

## Aplicações de Grafos E/Ou

**Uéverton dos Santos Souza**

UFF - Programa de Pós-Graduação em Computação - Niterói, RJ

UFRRJ - Instituto Três Rios - Três Rios, RJ

E-mail:usouza@ic.uff.br

**Fábio Protti**

UFF - Instituto de Computação - Niterói, RJ

E-mail:fabio@ic.uff.br

**Maise Dantas da Silva**

UFF - Instituto de Ciência e Tecnologia - Rio das Ostras, RJ

E-mail:maise@vm.uff.br

### RESUMO

Grafos *E/Ou* são digrafos acíclicos onde os vértices possuem rotulações *E* ou *Ou*. Esses rótulos determinam se a dependência de um vértice em relação a seus vizinhos de saída será conjuntiva ou disjuntiva. Através destes grafos é possível representar as possíveis formas de se decompor um problema, gerar padrões de cortes em placas, representar redes de atividades, modelar fórmulas booleanas, além de definir uma família de hipergrafos denominada F-grafos. Embora esses grafos possuam uma grande aplicabilidade, têm sido pouco utilizados, principalmente em textos em português. Sendo assim, neste trabalho temos como objetivo revisitar essa estrutura destacando suas características e exemplos de aplicações, além de mostrar que a utilização desses grafos muitas vezes torna mais simples a compreensão de diversos problemas.

**Palavras-chave:** Grafos *E/Ou*. Decomposição de Problemas. Hipergrafos. Teoria de Grafos.

### ABSTRACT

*And/Or* Graphs are acyclic digraphs where the vertices have labels *And* or *Or*. These labels determine whether dependence of a vertex with respect to its out-neighbors is disjunctive or conjunctive. Through these graphs we can represent possible ways to decompose a problem, generate cutting patterns in plates, represent activity networks, model boolean formulas, and define a family of hypergraphs called F-graphs. Although these graphs have a wide applicability, they have been little used, especially in Portuguese texts. Thus, in this work, we aim to revisit this structure, highlighting their characteristics and examples of applications, besides showing that the use of such graphs helps to understand several problems.

**Keywords:** *And/Or* Graphs. Problem Reduction. Hypergraphs. Graph Theory.

## 1 Introdução

Um grafo *E/Ou* é um digrafo  $G$ , tal que todo vértice  $v \in V(G)$  possui um rótulo  $f(v) \in \{E, Ou\}$ . Neste grafo, as arestas representam relações de dependência entre os vértices: vértices do tipo *E* dependem estritamente de todos os seus vizinhos de saída (dependência conjuntiva), enquanto vértices do tipo *Ou* dependem apenas de um dos seus vizinhos de saída (dependência optativa ou disjuntiva).

Grafos *E/Ou* vêm sendo utilizados em Inteligência Artificial desde o início da década de 1970, para *decomposição de problemas* [Nilsson (1971)]. Desde então, ao longo do tempo, novas aplicações para estes grafos vêm surgindo, como em: *versionamento de software* em engenharia de software [Corandi (1998)], geração de *padrões de*

*cortes em placas* em pesquisa operacional [Morabito (2007)], representação de *hipergrafos* em teoria dos grafos [Gallo (1993)] e representação de *game trees* em teoria dos jogos [Kumar (1984)], entre outras.

Na representação de grafos  $E/Ou$ , os vértices do tipo  $E$  são ilustrados com um arco entre suas arestas de saída. A Figura 1 ilustra um grafo  $E/Ou$ , onde  $s$  é um vértice do tipo  $E$  e  $t$  um vértice do tipo  $Ou$ . Na literatura, também são encontradas definições de grafos  $E/Ou$  onde os rótulos  $\{E, Ou\}$  se encontram nas arestas [Gallo (1993)].

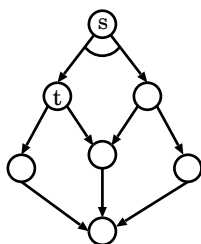


Figura 1. Exemplo de Grafo  $E/Ou$ .

