Métodos de Pesquisa em Vizinhança Variável aplicados ao Problema de Alocação de Salas

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

Alexandre Xavier Martins
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto
Campus Universitário, CEP 35400-000, Ouro Preto, MG, e-mail: xmartins@uai.com.br

Cássio Roberto de Araújo
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto
Campus Universitário, CEP 35400-000, Ouro Preto, MG, e-mail: craraujo@uaimail.com.br

Francisco Wagner Azevedo Costa
Graduando em Engenharia de Produção, Universidade Federal de Ouro Preto,
Campus Universitário, CEP 35400-000, Ouro Preto, MG, e-mail: fwcosta@yahoo.com.br

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. A recent heuristic, so-called Variable Neighborhood Search (VNS), finds a good solution by improving initial assignment through the change of neighborhood in the search. This technique is very easy to implement and does not need many parameters such as Tabu Search and Simulated Annealing. In this paper we test two versions of the local search phase of this technique applied to the classroom assignment problem. Some computational experiments are related.

KEYWORDS: *Classroom Assignment, Timetabling, Metaheuristics.*

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. Sendo o problema NP-difícil (Even et al, Carter and Tovey 1992), ele é normalmente abordado através de técnicas heurísticas. Dentre essas técnicas, destacam-se as chamadas metaheurísticas, as quais, ao contrário das heurísticas convencionais, são providas de mecanismos para escapar de ótimos locais.

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

Neste trabalho aplica-se uma técnica recente, baseada em trocas sistemáticas de vizinhança durante a pesquisa, conhecida como Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS), para resolver o problema. 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 VNS, o qual faz uso de três estruturas diferentes de vizinhança. Duas versões da fase de busca local deste método são testadas. Na primeira versão, faz-se uma busca local usando o Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND). Na segunda, procura-se apenas o melhor vizinho na estrutura de vizinhança corrente da solução em análise.

2. Descrição do problema

O problema considerado para análise é o do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). Trata-se de um instituto que recebe, em média, 1500 alunos por semestre e que oferece cerca de 250 turmas de disciplinas em aulas nos horários matutino, vespertino e noturno. As aulas são realizadas de segunda a sábado à tarde, mas a maioria delas está concentrada de terça a quinta-feira. Para a realização das aulas das turmas estão disponíveis 20 salas de aulas e 29 laboratórios específicos. Os horários de aula das turmas são confeccionados previamente pelos departamentos e encaminhados à diretoria do instituto para que esta faça a alocação das turmas às salas. As aulas que requerem laboratórios são alocadas pelos próprios departamentos e não fazem parte do problema objeto deste trabalho.

No processo de alocação de aulas a salas no ICEB, são observados vários requisitos, tais como: (a) Em uma mesma sala e horário não pode haver mais de uma aula; (b) Uma sala não pode receber uma turma cuja quantidade de alunos seja superior à sua capacidade; (c) Sempre que possível, alocar a uma mesma sala alunos de um mesmo curso e período; (d) Alocar as aulas das disciplinas que demandam recursos especiais (por exemplo, *datashow*) em salas que possuem esses recursos; (e) Certas salas têm restrições de uso e só podem ser usadas em situações especiais; (f) Utilizar o espaço das salas eficientemente, isto é, evitar alocar aulas de turmas pequenas em salas de maior capacidade; (g) Algumas salas têm alguns horários previamente reservados para a realização de outras atividades e nesses horários essas salas estão indisponíveis; (h) Se possível, cada uma das salas deve ser deixada vazia em pelo menos um horário ao longo do dia, de forma a possibilitar sua limpeza.

A alocação de salas no ICEB é feita manualmente e é um processo que se inicia normalmente duas semanas antes do início do período letivo, com base em uma pré-matrícula feita pela própria instituição. Dois dias antes do início das aulas, os alunos têm direito a fazer um ajuste de matrícula. Esta é uma situação que agrava o problema, uma vez que com este ajuste muitas turmas são reduzidas ou ampliadas ou mesmo criadas e as alocações anteriormente feitas têm que ser modificadas. Como não há tempo hábil para efetuar manualmente todas essas correções, em geral o semestre letivo se inicia com uma distribuição de salas que não agrada. Em muitos casos, turmas de 50 alunos são alocadas

em salas com capacidade para 40 alunos! A correção dessas distorções ocorre ao longo das primeiras semanas do período letivo e é sempre motivo de transtorno, porque para fazer essas correções é necessário realocar as aulas de várias outras turmas.

3. Modelagem do problema

3.1 Representação

Uma alocação (solução) do problema é representada por uma matriz $S = (s_{ij})_{mxn}$, onde m representa o número de horários reservados para a realização das aulas e n o número de salas disponíveis. Em cada célula s_{ij} é colocado o número da turma t alocada ao horário i e sala j. Uma célula vazia indica que a sala j está desocupada no horário i. Um exemplo simples de representação é dado pela Figura 1.

