SILVIO LUIZ BRAGATTO BOSS

CARACTERIZAÇÕES DE BUSCAS EM HIPERMULTIGRAFOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Jair Donadelli Junior
Co-orientador: Prof. Dr. André Luiz Pires Guedes

CURITIBA

2010

SILVIO LUIZ BRAGATTO BOSS

CARACTERIZAÇÕES DE BUSCAS EM HIPERMULTIGRAFOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Jair Donadelli Junior
Co-orientador: Prof. Dr. André Luiz Pires Guedes

CURITIBA

2010

SILVIO LUIZ BRAGATTO BOSS

CARACTERIZAÇÕES DE BUSCAS EM HIPERMULTIGRAFOS

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre no Programa de Pós-Graduação em Informática da Universidade Federal do Paraná, pela Comissão formada pelos professores:

Orientador: Prof. Dr. Jair Donadelli Junior
Departamento de Informática, UFPR

Prof. Dr. André Luiz Pires Guedes
Departamento de Informática, UFPR

Prof. Dr. André Vignatti
Departamento de Informática, UTFPR

Prof. Dr. Alexandre Ibrahim Direne
Departamento de Informática, UFPR

Curitiba, 23 de abril de 2010

Há aqueles que lutam um dia; e por isso são bons;
Há aqueles que lutam muitos dias; e por isso são muito bons;
Há aqueles que lutam anos; e são melhores ainda;
Porém há aqueles que lutam toda a vida; esses são os imprescindíveis.
Bertold Brecht

Dedico esta dissertação aos meus pais Sergio e Nilsa que me deram a vida e me ensinaram a lutar por ela, e à buscar meus ideais. A eles dedico este trabalho e que seja um exemplo que, ***mesmo em momentos difíceis e controversos, a perseverança alcança os seus frutos****.*

Agradecimentos

Aos meus pais e à minha família, agradeço todo o amor, carinho, compreensão e respeito.

Aos meus queridos irmãos Serginho e Jaqueline e minha cunhada Vanessa, sempre presente, pelo apoio e incentivo nos momentos de dificuldades, sempre indicando a direção a ser tomada nos momentos de maior dificuldade e, humildemente, pedindo ajuda quando preciso foi, assim como muitas vezes também o fiz.

Ao Prof. Dr. Jair Donadelli, orientador desta dissertação, por todo empenho, sabedoria, compreensão, paciência e principalmente, pela confiança, mais uma vez depositada, no meu trabalho e pelas sugestões que fizeram com que concluíssemos este trabalho.

Ao Prof. Dr. André Guedes, co-orientador desta dissertação, por sua ajuda e interesse, e por suas sábias ideias.

Ao Professor Dr. Alexandre Direne pelas excelentes sugestões por ocasião do Exame de Qualificação.

À Professora Sandra Mara Guse Scós Venske, sempre me incentivando com suas palavras gentis e alegres, e pelo seu apoio incondicional.

Ao meu grande amigo Leandro Zatesco, pelos valiosos momentos de discussão e conhecimentos compartilhados, pela indescritível solidariedade e afeto inestimável, que se traduziram sempre em entusiasmadas respostas, sou imensamente agradecido pela generosidade e carinho. Qualquer agradecimento aqui ficará muito aquém do justo.

Aos colegas do Mestrado pela excelente relação pessoal que criamos e que espero não se perca. Em especial a Maysa Alves, Márcia Valéria, Josynei de Souza e Rubens Suguimoto, pelo apoio nos momentos bons e menos bons, e pela amizade constante.

As também amigas especiais, Bruna Colnago e Francieli Triches pelo companheirismo e amizade e pelas conversas de desabafo.

Aos meus colegas de república Leandro Zatta, Marlon Kroetz e Marlon Neves, pelo convívio diário, pela preparação para as provas e seminários, cafés, discussões e pelas ideias. Meu muito obrigado támbem aos amigos de Curitiba: Karen Stadler, Claudia Carolina Stadler(Ina), Suelen Boff, André Pinz.

As minhas amigas Cíntia Manzoni e Natália Seron, que nunca esqueceram da verdadeira amizade que há entre nós, onde vimos que a distância não é suficiente para separar os amigos.

À secretária do ppginf Jucélia, por sua força, carinho, entusiasmo e otimismo contagiantes, sendo uma profissional extremamente competente e dedicada.

A todos os professores pelo carinho e dedicação demonstrado ao longo do mestrado.

A todas as pessoas que, direta ou indiretamente, contribuíram para a execução dessa Dissertação de Mestrado.

E finalmente à Deus por sempre me iluminar e me guiar...

SUMÁRIO

Lista de Figuras

Lista de Algoritmos

Lista de Abreviaturas

BG	Busca Genérica (*Generic Search*).
BFS	Busca em Largura (*Breadth-First Search*).
DFS	Busca em Profundidade (*Depth-First Search*).
MNS	Busca da Vizinhança Maximal (*Maximal Neighborhood Search*).
MLS	Busca do Rótulo Maximal (*Maximal Label Search*).
MCS	Busca da Cardinalidade Máxima (*Maximum Cardinality Search*).
LexBFS	Busca em Largura Lexicográfica (*Lexicographic Breadth-First Search*).
LexDFS	Busca em Profundidade Lexicográfica (*Lexicographic Depth-First Search*).

Resumo

Buscas em grafos é uma das ferramentas mais simples e mais utilizadas para algoritmos em grafos. Um algoritmo de busca examina os vértices e as arestas de um grafo a partir de um vértice inicial e, sistematicamente visita um novo vértice por iterativa travessias em arestas incidentes a um vértice anteriormente já visitado. A ordem em que esses vértices são visitados definem uma enumeração desses vértices em um dado grafo. Na literatura disponível, poucos resultados teóricos são conhecidos sobre uma enumeração que pode ser gerada por um algoritmo de busca específico, embora buscas como em Largura e em Profundidade sejam algoritmos tradicionais e bem conhecidos na literatura atual. Recentemente, dois novos algoritmos, Busca em Largura Lexicográfica e a Busca da Cardinalidade Máxima, têm sido aplicados em uma grande variedade de problemas e, além desses, outras estratégias também são conhecidas, como a Busca em Profundidade Lexicográfica, da Vizinhança Maximal e do Rótulo Maximal, usadas para o reconhecimento de certas classes de grafos, por exemplo. Muito dos resultados obtidos nas aplicações desses algoritmos de busca dependem da simples caracterização da enumeração que estes algoritmos podem computar. Neste trabalho, generalizamos o conceito de busca orientada por aresta para o caso de hipermultigrafo, apresentaremos características das enumerações e por fim provaremos que essas enumerações caracterizam um algoritmo de busca.

Abstract

Graph searching is one most used and simplest tools in graph algorithms. A graph search algorithm is just an algorithm to systematically go through all the vertices and all the edges of a given graph, and this usually is done by starting at an initial given vertex and, from that on, systematically discover new vertices by iteratively traversing all the edges incident to a previously discovered vertex. The order in which the vertices of a given graph are visited by a search algorithm define an enumeration of its vertices. Presently, very few results are known about an enumeration that can be produced by an specific search algorithm, while search strategies like breadth first search and depth first search are traditional and well known algorithms in the literature. Recently, two new algorithms, namely lexicographic breadth-first search and maximum cardinality search, has been applied in a wide variety of problems and another strategies, such as lexicographic depth-first search, maximal neighbourhood search and maximal label search has succeed in design of algorithms for recognition of various graph classes, for example, and much of this success is due to the fact of these results depend only on the characterisation of the enumeration produced by searching algorithms. In this work we generalise the concept of searching guided by edges algorithms to work in hypermultigraph case, we present characterisations of the enumerations produced by these algorithms and, finally, prove that those enumeration in fact characterise a search algorithm.

Capítulo 1

Introdução

Neste capítulo descrevemos a contextualização da proposta, os objetivos gerais e específicos. Ao final, são descritos brevemente os tópicos abordados pelos demais capítulos desta dissertação.

1.1 Contextualização

Busca é um nome que se refere a algoritmos que, sistematicamente, examinam todos os vértices e todas as arestas de um grafo. Nesse trabalho, tais algoritmos possuem como entrada um hipermultigrafo e um vértice que inicializa a busca e tem como saída uma enumeração do conjunto das arestas. Uma *enumeração* define a ordem em que as arestas são visitadas durante a execução do algoritmo e uma *ordem de busca* é uma enumeração das arestas de um hipermultigrafo. Esses algoritmos de busca possuem critérios na escolha de uma aresta para sua enumeração. A busca genérica, (ver Seção 3.1), não especifica qual vértice escolher; a busca em largura, (ver Seção 3.2), e a busca em profundidade, (ver Seção 3.3), (do inglês *breadth-first search* (BFS) e *depth-first search* (DFS)), respectivamente, são algoritmos tradicionais que têm uma estratégia para escolher o próximo vértice a ser explorado. Tanto BFS quanto DFS não determinam, entretanto, qual a ordem para colocar os vizinhos de um vértice não visitado em uma fila ou em uma pilha.

DFS foi analisado por Tarjan ((TARJAN, 1972) apud (CORNEIL, 2004a)) e tem sido utilizado por diversas aplicações como conectividade, planaridade, ordenações topológicas e componentes fortemente conectadas em digrafos. BFS tem sido aplicado em problemas como menor caminho e fluxo em redes. Outras estratégias de busca também existem, em 1970, Rose, Tarjan e Lueker (ROSE; TARJAN, 1975), introduziram uma variante do BFS, o LexBFS (*Lexicographic Breadth-first Search* - Busca em Largura Lexicográfica) para determinar esquemas da eliminação perfeita em grafos cordais. Assim como a busca em largura lexicográfica, o algoritmo MCS (*Maximum Cardinality Search* - Busca da Cardinalidade Máxima), recebe um grafo cordal como entrada e produz uma enumeração de eliminação perfeita. Ambos os algoritmos (LexBFS e MCS) têm a mesma descrição, mas MCS utiliza pesos para determinar o próximo vértice a ser escolhido. Mais recentemente Corneil e Krueger, introduziram três novos algoritmos denominados LexDFS (*Lexicographic Depth-first Search* - Busca em Profundidade Lexicográfica), MNS (*Maximal Neighborhood Search* - Busca da vizinhança Maximal) e MLS (*Maximal Label Search* - Busca do Rótulo Maximal) descritos em (CORNEIL; KRUEGER, 2008; BERRY et al., 2005). Os algoritmos de busca em largura lexicográfica e em profundidade lexicográfica utilizam rótulos para escolher qual o próximo vértice a ser visitado, a escolha para uma ordem de busca será aquele que tiver o maior rótulo, a diferença entre os dois algoritmos está no esquema da construção desses rótulos. O algoritmo busca da vizinhança maximal, também utiliza rótulos para determinar o próximo vértice a ser explorado, no entanto, o vértice candidato será aquele que tiver um maior número de vizinhos já visitados. Ambos os algoritmos LexBFS e LexDFS são restrições do algoritmo MNS. Já o algoritmo MLS utiliza um esquema de rotulagem para sua execução, assim, é possível computar qualquer um dos paradigmas de busca apresentados neste trabalho.

1.2 Objetivos do Trabalho

Nesta dissertação realizou-se uma revisão conceitual e teórica dos algoritmos de buscas em grafos, com destaque na busca em largura e profundidade lexicográficos e na busca da

vizinhança maximal. Propõe-se também o estudo do formalismo teórico das enumerações das buscas em grafos, conduzidos pelas arestas. O objetivo principal deste trabalho é apresentar simples generalizações das buscas em grafos apresentadas por (CORNEIL; KRUEGER, 2008). Assim os resultados obtidos são extensões das contribuições de (CORNEIL; KRUEGER, 2008) para o caso de hipergrafos com arestas múltiplas, ou hipermultigrafos.

Como objetivos secundários têm-se:

1. Levantamento da bibliografia básica e necessária para a elaboração desse trabalho;
2. Apresentar os algoritmos de busca conduzido pelas arestas, tal como, a busca genérica, em largura, em profundidade, em largura lexicográfica, em profundidade lexicográfica e a busca da vizinhança maximal, suas características e particularidas;
3. Propor adaptações dos teoremas que caracterizam enumerações de buscas em cada um dos algoritmos de busca citado acima para o caso de hipermultigrafos;
4. Apresentação do algoritmo *Busca do Rótulo Maximal*, generalizar e propor as caracterizações que rege tal algoritmo para o caso de hipermultigrafos.

1.3 Estrutura da Dissertação

Esta dissertação está organizada em 6 capítulos: uma introdução, a revisão bibliográfica, as buscas em hipermultigrafos, as caracterizações dos teoremas para hipermultigrafos, o algoritmo busca do rótulo maximal e as conclusões.
O capítulo de **Introdução** aborda algumas motivações para o desenvolvimento do trabalho.
O capítulo de **Revisão Bibliográfica** aborda conceitos relevantes sobre os temas envolvidos neste trabalho. Este capítulo está divido em quatro seções, de acordo com a seguinte ordem:

- **Notações e Definições.** Aborda as notações e definições necessárias para o desenvolvimento do trabalho.

- **Nota Histórica.** Aborda os trabalhos correlatos pertinentes à dissertação.

- **Definição e Caracterização de Buscas em hipermultigrafos.** Esta seção aborda conceitos relacionados à buscas em hipermultigrafos, tais como, os algoritmos de buscas, ordem de busca e caracterização de uma ordem de busca.

- **Grafo Linha.** Esta seção aborda as definições sobre o referido grafo e alguns exemplos.

- **Representação de Hipermultigrafos em Grafos Linha.** Esta seção aborda como um hipermultigrafo pode ser transformado em um grafo linha, seguido por um exemplo.

O capítulo **Algoritmos de Buscas em Hipermultigrafos** aborda como as diferentes buscas são executadas em hipermultigrafos e é mostrado alguns exemplos de execução. São apresentados também caraterísticas e particularidades de cada busca.
O capítulo **Caracterizações de Buscas em Hipermultigrafos** aborda as adaptações dos teoremas para o caso de hipermultigrafo e algumas considerações.
O capítulo **Algoritmo Busca do Rótulo Maximal** aborda definições e as caracterizações das buscas em hipermultigrafos para o referido algoritmo.
O capítulo **Conclusões** aborda as considerações finais, conclusões obtidas com o trabalho e sugestões de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

As seções seguintes trazem o problema a ser adordado, definições básicas sobre grafos, grafo linha, multigrafos e hipergrafos, caracterização e definição de buscas em hipermultigrafos, necessárias para o desenvolvimento do trabalho.

2.1 Notações e Definições

Seja $G = (V, E)$ um grafo não direcionado com conjunto de vértices V e conjunto de arestas E, sem perda de generalidade, $V = \{1, \ldots, n\}$. O número de vértices é denotado por $n = |V|$ e o número de arestas por $m = |E|$.

Dizemos que dois vértices u e v são *adjacentes* se a $e = \{u, v\} \in E$, analogamente duas arestas g e f são *adjacentes* se $g \cap f \neq \emptyset$. Dizemos ainda que u e v são as *extremidades* da aresta e e que a aresta e é *incidente* aos vértices u e v. Dizemos que uma aresta e é *incidente* a um vértice v se este vértice for uma de suas extremidades. A *vizinhança* de v é o conjunto dado por: $N(v) = \{u \in V : \{u, v\} \in E\}$. Chamamos de *grau* de v o valor $|N(v)|$ e denotamos por $\mathrm{d}(v)$. Uma *sequência* $P = (v_1, \ldots, v_k)$ de vértices é um caminho em um grafo G se $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)\} \subseteq E$. Estas arestas pertencem ao caminho P e o *comprimento de* P é o número de arestas do caminho. Se $u = v_1$ e $v = v_k$, dizemos que P é um caminho$-u, v$. Um caminho é simples se seus vértices são

distintos. A *distância* $d(u, v)$ entre dois vértices u e v no grafo G é o comprimento do menor *caminho*$-u, v$, ou ∞ se não existe tal caminho. Um grafo G é *conectado* ou *conexo* se existe um caminho entre todo par de vértices em G. Uma sequência $C = (v_1, \ldots, v_k)$ é um circuito em G se $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k), (v_k, v_1)\} \subseteq E$.

