Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.8447

Artigo de Pesquisa/Research Paper

Sobre a Computação de um Set Cover Minimal Máximo: Algoritmos Exatos e Limites Inferiores baseados em (S)ETH

On Computing a Maximum Minimal Set Cover: Exact Algorithms and (S)ETH-Based Lower Bounds

Lucas Fraga Damasceno [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) | *al.lucas.damasceno@impatech.edu.br*]
Felipe Ribeiro Mendonça [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) |*al.felipe.mendonca@impatech.edu.br*]
Vinicius Prestes do Nascimento [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) |*al.vinicius.nascimento@impatech.edu.br*]
Cristiane Magarinos Sampaio [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) |*al.cristiane.sampaio@impatech.edu.br*]
Alan David Santos [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) |*al.alan.santos@impatech.edu.br*]
Marcelo Miguel Alves da Silva [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA) |*al.marcelo.silva@impatech.edu.br*]
Uéverton dos Santos Souza [IMPA Tech - Instituto de Matemática Pura e Aplicada (IMPA), Universidade Federal Fluminense |*ueverton@ic.uff.br*]

IMPA Tech, Instituto de Matemática Pura e Aplicada (IMPA) Av. Professor Pereira Reis, 76, Santo Cristo, Rio de Janeiro, RJ, 20.220-800, Brasil

Resumo. Maximum Minimal Set Cover é a variante max-min do clássico problema Set Cover, cujo objetivo é encontrar um set cover minimal de cardinalidade máxima. Este problema generaliza Max Min Vertex Cover e Upper Domination. Embora (Maximum Minimal) Set Cover e (Maximum Minimal) Hitting Set sejam equivalentes sob certos parâmetros, diferenças surgem quando o tamanho do universo, denotado por n, é considerado. Maximum Minimal Hitting Set admite trivialmente um algoritmo em tempo $O^*(2^n)$, mas uma análise exaustiva para Maximum Minimal Set Cover demanda tempo $O^*(2^m)$, onde m é o número de conjuntos. Como m pode ser muito maior que n, investigamos algoritmos exatos exponenciais em função de n para o problema. Mostramos que Maximum Minimal Set Cover pode ser resolvido em tempo $O^*(2^n)$ e que um algoritmo subexponencial de tempo $O^*(2^{o(n)})$ implicaria a falsidade da ETH. Por fim, generalizamos os resultados anteriores apresentando um algoritmo em tempo $O^*(2^k)$, onde k é o tamanho do vertex cover mínimo do grafo de incidência, além de um limite inferior mais apertado baseado na SETH.

Abstract. Maximum Minimal Set Cover is the max-min variant of the classical Set Cover problem, whose goal is to find a minimal set cover of maximum cardinality. This problem generalizes both Max Min Vertex Cover and Upper Domination. Although (Maximum Minimal) Set Cover and (Maximum Minimal) Hitting Set are equivalent under specific parameterizations, differences arise when the size of the universe, denoted by n, is considered. Maximum Minimal Hitting Set trivially admits an $O^*(2^n)$-time algorithm, whereas an exhaustive approach for Maximum Minimal Set Cover requires $O^*(2^m)$ time, where m is the number of sets. Since m can be much larger than n, we investigate exact exponential algorithms parameterized by n for the problem. We show that Maximum Minimal Set Cover can be solved in $O^*(2^n)$ time, and that a subexponential-time algorithm running in $O^*(2^{o(n)})$ would imply that ETH fails. Finally, we generalize these results by presenting an $O^*(2^k)$-time algorithm, where k is the size of a minimum vertex cover of the incidence graph, together with a tighter lower bound based on SETH.

Palavras-chave: Set Cover, Maximum Minimal Set Cover, Algoritmos Exatos Exponenciais, Vertex Cover, ETH, SETH

Keywords: Set Cover, Maximum Minimal Set Cover, Exact Exponential Algorithms, Vertex Cover, ETH, SETH

Recebido/Received: 12 June 2026 • **Aceito/Accepted:** 15 June 2026 • **Publicado/Published:** 10 July 2026

1 Introdução

Problemas clássicos de otimização geralmente buscam por um subconjunto S de um conjunto de entrada que minimiza ou maximiza uma função objetivo dada, sujeita a restrições que caracterizam o espaço de soluções viáveis do problema. Neste contexto, uma solução viável é dita *minimal* caso não contenha propriamente nenhuma outra solução viável, e é dita *maximal* caso não esteja propriamente contida em nenhuma outra solução viável. A maioria dos problemas de minimização podem ser entendidos como a tarefa de encontrar uma solução minimal de custo mínimo e os de maximização, encontrar a solução maximal de custo máximo. Heurísticas para resolver problemas de minimização (ou maximização) geralmente envolvem computar soluções minimais (ou maximais) viáveis, e os estudos nesta área buscam analisar empirica ou teoricamente o quão "próximo" a solução obtida pelo método está da solução ótima. Logo, soluções minimais de custo máximo (*soluções minimais máximas*) representam o pior caso para a performance de certas heurísticas para problemas de minimização. O mesmo se aplica para soluções maximais de custo mínimo em problemas de maximização. Além disso, soluções minimais máximas são relevantes mesmo fora do contexto de seus respectivos problemas de minimização. Encontrar tais soluções, em muitos casos é um problemas mais difícil do que o problema de minimização original, e sua resolução possui aplicações e interesses próprios. Portanto, além dos problemas de minimização e maximização, existe o interesse em estudar problemas *max-min* e *min-max*, que buscam soluções minimais máximas e maximais mínimas, respectivamente.

