Resumo de Artigo
Parameterized Complexity of Critical Node Cuts

MO829 - Algoritmos Parametrizados

Celso Aimbiré Weffort Santos - RA:090743

15/12/2016

Resumo

Este texto apresenta de forma resumida os resultados obtidos por D. Hermelin et al em seu artigo "*Parameterized complexity of critical node cuts*" [7]. O artigo analisa o problema Critical Node Cut pela visão de algoritmos parametrizados, apresentando algoritmos FPT para algumas parametrizações, assim como *kernels* de tamanho polinomial e também provando a W[1]-dificuldade para outras. Neste resumo, mostraremos que CNC é W[1]-difícil quando parametrizado por k, que admite *kernel* polinomial quando parametrizado por $k + x$ e que é FPT quando parametrizado por $k + x$ e por y.

1 Introdução

Consideramos o problema de Teoria de Grafos entitulado Critical Node Cut (CNC). Nesse problema, recebemos um grafo simples G e dois parâmetros inteiros, k e x, e desejamos encontrar um conjunto de vértices $C \subseteq V(G)$, com $|C| \leq k$, tal que o grafo $G - C$ (resultado da remoção de C do grafo G), possui no máximo x pares de vértices conectados. Consideramos pares *ordenados* de vértices, ou seja, pares $(u, v) \in V(G)xV(G)$, $u \neq v$, tais que u e v pertencem a mesma componente conexa de $G - C$. O corte C é considerado *crítico* pois sua remoção do grafo G resulta em poucos (no máximo x) pares de vértices conectados. Em outras palavras, o objetivo de CNC é destruir ao máximo a conexidade de G removendo até k vértices.

Definimos formalmente o problema a seguir.

Critical Node Cut (CNC)
Entrada: Um grafo $G = (V, E)$ e $k, x \in \mathbb{N}$
Problema: Existe um corte $C \subseteq V(G)$ de tamanho no máximo k, tal que o grafo $G - C$ possui, no máximo, x pares de vértices conectados?

Destacamos o caso especial de CNC onde $x = 0$, que é o problema Vertex Cover (VC). Como VC é um dos problemas fundamentais em teoria de complexidade, é possível testar diversas técnicas de kernelização e construção de algoritmos FPT para VC em CNC.

1.1 Trabalhos Relacionados

O problema CNC foi provado ser NP-completo por Arulselvan et al. em 2009 [2] (apesar de sua NP-dificuldade vir diretamente do Vertex Cover). Em árvores, CNC pode ser resolvido em tempo polinomial, enquanto a versão de CNC para árvores com pesos nas arestas é NP-completa [8]. Em grafos com *bounded treewidth*, um algoritmo de programação dinâmica resolve o problema em tempo $\mathcal{O}(n^{w+1})$ [1]. Este artigo serviu como base para diversas provas de algoritmos parametrizados para outros problemas de cortes em grafos. Como exemplos, citamos Multicut [4, 9], MultiwayCut [6, 9] e Steiner Multicut [**?**].

1.2 Resultados do Artigo

O artigo examina o problema Critical Node Cut parametrizado por quatro parâmetros distintos e suas combinações. Estes parâmetros básicos são:

- O tamanho k da solução (ou do corte) C.
- O número máximo x de pares de vértices conectados por arestas em $G - C$.
- O número y de pares conectados que devem ser removidos de G; se G é um grafo conexo de n vértices, então $y = n(n-1) - x$.
- O *treewidth* w de G.

A Tabela 1 apresenta todos os resultados obtidos pelos autores para todas as possíveis combinações dos parâmetros acima. Alguns dos resultados podem ser provados facilmente. Primeiramente, como o caso de $x = 0$ é o problema Vertex Cover, sabemos que CNC parametrizado pelo parâmetro x não possui *kernel* polinomial a não ser que $\mathsf{P} = \mathsf{NP}$. Isso também implica que o problema provavelmente não admita *kernel* polinomial parametrizado por $w+x$, pois a existência de tal *kernel* implicaria em um *kernel* polinomial para Vertex Cover ao ser parametrizado por w, que causa o colapso da hierarquia polinomial [3]. Destacamos, também, que se o grafo G não possui vértices isolados, temos $x + y = \Omega(n)$ e, portanto, o problema é FPT e admite um *kernel* polinomial para esta parametrização. O mesmo argumento pode ser utilizado para os parâmetros $x + y + w$ e $k + x + y + w$.

O artigo prova que CNC parametrizado por k é W[1]-difícil. Sendo assim, é muito improvável que admita um algoritmo FPT nestas condições. Quando parametrizado por $k + x$, os autores utilizam duas técnicas para o problema Vertex Cover para encontrar um *kernel* de tamanho polinomial e um algoritmo FPT para resolver o problema.