Alguns dos algoritmos de busca, tal como LexBFS, LexDFS e MNS conceitualmente atribuem rótulos para as arestas, que são modificados cada vez que uma aresta e é visitada. Cada rótulo é uma *string* de dígitos e é utilizado para decidir qual aresta será a próxima escolhida. O algoritmo deve escolher a aresta que possui maior rótulo de acordo com a ordem lexicográfica. Essa ordem lexicográfica é conhecida também como a ordem do dicionário. Para a discussão e prova de corretude de alguns algoritmos de busca que computam uma enumeração σ das arestas, dizemos que $\text{rótulo}_i(e)$ denota o rótulo que foi atribuído à aresta e na iteração i. Se $k = \sigma(i)$, escrevemos $\text{rótulo}_k(e)$. Utilizamos a notação $\sigma = (e_1 \ldots e_m)$ para denotar $e_i = \sigma(i)$, que é o número atribuído a aresta e por σ. Uma atenção deve ser feita para o fato que os algoritmos de busca nesta dissertação geram enumerações das arestas de 1 a m, e escrevemos essa enumeração da *esquerda* para a *direita*. Utilizamos a notação $e \prec f$ para dizer que $\sigma(e) < \sigma(f)$. Em uma enumeração σ com duas arestas e e f, se a aresta e foi visitada antes da aresta f, denotado por $e \prec f$, dizemos que e *ocorre antes* de f ou que e foi *visitado antes* que f ou ainda que e está à *esquerda* de f em σ.

Um *multiconjunto* X é um conjunto que admite repetições dos seus elementos. Mais formalmente um *multiconjunto* é um par (X, μ_x), sendo X um conjunto e $\mu : X \to \mathbb{N}$ uma função que associa, para cada elemento x de X, um inteiro positivo $\mu_x(x)$, chamado de *multiplicidade* de x em X, que é o número de ocorrências de x em X. Quando, entretanto, o contexto for claro, escrevemos simplesmente $\mu(x)$. Um multisubconjunto de um conjunto S é um subconjunto X de S tal que $\mu_X(X) \leq \mu_S(X)$, para todo $_X \in X$.

Dado um conjunto finito V, chamado de conjunto de vértices, uma hiper-aresta f sobre V é um subconjunto não vazio e não unitário de V. Um *hipermultigrafo* H é uma tripla (V, E, μ_E), sendo E um multiconjunto de arestas sobre V com função de multiplicidade

$\mu_E = \mu$. Um *hipermultigrafo* H é um hipergrafo se todas as arestas possuem multiplicidade 1. Um *hipermultigrafo* H é um multigrafo se todas as arestas possuem exatamente 2 vértices distintos. Um *multigrafo* H é um grafo se todos os vértice de todas as arestas têm multiplicidade 1.

Uma relação binária $\mathbb{R}$ sobre E' é um subconjunto $\mathbb{R} \subseteq E' \times E'$. A relação binária $\mathbb{R}$ é *reflexiva* se $aa \in \mathbb{R}$ para todo $a \in E'$, *antisimétrica* se $ab \in \mathbb{R}$ e $ba \in \mathbb{R}$ implica que $a = b$ para todo $a, b \in E'$ e *transitiva* se $ab \in \mathbb{R}$ e $bc \in \mathbb{R}$ implica que $ac \in \mathbb{R}$ para todo $a, b, c \in E'$. Uma *ordem parcial* sobre o conjunto E' é uma relação binária $\mathbb{R}$ que é reflexiva, antisimétrica e transitiva. Escrevemos $a \trianglelefteq b$ para significar que $ab \in \mathbb{R}$ e $a \triangleleft b$ para significar que $a \trianglelefteq b$ e $a \neq b$.

2.2 Nota Histórica

Mencionamos na contextualização do trabalho que algoritmos como BFS e DFS são um dos mais fundamentais paradigmas de busca em grafos conhecidos e utilizados na literatura. Recentemente algoritmos como, os já citados, BFS e DFS, assim como LexBFS e LexDFS tem recebido uma maior atenção sobre a enumeração dos vértices que pode ser gerado em um grafo. Rose, Tarjan e Lueker (ROSE et al., 1976) introduziram o algoritmo LexBFS para determinar ordenação de eliminação perfeita em grafos cordais. Este algoritmo tem sido muito utilizado para reconhecimento de várias famílias de grafos e para encontrar estruturas chaves em certas classes de grafos (CORNEIL, 2004a). Em (BRANDSTÄDT et al., 1997) é apresentado a caracterização da enumeração dos vértices que o LexBFS pode computar. Esta caracterização é crucial para provas de corretude para vários outros algoritmos de busca. Corneil e Krueger (CORNEIL; KRUEGER, 2008) insvestigaram uma importante questão sobre a enumeração dos vértices que pode ser obtido por uma algoritmo de busca: tendo uma tripla de vértices a, b e c, onde a foi visitado antes que b e b visitado antes que c, sendo que ac tem aresta mas ab não tem aresta, como pode o vértice b ter sido visitado antes do vértice c? Esse trabalho, baseia-se na caracterização do LexBFS, e mostra que algoritmos de busca como BFS e DFS tem caracterizações

similares. Também são apresentadas respostas a essa questão, uma para cada paradigma de busca e também as provas de que estas respostas caracterizam uma enumeração de vértices. A seguir, será apresentado alguns dos algoritmos de buscas, as propriedades e teoremas, apresentados em (KRUEGER, 2005), que caracterizam suas enumerações para o caso de grafo, guiado por vértices.

O algoritmo 1 mostra a busca genérica guiado por vértices apresentado em (CORNEIL; KRUEGER, 2008).

Algoritmo 1: Busca Genérica por Vértices

Entrada: um grafo G = (V,E) e um vértice inicial s ∈ V
Saída : uma enumeração σ de V

```
1 S ← s
2 para i ← 1 to n faça
3 |   escolha e remova um vértice não enumerado v de S
4 |   σ(v) ← i
5 |   para cada vértice não enumerado w adjacente a v faça
6 |   |   adicione w em S
```

Propriedade 2.2.1: *Dada uma enumeração σ, se $a \prec b \prec c$, $ac \in E$ e $ab \notin E$, então existe um vértice $d \prec b$ tal que $db \in E$.*

Note que não existe nenhuma restrição quanto a posição de a e d em σ. Apenas dizemos que deve existir um vértice adjacente a b, que foi visitado antes de b em σ.

Teorema 2.2.2: *(CORNEIL; KRUEGER, 2008) Para um grafo arbitrário $G = (V, E)$, uma enumeração σ de V é uma ordem de busca genérica dos vértices de G se, e somente se σ tem a Propriedade 2.2.1.*

O Algoritmo 2 mostra a busca em largura guiado por vértices apresentado em (CORNEIL; KRUEGER, 2008).

Propriedade 2.2.3: *Dada uma enumeração σ, se $a \prec b \prec c$, $ac \in E$ e $ab \notin E$, então existe um vértice $d \prec a$ tal que $db \in E$.*

Esta propriedade restringe a Propriedade 2.2.1. Assim, qualquer enumeração computada pelo algoritmo de busca em largura tem a Proriedade 2.2.3 e que qualquer enumeração

Algoritmo 2: Busca em Largura por Vértices

Entrada: um grafo G = (V,E) e um vértice inicial s ∈ V
Saída : uma enumeração σ de V

```
1 Inicialize a fila com s
2 para i ← 1 to n faça
3     remova v do início da fila
4     σ(v) ← i
5     para cada vértice não enumerado w adjacente a v faça
6         se w não está na fila então
7             coloque w na fila
```

com esta propriedade, pode ser computada pelo algoritmo de busca em largura, conforme diz o Teorema .

Teorema 2.2.4: *(CORNEIL; KRUEGER, 2008) Para um grafo arbitrário $G = (V, E)$, uma enumeração σ de V é uma ordem de busca em largura dos vértices de G se, e somente se, σ tem a propriedade 2.2.3.*

O Algoritmo 3 mostra a busca em largura lexicográfica por vértices apresentado em (CORNEIL; KRUEGER, 2008).

Algoritmo 3: Busca em Largura Lexicográfica por Vértices

Entrada: um grafo G = (V,E) e um vértice inicial s ∈ V
Saída : uma enumeração σ de V

```
1 Atribui o rótulo zero para todos os vértices
2 rótulo(s) ← n
3 para i ← 1 to n faça
4     pegue um vértice não enumerado v com maior rótulo lexicográfico
5     σ(v) ← i
6     para cada vértice não enumerado w adjacente a v faça
7         Coloque como sufixo o valor (n − i) para o rótulo(w)
```

Propriedade 2.2.5: *Dada uma enumeração σ, se $a \prec b \prec c$, $ac \in E$ e $ab \notin E$, então existe um vértice $d \prec a$ tal que $db \in E$ e $dc \notin E$.*

Esta propriedade diz que se $dc \in E$, então o vértice c será escolhido antes de b. Como o vértice a é adjacente a c e não a b, isso quer dizer que c será escolhido antes de b, então deve existir um vértice antes que a na qual causou b ter sido escolhido antes de c.

Teorema 2.2.6: *(CORNEIL; KRUEGER, 2008) Para um grafo arbitrário G, uma enumeração σ de V é uma ordem de busca em largura lexicográfica dos vértices de G se, e somente se, σ tem Propriedade 2.2.5.*

Pode-se notar que a condição de que *dc* não é aresta, é apenas a diferença entre a lexicográfica e a busca em largura tradicional. Podemos obter o imediato corolário, sem mencionar seus respectivos algoritmos, que uma enumeração LexBFS é uma enumeração BFS, desde que a Propriedade 2.2.3 se submeta a Propriedade 2.2.5 (CORNEIL; KRUEGER, 2008). Apresentamos a seguir a ideia da rotulação de um vértice para o algoritmo de busca da vizinhança maximal.

Escolha um vértice v não visitado, tal que rótulo(v) não está contido no rótulo(u), para qualquer u que ainda não foi visitado.

Quadro 2.1: Ideia de rotulação do Algoritmo 4.

Neste algoritmo os rótulos são representados por conjunto e não mais *string* como apresentado nos algoritmos anteriores. O Algoritmo 4 mostra a busca da vizinhança maximal por vértices apresentado em (CORNEIL; KRUEGER, 2008).

Algoritmo 4: Busca da Vizinhança Maximal por Vértices

Entrada: um grafo G = (V,E) e um vértice inicial s ∈ V
Saída : uma enumeração σ de V

```
1 Atribui o rótulo zero para todos os vértices
2 rótulo(s) ← n
3 para i ← 1 to n faça
4 |   pegue um vértice não enumerado v com maior rótulo maximal
5 |   σ(v) ← i
6 |   para cada vértice não enumerado w adjacente a v faça
7 |   |   rótulo(w) ← rótulo(w) ∪ {i}
```

Propriedade 2.2.7: *Dada uma enumeração σ, se $a \prec b \prec c$, $ac \in E$ e $ab \notin E$, então existe um vértice d com $d \prec b$ tal que $db \in E$ e $dc \notin E$.*

Krueger se questionou sobre que tipo de busca poderia obter adicionando a restrição de que $dc \notin E$, para a Propriedade 2.2.1. A resposta para essa questão é dada pela Propridade 2.2.7 que caracteriza as enumerações apresentada pelo Algoritmo 4.

Teorema 2.2.8: *(CORNEIL; KRUEGER, 2008) Para um grafo arbitrário G, uma enumeração σ de V é uma ordem de busca da vizinhança maximal dos vértices de G se, e somente se, σ tem a propriedade 2.2.7.*

A Propriedade 2.2.7 é uma simples generalização das propriedades dos algoritmos LexBFS e LexDFS. Assim, podemos obter cada uma dessas buscas do Algoritmo 4, por restringir qual vizinhança maximal deve ser utilizada, especificada pelas suas respectivas buscas.

O Algoritmo 5, apresentado em (KRUEGER, 2005), é capaz de realizar uma busca, especificando um esquema de rotulagem como parâmetro de entrada. Esse esquema de rotulagem guia a busca de tal forma que, algoritmos como BFS e DFS podem ser computados por ele. Apresentamos o algoritmo, denominado *busca do rótulo maximal* como o Algoritmo 5 e a estrutura do esquema de rotulagem utilizada por ele.

Definição 2.2.9: (BERRY et al., 2005) Dizemos que (L, $\trianglelefteq$, l_0, INC) é um esquema de rotulagem se:

- L é um conjunto (o conjunto dos rótulos);
- $\trianglelefteq$ é uma ordem parcial sobre L, que será utilizada para escolher um vértice de rótulo maximal;
- l_0 é um elemento de L, utilizado para inicializar os rótulos;
- INC é uma função de $L \times \mathbb{Z}^+$ para L, que será utilizada para atualizar um rótulo.

Outras definições do algoritmo necessárias para o desenvolvimento do trabalho podem ser encontradas no capítulo 5.

Algoritmo 5: Busca do Rótulo Maximal por Vértice

Entrada: um grafo G = (V,E) e um esquema de rotulagem (L,$\trianglelefteq$,l_0,INC)
Saída : uma enumeração σ de V

```
1 Atribui o rótulo l_0 para todos os vértices
2 para i ← 1 to n faça
3     pegue um vértice não numerado v com rótulo maximal sobre a relação ⊴
4     σ(v) ← i
5     para cada vértice não numerado w adjacente a v faça
6         para cada aresta vw em G faça
7             rótulo(w) ← INC(rótulo(w),i)
```

2.2.1 Aplicações

Podemos citar algumas aplicações que as enumerações, computada por um determinado algoritmo de busca, pode gerar. Por exemplo, a propriedade que caracteriza uma enumeração LexBFS é fundamental para provar que o inverso de uma enumeração obtida executando o algoritmo LexBFS sobre um grafo cordal é uma ordenação de eliminação perfeita (ROSE et al., 1976). Um grafo é *cordal* se, e somente se, cada circuito em G de tamanho maior que três tem uma corda. Um vértice é *simplicial* se, e somente se, sua vizinhança é um subgrafo completo. Uma *ordenação de eliminação perfeita* é uma ordenação em G tal que v_i é *simplicial* em $G_i := G(\{v_i, \ldots, v_n\})$, para $i = 1, \ldots, n$. Rose, Tarjan e Lueker provou o seguinte teorema:

Teorema 2.2.10: *(ROSE et al., 1976) Se G é um grafo cordal e σ é uma enumeração LexBFS de G, então o inverso de σ é uma ordenação de eliminação perfeita.*

Assim, podemos descrever um algoritmo para o reconhecimento de um grafo cordal, apresentado no Algoritmo 6. Determinar se uma enumeração arbitrária é uma ordenação de elimação perfeita pode ser computado em tempo linear. O teste na linha 2 do Algoritmo 3 pode ser feito em tempo linear (CORNEIL, 2004a). A maioria das aplicações baseiam-se pesadamente nas enumerações que o algoritmo LexBFS pode computar. Muitos dos recentes algoritmos baseados no LexBFS utilizam multiplas varreduras para descobrir certas estruturas de uma dado grafo, como exemplo, o reconhecimento de grafo de intervalo único e cografo (ver CORNEIL,2004a para maiores informações).

Dahlhaus, Gustedt e McConnell (DAHLHAUS et al., 2002) apresentou como encontrar enumerações DFS de um complemento de um grafo em tempo linear (DAHLHAUS apud CORNEIL; KRUEGER, 2008). Para (CORNEIL; KRUEGER, 2008) podemos utilizar tais resultados para obter informações sobre o grafo derivado, como exemplo, se o complemento de um grafo é cordal.