Versões max-min e min-max de diversos problemas de otimização já foram estudados na literatura, como MINIMUM MAXIMAL INDEPENDENT SET (também conhecido como MINIMUM INDEPENDENT DOMINATING SET) [Bourgeois *et al*., 2013; Halldórsson, 1993; Hurink and Nieberg, 2008], MAXIMUM MINIMAL VERTEX COVER [Boria *et al*., 2015; Zehavi, 2017], LAZY BUREAUCRAT SCHEDULING (uma versão min-max de KNAPSACK) [Arkin *et al*., 2003; Bender *et al*., 2008], MAXIMUM MINIMUM HITTING SET [Araújo *et al*., 2023; Bazgan *et al*., 2018] e MAXIMAL MINIMUM DOMINATING SET (também conhecido como UPPER DOMINATION) [AbouEisha *et al*., 2014; Bazgan *et al*., 2018; Fomin *et al*., 2008].

Como pode ser visto na literatura, versões max-min de problemas clássicos tendem a ser mais difíceis de resolver sob várias perspectivas de complexidade computacional. A menos que P=NP, MAXIMUM MINIMAL VERTEX COVER não admite nenhuma $n^{\frac{1}{2}-\epsilon}$-aproximação, enquanto VERTEX COVER admite uma 2-aproximação [Boria *et al*., 2015]; UPPER DOMINATION não admite nenhuma $n^{1-\epsilon}$-aproximação [Bazgan *et al*., 2018], enquanto DOMINATING SET admite uma $O(\log n)$-aproximação; e LAZY BUREAUCRAT SCHEDULING é APX-difícil enquanto KNAPSACK admite PTAS [Arkin *et al*., 2003]. Quanto a algoritmos exponenciais exatos, o melhor algoritmo conhecido para UPPER DOMINATION requer tempo $O^*(1.7159^n)$ [Fomin *et al*., 2008], enquanto DOMINATING SET pode ser resolvido em tempo melhor do que $O^*(1.5^n)$ [Van Rooij and Bodlaender, 2011]; INDEPENDENT SET pode ser resolvido em tempo $O^*(1.1996^n)$ [Xiao and Nagamochi, 2017], enquanto o melhor conhecido para MINIMUM MAXIMAL INDEPENDENT SET (INDEPENDENT DOMINATING SET) é, até onde sabemos, $O^*(1.3351^n)$ [Bourgeois *et al*., 2013]. Lembre-se que, devido à dualidade entre minimal vertex cover e maximal independent set, o estado da arte quanto a algoritmos exponenciais exatos para VERTEX COVER e MAXIMUM MINIMAL VERTEX COVER é parecido com aquele dos respectivos problemas de independent set.

Neste artigo, estamos principalmente interessados no problema do MAXIMUM MINIMAL SET COVER, a versão max-min do SET COVER. Seja $\mathcal{U} = \{u_1, \ldots, u_n\}$ um conjunto finito dito o universo, e seja $\mathcal{C} = \{S_1, \ldots, S_m\}$ uma coleção onde cada membro S_j de $\mathcal{C}$ é um conjunto $S_j \subseteq \mathcal{U}$. Uma sub-coleção $\mathcal{S} \subseteq \mathcal{C}$ é dita um *set cover* de $\mathcal{U}$ se cada elemento de $\mathcal{U}$ pertence a pelo menos um conjunto em $\mathcal{S}$. Além disso, um subconjunto $H \subseteq \mathcal{U}$ *atinge* um conjunto $S_j \in \mathcal{C}$ se $H \cap S_j \neq \emptyset$, e H é dito um *hitting set* (de $\mathcal{C}$) se atinge todos os conjuntos em $\mathcal{C}$. Os problemas SET COVER e HITTING SET clássicos consistem em encontrar um set cover e um hitting set de cardinalidade mínima, respectivamente. Portanto, suas variantes max-min consistem em encontrar um set cover/hitting set minimal de cardinalidade máxima.

MAXIMUM MINIMAL VERTEX COVER e UPPER DOMINATION são casos particulares de MAXIMUM MINIMAL SET COVER. Além disso, (MAXIMUM MINIMUM) SET COVER e (MAXIMUM MINIMUM) HITTING SET são equivalentes dependendo do aspecto de complexidade analisado – uma conversão natural consiste em considerar cada S_j como um elemento e formar uma coleção com os conjuntos de incidência de cada u_i. Entretanto, quando consideramos o tamanho do conjunto universo como parâmetro, estas duas famílias podem se comportar de formas diferentes. Um algoritmo em tempo $O^*(2^{|\mathcal{U}|})$ para (MAXIMUM MINIMAL) HITTING SET é trivial, enquanto que testar exaustivamente todos os possíveis set covers leva tempo $O^*(2^{|\mathcal{C}|})$. Como $|\mathcal{C}| = m$ pode ser muito maior do que $n = |\mathcal{U}|$, buscamos o algoritmo para MAXIMUM MINIMAL SET COVER com a melhor complexidade de tempo quando analisado em função de n, o tamanho do conjunto universo (c.f. [Cygan *et al*., 2016]).

