well known that the timetabling problem is difficult to solve, since the problem has a huge search space and is highly constrained. So, many optimization methods such as Simulated Annealing, Tabu Search and Genetic Algorithm have been applied to find good timetables. In this work a hybrid technique, based on the metaheuristics GRASP, Genetic Algorithms and Tabu Search, is presented as proposal for solving this problem.*

**KEYWORDS:** *School Timetabling, Metaheuristics, Genetic Algorithm.*

## 1. Introdução

O problema de programação de horários em escolas (PPHE) diz respeito à alocação das aulas de uma escola a um conjunto restrito de horários, satisfazendo diversas restrições (Schaefer 1999). A solução manual deste problema é uma tarefa penosa e complexa e normalmente requer vários dias de trabalho (Bardadym 1996). Além do mais, a solução obtida pode ser insatisfatória com relação a vários aspectos. Por exemplo, em função da programação feita, pode haver aulas de uma mesma matéria sendo ministradas em dias consecutivos, com prejuízo na sedimentação da aprendizagem.

Em função de situações como essa, uma atenção especial vem sendo dada à automação do PPHE. Sendo o mesmo NP-difícil (Even, Itai e Shamir 1976), ele é comumente abordado através de técnicas heurísticas, dentre as quais destacam-se as chamadas metaheurísticas. Essas técnicas, ao contrário das heurísticas convencionais, têm caráter geral e são providas de mecanismos para escapar de ótimos locais. Como exemplos de aplicações bem sucedidas de metaheurísticas a problemas de programação de horários pode-se citar: *Simulated Annealing* (Abramson 1991), Busca Tabu (Costa 1994, Burke et al. 2001) e Programação Genética (Ueda et. al. 2001, Carrasco and Pato 2001, Collorni et. al. 1998, Erben and Keppler 1996).

Em geral, no entanto, as metaheurísticas sofrem grande influência das soluções iniciais, isto é, uma solução inicial de boa qualidade induz a um processo de busca mais rápida, com

[1] Souza, M. J. F.; Guimarães, I. F.; Costa, F. P. Um Algoritmo Evolutivo Híbrido para o Problema de Programação de Horários em Escolas. In XXII Encontro Nacional de Engenharia de Produção, 2002, Curitiba. Anais do XXII ENEGEP, Santa Bárbara D´Oeste, ABEPRO, 2002, CD-ROM, 8 p.

produção de soluções finais melhores. A fase de construção da metaheurística GRASP (Feo e Resende 1995) é um dos instrumentos atualmente utilizados em várias aplicações (Souza 2000) para alcançar este objetivo.

Este trabalho propõe uma técnica híbrida envolvendo as metaheurísticas Programação Genética, Busca Tabu e GRASP. Neste algoritmo evolutivo, a população inicial é gerada pelo procedimento construtivo da técnica GRASP. A cada geração deste algoritmo, o melhor indivíduo da população sofre um refinamento através do procedimento de Busca Tabu.

Este trabalho está organizado como segue. Na seção 2 descreve-se o problema. Nas seções 3 e 4 faz-se uma breve descrição do Algoritmo Genético e dos operadores utilizados no problema de programação de horários. Na seção 5 mostra-se como é gerada a população inicial. Na seção 6 faz-se uma descrição sumária da metaheurística Busca Tabu (BT). O algoritmo evolutivo híbrido proposto é apresentado na seção 7, enquanto que na seção 8 apresenta-se seu comportamento típico. Na seção 9 o trabalho é concluído.

## 2. Descrição do problema

O problema considerado para análise é o da programação de horários da Escola Estadual Dom Velloso em Ouro Preto, MG. Esta escola atende o 2º ciclo do ensino fundamental (5º a 8º ano), atendendo 11 turmas no turno matutino e 3 no vespertino.

No processo de programação de horários desta escola são observados vários requisitos, tais como: a) Não permitir a alocação de mais de um professor em uma mesma turma no mesmo horário; b) Evitar quebras de aulas, isto é, aulas não consecutivas de uma matéria para uma turma em um mesmo dia; c) Respeitar o limite diário de aulas de uma mesma matéria para uma mesma turma; d) Eliminar, sempre que possível, as janelas nos horários dos professores; e) Atender ao maior número possível de aulas geminadas; f) Minimizar a quantidade de dias que cada professor necessita ir à escola; g) Espalhar as aulas de uma mesma matéria para uma mesma turma ao longo da semana.

