**Uma abordagem de particionamento
de conjuntos através de
coloração em grafos**

Lana Mara Rodrigues dos Santos

Orientador : Prof. Dr. José Francisco Ferreira Ribeiro

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências - Área: "*Ciências de Computação e Matemática Computacional*".

São Carlos
1996

*à minha família
e aos amigos sempre presentes*

nem toda *hora*
é obra
nem toda obra
é *prima*
algumas são mães
outras *irmãs*
algumas
clima

Paulo Leminski

Agradecimentos

Ao meu orientador, José Francisco Ferreira Ribeiro, que me introduziu no estudo de coloração em grafos, pelo incentivo e palavras de encorajamento.

Ao prof. Luis Aurélio Raggi pelas sugestões dadas durante toda a elaboração deste trabalho.

Aos colegas do DMA-UFV pela concessão do afastamento.

Ao prof. José Geraldo Carvalho pela confiança e apoio.

Aos professores do ICMSC-USP, Antônio Castelo Filho, Antônio Cuminato, Neide Bertoldi Franco e Marcos Nereu Arenales pelos ensinamentos e disponibilidade com que sempre me receberam.

Ao prof. Olímpio Hiroshi Miyagaki por ter acreditado em algum potencial meu e ter sido um grande incentivador na minha graduação.

Ao prof. Alain Guénoche pela gentileza e presteza com que me esclareceu dúvidas e atendeu minhas solicitações.

À minha mãe, Celi, ao meu pai, Maurílio, e aos tios, Elisabeth, Maria Josepha, Salassier e Sediléa pelo carinho, ajuda, conselhos e orientação durante toda a minha vida.

Às amigas Giselle, Rosana e Simone pela amizade, pelas palavras de encorajamento e pelo grande apoio nos momentos mais difíceis.

Às secretárias, Beth e Laura, e aos funcionários da gráfica pela simpatia e eficiência com que sempre me atenderam.

À CAPES pelo suporte financeiro.

Finalmente, agradeço a todos que ajudaram na elaboração deste trabalho.

RESUMO

Esta dissertação apresenta o desenvolvimento de um método para o particionamento de conjuntos através de coloração em grafos. Uma aplicação deste método para a determinação de um horário escolar é igualmente apresentada. O programa computacional desenvolvido foi escrito em linguagem Pascal e testado utilizando-se um microcomputador IBM-PC compatível. Os resultados obtidos são de boa qualidade apesar do caráter não polinomial do problema estudado.

ABSTRACT

This thesis reports on the development of a method for cluster analysis using graph colouring. An application of this method on the school timetabling problem is presented. The corresponding program is written in Pascal and runs on a microcomputer IBM-PC compatible. The results obtained are high standard despite the non polynomial nature of the problem studied.

Introdução	1
Capítulo 1 Iniciação à Teoria dos Grafos	4
1 Introdução	4
2 Conceitos	4
2.1 Grafo	5
2.2 Subgrafo e Grafo Parcial	7
2.3 Grafo Valorado	7
2.4 Grafo Teto	8
2.5 Representação matricial de um grafo	8
2.6 Cadeia, Ciclo e Conexidade	9
2.7 Ciclo Hamiltoniano e Euleriano	10
2.8 Grafo Completo e Bipartido	10
2.9 Clique, Conjunto Independente de Vértices e “Matching”	11
2.10 Número e Índice Cromático	12
2.11 Árvore e Árvore geradora	12
2.12 Planaridade	14
Capítulo 2 Coloração em Grafos	15
1 Introdução	15
2 Conceitos	17
3 Aplicações	18
3.1 O problema dos exames escolares	18
3.2 O problema do pensionato	18
3.3 O projeto de células de manufatura	19
4 Alteração Estrutural de um Grafo	20
5 Homomorfismo de Grafos	21
5.1 Homomorfismo Quociente	23
6 Coloração Completa	24
7 Coloração com restrições	25
8 Algoritmos para a determinação do número cromático	26

8.1	Heurísticas seqüenciais	27
8.2	Algoritmo de Zikov	30
Capítulo 3	Algoritmo de Guénoche para uma p-coloração	32
1	Introdução	32
2	Matriz de Dissimilaridades e Partição	34
3	Algoritmo de Guénoche (1993)	34
3.1	Obtenção do grafo teto G_s p-colorível	36
3.2	Enumeração das p-colorações de G_s	38
3.3	Eliminação das partições que não são de diâmetro mínimo	40
4	Exemplo Ilustrativo	40
5	Comentários	44
Capítulo 4	Aplicação: planejamento de horário escolar	50
1	Introdução	50
2	Enunciado do Problema	51
3	Algoritmo	51
4	Cálculo da dissimilaridade entre as disciplinas	52
5	Matriz Potencial de Conflito	52
6	Medida de eficiência para auxílio à tomada de decisão	53
7	Resolução do Problema Teste	53
7.1	Dados do problema	53
7.2	Obtenção das matrizes de potencial de conflito e dissimilaridades	55
7.3	Soluções obtidas	55
7.4	Eficiência das soluções obtidas	56
7.5	Horário de Aulas	56
8	Comentários	58
Conclusão		59
Referências Bibliográficas		61

Introdução

Os grafos permitem a modelagem de um grande número de problemas práticos, tais como: otimização do fluxo em redes (rodoviárias, elétricas, telecomunicações, informáticas, etc.), seqüenciamento das atividades de projeto ou fabricação, projeto de células de manufatura, alocação de recursos, etc. Trata-se de uma ferramenta muito eficiente para a compreensão de uma situação concreta, para a sua modelagem e resolução. Numerosos algoritmos exatos ou aproximados estão disponíveis na área de teoria dos grafos para tratamento de problemas reais: algoritmos para a resolução de problemas de fluxo de custo mínimo, algoritmos (Bellman, Dijkstra, etc.) para a determinação de caminhos mínimos e/ou máximos em grafos com/sem ciclos e outros.

O desenvolvimento de algoritmos para a resolução de problemas modelados como grafos teve início em 1736 quando Euler propôs e resolveu o problema de determinar se era possível atravessar as sete pontes existentes no Parque Central de Königsberg (hoje a cidade de Kaliningrado) e voltar ao ponto de partida utilizando cada uma dessas pontes uma única vez. Um primeiro algoritmo, para resolver este problema, consiste em enumerar todos os itinerários possíveis e verificar se existe um que satisfaz a condição imposta. Euler considerou este algoritmo insatisfatório em razão do grande número de combinações possíveis e inaplicável no caso em que o número de pontes fosse muito elevado (Lewis e Papadimitriou, 1978). Na construção de um modelo em grafos para o problema, os vértices representam as margens dos rios e ilhas e as arestas as pontes. Euler propôs, então, um outro algoritmo de resolução: (1) verificar se o grafo correspondente não apresenta rupturas, ou seja, se é conexo; (2) verificar se o grau de cada nó do grafo é par. Um grafo que satisfaz (1) e (2) admite um ciclo que utiliza cada uma das arestas uma única vez (chamado “ciclo euleriano” em homenagem a Euler). O Parque de Königsberg não satisfaz (2) e, portanto, não possui um ciclo euleriano.

O matemático irlandês W. R. Hamilton formulou um outro problema clássico da teoria dos grafos, conhecido como o problema do “ciclo hamiltoniano”, e que possui, à primeira vista, bastante semelhança com o ciclo de Euler: dadas n cidades, trata-se de encontrar um caminho fechado que passe uma e só uma vez por todas essas cidades.

Estando as n cidades associadas aos vértices de um grafo, o problema do ciclo hamiltoniano consiste em determinar um ciclo que passe uma única vez por todos os vértices deste grafo. Apesar dos esforços de pesquisa realizados, não foi possível, até hoje, estabelecer uma condição necessária e suficiente para a determinação da existência de um ciclo hamiltoniano em um grafo. Os métodos disponíveis não são fundamentalmente melhores que o processamento de uma busca exaustiva sobre o conjunto de soluções possíveis do problema.

Ao “Problema do Carteiro Chinês” (Gondran e Minoux, 1990) está associada a busca de um ciclo euleriano em um grafo e ao “Problema do Caixeiro Viajante” (Gondran e Minoux, 1990) a busca de um ciclo hamiltoniano. Tais problemas são clássicos em otimização e possuem inúmeras aplicações: roteamento, sequenciamento, distribuição, etc.

Na área de teoria dos grafos, interessa-nos, o estudo do problema e dos algoritmos disponíveis para efetuar a coloração dos vértices de um grafo em um número pré-determinado, p , de cores. Não existe um algoritmo polinomial ou eficiente para a resolução do problema da coloração em grafos para toda e qualquer entrada (Aho et al, 1974). Outros problemas para os quais não é conhecido um algoritmo polinomial para a sua resolução são: o problema da cláusula booleana, os problemas de programação inteira, os problemas de “matching”, clique, “stable”, entre outros. Estes problemas pertencem a classes dos problemas NP-completos (Garey e Johnson, 1979), e têm sido objeto de numerosos estudos, dada a grande aplicação prática dos mesmos.

O estudo realizado nesta dissertação teve por objetivo o desenvolvimento e implementação de um programa computacional para efetuar o particionamento de conjuntos, em um número fixo de classes, através de coloração em grafos e, como aplicação, resolver o problema de programação de um horário escolar (“School Timetabling”) em uma universidade ou planejar a apresentação de artigos em um congresso (De Werra, 1985). Trata-se de um problema relevante uma vez que não é possível garantir um bom uso dos recursos disponíveis quando o número de alunos ou de disciplinas ou de artigos a apresentar é elevado.

A apresentação do trabalho que realizamos está organizada da seguinte maneira:

No capítulo 1 é feita uma introdução à teoria dos grafos, onde são apresentados os conceitos fundamentais à compreensão dos capítulos seguintes. Entre estes conceitos, destacam-se os seguintes: grafo, grafo valorado, grafo conexo, grafo teto, grau, ciclo, clique, "matching", "stable", árvore, número e índice cromático.

O capítulo 2 é dedicado ao estudo da coloração em grafos. Após a introdução de conceitos importantes em coloração tais como número cromático, homomorfismo e p-coloração, são apresentados alguns algoritmos de coloração de vértices, como as heurísticas sequenciais de Brélaz (1979) e Leighton (1979) e o algoritmo exato de religações-condensações de Zikov (1952).

No capítulo 3 é feito um estudo e apresentação do algoritmo de Guénoche (1989, 1993) para determinação de uma p-coloração sobre um grafo construído a partir de uma matriz de dissimilaridades entre objetos. O algoritmo é constituído de 3 etapas e faz uso de procedimentos aproximados para determinação de um teto e de uma coloração para o subgrafo obtido. Um exemplo ilustrativo do algoritmo estudado é apresentado e algumas propostas de variantes para o mesmo são comentadas e discutidas. Uma das variantes do algoritmo de Guénoche (1993) implementadas consistiu em incorporar à 1ª etapa do mesmo uma fase em duas versões para a otimização do valor do teto, necessário à construção e coloração do grafo. O algoritmo de Guénoche (1989, 1993) e suas variantes foi testado para 20.000 exemplos gerados aleatoriamente e forneceu bons resultados.

A implementação de um programa computacional desenvolvido para resolução do problema de determinação de um horário escolar é apresentada no capítulo 4. Nesta implementação é proposta uma maneira de se calcular a matriz de dissimilaridades, um índice de eficiência para auxiliar no processo de tomada de decisão e utiliza-se um algoritmo baseado no algoritmo de Guénoche (1989, 1993) para obter todas as p-colorações do grafo associado ao problema. Um exemplo, gerado aleatoriamente, para determinar a disposição de 12 disciplinas em quatro horários para 48 alunos de 3 cursos distintos é resolvido com o auxílio do programa desenvolvido. Este programa foi escrito em linguagem Pascal e testado utilizando-se um microcomputador IBM-PC compatível.

Na conclusão é feita uma reflexão sobre o trabalho realizado e são apresentadas as perspectivas de continuidade para o mesmo.

1

Iniciação à Teoria dos Grafos

1 - Introdução

Este capítulo é dedicado à apresentação dos conceitos fundamentais da Teoria dos Grafos, imprescindíveis à compreensão do trabalho desenvolvido nesta dissertação. A apresentação é breve e para um estudo mais completo do assunto muitas referências bibliográficas estão disponíveis, entre as quais podemos citar: Berge (1976), Netto (1979), Sakarovitch (1984), Gondran e Minoux (1990) e Szwarcfiter (1993).

2 - Conceitos

Os seguintes conceitos são estudados nos itens abaixo: grafo, grau dos vértices de um grafo, grafo orientado, grafo valorado (ou com pesos nas arestas), grafo teto, laço, ciclo, cadeia, conexidade, ciclos hamiltoniano e euleriano, grafo completo e bipartido, clique, “stable”, “matching”, número e índice cromático, árvore, árvore geradora e planaridade. Diversos exemplos ilustram os conceitos acima relacionados.

2.1 - Grafo

- Um **grafo** $G(S,A)$ é um conjunto finito não-vazio S constituído de elementos denominados **vértices** ou **nós** e um conjunto A que pode ser de pares não-ordenados ou ordenados de elementos de S , denominados **arestas** ou **arcos** de G , respectivamente.
- Se A é constituído de pares ordenados, diz-se que G é um grafo **orientado**; caso contrário G será **não orientado**.
- O termo grafo será utilizado tanto para grafos orientados quanto para grafos não orientados. Boa parte da nomenclatura e dos conceitos são análogos tanto em um caso como no outro.
- Cada aresta $a \in A$ é descrita pelo par de vértices $a = (i, j)$ que a define. Neste caso, os vértices i e j são os extremos da aresta a e denominados adjacentes. A aresta a é dita incidente a ambos i e j . Duas arestas que possuem um extremo comum são chamadas adjacentes.
- A notação $|S|$ e $|A|$ indica o número de elementos de S e A , respectivamente.

Um grafo pode ser visualizado através de sua representação geométrica, na qual os vértices correspondem a pontos distintos do plano e a cada aresta (i, j) é associada uma linha unindo os pontos correspondentes a i e j . O termo grafo é igualmente utilizado para designar sua representação geométrica.

A Figura 1.1, abaixo, ilustra o exemplo de um grafo $G(S,A)$ com 6 vértices e 11 arestas.

$$S = \{1, 2, 3, 4, 5, 6\}, |S| = 6$$

$$A = \{(1, 2), (1, 3), (1, 5), (1, 6), (2, 3), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5), (5, 6)\}, |A| = 11$$

Figura 1.1

Na visualização geométrica de um grafo orientado, um arco é representado por uma seta cujo sentido corresponde à orientação do par ordenado.

- Em um grafo orientado, um arco do tipo $a = (i, i)$ é admissível e chamado **laço**. Não é possível, porém, definir um sentido desse arco. Na Figura 1.2, o arco $a = (2, 2)$ é um laço.

Figura 1.2

Um grafo não orientado pode ser associado a um grafo orientado no qual cada aresta é substituída por dois arcos de sentidos opostos e no qual não existem laços.

