Complexidade de Resolução de Deadlocks em Grafos de Espera de Sistemas Distribuídos

Alan Diêgo Aurélio Carneiro
Universidade Federal Fluminense - Instituto de Computação
aaurelio@ic.uff.br

Fábio Protti
Universidade Federal Fluminense - Instituto de Computação
fabio@ic.uff.br

Uéverton dos Santos Souza
Centro Federal de Educacao Tecnologica Celso Suckow da Fonseca
Universidade Federal Fluminense - Instituto de Computação
usouza@ic.uff.br

RESUMO

Sistemas distribuídos são sistemas de memória compartilhada e compreendem um conjunto de processadores independentes interligados por uma rede de comunicação que permita o compartilhamento de recursos. Um deadlock ocorre em um sistema distribuído quando um conjunto de processos espera indefinidamente por recursos de processos do mesmo conjunto e, usualmente, sistemas distribuídos são representados por grafos de espera. Neste trabalho consideramos a representação de tais sistemas através de grafos E/Ou e Grafos X-de-Y, grafos mais conhecidos na literatura, e mostramos que detectar um deadlock utilizando estas representações pode ser feito em tempo polinomial. Finalmente, é desenvolvido um estudo aprofundado sobre a complexidade de problemas combinatórios relativos à resolução de deadlocks.

PALAVRAS CHAVE. sistemas distribuídos, deadlock, grafos de espera, grafos E/Ou, complexidade computacional.

Área Principal: Teoria e Algoritmos em Grafos

ABSTRACT

Distributed systems are systems with shared memory and consist of a set of independent processors interconnected by a communication network that supports resource sharing. A deadlock occurs in a distributed system when a set of processes wait indefinitely for resources from each other. Distributed systems are usually represented by wait-for graphs. In this paper we consider the representation of such systems by means of And/Or graphs and X-Y graphs, which are more known in the literature, and we show that detecting a deadlock in such representations can be done in polynomial time. Finally, we develop a detailed study of the complexity of combinatorial problems related to deadlock resolution.

KEYWORDS. distributed systems, deadlock, wait-for graphs, And/Or graphs, computational complexity.

Main Area: Theory and Algorithms on Graphs

1. Introdução

Sistemas distribuídos são sistemas de memória compartilhada (Barbosa, 1996) e compreendem um conjunto de processadores independentes (coleção de computadores autônomos) interligados por uma rede de comunicação que permita o compartilhamento de recursos. Tais processadores não compartilham fisicamente memória de forma direta; dessa forma, informações são necessariamente compartilhadas por troca de mensagens através da rede de comunicação.

Os sistemas distribuídos são representados por grafos direcionados e seguem algum modelo como regra de troca de mensagens. Os modelos são divididos em síncronos e assíncronos (Barbosa, 1993). No modelo síncrono (Tanenbaum e Van Steen, 2007) de sistemas distribuídos há um relógio com tempo global a todos os processadores, e um limite superior para a troca de mensagens entre pares de processos. Já no modelo assíncrono (Chandy et al., 1983; Cristian e Fetzer, 1999; Atreya et al., 2007) de sistemas distribuídos não há quaisquer tipos de restrições temporais entre as trocas de mensagens, não há relógio em comum, e não há memória compartilhada.

Podemos desconsiderar a natureza da dependência entre os nós da rede, isto é, não será relevante para os propósitos deste estudo o que faz um nó esperar por outro e sim se a espera ocorre, uma vez que o estudo é direcionado a deadlocks, que é uma propriedade estável destes modelos de sistemas. Uma propriedade é dita estável se, existindo para um tempo Ψ, também existirá em qualquer tempo subsequente a Ψ.

Um conjunto de processos está em deadlock se cada processo deste conjunto está bloqueado, aguardando resposta de um outro processo desse mesmo conjunto; isto é, os processos não conseguem prosseguir a execução, esperando um evento ou resposta que somente outro processo do próprio conjunto pode enviar. O objetivo deste trabalho é estudar problemas combinatórios relativos à resolução de deadlocks em sistemas distribuídos; para tal, consideraremos os grafos de espera de um sistema distribuído definido a seguir.

1.1. Grafos de Espera

Barbosa e Benevides (Barbosa e Benevides, 1998) definem grafos de espera como estruturas de análise e abstração de sistemas distribuídos. Os grafos de espera são dinâmicos, ou seja, mudam de acordo com as trocas de requisições e respostas do sistema. Novamente podemos desconsiderar alguns aspectos, e trabalhar apenas com o grafo em um instante específico de tempo (chamado de *snapshot*), visto que deadlocks existentes no grafo nesse instante implicam na sua existência em instantes subsequentes.