Em 2004, Fomin, Kratsch e Woeginger apresentaram um algoritmo de programação dinâmica para resolver SET COVER em tempo $O^*(2^n)$ [Fomin *et al*., 2004]. Em 2016, Cygan et. al [Cygan *et al*., 2016] provaram que, a menos que SETH seja provada falsa, não existe um algoritmo para HITTING SET que roda em tempo $O^*((2-\epsilon)^n)$ para qualquer $\epsilon > 0$. Quanto ao SET COVER, não existe algoritmo conhecido melhor do que $O^*(2^n)$ ou alguma conexão de sua não existência com SETH [Cygan *et al*., 2015]. Isso motiva a *Set Cover Conjecture* proposta por Cygan et al. [Cygan *et al*., 2016], que diz que não existe um algoritmo para SET COVER em tempo $O^*((2-\epsilon)^n)$ para qualquer $\epsilon > 0$ (cf. [Cygan *et al*., 2015]). Cygan et al. conectaram a existência de um algoritmo melhor para SET COVER com a existência de algoritmos melhores para diversos outros problemas [Cygan *et al*., 2015].

Neste artigo, focamos em algoritmos exponenciais exatos para MAXIMUM MINIMAL SET COVER. Dado que as versões max-min tendem a ser mais difíceis de resolver do que os seus respectivos problemas de minimização, e que, assumindo a Set Cover Conjecture, não existe um algoritmo para SET COVER mais rápido do que $O^*(2^n)$, analisamos o melhor tempo de execução possível em termos de n para MAXIMUM MINIMAL SET COVER. Na Seção 2, estabelecemos resultados preliminares que eliminam métodos elementares para obter um algoritmo em tempo $O^*(2^n)$ para MAXIMUM MINIMAL SET COVER. Na Seção 3, apresentamos um algoritmo em tempo $O^*(2^n)$ e provamos que, assumindo ETH, o problema não admite solução em tempo $O^*(2^{o(n)})$. Na Seção 4, obtemos um algoritmo melhorado em tempo $O^*(2^k)$, onde k é o tamanho do vertex cover mínimo do grafo de incidência da entrada do problema ($k \leq n$), por fim, provamos que, assumindo SETH, não existe um algoritmo em tempo $O^*((2-\epsilon)^{\frac{k}{4}})$. Na Seção 5 apresentamos nossas considerações finais.

2 Preliminares: Algoritmos de programação dinâmica

Primeiramente, vamos considerar uma variante do SET COVER, chamada MULTICOLORED SET COVER (MSC) que poderá aparecer como um subproblema a ser resolvido, dependendo de como o problema MAXIMUM MINIMAL SET COVER for abordado. Podemos interpretar MSC como uma variante de SET COVER no qual cada conjunto possui uma cor, e objetivamos encontrar um set cover com exatamente um conjunto por classe de cor.

MULTICOLORED SET COVER (MSC)

Instance: Um conjunto universo finito $\mathcal{U} = \{u_1, \ldots, u_n\}$, e coleções $\mathcal{C}_1 = \{S_1, \ldots, S_{m_1}\}, \mathcal{C}_2 = \{S_{m_1+1}, \ldots, S_{m_2}\}, \ldots, \mathcal{C}_k = \{S_{m_{k-1}+1}, \ldots, S_{m_k}\}$, onde cada $S_j \in \mathcal{C}_i$ é um conjunto $S_j \subseteq \mathcal{U}$.

Question: Determinar se existe um set cover S de $\mathcal{U}$ de tamanho k contendo exatamente um membro de cada coleção $\mathcal{C}_i$, $1 \leq i \leq k$.

Cada coleção de uma entrada do MSC representa uma cor. Podemos assumir que os conjuntos estão ordenados por cor; logo, dados $S_1, \ldots, S_{m_k}$ e $m_1, \ldots, m_k$, as coleções estão bem-definidas. Seja $\kappa(\ell)$ o menor inteiro j tal que $\ell \leq m_j$, para $1 \leq \ell \leq m_k$.

Uma programação dinâmica simples para MSC consiste em determinar para cada $U \subseteq \mathcal{U}$ e $\ell \in \{1, 2, \ldots, m_k\}$ se existe um multicolored set cover S para U de tamanho $\kappa(\ell)$ considerando os conjuntos $S_1, \ldots, S_\ell$, i.e., considerando as primeiras $\kappa(\ell)$ coleções de cores, onde para a $\kappa(\ell)$-ésima coleção apenas os conjuntos em $C_{\kappa(\ell)} \cap \{S_1, \ldots, S_\ell\}$ (isto é, $\{S_{m_{\kappa(\ell)-1}+1}, \ldots, S_\ell\}$) podem ser utilizados. A relação de recorrência para esta programação dinâmica é dada abaixo.

$$MSC[U, \ell] = \begin{cases} \text{Se } U = \emptyset: \\ \texttt{true} \\ \text{Se } U \neq \emptyset \text{ e } \ell = 0: \\ \texttt{false} \\ \text{Se } U \neq \emptyset \text{ e } \ell \neq 0: \\ MSC[U, \ell-1] \vee MSC[U \setminus S_\ell, m_{\kappa(\ell)-1}] \end{cases} \quad (1)$$

Agora, denotamos por q-SET COVER a versão do problema SET COVER com a suposição adicional de que cada conjunto tem cardinalidade no máximo q. Em 2016, Cygan et al. [Cygan *et al.*, 2016] propuseram a *Set Cover Conjecture* dada a seguir.

