Karlsruhe Institute of Technology

A Lower Bound for Minimal-to-Maximal Conversion Search

Bachelor's Thesis of

Bennet Hörmann

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

First Reviewer	T.T.-Prof. Dr. Thomas Bläsius
Second Reviewer	PD Dr. Torsten Ueckerdt
Advisor	Dr. Martin Schirneck

17 November 2025 – 17 March 2026

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Generative artificial intelligence has been used in the writing of this thesis for improving spelling, grammar, and style, and for generating code to draw graphics. Further, it was used as a tool to aid in understanding the literature, but all claims are taken directly from the literature itself. Models used were GPT-5, GPT-5.1 GPT-5.2, and GPT-5.4 by OpenAI, Gemini 2.5 Pro, 3 Pro, and 3.1 Pro by Google, and Claude Sonnet 3.7, 4, 4.5, and 4.6 by Anthropic. All new ideas presented in the thesis have been developed entirely without artificial intelligence.

Karlsruhe, 17 March 2026

—

. . . . . .

(Bennet Hörmann)

Abstract

Hitting sets are vertex sets of hypergraphs that have a nonempty intersection with every hyperedge. The problem of enumerating all (inclusion-)minimal hitting sets is known as the Transversal Hypergraph problem and has many applications, for example in data profiling, system diagnosis and computational biology. It is a long-standing question whether output-polynomial algorithms exist, i.e. algorithms whose runtime is polynomial in the combined input and output size. The most efficient algorithm in practice is arguably Minimal-to-Maximal Conversion Search (MMCS) by Murakami and Uno. While lower bounds have been proven for some other hitting set enumeration algorithms, little is known about the theoretical worst-case complexity of MMCS—despite its relevance in practice. We close that gap by providing a class of graphs for which MMCS takes exponential time to output the polynomial number of minimal hitting sets. Thereby, we prove that MMCS is not output-polynomial, not even if we restrict the inputs to graphs. To be precise, this result assumes the worst case regarding the internal decisions of MMCS. We consequently suggest introducing a heuristic that makes these decisions in a way that ensures output-polynomially on our particular class of graphs. Further, we argue that this heuristic is beneficial on general inputs. We leave the question, whether MMCS with our heuristic enhancement is output-polynomial on general inputs as an open problem. Moreover, we investigate how the existing heuristic and pruning strategy of MMCS affect its runtime. Furthermore, we prove that MMCS is output-polynomial on paths and cycles.

Zusammenfassung

Hitting Sets sind Knotenmengen von Hypergraphen, die einen nichtleeren Schnitt mit jeder Hyperkante haben. Das Problem, alle (inklusions-)minimalen Hitting Sets aufzuzählen, ist als Transversal Hypergraph-Problem bekannt und hat viele Anwendungen, bspw. in Data Profiling, Systemdiagnose und Bioinformatik. Es ist eine langjährig offene Frage, ob ausgabepolynomielle Algorithmen existieren, also Algorithmen, deren Laufzeit polynomiell in der kombinierten Eingabe- und Ausgabegröße ist. Der in der Praxis wohl effizienteste Algorithmus ist Minimal-to-Maximal Conversion Search (MMCS) von Murakami und Uno. Während für einige andere Aufzählungsalgorithmen untere Laufzeitschranken bewiesen wurden, ist wenig über die theoretische Worst-Case-Komplexität von MMCS bekannt – trotz seiner Relevanz in der Praxis. Wir schließen diese Lücke, indem wir eine Graphklasse angeben, für die MMCS exponentielle Zeit benötigt, um die polynomiell vielen minimalen Hitting Sets auszugeben. Dadurch beweisen wir, dass MMCS nicht ausgabepolynomiell ist, auch nicht, wenn wir die Eingaben auf Graphen beschränken. Um genau zu sein, geht dieses Resultat vom Worst Case bezüglich der internen Entscheidungen von MMCS aus. Folglich schlagen wir die Einführung einer Heuristik vor, die diese Entscheidungen so trifft, dass auf unserer speziellen Graphklasse Ausgabepolynomialität sichergestellt wird. Zudem argumentieren wir, dass diese Heuristik auch auf allgemeinen Eingaben von Vorteil ist. Ob MMCS mit unserer heuristischen Erweiterung auf allgemeinen Eingaben ausgabepolynomiell ist, belassen wir als offenes Problem. Außerdem untersuchen wir, wie sich die bestehende Heuristik und Pruning-Strategie von MMCS auf dessen Laufzeit auswirken. Weiterhin beweisen wir, dass MMCS auf Pfaden und Kreisen ausgabepolynomiell ist.

Contents

1 Introduction

A *hypergraph* $\mathcal{H}$ is a set of subsets of a finite vertex set V. Elements of $\mathcal{H}$ are called *hyperedges*, or simply *edges*. Hypergraphs generalize graphs by allowing more or less than two vertices per edge. A *hitting set* of a hypergraph $\mathcal{H}$ is a vertex set that has a nonempty intersection with every hyperedge of $\mathcal{H}$. Hitting sets of graphs are better known as *vertex covers*. Since any superset of a hitting set is itself a hitting set, we are usually interested only in the (inclusion-)minimal hitting sets, i.e. hitting sets whose proper subsets are all non-hitting. The set of minimal hitting sets of $\mathcal{H}$ is also known as the *transversal hypergraph* or the *dual* of $\mathcal{H}$.

In many applications, it is required to list all minimal hitting sets, which is known as the Transversal Hypergraph problem. This enumeration problem was introduced in 1987, independently by Demetrovics and Thi, Mannila and Räihä, and Reiter [DV87 | MR87 | Rei87] in the contexts of data profiling and system diagnosis. In data profiling, hitting sets are equivalent to *unique column combinations*, i.e. sets of columns whose value combinations are without duplicates. To be specific, for a given dataset, the considered hypergraph is the one, where each column appears as a vertex, and where, for each pair of rows, the set of columns in which the two rows differ, is a hyperedge. Unique column combinations can be used for query optimization, as surveyed in [KPN22]. Another application relates to monotone Boolean functions, where we consider formulas consisting of finitely many variables and the conjunction and disjunction operators, but not the unary negation operator. Here, the problem of computing the disjunctive normal form from the conjunctive normal form is equivalent to solving the Transversal Hypergraph problem, as explained in [GV17]. Moreover, the Transversal Hypergraph problem is equivalent to the problem of listing all maximal independent sets, giving rise to further applications. To be specific, a vertex set is a hitting set if and only if its complement is an independent set, which is a set that does not contain any edge. Furthermore, the Transversal Hypergraph problem is computationally equivalent to the problem of enumerating all minimal dominating sets of graphs, as shown in [KLMN14]. That means that there exist polynomial time reductions between both problems. Further applications in various domains such as system diagnosis, computational biology and data mining are surveyed in [GV17].

