# EXPERIÊNCIAS COM *SIMULATED ANNEALING* E BUSCA TABU NA RESOLUÇÃO DO PROBLEMA DE ALOCAÇÃO DE SALAS

**Marcone Jamilson Freitas Souza**
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, CEP 35400-000, Ouro Preto, MG, Brasil
e-mail: marcone@iceb.ufop.br

**Alexandre Xavier Martins**
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto,
e-mail: xmartins@uai.com.br

**Cássio Roberto de Araújo**
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto,
e-mail: caraujo@uaimail.com.br

**RESUMO**

Este trabalho trata do problema de alocação de salas. Uma vez que este problema é NP-difícil, muitos métodos heurísticos têm sido propostos para resolvê-lo. *Simulated Annealing* (SA) e Busca Tabu (BT) são exemplos de tais métodos que têm sido aplicados com sucesso na resolução deste problema. Esta classe de heurísticas encontra uma boa solução melhorando uma alocação inicial através de técnicas de pesquisa em vizinhança. Contrariamente às heurísticas convencionais, SA e BT permitem movimentos de piora como forma de escapar de ótimos locais. Neste artigo relata-se uma experiência com a utilização destas técnicas e propõe-se uma técnica híbrida SA+TS, a qual combina as características mais apropriadas de SA e BT de forma a obter um procedimento mais eficaz. A eficiência dessa combinação é avaliada comparando-a com as versões puras dos métodos envolvidos.

**PALAVRAS-CHAVE**: Problema de Alocação de Salas, Metaheurísticas, Otimização Combinatória

**ABSTRACT**

This work deals with the classroom assignment problem. Since this scheduling problem is NP-hard, many heuristic methods have been proposed to solve it. Simulated Annealing (SA) and Tabu Search (TS) are examples of such methods that have been applied successfully. This class of heuristics finds a good solution by improving initial assignment through neighborhood search techniques. Contrarily to the conventional heuristics, SA and TS allow some uphill moves to avoid becoming trapped early in a local optimum. In this paper an experience with the application of this techniques for solving the problem is related and a hybrid SA-TS heuristic is proposed. This hybrid method combines the suitable features of SA and TS in order to obtain a procedure more effective than the pure SA and TS methods. The effectiveness of the hybridization is evaluated by comparing the hybrid method with the pure TS and SA methods.

**KEYWORDS**: Classroom Assignment Problem, Metaheuristics, Combinatorial Optimization.

## 1. Introdução

O problema de alocação de salas (PAS) diz respeito à distribuição de aulas, com horários previamente estabelecidos, a salas, respeitando-se um conjunto de restrições de várias naturezas (Schaefer 1999).

A alocação de salas é tratada ou como parte integrante do problema de programação de cursos universitários (*course timetabling*) ou como um problema derivado dele (*classroom assignment*) (Bardadym 1996). Nesta última situação, considera-se que as aulas dos cursos universitários já estejam programadas, isto é, que já estejam definidos os horários de início e término das aulas de cada turma de cada disciplina e o problema, então, é o de alocar essas aulas às salas.

A solução manual deste problema é uma tarefa árdua e normalmente requer vários dias de trabalho. Além do mais, a solução obtida pode ser insatisfatória com relação a vários aspectos. Por exemplo, em função da alocação feita, pode haver em um dado horário um fluxo acentuado de alunos deslocando-se de salas com conseqüente perturbação no ambiente.

Em função de situações como essa, uma atenção especial vem sendo dada à automação deste problema. Entretanto, sua automação não é uma tarefa das mais simples, pois o problema é NP-difícil (Even et al. 1976, Carter and Tovey 1992), o que, em casos reais, dificulta ou até mesmo impossibilita sua resolução por técnicas de programação matemática, ditas exatas.