Definiremos o grafo de espera como um grafo direcionado $W = (P, E)$, onde P é o conjunto de processos (vértices) e E o conjunto de requisições (arcos). Um arco existe de um processo P_i para P_j se e somente se P_i fez a uma solicitação a P_j que não foi atendida até o instante corrente. Para cada processo $P_i \in P$, denotaremos por $O_i \subseteq P$ o conjunto de vértices dos quais P_i aguarda resposta.

Sistemas distribuídos são representados por grafos de espera atrelados a um modelo de dependência. Modelos de dependência proporcionam abstração das regras que governam o aguardo de um processo para sua execução. Existem quatro modelos clássicos de espera na literatura (Brzezinski et al., 1995; Kshemkalyani e Singhal, 1994):

Modelo E: Um processo P_i somente torna-se executável quando toda requisição (P_i, P_j) é atendida, isto é, as requisições (P_i, P_j) foram satisfeitas, sendo portanto removidas do grafo de espera corrente, tornando P_i um vértice sumidouro. Este modelo (Figura 1) é caracterizado por situações onde a conjunção de respostas de todas as requisições são necessárias para execução de P_i.

Modelo Ou: Neste modelo, para tornar um processo P_i executável, basta que uma requisição (P_i, P_j) seja atendida; neste caso todas as demais requisições são desconsideradas, tornando P_i um vértice sumidouro. O modelo (Figura 2) é caracterizado por situações de característica disjuntiva onde apenas uma resposta é necessária para P_i.

Modelo E-Ou: Neste modelo, existem $t_i \geq 1$ subconjuntos de O_i associados a P_i. Tais conjuntos são denotados por $O_i^1, \cdots, O_i^{t_i}$ onde $O_i = O_i^1 \cup \cdots \cup O_i^{t_i}$. Para que o processo P_i se torne executável, basta que todas as requisições (P_i, P_j) de pelo menos um subconjunto de O_i sejam atendidas; neste caso todas as demais requisições são desconsideradas, tornando P_i um vértice sumidouro. Este modelo (Figura 3) é caracterizado por processos P_i em que a conjunção de recursos recebida por cada grupo O_i é equivalente.

Modelo X de Y: Neste modelo, para tornar um processo P_i executável, basta que x_i requisições (P_i, P_j) sejam atendidas, neste caso todas as demais requisições são desconsideradas, tornando P_i um vértice sumidouro. Este modelo (Figura 4) é caracterizado por situações onde P_i requisita mais do que ele realmente precisa e espera somente x_i requisições.

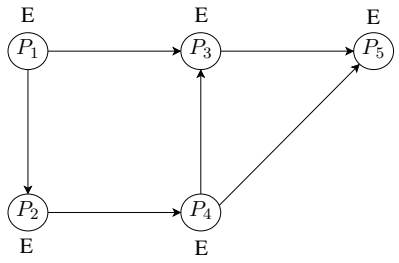

Figura 1: Grafo de Espera: Modelo E.

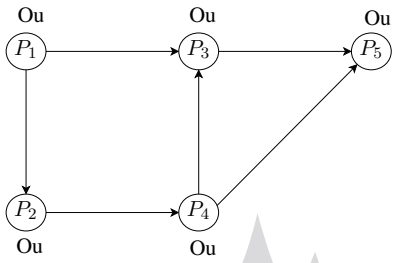

Figura 2: Grafo de Espera: Modelo Ou.

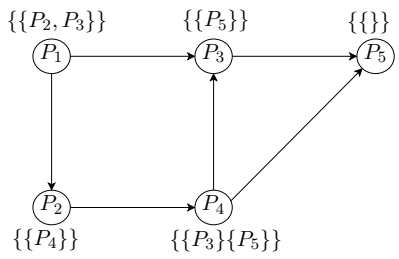

Figura 3: Grafo de Espera: Modelo E/Ou.

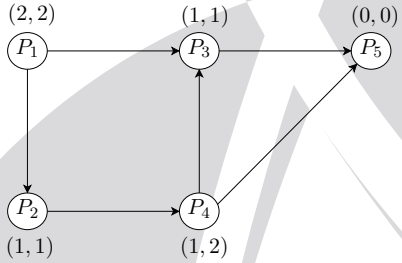

Figura 4: Grafo de Espera: Modelo X-de-Y.