The time complexity of the Transversal Hypergraph problem is still unknown. Since the number of minimal hitting sets can be exponential in the input size, a polynomial algorithm cannot exist. As a result, we express the complexity of the problem in terms of the input size and output size. It remains an open question whether an output-polynomial algorithm exists, i.e. an algorithm whose runtime is polynomial in the input size plus output size. Progress has been made in showing that the problem is output-polynomial on certain classes of hypergraphs. For example, even before the problem was investigated on hypergraphs, there were already output-polynomial algorithms enumerating all minimal vertex covers of graphs, e.g., [TIAS77 | JYP88].

Further, in [EG95, Theorem 5.2], an algorithm is presented that is output-polynomial on classes of hypergraphs with bounded edge size, in particular on the class of all graphs. This is further generalized in [KBEG07], where an algorithm is presented that is output-polynomial on classes with bounded edge intersections. These are classes for which $k \in \mathbb{N}$ and $r \in \mathbb{N}_0$ exist, so that the intersection of k arbitrary edges has cardinality at most r. In particular this includes classes with bounded edge size and also classes with bounded degree. For multiple other classes of hypergraphs, output-polynomial algorithms have been presented in the literature as well. Recently, many of these results have been generalized in a preprint by Mary [Mar26], where an algorithm is presented that is output-polynomial on all classes of hypergraphs with bounded VC-dimension.

Many algorithms for the Transversal Hypergraph problem and equivalent problems have been proposed. The one with the lowest asymptotic bound $N^{o(\log N)}$ is presented in [FK96], where N is the combined input and output size. In practice, the fastest algorithm on most instances is *Minimal-to-Maximal Conversion Search (MMCS)* by Murakami and Uno [MU14]. Its efficiency on real-world instances has been shown in [GV17]. As a result, MMCS is used in practical settings. For example, in [Bir+20], the authors propose a specialized algorithm for the discovery of unique column combinations, that uses MMCS under the hood. A similar approach, also based on MMCS, is proposed in [Ble+24] for discovering functional dependencies of datasets.

For some other hitting set enumeration algorithms, lower bounds have been shown, implying that they are not output-polynomial [Tak08 | Hag09]. Despite the practical relevance of MMCS, no similar results exist for MMCS. Only the runtime of a single step of MMCS is known to be polynomial in the input size. We close that gap by providing a class of graphs for which MMCS needs exponential time to list the polynomial number of minimal hitting sets. In that way, we prove that MMCS is not output-polynomial, not even if we restrict the inputs to graphs instead of general hypergraphs. This result considers the worst case regarding the internal branching decisions of MMCS. We also provide a class of hypergraphs, very similar to the mentioned class of graphs, on which this worst-case assumption is not necessary to prove exponential runtime. Both results, however, depend on MMCS choosing a particular ordering of vertices in every step. Since the ordering of vertices is not specified in MMCS, we propose to augment MMCS with a certain heuristic for ordering vertices. We show that this makes MMCS output-polynomial on the mentioned classes, but whether it becomes output-polynomial on general inputs is left as an open problem. Further, we argue that the proposed heuristic for ordering vertices is useful in general, by proving that it minimizes the number of potential branching points among all direct child nodes in every step.

Besides, we show that the choice of the heuristic which MMCS uses to select an edge to branch on can have a large impact on its runtime. Specifically, the default heuristic can lead to an exponential increase or an exponential decrease in runtime compared to some other heuristic, depending on the input hypergraph. Additionally, we show that disabling the so-called violator pruning strategy can increase the runtime of MMCS from polynomial to exponential on certain inputs.

In the presence of an exponential lower bound to the runtime on general inputs, it becomes an interesting question on which classes of hypergraphs MMCS is output-polynomial. We show that this is the case on paths and cycles by proving that they have exponentially many minimal vertex covers. As secondary results, we include two observations about MMCS and minimality that, to our knowledge, have not been published before. Namely, that the default heuristic prevents a certain kind of dead end in the search tree and that MMCS can behave differently on a hypergraph and its minimization.

2 Preliminaries

We start by repeating the definitions from the introduction and introducing some notation that we use throughout this thesis.

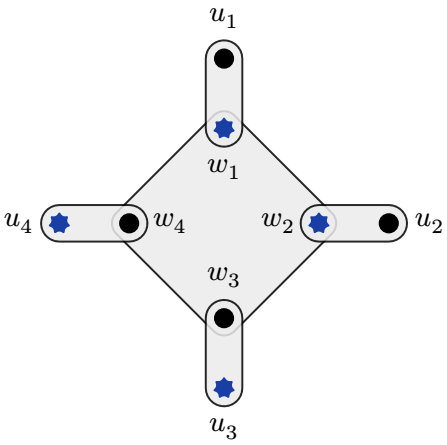
Notation 2.1 We write $S \sqcup T$ for the disjoint union of two sets S and T.

Definition 2.2 A *hypergraph* is a pair $(V, \mathcal{H})$, where V is a finite set of vertices and $\mathcal{H}$ is a set of subsets of V. The elements of $\mathcal{H}$ are called hyperedges or simply edges. When the vertex set is clear from the context, we often denote a hypergraph just by the edge set $\mathcal{H}$.

Definition 2.3 We say a vertex set $T \subseteq V$ is a *hitting set* if and only if $T \cap E \neq \emptyset$ holds for all edges $E \in \mathcal{H}$. Further, we call a hitting set T *(inclusion-)minimal* if and only if there is no proper subset $T' \subsetneq T$ that also is a hitting set.

We define the *transversal hypergraph* $\mathrm{Tr}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ as the set of all its inclusion-minimal hitting sets.

As an example, let us consider the following hypergraph.



$$\mathcal{H} = \{\{w_1, w_2, w_3, w_4\}, \{w_1, u_1\}, \{w_2, u_2\}, \{w_3, u_3\}, \{w_4, u_4\}\}$$

Here, the highlighted vertices form the minimal hitting set $\{w_1, w_2, u_3, u_4\}$. All proper supersets of it are non-minimal hitting sets and all proper subsets are non-hitting.