## 3. Algoritmo Genético

Trata-se de uma metaheurística que se fundamenta em uma analogia com processos naturais de evolução, nos quais, dada uma população, os indivíduos com características genéticas melhores têm maiores chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer.

Nos Algoritmos Genéticos (AGs), cada cromossomo (indivíduo da população) está associado normalmente a uma solução do problema e cada gene está associado a uma componente da solução. Um mecanismo de reprodução, baseado em processos evolutivos,é aplicado sobre a população com o objetivo de explorar o espaço de busca e encontrar melhores soluções para o problema. Para atingir esse objetivo, os n indivíduos da população passam por uma fase de reprodução, a qual consiste em selecionar indivíduos para operação de recombinação e/ou mutação. Na operação de recombinação, os genes de dois cromossomos pais são combinados de forma a gerar um ou dois cromossomos filhos, de sorte que para cada cromossomo filho há um conjunto de genes de cada um dos cromossomos pais. A operação de mutação consiste em alterar aleatoriamente uma parte dos genes de cada cromossomo (componentes da solução). Ambas as operações são realizadas com uma certa probabilidade.

Gerados os cromossomos filhos, define-se a população sobrevivente a partir da avaliação de cada cromossomo por uma certa função de aptidão. Os critérios comumente usados para escolher os cromossomos sobreviventes são: 1) aleatório; 2) roleta (onde a chance de sobrevivência de cada cromossomo é proporcional ao seu nível de aptidão) e 3) misto (isto é, uma combinação dos dois critérios anteriores). Em qualquer um desses critérios admite-se,

portanto, a sobrevivência de indivíduos menos aptos, o que é feito de forma a tentar-se escapar de ótimos locais. O método termina, em geral, quando um certo número de populações é gerado ou quando a melhor solução encontrada atinge um certo nível de aptidão ou ainda, quando não há melhora após um certo número de iterações.

## 4. Algoritmo Genético Aplicado ao Problema de Programação de Horários

### 4.1 Representação de uma solução

Neste problema um cromossomo representa um quadro de horário semanal, isto é, um cromossomo é uma matriz $S = (s_{ij})_{m \times n}$ de valores inteiros $s_{ij}$ , sendo $m$ o número de professores e $p$ o número de horários. Uma célula $s_{ij}$ representa a situação de um professor $i$ em um horário $j$. Estas situações são representadas da seguinte forma: (a) Se o professor está disponível, $s_{ij}$ recebe um valor infinito; (b) Se o professor está indisponível, $s_{ij}$ recebe um valor negativo; (c) Se o professor está lecionando, $s_{ij}$ recebe o número correspondente ao da turma para a qual está lecionando.

### 4.2 Operadores genéticos

#### 4.2.1 Operadores do Tipo *Crossover*

Para o processo de cruzamento (*crossover*), o qual é aplicado com uma certa probabilidade, foram considerados dois tipos deste operador (cada qual aplicado, também, com uma certa probabilidade):

***Crossover* Simples**: O *crossover* Simples consiste em escolher aleatoriamente dois quadros de horários (pais) dentro da população para gerar dois novos quadros (filhos), de acordo com a seguinte operação. Escolhe-se deterministicamente um ponto de corte (no nosso caso, 80% do número de professores). O filho 1 recebe todas as aulas que estiverem acima do ponto de corte do pai 1, bem como as aulas abaixo do ponto de corte do pai 2. Entretanto, essas aulas do pai 2 não são automaticamente alocadas, sendo submetidas, inicialmente, a uma verificação de sobreposição. Isto é, inicialmente elas são alocadas na mesma posição do pai somente se não gerarem sobreposição. Para as que gerarem sobreposição, procura-se outros horários onde isto não ocorra. Se ainda assim houver aulas sem alocação, estas são alocadas aleatoriamente.