Assim sendo, esse problema é normalmente abordado através de técnicas heurísticas. Apesar de estas técnicas não garantirem a otimalidade, elas conseguem, em geral, gerar soluções de boa qualidade sem um elevado custo computacional e são relativamente fáceis de serem implementadas. Dentre as heurísticas, destacam-se as chamadas metaheurísticas, as quais, ao contrário das heurísticas convencionais, têm caráter geral e são providas de mecanismos para escapar de ótimos locais.

Dentre as metaheurísticas que vêm sendo aplicadas com relativo sucesso em problemas de programação de horários, destacamos, dentre outras: *Simulated Annealing* (Dowsland 1998, Abramson 1991), Busca Tabu (Burke et al. 2001, Costa 1994, Hertz 1992) e Programação Genética (Ueda et al. 2001, Santos et al. 1997, Erben and Keppler 1996, Rich 1996).

Neste trabalho relata-se uma experiência com a utilização das técnicas *Simulated Annealing* (SA) e Busca Tabu (BT) na resolução do problema de alocação de salas e propõe-se, em função dos resultados obtidos, uma metodologia que combina essas duas técnicas. Nessa técnica híbrida proposta, uma solução inicial é construída por um procedimento que segue as idéias da fase de construção do método GRASP (Feo and Resende 1995). Essa solução é então submetida ao método SA e a solução resultante deste é refinada pelo método BT. Os métodos SA e BT são projetados para trabalhar com uma estrutura de vizinhança que combina dois tipos diferentes de movimentos.

A idéia de combinar o mecanismo de construção GRASP e os métodos SA e BT surgiu após a verificação dos seguintes fatos observados durante a fase de testes: (a) Partindo-se de uma solução inicial de baixa qualidade, o método BT necessitava de muito tempo de processamento para alcançar uma boa solução; (b) Partindo-se de uma boa solução inicial, o método BT conseguia encontrar soluções de melhor qualidade do que aquelas obtidas pelo método SA; (c) Só o mecanismo de construção GRASP não era suficiente para gerar uma boa solução inicial para o método BT; (d) O método SA conseguia gerar uma boa solução inicial com um baixo custo computacional.

Este trabalho está organizado como segue. Na seção 2 descreve-se o problema e na seção seguinte mostra-se como ele é modelado. A seção 4 trata da descrição do procedimento de construção de uma solução inicial. Nas seções 5 e 6 faz-se uma breve descrição dos algoritmos *Simulated Annealing* e Busca Tabu, respectivamente. Na seção 7 apresenta-se o algoritmo híbrido proposto. Nas seções 8 e 9 apresentam-se os resultados relativos à comparação entre a técnica híbrida proposta e as técnicas *Simulated Annealing* e Busca Tabu, bem como as conclusões obtidas.

Salas

| Horários | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **1** | 3 | | | | 4 |
| **2** | 3 | | 1 | 6 | 4 |
| **3** | 3 | 5 | | 6 | 7 |
| **4** | | 5 | 2 | 6 | 7 |
| **5** | 12 | | 2 | | |
| **6** | 12 | 13 | 11 | 9 | |
| **7** | | 13 | 11 | 9 | 10 |
| **8** | **8** | | 11 | | 10 |
| **9** | **8** | | | | 10 |
| | | | | | |

Figura 1 – Exemplo de uma alocação

**3.2 Estrutura de vizinhança**

Dada uma solução *s*, para atingir uma solução *s'*, onde *s'* é dito vizinho de *s*, são usados dois tipos de movimento: Alocação e Troca.

O movimento de alocação consiste em realocar as aulas de uma dada turma e sala a uma outra sala que esteja vazia nos horários de realização das aulas. Para a realização desse movimento é exigido que a sala que receberá as aulas de uma turma esteja disponível nos horários das aulas. Este tipo de movimento é ilustrado na Figura 2.