- Pode-se substituir na definição de grafo, o conjunto de arestas, A , por um multiconjunto. Com isto, é possível ter mais de uma aresta entre o mesmo par de vértices. Estas arestas são denominadas de arestas paralelas e a estrutura (S, A) é dita um **multigrafo**.

A Figura 1.3 mostra um multigrafo com 3 arestas entre os nós 2 e 4.

Figura 1.3

- **Grau** de um vértice i de um grafo G é o número de arestas incidentes a i . Um vértice que possui grau zero é chamado de isolado. Na Figura 1.3, o nó 1 tem grau 2 e o nó 2 tem grau 5.

2.2 - Subgrafo e Grafo Parcial

Se G é um grafo e X um conjunto de arestas ou vértices, $G-X$ e $G+X$ denotam o grafo obtido de G pela exclusão e inclusão de X , respectivamente. Excluir um vértice implica em remover de G o vértice em questão e as arestas a ele incidentes.

- Um **subgrafo** $G_1 (S_1, A_1)$ de um grafo $G(S, A)$ é um grafo tal que $S_1 \subseteq S$ e $A_1 \subseteq A$. Se, além disso, G_1 possuir toda aresta (i, j) de G tal que i e j estejam em S_1 , então G_1 é o subgrafo induzido de G pelo subconjunto de vértices S_1 .
- Um **grafo parcial** $G_1 (S, A_1)$ de um grafo $G(S, A)$ é um grafo tal que $A_1 \subseteq A$.

2.3 - Grafo Valorado

Diz-se que um grafo é **valorado** se existem uma ou mais funções relacionando S e/ou A com um conjunto de números :

$$f: S \rightarrow P$$

$$i \rightarrow p_i$$

$$g: A \rightarrow Q$$

$$(i, j) \rightarrow q_{ij}$$

- **Comprimento** de uma aresta $a = (i, j)$ em um grafo valorado é a imagem de uma função peso λ associado à aresta a .

Na Figura 1.4, a aresta $a = (5, 6)$ tem peso $\lambda(a) = 5$

Figura 1.4

2.4 - Grafo Teto

Dado um índice $s \geq 0$, define-se como **grafo teto**, $G_s(S, A_s)$, o subgrafo parcial de $G(S, A)$ tal que $\exists (x_i, x_j) \in A_s \Leftrightarrow \lambda(x_i, x_j) \geq s$.

A Figura 1.5 ilustra o grafo teto G_6 , subgrafo parcial do grafo da Figura 1.4

Figura 1.5

2.5 - Representação matricial de um grafo

Pode-se associar a um grafo não-orientado uma **matriz de Valores ou Pesos** $P=[a_{ij}]$ definida da seguinte maneira:

$$a_{ij} = \begin{cases} \lambda(i, j) & \Leftrightarrow \exists(i, j) \\ 0 & \text{caso contrario} \end{cases}$$

Obs.: A matriz de valores de um grafo não-orientado é sempre simétrica pois se $a_{ij} \neq 0$, $a_{ij} = \lambda(i, j) = \lambda(j, i) = a_{ji}$

A Figura 1.6 nos fornece a matriz P associada ao grafo da Figura 1.5.

$$P = \begin{bmatrix} 0 & & & & & & \\ 0 & 0 & & & & & \\ 0 & 0 & 0 & & & & \\ 0 & 7 & 6 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & & \\ 6 & 0 & 0 & 11 & 0 & 0 & \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 1.6

Existem outras representações de um grafo G por meio de matriz (Netto, 1979).

2.6 - Cadeia, Ciclo e Conexidade

- **Cadeia** é uma seqüência de arestas que estão ligadas umas às outras por uma extremidade qualquer. Uma cadeia é **elementar** se não utiliza mais de uma vez um mesmo vértice e **simples** se não utiliza mais de uma vez a mesma aresta. Logo, toda cadeia elementar é simples, porém nem toda cadeia simples é elementar. Na Figura 1.7, a seqüência 1,2,3,6 é uma cadeia elementar enquanto 1,2,3,6,1,5 é uma cadeia simples, mas não elementar.
- **Comprimento** de uma cadeia é o número de arestas que a compõe (se o grafo for valorado, o comprimento da cadeia é dado pela soma dos pesos das arestas que a compõe).
- **Distância** $d(i, j)$ entre dois vértices i, j de um grafo é o comprimento da menor cadeia entre i e j . Na Figura 1.7, por exemplo, $d(1,4) = 2$
- **Ciclo** é uma cadeia onde o vértice inicial é idêntico ao vértice final. Define-se, da mesma maneira ciclo simples e elementar. Na Figura 1.7, a seqüência 1,2,3,6,1 é um ciclo elementar enquanto 5,1,6,2,3,6,5 é uma ciclo simples.
- Um grafo $G(S,A)$ é dito **conexo** quando existe uma cadeia unindo cada par de seus vértices. Caso contrário G é desconexo. A representação de um grafo desconexo é necessariamente descontínua. O grafo da Figura 1.7 é conexo enquanto o grafo da Figura 1.5 é desconexo.

Obs.: Seja S um conjunto e $S' \subseteq S$. Diz-se que S' é maximal em relação a uma certa propriedade P quando S' satisfaz P e não existe $S'' \supset S'$ ($S'' \subseteq S$) tal que S'' satisfaça P . De maneira análoga, define-se conjunto minimal. Observe que S' não precisa ser o maior subconjunto de S que satisfaça P .

- **Componentes conexos** de um grafo G são os subgrafos maximais de G conexos.

Figura 1.7

2.7 - Ciclo Hamiltoniano e Euleriano

- Uma cadeia ou ciclo é **euleriano** quando contém cada aresta do grafo somente uma vez.
- Uma cadeia é **hamiltoniana** se for elementar e contenha todo vértice do grafo. Da mesma forma define-se ciclo **hamiltoniano**.
- Se $G(S,A)$ é um grafo que possui um ciclo hamiltoniano ou euleriano então G é denominado Hamiltoniano ou Euleriano, respectivamente.

Os teoremas abaixo são importantes para identificação de grafos Hamiltonianos ou Eulerianos, respectivamente:

- *Teorema: Seja $G(S,A)$ um grafo com pelo menos 3 vértices tal que grau $(i) \geq |S|/2$ para todo vértice $i \in S$. Então G é Hamiltoniano.*
- *Teorema de Euler: Um grafo é Euleriano se e somente se ele for conexo e todos os seus vértices tiverem grau par.*

O teorema de Euler fornece a condição necessária e suficiente para a existência de um ciclo euleriano em um grafo. O grafo da Figura 1.7 não é Euleriano pois os vértices 2 e 3 têm grau ímpar.

Até hoje, não é conhecida uma condição necessária e suficiente para a existência de um ciclo hamiltoniano. O grafo da Figura 1.7 é Hamiltoniano pois contém um ciclo hamiltoniano dado por 1,2,3,4,5,6,1 embora grau (4) < 3.

2.8 - Grafo Completo e Bipartido

- Um grafo é **completo** quando existe uma aresta entre cada par de seus vértices. Utiliza-se a notação K_n para designar um grafo completo com n vértices.
- Um grafo é **bipartido** quando seu conjunto de vértices S puder ser particionado em dois conjuntos S_1 e S_2 tal que toda aresta de G une um vértice de S_1 a outro de S_2 . Denota-se o grafo bipartido G por $G(S_1 \cup S_2, A)$.
- Um grafo **bipartido completo** possui uma aresta para cada par de vértices s_1, s_2 sendo $s_1 \in S_1$ e $s_2 \in S_2$. Se $n_1 = |S_1|$ e $n_2 = |S_2|$, denotamos G por K_{n_1, n_2} .

Um resultado importante para identificação de grafos bipartidos é dado pelo seguinte teorema: *Um grafo é bipartido se e somente se todo ciclo de G possuir número par de vértices.*

Na Figura 1.8, $G(S,A)$ é o grafo bipartido completo $K_{2,3}$ sendo $S = S_1 \cup S_2$ com $S_1 = \{1,2\}$ e $S_2 = \{3,4,5\}$.

Figura 1.8

Os grafos bipartidos são importantes no estudo de problemas relacionados com pares de conjuntos de propriedades diversas. Exemplo: problemas de transporte em redes onde S_1 é o conjunto de pontos de origem e S_2 é o conjunto de pontos de destino.

2.9 - Clique, Conjunto Independente de Vértices e “Matching”

- Uma **clique** de um grafo G é um subgrafo de G que seja completo.
- **Conjunto independente de vértices** (SCIE) ou “stable” é um subgrafo induzido de G que seja totalmente desconexo.
- O tamanho de uma clique ou de um conjunto independente de vértices é igual ao número de elementos de seu conjunto de vértices.
- Um “**matching**” de um grafo G é um subconjunto de arestas de G tal que duas arestas quaisquer pertencentes a este subconjunto não sejam adjacentes.

Na Figura 1.9, $\{2,4,5\}$, $\{1,3\}$ e $\{3,6\}$ são SCIE maximais, $\{1,2,6\}$ é uma clique e $\{(1,2),(5,6),(3,4)\}$ é um “matching” de G .

Figura 1.9

- O número de vértices de um SCIE de maior cardinalidade de um grafo G é chamado **número de estabilidade interna**, $\alpha(G)$. No pior caso, o problema de determinação dos SCIE maximais é de complexidade exponencial (Netto, 1979).
- **Complemento de um grafo** $G(S,A)$ é o grafo $\overline{G}(S,A')$ tal que $(i,j) \in A \Leftrightarrow (i,j) \notin A'$.

Como a cada SCIE de um grafo G corresponde uma clique do grafo complementar $\overline{G}$, o problema determinação dos SCIE maximais pode ser apresentado como o da determinação de cliques maximais no grafo $\overline{G}$.

2.10 - Número e Índice Cromático

- **Número cromático**, $\lambda(G)$, de um grafo G é o menor número de cores necessárias para colorir os vértices de G de modo que dois vértices adjacentes não recebam a mesma cor.
- **Índice cromático**, $\lambda_1(G)$, de um grafo G é o menor número de cores necessárias para colorir as arestas de G tal que duas arestas adjacentes não possuam a mesma cor.

Na Figura 1.10, o grafo G possui $\lambda(G) = 3$ e $\lambda_1(G) = 3$

Figura 1.10

2.11 - Árvore e Árvore Geradora

Se um grafo não possui ciclos é dito acíclico. Uma **árvore** $Ar(S,A)$ é um grafo acíclico e conexo.

- **floresta**: conjunto de árvores.
- **folha**: se i é um vértice de Ar tal que $\text{grau}(i) = 1$.

A Figura 1.11 ilustra uma árvore com folhas 2, 4 e 5.

Figura 1.11

Alguns teoremas que podem ser úteis para caracterização de árvores são apresentados abaixo:

Teorema: Toda árvore com n vértices possui $n-1$ arestas

Teorema: Um grafo G é uma árvore se e somente se existir uma única cadeia entre cada par de vértices de G .

Teorema: Seja $G(S,A)$ um grafo. As seguintes afirmativas são equivalentes:

- (i) G é uma árvore
- (ii) G é conexo e $|A|$ é mínimo
- (iii) G é acíclico e $|A|=|S|-1$
- (iv) G é conexo e $|A|=|S|-1$
- (v) G é acíclico e $\forall i, j \in S$, a inserção de (i, j) produz um grafo contendo exatamente um ciclo.

- Um **subgrafo gerador** de um grafo $G(S,A)$ é um subgrafo parcial de G . Se este subgrafo for uma árvore, ele recebe o nome de **árvore geradora**.

A existência de uma árvore geradora em um grafo conexo fica garantida pelo teorema: *Todo grafo conexo G possui uma árvore geradora*

As aplicações de se determinar a árvore geradora mínima (ou máxima) são numerosas: otimização em redes, classificação hierárquica, etc.

O problema da determinação de uma árvore geradora de peso mínimo pode ser enunciado da seguinte forma (Gondran e Minoux, 1990):

- Considere um grafo $G(S,A)$ valorado. Para cada aresta $a \in A$ associa-se o peso da aresta, $\lambda(a)$. Seja $G'(S,A')$ um grafo parcial de G . O peso de G' é um parâmetro calculado como se segue:

$$\lambda(G') = \sum_{a \in G'} \lambda(a)$$

- Supõe-se G conexo (se não o for, resolve-se separadamente para cada componente conexa). Procura-se, então, uma árvore Ar^* de G tal que :

$$\lambda(Ar^*) = \min \{ \lambda(Ar), \forall Ar \subset G \}$$

2.12 - Planaridade

Seja G um grafo e R uma representação geométrica de G em um plano.

- A representação R é **plana** quando não houver cruzamento de arestas em R , a não ser nos vértices.
- Um grafo é **planar** quando admitir alguma representação plana.

Uma caracterização de planaridade é dada pelo teorema abaixo:

Teorema (Kuratowski): Um grafo é planar se e somente se ele não contém como subgrafo parcial um grafo $K_{3,3}$ ou K_5 .

A Figura 1.12(a), fornece uma representação não plana do grafo completo K_4 .

A Figura 1.12(b) mostra uma representação plana desse grafo; portanto K_4 é planar.

Figura 1.12

2

Coloração em Grafos

1 - Introdução

O estudo da coloração em grafos teve início com o chamado “problema das quatro cores”. Em 1852, Francis Guthrie formulou a conjectura de que qualquer mapa poderia ser colorido com quatro cores de modo que países fronteiros não tivessem a mesma cor.

Se M é uma região do plano, um mapa é um particionamento de M em um número finito de regiões, as quais são delimitadas por linhas. Duas regiões são adjacentes quando possuírem uma linha em comum. Uma coloração de M é uma atribuição de alguma cor a cada região de M , de modo que regiões adjacentes possuam cores diferentes. Pode-se definir um grafo G associando-se um vértice para cada região de M e uma aresta a um par de vértices adjacentes quando as respectivas regiões o forem (Figura 2.1). Então, colorir o mapa M é equivalente a colorir G . Logicamente G tem de ser planar. O “problema das quatro cores” consiste, pois, em provar que todo grafo planar é 4-colorível.

Figura 2.1

Ao longo desses 125 anos, ninguém foi capaz de exibir um mapa em que cinco cores fossem necessárias e muito menos exibir uma prova direta não enumerativa que quatro cores são suficientes. K. Appel e W. Haken (1976) realizaram esta demonstração através de uma técnica por computador. Eles utilizaram resultados da teoria dos grafos, entre os quais pode-se citar o trabalho de Heawood (1890), para provar que o número geral pode ser reduzido a um número finito de casos particulares, tais que todo grafo planar pode ser colorido com quatro cores. O resultado ficou conhecido como “teorema das quatro cores”.

Neste capítulo é apresentado um estudo teórico do problema de coloração em grafos, três exemplos de aplicações e alguns algoritmos para resolução de problemas concretos envolvendo coloração. Para a redação deste capítulo, foram consultadas várias referências bibliográficas clássicas em Teoria dos Grafos, entre as quais podemos citar: Berge (1976), Netto (1979), Christofides et al (1979), Sakarovitch (1984), Gondran e Minoux (1990), Gouvêa (1993), Szwarcfiter (1993) e Campello e Maculan (1994). A estrutura de base utilizada para a apresentação dos principais conceitos e algoritmos é aquela utilizada por Campello e Maculan (1994), por ser a referência bibliográfica mais recente e realizar um tratamento do assunto útil aos nossos propósitos.