An interesting property of the transversal hypergraph is that the transversal of the transversal consists exactly of the inclusion-minimal edges of $\mathcal{H}$. A proof can be found in the standard textbook by Berge from 1984 [Ber84, Chapter 2]. For that reason, the transversal hypergraph is also known as the *dual*.

Interestingly, the Transversal Hypergraph problem is equivalent to its corresponding recognition problem. More specifically, one can enumerate the transversal hypergraph in output-polynomial time if and only if one can solve the following decision problem in polynomial time. Given two hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$, decide if $\mathcal{H}_1$ is the transversal of $\mathcal{H}_2$, i.e. if $\mathcal{H}_1 = \mathrm{Tr}(\mathcal{H}_2)$. This is proven in [BI95]. Other equivalent problems are enumerating all maximal independent sets of hypergraphs and enumerating all minimal dominating sets of graphs [KLMN14], as described in the introduction.

2.1 MMCS

Minimal-to-Maximal Conversion Search (MMCS) is an algorithm that enumerates all minimal hitting sets of a hypergraph $(V, \mathcal{H})$. In other words, it lists all the edges of the transversal hypergraph $\mathrm{Tr}(\mathcal{H})$. It was first described by Murakami and Uno in the conference paper preceding [MU14].

It generates a search tree, where every node consists of a partial solution $S \subseteq V$ and a *candidate set* $C \subseteq V \setminus S$. MMCS generates the search tree in such a way that the subtree under a node (S, C) includes all minimal hitting sets that contain the partial solution S and consist only of vertices from $S \sqcup C$. In particular, the entire search tree under the root node $(\emptyset, V)$ includes all minimal hitting sets.

To understand MMCS, we need to understand how the children of a given node are generated, i.e. we need to understand the recursive step of MMCS. For that, we first need a few concepts and some notation.

Definition 2.4 *Unhit Edges*
For a vertex set $S \subseteq V$, we define $\mathrm{unhit}(S) := \{E \in \mathcal{H} \mid E \cap S = \emptyset\}$ as the set of edges that are not hit by S.

Clearly, S is a hitting set if and only if $\mathrm{unhit}(S) = \emptyset$. Besides that, MMCS relies heavily on the concept of *irredundancy*.

Definition 2.5 *Irredundancy and Private Edges*
For a vertex set $S \subseteq V$ and a vertex $v \in S$, we define the set of private edges $\mathrm{priv}(S, v) := \{E \in \mathcal{H} \mid E \cap S = \{v\}\}$. This is the set of edges that are only hit by the vertex v out of S.

If every vertex of a set S has at least one private edge, i.e. $\mathrm{priv}(S, v) \neq \emptyset$ for all $v \in S$, we call S *irredundant*. Otherwise, we call it *redundant*.

A central idea behind MMCS is to restrict the search to irredundant partial solutions. To see why we can still find all minimal hitting sets, we need the following two lemmas about irredundancy.

Lemma 2.6 *Characterisation of Minimal Hitting Sets*
$S \subseteq V$ is a minimal hitting set if and only if it is an irredundant hitting set.

Proof. A set $S \subseteq V$ is irredundant if and only if leaving out any vertex causes an edge that was previously hit to become unhit. This is because when leaving out a vertex v, precisely the edges $\operatorname{priv}(S, v)$ become unhit. In other words, S is irredundant if and only if it is minimal among the sets hitting the same edges as S. Hence, for hitting sets, irredundancy coincides with minimality. This is a well-known result. ∎

Lemma 2.7 If a set $S \subseteq V$ is redundant, then any superset $\tilde{S} \supseteq S$ is also redundant.

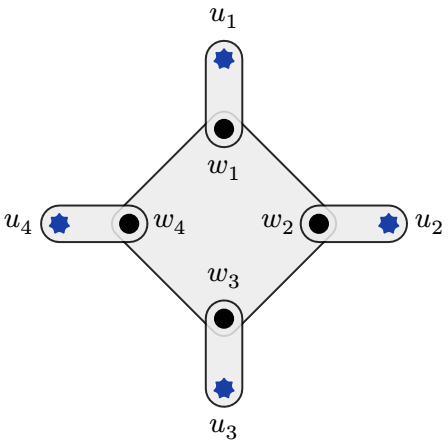
Proof. We show that, when adding a single vertex u to a set S, then for all $v \in S$, the set of their private edges does not grow. More precisely, for all $u \notin S$ and all $v \in S$, it holds that $\operatorname{priv}(S \cup \{u\}, v) = \operatorname{priv}(S, v) \setminus \{E \in \mathcal{H} \mid u \in E\}$, cf. [MU14, Lemma 2.1]. This is derived in the following.

$$E \in \operatorname{priv}(S \cup \{u\}, v) \iff \underbrace{E \cap (S \cup \{u\})}_{=(E \cap S) \cup (E \cap \{u\})} = \{v\} \iff \underbrace{E \cap S = \{v\}}_{\iff E \in \operatorname{priv}(S, v)} \text{ and } u \notin E$$

Consequently, any vertex that makes S redundant, i.e. has no private edge with respect to S, will not have a private edge with respect to $S \cup \{u\}$ either. That implies that $S \cup \{u\}$ is redundant too. By induction, this result transfers to any superset of S. ∎

These two lemmas allow MMCS to prune all redundant partial solutions, since any superset of a redundant set cannot be a minimal hitting set. This is because, all minimal hitting sets are irredundant by Lemma 2.6 and once a partial solution is redundant, any superset of it will remain redundant by Lemma 2.7. Recall here that the descendants of a partial solution are all supersets of it. So essentially, by adding more and more vertices, MMCS generates all maximal irredundant sets. That includes all minimal hitting sets, hence the name Minimal-to-Maximal Conversion Search[1].

As an example, let us consider the hypergraph from before again.

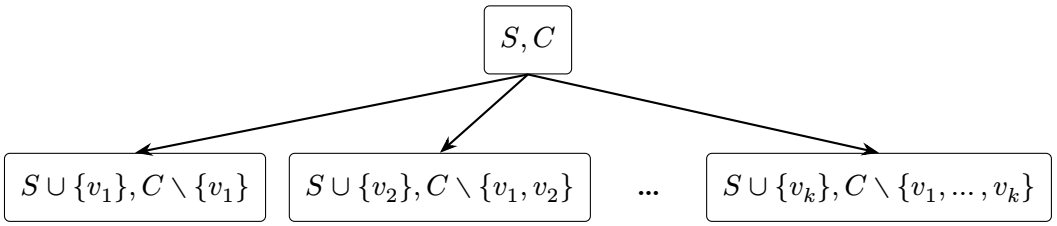


$$\mathcal{H} = \{\{w_1, w_2, w_3, w_4\}, \{w_1, u_1\}, \{w_2, u_2\}, \{w_3, u_3\}, \{w_4, u_4\}\}$$

[1]The meaning of the name is not mentioned in the original paper by Murakami and Uno [MU14].