1.2. Deadlocks

Uma condição necessária para o surgimento de deadlocks, segundo (Barbosa, 1996), é a existência de ciclos no grafo de espera associado ao sistema. Nós em deadlock sempre estão à espera de pelo menos um nó pertencente a algum ciclo. Mais especificamente, deadlocks ocorrem se, e somente se, o grafo de espera contém um knot. Um knot é definido como uma estrutura genérica minimal que caracteriza o deadlock.

Embora o princípio básico de um knot independa do modelo, sua caracterização apresenta distinções de acordo com o modelo de espera do grafo em análise; sendo assim, denominamos E-knot, Ou-knot, (E/Ou)-knot e (X-Y)-knot os knots associados aos modelos E, Ou, E/Ou e X-de-Y, respectivamente (Barbosa, 2002):

E-Knot: Um deadlock em um grafo W no modelo E existe se, e somente se, W contém um ciclo; ciclos são chamados E-Knots.

Ou-Knot: Um deadlock em um grafo W no modelo Ou existe se, e somente se, W contém uma componente fortemente conexa C onde não há caminhos de um vértice de C para algum vértice de $W[P \setminus C]$, isto é, uma componente fortemente conexa sem saídas.

(E/Ou)-Knot: Um deadlock em um grafo W no modelo E/Ou existe se, e somente se, W contém um subgrafo $W^{'}$ fortemente conexo tal que para cada vértice $P_i \in P(W^{'})$, pelo menos um vértice de cada subconjunto de O_i também pertence a $W^{'}$.

(X-Y)-Knot: Um deadlock em um grafo W no modelo E/Ou existe se, e somente se, W contém um subgrafo $W^{'}$ fortemente conexo tal que para cada vértice $P_i \in P(W^{'})$, pelo menos $(y_i - x_i + 1)$ nós do conjunto O_i também pertencem a $W^{'}$.

2. Grafos E/Ou e Grafos X-de-Y

Neste trabalho utilizaremos grafos E, Ou, E/Ou, e X-de-Y, estruturas mais conhecidas na literatura (Souza et al., 2012, 2013, 2014), que chamaremos de *convencionais*, para a representação dos sistemas distribuídos, e mostramos transformações polinomiais dos grafos de espera em cada modelo para os grafos convencionais equivalentes. De fato, a representação de grafos de espera para os modelos E, Ou e X-de-Y não sofrem alteração em tamanho de instância.

Definiremos um grafo convencional como um grafo direcionado $G = (V, E)$ onde V é o conjunto de vértices e E o conjunto de arcos. Cada vértice $v_i \in V$ possui um rótulo $f(v_i)$ a ele associado; este rótulo pode ser e, ou, ou um par x-y, o qual indica o tipo de grafo. Os grafos de espera possuem em sua definição o conjunto O_i relacionado a um processo P_i, portanto para a transformação basta que v_i receba um rótulo correspondente ao modelo E, Ou, ou X-de-Y.

No caso do grafo de espera para o modelo E/Ou, para cada $O_i = O_i^1, O_i^2, \ldots, O_i^q$ temos relações disjuntivas entre os subconjuntos, e conjuntivas entre os elementos de cada conjunto; portanto não é possível uma representação precisa utilizando apenas um rótulo.

Dado um grafo de espera $W = (P, E)$ no modelo E/Ou, a transformação para Grafo E/Ou $G = (V, E)$ (Figura 5) é apresentada em (Ryang and Park, 1995) e obtida seguindo os passos abaixo.

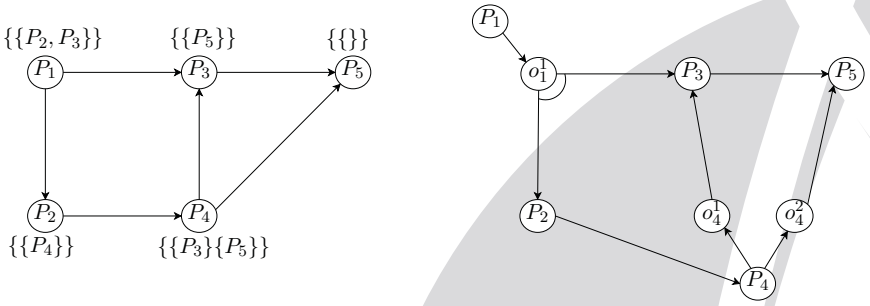

Figura 5: Transformação de grafo de espera no modelo E/Ou para grafo E/Ou.