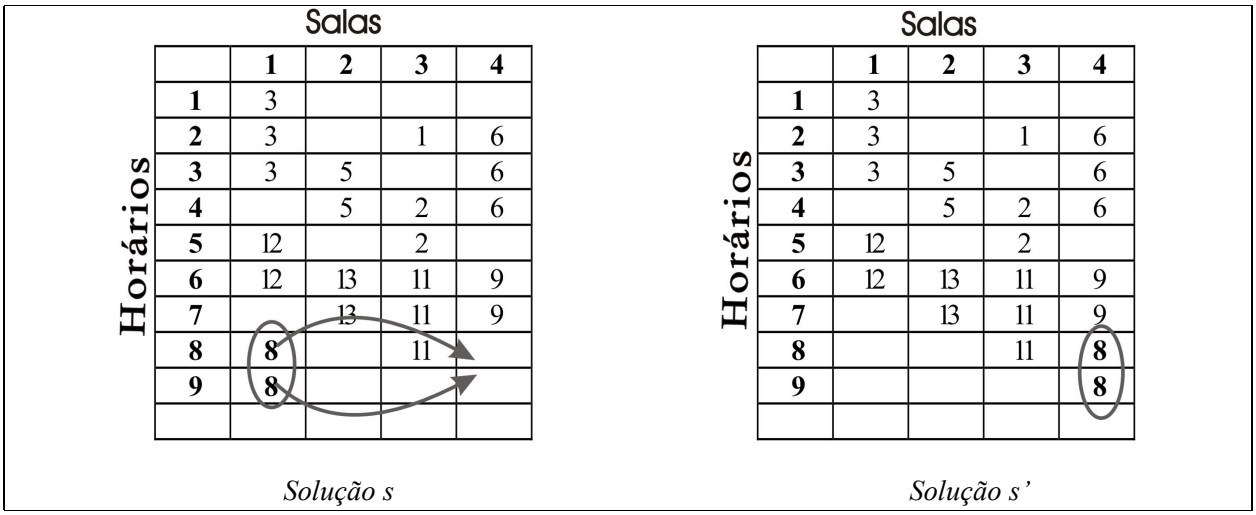
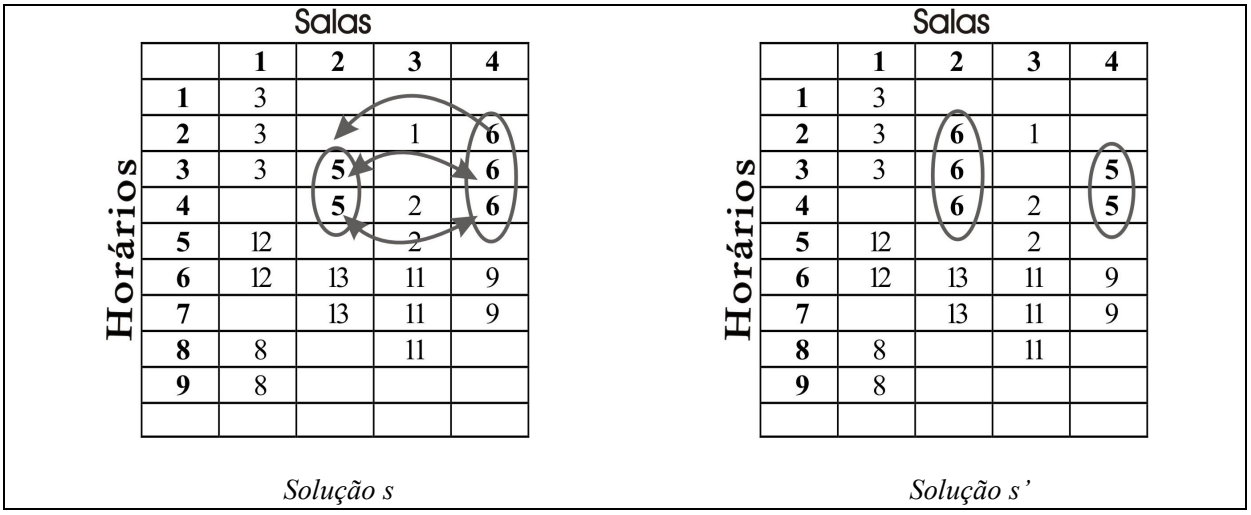



Figura 2 - Movimento de Alocação

Nesta figura, as aulas da turma 8 realizada nos horários 8 e 9 na sala 1 são transferidas para a sala 4.