Here, the selected vertices form the set $S = \{u_1, u_2, u_3, u_4\}$, which is irredundant, since every $u_i \in S$ has a private edge $\{w_i, u_i\}$. In fact, it is a maximal irredundant set, since adding any vertex w_i causes the vertex u_i to lose its private edge. But S is not a hitting set since the edge $\{w_1, w_2, w_3, w_4\}$ remains unhit. This demonstrates that, while all minimal hitting sets are maximally irredundant, the reverse is not true.

Now, we can describe the recursive step of MMCS, which determines how MMCS recursively generates the search tree. To generate the direct children of a node (S, C), MMCS checks if there is an edge $E \in \text{unhit}(S)$. If not, then S is a hitting set. Moreover, S even is a *minimal* hitting set, since MMCS only generates irredundant nodes. Therefore, MMCS outputs S in that case and does not generate any children. If, on the other hand, there are unhit edges, then MMCS selects one edge $E \in \text{unhit}(S)$ according to a heuristic, which is described later. In principle, any edge $E \in \text{unhit}(S)$ would be a valid choice. Then MMCS generates child nodes for the partial solutions $S \cup \{v_1\}, \dots, S \cup \{v_k\}$, where $\{v_1, \dots, v_k\} = E \cap C$. These are all the possibilities to hit the selected edge E with the available candidates. To prevent duplicates in the enumeration, the candidate sets for the child nodes are chosen so that $S \cup \{v_j\}$ has the candidate set $C \setminus \{v_1, \dots, v_j\}$. This is visualized in the following figure.



A visualization of child node generation in MMCS

As described earlier, MMCS checks if a child partial solution is redundant and in that case does not further process it, causing redundant child nodes to be leaves in MMCS' search tree. Optionally, MMCS can prune so-called *violating vertices* from the candidate sets of all child nodes, which is described later. This pruning technique is enabled by default. Finally, MMCS continues recursively with every generated irredundant child node.

For details on how to efficiently implement MMCS, in particular the redundancy check, see the original paper [MU14]. They achieve polynomial runtime for a single step of MMCS, cf. [MU14, Theorem 3.1]. As a consequence, MMCS is output-polynomial on a certain class of hypergraphs if and only if the generated search tree has polynomial size on that class.

Note that MMCS leaves some decisions to the specific implementation. It does not specify how to order the vertices $\{v_1, \dots v_k\} = E \cap C$ for the child node creation, which has an effect on the child nodes' candidate sets. Later, in Chapter 4, we will see that these decisions can make a significant runtime difference.

Edge Selection Heuristics

As described above, MMCS selects one edge $E \in \text{unhit}(S)$ in every recursive step according to a heuristic. Regarding the correctness of the algorithm, any unhit edge may be selected. But as we will see in Chapter 4, different heuristics can lead to drastically different runtimes for certain inputs. The default heuristic for selecting an edge is the *min-heuristic*.

Definition 2.8 The *min-heuristic* chooses an edge $E \in \text{unhit}(S)$, for which $|E \cap C|$ is minimum, where C is the current candidate set.

Note that if multiple edges satisfy this condition, the min-heuristic does not specify which of them to select. This is up to the implementation.

The intuition behind the min-heuristic is that it minimizes the number of child nodes that are created for the current node, since one child node is created for every $v \in E \cap C$. An edge case in which the benefits of that become particularly clear, is the case, where there is an unhit edge E with $|E \cap C| = 1$. In that case, MMCS will create only a single child node by adding the single vertex $v \in E \cap C$ to the partial solution. Intuitively this makes sense, since v is the only available vertex that hits the edge E, and thus has to be added to the partial solution sooner or later.

Violator Pruning

As mentioned earlier, MMCS can optionally prune *violating vertices* from the candidate sets of all child nodes, before it continues recursively with all child nodes.

Lemma and Definition 2.9 *Violating Vertices*
If $S \cup \{v\}$ is redundant and $\tilde{S}$ is a (proper) superset of S, then $\tilde{S} \cup \{v\}$ is also redundant. Similarly, if $S \cup \{v\}$ is a minimal hitting set and $\tilde{S}$ a proper superset of S, then $\tilde{S} \cup \{v\}$ is redundant.

In both cases, we say that v is *(S-)violating*.

When violator pruning is enabled, MMCS prunes all vertices of $E \cap C$ that are violating from all candidate sets, where E is the selected edge and C the current candidate set. Note that for every $v \in E \cap C$, MMCS checks if $S \cup \{v\}$ is redundant or hitting anyway, so enabling violator pruning does not require any additional computations. In theory, one could extend violator checking and pruning to all candidate vertices, but that would require additional computation, so we restrict violator pruning to the vertices of $E \cap C$. To see why pruning violating vertices is justified, we consider a partial solution S and some partial solution $\tilde{S} \supseteq S$ in the subtree under S. If we then added an S-violating vertex v to $\tilde{S}$, then $\tilde{S} \cup \{v\}$ would be redundant by the definition of violating vertices, cf. Lemma and Definition 2.9. Therefore, any S-violating vertex can be safely pruned from the candidate sets of all child nodes of S.

In the explanation above, it seems that violator pruning merely avoids the redundancy check for $\tilde{S} \cup \{v\}$, but it has a larger impact, due to how it interacts with the selection of edges. In particular, when using the min-heuristic the reduced candidate sets can

affect which edge is selected. Intuitively, violator pruning excludes vertices from the candidate sets, from which we already know that their addition will lead directly to a leaf node. Recall that the intuition behind the min-heuristic was to minimize the number of branches at the current node. But the min-heuristic does not distinguish between branches that directly lead to leaf nodes and branches that go on longer. Intuitively, we should minimize the latter, since branches leading directly to a leaf node are processed very quickly. By enabling violator pruning, we take some of the branches that lead directly to leaf nodes out of the calculation—the branches corresponding to the addition of a vertex that has been identified as a violator. Intuitively this leads to the min-heuristic minimizing the number of branches from which we currently have to expect that they go on for longer than just a single leaf node. In section Section 4.2, we see a class of hypergraphs where enabling violator pruning along with the min-heuristic reduces the runtime significantly. This is precisely because violator pruning removes many vertices from the candidate sets, causing the min-heuristic to "focus" on the vertices that actually matter. This results in the selection of the edge that easily resolves the problem over one of the edges that inflates the search tree.