2 - Conceitos

- Uma **coloração** de um grafo, $G(S,A)$, é uma atribuição de alguma cor de um conjunto de cores, $C\{c_i\}$, para cada vértice de S , de tal modo que dois vértices adjacentes de $G(S,A)$ não recebam a mesma cor. Isto é, uma coloração de $G(S,A)$ é uma função $f : S \rightarrow C$ tal que se $(i, j) \in A \Rightarrow f(i) \neq f(j)$
- Um **k-coloração** de $G(S,A)$ é uma coloração de $G(S,A)$ que utiliza k cores o que equivale a uma partição de S em k SCIE, maximais ou não. Dizemos, então, que G é **k-colorível**. A Figura 2.2 ilustra um grafo com três k-colorações possíveis.

Figura 2.2

- **Número cromático**, $\lambda(G)$, de um grafo G é o menor número de cores para o qual existe uma k -coloração de G . Na Figura 2.2, $\lambda(G) = 3$.

Não é conhecido algoritmo eficiente para determinar o número cromático de um grafo G . Entretanto, pode-se obter alguns resultados imediatos da definição, tais como:

- (i) Um grafo G é bicromático se e somente se for bipartido.
- (ii) Um grafo completo K_t é t -cromático.
- (iii) Um grafo G com uma ou mais arestas é pelo menos 2-cromático.

Os conceitos de coloração, clique e conjunto independente de vértices (SCIE) estão relacionados:

- Como são necessárias k cores para colorir os k vértices de uma clique de tamanho k , isto implica que $\lambda(G)$ é maior ou igual ao tamanho da maior clique de G .
- Sejam $S_1, S_2, \dots, S_k$ subconjuntos disjuntos de S , induzidos pela k -coloração. Então, $\cup S_i = S$, $i = 1, \dots, k$, e cada S_i é um conjunto independente de vértices (SCIE). Com isto, determinar $\lambda(G)$ pode ser formulado em termos de particionar S em um número mínimo de SCIE (maximais ou não).

- **Número de estabilidade interna**, $\alpha(G)$, de um grafo G é o número de vértices de um SCIE de maior cardinalidade do grafo. Na Figura 2.2, $\alpha(G) = 2$.

Seja $G(S,A)$ um grafo não orientado com $|S| = n$ e $|A| = m$. Alguns resultados importantes unindo $\lambda(G)$ a $\alpha(G)$, m , n e ao grau máximo de G são dados pelos teoremas:

- (i) $\lambda(G) + \alpha(G) \leq n+1$
- (ii) Se $\overline{G}$ é o grafo complementar de G então $\lambda(G) + \lambda(\overline{G}) \leq n+1$
- (iii) $\lambda(G) \geq n^2 / (n^2 - 2m)$
- (iv) Se o grau máximo de G é h então $\lambda(G) \leq h+1$
- (v) Um grafo é bicromático se e somente se não contém ciclos de comprimento ímpar

Obs.: A demonstração desses teoremas podem ser encontradas em Netto (1979).

3 - Aplicações

Para ilustrar a aplicação dos estudos em coloração em grafos, apresentamos três problemas. Outras aplicações são discutidas em Leighton (1979), Opsut e Roberts (1981) e De Werra (1985).

3.1 - O problema dos exames escolares (Netto, 1979)

Considera-se um conjunto X de provas a serem realizadas por um conjunto Y de estudantes onde cada prova $x \in X$ deve ser realizada por um conjunto $S(x)$ de estudantes. Cada prova deve ser aplicada uma única vez, o que determina o comparecimento de todos os alunos que devem realizá-la. Se cada aluno comparece no máximo a uma prova por dia, o problema consiste em se determinar o menor número de dias exigidos para a programação das provas. Pode-se resolver este problema construindo um grafo $G(X,U)$ no qual cada aresta $(x, y) \in U$ se $S(x) \cap S(y) \neq \emptyset$, i.e., as provas x e y não podem ser realizadas no mesmo momento pois existem alunos que devem estar presentes a ambas. Cada coloração dos vértices corresponde a uma alocação possível de provas. As provas correspondentes a vértices de uma mesma cor poderão ser previstas para o mesmo dia. O número de dias necessários será, então, $\lambda(G)$.

3.2 - O problema do pensionato (Berge, 1976)

Quinze moças que estudam em um pensionato saem para passear todos os dias dispostas em 3 filas. Pergunta-se: é possível realizar sete passeios consecutivos sem que duas moças estejam lado a lado mais de uma vez? Este é um problema clássico da teoria dos grafos e foi objeto de numerosos estudos. Trata-se de um problema próximo

de outro igualmente célebre, o problema de Steiner, onde dispõe-se de 15 moças para formar 35 conjuntos distintos de 3 moças de modo que duas moças não façam parte do mesmo conjunto mais de uma vez.

A resolução do problema de Steiner pode ser feita através de um grafo G cujos nós são os $C_{15,3} = 455$ conjuntos possíveis, dois conjuntos estando unidos por uma aresta se entre eles existir duas moças em comum. Desta forma, procura-se o “stable” S de cardinal máximo. Têm-se $|S| \leq 35$, uma vez que em S a mesma moça aparece no máximo em 7 conjuntos distintos, o que representa um total de $15 \times 7 \times 1/3 = 35$ conjuntos no máximo. Para ver que uma solução do problema de Steiner, relaciona-se com o problema do pensionato, constrói-se um grafo G_1 cujos nós são os 35 conjuntos relacionados, dois conjuntos estando unidos por uma aresta se entre eles existir uma moça em comum. Se o número cromático $\lambda(G_1)$ for igual a 7, o problema está resolvido; se $\lambda(G_1) > 7$, será necessário relacionar outros conjuntos de moças.

3.3 - O projeto de células de manufatura (Ferreira Ribeiro e Ribeiro, 1993; Ferreira Ribeiro e Alves, 1994)

O projeto de células de manufatura, para implantação em uma indústria, consiste em encontrar uma partição dos produtos a fabricar em famílias e do parque de máquinas disponíveis em grupos ou células. A cada família de produtos corresponde um só grupo ou célula de máquinas e vice-versa. Espera-se uma maior facilidade para administrar a fábrica proveniente da decomposição do sistema global de produção em sub-sistemas de menor dimensão.

Para realizar esta tarefa, estabelece-se uma distância d entre quaisquer 2 pares de produtos a fabricar com base, por exemplo, nos roteiros de fabricação dos mesmos (Ferreira Ribeiro e Pradin, 1993). Constrói-se, então, um grafo $G(N,A)$, onde N é o conjunto de produtos, A é o conjunto de arestas e d o comprimento de cada aresta inserida no grafo. Existe uma aresta unindo 2 nós i e j se a distância d_{ij} for maior ou igual ao valor de uma distância crítica fixada “a priori”. Como 2 nós adjacentes não podem receber a mesma cor, a coloração do grafo G acima descrito não colocará em uma mesma célula de manufatura produtos cujas distâncias entre si sejam maiores ou iguais a um valor dado de distância mínima. Os produtos que receberem a mesma cor estarão reunidos em uma mesma família de produtos e serão fabricados por uma mesma célula de máquinas.

4 - Alteração Estrutural de um Grafo

A técnica de alteração estrutural de um grafo $G(S,A)$ consiste em aplicar uma operação sobre este grafo de modo a transformá-lo em outro grafo $G'(S',A')$. As transformações relacionadas abaixo são as mais comuns:

(1) *Eliminação de vértices ou arestas (E)*: Escolhe-se um vértice v ou aresta a de G , que é retirado de G . Obtêm-se $G'(S-\{v\},A')$ ou $G'(S,A-\{a\})$, respectivamente.

(2) *Adição de vértices ou arestas (religação) (R)*: Escolhe-se um vértice v ou aresta a de G , que é acrescentado a G . Obtêm-se os grafos $G'(S+\{v\},A')$ ou $G'(S,A+\{a\})$, respectivamente.

(3) *Contração (C)*: Seja $G(S,A)$ um grafo com pelo menos dois vértices não adjacentes x e y . Uma contração dos vértices $x, y \in S$ corresponderá à assimilação de x e y em um único vértice unido a todos os $s \in S, s \neq x, y$ que sejam adjacentes a um dos dois.

A Figura 2.3, abaixo, ilustra as 3 técnicas E, R e C :

Figura 2.3

A determinação do número cromático de um grafo G pode ser feita através de um esquema iterativo de alterações estruturais que visa a determinação de cliques. O objetivo final é a obtenção da clique de cardinalidade mínima.

Algoritmos que se utilizam de operações de contração de vértices ou adição de arestas são do tipo alteração estrutural. Podem ser exatos ou heurísticos.

Obs.: As operações (1) e (2) não preservam no grafo resultante as adjacências de G .

5 - Homomorfismo de Grafos

Seja $G(S,A)$ um grafo conexo com pelo menos dois vértices não adjacentes, x e y e um grafo $G'(S',A')$ tal que:

$$S' = \{S - \{x, y\} \cup \{xy\}\}$$

$$A' = \{A - \{(i, j)\} / \{i, j\} \cap \{x, y\} \neq \emptyset\} \cup \{(s, xy) / s \in \Gamma(x) \cap \Gamma(y)\}$$

sendo $\Gamma(x)$ = conjunto dos vértices adjacentes a x

- G' será a **imagem homomorfa** de G obtida por contração dos vértices x e y . Esta operação pode ser caracterizada por uma função g_{xy} definida em G , da forma:

$$g_{xy}(u) = \begin{cases} u & \text{se } u \notin \{x, y\} \\ xy & \text{caso contrário} \end{cases}$$

tal que $\forall (g_{xy}(u), g_{xy}(v)) \in A' \exists (u, v) \in A$

- Um **homomorfismo elementar** foi definido por Harary (1969) como a contração de dois vértices não adjacentes. Com isto, g_{xy} é um homomorfismo sobrejetor elementar de G sobre G' .
- Uma composição finita dessas funções, g_{xy} , da forma $h = g_1 \circ g_2 \circ \dots \circ g_k$ é um **homomorfismo** de G sobre $h(G)$.

A Figura 2.4 ilustra uma sequência de imagens geradas por um homomorfismo, de comprimento cinco, aplicado ao grafo G .

Figura 2.4

Um teorema que relaciona $\lambda(G)$ e $\lambda(h(G))$, ie., o número cromático dos grafos G e $h(G)$ é descrito a seguir:

Teorema (Hajós, 1961): *Se h é um homomorfismo elementar em G então $\lambda(G) \leq \lambda(h(G)) \leq \lambda(G) + 1$*

Demonstração: *sejam x, y dois vértices não adjacentes em G , quando a $\lambda(G)$ -coloração adota cor idêntica para ambos (Figura 2.5), uma coloração ótima para $h(G)$ pode ser obtida mantendo-se as atribuições de cores dos demais vértices e atribuindo ao vértice xy , a cor adotada para colorir x e y . Com isto, $\lambda(G) = \lambda(h(G))$.*

Figura 2.5

Se a $\lambda(G)$ -coloração adota cores distintas para x e y , duas situações podem se aplicar. Se uma das cores atribuídas a x e y não foi utilizada para colorir nenhum dos vértices adjacentes a xy , uma coloração ótima para $h(G)$ é obtida atribuindo-se esta cor a xy e mantendo as demais; neste caso $\lambda(G) = \lambda(h(G))$ (Figura 2.6(a)) Caso contrário, a coloração ótima para $h(G)$ é obtida utilizando uma nova cor para o vértice xy e mantendo-se as demais; portanto $\lambda(h(G)) = \lambda(G) + 1$ (Figura 2.6(b)).

Figura 2.6

Pode-se concluir, então, que sendo h_p um homomorfismo de comprimento p em G , $\lambda(G) \leq \lambda(h_p(G)) \leq \lambda(G) + p$.

5.1 - Homomorfismo Quociente

Dado um grafo $G(S,A)$, existem propriedades importantes que induzem partições no conjuntos S e/ou A e equivalências naturais entre seus vértices e/ou arestas. O Conceito de “Grafo Quociente” foi introduzido por Golumbic (1977) em um estudo acerca da relação de vértices com mesmos adjacentes: para dois vértices x e y , uma relação de equivalência $x \sim y$ é definida em $S \Leftrightarrow \Gamma(x) = \Gamma(y)$. Logo, vértices equivalentes não são adjacentes. O grafo quociente $\tilde{G}(\tilde{S}, \tilde{A})$ fica definido pela contração dos vértices equivalentes, sendo $\tilde{A} = \{(\tilde{x}, \tilde{y}) / (x, y) \in A\}$. A Figura 2.7 ilustra este conceito.

Figura 2.7

Abreu (1991) generalizou o conceito de grafo quociente para qualquer relação de equivalência sobre S da seguinte forma: considere $\tilde{S}$ um conjunto formado por classes de equivalência, $\tilde{x} = \{y \in S / y \sim x\}$. $G(S,A)$ é o quociente de $\tilde{G}(\tilde{S}, \tilde{A})$ com respeito à equivalência $\sim$ sobre S .

- Quando $\tilde{G}$ é obtido de G por partição em SCIE, a função $f: G \rightarrow \tilde{G}$ é um homomorfismo de G sobre $\tilde{G}$ chamado **homomorfismo Quociente** de G sobre $\tilde{G}$.

Logo, todo homomorfismo Quociente é resultado de uma sequência de condensações de homomorfismos elementares e por isso determina sempre uma coloração para G .

Um quociente de Golumbic $\tilde{G}$ é a imagem homomorfa de G , por um homomorfismo quociente com respeito à relação de adjacentes iguais de acordo com o teorema :

Teorema: Quocientes de Golumbic são determinados por homomorfismos quocientes.

6 - Coloração Completa

Uma **k-coloração completa** ocorre quando, para qualquer par de cores distintas, existe um par de vértices adjacentes em G coloridos por este par de cores. A Figura 2.8 apresenta uma 3-coloração completa.

Figura 2.8

– O homomorfismo h é completo e de ordem r quando $h(G) = K_r$.

Um resultado que relaciona homomorfismo completo com coloração completa é dado pelo teorema:

Teorema: h é um homomorfismo completo e de ordem r em G se e somente se G admite uma r -coloração completa.

A sequência da Figura 2.9 ilustra a obtenção de duas imagens homomorfas de G em relação a $h_5(G)$ e $h_6(G)$ ambas completas, dependendo da composição de homomorfismos elementares aplicadas.

Figura 2.9

A família de todos os homomorfismos completos de ordem r em G define todas as r -colorações completas para o grafo G . Portanto, pelo resultado de Hajós (1961), têm-se: $\lambda(G) = \min \{r / h(G) = K_r\}$. O grafo G da Figura 2.8 é 2-cromático, o que corresponde à menor ordem de r dentre todos os homomorfismos completos de G .