Já o movimento de troca consiste em trocar de sala as aulas de duas turmas realizadas em um mesmo bloco de horários. Este tipo de movimento é ilustrado na Figura 3.




| Horários \ Salas | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 3 | | | |
| 2 | 3 | | 1 | 6 |
| 3 | 3 | 5 | | 6 |
| 4 | | 5 | 2 | 6 |
| 5 | 12 | | 2 | |
| 6 | 12 | 13 | 11 | 9 |
| 7 | | 13 | 11 | 9 |
| 8 | 8 | | 11 | |
| 9 | 8 | | | |

| Horários \ Salas | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 3 | | | |
| 2 | 3 | 6 | 1 | |
| 3 | 3 | 6 | | 5 |
| 4 | | 6 | 2 | 5 |
| 5 | 12 | | 2 | |
| 6 | 12 | 13 | 11 | 9 |
| 7 | | 13 | 11 | 9 |
| 8 | 8 | | 11 | |
| 9 | 8 | | | |

Figura 3 - Movimento de Troca

Nesta figura, as aulas das turmas 5 e 6 são permutadas de sala. As aulas da turma 5 realizadas na sala 2 nos horários 3 e 4 são transferidas para a sala 4, enquanto que as aulas da turma 6 realizadas na sala 4 nos horários 2, 3 e 4 são transferidas para a sala 2.

Para a realização desse movimento exige-se que nos horários envolvidos as salas estejam vazias ou com aulas apenas das turmas relacionadas com a operação. Desta forma, não é permitido, por exemplo, permutar as aulas das turmas 2 e 6, porque no horário 2 a sala 3 está ocupada com uma aula da turma 1, impedindo que as aulas da turma 6 sejam transferidas da sala 4 para a sala 3.

Na estrutura de vizinhança N($s$) considerada, diz-se que uma solução $s' \in N(s)$ é um vizinho de $s$ se ela pode ser acessada a partir desta através de um movimento ou de alocação ou de troca.

**3.3 Função objetivo**

Para avaliar uma alocação, os requisitos do problema são primeiramente divididos em duas categorias: (i) requisitos essenciais, que são aqueles que se não forem satisfeitos, gerarão uma alocação inviável; (ii) requisitos não-essenciais, que são aqueles cujo atendimento é desejável mas que, se não satisfeitos, não gerarão alocações inviáveis. No caso em análise, são essenciais os requisitos (a), (b) e (c) descritos na seção 2 e não-essenciais os requisitos (d), (e), (f) e (g).

Desse modo, uma alocação (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 avalia o não atendimento aos requisitos considerados não-essenciais. A função objetivo $f$ que associa cada solução $s$ do espaço de soluções a um número real $f(s)$, e que deve ser minimizada, pode ser expressada, portanto, na forma:

$$f(s) = g(s) + h(s) \tag{1}$$

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

$$g(s) = \sum_{k=1}^{K} \alpha_k I_k \tag{2}$$

onde $K$: número de medidas de inviabilidade;
$I_k$: valor da $k$-ésima medida de inviabilidade.
$\alpha_k$: peso associado à $k$-ésima medida de inviabilidade;

Já a componente $h(s)$, que mensura a qualidade de uma solução $s$, é avaliada por:

$$h(s) = \sum_{l=1}^{L} \beta_l Q_l \qquad (3)$$

onde $L$: número de medidas de qualidade;
$Q_l$: valor da $l$-ésima medida de qualidade;
$\beta_l$: peso associado à $l$-ésima medida de qualidade;

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. Observa-se, também, que algumas das medidas de qualidade são conflitantes com outras como, por exemplo, encontrar uma utilização eficiente de espaço (requisito (f)) e agrupar todas as aulas semanais dos alunos de um mesmo curso e período em uma mesma sala de aula (requisito (e)). Esse conflito aparece porque muitas disciplinas são compartilhadas com vários cursos, enquanto outras são ministradas apenas para um determinado curso. Dessa forma, as turmas das disciplinas são geralmente de tamanhos diferentes. Entretanto, faz-se necessário estabelecer, através de uma relação de pesos, um compromisso adequado de atendimento aos requisitos envolvidos.