1. Para cada processo P_i em P, criar um vértice v_i em V.
   (a) Se $|O_i| > 1$, associar a v_i um rótulo $f(v_i) = or$.
   (b) Caso contrário, associar a v_i um rótulo $f(v_i) = e$.
2. Para cada conjunto $|O_i| > 1$, criar q vértices, $o_i^1, o_i^2, \ldots, o_i^q$ em V que chamaremos de artificiais, para cada subconjunto em $O_i = O_i^1, O_i^2, \ldots, O_i^q$.
   (a) Associar a cada vértice v_i criado um rótulo $f(v_i) = e$.
   (b) Criar arcos direcionados de v_i para todos os q vértices $o_i^1, o_i^2, \ldots, o_i^q$ em V.
   (c) Para cada vértice artificial o_i^j, relacionado a um subconjunto O_i^j de tamanho k, criar k arcos direcionados de o_i^j para todos os vértices em V correspondentes aos vértices contidos em O_i^j.

Os problemas de otimização relacionados à resolução de deadlocks nos modelos de grafos E, Ou, E/Ou e X-de-Y são formalmente definidos abaixo.

Deadlock Free Arc Deletion (DFAD).
Entrada: Grafo $G = (V, E)$.
Objetivo: Determinar o número mínimo de arcos que ao serem removidos livra o sistema de deadlock.

Deadlock Free Vertex Deletion (DFVD).
Entrada: Grafo $G = (V, E)$.
Objetivo: Determinar o número mínimo de vértices que ao serem removidos livra o sistema de deadlock.

Deadlock Free Vertex Resolution (DFVR).
Entrada: Grafo $G = (V, E)$.
Objetivo: Determinar o número mínimo de vértices que ao serem resolvidos livram o sistema de deadlock. Um vértice é resolvido ao anular todas suas requisições, tornando-o um sumidouro.

Decorrentes dos problemas acima, de acordo com cada um dos modelos, temos doze diferentes problemas distintos. O objetivo deste trabalho é investigar a complexidade de cada um desses problemas, buscando responder as questões ilustradas na Tabela 1.

	Complexidade			
	Modelo E	Modelo Ou	Modelo E/Ou	Modelo X-de-Y
Deadlock Free Arc Deletion	?	?	?	?
Deadlock Free Vertex Deletion	?	?	?	?
Deadlock Free Vertex Resolution	?	?	?	?

Tabela 1: Complexidade dos problemas de resolução de deadlock associados aos modelos de sistemas distribuídos.

3. Resultados

Nesta seção apresentamos os resultados obtidos para cada modelo de dependência.

3.1. Modelo E

Para determinar se há deadlock em um grafo G no modelo E, basta verificar se G possui ciclo (condição necessária e suficiente). Ciclos podem ser encontrados em tempo linear utilizando uma busca em profundidade. Temos três problemas otimização relacionados ao modelo E:

- DFAD no modelo E: para um dado grafo G, encontrar o número mínimo de arcos formando um conjunto E' tal que $G[E \setminus E']$ seja acíclico, isto é, um grafo livre de deadlock.
- DFVD no modelo E: para um um dado grafo G, encontrar o número mínimo de vértices formando um conjunto S tal que $G[V \setminus S]$ seja acíclico, isto é, um grafo livre de deadlock.
- DFVR no modelo E: para um um dado grafo G, encontrar o número mínimo de vértices formando um conjunto S tal que ao resolver todos os vértices de S (transformá-los em executáveis), o grafo se torna acíclico, isto é, um grafo livre de deadlock.

A complexidade de cada um dada é abaixo:

Teorema 1.

(a) DFAD no modelo E é NP-Difícil.

(b) DFVD no modelo E é NP-Difícil.

(c) DFVR no modelo E é NP-Difícil.

Demonstração. (a) Dado um grafo G no modelo E, encontrar o menor número de arestas a serem removidas para o tornar livre de deadlock corresponde a escolher um conjunto mínimo E' de arestas de forma que $G[E \setminus E']$ seja acíclico, o que se resume ao *Directed Feedback Arc Set Problem* (DFAS), provado NP-Difícil em (Karp, 1972).