Algoritmo 6: Algoritmo para reconhecer um grafo cordal (ROSE et al., 1976)

Entrada: um grafo $G = (V, E)$
Saída : uma confirmação se G é ou nao um grafo cordal

```
1 Compute uma enumeração σ para o algoritmo LexBFS
2 se o inverso de σ é uma ordenação de eliminação perfeita então
3 |  concluí-se que G é uma grafo cordal
4 senão
5 |  concluí-se que G não é um grafo cordal
```

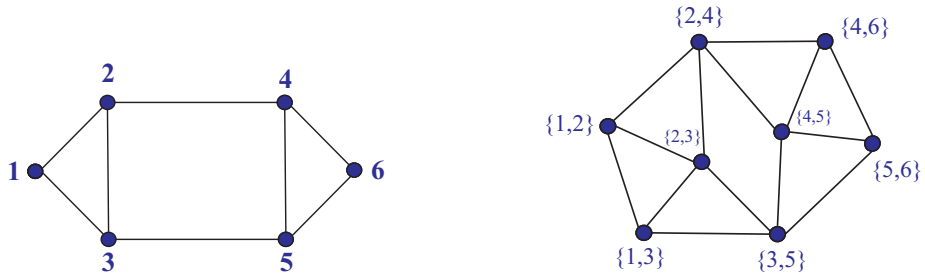
2.3 Definição e Caracterização de Buscas em Hipermultigrafos

Busca é um problema computacional cuja entrada é um grafo e um vértice dele e cuja saída é uma enumeração do conjunto das arestas que contempla todos os vértices alcançáveis a partir do vértice inicial. Muitos algoritmos em grafos empregam algum mecanismo para visitar os vértices e arestas. Após escolher um vértice inicial, a busca em um grafo conexo, visita os vértices e arestas, de tal forma que, uma nova aresta é explorada se, e somente se, é adjacente a alguma outra anteriormente visitada. Nesse ponto podem existir várias arestas a serem escolhidas, e o critério para a escolha é feito de acordo com cada algoritmo de busca. Uma *ordem de busca* é uma enumeração das arestas de E, que foi obtida por um algoritmo de busca. Essa enumeração é uma bijeção, definida pela função $\sigma : \{1, \ldots, m\} \rightarrow E$. Quando do início da execução de um algoritmo de busca temos um conjunto de arestas $(e_1, e_2, \ldots, e_m)$, denominado de arestas não enumeradas, ou não visitadas ou ainda não exploradas. Para cada iteração do algoritmo, temos que uma aresta e é escolhida e essa aresta recebe um rótulo que passa a fazer parte de σ. Portanto, dizemos que uma aresta e é enumerada, ou visitada, ou ainda explorada quando $e = \sigma(i)$ para $1 \leq i \leq m$. Uma *caracterização* consiste em, dada uma enumeração σ das arestas

de E, encontrar propriedades satisfeitas se, e somente se, essa enumeração vem de um determinado algoritmo de busca.

2.4 Grafo Linha

Nessa seção será apresentado um tipo de grafo capaz de representar as características do grafo, por reduzir instâncias de um hipermultigrafo em instância de um grafo. Com isso faz-se necessário a formalização dos conceitos do novo grafo, conhecido na literatura como *grafo linha*. Esse grafo será importante para o processo adaptativo dos teoremas proposto por esta dissertação. Para formalizar a ideia do grafo linha, deve-se explicar alguns conceitos simples. Tendo um grafo G e o grafo linha de G, denotado por G^{L}, temos que (GODSIL; ROYLE, 2001): (i) o conjunto de vértices de G^{L} é o conjunto de arestas de G; (ii) duas arestas de G são adjacentes em G^{L} se, e somente se, são incidentes em G. A Figura 2.1(a) apresenta um grafo e seu respectivo grafo linha na Figura 2.1(b). Cada vértice do grafo linha é mostrado rotulado com um par de vértices, que representa uma aresta no grafo G. Por exemplo, o vértice no grafo linha rotulado de {1,3}, corresponde a aresta no grafo G entre os vértices 1 e 3. O vértice {1,3} no grafo linha é adjacente a três vértices: {1,2}, {2,3} e {3,5}, todos os três vértices são adjacentes ao vértice {1,3} no grafo linha pois possuem adjacências no grafo original.



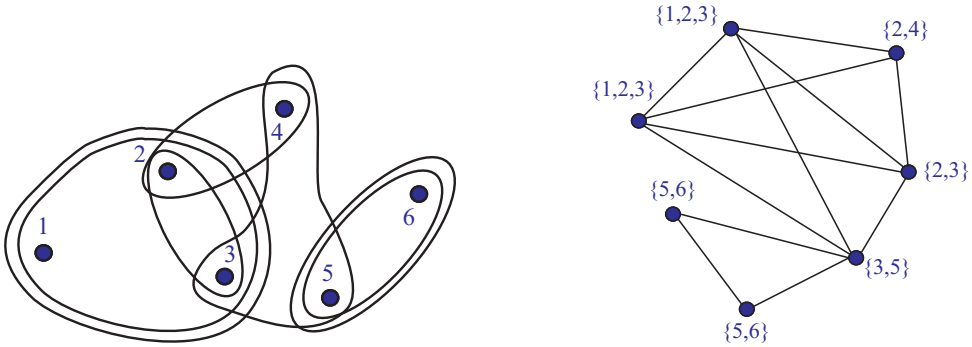
(a) Grafo (b) Grafo Linha

Figura 2.1: Exemplo de um grafo e seu respectivo grafo linha.

FONTE (GODSIL; ROYLE, 2001)

2.5 Representação de Hipermultigrafos em Grafos Linhas

Nessa seção será apresentado como um hipermultigrafo pode ser representado por um tipo particular de grafo. Essa representação será importante para provar os teoremas que caracterizam cada uma das buscas apresentadas no Capítulo 2. Para tal, utilizou-se o conceito de grafo linha para a transformação dos hipermultigrafos, tornando possível a veracidade das provas dos teoremas que serão apresentadas no Capítulo 4. Tomemos como exemplo o hipermultigrafo $H = (\{1,2,3,4,5,6\}, \{\{1,2,3\},\{1,2,3\},\{2,3\},\{2,4\},\{3,4,5\},\{5,6\},\{5,6\}\})$, ilustrado na figura 2.2(a). Para a construção do grafo linha devemos considerar que cada aresta do hipermultigrafo se transformará em um vértice no grafo linha. Observe que, como dito anteriormente, para cada aresta do hipermultigrafo temos um vértice no grafo linha, assim, pode-se verificar que existem múltiplas arestas entre os vértices 1, 2 e 3 e entre os vértices 5 e 6, portanto, será considerado um vértice para cada aresta {1,2,3} e {5,6} do hipermultigrafo, como ilustrado na Figura 2.2(b).



(a) Hipermultigrafo (b) Grafo Linha

Figura 2.2: Exemplo de um grafo linha gerado a partir do hipermultigrafo .

FONTE o autor.

Capítulo 3

Algoritmos de Buscas em Hipermultigrafos

Neste capítulo serão apresentadas adaptações dos algoritmos de buscas de (KRUEGER, 2005) para o caso de hipermutigrafos, como a busca genérica, em largura, em profundidade e também outras buscas como LexBFS, LexDFS e MNS.

3.1 Busca Genérica (BG)

Começaremos por apresentar a busca genérica, um algoritmo simples, que se utiliza de uma estrutura de dados S para armazenar as arestas candidatas a serem escolhidas. Esse algoritmo tem esse nome porque tem a liberdade de escolher qualquer aresta que está em S. Assim, se uma aresta e está em S, então será candidata a ser visitada. Neste trabalho, os algoritmos de busca foram adaptados para executar uma busca conduzida por aresta em um hipermultigrafo, como ilustrado no Algoritmo 7.

Qualquer enumeração de arestas gerada por esse algoritmo é uma ordem de busca em um hipermultigrafo. Isso é possível pelo critério de escolha na linha 6 do Algoritmo 7. Podemos optar pelo uso de uma estrutura de dados mais restrita para obter outras propriedades para a busca, tal como, substituindo S por uma fila para obter uma busca

em largura ou substituir S por uma pilha para obter uma busca em profundidade. Como exemplo, no grafo apresentado na Figura 3.1(a), com vértice inicial e, a enumeração $\sigma_{generica} = (e_5, e_6, e_7, e_{10}, e_2, e_8, e_1, e_{11}, e_4, e_3, e_9)$ é uma ordem de busca válida pela busca genérica. Note que, as arestas e_4, e_3 e e_9 são arestas de não-árvore. A partir da enumeração das arestas e do algoritmo 8, identificando quais delas são de não-árvore, pode-se construir uma enumeração dos vértices, $\sigma_{generica(v)} = (e, c, d, f, g, b, h, a, i)$. Note que, esta enumeração não é BFS nem DFS, isso porque para uma busca em largura após o vértice d ter sido enumerado, o vértice b deveria ter sido escolhido antes do vértice f, e para uma busca em profundidade após o vértice c ter sido enumerado, b deveria ter sido escolhido antes de d. No Capítulo 4 será apresentada propriedades que caracterizam as buscas apresentadas neste trabalho.

Algoritmo 7: Busca Genérica para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  U ← {v_0}
2  i ← 1
3  para e adjacente a v_0 faça
4      adicione a aresta e em S
5  enquanto S ≠ ∅ faça
6      remova uma aresta e de S
7      σ(e) ← i
8      se U ≠ U ∪ e então
9          marque e como aresta de árvore
10         para cada aresta não enumerada f adjacente a e faça
11             se f não está em S então
12                 adicione f em S
13         U ← U ∪ e
14     senão
15         marque e como aresta de não-árvore
16     i ← i + 1
```

Para o multigrafo apresentado na Figura 3.1(b), com vértice inicial g, temos a enumeração $\sigma_{genericaMult} = (e_{11}, e_4, e_2, e_1, e_3, e_5, e_9, e_6, e_7, e_8, e_{10}, e_{15}, e_{14}, e_{16}, e_{12}, e_{13})$ válida das arestas. As arestas $e_1, e_5, e_6, e_8, e_{14}$, e_{16}, e_{12} e e_{13} são arestas de não-árvore. Uma possível enumeração válida dos vértices seria $\sigma_{genericaMult(v)} = (g, d, b, a, c, f, e, h, i)$. Neste con-

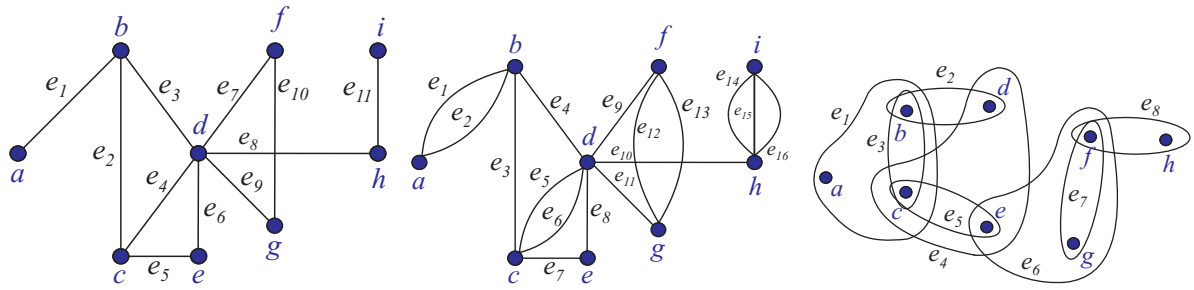
texto pode-se verificar que, para o caso de arestas múltiplas, como as arestas, e_5 e e_6, temos que apenas uma delas será aresta de árvore, a(s) outra(s) será/serão consequêntemente aresta(s) de não-árvore. Como neste trabalho não estamos considerando pesos nas arestas, ou alguma função que diferencie estas múltiplas arestas, e que poderia determinar qual delas poderia ser a mais promissora, temos que, nesse caso, o algoritmo tem a liberdade de escolha de qualquer multiaresta que esteja em S.

Algoritmo 8: Algoritmo para obter uma enumeração de vértices

Entrada: uma enumeração σ E e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ_v sobre o conjunto de vértices

```
1 σ_v({v_0}) ← 0
2 W ← v_0
3 j ← 1
4 enquanto W ≠ V faça
5 |   pegue a aresta f tal que σ(f) = j
6 |   σ_v(f \ W) ← j
7 |   W ← W ∪ (f \ W)
8 |   j ← j + 1
```

Para o hipergrafo apresentado na Figura 3.1(c), com vértice inicial a, uma possível enumeração válida é $\sigma_{genericaHiper} = (e_1, e_5, e_6, e_4, e_7, e_8, e_3, e_2)$. Novamente, temos que, e_7, e_3 e e_2 são arestas de não-árvore. A partir dessa enumeração temos a enumeração $\sigma_{genericaHiper(v)} = (a, b, c, e, g, f, d, h)$ dos vértices. Observe que, a partir do vértice inicial a tem-se que e_1 será a primeira aresta escolhida e assim os vértices b e c serão alcançados simultaneamente e consequentemente terão o mesmo valor na enumeração. Portanto, para o caso da representação sequencial dos vértices, utilizaremos a notação de conjunto, portanto temos a enumeração, $\sigma_{genericaHiper(v)} = (\{a\}, \{c, b\}, \{e\}, \{g, f\}, \{d\}, \{h\})$ para o hipergrafo da Figura 3.1(c).



(a) Grafo (b) Multigrafo (c) Hipergrafo

Figura 3.1: Exemplo a ser utilizado para obter uma sequência de arestas.

3.2 Busca em Largura (BFS)

Busca em largura é uma restrição da busca genérica que utiliza uma fila para armazenar e determinar qual aresta será a próxima escolhida, como pode ser observado na linha 5 do Algoritmo 9. Este algoritmo é a base para muitos algoritmos em grafos, tais como o algoritmo de *Prim* para obter a árvore geradora mínima e o algoritmo de *Dijkstra* para obter o caminho mais curto de um vértice a todos os outros vértices. BFS tem sido muito utilizado em várias áreas desde meados do século XX. Segundo (KRUEGER, 2005), BFS foi originalmente apresentado por Moore (MOORE, 1959) em um artigo intitulado *The shortest path through a maze* e tem sido aplicado em muitos problemas de grafos, incluíndo caminhos mínimos (EVEN, 1979), redes de fluxo (EDMONDS; KARP, 1970), estimativa de diâmetro (CORNEIL et al., 2003) e no reconhecimento de várias classes de grafos, tal como, grafos de intervalos unitários (CORNEIL et al., 1995). O Algoritmo 9 mostra a busca em largura por arestas.

Como exemplo, o algoritmo de busca em largura pode visitar as arestas da Figura 3.1(a), com vértice inicial b, na ordem $\sigma_{BFS} = (e_3, e_1, e_2, e_7, e_6, e_9, e_8, e_{11})$[1]. A partir da enumeração das arestas, e do Algoritmo 8, pode-se construir uma enumeração dos vértices, $\sigma_{BFS(v)} = (b, d, a, c, f, e, g, h, i)$. Como apresentado na Figura 3.2, percebe-se que as arestas de não-árvore[2] estão no máximo um nível abaixo (ou acima) na árvore BFS. Isso porque para uma busca em largura os vértices visitados são sempre aqueles menos recentemente alcançados na busca, assim, em hipótese alguma, em uma busca

[1] A partir de agora as enumerações serão apresentadas sem as arestas de não-árvore.
[2] Arestas pontilhadas indicam arestas de não-árvore em G e as linhas indicam a direção do percurso.

BFS tem-se uma diferença maior que dois na árvore BFS.

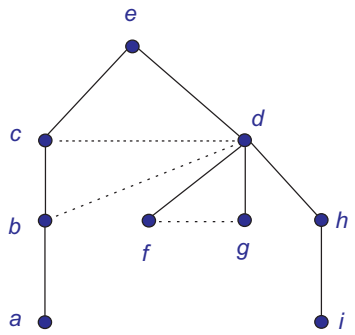


Figura 3.2: Exemplo de uma árvore de busca BFS do exemplo ilustrado na Figura 3.1(a).

Algoritmo 9: Busca em Largura para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  U ← {v_0}
2  i ← 1
3  para e adjacente a v_0 faça
4      enfileira a aresta e
5  enquanto Fila ≠ ∅ faça
6      desenfila uma aresta e
7      σ(e) ← i
8      se U ≠ U ∪ e então
9          marque e como aresta de árvore
10         para cada aresta não enumerada f adjacente a e faça
11             se f não está na fila então
12                 enfileira f
13         U ← U ∪ e
14     senão
15         marque e como aresta de não-árvore
16     i ← i + 1
```

Para o caso de um multigrafo, como ilustrado na Figura 3.1(b) a enumeração $\sigma_{BFSMulti} = (e_4, e_2, e_3, e_8, e_{10}, e_9, e_{11}, e_{14})$ é uma ordem de busca válida pela busca em largura apresentada pelo Algoritmo 9. Uma enumeração dos vértices válida pode ser dada por $\sigma_{BFSMulti(v)} = (b, d, a, c, e, h, f, g, i)$. Para o caso de hipergrafo, como ilustra a Figura 3.1(c), com vértice inicial a, a enumeração $\sigma_{BFSHiper} = (e_1, e_2, e_4, e_6, e_8)$ é uma ordem de busca válida e a enumeração dos vértices pode ser dada por $\sigma_{BFSHiper(v)} = (\{a\}, \{b, c\}, \{d\}, \{e\}, \{g, f\}, \{h\})$.