## 4. Geração de uma solução inicial

Uma solução inicial para o problema de alocação de salas é gerada por um procedimento construtivo que segue as idéias da fase de construção do método GRASP (Feo and Resende 1995).

Inicialmente, toma-se a aula ainda não alocada da turma com maior demanda e constrói-se uma lista restrita de candidatos (LRC) das salas vagas nos horários da aula, ordenadas pela capacidade. Caso não exista uma sala vaga no horário, uma sala virtual é criada e a lista se resume a essa sala. A seguir, uma dessas |LRC| salas é escolhida aleatoriamente para receber a aula. Esse procedimento continua até que todas as aulas sejam alocadas.

## 5. *Simulated Annealing*

*Simulated Annealing* é um método de busca local que aceita movimentos de piora para escapar de ótimos locais. Tal como os métodos de busca local tradicionais, ele requer que o problema seja especificado em termos de um espaço de soluções com uma estrutura de vizinhança definida sobre ele, bem como uma função custo mapeando cada solução em um custo ou valor numérico. O processo se inicia com um membro qualquer do espaço de soluções, normalmente gerado aleatoriamente, e seleciona um de seus vizinhos randomicamente. Se este vizinho for melhor que o original ele é aceito e substitui a solução corrente. Se ele for pior por uma quantidade $\Delta$, ele é aceito com uma probabilidade $e^{-\Delta/T}$, onde T decresce gradualmente conforme o progresso do algoritmo. Esse processo é repetido até que T seja tão pequeno que mais nenhum movimento seja aceito. A melhor solução encontrada durante a busca é tomada como uma boa aproximação para a solução ótima. Originalmente, *Simulated Annealing* foi derivado de simulações em termodinâmica e por esta razão o parâmetro T é referenciado como temperatura e a maneira pela qual ela é reduzida é chamada de processo de resfriamento.

O pseudocódigo do algoritmo é apresentado pela Figura 4. Detalhes adicionais desse algoritmo podem ser encontrados em Dowsland (1993).

**procedimento** *SA*

1. Seja $s_0$ uma solução inicial, $T_0$ a temperatura inicial, $\alpha$ a taxa de resfriamento e SAmax o número máximo de iterações para se atingir o equilíbrio térmico;
2. $s \leftarrow s_0$; {Solução corrente}
3. $s' \leftarrow s$; {Melhor solução obtida até então}
4. $T \leftarrow T_0$; {Temperatura corrente}
5. IterT $\leftarrow 0$; {Número de iterações na temperatura T}
6. enquanto (T > 0) faça
7. enquanto (IterT < SAmax) faça
8. IterT $\leftarrow$ IterT + 1;
9. Gere um vizinho qualquer $s' \in N(s)$;
10. $\Delta = f(s') - f(s)$;
11. se ($\Delta < 0$)
12. então
13. $s \leftarrow s'$;
14. se $f(s') < f(s^*)$ então $s^* \leftarrow s'$;
15. senão
16. Tome $x \in [0,1]$;
17. se $x < e^{-\Delta/T}$ então $s \leftarrow s'$;
18. fim-se;
19. fim-enquanto;
20. $T \leftarrow \alpha \times T$;
21. IterT $\leftarrow 0$;
22. fim-enquanto;
23. Retorne $s^*$;