Com relação ao parâmetro w, o *treedwidth* de G, o artigo demonstra que CNC é W[1]-dfícil nessa parametrização - algo inesperado pelos autores, pois muitos dos problemas de cortes em grafos não são W[1]-difíceis quando parametrizados por *treewidth*. As provas realizadas no artigo para este parâmetro são muito extensas e detalhosas e, portanto, não serão apresentadas neste resumo. Finalmente, ao parametrizar CNC pelo parâmetro y, demonstram que este problema é FPT e que não admite um *kernel* polinomial.

2 Parâmetro k

Consideramos o problema Critical Node Cut parametrizado pelo tamanho do corte k.

Teorema 1. *Critical Node Cut é W[1]-difícil quando parametrizado por k.*

Demonstração. Reduzimos Clique para o nosso problema. Este problema pergunta se, dados um grafo G de n vértices e um parâmetro l, existe K_l como subgrafo induzido em G. Seja (G, l) uma instância de Clique. Construímos o grafo H para nossa instância de CNC como segue. Substituímos cada aresta de G por um *edge-gadget* composto por n arestas paralelas, cada uma das quais é, então, subdividida (operação inversa da contração de arestas). Os vértices criados nestas subdivisões são chamados de *dummy vertices*. Em seguida, acrescentamos uma aresta em H para cada par de vértices não adjacentes em G. Por fim, definimos $k := l$. A construção de H é feita de modo que G possui uma clique de l vértices se e somente se H possui um conjunto de $k = l$ vértices que, ao serem removidos, removem y pares conectados em H, com

$$y = k(k-1) + 2k(N-k) + \binom{k}{2}n \cdot (\binom{k}{2}n - 1) + 2\binom{k}{2}n(N - k - \binom{k}{2}n)$$

Parâmetro				Resultado	
k	x	y	w	FPT	P-*kernel*
✓				*não*	*não*
	✓			*não*	*não*
		✓		*sim*	*não*
			✓	*não*	*não*
✓	✓			*sim*	*sim*
✓		✓		*sim*	*não*
✓			✓	**?**	*não*
	✓	✓		*sim*	*sim*
	✓		✓	*sim*	*não*
		✓	✓	*sim*	*não*
✓	✓	✓		*sim*	*sim*
✓	✓		✓	*sim*	*sim*
✓		✓	✓	*sim*	*não*
	✓	✓	✓	*sim*	*sim*
✓	✓	✓	✓	*sim*	*sim*

Tabela 1: Resultados do artigo *Parameterized compelexity of critical node cuts*

e $N := |V(H)|$.

($\Rightarrow$) Suponha que C é uma clique de tamanho l em G e seja $D(C)$ os $\binom{k}{2}n$ *dummy vertices* que possuem ambos os vizinhos em C. Ao remover C de H, deletamos um total de:

- $k(k-1)$ pares conectados que envolvem os vértices de C,
- $2k(N-k)$ pares que envolvem um vértice de C e um vértice de $V(H)\backslash C$,
- $\binom{k}{2}n \cdot (\binom{k}{2}n - 1)$ pares que envolvem os vértices de $D(C)$, e
- $2\binom{k}{2}n(N-k-\binom{k}{2}n)$ pares que envolvem um vértice em $D(C)$ e outro vértice em $V(H)\backslash(C \cup D(C))$.

($\Leftarrow$). Suponha que exista um corte C que remove y pares conexos em H. Observe que se C possui um subconjunto $C' \subseteq C$ de *dummy vertices*, então podemos substituir C' por um conjunto de tamanho arbitrário de *non-dummy vertices* sem alterar y. Portanto, sem perda de generalidade, consideramos que C é composto apenas por *non-dummy vertices*. Nota-se que ao remover um *non-dummy vertex* v (ou seja, $v \in V(G)$), então os únicos pares conectados removidos são aqueles que envolvem v ou vértices vizinhos de v. Isso ocorre porque cada par de vértices de G está conectado por uma aresta ou por um *edge-gadget* em H. Agora, para cada conjunto C de tamanho k, a remoção de C deleta $k(k-1) + 2k(N-k)$ pares conectados que contém exatamente um vértice de C. Logo, a única maneira de remover y pares conetados em H é se isolarmos os $\binom{k}{2}n$ *dummy vertices*. Isso só é possível se removermos k vértices que são conectados por *edge-gadgets*, que correspondem aos $l = k$ vértices que formam uma clique em G. □

Os autores mostram, também, que através de pequenas alterações na construção do grafo H, o Teorema 1 se mantém para grafos bipartidos, *split* e d-degenerados para qualquer $d \geq 2$ fixo.

