UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLóGICO

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS

PROGRAMA DE PóS GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS

PROPOSTA DE UM ALGORITMO HEURÍSTICO PARA OBTENÇÃO DE UMA SOLUÇÃO PARA O MÁXIMO CONJUNTO INDEPENDENTE

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

SALOMÃO WESTPHAL SANDRINI

Florianópolis, 29 de dezembro de 1992.

PROPOSTA DE UM ALGORITMO HEURÍSTICO PARA OBTENÇÃO DE UMA SOLUÇÃO PARA O MÁXIMO CONJUNTO INDEPENDENTE

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

Salomão Westphal Sandrini

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA
ESPECIALIDADE ENGENHARIA DE PRODUÇÃO E SISTEMAS E APROVADA NA SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

Prof. Neri dos Santos, Dr.
Coordenador

BANCA EXAMINADORA:

Prof. Ricardo Miranda Barcia, Ph.D.
Orientador

Prof. Antônio Galvão Novaes, Dr.
Sub-orientador

Prof. Sérgio F. Mayerle, M. Eng.
Sub-orientador

AGRADECIMENTOS

Ao Prof. Ricardo Miranda Barcia pela confiança e pelo apoio recebido ao longo do desenvolvimento deste trabalho.

Ao Prof. Sérgio Fernando Mayerle pela coorientação dedicada e acima de tudo pela paciência, amizade e compreensão.

Ao corpo do Curso de Pós-Graduação em Engenharia de Produção pelos conhecimentos adquiridos.

Ao amigo Adalberto Gassenferth Júnior, pelas dicas heurísticas no início do trabalho.

Ao colega Antônio Carlos da Silva, pela implementação computacional dos algoritmos.

À UFSC - Universidade Federal de Santa Catarina pela oportunidade.

A DEUS pela esperança.

A minha esposa Vera Lúcia e meus filhos, Caroline e Ury, pelo amor, que transformado em força me levaram até ao fim deste trabalho.

Sumário

ABSTRACT

The present work, primaly, realises an exploration between the methods or algorhythmns, made with the expectation being to find, for whatever graph, the independant maximum groups, to thus obtain the best solution of maximum independant sets.

After the research of traditional algorhythmns, concluding in heuristic algorhythmns, efficient with the complexity of n^2 - for entries the size of n - which find a solution for the maximum independant set problem.

Thus also bringing into effect comparitive analysis, related to the computed efficiency, between the best traditionally researched heuristic algorhythmn (Bron and Kerbosch) and the proposed heuristic algorhythmn in this work.

Conclude and work with commentaries about the advantages of utilising the heuristic algorhythmn, beyond the recomendations and indications for new reseraches and studies.

RESUMO

O presente trabalho, a priori, realiza uma exploração dentre os métodos ou algoritmos, desenvolvidos com a finalidade de encontrar, para um grafo qualquer, os conjuntos independentes maximais. Obtem-se, assim, a solução ótima para o problema do máximo conjunto independente (número de independência).

Após a pesquisa dos algoritmos tradicionais, desenvolve-se um algoritmo heurístico, eficiente, com complexidade n^2 - para entradas de tamanho n - que encontra uma boa solução para o problema do máximo conjunto independente.

Efetua-se, também, uma análise comparativa, relacionada ao desempenho computacional, entre o melhor algoritmo tradicional pesquisado (Bron e Kerbosch) e o algoritmo heurístico proposto neste tabalho.

Conclui-se o trabalho com comentários sobre vantagens da utilização do algoritmo heurístico, além de recomendações e indicações para novas pequisas e estudos.

LISTA DE FIGURAS

TABELA

LISTA DE GRÁFICOS

CAPÍTULO I

INTRODUÇÃO

1.1. - Histórico [71]

O motivo que se constituiu na origem da teoria de grafos é um problema algorítmico, já que a solução é obtida atráves da elaboração de um algoritmo. Este problema, conhecido como *problema das pontes de Konigsberg*, foi primeiramente resolvido por Euler em 1736. No rio Pregel, junto à cidade de Königsberg na então Prússia, existiam duas ilhas formando portanto quatro regiões distinguíveis de terra, margem direita, margem esquerda e duas ilhas separadas, tanto uma da outra quanto das margens. Havia um total de sete pontes interligando-as. O problema consiste em, partindo de uma dessas regiões, determinar um trajeto pelas pontes segundo o qual se possa retornar à região de partida, atravessando cada uma das pontes somente uma vez. Euler mostrou que não existe tal trajeto, ao elaborar um método algorítmico utilizando um modelo em grafos para generalização deste problema. Através desse modelo ele verificou que existe tal trajeto se e somente se cada região for ligada por um número par de pontes.

A solução deste problema é considerada como o marco inicial da teoria de grafos. Porém, desde seu aparecimento, há mais de 200 anos, muito pouco foi acrescentado, nos anos seguintes, ao trabalho de Euler.

Somente, em meados do século XIX, é que três desenvolvimentos isolados viriam contribuir para despertar o interesse pelo assunto.

O primeiro desses fatos é a formulação do *problema das quatro cores*, cuja autoria supõe-se ser de Francis Guthrie. O problema consiste em colorir os países de um mapa arbitrário plano, cada país com uma cor, de tal forma que países fronteiriços possuam cores diferentes. Será possível então obter tal coloração usando não mais de 4 cores? Esta conjectura permaneceu em aberto até 1977, quando foi provada por Appel e Haken. Além da importância do tópico de coloração, o problema das quatro cores desempenhou um papel muito relevante para o desenvolvimento geral da teoria dos grafos.

Outro desenvolvimento importante foi a formulação do *problema do caminho Hamiltoniano*, por Hamilton. No caminho Hamiltoniano existem n cidades, cada par de cidades pode ser adjacente ou não. Partindo de uma cidade qualquer, o problema consiste em determinar um trajeto que passe apenas uma vez em cada cidade e retorne ao ponto de partida. Até a data atual, não foi encontrada uma solução algorítmica satisfatória, ou seja, não são conhecidas condições necessárias e suficientes, razoáveis, de existência de tais trajetos.

E o terceiro acontecimento marcante do século XIX foi o desenvolvimento da *teoria das árvores*, realizado, inicialmente, por Kirchoff e Cayley. O primeiro visava a sua aplicação em circuitos elétricos, enquanto o último a empregava em química orgânica. As árvores constituem uma classe especial de grafos, com larga aplicação nas demais diferentes áreas.

No século XX o interesse por grafos aumentou. Por volta da década de 1930, resultados fundamentais na teoria foram obtidos por Kuratowski, Konig, Menger e outros. Atualmente, tem-se, a idéia de ser, a teoria de grafos, uma área pouco explorada.

1.2. - Origem

Dado um grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$, muitas vezes o interesse está em descobrir subconjuntos do conjunto $\mathbb{X}$ dos vértices que formam $\mathbb{G}$ e que possuam alguma propriedade pré-definida. Por exemplo, qual a máxima cardinalidade para que, um determinado subconjunto $\mathbb{V} \subseteq \mathbb{X}$, forme um subgrafo $\langle \mathbb{V} \rangle$ completo? Ou qual será a máxima cardinalidade para que o subconjunto $\mathbb{V}$, gere um subgrafo $\langle \mathbb{V} \rangle$ inteiramente disconecto? A resposta para a primeira questão é conhecida como *número facção* (*clique*) de $\mathbb{G}$ e a resposta para a segunda, como o *número de independência*. Estes números e os subconjuntos de vértices dos quais são derivados, descrevem importantes propriedades estruturais do grafo, e há uma variedade de aplicações diretas tais como: planejamento de projetos, análise de grupos, processamento paralelo em computadores, locação e instalação de componentes elétricos, geração de horários, coloração, associação ("*matching*"), etc.

Neste trabalho, a atenção estará voltado para o máximo conjunto independente, que é obtido por seleção do maior conjunto pertencente à família dos conjuntos independentes maximais.

Para obter todos os conjuntos independentes maximais de um grafo $\mathbb{G}$, existem alguns métodos, dentre os quais: o método da

enumeração sistemática, o método de Maghout (algébrico), e também o método de Bednarek e Taulbee que trabalha com as facções do grafo $\mathbb{G}$. Com o aparecimento do método enumerativo sistemático de Bron e Kerbosch, houve um avanço computacional significativo. Contudo, quanto maior for o número de vértices do grafo $\mathbb{G}$, maior será o número de conjuntos independentes maximais que terão de ser encontrados, o que certamente exigirá muito tempo de processamento e uma grande quantidade de memória.

Embora encontrem todos os conjuntos independentes maximais do grafo, esses métodos servem tão somente a propósitos teóricos, pois quanto maior for o número de vértices de um determinado grafo $\mathbb{G}$, maior será a quantidade de memória necessária e mais tempo de processamento será gasto, o que conseqüentemente, os torna inviáveis computacionalmente. A finalidade destes métodos é, portanto, encontrar os conjuntos independentes maximais para grafos pequenos de até 50 vértices.

Por isso, surgiu a idéia de criar um algoritmo heurístico eficiente, com o objetivo de gerar uma boa solução para o problema da determinação do máximo conjunto independente e, conseqüentemente, do número de independência $\alpha[\mathbb{G}]$. Assim, o principal objetivo que este algoritmo deveria perseguir seria o de obter uma solução sem ter de encontrar e comparar todos os conjuntos independentes maximais do grafo e, além disso, utilizar pouca quantidade de memória com baixo tempo de processamento computacional.

1.3. - Objetivos

O objetivo central deste trabalho é propor um algoritmo heurístico, independente do tamanho do grafo, que otimize o processo de obtenção do número de independência $\alpha[\mathbb{G}]$ e, de preferência, possua complexidade polinomial. Há também uma preocupação em reduzir o tempo e a quantidade de memória requerida no processamento computacional.

O trabalho também realiza um estudo comparativo entre dois algoritmos completamente distintos. O primeiro algoritmo, escolhido dentre os vários que existem atualmente na literatura, é considerado como sendo o melhor dentre os que selecionam a solução ótima para o problema da determinação do máximo conjunto independente. Este algoritmo encontra a solução ótima para o problema de obtenção do número de independência $\alpha[\mathbb{G}]$ de um grafo $\mathbb{G}$ qualquer, tendo que, para isso, gerar e comparar todos os conjuntos independentes maximais. Já o segundo algoritmo que é o proposto neste trabalho, tem como objetivo apresentar somente um conjunto, sem necessidade de gerar todos os conjuntos independentes maximais, nem de compará-los entre si.

1.4. - Importância e limitações

A importância dos conjuntos independentes maximais de um grafo $\mathbb{G}$ qualquer reside, não apenas no fato do grande número de aplicações diretas, mas também porque aparece muitas vezes como um subproblema em outras áreas abrangidas pela teoria de grafos.

Em particular, desempenha um papel muito importante na determinação do número cromático de um grafo, em problemas de associação ("*matching*"), na otimização da geração de horários (professores, disciplinas, salas, cursos, etc.) e em muitos outros. No capítulo III serão apresentadas algumas aplicações relacionadas com conjuntos independentes maximais e com a determinação do número de independência. Como o interesse principal em relação a propriedade da independência maximal é obter o número de independência $\alpha[G]$ e o conjunto independente maximal por ele gerado, o objetivo é, obviamente, encontrá-los. Contudo ao se tentar encontrar os conjuntos independentes maximais, surgem com alguns problemas, dos quais destacam-se:

(*i*) **Tratamento teórico**

Apresenta-se, a seguir um breve resumo do principal problema teórico encontrado em cada um dos métodos mencionados.

Método da enumeração sistemática

Tem-se que verificar a cada conjunto gerado se este conjunto não esta contido em um conjunto anteriormente gerado, para saber se é maximal ou não. Além disso, tem que efetuar todas as possíveis combinações.

Método de Maghout

A condição de independência é formulada utilizando a álgebra booleana. Efetua operações e reduções algébricas e, utiliza uma função que gera as combinações que fornecem todos os conjuntos independentes maximais do grafo.

Método de Bednareck e Toulbee

Trabalha com subgrafos, escolhendo um vértice p qualquer, para formar subgrafos que contenham p e subgrafos que não contenham. Pode-se também, aumentar a cada etapa, o tamanho do subgrafo.

Algoritmo de Bron e Kerbosch

Guarda subconjuntos auxiliares $\mathbb{E}_k^-$ e $\mathbb{E}_k^+$, que dependem do conjunto independente em formação, efetua testes heurísticos e utiliza o recurso *"backtrak"*.

(*ii*) **Complexidade dos algoritmos disponíveis**

Todos os algoritmos apresentados possuem complexidade exponencial $\Theta(\exp\ m)$, ou seja o tempo de execução cresce exponencialmente a medida que aumenta-se o tamanho do grafo;

Memória requerida para armazenamento muito grande.

Por isso a importância do algoritmo heurístico proposto, deve-se ao fato de possuir, dentre outras vantagens, as seguintes:

(*i*) Heurística eficiente;

(*ii*) Complexidade da ordem de n^2, ou seja, $\Theta(n^2)$;

(*iii*) Encontra uma solução para grafos de grande porte;

(*iv*) Apesar da não garantia da solução ótima garante uma boa solução, que eventualmente serve, como um limite inferior para a obtenção da solução ótima, ao utilizar-se o algoritmo de Bron e Kerbosch.

1.5. - Organização do trabalho

Este trabalho está dividido em sete capítulos.

O capítulo I consiste de uma apresentação preliminar da dissertação, constando de um histórico sobre a teoria de grafos; origem, objetivos, importância e organização do trabalho.

No capítulo II elabora-se um resumo dos conceitos e definições básicas utilizados em teoria de grafos introduzindo também o conceito de algoritmo e complexidade. Apresenta-se também exemplos ilustrativos dos conceitos apresentados.

O capítulo III introduz, para um determinado grafo $\mathbb{G}$, a noção de conjuntos independentes. Define conjuntos independentes maximais, número de independência, máximo conjunto independente e subgrafos completos maximais (facções), com a apresentação de alguns exemplos modulados em forma de grafos. A solução para cada um desses exemplos está relacionada a propriedade de independência maximal de um ou de alguns conjuntos de $\mathbb{G}$. Mostra-se, ainda, uma aplicação da propriedade de independência aos grafos orientados. Finaliza conceituando problemas NP com apresentação de alguns exemplos.

No capítulo IV mostra-se como obter, num determinado grafo $\mathbb{G}$, todos os conjuntos independentes maximais. Apresenta um a um os métodos ou algoritmos que determinam esses conjuntos. Elabora-se, em seguida, um algoritmo com base no algoritmo enumerativo sistemático desenvolvido por Bron e Kerbosch, com a sua descrição formal e o desenvolvimento deste algoritmo aplicado a um exemplo, executado passo a passo.

O capítulo V aponta os problemas do algoritmo de Bron e

Kerbosch, enfoca o problema da determinação do máximo conjunto independente, apresentando métodos para encontrar-se este conjunto. Descreve, em seguida, a base conceitual do algoritmo proposto para a obtenção de uma solução para o máximo conjunto independente, com as respectivas explicações sobre a heurística básica elaborada. Desenvolve-se, também, exemplos didáticos com a utilização do algoritmo heurístico proposto, executado, passo por passo. Finaliza com uma análise da complexidade do algoritmo heurístico proposto.

O capítulo VI mostra os testes realizados, bem como as vantagens do algoritmo heurístico proposto sobre o de Bron e Kerbosch. Caracteriza a metodologia utilizada enfocando os exemplos, comentando como estes foram gerados, como foram feitos os testes, que resultados foram obtidos, como foram comparados e quais as limitações de tempo. Nos resultados obtidos mostram-se os exemplos testados e os gráficos comparativos em relação à convergência do algoritmo heurístico proposto, com apresentação dos contra-exemplos encontrados.

O capítulo VII conclui o trabalho e faz recomendações quanto ao algoritmo heurístico proposto, indicando novas pesquisas.

A bibliografia citada pode ser considerada como um levantamento completo do que foi publicado no país sobre teoria de grafos, algoritmos e complexidade.

CAPÍTULO II

INTRODUÇÃO À GRAFOS

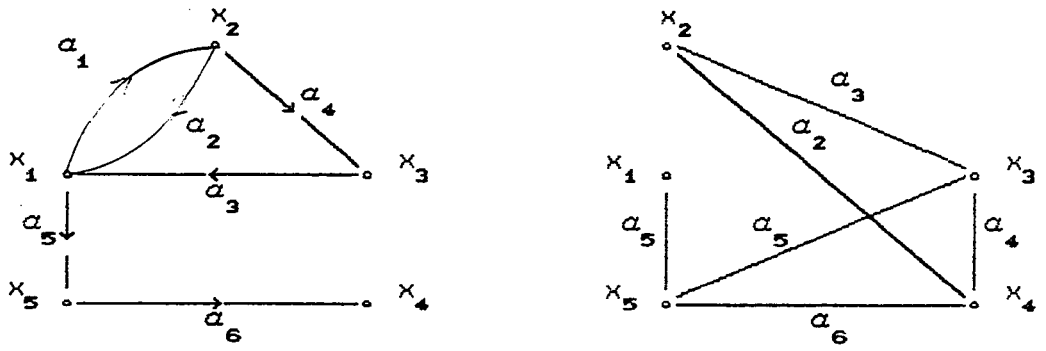
2.1 - Definições básicas

2.1.1. - Grafo

Um *grafo* $\mathbb{G}$ é uma coleção de *pontos*, *nós* ou *vértices*, x_1, x_2, x_3, ..., x_n (denotado pelo conjunto $\mathbb{X}$), e uma coleção de *linhas* a_1, a_2, a_3, ..., a_m (denotado pelo conjunto $\mathbb{A}$), que ligam todos ou alguns destes pontos. O grafo $\mathbb{G}$ será denotado pelo par $(\mathbb{X}, \mathbb{A})$.

2.1.2. - Grafo direcionado e grafo não direcionado

Seja $\mathbb{G} = (\mathbb{X}, \mathbb{A})$ um grafo qualquer. Se as linhas em $\mathbb{A}$ possuem uma direção, que são indicadas por uma flecha, serão chamadas *arcos* e o grafo resultante será chamado grafo *direcionado*, Fig. 2.1.(a). Se as linhas não possuem orientação serão chamadas de *arestas* e o grafo será *não-direcionado*, Fig. 2.1.(b).



(a) Grafo direcionado (b) Grafo não direcionado

Figura 2.1. Grafos

2.1.3. - Direção

Quando um arco $\alpha = (x_i, x_j)$, pertencente a um grafo direcionado, estiver denotado pelo seu par de vértices, *inicial* e *final* (isto é, pelos seus dois vértices *terminais* x_i e x_j), pode-se tomar sua direção como sendo do primeiro vértice para o segundo (ou seja, de x_i para x_j). Deste modo na Fig. 2.1.(a), (x_1, x_2) refere-se ao arco α_1 e, (x_2, x_1) ao arco α_2.

2.1.4. - Adjacência

Seja $\alpha = (x_i, x_j)$ uma aresta de um grafo $\mathbb{G}$ qualquer. Então diz-se que α liga os vértices x_i e x_j, e estes vértices são ditos adjacentes, ou seja, se existe uma aresta ligando dois vértices quaisquer do grafo $\mathbb{G}$ eles são adjacentes. Quando o grafo for direcionado e $\alpha = (x_i, x_j)$, representar um arco, então x_j será adjacente a x_i, o que não quer dizer que x_j seja adjacente a x_i.

2.1.5. - A relação sucessores - $\Gamma(\cdot)$

Uma alternativa, e muitas vezes o melhor meio para descrever um grafo $\mathbb{G}$, é pela especificação do conjunto de vértices $\mathbb{X}$ e por uma *relação* Γ (*sucessores*) que mostra como cada um dos vértices esta relacionado com todos os outros vértices do grafo. Tem-se:

Nota: A relação sucessores Γ é chamada um mapeamento do conjunto $\mathbb{X}$ em $\mathbb{X}$ e o grafo poderá também ser denotado pelo par $(\mathbb{X}, \Gamma)$

$$\Gamma(x_i) = \{x_j \in \mathbb{X} / \exists\ \alpha = (x_i, x_j) \in \mathbb{A}\} = \{x_j \in \mathbb{G} / \ x_j \text{ é adjac. a } x_i\}$$

No exemplo da Fig. 2.1(a) temos $\Gamma(x_1) = (x_2, x_5)$ ou seja, x_2 e x_5 são os vértices finais dos arcos cujo vértice inicial é x_1. Outros exemplos para o grafo da Fig. 2.1.(a):

$\Gamma(x_4) = \emptyset$ (conjunto vazio) $\qquad\qquad$ $\Gamma(x_3) = (x_1)$

E para a Fig. 2.1.(b):

$\Gamma(x_5) = (x_1, x_3, x_4)$ $\qquad\qquad$ $\Gamma(x_1) = (x_5)$

Obs:

No caso de um grafo não-direcionado, ou de um grafo contendo arcos e arestas (tais como os grafos desenhados nas Fig. 2.1.(b) e 2.2), as relações Γ serão tomadas conforme as de um grafo direcionado equivalente, no qual cada aresta será substituída por dois arcos em direções opostas.

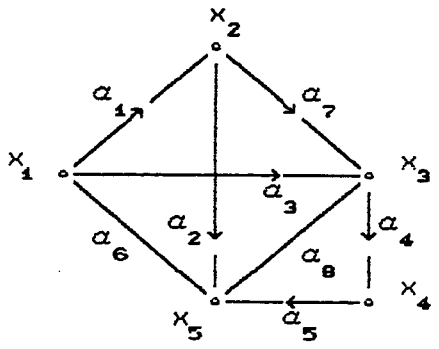


Figura 2.2. Grafo misto

Desta forma, para o grafo da figura 2.2, por exemplo:

$\Gamma(x_1) = (x_2, x_3, x_5)$ $\qquad$ e $\qquad$ $\Gamma(x_5) = (x_1, x_3)$

2.1.6. - A relação antecessores - $\Gamma^{-1}(\cdot)$ (inversa sucessores)

Sabe-se que $\Gamma(x_i)$ é o conjunto de vértices $x_j \in \mathbb{X}$ para os quais um arco (x_i, x_j) existe em $\mathbb{G}$. Assim, é natural escrever $\Gamma^{-1}(x_i)$ como sendo o conjunto dos vértices x_k para os quais exista um arco $\alpha = (x_k, x_i)$ em $\mathbb{G}$. A relação $\Gamma^{-1}(x_i)$ será por isso chamada *relação inversa* e será dada por:

$$\Gamma^{-1}(x_i) = \{x_k \in \mathbb{X} / \exists\ (x_k, x_i) \in \mathbb{A}\} = \{x_k \in \mathbb{G} / \ x_i \text{ é adjac. a } x_k\}$$

Deste modo, para o grafo da Fig. 2.1(a), temos:

$$\Gamma^{-1}(x_1) = \{x_2, x_3\} \qquad e \qquad \Gamma^{-1}(x_2) = \{x_1\}$$

e para o grafo da Fig 2.2,

$$\Gamma^{-1}(x_5) = \{x_1, x_2, x_3, x_4\}$$

Nota: é óbvio que num grafo não-direcionado $\Gamma(x_i) = \Gamma^{-1}(x_i)$ para todo $x_i \in \mathbb{X}$.

2.1.7. - Relação Γ aplicada à um conjunto

Quando a relação Γ não operar sobre um único vértice mas sobre um conjunto de vértices tal como $\mathbb{V} = \{x_1, x_3, \ldots, x_q\}$, então $\Gamma(\mathbb{V})$ será descrita por:

$$\Gamma(\mathbb{V}) = \Gamma(x_1) \cup \Gamma(x_3) \cup \ldots \cup \Gamma(x_q)$$

isto é, $\Gamma(\mathbb{V})$ é um subconjunto de vértices $x_j \in \mathbb{X}$ (ou seja, $\Gamma(\mathbb{V}) \subseteq \mathbb{X}$) para os quais existe ao menos um arco (aresta) $\alpha = (x_i, x_j)$ em $\mathbb{A}$, para algum $x_i \in \mathbb{V}$. Deste modo para o grafo da Fig. 2.1.(b) e com $\mathbb{V} = \{x_1, x_2, x_5\}$, tem-se:

$$\Gamma(\mathbb{V}) = \Gamma(\{x_1, x_2, x_5\}) = \Gamma(x_1) \cup \Gamma(x_2) \cup \Gamma(x_5) =$$
$$= \{x_5\} \cup \{x_3, x_4\} \cup \{x_1, x_3, x_4\} = \{x_1, x_3, x_4, x_5\}$$

Nota: $\Gamma(\mathbb{V}) = \{x_j \in \mathbb{V} / \exists\ \alpha = (x_i, x_j) \in \mathbb{A}$, para algum $x_i \in \mathbb{V}\}$

2.2 - Graus de um vértice

2.2.1. - Grau-de-entrada e grau-de-saída de um vértice

O número de arcos que tenham o vértice x_i como seu vértice inicial será definido como o *grau-de-saída do* vértice x_i. Similarmente, o número de arcos que tenham x_i como seu vértice final será chamado *grau-de-entrada*.

Deste modo, referindo-se a Fig. 2.2, o grau-de-saída de x_5, denotado por $g_S(x_5) = |\Gamma(x_5)| = 2$, e o grau-de-entrada de x_5, denotado por $g_E(x_5) = |\Gamma^{-1}(x_5)| = 4$.

Nota: $\sum_{i=1}^{n} g_S(x_i) = \sum_{i=1}^{n} g_E(x_i) = m$

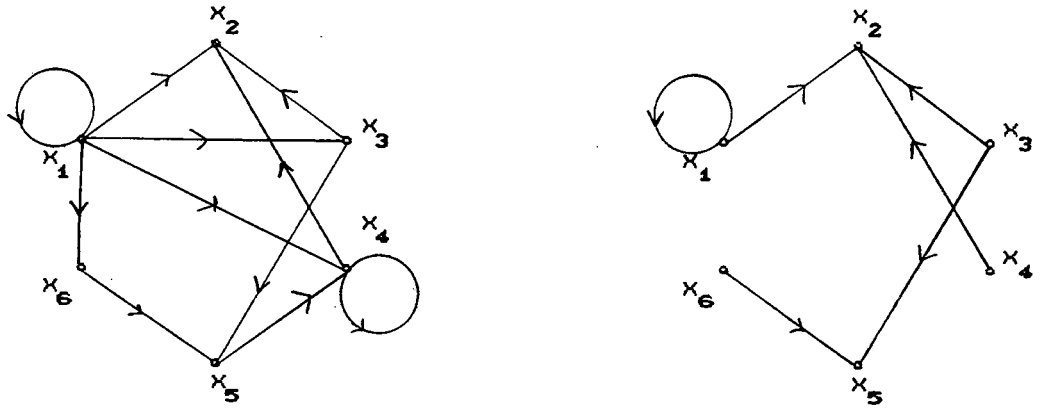
2.2.2. - Grau dos vértices de um grafo não-direcionado

Para um grafo não-direcionado $\mathbb{G} = (\mathbb{X}, \Gamma)$, o grau de um vértice x_i é definido por $g(x_i) = |\Gamma(x_i)|$.

2.3. - Grafos parciais, subgrafos e subgrafos parciais

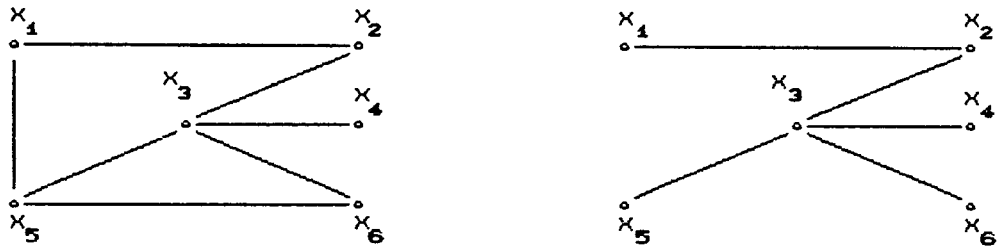
2.3.1. - Grafo parcial

Dado um grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$, um grafo *parcial* de $\mathbb{G}$ é o grafo $\mathbb{G}_p = (\mathbb{X}, \mathbb{A}_p)$ com $\mathbb{A}_p \subset \mathbb{A}$. Desta forma, um grafo parcial é um grafo com o mesmo número de vértices, mas apenas com um subconjunto dos arcos (arestas) do grafo original.



(i) Grafo (ii) Grafo parcial

(a) Grafo parcial direcionado



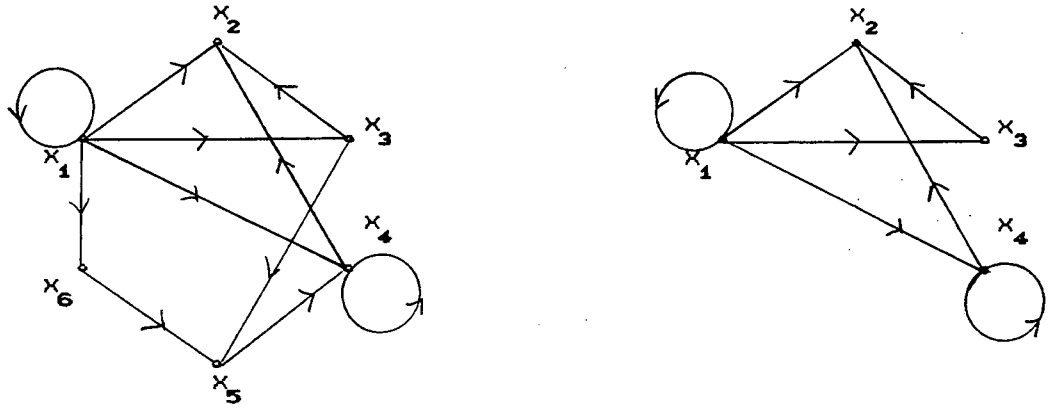
(i) Grafo (ii) Grafo parcial

(b) Grafo parcial não-direcionado

Figura 2.3. Grafo parcial

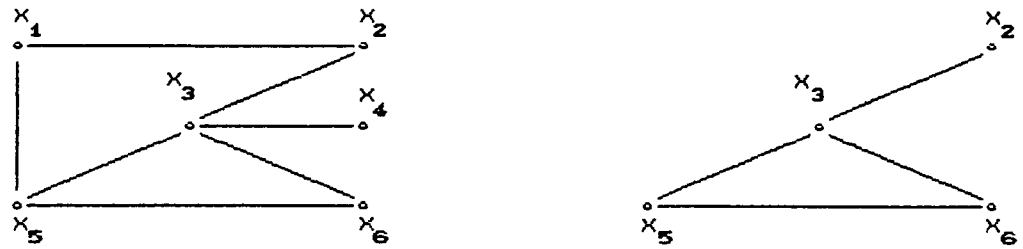
2.3.2. - Subgrafo

Dado um grafo $\mathbb{G} = (\mathbb{X}, \Gamma)$ um *subgrafo* é o grafo $\mathbb{G}_s = (\mathbb{X}_s, \Gamma_s)$ com $\mathbb{X}_s \subset \mathbb{X}$ tal que para todo $x_i \in \mathbb{X}_s$ tem-se $\Gamma_s(x_i) = \Gamma(x_i) \cap \mathbb{X}_s$. Portanto, um subgrafo possui somente um subconjunto $\mathbb{X}_s$ do conjunto de vértices $\mathbb{X}$ do grafo original $\mathbb{G}$, o qual contém todos os arcos (arestas) cujos vértices inicial e final estejam ambos dentro deste conjunto. Subgrafo é o grafo induzido por seus vértices. Denota-se o subgrafo $\mathbb{G}_s$ por $(\mathbb{X}_s)$.



(i) Grafo (ii) Subgrafo

(a) Subgrafo direcionado



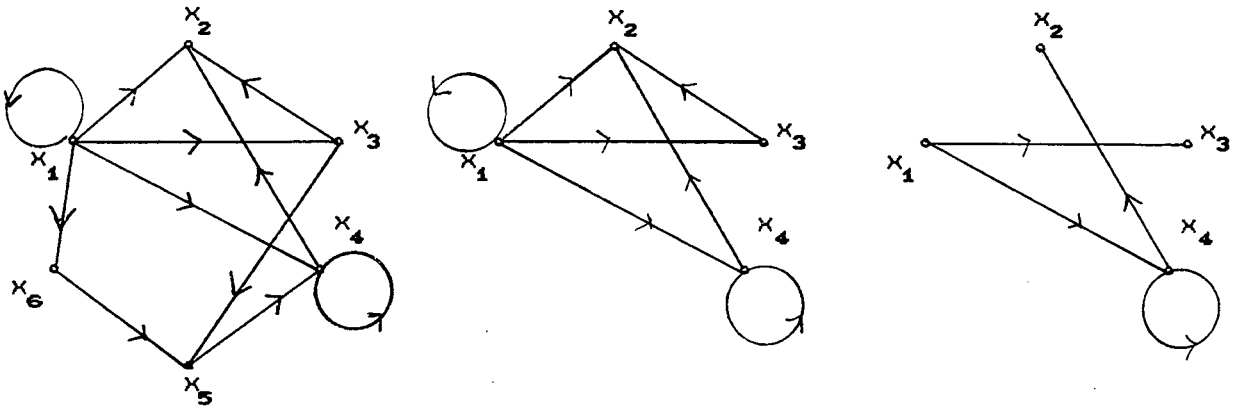
(i) Grafo (ii) Subgrafo

(b) Subgrafo não-direcionado

Figura 2.4. Subgrafo

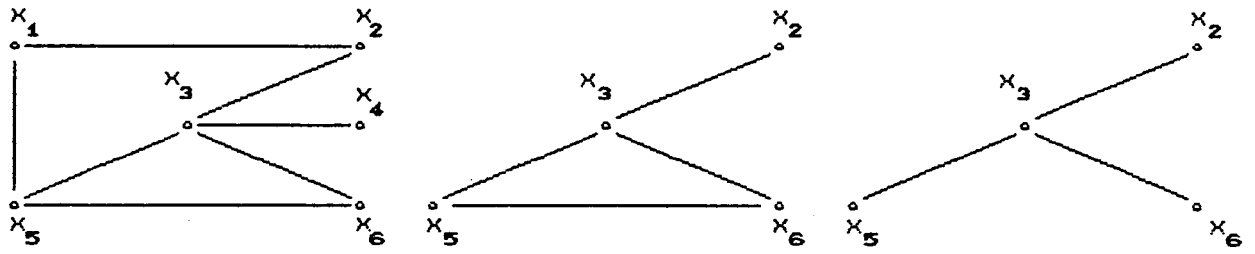
2.3.3. - Subgrafo parcial

As duas definições acima podem ser combinadas para definir *subgrafo parcial*, que é um grafo parcial de um subgrafo dado.