**fim** *SA;*

Figura 4: Algoritmo *Simulated Annealing*

## 6. Busca Tabu

De forma semelhante ao método *Simulated Annealing*, Busca Tabu é um procedimento de otimização local que admite soluções de piora. Em sua forma clássica, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho *s'* de um subconjunto *V* 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 que têm 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 *A*, 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 alocação de salas, apesar de se trabalhar com dois tipos diferentes de movimentos, utilizou-se somente uma lista tabu. Por exemplo, se o movimento realizado é de alocação da aula do horário *k* e sala *j* para a sala *i* ou de troca da aula do horário *k* e sala *i* com a da aula realizada na sala *j*, então considera-se tabu durante |T| iterações qualquer movimento envolvendo a alocação da aula do horário *k* e sala *i*..

O pseudocódigo do algoritmo é apresentado pela Figura 5. Detalhes adicionais desse algoritmo podem ser encontrados em Glover (1986, 1997).

**procedimento** *BT*
1. Seja $s_0$ solução inicial;
2. $s^* \leftarrow s$; {Melhor solução obtida até então}
3. Iter ← 0; {Contador do número de iterações}
4. MelhorIter ← 0; {Iteração mais recente que forneceu $s^*$}
5. Seja BTmax o número máximo de iterações sem melhora em $s^*$;
6. T ← ∅; {Lista Tabu}
7. Inicialize a função de aspiração A;
8. enquanto (Iter – MelhorIter ≤ BTmax) faça
9. Iter ← Iter + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que o movimento $m$ não seja tabu ($m \notin T$) ou $s'$ atenda a condição de aspiração ($f(s') < A(f(s))$);
11. T ← T – {movimento mais antigo} + {movimento que gerou $s'$};
12. $s \leftarrow s'$;
13. se $f(s) < f(s^*)$ então
14. $s^* \leftarrow s$;
15. MelhorIter ← Iter ;
16. fim-se;
17. Atualize a função de aspiração A;
18. fim-enquanto;
19. Retorne $s^*$;
**fim** *BT*;

Figura 5: Algoritmo Busca Tabu

## 7. Algoritmo híbrido

O algoritmo híbrido proposto é um método de 3 fases. Na primeira fase constrói-se uma solução inicial (seção 4). Na segunda, essa solução construída é submetida ao Algoritmo *Simulated Annealing* (seção 5) e a resultante deste é, na última fase, refinada pelo método de Busca Tabu (seção 6).

O pseudocódigo do algoritmo híbrido SA+BT é apresentado na Figura 6.

**procedimento** *SA+BT*
1. $s^0 \leftarrow$ ConstruaSolucaoInicial();
2. $s^1 \leftarrow SA(s^0)$;
3. $s^* \leftarrow BT(s^1)$;
**fim** *SA+BT;*

Figura 6**:** Algoritmo Híbrido *Simulated Annealing* + Busca Tabu

## 8. Resultados Computacionais

Os algoritmos foram implementados na linguagem C++ usando o compilador Borland C++ Builder 5.0 e testados em um microcomputador PC AMD Atlhon, 800 MHz, com 128 MB de RAM sob sistema operacional Windows 2000.

Para testar os algoritmos foram usados dados relativos à distribuição de salas do segundo semestre letivo do ano 2001 (ICEB2001/2), acrescido de dois outros problemas teste, um com 17 salas (Teste17) e outro com 22 salas (Teste22). Esses dois últimos problemas foram gerados aleatoriamente, mantendo-se uma proporção entre a quantidade de horários de aulas e horários disponíveis em salas semelhante à

existente no problema real ICEB2001/2. Algumas das características desses problemas encontram-se explicitadas na Tabela 1.

| Instância | Número de salas | Número de turmas | Número de horas-aula |
|---|---|---|---|
| **Teste17** | 17 | 214 | 713 |
| **ICEB2001/2** | 20 | 233 | 763 |
| **Teste22** | 22 | 281 | 938 |