Conjectura 1 (Set Cover Conjecture [Cygan *et al.*, 2016])**.** *Seja λ_q a ínfimo do conjunto das constantes c tais que q-SET COVER pode ser resolvido em tempo $O^*(2^{cn})$, onde n é o tamanho do conjunto universo. Então $\lim_{q\to\infty} \lambda_q = 1$. Em particular, não existe um algoritmo para SET COVER que roda em tempo $O^*((2-\epsilon)^n)$ para qualquer $\epsilon > 0$.*

Dada uma instância de set cover $(\mathcal{U}, \mathcal{C})$, podemos reduzir o problema de determinar se existe um set cover de $\mathcal{U}$ de tamanho no máximo k para o problema do MULTICOLORED SET COVER tomando k cópias de $\mathcal{C}$ e considerando cada uma como uma classe de cor diferente. Portanto, vale o seguinte.

Teorema 1. *MULTICOLORED SET COVER pode ser resolvido em tempo $O^*(2^n)$, mas não pode ser resolvido em tempo $O^*((2-\epsilon)^n)$ para qualquer $\epsilon > 0$, a menos que Set Cover Conjecture seja provada falsa.*

Em seguida, apresentamos um algoritmo simples de programação dinâmica para o problema do MAXIMUM MINIMUM SET COVER, formalmente definido a seguir.

MAXIMUM MINIMAL SET COVER (MAX-MIN SC)

Instance: Um conjunto universo finito $\mathcal{U} = \{u_1, \ldots, u_n\}$, e uma coleção $\mathcal{C} = \{S_1, \ldots, S_m\}$, onde cada $S_j \in \mathcal{C}$ é um conjunto $S_j \subseteq \mathcal{U}$.

Question: Determinar um set cover minimal S de $\mathcal{U}$ de tamanho máximo.

Intuitivamente, o algoritmo de programação dinâmica para MSC se basea em, dado um inteiro ℓ, "adivinhar" a partição de $\mathcal{U}$ em elementos que ainda precisam ser cobertos, U, e os elementos, $\overline{U}$, que já foram cobertos por conjuntos verificados previamente ($S_{\ell+1}, \ldots, S_{m_k}$). Entretanto, para o problema do MAXIMUM MINIMAL SET COVER, os pares (U, ℓ) parecem não fornecer informação suficiente para a programação dinâmica, já que é necessário verificar a condição de minimalidade.

Lembre-se que um set cover S de $\mathcal{U}$ é minimal se e somente se cada $S_j \in S$ possui um *elemento privado*, i.e., um elemento $w \in S_j$ que não pertence a nenhum outro conjunto de S. Portanto, podemos construir um algoritmo de programação dinâmica para MAXIMUM MINIMAL SET COVER "adivinhando": U – o conjunto de elementos que ainda devem ser cobertos; P – um conjunto minimal de elementos privados de conjuntos já selecionados; e ℓ – o índice do maior conjunto a ser considerado para a solução.

Seja $M[U, P, \ell]$ um set cover minimal máximo de U usando apenas conjuntos em $\{S_1, \ldots, S_\ell\}$ que não intersecta P. Observe que $M[\mathcal{U}, \emptyset, m]$ é uma solução ótima para o MAXIMUM MINIMAL SET COVER. Podemos computar $M[U, P, \ell]$ para cada tripla (U, P, ℓ) da seguinte forma.

Definimos a função $\max^*$ que retorna um conjunto de cardinalidade máxima de uma coleção dada. Além disso, a função retorna `null` se todas as entradas forem `null`.

$$M[U, P, \ell] = \begin{cases} \text{Se } U = \emptyset: \\ \emptyset \\ \text{Se } U \neq \emptyset \text{ e } \ell = 0: \\ \texttt{null} \\ \text{Se } P \cap S_\ell \neq \emptyset: \\ M[U, P, \ell-1] \\ \text{se } U \neq \emptyset, P \cap S_\ell = \emptyset \text{ e } \ell > 0: \\ \max^* \begin{cases} M[U, P, \ell-1], \\ \{w\} \cup M[U \setminus S_\ell, P \cup \{w\}, \ell-1] \\ \qquad \forall w \in S_\ell \cap U \end{cases} \end{cases} \quad (2)$$

A partir da relação de recorrência acima, podemos resolver MAXIMUM MINIMAL SET COVER em tempo $O^*(3^n)$. Sob o paradigma da programação dinâmica, não parece possível fazer melhor do que tempo $O^*(3^n)$ para resolver o problema do MAXIMUM MINIMAL SET COVER em vista da necessidade de distiguir os elementos que já foram cobertos dos que ainda devem ser cobertos, além de guardar alguma

informação que garante a minimalidade. Dessa forma, questionamos se $O^*(3^n)$ é um tempo ótimo para este problema.

A seguir, mostramos como obter um algoritmo em tempo $O^*(2^n)$ para o problema sem utilizar programação dinâmica.

3 O truque dos elementos privados

Para obtermos um algoritmo em tempo $O^*(2^n)$ para MAX-MIN SC, não podemos recorrer a enumerar exaustivamente todas as partições de $\mathcal{U}$ em três partes, como fizemos anteriormente na abordagem por programação dinâmica. Uma alternativa natural seria particionar os elementos de $\mathcal{U}$ naqueles que "já foram cobertos" e nos que "ainda não foram cobertos". Entretanto, esta estratégia não parece suficiente: tal partição não retém informação suficiente sobre a estrutura do problema para garantir minimalidade quando novos conjuntos forem adicionados à solução parcial. Em particular, uma vez que os novos conjuntos são adicionados, perdemos a habilidade de verificar se o set cover resultante é minimal, já que não temos informação sobre os elementos unicamente cobertos (privados).