(i) Grafo (ii) Subgrafo (iii) Subgrafo parcial

(a) Subgrafo parcial direcionado



(i) Grafo (ii) Subgrafo (iii) Subgrafo parcial

(b) Subgrafo parcial não-direcionado

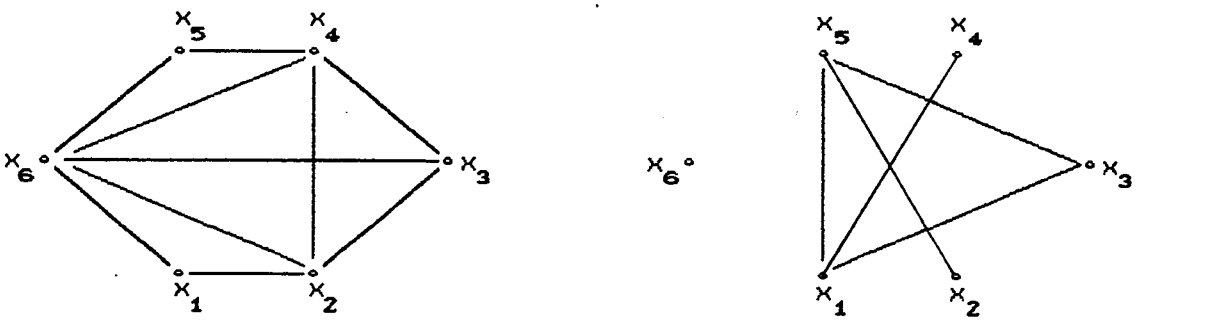
Figura 2.5. Subgrafo parcial

2.4 - Grafo complementar e grafo completo

2.4.1. - Grafo complementar

Denomina-se *complementar* de um grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$ ao grafo $\overline{\mathbb{G}}$, que possui o mesmo conjunto de vértices de $\mathbb{G}$ e que para todo par

de vértices distintos x_i, $x_j \in \mathbb{X}$, tem-se que (x_i, x_j) é arco (aresta) de $\overline{\mathbb{G}}$, se e somente se, não o for de $\mathbb{G}$. Ver Figura 2.7.



(a) Grafo $\mathbb{G}$ (b) $\overline{\mathbb{G}}$ - Complementar do Grafo $\mathbb{G}$

Figura 2.6. Grafo $\mathbb{G}$ e seu complementar $\overline{\mathbb{G}}$

Nota: $\overline{\mathbb{G}} = (\mathbb{X}, \overline{\mathbb{A}})$ onde $\overline{\mathbb{A}} = \{\overline{a} = \overline{(x_i, x_j)} \in \overline{\mathbb{G}} / a = (x_i, x_j) \notin \mathbb{A}\}$

2.4.2. - Grafo completo $\mathbb{K}_n$

Um grafo é *completo* quando:

(i) existe uma arco (aresta) entre cada par de seus vértices; ou

(ii) um grafo no qual todo par de vértices distintos são adjacentes.

O grafo não direcionado completo com n vértices e $\frac{1}{2}$ n(n - 1) arestas é denotado por $\mathbb{K}_n$ ou n(n - 1) arcos para grafos direcionados (figura 2.8).

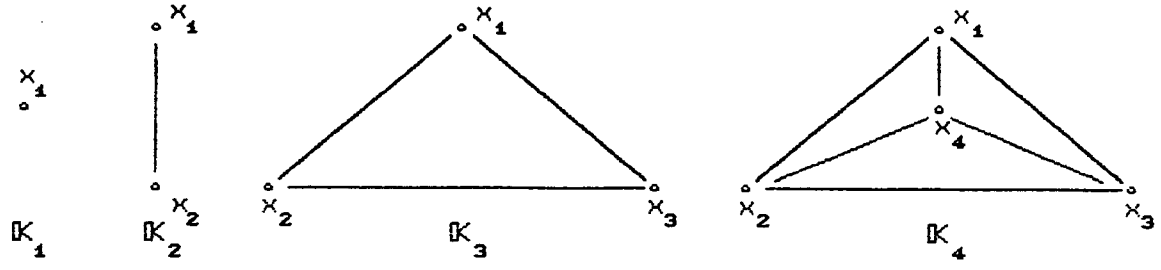
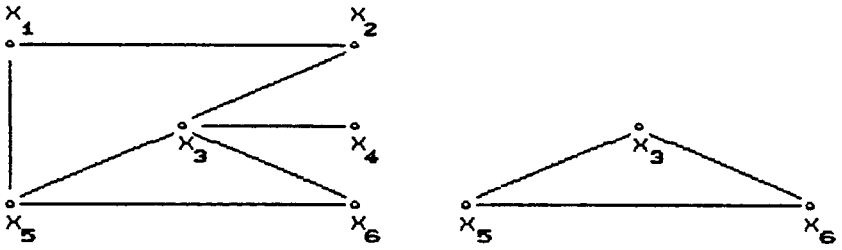


Figura 2.7. Grafos completos.

2.5. - Facção (Subgrafo completo maximal)

Uma facção num grafo $\mathbb{G}$ é um subgrafo completo maximal - maximal no sentido de que não está contida em nenhum subgrafo completo maior. (i. é, ∄ nenhum subgrafo completo que o contenha)



(a) Grafo (b) Facção

Figura 2.8. Facção de um grafo $\mathbb{G}$ não direcionado

2.6. Representação Matricial

Um modo conveniente de representar algebricamente um grafo é através do uso de matrizes conforme segue.

2.6.1. - A matriz de adjacência

Dado um grafo $\mathbb{G}$, sua matriz de *adjacência* $\mathbf{A} = [a_{ij}]_{n \times n}$, será determinada conforme:

$$a_{ij} = \begin{cases} 1 \text{ se o arco } (x_i, x_j) \underline{\text{existir}} \text{ em } \mathbb{G} \\ 0 \text{ caso contrário} \end{cases}$$

Assim, a matriz de adjacência do grafo mostrado na Fig. 2.9 é:

$$A = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{matrix}$$

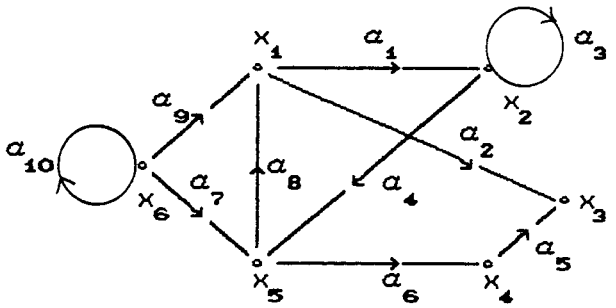


Figura 2.9. Grafo $\mathbb{G}$ para exemplo das matrizes de adjacência e de incidência

A matriz de adjacência define completamente a estrutura do grafo. Por exemplo, a soma de todos os elementos na linha do x_i da matriz fornece o grau de saída do vértice x_i, ou seja $|\Gamma(x_i)|$, e a soma dos elementos na coluna do x_i fornece o grau de entrada do vértice x_i, isto é $|\Gamma^{-1}(x_i)|$.

2.6.2. - A matriz de incidência

Dado um grafo $\mathbb{G}$ direcionado de n vértices e m arcos, a *matriz de incidência* de $\mathbb{G}$, $\mathbf{B} = [b_{ij}]_{nxm}$, será definida segundo:

$$b_{ij} = \begin{cases} 1 & \text{se } x_i \text{ é o } \underline{\text{vértice inicial}} \text{ do arco } a_j \\ -1 & \text{se } x_i \text{ é o } \underline{\text{vértice final}} \text{ do arco } a_j \end{cases}$$

e $b_{ij} = 0$ se x_i não for um vértice terminal do arco a_j ou se a_j for um laço.

Para o grafo mostrado na Fig. 2.9, a matriz incidência é:

$$B = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{matrix}$$

Visto que os vértices x_i e x_j pertencentes a um arco $a = (x_i, x_j)$ são adjacentes, toda coluna da matriz de incidência conterá um 1 e um -1, exceto quando o arco formarum laço no qual teremos apenas zeros.

Nota: Se $\mathbb{G}$ é um grafo não direcionado, então a matriz incidência é definida conforme visto acima, exceto que todos os -1 serão neste caso trocados por 1.

2.7. - Algoritmos e Complexidade de algoritmos [71]

Um elemento primordial em teoria de grafos é o *algoritmo*. Ele pode ser associado ao desenvolvimento de uma técnica para resolver um desejado problema. Os dados do problema constituem a *entrada* (*E*) ou *dados* do algoritmo. A solução do problema corresponde a sua *saída* (*S*). Um algoritmo poderia ser então caracterizado por uma função f, que associa uma saída $S = f(E)$, a cada entrada E. Assim sendo, considera-se a entrada como uma variável independente básica, em relação a qual são produzidas as saídas do algoritmo, bem como são analisados os comportamentos de tempo e espaço do mesmo.

De um modo geral, o interesse pelo algoritmo é inerente ao estudo do problema. Contudo, este conceito, somente foi fundamentado nas primeiras décadas do século corrente, através da formalização de uma computação. Isto possibilitou demonstrar a existência de problemas algorítmicos que não podem ser resolvidos. Com isso abriu-se um novo campo de interesses, que consiste em identificar e classificar os problemas, segundo a existência ou não de algoritmos para resolvê-los.

Desenvolveu-se, então um enorme número de aplicações de interesse prático que dependem de resultados reais obtidos através desses algoritmos. Como conseqüência natural, deixou de ser o bastante assinalar apenas a existência ou não de algum algoritmo que resolva um dado problema. Tornou-se necessário também impor que o tempo e espaço consumidos pela máquina para a implementação do algoritmo estejam dentro do limite da prática. Por exemplo, resolver o problema do caminho hamiltoniano segundo o critério das permutações, para um valor $n = 20$ é, do ponto de vista da aplicação, como se esse algoritmo talvez não existisse.

De início, os critérios de medida de eficiência eram em geral empíricos, principalmente no que se refere à eficiência de tempo. Baseado em uma certa estratégia, um algoritmo era descrito e implementado. Em seguida, efetuava-se uma avaliação prática de seu comportamento. Isto é, um programa implementando o algoritmo era executado em um computador, para alguns conjuntos de dados diferentes. Para cada execução media-se o tempo correspondente. Ao final da experiência eram obtidas, algumas curvas destinadas a avaliar o comportamento do algoritmo.

Em geral, esses eram os critérios utilizados até

aproximadamente a primeira metade da década de 60. Se bem que as informações sejam também úteis, os critérios mencionados não permitem medir, realmente, o comportamento do algoritmo. E por vários motivos. As curvas obtidas traduzem apenas os resultados de medidas empíricas para dados particulares. Além disso, essas medidas são dependentes de uma implementação particular. Estes fatos justificam a necessidade da adoção de algum processo que seja analítico, para avaliação da eficiência.

Ao invês de escolher uma máquina particular, em relação a qual a eficiência dos algoritmos seria avaliada, é certamente mais conveniente utilizar-se um modelo matemático de um computador. Uma possível formulação desse modelo é a RAM (*random acess machine*). Este é composto de uma *unidade de entrada*, *unidade de saída*, *memória e controle/processador*.

O funcionamento de uma RAM é semelhante ao funcionamento de um computador hipotético elementar. O processador dipõe de *intruções* que podem ser executadas. A memória armazena os *dados* e o *programa*. Este último consiste de um conjunto de intruções que implementa o algoritmo. Cada instrução **I** do modelo possui um *tempo de execução* **t(I)**. Assim sendo, se para a execução de um programa **P**, para certa entrada fixa, são processadas r_1 instruções do tipo I_1, r_2 instruções do tipo I_2, . . ., r_m intruções do tipo I_m, então o *tempo de execução* do programa **P** é dado por $\sum_{j=1}^{m} r_j \cdot t(I_j)$.

Introduz-se a seguinte simplificação. Supõe-se que **t(I) = 1** para toda instrução **I**, isto é, que o tempo de execução de cada instrução seja constante e igual a 1. Com **t(I) = 1**, o valor do tempo de execução de um programa torna-se igual ao número total

de instruções computadas. Denomina-se *passo de um algoritmo* α à computação de uma instrução do programa **P** que o implementa.

A *complexidade local* do algoritmo α é o número total de passos necessários para a computação completa de **P**, para uma certa entada E. Assim sendo, considerando as simplificações introduzidas, a complexidade local do α é equivalente ao seu tempo de execução, para a entrada E. O interesse é determinar o número total de passos acima, para entradas consideradas suficientemente grandes.

Nesse sentido, é importante avaliar o tamanho de entrada. Supondo que esta seja composta de n símbolos, o seu comprimento seria o somatório dos tamanhos das codificações correspondentes aos n símbolos, segundo o critério de codificação utilizado. Para simplificar a análise, o tamanho da codificação de cada símbolo será considerado constante. Por isso, o tamanho da entrada pode ser expresso por um número proporcional a n, sendo considerado grande quando n o for.

A avaliação da eficiência para problemas grandes é de certa forma a mais importante. Além disso, facilita a manipulação de sua expressão analítica. Conseqüentemente, seria também razoável aceitar uma medida analítica que refletisse um limite superior para o número de passos, em lugar de um cálculo exato. Defini-se então *complexidade (local) assintótica* de uma algoritmo como sendo um limite superior da sua complexidade local, para uma certa entrada suficientemente grande. A complexidade assintótica deve ser escrita em relação às váriaveis que descrevem o tamanho da entrada do algoritmo.

Para exprimir analiticamente a complexidade assintótica é

conveniente utilizar a notação seguinte, denominada notação $\oplus$. Seja f uma função real não-negativa da variável inteira $n \geq 0$. Diz-se que f é $\oplus(h)$, denotando-se $f = \oplus(h)$, quando existirem constantes c, $n_o > 0$ tais que $f(n) \leq ch$, para $n \geq n_o$.

Por exemplo, $3n^2 + 2n + 5$ é $\oplus(n^2)$, $4n^2 + 2q + 5n$ é $\oplus(n^2 + q)$, $2n + \log n + 1$ é $\oplus(n)$, 231 é $\oplus(1)$, e assim por diante. É fácil verificar que $\oplus(h_1+h_2) = \oplus(h_1) + \oplus(h_2)$ e $\oplus(h_1.h_2) = \oplus(h_1).\oplus(h_2)$. Se $h_1 \geq h_2$, então $\oplus(h_1+h_2) = \oplus(h_1)$. Seja k uma constante, então $\oplus(k.h) = k.\oplus(h) = \oplus(h)$. Quando a função f é $\oplus(h)$ diz-se também que f é da *ordem* de h.

Defini-se *complexidade de pior caso* (ou simplismente *complexidade*) de um algoritmo como o valor máximo dentre todas as suas complexidades assintóticas, para entradas suficientemente grandes. Ou seja, a complexidade de pior caso traduz um limite superior do número de passos necessários à computação da entrada mais desfavorável, de tamanho suficientemente grande.

A complexidade de um algoritmo é sem dúvida um indicador importante para a avaliação da sua eficiência de tempo. Mas certamente também possui aspectos desvantajosos e tampouco é o único indicador existente. A complexidade procura traduzir analiticamente uma expressão da eficiência de tempo do pior caso. Porém, o pior caso pode corresponder a um número de passos bastante maior do que os casos mais freqüentes. Além disso, a expressão de complexidade não considera as constantes. Este é um outro fator onde distorções podem surgir.

Por analogia, defini-se a *complexidade de melhor caso* como sendo o valor mínimo dentre todas complexidades assintóticas do algoritmo, para entradas suficientemente grandes. Isto é, a

complexidade de melhor caso corresponde a um limite superior do número de passos necessários à computação da entrada mais favorável, de tamanho suficientmentee grande. Analogamente ao pior caso, este novo indicador deve ser expresso em função do tamanho da entrada.

A notação Θ é útil no estudo de complexidade do melhor caso. Seja f uma função real não negativa da variável inteira $n > 0$. Então $f = \Theta(h)$ significa que existem constantes c, $n_o > 0$, tais que $f(n) \geq ch$, para $n \geq n_o$. Por exemplo, $2n^3 + 5n$ é $\Theta(n^2)$.

Seja **P** um problema algorítmico, cuja entrada possui tamanho $n > 0$ e g uma função real positiva da variável n. Diz-se que $\Theta(g)$ é um *limite inferior* de **P** quando qualquer algoritmo α que resolva **P** requerer pelo menos $\Theta(g)$ passos. Isto é, se $\Phi(f)$ for a complexidade (pior caso) de α então g é $\Phi(f)$. Por exemplo, se $\Theta(n^2)$ for um limite inferior para **P** não poderá existir algoritmo que o resolva em $\Phi(n)$ passos. Um algoritmo α_o que possua complexidade $\Phi(g)$ é denominado *ótimo* e, neste caso, g é um *limite inferior máximo* de **P**. Observe que α_o é o algoritmo de complexidade mais baixa dentre todos os que resolvem **P**. Este novo indicador é obviamente importante e, freqüentemente, de difícil determinação. é também mais geral que os anteriores, pois é relativo ao problema e não a alguma solução específica.

Como exemplo, considere o *problema de ordenação*. Dada uma seqüência S de números, o objetivo consiste em dispô-los em ordem não-crescente. Sejam s_i e $s_j \in S$. Se $s_i < s_j$ e s_i sucede s_j em S, então diz-se que o par s_i, s_j forma uma *inversão*. Por exemplo, a seqüência 6 8 5 7 apresenta duas inversões 6 8 e 5 7. A seqüência estará ordenada exatamente quando não apresentar inversões. Seja

s_i, s_j uma inversão. Se s_i, s_j intercambiarem posições em S, então s_i, s_j obviamente deixa de ser inversão. Esta troca de posições pode também criar ou eliminar outras inversões. Contudo, o número total das inversões eliminadas é maior do que o das criadas. Essas observações conduzem ao seguinte algoritmo de ordenação.

Algoritmo: Ordenação de uma seqüência
dados: seqüência: $s_1, s_2, \ldots, s_n$
enquanto existir uma inversão s_i, s_j *faça*
trocar de posição s_i com s_j

Observe que o algoritmo apresenta basicamente duas operações: (i) a identificação de uma inversão s_i, s_j e (ii) a correspondente troca de posições se s_i com s_j. Seja **i** o número total de inversões de S. A operação (ii) pode ser realizada obviamente em tempo constante, logo requer $\oplus(i)$ passos no total. A operação (i) pode ser realizada, sem dificuldade em $\oplus(n)$ passos por inversão. Para tal, basta percorrer S da esquerda para a direita, comparando cada número da seqüência como seu antecessor. Portanto, para indentificar todas as inversões do processo o algoritmo de ordenação requer tempo $\oplus(n!)$. A leitura dos dados é realizada em tempo $\oplus(n)$. Logo, sua complexidade assintótica é $\oplus(n! + n)$. As complexidades de pior e melhor caso podem ser calculadas através da atribuição de valores específicos a **i**. O valor máximo do número de inversões **i** corresponde ao caso em que a seqüência de entrada S se encontra em ordem decrescente. Portanto, $\mathbf{i}_{máx} = n(n-1)/2$. O valor mínimo de **i** ocorre quando S já se encontra ordenada. Isto é, $\mathbf{i}_{mín} = 0$. Logo, as complexidades de pior e melhor caso são $\oplus(n^3)$ e $\Theta(n)$, respectivamente.

2.8. - Problemas NP-completo ([71])

2.8.1. - Introdução

Considere o problema de avaliar satisfatoriamente a eficiência de algoritmos. Na seção 2.7., foi ressaltada a conveniência de utilizar a complexidade como medida de eficiência. Contudo, uma vez de posse dessa complexidade de que forma reconhecer se o algoritmo é ou não eficiente?

O critério a seguir procura responder a esta nova questão? Um *algoritmo é eficiente precisamente quando a sua complexidade for um polinômio no tamanho de sua entrada*. Esta classificação é aceitável para a grande maioria dos casos.

Considere a coleção de todos os algoritmos que resolvem um certo problema P. O interesse é conhecer se nessa coleção existe algum que seja eficiente, isto é, de complexidade polinomial. Se existir tal algoritmo, o problema P, será denominado *tratável*, e *intratável* caso contrário.

De acordo com a definição, um problema seria classificado como tratável, quando elabora-se algum algoritmo de complexidade polinomial, que o resolva. Por outro lado, para verificar que é intratável, há necessidade de provar que todo possível algoritmo que o resolva não possui complexidade polinomial.

2.8.2. - Problemas de Decisão

Resolver um problema algorítmico consiste em desenvolver um algorimo, cuja entrada são dados específicos retirados desse

conjunto e cuja saída, denominada *solução*, responda ao objetivo do problema. Os dados específicos que constituem uma entrada formam uma *instância do problema*. Isto é, um problema possui tantas instâncias diferentes quantas são as variações possíveis de seus dados.

Supõe-se que cada instância do problema seja apresentada ao algoritmo, segundo uma codificação conveniente. O comprimento total dessa codificação constitui o *tamanho* de entrada do algoritmo. Naturalmente, este parâmetro é importante, pois é em relação a ele que são tomadas as medidas de complexidade.

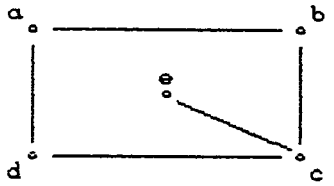
Há certas classes gerais de problemas algorítmicos. Por exemplo, existem os *Problemas de Decisão*, os de *Localização* e os de *Otimização*. Num problema de decisão, o objetivo consiste em decidir a resposta SIM ou NÃO a uma questão q. Num problema de localização, o objetivo é localizar uma certa estrutura $\mathbb{V}$ que satisfaça um conjunto de propriedades dadas. Se as propriedades q que $\mathbb{V}$ deve satisfazer envolverem critérios C de otimização, então o problema torna-se de otimização. Observe que é possível formular um problema de decisão cujo objetivo é indagar se existe ou não a mencionada estrutura $\mathbb{V}$, satisfazendo às propriedades dadas. Isto é, existem problemas, que podem ser *associados*, de decisão, de localização e de otimização.

PROBLEMA DE DECISÃO
Existe estrutura V que
satisfaça propriedades q

PROBLEMA DE LOCALIZAÇÃO
Encontrar estrutura V que
satisfaça propriedade q

PROBLEMA DE OTIMIZAÇÃO
Encontrar estrutura V que
satisfaça critérios de otimização

Como exemplo, considere o problema do *máximo conjunto independente*. Por exemplo no grafo da figura abaixo, um conjunto independente maximal é, por exemplo, a e e, enquanto que um máximo conjunto independente é b, e e d.



$\Gamma(a) = \langle b, d \rangle$
$\Gamma(b) = \langle a, c \rangle$
$\Gamma(c) = \langle b, d, e \rangle$
$\Gamma(d) = \langle a, c \rangle$
$\Gamma(e) = \langle c \rangle$

Figura 2.10. Grafo para exemplo de problemas associados (decisão, localização e otimização) do máx. conj. indep.

Os seguintes problemas podem ser associados

PROBLEMA DE DECISÃO

Dados: Um grafo $\mathbb{G}$ e um inteiro $k > 0$

Objetivo: Verificar se $\mathbb{G}$ possui um conjunto independente maximal $\mathbb{V}$ de cardinalidade $\leq k$.

PROBLEMA DE LOCALIZAÇÃO

Dados: Um grafo $\mathbb{G}$ e um inteiro $k > 0$

Objetivo: Localizar, em $\mathbb{G}$, um conjunto independente maximal $\mathbb{V}$, de cardinalidade $\leq k$.

PROBLEMA DE OTIMIZAÇÃO

Dados: Um grafo $\mathbb{G}$ e um inteiro $k > 0$

Objetivo: Verificar, em $\mathbb{G}$, se existe um conjunto independente maximal $\mathbb{V}$ de cardinalidade $\leq k$ e que seja o máximo conjunto independente $\mathbb{V}^*$, $(\mathbb{V} = \mathbb{V}^*)$.

Os três problemas associados, acima, estão relacionados. Suponha que o *Problema de Otimização* seja resolvido e o máximo

conjunto independente encontrado seja denominado por $\mathbb{V}^*$. Então $\mathbb{V}^*$ pode ser utilizado para resolver o *Problema de Localização* associado, da seguinte maneira: Seja t a cardinalidade do conjunto $\mathbb{V}^*$. Então se $t \leq k$, $\mathbb{V}^*$ é também uma solução para o *Problema de Lacalização*. Caso contrário, se $t > k$, não existe em $\mathbb{G}$ um máximo conjunto independente de cardinalidade $\leq k$. Obviamente, isto resolve também o *Problema de Decisão* associado. Então, para o caso do máximo conjunto independente, o *Problema de Decisão* é de dificuldade não maior do que o de *Localização*, e este de dificuldade não maior do que o de *Otimização*. Aliás, é natural assim que o seja. Contudo é bastante menos intuitivo que, também em diversos casos, os problemas de otimização e localização apresentam ambos dificuldades não maior do que o da decisão associada.

Os problemas, aqui em discusão, serão todos de decisão. Uma justificativa para esta escolha é que, em geral, o problema de decisão é o mais simples entre os três associados. Por isso, alguma prova de sua possível instabilidade pode ser estendida aos ouros casos.

A notação $\P(\mathbf{D}, \mathbf{Q})$ será utilizada para representar um problema de decisão $\P$. **D** representa o conjunto de dados e **Q** a questão (decisão) correspondente. Quando conveniente, utiliza-se $\P(\mathbf{I})$ para denotar o problema $\P(\mathbf{D}, \mathbf{Q})$ aplicado à instância $\mathbf{I} \in \mathbf{D}$.

2.8.3. - A classe P

Um algoritmo efiente é aquele cuja complexidade é uma função polinomial no tamanho dos dados de entrada. Observe que para um

problema de decisão, o tamanho da saída é constante, o que possibilita ignorá-lo. Por exemplo, se n for o tamanho de entrada, algoritmos cujas complexidades sejam $\oplus(1)$, $\oplus(n)$, $\oplus(n^2 \log n)$ ou $\oplus(n^{10})$, seriam todos classificados como eficientes. Por outro lado, complexidades como, por exemplo $\oplus(2^n)$ ou $\oplus(n!)$, $\oplus(exp(n))$ corresponderiam a algoritmos não eficientes.

Um algoritmo será denominado *polinomial (exponencial)* quando sua complexidade for uma função polinomial (exponencial) no tamanho dos dados de entrada.

A adoção do critério acima de classificação de algoritmos quanto a sua eficiência foi também motivada por outros fatores. Por exemplo, as expressões de complexidade dos algoritmos polinomiais são freqüentemente polinômios de baixo grau. Um outro argumento diz respeito ao modelo de computação correspondente à medida de complexidade. Como foi observado, na secão 2.7. a idéia de *passo* de um algoritmo é fundamental para a conceituação de sua complexidade. E esta idéia está relacionada ao *modelo de computação* utilizado. Existem alguns modelos que podem ser considerados como abstrações dos computadores reais (como o RAM). Entre esses, em geral, é preservado o caráter polinomial de algoritmos. Isto é, um algoritmo polinomial segundo um certo modelo, permaneceria polinomial quando traduzido para um outro. Assim sendo, o conceito de eficiência seria independente do modelo de computação adotado.

Define-se a classe P de problemas de decisão, como sendo aquela que compreende precisamente aqueles problemas que admitem algoritmos polinomiais para resolvê-los.

Obs: Observe que se os algoritmos conhecidos para um certo

problema ¶ forem todos exponenciais não necessariamente ¶ ∉ P. Se de fato $\pi \notin P$ então deve existir alguma *prova* de que todo possível algoritmo para resolvr ¶ não é polinomial. Por exemplo, os algoritmos conhecidos até agora para o problema do *máximo conjunto independente*, são todos exponenciais. Contudo não é conhecida prova de que seja impossível a formulação de algoritmo polinomial para o problema. Isto é, se desconhece se o *problema máximo conjunto independente* pertence ou não a P.

2.8.4. - A classe NP

Considere um problema de decisão ¶. Se ¶ for solúvel através da aplicação de algum processo, então necessariamente existe uma *justificativa* para a solução de ¶. Esta justificativa pode ser representada por um determinado conjunto de argumentos, os quais, quando interpretados convenientemente, podem atestar a veracidade da resposta SIM ou NÃO dada ao problema.

Considere o problema *facção*, onde o objetivo é decidir, se um dado grafo $\mathcal{G}$ contém uma facção de tamanho $\geq k$, onde $k > 0$ é um inteiro dado. Uma justificativa para a resposta SIM pode ser obtida exibindo-se uma facção P de tamanho $\geq k$. Efetua-se então uma verificação para reconhecer (*i*) que P de fato é uma facção (ou seja, todo par de vértices de P é adjacente em $\mathcal{G}$) e (*ii*) que $|P| \geq k$. Uma justificativa para a resposta NÃO pode consistir de uma lista de todas as facções de $\mathcal{G}$. Efetuamos então uma verificação para confirmar que a lista de fato está completa e que o tamanho de cada facção é $< k$.

Observe que o processo acima de justificar respostas a

problemas de decisão se compõe de duas fases distintas:

FASE 1: EXIBIÇÃO

Consiste em exibir a justificativa.

FASE 2: RECONHECIMENTO

Consiste em verificar que a justificativa apresentada (no passo de exibição) é, de fato, satisfatória.

Seja novamente, o problema do *máximo conjunto independente* para o grafo da Fig. 2.10. com $k \leq 3$, cuja solução é SIM. Na justificativa, o passo de exibição consiste da seqüência $\mathbb{V}$ de vértices a, d e e. O passo de reconhecimento consiste em verificar se $\mathbb{V}$ é de fato um dos máximo conjunto independente. Conforme visto, o processo de reconhecimento é simples. Basta testar se: (i) $\mathbb{V}$ é um conjunto independente maximal (ou seja, $\mathbb{V} \cap \Gamma(\mathbb{V}) = \emptyset$ e $\mathbb{V} \cup \Gamma(\mathbb{V}) = \mathbb{X}$) e ($ii$) verificar se $\mathbb{V}$ é o maior. O algoritmo correspondente ao processo é facilmente implementável em tempo polinomial no tamanho do grafo.

Retorne agora ao problema do *máximo conjunto independente* para o grafo da Fig. 2.10. com $k \leq 2$, cuja solução é NÃO. O passo de exibição da justificativa é o conjunto das 2 seqüências: (a, c; a, e), conforme indicado. O passo de reconhecimento consiste em comprovar que (i) cada seqüência de vértices é um conjunto independente maximal e (ii) todo conjunto independente maximal possui cardinalidade $< k$. O algoritmo de reconhecimento não é tão simples quanto aquele da justificativa SIM. A operação (i) deve ser realizada para *cada* conjunto exibido (ao contrário do caso anterior que requeria verificações sobre um único). Para comprovar (ii) uma idéia seria enumerar *todos* os conjuntos independentes maximais do grafo $\mathbb{G}$. Como o número total de

conjuntos pode ser exponencial com o tamanho de G, esse algoritmo é de natureza exponencial. Até o momento, é desconhecido se existe algum processo para justificar a resposta NÃO, ao *máximo conjunto independente*, tal que o passo de reconhecimento corresponda a um algoritmo polinomial.

Define-se então a clase NP como sendo aquela que compreende todos os problemas de decisão Π, tais que existe uma justificativa à resposta SIM para Π, cujo passo de reconhecimento pode ser realizado por um algoritmo polinomial no tamanho da entrada de Π.

Ressalte-se na definição de classe NP, que não se exige uma solução polinomial para π. Além disso, para existir algoritmo de reconhecimento polinomial é necessário (mas não suficiente) que o *tamanho da justificativa*, dada pelo passo de exibição, seja polinomial no tamanho da entrada do problema. Ainda em relação à definição de NP, observe que nada se exige da justificativa NÃO. De fato, há problemas de NP que admitem algoritmos polinomiais para suas justificativas NÃO. Como há também aqueles para os quais tais algoritmos não são conhecidos,

Os exemplos de problemas pertencentes à classe NP são inúmeros. A justificativa SIM dada ao *Ciclo Hamiltoniano* possui como passo de reconhecimento um algoritmo polinomial no tamanho da entrada do grafo. Logo, *Ciclo Hamiltoniano* pertence a NP.

Para verificar se um problema Π pertence ou não a NP procede-se da seguinte maneira:

(1) Define-se uma justificativa J conveniente para a resposta SIM ao problema;

(2) Elabora-se um algoritmo para reconhecer se J está

correta. A entrada desse algoritmo consiste de J e da entrada de ¶.

Se o algoritmo resultante do passo (2) for polinomial no tamanho da entrada de π, então π pertence a NP. O caso contrário, naturalmente, não implica necessariamente em não pertinência a NP.

Mostra-se abaixo, que os problemas *Conjunto Independente maximal* e *Cobertura de Vértices* pertencem a NP.

PROBLEMA: Conjunto independente maximal

Dados: Um grafo $G(X, A)$ e um um inteiro $k > 0$

Decisão: G possui um conjunto independente maximal, de tamanho $\geq k$?