(b) Dado um grafo G no modelo E, encontrar o menor número de vértices para o tornar livre de deadlock corresponde a escolher um conjunto mínimo S de vértices tal que $G[V \setminus S]$ seja acíclico, e isto se resume ao *Directed Feedback Vertex Set Problem* (DFVS), provado NP-Difícil também em (Karp, 1972).

(c) Provaremos que DFVS $\propto$ DFVR, o que mostra que DFVR é NP-Difícil. Seja $G = (V, E)$ uma instância de DFVS, ou seja, podemos assumir que G é um grafo no modelo E. Seja $G^{'}$ uma instância de DFVR tal que $G' = G$. Vamos mostrar que $G^{'}$ possui k vértices em um conjunto S cuja resolução torna G' livre de deadlock se, e somente se, $G[V \setminus S]$ é acíclico.

Suponha que G possua um conjunto S de forma que $G[V \setminus S]$ seja acíclico. Podemos então considerar que removendo em $G^{'}$ todos os vértices do conjunto S teremos um grafo acíclico; portanto, ao converter em G' os vértices de S em sumidouros, o grafo resultante permanecerá acíclico pois não há arcos de "feedback"entre os vértices de S e vértices de $V \setminus S$, isto é, o grafo permanece livre de deadlock.

Por outro lado, suponha que $G^{'}$ possui um conjunto S de vértices tal que removendo os arcos de saída de cada vértice de S em $G^{'}$ o grafo resultante seja livre de deadlock. Como os vértices resolvidos se tornam sumidouros, eles não provêem ao grafo nenhum arco de retorno (arco que feche ciclo); portanto, se removidos, o grafo resultante $G[V \setminus S]$ será acíclico. □

3.2. Modelo Ou

Para determinar se há deadlock em um grafo G no modelo OU basta que exista uma componente fortemente conexa sem saídas, ou seja, um Ou-knot. Os Ou-knots podem ser encontrados em tempo linear pelo algoritmo em 1, que por sua vez utiliza o algoritmo apresentado em (Cormen et al., 2009), que realiza duas buscas em profundidade para encontrar as componentes fortemente conexas no grafo. Cada componente fortemente conexa encontrada pode ser contraída a um único vértice, formando assim um digrafo acíclico D onde sumidouros são knots. Veja a Figura 6.

Algoritmo 1: Detecção de knots

Entrada:
G: Grafo Ou
Saída:
s: Lista de knots

```
1 início
2 |  Encontrar as CFC's de G via busca em profundidade
3 |  Computar D
4 |  Para vértice de D (CFC de G), incluí-lo em s se for sumidouro em D.
5 |  retorne s
6 fim
```

Os lemas a seguir serão úteis.

Lema 2. *Considere H um knot em um grafo G no modelo Ou. O menor número de arestas a serem removidas de H para torná-lo livre de deadlock é igual ao menor grau de saída de H.*

Demonstração. Seja H um knot. H deixará de estar em deadlock se, e somente se, para todo vértice de H houver um caminho direcionado deste vértice a um sumidouro de H. Um sumidouro pode ser obtido removendo-se de um vértice v todas os seus arcos de saída. Em H, o menor número

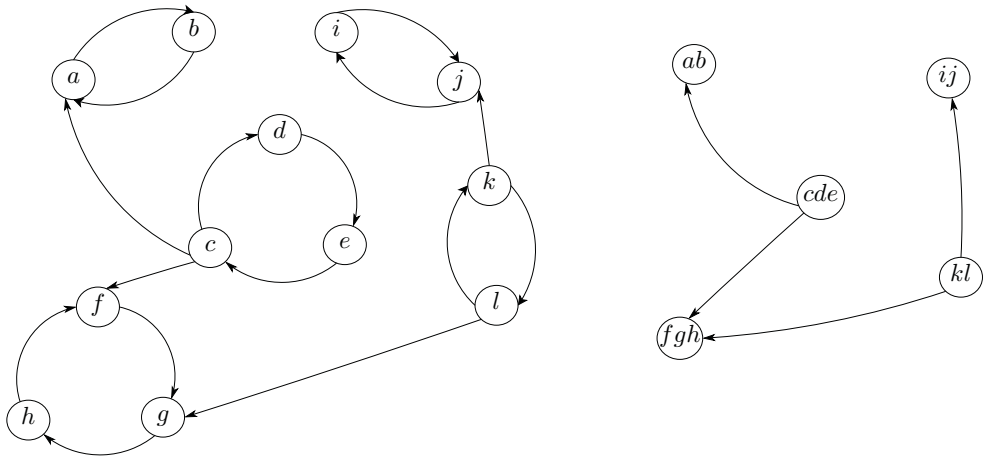

Figura 6: Grafo e árvore obtida pela contração de CFCs.