## 2 Decomposição de Problemas

Decomposição de Problemas (*problem reduction*) é uma estratégia utilizada para soluções de problemas complexos, que consiste em decompor problemas em subproblemas com complexidade menor que a do original. A idéia desta estratégia é seguir esta decomposição recursivamente, até alcançar um conjunto de problemas de solução imediata, de forma que a solução desses problemas gere uma solução para o problema original.

Em Inteligência Artificial, grafos  $E/Ou$  são amplamente utilizados para representação de todas as possíveis decomposições de um problema, ou seja, para representar o espaço de busca de decomposição de um problema [Nilsson (1971)]. Nesta estratégia, uma solução para o problema original pode ser encontrada a partir de buscas no espaço das possíveis decomposições para o problema. Esse espaço de busca é geralmente representado graficamente por um grafo  $E/Ou$ . Neste grafo, os vértices contêm a descrição dos problemas (subproblemas) e as arestas indicam como os problemas são decompostos. O vértice fonte  $s$  representa o problema original, os sumidouros subproblemas triviais, e os vértices intermediários subproblemas não triviais. Os vértices do tipo  $Ou$  indicam que os vizinhos de saída são subproblemas optativos, enquanto os vértices do tipo  $E$  indicam que os vizinhos de saída são subproblemas conjuntivos.

A seguir, será apresentado o problema de multiplicação de cadeias de matrizes e a representação do seu espaço de busca como grafo  $E/Ou$ .

**Problema:** MULTIPLICAÇÃO DE CADEIAS DE MATRIZES

**Instância:** Uma cadeia de  $n$  matrizes  $(M_1, M_2, \dots, M_n)$ , onde  $n \geq 2$  e  $M_i$  possui  $r_i$  linhas e  $r_{i+1}$  colunas,  $1 \leq i \leq n$ .

**Questão:** Determinar a ordem em que  $M_1, \dots, M_n$  devem ser multiplicadas, produzindo a matriz  $M_{1,n}$ , de modo a minimizar o número de operações, considerando que o produto de uma matriz  $p \times q$  por outra matriz  $q \times r$  requer  $O(pqr)$  operações.

**Exemplo:** Seja  $M_{1,4} = M_1[10, 20] \times M_2[20, 50] \times M_3[50, 1] \times M_4[1, 100]$ .

A obtenção de  $M_{1,4}$  pela multiplicação  $M_1 \times (M_2 \times (M_3 \times M_4))$  requer 125.000

operações. Já a parentização  $(M_1 \times (M_2 \times M_3)) \times M_4$ , que é ótima, requer apenas 2.200 operações.

## 2.1 Representação do problema utilizando grafos $E/Ou$

Considere  $M_{i,j} = (M_i \times \dots \times M_j)$ . O problema de multiplicação de cadeias de matrizes pode ser modelado como uma árvore  $E/Ou$  da seguinte forma:

$M_{i,j}$  será um nó  $Ou$  com  $(j - i)$  filhos, onde cada filho  $k$  ( $i \leq k \leq j - 1$ ) será um nó  $E$  representando a multiplicação  $(M_{i,k} \times M_{k+1,j})$ . Cada um destes nós  $E$  possui dois filhos, representando, respectivamente,  $M_{i,k}$  e  $M_{k+1,j}$ , onde  $M_{l,l}$  equivale a  $M_l$ . A construção da árvore segue recursivamente até que cada folha represente exatamente uma matriz da instância original do problema ou a multiplicação de duas dessas matrizes. A Figura 2 ilustra a representação das possíveis soluções para a matriz  $M_{1,4}$ , exemplificada acima.