JUSTIFICATIVA: Sim

(1) Exibição: O grafo G e um subconjunto de vértices $V \subseteq X$.

(2) Algoritmo de reconhecimento: Examina-se cada lista de adjacência $\Gamma(x_i)$, $x_i \in V$, para verificar se todo $x_j \in \Gamma(x_i)$ é tal que $x_j \notin V$. Seja agora $k' = |V|$. É imediato concluir que a justificativa está correta se e só se essas verificações forem todas satisfeitas, e além disso, $k' \geq k$.

CONCLUSÃO: O algoritmo (2) é polinomial no tamanho dos dados do CONJUNTO INDEPENDENTE MAXIMAL. Logo, pertence a NP.

PROBLEMA: Cobertura de vértices

Dados: Um grafo $G(X, A)$ e um um inteiro $k > 0$

Decisão: G possui uma cobertura de vértices, de tamanho $\leq k$?

JUSTIFICATIVA: Sim

(1) Exibição: O grafo G e um subconjunto de vértices $V \subseteq X$.

(2) Algoritmo de reconhecimento: Examina-se cada aresta

$(x_i, x_j) \in \mathbb{X}$ com o intuito de verificar se x_i ou $x_j \in \mathbb{V}$. Seja agora $k' = |\mathbb{V}|$. é imediato concluir que a justificativa está correta se e só se as verificações forem todas satisfeitas, e além disso, $k' \leq k$.

CONCLUSÃO: O algoritmo (2) é polinomial no tamanho dos dados da COBERTURA DE VÉRTICES. Logo, pertence a NP.

Existem também diversos problemas para os quais se desconhece sua pertinência ou não a essa classe. Um exemplo é a *Máxima Facção* abaixo.

PROBLEMA: Máxima facção

Dados: Um grafo $\mathbb{G}(\mathbb{X}, \mathbb{A})$ e um um inteiro $k > 0$

Decisão: A facção de tamanho máximo de $\mathbb{G}$ possui k vértices?

Uma maneira de justificar uma resposta SIM ao problema acima pode ser assim descrita. No passo de exibição apresenta-se uma família $\mathcal{F}$ de conjuntos contendo todas as facções maximais de $\mathbb{G}$. No passo de reconhecimento, comprova-se que $\mathcal{F}$ contém, de fato, todas as facções maximais de $\mathbb{G}$. Seja k' o tamanho da maior facção de $\mathcal{F}$. Obviamente, a justificativa está correta se e só se $k = k'$. Como o tamanho de $\mathcal{F}$ pode ser exponencial quanto ao tamanho de $\mathbb{G}$, esse algoritmo é também exponencial. A conclusão natural é que a justificativa apresentada não permite afirmar que a *máxima facção* pertence a NP. Isto não exclui a possibilidade de existir alguma outra que assim o permita. Contudo, tal justificativa não foi encontrada até o omomento, mas também não foi provado que ela não possa existir. Conseqüentemente, não é conhecido se *facção máxima* pertence ou não a NP. Todas as evidências, no entanto, conduzem a conjectura de que *facção máxima* $\notin$ NP.

CAPÍTULO III

CONJUNTOS INDEPENDENTES MAXIMAIS

3.1. - Conjuntos independentes

Considere um grafo não-direcionado $\mathbb{G} = (\mathbb{X}, \Gamma)$. Um *conjunto de vértices independentes* $\mathbb{V}$ (ou *conjunto internamente estável*) é um conjunto de vértices do grafo $\mathbb{G}$, para o qual dois quaisquer vértices distintos não são adjacentes ($\forall\ x_i, x_j \in \mathbb{V};\ x_i \neq x_j$ implica $x_j \notin \Gamma(x_i)$; ou seja, não há dois vértices neste conjunto ligados por uma aresta. Portanto, qualquer conjunto $\mathbb{V} \subseteq \mathbb{X}$ que satisfizer a relação:

$$\mathbb{V} \cap \Gamma(\mathbb{V}) = \emptyset \qquad (3.1)$$

é um conjunto de vértices independentes. Para o grafo da Fig. 3.1. por exemplo, os conjuntos: (x_1, x_3, x_7), (x_3, x_7), (x_2, x_3, x_5, x_7), etc. são todos conjuntos de vértices independentes. Quando não houver dúvidas a palavra "vértices" será suprimida e, então, serão chamados de conjuntos independentes.

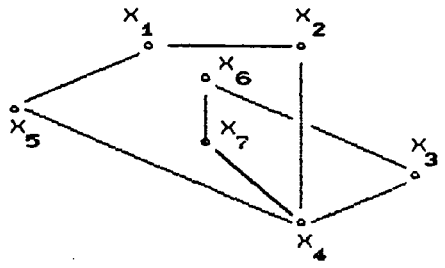


Figura 3.1. Grafo para exemplo de conjunto independente

3.2. - Conjunto independente maximal

Um conjunto independente é dito maximal quando não existir outro conjunto independente que o contenha. Assim, um conjunto $\mathbb{V}$ é um *conjunto independente maximal* se satisfizer:

$$\mathbb{V} \cap \Gamma(\mathbb{V}) = \emptyset$$

e além disso

$$\mathbb{W} \cap \Gamma(\mathbb{W}) \neq \emptyset \quad , \quad \forall\, \mathbb{W} \supset \mathbb{V} \tag{3.2}$$

Por isso, no grafo da Fig. 3.1., o conjunto $\{x_2, x_5, x_7\}$ é um conjunto independente, porém não maximal. Já o conjunto $\{x_2, x_5, x_7, x_3\}$ é um conjunto independente maximal. Os conjuntos $\{x_1, x_7, x_3\}$ ou $\{x_1, x_6, x_4\}$ são também conjuntos independentes maximais do grafo da Fig. 3.1. Existirá, portanto, mais do que um conjunto independente maximal para um determinado grafo. Observa-se que o número de elementos (vértices) de cada um dos vários conjuntos independentes maximais podem ser distintos.

Nota: Pode-se dizer também que um conjunto $\mathbb{V}$ será independente maximal quando satisfizer:

$$\mathbb{V} \cap \Gamma(\mathbb{V}) = \emptyset \qquad e \qquad \mathbb{V} \cup \Gamma(\mathbb{V}) = \mathbb{X}$$

3.3. - Número de independência e máximo conjunto independente

Se $\mathcal{F}$ for família dos conjuntos independentes maximais $\mathbb{V}$,

então o número

$$\alpha[\mathbb{G}] = \max_{\mathbb{V} \in \mathcal{F}} |\mathbb{V}| \qquad (3.3)$$

será chamado *número de independência* do grafo $\mathbb{G}$, e o conjunto $\mathbb{V}^*$ do qual é derivado, será chamado *máximo conjunto independente*.

Para o grafo da Fig. 3.1., é apresentado na Fig. 3.2., a família de conjuntos independentes maximais.

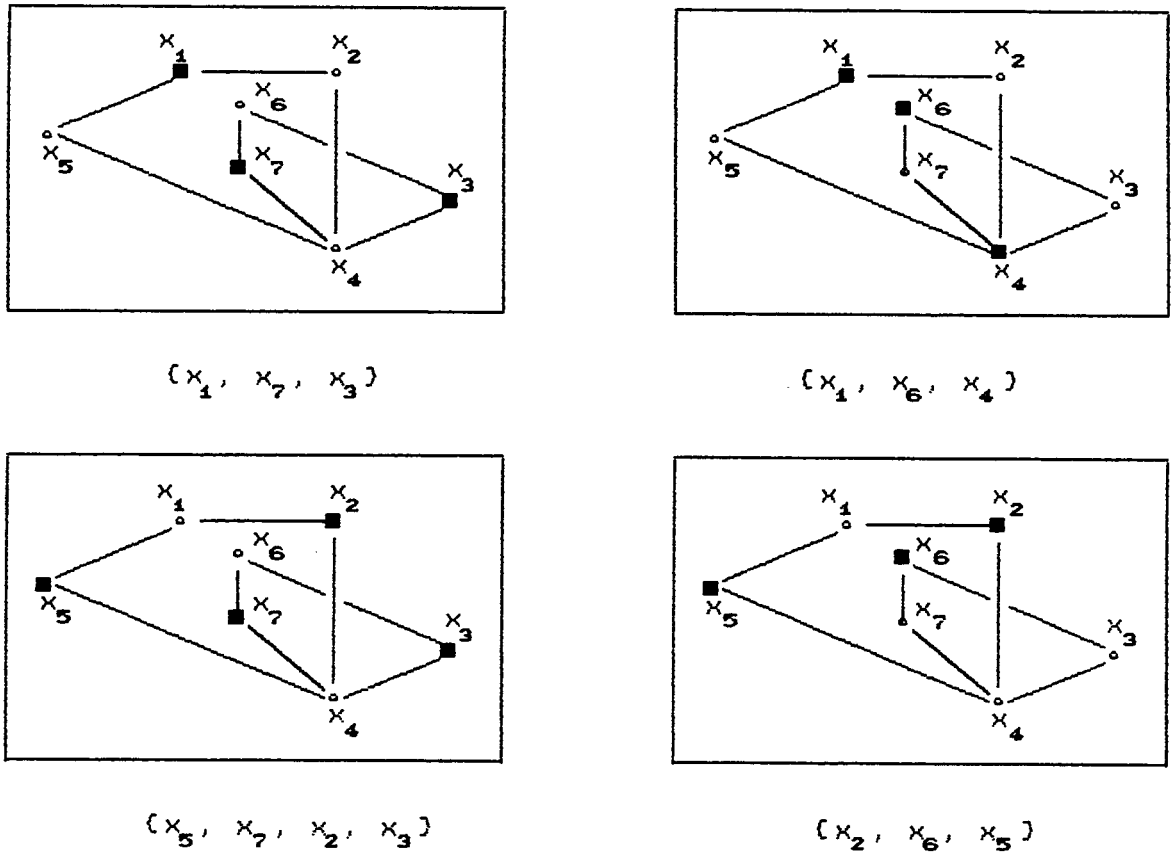


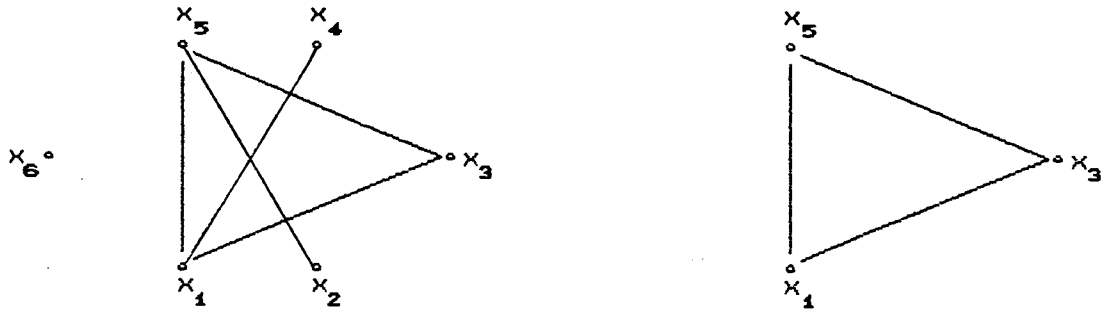
Figura 3.2. Conjuntos indep. maximais do grafo $\mathbb{G}$ da Fig. 3.1.

O maior destes conjuntos possui quatro elementos e por isso $\alpha[\mathbb{G}] = 4$. Conseqüentemente, o conjunto $\mathbb{V}^* = (x_2, x_5, x_7, x_3)$ é, o máximo conjunto independente.

3.4. - Facções (Subgrafos completos maximais)

Um conceito que é oposto ao de conjunto independente maximal é o de subgrafo completo maximal. Portanto, *subgrafo completo maximal (facção)* de um grafo $\mathbb{G}$, é um subgrafo formado pelo conjunto $\mathbb{V}$ de vértices que é completo e que é maximal no sentido de que qualquer outro subgrafo $\mathbb{W}$ de $\mathbb{G}$ estando contido em $\mathbb{V}$ (isto é, $\mathbb{W} \subset \mathbb{V}$), não será completo. Assim, ao contrário do conjunto independente maximal, onde dois vértices não são adjacentes, numa facção cada vértice é adjacente a todos os outros. É óbvio, portanto, que o conjunto independente maximal de um grafo $\mathbb{G}$ corresponde a uma facção do grafo $\overline{\mathbb{G}}$ e vice-versa, onde $\overline{\mathbb{G}}$ é o complementar de $\mathbb{G}$.

Para o grafo complementar $\overline{\mathbb{G}}$ do grafo $\mathbb{G}$ da Figura 2.6. do Cap. 2, seção 2.4., tem-se:



(a) Grafo $\overline{\mathbb{G}}$ (b) máxima facção contida em $\overline{\mathbb{G}}$

Figura 3.3. Grafo $\overline{\mathbb{G}}$ e a máxima facção de $\overline{\mathbb{G}}$

Analogamente como na definição do número de independência, relação (3.3), pode-se definir o *número facção (densidade)* como sendo o número máximo de vértices numa facção.

3.5. - Algumas aplicações relacionadas com conjuntos independentes maximais

3.5.1. - Introdução

A modelagem, com auxílio da noção de independência, é bastante versátil, aplicando-se a:

(i) problemas de coloração de vértices em grafos;

(ii) problemas de atribuição ("*matching*") em grafos;

(iii) geração e alocação de horários em empresas e instituições de ensino;

(iv) compatibilidade, semelhança, comparação, comunicação.

Nestes problemas trabalha-se com o grafo que representa a situação, ou com seu grafo complementar, ou com um grafo associado, construído de forma a representar convenientemente as restrições do problema.

Na seqüência serão apresentados alguns exemplos de problemas que podem ser resolvidos através da modelagem em grafos utilizando-se a propriedade da independência maximal.

3.5.2. - Seleção de projetos ([13])

Considere n projetos que devem ser executados, e suponha que cada projeto x_i requeira um subconjunto $R_i \subseteq \{1, \ldots, r\}$, onde $\{1, \ldots, r\}$ é um conjunto de r recursos disponíveis para a execussão deste projeto. Além disso, suponha que cada projeto (fornecido seus recursos necessários), possa ser executado num determinado período de tempo. Forme agora um grafo G, onde a cada

vértice corresponde um projeto, e as arestas (x_i, x_j) serão acrescentadas todas as vezes que x_i e x_j tiverem alguma necessidade de recursos em comum. Isto é, se $R_i \cap R_j \neq \emptyset$. Um conjunto independente maximal de $\mathbb{G}$ representa então um conjunto maximal de projetos que podem ser executados simultaneamente durante um único período de tempo.

Num sistema dinâmico novos projetos tornam-se necessários para execussão após determinado período de tempo, tal que a família de conjuntos independentes maximais de $\mathbb{G}$ são encontrados repetidamente de maneira a escolher qual conjunto maximal de projetos deve ser executado durante o período corrente.

3.5.3. - Rotação de tripulações em uma empresa de transporte aéreo. ([63], [7])

Neste problema, apresentado por Roy [62], conhece-se o grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$ representativo da rede da companhia e quer-se alocar tripulações de modo a atender a uma série de restrições (local de residência, tempo de viagem, repouso entre viagens, etc.); o itinerário de cada tripulação deve corresponder a um circuito no qual se considera o vértice inicial associado à cidade onde reside a respectiva tripulação.

O conjunto de circuitos viáveis é um dado do problema, fornecido pelo departamento pessoal da companhia. O que se deseja, é escolher um subconjunto de circuitos que forme uma partição do conjunto $\mathbb{A}$ de rotas.

Defini-se, então, um grafo $\mathbb{H} = (\mathbb{Y}, \mathbb{B})$ no qual $y_i \subset \mathbb{Y}$ é um circuito viável e $(y_i, y_j) \in \mathbb{B}$ existe se os circuitos y_i e y_j

possuem ao menos uma rota em comum.

A solução será, então, dada por um máximo conjunto independente $\mathbb{M} \subset \mathbb{Y}$ tal que

$$\bigcup_{y_i \in \mathbb{M}} y_i = \mathbb{A}$$

seja, por exemplo, a rede cujo grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$ está representado na figura 3.4.

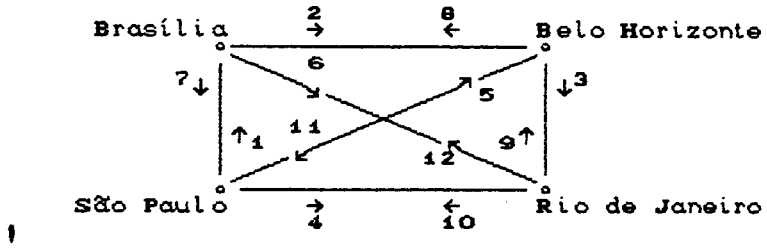


Figura 3.4. Rotas aéreas possíveis.

Cada rota é representada por um arco; suponhamos que os circuitos viáveis em relação à alocação de tripulantes sejam:

$y_1 = (1, 2, 11)$ $\qquad$ $y_5 = (4, 9, 3, 10)$

$y_2 = (1, 6, 10)$ $\qquad$ $y_6 = (5, 3, 10)$

$y_3 = (2, 3, 12)$ $\qquad$ $y_7 = (6, 12)$

$y_4 = (3, 10, 5)$ $\qquad$ $y_8 = (7, 5, 8)$

O grafo $\mathbb{H} = (\mathbb{Y}, \mathbb{B})$, indicativo das relações entre os circuitos viáveis, está indicado na figura 3.5.

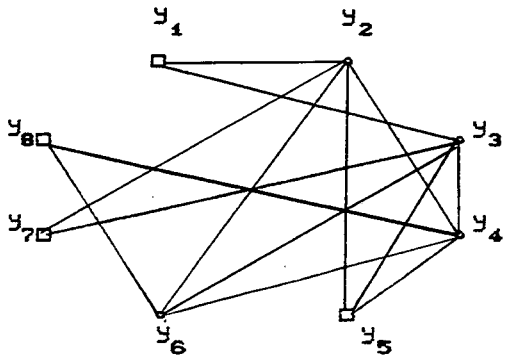


Figura 3.5. Grafo $\mathbb{H}$ (conjunto independente)

O máximo conjunto independente $M = \{y_1, y_8, y_7, y_5\}$ é uma solução do problema, pois todas as rotas estão presentes, somente uma vez, o que caracteriza uma partição para o conjunto de rotas.

3.5.4. - Coloração dos vértices de um determinado grafo G

Seja $G = (X, A)$ um grafo e $C = \{c_i\}$ um conjunto de cores. Uma coloração de G é uma atribuição de alguma cor de C para cada vértice de X, de tal modo que a dois vértices adjacentes sejam atribuídas cores diferentes.

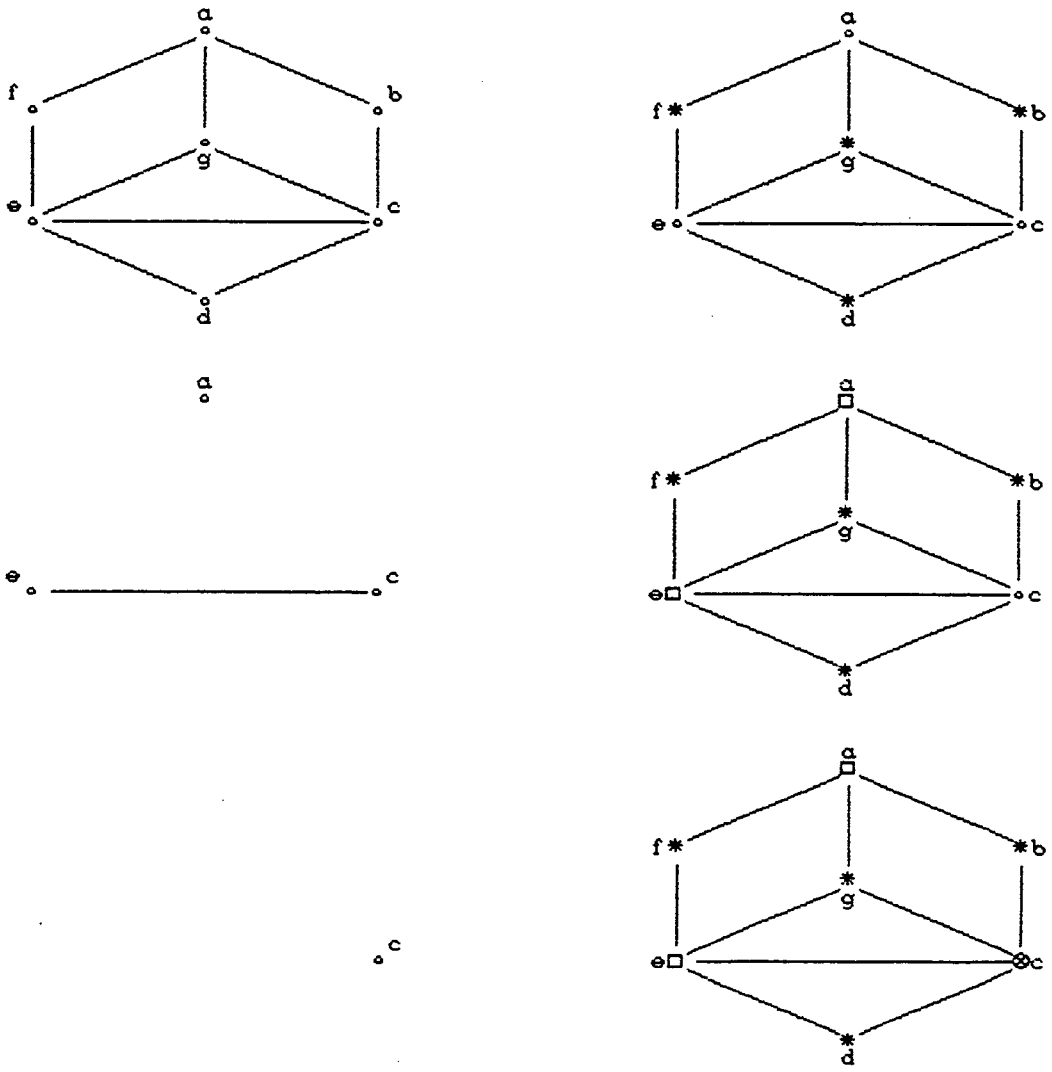


Figura 3.6. $C = \{*, \square, \otimes\}$: Coloração máxima para o Grafo G

Assim sendo, uma coloração de $\mathbb{G}$ é uma função $f:\mathbb{X} \longrightarrow \mathbb{C}$ tal que para cada par de vértices x_i, $x_j \in \mathbb{X}$ tem-se $(x_i, x_j) \in \mathbb{A} \Longrightarrow f(x_i) \neq f(x_j)$. Uma *k-coloração* de $\mathbb{G}$ é uma coloração que utiliza um total de k cores. Na Fig..3.6., tem-se uma 3-coloração.

Os conceitos de coloração, facção e conjunto independente maximal estão naturalmente relacionados. Por exemplo, como são necessárias ρ cores para colorir os ρ vértices de uma facção de tamanho k, conclui-se que o número cromático de um grafo $\mathbb{G}$ seja maior ou igual do que o tamanho da maior facção de $\mathbb{G}$. Considere uma k-coloração de $\mathbb{G} = (\mathbb{X}, \mathbb{A})$. Sejam, $\mathbb{X}_1$, $\mathbb{X}_2$, ..., $\mathbb{X}_k$ os subconjuntos (disjuntos) de $\mathbb{X}$, induzidos pela k-coloração. Então obviamente $\cup\mathbb{X}_i = \mathbb{X}$ e cada $\mathbb{X}_i$ é um conjunto independente maximal. Então o problema de determinar uma coloração mínima de $\mathbb{G}$ pode ser formulado em termos de particionar $\mathbb{X}$ em um número mínimo de conjuntos independentes maximais.

3.5.5. - Problema de Shannon ([7])

é um problema clássico que constitui uma aplicação direta da noção de indepêndencia maximal: ao se projetar um código a ser usado em um sistema de comunicações, é importante que seus sinais não sejam confundidos uns com os outros; se monta um grafo $\mathbb{G} = (\mathbb{X},\mathbb{A})$ onde $\mathbb{X}$ = {sinais} e a relação geradora de $\mathbb{A}$ seja (x_i pode ser confundido com x_j) então, o máximo conjunto independente fornecerá $\alpha[\mathbb{G}]$ sinais para os quais não haverá risco de se confundir as mensagens (figura 3.7).

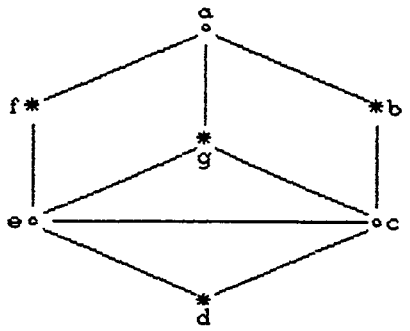


Figura 3.7. Sinais possíveis $\mathbb{V}^* = (b, d, f, g)$ e $\alpha[\mathbb{G}] = 4$

3.5.6. - Problema das oito rainhas ([6], [7])

Trata-se de achar todas as maneiras de colocar 8 rainhas em um tabuleiro de xadrez, de forma que nenhuma ataque a outra (Fig. 3.8). Há 23 soluções, e a cada uma corresponde um máximo conjunto independente do grafo $\mathbb{G} = (\mathbb{X}, \mathbb{A})$, onde $|\mathbb{X}| = 64$ e $\alpha = (x_i, x_j) \in \mathbb{A}$ existe, se x_i e x_j estiverem em casas da mesma linha, coluna ou diagonal do tabuleiro.

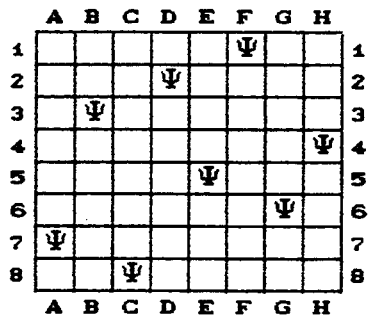


Figura 3.8. Tabuleiro de xadrez com uma das soluções para o problema das oito rainhas

3.5.7.- Pesquisa tipológica ([63], [7])

Um exemplo relacionado com pesquisa de mercado é, também, discutido por Roy [63]. Pesquisando as reações de consumidores frente aos diversos aspectos de um novo produto, constrói-se um

grafo baseado na semelhança de comportamento dos consumidores, o qual caracteriza determinados tipos de consumidores; para determinar esses tipos basta, portanto, achar os conjuntos independentes do grafo complementar.

3.5.8. - Aplicação de conjuntos independentes maximais à grafos orientados ([1])

Seja a construção de um objeto no qual interferem fundamentamente oito fatores:

1) Forma
2) Cor
3) Aspecto
4) Matéria Prima
5) Volume
6) Peso
7) Processo de Fabricação
8) Acabamento

Numa análise do relacionamento desses fatores, estabeleceu-se o seguinte grafo ($\mathbb{G}$) constando de $\mathbb{X}$ vértices e Γ relações, assim $\mathbb{G} = (\mathbb{X}, \Gamma)$.

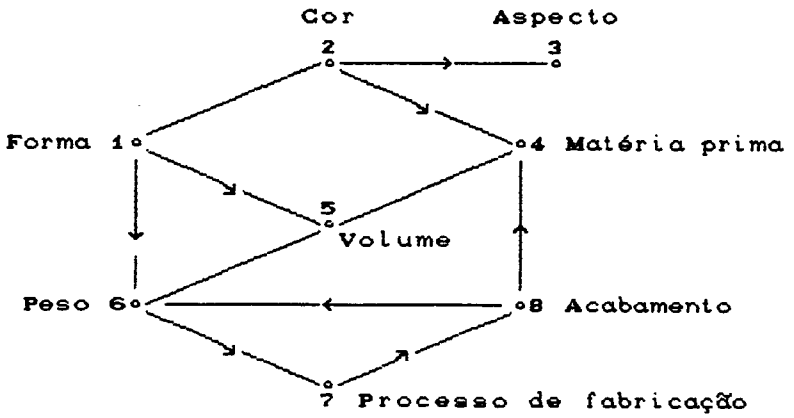
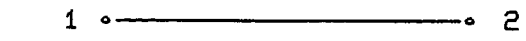


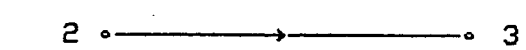
Figura 3.9. Relações Γ em $\mathbb{G}$

Isto quer dizer que:

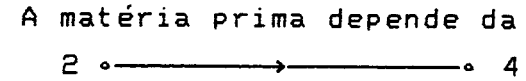
1°) A forma depende da cor como a cor depende da forma.



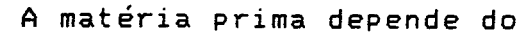
2°) O aspecto depende da cor.



3°) A matéria prima depende da cor.

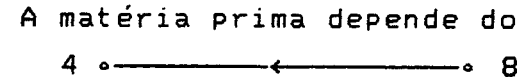


4°) A matéria prima depende do volume assim como o volume depende da matéria prima.

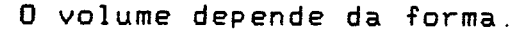


5°) A matéria prima depende do acabamento.

6°) O volume depende da forma.

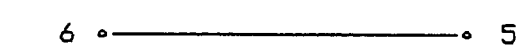


7°) O volume depende do peso assim como o peso depende da forma.

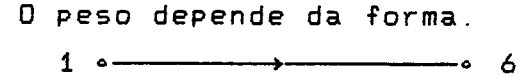


8°) O peso depende da forma.

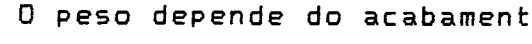
9°) O peso depende do acabamento.



10°) O processo de fabricação depende do peso.



11°) O acabamento depende do processo de fabricação.



Os vértices que determinam um conjunto independente se constituem em fatores autônomos, os quais podem orientar a escolha da opção inicial arbitrária de onde o projeto pode evoluir.

Existindo vários conjuntos independentes, é importante que

se escolha um dos máximos conjuntos independentes.

Os oito conjuntos independentes maximais são:

1 3 4 7	Forma Aspecto Matéria Prima Fabricação	3 4 6	Aspecto Matéria Prima Peso
2 6	Cor Peso	3 5 7	Aspecto Volume Fabricação
2 5 7	Cor Volume Fabricacão	1 3 8	Forma Aspecto Acabamento
3 5 8	Aspecto Volume Acabamento	2 5 8	Cor Volume Acabamento

O projeto, portanto, pode ser iniciado por qualquer desses conjuntos.

Essas oito opcões já orientam o projetista que então pode definir qual partida interessa sua concepção. Caso deseje começar o projeto por onde haja maior liberdade para escolha dos elementos iniciais, então trabalha-se com um conjunto mais numeroso (máximo conjunto independente) que neste caso é o 1 3 4 7, no qual inicia sua criação por forma-aspecto-matéria prima e fabricação para em seguida determinar e compatibilizar: peso-volume-acabamento e cor.

Caso, entretanto, a intenção do projetista seja partir do menor número de considerações livres (mínimo conjunto independente), deverá operar com 2 6, isto é, escolher inicialmente a cor e o peso e determinar e compatibilizar os seis outros componentes.

CAPÍTULO IV

MÉTODOS E/OU ALGORITMOS EXISTENTES PARA ENCONTRAR TODOS OS CONJUNTOS INDEPENDENTES MAXIMAIS DE UM GRAFO $\mathbb{G}$

4.1. - Introdução

Conforme Boaventura Netto [7], "o problema de determinação dos conjuntos independentes maximais é de complexidade exponencial; na prática, as técnicas baseadas na verificação direta da não adjacência exigem muito da memória e utilizam muito tempo de processamento. Por essa razão utilizam-se técnicas mais sofisticadas."

Cabe observar que a cada conjunto independente maximal de um grafo $\mathbb{G}$ qualquer, corresponderá uma facção (clique), do grafo complementar $\overline{\mathbb{G}}$ e vice versa; por isso, o problema é, muitas vezes, apresentado como o da determinação das facções maximais.

4.2. - Determinação de todos os conjuntos independentes maximais

Por causa da relação entre conjuntos independentes maximais e facções mencionadas anteriormente, os métodos de cálculo apresentados neste capítulo, e que serão descritos em termos de conjuntos independentes maximais, poderão também ser usados diretamente para o cálculo de facções.

Em princípio, supõe-se que o cálculo de todos os conjuntos independentes maximais de um grafo seja uma tarefa muito simples, que poderia ser feita pela enumeração sistemática dos conjuntos independentes, testando se são os maiores possíveis (pela tentativa de acrescentar qualquer outro vértice ao conjunto, enquanto preserva-se a independência) e armazenando então os maximais. A suposição será verdadeira para grafos pequenos de, no máximo, 20 vértices, mas, à medida que o número de vértices aumenta, este método de geração torna-se inviável computacionalmente. Isto deve-se, não ao fato de existir um grande número de conjuntos independentes maximais, mas ao fato de que, durante o processo, muitos conjuntos independentes seremformados e mais tarde rejeitados pelo fato de estarem contidos em outros anteriormente gerados e, por isso, não serão maximais.

Descreve-se então métodos que vencem substancialmente esta dificuldade de forma que os conjuntos independentes - uma vez gerados - não necessitam ser checados em relação à maximalidade dos conjuntos previamente gerados.

4.3. - Método de Maghout ([35], [7])

4.3.1. Base teórica do método

É um método que formula a condição de independência utilizando a algebra booleana.

Seja $\mathcal{G} = (\mathcal{X}, \mathcal{A})$ um grafo qualquer.