Violator pruning is not required for the correctness of MMCS, but we hope that it improves the runtime, as argued above. It is enabled by default.

2.2 Some Results about MMCS and Minimality

In this section, we present two observations about MMCS that, to our knowledge, have not been published before.

A Consequence of the Min-Heuristic

In general, MMCS can encounter a partial solution where there is still an unhit edge but no way to hit it with the available candidates. While the original MMCS paper [MU14] does not explicitly mention this possibility, its occurrence is not an issue for the correctness of MMCS. If MMCS encounters such a node, it will at some point select an edge that has an empty intersection with the current candidate set. At that point, it will implicitly abort the branch, since there are no vertices to add to the partial solution. That said, if we use the min-heuristic, which is the default, then this situation cannot occur anyway. We show this in the following lemma.

Lemma 2.10 If MMCS always selects an unhit edge E for which $E \cap C$ is inclusion-minimal, i.e. there is no unhit edge F with $F \cap C \subsetneq E \cap C$, then the candidate sets of all nodes of the MMCS tree have a nonempty intersection with every edge that is unhit at that node. As always, C stands for the current candidate set here. This result holds regardless of whether violator pruning is enabled.

Proof. The proof idea is by Max Göttlicher via personal communication. It goes by induction over the MMCS tree.

Base case. The candidate set of the root node contains all vertices and therefore has nonempty intersection with every edge.

Induction step. Let (S, C) be the current node and $E \in \text{unhit}(S)$ the edge selected for branching. We assume for contradiction that the child node $(S \cup \{v\}, C \setminus C_0)$ has an unhit edge $F \in \text{unhit}(S \cup \{v\})$ with $F \cap (C \setminus C_0) = \emptyset$. In other words, the set C_0 of vertices that are removed from the candidate set contains $F \cap C$ entirely. At the same time, C_0 is always a subset of $E \cap C$ by the definition of MMCS. We can conclude $F \cap C \subseteq C_0 \subseteq E \cap C$. Since $F \in \text{unhit}(S \cup \{v\})$, we know that $v \notin F$, while $v \in E \cap C$ holds by the definition of MMCS. Therefore, $F \cap C$ is a *proper* subset of $E \cap C$, contradicting the assumption that E was selected for branching. ∎

Note that if we use the min-heuristic, i.e. if we always select an edge whose intersection with the candidate set has minimum cardinality, then that intersection is in particular inclusion-minimal. We have therefore shown that the use of the min-heuristic prevents MMCS from encountering this particular kind of dead end in its search.

MMCS and Hypergraph Minimization

It is well known that the transversal hypergraph only depends on the inclusion-minimal edges. More specifically, with $\min(\mathcal{H}) := \{E \in \mathcal{H} \mid \nexists E' \in \mathcal{H} \text{ s.t. } E' \subsetneq E\}$, it holds that $\text{Tr}(\mathcal{H}) = \text{Tr}(\min(\mathcal{H}))$. This is because a set $S \subseteq V(\mathcal{H})$ that hits all inclusion-minimal edges, automatically hits all their supersets too. In the other direction, if a set S hits all edges of $\mathcal{H}$, then that includes, in particular, all edges of $\min(\mathcal{H})$. While the transversal hypergraph is only affected by inclusion-minimal edges, MMCS can behave differently on the inputs $\mathcal{H}$ and $\min(\mathcal{H})$. However, if we use the min-heuristic, this is only due to redundancy checks, which we show in the following.

The key observation is that irredundancy in $\mathcal{H}$ is not the same as irredundancy in $\min(\mathcal{H})$. Sets that are irredundant in $\min(\mathcal{H})$ are also irredundant in $\mathcal{H}$. This is because, if a set $S \subseteq V$ is irredundant in $\min(\mathcal{H})$, then every vertex $v \in S$ has a private edge $E_v \in \min(\mathcal{H})$. This private edge E_v is also part of $\mathcal{H}$, implying the irredundancy of S in $\mathcal{H}$. The other direction does not hold in general, as we can see in the following hypergraph.

$$\begin{aligned}\mathcal{H} &= P_6 \sqcup \{\{v_1, v_2, v_4\}\} \\ &= \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}\} \sqcup \{\{v_1, v_2, v_4\}\}\end{aligned}$$

Here, the set $S = \{v_3, v_4, v_5\}$ is irredundant in $\mathcal{H}$, since v_4 has the private edge $\{v_1, v_2, v_4\}$, v_3 has $\{v_2, v_3\}$ as a private edge, and v_5 has $\{v_5, v_6\}$ as a private edge. But S is redundant in the path $P_6 = \min(\mathcal{H})$, since the only private edge that v_4 has in $\mathcal{H}$ is not part of $\min(\mathcal{H})$.

MMCS can therefore behave differently on the inputs $\mathcal{H}$ and $\min(\mathcal{H})$. On the input $\mathcal{H}$, it might process the partial solution S, while on $\min(\mathcal{H})$, S is pruned, because it is redundant in $\min(\mathcal{H})$. This, however, is the only reason why MMCS might behave differently as long as we use the min-heuristic, as the following lemma states.

Lemma 2.11 With the min-heuristic, the MMCS tree for the input $\mathcal{H}$ is the same as the MMCS tree for the input $\min(\mathcal{H})$, if MMCS never considers partial solutions that are irredundant in $\mathcal{H}$, but redundant in $\min(\mathcal{H})$.

For this lemma to hold, we need to assume the following things about the implementation of MMCS. Firstly we assume that the chosen order of vertices in $E \cap C$ does not depend on E, but only on the set $E \cap C$. Secondly we assume that the selection of an edge among multiple edges $E_1, \dots, E_k$ with $|E_1 \cap C| = \cdots = |E_k \cap C|$ only depends on the intersections $E_1 \cap C, \dots, E_k \cap C$ and not on the edges themselves.

Proof. We have to show that all the decisions that MMCS makes, are the same for the input $\mathcal{H}$ and $\min(\mathcal{H})$. By assumption, redundancy checks do not depend on whether the input is $\mathcal{H}$ or $\min(\mathcal{H})$.