7 - Coloração com restrições

Seja $|S| \geq 4$ e $m \leq |S|/2$ SCIE de G pré-escolhidos, V_k , com $|V_k| \geq 2$, $k=1, \dots, m$, não sendo esses subconjuntos necessariamente uma partição de G . Suponha que os vértices de G devem ser coloridos de modo que:

- (i) Nenhum vértice adjacente tenha a mesma cor.
- (ii) Todo vértice $v \in V_k$ tenha a mesma cor.

A determinação do número mínimo de cores necessárias, $\lambda_c(G)$, é um problema de **coloração com restrições**. Logicamente $\lambda_c(G) \geq \lambda(G)$. A obtenção de $\lambda_c(G)$ pode ser feita considerando o número cromático de um grafo $G'(S', A')$, imagem homomorfa quociente de G , pelo homomorfismo quociente:

$$h: G \rightarrow G' \quad \text{tal que } h(v) = \begin{cases} u_k & \text{se } v \in V_k \\ v & \text{caso contrário} \end{cases}$$

Esta função é um homomorfismo quociente definido para a partição de S em SCIE, V_k , e conjuntos unitários $\{v\}$, $v \in \{S - V_k, k=1, \dots, m\}$. Então, G' é obtido pela substituição de cada subconjunto V_k por um vértice u_k através da contração dos vértices de V_k em um único vértice u_k .

A Figura 2.10 ilustra a obtenção de G' que é construído considerando como pré-definidos os seguintes SCIE de G , ou restrições de classe, $V_1 = \{v_1, v_9\}$ e $V_2 = \{v_6, v_{12}\}$. O grafo G é 3-cromático sendo uma possível coloração dada pelas classes: $C_1 = \{v_1, v_3, v_7, v_9, v_{11}\}$; $C_2 = \{v_2, v_4, v_6, v_{12}\}$, $C_3 = \{v_5, v_8, v_{10}\}$. Esta partição atende as restrições de classe impostas e, portanto, $\lambda_c(G) = 3$.

Figura 2.10

A relação entre $\lambda(G)$ e $\lambda(G')$ fica estabelecido pelo resultado: $\lambda_c(G) = \lambda(G')$

Demonstração: Seja $C_i(G')$ a i -ésima classe de coloração de G' em qualquer $\lambda(G')$ -coloração de G' . Como todo subconjunto V_k de S é um SCIE e toda classe de coloração é também um SCIE, uma coloração restrita de G pode ser obtida de uma coloração de G' da seguinte forma:

$$C_i(G) = \begin{cases} C_i(G') & \text{se } C_i(G') \cap \{u_k\} = \emptyset \\ C_i(G') \cup V_k & \text{se } u_k \in C_i(G') \end{cases} \Rightarrow \lambda_c(G) = \lambda(G')$$

8 - Algoritmos para a determinação do número cromático

O número de passos necessários a obtenção de uma solução ótima para a determinação do número cromático cresce exponencialmente com o número de variáveis. Aho et al (1974) demonstraram que este problema é NP-Completo. De fato, não existe até o momento um algoritmo limitado por um polinômio que produza solução ótima para toda e qualquer entrada.

Decidir se um grafo é k -cromático é um problema polinomial para $k = 2$ (Berge, 1976) mas é NP-Completo para qualquer $k \geq 3$ (Stockmeyer, 1973). Porém, para certas classes de grafos a determinação do $\lambda(G)$ é imediata. Por exemplo:

- (i) $\lambda(K_n) = n$
- (ii) $\lambda(Ar) = 2$ para qualquer árvore Ar com pelo menos uma aresta.
- (iii) $\lambda(K_{n,m}) = 2$ para grafos bipartidos com pelo menos uma aresta.
- (iv) $\lambda(C_{2i}) = 2$ e $\lambda(C_{2i+1}) = 3$ para grafos C_n com n vértices formando um único ciclo.

A maioria dos algoritmos heurísticos para a determinação do número cromático apresentam um bom desempenho quanto ao número de cores encontradas para determinadas classes de grafos; assim sendo, a pesquisa na área de coloração em grafos tem buscado construir heurísticas rápidas e de boa qualidade. Nesta direção, porém, há obstáculos relacionados à complexidade do problema. Garey e Johnson (1976) mostram que a menos que $P = NP$ nenhum esquema polinomial pode garantir a utilização de menos do que $2\lambda(G)$ cores. Além disso, para diversos algoritmos heurísticos, existem grafos 3-cromáticos de n vértices para os quais as colorações ótimas obtidas tem $\Omega(n)$ cores (Johnson, 1974), onde $\Omega(n)$ é a complexidade para o melhor caso.

8.1 - Heurísticas seqüenciais

- Os **algoritmos seqüenciais** coloreem os vértices de um grafo, dada uma ordenação inicial dos mesmos, atribuindo-se ao vértice a ser colorido a cor de menor índice distinta da de seus adjacentes. Por esta razão, heurísticas deste tipo são chamadas **gulosas ou míopes** diferenciando-se entre si somente pela regra de ordenação dos vértices. Esta classe de algoritmos pode ainda se subdividir em duas outras: seqüencial simples e seqüencial dinâmica (Gouvêa, 1993).
- **seqüencial simples** - constituída por algoritmos em que a ordenação inicial dos vértices permanece por toda a execução do algoritmo.
- **seqüencial dinâmica** - constituída por algoritmos em que a ordenação pode ser modificada ou até mesmo construída durante a execução do algoritmo considerado.

A classe dos grafos *perfeitamente ordenáveis* reúne todas as famílias de grafos para as quais é possível determinar regras de ordenação de vértices de forma que o algoritmo guloso colora seus grafos de maneira ótima. Por exemplo: árvore de peso mínimo.

O primeiro algoritmo seqüencial foi implementado por Welsh e Powell (1967). A idéia geral da heurística é obter $\lambda(G)$ ou pelo menos uma boa aproximação, examinando os vértices em ordem não-crescente de seus graus. Uma descrição deste algoritmo é feita abaixo e sua complexidade é de $O(n^2)$.

algoritmo de Welsh e Powell

Início

Dado $G(S,A)$, ordenar S em ordem não-crescente de graus $x_1, \dots, x_n$

$C_1 \leftarrow \{x_1\}$,

$C_i \leftarrow \emptyset, i = 2, \dots, n$

Para $j = 2$ **até** n **fazer**

$r \leftarrow \min \{i / \Gamma(x_j) \cap C_i = \emptyset\}$

$C_r \leftarrow C_r \cup \{x_j\}$

FimPara

Fim

No pior caso, o desempenho deste algoritmo é muito ruim. A família de grafos apresentada a seguir foi sugerida por Johnson (1974): seja $G(S_1 \cup S_2, A)$ um grafo bipartido com $|S_1| = |S_2| = n$ e $|A| = n(n-1)$ tal que $\forall x_i \in S_1$ e $y_i \in S_2$ tem-se $\Gamma(x_i) \cap \{y_i\} = \emptyset$ e $\Gamma(x_i) = \{y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n\}$. Aplicando o algoritmo considerando a sequência de vértices $\{x_1, y_1, \dots, x_i, y_i, \dots, x_n, y_n\}$, obtém-se $\lambda(G) = n$ sendo que para qualquer grafo bipartido, $\lambda(G) = 2$ (Figura 2.8).

Figura 2.8

As heurísticas seqüenciais dinâmicas exibem, em geral, resultados computacionais mais favoráveis. Como exemplo, pode-se citar as heurísticas Dsatur (*Degree of Saturation*), de Brélaz (1979) e RLF (*Recursive Largest First*) de Leighton (1979). Os dois algoritmos, citados acima, obtém resultados exatos para grafos bipartidos.

O algoritmo Dsatur, apresentado abaixo, tem complexidade $O(n^2)$. O vértice x a ser colorido em seguida é aquele com o maior grau de saturação, $Dsat(x)$, onde :

$$Dsat(x) = \begin{cases} |\Gamma(x)| & \text{se } \forall y \in \Gamma(x), y \text{ não foi colorido.} \\ \text{Numero de cores utilizadas para colorir } y \in \Gamma(x). \end{cases}$$

algoritmo Dsatur

Início

x_1 = vértice de maior grau em $G(S, A)$

$\lambda(G) \leftarrow 1, C_1 \leftarrow \{x_1\}, C_i \leftarrow \emptyset, i = 2, \dots, n$

Para $j = 2$ **até** n **fazer**

$x_j \leftarrow$ vértice de maior grau de saturação em G .

$r \leftarrow$ grau de saturação de $x_j + 1$

$C_r \leftarrow C_r \cup \{x_j\}, \lambda(G) \leftarrow \max \{\lambda(G), r\}$

FimPara

Fim

Abaixo é fornecida a heurística RLF. A atribuição de cores aos vértices é feita consolidando cada classe de coloração por vez através de um critério guloso.

algoritmo RLF

Dados:

C : próxima classe de coloração a ser construída.

V : conjunto de vértices a colorir

Q : conjunto de vértices não coloridos os quais não podem receber a cor da classe C .

Gr : Grafo residual induzido pelos vértices não alocados em qualquer classe de coloração após a consolidação da classe C .

Início

$V \leftarrow [1..n]$ {Inicialmente nenhum vértice está colorido}

$Q \leftarrow \emptyset$, $k \leftarrow 1$, $pare \leftarrow \text{falso}$

Repita até $pare = \text{verdadeiro}$

$C[k] \leftarrow \emptyset$

$\text{ConstrGs}(GR, V)$; {Construção do grafo residual Gr com os elementos de V }

{Escolher $v_0 \in V$: $|\Gamma(v_0)| = \max \{|\Gamma(v)|, v \in V \text{ no grafo residual } Gr\}$ }

$C[k] \leftarrow C[k] \cup \{v_0\}$,

$Q \leftarrow Q \cup \{v \in Gr / \Gamma(v_0) \cap V \neq \emptyset\}$,

$V \leftarrow V - \{\{v_0\} \cup Q\}$;

Enquanto $V \neq \emptyset$ **faça**

{Escolher $v_1 \in V$: $|\Gamma(v_1)| = \max \{|\Gamma(v) \cap Q| \text{ no grafo residual } Gr\}$ }

$C[k] \leftarrow C[k] \cup \{v_1\}$

$Q \leftarrow Q \cup \{u \in \Gamma(v_1) \cap V\}$

$V \leftarrow V - \{\{v_1\} \cup Q\}$

FimEnquanto

Se $((V = \emptyset) \text{ e } (Q = \emptyset))$ **então** $pare \leftarrow \text{verdadeiro}$ **FimSe**

Senão

$V \leftarrow Q$, $Q \leftarrow \emptyset$, $k \leftarrow k+1$

FimSe

FimRepita

Fim

8.2 - Algoritmo exato de Zikov

Zikov, em 1949, criou um algoritmo que através das operações de religação (R) e contração (C) altera a estrutura de um grafo $G(S,A)$ até obter grafos completos. Evidentemente, existem em G pelo menos dois vértices não adjacentes $x, y \in S$. Caso contrário, o grafo seria completo e $\lambda(G=K_n) = n$.

Seja $G_1 = \beta_{x,y}(G)$ o grafo obtido de G pela inserção da aresta (x, y) e $G_2 = \gamma_{x,y}(G)$ o grafo obtido de G pela condensação dos vértices x e y . Pela natureza das operações estruturais aplicadas, $\lambda(G) = \min\{\lambda(G_1), \lambda(G_2)\}$. Este resultado vem do fato de que para se colorir G , dados dois vértices não adjacentes, ou eles recebem cores iguais ou distintas. Com isto, ou $\lambda(G) = \lambda(G_1)$ ou $\lambda(G) = \lambda(G_2)$. Para obter $\lambda(G_1)$ e $\lambda(G_2)$ respectivamente, repete-se a operação de forma recursiva, até que todos os grafos obtidos sejam cliques.

A seqüência de condensações aplicada a um grafo qualquer G forma um caminho constituído pelas arestas mais à direita de Z_G da raiz (início da construção) até uma folha K_i (fim da construção no ramo K_i) chamada **seqüência completa de condensações de Z_G** . Com isto, uma árvore estritamente binária é construída, sendo que cada vértice corresponde a um grafo da forma $\beta_{x,y}(G')$ ou $\gamma_{x,y}(G')$ e cuja raiz é o grafo original G . Esta árvore é chamada de **árvore de Zikov**, Z_G , e $\lambda(G)$ é o tamanho da menor clique dentre suas folhas.

Como o número de nós da árvore binária pode eventualmente ser exponencial em relação aos vértices do grafo G , a complexidade do Algoritmo de Zikov é $O(2^n)$.

algoritmo Árvore de Zykov

Procedimento Cor($G(S,A)$)

$n \leftarrow |S|$

Se G é completo **então** $\lambda(G) \leftarrow \min\{\lambda(G), n\}$

Senão {Para $x, y \in S$ e $(x, y) \notin A$ }

Cor($\beta_{x,y}(G)$), Cor($\gamma_{x,y}(G)$)

FimSe

Fim {Procedimento Cor}

$\lambda(G) \leftarrow n, \text{Cor}(G)$

Fim

3

Algoritmo de Guénoche para uma p -coloração

1 - Introdução

Seja o problema de particionar um conjunto X de n objetos munidos de uma dissimilaridade, em um número p , fixo, de classes tal que o diâmetro entre as classes seja mínimo. Este problema corresponde a construir um grafo de teto mínimo, G_s , subgrafo parcial de $G(X,A)$, que seja p -colorível, de acordo com o seguinte resultado (Hansen e Delattre, 1978): *Existe uma p -partição de diâmetro inferior a $s \Leftrightarrow G_s$ é p -colorível. Então, para construir uma p -partição de diâmetro mínimo é suficiente encontrar um teto mínimo tal que G_s seja p -colorível.*

Para $p = 2$, os algoritmos polinomiais de Rao (1971), Hubert (1974), Leclerc (1986) e Hansen e Jaumard (1987) enumeram todas as partições em duas classes de diâmetro mínimo. Guénoche (1989) propõe um algoritmo de complexidade $O(n^2)$, baseado na bicoloração de uma árvore de peso máximo para construir uma bipartição de diâmetro mínimo.

Para $p > 2$, Hansen e Delattre (1978) demonstraram que o problema é NP-Completo. Eles propõem um algoritmo baseado em um método de coloração ótima de

grafos teto sucessivos, adaptado a partir do algoritmo de Brown (1972) e constróem para cada valor de p uma só partição de diâmetro mínimo. Karchmer e Naor (1988) descrevem um algoritmo implementado com processamento paralelo.

O algoritmo de Guénoche (1993) enumera todas as partições em um número fixo de classes de diâmetro mínimo. Este algoritmo baseia-se na coloração de um grafo teto em p cores, cada cor definindo uma classe. Utiliza-se várias heurísticas para aproximar o diâmetro e enumera-se as partições de diâmetro mínimo unicamente no passo final.

Existem muitos outros algoritmos propostos na literatura para a resolução do problema enunciado ($p > 2$). Escolhemos o algoritmo de Guénoche (1993) para a realização de um estudo mais aprofundado devido a qualidade das soluções descritas e, principalmente, por este fornecer todas as partições de diâmetro mínimo existentes, o que é importante como ferramenta para o processo de tomada de decisão.