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

Esta figura mostra, por exemplo, que na sala 4 os horários 2 a 4 e 6 e 7 estão ocupados com aulas das turmas 6 e 9, respectivamente. Nos demais horários esta sala está desocupada.

Figura 1 – Exemplo de uma alocação

3.2 Estruturas 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) para definir três estruturas diferentes de vizinhança, a saber: $N^{(1)}(s)$, $N^{(2)}(s)$ e $N^{(3)}(s)$.

O movimento de alocação (chamado movimento 1-*optimal*) consiste em realocar as aulas de uma dada turma e sala a uma outra sala que esteja vazia nos horários 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.

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

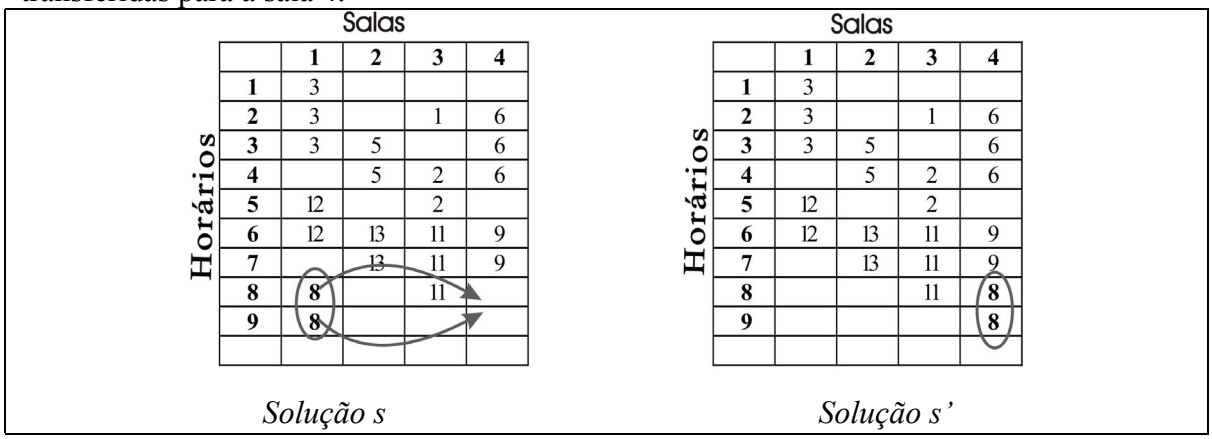

Figura 2 - Movimento de Alocação (1- *Optimal*)

O conjunto de todos os vizinhos de s gerados através de movimentos apenas 1-*optimal* define a estrutura de vizinhança $N^{(1)}(s)$.

Já o movimento de troca (chamado movimento 2-*optimal*) consiste em trocar de sala as aulas de duas turmas realizadas em um mesmo bloco de horários. Este tipo de movimento encontra-se ilustrado na Figura 3.

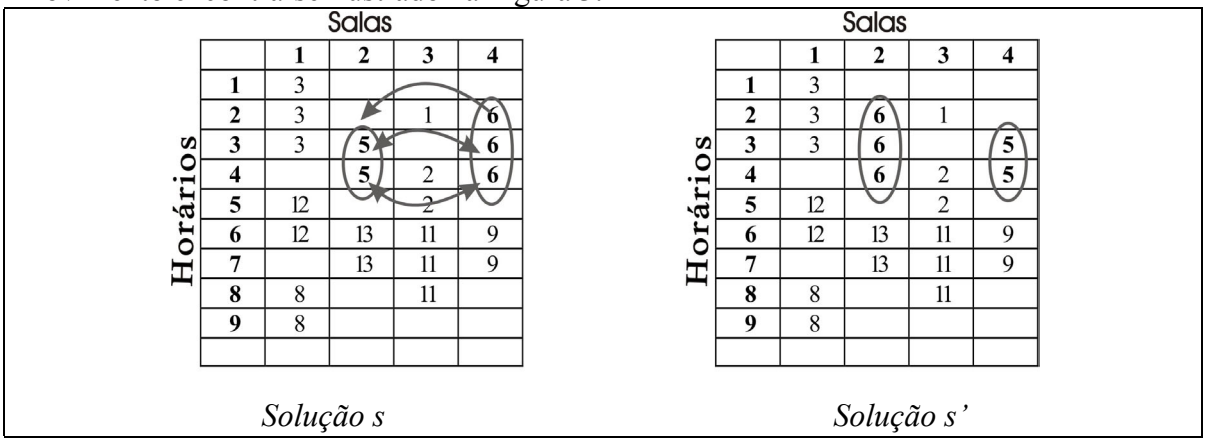

Figura 3 - Movimento de Troca (2- *Optimal*)

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.