***Crossover* Ordenado**- O *crossover* ordenado é semelhante ao *Crossover* Simples, diferindo deste apenas com relação ao ponto de corte, que neste caso é aleatório, e à organização das aulas dos pais. Antes de se submeter à operação, cada quadro é ordenado segundo a função de custo dos professores. A seguir, é selecionado aleatoriamente um ponto de corte, sendo o *crossover* efetuado conforme anteriormente. A Figura 2 ilustra esta operação.Nestes cromossomos, cada linha $i$ representa um professor, cada coluna $j$ um horário e cada célula $(i,j)$ a turma em que o professor $i$ está ministrando no horário $j$. A linha tracejada indica o ponto de corte.

| | H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
|---|---|---|---|---|---|---|
| 8 | 5 | 5 | 4 | 4 | 2 | 10 |
| 3 | 0 | 0 | 1 | 1 | 4 | 20 |
| 5 | 1 | 1 | 2 | 2 | 0 | 24 |
| 7 | 4 | 4 | 5 | 5 | 3 | 25 |
| 2 | 3 | 3 | 0 | 0 | 1 | 30 |
| 6 | 2 | 2 | 3 | 3 | 5 | 32 |
| 1 | 0 | 0 | 1 | 1 | 8 | 50 |
| 4 | 5 | 5 | 4 | 4 | 6 | 86 |

| | H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
|---|---|---|---|---|---|---|
| 3 | 7 | 7 | 8 | 8 | 2 | 25 |
| 5 | 6 | 6 | 7 | 7 | 4 | 27 |
| 1 | 2 | 2 | 3 | 3 | 0 | 30 |
| 4 | 3 | 3 | 0 | 0 | 3 | 32 |
| 8 | 4 | 4 | 5 | 5 | 1 | 36 |
| 2 | 1 | 1 | 2 | 2 | 5 | 51 |
| 6 | 6 | 6 | 7 | 7 | 8 | 52 |
| 7 | 7 | 7 | 8 | 8 | 6 | 80 |

Figura 1 – Cromossomos pais ordenados de acordo com a função de custo dos professores

| | H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
|---|---|---|---|---|---|---|
| 8 | 5 | 5 | 4 | 4 | 6 | 10 |
| 3 | 1 | 1 | 2 | 2 | 0 | 20 |
| 5 | 3 | 3 | 0 | 0 | 1 | 24 |
| 7 | 0 | 0 | 1 | 1 | 8 | 25 |
| 2 | 0 | 0 | 1 | 1 | 4 | 30 |
| 6 | 2 | 2 | 3 | 3 | 5 | 32 |
| 6 | 6 | 6 | 7 | 7 | 8* | ** |
| 7 | 7 | 7 | 8 | 8 | 6* | ** |

| | H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 3 | 3 | 0 | 25 |
| 5 | 4 | 4 | 5 | 5 | 1 | 27 |
| 1 | 7 | 7 | 8 | 8 | 2 | 30 |
| 4 | 3 | 3 | 0 | 0 | 3 | 32 |
| 8 | 7 | 7 | 8 | 8 | 6 | 36 |
| 2 | 6 | 6 | 7 | 7 | 4 | 51 |
| 1 | 0 | 0 | 1 | 1 | 8 | ** |
| 4 | 5 | 5 | 4 | 4 | 6* | ** |

Figura 2– Cromossomos filhos gerados pelo operador Crossover Ordenado

As aulas assinaladas com * foram alocadas aleatoriamente. O sinal ** indica que a função de custo do professor ($f_i(s)$) necessitará ser recalculada.

### 4.2.1.Operadores do Tipo mutação

Este operador consiste em trocar uma ou mais aulas, estudando a melhor posição de troca, desde que haja uma diminuição no valor da função de custo do cromossomo. Três tipos de operadores de mutação foram considerados, a saber:

**Mutação de ordem 1:** consiste em trocar a posição de um gene com a de outro na programação de um mesmo professor.

**Mutação de ordem *k***: consiste em trocar as posições de *k* genes contíguos da programação de um professor com outro conjunto de *k* genes contíguos.