Antes de descrever o algoritmo de Guénoche (1993), em detalhes, pode-se mostrar que este não é polinomial.

Proposição (Guénoche, 1993): *O número máximo de partições de diâmetro mínimo de um conjunto com n elementos em p classes é p^{n-p} .*

Demonstração: *Considere um grafo completo com pesos nas arestas de n nós cujas $p(p-1)/2$ maiores arestas formam uma clique com p elementos. Se as p arestas seguintes (em ordem não-crescente) de valores iguais ao diâmetro entre as classes, $d(P)$, ligam um mesmo nó fora da clique a cada nó da clique, este nó e todos os outros podem ser inseridos em qualquer uma das classes que serão todas de diâmetro $d(P)$. Com isto, têm-se p^{n-p} possibilidades.*

2 - Matriz de Dissimilaridades e Partição

Seja $X = \{x_1, x_2, \dots, x_n\}$ um conjunto de n objetos e $G(X, A)$ um grafo valorado, não-direcionado, cujas arestas tem pesos fornecidos por uma matriz de pesos, $D = [d_{ij}]$, chamada matriz de dissimilaridades sobre X , tal que : $d_{ij} = \lambda(x_i, x_j) = \lambda(x_j, x_i) \geq 0$.

A dissimilaridade em questão mede as diferenças existentes entre os objetos com relação a uma série de atributos considerados relevantes.

Seja $\{C_1, C_2, \dots, C_p\}$ uma partição de X em p classes $\Rightarrow C_i \cap C_j = \emptyset$ e $\cup C_i = X$.

Para comparar as partições construídas sobre X , é definido :

- $d(C_k) = \max\{d_{ij} \text{ , para } x_i, x_j \in C_k\}$ isto é, o diâmetro de uma classe igual à maior dissimilaridade intra-classe.
- $d(P) = \max\{d(C_k), k = 1, \dots, p\}$ isto é, o diâmetro de uma partição igual ao maior diâmetro de suas classes.

3 - Algoritmo de Guénoche (1993)

Seja o problema da enumeração de todas as partições de um conjunto X em um número de classes p , fixado, e de diâmetro $d(P)$ mínimo. Este problema corresponde a construir um grafo teto G_s onde as arestas têm um comprimento superior a $d(P)$ e que seja p -colorível (Hansen e Delattre, 1978). No algoritmo de Guénoche (1993), um limite superior, s , para $d(P)$ é determinado de maneira heurística. Em seguida, enumera-se, as p -colorações para G_s . Enquanto existir ao menos uma p -coloração possível, decrementa-se o valor de s e uma nova iteração é processada. Quando o algoritmo pára, o teto obtido é igual ao diâmetro das partições restantes.

Para enumerar todas as possíveis partições de X , em p classes de diâmetro mínimo, aplica-se um algoritmo constituído de 3 etapas:

1ª etapa - Obtenção do grafo teto G_s p -colorível

2ª etapa - Enumeração de todas as p -colorações de G_s

3ª etapa - Eliminação das partições que não são de diâmetro mínimo

1ª etapa - Obtenção do grafo teto G_s p-colorível.

- Utiliza-se um método heurístico para determinar o teto s , aproximação superior de $d(P)$ tal que G_s seja p-colorível. O teto s é determinado por meio de uma sequência de subdivisões dicotômicas do intervalo de variação das dissimilaridades. Para cada tentativa, utiliza-se um algoritmo guloso de coloração.

2ª etapa - Enumeração de todas as p-colorações de G_s .

- Uma vez fixado o teto s , enumera-se todas as colorações de G_s em p cores. Utiliza-se uma clique, se possível máxima, de G_s onde cada nó é o representante de uma classe, obtida pelo algoritmo D_{sat} . Em seguida, colore-se os outros nós, considerando-os na ordem de saturação, de todas as maneiras possíveis. Se um nó é adjacente a nós coloridos, estas cores não poderão ser utilizadas para colori-lo. Obtém-se assim todas as partições em p classes de diâmetro inferior a s .

3ª etapa - Eliminação das partições que não são de diâmetro mínimo.

- Considera-se as arestas na ordem não-crescente dos valores de dissimilaridades a partir de s . Uma aresta pode ser inserida no grafo enquanto existir uma partição compatível isto é, se a aresta liga nós de classes diferentes. Cada aresta inserida elimina assim algumas partições obtidas anteriormente, se elas existirem. A primeira aresta que não pode ser inserida, uma vez que se for não restariam mais partições compatíveis, tem valor igual a $d(P)$ que é o maior valor de dissimilaridade inter-classes. As partições restantes são de diâmetro mínimo (para um número fixo de classes). Esta etapa é mais rápida se s é uma boa aproximação de $d(P)$.

Obs.: As três etapas do algoritmo serão descritas com mais detalhes nas seções 3.1, 3.2 e 3.3. A seguir, na seção 4 apresentaremos um exemplo ilustrativo e faremos um breve comentário sobre o algoritmo na seção 5.

3.1 - Obtenção do grafo G_s p-colorível

Na primeira etapa, constrói-se um grafo p-colorível cujas arestas são aquelas que apresentam valor superior ou igual ao valor de um teto s a determinar. Para obter este valor de s utiliza-se um método de subdivisão dicotômica e um método seqüencial de coloração, no caso, o algoritmo Dsatur de Brélaz. O algoritmo de *subdivisão*, fornecido a seguir, executa esta tarefa.

Algoritmo de subdivisão

{Num_Cor = número de cores obtidas usando o algoritmo Dsatur.}

dMin $\leftarrow$ Mínimo $\{d_{ij}, (i,j) \in A\}$

dMax $\leftarrow$ Máximo $\{d_{ij}, (i,j) \in A\}$

Enquanto (dMax-dMin) > 2 **faça**

s $\leftarrow$ (dMin+dMax)/2

Construção de G_s colorido por Dsatur

Se num_cor $\leq$ p **então** dMax $\leftarrow$ s **Senão** dMin $\leftarrow$ s

FimSe

FimEnquanto

Fim

Os algoritmos seqüenciais de coloração colorem os nós em uma certa ordem. No algoritmo Dsatur, parte-se de um nó de grau máximo ao qual é atribuído a cor 1. Todos os nós que lhe são adjacentes estão “saturados” por esta cor. Define-se um índice de saturação, para cada nó não colorido, igual ao número de cores diferentes ao qual ele é adjacente. A cada iteração escolhe-se colorir o nó mais saturado e, em caso de empate, retém-se aquele que possui grau máximo no subgrafo parcial dos nós não coloridos. Dá-se a este nó a cor de menor índice.

Guénoche (1993) propõe uma modificação no Dsatur: a cada etapa, colore-se o mesmo nó que seria colorido pelo Dsatur, mas, dá-se a este nó mais saturado, a cor que satura o menor número de nós não coloridos, insaturados por esta cor. Isto consiste em deixar livre mais cores para os nós que serão coloridos depois; e a complexidade do algoritmo continua a mesma.

Exemplo 3.1 (Guénoche, 1993) - O grafo correspondente ao exemplo está ilustrado na Figura 3.1. Inicia-se com o nó de grau máximo x_3 e seus 5 nós adjacentes são saturados com a cor 1. Entre estes, x_2 tem grau máximo; ele recebe a cor 2 e seus 4 nós adjacentes são saturados pela cor 2. Entre os nós saturados pelas duas cores, escolhe-se x_6 que tem grau máximo e recebe a cor 3. Não tem mais, no momento, nova cor que seja necessário utilizar, uma vez que nenhum nó é adjacente aos três nós coloridos. Continua-se a colorir os nós mais saturados e de grau máximo no subgrafo parcial de nós não coloridos. Então x_9 recebe a cor 3, assim como x_5 , o que satura x_1 para esta cor. Até agora não houve escolha a fazer. Obtém-se a Tabela 3.1 de saturação abaixo, na qual x designa a cor do nó e 1 indica os nós saturados por esta cor.

nós	1	2	3	4	5	6	7	8	9
graus	3	4	5	3	2	4	4	5	4
cor 1		1	x		1	1	1		1
cor 2		x	1		1	1			1
cor 3	1	1	1	1	x	x		1	x

Tabela 3.1

Figura 3.1

No grafo parcial restante, os nós x_1 e x_4 tem grau 2 e x_8 grau 3. É preciso colorir x_7 e pode-se escolher entre as cores 2 e 3:

O algoritmo clássico lhe daria a primeira, o que saturaria x_1 , x_4 e x_8 para a cor 2. Logo x_8 só pode receber a cor 1, o que satura x_1 e x_4 a quem não resta nenhuma cor. Chega-se, então, a uma coloração em 4 cores: $\{x_3, x_8\}, \{x_2, x_7\}, \{x_5, x_6, x_9\}, \{x_1, x_4\}$.

Seguindo a modificação proposta, prefere-se colorir x_7 com a cor 3, pois x_1 , x_4 e x_8 já estão saturados por esta cor. Para x_8 que satura x_1 e x_4 a escolha é indiferente, uma vez que nenhum desses nós está saturado pelas cores 1 ou 2 e se x_8 recebe a cor 1, x_1 e x_4 receberão a cor 2. Então, chega-se a uma coloração em três cores, obtendo-se a seguinte partição: $\{x_3, x_8\}, \{x_1, x_2, x_4\}, \{x_5, x_6, x_7, x_9\}$

3.2 - Enumeração das p-colorações de G_s

Dispõe-se agora de um grafo p-colorível com uma ordem sobre os nós onde os primeiros coloridos com cores distintas formam uma clique, obtida pelo algoritmo Dsatur: o primeiro nó estando colorido, colore-se com uma nova cor um nó que lhe seja adjacente, depois se existe um nó adjacente a estes dois, este recebe uma nova cor. Se o cardinal NbQ desta clique é o número de cores obtidas no fim do algoritmo, esta coloração é ótima. É o caso do exemplo 3.1 onde $\{x_2, x_3, x_6\}$ é uma clique.

A clique obtida não é necessariamente máxima dado que, de um lado o problema é NP-Completo e de outro, um grafo p-cromático não possui necessariamente uma clique com p elementos (Capobianco e Molluzzo, 1978).

Para construir todas as p-colorações de G_s , utiliza-se esta clique como inicialização do processo de enumeração. Desenvolve-se uma estrutura arborescente em largura, na qual cada nó da árvore designa um nó do grafo e uma cor associada a este nó. Nesta arborescência, um caminho entre uma folha e a raiz é um início de uma coloração no qual um nó só pode aparecer uma vez. Inicializa-se a arborescência por uma cadeia constituída dos NbQ nós da clique Cl(.) tendo as cores 1, 2, ..., NbQ. A cada iteração, prolonga-se todas as folhas de maior profundidade. Elas são todas associadas a um mesmo nó i. Colore-se o nó j que segue i na ordem de coloração. Para cada folha, retrocedendo-se na arborescência, marca-se as cores dos nós (coloridos) adjacentes a j; estas cores estão proibidas para ele. Todas as outras cores iniciam uma nova folha da arborescência. Quando todos os nós estão coloridos, todas as folhas e os caminhos associados da arborescência correspondem a colorações diferentes.

O grafo G_s é codificado em uma matriz G com n linhas, a i-ésima linha contendo a lista dos nós adjacentes a x_i . $G(i,j)$ é o número do j-ésimo nó adjacente a x_i .

A estrutura de dados utilizada, para a enumeração de todas as p-colorações de G_s , é constituída de 3 vetores V, C, P indexados sobre os nós da arborescência:

- V(i) contém o número de um nó de G_s
- C(i) contém a cor desse nó.
- O índice i designa o i-ésimo nó da arborescência.
- P(i) é o predecessor de i na arborescência.
- O nó seguinte s na ordem de Dsatur é codificado Suc(s)
- O último nó a colorir é chamado Der.

O algoritmo, descrito a seguir, é o responsável pela construção e pela enumeração de todas as diferentes partições do grafo G_s em p classes.

algoritmo Enumeração das p-partições de diâmetro $< s$

$i_1 \leftarrow \text{NbQ}, i_2 \leftarrow \text{NbQ}$

Para $i \leftarrow 1$ a NbQ **faça**

$V(i) \leftarrow \text{Cl}(i), C(i) \leftarrow i, P(i) \leftarrow i-1$ {somente os nós da clique estão em classes}

FimPara

Enquanto $i_1 \leq i_2$ **faça**

$s \leftarrow V(i_1),$

$v \leftarrow \text{Suc}(s)$ {tenta-se colorir s }

Para $c \leftarrow 1$ até p **faça** $\text{coul}(c) \leftarrow 0$ **FimPara**

$k \leftarrow i_1$

Enquanto $k \neq 0$ **faça**

$t \leftarrow V(k)$

Se $t \in \Gamma(v)$ **então** $\text{coul}(C(k)) \leftarrow 1$ **FimSe**

$k \leftarrow P(k)$

FimEnquanto

Para $c \leftarrow 1$ até p **faça**

Se $\text{Coul}(c) = 0$ **então** {a cor c é possível}

Se $v \neq \text{Der}$ **então**

$i_2 \leftarrow i_2 + 1, V(i_2) \leftarrow v, C(i_2) \leftarrow c, P(i_2) \leftarrow i_1$

Senão $\text{Coul}(v) \leftarrow c$ {sai a partição}

FimSe

FimSe

FimPara

FimEnquanto

Fim

Este algoritmo é de complexidade proporcional ao número de nós e ao número de colorações em p cores do grafo G_s (Guénoche, 1993). Quanto menor o teto s , maior o número de arestas do grafo e menos colorações existem. É por isto que se toma tanto cuidado, através de uma busca dicotômica, em diminuir s para depois inserir arestas, sem se contentar com o primeiro teto que permita uma p -coloração.

3.3 - Eliminação das partições que não são de diâmetro mínimo

Até o presente momento, construiu-se todas as partições de diâmetro inferior ao teto s que define o grafo G_s . Considera-se as arestas de comprimento inferior a s , em ordem não-crescente. Se para uma aresta (i, j) existe ao menos uma partição tal que i e j não estejam na mesma classe, existe então uma partição de diâmetro $d_{ij} < s$. Se todas as arestas de valor d_{ij} satisfazem esta condição, as partições restantes são no máximo de diâmetro $d_{ij} < s$.

Considera-se, então, todas as arestas de valor $d(P)$ imediatamente inferior a s . Se existir ao menos uma partição compatível com estas arestas, todas as partições compatíveis têm diâmetro inferior a s ; itera-se o procedimento fazendo $s \leftarrow d(P)$ e armazenando-se apenas as partições compatíveis. Se nenhuma partição é compatível com todas as arestas de valor $d(P)$, as partições restantes tem diâmetro $d(P)$ e o algoritmo pára.

4 - Exemplo Ilustrativo

Uma aplicação do algoritmo de Guénoche (1993) é apresentada a seguir. O exemplo ilustrativo utiliza um conjunto, de 9 objetos, que se deseja particionar em 3 classes de modo que o diâmetro entre as classes seja mínimo. A Tabela 3.2, abaixo, apresenta a matriz de dissimilaridades D entre os objetos.