O conjunto de todos os vizinhos de s gerados a partir de movimentos do tipo 2-*optimal* define a estrutura de vizinhança $N^{(2)}(s)$.

Finalmente, considera-se uma terceira estrutura de vizinhança, $N^{(3)}(s)$, na qual 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 1-*optimal* ou 2-*optimal*.

3.3 Função de avaliação

O problema de alocação de salas é um problema de decisão multicritério porque para determinar a qualidade de uma alocação faz-se necessário considerar diferentes objetivos, tais como: encontrar uma utilização eficiente de espaço e manter os alunos de mesmo curso e período em uma mesma sala de aula durante todo o período letivo.

Para avaliar uma alocação, os requisitos do problema são divididos em duas categorias: (i) requisitos essenciais, que são aqueles que se não forem satisfeitos, gerarão uma alocação inviável, como por exemplo, alocar duas ou mais turmas em uma mesma sala e horário; (ii) requisitos não-essenciais, que são aqueles cujo atendimento é desejável mas que, se não satisfeitos, não geram alocações inviáveis, como por exemplo, não alocar as diversas aulas semanais de uma dada turma em uma mesma sala ou disponibilizar uma sala muito grande para uma turma com poucos alunos.

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 mede o não atendimento aos requisitos considerados não essenciais. Assim, a função de avaliação de uma solução s, $f(s)$, que deve ser minimizada, pode ser calculada na forma: $f(s) = g(s) + h(s)$ (1)

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

onde K representa o número de medidas de inviabilidade, I_k o valor da k-ésima medida de inviabilidade e α_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: $h(s) = \sum_{l=1}^{L} \beta_l Q_l$ (3)

onde L representa o número de medidas de qualidade, Q_l o valor da l-ésima medida de qualidade e β_l o peso associado à 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. 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. A seguir, uma dessas |LRC| salas é escolhida aleatoriamente para receber a aula. Esse procedimento continua até que todas as aulas sejam alocadas.

5. Método de Pesquisa em Vizinhança Variável (VNS)

O Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Contrariamente a outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança. O pseudocódigo do algoritmo é apresentado pela Figura 4. Detalhes adicionais desse algoritmo podem ser encontrados em Mladenovic and Hansen (1997, 1999).

```
procedimento VNS
1.  Seja s0 uma solução inicial e r o número de estruturas diferentes de vizinhança;
2.  s ← s0;                    {Solução corrente}
3.  enquanto (Critério de parada não satisfeito) faça
4.     k ← 1;                  {Tipo de estrutura de vizinhança}
5.     enquanto (k ≤ r) faça
6.        Gere um vizinho qualquer s' ∈ N(k)(s);
7.        s" ← BuscaLocal(s');
8.        se  f(s") < f(s)
9.           então  s ← s";
10.                 k ← 1;
11.          senão k ← k + 1;
12.       fim-se;
13.    fim-enquanto;
14. fim-enquanto;
15. Retorne s;
fim VNS;
```

Figura 4: Algoritmo VNS

Nesse algoritmo, parte-se de uma solução inicial qualquer e a cada iteração seleciona-se aleatoriamente um vizinho s' dentro da vizinhança $N^{(k)}(s)$ da solução s

corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, *s"*, for melhor que a solução *s* corrente, a busca continua de *s"* recomeçando da primeira estrutura de vizinhança $N^{(1)}(s)$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$. Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de CPU, o número máximo de iterações ou número máximo de iterações consecutivas entre dois melhoramentos. A solução s' é gerada aleatoriamente no passo 6 de forma a evitar ciclagem, situação que pode ocorrer se alguma regra determinística for usada.

6. Método de Descida em Vizinhança Variável (VND)

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND) (Mladenovic and Hansen (1997, 1999)) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada. O pseudocódigo desse algoritmo é apresentado pela Figura 5.

procedimento *VND*
1. Seja s_0 uma solução inicial e r o número de estruturas diferentes de vizinhança;
2. $s \leftarrow s_0$; {Solução corrente}
3. $k \leftarrow 1$; {Tipo de estrutura de vizinhança}
4. enquanto ($k \leq r$) faça
5. Encontre o melhor vizinho $s' \in N^{(k)}(s)$;
6. se $f(s') < f(s)$
7. então $s \leftarrow s'$; $k \leftarrow 1$;
8. senão $k \leftarrow k + 1$;
9. fim-se;
10. fim-enquanto;
11. Retorne s;

fim *VND;*

Figura 5: Algoritmo VND

Observa-se que para o caso do problema de alocação de salas, apenas as estruturas de vizinhança $N^{(1)}(s)$ e $N^{(2)}(s)$ foram consideradas, já que neste método de descida a terceira estrutura, $N^{(3)}(s)$, é redundante.

