And/Or-Convexity: A Graph Convexity Based on Processes And Deadlock Models

Alan Diêgo Aurélio Carneiro1

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

{aaurelio}@ic.uff.br

Abstract. *Deadlock prevention techniques are essential in the design of robust distributed systems. However, despite the large number of different algorithmic approaches to detect and solve deadlock situations, yet there remains quite a wide field to be explored in the study of deadlock-related combinatorial properties. In this work we consider a simplified AND-OR model, where the processes and their communication are given as a graph G. Each vertex of G is labelled AND or OR, in such a way that an AND-vertex (resp., OR-vertex) depends on the computation of all (resp., at least one) of its neighbors. We define a graph convexity based on this model, such that a set $S \subseteq V(G)$ is convex if and only if every AND-vertex (resp., OR-vertex) $v \in V(G) \backslash S$ has at least one (resp., all) of its neighbors in $V(G) \backslash S$. We relate some classical convexity parameters to blocking sets that cause deadlock. In particular, we show that those parameters in a graph represent the sizes of minimum or maximum blocking sets, and also the computation time until system stability is reached. Finally, a study on the complexity of combinatorial problems related to such graph convexity is provided.*

1. Introdução

Seja V um conjunto de processos que denotam uma computação distribuída. Informalmente, dizemos que um conjunto de processos S está em deadlock quanto todo $i \in S$ está à espera de que se realize alguma condição que somente pode se tornar verdadeira por ação de um ou mais membros do próprio conjunto S. Em outras palavras, uma situação de deadlock é caracterizada pelo impedimento permanente para um conjunto de processos proceder com suas tarefas devido a uma condição que bloqueia pelo menos um recurso essencial a ser adquirido [Barbosa 2002].

Neste apresenta uma relação proxima entre convexidade em grafos com propriedades estáveis em computação distribuída em grafos de espera no modelo E-OU de deadlock.

2. Conceitos Adicionais e Notações

Serão apresentadas nessa seção os principais conceitos e definições utilizadas ao longo do trabalho.

2.1. O modelo E-OU

Neste trabalho é utilizada a terminologia e notação de grafos finitos [Bondy and Murty 2008] com n vértices e m arestas. Seja G um grafo direcionado

com um conjunto de vértices $V(G) = \{v_1, v_2, \ldots, v_n\}$, para $v_i \in V(G)$, $N_G^+(v_i)$ denota o conjunto de descendentes e $N_G^-(v_i)$ o conjunto de ancestrais de v_i em G. Seja $D_i \subseteq N_G^+(v_i)$ o conjunto de descendentes imediatos e $A_i \subseteq N_G^-(v_i)$ o conjunto de ancestrais imediatos em G.

[Barbosa and Benevides 1998] apresenta o modelo de deadlock E-OU de uma computação distribuída como um grafo direcionado G, e para cada $v_i \in V(G)$ existe uma coleção de subconjuntos $W_i^1, \ldots, W_i^{p_i}$ onde:

- $W_i^1 \cup \cdots \cup W_i^{p_i} = D_i$;
- Não há dois conjuntos não vazios em $W_i^1, \ldots, W_i^{p_i}$ tal que um é subconjunto do outro;
- Para que um vértice v_i saia de um estado bloqueado é necessário que receba um sinal de pelo menos um conjunto não vazio de $W_i^1, \ldots, W_i^{p_i}$.

Afim de simplificar a representação, utilizaremos grafos bem conhecidos na literatura (grafos e/ou) que pode ser visto como uma simplificação dos grafos no modelo E-OU de deadlock. Um grafo *e/ou*, em princípio, consiste em um digrafo acíclico G que possui uma fonte e tal que cada vértice $v \in V(G)$ possui um rótulo $f(v) \in \{$*e, ou*$\}$. Em tais digrafos, arcos representam relações de dependência entre os vértices: um vértice rotulado '*e*' depende de todos os seus vizinhos de saída (dependência conjuntiva), enquanto que um vértice rotulado 'ou' depende de apenas um vizinho de saída (dependência disjuntiva). Grafos *e* possuem apenas vértices com rótulos '*e*', e grafos *ou* apenas vértices com rótulos '*ou*'.

2.2. A convexidade E/Ou

Formalmente, dado um grafo G, uma convexidade em grafos é definida por um par $(G, \mathcal{C})$ tal que:

- ϕ é uma coleção de subconjuntos de $V(G)$ fechado em interseções;
- $\emptyset \in \mathcal{C}$;
- $V(G) \in \mathcal{C}$.