de remoções de arcos referente a tal operação é igual ao menor grau de saída em H. Como H inicialmente é uma componente fortemente conexa, todo vértice de H possui caminho direcionado para v; após a remoção das arestas de saída de v, os caminhos direcionados a v permanecerão inalterados. □

Lema 3. *Seja G um grafo no modelo Ou, e suponha que G está em deadlock. Então, o menor número de arestas a serem removidas de G para torná-lo livre de deadlock é igual à soma dos menores graus de saída nos knots de G.*

Demonstração. Consideraremos sem perda de generalidade que o grafo G é conexo. Todas as componentes fortemente conexas de G que não são knots possuem um caminho de saída para uma ou mais componentes, podendo assim serem colapsadas em um digrafo acíclico onde o knot sempre será sumidouro. Portanto, resolvendo o deadlock de cada sumidouro, o sistema estará livre de deadlock. Aplicando o lemma 2 a cada knot, teremos um sistema livre de deadlock. □

Através dos Lemas 2 e 3, podemos remover todos os Ou-knots de um grafo no modelo Ou por meio de remoções de arcos com o algoritmo de tempo polinomial apresentado a seguir (Algoritmo 2).

Algoritmo 2: Deadlock Free Arc Deletion

Entrada:
G: Grafo Ou
Saída:
H: Grafo livre de deadlock

```
1 início
2     H ← G
3     s ← lista de knots utilizando o algoritmo de detecção de knots
4     para cada knot em s faça
5         Encontre o vértice v com menor grau de saída em s
6         Remova todos os arcos de saída do vértice v em H
7     fim para cada
8     retorne H
9 fim
```

Como consequência, podemos enunciar:

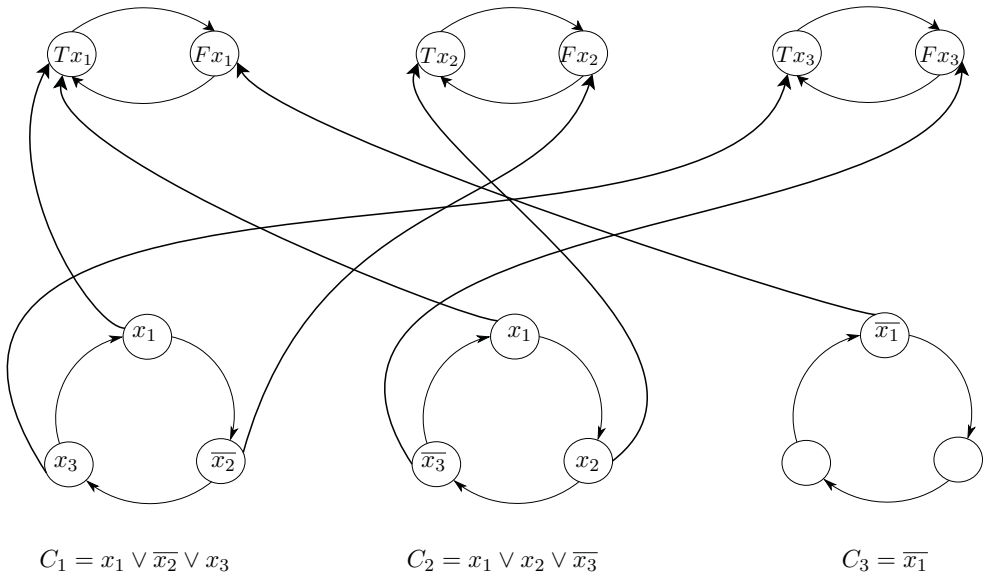

Figura 7: Grafo G resultante da transformação polinomial da fórmula $P = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1})$.

Teorema 4. *DFAD no modelo Ou pode ser resolvido em tempo polinomial.*

Agora trataremos do problema DFVD no modelo Ou:

Teorema 5. *DFVD no modelo Ou é NP-Difícil.*

Demonstração. Provaremos que 3-SAT $\propto$ DFVD, o que mostra que DFVD é NP-Difícil. Seja F uma instância do problema 3-SAT, ou seja, uma fórmula booleana na forma conjuntiva normal, com n variáveis e tendo cada cláusula no máximo 3 literais. Construímos a partir de F um grafo G_F tal que G_F possui $k = n$ vértices que ao serem removidos o tornam livre de deadlock se, e somente se, F for satisfatível. A construção é descrita a seguir.