Uma abordagem mais promissora é considerar "adivinharmos" uma partição de $\mathcal{U}$ nos elementos que serão elementos privados na cobertura final e no restante dos elementos. Mais precisamente, suponha que particionamos $\mathcal{U}$ em P e $\mathcal{U} \setminus P$, onde P é o conjunto dos elementos que gostaríamos que servissem como elementos privados na cobertura a ser construída, e é minimal quanto a condição de que cada conjunto selecionado deve possuir um elemento privado. Agora, a principal questão se torna como estender tal partição para uma solução válida, i.e., como selecionar uma família de conjuntos que cobre $\mathcal{U}$ para a qual cada conjunto escolhido contém um único elemento de P, e cada elemento de P está contido em exatamente um dos conjuntos escolhidos, de forma que P certifica a minimalidade da solução. Dado $P \subseteq \mathcal{U}$ minimal em relação a condição de servir como conjunto de elementos privados do set cover a ser construído, a regra a seguir é segura, i.e., sua aplicação não elemina nenhuma solução viável que seja consistente com P:

Regra 1. *Remova qualquer conjunto S_j tal que $|S_j \cap P| = 0$ ou $|S_j \cap P| \geq 2$.*

Esta regra é segura já que, para qualquer solução viável consistente com P, cada conjunto escolhido deve conter exatamente um elemento de P. Neste ponto, cada conjunto restante contêm exatamente um elemento de P. Seja $P = \{p_1, p_2, \ldots, p_k\}$. Definimos uma partição da família $\mathcal{C}$ em $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, onde

$$\mathcal{C}_i = \{S_j \in \mathcal{C} \mid S_j \cap P = \{p_i\}\}, \text{ para cada } 1 \leq i \leq k.$$

Portanto, basta encontrarmos um set cover que selecione exatamente um conjunto de cada coleção $\mathcal{C}_j$. Isto corresponde a uma instância de MULTICOLORED SET COVER, que pode ser resolvida em tempo $O^*(2^n)$, veja o Theorem 1. Entretanto, enumerar todas as 2^n escolhas de $P \subseteq \mathcal{U}$ e, para cada escolha, resolver a instância resultante de MULTICOLORED SET COVER fornece um algoritmo que roda em tempo $O^*(4^n)$. Logo, precisamos evitar resolver MULTICOLORED SET COVER como subrotina para uma escolha fixa de P. De fato, apenas enumerar todos os subconjuntos $P \subseteq \mathcal{U}$ já requer tempo $O^*(2^n)$, esgotando todo o tempo exponencial do algoritmo que queremos construir. Consequentemente, para cada P fixo, a verificação de consistência deve ser realizada em tempo polinomial. A seguir, apresentamos um algoritmo em tempo $O^*(2^n)$ para o problema. O algoritmo enumera todos os subconjuntos $P \subseteq \mathcal{U}$ e, para cada escolha, realiza uma verificação em tempo polinomial.

Teorema 2. *MAXIMUM MINIMAL SET COVER pode ser resolvido em tempo $O^*(2^n)$.*

Demonstração. Assumindo que existe um set cover minimal máximo S, pela minimalidade de S, existe um conjunto P_S de cardinalidade $|S|$ contendo exatamente um elemento privado de cada membro de S. Podemos enumerar todos os subconjuntos $P \subseteq \mathcal{U}$ em tempo $O^*(2^n)$. Para cada P, assumindo que $P = P_S$, isto é, assumindo que P é um conjunto representativo dos elementos privados de algum set cover minimal máximo (que deverá ter tamanho $|P|$), prosseguimos da seguinte forma:

1. Aplique a Regra 1.
2. Caso haja algum elemento u que não atinge nenhum dos conjuntos restantes, então estamos lidando com uma escolha incorreta de P e podemos descartá-lo com segurança. Caso contrário:
    - cada conjunto restante contêm exatamente um elemento de $P = \{p_1, p_2, \ldots, p_k\}$. Logo,
    - encontre um set cover minimal S_P de $\mathcal{U}$ considerando apenas os conjuntos restantes (note que S_P pode ser encontrado em tempo polinomial já que requermos apenas minimalidade).

Se o set cover S_P obtido a partir de P tiver cardinalidade maior que $|P|$ então P não é um conjunto representativo dos elementos privados de algum set cover minimal máximo. Além disso, se o set cover tiver tamanho $|P|$ então ele é uma solução da instância de MSC que poderíamos ter construído. Por fim, S_P não pode ser menor do que P já que existe uma partição de $\mathcal{C}$ em $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, onde $\mathcal{C}_i = \{S_j \in \mathcal{C} \mid S_j \cap P = \{p_i\}\}$, para $1 \leq i \leq k$.

Em outras palavras, depois de aplicarmos a Regra 1, temos que o conjunto P define uma partição dos conjuntos restantes. Logo, qualquer set cover minimal tem tamanho no mínimo $|P|$. Em particular, para um conjunto representativo P de uma solução ótima, a Regra 1 é segura; o set cover minimal computado deve ter tamanho $|P|$; e será o maior possível (a solução ótima deve ter tamanho $|P|$). Portanto, o maior S_P obtido a partir de algum P é um set cover minimal máximo de $\mathcal{U}$. □

3.1 Limite inferior algorítmico baseado em ETH

Agora, vamos discutir possíveis cotas inferiores para MAX-MIN SC baseado na Hipótese de Tempo Exponencial (Exponential Time Hypothesis) e no Lema de Esparcificação (Sparsification Lemma). Para mais detalhes sobre o Sparsification Lemma, veja [Cygan *et al.*, 2015]. A Exponential Time Hypothesis é enunciada a seguir.