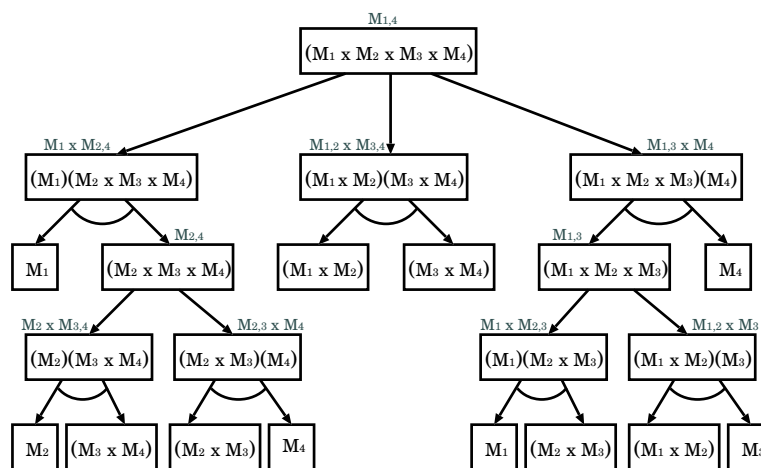


Figura 2. Árvore  $E/Ou$  para multiplicação de cadeias de matrizes.

## 3 Geração de Padrões de Cortes Bidimensionais em Placas

Na área de Pesquisa Operacional, grafos  $E/Ou$  são utilizados para auxiliar na tomada de decisões e na análise de sistemas complexos do mundo real, tipicamente com o objetivo de melhorar ou otimizar a sua performance. Uma das aplicações de grafos  $E/Ou$  em Pesquisa Operacional é a sua utilização na geração de padrões de cortes bidimensionais em placas.

Este problema, que é NP-Difícil [Morabito (2007)], aparece em diversos processos industriais, tais como: corte de bobinas de papel e alumínio, barras de aço, chapas metálicas e de madeira, placas de circuito impresso, caixas de papelão, rolos de tecido, entre outros. A seguir, uma descrição do problema de geração de padrões de cortes guilhotinados em placas é apresentada.

O modo como as peças são arranjadas na placa é chamado *padrão de corte*.

A Figura 3 ilustra dois exemplos de padrões de corte.

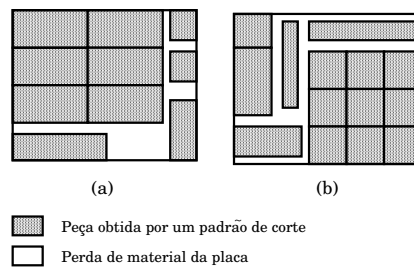


Figura 3. Ilustrações de padrões de cortes.

**Problema:** GERAÇÃO DE PADRÕES DE CORTES BIDIMENSIONAIS EM PLACAS

**Instância:** Uma placa retangular  $P$  de dimensões  $(L, W)$ , onde  $L$  é o comprimento e  $W$  a largura; um conjunto de  $m$  peças retangulares  $p_i$  de dimensões  $(l_i, w_i)$ , tais que  $l_i \leq L$  e  $w_i \leq W$ ,  $1 \leq i \leq m$ ; e um valor utilidade  $v_i$  associado a cada peça  $p_i$ .

**Questão:** Encontrar um padrão de corte da placa  $P$  que produza um conjunto de peças que maximize o valor de utilidade total, ou seja, maximize o somatório dos valores utilidade  $v_i$  das peças obtidas pelo corte. (Se o valor utilidade  $v_i$  for a área da peça  $p_i$ , então o objetivo é equivalente a minimizar a perda de material da placa  $P$ .)

Dada uma placa  $P$ , uma árvore  $E/Ou T$  pode ser definida para representar todos os possíveis padrões de corte de  $P$ . Os vértices do tipo  $Ou$  representam partes da placa, e os vértices do tipo  $E$  possíveis cortes na placa. Nesta representação, cada filho de um vértice  $v$  do tipo  $Ou$  é um vértice do tipo  $E$ , que representa uma possível alternativa de corte na placa representada por  $v$ , e cada vértice  $u$  do tipo  $E$  possui dois filhos (placas obtidas após o corte representado por  $u$ ). A raiz da árvore é um vértice do tipo  $Ou$  que representa a placa  $P$ .

Na árvore  $T$  assim obtida, a busca por um padrão de corte pode ser feita examinando-se todas as possíveis alternativas de cortes a partir de um nó  $Ou$  e, para cada vértice  $E$ , as peças obtidas após o corte.