**Mutação dia**: É análogo ao anterior, porém os k genes contíguos referem-se à programação de um dia do professor.

Cada um destes operadores é aplicado à programação de um dado professor (escolhido aleatoriamente), com uma certa probabilidade de ocorrência. As figuras 4 e 5 ilustram a aplicação do operador mutação de ordem *k*=2.

| Segunda- Feira | | | | | |
|---|---|---|---|---|---|
| | H1 | H2 | H3 | H4 | H5 |
| ... | ... | ... | ... | ... | ... |
| 4 | 7 | 7 | **4** | **3** | 8 |
| ... | ... | ... | ... | ... | ... |

| Terça – Feira | | | | | |
|---|---|---|---|---|---|
| H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
| ... | ... | ... | ... | ... | ... |
| **7** | **6** | 6 | 5 | 5 | 55 |
| ... | ... | ... | ... | ... | ... |

Figura 3: Programação do professor 4 antes da mutação de ordem 2**.**

| Segunda- Feira | | | | | |
|---|---|---|---|---|---|
| | H1 | H2 | H3 | H4 | H5 |
| ... | ... | ... | ... | ... | ... |
| 4 | 7 | 7 | **7** | **6** | 8 |
| ... | ... | ... | ... | ... | ... |

| Terça – Feira | | | | | |
|---|---|---|---|---|---|
| H1 | H2 | H3 | H4 | H5 | $f_i(s)$ |
| ... | ... | ... | ... | ... | ... |
| **4** | **3** | 6 | 5 | 5 | ** |
| ... | ... | ... | ... | ... | ... |

Figura 4: Programação do professor 4 depois da mutação de ordem 2**.**

Neste exemplo considera-se que a melhor opção de troca envolve as aulas dos horários H3-H4 de segunda-feira e H1-H2 da terça-feira. O símbolo ** indica uma função de custo com valor menor que 55.

### 4.3 Função de custo dos cromossomos

A programação de horários é um problema de decisão multicritério porque para determinar a qualidade de uma programação faz-se necessário considerar diferentes objetivos (custos), tais como: minimizar o número de dias que um professor vai à escola, espalhar as aulas de uma mesma matéria para uma mesma turma, etc.

Para avaliar uma programação de horários, os requisitos do problema são classicamente organizados em: (i) requisitos essenciais, que são aqueles que se não forem

satisfeitos, gerarão uma alocação inviável, como por exemplo, alocar mais de um professor para uma mesma turma em um mesmo horário; (ii) requisitos não-essenciais, que são aqueles cujo atendimento é desejável mas que, se não satisfeitos, não geram programações inviáveis, como por exemplo, apresentar uma distribuição de aulas com janelas para os professores.

Desse modo, uma programação (ou solução) *s* pode ser medida com base em duas componentes, uma de inviabilidade (*g(s)*), a qual mede o não atendimento aos requisitos essenciais, e outra de qualidade (*h(s)*), a qual mede o não atendimento aos requisitos considerados não essenciais. A função de custo de uma solução *s*, a qual deve ser minimizada, pode ser calculada, portanto, pela seguinte expressão: $f(s) = g(s) + h(s)$ (1)

A parcela *g(s)*, que mensura o nível de inviabilidade de uma solução *s*, é avaliada com base na expressão: $$g(s) = \sum_{k=1}^{K} \alpha_k I_k \quad (2)$$

onde *K* é número de medidas de inviabilidade, $I_k$ o valor da *k*-ésima medida de inviabilidade e $\alpha_k$ o peso associado à essa *k*-ésima medida.

A parcela *h(s)*, que mensura a qualidade de uma solução *s*, é avaliada com base na seguinte função: $$h(s) = \sum_{l=1}^{L} \beta_l Q_l \quad (3)$$

onde *L* representa o número de medidas de qualidade, $Q_l$ o valor da *l*-ésima medida de qualidade e $\beta_l$ o peso associado a essa *l*-ésima medida.

Deve ser observado que uma solução *s* é viável se e somente se $g(s) = 0$. 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 tomar $\alpha_k >> \beta_l \quad \forall k, l$, de forma a privilegiar a eliminação das soluções inviáveis.