Conjectura 2 (Exponential Time Hyphothesis (ETH) [Cygan *et al.*, 2015])**.** *Seja λ_k o ínfimo do conjunto de constantes c tais que k-SAT pode ser resolvido em tempo $O^*(2^{cn'})$, onde n' é o número de variáveis. Vale que $\lambda_3 > 0$. Em particular, não existe algoritmo para 3-SAT que roda em tempo $2^{o(n'+m')}$ pelo Sparsification Lemma, onde m' é o número de cláusulas.*

Teorema 3. *A menos que ETH seja provada falsa, Maximum Minimal Set Cover não pode ser resolvido em tempo $O^*(2^{o(n)})$.*

Demonstração. Dada uma fórmula CNF F com variáveis $x_1, \ldots, x_{n'}$ e cláusulas $C_1, \ldots, C_{m'}$, construímos a seguinte instância de Max-Min SC:

- para cada cláusula C_j de F, adicione um elementos u_{C_j} em $\mathcal{U}$;
- para cada variável x_i de F, adicione os elementos $u^1_{x_i}, u^2_{x_i}, u^3_{x_i}, u^4_{x_i}, u^5_{x_i}$ em $\mathcal{U}$.
- para cada variável x_i de F, adicione em $\mathcal{C}$ os seguintes conjuntos (gadget G_{x_i}):
  - $T_{x_i} = \{u^1_{x_i}, u^2_{x_i}, u^3_{x_i}\} \cup \{u_{C_j} : x_i \text{ is a literal of } C_j\}$;
  - $F_{x_i} = \{u^1_{x_i}, u^4_{x_i}, u^5_{x_i}\} \cup \{u_{C_j} : \overline{x_i} \text{ is a literal of } C_j\}$;
  - $A_{x_i} = \{u^2_{x_i}, u^5_{x_i}\}$;
  - $B_{x_i} = \{u^3_{x_i}, u^4_{x_i}\}$;

  Isso termina a construção. Note que $n = |\mathcal{U}| = 5n' + m'$.

Se F possui uma atribuição verdadeira $\tau(F)$, então podemos obter um set cover minimal S de tamanho $3n'$, tomando para cada variável x_i os seguintes conjuntos:

- A_{x_i};
- B_{x_i};
- T_{x_i} se x_i é `true` em $\tau(F)$; ou
- F_{x_i} se x_i é `false` em $\tau(F)$;

Claramente, S é um set cover de $\mathcal{U}$ já que $\tau(F)$ é uma atribuição verdadeira. Além disso, é minimal já que ou $(u^1_{x_i}, u^2_{x_i}, u^3_{x_i})$ ou $(u^1_{x_i}, u^4_{x_i}, u^5_{x_i})$ são elementos privados para cada gadget G_{x_i}.

Reciprocamente, suponha que existe um set cover minimal S de $\mathcal{U}$ com pelo menos $3n'$ conjuntos. Por construção, para cada gadget G_{x_i} no máximo 3 conjuntos podem ser utilizados em um set cover minimal. Logo, S contêm exatamente 3 conjuntos por gadget G_{x_i}, e deve conter ou T_{x_i} ou F_{x_i}, o que define uma atribuição para x_i (x_i = `true` sse $T_{x_i} \in S$). Como cada v_{C_i} é coberto por S, F possui uma atribuição que a satisfaz. Isto conclui a prova. □

4 Sobre o vertex cover mínimo do grafo de incidência

Dada uma instância $(\mathcal{U}, \mathcal{C})$ de (Max-Min) Set Cover, o grafo de incidência de $(\mathcal{U}, \mathcal{C})$ é um grafo bipartido B obtido criando um vértice v_u para cada elemento $u \in \mathcal{U}$, um vértice v_{S_j} para cada $S_j \in \mathcal{C}$, e adicionando arestas entrea v_u e v_{S_j} se $u \in S_j$.

Note que o vertex cover mínimo de B tem cardinalidade no máximo $\min\{m, n\}$.

A seguir, apresentamos um algoritmo em tempo $O^*(2^k)$ para Max-Min SC, onde k é o número de vertex cover de B. Lembre-se que um vertex cover mínimo em um grafo bipartido pode ser encontrado em tempo polinomial, e $k \leq \min\{n, m\}$. Logo, este resultado é uma melhoria do algoritmo de tempo $O^*(2^n)$ apresentado no Theorem 2.

Teorema 4. *Maximum Minimal Set Cover pode ser resolvido em tempo $O^*(2^k)$, onde k é o tamanho do vertex cover mínimo do grafo de incidência da entrada.*

Demonstração. Seja B o grafo de incidência de uma instância $(\mathcal{U}, \mathcal{C})$. Seja R um vertex cover mínimo de B, e seja $I = V(B) \setminus R$.

Defina $R_\mathcal{U} = R \cap \mathcal{U}$, $R_\mathcal{C} = R \cap \mathcal{C}$, $I_\mathcal{U} = I \cap \mathcal{U}$ e $I_\mathcal{C} = I \cap \mathcal{C}$.

Primeiramente, analizamos exaustivamente todos os subconjuntos S' de $R_\mathcal{C}$. Intuitivamente, estamos procurando pela interseção de um set cover ótimo S com $R_\mathcal{C}$ e testando cada S' como um possível candidato.