3.3 Busca em Profundidade (DFS)

Busca em largura e em profundidade são duas das mais fundamentais estratégias de algoritmos em grafos. A busca em profundidade utiliza uma pilha para armazenar as arestas candidatas a serem visitadas. Nesta busca, as arestas são exploradas a partir da aresta mais recentemente descoberta ainda inexploradas. Quando todas as arestas $(e_1, e_2, e_3, \ldots, e_i)$ adjacentes a f tiverem sido exploradas, a busca regressa[3] um nível para explorar arestas que ainda não foram visitadas. Esse processo continua até que todas as arestas terem sidos descobertas a partir da aresta inicial. Apresentamos a busca em profundidade mostrado no Algoritmo 10.

Segundo (KRUEGER, 2005) DFS apresenta duas enumerações de vértices: o processo de enumeração, que é a ordem em que os vértices são visitados, e uma enumeração de fim, que indica a ordem em que cada vizinhança de um vértice foi completamente explorada, causando processo de retrocesso.

Como exemplo, o algoritmo de busca em profundidade pode visitar as arestas da Figura 3.1(a), com vértice inicial b, na ordem $\sigma_{DFS} = (e_3, e_9, e_{10}, e_8, e_{11}, e_6, e_5, e_1)$. A partir da enumeração das arestas, identificando quais delas são de árvore, pode-se construir uma enumeração dos vértices, $\sigma_{DFS(v)} = (b, d, g, f, h, i, e, c, a)$. Para o caso de um multigrafo, como ilustrado na Figura 3.1(b), com vértice inicial a, a enumeração $\sigma_{DFSMulti} = (e_1, e_4, e_{11}, e_{13}, e_{10}, e_{16}, e_8, e_7)$ é uma ordem de busca válida pela busca em profundidade apresentada pelo Algoritmo 10. Uma enumeração dos vértices válida pode ser dada por $\sigma_{DFSMulti(v)} = (a, b, d, g, f, h, i, e, c)$. Para o caso do hipergrafo, como ilustra a Figura 3.1(c) a enumeração $\sigma_{BFSHiper} = (e_1, e_4, e_6, e_8)$ é uma ordem de busca válida e a enumeração dos vértices pode ser dada por $\sigma_{BFSHiper(v)} = (a, c, e, f, h, g, d, b)$.

[3]do inglês *backtrack*.

Algoritmo 10: Busca em Profundidade em Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  U ← {v_0}
2  i ← 1
3  para e adjacente a v_0 faça
4      empilha a aresta e
5  enquanto Pilha ≠ ∅ faça
6      desempilha uma aresta e
7      σ(e) ← i
8      se U ≠ U ∪ e então
9          marque e como aresta de árvore
10         para cada aresta não enumerada f adjacente a e faça
11             se f já está na pilha então
12                 desempilha f
13             empilha f
14         U ← U ∪ e
15     senão
16         marque e como aresta de não-árvore
17     i ← i + 1
```

3.4 Busca em Largura Lexicográfica (LexBFS)

A busca em largura lexicográfica é uma restrição da busca em largura, em que a preferência de escolha é dada para um vértice v tal que os vizinhos de v foram visitados pela busca (ROSE et al., 1976). Existem diferentes formas de implementação do LexBFS. O termo *lexicográfico* significa que a busca será orientada através de rótulos, assim todos os vértices não enumerados adjacentes a v receberão um rótulo. Um vértice é escolhido tal que ele tem o maior rótulo lexicográfico. Uma outra conceituação do LexBFS envolve pivô e particionamento (ver (HABIB et al., 2000) para maiores detalhes). Recentemente, o algoritmo LexBFS tem sido mostrado ser uma ferramenta poderosa para o desenvolvimento em tempo linear para várias famílias de grafos. Uma das razões para sua utilidade é que existe uma caracterização simples da enumeração dos vértices (BRANDSTÄDT et al., 1997). Esta caracterização é crucial para provas de corretude para vários algoritmos. A busca da cardinalidade máxima (MCS) de Tarjan e Yannakakis (TARJAN; YANNAKAKIS, 1984) tem uma caracterização similar como será apresentada na Seção 3.7. A

técnica utilizada pelo LexBFS para construir uma enumeração dos vértices, caracterizando propriedades em grafos foi primeiramente introduzido por Rose, Tarjan e Lueker (ROSE et al., 1976) em 1976 para reconhecimento de grafos cordais. Mais recentemente este algoritmo foi utilizado para reconhecer grafos intervalar (HABIB et al., 2000), grafos intervalar única (CORNEIL, 2004b; HELL; HUANG, 2004/05) e permutações bipartidas (CHANG et al., 1999; HELL; HUANG, 2004/05).

A rotulação de uma aresta é feita como segue:

Inicialmente atribui-se o rótulo zero para todas as arestas, e para cada aresta e escolhida, aquelas que são adjacentes a ela tem seu rótulo modificado, sendo adicionado ao final de seu rótulo o valor $(m - i)$, onde m é o número de arestas do grafo e i é o número da iteração do algoritmo.

Quadro 3.1: Ideia do Algoritmo 11.

O Algoritmo 11 ilustra a busca LexBFS por arestas.

Algoritmo 11: Busca em Largura Lexicográfica para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  Atribui o rótulo zero para todas as arestas de G
2  U ← {v_0}
3  i ← 1
4  para e adjacente a v_0 faça
5      coloque o valor m para o rótulo da aresta e
6  para i ← 1 até m faça
7      pegue uma aresta não enumerada e com maior rótulo lexicográfico
8      σ(e) ← i
9      se U ≠ U ∪ e então
10         marque e como aresta de árvore
11         para cada aresta não enumerada f adjacente a e faça
12             Coloque como sufixo o valor (m − i) para o rótulo da aresta f
13         U ← U ∪ e
14     senão
15         marque e como aresta de não-árvore
```

Observe que, o grafo apresentado na Figura 3.1(a), com vértice inicial b, a enumeração $\sigma_{LexBFS} = (e_3, e_1, e_2, e_7, e_9, e_8, e_6, e_{11})$ é uma ordem de busca em largura lexicográfica válida. Note que, após a aresta e_1 ter sido enumerada, a única opção de escolha é a

aresta e_2, uma vez que, rótulo(e_2) = (9), rótulo(e_4) = (8), rótulo(e_6) = (8), rótulo(e_7) = (8), rótulo(e_8) = (8) e rótulo(e_9) = (8). Note também que após a aresta e_8 ter sido enumerada, temos que, rótulo(e_{10}) = (43) e rótulo(e_{11}) = (2), portanto e_{10} será escolhida antes de e_{11}. Esta enumeração também é uma ordem de busca em largura válida. Para ilustrar a diferença entre a busca em largura e a busca em largura lexicográfica note que a enumeração σ_{BFS}, já apresentada na Seção 3.2, não é uma busca em largura lexicográfica válida, isso porque, em uma ordem de busca em largura lexicográfica as arestas e_4 e e_5 devem ser escolhida antes da aresta e_7. Observe a enumeração dos vértices $\sigma_{LexBFS(v)} = (b, d, a, c, f, g, h, e, i)$.

Enumeração para multigrafo e hipergrafo das Figuras 3.1(b) 3.1(c):

- $\sigma_{LexBFSMult} = (e_4, e_2, e_3, e_9, e_{10}, e_{11}, e_8, e_{14})$. Note que, após a aresta e_8 ter sido enumerada a única opção é a escolha da aresta e_7, isso porque, rótulo(e_7) = (62), rótulo(e_{12}) = (53), rótulo(e_{13}) = (53), rótulo(e_{14}) = (4), rótulo(e_{15}) = (4) e rótulo(e_{16}) = (4). A enumeração dos vértices pode ser dada por $\sigma_{LexBFSMult(v)} = (b, d, a, c, f, h, g, e, i)$.

- $\sigma_{LexBFSHiper} = (e_3, e_2, e_1, e_4, e_6, e_8)$. Enumerando os vértices a partir das arestas temos $\sigma_{LexBFSHiper(v)} = (b, c, d, a, e, g, f, h)$.

3.5 Busca em Profundidade Lexicográfica (LexDFS)

A busca em profundidade lexicográfia foi proposta por Krueger (KRUEGER, 2005) em sua tese de doutorado. Esse algoritmo é bastante similar ao algoritmo de busca em largura lexicográfica. Analogamente ao LexBFS, a rotulação de uma aresta no LexDFS é feita da seguinte forma:

Inicialmente atribui-se o rótulo zero para todas as arestas, e para cada aresta e escolhida, aquelas que são adjacentes a ela tem seu rótulo modificado, sendo adicionado ao início de seu rótulo o valor (i), onde i é o número da iteração do algoritmo.

Quadro 3.2: Ideia do Algoritmo 12.

O Algoritmo 12 ilustra a busca por aresta em profundidade lexicográfica. Observe que, o grafo apresentado na Figura 3.1(a), com vértice inicial b, a enumeração $\sigma_{LexDFS} = (e_3, e_7, e_{10}, e_8, e_{11}, e_6, e_5, e_1)$ é uma ordem de busca em profundidade lexicográfica válida. Note que, após a aresta e_8 ter sido enumerada, a única opção de escolha é a aresta e_{11}, uma vez que, rótulo$(e_{11}) = (5)$, rótulo$(e_4) = (2)$, rótulo$(e_6) = (2)$, rótulo$(e_1) = (2)$ e rótulo$(e_1) = (1)$. Note também que após a aresta e_6 ter sido enumerada e_5 será escolhida antes de e_2 e e_4, isso porque, rótulo$(e_5) = (6)$, rótulo$(e_4) = (2)$ e rótulo$(e_2) = (1)$. Esta enumeração também é uma ordem de busca em profundidade válida. Observe que a enumeração $\sigma_{DFS} = (e_3, e_9, e_{10}, e_8, e_{11}, e_6, e_5, e_1)$, já apresentada na Seção 3.3 também é uma ordem de busca em profundidade lexicográfica válida.

Algoritmo 12: Busca em Profundidade Lexicográfica para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  Atribui o rótulo zero para todas as arestas de G
2  U ← {v_0}
3  i ← 1
4  para e adjacente a v_0 faça
5      coloque o valor i para o rótulo da aresta e
6  para i ← 1 até m faça
7      pegue uma aresta não enumerada e com maior rótulo lexicográfico
8      σ(e) ← i
9      se U ≠ U ∪ e então
10         marque e como aresta de árvore
11         para cada aresta não enumerada f adjacente a e faça
12             Coloque como prefixo o valor i para o rótulo da aresta f
13         U ← U ∪ e
14     senão
15         marque e como aresta de não-árvore
```

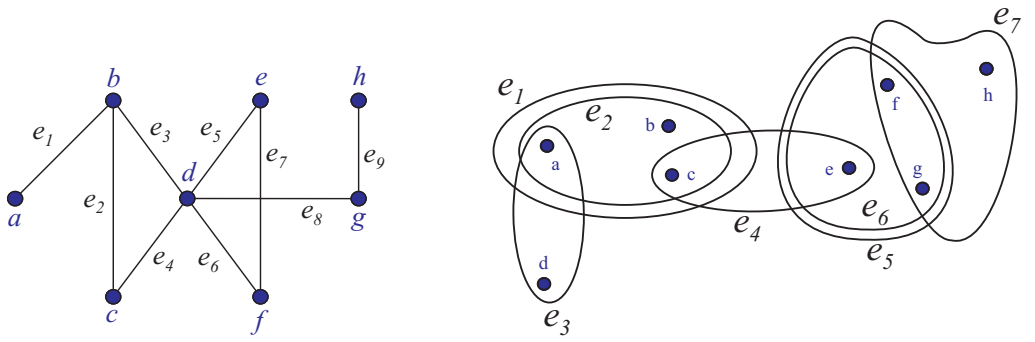
Enumeração para multigrafo e hipergrafo das Figuras 3.1(b) e 3.1(c):

- $\sigma_{LexDFSMult} = (e_3, e_6, e_9, e_{12}, e_{10}, e_{16}, e_8, e_1)$. Após a aresta e_{12} ter sido enumerada a única opção de escolha é a aresta e_{11}, uma vez que, rótulo$(e_{11}) = (35)$, rótulo$(e_{10}) = (3)$, rótulo$(e_8) = (3)$, rótulo$(e_5) = (2)$ e rótulo$(e_{13}) = (4)$. Note também que após a aresta e_{16} ter sido enumerada e_8 será escolhida antes da e_7.

- $\sigma_{LexDFSHiper} = (e_8, e_7, e_6, e_5, e_4, e_3, e_1)$.

3.6 Busca da Vizinhança Maximal (MNS)

O algoritmo busca da vizinhança maximal (MNS)[4], pode ser descrito como segue: a cada passo, escolha uma aresta na qual a vizinhança visitada, ou seja, aqueles vizinhos que já foram visitados, é maximal sobre todas as arestas não visitadas. Segundo Corneil e Krueger (CORNEIL; KRUEGER, 2005), este algoritmo tem se mostrado importante para encontrar esquemas da eliminação perfeita e triangulações minimais para grafos arbitrários. Este algoritmo, assim como o LexBFS e LexDFS, utiliza rótulos para escolher qual será a próxima aresta a ser visitada, entretanto, esses rótulos são representados através de conjunto. Tanto LexBFS quanto LexDFS terá uma vizinhança maximal, assim esses algoritmos são uma restrição do MNS. O Algoritmo 13 ilustra a busca da vizinhança maximal por arestas.



(a) Grafo (b) Hipermultigrafo

Figura 3.3: Exemplo a ser utilizado para obter uma sequência de arestas para o MNS.

Como exemplo, no grafo ilustrado na Figura 3.3(a), a enumeração $\sigma_{MNS} = (e_3, e_5, e_7, e_8, e_9, e_4, e_1)$ é uma ordem de busca da vizinhança maximal válida. Observe que, após a aresta e_7 ter sido enumerada a aresta e_6 deve ser escolhida antes das arestas e_4 e e_8, isso porque, rótulo$(e_4) = \{2\}$, rótulo$(e_8) = \{2\}$ e rótulo$(e_6) = \{2, 4\}$, assim o rótulo das arestas e_4 e e_8 está contido na aresta e_6. Para o hipermultigrafo apresentado na Figura 3.3, a enumeração $\sigma_{MNSHiper} = (e_3, e_1, e_4, e_6, e_7)$ é uma ordem de busca da vizinhança maximal

[4]Do inglês *Maximal Neighbourhood Search.*

válida.

Algoritmo 13: Busca da Vizinhança Maximal para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  Atribui o rótulo zero para todas as arestas de G
2  escolha uma aresta e adjacente a v_0
3  rótulo(e) ← m
4  U ← {v_0}
5  i ← 1
6  para e adjacente a v_0 faça
7      coloque o valor i para o rótulo da aresta e
8  para i ← 1 até m faça
9      pegue uma aresta não enumerada f com rótulo maximal
10     σ(e) ← i
11     se U ≠ U ∪ e então
12         marque e como aresta de árvore
13         para cada aresta não enumerada f adjacente a e faça
14             rótulo(f) ← rótulo(f) ∪ {i}
15         U ← U ∪ e
16     senão
17         marque e como aresta de não-árvore
```

3.7 Busca da Cardinalidade Máxima (MCS)

Muitos problemas importantes na teoria dos grafos dependem em computar um grafo cordal ou, equivalentemente, um grafo triangularizado. Geralmente a meta é computar uma *triangulação minimal*, que é uma triangularização com o menor número de arestas. Algoritmos como Lex-M e MCS-M (derivados dos algoritmos LexBFS e MCS, respectivamente e utilizados para reconhecimentos de grafos cordais) são utilizados para computar triangularização minimal em um dado grafo. Segundo (BERRY et al., 2004) computar triangularização minimal é NP-Dificíl (YANNAKAKIS, 1981). O teorema a seguir caracteriza triangularização minimal:

Teorema 3.7.1 ((ROSE et al., 1976))**:** *Dado um grafo $G = (V, E)$, uma triangularização $H = (V, E\ U\ F)$ de G, é minimal se, e apenas se, toda aresta em F tem uma única corda em um ciclo de tamanho quatro em H.*

Teorema 3.7.2 ((ROSE et al., 1976))**:** *Dado um grafo não direcionado $G = (V, E)$, é um grafo de eliminação perfeita se, e somente se, é triangularizado.*

Para maiores informações sobre grafo triangularizado e triangulações minimais consutar (ROSE et al., 1976; BERRY et al., 2004).