Define-se, então, as seguintes variáveis booleanas:

$$\alpha_{ij} = \begin{cases} 1 & \text{se } \exists\,(x_i,\ x_j) \in \mathbb{A} \\ 0 & \text{caso contrário} \end{cases}$$

$$x_i = \begin{cases} 1 & \text{se o } \underline{\text{vértice}}\ x_i \in \mathbb{V} \\ 0 & \text{caso contrário} \end{cases}$$

$$x_j = \begin{cases} 1 & \text{se o } \underline{\text{vértice}}\ x_j \in \mathbb{V} \\ 0 & \text{caso contrário} \end{cases}$$

Então, se $\mathbb{V} \subset \mathbb{X}$ for um conjunto independente, a proposição

$$\exists\ (x_i,\ x_j) \in \mathbb{A} \text{ com } x_i,\ x_j \in \mathbb{V}$$

será falsa e, portanto, não se terá

$$\alpha_{ij}\ x_i\ x_j = 1.$$

Se uma das três variáveis for nula, a condição de independência será satisfeita e, por conseguinte, dois vértices quaisquer de $\mathbb{G}$, pertencerão a um conjunto independente quando

$$\alpha_{ij}\ x_i\ x_j = 0.$$

Generalizando a todos os possíveis pares de vértices, obtem-se:

$$\sum_i \sum_j \alpha_{ij}\, x_i\, x_j = \emptyset$$

Tomando o conjugado, utilizando a propriedade de De Morgan ($\overline{a + b} = \overline{a}\cdot\overline{b}$), e lembrando que $\overline{a\cdot b\cdot c} = \overline{a} + \overline{b} + \overline{c}$, tem-se:

$$\overline{\sum_i \sum_j \alpha_{ij}\, x_i\, x_j} = \overline{\emptyset}$$

$$\overline{\alpha_{11}x_1x_1 + \alpha_{12}x_1x_2 + \ldots + \alpha_{kl}x_kx_l + \ldots + \alpha_{n;n}x_nx_n} = 1$$

$$(\overline{\alpha_{11}x_1x_1})(\overline{\alpha_{12}x_1x_2}) \ldots (\overline{\alpha_{kl}x_kx_l}) \ldots (\overline{\alpha_{n;n}x_nx_n}) = 1$$

$$(\overline{\alpha}_{11}+\overline{x}_1+\overline{x}_1)(\overline{\alpha}_{12}+\overline{x}_1+\overline{x}_2)\ldots(\overline{\alpha}_{kl}+\overline{x}_k+\overline{x}_l)\ldots(\overline{\alpha}_{n;n}+\overline{x}_n+\overline{x}_n) = 1$$

$$\prod_i \prod_j \left(\overline{\alpha}_{ij} + \overline{x}_i + \overline{x}_j \right) = 1$$

O produtório tem n^2 termos; se $\alpha_{ij} = \emptyset$ então os vértices x_i e x_j não são adjacentes, portanto $\overline{\alpha}_{ij} = 1$, e assim, esses termos não precisarão ser considerados pois quaisquer que sejam os valores de x_i e x_j, a soma $\overline{\alpha}_{ij} + \overline{x}_i + \overline{x}_j$ será sempre igual a 1 (na álgebra booleana, $\emptyset + \emptyset = \emptyset$, $\emptyset + 1 = 1$ e $1 + 1 = 1$). Restam, portanto, termos, cujo desenvolvimento fornecerá todos os conjuntos independentes (maximais ou não) do grafo; porém, nestes termos pode-se desconsiderar os elementos onde i = j pois $\overline{\alpha_{ij}} = \emptyset$ (todo vértice é adjacente a ele mesmo), o produtório será, portanto, um enumerador de conjuntos independentes. *Enumerador*, ou *função geradora*, é uma função capaz de reproduzir conjuntos escolhidos por um critério combinatório.

Como as variáveis do produtório estão todas representadas na forma de conjugado, cada termo fornecerá um conjunto de vértices de uma facção do grafo complementar e, portanto, o complementar de cada um desses conjuntos será um conjunto independente.

Aplicando operações e propriedades de absorção da álgebra booleana, conforme:

$a + a = a$ $\qquad$ $a + a \cdot b = a$

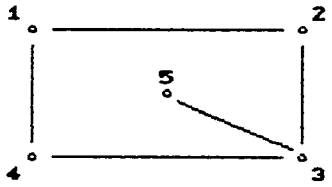
$a \cdot a = a$ $\qquad$ $\overline{(a + b)} = \overline{a} \cdot \overline{b}$

$a + b = b + a$ (comut. ad.) $\qquad$ $\overline{a \cdot b \cdot c} = \overline{a} + \overline{b} + \overline{c}$

$a \cdot b = b \cdot a$ (comut. mult.) $\qquad$ $\overline{a} \cdot \overline{a} = \overline{(a + a)} = \overline{(a)} = \overline{a}$

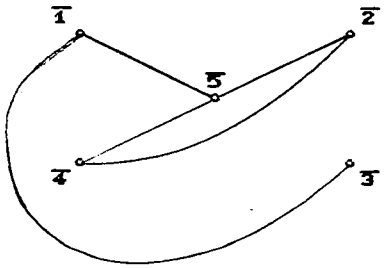
tem-se, ao final, os menores conjuntos e portanto, os seus complementares serão os conjuntos independentes maximais.

Exemplo 4.1: Exemplo para o método de Maghout



$\tau(1) = \langle 2, 4 \rangle$
$\tau(2) = \langle 1, 3 \rangle$
$\tau(3) = \langle 2, 4, 5 \rangle$
$\tau(4) = \langle 1, 3 \rangle$
$\tau(5) = \langle 3 \rangle$

Figura 4.1. Grafo $\mathbb{G}$ para o exemplo do método de Maghout



$\tau(\overline{1}) = \langle \overline{3}, \overline{5} \rangle$
$\tau(\overline{2}) = \langle \overline{4}, \overline{5} \rangle$
$\tau(\overline{3}) = \langle \overline{1} \rangle$
$\tau(\overline{4}) = \langle \overline{2}, \overline{5} \rangle$
$\tau(\overline{5}) = \langle \overline{1}, \overline{2}, \overline{4} \rangle$

Figura 4.2. - Grafo complementar $\overline{\mathbb{G}}$ do grafo $\mathbb{G}$ da Fig. 4.1. com os sucessores de cada vértice

$$\overline{\Sigma \Sigma \alpha_{ij} x_i x_j} = \overline{0}$$

$$\overline{\alpha_{12}x_1x_2 + \alpha_{14}x_1x_4 + \alpha_{23}x_2x_3 + \alpha_{34}x_3x_4 + \alpha_{35}x_3x_5} = 1$$

Com a eliminação dos termos onde i = j (todo vértice é adjacente à ele mesmo), e também dos termos onde $\nexists (x_i, x_j)$ (ou seja, x_i não é adjacente a x_j), tem-se:

$$\left(\overline{\alpha_{12}x_1x_2}\right)\left(\overline{\alpha_{14}x_1x_4}\right)\left(\overline{\alpha_{23}x_2x_3}\right)\left(\overline{\alpha_{34}x_3x_4}\right)\left(\overline{\alpha_{35}x_3x_5}\right) = 1$$

$$\left(\overline{\alpha_{12}} + \overline{x_1} + \overline{x_2}\right)\times\left(\overline{\alpha_{14}} + \overline{x_1} + \overline{x_4}\right)\times\left(\overline{\alpha_{23}} + \overline{x_2} + \overline{x_3}\right)\times$$
$$\times\left(\overline{\alpha_{34}} + \overline{x_3} + \overline{x_4}\right)\times\left(\overline{\alpha_{35}} + \overline{x_3} + \overline{x_5}\right) = 1$$

Como os $\overline{\alpha_{ij}}$ restantes são todos iguais a zero, tem-se:

$$\left(\overline{x_1} + \overline{x_2}\right)\left(\overline{x_1} + \overline{x_4}\right)\left(\overline{x_2} + \overline{x_3}\right)\left(\overline{x_3} + \overline{x_4}\right)\left(\overline{x_3} + \overline{x_5}\right) = 1$$

Suprimindo-se a letra x,

$$(\overline{1} + \overline{2})(\overline{1} + \overline{4})(\overline{2} + \overline{3})(\overline{3} + \overline{4})(\overline{3} + \overline{5}) = 1$$

Pode-se representar também o produtório através da matriz de adjacência triângular (para que cada aresta seja levada em conta uma só vez), do grafo complementar $\overline{G}$ tomando-se somente os vértices não adjacentes.

Para o exemplo acima, tem-se:

$$\overline{A} = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} - & - & 1 & - & 1 \\ & - & - & 1 & 1 \\ & & - & - & - \\ & & & - & 1 \\ & & & & - \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

Que é equivalente ao produtório. Usando as leis e propriedades da álgebra booleana, tem-se:

$$(\overline{1} + \overline{2})(\overline{1} + \overline{4})(\overline{2} + \overline{3})(\overline{3} + \overline{4})(\overline{3} + \overline{5}) = 1$$

$(\overline{2} + \overline{3}) = (\overline{3} + \overline{2})$ (comutativa da adição)

$$(\overline{1} + \overline{2})(\overline{1} + \overline{4})(\overline{3} + \overline{2})(\overline{3} + \overline{4})(\overline{3} + \overline{5}) = 1$$

$$(\overline{1} + \overline{2})(\overline{1} + \overline{4}) = \overline{1}\cdot\overline{1} + \overline{1}\cdot\overline{4} + \overline{2}\cdot\overline{1} + \overline{2}\cdot\overline{4} = \overline{1} + \overline{1}\cdot\overline{4} + \overline{1}\cdot\overline{2} + \overline{2}\cdot\overline{4}) =$$
$$= \overline{1} + \overline{1}\cdot\overline{2} + \overline{2}\cdot\overline{4} = \overline{1} + \overline{2}\cdot\overline{4}$$

(idem para $(\overline{3} + \overline{2})(\overline{3} + \overline{4})$, ou seja, $(\overline{3} + \overline{2})(\overline{3} + \overline{4}) = (\overline{3} + \overline{2}\cdot\overline{4})$).

Assim, tem-se:

$$(\overline{1} + \overline{2}\cdot\overline{4})(\overline{3} + \overline{2}\cdot\overline{4})(\overline{3} + \overline{5}) = 1$$

$$(\overline{2}\cdot\overline{4} + \overline{1})(\overline{2}\cdot\overline{4} + \overline{3})(\overline{3} + \overline{5}) = 1$$

$$(\overline{2}\cdot\overline{4} + \overline{1}\cdot\overline{3})(\overline{3} + \overline{5}) = 1$$

$$\overline{2}\cdot\overline{4}\cdot\overline{3} + \overline{2}\cdot\overline{4}\cdot\overline{5} + \overline{1}\cdot\overline{3}\cdot\overline{3} + \overline{1}\cdot\overline{3}\cdot\overline{5} = 1$$

$$\overline{2}\cdot\overline{4}\cdot\overline{3} + \overline{2}\cdot\overline{4}\cdot\overline{5} + \overline{1}\cdot\overline{3} + \overline{1}\cdot\overline{3}\cdot\overline{5} = 1$$

$$\overline{2}\cdot\overline{3}\cdot\overline{4} + \overline{2}\cdot\overline{4}\cdot\overline{5} + \overline{1}\cdot\overline{3} = 1$$

Tomando o complementar de cada um dos termos acima,

$(\overline{1}, \overline{5})$ $(\overline{1}, \overline{3})$ $(\overline{2}, \overline{4}, \overline{5})$

tem-se as três facções maximais do grafo complementar $\overline{G}$, que são os conjuntos independentes maximais do grafo G:

$(1, 5)$ $(1, 3)$ $(2, 4, 5)$

4.4. - Método de Bednareck e Toulbee ([3], [61])

4.4.1. Base teórica do método

Considere o problema do descobrimento de todos as facções de um grafo. Se p é um vértice qualquer do grafo $\mathbb{G}$, então as facções podem ser divididos entre duas espécies; aquelas que contém p, e aquelas que não contém p.

Qualquer facção que contém p consiste de p juntamente com um conjunto $\mathbb{Y}$ de vértices do grafo $\mathbb{G} - p$ obtido pela eliminação de p do grafo $\mathbb{G}$.

Se $\mathbb{N}$ denotar o subgrafo de $\mathbb{G}$ induzido pela vizinhança (sucessores) de p, então $\mathbb{Y}$ pode ser uma facção em $\mathbb{N}$. Portanto todas as facções de $\mathbb{G}$ que contém p poderão ser obtidas pelo descobrimento das facções de $\mathbb{N}$.

Quanto às facções que não contém p, elas podem ser facções de $\mathbb{G} - p$, entretanto, o contrário não é verdadeiro. No meio das facções de $\mathbb{G} - p$ podem estar algumas que são facções de $\mathbb{N}$, e estas não são maximais em $\mathbb{G}$.

Por isso o conjunto de facções de $\mathbb{G}$ é a união dos dois conjuntos:

(i) o conjunto de facções de $\mathbb{N}$, cada uma delas aumentada por p;

(ii) o conjunto de facções de $\mathbb{G} - p$ que não são facções de $\mathbb{N}$.

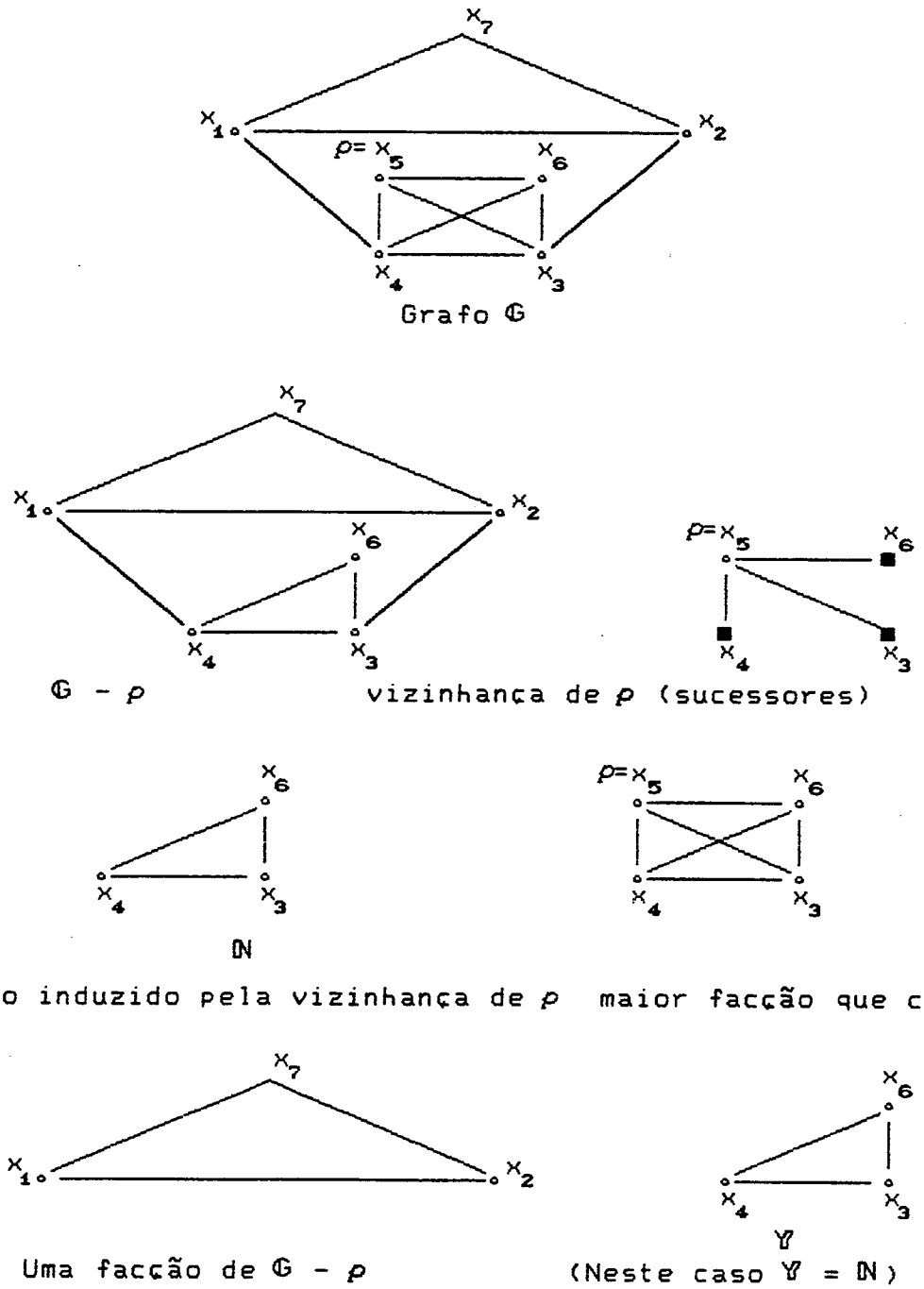
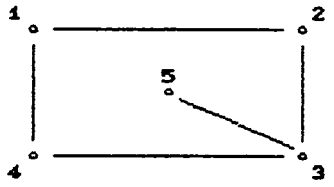


Figura 4.3. - Caracterização dos termos utilizados no método de Bednarek e Toulbee

Assim tem-se um típico problema onde deve-se aplicar um algoritmo de ramificação - a divisão do problema em dois problemas menores. Contudo, existem complicações aí; a condição

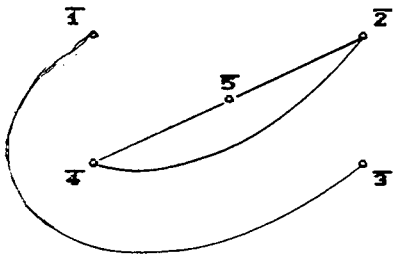
(ii) requer que a informação obtida durante uma parte do processo de ramificação seja usada na outra parte. Este método poderá funcionar, mas origina problemas de conservação e de armazenagem.

Exemplo 4.3: Desenvolvimento gráfico de um exemplo



$\Gamma(1) = \langle 2, 4 \rangle$
$\Gamma(2) = \langle 1, 3 \rangle$
$\Gamma(3) = \langle 2, 4, 5 \rangle$
$\Gamma(4) = \langle 1, 3 \rangle$
$\Gamma(5) = \langle 3 \rangle$

Figura 4.4. Grafo $\mathbb{G}$ para exemplo do método de Bednarek e Toulbee



$\Gamma(\overline{1}) = \langle \overline{3} \rangle$
$\Gamma(\overline{2}) = \langle \overline{4}, \overline{5} \rangle$
$\Gamma(\overline{3}) = \langle \overline{1} \rangle$
$\Gamma(\overline{4}) = \langle \overline{2}, \overline{5} \rangle$
$\Gamma(\overline{5}) = \langle \overline{1}, \overline{2}, \overline{4} \rangle$

Figura 4.5. - Grafo complementar $\overline{\mathbb{G}}$ do grafo $\mathbb{G}$ da Fig. 4.4.

O vértice escolhido é $p = x_{\overline{5}}$

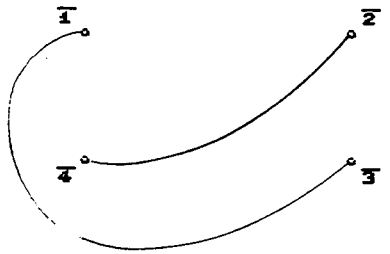


Figura 4.6. $\mathbb{G} - x_{\overline{5}}$

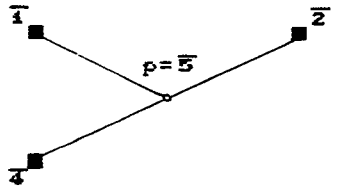


Figura 4.7. Vizinhança de $x_{\overline{5}}$ (sucessores de $x_{\overline{5}}$)

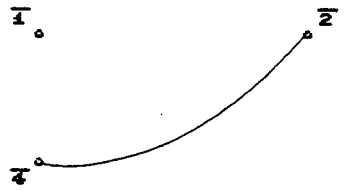


Figura 4.8. $\overline{\mathbb{N}}$ (subgrafo induzido pela vizinhança de x_5)

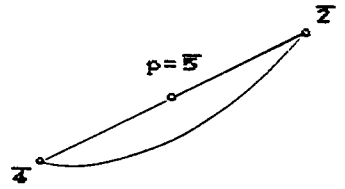


Figura 4.9. Maior facção que contém x_5

A maior facção que contém x_5, consiste de x_5 juntamente com $\overline{Y} = (x_2, x_4)$. $\overline{Y}$ é uma facção em $\overline{\mathbb{N}}$. Assim, facções que contém x_5, podem ser encontradas pelo descobrimento das facções de $\overline{\mathbb{N}}$. Por outro lado, as facções que não contém x_5, poderão ser facções de $\overline{\mathbb{G}} - x_5$.

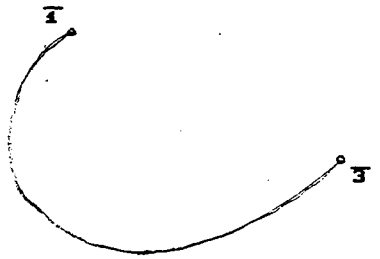


Figura 4.10. Facção de $\overline{\mathbb{G}} - x_5$

Nas facções de $\overline{\mathbb{G}} - x_5$ podem estar algumas que são facções de $\overline{\mathbb{N}}$.

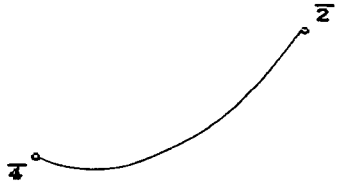


Figura 4.11. Facções de $\overline{\mathbb{G}} - x_5$ que são também facções de $\overline{\mathbb{N}}$

e estas não são maximais em $\overline{G}$. Por isso, o conjunto de facções de $\overline{G}$ é a união dos dois conjuntos:

(i) conjunto de facções de $\overline{N}$, cada uma das quais aumentada por $x_{\overline{5}}$;

Figura 4.12. Facções de $\overline{N}$, cada uma das quais aumentada por $x_{\overline{5}}$

(ii) conjunto de facções de $\overline{G} - x_{\overline{5}}$ que não são facções de $\overline{N}$.

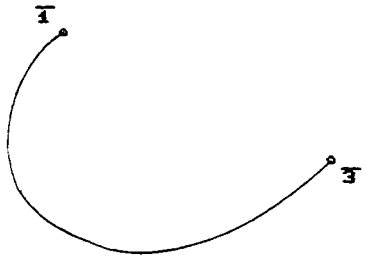


Figura 4.13. Facções de $\overline{G} - x_{\overline{5}}$ que não são facções de $\overline{N}$

Portanto, as facções de $\overline{G}$ são

$\{\overline{1}, \overline{3}\}$ $\{\overline{1}, \overline{5}\}$ $\{\overline{2}, \overline{4}, \overline{5}\}$

e seus complementares são exatamente os conj. indep. maximais.

$\{1, 5\}$ $\{1, 3\}$ $\{2, 4, 5\}$

Uma outra forma é trabalhar de baixo para cima - isto é, ir do menor para o maior grafo. Faça H_i ser o subgrafo induzido

pelos vértices 1, 2, 3, ..., i; e suponha que se tenha encontrado todas as suas facções. Agora toma-se o vértice i + 1. Encontra-se o conjunto $\mathbb{N}$ para este vértice, e encontramos sua interseção com as facções de $\mathbb{H}_i$. Estas serão as facções de $\mathbb{N}$, e, aumentadas pelo vértice i + 1, serão facções de $\mathbb{H}_{i+1}$. Aquelas facções de $\mathbb{H}_i$ que não estão contidas em $\mathbb{N}$ serão as facções de $\mathbb{H}_{i+1}$. Deste modo as facções de um dado grafo são eventualmente obtidas. Este algoritmo, essencialmente simples, é o de Bednarek e Taulbee [3].

Exemplo 4.4:

Figura 4.14. $\overline{\mathbb{H}}_{\overline{2}}$

$$\rho = i + 1 = \overline{3}$$

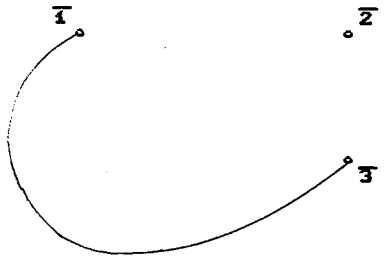


Figura 4.15. $\overline{\mathbb{H}}_{\overline{3}}$

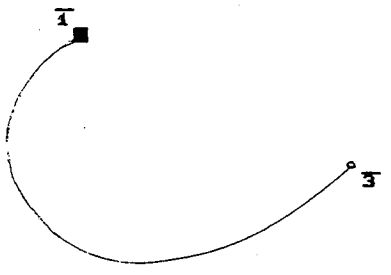


Figura 4.16. Vizinhança de $x_{\overline{3}}$

Figura 4.17. $\overline{\mathbb{N}}$

Figura 4.18. $\overline{\mathbb{N}} \cap$ (facções de $\overline{H}_2$)

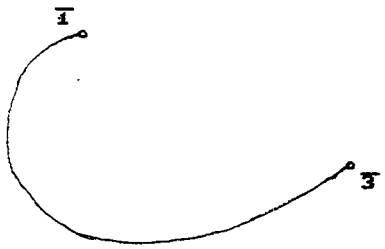


Figura 4.19. Facções da $\cap$ aumentadas por x_3

Figura 4.20. Facções de $\overline{H}_2$ não contidas em $\overline{\mathbb{N}}$ serão facções de $\overline{H}_3$

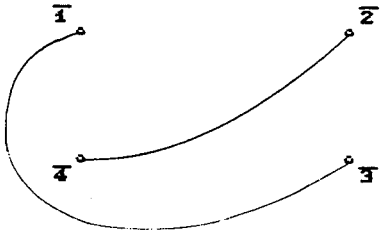


Figura 4.21. $\overline{H}_4$

$$\rho = i + 1 = \overline{4}$$

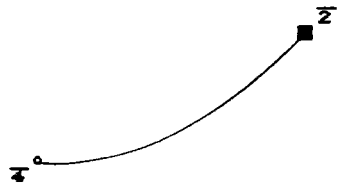


Figura 4.22. Vizinhança de $x_{\overline{4}}$

Figura 4.23. $\overline{\mathbb{N}}$

Figura 4.24. $\overline{\mathbb{N}} \cap$ (facções de $\overline{H}_{\overline{3}}$)

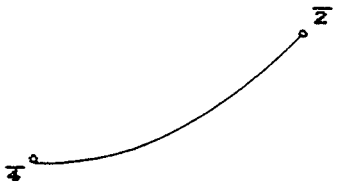


Figura 4.25. Facções da $\cap$ aumentadas por $x_{\overline{4}}$

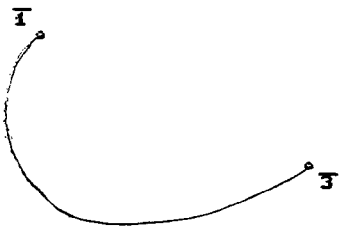


Figura 4.26. Facções de $\overline{H}_{\overline{3}}$ não contidas em $\overline{\mathbb{N}}$ serão facções de $\overline{H}_{\overline{4}}$

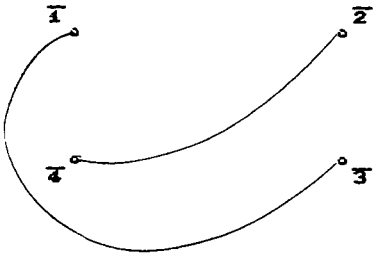


Figura 4.27. $\overline{H}_{\overline{4}}$

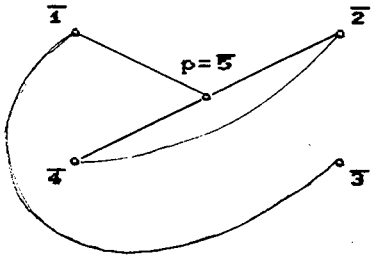


Figura. 4.28. $\overline{H}_{\overline{5}}$

$$p = i + 1 = \overline{5}$$

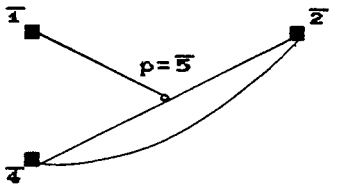


Figura 4.29. Vizinhança de $x_{\overline{5}}$

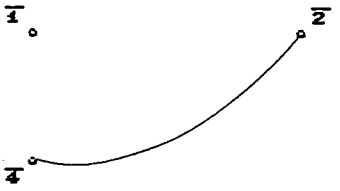


Figura 4.30. $\overline{N}$

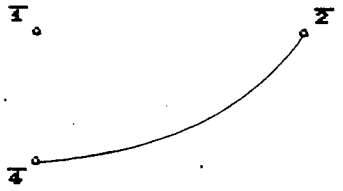


Figura 4.31. $\overline{N} \cap$ (facções de $\overline{H}_4$)

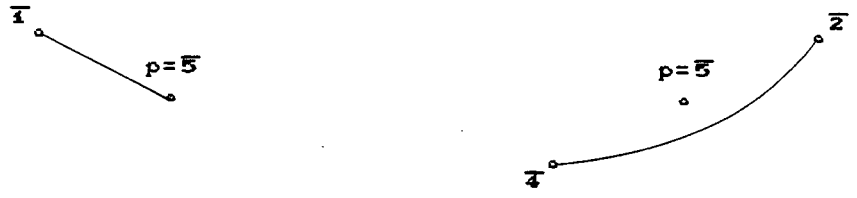


Figura 4.32. Facções da $\cap$ aumentadas por x_5

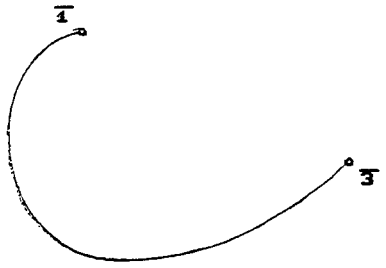


Figura 4.33. Facções de $\overline{H}_4$ que não estão contidas em $\overline{N}$, serão facções de $\overline{H}_5$

4.5. - Algoritmo de Bron e Kerbosch ([10], [13], [14])

4.5.1. - A base do algoritmo

4.5.1.1. - Introdução

Este algoritmo é essencialmente uma busca enumerativa simples em árvore, durante a qual em algum estágio k, um conjunto de vértices independentes $\mathbb{V}_k$ será ampliado pela adição de um vértice $x_{i_{k+1}}$ adequadamente escolhido, para formar um conjunto independente $\mathbb{V}_{k+1}$, no estágio k, até que mais nenhuma adição seja possível e, por isso, tem-se um novo conjunto independente maior. No estágio k toma-se, por exemplo, $\mathbb{E}_k$ como sendo o maior conjunto de vértices para o qual $\mathbb{V}_k \cap \mathbb{E}_k = \emptyset$, isto é, qualquer vértice de $\mathbb{E}_k$ acrescentado a $\mathbb{V}_k$ resultará num conjunto $\mathbb{V}_{k+1}$ que será independente.

Em algum ponto durante o desenvolvimento do algoritmo, $\mathbb{E}_k$ consistirá de dois tipos de vértices: vértices em $\mathbb{E}_k^-$ que já foram usados anteriormente na busca para aumentar $\mathbb{V}_k$ e vértices em $\mathbb{E}_k^+$ que ainda não foram usados.

4.5.1.2. - **Avanço**

Uma ramificação à frente durante a busca na árvore implicará na escolha de um vértice $x_{i_{k+1}} \in \mathbb{E}_k^+$, e na sua adição a $\mathbb{V}_k$, obtendo-se desta forma:

$$\mathbb{V}_{k+1} = \mathbb{V}_k \cup \{x_{i_{k+1}}\} \qquad (3.4)$$

juntamente com os novos conjuntos $\mathbb{E}_{k+1}$ conforme:

$$\mathbb{E}^{+}_{k+1} = \mathbb{E}^{+}_{k} - \{x_{i_{k+1}}\} - \Gamma(x_{i_{k+1}}) = \mathbb{E}^{+}_{k} - \left\{\{x_{i_{k+1}}\} \cup \Gamma(x_{i_{k+1}})\right\} \qquad (3.5)$$

e

$$\mathbb{E}^{-}_{k+1} = \mathbb{E}^{-}_{k} - \Gamma(x_{i_{k+1}}) \qquad (3.6)$$

4.5.1.3. - **Recuo** *"Backtrack"*

Um recuo (*"backtrack"*) implica na remoção de $x_{i_{k+1}}$ de $\mathbb{V}_{k+1}$ para retornar a $\mathbb{V}_{k}$, removendo também $x_{i_{k+1}}$ do conjunto $\mathbb{E}^{+}_{k}$ para então inseri-lo no conjunto $\mathbb{E}^{-}_{k}$, formando deste modo dois novos conjuntos $\mathbb{E}^{+}_{k}$ e $\mathbb{E}^{-}_{k}$. ($\mathbb{E}^{+}_{k}$ agora sem $x_{i_{k+1}}$ e $\mathbb{E}^{-}_{k}$ agora com $x_{i_{k+1}}$).

Um conjunto $\mathbb{V}_{k}$, somente não poderá ser mais aumentado quando $\mathbb{E}^{+}_{k} = \emptyset$. Neste caso:

(*i*) Se $\mathbb{E}^{-}_{k} \neq \emptyset$, decorre imediatamente que no nível k atual, $\mathbb{V}_{k}$ teria, num nível anterior (k - 1), sido aumentado por algum vértice que agora se encontra em $\mathbb{E}^{-}_{k}$ e por isso não é maximal. Desta forma tem-se que efetuar um recuo.