A partir desta representação por grafos  $E/Ou$ , algoritmos de busca e programação dinâmica podem ser aplicados para encontrar uma solução ótima. Em 2006, Arenales [Arenales (2006)] propôs uma abordagem utilizando grafos  $E/Ou$  para geração de padrões de corte em placas defeituosas. Em seguida, em 2007, Morabito [Morabito (2007)] desenvolveu um método para geração de padrões de cortes restritos utilizando esses grafos.

Nesta representação, um padrão de corte pode ser obtido a partir de uma subárvore (subárvore-solução) com a mesma raiz de  $T$ , onde cada vértice  $Ou$  desta subárvore possui um filho (corte escolhido) e cada vértice  $E$  possui 2 filhos (placas obtidas após o corte), sendo que as peças deste padrão de corte são as folhas dessa subárvore-solução.

A Figura 4 ilustra em (a) uma placa  $P$ , em (b) um conjunto de  $m$  peças que pode ser obtido a partir de  $P$ , e em (c) um possível padrão de corte para  $P$ .

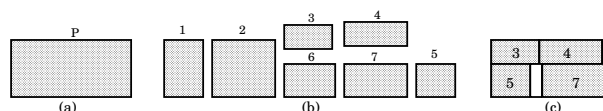


Figura 4. (a) Placa  $P$ . (b) Conjunto das  $m$  peças. (c) Padrão de corte de  $P$ .

A Figura 5 ilustra uma árvore  $E/Ou T$  que modela possíveis cortes na placa  $P$ , onde as folhas são possíveis peças obtidas pelos cortes. Nesta árvore, os vértices do tipo  $Ou$  indicam que cada filho é um possível corte e os vértices  $E$  indicam que seus filhos são placas obtidas a partir de um corte. As arestas em destaque formam a subárvore-solução do padrão de corte representado na Figura 4 (c).

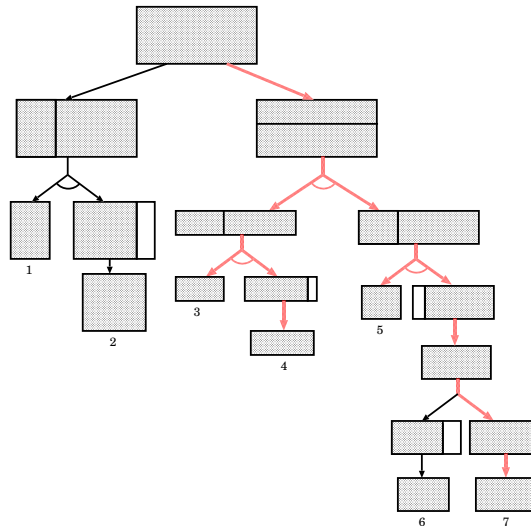


Figura 5. Árvore  $E/Ou$  de padrões de corte da Figura 4.

Muitos trabalhos limitam seus estudos a padrões guilhotinados, isto é, padrões obtidos por uma sequência de cortes em placas retangulares que produzem sempre dois novos retângulos. Alguns trabalhos adicionam ainda uma restrição para o número máximo de vezes que um tipo de peça poderá ser cortado a partir de uma chapa. A existência de possíveis defeitos nas placas também é considerada por alguns autores. Para mais informações sobre o problema, as referências [Morabito (2007), Arenales (2006)] podem ser consultadas.

#### 4 Redes de Atividades

Grafos  $E/Ou$  também são muito utilizados na área de Engenharia de Software. Em 1998, Corandi e Westlehtel [Corandi (1998)] apresentaram um modelo para configurar e administrar versões de software, que se baseia nestes grafos. Uma outra aplicação desses grafos nesta área é a sua utilização na representação de redes de atividades.

Redes de Atividades são basicamente um conjunto de atividades e de relações de dependência entre elas, onde a execução de uma atividade pode ser iniciada somente após a execução de um subconjunto de atividades da qual esta depende. Redes de atividades são muito utilizadas para modelar projetos de software, onde, a partir dessa rede, analisa-se a melhor forma de concluir o projeto.

Em uma rede de atividades, para cada atividade  $p_i$ , existe um subconjunto de atividades  $O_i$ , formado pela união de  $r_i$  subconjuntos não necessariamente disjuntos ( $O_i = O_i^1 \cup O_i^2 \cup \dots \cup O_i^{r_i}$ ), de forma que a atividade  $p_i$  pode ser executada somente após a execução de todas as atividades de pelo menos um subconjunto  $O_i^k$ ,  $1 \leq k \leq r_i$ .