O algoritmo busca da cardinalidade máxima, ilustrado na Figura 14 é um simples algoritmo de tempo linear que primeiro processa um vértice arbitrário v atribuindo a ele o peso $w(v) = n$. MCS mantém para cada aresta e não visitada um peso inteiro $w(e)$ que é a cardinalidade dos vizinhos já processados. Na iteração i um vértice v com rótulo de peso máximo é escolhido e recebe o número i, sendo que i é adicionado ao final dos rótulos de todos os vizinhos não enumerados de v. MCS produz um esquema da eliminação perfeita quando, dado um grafo cordal como entrada (BERRY et al., 2004).

Algoritmo 14: Busca da Cardinalidade Máxima para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$ e um vértice inicial $v_0 \in$ V
Saída : uma enumeração σ de E

```
1  w(e) ← 0, ∀ e ∈ E
2  escolha uma aresta e adjacente a v_0
3  w(e) ← m
4  U ← {v_0}
5  i ← 1
6  para i ← 1 até m faça
7      pegue uma aresta não enumerada f com peso máximo
8      σ(e) ← i
9      se U ≠ U ∪ e então
10         marque e como aresta de árvore
11         para cada aresta não enumerada f adjacente a e faça
12             w(f) ← w(f) + 1
13         U ← U ∪ e
14     senão
15         marque e como aresta de não-árvore
```

Capítulo 4

Caracterizações de Buscas em Hipermultigrafos

Neste capítulo serão apresentadas características das buscas em hipermultigrafos da perspectiva da ordem de busca por arestas. O foco será em torno da questão:

- Para uma tripla de arestas $a \prec b \prec c$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ (ver Figura 4.1), como poderia um algoritmo de busca, apresentado no Capítulo 3, ter escolhido a aresta b antes da aresta c?

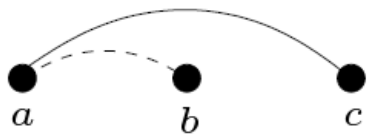


Figura 4.1: Uma tripla de arestas para uma ordem de busca.

Pode-se observar, através da Figura 4.1, a linha tracejada indica a não existência de vértices em comum entre as arestas a e b, que, após o algoritmo ter escolhido a aresta a, que é adjacente a c mas não a b, a aresta c deveria ter sido escolhida antes da aresta b. Dessa forma o que se pode deduzir da Figura 4.1 de modo que a busca poderia ter

escolhido b antes de c? Essa questão nos leva a uma caracterização para cada um dos paradigmas de busca apresentados no Capítulo 3.

4.1 Caracterização da Busca Genérica

Questão sobre a ordem de busca das arestas 1: *Para uma tripla de arestas $a \prec b \prec c$ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, como poderia a aresta b ter sido escolhida antes da aresta c pela busca genérica? A resposta para esta questão é mostrada na Propriedade 4.1.1, como ilustra a Figura 4.2*

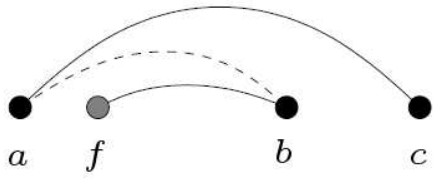


Figura 4.2: Exemplo para uma ordem de busca para a busca genérica.

Propriedade 4.1.1: *Dada uma enumeração σ de E, que contempla todos os vértices em V, se a aresta $a \prec b \prec c$ e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então existe uma aresta f tal que $f \prec b$ e $f \cap b \neq \varnothing$.*

Teorema 4.1.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca genérica das arestas de H se, e somente se, σ tem a Propriedade 4.1.1.*

Prova. Vamos assumir que σ é uma ordem de busca genérica gerada pelo Algoritmo 7. Sejam as arestas a, b, c tais que $a \prec b \prec c$ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$. Quando b foi escolhida, ambas as arestas b e c deviam estar em S. Como b está em S, então, alguma aresta adjacente a b já deve ter sido escolhida. Chamemos essa aresta de f. Assim, existe uma aresta f adjacente a b tal que $f \prec b$ e $f \cap b \neq \varnothing$. Portanto, como σ é uma ordem de busca genérica, então σ tem a Propriedade 4.1.1.

Por contradição, suponhamos que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.1.1, mas não é uma ordem de busca genérica em H, ou seja, para algum $2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 7, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo algoritmo. Considere a próxima iteração do algoritmo após escolher e_{i-1}. Como e_i não pode ser escolhida, então $e_i \notin S$. Seja h a próxima aresta escolhida pelo algoritmo. Então, h deve estar em S. Seja g a aresta adjacente a h que causou sua adição em S. Como $e_i \notin S$, sabemos que $g \cap e_i = \varnothing$. Aplicando a Propriedade 4.1.1 para a tripla de arestas $(g \prec e_i \prec h)$, existe uma aresta f tal que $f \prec e_i$ e $f \cap e_i \neq \varnothing$. Portanto, $e_i \in S$, contradizendo a suposição de que e_i não poderia ser a próxima aresta escolhida. □

4.2 Caracterização da Busca em Largura

Questão sobre a ordem de busca das arestas 2: *Para uma tripla de arestas $a \prec b \prec c$ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, como poderia a aresta b ter sido escolhida antes da aresta c pela busca em largura? A resposta é uma simples restrição da demonstração apresentada pela caracterização da busca genérica, como ilustra a Figura 4.3.*

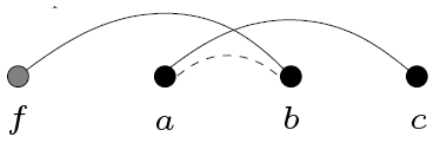


Figura 4.3: Exemplo para uma ordem de busca em largura.

Propriedade 4.2.1: *Dada uma enumeração σ de E, que contempla todos os vértices de V, se a aresta $a \prec b \prec c$, e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então existe uma aresta f tal que $f \prec a$ e $f \cap b \neq \varnothing$.*

Teorema 4.2.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca em largura das arestas de H se, e somente se, σ tem a Propriedade 4.2.1*

PROVA. Suponha que σ seja uma ordem de busca BFS. Suponha a tripla de arestas $a \prec b \prec c$ de σ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$. Se a aresta b foi removida da fila antes da aresta c, então deve existir uma aresta adjacente a b, nomeada de f, que foi enumerada antes que a em σ. Como a aresta a não é adjacente a b, então f deve estar antes que a em σ. Portanto, como σ é uma ordem de busca em largura, então σ tem a Propriedade 4.2.1.

Por contradição, suponha que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.2.1, mas não é uma ordem de busca em largura em H, ou seja, para algum $2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 9, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo algoritmo. Seja h a aresta que a busca em largura pode escolher. Deve existir uma aresta g que foi enumerada antes de e_i adjacente a h mas não a e_i, escolha g para ser a aresta mais a esquerda em σ para a tripla de arestas. Aplicando a Propriedade 4.2.1 na tripla $(g \prec e_i \prec h)$, deve existir uma aresta $f \prec g$ adjacente a e_i. Portanto, existe uma aresta à esquerda de g adjacente a e_i chamada de f. Assim a busca BFS poderia escolher e_i antes de h, contradizendo a suposição acima. □

4.3 Caracterização da Busca em Profundidade

Questão sobre a ordem de busca das arestas 3: *Dada uma ordem de busca em profundidade com $a \prec b \prec c$, $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, como poderia a aresta b ter sido escolhida antes da aresta c pela busca DFS? A resposta está na Propriedade 4.3.1 que caracteriza uma ordem de busca em profundidade, como ilustra a Figura 4.4.*

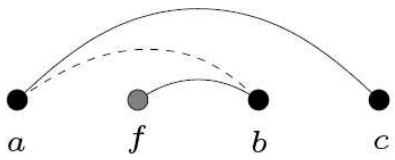


Figura 4.4: Exemplo para uma ordem de busca em profundidade.

Propriedade 4.3.1: *Dada uma enumeração σ de E, que contempla todos os vértices de V, se a aresta $a \prec b \prec c$ e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então existe uma aresta f tal que $a \prec f \prec b$ e $f \cap b \neq \varnothing$.*

Teorema 4.3.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca em profundidade das arestas de H se, e somente se, σ tem a Propriedade 4.3.1.*

Prova. Suponha que σ seja uma enumeração DFS. Suponha também a tripla de arestas $a \prec b \prec c$ de σ, $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$. Quando a aresta b foi escolhida, b estava na pilha. Como b não foi a primeira aresta escolhida pela algoritmo, alguma aresta adjacente a b já deva ter sido escolhida. Seja f a aresta adjacente a b dentre todas as arestas que precede b em σ. Assim, $f \prec b$ e $f \cap b \neq \varnothing$.

Observe que, entre as arestas a e b, podem existir várias outras que precedem b e sucedem a em σ. No entanto, a deve ser adjacente à aresta mais à esquerda dentre todas as arestas que precedem b em σ. Portanto, deve existir um caminho $(a \prec x_1 \prec x_2 \prec \cdots \prec f)$ que sai da aresta a e chega na aresta f.

Suponha que $f \prec a$. Neste ponto, sabemos que a foi escolhida depois de f, e que a aresta a foi substituída na pilha por c. Como não existe nenhuma aresta adjacente a b antes dela que poderia ter sido escolhida, temos que c foi escolhida antes de b, pois está acima de b na pilha, uma contradição.

Por contradição, suponha que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.3.1, mas não é uma ordem de busca em profundidade em H, ou seja, para algum $2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 10, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo algoritmo. Seja h a aresta que a busca em profundidade pode escolher e seja e_j a aresta mais a direita em σ_{i-1}, tal que $e_j \cap h \neq \varnothing$. Claramente, $e_j \cap e_i = \varnothing$ e mais especificamente, $e_k \cap e_i = \varnothing$, para todo k, $j \leq k < i$. Por aplicar a Propriedade $\mathcal{P}3$ na tripla de arestas $(e_j \prec e_i \prec h)$, sabemos que existe uma aresta f entre as arestas e_j e e_i tal que, $f \cap e_i \neq \varnothing$, portanto,

uma contradição. □

4.4 Caracterização da Busca em Largura Lexicográfica

Questão sobre a ordem de busca das arestas 4: *Tendo uma ordem de busca em largura lexicográfica com $a \prec b \prec c$, $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, como poderia a aresta b ter sido escolhida antes da aresta c pela busca em largura lexicográfica? Nesse caso temos a Propriedade 4.4.1 adicionado a restrição de que $f \cap c = \varnothing$, como ilustra a Figura 4.5.*

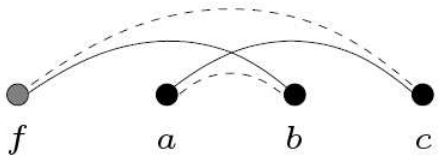


Figura 4.5: Exemplo para uma ordem de busca em largura lexicográfica.

Propriedade 4.4.1: *Dada uma enumeração σ de E que contempla todos os vértices de V, se a aresta $a \prec b \prec c$ e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então existe uma aresta $f \prec a$ tal que $f \cap b \neq \varnothing$ e $f \cap c = \varnothing$*[1].

Teorema 4.4.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca em largura lexicográfica das arestas de H se, e somente se, σ tem a Propriedade 4.4.1.*

Prova. Suponha que σ seja uma enumeração em largura lexicográfica. Suponha a tripla de arestas $a \prec b \prec c$ de σ com $f \cap c \neq \varnothing$. Quando a aresta a for escolhida, a próxima aresta a ser enumerada será aquela que tiver maior rótulo lexicográfico. Assim, c será escolhida antes de b, pois terá os valores dos rótulos das arestas de f e de a, enquanto b terá apenas o valor do rótulo de f. Portanto, $f \cap c = \varnothing$.

Suponha, por contradição, que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.4.1, mas não é uma ordem de busca em largura lexicográfica de H, ou seja, para algum

[1]Esta propriedade diz que se $f \cap c \neq \varnothing$, então c será escolhida antes de b. Como a aresta a é adjacente a c e não a b, isso quer dizer que c será escolhida antes de b, então deve existir uma aresta antes de a na qual causou b ser escolhida antes de c.

$2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 11, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo LexBFS. Seja h a próxima aresta escolhida, então deve existir uma aresta g tal que $g \prec e_i$, adjacente a h mas não a e_i. Escolha g para ser a aresta mais à direita em σ_{i-1}. Aplicando a Propriedade 4.4.1 na tripla $(g \prec e_i \prec h)$, deve existir uma aresta à esquerda de g, nomeada f, na qual $f \prec g$ adjacente a e_i mas não a h. Assim, $f \cap e_i \neq \varnothing$ e $f \cap h = \varnothing$. Portanto, a busca em largura lexicográfica poderia escolher e_i antes de h, contradizendo a suposição acima. □

4.5 Caracterização da Busca em Profundidade Lexicográfica

Questão sobre a ordem de busca das arestas 5: *Para uma ordem de busca em profundidade lexicográfica com $a \prec b \prec c$, $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, a aresta b, apenas poderia ter sido escolhida antes da aresta c se adicionarmos na Propriedade 4.5.1 a restrição que $f \cap c = \varnothing$, como ilustra a Figura 4.6*

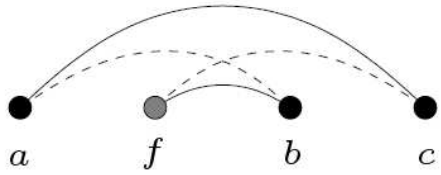


Figura 4.6: Exemplo para uma ordem de busca em profundidade lexicográfica.

Propriedade 4.5.1: *Dada uma enumeração σ de E que contempla todos os vértices de V, se a aresta $a \prec b \prec c$ e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então, existe uma aresta f com, $a \prec f \prec b$ tal que $f \cap b \neq \varnothing$ e $f \cap c = \varnothing$.*

Teorema 4.5.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca em profundidade lexicográfica das arestas de H se, e somente se, σ tem a Propriedade 4.5.1.*

Prova. Suponha que σ é uma enumeração gerada pela busca em profundidade lexicográfica. Seja $a \prec b \prec c$ uma tripla de arestas em σ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$. Para

que a aresta b tenha sido escolhida antes de c, rótulo(b) $\geq$ rótulo(c), isso quer dizer que a aresta b deve possuir um rótulo lexicograficamente maior ou igual ao rótulo de c. Como rótulo(a) está contido no rótulo(c) mas não no rótulo(b), então rótulo(b) deve conter algum outro rótulo maior que o rótulo(a) e que não está contido no rótulo(c). Seja f a aresta que fornecerá o rótulo para a aresta b. Assim, $f \cap b \neq \varnothing$ e $f \cap c = \varnothing$. Como a aresta f fornecerá um rótulo para b maior que a aresta a forneceu para c, então $a \prec f$. Como σ é uma ordem de busca em profundidade lexicográfica então, a Propriedade 4.5.1 é satisfeita.

Por contradição, suponha que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.5.1, mas não é uma ordem de busca em profundidade lexicográfica de H, ou seja, para algum $2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 12, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo LexDFS. Seja h a aresta que a busca em profundidade lexicográfica pode escolher. Desde que h tenha sido escolhida antes de e_i, rótulo(h) deve ser lexicograficamente maior ou igual que o rótulo(e_i). Seja j o maior dígito no rótulo(h) e que não está contido no rótulo(e_i). Isso é, e_j é a aresta mais à direita em σ_{i-1} com $e_j \cap h \neq \varnothing$ e $e_j \cap e_i = \varnothing$. Por aplicar a Propriedade 4.5.1 na tripla de arestas $(e_j \prec e_i \prec h)$, sabe-se que existe uma aresta f entre e_j e e_i tal que $f \cap e_i \neq \varnothing$ e $f \cap h = \varnothing$. Desde que $e_j \prec f$, rótulo(e_i) $>$ rótulo(h), então e_i deveria ter sido escolhida antes de h, uma contradição. □

4.6 Caracterização da Busca da Vizinhança Maximal

Questão sobre a ordem de busca das arestas 6: *Para a Propriedade 4.6.1, é adicionado a restrição que $f \cap c = \varnothing$ na Propriedade 4.1.1, como ilustra a Figura 4.7. Assim, a aresta b poderia ter sido escolhida antes da aresta c pela busca da vizinhança maximal.*

Propriedade 4.6.1: *Dada uma enumeração σ de E que contempla todos os vértices de V, se a aresta $a \prec b \prec c$, e $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então, existe uma aresta $f \prec b$ tal que $f \cap b \neq \varnothing$ e $f \cap c = \varnothing$.*

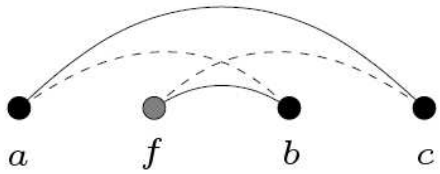


Figura 4.7: Exemplo para uma ordem de busca da vizinhança maximal.