(*ii*) Se $\mathbb{E}^{-}_{k} = \emptyset$, o atual conjunto $\mathbb{V}_{k}$, não teria sido aumentado anteriormente e, visto que os conjuntos são gerados sem duplicação, $\mathbb{V}_{k}$ será um conjunto independente maximal. Portanto, a condição necessária e suficiente para $\mathbb{V}_{k}$ ser um conjunto independente maximal é:

$$\mathbb{E}^{-}_{k} = \mathbb{E}^{+}_{k} = \emptyset \qquad (3.7)$$

É evidente que jamais será acessado o estágio k + 1 quando existir algum vértice $\hat{x} \in \mathbb{E}_k^-$ e, para o qual $\Gamma(\hat{x}) \cap \mathbb{E}_k^+ = \emptyset$, assim é indiferente qual vértice de $\mathbb{E}_k^+$ escolher para aumentar $\mathbb{V}_k$ já que em qualquer ramificação à frente, o vértice $\hat{x}$ nunca poderá ser removido de $\mathbb{E}_{k+1}^-$ pois, $\mathbb{E}_{k+1}^- = \mathbb{E}_k^- - \Gamma(x_{i_{k+1}}) = \mathbb{E}_{k+1}^-$, ou seja, $\Gamma(x_{i_{k+1}}) \cap \mathbb{E}_k^- = \emptyset$. Por isso, a condição:

$$\exists\ \hat{x} \in \mathbb{E}_k^- \text{ tal que } \Gamma(\hat{x}) \cap \mathbb{E}_k^+ = \emptyset \qquad (3.8)$$

será suficiente para que um recuo deva ser efetuado, pois nenhum conjunto independente maximal resultará de qualquer ramificação à frente de $\mathbb{V}_k$.

4.5.1.4. - **Heurística**

Como em todos os métodos de busca em árvore, é vantagem ter como meta antecipar recuos em ramos que não levam a conjuntos independentes maximais, já que estes recuos tendem para a parte desnecessária da busca. é importante, então, forçar a condição (3.8) tanto quanto possível por uma escolha adequada dos vértices que serão usados na ampliação do conjunto $\mathbb{V}_k$.

Durante o avanço, pode-se escolher para $x_{i_{k+1}}$, qualquer vértice pertencente a $\mathbb{E}_k^+$ e com o qual aumentar-se $\mathbb{V}_k$, obtendo-se, assim $\mathbb{V}_{k+1}$. Já no recuo, o vértice escolhido $x_{i_{k+1}}$ deve ser removido de $\mathbb{E}_k^+$ e inserido em $\mathbb{E}_k^-$. Porém se $x_{i_{k+1}}$ for escolhido dentre os vértices pertencentes à $\Gamma(x^*)$ para algum $x^* \in \mathbb{E}_k^-$, então quando ocorrer o teste para verificar se deve ser efetuado um

avanço ou um recuo, o número:

$$\Delta(x^*) = |\Gamma(x^*) \cap \mathbb{E}_k^+| \qquad (3.9)$$

poderá ser reduzido (a partir de seu valor anterior ao término dos passos de avanço e recuo) tal que a condição (3.8) agora seja satisfeita, devendo, portanto, efetuar-se o recuo.

Portanto, uma boa maneira de escolher o vértice $x_{i_{k+1}}$ e com o qual aumentar-se $\mathbb{V}_k$, é primeiro determinar o vértice $x^* \in \mathbb{E}_k^-$ com o menor valor de Δ e depois escolher $x_{i_{k+1}}$ a partir do conjunto $\Gamma(x^*) \cap \mathbb{E}_k^+$. Tal escolha de $x_{i_{k+1}}$ no nível k, poderia ocasionar a Δ uma diminuição quando for feito o teste para o recuo, até que eventualmente um vértice x^* satisfaça a condição (3.8), devendo efetuar-se, portanto, um recuo.

Como num recuo, $x_{i_{k+1}}$ entra em $\mathbb{E}_k^-$, pode ser que esta nova entrada tenha, agora, um valor menor de Δ do que o vértice x^* fixado anteriormente. Esta nova entrada poderá forçar a condição (3.8). Isto é importante ao efetuar-se o primeiro avanço nas ramificaçoes, quando $\mathbb{E}_k^- \neq \emptyset$, pois neste caso, deve-se utilizar a heurística proposta, que está baseada em $\mathbb{E}_k^-$.

4.5.2. - Descrição do algoritmo

Inicialização

Passo 1:

Faça

$l = 1$

$k = \emptyset$

$\mathbb{V}_k^l = \emptyset$

$\mathbb{E}_k^+ = \mathbb{X}$

$\mathbb{E}_k^- = \emptyset$

Escolha

Passo 2:

Escolha um vértice $x_{i_{k+1}} \in \mathbb{E}_k^+$ conforme a seguinte heurística:

se $\mathbb{E}_k^- \neq \emptyset$

então

se $\exists ! \; x \in \mathbb{E}_k^-$

então

escolhemos algum $x_{i_{k+1}} \in \Gamma(x) \cap \mathbb{E}_k^+$

senão

selecionamos x^* de modo que:

$$\Delta(x^*) = |\Gamma(x^*) \cap \mathbb{E}_k^+| = \min_{x \in \mathbb{E}_k^-} |\Gamma(x) \cap \mathbb{E}_k^+|$$

e em seguida escolhemos $x_{i_{k+1}} \in \Gamma(x^*) \cap \mathbb{E}_k^+$

senão

escolha o primeiro $x_{i_t} \in \mathbb{E}_k^+$

Avanço

Passo 3:

$$\mathbb{V}^{l}_{k+1} = \mathbb{V}^{l}_{k} \cup \{x_{i_{k+1}}\}$$

$$\mathbb{E}^{+}_{k+1} = \mathbb{E}^{+}_{k} - \{x_{i_{k+1}}\} - \Gamma(x_{i_{k+1}}) = \mathbb{E}^{+}_{k} - \left\{\{x_{i_{k+1}}\} \cup \Gamma(x_{i_{k+1}})\right\}$$

$$\mathbb{E}^{-}_{k+1} = \mathbb{E}^{-}_{k} - \Gamma(x_{i_{k+1}})$$

$k = k + 1$

Teste (verifica se o conjunto gerado é independente maximal)

Passo 4:

se $(\mathbb{E}^{-}_{k} = \mathbb{E}^{+}_{k} = \emptyset)$

então

Guarde o **CIM** $\mathbb{V}^{l}_{k}$

$l = l + 1$

e vá ao **Passo 6:**

Teste (avanço ou recuo)

Passo 5:

se $\mathbb{E}^{+}_{k} \neq \emptyset$

então

se $\mathbb{E}^{-}_{k} = \emptyset$

então

vá ao **Passo 2:**

senão

se $\nexists\ x \in \mathbb{E}^{-}_{k}$ tal que $\Gamma(x) \cap \mathbb{E}^{+}_{k} = \emptyset$

então

vá ao **Passo 2:**

Recuo (backtrack)

Passo 6:

$k = k - 1$

se $k \neq -1$

então

$$\mathbb{V}_k^l = \mathbb{V}_{k+1}^l - (x_{i_{k+1}})$$

$$\mathbb{E}_k^+ = \mathbb{E}_k^+ - (x_{i_{k+1}})$$

$$\mathbb{E}_k^- = \mathbb{E}_k^- \cup (x_{i_{k+1}})$$

se $(k \neq \emptyset$ ou $\mathbb{E}_k^+ \neq \emptyset)$

então

vá ao Passo 5.

senão

TERMINOU A BUSCA

(Todos os **MCIM** foram encontrados)

senão

TERMINOU A BUSCA

(Todos os **MCIM** foram encontrados)

4.5.3. - Desenvolvimento ilustrativo de um exemplo utilizando-se o algoritmo passo a passo

LEGENDA

∘ **VÉRTICES PARA ESCOLHER**

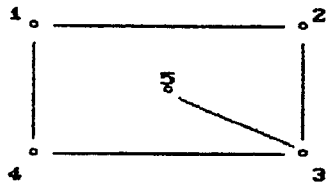
⊗ **VÉRTICES ESCOLHIDOS**

■ **SUCESSORES DOS VÉRTICES ESCOLHIDOS**

ℵ **VÉRTICES RETIRADOS NO NÍVEL 0**

□ **VÉRTICES RETIRADOS NO NÍVEL 1**

⊖ **VÉRTICES RETIRADOS NO NÍVEL 2**



$\Gamma(1) = \langle 2, 4 \rangle$
$\Gamma(2) = \langle 1, 3 \rangle$
$\Gamma(3) = \langle 2, 4, 5 \rangle$
$\Gamma(4) = \langle 1, 3 \rangle$
$\Gamma(5) = \langle 3 \rangle$

Figura 4.34. Grafo $\mathcal{G}$ para exemplo do algoritmo de Bron e Kerbosch

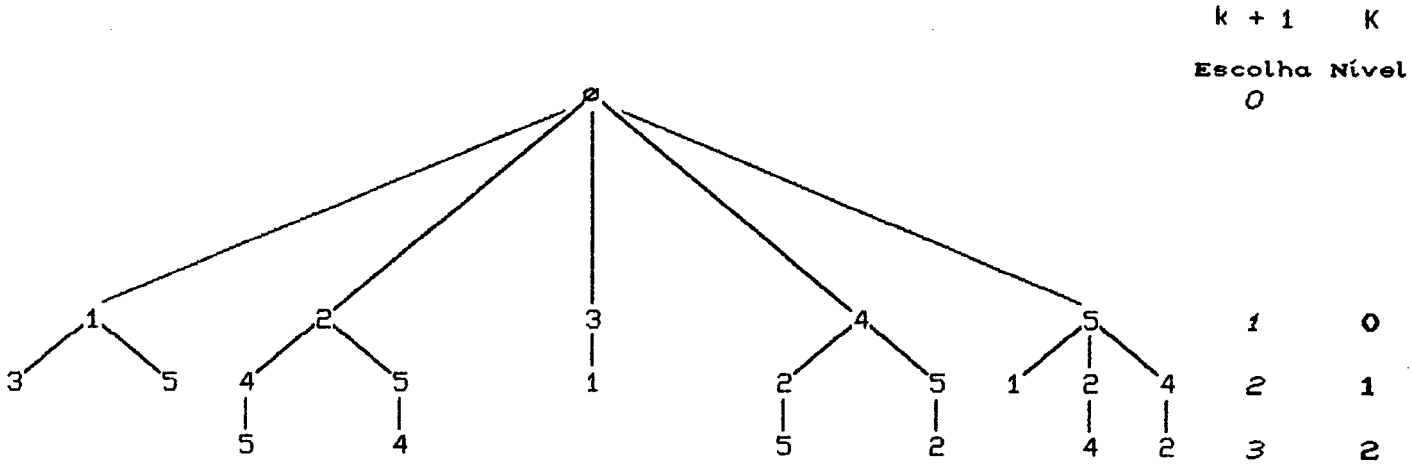


Figura 4.35. Árvore com todas as possíveis soluções

P1.

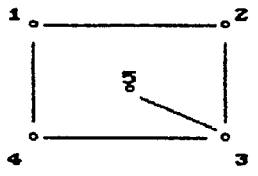
$l = 1$

$k = \emptyset$ ø

$V_o^1 = \emptyset$

$E_o^+ = X = (1, 2, 3, 4, 5)$

$E_o^- = \emptyset$



P2.

Não

$x_{i_1} = 1$

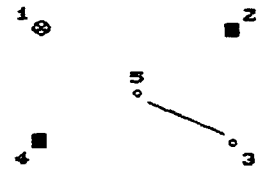
P3.

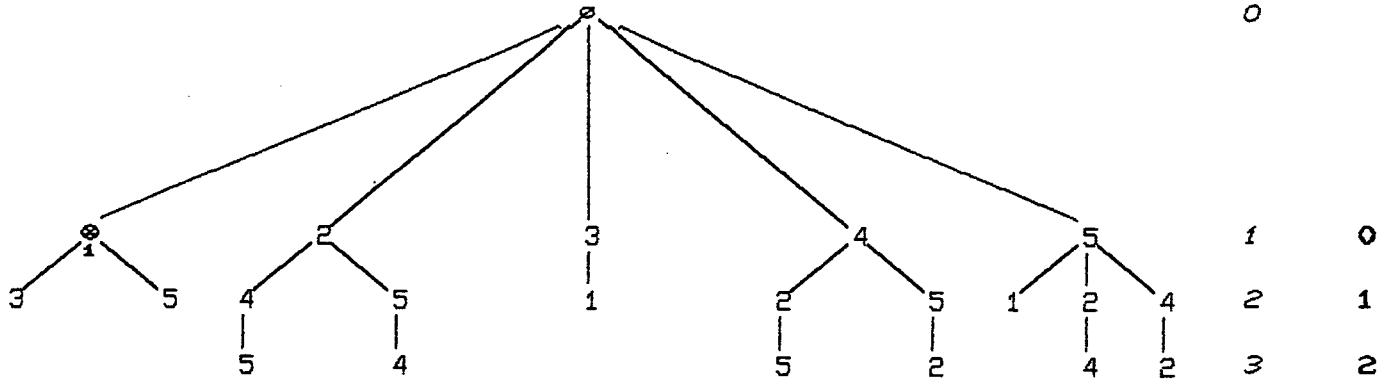
$V_1^1 = (1)$

$E_1^+ = (3, 5)$

$E_1^- = \emptyset$

$k = 1$





P4. Não $E_1^+ \neq \emptyset$ e $E_1^- = \emptyset$

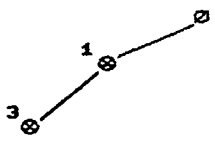
P5. Sim $E_1^+ \neq \emptyset$

Sim $E_1^- = \emptyset$

P2.

Não

$x_{i_2} = 3$



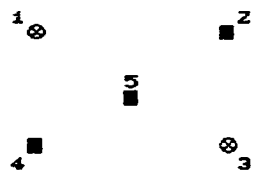
P3.

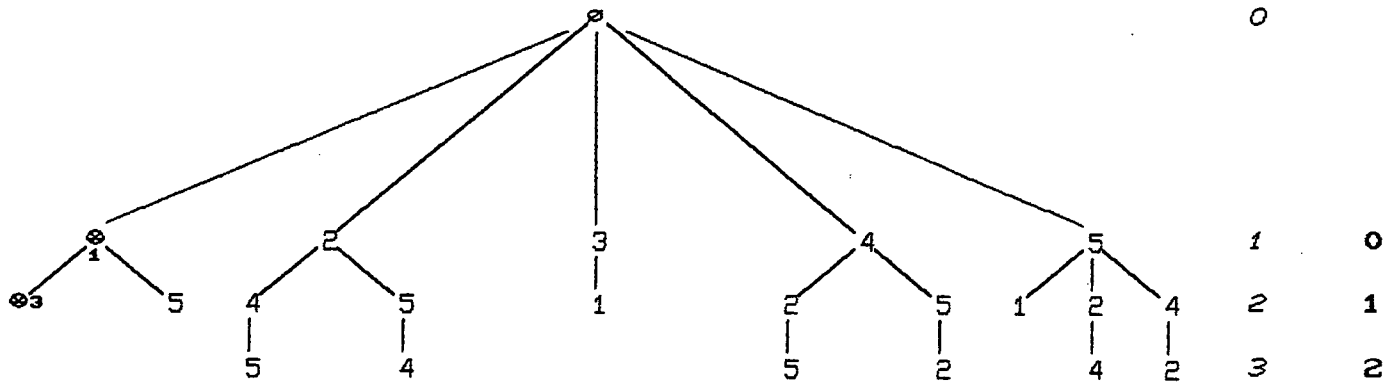
$V_2^1 = (1, 3)$

$E_2^+ = \emptyset$

$E_2^- = \emptyset$

$k = 2$





P4. Sim

$\mathbb{V}_2^1 = (1, 3)$

$1 = 2$

$\mathbb{E}_2^+ = \emptyset$ e $\mathbb{E}_2^- = \emptyset$

P6.

$k = 1$

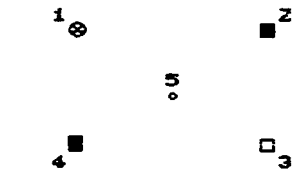
Sim

$\mathbb{V}_1^2 = (1)$

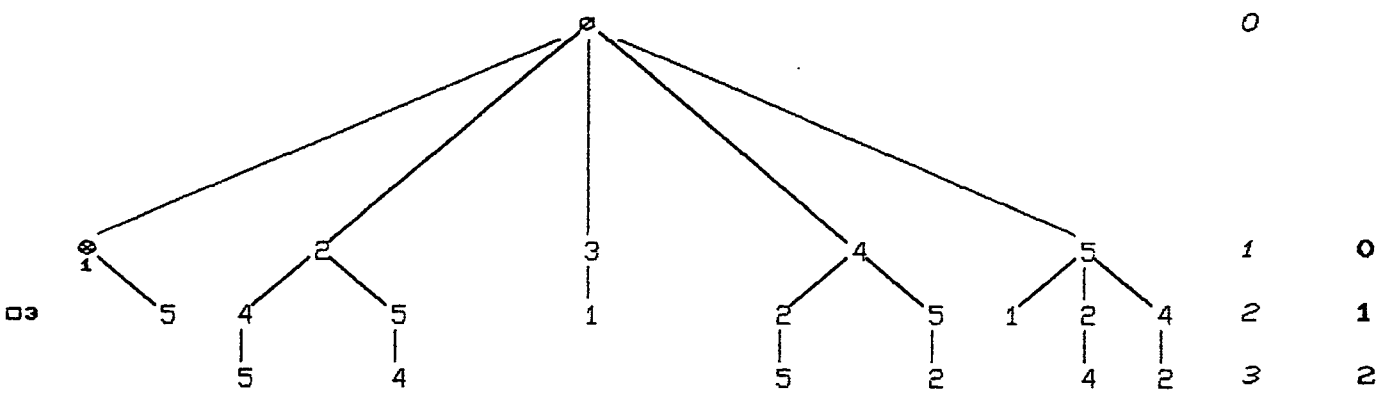
$\mathbb{E}_1^+ = (5)$

$\mathbb{E}_1^- = (3)$

Sim



$k = 1$ e $\mathbb{E}_1^+ = (5)$



P5. Sim $\quad \mathbb{E}_1^+ \neq \emptyset$

Não $\quad \mathbb{E}_1^- = \emptyset$

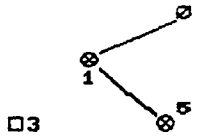
Sim $\quad \nexists\ x \in \mathbb{E}_1^-$ tq $\Gamma(x) \cap \mathbb{E}_1^+ = \emptyset$

P2.

Sim

Sim

$x_{i_2} = 5$



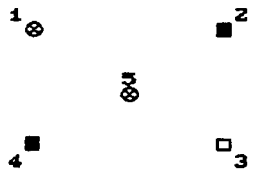
P3.

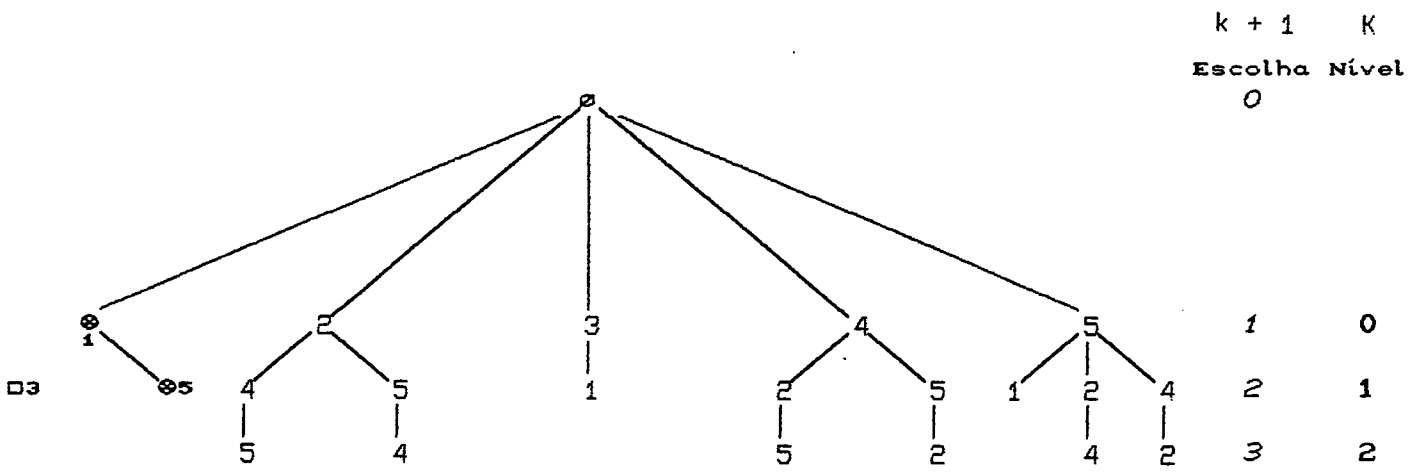
$\mathbb{V}_2^2 = (1, 5)$

$\mathbb{E}_2^+ = \emptyset$

$\mathbb{E}_2^- = \emptyset$

$k = 2$



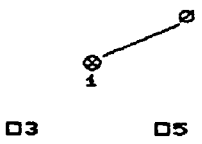


P4. Sim $\quad \mathbb{E}_2^+ = \emptyset$ e $\mathbb{E}_2^- = \emptyset$

$\boxed{\mathbb{V}_2^2 = (1, 5)}$

$l = 3$

P6.



k = 1

Sim

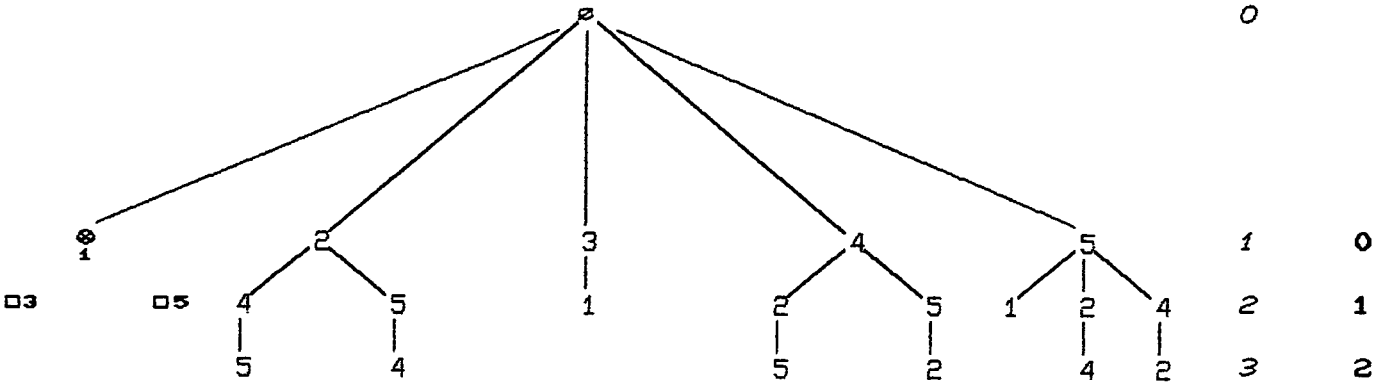
$\mathbb{V}_1^3 = \{1\}$

$\mathbb{E}_1^+ = \emptyset$

$\mathbb{E}_1^- = \{3, 5\}$

Sim

k = 1 e $\mathbb{E}_1^+ = \emptyset$



P5. Não

$\mathbb{E}_1^+ = \emptyset$

P6.

k = 0

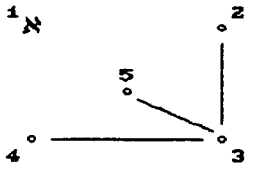
Sim

$\mathbb{V}_0^3 = \emptyset$

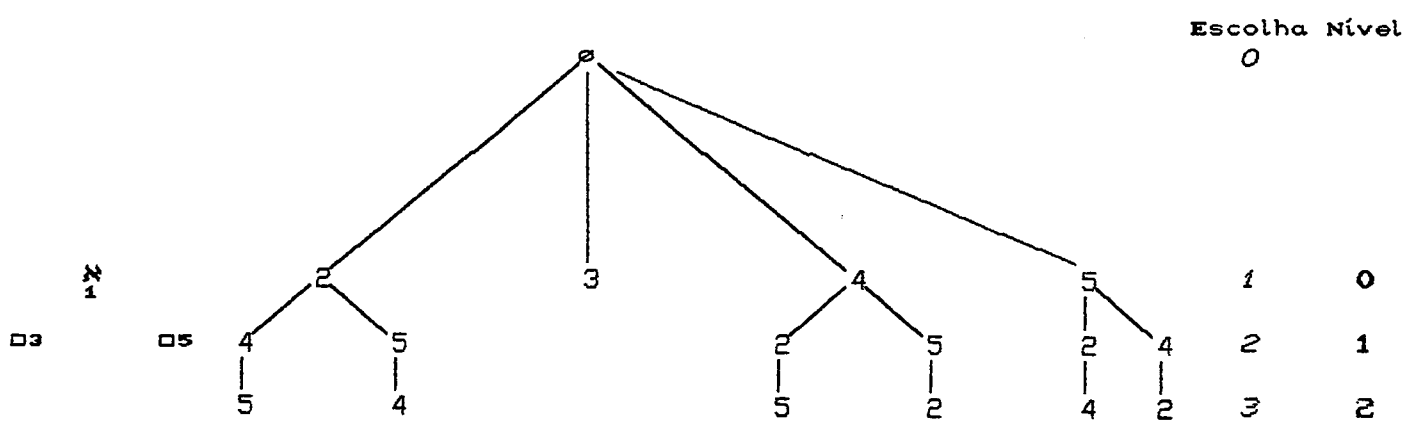
$\mathbb{E}_0^+ = \{2, 3, 4, 5\}$

$\mathbb{E}_0^- = \{1\}$

Sim



k = 0 e $\mathbb{E}_2^+ \neq \emptyset$



P5. Sim $\quad \mathbb{E}_o^+ \neq \emptyset$

Não $\quad \mathbb{E}_o^- \neq \emptyset$

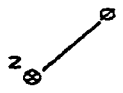
Sim $\quad \nexists\ x \in \mathbb{E}_o^-$ tq $\Gamma(x) \cap \mathbb{E}_o^+ = \emptyset$

P2.

Sim

Sim

$x_{i_1} = 2$



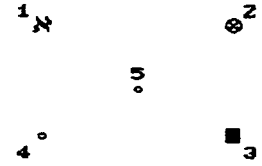
P3.

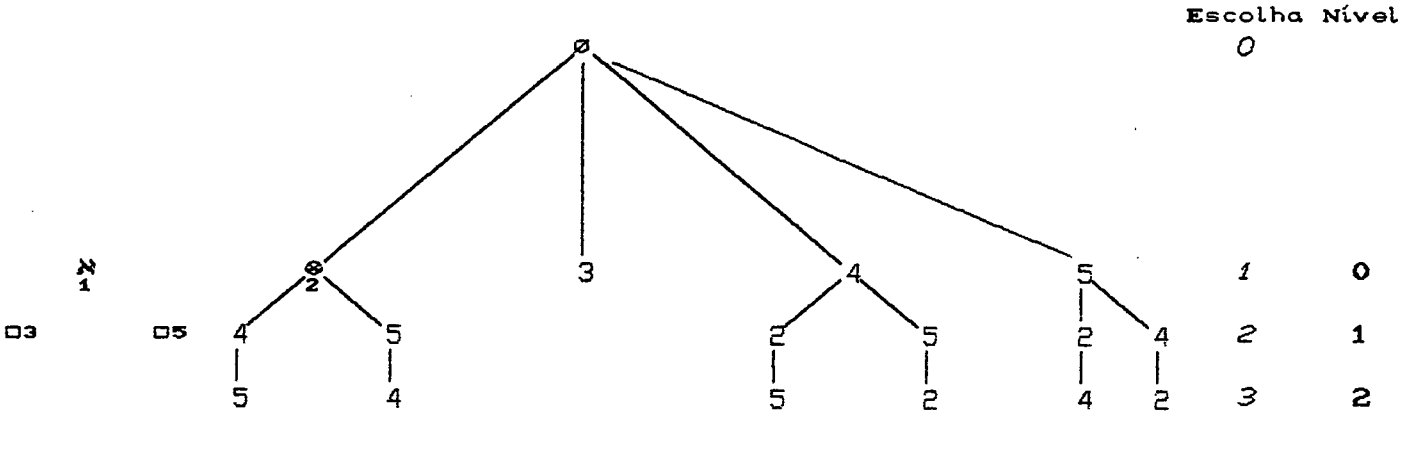
$\mathbb{V}_1^3 = (2)$

$\mathbb{E}_1^+ = (4, 5)$

$\mathbb{E}_1^- = \emptyset$

$k = 1$





P4. Não $\quad \mathbb{E}_1^+ \neq \emptyset$ e $\mathbb{E}_1^- = \emptyset$

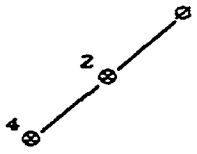
P5. Sim $\quad \mathbb{E}_1^+ \neq \emptyset$

Sim vá ao P2. $\quad \mathbb{E}_1^- = \emptyset$

P2.

Não

$x_{i_2} = 4$



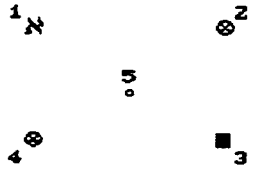
P3.

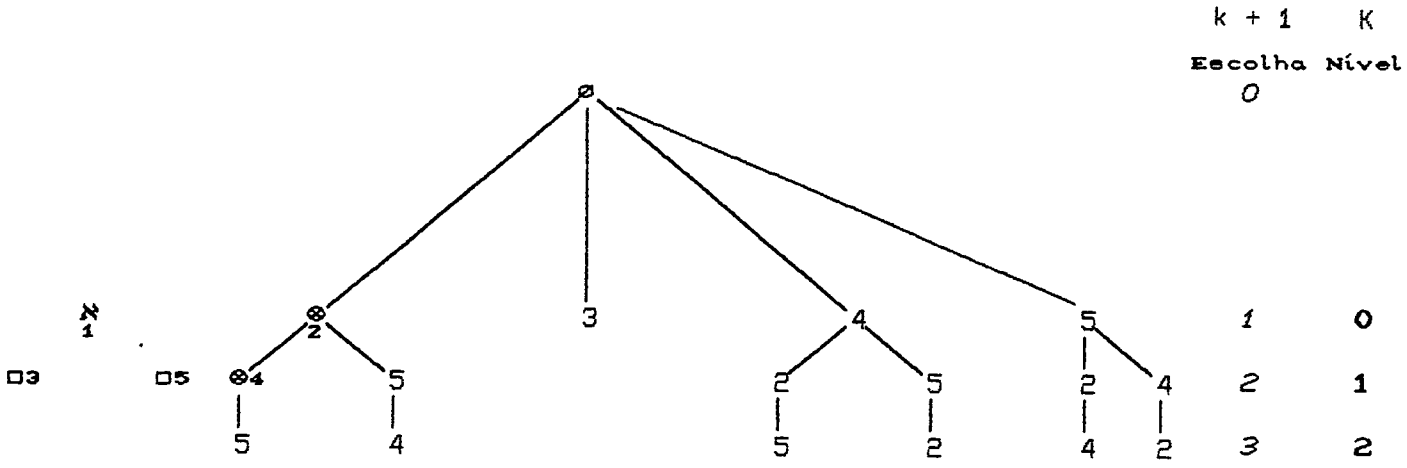
$\mathbb{V}_2^3 = (2, 4)$

$\mathbb{E}_2^+ = (5)$

$\mathbb{E}_2^- = \emptyset$

$k = 2$





P4. Não $\quad \mathbb{E}_2^+ \neq \emptyset$ e $\mathbb{E}_2^- = \emptyset$

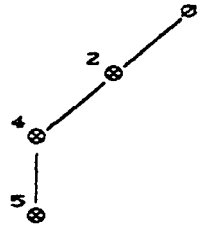
P5. Não $\quad \mathbb{E}_2^+ \neq \emptyset$

Sim vá ao P2. $\quad \mathbb{E}_2^- \neq \emptyset$

P2.

Não

$x_{i_3} = 5$



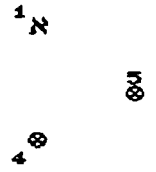
P3.

$V_3^3 = (2, 4, 5)$

$E_3^+ = \emptyset$

$E_3^- = \emptyset$

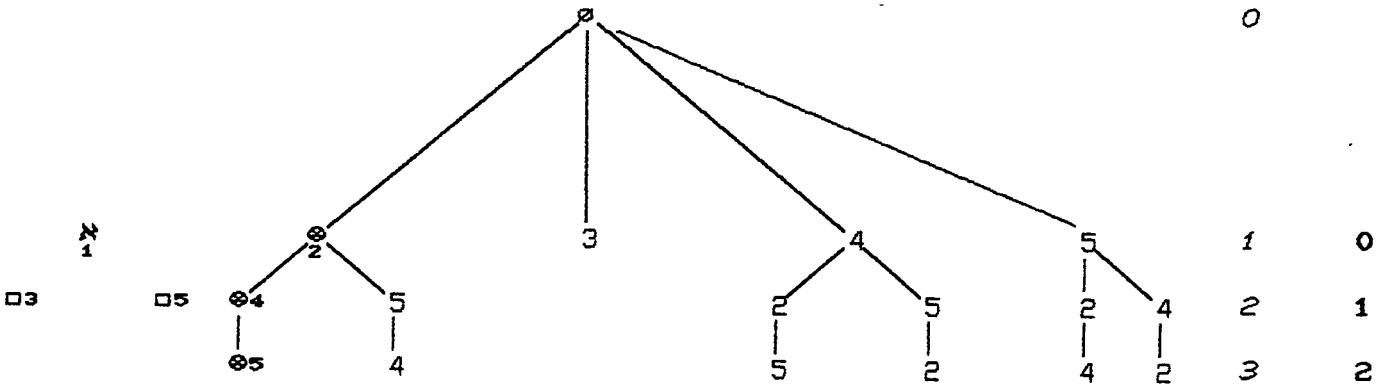
$k = 3$



k + 1 K

Escolha Nível

0



P4. Sim

$E_3^+ = \emptyset$ e $E_3^- = \emptyset$

$\boxed{V_3^3 = (2, 4, 5)}$

$l = 4$

P5.