Todo $C \in \mathcal{C}$ é chamado convexo. É definida $\mathcal{C}^*$ a família de subconjuntos de vértices de um grafo não direcionado e/ou como a seguir:

(*) Um conjunto C é um membro de $\mathcal{C}$ se e somente se todo vértice ou em $V(G)\backslash C$ não possui vizinhos em C e todo vértice e em $V(G)\backslash C$ tem pelo menos um vizinho em $V(G)\backslash C$.

Analogamente, para a versão de grafos direcionados, definimos a família $\mathcal{C}^{**}$ definida a seguir:

(**) Um conjunto C é um membro de $\mathcal{C}^{**}$ se e somente se todo vértice com rótulo ou não possui vizinhos de saída em C e todo vértice com rótulo e possui pelo menos um vizinho de saída em $V(G)\backslash C$.

Teorema 1 $(C, \mathcal{C}^*)$ *define uma convexidade em grafos.*

Esboço da prova. É trivial para $\emptyset \in \mathcal{C}$ e $V(G) \in \mathcal{C}$. O resto da prova é considerado os dois casos possíveis para vértices considerando dois conjuntos convexos A e B, vértices do tipo ou (consiste em mostrar que se o vértice está fora de $A \cup B$) não interfere na convexidade de $A \cap B$. Para vértices e é análogo.

Corolário 2 $(C, \mathcal{C}^{**})$ *define uma convexidade em grafos.*

Esboço da prova. Similar ao Teorema 1.

2.3. Parametros de Convexidade

A seguir são apresentados algumas convexidades:

- O **convex hull** de um subconjunto $S \subseteq V(G)$ é o menor conjunto convexo $H(S)$ tal que contenha S. A cardinalidade $hn(G)$ de um **hull set** mínimo de G é o **hull number** de G.
- O **convexity number** de G é a maior cardinalidade $cx(G)$ de um conjunto convexo não trivial $C \subseteq V(G)$.
- O **carathéodory number** é o menor inteiro $c(G)$ tal que para todo $S \subseteq V(G)$ e todo $u \in H(S)$, existe um subconjunto $X \subseteq S$ ao qual $|X| \leq c(G)$ e $u \in H(X)$.
- O **interval number** de G é a menor cardinalidade $in(G)$ de um subconjunto $S \subseteq V(G)$ que satisfaz $I_1[S] = V(G)$.

3. Computação de Parâmetros de Convexidade

Para este trabalho, consideraremos sem perda de generalidade apenas grafos conexos. Seja G_{ou} o subgrafo induzido por todos os vértices com rótulo ou (G_e o subgrafo induzido por todos os vértices com rótulo e). Além disso, seja G^i_{ou} a i-ésima componente conexa de G_{ou}, $1 \leq i \leq \omega(G_{ou})$, tal que $\omega(G_{ou})$ denota o número de componentes conexas de G_{ou}. A vizinhança fechada de $S \in V(G)$ é definida como $N[S] = S \cup N_G(S)$, onde $N_G(X) = \bigcup_{v \in X} N_G(V)$.

3.1. Convexity Number

O convexity number de um grafo direcionado G e o tamanho mínimo de um b-knot (estrutura minimal que caracteriza o deadlock) de G são quantidades complementares. Portanto computar $cx(G)$ é equivalente a calcular o tamanho do menor b-knot de G, que é $n - cx(G)$.

Teorema 3 *Dado um grafo e/ou não direcionado G, segue que:*

$$cx(G) = \begin{cases} 0 & \text{, se } V(G_e) = \emptyset) \\ n-2 & \text{, se } V(G_e) \neq \emptyset \text{ e } V(G_e) \text{ não é um conjunto independente;} \\ n - \theta(G) & \text{, caso contrário.} \end{cases}$$

Esboço da prova. Para o primeiro caso é trivial ja que G somente possui vértices do tipo ou. No segundo caso, apontar que se $V(G_e)$ não forma um conjunto independente ha pelo menos um par de vértices vizinhos $uev \in C$ com rótulo e tal que $V(G) \backslash C$ é convexo. Por ultimo, mostrar que a vizinhança fechada de maior cardinalidade de G_{ou} não possui vizinhança entre pares de vértices com rótulo ou e portanto $n - \theta(G)$ é convexo.