Teorema 4.6.2: *Dado um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca da vizinhança maximal das arestas de H se, e somente se, σ tem a Propriedade 4.6.1.*

PROVA. Por contradição, suponha que σ é uma ordem de busca da vizinhança maximal. Suponha que $a \prec b \prec c$ seja uma tripla em σ que viola a Propriedade 4.6.1, isto é:

$$\nexists f \prec b, \text{ tal que } f \cap b \neq \varnothing \text{ e } f \cap c = \varnothing \tag{4.1}$$

Considere o ponto de execução quando $(b = e_i)$ é escolhida. Então $N(b) \subseteq N(c)$, assim b não poderia ter sido escolhida pela busca, uma contradição. Conclui-se que:

$$\forall f \prec b \text{ se } f \cap b \neq \varnothing \text{ então } d \cap c = \varnothing \tag{4.2}$$

Suponha que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.6.1, mas não é uma ordem de busca da vizinhança maximal de H, ou seja, para algum $2 \leq i \leq m$, $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 13, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo MNS. Seja: (i) h a próxima aresta a ser escolhida, assim, $N(e_i) \subseteq N(h)$; (ii) $g \in (N(h) - N(e_i))$. Por aplicar a Propriedade 4.6.1 na tripla de arestas $(g \prec e_i \prec h)$, tem-se a aresta $f \prec e_i$ tal que $f \cap e_i \neq \varnothing$ e $f \cap h = \varnothing$. Portanto, e_i poderia ser a próxima aresta escolhida. □

4.7 Caracterização da Busca da Cardinalidade Máxima

Questão sobre a ordem de busca das arestas 7: *Para a tripla de arestas $a \prec b \prec c$, $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, como poderia a aresta b ter sido escolhida antes da aresta c pela busca da cardinalidade máxima? A resposta para esta questão é dada pela Propriedade 4.7.1 e ilustrada na Figura 4.8.*

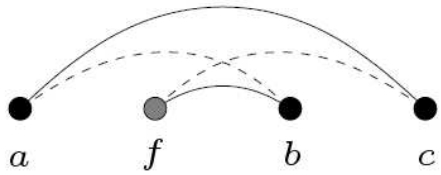


Figura 4.8: Exemplo para uma ordem de busca da cardinalidade máxima.

Propriedade 4.7.1: *Dada uma enumeração σ de E, que contempla todos os vértices de V, se $a_1 \prec \cdots \prec a_k \prec b \prec c$, para $1 \leq i \leq k$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, então, existe um conjunto de arestas $(f_1 \prec \cdots \prec f_i \prec b)$ tal que $f_i \cap b \neq \varnothing$ e $d_i \cap c = \varnothing$, para $1 \leq i \leq k$ e $|e_1 b| \cup |e_i b| \geq$ rótulo(c).*

Teorema 4.7.2: *Para um hipermultigrafo arbitrário $H = (V, E)$, uma enumeração σ de E é uma ordem de busca da cardinalidade máxima das arestas de H se, e somente se, σ tem a Propriedade 4.7.1.*

Prova. Suponha que σ é uma enumeração gerada pela busca da cardinalidade máxima. Seja $a \prec b \prec c$ uma tripla de arestas em σ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$. Para que a aresta b tenha sido escolhida antes da aresta c a cardinalidade do rótulo de b deve ser igual ou maior que a cardinalidade do rótulo de c. Como o rótulo da aresta c pode possuir um valor maior que de b, devem existir arestas tais que, $\sum_{i=1}^{k}$ rótulo$(e_i) \geq$ rótulo(c). Assim, existem arestas $f_1 \prec \cdots \prec f_i$, tais que o rótulo de b é maior que o rótulo de c. Portanto, a aresta b poderia ter sido escolhida antes da aresta c.

Por contradição, suponha que $\sigma = (e_1, \ldots, e_m)$, sendo $m = |E|$, tem a Propriedade 4.7.1, mas não é uma ordem de busca da cardinalidade máxima de H, ou seja, para

algum $2 \leq i \leq m$ $(e_1, \ldots, e_{i-1})$ pode ser obtido pelo Algoritmo 14, mas $(e_1, \ldots, e_i)$ não pode. Assim, e_i é a primeira aresta em σ que não pode ser escolhida pelo MCS. Seja: (i) h a próxima aresta escolhida de rótulo máximo na iteração i. Deve existir uma aresta $g \prec e_i \prec h$, com $g \cap h \neq \varnothing$ e $g \cap e_i = \varnothing$. Aplicando a Propriedade 4.7.1 na tripla de arestas $(g \prec e_i \prec h)$, existe um conjunto de arestas $(f_1 \prec \cdots \prec f_i)$, com $(f_1 \ldots f_i) \cap e_i \neq \varnothing$, tal que, rótulo$(b) \geq$ rótulo(c), portanto uma contradição. □

4.8 Sintetização das Buscas

Nesta seção será aprentada uma síntese das caracterizações apresentadas neste trabalho. Assim como apresentado em (CORNEIL; KRUEGER, 2008), pode-se sumarizar as caracterizações dos seis paradigmas de buscas como ilustrado na Figura 4.9. Por exemplo, uma enumeração σ é uma enumeração LexBFS se, e somente se, σ satisfaz a condição da busca genérica com as condições $f < a$ e $f \cap c = \varnothing$.

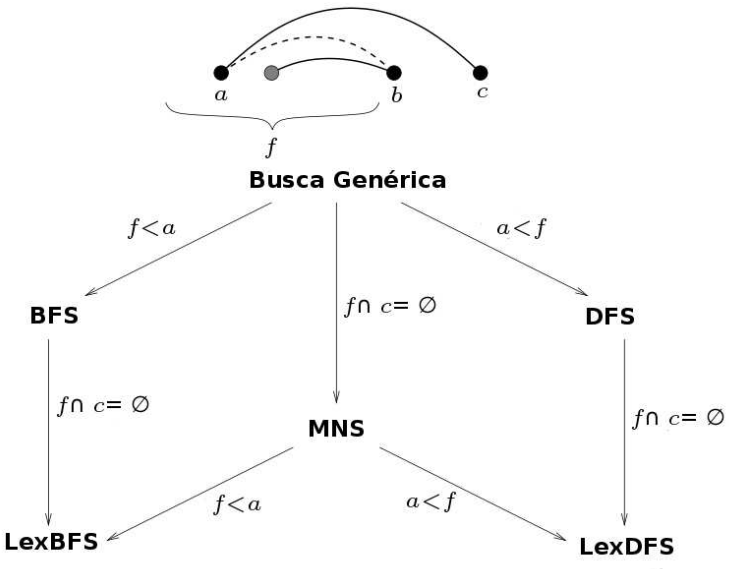


Figura 4.9: Sintetização da caracterização das buscas para quatro arestas

Capítulo 5

Algoritmo Busca do Rótulo Maximal

Neste capítulo será apresentado um algoritmo capaz de expressar as buscas discutidas no Capítulo 2, por aplicar um apropriado tipo de estrutura. Com isso, faz-se necessário a formalização dos conceitos da rotulagem de arestas que o algoritmo, denominado *Busca do Rótulo Maximal - MLS*[1] e proposto por (KRUEGER, 2005), utiliza para sua execução. Neste algoritmo utiliza-se um esquema de rotulagem como parâmetro de entrada, que especifica qual paradigma de busca será obtido pelo MLS. Nesse esquema de rotulagem é descrito como os rótulos das arestas são modificados e atualizados para assim determinar qual aresta deverá ser escolhida, uma vez que a busca será executada em um determinado tipo de grafo. Serão apresentadas também características dos esquemas de rotulagem que correspondem a cada uma das buscas.

5.1 Contextualização do MLS

Todos os algoritmos de busca em grafos nesta dissertação são executados de uma maneira padrão: iterativamente uma aresta é visitada se é adjacente a alguma outra anteriormente visitada. Uma vez que uma aresta foi explorada, qualquer um dos seus vizinhos é candidato a ser o próximo escolhido, através do critério de seleção de cada algoritmo de busca.

[1] Do inglês *Maximal Label Search*.

Segundo (KRUEGER, 2005) o funcionamento do MLS dá-se da seguinte forma: quando uma busca é executada em um tipo específico de grafo, considera-se para cada aresta não visitada do grafo um rótulo que determina a prioridade dentre todas as arestas não visitadas. Assim uma aresta que não é vizinha de nenhuma outra já explorada pode ter baixa prioridade em relação a outra que possui alguns vizinhos visitados e, portanto, arestas que possuem uma grande vizinhança de arestas já visitadas pode ter sua prioridade aumentada (ou decrementada) para um paradigma de busca específico.

Para formalizar a ideia do algoritmo deve-se especificar alguns conceitos. Primeiramente, dizemos que o rótulo de uma aresta e, rótulo(e) deve fazer parte de algum conjunto L. O rótulo de uma aresta pode ser mudado várias vezes durante a execução do algoritmo.

Após uma aresta ter sido escolhida, a sua vizinhança (de arestas não visitada) pode se tornar mais promissora (i.e. com uma maior ou menor prioridade) para a próxima iteração, e consequente escolha do algoritmo. A atualização do rótulo de uma aresta é executada através de uma função, $INC(l,i)$ que atualizará o rótulo das arestas ainda não exploradas, quando um de seus vizinhos foi escolhido na i-ésima iteração. A função $INC(l,i)$ atualizará o rótulo de um novo elemento do conjunto L, que depende do valor de i e que reflete a nova prioridade de escolha de uma aresta.

Como a busca pode começar por qualquer vértice do grafo e como a função INC precisa de um rótulo inicial para uma determinada aresta, então será atribuído o rótulo l_0 para todas as arestas de G, assim todas as arestas terão a mesma prioridade de escolha na primeira iteração do algoritmo.

Utilizando estes conceitos, podemos definir uma estrutura para os vários tipos de buscas:

Definição 5.1.1: (BERRY et al., 2005) Dizemos que (L, $\trianglelefteq$, l_0, INC) é um *esquema de rotulagem* se:

- L é um conjunto (o conjunto dos rótulos);
- $\trianglelefteq$ é uma ordem parcial sobre L, que será utilizada para escolher uma aresta de

rótulo maximal;

- l_0 é um elemento de L, utilizado para inicializar os rótulos;
- *INC* é uma função de $L \times \mathbb{Z}^+$ para L, que será utilizada para atualizar um rótulo.

Com esta definição, a execução do algoritmo é feita da seguinte forma: iterativamente deve-se escolher uma aresta e de rótulo maximal sobre $\trianglelefteq$ de todas as arestas não enumeradas e atualizar os rótulos dos vizinhos de e através da função *INC*. O Algoritmo 15 mostra a busca do rótulo maximal guiado por aresta para um hipermultigrafo.

Algoritmo 15: Busca do Rótulo Maximal para Hipermultigrafo

Entrada: um hipermultigrafo conexo $G = (V, E)$, um vértice inicial $v_0 \in$ V e um esquema de rotulagem (L,$\trianglelefteq$,l_0,INC)
Saída : uma enumeração σ de E

```
1  Atribui o rótulo l_0 para todas as arestas de G
2  escolha uma aresta e adjacente a v_0
3  rótulo(e) ← {m}
4  U ← {v_0}
5  i ← 1
6  para i ← 1 to m faça
7      pegue uma aresta não enumerada e com rótulo maximal sobre a relação ⊴
8      σ(e) ← i
9      se U ≠ U ∪ e então
10         para cada aresta não enumerada f adjacente a e faça
11             rótulo(f) ← INC(rótulo(f),i)
12         U ← U ∪ e
13     senão
14         marque e como aresta de retorno
15     i ← i + 1
```

Dizemos que σ é uma enumeração S-MLS de G se σ é uma enumeração encontrada pelo Algoritmo 15 utilizando um esquema de rotulagem S. Utilizamos a notação rótulo$_i(e)$ para denotar o rótulo atribuído para a aresta e na iteração i, como descrito no Algoritmo 15 .

Lema 5.1.2: *(KRUEGER, 2005) Seja G um grafo e S um esquema de rotulagem. Então em qualquer execução de S-MLS em G resultando em uma enumeração* $\sigma = (e_1 \prec \cdots \prec e_n)$*, a seguinte asserção é válida para todos os inteiros* $i, j \geq 1$ *e arestas f,g* $\in E$*:*

i Se $i < j$, então rótulo$_i(e_i) \not\lhd$ rótulo$_i(e_j)$;

ii Se $e_i \cap f = \varnothing$, então rótulo$_{i+1}(f) =$ rótulo$_i(f)$;

iii Se rótulo$_{i+1}(f) \neq$ rótulo$_i(f)$, então $e_i \cap f \neq \varnothing$ e rótulo$_{i+1}(f) =$ INC(rótulo$_i(f), i)$;

iv Se $e_i \prec f \prec g$ em σ e, para todo $j < i$, $e_i \cap f \neq \varnothing$ precisamente quando $e_j \cap g \neq \varnothing$, então rótulo$_i(f) =$ rótulo$_i(g)$.

Como os rótulos das arestas são ou l_0 ou o resultado da aplicação da função *INC*, então podemos definir os rótulos que podem ser atribuídos a uma aresta a cada iteração da busca (KRUEGER, 2005):

Definição 5.1.3: Seja $S = (L, \trianglelefteq, l_0, INC)$ um esquema de rotulagem. Para $i \geq 1$, seja $L_i \subseteq L$ definido recursivamente por:

- $L_1 = \{l_0\}$, e
- $L_{i+1} = \mathrm{L}_i \cup \{l \in L$: $l =$ INC(l',i) para algum $l' \in L_i\}$ para $i \geq 1$.

5.2 Definição de Esquemas de Rotulagem

Nota-se que o algoritmo de busca de rótulo maximal pode descrever vários tipos de busca, portanto, nesta seção será apresentada esquemas de rotulagem que corresponde a cada uma das buscas do Capítulo 2, como apresentado em (KRUEGER, 2005).

`Busca Genérica:` (Esquema de Rotulagem S_{BG}): $L = \{0,1\}$, $\trianglelefteq$ é a ordem natural $\leq$ sobre os inteiros, $l_0 = 0$, e INC(l,i) $= 1$. O rótulo corresponde a um booleano que indica se uma aresta é adjacente a outra já visitada.

`Busca em Largura:` (Esquema Rotulagem S_{BFS}): $L = \{1,2,\dots\} \cup \{\infty\}$, $\trianglelefteq$ é a ordem natural $\geq$ sobre os inteiros, $l_0 = \infty$, e INC(l,i) é igual a i se $l = \infty$ e iqual l caso contrário. O rótulo ∞ de uma aresta e significa que nenhum de seus vizinhos foi visitado.

`Busca em Profundidade:` (Esquema de Rotulagem S_{DFS}): $L = \{0,1,\dots\}$, $\trianglelefteq$ é a ordem natural $\leq$ sobre os inteiros, $l_0 = 0$, e INC(l,i) = i. O rótulo corresponde a aresta mais recentemente visitada ou zero se nenhum vizinho foi visitado.

`Busca em Largura Lexicográfica:` (Esquema de Rotulagem S_{LexBFS}): L é o conjunto de *strings* sobre o alfabeto $\{1,2,\dots\}$, $\trianglelefteq$ é a ordem lexicográfica em que o dígito i é considerado maior que o dígito j, quando $i < j$, l_0 é a *string* vazia ε, e INC(l,i) é o dígito i colocado como sufixo na *strings* l.

`Busca em Profundidade Lexicográfica:` (Esquema de Rotulagem S_{LexDFS}): L é o conjunto de *strings* sobre o alfabeto $\{1,2,\dots\}$, $\trianglelefteq$ é a ordem lexicográfica natural, l_0 é a *string* vazia ε, e INC(l,i) é o dígito i colocado como prefixo na *string* l.