$k = 2$

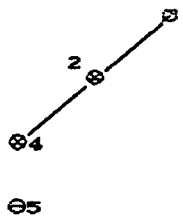
Sim

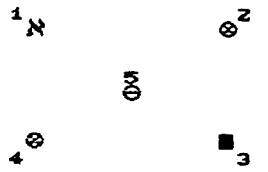
$V_2^4 = (2, 4)$

$E_2^+ = \emptyset$

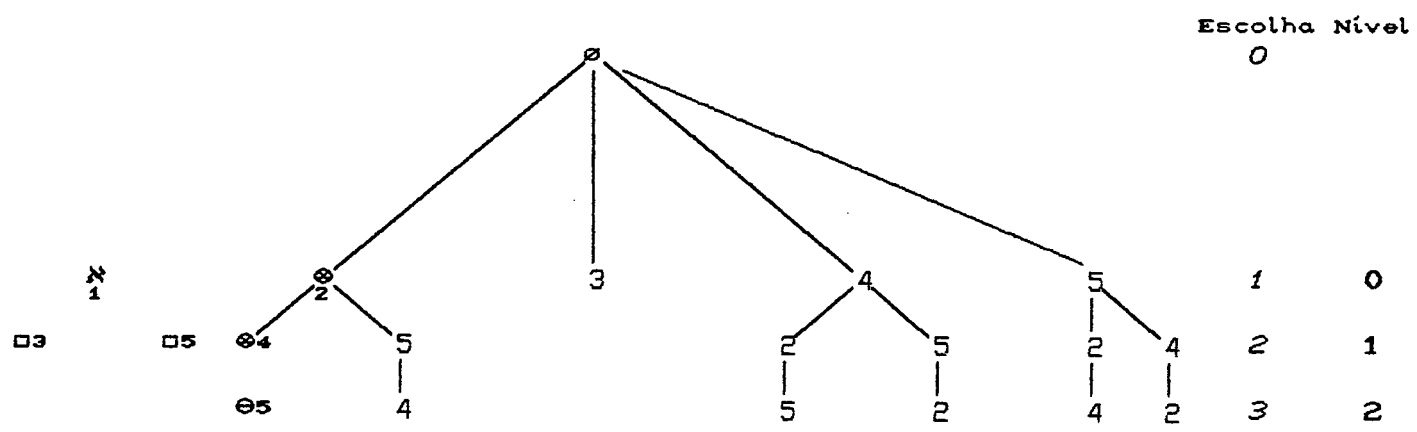
$E_2^- = (5)$

Sim





$k = 2$ e $E_2^+ = \emptyset$



P5. Não $\quad \mathbb{E}_2^+ = \emptyset$

P6.

$k = 1$

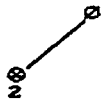
Sim

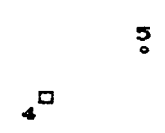
$\mathbb{V}_1^4 = (2)$

$\mathbb{E}_1^+ = (5)$

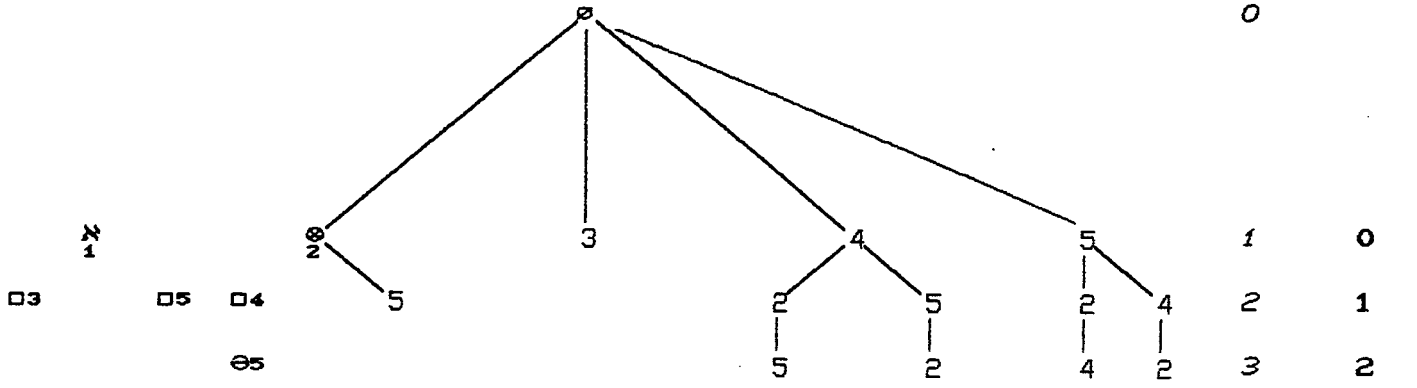
$\mathbb{E}_1^- = (4)$

Sim





$k = 1$ e $\mathbb{E}_1^+ = (5)$



P5. Sim $\quad \mathbb{E}_1^+ \neq \emptyset$

Não $\quad \mathbb{E}_1^- \neq \emptyset$

Não $\quad \exists\ x = 4 \in \mathbb{E}_1^-$ tq $\Gamma(4) \cap \mathbb{E}_1^+ = \emptyset$

$\langle 1, 3 \rangle \cap \langle 5 \rangle = \emptyset$

P6.

$k = 0$

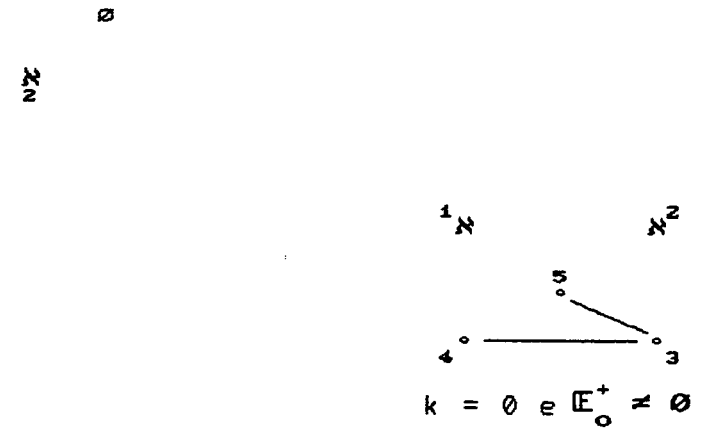
Sim

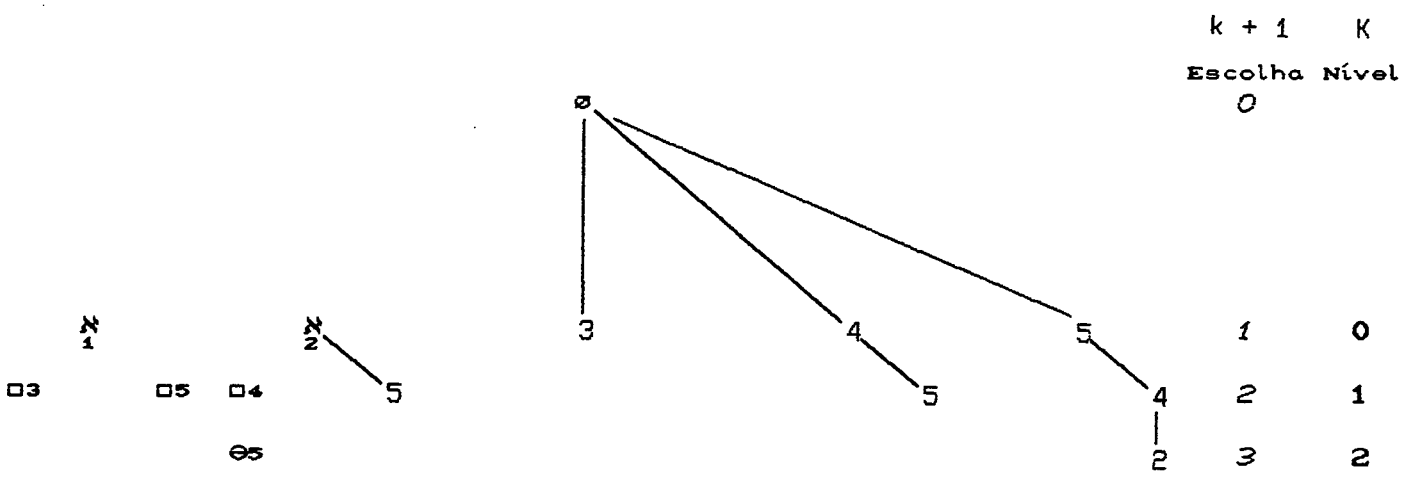
$V_o^4 = \emptyset$

$E_o^+ = (3, 4, 5)$

$E_o^- = (1, 2)$

Sim





P5. Sim $E_o^+ \neq \emptyset$

Não $E_o^- \neq \emptyset$

Sim $\nexists\ x \in E_o^-$ tq $\Gamma(x) \cap E_o^+ = \emptyset$

P2.

Sim

Não

$x_{i_1} = 4$

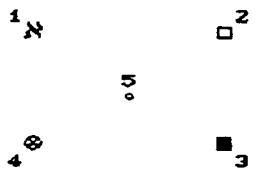
P3.

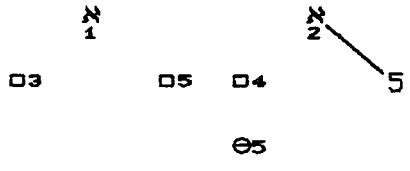
$V_1^4 = (4)$

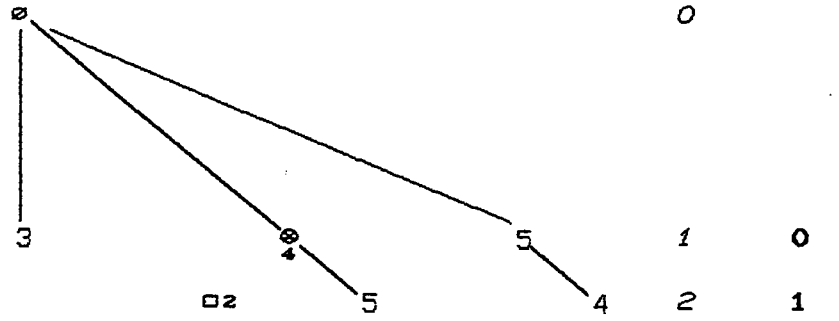
$E_1^+ = (5)$

$E_1^- = (2)$

$k = 1$







P4. Não $\quad \mathbb{E}_1^+ \neq \emptyset \text{ e } \mathbb{E}_1^- \neq \emptyset$

P5. Sim $\quad \mathbb{E}_1^+ \neq \emptyset$

Não $\quad \mathbb{E}_1^- \neq \emptyset$

Não $\quad \exists\ x = 2 \in \mathbb{E}_1^- \text{ tq } \Gamma(2) \cap \mathbb{E}_1^+ = \emptyset$

$\langle 1, 3 \rangle \cap \langle 5 \rangle = \emptyset$

P6.

$k = 0$

Sim

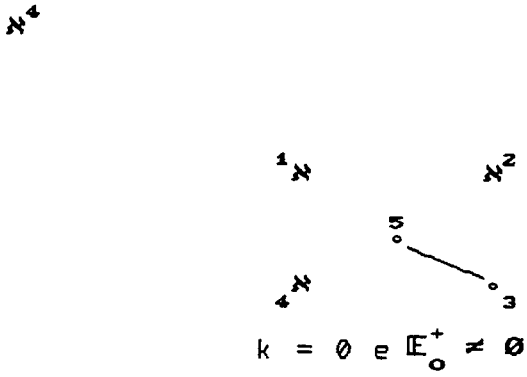
$\mathbb{V}_o^4 = \emptyset$

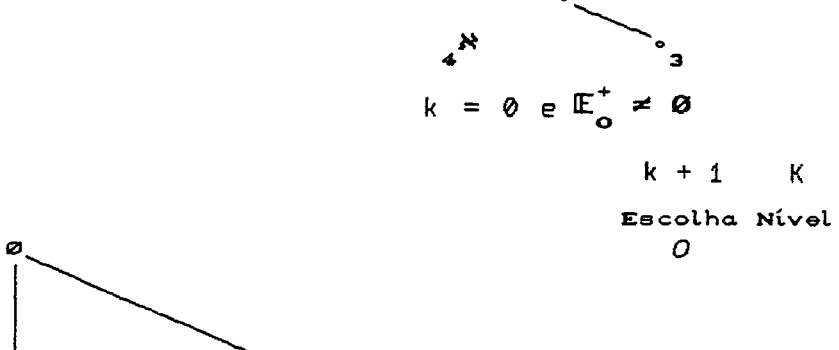
$\mathbb{E}_o^+ = \{3, 5\}$

$\mathbb{E}_o^- = \{1, 2, 4\}$

Sim

ø





P5. Sim $\quad \mathbb{E}_o^+ \neq \emptyset$

Não $\quad \mathbb{E}_o^- \neq \emptyset$

Não $\quad \exists\ x = 1 \in \mathbb{E}_o^-$ tq $\Gamma(1) \cap \mathbb{E}_o^+ = \emptyset$

$\langle 2, 4 \rangle \cap \langle 3, 5 \rangle = \emptyset$

P6.

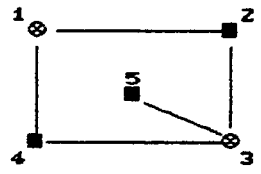
$k = -1$.

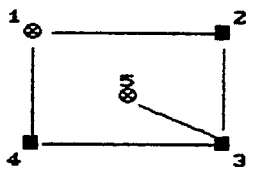
Não

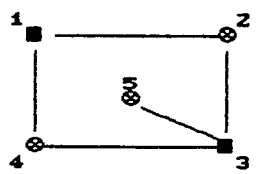
TERMINOU A BUSCA

(Todos os conj. indep. max. foram encontrados)

$\mathbb{V}_2^1 = \{1, 3\}$ $\quad \mathbb{V}_2^2 = \{1, 5\}$ $\quad \mathbb{V}_3^3 = \{2, 4, 5\}$







CAPÍTULO V

PROPOSTA DE UM ALGORITMO HEURÍSTICO PARA OBTENÇÃO DE UMA SOLUÇÃO PARA O MÁXIMO CONJUNTO INDEPENDENTE

5.1. - Introdução

Na área de algoritmos em grafos (assim como em algoritmos de um modo geral) é inerente uma preocupação computacional para com os processos desenvolvidos. Essa preocupação, implica no objetivo de procurar elaborar algoritmos que sejam eficientes e que, de preferência, possuam complexidade polinomial.

Segundo Ronaldo C. Read [61]: "Desde a publicação do algoritmo de Bednarek e Toulbee [3], vários outros algoritmos para o descobrimento de facções (cliques) tem aparecido. Parece que em quaisquer desses algoritmos é necessário, até certo ponto, checar se alguns subgrafos completos gerados estão contidos num subgrafo já gerado. Os algoritmos diferem sobretudo no exito obtido na redução do tempo gasto. O melhor algoritmo fornecido até agora para descobrir todas as facções é o de Bron e Kerbosch [10]".

5.1.1. - Problemas do algoritmo de Bron e Kerbosch

Considerado por muitos como o melhor algoritmo para selecionar os conjuntos independentes maximais, o método de Bron e Kerbosch, entretanto, não é bom para determinar o número de

independência, bem como um dos conjuntos determinado por ele (se é que existe mais de um), pois apresenta dificuldades tais como:

a) complexidade exponencial $\Theta(exp(n))$;

b) tem que gerar todos os conjuntos independentes maximais $\mathbb{V}_k^l$ do grafo;

c) o elevado número (l) de conjuntos independentes maximais ($\mathbb{V}_k^l$) gerados, a medida que aumenta-se o número de vértices do grafo;

d) aleatoriedade na geração dos conjuntos independentes maximais no que diz respeito à sua cardinalidade;

e) compara conjuntos entre si, isto é, dois a dois, todos os conjuntos independentes maximais. Para fazer isso além de ter que gerar todos os conjuntos independentes maximais, tem-se que compará-los e também vasculhar todos os conjuntos até o fim da busca enumerativa;

f) armazena dois subconjuntos $\mathbb{E}_k^-$ e $\mathbb{E}_k^+$ a cada nível k, onde k, indica o tamanho do conjunto independente $\mathbb{V}_k^l$ que se está gerando. Além disso, aumentam conforme aumenta a cardinalidade k do conjunto independente maximal $\mathbb{V}_k^l$ que esta sendo gerado;

g) número de subconjuntos $\mathbb{E}_k^-$ e $\mathbb{E}_k^+$ produzidos que também dependem da cardinalidade do conjunto $\mathbb{V}_k^l$ que se esta gerando. Assim quanto maior for a cardinalidade de $\mathbb{V}_k^l$ mais níveis k de subconjuntos $\mathbb{E}_k^-$ e $\mathbb{E}^+$ tem-se de armazenar;

h) o tempo de processamento computacional gasto aumenta muito conforme o número de vértices do grafo $\mathbb{G}$, pois a medida em que cresce o número de vértices do grafo,

cresce também o número de conjuntos independentes maximais;

i) memória necessária para guardar os conjuntos $\mathbb{V}_k^l$ e os subconjuntos $\mathbb{E}_k^-$ e $\mathbb{E}_k^+$;

j) recuo (*backtrack*) do ramo na árvore da busca enumerativa quando não levar a nenhum conjunto independente maximal, retrocedendo um passo de cada vez (digamos passo a passo), e verificando a possibilidade de continuar à frente ou então recuar mais um passo por vez, tendo, se for o caso, que encerrar o ramo e iniciar outro, ou seja, poda-se esse ramo e recomeça-se a busca em outro;

k) crescimento da cardinalidade dos conjuntos independentes maximais $\mathbb{V}_k^l$ conforme o tamanho do grafo $\mathbb{G}$, ou seja, conforme o aumento do conjunto de vértices $\mathbb{X}$, aumenta-se o tamanho k dos conjuntos $\mathbb{V}_k^l$, o que torna difícil o procedimento de formação bem como o de recuo;

l) heurísticas propostas são demoradas e de difícil implementação computacional;

m) a heurística para a escolha de vértices, apresentada no algoritmo de Bron e Kerbosch, não funciona para encontrar o primeiro conjunto independente maximal ($\mathbb{V}_k^1$), já que o $\mathbb{E}_k^-$ será sempre vazio enquanto buscar o primeiro conjunto, isto é, a escolha do primeiro elemento é aleatória, bem como a do segundo e o do terceiro, e deste modo sucessivamente, até formar o primeiro conjunto independente maximal;

n) há necessidade de se modificar o algoritmo original de Bron e Kerbosch, para gerar somente o máximo conjunto

independente, tendo que comparar os conjuntos independentes maximais entre si, guardando, a cada comparação, o maior.

5.2. - Formas para encontrar-se o máximo conjunto independente

5.2.1. - Gerar todos os conjuntos independentes maximais

O descobrimento do número de independência $\alpha[G]$, juntamente com a obtenção de um dos máximos conjuntos independentes (se existir mais de um), pode ser considerada uma tarefa fácil para grafos pequenos, menores que 15 vértices. Entretanto, quando pretende-se encontrar os conjuntos independentes maximais, de grafos grandes, surgem problemas. Tais problemas, conforme citado na seção 5.1.1., também estarão presentes no problema de encontrar o número de independência e um dos máximos conjuntos independentes determinado por tal número.

Acontece que os algoritmos existentes para o cálculo dos conjuntos independentes maximais geram tais conjuntos aleatoriamente, no sentido da cardialidade desses conjuntos, e portanto, seria necessário gerar todos os conjuntos independentes maximais e então escolher o que tenha a maior cardinalidade obtendo-se o máximo conjunto independente. Isto, certamente, exige muito tempo de processamento e grande quantidade de memória computacional, além de ser um algoritmo de complexidade exponencial na ordem de entrada.

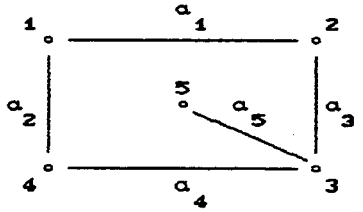
5.2.2. - Tratamento via Problema da Cobertura do Conjunto

Uma outra maneira é formular o problema da obtenção do máximo conjunto independente, através do Problema da Cobertura de Conjunto (PCC), onde as linhas de uma determinada matriz de incidência $[T]$, representam as arestas de $\mathbb{G}$ e as colunas representam os vértices. Desta forma:

$$T = [t_{ij}]_{mxn} = \begin{cases} 1 \text{ se } x_j \text{ for um vértice terminal da aresta } a_i \\ 0 \text{ caso contrário} \end{cases}$$

Assim, se $\breve{\mathbb{X}} \subseteq \mathbb{X}$ for a mínima cobertura (solução ótima) para o Problema da Cobertura de Conjunto (PCC), o conjunto $\mathbb{X} - \breve{\mathbb{X}}$ será então, o máximo conjunto independente de $\mathbb{G}$.

Exemplo 5.1.: Como se achar o máximo conjunto independente utilizando o problema da cobertura do conjunto



$\Gamma(1) = \langle 2, 4 \rangle$
$\Gamma(2) = \langle 1, 3 \rangle$
$\Gamma(3) = \langle 2, 4, 5 \rangle$
$\Gamma(4) = \langle 1, 3 \rangle$
$\Gamma(5) = \langle 3 \rangle$

Figura 5.1. Grafo $\mathbb{G}$ com vértices e arestas para o exemplo do PCC

A matriz de incidência $T = [t_{ij}]_{mxn}$ é dada por:

$$T = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \begin{array}{c} \begin{array}{ccccc} \overset{*}{x_1} & x_2 & \overset{*}{x_3} & x_4 & x_5 \end{array} \\ \begin{bmatrix} 1 & 1 & - & - & - \\ 1 & - & - & 1 & - \\ - & 1 & 1 & - & - \\ - & - & 1 & 1 & - \\ - & - & 1 & - & 1 \end{bmatrix} \end{array}$$

Como, $\breve{\mathbb{X}} = (x_1, x_3)$ é a mínima cobertura para o conjunto $\mathbb{X}$, então $\mathbb{X} - \breve{\mathbb{X}} = (x_2, x_4, x_5)$ será o máximo conjunto independente.

Para fazer isto, necessita-se de uma quantidade de memória muito grande, já que com o aumento do número de vértices aumenta o número de arestas. Por outro lado, tem-se que implementar um algoritmo mais complexo do ponto de vista teórico e computacional, que resolva o Problema de Cobertura de Conjunto (PCC) e, além disso, determinar o mínimo conjunto de Cobertura $\breve{\mathbb{X}}$ do grafo $\mathbb{G}$.

5.2.2.1. - Complexidade do PCC

Para implementar o algoritmo do Problema da Cobertura de Conjuntos (PCC) tem-se que efetuar à priori reduções inerentes da parte teórica e, além disso, construir um tableau inicial (formando blocos). Deve-se também armazenar listas de coberturas provisórias e realizar testes de domínio que são usados para limitar a busca em árvores e aperfeiçoar a eficiência do algoritmo.

5.3. - Descrição do algoritmo heurístico proposto

5.3.1. - Introdução

Em aplicações práticas a principal característica relativa a propriedade da independência de um grafo $\mathbb{G}$ é o número de independência $\alpha[\mathbb{G}]$ e, conseqüentemente, um conjunto independente

que tenha tal cardinalidade. Porém existem dificuldades conforme já apresentadas anteriormente, para se encontrar tal número bem como e um dos conjuntos (se existir mais de um) determinado por ele ao utilizar-se os métodos existentes na literatura. Neste trabalho apresenta-se um algoritmo que encontra uma solução aproximada, quando não a ótima, de rápido processamento e baixa utilização de memória.

Pode-se dizer que o passo mais importante no algoritmo, para o descobrimento do máximo conjunto independente é o da escolha do vértice que entrará no conjunto que se pretende formar. Desta forma, deve-se estabelecer ou mesmo desenvolver um critério de escolha de vértices que seja eficaz, isto é, um critério selecione elementos de tal modo que forme um dos máximos conjuntos independentes (se existir mais de um).

Para cada um dos vértices pertencentes ao conjunto $\mathbb{X}$ de vértices do grafo $\mathbb{G}$, verifica-se o grau correspondente. Se existir um único vértice que possua o menor grau, dentre todos os outros vértices em $\mathbb{X}$, escolhe-se este para ser acrescentado ao conjunto que se pretende formar. Caso contrário, isto é, se existe mais de um vértice dentro do conjunto $\mathbb{X}$, que possuem o menor grau, seleciona-se, em princípio, esses vértices para formar o subgrafo auxilir $\langle \mathbb{E}_o \rangle$.

No subgrafo $\langle \mathbb{E}_o \rangle$ verifica-se o grau de cada um de seus vértices, e existindo um único vértice cujo grau em $\langle \mathbb{E}_o \rangle$ é mínimo, escolhe-se o mesmo para ser acrescentado ao conjunto independente maximal em formação. Em caso contrário forma-se um novo subgrafo $\langle \mathbb{E}_1 \rangle$, com os vértices de $\langle \mathbb{E}_o \rangle$ que possuam grau mínimo, e assim sucessivamente, até que um único vértice com o

menor número de sucessores seja identificado ou até que o subgrafo $\langle \mathbb{E}_k \rangle$ e $\langle \mathbb{E}_{k+1} \rangle$ sejam iguais, quando escolhe-se qualquer um dos vértices de $\langle \mathbb{E}_{k+1} \rangle$ para inclusão no conjunto independente em formação. A heurística que determina a formação de uma boa solução para o máximo conjunto independente, consiste no processo de escolha acima.

Identificado um vértice $x_{i_{k+1}}$ a ser incluido no conjunto independente, elimina-se o mesmo, juntamente com seus sucessores, do conjunto de vértices $\mathbb{X}$, isto é, $\mathbb{X} = \mathbb{X} - \{x_{i_{k+1}}\} - \Gamma(x_{i_{k+1}})$.

O processo continua até o conjunto $\mathbb{X}$ ficar completamente vazio, isto é, quando não houver mais vértices a escolher em $\mathbb{X}$.

O conjunto de vértices formado será um conjunto independente maximal. Normalmente o processo de busca descrito acima forma, inicialmente, um conjunto independente maximal que corresponde, quando não ao máximo conjunto independente, ao menos a uma boa aproximação para o mesmo.

A grande vantagem na implementação do algoritmo conceitual proposto deve-se ao fato de não haver necessidade de armazenagem de conjuntos, e ao baixo tempo de processamento.

5.3.2. - O algoritmo formal

Inicializações

Passo 1.

$k = 0$

$\mathbb{V}_k = \emptyset$

$\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$

Determinação do suposto conjunto independente

Passo 2.

repita

Passo 2.1. *(Escolha de vértice)*

$\mathbb{E} = \mathbb{X}$

cardan = $|\mathbb{E}|$

$x_{i_{k+1}} = \emptyset$

repita

se (existem mais de um $x_{i_j} \in \mathbb{E}$ com

$$\Delta(x^*_{i_1}) = \ldots = \Delta(x^*_{i_j}) = \min_{x_{i_j} \in \mathbb{E}} |\Gamma(x_{i_j}) \cap \mathbb{E}|)$$

então

$\mathbb{E} = \{x^*_{i_1}, \ldots, x^*_{i_j}\}$

se ((j ≠ 2) ou (j ≠ cardan))

então

cardan = j

senão

$x_{i_{k+1}} = x^*_{i_1}$

senão

$x_{i_{k+1}} = x_{i_t}$ onde $\Delta(x_{i_t}) = \min_{x_{i_t} \in \mathbb{E}} |\Gamma(x_{i_t}) \cap \mathbb{E}|$

até que $x_{i_{k+1}} \neq \emptyset$

Passo 2.2. *(Inclusão do vértice escolhido no conjunto)*

$\mathbb{V}_{k+1} = \mathbb{V}_k \cup \{x_{i_{k+1}}\}$

k = k + 1 { nº de elementos }

Passo 2.3. *(Exclusão)*

$$\mathcal{X} = \mathcal{X} - (x_{i_{k+1}}) - \Gamma(x_{i_{k+1}})$$

até que $\mathcal{X} = \emptyset$.

Finalizações

Passo 3.

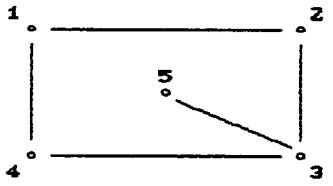
escreva('O conjunto independente maximal proposto é $\mathbb{V}$ = ', $\mathbb{V}_k$)

escreva('Cardinalidade = ',k)

fim.

5.3.4. - Exemplos para o algoritmo proposto, desenvolvidos passo por passo

Exemplo 5.2:



$\Gamma(1) = \langle 2, 4 \rangle$
$\Gamma(2) = \langle 1, 3 \rangle$
$\Gamma(3) = \langle 2, 4, 5 \rangle$
$\Gamma(4) = \langle 1, 3 \rangle$
$\Gamma(5) = \langle 3 \rangle$

Figura 5.2. Grafo $\mathcal{G}$ para exemplo do algoritmo heurístico proposto

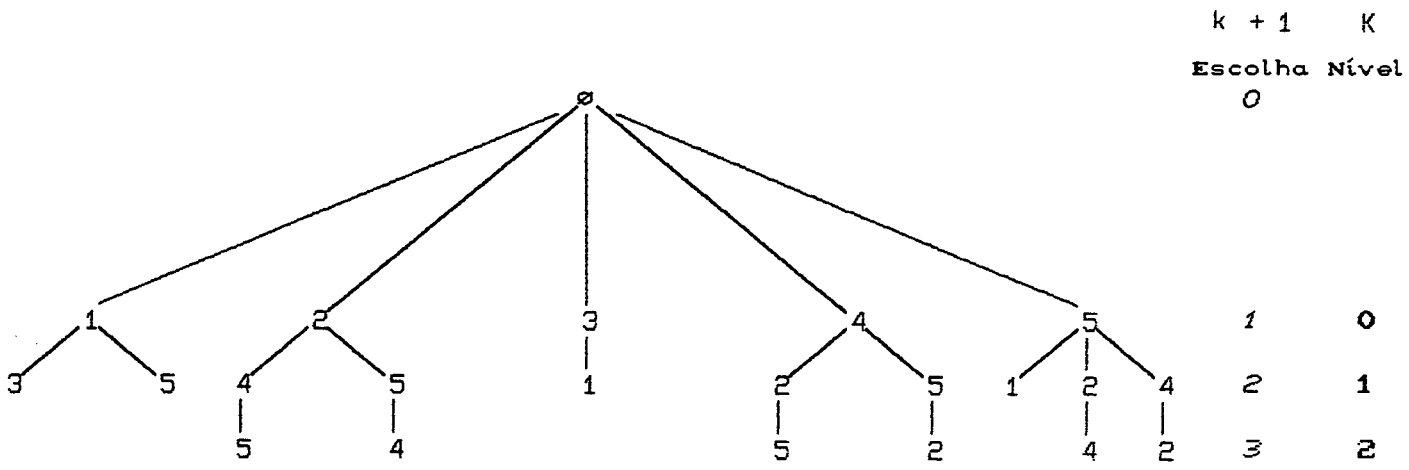


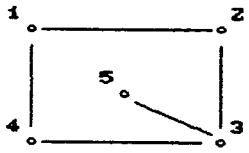
Figura 5.3. Árvore para o grafo $\mathcal{G}$ com todas as possíveis soluções

P 1. ø

$k = 0$

$\mathbb{V} = \emptyset$

$\mathbb{X} = (1, 2, 3, 4, 5)$



P 2.

P 2.1.

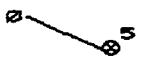
$\mathbb{E} = (1, 2, 3, 4, 5)$ (2 2 3 2 1)

cardan = 5

$x_{i_1} = \emptyset$

Não

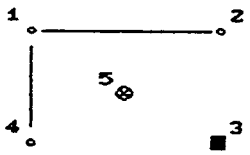
$x_{i_1} = 5$



P 2.2

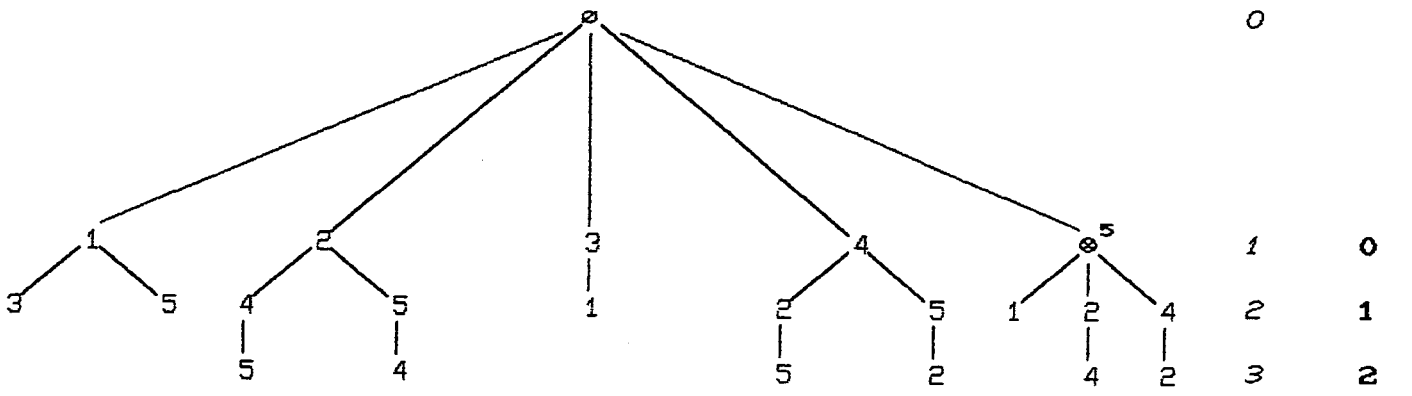
$\mathbb{V} = \emptyset \cup (5) = (5)$

$k = 1$



P 2.3.