**4.4 Função de aptidão de um cromossomo**

O papel desta função é avaliar o nível de aptidão, ou seja, a possibilidade de sobrevivência de cada indivíduo gerado pelo AG. A partir deste processo de avaliação pode–se saber o grau de adaptação dos indivíduos e eliminar aqueles que não são aptos ao meio. Para se fazer a avaliação de uma solução *s* é utilizada a seguinte função :

$$faptidão(s) = \begin{cases} x \text{ se } x > 0; \\ 0 \text{ caso contrário;} \end{cases}$$

sendo: $x$ = Média($f(pop)$) + $\alpha\times$ desvio($f(pop)$) - $f(s)$ (4)

*faptidão(s)*= Valor da função de aptidão da solução *s;*

*f(s)*=Função de custo da solução *s;*

média*(f(pop))* = Média da função de custo da população*;*

$\alpha$=Fator que determina o percentual de aptidão de *s;*

desvio*(f(pop))*= Desvio padrão da função de custo da população*;*

**4.5 Seleção**

Nesta fase são escolhidos os indivíduos que farão parte da próxima geração. Neste processo utiliza-se do grau de adaptação de cada um para fazer a seleção. Os valores do grau de adaptação de cada indivíduo *s* são convertidos em números entre zero e um através da seguinte função: *Adaptação(s) = faptidão(s) / faptidão(pop)* (5)

Os indivíduos são selecionados pelo mecanismo de roleta russa, no qual os mais aptos possuem maior probabilidade de sobreviverem, sendo que todos, mesmos os menos aptos, têm chances de sobrevivência.

**5. Geração de uma solução inicial**

Para gerar uma população inicial normalmente são utilizados vários procedimentos, desde os aleatórios até os heurísticos. A geração de soluções aleatórias diversifica o espaço de soluções; no entanto, uma solução aleatória é, em geral, de baixa qualidade, exigindo um tempo de processamento muito elevado para ser melhorada. Por outro lado, soluções gulosas, apesar de serem de boa qualidade, apresentam baixa diversidade. Recentemente, Feo e Resende (1995) idealizaram uma heurística que representa uma alternativa a esses métodos puramente aleatórios ou puramente gulosos. Aplicações bem sucedidas dessa técnica, chamada GRASP, têm sido relatadas na literatura (Souza 2000). Desta forma, optou-se por gerar uma população inicial através da aplicação sucessiva da fase de construção do algoritmo GRASP.

Mais especificamente, cada cromossomo da população inicial do problema de programação de horários é gerado conforme a seguir se descreve. Inicialmente, determinam-se os horários mais críticos, isto é, aqueles que têm o menor número de professores disponíveis. A seguir enumeram-se e ordenam-se todas as aulas, priorizando aquelas mais difíceis de serem alocadas, ou seja, aquelas que têm os professores com maior carga horária e maior número de horários indisponíveis. Forma-se então uma lista restrita de candidatos destas aulas, de tamanho |LRC| e seleciona-se aleatoriamente uma dela. A seguir procura-se alocá-la (usando a ordem dos horários críticos) de forma que todas as restrições essenciais sejam satisfeitas. Não sendo possível admite-se, agora, a violação a apenas uma das restrições essenciais. Persistindo a impossibilidade de alocação tenta-se alocar a aula admitindo que duas restrições essenciais sejam satisfeitas. Essas tentativas de alocação persistem aumentando-se o número de restrições essenciais não atendidas até que a aula seja alocada. Toda vez que uma aula é alocada, os horários críticos são atualizados, bem como as aulas mais difíceis remanescentes.