`Busca da Vizinhança Maximal:` (Esquema de Rotulagem S_{MNS}): L é o conjunto dos subconjuntos de $\{1,2,\dots\}$, $\trianglelefteq$ é o conjunto inclusão $\subseteq$ (não é ordem total), l_0 é o conjunto vazio, e INC(l,i) = $l \cup \{i\}$. O rótulo corresponde ao conjunto dos vizinhos já visitada de uma aresta.

Definição 5.2.1: (KRUEGER, 2005) Dizemos que um esquema de rotulagem (L, $\trianglelefteq$, l_0, *INC*) é um esquema de rotulagem em um grafo se a seguinte propriedade é válida:

- **Rótulo Mínimo Inicial** $\mathbb{L}1$**:** $l_0 \lhd$ INC(l,i), para todo $i \geq 1$ e $l \in \mathrm{L}_i$,

Mostraremos agora que qualquer esquema que forma um esquema de rotulagem de busca em um grafo corresponde a algum algoritmo de busca.

Lema 5.2.2: *(KRUEGER, 2005) Seja G um hipermultigrafo, seja S um esquema de rotulagem de busca em um grafo, e seja σ uma enumeração de E. Se σ pode ser computado pelo $S - MLS(G)$, então σ é uma enumeração válida.*

Prova. Suponha que σ é uma enumeração encontrado pelo S-MLS utilizando algum esquema de rotulagem S. Suponha por contradição que σ não é uma enumeração válida. Então pelo Teorema 4.1.2 a Propriedade 4.1.1 é violada. Então, existe uma tripla de arestas $a \prec b \prec c$ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ tal que, para toda aresta $f \prec b$, $f \cap b = \varnothing$.

Então, rótulo$_b(b) = l_0$. Mas, como $a \cap c \neq \varnothing$, temos que rótulo$_b(c) = INC($rótulo$_i(c), i)$, para algum $i < \sigma^{-1}(b)$. Pela Propriedade $\mathbb{L}1$ da Definição 5.2.1, rótulo$_b(c) \trianglerighteq l_0 =$ rótulo$_b(b)$, contradizendo a escolha de b pelo algoritmo. □

Lema 5.2.3: *(KRUEGER, 2005) Seja G um hipermultigrafo e seja σ uma enumeração de E. Então σ pode ser computada pelo $S_{GS} - MLS(G)$.*

PROVA. Suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração de G. Suponha que $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo algoritmo S_{GS}, mas que $(e_1 \prec \cdots \prec e_i)$ não pode, para algum $1 < i \leq n$. Então o rótulo$_i(e_i)$ não é maximal na iteração i. Seja $g > e_i$ uma aresta não enumerada de rótulo maximal na iteração i com, rótulo$_i(g) \trianglerighteq$ rótulo$_i(e_i)$. Então rótulo$_i(g) = 1$ e rótulo$_i(e_i) = 0$. Assim, e_i não tem vizinho antes dele em σ. Desde que o rótulo da aresta g foi atualizada, então, existe um $j < i$ com, $e_j \cap g \neq \varnothing$ e $e_j \cap e_i = \varnothing$. Pelo Teorema 4.1.2 e aplicando a Propriedade 4.1.1 na tripla de arestas $e_j \prec e_i \prec g$, existe um $k < i$ com $e_k \cap e_i \neq \varnothing$, uma contradição. □

O seguinte teorema segue diretamente.

Teorema 5.2.4: *(KRUEGER, 2005) Seja G um grafo e σ uma enumeração de E. Então σ é uma ordem de busca de G se, e somente se, existe um esquema de rotulagem S tal que, σ pode ser computado pelo $S - MLS(G)$.*

PROVA. Pelos Lemas 5.2.2 e 5.2.3 e pelo fato que S_{GS} é um esquema de rotulagem. □

5.3 Caracterização das Buscas em Hipermultigrafos Utilizando o Algoritmo MLS

Adicionando propriedades, e quando satisfeita por um esquema de rotulagem S, pode-se assegurar que $S-MLS$ irá computar um tipo específico de busca. A seguinte propriedade será utilizada para caracterizar tais esquemas.

Propriedade 5.3.1: *(KRUEGER, 2005) Seja (L, $\trianglelefteq$, l_0, INC) um esquema de rotulagem. Definimos as seguintes propriedades para S:*

- ***Não Decremento Monotônico*** $\mathbb{L}2$***:*** $l \trianglelefteq INC(l, i)$*, para todo* $i \geq 1$ *e* $l \in L_i$*,*
- ***Incremento Monotônico*** $\mathbb{L}3$***:*** *l* $\trianglelefteq$ *INC(l,i), para todo* $i \geq 1$ *e* $l \in L_i$*,*
- ***Estabilidade*** $\mathbb{L}4$***:*** *se* $l \trianglelefteq l'$ *então INC(l,i)* $\trianglelefteq$ *INC(l',i), para todo* $i \geq 1$ *e* $l, l' \in L_i$*.*

Lema 5.3.2: *(KRUEGER, 2005) Seja G um grafo e S um esquema de rotulagem. Então em qualquer execução do* $S - MLS$ *em G resultando em uma enumeração* $\sigma = (e_1, \ldots, e_n)$ *a seguinte a asserção é válida para todos os inteiros* $i, j \geq 1$ *e arestas* $f, g \in E'$*.*

i Se S é não decremento monotônico, então, para $i < j$, rótulo$_i(f) \trianglelefteq$ rótulo$_j(f)$.

ii Se S é não decremento monotônico e estável, e se rótulo$_i(f) \trianglelefteq$ rótulo$_j(g)$ para alguma $e_i \prec f, g$ então, existe $j < i$ tal que, $e_j \cap g \neq \varnothing$ e $v_j \cap f = \varnothing$

Proposição 5.3.3: *Seja H um hipermultigrafo e* σ *uma enumeração de E, que contempla todos os vértices em V, então* σ *é uma enumeração DFS de H se, e somente se, existe um esquema de rotulagem S tal que,* σ *pode ser computada pelo S-MLS, onde S satisfaz a propriedade:*

- **Primazia em Profundidade** $\mathbb{L}5$**:** $l \trianglelefteq$ INC(l',i), para todo $i \geq 1$ e l,$l' \in$ L$_i$,

Prova. Suponha que σ seja uma enumeração encontrada pelo S-MLS por algum esquema de rotulagem S satisfazendo a propriedade $\mathbb{L}5$. Se σ não é uma enumeração DFS então, pelo Teorema 4.3.2, a Propriedade 4.3.1 é violada. Então existe arestas $a \prec b \prec c$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ tal que, para toda aresta $a \prec f \prec b$, $f \cap b = \varnothing$. Então rótulo$_a(b) =$ rótulo$_b(b)$ por indução no Lema 5.1.2(ii). Por $\mathbb{L}5$, rótulo$_a(b) \lhd INC($rótulo$_a(c), \sigma^{-1}(a)) =$ rótulo$_{a+1}(c) \trianglelefteq$ rótulo$_b(c)$ pelo Lema 5.3.2(i) e pelo fato que $\mathbb{L}5$ se submete a $\mathbb{L}2$. Assim, rótulo$_b(b) \lhd$ rótulo$_b(c)$, contradizendo a escolha de b pelo MLS.

Por contradição, suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração DFS. Seja S um esquema de rotulagem $S_{DFS} = (\{0,1\ldots\}, \leq, 0, INC(l,i) = i)$, descrito na Seção 5.2. Note que S_{DFS} satisfaz $\mathbb{L}5$. Suponha que $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo algoritmo S-MLS, mas que $(e_1 \prec \cdots \prec e_i)$ não pode para algum $1 < i \leq n$. Então o rótulo$_i(e_i)$ não é maximal na iteração i. Seja g tal que, $e_i \prec g$ ser uma aresta não enumerada de rótulo maximal na iteração i, com rótulo$_i(e_i) \prec$ rótulo$_i(g)$. Assim rótulo$_i(g) = INC($rótulo$_j(g), j) = j$ para algum $j < i$. Então $e_j \cap g \neq \varnothing$ e $e_j \cap e_i = \varnothing$. Pelo Teorema 4.3.2, a Proriedade 4.3.1 é válida na tripla $e_j \prec e_i \prec g$, então existe algum k, $j \prec k \prec i$, com $e_k \cap e_i \neq \varnothing$; escolha o maior k. Então rótulo$_i(e_i) = k > j$, contradizendo a suposição acima. □

Proposição 5.3.4: *Seja H um hipermultigrafo e σ uma enumeração de E, que contempla todos os vértices de V, então σ é uma enumeração BFS de H se, e somente se, existe um esquema de rotulagem S, tal que σ pode ser computada pelo S-MLS, onde S satisfaz as propriedades:*

- **Rótulo Mínimo Inicial** $\mathbb{L}1$**:** $l_0 \lhd INC(l,i)$, para todo $i \geq 1$ e $l \in L_i$;
- **Não Decremento Monotônico** $\mathbb{L}2$**:** $l \trianglelefteq INC(l,i)$, para todo $i \geq 1$ e $l \in L_i$;
- **Primazia em Largura** $\mathbb{L}6$**:** se $l \trianglelefteq l'$ então $INC(l,i) \trianglelefteq l'$, para todo $i \geq 1$ e l,$l' \in$ L$_i$.

Prova. Suponha que σ seja uma enumeração encontrada pelo S-MLS por algum esquema de rotulagem S satisfazendo as Propriedades $\mathbb{L}1$, $\mathbb{L}2$ e $\mathbb{L}6$. Se σ não é uma enumeração BFS então, pelo Teorema 4.2.2, a Propriedade 4.2.1 é violada. Então existe arestas $a \prec b \prec c$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ tal que, para toda aresta $f \prec a$, $f \cap b = \varnothing$. Então rótulo$_{a+1}(b) = l_0$ por repetidas aplicações do Lema 5.1.2(ii) e por $\mathbb{L}1$, rótulo$_{a+1}(c) = INC($rótulo$_a(c), \sigma^{-1}(a)) \rhd l_0$. Por indução em $\mathbb{L}6$, rótulo$_b(b) \lhd$ rótulo$_{a+1}(c)$. Por $\mathbb{L}2$, rótulo$_{a+1}(c) \trianglelefteq$ rótulo$_b(c)$. Assim rótulo$_b(b) \lhd$ rótulo$_b(c)$, contradizendo a escolha de b pelo S-MLS.

Por contradição, suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração BFS. Seja S um esquema de rotulagem $S_{BFS} = (\{1,2\ldots\} \cup \{\infty\}, \geq, \infty, INC)$, onde $INC(l,i) = i$ se $l = \infty$ e $INC(l,i) = l$ descrito na Seção 5.2. Note que S_{BFS} satisfaz $\mathbb{L}1$, $\mathbb{L}2$ e $\mathbb{L}6$. Suponha que $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo algoritmo S-MLS, mas que $(e_1 \prec \cdots \prec e_i)$ não pode para algum $1 < i \leq n$. Então, o rótulo$_i(e_i)$ não é maximal na iteração i. Seja $g \succ e_i$ uma aresta não enumerada de rótulo máximal na iteração i com rótulo$_i(e_i) \lhd$ rótulo$_i(g)$. Então rótulo$_i(g) = INC($rótulo$_j(g), j)$ para algum $j < i$ com $e_j \cap g \neq \varnothing$; escolha o menor j. Então rótulo$_i(g) = j$. Por $\mathbb{L}2$ e pelo Lema 5.3.2(i), rótulo$_{j+1}(e_i) \trianglelefteq$ rótulo$_i(e_i)$. Se $e_j \cap e_i \neq \varnothing$, então rótulo$_{j+1}(e_i) = INC($rótulo$_j(e_i, j) \trianglerighteq j =$ rótulo$_i(g)$, que contradiz a escolha de g. Portanto, $e_j \cap e_i = \varnothing$. Pelo Teorema 4.2.2, a propriedade 4.2.1 é válida na tripla $e_j \prec e_i \prec g$, então existe um $k < j$ tal que, $e_k \cap e_i \neq \varnothing$. Assim, rótulo$_i(e_i) \trianglerighteq$ rótulo$_{k+1}(e_i) = INC($rótulo$_k(e_i), k \trianglerighteq k \trianglerighteq j =$ rótulo$_i(g)$, que contradiz a escolha de g. □

Proposição 5.3.5: *Seja H um hipermultigrafo e σ uma enumeração de E, que contempla todos os vértices de V, então σ é uma enumeração LexBFS de H se, e somente se, existe um esquema de rotulagem S tal que, σ pode ser computada pelo S-MLS, onde S satisfaz as propriedades:*

- **Incremento Monotônico** $\mathbb{L}3$**:** $l \trianglelefteq$ INC(l,i), para todo $i \geq 1$ e $l \in \mathrm{L}_i$,
- **Primazia em Largura** $\mathbb{L}6$**:** se $l \trianglelefteq l'$ então $INC(l,i) \trianglelefteq l'$, para todo $i \geq 1$ e l,$l' \in \mathrm{L}_i$.

Prova. Suponha que σ seja uma enumeração encontrada pelo S-MLS por algum esquema de rotulagem S satisfazendo as propriedades $\mathbb{L}3$ e $\mathbb{L}6$. Se σ não é uma enumeração LexBFS então, pelo Teorema 4.4.2, a Propriedade 4.4.1 é violada. Isso quer dizer que existe uma tripla de arestas $a \prec b \prec c$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ tal que, para toda aresta $f \prec a$, $f \cap b \neq \varnothing$ implica que $f \cap c \neq \varnothing$; escolha uma tripla com a aresta a mais a esquerda em σ. Então para toda aresta $f \prec a$, temos que $f \cap b \neq \varnothing$ se, e somente se, $f \cap c \neq \varnothing$, então rótulo$_a(b) =$ rótulo$_a(c)$ pelo Lema 5.1.2(iv). Então rótulo$_a(c) \lhd$ rótulo$_{a+1}(c)$ pelo

$\mathbb{L}3$ e rótulo$_{a+1}(b)$ = rótulo$_a(b)$. Por indução e por $\mathbb{L}6$, rótulo$_b(b) \lhd$ rótulo$_{a+1}(c)$. Já que $\mathbb{L}3$ se submete a $\mathbb{L}2$, rótulo$_{a+1}(c) \trianglelefteq$ rótulo$_b(c)$ pelo Lema 5.3.2(i). Assim rótulo$_b(b) \lhd$ rótulo$_b(c)$, contradizendo a escolha do algoritmo.

Por contradição, suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração LexBFS. Seja S um esquema de rotulagem S_{LexBFS} = (strings sobre 1,2..., $\leq$ invertido, ϵ,INC), onde INC(l,i) = $l \circ i$, descrito na Seção 5.2. Note que S_{LexBFS} satisfaz as propriedades $\mathbb{L}3$ e $\mathbb{L}6$. Suponha que, $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo algoritmo MLS, mas que $(e_1 \prec \cdots \prec e_i)$ não pode para algum $1 < i \leq n$. Então o rótulo$_i(e_i)$ não é maximal. Seja g tal que, $e_i \prec g$ ser uma aresta não enumerada de rótulo maximal na iteração i com rótulo$_i(e_i) \lhd$ rótulo$_i(g)$. Já que $\mathbb{L}3$ e $\mathbb{L}6$ implicam que $\mathbb{L}2$ e $\mathbb{L}4$ sejam válidos, pelo Lema 5.3.2(ii) existe um $j < i$ com $e_j \cap e_i = \varnothing$ e $e_j \cap g \neq \varnothing$; pegue o menor j. Pelo Teorema 4.4.2, a Propriedade 4.4.1 é válida na tripla $e_j \prec e_i \prec g$, então, existe um $k < j$ tal que, $e_k \cap e_i \neq \varnothing$ e $e_k \cap g = \varnothing$; pegue o menor k. Então, para toda aresta $h \prec e_k$, $h \cap e_i \neq \varnothing$ e $h \cap g \neq \varnothing$, então rótulo$_k(e_i)$ = rótulo$_k(g)$ pelo Lema 5.1.2(iv), e rótulo$_{k+1}(e_i) = INC($rótulo$_k(e_i), k) \trianglerighteq$ rótulo$_k(e_i)$ = rótulo$_k(g)$ = rótulo$_{k+1}(g)$ por $\mathbb{L}3$. Por $\mathbb{L}6$ e indução, rótulo$_i(g) \lhd$ rótulo$_{k+1}(e_i) \trianglelefteq$ rótulo$_i(e_i)$, que contradiz nossa escolha de g. □

Proposição 5.3.6: *Seja H um hipermultigrafo e σ uma enumeração de E, que contempla todos os vértices de V. Então σ é uma enumeração LexDFS de H se, e somente se, existe um esquema de rotulagem S tal que, σ pode ser computada pelo S-MLS, onde S satisfaz as propriedades:*

- **Primazia em Profundidade** $\mathbb{L}5$**:** $l \trianglelefteq$ INC(l',i), para todo $i \geq 1$ e l,$l' \in$ L$_i$,
- **Estabilidade** $\mathbb{L}4$**:** se $l \trianglelefteq l'$ então INC(l,i) $\trianglelefteq$ INC(l',i).