Corolário 4 *O convexity number de um grafo direcionado pode ser computado em tempo polinomial.*

Esboço da prova. Para o primeiro caso, basta apontar que existe um vértice v sumidouro. No segundo caso, encontrar o menor ciclo direcionado (pode ser obtido por [Cormen 2009]). Por ultimo, consiste em encontrar um b-knot mínimo B, já que $V(G) \backslash B$ é convexo e pode ser encontrado em tempo polinomial [Barbosa and Benevides 1998].

3.2. O Interval Number

Na prática, o interal number expressa o menor número de processos que devem enviar mensagens de concessão no passo inicial tal que todos os processos restantes convergirem em um passo. De forma geral computar $in(G)$ e NP-Difícil. Se todos os vértices são de rotulo ou $in(G)$ e equivalente a encontrar um conjunto dominante mínimo de G, denotado por $\gamma(G)$ [Bertossi 1984]. Se todos os vértices são de rotulo e $in(G)$ e equivalente a determinar a cardinalidade da cobertura de vértices mínima de G, denotado por $\beta(G)$ [Johnson and Garey 1992].

Teorema 5 *Interval number é NP-Completo para grafos bipartidos conexos com exatamente um vértice com rótulo e.*

Esboço da prova. É considerada a redução do cobertura de vértices em grafos cúbicos. Dado um grafo cubico $G = (V, E)$, é construído um grafo bipartido com um vértice rótulo e, ligado por uma aresta a $|V(G)|$ vértices, cada um representando um vértice de $V(G)$. Estes por sua vez ligados a 3 outros vértices (cada um correspondente a uma aresta). No grafo resultante prova-se que $in(G) = \beta(G)$.

Corolário 6 *Determinar o interval number de um grafo direcionado conexo G é NP-Difícil.*

Esboço da prova. Pode-se representar qualquer grafo não direcionado como um grafo direcionado. Assumindo que G possui apenas vértices do tipo e, é fácil ver que G possui uma cobertura de vértices de tamanho k se, e somente se, G' possui um interval set de tamanho k. Note que se G possua apenas vértices tipo ou obtemos uma redução do problema conjunto dominante.

3.2.1. Algoritmo Aproximativo

Algoritmo 1: Algoritmo $(2 + \log |G_{ou}|)$-aproximativo para $in(G)$

Entrada:
G: Grafo e/ou.
Saída:
S: Um interval set com tamanho no máximo $(2 + \log |G_{ou}|) * OPT$

```
1  início
2      S ← vértices de um emparelhamento maximal de G_e;
3      U ← vértices não cobertos por S;
4      se U ≠ ∅ então
5          Ordenar vértices de G_ou pelo grau em G_ou;
6          enquanto S não é um interval set de G faça
7              S ← S∪ o primeiro vértice da sequência;
8          fim enqto
9      fim se
       Resultado: S;
10 fim
```

Teorema 7 *Algoritmo 1 é* $(2+\log|G_{ou}|)$*-aproximativo para* $in(G)$*. Além disso, computar* $in(G)$ *é um* $Log-APX-Problema.$

Esboço da prova. Sabemos que todo interval set de G deve incluir uma cobertura de G_e. Portanto linha 2 pode ser representada por um algoritmo clássico 2-aproximativo. Já que nenhum vértice com rótulo e é adicionado posteriormente, e é suficiente encontrar uma cobertura de vértices tipo ou, linhas 5-7 apresenta um algoritmo $\log n$ para o problema do conjunto dominante.

3.3. Hull Number

Um conjunto convexo C pode ser visto como um conjunto de processos onde ao liberar todos os processos de C de sua condição de espera, não se libera qualquer processo $w \notin C$ de sua espera. Portanto o hull set de G contém um conjunto mínimo de processos tal que, liberados de sua condição de espera, garante a liberação de todos os processos em G em tempo finito.