Matriz de dissimilaridades									
	1	2	3	4	5	6	7	8	9
1	0								
2	17	0							
3	49	3	0						
4	68	69	29	0					
5	23	39	67	23	0				
6	12	63	3	44	6	0			
7	73	38	89	56	71	81	0		
8	84	94	39	95	68	51	40	0	
9	90	66	55	28	19	51	65	15	0

Tabela 3.2

Aplica-se as três etapas do algoritmo sobre a matriz de dissimilaridades D .

1ª etapa:

$D_{\max} = 95$ e $D_{\min} = 3$. Usando o algoritmo de subdivisão, obtém-se 4 cores para o teto 49, 2 cores para o teto 72, 4 cores para o teto 60, 3 cores para o teto 66, 4 cores para o teto 63 e, finalmente, 3 cores para o teto 64. A partição obtida para este teto é fornecida na Tabela 3.3 abaixo.

Classe 1	Classe 2	Classe 3
7,8	1,2,3,6	4,5,9

Tabela 3.3

A representação geométrica do grafo G_{64} , considerando a clique $\{7,1,9\}$ obtida no algoritmo Dsatur, podem ser vista na Figura 3.1 abaixo.

Figura 3.2

2ª Etapa:

Utiliza-se a clique $C_1 = \{7,1,9\}$ para enumerar todas as possíveis colorações deste grafo em 3 cores, obtendo-se 6 partições de diâmetro estritamente inferior a 64. É fácil verificar que essas partições são as únicas existentes. A Tabela 3.4, fornecida a seguir, apresenta as 6 partições obtidas.

Partição	Classe 1	Classe 2	Classe 3
1	7,8	1,2,3	4,5,6,9
2	7,8	1,2,3,6	4,5,9
3	7,8	1,2,5	3,4,6,9
4	7,8	1,2,5,6	3,4,9
5	4,7	1,2,5	3,6,8,9
6	4,7	1,2,5,6	3,8,9

Tabela 3.4

A Figura 3.3 ilustra a obtenção das 6 partições em 3 cores do grafo G_{64} utilizando a matriz de adjacências do grafo (Tabela 3.5).

Grafo G_{64}	
vértices	adjacências
1	4, 7, 8, 9
2	4, 8, 9
3	5, 7
4	1, 2, 8
5	3, 7, 8
6	7
7	1, 3, 5, 6, 9
8	1, 2, 4, 5
9	1, 2, 7

Tabela 3.5

Figura 3.3

3ª Etapa

Na terceira etapa, ordena-se as arestas em ordem não-crescente de dissimilaridades cujo valor seja inferior ao teto $s = 64$, obtendo-se:

(2, 6) / (4, 7) / (3, 9) / (6, 8), (6, 9) / (1, 3) / (4, 6) / (7, 8) / (2, 5), (3, 8) / (2, 7) /
(3, 4) / (4, 9), / (1, 5), (4, 5) / (5, 9) / (1, 2) / (8, 9) / (1, 6) / (5, 6) / (2, 3), (3, 6)

Examina-se as arestas de comprimento imediatamente inferior a s : (2, 6) de valor 63 que é compatível com a primeira, terceira e quinta partições. Com isto, elimina-se a segunda, quarta e sexta partições. Examina-se a aresta (4, 7) de valor 56 que é compatível com a primeira e terceira partição. A quinta partição é, então, eliminada. A próxima aresta, (3, 9), só é compatível com a primeira partição.

Examina-se as arestas (6,8) e (6,9), de valor 51: a aresta (6,8) é compatível com a primeira partição mas (6,9) não o é. Com isto, a partição restante tem diâmetro igual a 51 e é a única existente: o vértice x_3 está necessariamente na classe de x_1 assim como x_4 está na classe de x_9 o que obriga x_8 a estar na classe de x_7 e x_2 a estar na classe de x_1 . Com isto, o vértice x_6 vai para a classe de x_9 .

A Tabela 3.6 fornece a partição resultante e a Figura 3.4 apresenta uma representação geométrica do grafo G_{55} com a única 3-coloração possível para o mesmo.

Classe 1	Classe 2	Classe 3
7,8	1,2,3	4,5,6,9

Tabela 3.6

Figura 3.4

O teto obtido, $s = 55$, é o mínimo pois para $s = 51$ ($G_{55} = G_{54} = G_{53} = G_{52}$), o grafo será 4-cromático (Figura 3.5).

Figura 3.5

5 - Comentários

Com o objetivo de realizar estudos e explorar as possíveis vantagens e desvantagens das diferentes versões do algoritmo de Guénoche, implementamos duas versões por ele propostas: Guénoche (1989) e Guénoche (1993). A idéia geral da heurística nas duas versões é a mesma e pode ser dividida em três etapas:

1ª etapa - Obtenção do grafo teto G_s p-colorível.

2ª etapa - Enumeração de todas as p-colorações de G_s .

3ª etapa - Eliminação das partições que não são de diâmetro mínimo.

O algoritmo de 1993 exhibe melhores resultados, pois utiliza algoritmos mais eficientes para a coloração e para a enumeração de todas as p-colorações de G_s . Além disso, na versão de 1989, ao se obter o grafo p-colorível e passar para a segunda etapa é necessária a construção de uma clique através de um método aproximado, utilizada como inicialização do processo de enumeração; na versão de 1993, os NbQ primeiros nós já formam uma clique obtida pelo algoritmo Dsatur.

Na primeira etapa da heurística proposta por Guénoche (1989), ao se obter o teto s , faz-se uma fase de *otimização do teto*: considera-se as arestas em ordem não-crescente, de comprimento imediatamente inferior a s ; seja (i, j) esta aresta:

- Se i e j estão em classes diferentes, insere-se (i, j) a G_s (*condição1*).
- Se (i, j) liga nós de uma mesma classe, tenta-se colocar i ou j em uma outra classe sem aumentar o diâmetro. Sendo isto possível, insere-se (i, j) a G_s (*condição2*).

Realizamos 20.000 testes computacionais, com conjuntos de 9 elementos para serem distribuídos em três classes, com a versão de 1989 e com a versão de 1993. Implementamos três versões para a realização da primeira etapa do algoritmo proposto em Guénoche (1993): sem a fase de *otimização do teto* (*versão1*), isto é conforme descrito em Guénoche (1993); com a fase de *otimização do teto* como descrito em Guénoche (1989), ie., com a *condição1* e a *condição2* (*versão2*) e uma versão, proposta por nós, com a fase de *otimização do teto* utilizando somente a *condição1* (*versão3*).

O objetivo destes testes foi observar o comportamento do algoritmo nas duas versões e verificar o número de grafos em que foi necessária a utilização da fase de *otimização do teto*. Os resultados podem ser vistos na Tabela 3.7.

Obs.: O acréscimo de tempo computacional exigido pela *versão3* do algoritmo é pequeno, pois nesta versão a *otimização do teto* se reduz a comparar os vértices das arestas de valor inferior a s , e estando esses em classes distintas, inseri-los no grafo.

Vamos considerar:

N - número de elementos do conjunto a particionar.

p - número de classes em que se deseja particionar o conjunto.

novoteto - teto obtido na primeira etapa com a fase de otimização do teto.

velhoteto - teto obtido na primeira etapa sem a fase de otimização do teto.

X1 - número de grafos em que $\text{novoteto} < \text{velhoteto}$ na versão de 1989.

X2 - número de grafos em que $\text{novoteto} < \text{velhoteto}$ na *versão2*.

X3 - número de grafos em que $\text{novoteto} < \text{velhoteto}$ na *versão3*.

Ng1 = Número de grafos em que se gastou menos tempo computacional na *versão1* em relação à *versão2*, sendo essa diferença superior a 10^{-3} segundos.

Ng2 = Número de grafos em que se gastou menos tempo computacional na *versão2* em relação à *versão1*, sendo essa diferença superior a 10^{-3} segundos.

TerceiraEtapa = Número de grafos em que não houve necessidade da 3ª etapa na *versão3*, considerando-se apenas aqueles grafos em que houve diferença no número de partições obtidas na 2ª etapa entre a *versão1* e a *versão3*.

A Tabela 3.7, abaixo, fornece os resultados obtidos.

Resultados obtidos utilizando-se as diferentes versões do algoritmo implementadas	
N	9
p	3
Número total de grafos testados	20.000
X1	2.370
X2	148
X3	139
Ng1	10
Ng2	14
TerceiraEtapa	127

Tabela 3.7

A partir dos resultados obtidos nos testes computacionais, mostrados na Tabela 3.7, pode-se fazer as seguintes observações:

1. Na maioria dos casos foi dispensável a fase de *otimização do teto*. Porém, na versão de 1989, em 11,84% dos casos ela foi utilizada enquanto na versão de 1993 em apenas 0,740% dos casos.
2. A terceira etapa mostrou-se necessária, mesmo com *otimização do teto*, embora seu uso tenha sido menos exigido (14,2% dos casos).
3. Utilizando a *versão3* do algoritmo, a fase de *otimização do teto*, como descrita, foi útil em 0,695% dos casos.

Para a aplicação de particionamento de um conjunto de disciplinas em um número fixo de horários, que apresentaremos no capítulo 4, utilizamos a *versão3* do algoritmo implementado, pois:

1. O acréscimo de tempo computacional exigido por esta versão é pequeno, pois a fase de *otimização do teto* se reduz a comparar os vértices das arestas de valor inferior a s , e estando estes em classes distintas, inseri-los no grafo.
2. Embora, na maioria dos exemplos testados, tenha sido dispensável a fase de *otimização do teto*, nos casos em que isto não ocorreu (0,695 %), o número de partições obtidas na segunda etapa, usando a *versão3* do algoritmo, foi inferior ao número de partições obtidas na segunda etapa, na versão descrita em Guénoche (1993).
3. A terceira etapa é mais rápida (elimina-se um número menor de partições) ou desnecessária se s é uma boa aproximação de $d(P)$. Pelos resultados obtidos, fazendo-se uso somente da *condição1*, na maioria dos casos (85,8 %), a terceira etapa foi desnecessária.

Apresentamos, a seguir, dois exemplos ilustrativos: Exemplo 3.2 e Exemplo 3.3. Em ambos, deseja-se particionar um conjunto de 9 objetos em 3 classes tal que o diâmetro $d(P)$ seja mínimo. Para este fim utilizamos a *versão1*, *versão2* e a *versão3* do algoritmo implementadas.

Exemplo 3.2 - Neste exemplo, o objetivo é ilustrar os casos em que o número de partições obtidas com a *versão1*, na 2ª etapa, é consideravelmente maior que o número de partições obtidas com a *versão3*. Além disso, a utilização da *versão3*, neste caso, implica na não utilização da 3ª etapa. A Tabela 3.8, abaixo, apresenta a matriz de dissimilaridades.

Matriz de Dissimilaridades									
	1	2	3	4	5	6	7	8	9
1	0	0							
2	66	0							
3	83	5	0						
4	35	89	86	0					
5	30	95	68	64	0				
6	90	22	93	26	93	0			
7	47	27	21	38	30	53	0		
8	29	86	30	83	98	43	5	0	
9	21	18	69	22	0	98	43	84	0

Tabela 3.8

Se na 1ª etapa do algoritmo não se faz uso da fase de *otimização do teto*, o teto encontrado é $s = 84$ e obtém-se 48 partições na 2ª etapa (Tabela 3.9).

PARTIÇÕES COM DIÂMETRO $d(P) < 84$							
	Classe 1	Classe 2	Classe 3		Classe 1	Classe 2	Classe 3
1	2,6,7	1,3,5,9	4,8	25	4,6,8,	1,3,5,7	2,9
2	4,6,7,8	1,3,5,9	2	26	4,6,8,	1,3,5	2,7,9
3	2,6,7,	1,4,5,9	3,8	27	4,6,8,	1,5,7	2,3,9
4	4,6,7,8,	1,5,9	2,3	28	4,6,8,	1,5	2,3,7,9
5	4,6,7,8,	1,3,5	2,9	29	6,8,	1,4,5,7,9	2,3,
6	4,6,7,8,	1,5	2,3,9	30	6,8,	1,4,5,9	2,3,7
7	6,7,8,	1,4,5,9	2,3	31	6,8,	1,4,5,7	2,3,9
8	6,7,8,	1,4,5	2,3,9	32	6,8,	1,4,5	2,3,7,9
9	2,6,7,	3,5,9	1,4,8	33	2,6,	3,5,7,9	1,4,8
10	4,6,7,8,	3,5,9	1,2	34	2,6,	3,5,9	1,4,7,8
11	2,6,7,	4,5,9	1,3,8	35	4,6,8,	3,5,7,9	1,2
12	4,6,7,8,	5,9	1,2,3	36	4,6,8,	3,5,9	1,2,7
13	4,6,7,8,	3,5	1,2,9	37	2,6,	4,5,7,9	1,3,8
14	4,6,7,8,	5	1,2,3,9	38	2,6,	4,5,9	1,3,7,8
15	6,7,8,	4,5,9	1,2,3	39	4,6,8,	5,7,9	1,2,3
16	6,7,8,	4,5	1,2,3,9	40	4,6,8,	5,9	1,2,3,7
17	2,6,	1,3,5,7,9	4,8	41	4,6,8,	3,5,7	1,2,9
18	2,6,	1,3,5,9	4,7,8	42	4,6,8,	3,5	1,2,7,9
19	4,6,8,	1,3,5,7,9	2	43	4,6,8,	5,7	1,2,3,9
20	4,6,8,	1,3,5,9	2,7	44	4,6,8,	5	1,2,3,7,9
21	2,6,	1,4,5,7,9	3,8	45	6,8,	4,5,7,9	1,2,3
22	2,6,	1,4,5,9	3,7,8	46	6,8,	4,5,9	1,2,3,7
23	4,6,8,	1,5,7,9	2,3	47	6,8,	4,5,7	1,2,3,9
24	4,6,8,	1,5,9	2,3,7	48	6,8,	4,5	1,2,3,7,9

Tabela 3.10

Para aplicar a fase de *otimização do teto*, ordena-se as dissimilaridades em ordem não-crescente, de valor inferior a $s = 84$:

(1,3),(4,8) / (3,9) / (3,5) / (1,2) / (4,5) / (6,7) / (1,7) / (6,8),(7,9) / (4,7) / (1,4) /
(1,5),(5,7),(3,8) / (1,8) / (2,7) / (4,6) / (2,6),(4,9) / (3,7),(1,9) / (2,9) / (2,3),(7,8)

- As arestas (1,3), (4,8), (3,9), (3,5), (1,2) podem ser inseridas no grafo pois estão em classes diferentes o que faz com que o teto seja $s = 66$. A aresta (4,5), de valor 64, não pode ser inserida no grafo pois os vértices 4 e 5 estão na mesma classe. Assim, obtemos já na 2ª etapa as 6 partições de diâmetro mínimo, apresentadas na Tabela 3.10, e que só são obtidas na *versão1* na 3ª etapa.