Tabela 1: Características das instâncias consideradas

Os algoritmos SA e BT passaram inicialmente por uma bateria preliminar de testes visando à calibragem de seus parâmetros. Foram os seguintes os parâmetros utilizados, conforme notação adotada nas seções 5 e 6: SAmax = 1000 (número de iterações em uma dada temperatura), $T_0 = 1000$ (temperatura inicial), $\alpha = 0.95$ (taxa de resfriamento), $|T| = 100$ (tamanho da lista tabu); BTmax = 1000 (número máximo de iterações sem melhora). Nesta bateria de testes verificou-se que o método BT, ao contrário do método SA, era fortemente influenciado pela solução inicial, isto é, soluções iniciais de baixa qualidade exigiam um tempo de processamento muito elevado para serem melhoradas. Assim, para efeito de comparação com os métodos SA e SA+BT, ainda que estes não fossem significativamente dependentes de uma boa solução inicial, optou-se por gerar soluções iniciais conforme seção 4, utilizando uma lista restrita de candidatos de tamanho $|LRC| = 3$ (também determinado empiricamente), já que o procedimento lá descrito cumpre o objetivo de construir soluções de boa qualidade.

Para cada problema teste foram realizadas 5 execuções dos algoritmos SA, BT e SA+BT, cada qual com uma semente diferente de números aleatórios. Utilizou-se como critério de parada um tempo de execução de 5400 segundos. Para o algoritmo SA (na sua versão pura), sempre que o sistema tornava-se estável e o tempo total de execução não era atingido, fazia-se um reaquecimento a partir da última solução gerada, tomando-se como temperatura de partida a metade da temperatura inicial. Para o algoritmo BT (puro ou associado ao método híbrido SA+BT) analisou-se a cada iteração apenas um percentual da vizinhança completa de uma solução, uma vez que a análise de toda a vizinhança exigia um tempo de processamento muito elevado. No entanto, sempre que este procedimento entrava em uma região plana (situação na qual não havia movimentos de melhora ou piora em NUMPLANAS=10 iterações consecutivas), aumentava-se a lista tabu progressivamente até que fosse encontrado um movimento de melhora ou piora, situação na qual a lista tabu voltava ao seu tamanho inicial. Adicionalmente, passadas BTmax/2 iterações sem melhora sobre a melhor solução encontrada durante toda a pesquisa, acionava-se a análise na vizinhança completa da solução corrente. Encontrada uma solução melhor, a análise voltava a ser parcial. O subconjunto da vizinhança analisada a cada iteração da Busca Tabu era assim escolhido: Na primeira iteração consideravam-se as aulas que se iniciavam nos horários compreendidos entre 1 e 20. Na segunda iteração eram consideradas as aulas que se iniciavam nos horários entre 2 e 21. Este mecanismo prosseguia até que fossem consideradas as aulas que se iniciavam nos horários compreendidos entre 71 e 90, ao fim do qual voltava-se a analisar as aulas que se iniciavam nos horários 1 a 20.

A Tabela 2 mostra, para cada instância considerada e para cada método, o melhor valor pelo método, o desvio percentual médio em relação à melhor solução encontrada por todos os métodos, bem como o tempo de CPU gasto para encontrar a melhor solução.

| Instância | Algoritmo | Melhor Solução | Tempo de CPU (segundos) | Desvio (%) |
|---|---|---|---|---|
| **Teste17** | SA | 7320 | 4767 | 4.93 |
| | BT | 7510 | 2099 | 5.94 |
| | SA+BT | **7144** | 5199 | **1.29** |
| **ICEB2001/2** | SA | 10600 | 5358 | 16.39 |
| | BT | 10191 | 5386 | 34.81 |
| | SA+BT | **9208** | 4532 | **8.09** |
| **Teste22** | SA | 26329 | 2835 | 10.87 |
| | BT | 28677 | 5276 | 27.7 |
| | SA+BT | **25212** | 1655 | **7.35** |

Tabela 2: Resultados computacionais

A Figura 7 ilustra a evolução da melhor solução encontrada pelos algoritmos SA, BT e SA+BT em uma execução do problema Teste17.