A Tabela 1 exemplifica uma rede de atividades.

### 4.1 Representação de Redes de Atividades como grafos $E/Ou$ .

Uma rede de atividades pode ser representada como um grafo  $E/Ou$  da seguinte forma:

- Cada atividade  $p_i$  é representada por um vértice.
- Se  $r_i = 1$ , então o vértice  $p_i$  é rotulado como  $E$  e uma aresta  $(p_i, p_j)$  é criada para toda atividade  $p_j \in O_i$ .
- Se  $r_i \neq 1$  e  $|O_i^k| = 1$  para todo  $1 \leq k \leq r_i$ , então o vértice  $p_i$  é rotulado como  $Ou$  e uma aresta  $(p_i, p_j)$  é criada para toda atividade  $p_j \in O_i$ .
- Se  $r_i \neq 1$  e  $|O_i^k| \neq 1$  para algum  $1 \leq k \leq r_i$ , o vértice  $p_i$  é rotulado como  $Ou$  e, para todo subconjunto  $O_i^k$  ( $1 \leq k \leq r_i$ ), cria-se um vértice auxiliar  $p_i^k$  do tipo  $E$  e uma aresta  $(p_i, p_i^k)$ , e cria-se uma aresta  $(p_i^k, p_j)$  para toda atividade  $p_j \in O_i^k$ .

Atividade	$O_i$	Dependência
$p_1$	$\{p_2, p_3, p_4, p_5\}$	$\{p_2, p_3\}$ ou $\{p_3, p_4\}$ ou $\{p_4, p_5\}$
$p_2$	$\{p_6\}$	$\{p_6\}$
$p_3$	$\{p_6, p_7, p_8\}$	$\{p_6, p_7, p_8\}$
$p_4$	$\{p_6, p_7, p_8\}$	$\{p_6\}$ ou $\{p_7\}$ ou $\{p_8\}$
$p_5$	$\{p_8\}$	$\{p_8\}$
$p_6$	$\{\}$	Nenhuma
$p_7$	$\{\}$	Nenhuma
$p_8$	$\{\}$	Nenhuma

Tabela 1. Descrição de uma rede de atividades.

A Figura 6 ilustra a representação da rede de atividades correspondente às atividades e relações de dependência da Tabela 1.

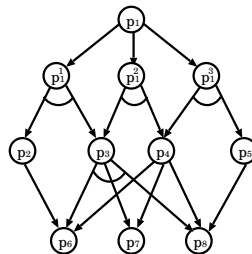


Figura 6. Representação da rede de atividades da Tabela 1.

### 4.2 Escalonamento de grafos $E/Ou$

A interpretação de grafos  $E/Ou$  como redes de atividades muitas vezes possibilita uma melhor compreensão de diversos problemas relativos a esses grafos. Um exemplo de problema onde a interpretação como redes de atividades se torna apropriada é o problema de escalonamento de grafos  $E/Ou$ .

**Problema:** ESCALONAMENTO DE GRAFOS  $E/Ou$   
**Instância:** Um grafo  $E/Ou$  conexo e ponderado  $G$  com um vértice fonte  $s \in V(G)$ , onde cada aresta tem um peso  $\tau(e) \in \mathbb{Z}^+$ .  
**Questão:** Encontrar o menor valor  $t(v_j), \forall v_j \in V(G)$ , satisfazendo às seguintes condições:

$$\begin{cases} t(v_j) = 0 \text{ se } v_j \text{ for sumidouro,} \\ t(v_j) \geq \max_{v_i \in O(v_j)} \{t(v_i) + \tau(v_j, v_i)\} \text{ se } f(v_j) = E, \\ t(v_j) \geq \min_{v_i \in O(v_j)} \{t(v_i) + \tau(v_j, v_i)\} \text{ se } f(v_j) = Ou, \\ t(v_j) \geq 0, \forall v_j \end{cases}$$

O problema de escalonamento de grafos  $E/Ou$  pode ser facilmente interpretado como uma rede de atividades, onde as arestas  $(u, v)$ , para  $u, v \in V(G)$ , representam o tempo necessário para que a atividade  $u$  seja informada do término da execução da atividade  $v$ . O objetivo do problema é encontrar o menor instante de tempo em que cada atividade pode iniciar sua execução.