Partições de diâmetro $D(P) = 64$			
1	2,6,7,	1,4,5,9,	3,8
2	6,7,8	1,4,5,9	2,3
3	2,6	1,4,5,7,9	3,8
4	2,6	1,4,5,9,	3,7,8
5	6,8	1,4,5,7,9	2,3
6	6,8	1,4,5,9	2,3,7

Tabela 3.10

Exemplo 3.3 - Este exemplo faz uso de um grafo em que, embora o número de partições obtidas na *versão2* ou *versão3* seja inferior ao número de partições obtidas na *versão1*, é necessário o processamento da terceira etapa para a obtenção das partições de diâmetro mínimo. A Tabela 3.11 apresenta a matriz de dissimilaridades.

Matriz de Dissimilaridades									
	1	2	3	4	5	6	7	8	9
1	0								
2	76	0							
3	33	82	0						
4	87	40	63	0					
5	2	84	7	40	0				
6	16	51	84	44	53	0			
7	77	48	15	64	80	84	0		
8	91	19	49	94	2	67	37	0	
9	38	0	37	85	79	40	5	35	0

Tabela 3.11

A Tabela 3.12 exhibe o resultado da partição obtida com o algoritmo de coloração em que se particiona o conjunto de 9 elementos em 3 cores:

Classe 1	Classe 2	Classe 3	Clique	Teto s
2,4,6	1,5	3,7,8,9	4,1,7	54

Tabela 3.12

Se na primeira etapa do algoritmo, não se faz uso da fase de *otimização do teto*, o teto encontrado é $s = 54$ e obtém-se 6 partições na segunda etapa (Tabela 3.13).

Partições de diâmetro $D(P) < 54$			
1	2,4,6	1,3,5	7,8,9
2	4,5,6	1,3,9	2,7,8
3	4,6	1,3,5	2,7,8,9
4	4,5,6	1,3	2,7,8,9
5	2,4,6	1,5	3,7,8,9
6	2,4	1,5,6	3,7,8,9

Tabela 3.13

Para aplicar a fase de *otimização do teto*, ordena-se as dissimilaridades em ordem não-crescente inferior a $s = 54$:

(5,6) / (2,6) / (3,8) / (2,7) / (4,6) / (2,4),(4,5),(6,9) / (1,9) / (7,8),(3,9) / (8,9) / (1,3) / (2,8) / (1,6) / (3,7) / (3,5) / (7,9) / (1,5),(5,8)

- A aresta (5,6) pode ser inserida no grafo, mas a aresta (2,6) não, o que torna $s = 53$ e as partições obtidas para este teto são as partições (1), (3) e (5) da Tabela 3.13.
- Ao se utilizar a terceira etapa do algoritmo, a partição restante têm diâmetro mínimo, $d(P) = 49$, sendo a partição (3), da Tabela 3.13, a única existente.

4

Aplicação: planejamento de horário escolar

1 - Introdução

Neste capítulo é apresentada a implementação computacional realizada e os resultados obtidos para a resolução de um problema de matrícula de alunos na universidade, com dados gerados aleatoriamente. O método codificado tem por base o algoritmo de Guénoche apresentado no capítulo 3.

O problema de se determinar um horário escolar pode, impondo certas restrições, ser resolvido por coloração em grafos e tem grande interesse prático. Numerosos outros problemas de alocação de recursos e partição de objetos em classes constituem exemplos que podem ser resolvidos com a ajuda desta ferramenta, introduzindo-se na mesma os dados gerais, restrições e recursos específicos de cada um dos exemplos estudados.

O problema computacional foi escrito em linguagem Pascal e testado utilizando-se um microcomputador IBM-PC compatível.

2 - Enunciado do Problema

Procura-se determinar um horário de aulas para um certo período escolar. Diversos fatores devem ser levados em conta, tais como: quais disciplinas serão oferecidas no semestre, quantos alunos desejam cursá-las (para isso, é necessário o processamento de uma pré-matrícula), quantas turmas de cada disciplina serão oferecidas, a carga horária semanal, a periodicidade das disciplinas, se a disciplina é obrigatória ou optativa, o espaço físico necessário, se há necessidade de aulas práticas, se um professor ministrará mais de uma disciplina, etc.

Para simplificação do problema e para que o resultado do exemplo ilustrativo seja de fácil verificação, apenas três fatores são considerados:

1. O número de dias por semana em que haverá aulas, o número de aulas por dia e a carga horária das disciplinas.
2. A prioridade de um aluno em relação a outro, para conseguir matrícula em determinada disciplina, em função do rendimento escolar em uma escala variando de 1 a 10.
3. A escola oferece diversos cursos. O número total de disciplinas oferecidas no período é igual a n . Na pré-matrícula o aluno pode escolher um número $r_i \leq n_i$ de disciplinas entre aquelas oferecidas para o seu curso (i) específico.

3 - Algoritmo

O algoritmo desenvolvido é constituído dos seguintes passos fundamentais:

algoritmo matrícula

Início

Passo 1: Computação da matriz de dissimilaridades

Passo 2: Aplicação da *versão 3* do algoritmo de p-coloração (Cap. 3 p. 44)

Passo 3: Escolha da solução para o problema segundo uma medida de eficiência

Fim {algoritmo matrícula}

4 - Cálculo da dissimilaridade entre as disciplinas

A dissimilaridade entre os elementos de um conjunto pode ser calculada de diferentes maneiras, procurando explorar as características específicas de cada problema tratado. Witte (1980) faz um amplo estudo a este respeito.

Em nossa aplicação, a dissimilaridade entre as disciplinas é calculada de maneira original, através da seguinte fórmula:

$$D(Dis_i, Dis_j) = \begin{cases} \sum_{k \in S(i) \cap S(j)} \text{prioridade\_aluno}(k) & \text{se } i \neq j \\ 0 & \text{se } i = j \end{cases}$$

onde $\begin{cases} S(d) = \{\text{conjunto de alunos pré-matriculados na disciplina } Dis_d\} \\ \text{prioridade\_aluno}(k) = \text{prioridade do aluno } k. \end{cases}$

Esta fórmula leva em consideração o número de alunos pré-matriculados em cada disciplina e a prioridade de cada aluno. Uma variante para esta fórmula consiste em se efetuar o cálculo das dissimilaridades entre as disciplinas multiplicando o somatório acima por $|S_i \cap S_j| / |S_i \cup S_j|$.

Outras variantes para a fórmula apresentada acima podem ser propostas, com o objetivo de eliminar ou diminuir possíveis distorções existentes no cálculo da dissimilaridade em cada problema concreto.

5 - Matriz Potencial de Conflito

A matriz potencial de conflito das disciplinas é obtida pela interseção do número de alunos pré-matriculados entre duas disciplinas, ou seja :

$$P(Dis_i, Dis_j) = |S(i) \cap S(j)|, \quad i, j = 1, \dots, ndis$$

$ndis$ = número total de disciplinas

$P(Dis_i, Dis_j)$ é uma matriz que traduz o número de alunos interessados em cursar as disciplinas Dis_i e Dis_j .

6 - Medida de eficiência para auxílio à tomada de decisão

Entre as diferentes soluções encontradas pelo algoritmo de coloração utilizado em nosso método, é preciso escolher uma das partições obtidas para implementação. Com o objetivo de auxiliar no processo de tomada de decisão, propomos o cálculo de uma medida de eficiência correspondendo a cada uma das soluções apresentadas. A fórmula proposta é dada por:

$$\text{Eficiência} = \sum_{i=1}^{\text{npedidos}} x_i \cdot na_i / \text{nalunos}$$

onde: x_i = fator de ponderação associado ao atendimento em i pedidos.

na_i = número de alunos atendidos em i pedidos, $i = 1, \dots, \text{npedidos}$

npedidos = número de disciplinas que o aluno pode pedir para cursar.

nalunos = número total de alunos.

7 - Resolução do Problema Teste

Entre os testes computacionais realizados, escolhemos o problema teste resolvido abaixo para ilustrar detalhadamente os passos do procedimento.

7.1 - Dados do problema

- A escola mantém 3 cursos distintos e um total de 12 disciplinas estão sendo oferecidas no semestre.
- Na pré-matrícula, o aluno deve escolher 4 disciplinas para cursar no referido semestre.
- A prioridade de cada aluno é pré-estabelecida por uma rotina específica, baseada no rendimento escolar, por exemplo.
- É feito o processamento da pré-matrícula de 48 alunos.
- Caso o aluno (i) não tenha sido plenamente atendido nas disciplinas escolhidas na pré-matrícula, a escola oferece as opções de disciplinas que ele pode se matricular.
- As 12 disciplinas serão distribuídas em 4 horários.
- A relação disciplina/curso está relacionada na Tabela 4.1 abaixo.

Curso	Disciplinas
Curso(1)	1,...,8
Curso(2)	1,...,3,5,9,...,12
Curso(3)	1,5,...,10

Tabela 4.1

A Tabela 4.2 mostra o resultado do processamento de uma pré-matrícula realizada por 48 alunos, assim como o curso e a prioridade de cada aluno.

PRÉ-MATRÍCULA			
Aluno	Prioridade	Curso	Matérias
1	07	1	1,2,3,5
2	05	1	1,4,6,7
3	04	1	4,6,7,8
4	04	1	3,4,6,8
5	02	1	1,6,7,8
6	10	1	1,2,3,4
7	06	1	1,2,7,8
8	01	1	4,5,6,7
9	08	1	1,3,4,5
10	07	1	1,3,4,5
11	02	1	1,4,5,8
12	10	1	1,2,3,4
13	09	1	1,2,3,4
14	09	1	1,3,4,8
15	07	1	2,4,6,7
16	06	1	2,3,5,6
17	09	1	1,2,3,5
18	01	1	5,6,7,8
19	02	2	2,3,5,8
20	03	2	2,5,10,12
21	04	2	1,3,9,10
22	05	2	2,5,9,11
23	06	2	5,9,10,12
24	09	2	5,9,11,12
25	03	2	1,3,5,12
26	10	2	9,10,11,12
27	10	2	9,10,11,12
28	08	2	5,10,11,12
29	07	2	1,9,11,12
30	04	2	1,2,9,12
31	09	2	9,10,11,12
32	05	2	1,5,9,12
33	01	2	1,2,11,12
34	10	2	9,10,11,12
35	07	2	1,3,5,9
36	06	3	1,6,8,11
37	02	3	1,6,7,10
38	01	3	7,8,9,10
39	04	3	1,6,7,10
40	04	3	1,7,9,10
41	07	3	6,7,8,9
42	09	3	7,9,10,11
43	10	3	1,6,9,11
44	10	3	1,6,9,11
45	08	3	5,6,7,11
46	05	3	1,7,9,11
47	03	3	5,6,9,11
48	02	3	1,7,9,11

Tabela 4.2

7.2 - Obtenção das matrizes de potencial de conflito e de dissimilaridades

A Tabela 4.3 fornece a matriz potencial de conflito e a Tabela 4.4 a matriz de dissimilaridades, calculadas conforme fórmulas descritas nos itens 5 e 6.

Matriz Potencial de Conflito												
	1	2	3	4	5	6	7	8	9	10	11	12
1	27											
2	08	13										
3	11	07	14									
4	08	06	08	12								
5	08	06	08	04	19							
6	07	02	02	05	05	16						
7	08	02	00	04	03	10	16					
8	05	01	02	04	02	06	06	10				
9	10	02	02	00	06	04	06	02	21			
10	04	01	01	00	04	03	05	01	10	14		
11	07	02	00	00	04	04	04	01	12	06	16	
12	05	04	02	00	07	00	00	00	09	07	08	14

Tabela 4.3

Matriz de Dissimilaridades												
	1	2	3	4	5	6	7	8	9	10	11	12
1	0											
2	56	0										
3	83	53	0									
4	60	36	57	0								
5	48	32	49	18	0							
6	39	13	10	21	19	0						
7	30	13	0	17	10	41	0					
8	25	6	13	19	3	24	21	0				
9	58	9	11	0	35	30	28	8	0			
10	14	3	4	0	20	9	20	1	66	0		
11	41	6	0	0	30	34	24	6	96	56	0	
12	20	10	5	0	36	0	0	0	70	56	64	0

Tabela 4.4

7.3 - Soluções obtidas

A aplicação do algoritmo de p-coloração fornece um teto mínimo $s = 25$. A Tabela 4.5 fornece todas as partições com diâmetro inferior a s .

Partição	Classe 1	Classe 2	Classe 3	Classe 4
1	1,12	2,8,9	4,5,6,10	3,7,11
2	1,12	3,8,9	4,5,6,10	2,7,11
3	1,12	2,9	4,5,6,8,10	3,7,11
4	1,12	2,9	4,5,6,10	4,7,8,11
5	1,12	3,9	4,5,6,8,10	2,7,11
6	1,12	3,9	4,5,6,10	2,7,8,11

Tabela 4.5

7.4 - Eficiência das soluções obtidas

O fator de ponderação escolhido associado ao número de alunos atendidos em i pedidos efetuados é igual a:

$$x_i = i/npedidos$$

Desta forma, procura-se privilegiar soluções que atendam o maior número de pedido de disciplinas realizado pelos alunos.

Para o nosso exemplo particular, a fórmula da eficiência calculada de acordo com o item 6 é da forma:

$$\text{Eficiência} = (0.25na_1 + 0.50na_2 + 0.75na_3 + na_4) / 48$$

Entre as seis partições encontradas no item anterior, a partição (1) é a que apresenta maior eficiência, igual a 82,8%, o que significa que dentre todas as partições obtidas pelo algoritmo, a partição (1) é a que satisfaz o maior número de alunos em função do atendimento aos pedidos de disciplinas na pré-matrícula.

A Tabela 4.6 apresenta o número de alunos atendidos em i pedidos, $i = 1, \dots, 4$ e a eficiência.

na_1	na_2	na_3	na_4	$nalunos$	Eficiência
00	04	25	19	48	0.82813

Tabela 4.6

7.5 - Horário de Aulas

Um possível horário de aulas é fornecido na Tabela 4.7. A relação da matrícula, pré-matrícula e prioridade dos alunos correspondentes, podem se vistos na Tabela 4.8, assim como as opções de disciplinas que o aluno (i) pode se matricular, caso ele não tenha sido plenamente atendido nas disciplinas escolhidas na pré-matrícula.