Corolário 2. Critical Node Cut *é W[1]-difícil quando parametrizado por k mesmo se o grafo G for* split*, bipartido ou d-degenerado, para qualquer $d \geq 2$ fixo.*

3 Parâmetro $k + x$

Consideramos agora uma outra parametrização para Critical Node Cut. Mostramos que para o parâmetro $k + x$, CNC admite um *kernel* de tamanho polinomial, bem como um algoritmo FPT baseado na técnica de *branching*.

3.1 Algoritmo FPT

Seja (G, k, x) uma instância de CNC e seja n o número de vértices de G. Note que se existe $C \subseteq V(G)$ de tamanho k tal que $G - C$ possui no máximo x pares conectados, então $G - C$ possui, no máximo, x arestas. Utilizando desta observação, resolvemos um problema auxiliar de modo a determinar se (G, k, x) tem solução. O objetivo deste problema auxiliar é determinar se existe k vértices $C' \subseteq V(G)$ tal que $G - C'$ tem, no máximo, x arestas. Este problema pode ser resolvido utilizando a técnica de *bounded search trees*. Para uma aresta arbitrária $e = \{u, v\} \in E(G)$, realizamos um *branch* com novas instâncias $(G - u, k - 1, x)$, $(G - v, k - 1, x)$, e $(G - e, k, x - 1)$. Realizamos este processo recursivamente até não existirem mais arestas em G, ou $k = x = 0$.

Após realizar o algoritmo acima, temos um conjunto $\mathcal{C}'$ contendo todas as possíveis soluções mínimas do problema auxiliar. Se existir uma solução C para nossa instância de CNC, então C também é uma solução do problema auxiliar, embora não seja necessariamente mínima. Verificamos, então, se cada solução mínima de $\mathcal{C}'$ pode ser extendida para uma solução de CNC. Se isso não for possível para toda solução em $\mathcal{C}'$, então concluímos que nossa instância original (G, k, x) é uma instância *não*.

Por fim, analisamos o tempo de execução deste algoritmo. É fácil verificar que resolver o problema auxiliar requer tempo $3^{x+k} \cdot n$ e o tamanho do conjunto $\mathcal{C}'$ de soluções mínimas gerado pelo algoritmo é, no máximo, 3^{x+k}. Analisamos, agora, o tempo para processar cada solução mínima. Suponha que uma solução $C' \in \mathcal{C}'$ contém k_1 vértices. Então, podemos ainda remover $k_2 = k - k_1$ vértices para obter nosso corte. Removemos todos os vértices isolados de $G - C'$, obtendo um grafo com, no máximo, $2x$ vértices. Se $k_2 > 2x$, então todos os vértices podem ser removidos seguramente. Caso contrário, analisamos todas as possíveis maneiras de extender C' para um corte crítico em tempo $\mathcal{O}(\binom{2x}{k_2} x^2 + n)$. Portanto, o tempo de execução total do algoritmo é $\mathcal{O}(3^{x+k}(x^{k+2} + n))$.

3.2 *Kernel* Polinomial

Mostramos agora que o problema Critical Node Cut tem *kernel* de tamanho polinomial quando parametrizado por $k + x$. Seja (G, k, x) uma instância de CNC. Reduzimos essa instância para uma instância (G', k', x) equivalente tal que o número de vértices em G' é polinomial em $k + x$. Inspirado no algoritmo de Buss para o problema Vertex Cover, construímos a instância G' aplicando as seguintes reduções exaustivamente.

Redução CNC.1 Se existe vértice $v \in V(G)$, tal que $d(v) = 0$, remova v. A nova instância é $(G - v, k, x)$.

Redução CNC.2 Se existe vértice $v \in V(G)$, tal que $d(v) > k' + \sqrt{x}$, remova v de G e diminuia k' por um. A nova instância é $(G - v, k' - 1, x)$.

Lema 3. *Se as reduções CNC.1 e CNC.2 não são aplicáveis, então* $|V(G')| \leq k(k + \sqrt{x}) + x + k$.

Demonstração. Suponha que exista solução $C \subseteq V(G')$ de tamanho $k' \leq k$. Então, $G' - C$ possui, no máximo, x pares de vértices conectados. Particionamos os vértices restantes em dois conjuntos A e B, $A \cup B = V(G) \backslash C$. O conjunto A contém todos os vértices isolados em $G' - C$, enquanto B contém todos os vértices que são extremidades de arestas em $G' - C$. Sabemos que $|B| \leq x$. Como a redução CNC.1 não é aplicável, G' não contém vértices isolados, cada vértice em A contém no mínimo um vizinho em C.