• Para um $S' \subseteq R_\mathcal{C}$ dado, prosseguimos da seguinte forma:

1. Se S' não cobre $I_\mathcal{U}$, então ele não pode ser a interseção de nenhum set cover com $R_\mathcal{C}$. Logo, podemos descartá-lo.
2. Se algum S' não possui um elemento privado com respeito aos outros membros de S', então S' não pode ser estendido para um set cover minimal. Logo, podemos descartá-lo.
   Agora, suponha que S' cobre $I_\mathcal{U}$ e cada membro de S' possui um elemento privado com respeito aos outros elementos de S'.
3. Seja S'_0 a família dos conjuntos em S' que não possuem um elemento privado em $I_\mathcal{U}$. Para cada $P' \subseteq R_\mathcal{U}$ minimal quanto a condição de que cada conjunto em S'_0 deve possuir um elemento privado em P', prosseguimos da seguinte forma.
4. Remova: $R_\mathcal{C}$; $I_\mathcal{U}$; cada elemento de $R_\mathcal{U}$ que pertence a algum conjunto de S'; e cada conjunto de $I_\mathcal{C}$ que contêm algum elemento de P'.
   Intuitivamente, estamos assumindo que $S' = S \cap R_\mathcal{C}$ e P' um conjunto representativo dos elementos privados de S'_0 (eles podem também ocorrer em conjuntos de $I_\mathcal{C}$). Logo, queremos estender S' para um set cover minimal de $\mathcal{U}$. Para isso, resta cobrir os elementos de $R_\mathcal{U}$ que não foram cobertos por S' utilizando conjuntos em $I_\mathcal{C}$ que não intersectam P'.
5. Neste ponto, após as remoções descritas anteriormente, é suficiente executar o algoritmo para Max-Min SC apresentado no Theorem 2. Se o algoritmo não encontrar nenhuma solução, podemos descartar o par (S', P'). Caso contrário, seja S'' o set cover retornado pelo algoritmo.
6. Retorne $S' \cup S''$ para o par (S', P').

• Finalmente, tomamos como solução máxima o maior entre todos os set covers retornados dentre os pares viáveis (S', P').

Enumeramos todos os possíveis S' em tempo $O(2^{|R_C|})$. Para um S' dado, verificar Passo 1 e Passo 2, bem como computar S'_0 pode ser feito em tempo polinomial. Temos também que os elementos de $R_{\mathcal{U}}$ podem ser particionados de foram eficiente em: $R^1_{\mathcal{U}}$ – o conjunto dos elementos cobertos por S', e $R^2_{\mathcal{U}}$ – os elementos que não foram cobertos por S'. Além disso, para um S' dado, podemos enumerar todos os possíveis P' em tempo $O^*(2^{|R^1_{\mathcal{U}}|})$ (Passo 3). Dado (S', P'), podemos realizar o Passo 4 em tempo polinomial.

Finalmente, aplicamos o algoritmo apresentado no Teorema 2 para encontrar um set cover minimal máximo dos elementos restantes ($R^2_{\mathcal{U}}$) utilizando os conjuntos restantes, o que realizado em tempo $O^*(2^{|R^2_{\mathcal{U}}|})$. Portanto, o tempo de execução total do algoritmo é $O^*(2^{|R_C|} \cdot 2^{|R^1_{\mathcal{U}}|} \cdot 2^{|R^2_{\mathcal{U}}|}) = O^*(2^k)$, onde k é o tamanho do vertex cover mínimo de B.

Note que P' e a remoção dos conjuntos que o intersectam garantem que S'' é compatível para estender S' para um set cover minimal de $\mathcal{U}$ (i.e., $S' \cup S''$); ademais, todos os pares (S', P') relevantes estão enumerados, implicando que na corretude. □

4.1 Limite inferior algorítmico baseado em SETH

A seguir, apresentamos uma cota inferior para MAX-MIN SC parametrizada por k, baseada na Hipótese de Tempo Exponencial Forte (Strong Exponential Time Hypothesis), enunciada a seguir.

Conjectura 3 (Strong Exponential Time Hyphothesis (SETH) [Cygan *et al.*, 2015])**.** *Seja λ_k o ínfimo do conjunto das constantes c tais que k-SAT pode ser resolvido em tempo $O^*(2^{cn'})$, onde n' é o número de variáveis. Vale que $\lim_{k\to\infty} \lambda_k = 1$. Em particular, não existe um algoritmo para SAT que roda em tempo $O^*((2-\epsilon)^{n'})$ para nenhum $\epsilon > 0$.*

Corolário 1. *A menos que SETH seja provada falsa, para qualquer $\epsilon > 0$ vale os seguintes resultados:*

- *MAX-MIN SC não pode ser resolvido em tempo $O^*((2-\epsilon)^{\frac{k}{4}})$, onde k é o tamanho do vertex cover mínimo do grafo de incidência.*
- *MAX-MIN SC não pode ser resolvido em tempo $O^*((2-\epsilon)^{\frac{m}{4}})$.*

Demonstração. Segue diretamente da redução apresentada no Teorema 3, e do fato de que a instância resultante possui exatamente $4n'$ conjuntos. □

5 Conclusão

Neste trabalho, investigamos o problema MAXIMUM MINIMAL SET COVER sob a perspectiva de algoritmos exponenciais exatos e complexidade parametrizada. Embora MAXIMUM MINIMAL SET COVER e MAXIMUM MINIMAL HITTING SET possam ser considerados equivalentes sob diferentes transformações clássicas, mostramos que diferenças fundamentais emergem quando o tamanho do universo é tomado como parâmetro de análise. Em particular, enquanto MAXIMUM MINIMAL HITTING SET admite trivialmente um algoritmo em tempo $O^*(2^n)$, não é imediato obter a mesma complexidade para MAXIMUM MINIMAL SET COVER, uma vez que abordagens exaustivas naturais dependem exponencialmente do número de conjuntos.