Computar $hn(G)$ é trivial para grafos que possuem apenas vértices tipo ou, neste caso, qualquer vértice define um hull set. Contudo, se G não possui vértices tipo ou é NP-Difícil já que $\beta(G_e) \leq hn(G)$ e $hn(G) = \beta(G) = \beta(G_e)$. Baseado neste fato obtemos o seguinte resultado:

Lema 8 *Seja* G *um grafo conexo contendo apenas vértices* e*. Para qualquer hull set* S *de* G*, existe um hull set* S' *com nenhum vértice com rótulo* ou *tal que* $|S'| \leq |S|$*.*

Esboço da prova. Seja S um hull set de G, e $v \in S$ um vértice or. Se $N_G(v) \cap S \backslash \{v\}$ não é vazia, neste caso $S' = S \backslash \{v\}$ é hull set de G. Caso contrário existe pelo menos um vértice vizinho a v que não está em S tal que $S' = \{u\} \cup S \backslash \{v\}$, que também é hull set de G já que $S \subset I_1[S']$ (Aplicar mesmo raciocínio para G^i_{ou}).

Seja $G*$ um grafo e/ou obtido a partir da contração de cada componente G^i_{ou} a um único vértice v^i. Observe que os vértices or de $G*$ induzem um conjunto independente.

Lema 9 *Seja* G^* *um grafo resultante da contração de* G*. Para todo* $S \subseteq V(G)$*,* $H(S) \cap V(G_e) = H(S^*) \cap V(G_e)$*.*

Esboço da prova. Segue do Lema 8.

Lema 10 *Para todo grafo e/ou conexo* G*,* $hn(G) = hn(G^*)$

Esboço da prova. Segue do Lema 8.

Corolário 11 *Se* $V(G_e)$ *é um conjunto independente, então computar* $hn(G)$ *é NP-Difícil.*

Esboço da prova. Consiste em mostrar que encontrar o hull number é equivalente a encontrar a cobertura por vértices de G.

3.4. O Carathéodory Number

Qualquer vértice pode se tornar executável por um grupo de processos, isto é, um grupo de processos tal que transitivamente todos os processos recebam em tempo finito todas as concessões necessárias para que executem suas tarefas. O carathéodory number de G, $c(G)$ indica que cada processo p pode ser liberado por um subconjunto de processos $|X_p| \leq c(G)$.

Teorema 12 *Dado um grafo G:*

$$c(G) = \begin{cases} 1 & \text{, se } G \text{ Contém apenas vértices ou} \\ \Delta(G) & \text{, se } G \text{ Contém apenas vértices ou} \end{cases}$$

Esboço da prova. Já que cada vértice define um hull set de G, se G contém apenas vértices com rótulo ou, é trivial para esse caso. Caso G contém apenas vértices com rótulo e, o convex hull de qualquer subconjunto $S \subset V(G)$ deve conter toda a vizinhança de cada vértice em $V(G) \backslash S$.

O Teorema 12 implica que $c(G) \geq \Delta(G_e)$. A seguir é apresentado um algoritmo polinomial que computa $c(G)$ para qualquer grafo G. Seja $d_e(V)$ o número de vizinhos com rótulo e vizinhos de v. Adicionalmente, dado um vértice v com rótulo e, $\overline{d_{ou}}$ denota o número de vizinhos de v com rótulo ou que não são adjacentes a nenhum vértice com rótulo e na vizinhança de v.

Teorema 13 *Dado um grafo G, $c(G) = max\{1, max_{v \in V(G_e)}\{d_e(v) + \overline{d_{ou}}(v)\}\}$, além disso $c(g)$ é computado em tempo linear.*

Esboço da prova. para $c(G) = 1$ é trivial. Como para qualquer conjunto conexo de vértices tipo em $H(S)$ pode ser obtidos tomando subconjuntos $X \subset S$ com $|X| = 1$, pode-se então considerar apenas vértices e em $H(S) \backslash S$. Já que $|X| \geq d_e(v) + \overline{d_{ou}}(v)$, não é complicado ver que $c(G) \geq max_{v \in V(G_e)}\{d_e(v) + \overline{d_{ou}}(v)\}$. Além disso, não é difícil ver que $c(G) \leq max_{v \in V(G_e)}\{d_e(v) + \overline{d_{ou}}(v)\}$ e portanto $c(G)$ pode ser computado em tempo $O(n + m)$.

References

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. (1998). A graph-theoretic characterization of and-or deadlocks. In *UFRJ Technical Report COPPE-ES-472/98, Rio de Janeiro, Brazil*.

Bertossi, A. A. (1984). Dominating sets for split and bipartite graphs. *Information processing letters*, 19(1):37–40.

Bondy, J. A. and Murty, U. S. (2008). Graph theory, volume 244 of graduate texts in mathematics.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Johnson, D. and Garey, M. (1992). The np-completeness column: an ongoing guide. *Journal of algorithms*, 13(3):502–524.