Prova. Suponha que σ seja uma enumeração encontrada pelo S-MLS por algum esquema de rotulagem S satisfazendo as Propriedades $\mathbb{L}4$ e $\mathbb{L}5$. Se σ não é uma enumeração LexDFS então, pelo Teorema 4.5.2 a Propriedade 4.5.1 é violada. Isso quer dizer que existe uma tripla de arestas $a \prec b \prec c$, com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$ tal que, para toda aresta f,

$a \prec f \prec b$, $f \cap b \neq \varnothing$ implica que $f \cap c \neq \varnothing$; escolha a tripla de arestas com a mais a direita. Por $\mathbb{L}5$, rótulo$_{a+1}(c) \trianglerighteq$ rótulo$_a(b) =$ rótulo$_{a+1}(b)$. Por indução em $\mathbb{L}4$, rótulo$_b(b)$ $\lhd$ rótulo$_b(c)$, contradizendo a escolha de b pelo algoritmo.

Por contradição, suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração LexDFS. Seja S um esquema de rotulagem S_{LexDFS} = (strings sobre {1,2...}, $\leq_L$, ϵ,INC), onde INC$(l, i) = l \circ i$, descrito na Seção 5.2. Note que S_{LexDFS} satisfaz as Propriedades $\mathbb{L}5$ e $\mathbb{L}4$. Suponha que $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo algoritmo S-MLS, mas que $(e_1 \prec \cdots \prec e_i)$ não pode para algum $1 < i \leq n$. Então o rótulo$_i(e_i)$ não é maximal. Seja g tal que, $e_i \prec g$ ser uma aresta não enumerada de rótulo maximal na iteração i com rótulo$_i(e_i)$ $\lhd$ rótulo$_i(g)$. Já que $\mathbb{L}5$ submete a $\mathbb{L}2$, pelo Lema 5.3.2(ii), existe algum $j < i$, tal que, $e_j \cap g \neq \varnothing$ e $e_j \cap e_i = \varnothing$ e para todo k, $j < k < i$, $e_k \cap g \neq \varnothing$ se, e apenas se, $e_k \cap e_i \neq \varnothing$. Pelo Teorema 4.5.2, a Propriedade 4.5.1 é válido na tripla $e_j \prec e_i \prec g$, então existe um vértice f, $e_j \prec f \prec e_i$ com $f \cap e_i \neq \varnothing$ mas $f \cap g = \varnothing$, uma contradição. □

Proposição 5.3.7: *Seja H um hipermultigrafo e σ uma enumeração de E, que contempla todos os vértices de V. Então σ é uma enumeração MNS de H se, e somente se, existe um esquema de rotulagem S tal que, σ pode ser computada pelo S-MLS, onde S satisfaz as propriedades:*

- **Incremento Monotônico** $\mathbb{L}3$**:** $l \trianglelefteq$ INC(l',i), para todo $i \geq 1$ e $l \in \mathrm{L}_i$,
- **Estabilidade** $\mathbb{L}4$**:** se $l \trianglelefteq l'$ então INC(l,i) $\trianglelefteq$ INC(l',i), para todo $i \geq 1$ e $l, l' \in L_i$.

Prova. Suponha que σ seja uma enumeração encontrada pelo S-MLS por algum esquema de rotulagem S satisfazendo as Propriedades $\mathbb{L}3$ e $\mathbb{L}4$. Se σ não é uma enumeração MNS então, pelo Teorema 4.6.2 a Propriedade 4.6.1 é violada. Então existe arestas $a \prec b \prec c$ com $a \cap c \neq \varnothing$ e $a \cap b = \varnothing$, e para toda aresta $f \prec b$, se $f \cap b \neq \varnothing$ então, $f \cap c \neq \varnothing$. Seja $a \prec b \prec c$ tais arestas com a mais esquerda em σ. Então rótulo$_a(b) =$ rótulo$_a(c)$ pelo Lema 5.1.2(iv). Por ($\mathbb{L}3$), rótulo$_a(c) =$ rótulo$_a(b) \lhd$ rótulo$_{a+1}(b) \trianglelefteq$ rótulo$_{a+1}(c)$ = INC(rótulo$_a(c)$,$\sigma^{-1}(a)$) $\cup$ INC(rótulo$_a(c)$,$\sigma^{-1}(a)$). Por ($\mathbb{L}4$) rótulo$_b(b) \trianglelefteq$ rótulo$_b(c)$, contradizendo a escolha de b pelo MLS.

Por contradição, suponha que $\sigma = (e_1 \prec \cdots \prec e_n)$ é uma enumeração MNS. Seja S um esquema de rotulagem S_{MNS} = (subconjunto de {1,2...}, $\subseteq$, ϕ, INC(l,i) = $l \cup \{i\}$ descrito na Seção 5.2. Note que S_{MLS} satisfaz ($\mathbb{L}$3) e ($\mathbb{L}$4). Suponha $(e_1 \prec \cdots \prec e_{i-1})$ pode ser encontrado pelo S-MLS, mas $(e_1 \prec \cdots \prec e_i)$ não, para algum $1 \leq i \leq n$. Então rótulo$_i(e_i)$ não é maximal na iteração i. Seja g tal que, $e_i \prec g$ uma aresta não enumerada de rótulo maximal na iteração i com rótulo$_i(e_i) \subset$ rótulo$_i(g)$. Então existe um $j < i$ com $e_j \cap g \neq \varnothing$ e $e_j \cap e_i = \varnothing$. Pelo Teorema 4.6.2 e aplicando a Propriedade 4.6.1 na tripla de arestas $e_j \prec e_i \prec g$, existe um $k < i$, tal que, $e_k \cap e_i \neq \varnothing$ e $e_k \cap g = \varnothing$. Assim, rótulo$_i(e_i)$ $\nsubseteq$ rótulo$_i(g)$, uma contradição. □

5.4 Validade da Generalização da Busca

O Algoritmo 15, que mostra a busca do rótulo maximal, é capaz de expressar enumerações de arestas para cada um dos tipos de busca apresentada nesta dissertação, através de um esquema de rotulagem. Neste trabalho, as buscas apresentadas no Capítulo 3 são uma generalização das buscas apresentadas por (CORNEIL; KRUEGER, 2008). Assim, queremos mostrar que, para os casos que já eram comtemplados pelo Algoritmo 5, o comportamento do nosso algoritmo (Algoritmo 15) é equivalente ao comportamento do algoritmo original.

Teorema 5.4.1: *Seja $G = (V, E)$ um grafo e seja S um esquema de rotulagem. Seja σ_v a enumeração obtida pelo Algoritmo 5 com entrada G e S. Seja σ_e a enumeração obtida pelo Algoritmo 15 com entrada G e S. Seja σ_p a enumeração obtida pelo Algoritmo 8 com entrada σ_e. Como todas as arestas de G possuem 2 e só 2 elementos, é imediato que as partes da partição que σ_p enumera possuem 1 e só 1 elemento cada. Portanto, tomemos σ' a enumeração do conjunto de vértices obtida trivialmente a partir de σ_p. Então, $\sigma_v = \sigma'$.*

Prova. Suponhamos, por contradição, que $\sigma_v \neq \sigma'$. Então, existe algum $j \geq 1$ para o qual $v_j \neq u_j$ mas $v_k = u_k$ para todo $k < j$. Assim, consideremos a iteração do Algoritmo 5 quando i vale j. O vértice a ser escolhido na linha 3 é exatamente o vértice que será

enumerado com $i = j$. Como essa enumeração não poderá ser alterada no restante da execução do algoritmo, esse vértice é precisamente o v_j.

Agora, consideremos a iteração do Algoritmo 15 quando i vale j. Como na entrada do Algoritmo 8, não são consideradas a múltiplas arestas, consideremos, sem perda de generalidade, que a aresta e escolhida na linha 7 do Algoritmo 15 é uma aresta que leva a um vértice u que ainda não está em U. Como imediatamente ocorrerá a enumeração da aresta e, sendo que essa enumeração não será alterada, podemos concluir que $\sigma'(u) = j$. Portanto, $u = u_j$.

Da execução do Algoritmo 5, temos que v_j é adjacente a $v_{j-1} = u_{j-1}$. Tomemos, então, a aresta $f = \{u_{j-1}, v_j\}$. Consideremos agora a iteração do Algoritmo 15 quando $i = j - 1$. É evidente que a aresta f ainda não foi enumerada; do contrário, teríamos que v_j seria igual a algum v_k, $k < j$. Portanto, no laço da linha 10, a aresta f será escolhida e rotulada. O mesmo acontecerá com a aresta $e = \{u_{j-1}, u_j\}$, que também ainda não foi enumerada. Porém, como a aresta e é a que vai ser escolhida na iteração j, como já vimos, temos que rótulo $f \trianglelefteq$ rótulo e. Construindo o mesmo raciocínio para a execução do Algoritmo 5, temos que rótulo $u_j \trianglelefteq$ rótulo v_j. Como os rótulos são construídos exatamente com o mesmo esquema de rotulagem, temos tanto que rótulo $f =$ rótulo e quanto que rótulo $u_j =$ rótulo v_j, caindo no caso em que os algoritmos escolhem arbitrariamente a aresta (ou o vértice) maximal entre os empatados. Todavia, assumimos o mesmo critério de desempate entre os algoritmos, já que as comparações ocorrem exatamente sobre os mesmos rótulos. Dessa forma, ou a aresta f deveria ter sido escolhida pelo Algoritmo 15 ou o vértice u_j deveria ter sido escolhido pelo Algoritmo 15. Chegamos, então, a uma contradição. □

Capítulo 6

Conclusões

Neste trabalho foi apresentado alguns algoritmos de busca para o caso de um hipermultigrafo e como, através dessas buscas pode-se conseguir diferentes tipos de enumerações. Tais enumerações, respeitam algumas proriedades que caracterizam cada uma das buscas apresentada nesta dissertação.

Demostramos também sete diferentes algoritmos de busca da perspectiva: se em uma busca, o algoritmo visita a aresta a antes da aresta b e b antes da aresta c, sendo que $a \cap b = \varnothing$ e $a \cap c \neq \varnothing$, como poderia um algoritmo de busca ter visitado b antes de c? Com essa questão, encontramos propriedades satisfeitas pelas buscas que caracterizou-se buscas clássicas em hipermultigrafo encontrados na literatura, tais como, buscas em largura e em profundidade, e outras buscas apresentadas mais recentemente como, em profundidade lexicográfica e da vizinhança maximal. Tais caracterizações, assim como apresentada em (CORNEIL; KRUEGER, 2008), pode ser uma ferramenta poderosa e útil para o estudo e melhor compreensão dos algoritmos de grafos, além de nos dar um melhor entendimento de como uma determinada busca revela a estrutura em um dado grafo.

6.1 Trabalhos Futuros

Como trabalhos futuros podemos citar:

- Como utilizar as enumerações para recuperar uma busca ou executar uma nova busca com pouco esforço. Para tal, podemos pensar na junção de partes de buscas sem necessariamente executar a busca em todo o grafo;

- Caracterizações de outros tipos de busca e/ou outras estruturas.

Referências

BERRY, A. et al. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, v. 39, n. 4, p. 287–298, 2004.

BERRY, A.; KRUEGER, R.; SIMONET, G. Ultimate generalizations of LexBFS and LEX M. In: *WG*. [S.l.: s.n.], 2005. p. 199–213.

BRANDSTÄDT, A.; DRAGAN, F. F.; NICOLAI, F. LexBFS-orderings and powers of chordal graphs. *Discrete Math.*, v. 171, n. 1-3, p. 27–42, 1997. ISSN 0012-365X. Disponível em: <http://dx.doi.org/10.1016/S0012-365X(96)00070-2>.

CHANG, J.-M.; HO, C.-W.; KO, M.-T. LexBFS-ordering in asteroidal triple-free graphs. In: *ISAAC '99: Proceedings of the 10th International Symposium on Algorithms and Computation*. London, UK: Springer-Verlag, 1999. p. 163–172. ISBN 3-540-66916-7.

CORNEIL, D. G. Lexicographic breadth first search—a survey. In: *Graph-theoretic concepts in computer science*. Berlin: Springer, 2004, (Lecture Notes in Comput. Sci., v. 3353). p. 1–19.

CORNEIL, D. G. A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Appl. Math.*, v. 138, n. 3, p. 371–379, 2004. ISSN 0166-218X. Disponível em: <http://dx.doi.org/10.1016/j.dam.2003.07.001>.

CORNEIL, D. G.; DRAGAN, F. F.; KÖHLER, E. On the power of BFS to determine a graph's diameter. *Networks*, v. 42, n. 4, p. 209–222, 2003. ISSN 0028-3045. Disponível em: <http://dx.doi.org/10.1002/net.10098>.

CORNEIL, D. G. et al. Simple linear time recognition of unit interval graphs. *Inform. Process. Lett.*, v. 55, n. 2, p. 99–104, 1995. ISSN 0020-0190. Disponível em: <http://dx.doi.org/10.1016/0020-0190(95)00046-F>.

CORNEIL, D. G.; KRUEGER, R. M. Simple vertex ordering characterizations for graph search: (expanded abstract). In: *7th International Colloquium on Graph Theory*. [S.l.]: Electronic Notes in Discrete Mathematics, 2005. v. 22, p. 445–449.

CORNEIL, D. G.; KRUEGER, R. M. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, SIAM, v. 22, n. 4, p. 1259–1276, 2008.

DAHLHAUS, E.; GUSTEDT, J.; MCCONNELL, R. M. Partially complemented representations of digraphs. *Discrete Math. Theor. Comput. Sci.*, v. 5, n. 1, p. 147–168 (electronic), 2002. ISSN 1365-8050.

EDMONDS, J.; KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. In: *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*. New York: Gordon and Breach, 1970. p. 93–96.

EVEN, S. *Graph algorithms*. Woodland Hills, Calif.: Computer Science Press Inc., 1979. ix+249 p. Computer Software Engineering Series. ISBN 0-914894-21-8.

GODSIL, C.; ROYLE, G. *Algebraic graph theory*. New York: Springer-Verlag, 2001. xx+439 p. (Graduate Texts in Mathematics, v. 207). ISBN 0-387-95241-1; 0-387-95220-9.

HABIB, M. et al. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoret. Comput. Sci.*, v. 234, n. 1-2, p. 59–84, 2000. ISSN 0304-3975. Disponível em: <http://dx.doi.org/10.1016/S0304-3975(97)00241-7>.

HELL, P.; HUANG, J. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM J. Discrete Math.*, v. 18, n. 3, p. 554–570 (electronic), 2004/05. ISSN 0895-4801. Disponível em: <http://dx.doi.org/10.1137/S0895480103430259>.

KRUEGER, R. M. *Graph Searching*. Tese (Doutorado) — Department of Computer Science, University of Toronto, 2005.

MOORE, E. F. The shortest path through a maze. In: *Proc. Internat. Sympos. Switching Theory 1957, Part II*. Cambridge, Mass.: Harvard Univ. Press, 1959. p. 285–292.

ROSE, D. J.; TARJAN, R. E. Algorithmic aspects of vertex elimination. In: *STOC 75: Proceedings of seventh annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1975. p. 245–254.

ROSE, D. J.; TARJAN, R. E.; LUEKER, G. S. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, v. 5, n. 2, p. 266–283, 1976. ISSN 0097-5397.

TARJAN, R. E. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, v. 1, n. 2, p. 146–160, 1972.

TARJAN, R. E.; YANNAKAKIS, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, v. 13, n. 3, p. 566–579, 1984. ISSN 0097-5397. Disponível em: <http://dx.doi.org/10.1137/0213035>.

YANNAKAKIS, M. Computing the minimum fill-in is NP-complete. *SIAM J. Algebraic Discrete Methods*, v. 2, n. 1, p. 77–79, 1981. ISSN 0196-5212. Disponível em: <http://dx.doi.org/10.1137/0602010>.