**6. Busca Tabu**

Busca Tabu (Glover and Laguna 1997) é um procedimento de otimização local que admite soluções de piora para escapar de ótimos locais. Em sua forma original, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho $s'$ da vizinhança N($s$) da solução corrente $s$. Independentemente de f($s'$) ser melhor ou pior que f($s$), $s'$ será sempre a nova solução corrente. Entretanto, apenas esse mecanismo não é suficiente para escapar de ótimos locais, uma vez que pode haver retorno a uma solução previamente gerada. Para evitar isso, o algoritmo usa o conceito de lista tabu. Esta lista define todos os movimentos com um certo atributo como sendo tabu por um determinado número de iterações, conhecido como tempo tabu. Tais movimentos são proibidos a menos que a solução satisfaça a um certo critério de aspiração, em geral que essa solução seja melhor que a melhor solução encontrada até então. Os atributos são escolhidos para prevenir o retorno a soluções visitadas recentemente e são escolhidos por características que são fáceis para detectar. No caso do problema de programação de horários, se o melhor vizinho $s'$ de um quadro de horários $s$ for obtido a partir deste pelo movimento de troca entre a aula do professor $i$ do horário $k_1$ e a aula do horário $k_2$, então considera-se tabu durante |T| iterações qualquer movimento que envolva a troca das aulas do professor $i$ entre os horários $k_1$ e $k_2$.

**7. Algoritmo híbrido**

O algoritmo híbrido proposto é um método evolutivo na qual a população inicial é gerada por um procedimento construtivo (seção 5) e, em cada geração, o melhor indivíduo é

submetido a um refinamento pelo método de Busca Tabu. A Figura 5 apresenta o pseudocódigo do algoritmo.

```
procedimento Híbrido( );
1. Inicializar a população;
2. t ← 0;
3. Gerar uma população inicial P(t), conforme seção 5;
4. Avaliar P(t);
5. enquanto "os critérios de parada não forem satisfeitos" faça
6.          Cruzar selecionados ;
7.          Mutar Resultantes;
8.          Avaliar P(t) ;
9.          Refinar o melhor indivíduo de P(t) usando Busca Tabu;
10.         Selecionar P(t+1) à partir de P(t);
11.         t ← t+1;
12. fim-enquanto;
13. Retornar melhor indivíduo;
fim Híbrido;
```

Figura 5: Algoritmo Evolutivo Híbrido

## 8. Resultados Computacionais Preliminares

Para testar o algoritmo foram utilizados dados relativos à programação de horários da Escola Estadual Dom Velloso de Ouro Preto dos anos 2001 e 2002. Esta escola conta com 22 professores para ministrarem 350 horas-aula para 14 turmas.

O algoritmo foi implementado na linguagem C++ usando o compilador Borland C++ Builder 5.0 e testado em um microcomputador Pentium II, MMX com 300 MHz e 128 MB de RAM. Inicialmente, ele foi submetido a uma bateria preliminar de testes para calibrar os diversos parâmetros existentes (tamanho da lista tabu, probabilidade de crossover e de cada um de seus tipos, probabilidade de mutação e de cada um de seus tipos etc), bem como para escolher os pesos mais adequados para a função custo.