Also, the selection of vertices to branch on is the same in both cases. The min-heuristic selects an edge E for which $|E \cap C|$ is minimum, where C is the current candidate set. Let $E_1, \dots, E_p$ be the edges of $\min(\mathcal{H})$ that have this property. If there is an edge $\tilde{E}_i \in \mathcal{H}$ with $\tilde{E}_i \supsetneq E_i$ that also satisfies that property, we can conclude that $\tilde{E}_i \cap C \supseteq E_i \cap C$. But since both these intersections have the same cardinality, we can conclude $\tilde{E}_i \cap C = E_i \cap C$. Therefore, MMCS on $\mathcal{H}$ will select an inclusion-minimal edge E_j for some j or a superset $\tilde{E}_j$ of it. Since the branching happens on the vertices $E_j \cap C = \tilde{E}_j \cap C$, both edge selections result in the same child partial solutions.

Finally, by assumption, the order of the vertices in $E_j \cap C = \tilde{E}_j \cap C$ does not depend on whether E_j or $\tilde{E}_j$ was selected, resulting in the same candidate sets in both cases. Recall here that the order of vertices determines the candidate sets of the child nodes. ∎

Having introduced all required concepts and notation, and having made two interesting new observations about MMCS, we are now ready for the main chapters of this thesis.

3 Output Polynomiality on Certain Classes of Hypergraphs

Before we investigate lower bounds for MMCS, we first spend a chapter looking at certain classes of hypergraphs on which MMCS is output-polynomial. More specifically, we look at paths and cycles. Both are classes of *graphs*, i.e. hypergraphs, where every edge consists of exactly two vertices. On graphs, (minimal) hitting sets are better known as (minimal) *vertex covers*. The output-polynomiality in both cases relies on the following lemma.

Lemma 3.1 MMCS is output-polynomial on all classes of hypergraphs that have exponentially many minimal hitting sets, regardless of which heuristic is used and whether violator pruning is enabled. More precisely, if for a class of hypergraphs, $|\mathrm{Tr}(\mathcal{H})| \geq Ca^{|V|+|\mathcal{H}|}$ holds for all sufficiently large $\mathcal{H}$ in the class with constants $C > 0$ and $a > 1$, then there is a polynomial in the input size plus output size that bounds the runtime of MMCS from above.

Proof. Let us denote the number of vertices by n and the number of edges by m. First, we observe that MMCS takes at most exponential time, regardless of which heuristic is used and whether violator pruning is enabled. This is because the MMCS tree can contain at most 2^n nodes, one for every possible subset of $V(\mathcal{H})$. Note that MMCS avoids duplicate nodes by using candidate sets. Since the runtime per node is polynomial in n and m, we can bound it by $\mathcal{O}(2^{n+m})$ to conclude that the total runtime of MMCS is in $\mathcal{O}(4^{n+m})$.

Note that exponentials with different basis can be written as polynomials of each other. Therefore, the total runtime of MMCS grows at most polynomially in Ca^{n+m}, which bounds the number of minimal hitting sets from below. More rigorously, we can rewrite $4^{n+m} = (a^{n+m})^{\log_a 4}$ which is a polynomial in the term a^{n+m}. ∎

3.1 Paths

The goal of this section is to show the following theorem.

Theorem 3.2 MMCS is output-polynomial on paths, regardless of which heuristic is used and whether violator pruning is enabled.

Definition 3.3 We denote the *path on n vertices* and $m = n - 1$ edges with P_n and its vertices with $V(P_n) = \{v_1, ..., v_n\}$.

To prove this theorem, we use Lemma 3.1. For that, we need to find exponentially many minimal vertex covers of paths. To be able to explicitly construct these, we first characterize minimal vertex covers of paths. For that, it is helpful to view vertex sets of P_n as a sequence of n selected and unselected vertices. For example, ●○● corresponds to the vertex set $\{v_1, v_3\} \subseteq V(P_3)$. We use this notation to formulate the following two lemmas.

Lemma 3.4 A set $S \subseteq V(P_n)$ is a vertex cover if and only if there are no two consecutive vertices absent from S, that is, there is no subsequence ○○ in S.

Proof. The statement follows directly from the fact that S hits an edge if and only if it selects one of its two vertices. ■

Lemma 3.5 A set $S \subseteq V(P_n)$ is irredundant if and only if there are no three consecutive vertices included in S and neither $\{v_1, v_2\} \subseteq S$ nor $\{v_{n-1}, v_n\} \subseteq S$. In other words, S is irredundant if and only if it does not contain the subsequence ●●●, does not start on ●●, and does not end on ●●.

Proof. Recall from Definition 2.5 that S is redundant if and only if it contains a vertex that has no private edge, i.e. a vertex whose neighbors are all contained in S too. Hence, S is redundant if and only if there are three consecutive vertices selected or if the first two vertices or the last two vertices are selected. ■

Since minimal vertex covers are the same as irredundant vertex covers by Lemma 2.6, these two lemmas provide a characterization of minimal vertex covers of paths. With this characterization, we can now explicitly construct exponentially many minimal vertex covers in the following theorem.

Theorem 3.6 Paths have exponentially many minimal vertex covers, specifically

$$|\mathrm{Tr}(P_n)| \geq \tfrac{1}{2}\left(\sqrt[6]{2}\right)^n \geq \tfrac{1}{2}\left(\sqrt[12]{2}\right)^{n+m}.$$

Proof. We consider sequences starting on ○, followed by a concatenation of the *building blocks* ●●○●●○ and ●●○●○●○ in arbitrary order and of arbitrary length. That means we consider sequences of the following form.

○ *building block* ○ *building block* ○ ⋯ ○ *building block*

With Lemma 3.5, we can see that the constructed sequences are irredundant. The building blocks do not contain ●●● and the concatenation cannot generate any new ●●●, since both building blocks end on ○. For the same reason, the sequences cannot end on ●●, and because of the starting segment ○, the sequences can not start on ●● either, allowing us to conclude their irredundancy.

With Lemma 3.4, we can see that the constructed sequences are vertex covers and therefore even minimal vertex covers, by Lemma 2.6. The building blocks do not contain ○–○ and since they both start with ●–, the concatenation cannot generate any new ○–○. For the same reason the additional start on ○– does not generate any new ○–○.

For every $k \in \mathbb{N}$, this construction provides us with 2^k sequences of length $6k+1$ which prove $|\mathrm{Tr}(P_{6k+1})| \geq 2^k$.