Horário/Dia	Dia 1	Dia 2
Horário 1	Dis(1), Dis(12)	Dis(4), Dis(5), Dis(6), Dis(10)
Horário 2	Dis(2), Dis(8), Dis(9)	Dis(3), Dis(7), Dis(11)

Tabela 4.7

Aluno	Prioridade	Curso	Pré-Matricula	Matrícula	Opções
1	07	1	1,2,3,5	1,2,3,5	
2	05	1	1,4,6,7	1,4,7	2,8
3	04	1	4,6,7,8	4,7,8	1
4	04	1	3,4,6,8	3,4,8	1
5	02	1	1,6,7,8	1,6,7,8	
6	10	1	1,2,3,4	1,2,3,4	
7	06	1	1,2,7,8	1,2,7	4,5,6
8	01	1	4,5,6,7	4,7	1,2,8
9	08	1	1,3,4,5	1,3,4	2,8
10	07	1	1,3,4,5	1,3,4	2,8
11	02	1	1,4,5,8	1,4,8	3,7
12	10	1	1,2,3,4	1,2,3,4	
13	09	1	1,2,3,4	1,2,3,4	
14	09	1	1,3,4,8	1,3,4,8	
15	07	1	2,4,6,7	2,4,7	1
16	06	1	2,3,5,6	2,3,5	1
17	09	1	1,2,3,5	1,2,3,5	
18	01	1	5,6,7,8	5,7,8	1
19	02	2	2,3,5,8	2,3,5,12	
20	03	2	2,5,10,12	2,5,12	3,11
21	04	2	1,3,9,10	1,3,9,10	
22	05	2	2,5,9,11	2,5,11	1,12
23	06	2	5,9,10,12	5,9,12	3,11
24	09	2	5,9,11,12	5,9,11,12	
25	03	2	1,3,5,12	1,3,5	2,9
26	10	2	9,10,11,12	9,10,11,12	
27	10	2	9,10,11,12	9,10,11,12	
28	08	2	5,10,11,12	5,11,12	2,9
29	07	2	1,9,11,12	1,9,11	5,10
30	04	2	1,2,9,12	1,2	3,5,10,11
31	09	2	9,10,11,12	9,10,11,12	
32	05	2	1,5,9,12	1,5,9	3,11
33	01	2	1,2,11,12	1,2,11	5,10
34	10	2	9,10,11,12	9,10,1 ¹ ,12	
35	07	2	1,3,5,9	1,3,5,9	
36	06	3	1,6,8,11	1,6,8,11	
37	02	3	1,6,7,10	1,6,7	8,9
38	01	3	7,8,9,10	7,8,10	1
39	04	3	1,6,7,10	1,6,7	8,9
40	04	3	1,7,9,10	1,7,9,10	
41	07	3	6,7,8,9	6,7,8	1
42	09	3	7,9,10,11	7,9,10	1
43	10	3	1,6,9,11	1,6,9,11	
44	10	3	1,6,9,11	1,6,9,11	
45	08	3	5,6,7,11	5,7	1,8,9
46	05	3	1,7,9,11	1,7,9	5,6,10
47	03	3	5,6,9,11	5,9	1,7
48	02	3	1,7,9,11	1,7,9	5,6,10

Tabela 4.8

8 - Comentários

O exemplo resolvido neste capítulo foi gerado aleatoriamente e é um dos vários testes que realizamos com o programa computacional desenvolvido. Os resultados obtidos mostraram-se de boa qualidade, o que constitui uma perspectiva animadora para a continuidade dos estudos.

Na implementação realizada é proposta uma maneira de computar a dissimilaridade entre as disciplinas e calcular um índice de eficiência com o objetivo de auxiliar no processo de tomada de decisão. O programa computacional, além de propor uma solução para o problema tratado, apresenta aos alunos opções de disciplinas que eles podem se matricular (ver Tabela 4.8 coluna opções), uma vez que não há conflitos entre as disciplinas demandadas por estes e as disciplinas relacionadas como opções.

Para o exemplo apresentado, obteve-se índice de eficiência = 82,8% e tempo computacional de 0,14 segundos (processador INTEL DX4-100 MHz).

A fórmula utilizada para o cálculo da matriz de dissimilaridades mostra que os alunos com prioridade maior foram beneficiados, o que era a intenção. Para este fim, fez-se o cálculo da correlação (r) entre a prioridade dos alunos e o número de disciplinas em que estes foram matriculados (considerando aquelas escolhidas na pré-matrícula) encontrando $r = 0,49357$, o qual não difere estatisticamente de zero, ao nível de 0,5% de probabilidade, pelo teste t de Student (Mood et al, 1974).

Conclusão

O trabalho realizado nesta dissertação pode ser dividido em três partes fundamentais:

- 1- Estudo dos principais conceitos da Teoria dos Grafos e, em particular, de coloração em grafos e dos algoritmos disponíveis na literatura para resolvê-los.
- 2- Escolha de um algoritmo de coloração em grafos, desenvolvido com o objetivo de resolver o problema de particionamento de conjuntos em um número fixo de classes tal que o diâmetro inter-classes seja mínimo, para a realização de um estudo mais aprofundado.
- 3- Implementação computacional do algoritmo selecionado e das modificações introduzidas sobre o mesmo, tendo em vista a aplicação deste na resolução de problemas, tais como, a realização da matrícula em uma universidade ou o planejamento das apresentações de artigos em um congresso.

O estudo da Teoria dos Grafos realizado e apresentado nos capítulos 1 e 2, desta dissertação, introduz conceitos para a compreensão de algoritmos de coloração. Os algoritmos descritos nos Capítulos 2 e 3 ressaltam a necessidade e importância de um estudo teórico mais aprofundado para a proposição e desenvolvimento de algoritmos de coloração em grafos.

O algoritmo de Guénoche (1989, 1993), apresentado no capítulo 3, para particionamento de conjuntos em um número p , fixo, de classes, baseia-se na construção e de todas as possíveis colorações de um grafo teto p -colorível. Tendo em vista o caráter NP-completo do problema tratado (Garey e Johnson, 1979), o autor faz uso de heurísticas para a determinação do teto s , usando heurísticas sequenciais de coloração, para a obtenção de uma clique, no grafo teto construído e para a coloração efetuada a seguir. Apesar do uso de algoritmos aproximados, Guénoche (1989, 1993) descreve resultados de boa qualidade, comprovados pelos testes computacionais, com dados gerados aleatoriamente, que realizamos com o programa correspondente ao método proposto. Um procedimento proposto em Guénoche (1989) para otimização do teto na 1ª etapa do algoritmo foi incorporado ao método descrito por Guénoche (1993)

em duas versões: (1) a fase de *otimização do teto* é implementada conforme a técnica descrita em Guénoche (1989); (2) a fase de *otimização do teto* implementada utiliza apenas a fase em que as arestas adjacentes a vértices atribuídos a classes distintas, são inseridas no grafo teto. Esta modificação revelou-se benéfica, uma vez que o acréscimo de tempo computacional é pequeno, o número de partições obtidas na 2ª etapa do algoritmo diminui em certos casos e ao se utilizar esta versão, somente em muito poucos exemplos testados houve a necessidade da 3ª etapa do algoritmo.

O problema da matrícula de alunos ("scholl timetabling") pode ser resolvido como um problema de particionamento de conjuntos em p classes distintas. O programa computacional desenvolvido usando o algoritmo de Guénoche (1989, 1993) para obtenção de todas as possíveis partições das disciplinas um número fixo de horários, forneceu resultados de boa qualidade para dados gerados aleatoriamente. A medida de eficiência da solução obtida é calculada através da média ponderada proposta no capítulo 4. O diâmetro inter-classes a ser minimizado é obtido com base na computação da dissimilaridade entre as disciplinas oferecidas, calculada em função do número de alunos pré-matriculados e da prioridade de cada aluno.

O exemplo resolvido no capítulo 4, envolvendo a matrícula de 48 alunos de 3 cursos em um total de 12 disciplinas oferecidas, forneceu uma medida de eficiência em torno de 83% e exigiu um tempo computacional um pouco maior que 0,1 segundos em um microcomputador IBM-PC 486 DX4-100 MHz.

Os estudos e a implementação computacional realizados nesta dissertação proporcionaram a publicação de um artigo (Santos e Ferreira Ribeiro, 1995) e a redação de dois novos artigos a serem submetidos à apreciação de um congresso nacional e de uma revista nacional. Entre as tarefas a executar, como continuidade do trabalho apresentado nesta dissertação, destacam-se:

- 1 - Reestudo de procedimentos específicos empregados no método de coloração de Guénoche (1993) com o objetivo de substituí-los por rotinas mais eficientes.
- 2 - Realização de testes computacionais para exemplos de grande porte no sentido de avaliar o tempo de cálculo e a utilização de memória.
- 3 - Estudo da aplicação do programa desenvolvido para a resolução de problemas em outras áreas, notadamente: genética das populações, ecologia, e outras.

Referências Bibliográficas

- ABREU, N. M. M. - *Grafos Quocientes e Algumas Aplicações em Problemas Discretos de Otimização*, Niterói, RJ: Instituto de Lógica, Filosofia e Teoria da Ciência - ILTC, 1991. Ata Científica B 015/91
- APPEL, K. I., HAKEN, W. - *The Solution to the Four-color-map Problem*, Scientific American, n. 108, october 1977.
- AHO, A. V., HOPCROFT, J. E., ULMAN, J. D. - *The Design and Analysis of Computer Algorithms*, Add.-Wesley, 1974.
- BERGE, C. - *Graphs and Hypergraphs*, North-Holland Publishing Company, 1976
- BRÉLAZ, D. - *New Methods to Color Vertices of a Graph*, Comm. ACM, v. 22, p. 251-256, 1979.
- BROWN, J. R. - *Chromatic Scheduling and Chromatic Number Problem*, Manag. Science, 19, pp. 456-463, 1972.
- CAMPELO, R. E., MACULAN, N - *Algoritmos e Heurísticas: Desenvolvimento e Avaliação de Performance*, EDUFF, 1994.
- DE WERRA, D. - *An Introduction to Timetabling*, European Journal Of Operational Research - EJOR 19, p. 151-162, 1985.
- DUTTON, R. D., BRIGHAM, R. C. - *A New Graph Colouring Algorithm*, The Computer Journal, v. 24, p. 85-86, 1981.
- FERREIRA RIBEIRO, J. F., PRADIN, B - *A Methodology for Cellular Manufacturing Design*, International Journal of Production Research, v. 31,1, pp. 235-250, 1993.
- FERREIRA RIBEIRO, J. F. e RIBEIRO, C. M. - *Um Algoritmo de Coloração de Grafos para o Projeto de Células de Manufatura*, XIV CILAMCE, p. 898-903, 1993.

- FERREIRA RIBEIRO, J. F. e ALVES, M. R. P. A. - *Projeto de Células de Manufatura por meio de Ligações-Contrações*, XV CILAMCE, p. 1470-1476, 1994.
- GAREY, M. R., JOHNSON, D. S. - *The Complexity of Near-optimal Graph Colouring*, J. of the Association of Computing and Machinery, n. 23, p. 43-49, 1976.
- GAREY, M. R., JOHNSON, D. S. - *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.
- GOLUMBIC, M. C. - *Algorithmic Graph Theory and Perfect Graphs*, Computer Science and Applied Mathematics Series, Academic Press Inc., 1977.
- GONDRAN, M., MINOUX, M. - *Graphs and Algorithms* - John Wiley & Sons, 1990.
- GOUVÊA, E. F. - *Contribuições às Heurísticas do Problema de Coloração em Grafos*, Tese de mestrado do Programa de Engenharia de Sistemas do Instituto Militar de Engenharia - IME, 1993.
- GUÉNOCHE, A. - *Partitions with Minimum Diameter*, 2nd IFCS, Charlotteville, 1989.
- GUÉNOCHE, A. - *Énumération des Partitions de Diamètre Minimum*, Discrete Mathematics, 111, p. 227-287, 1993.
- HÁJOS, G. - *Über eine Klassifikation Nicht n -Färbbarer Graphen*, Martin Luther Univ. Halle-Wittenberg, A 10, p. 116-117, 1961.
- HANSEN, P., DELATTRE, M. - *Complete-link Cluster Analysis by Graph Coloring*, J. Am. Stat. Ass., 73, pp. 397-403, 1978.
- HANSEN, P., JAUMARD, B. - *Minimum Sum of Diameters Clustering*, Journal of Classification, 4, 2, pp. 215-226, 1987.
- HARARY, F. - *Graph Theory*, Addison-Wesley Publishing Company, Inc., 1969.
- HEAWOOD, P. J., *Map-Color Theorems*, Quart. J. Math., v. 24, p. 332-338, 1890.
- HERTZ, A. - *COSINE: A New Graph Coloring Algorithm*, Operations Research Letters, v. 10, p. 411-415, 1991.

HUBERT, L. J. - *Spanning Trees and Aspects of Clustering*, British J. Math. Statist. Psych., 27, pp. 14-28, 1974.

JOHNSON, D. S. - *Worst Case Behavior of Graph Coloring Algorithms*, Proc. 5th s. ed. Conf. Combinatorics, Graph Theory and Computing, p. 513-527, 1974

KARCHMER M. e NAOR, J. A *Fast Parallel Algorithm to Color a Graph with p Colors*, J. of Algorithms, 9, p. 83-91, 1988.

KORMAN, S. M. - *The Graph-coloring Problem*, Em: CHRISTOFIDES et al (ed.) - *Combinatorial Optimization* - John Wiley & Sons, 1979.

LECLERC, B. - *Caractérisation, Construction et Dénombrement des Ultramétriques Supérieures Minimales*, Statistiques et Analyse des Données, 11(2), pp. 26-50, 1986.

LEIGHTON, F. T. - *A Graph Coloring Algorithm for Large Scheduling Problems*, J. Res. Nath. Bur. Standards, v. 84, p. 489-506, 1979.

LEWIS, H. R., PAPADIMITRIOU, C. H. - *L'efficacité des Algorithmes*, Pour la Science, p.114-127.

MOOD, A. M., GRAYBILL, F. A., BOES, D. C. - *Introductions to the Theory of Statistics*, McGraw-Hill, 1974.

NETTO, P. O. B. - *Teoria e Modelos de Grafos* - Ed. Edgard Blucher Ltda, 1979.

OPSUT, R. J. e ROBERTS, F. S. - *On the Fleet Maintenance, Mobile Radio Frequency, Task Assignment and Traffic Phasing Problems*, Em: CHARTRAND, G. et al (ed.) - *The Theory and Applications of Graphs*, John Wiley & Sons, 1981.

RAO, M. R. - *Cluster Analysis and Mathematical Programming*, J. Am. Stat. Ass., 66, pp. 622-626, 1971.

SAKAROVITCH, M. - *Optimization Combinatoire: Graphes et Programmation Linéaire (vol 1) / Programmation Discrète (vol 2)*, Hermann, 1984.

SANTOS, L. M. R., FERREIRA RIBEIRO, J. F. - *Planejamento de um Horário Escolar através de Coloração em Grafos*, XXVII SOBRAPO, 1995.

STOCKMEYER, L. J. - *Planar 3-Colorability is NP-Complete*, Sigact News, p. 19-25, 1973.

SZWARCFITER, J. L. - *Grafos e Algoritmos Computacionais* - Campus, 1993.

WELSH, D. J. A., POWELL, M. B. - *An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems*, Comput. J., n. 10, p. 85-86, 1967.

WITTE, J. - *The Use of Similarity Coefficients in Production Flow Analysis*, International Journal of Production Research, 27,12, p. 2053-2062, 1980.

ZIKOV, A. A. - *On Some Properties of Linear Complexes*, Mat. Sbornik, 24, p. 163-168, 1949; Amer. Math. Soc. Translation, 79, 1952.