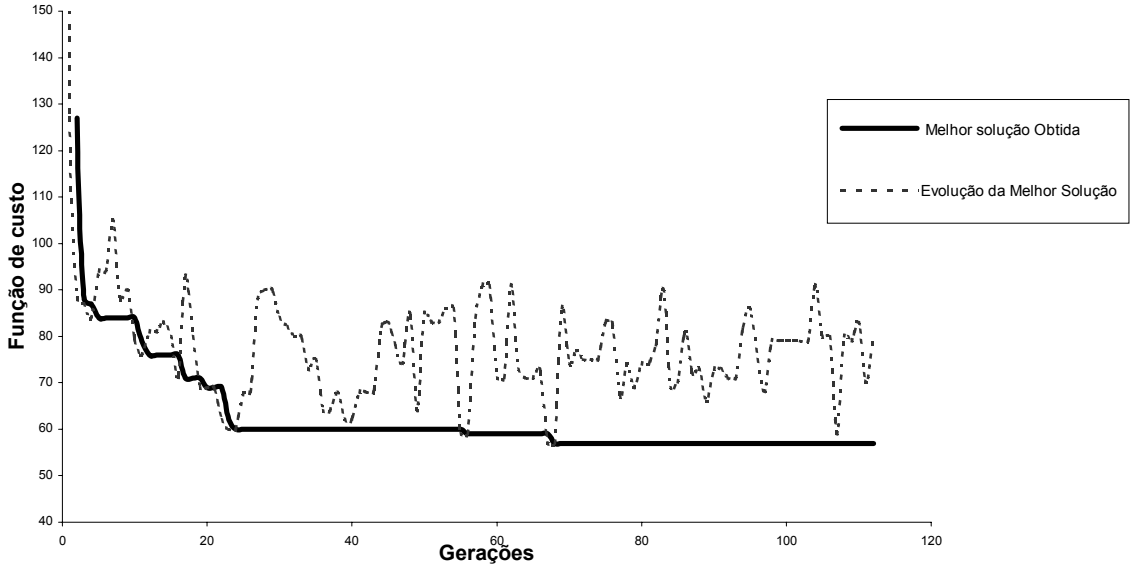


Figura 6: Comportamento típico do algoritmo evolutivo híbrido

A Figura 6 mostra o comportamento típico do algoritmo híbrido. Nesta figura a linha tracejada indica os valores da função de custo do melhor indivíduo obtido em cada geração. A linha contínua indica a evolução da melhor solução gerada ao longo das iterações.

## 9. Conclusões e trabalhos futuros

Foi apresentado um algoritmo evolutivo híbrido para resolver o problema de programação de horários em escolas. Uma população inicial é gerada por um procedimento construtivo adaptativo e parcialmente guloso e, a cada geração, o melhor membro da população é submetido a um refinamento pelo método de Busca Tabu. Nos testes preliminares realizados esse procedimento híbrido conseguiu produzir soluções de melhor qualidade final do que aquele sem a fase de refinamento por Busca Tabu. O algoritmo sem esta fase de refino tem enorme dificuldade para encontrar soluções viáveis. Encontra-se em estudo a utilização de técnicas para reparar as inviabilidades geradas pelos operadores *crossover*, como aquelas descritas em Collorni et al. (1998).

## 10. Agradecimentos

Os autores agradecem ao CNPq pela bolsa de iniciação científica concedida ao projeto, à Escola Estadual Dom Velloso de Ouro Preto pela disponibilização das informações relativas ao processo de elaboração de horários e à Borland Latin América pela cessão de uma licença de uso do *software* C++ Builder Professional 5.0.

## Referências Bibliográficas

Abramson, D. (1991). "Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms", *Management Science*, 37:98-113.

Bardadym, V. A (1996) "Computer-Aided School and University Timetabling: The New Wave", *Lecture Notes in Computer Science*, 1153:22-45, Springer-Verlag.

Burke, E.K., Cowling, P., Landa Silva, J.D. and McC, B. (2001) "Three Methods to Automate the Space Allocation Process in UK Universities", *Lecture Notes in Computer Science*, 2079: 254-276.

Carrasco, M.P. and Pato, M.V. (2001). "A multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem", *Lecture Notes in Computer Science*, 2079:3-17.

Collorni, A., Dorigo, M. and Maniezzo, V. (1998) "Metaheuristics for High School Timetabling", *Computational Optimization and Applications*, 9:275-298.

Costa, D. (1994). "A tabu search algorithm for computing an operational timetable". *European Journal of Operational Research*, 76:98-110.

Erben, W. and Keppler, J. (1996). "A Genetic Algorithm Solving a Weekly Course-Timetabling Problem", *Lecture Notes in Computer Science*, 1153:198-211.

Even, S., Itai, A. and Shamir, A. (1976) "On the complexity of timetabling and multicommodity flow problems", *SIAM Journal of Computation*, 5:691-703.

Feo, T.A. and Resende, M.G.C. (1995) "Greedy randomized adaptive search procedures", *Journal of Global Optimization*, 6:109-133.

Glover, F. and Laguna, M. (1997). *Tabu Search*, Kluver academic Publishers, Boston.

Reeves, C.R. (1996) "Genetic Algorithms", In Reeves, C.R. (ed), Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, 151-196.

Schaefer, A. (1999) "A survey of automated timetabling", *Artificial Intelligence Review*, 13:87-127.

Souza, M.J.F. (2000) "Programação de horários em escolas: uma aproximação por metaheurísticas", Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, UFRJ, Rio de Janeiro.

Ueda, H., Ouchi, D., Takahashi, K. and Miyahara, T. (2001) "A Co-evolving Timeslot/Room Assignment Genetic Algorithm Technique for Universities Timetabling", Lecture Notes in Computer Science, 2079:48-63.