To get the same bound also for paths on $6k+i$ vertices for $i \in \{2, \dots, 6\}$, we can simply extend our constructed minimal vertex covers by $1, 2, \dots, 5$ vertices. If a minimal vertex cover starts on ○–, as all our constructed sequences do, we can just concatenate an additional ●– to the start. The sequence, that is now one vertex longer, is still a minimal vertex cover, since the added selected vertex hits the added edge, which would be otherwise be unhit. To further extend our constructed minimal vertex covers, we need to define how to extend minimal vertex covers that start on ●–. In that case, we can concatenate an additional ○– to the start. The resulting sequence is still a minimal vertex cover, since the added edge is already hit by the selected vertices. By alternately concatenating ○– and ●– to the start, we can therefore construct 2^k minimal vertex covers of $P_{6k+2}, P_{6k+3}, \dots, P_{6k+6}$. For example, for P_{6k+6}, we get 2^k minimal vertex covers of the following form.

●–○–●–○–●–○– *building block* – *building block* – ⋯ – *building block*

We can conclude that $|\mathrm{Tr}(P_{6k+i})| \geq 2^k$ for $i \in \{1, 2, \dots, 6\}$. Using[1] $\lfloor \frac{n-1}{6} \rfloor \geq \frac{n}{6} - 1$, we can finally conclude

$$|\mathrm{Tr}(P_n)| \geq 2^{\lfloor \frac{n-1}{6} \rfloor} \geq 2^{\frac{n}{6}-1} = \tfrac{1}{2}\left(\sqrt[6]{2}\right)^n.$$

■

This lower bound is not tight, but sufficient for us. To count the number of minimal vertex covers exactly, one could consider all concatenations of the building blocks ●–○– and ●–●–○–. However, their different lengths make counting more tedious.

3.2 Cycles

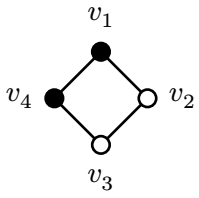
Having shown that MMCS is output-polynomial on paths, we now show the same result for cycles.

Theorem 3.7 MMCS is output-polynomial on cycles, regardless of which heuristic is used and whether violator pruning is enabled.

Definition 3.8 We denote the *cycle on n vertices* and $m = n$ edges with C_n and its vertices with $V(C_n) = \{v_1, \dots, v_n\}$.

[1] This inequality can be seen by writing $n - 1 = 6\ell + j$ for some $\ell, j \in \mathbb{N}_0$ with $j < 6$.

Again, we rely on Lemma 3.1 to deduce output-polynomiality from an exponential number of minimal vertex covers. For that, we use the fact that paths have exponentially many minimal vertex covers to construct exponentially many minimal vertex covers of cycles. As earlier, we first characterize minimal vertex covers of cycles before going into the construction. For that, we again interpret subsets $S \subseteq V(C_n)$ as sequences of selected and unselected vertices, except that the sequences are now cyclic. The following example is the cyclic sequence representing the subset $\{v_1, v_4\} \subseteq V(C_4)$. The top-most vertex always corresponds to the vertex v_1, and we go around clockwise.



We use this notation to formulate the following two lemmas.

Lemma 3.9 A set $S \subseteq V(C_n)$ is a vertex cover if and only if there are no two consecutive vertices absent from S, that is, there is no subsequence ○○ in S.

Proof. As with Lemma 3.4, the statement follows directly from the fact that S hits an edge if and only if it selects one of its two vertices. ∎

Lemma 3.10 A set $S \subseteq V(C_n)$ is irredundant if and only if there are no three consecutive vertices included in S, i.e., there is no subsequence ●●● in S.

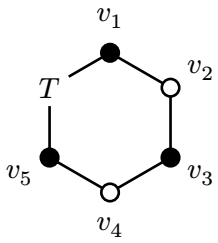
Proof. Analogously to the proof of Lemma 3.5, this statement follows from the fact that a vertex in S has no private edge if and only if both its neighbors are contained in S. ∎

Note that Lemma 3.9 and Lemma 3.10 are very similar to the respective lemmas that we had for paths, namely Lemma 3.4 and Lemma 3.5. However, for cycles it is important to read the sequences as cyclic. For example, we consider ●●● a subsequence of the sequence corresponding to the vertex set $\{v_1, v_2, v_4, v_6\} \subseteq C_6$. Here, the three consecutive vertices making the set redundant are v_6, v_1 and v_2. Having understood vertex covers and irredundancy in cycles, we can now construct exponentially many minimal vertex covers of cycles in the following theorem.

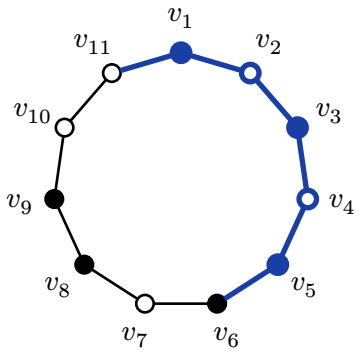
Theorem 3.11 Cycles have exponentially many minimal vertex covers, specifically

$$|\mathrm{Tr}(C_n)| \geq \left(\tfrac{1}{2}\right)^{11/6} \left(\sqrt[6]{2}\right)^n = \left(\tfrac{1}{2}\right)^{11/6} \left(\sqrt[12]{2}\right)^{n+m}.$$

Proof. We construct exponentially many minimal vertex covers of C_n from the minimal vertex covers of P_{n-5}, where $n \geq 5$. For that, we consider the following construction. For each $T \in \mathrm{Tr}(P_{n-5})$, we construct a unique set $\hat{T} \in \mathrm{Tr}(C_n)$ in the following way.



We call the inserted sequence $N =$ ●-○-●-○-●-. For example, we get the following $\hat{T}$, when T is ●-○-●-●-○-○-. The inserted sequence N as well as the edges connecting to T are highlighted.



We show that for every $T \in \mathrm{Tr}(P_{n-5})$, the constructed $\hat{T}$ is a minimal vertex cover of C_n. Because then, since the map $T \mapsto \hat{T}$ is injective, we can conclude that

$$|\mathrm{Tr}(C_n)| \geq |\mathrm{Tr}(P_{n-5})| \geq \tfrac{1}{2}\left(\sqrt[6]{2}\right)^{n-5} = \underbrace{\tfrac{1}{2}\left(\sqrt[6]{2}\right)^{-5}}_{=2^{-11/6}}\left(\sqrt[6]{2}\right)^{n}.$$

We can see that $\hat{T}$ is a vertex cover. The selected vertices in N hit all edges in the inserted pattern including the two edges connecting to T. Since T hits the remaining edges by assumption, all edges of C_n are hit.