$\mathbb{X} = (1, 2, 4)$



P 2.

P 2.1.

$\mathbb{E} = (1^{2}, 2^{1}, 4^{1})$

cardan = 3

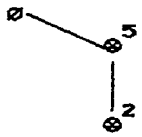
$x_{i_2} = \emptyset$

Sim

$\mathbb{E} = (2^{o}, 4^{o})$

Não

$x_{i_2} = 2$



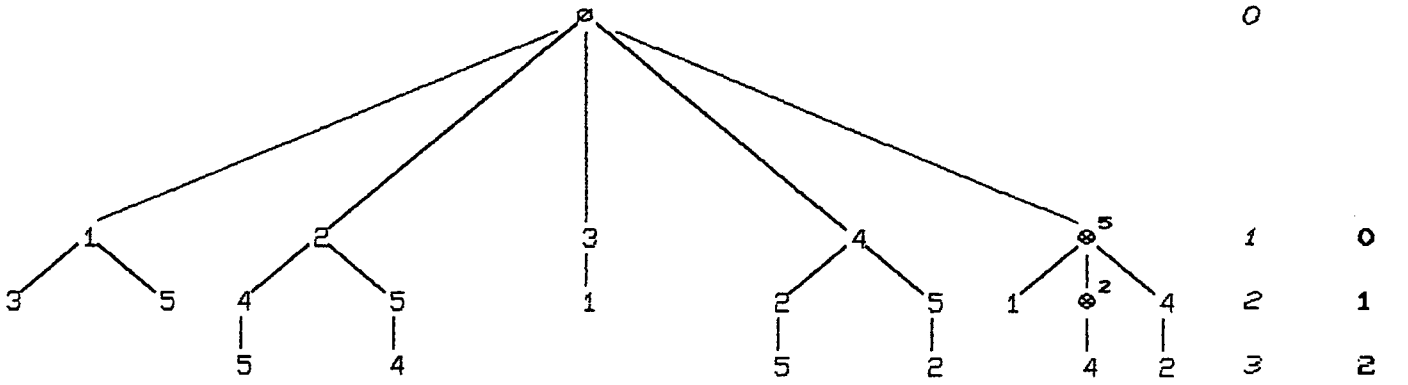
P 2.2

$\mathbb{V} = (5) \cup (2) = (5, 2)$

$k = 2$

P 2.3.

$\mathbb{X} = (4)$



P 2.

P 2.1.

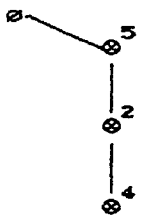
$\mathbb{E}^{o} = (4)$

cardan = 1

$x_{i_3} = 0$

Não

$x_{i_3} = 4$



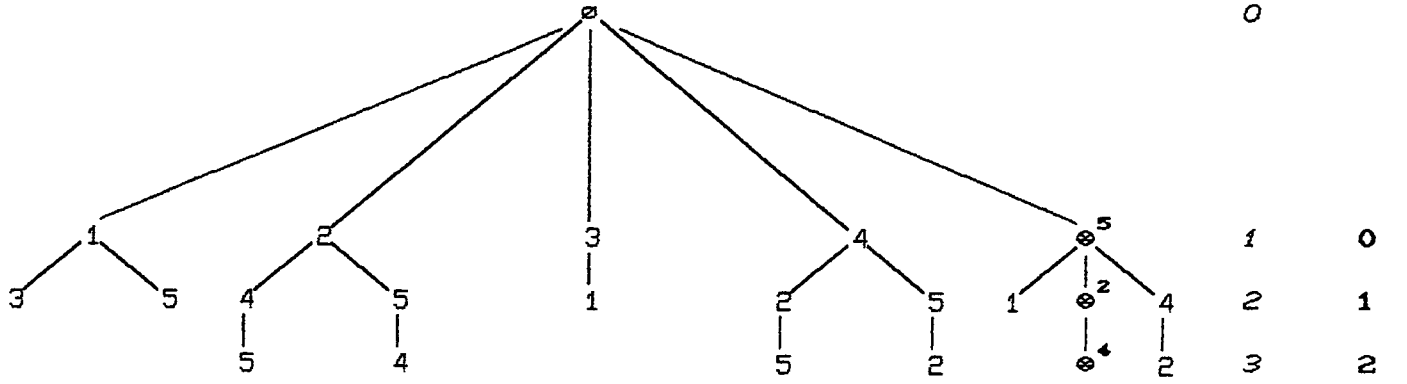
P 2.2.

$\mathbb{V} = (5, 2) \cup (4) = (5, 2, 4)$

k = 3

P 2.3.

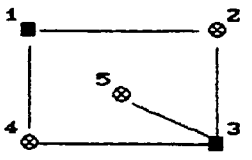
$\mathbb{X} = \emptyset$



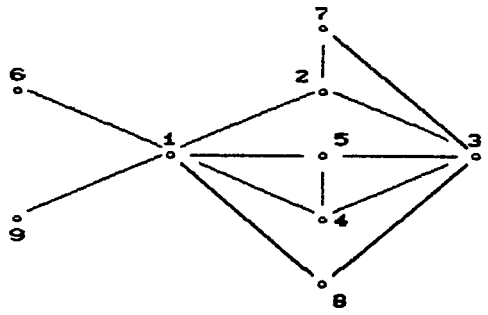
P 3.

O conjunto independente maximal proposto é $\mathbb{V}_3 = (5, 2, 4)$

Cardinalidade = 3



Exemplo 5.3: Outro exemplo para o algoritmo heurístico proposto



$\Gamma(1) = \langle 2, 4, 5, 6, 8, 9 \rangle$

$\Gamma(2) = \langle 1, 3, 7 \rangle$

$\Gamma(3) = \langle 2, 4, 5, 7, 8 \rangle$

$\Gamma(4) = \langle 1, 3, 5 \rangle$

$\Gamma(5) = \langle 1, 3, 4 \rangle$

$\Gamma(6) = \langle 1 \rangle$

$\Gamma(7) = \langle 2, 3 \rangle$

$\Gamma(8) = \langle 1, 3 \rangle$

$\Gamma(9) = \langle 1 \rangle$

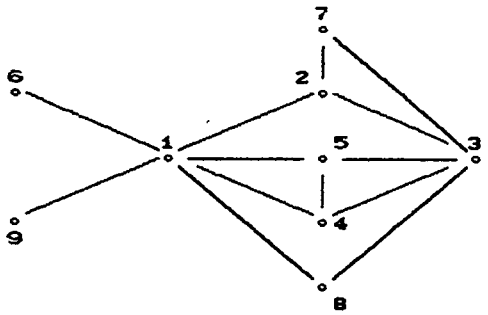
Figura 5.4. Outro grafo para exemplo do algoritmo proposto

P 1. $\varnothing$

$k = 0$

$\mathbb{V} = \emptyset$

$\mathbb{X} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$



P 2.

P 2.1.

$\mathbb{E} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (6 3 5 3 3 1 2 2 1)

cardan = 9

$x_{i_1} = \emptyset$

Sim

$\mathbb{E} = \{6, 9\}$ (o o)

Não

$x_{i_1} = 6$ $\varnothing$ 6

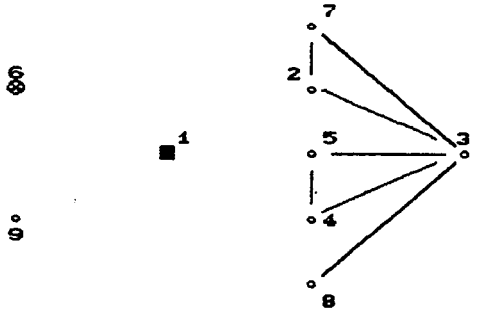
6

9

P 2.2.

$\mathbb{V} = \emptyset \cup (6) = (6)$

$k = 1$



P 2.3.

$\mathbb{X} = (2, 3, 4, 5, 7, 8, 9)$

P 2.

P 2.1.

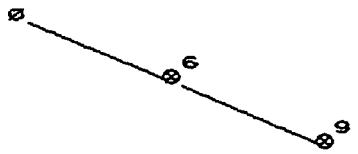
$\mathbb{E} = (2, 3, 4, 5, 7, 8, 9)$ (2 5 2 2 2 1 0)

cardan = 7

$x_{i_2} = \emptyset$

Não

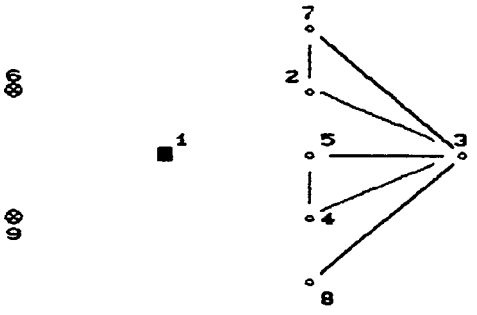
$x_{i_2} = 9$



P 2.2.

$\mathbb{V} = (6) \cup (9) = (6, 9)$

$k = 2$



P 2.3.

$\mathbb{X} = (2, 3, 4, 5, 7, 8)$

P 2.

P 2.1.

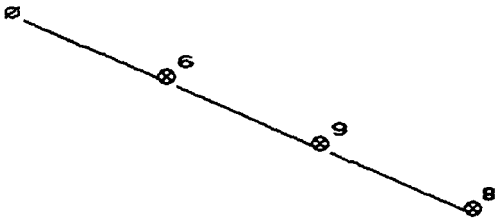
$$\mathbb{E} = \{2, 3, 4, 5, 7, 8\}$$

cardan = 6

$x_{i_3} = 0$

Não

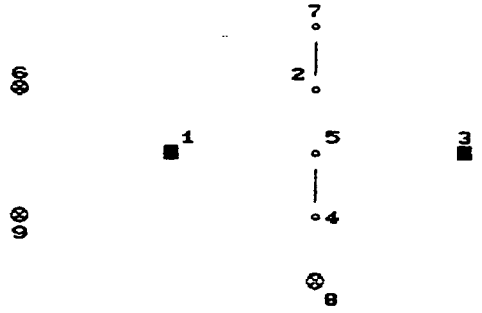
$x_{i_3} = 8$



P 2.2.

$\mathbb{V} = \{6, 9\} \cup \{8\} = \{6, 9, 8\}$

k = 3



P 2.3.

$\mathbb{K} = \{2, 4, 5, 7\}$

P 2.

P 2.1.

$$\mathbb{E} = \{2, 4, 5, 7\}$$

cardan = 4

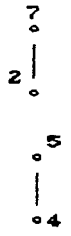
$x_{i_4} = 0$

Sim

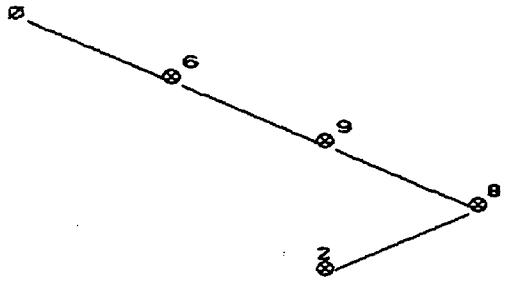
$\mathbb{E} = \{2, 4, 5, 7\}$

Não

SUBGRAFO



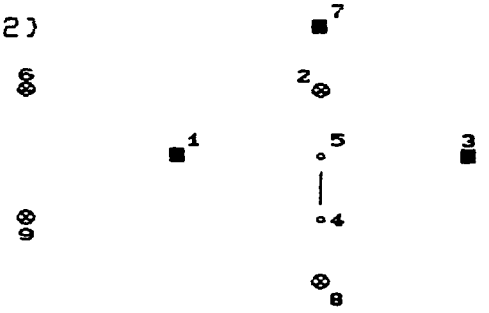
$x_{i_4} = 2$



P 2.2.

$\mathbb{V} = (6, 9, 8) \cup (2) = (6, 9, 8, 2)$

$k = 4$



P 2.3.

$\mathbb{X} = (4, 5)$

P 2.

P 2.1.

$\mathbb{E} = (\overset{1}{4}, \overset{1}{5})$

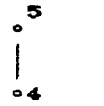
cardan = 2

$x_{i_5} = 0$

Sim

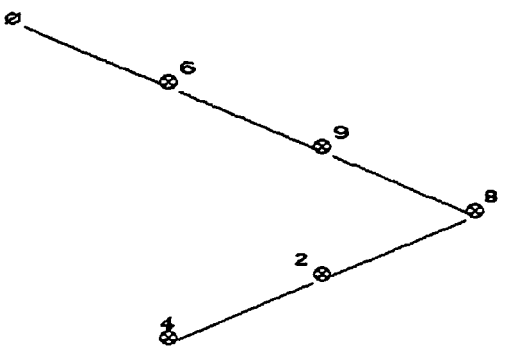
$\mathbb{E} = (4, 5)$

SUBGRAFO



Não

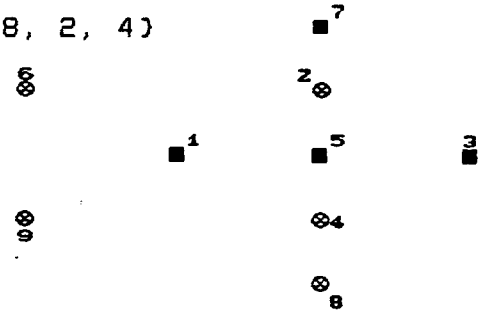
$x_{i_5} = 4$



P 2.2.

$\mathbb{V}$ = (6, 9, 8, 2) ∪ (4) = (6, 9, 8, 2, 4)

k = 5



P 2.3.

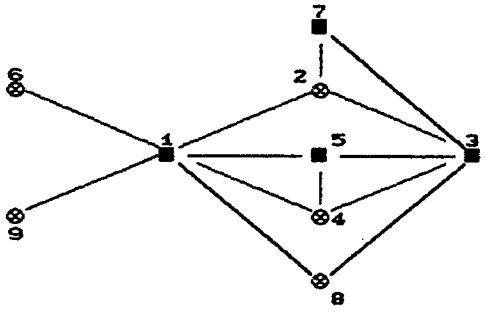
$\mathbb{X}$ = Ø

P 3.

O conjunto independente maximal proposto é $\mathbb{V}$ = (6, 9, 8, 2, 4)

Cardinalidade = 5

Fim.



7.4. - Análise da complexidade do algoritmo

O algoritmo proposto possui algumas vantagens, dentre as quais destacam-se as seguintes:

a) complexidade da ordem n^2, isto é, $\Theta(n^2)$;

b) não requer subconjuntos auxiliares e, além disso, vai destruindo o conjunto inicial $\mathbb{X}$ até torná-lo vazio, acelerando o processo de escolha de vértices a cada vértice incluido no conjunto independente;

c) não necessita de *backtrack*, o que reduz significativamente o tempo de processamento. O algoritmo somente avança, nunca recua. Vai até ao final, quando esgotam todas as possibilidades de escolha de elementos, isto é, quando $\mathcal{X} = \emptyset$;

d) gera somente um conjunto, que é uma boa aproximação para o máximo conjunto independente. Como encontra apenas um conjunto não é necessário gerar e comparar os conjuntos independentes maximais - que são gerados aleatoriamente sob o aspecto da cardinalidade desses conjuntos bem como também não precisa guardar os subconjuntos, $\mathbb{E}_k^-$ e $\mathbb{E}_k^+$, como por exemplo, no algoritmo de Bron e Kerbosch.

e) o resultado encontrado pelo algoritmo proposto é um limite inferior para o número de independência $\alpha[\mathbb{G}]$, e poderá ser utilizado como um limite mínimo para efeito de comparação dos conjuntos independentes maximais gerados pelo o algoritmo de Bron e Kerbosch;

CAPÍTULO VI

TESTES E VALIDADE DO ALGORITMO PROPOSTO

6.1. Metodologia utilizada

6.1.1. Como foram gerados os exemplos

Alguns grafos utilizados nos testes para verificar a validade do algoritmo heurístico proposto, foram coletados em livros, e são exemplos clássicos de algumas áreas da teoria de grafos. Outros foram escolhidos aleatoriamente.

Dentre os exemplos testados encontram-se o grafo de Peterson, o grafo completo $\mathbb{K}_n$ de n vértices, o grafo estrela com cinco pontas, o grafo bipartido $\mathbb{K}_{6,6}$, o grafo cactos (blocos), o grafo do contra-exemplo de Heawood para a prova de Kempe (Teorema das cinco cores), o grafo da dupla estrela, o grafo de Thomassen, o grafo de Közyrer e Grinberg, e o grafo do tabuleiro de xadrez (probl. das rainhas), além de vários outros. A seguir, mostra-se os grafos citados acima e alguns outros. Todos os exemplos apresentados abaixo, foram testados nas versões implementadas do algoritmo de Bron e Kerbosch e do algoritmo heurístico proposto.