Ademais, como cada vértice em C tem, no máximo, $k + \sqrt{x}$ vizinhos e C tem, no máximo, k vértices, isso implica que $A \leq k(k + \sqrt{x})$. Portanto,

$$|V(G')| = |A| + |B| + |C| \leq k(k + \sqrt{x}) + x + k \qquad \square$$

4 Parâmetro y

Consideramos agora o problema CNC parametrizado por y (o número de pares conectados de vértices a serem removidos de G). No artigo, os autores demonstram que o problema não admite *kernel* polinomial quando parametrizado por y e pelo parâmetro composto $k + y + w$. Neste resumo, apresentamos apenas o algoritmo FPT para esta parametrização utilizando da técnica de Programação Dinâmica.

Teorema 4. Critical Node Cut *é FPT quando parametrizado por y.*

Demonstração. Seja (G, k, y) uma instância de CNC, e sejam $G_1, \ldots, G_t$ as componentes conexas de G. Note que se uma das componentes de G for maior que y, ao remover um vértice desta componente estamos, automaticamente, removendo no mínimo y pares conectados de vértices. O mesmo ocorre nos casos em que $k \geq y$. Sendo assim, nos interessam os casos onde $k < y$ e cada componente conexa G_i de G tem tamanho no máximo y. Construímos um algoritmo de Programação Dinâmica como segue.

1. Para cada componente conexa G_i de G e cada k', $1 \leq k' \leq k$, utilizamos força bruta para calcular o número máximo de pares conectados em G_i que podem ser removidos ao deletar exatamente k' vértices de G_i. Denotamos tal número por $T[i, k']$.

2. Calculamos, para cada i, o maior número de pares conectados de vértices que podem ser removidos ao remover exatamente k' vértices nas componentes $G_1, \ldots, G_i$. Denotamos tal número por $Q[i, k']$. Para $i = 1$, temos $Q[1, k'] = T[1, k']$, e para $i > 1$, temos

$$Q[i, k'] = \max_{k'' \leq k'} Q[i - 1, k''] + T[i, k' - k''].$$

3. Se $Q[t, k] < y$, devolvemos *não*; caso contrário, devolvemos *sim*.

A correção do algoritmo segue diretamente: soluções ótimas para diferentes componentes conexas podem ser trivialmente combinadas, pois cada par de vértices conectados pertence a uma mesma componente. Quanto ao tempo de execução do algoritmo, note que cada componente tem, no máximo, y vértices e, portanto, existem $\mathcal{O}(2^y)$ possibilidades na etapa de força bruta. Para cada uma destas possibilidades, calcular o tamanho das componentes conexas restantes pode ser feito em tempo $\mathcal{O}(y^2)$. Finalmente, a programação dinâmica no passo 2 é realizada para $t \leq n$ diferentes valores de i. Para cada i, $k^2 \leq y^2$ combinações de k' e k'' são levadas em conta. Portanto, o tempo de execução total do algoritmo é $\mathcal{O}(2^y \cdot y^2 \cdot n)$. $\square$

5 Exercício

Prove que a *Redução CNC.2* é segura.

Referências

[1] B. Addis, M. D. Summa, A. Grosso, *Removing critical nodes from a graph: complexity results and polynomial algorithms for the case of bounded treewidth* Optimization online (www.optimization-online.org) (2011).

[2] A. Arulselvan, C. W. Commander, L. Elefteriadou, P. M. Pardalos, *Detecting critical nodes in sparse graphs* Computers & Operations Research, **36(7)** (2009), 2193-2200.

[3] H. L. Boadlaender, R. G. Downey, M. R. Fellows, D. Hermelin, *On problems without polynomial kernels* Journal of Computer and System Sciences, **75(8)** (2009), 423-434.

[4] N. Bousquet, J. Daligault, S. Thomassé, *Multicut is FPT* Proceedings of the 43rd Annual Symposium on Theory of Computing (2011), 459-468.

[5] K. Bringmann, D. Hermelin, M. Mnich, E. J. Leeuwen, *Parameterized complexity dichotomy for steiner multicut* Proceedings of the 32nd Annual Symposium on Theoretical Aspects of Computer Science (2015), 157-170.

[6] Y. Cao, J. Chen, J. Fan, *An* $O(1.84^k)$ *parameterized algorithm for the multiterminal cut problem*, Proceedings of the 19th Annual Symposium on Fundamentals of Computer Theory (2013), 84-94.

[7] D. Hermelin, M. Kaspi, C. Komusiewicz, B. Navon, *Parameterized complexity of critical node cuts* Theoretical Computer Science **651** (2015), 62-75.

[8] M. D. Summa, A. Grosso, M. Locatelli, *Complexity of the critical node problem over trees* Computers & Operations Research, **38(12)** (2011), 1766-1774.

[9] M. Xiao, *Simple and improved parameterized algorithms for multiterminal cuts* Theorey of Computer Systems, **46(4)** (2010), 723-736.