7. Algoritmos analisados

Foram analisadas duas versões do algoritmo VNS. A primeira, chamada VNS+VND, consiste em fazer a busca local (passo 7 do algoritmo VNS) usando o procedimento VND descrito na seção 6. Na segunda versão, chamada VNS+MV, a busca local consiste em determinar o melhor vizinho da solução corrente usando a estrutura corrente de vizinhança. Em ambas as versões, parte-se de uma solução inicial gerada conforme seção 4.

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 (Testereal), acrescido de dois outros problemas teste (Teste17 e Teste22). Esses dois últimos problemas foram gerados aleatoriamente, mantendo-se uma proporção de aulas semelhante ao existente no problema real. 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
Testereal	20	233	763
Teste22	22	281	938

Tabela 1: Características das instâncias consideradas

Uma bateria preliminar de testes indicou que os algoritmos VNS+VND e VNS+MV eram altamente dependentes de uma boa solução inicial. Este foi o motivo pelo qual resolveu-se partir de uma solução parcialmente gulosa e não completamente aleatória. Para cada problema teste foram realizadas 5 execuções dos algoritmos VNS+VND e VNS+MV, cada qual com uma semente diferente de números aleatórios. Utilizou-se como critério de parada um tempo de execução de 5000 segundos. A Tabela 2 mostra os valores médios das melhores soluções encontradas e a melhor solução encontrada por cada algoritmo nas 5 execuções. Observa-se que no segundo semestre letivo de 2001 o Instituto de Ciências Exatas e Biológicas da UFOP utilizou uma alocação de salas de valor 74320, encontrada manualmente (problema Testereal).

Instância	Algoritmo	Melhor Solução	Média
Teste17	VNS+VND	**7330**	**8378**
	VNS+MV	7750	8978
Testereal	VNS+VND	**10733**	**15280**
	VNS+MV	11210	16273
Teste22	VNS+VND	**31339**	**38035**
	VNS+MV	32189	39581

Tabela 2: Resultados computacionais

A Figura 6 ilustra o comportamento típico dos algoritmos VNS+VND, VNS+MV em uma execução do problema Testereal.

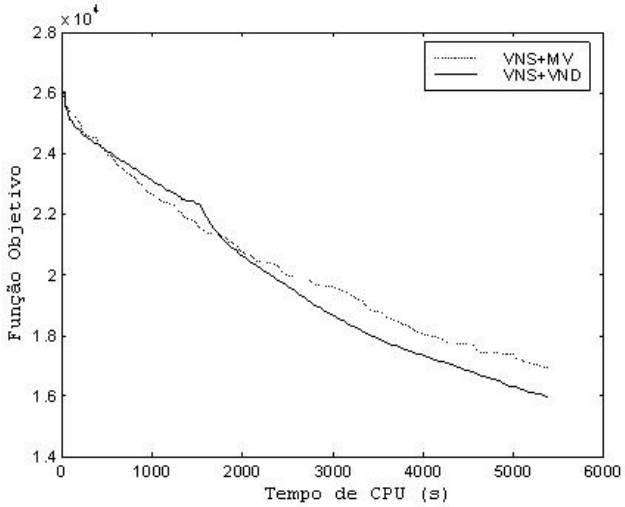

Figura 6: Comportamento típico dos algoritmos VNS+VND e VNS+MV

9. Conclusões e trabalhos futuros

Este trabalho contribui com o estudo do problema de alocação de salas. Apresentam-se três estruturas diferentes de vizinhança, as quais são usadas para testar duas versões do Método de Pesquisa em Vizinhança Variável. Esses dois algoritmos partem de soluções iniciais geradas por um procedimento construtivo parcialmente guloso.

O Método de Pesquisa em Vizinhança Variável é de fácil implementação e, basicamente, requer somente a definição de algumas estruturas de vizinhança, ao contrário

de outras metaheurísticas, tais como Busca Tabu e *Simulated Annealing*, que requerem a calibragem de diversos parâmetros.

O algoritmo VNS+VND mostrou ser mais eficaz que o algoritmo VNS+MV, superando-o em todos os testes realizados com relação à qualidade da solução final obtida.

Como trabalho futuro pretende-se analisar a influência do tamanho das vizinhanças analisadas a cada iteração do procedimento VND, o qual é responsável pela busca local dentro do algoritmo VNS, já que esse procedimento requer um tempo de processamento muito elevado para avaliar a vizinhança completa de uma solução.

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 UFOP 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.W. 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.

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", Kluver academic Publishers, Boston.

Mladenovic, N. and Hansen, P. (1997) "A Variable Neighborhood Search", *Computers and Operations Research*, 24: 1097-1100.

Mladenovic, N. and Hansen, P. (1999) "Variable Neighborhood Search: Methods and Recent Applications", *In* Third Metaheuristics International Conference, Angra dos Reis, Brazil, pp.275-280.

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.