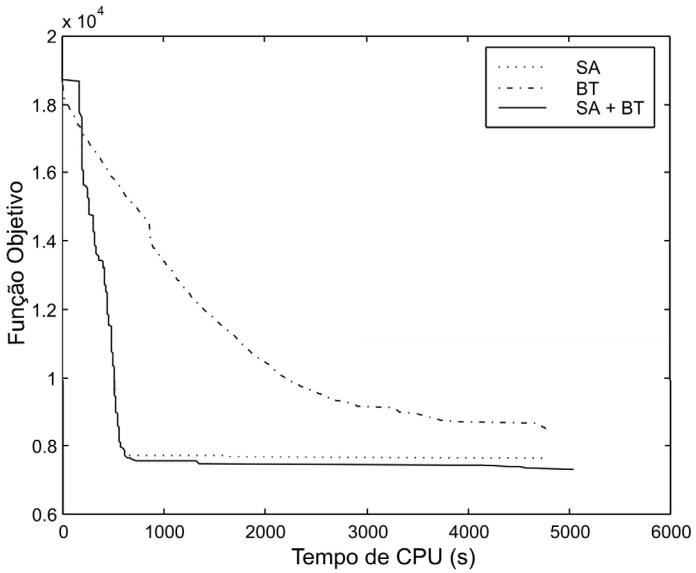



Figura 7: Comportamento típico dos algoritmos SA, BT e SA+BT

**9. Conclusões e trabalhos futuros**

Foi apresentada uma heurística híbrida para resolver o problema de alocação de salas. Uma solução inicial é gerada por um procedimento construtivo parcialmente guloso e submetida à heurística *Simulated Annealing*. A solução final desse método é, então, refinada por um algoritmo de Busca Tabu.

Conforme pode ser observado a partir da Tabela 2 e Figura 7, o procedimento híbrido conseguiu produzir as melhores soluções finais em todos os casos. Esse procedimento mostrou também ser mais robusto, ao gerar soluções finais com menor desvio em relação às melhores soluções encontradas para cada instância. Esses fatos comprovam a supremacia da técnica híbrida sobre as técnicas Busca Tabu e *Simulated Annealing* tomadas isoladamente.

Em vista do desempenho satisfatório da técnica proposta, o ICEB adotou o sistema desenvolvido, tendo-o utilizado na confecção do horário do primeiro semestre letivo de 2002. Atualmente encontra-se

em desenvolvimento a inclusão de novos requisitos no modelo e de algumas facilidades de interface para o usuário.

## 10. Agradecimentos

Os autores agradecem à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG, processo nº 85085/01) e ao Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto pelo suporte financeiro ao projeto, bem como 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.

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

Carter, M.V. and Tovey, C.A. (1992). "When Is the Classroom Assignment Problem Hard?", *Operations Research*, 40:S28-S39.

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

de Werra, D. (1995) "An introduction to timetabling", *European Journal of Operational Research*, 19:151-162.

Dowsland, K.A. (1993) "Simulated Annealing", In Reeves, C.R. (ed), Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, 20-69.

Dowsland, K.A. (1998). "Off-the-Peg or Made-to-Measure? Timetabling and Scheduling with SA and TS", Lecture Notes in Computer Science, 1408:37-52.

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. (1986) "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, 5: 553-549.

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

Hertz, A. (1992) "Tabu search for large scale timetabling problems", *European Journal of Operational Research*, 54:39-47.

Rich, D.C. (1996) "A Smart Genetic Algorithm for University Timetabling", *Lecture Notes in Computer Science*, 1153: 181-197.

Santos, A.M., Marques, E. and Ochi, L.S. (1997). "Design and implementation of a timetable system using genetic algorithm". Second International Conference on Practice and Theory of Automated Timetabling, Toronto, Canada.

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

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.