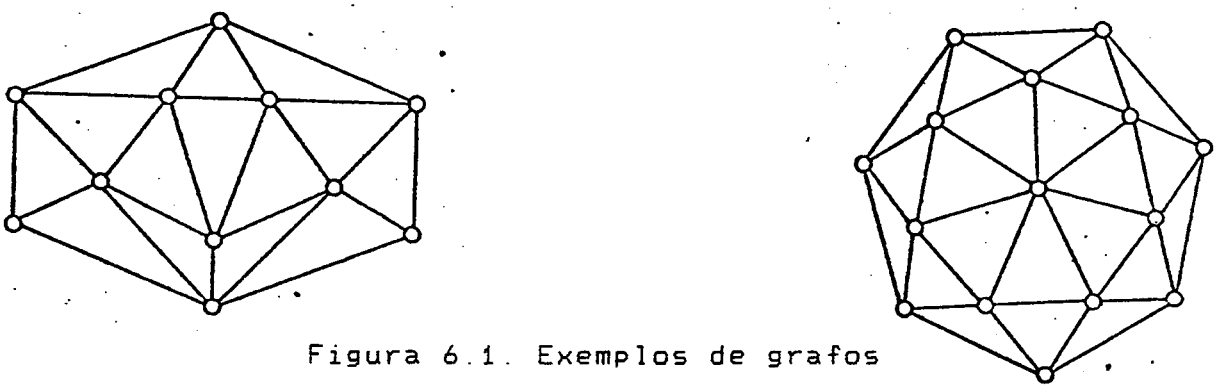


Figura 6.1. Exemplos de grafos

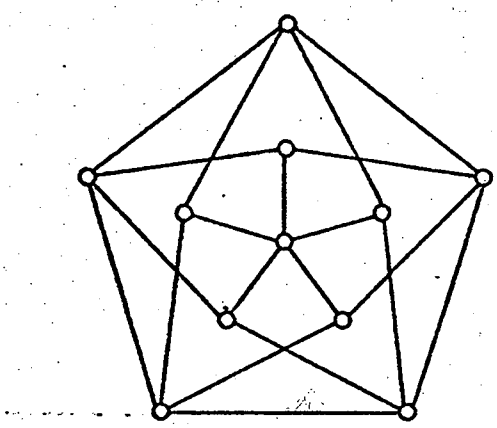


Figura 6.2. O grafo de Grötzsch

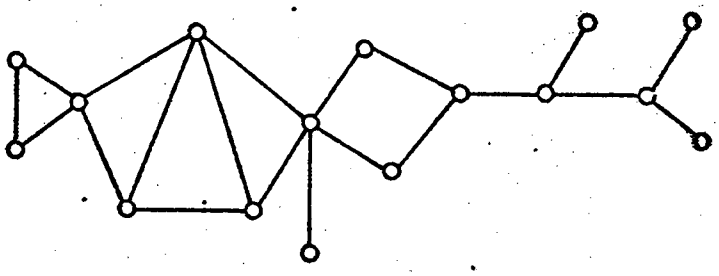


Figura 6.3. Um grafo plano

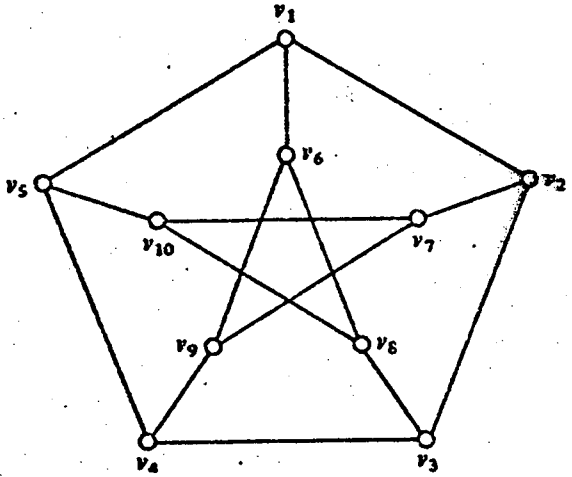


Figura 6.4. O grafo de Peterson

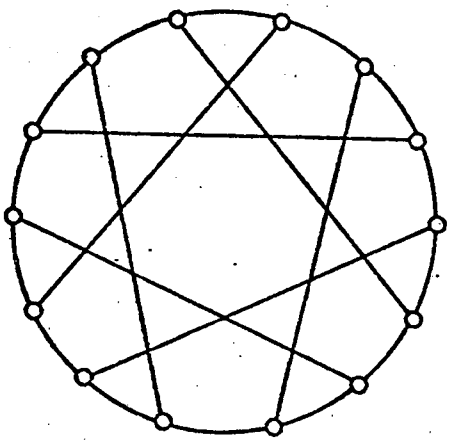


Figura 6.5. O grafo de Heawood: o único 6-cage

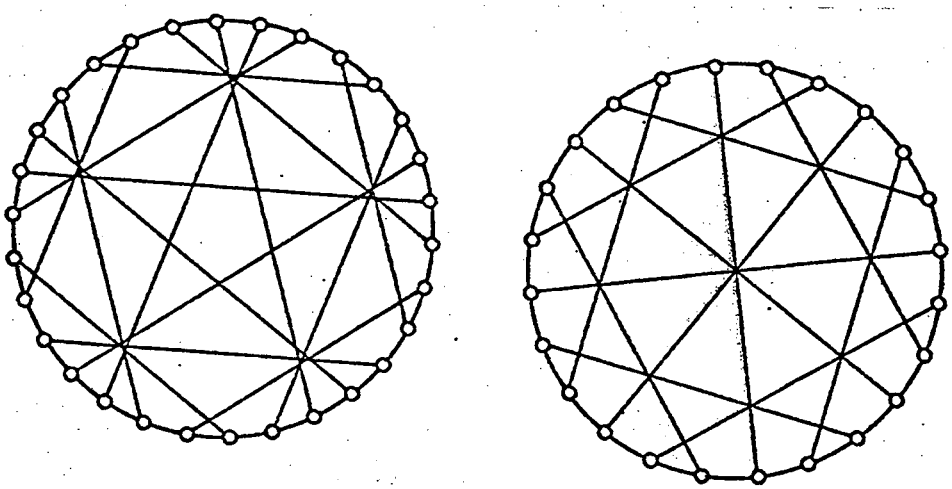


Figura 6.6. O *7-cage* e o *8-cage*

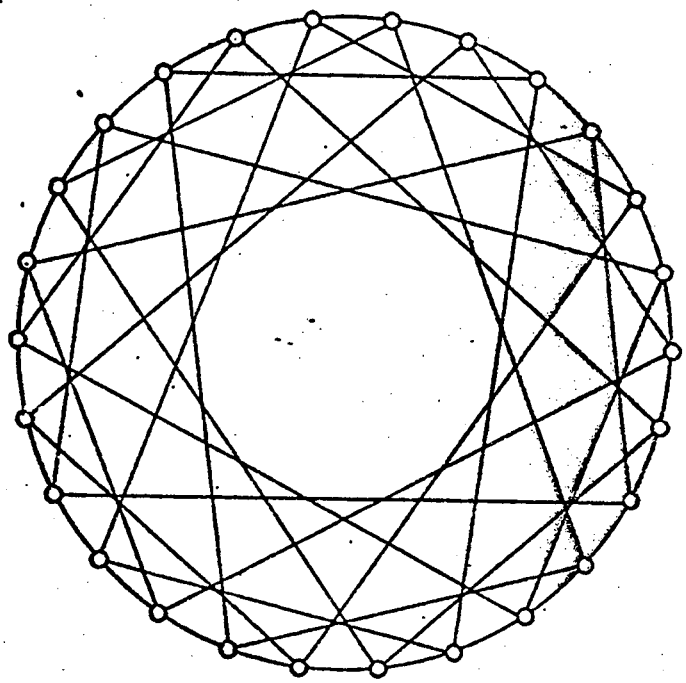


Figura 6.7. O (4, 6)-*cage*

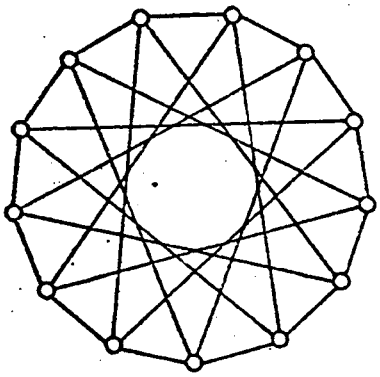


Figura 6.8. Um grafo extremo $r(3, 5) \geq 14$

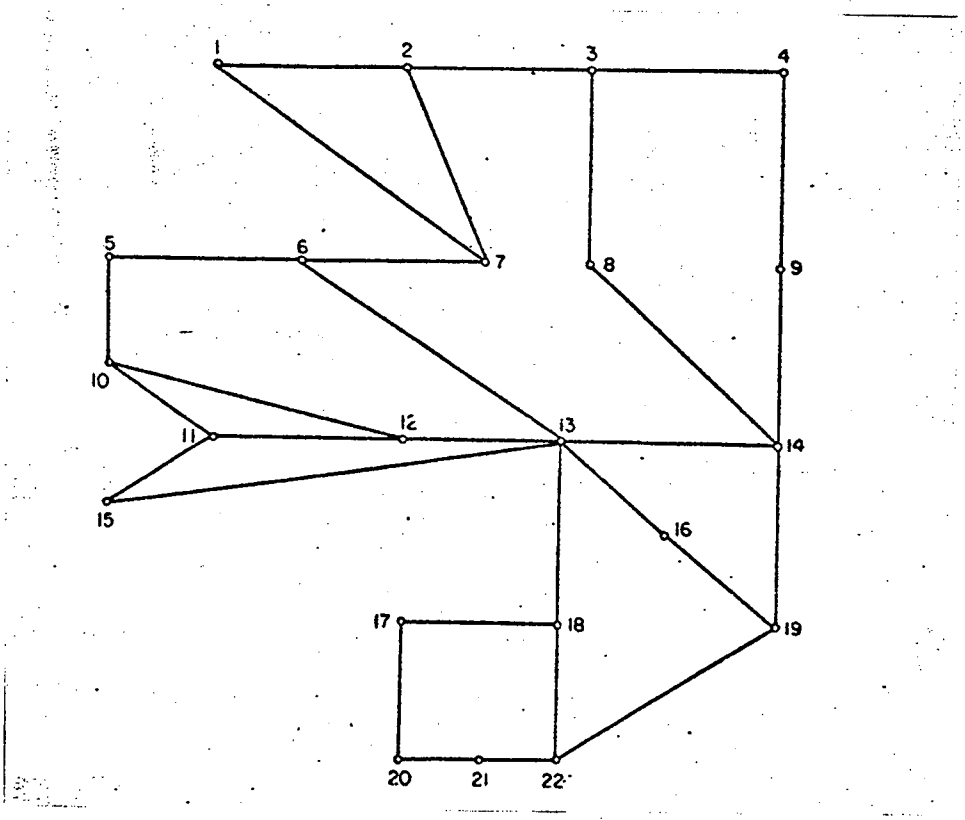


Figura 6.9. Exemplo de grafo com 22 vértices

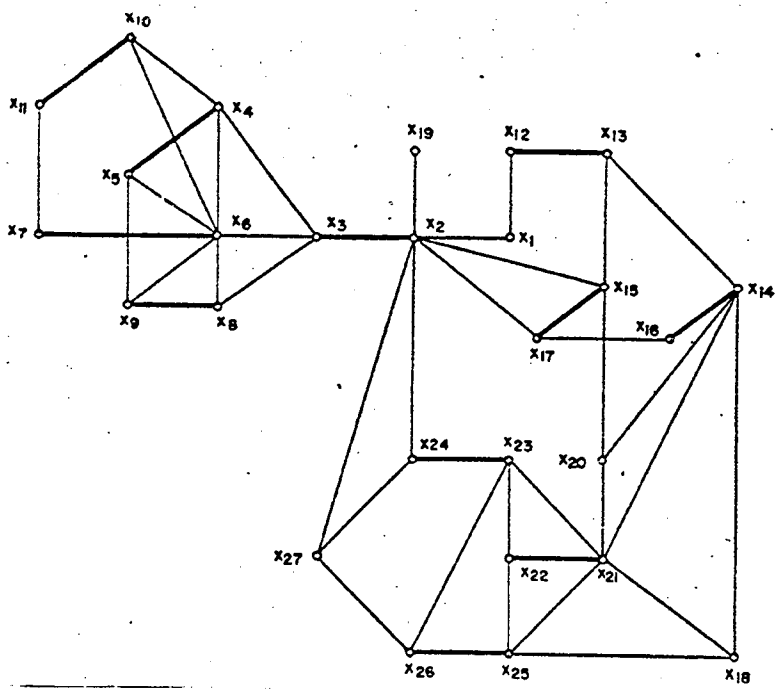


Figura 6.10. Exemplo de grafo com 27 vértices

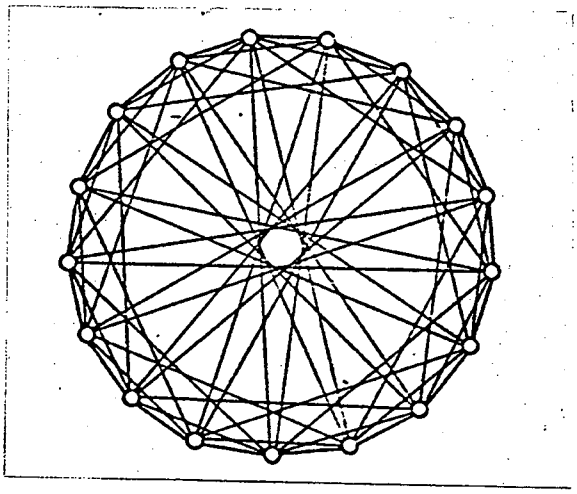


Figura 6.11. Um grafo extremo $r(4, 4) \geq 18$

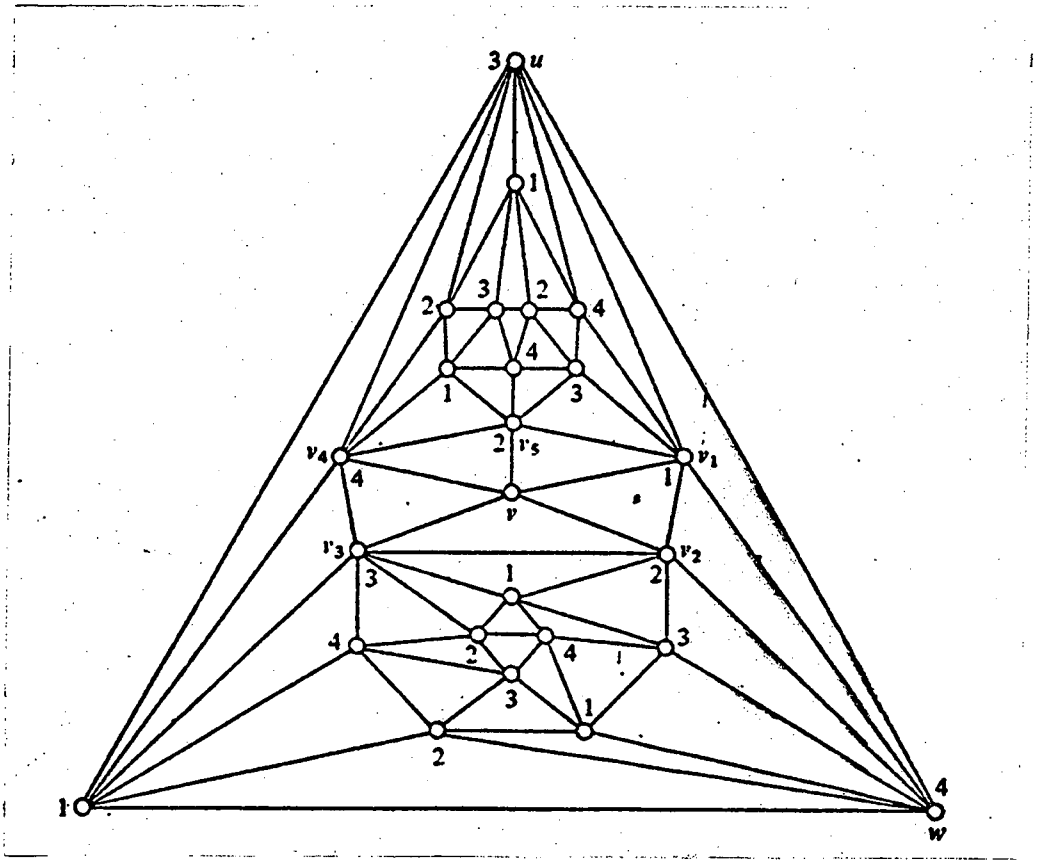


Figura 6.12. O contra-exemplo de Heawood para a prova de Kempe

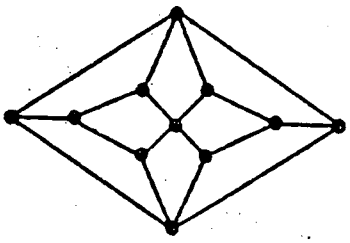


Figura 6.13. O grafo de Herschel

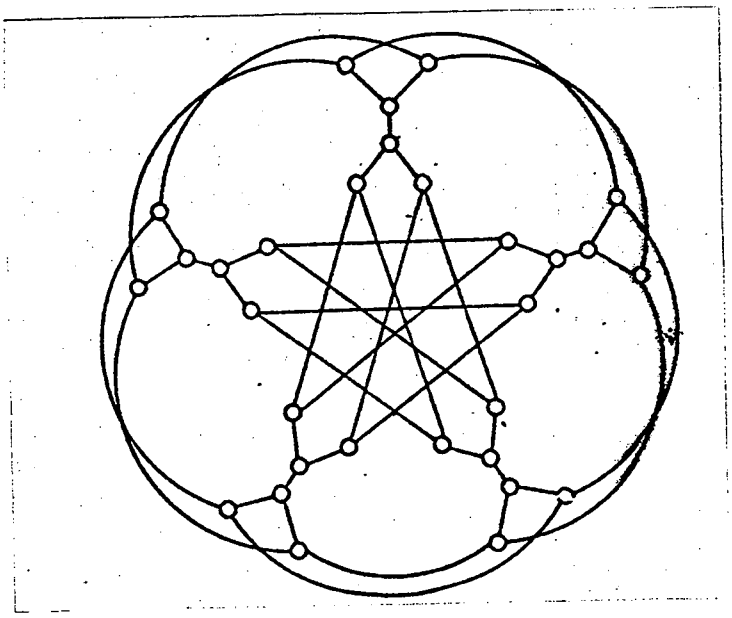


Figura 6.14. A dupla estrela: uma cúbica

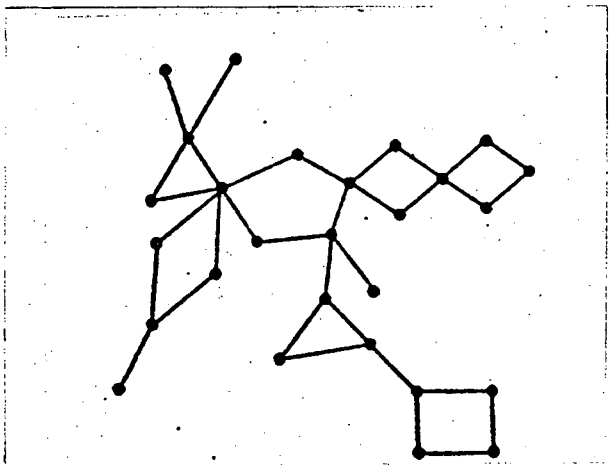


Figura 6.15. O grafo Cactus

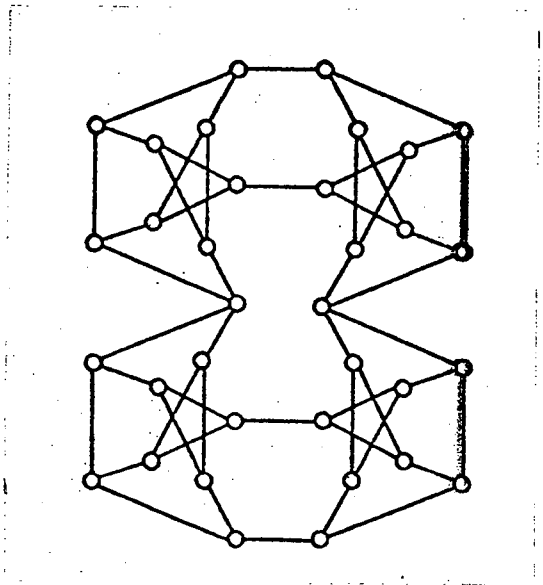


Figura 6.16. O grafo Thomassen

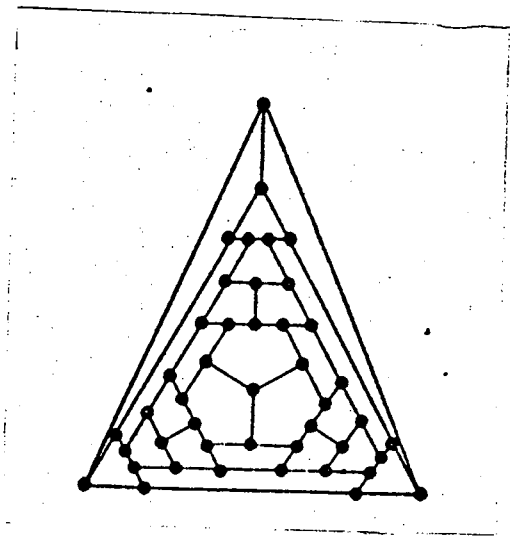


Figura 6.17. Grafo de Grinberg

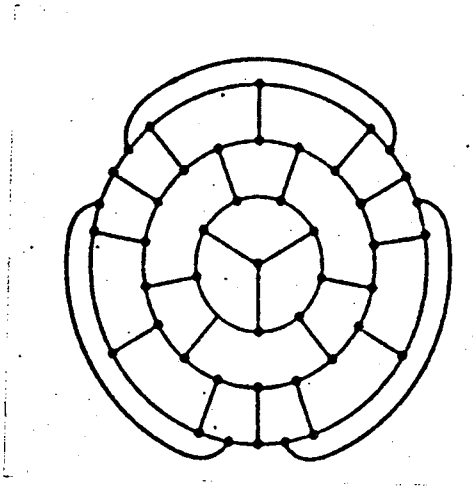


Figura 6.18. O grafo de Kozyrev e Grinberg

6.1.2. Como foram feitos os testes

Escolheu-se, dentre os vários algoritmos existentes na literatura, o de Bron e Kerbosch por ser considerado o melhor. Além disso o algoritmo de Bron e Kerbosch é o que gera os conjuntos independentes maximais sem necessidade de verificar se dentre os conjuntos anteriormente gerados há algum subconjunto contido no atual conjunto gerado.

Após esta escolha, foram feitas algumas modificações no algoritmo a fim de que fosse apresentado apenas o máximo conjunto independente e não todos os conjuntos independentes gerados.

6.1.3. Como foram comparados

Os resultados obtidos com o algoritmo de Bron e Kerbosch, e o algoritmo heurístico proposto foram comparados entre si, sob dois aspectos distintos:

(i) tempo de processamento necessário, e

(ii) cardinalidade do máximo conjunto independente apresentado.

6.1.4. Limitações de tempo

Como o algoritmo de Bron e Kerbosch gera todos os conjuntos independentes maximais de um determinado grafo G e, o número destes conjuntos cresce conforme aumenta-se o tamanho do grafo, se impôs um limite no tempo de processamento para este algoritmo.

Tentou-se executar um exemplo de 100 vértices, com o algoritmo de Bron e Kerbosch implementado, contudo não foi possível, pois ao se impor um limite de 100 horas de processamento, não se chegou ao fim da busca. Por isso, o algoritmo não obteve a solução ótima para efeito de comparação com a solução do algoritmo heurístico proposto.

Para o algoritmo proposto neste trabalho, praticamente não há restricões quanto ao tamanho prático do grafo, bem como quanto ao limite do tempo de processamento computacional.

6.2. Resultados obtidos

6.2.1. Introdução

Os resultados obtidos através do algoritmo proposto foram considerados satisfatórios, tendo em vista a obtenção de uma boa aproximação para o máximo conjunto independente, quer seja pelo aspecto teórico quer seja pela facilidade de implementacão computacional, e também pela quantidade de memória requerida e pelo tempo de processamento dispendido.

Foi realizada uma análise comparativa dos resultados obtidos quanto ao desempenho computacional do algoritmos, de Bron e Kerbosch e do heurístico proposto, um em relação ao outro.

Os dois algoritmos implementados foram submetidos a testes com exemplos de 6, 8, 10, 12, 15, 16, 17, 20, 22, 25, 26, 27, 30, 34, 46, 50 e 64 vértices e ao final, testou-se somente a versão implementada do algoritmo heurístico proposto em exemplos de 100, 140 e 200 vértices.

6.2.2. Exemplos testados

Mostra-se abaixo, três exemplos da estrutura de dados utilizada para os grafos nos testes. Considerados como pequeno, médio e grande quando se utiliza o algoritmo de de Bron e Kerbosch, porém, quando usa-se o algoritmo heurístico proposto, são considerados todos pequenos.

Apresenta-se ainda, a saída fornecida pelo algoritmo de Bron

e Kerbosch - constando da solução obtida, cardinalidade da solução, número de soluções (conjuntos independentes maximais do grafo) e tempo de processamento computacional gasto para obter a solução. Em seguida, apresenta-se também a saída fornecida pelo algoritmo heurístico proposto - constando da solução obtida, cardinalidade da solução e tempo de processamento computacional gasto para obter a solução.

```
  ox     oy    dx    dy

  40    160    48    14

vert    (x   , y)       sucessores

  1       0.5    9        2    6    8    0
  2       4.5    9        1    3    6    0
  3       8.5    9        2    4    5    0
  4      10.5    5        3    5    7    0
  5       8.5    1        3    4    6    0
  6       6.5    5        1    2    5    7    8    0
  7       4.5    1        4    6    0
  8       2.5    5        1    6    0
  0
```

```
Solucao otima obtida pelo algoritmo de Bron e Kerbosch

S = (2  5  7  8)

cardinalidade =  4

num. de solucoes =   8

tempo de processamento = 0h  0m  0, 5s
```

```
Solucao obtida pelo algoritmo heuristico proposto

S = (7  5  2  8)

cardinalidade =  4

tempo de processamento = 0h  0m  0, 5s
```

```
  ox     oy     dx     dy

-20    180     80     24

vert     (x   ,    y)        sucessores:

  1       1.5      6.0        3    0
  2       2.5      6.2        3    0
  3       2.0      5.0        1    2    4   27    0
  4       2.5      4.5        3    5   22   23   26   27    0
  5       3.2      4.8        4    6    0
  6       4.0      4.5        5    7   12   13    0
  7       4.8      5.1        6    8    0
  8       5.3      4.5        7    9   11   12    0
  9       5.8      5.0        8   10    0
 10       6.4      4.5        9   11    0
 11       5.8      4.2        8   10    0
 12       4.8      4.2        6    8    0
 13       3.8      3.8        6   14   15   22    0
 14       4.2      3.2       13    0
 15       3.2      3.0       13   16   21    0
 16       4.4      2.6       15   17   21    0
 17       4.8      1.8       16   18   20    0
 18       5.8      1.8       17   19    0
 19       5.8      1.0       18   20    0
 20       4.8      1.0       17   19    0
 21       3.2      2.2       15   16    0
 22       3.0      3.7        4   13    0
 23       2.5      3.4        4   24    0
 24       1.5      2.6       23   25   26    0
 25       1.0      1.8       24    0
 26       1.5      3.7        4   24    0
 27       1.5      4.4        3    4    0
  0


Solucao otima obtida pelo algoritmo de Bron e Kerbosch

S = (1   2   5   7   9  11  12  14  15  17  19  22  23  25  26
                                                          27)

cardinalidade = 16

num. de solucoes = 240

tempo de processamento = 0h  0m  3,84s


Solucao obtida pelo algoritmo heuristico proposto

S = (1   2  14  22  27   5   7  12   9  11  23  25  26  18  20
                                                          15)

cardinalidade = 16

tempo de processamento = 0h  0m  0,27s
```

ox	oy	dx	dy
40	160	50	16

vert	(x	y)	sucessores			
1	5.00	9.00	2	18	19	0
2	7.00	8.46	1	3	21	0
3	7.83	7.83	2	4	17	0
4	8.46	7.00	3	5	22	0
5	8.86	6.04	4	6	9	0
6	9.00	5.00	5	7	23	0
7	8.46	3.00	6	8	25	0
8	7.00	1.54	7	9	27	0
9	6.04	1.14	5	8	10	0
10	5.00	1.00	9	11	28	0
11	3.96	1.14	10	12	15	0
12	3.00	1.54	11	13	29	0
13	1.54	3.00	12	14	31	0
14	1.00	5.00	13	15	33	0
15	1.14	6.04	11	14	16	0
16	1.54	7.00	15	17	34	0
17	2.17	7.83	3	16	18	0
18	3.00	8.46	1	17	35	0
19	5.00	8.00	1	20	36	0
20	5.78	7.90	19	21	37	0
21	6.50	7.60	2	20	22	0
22	7.60	6.50	4	21	23	0
23	8.00	5.00	6	22	24	0
24	7.90	4.22	23	25	39	0
25	7.60	3.50	7	24	26	0
26	7.12	2.88	25	27	40	0
27	6.50	2.40	8	26	28	0
28	5.00	2.00	10	27	29	0
29	3.50	2.40	12	28	30	0
30	2.88	2.88	29	31	42	0
31	2.40	3.50	13	30	32	0
32	2.10	4.22	31	33	43	0
33	2.00	5.00	14	32	34	0
34	2.40	6.50	16	33	35	0
35	3.50	7.60	18	34	36	0
36	4.22	7.90	19	35	45	0
37	5.36	6.35	20	38	45	0
38	6.21	5.70	37	39	46	0
39	6.35	4.64	24	38	40	0
40	5.99	4.01	26	39	41	0
41	5.00	3.60	40	42	46	0
42	4.01	4.01	30	41	43	0
43	3.65	4.64	32	42	44	0
44	3.79	5.70	43	45	46	0
45	4.64	6.35	36	37	44	0
46	5.00	5.00	38	41	44	0
0						

Solucao otima obtida pelo algoritmo de Bron e Kerbosch

S = {1 3 5 7 10 13 15 20 22 24 27 29 32 34 36
38 40 42 44}

cardinalidade = 19

num. de solucoes = 37820

tempo de processamento = 1h 9m 57,46s

Solucao obtida pelo algoritmo heuristico proposto

S = {1 3 5 10 7 27 40 24 22 20 38 42 29 44 36
13 15 32 34}

cardinalidade = 19

tempo de processamento = 0h 0m 0,82s

6.2.3. Resultados dos exemplos testados

Na tabela 6.1 são apresentados os resultados para os testes realizados e os gráficos 6.1 e 6.4, apresentam os resultados sob a forma de gráficos[1].

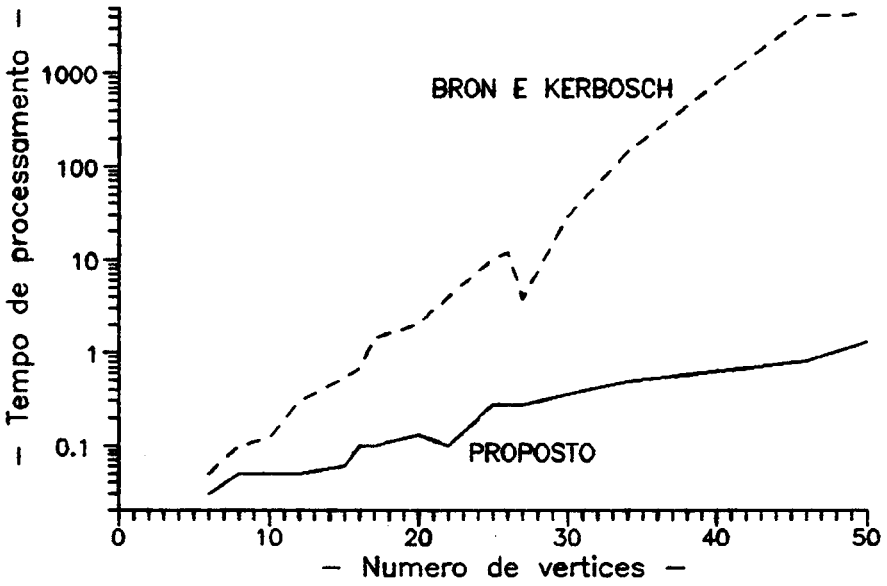
Observa-se que os resultados obtidos pelo algoritmo heurístico proposto são claramente melhores que os resultados obtidos pelo algoritmo de Bron e Kerbosch modificado para encontrar o máximo conjunto independente. Nota-se também que conforme aumenta-se o número de vértices do grafo, a diferença entre um e outro, tende a aumentar e não se pode dizer que o algoritmo de Bron e Kerbosch modificado seja competitivo com o heurístico proposto em termos de velocidade de processamento e em relação a quantidade de memória utilizada.

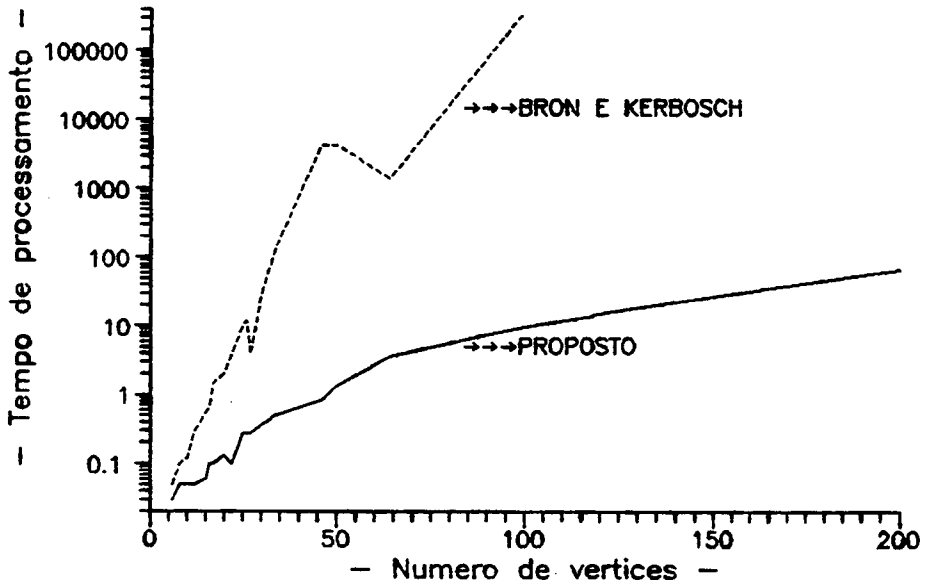
[1] Os tempos de processamento, em segundos, apresentados na tabela 6.1 e em escala logarítmica, nos gráficos 6.1 e 6.2, referem-se a implantação computacional dos algoritmos em um microcomputador **IS30 PLUS II** da Itautec.

número de vértices	tempo de execução		Diferenças (B/K - P)	
	Bron e Kerbosch	Proposto	Absoluta	%
06	0.05s	0.03s	0.02s	66.66
08	0.10s	0.05s	0.05s	100.00
10	0.12s	0.05s	0.07s	140.00
12	0.30s	0.05s	0.25s	500.00
15	0.53s	0.06s	0.47s	783.33
16	0.68s	0.10s	0.58s	580.00
17	1.42s	0.10s	1.32s	1320.00
20	2.05s	0.13s	1.92s	1476.92
22	3.89s	0.10s	3.79s	3790.00
25	10.21s	0.27s	9.94s	3681.48
26	11.97s	0.27s	11.70s	4333.33
27	3.84s	0.27s	3.57s	1322.22
30	29.27s	0.36s	28.91s	8030.55
34	2m 24.67s	0.49s	2m 24.18s	29424.48
46	1h 9m 57.46s	0.82s	1h 9m 56.64s	3958126.80
50	1h 12m 14.46s	1.32s	1h 12m 13.14s	3273722.70
64	23m 17.52s	3.57s	23m 13.95s	3904.62
100	-	9.67s	-	-
140	-	22.79s	-	-
200	-	1m 05.52s	-	-

Tabela 6.1. Tempo de execução e diferença entre entre os tempos de processamento

6.2.2. Gráficos comparativos




- Tempo de processamento -
100000
10000
1000
100
10
1
0.1
0
50
100
150
200
- Numero de vertices -
→→→BRON E KERBOSCH
→→→PROPOSTO

6.3. Excessão para convergência

Em todos os exemplos testados, descobriu-se apenas dois contra-exemplos para a convergência do algoritmo heurístico proposto e, mesmo assim o algoritmo proposto encontrou uma boa aproximação para a solução, com cardinalidade diferindo em apenas uma unidade, quando comparada com a solução ótima encontrada pelo algoritmo de Bron e Kerbosch modificado.

O primeiro contra-exemplo encontrado para a convergência da solução ótima, foi o das oito damas do tabuleiro de xadrez, onde ficam sem atacar-se umas as outras. As soluções ótima (já que existe mais de uma) para esse problema são os conjuntos independentes maximais com cardinaldades iguais a oito. O algoritmo heurístico proposto encontrou um conjunto independente maximal com 7 elementos em 3.57^{s}, enquanto que o algoritmo de Bron e Kerbosch levou 23^{m} 17.52^{s} para encontrar a solução ótima, com cardinalidade oito.

O outro contra-exemplo encontrado para o algoritmo heurístico proposto foi um grafo de 50 vértices. A solução ótima, com cardinalidade 17 foi encontrada pelo algoritmo de Bron e Kerbosch em 1^{h} 21^{m} 49.51^{s} e o algoritmo heurístico proposto encontrou uma solução de cardinalidade 16 em 1.32^{s}.

CAPÍTULO VII

CONCLUSÕES E RECOMENDAÇÕES

7.1. - Conclusões

Dentre vários aspectos, chegou-se as seguintes conclusões:

(*i*) O algoritmo heurístico proposto tem complexidade n^2, ou seja, $\Theta(n^2)$;

(*ii*) O algoritmo possui uma convergência muito boa, pois em todos os exemplos testados, sempre selecionou, quando não a ótima solução, pelo menos uma solução muito próxima da ótima. Sendo que a solução proposta encontrou até mesmo nos dois contra-exemplos uma solução que não difere em mais de uma unidade em relação a cardinalidade da solução ótima;

(*iii*) Torna-se uma busca muito rápida, já que sugere a solução encontrada no algoritmo proposto como máximo conjunto independente, sendo muito eficiente pois avança sempre em frente sem necessidade de recuos;

(*iv*) Em quase todos os exemplos testados, o algoritmo heurístico proposto encontrou uma solução que, quando não a ótima, ao menos uma muito próximo da ótima, com a cardinalidade não diferindo em mais de uma unidade quando comparada com a ótima solução fornecida pelo algoritmo de Bron e Kerbosch.

7.2. - Recomendações

Às recomendações citadas, indicam algumas possibilidades para estudos e pesquisas

(*i*) Possibilidade de modificação no algoritmo heurístico proposto para que encontre também uma boa solução para o mínimo conjunto independente. Em alguns casos é necessário encontrar o mínimo conjunto independente, em vez do máximo conjunto independente. Nestes casos, é muito simples modificar a heurística básica apresentada no algoritmo proposto para que apresente uma boa solução para o mínimo conjunto independente.

(*ii*) Alteração de algoritmos de coloração para um grafo $\mathcal{G}$ qualquer, com a utilização do algoritmo heurístico proposto neste trabalho, o que certamente acelera o procedimento.

(*iii*) Formulação de algoritmos que geram horários em instituições governamentais e de ensino, baseados na propriedade da independência maximal (salas, professores, cursos, horários, turnos, etc.)

(*iv*) Possibilidade de modificação na heurística básica do algoritmo proposto neste trabalho, bem como o acréscimo de heurística para verificar a adjacência, quando pretende-se encontrar o máximo conjunto independente em grafos direcionados. Note que num grafo direcionado, se x_i é adjacente a x_j, não implica que x_j seja adjacente a x_i.

BIBLIOGRAFIA

1. Andrade, M. C. Q. de, *A criação no processo decisório*, Recife, Editora Universitária, 1980.

2. Barbosa, R. M., *Combinatória e grafos*, São Paulo, Nobel, 1974.

3. Bednarek, A. R and Taulbee, O. C., Sobre cadeias maximais, Rev. Raimane Math. Pures Appl. **11** (1966), 23-25; MR **36 #80**.

4. Beineke, L. W. and Wilson, R. J., *Selected topics in graph theory*, New York, Academic Press, 1978.

5. Berge, C., *The theory of graphs and its aplications*, New York, John Wiley, 1962.

6. Berge, C., *Graphs and hypergraphs*, Amsterdam, North-Holland, 1973.

7. Boaventura, P. O., Netto, *Teoria e modelos de grafos*, São Paulo, Edgar Blher, 1979.

8. Bollobás, B., *Extremal graph theory*, London, Academic Press, 1978.

9. Bollobás, B., *Graph theory: an introductory course*, New York, Springer-Verlag, 1979.

10. Bron, C. and Kerbosch, J., Algorithm 457: Finding all cliques of an undirected graph, *Comm.* ACM **16** (1973), 575-577.

11. Busaker, R. G. and Saaty, T. L., *Finite graphs and networks: an introducion with applications*, New York, MacGraw-Hill, 1965.

12. Carvalho, R. L., *Máquinas, programas e algoritmos*, Campinas, 2ª Escola de Computação, Inst. de Mat., Estat. e Ciências da Computação, Universidade de Campinas, 1981.

13. Christofides, N., An algorithm for the chromatic number of a graph, *Comput J.* **14** (1971), 38-39; MR **43 #4719**.

14. Christofides, N., *Graph theory: an algorithmic approach*, New York, Academic Press, 1975.

15. Corneil, D. G. and Graham, B., An algorithm for determining the chromatic number of a graph, *SIAM J. Comput.* 2 (1973), 311-318.

16. Deo, N., *Graph theory with applications to engineering and computer science*, Englewood Cliffs, Prentice-Hall, 1974.

17. Even, S., *Algorithmic combinatorics*, New York, Mac Millan, 1973.

18. Even, S., *Graph algorithms*, Potomac, Computer Science Press, 1979.

19. Furtado, A. L., *Teoria dos grafos: algoritmos*, Rio de Janeiro, Livros Técnicos e Científicos, 1973.

20. Furtado, A. L., Santos, C. S., Veloso, P. A. e Azevedo, P. A., *Estrutura de dados*, Rio de Janeiro, Editora Campus, 1973.

21. Garey, M. R. and Johnson, D. S., The complexity of near-optimal graph coloring, *J. of the ACM*, 23, 43-49, 1976.

22. Garey, M. R. and Johnson, D. S., *Strong NP-completeness results: motivation, examples and implications*, J. of the ACM, 25, 439-508, 1978.

23. Garey, M. R. and Johnson, D. S., *Computers and intractability: a guide of NP-completeness*, San Francisco, Freeman, 1979.

24. Columbic, M, C., *Algorithmic graph theory and perfect graphs*, New York, Academic Press, 1980.

25. Gondran, M. et Minoux, M., *Graphes e algorithmes*, Paris, Editions Eyrolles, 1979.

26. Goodman, S. E. and Hedetniemi, S. T., *Introdution to design and analysis of algorithms*, New York, MacGraw-Hill, 1977.

27. Greene, D. H. and Knuth, D. E., *Mathematics for analysis of algorithmis*, Boston, Birkhauser, 1981.

28. Harary, F., *Graph theory*, Massachuts, Addison-Wesley, Reading, 1969.

29. Harary, F., Normam, R. Z. and Cartwright, D., *Structural models: an introduction to the theory of directed graphs*, New York, John Wiley & Sons, 1965.

30. Harrison, M. C., *Data strutures and programming*, Glenview, Scott, Foresman & Co., 1973.

31. Heawood, P. J., Map colour theorems, *Quart. J. Math.*, 24, 332-338, 1890.

32. Hopcroft, J. and Tarjan, R., Efficient algorithms for manipulation, *Comm. of the ACM*, 16, 372-378, 1973b.

33. Horowitz, E. and Sahni, S., *Fundamentals of data strutures*, Woodland Hills, Computer Science Press, 1976.

34. Horowitz, E. and Sahni, S., *Fundamentals of computer algorithms*, Rockville, Computer Science Press, 1978.

35. Kaufmann, A., *Méthodes et modèles de recherche opérationnelee*, Vol II, Dunod, 1968.

36. Kempe, A. B., On the geographical problem of four colors, *Amer. J. Math.*, 2, 193-204, 1879.

37. Knuth, D. E., *The art of computer programming, volume I: Fundamental algorithms*, Reading, Addison-Wesley, 1968.

38. Knuth, D. E., Estimating The efficiency of backtrak programs, *Mathematics of computation*, 29, 121-136, 1975.

39. Luccheci, C. L., Introdução à teoria dos grafos, Poços de Caldas, 12 Colóquio Brasileiro de Matemática, 1979.

40. Luccheci, C. L., Simon, I., Simon, J. e Kowaltowski, T., *Aspectos teóricos de computação*, Rio de Janeiro, Inst. de Mat. Pura e Aplicada (Projeto Euclides), 1979.

41. Lucena, C. J. P., *Introdução às estruturas de informação*, Rio de Janeiro, Livros Técnicos e Científicos, 1972.

42. MacDiarmid, C., *Determining the chromatic number of a graph*, Report STAN-CS-76-576, Computer Science Depto., Stanford University (1976).

43. Machtey, M. and Yong, P., *An introduction to the general theory of algorithms*, New York, Elsevier North-Holland, 1978.

44. Manna, Z., *Mathematical theory of computation*, New york, MacGraw-Hill 1974.

45. Matula, D. W., Marble, G. and Isaacson, J. D., Graph coloring algorithms, *in* Read, R. C., ed., *Graph theory and computing*, New York, 109-122, Academic Press, 1972.

46. Mithchen, J., On various algorithms for estimating the chromatic number of a graph, *the Computer Journal*, 19, 182-183, 1076.

47. Moon, J. W. and Moser, L., On cliques in graphs, *Israel J. Math.* 3 (1965), 23-28; MR 32 #60.

48. Mulligan, G. D., *Algorithms for finding cliques of a Graph*, Tech. Report 41, Computer Science Depto., University of Toronto (1972).

49. Mycielski, J., Sur le coloriage des graphs, *Colloq. Math.*, 3, 161-162, 1955.

50. Nijenhius, A. and Wilf, H. S., *Combinatorial algorithms*, New York, Academic Press, 1975.

51. Ore, O., *Theory of graphs*, Providence, American Math. Soc., 1962.

52. Ore, O., *The four-color problem*, New York, Academic Press, 1967.

53. Pacitti, T. e Atkinson, *Programação e métodos computacionais*, vols. 1 e 2, Rio de Janeiro, Livros técnicos e científicos, 1975.

54. Page, E. S. and Wilson, L. B., *Information representation and manipulation in a computer*, London, Cambridge University Press, 1973.

55. Page, E. S. and Wilson, L. B., *An introduction to computacional combinatorics*, Cambridge, Cambridge University Press, 1979.

56. Papadimitriou, C. H. and Steiglitz, K., *Combinatorial optimization: algorithms and complexity*, Englewood Cliffs, Printice Hall, 1982.

57. Pfaltz, J. C., Computer data structures, New York, MacGraw-Hill, 1977.

58. Pombo, H. C. R., *Representação de grafos em computador*, Rio de Janeiro, Tese de mestrado, COPPE/UFRJ, 1979.

59. Rabin, M. O., *Probabilistic algorithms, algorithms and complexity: new directions and recents results*, J. F. Tramb, ed., New York, Academic Press, 21-40, 1976.

60. Rabuske, M. A., *Introdução à teoria de grafos*, Florianópolis, SC, Ed. da UFSC, 1992.

61. Read, R. C., Graph theory algoritms, *Graph theory and its applications*, (ed. B. Harris), New York and London, Academic Press (1970), 51-78; MR **41** #8281.

62. Reingold, E. M., Nievergelt, J. and Deo, N., *Combinatorial algorithms: theory and practice*, New York, Englewood Cliffs, 1977.

63. Roy, B., *Algebre moderne et théorie des graphes*, Vol 1 and Vol 2, Dunod, Paris, (1969, 1970).

64. Roy, B., An algorithm for a general constrained set covering problem, *In: Computing and Graph Theory*, Read, Ed. Academic Press, New York, 1972.

65. Saaty, T. and Kainen, P., *The four problem: assaults and conquests*, New York, MacGraw-Hill, 1977.

66. Santos, R. N., *Obtenção do número cromático de um grafo*, Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1979.

67. Schwarz, G. A., *Geração de horário em instituições de ensino com otimização simultânea de tempo e espaço*, Florianópolis, SC, Tese de mestrado, Dpto. de Eng. de Prod. e Sist./UFSC, 1990.

68. Simon, Istvan, *Inrodução à teoria de complexidade de algoritmos*, São Paulo, Inst. de Mat. e Estat., Univ. de São Paulo, 1979.

69. Standish, T. A., *Data structure techniques*, Reading, Addison-Wesley, 1980.

70. Syslo, M. M., Deo, N. and Kowalik, J. S., *DISCRETE OPTIMIZATION ALGORITHMS with pascal programs*, Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1983.

71. Szwarcfiter, J. L., *GRAFOS e algoritmos computacionais*, Rio de Janeiro - RJ, Ed. Campuus Ltda., 1984.

72. Tarjan, R. E., Complexity of combinatorial algorihms, *SIAM Rewiew*, 20, 457-491, 1978.

73. Tarjan, R. E. and Trojanowski, *Finding a maximal independent set*, Report STAN-CS-76-550, Computer Science Depto., Stanfort University (1976).

74. Terada, R., *Desenvolvimento de algoritmos e complexidade de computação*, Rio de Janeiro, 3ª Escola de Computação, Depto. de Informática, Pontifícia Univ. Católica, 1982.

75. Tremblay, J. P. e Bunt, R. B., *CIÊNCIA DOS COMPUTADORES uma abordagem algorítmica*, São Paulo - SC, MacGraw Hill do Brasil, 1983.

76. Tutte, W. T., *The connectivity of graphs*, Toronto, Toronto University Press, 1966.

77. Wang., C. C., An algorithm for the chromatic number of a graph, *J. Assoc. Comput. Mach.* **21** (1974), 385-391.

78. Wilson, R. J., *Introduction to graph theory*, Edinburgh, Oliver and Boyd, 1972.

79. Wilson, R. J. and Beineke, L. W., *Applications of graph theory*, New York, Academic Press, 1970.