1. Para cada variável x_i em F, criaremos um ciclo direcionado com dois vértices, Tx_i e Fx_i em G.

2. Para cada clausula C_j em F, criaremos um ciclo direcionado com três vértices. Cada literal da cláusula C_j terá um vértice correspondente no ciclo.

3. Para cada vértice x_i ou $\overline{x_i}$, criar um arco direcionado ligando ao vértice correspondente Tx_i em caso de literal positivo, ou Fx_i caso contrário.

Suponha que F possui uma atribuição A que a satisfaça. Podemos construir um conjunto S de vértices com cardinalidade n, de forma que $G_F^{'} = G_F[V \setminus S]$ seja livre de deadlock. A construção do conjunto S é dada a seguir. Para cada variável de F, selecionaremos um vértice de G_F para fazer parte do conjunto S de acordo com a atribuição em A, isto é, se a variável x_i possui uma atribuição positiva em A, Fx_i será incluído em S, caso contrário Tx_i será incluído em A. Desta forma, já que A satisfaz F, todos os ciclos referentes a cláusulas possuirão caminho para pelo menos um vértice que corresponde ao valor da atribuição de A, assim como todos os knots de G_F formados por ocorrências positivas e negativas de cada variável deixarão de existir em $G_F^{'}$, tornando o sistema livre de deadlock.

Por outro lado, suponha que G_F possua um conjunto S de vértices de cardinalidade n tal que $G_F^{'} = G_F[V \setminus S]$ seja livre de deadlock. Podemos construir uma atribuição A para F da seguinte forma. Por construção, G_F possui n knots, portanto será necessário no mínimo n remoções

para livrar o grafo de deadlock. Removendo um vértice de cada knot teremos uma atribuição de valores verdadeiro/falso para as variáveis em F, e o deadlock permanecerá somente se cada ciclo direcionado em G correspondente às cláusulas não possuir caminho a um sumidouro (vértice de representação do valor atribuído à variável). Logo, se G_F possui solução com n remoções então F é satisfatível. □

Lema 6. *O número de resoluções necessárias para tornar livre de deadlock um grafo G no modelo Ou é igual ao número de knots em G.*

Demonstração. Considerando cada knot em G separadamente, devido à natureza semelhante dos problemas, temos a mesma propriedade usada no lemma 2: um vértice qualquer v de uma componente fortemente conexa C possui caminho direcionado a qualquer outro vértice de C. A resolução de um vértice v qualquer de um uma componente fortemente conexa sem saídas C, isto é, um Ou-knot, tornará v um sumidouro, e todo vértice $v' \in C$ continuará com caminho direcionado a v. Portanto, a solução total do sistema, assim como no lemma 3, será a soma das soluções de cada knot em G. □

Através do Lema 6, podemos remover todos os Ou-knots de um grafo no modelo Ou por meio de resoluções de vértices com o algoritmo de tempo polinomial apresentado a seguir (Algoritmo 3).

Algoritmo 3: DFVR

Entrada:
G: Grafo Ou
Saída:
G': Grafo livre de deadlock

```
1 início
2     G' ← G
3     s ← lista de knots utilizando o algoritmo de detecção de knots
4     para cada knot em s faça
5         Selecione um vértice v de s aleatoriamente
6         Remova todos os arcos de saída do vértice v em G'
7     fim para cada
8     retorne S
9 fim
```

Como consequência, temos:

Teorema 7. *DFVR no modelo Ou pode ser resolvido em tempo polinomial.*

3.3. Modelo E/Ou e Modelo X-de-Y

o modelo E/Ou é uma generalização do modelo E e do modelo Ou, portanto toda instância de um problema de resolução de deadlock no modelo E ou no modelo Ou também será uma instância do problema no modelo E/Ou. Desta observação segue o seguinte teorema:

Teorema 8. *DFAD, DFVD e DFVR no modelo E/Ou são problemas NP-Difíceis.*

Demonstração. Podemos restringir os problemas a DFAS, DFVS e DFVR no modelo E, respectivamente, considerando apenas instâncias onde todo vértice v terá rótulo $f(v) = e$. □