Nossos resultados mostram que MAXIMUM MINIMAL SET COVER pode ser resolvido em tempo $O^*(2^n)$, estabelecendo que a dependência exponencial no tamanho do universo é suficiente para o problema. Além disso, provamos que, assumindo a Hipótese de Tempo Exponencial (ETH), não existe algoritmo em tempo $O^*(2^{o(n)})$, indicando que o resultado obtido é essencialmente ótimo sob hipóteses amplamente aceitas em Complexidade Computacional. Também apresentamos um algoritmo parametrizado em tempo $O^*(2^k)$, onde k é o tamanho de um vertex cover mínimo do grafo de incidência da instância. Como $k \leq \min\{n, m\}$, este resultado refina a análise baseada apenas no tamanho do universo. Complementarmente, apresentamos um limite inferior mais apertado assumindo a Hipótese de Tempo Exponencial Forte (SETH), mostrando que não existe algoritmo em tempo $O^*((2-\epsilon)^{k/4})$, para qualquer $\epsilon > 0$.

Os resultados apresentados contribuem para uma melhor compreensão das diferenças estruturais entre variantes de problemas de cobertura sob diferentes parametrizações, além de motivarem novas investigações sobre algoritmos exatos exponenciais e limites inferiores condicionais para problemas max-min em otimização combinatória.

Declarações complementares

Financiamento

Os autores agradecem o apoio financeiro recebido por meio de bolsas do Programa de Iniciação Científica e Mestrado (PICME), promovido pelo IMPA em parceria com a CAPES e o CNPq.

Contribuições dos autores

Todos os autores contribuíram igualmente para a conceitualização, metodologia e análise formal do manuscrito. Vinicius Prestes do Nascimento é o escritor deste manuscrito. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Não se aplica. Este trabalho é de natureza puramente teórica e matemática e não envolve dados, materiais ou códigos computacionais passíveis de disponibilização.

Outras informações relevantes

Em conformidade com as convenções das comunidades de Teoria da Computação e Matemática, os autores estão listados em ordem alfabética de sobrenome.

Referências

AbouEisha, H., Hussain, S., Lozin, V., Monnot, J., and Ries, B. (2014). A dichotomy for upper domination in monogenic classes. In *International Conference on Combinatorial Optimization and Applications*, pages 258–267. Springer. DOI: 10.1007/978-3-319-12691-3_20.

Araújo, J., Bougeret, M., Campos, V. A., and Sau, I. (2023). Parameterized complexity of computing maximum mini-

mal blocking and hitting sets. *Algorithmica*, 85(2):444–491. DOI: 10.1007/s00453-022-01036-5.

Arkin, E. M., Bender, M. A., Mitchell, J. S., and Skiena, S. S. (2003). The lazy bureaucrat scheduling problem. *Information and Computation*, 184(1):129–146. DOI: 10.1016/S0890-5401(03)00060-9.

Bazgan, C., Brankovic, L., Casel, K., Fernau, H., Jansen, K., Klein, K.-M., Lampis, M., Liedloff, M., Monnot, J., and Paschos, V. T. (2018). The many facets of upper domination. *Theoretical Computer Science*, 717:2–25. DOI: 10.1016/j.tcs.2017.05.042.

Bender, M. A., Clifford, R., and Tsichlas, K. (2008). Scheduling algorithms for procrastinators. *Journal of Scheduling*, 11(2):95–104. DOI: 10.1007/s10951-007-0038-4.

Boria, N., Della Croce, F., and Paschos, V. T. (2015). On the max min vertex cover problem. *Discrete Applied Mathematics*, 196:62–71. DOI: 10.1016/j.dam.2014.06.001.

Bourgeois, N., Della Croce, F., Escoffier, B., and Paschos, V. T. (2013). Fast algorithms for min independent dominating set. *Discrete Applied Mathematics*, 161(4-5):558–572. DOI: 10.1016/j.dam.2012.01.003.

Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., and Wahlström, M. (2016). On problems as hard as CNF-SAT. *ACM Transactions on Algorithms (TALG)*, 12(3):1–24. DOI: 10.1145/2925416.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. (2015). *Parameterized Algorithms*. Springer.

Fomin, F. V., Grandoni, F., Pyatkin, A. V., and Stepanov, A. A. (2008). Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. *ACM Transactions on Algorithms (TALG)*, 5(1):1–17. DOI: 10.1145/1435375.1435384.

Fomin, F. V., Kratsch, D., and Woeginger, G. J. (2004). Exact (exponential) algorithms for the dominating set problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer. DOI: 10.1007/978-3-540-30559-0_21.

Halldórsson, M. M. (1993). Approximating the minimum maximal independence number. *Information Processing Letters*, 46(4):169–172. DOI: 10.1016/0020-0190(93)90022-2.

Hurink, J. L. and Nieberg, T. (2008). Approximating minimum independent dominating sets in wireless networks. *Information Processing Letters*, 109(2):155–160. DOI: 10.1016/j.ipl.2008.09.021.

Van Rooij, J. M. and Bodlaender, H. L. (2011). Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164. DOI: 10.1016/j.dam.2011.07.001.

Xiao, M. and Nagamochi, H. (2017). Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146. DOI: 10.1016/j.ic.2017.06.001.

Zehavi, M. (2017). Maximum minimal vertex cover parameterized by vertex cover. *SIAM Journal on Discrete Mathematics*, 31(4):2440–2456. DOI: 10.1137/16M109017X.