Na década de 90, Dinic, ao estudar o problema de escalonamento de grafos  $E/Ou$ , exibiu um algoritmo de tempo polinomial para solucioná-lo. Posteriormente, seus resultados foram melhorados, sendo o mais recente um algoritmo de complexidade  $O(nm)$  proposto por Velsky em 2002 [Adelson-Velsky (2002)], para grafos  $E/Ou$  possivelmente cíclicos. Em 2010, Souza [Souza (2010)] desenvolveu um algoritmo de complexidade  $O(m)$  para grafos  $E/Ou$  acíclicos.

A Figura 7 ilustra um exemplo de escalonamento de um grafo  $E/Ou$ .

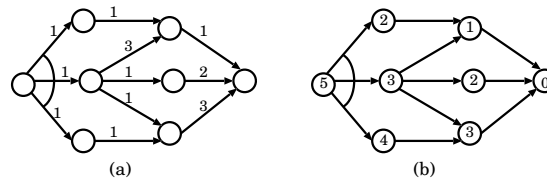


Figura 7. (a) Um grafo  $E/Ou$   $G$ . (b) Grafo  $G$  escalonado.

## 5 Fórmulas Booleanas

Grafos  $E/Ou$  também podem ser utilizados na modelagem de fórmulas booleanas. Estas são basicamente um conjunto  $U$  de variáveis e um conjunto  $F$  de operações (conjunções/disjunções) entre cláusulas. Cada cláusula é composta por operações (conjunções/disjunções) entre literais das variáveis de  $U$ .

Em um grafo  $E/Ou$ , os vértices possuem características que se assemelham a conjunções e disjunções de uma fórmula booleana. Para uma conjunção ser satisfatível, esta depende da satisfabilidade de todos os termos que a compõem, enquanto uma disjunção depende da satisfabilidade de pelo menos um dos termos que a compõem. Assim, conjunções de cláusulas podem ser facilmente modeladas por vértices  $E$ , e disjunções por vértices  $Ou$ . A Figura 8 ilustra em (a) um vértice  $E$  representando a conjunção  $(C_1)(C_2)(C_3)$  e em (b) um vértice  $Ou$  representando a disjunção  $(C_1) + (C_2) + (C_3)$ .

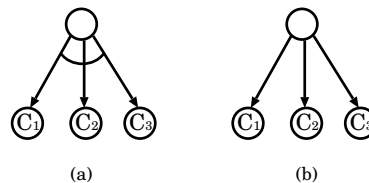


Figura 8. (a) Representação de conjunção. (b) Representação de disjunção.

Desta forma, a partir das características dos vértices  $E/Ou$ , podemos construir facilmente uma árvore  $E/Ou$  modelando uma fórmula booleana  $F$  (não necessariamente  $CNF$ ).

A construção de árvores *E/Ou* modelando fórmulas booleanas pode ser utilizada por exemplo para o desenvolvimento de novos algoritmos probabilísticos para o problema de satisfabilidade de fórmulas booleanas.

A Figura 9 ilustra uma árvore *E/Ou* construída a partir da fórmula  $F = (\bar{x}y\bar{z})(\bar{x} + y + z) + (xyz) + ((x + y + z)(\bar{x}\bar{y}\bar{z}))$ .

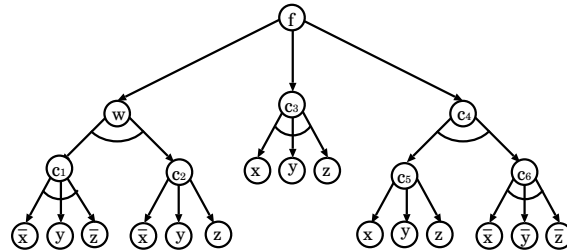


Figura 9. Árvore *E/Ou* de uma fórmula booleana.