Todo sistema distribuído no modelo E/Ou pode ser transformado em tempo polinomial a um sistema distribuído equivalente no modelo X-de-Y. Logo é facil ver que:

Teorema 9. *DFAD, DFVD e DFVR no modelo X-de-Y são problemas NP-Difíceis.*

Demonstração. Podemos restringir os problemas a, respectivamente, DFAD, DFVD e DFVD novamente, considerando apenas instâncias onde $x_i = y_i$ para todo vértice v_i. □

4. Conclusões

Neste trabalho estudamos a complexidade de problemas de otimização relacionados a deadlocks. Provamos que, no modelo E, DFAD e DFVD são equivalentes aos problemas DFAS e DFVS, provados NP-Difíceis por Karp. Provamos também que DFVR no modelo E é NP-Difícil. No modelo Ou, apresentamos provas de que DFAD e DFVR são problemas polinomiais, e DFVD é NP-Difícil. Finalmente, no modelo E/Ou, generalização dos modelos E e Ou, e no modelo X-de-Y, generelização de todos os demais, todos os problemas estudados NP-Difíceis. Temos assim a Tabela a seguir.

	Complexidade			
	Modelo E	Modelo Ou	Modelo E/Ou	Modelo X-de-Y
Deadlock free arc deletion	NP-D	P	NP-D	NP-D
Deadlock free vertex deletion	NP-D	NP-D	NP-D	NP-D
Deadlock free vertex resolution	NP-D	P	NP-D	NP-D

Tabela 2: Complexidade dos problemas de deadlock associados aos modelos de sistemas distribuídos.

Algoritmos de detecção de knots nos modelos E e Ou também foram apresentados. Esta questão permanece em aberto para os modelos E/Ou e X-de-Y.

Referências

Atreya, R., Mittal, N., Kshemkalyani, A. D., Garg, V. K., and Singhal, M. (2007). Efficient detection of a locally stable predicate in a distributed system. *Journal of Parallel and Distributed Computing*, 67(4):369–385.

Barbosa, V. C. (1993). *Massively Parallel Models of Computation: Distributed Parallel Processing in Artificial Intelligence and Optimisation*. Ellis Horwood.

Barbosa, V. C. (1996). *An Introduction to Distributed Algorithms*. MIT Press.

Barbosa, V. C. (2002). The Combinatorics of Resource Sharing. In *Models for Parallel and Distributed Computation*, pages 27–52. Springer.

Barbosa, V. C. and Benevides, M. R. F. (1998). A graph-theoretic characterization of AND-OR deadlocks. Technical report, UFRJ technical report COPPE-ES-472/98, Rio de Janeiro, Brazil.

Brzezinski, J., Helary, J.-M., Raynal, M., and Singhal, M. (1995). Deadlock models and a general algorithm for distributed deadlock detection. *Journal of parallel and distributed computing*, 31(2):112–125.

Chandy, K. M., Misra, J., and Haas, L. M. (1983). Distributed deadlock detection. *ACM Transactions on Computer Systems (TOCS)*, 1(2):144–156.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT press.

Cristian, F. and Fetzer, C. (1999). The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):642–657.

Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R., Thatcher, J., and Bohlinger, J., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US.

Kshemkalyani, A. D. and Singhal, M. (1994). Efficient detection and resolution of generalized distributed deadlocks. *IEEE Transactions on Software Engineering*, 20(1):43–54.

Ryang, D.-S. and Park, K. H. (1995). A two-level distributed detection algorithm of AND/OR deadlocks. *Journal of Parallel and Distributed Computing*, 28(2):149–161.

Souza, U. d. S., Pinheiro, R. G. S., and Martins, I. C. (2014). Métodos heurísticos e exatos para busca de um subgrafo-solução ótimo de um grafo xy. *Simpósio Brasileiro de Pesquisa Operacional*, XLVI.

Souza, U. d. S., Protti, F., and da Silva, M. D. (2012). Complexidade parametrizada para problemas em grafos E/OU. *Pesquisa Operacional para o Desenvolvimento*, 4(2):160–174.

Souza, U. d. S., Protti, F., and da Silva, M. D. (2013). Revisiting the complexity of and/or graph solution. *Journal of Computer and System Sciences*, 79(7):1156–1163.

Tanenbaum, A. and Van Steen, M. (2007). *Distributed Systems*. Pearson Prentice Hall.