### 6 Hipergrafos direcionados

Um hipergrafo direcionado  $H = (V, E)$  é uma generalização de digrafos, onde  $V(H) = \{v_1, v_2, \dots, v_n\}$  é um conjunto de vértices e  $E(H) = \{E_1, E_2, \dots, E_m\}$  é um conjunto de hiperarcos. Todo hiperarco  $E_i \in E(H)$  é um par ordenado de subconjuntos disjuntos de vértices  $(S, T)$ , onde  $S(E_i)$  é o conjunto de saída de  $E_i$  e  $T(E_i)$  o seu conjunto de entrada. Neste trabalho, hipergrafos direcionados serão denominados simplesmente como hipergrafos. A Figura 10 ilustra um hipergrafo direcionado.

A dimensão de um hipergrafo  $H$  não é medida simplesmente pelo seu número de vértices  $|V(H)| = n$  e pelo seu número de hiperarcos  $|E(H)| = m$ . Em um hipergrafo, a sua dimensão depende da cardinalidade de suas hiperarestas, sendo o tamanho de  $H$  definido como o somatório da cardinalidade das hiperarestas:

$$tamanho(H) = \sum_{E_i \in E(H)} |E_i|, \text{ onde } |E_i| = |S(E_i)| + |T(E_i)|$$

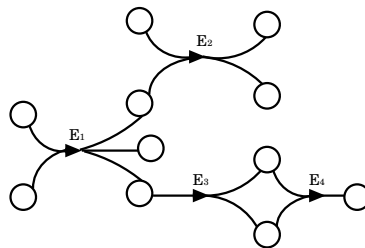


Figura 10. Exemplo de um hipergrafo direcionado.

Um *B-arco* (*backward hyperarc*) é um hiperarco  $E_i = (S, T)$  onde  $|T(E_i)| = 1$ , já um *F-arco* (*forward hyperarc*) é um hiperarco  $E_i = (S, T)$  onde  $|S(E_i)| = 1$  [Gallo (1993)].

Um *B-grafo* (ou *B-hipergrafo*) é um hipergrafo onde todos os hiperarcos são B-arcos. Um *F-grafo* (ou *F-hipergrafo*) é um hipergrafo onde todos os hiperarcos são F-arcos.

F-grafos têm sido muito estudados no contexto de problemas de transporte urbano [Gallo (1993)].

### 6.1 Transformação de $F$ -grafos em grafos $E/Ou$

Dado um  $F$ -grafo  $H$ , este pode ser facilmente transformado em um grafo  $E/Ou$   $G$  equivalente da seguinte forma:

1. Para todo vértice  $v \in V(H)$  atribua valor  $Ou$  para o rótulo  $f(v)$ .
2. Para cada  $F$ -arco  $E_i$ , onde  $T(E_i) \geq 2$ , faça:
  - Criar um vértice  $v_i$ , onde  $f(v_i) = E_i$ .
  - Adicionar uma aresta  $(u, v_i)$ , onde  $u \in S(E_i)$ .
  - Para cada vértice  $w_j \in T(E_i)$  adicionar uma aresta  $(v_i, w_j)$ .
  - Remover o  $F$ -arco  $E_i$ .
3. Transforme todo  $F$ -arco  $E_i = (S, T)$ , tal que  $|S(E_i)| = |T(E_i)| = 1$ , em uma aresta  $e_i = (u, v)$ , onde  $S(E_i) = \{u\}$  e  $T(E_i) = \{v\}$ .

Considerando que cada operação de inserção e remoção de vértices e arestas é feita em tempo constante, a transformação acima será executada em tempo  $O(\text{tamanho}(H))$ .

Como podemos observar, dado o hipergrafo  $H$ , onde

$$|V(H)| = n, |E(H)| = m \text{ e } \text{tamanho}(H) = \sum_{E_i \in E(H)} |E_i|,$$

$H$  será transformado em um grafo  $E/Ou$   $G$ , onde

$$|V(G)| \leq (|V(H)| + |E(H)|) \text{ e } |E(G)| \leq \text{tamanho}(H).$$

A primeira desigualdade se deve ao fato de hiperarestas com apenas um destino não adicionarem vértices na construção do grafo  $E/Ou$ .

A Figura 11 ilustra a transformação de um  $F$ -grafo em um grafo  $E/Ou$ .

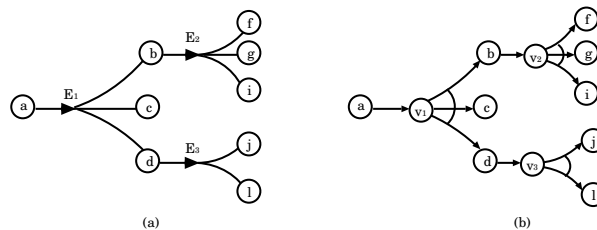


Figura 11. Transformação de um  $F$ -grafo em um grafo  $E/Ou$ .

### 6.2 Transformação de grafos $E/Ou$ em $F$ -grafos

Conforme vimos na subseção anterior,  $F$ -grafos podem ser facilmente transformados em grafos  $E/Ou$ . De forma semelhante, podemos transformar grafos  $E/Ou$  em  $F$ -grafos da seguinte forma:

- Substituir as arestas de saída de todo vértice  $v$  do tipo  $E$  por um  $F$ -arco  $(\{v\}, O_v)$ .

A Figura 12 ilustra a transformação de um grafo  $E/Ou$  em um  $F$ -grafo.

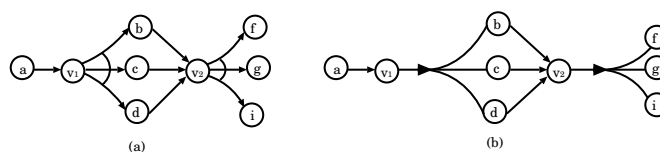


Figura 12. Transformação do grafo  $E/Ou$  em (a) para o  $F$ -grafo em (b).

## 7 Considerações Finais

Grafos  $E/Ou$  são estruturas de dados que possibilitam a modelagem de diversos problemas reais, de forma que soluções para estes problemas possam ser buscadas computacionalmente.

Neste trabalho, é apresentada algumas das aplicações de grafos  $E/Ou$ . Motivou-nos o fato de apesar destes grafos possuírem várias e interessantes aplicações em problemas naturais, não vinham sendo muito estudados na literatura especializada. As referências encontradas eram poucas e, de forma geral, pouco recentes. Sendo assim, é investigado algumas destas aplicações, mostrando a aplicabilidade destes grafos nas mais diversas áreas, como Inteligência Artificial, Pesquisa Operacional e Engenharia de Software.

## Referências

- Adelson-Velsky, G. M., Gelbukh, A. F., and Levner, E.** (2002). On fast path-finding algorithms in and-or graphs. *Mathematical Problems in Engineering*, 8(4):283–293.
- Arenales, M. N. and Vianna, A. C. G.** (2006). O problema de corte de placas defeituosas. *Pesquisa Operacional*, 26(2):185–202.
- Corandi, R. and Westfechtel, B.** (1998). Version models for software configuration management. *ACM Computing Surveys*, 30(2):332–282.
- Gallo, G., Longo, G., Nguyen, S., and Pallottino, S.** (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201.
- Kumar, V. and Kanal, L. N.** (1984). Parallel branch-and-bound formulations for and/or tree search. *IEEE Transactions Pattern Analysis and Machine Intelligence, PAMI-6*, pages 768–778.
- Morabito, R. and Pureza, V.** (2007). Geração de padrões de cortes bidimensionais guilhotinados restritos via programação dinâmica e busca em grafo-e/ou. *Produção, São Paulo*, 17(1):033–051.
- Nilsson, N. J.** (1971). Problem-reduction representations. In Dojny, R. F. and Eakins, M., editors, *Problem Solving Methods in Artificial Intelligence*, pages 80–112. McGraw-Hill, United States of America.
- Souza, U. S.** (2010). *Uma abordagem parametrizada para grafos e/ou e grafos x-de-y*. Mestrado em Informática, Instituto de Matemática, NCE, Universidade Federal do Rio de Janeiro, Rio de Janeiro.