Similarly, we can show that $\hat{T}$ is irredundant by contraposition. If $\hat{T}$ is redundant, i.e., contains the subsequence ●-●-●-, we consider two cases.

(1) If that subsequence uses no vertices of the inserted sequence N, then ●-●-●- must be a subsequence of T, implying the redundancy of T.

(2) In the other case, we know that one of the two ends of the inserted sequence $N =$ ●-○-●-○-● must be used, since the middle one is neighbored by -○- on both sides. Therefore, T must start with ●-● or end on -●-●, which implies that T is redundant.

■

In conclusion of this chapter, we have seen that MMCS is output-polynomial on cycles and paths. The central argument was that both of these classes of graphs have exponentially many minimal vertex covers, allowing MMCS exponential runtime while

still being output-polynomial on these classes. Having shown upper bounds for certain classes of inputs, we now move on to showing lower bounds for MMCS in the next chapter.

4 Lower Bounds for MMCS

In this chapter we prove lower bounds for different configurations of MMCS. We achieve this by constructing hypergraphs for which MMCS needs exponential runtime, while the number of minimal hitting sets, i.e. the output size, is at most polynomial. Note that in that case output-polynomiality and (input-)polynomiality are the same. Thereby we see what effects different heuristics and violator pruning can have on the runtime of MMCS. But most importantly, we prove that MMCS is not output-polynomial in its default configuration, i.e. with the min-heuristic and enabled violator pruning. Since this result relies on assuming the worst case regarding the ordering of vertices, we propose to use a certain heuristic to determine the ordering of vertices, which is otherwise not specified by MMCS. With that particular heuristic in place, MMCS now only needs polynomial time for the particular class of hypergraphs for which it needed exponential runtime before.

4.1 MMCS with a Non-Min-Heuristic

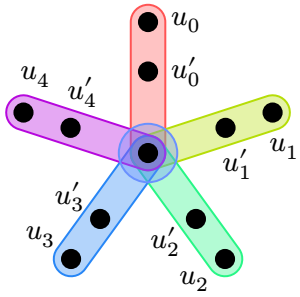
In this section, we consider MMCS with a heuristic that always selects an unhit edge E for which $|E \cap C|$ is not minimum, as long as there is such an edge. We call such a heuristic a *non-min-heuristic*, since it decides differently than the default min-heuristic, whenever possible. In the following, we construct a particular class of hypergraphs on which MMCS with the min-heuristic is output-polynomial, while it is not output-polynomial with a non-min-heuristic. We prove the following theorem.

Theorem 4.1 MMCS with a non-min-heuristic, with or without violator pruning, is not output-polynomial.

To prove this, we construct a class of hypergraphs for which the MMCS tree grows exponentially, while all hypergraphs have just one minimal hitting set and polynomially many edges and vertices. Consequently, the size of the MMCS tree is not output-polynomial, implying that the runtime cannot be output-polynomial either. In the rest of this section, we will consider the following class of hypergraphs, parameterized by a positive integer ℓ.

$$\begin{aligned} \mathcal{H}_\ell &:= \{\{w, u_i, u'_i\} \mid i \in \{0, ..., \ell - 1\}\} \cup \{\{w\}\} \\ V(\mathcal{H}_\ell) &:= \{w, u_0, \dots, u_{\ell-1}, u'_0, \dots u'_{\ell-1}\} \end{aligned}$$

The following is a visualization of $\mathcal{H}_5$, where w is the vertex in the center.



Visualization of $\mathcal{H}_5$

First, we investigate the number of minimal hitting sets of $\mathcal{H}_\ell$.

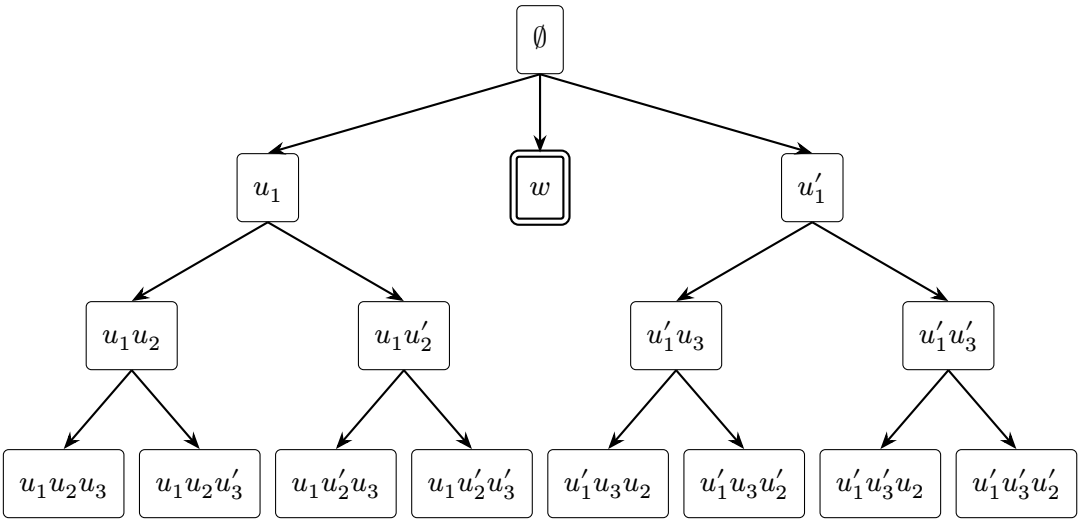
Lemma 4.2 $\mathcal{H}_\ell$ has exactly one minimal hitting set, namely $\{w\}$.

Proof. Any minimal hitting set of $\mathcal{H}_\ell$ must contain w, since this is the only vertex hitting the edge $\{w\}$. Because all other edges are supersets of that edge, the set $\{w\}$ already is a minimal hitting set. ∎

Having understood the number of minimal hitting sets, we now want to build some intuition on why the MMCS tree of $\mathcal{H}_\ell$ grows exponentially. Note that a non-min-heuristic will select all edges $\{w, u_i, u'_i\}$ for all i before it selects the edge $\{w\}$. This causes MMCS to build up all combinations of the form (u_0 or u'_0), (u_1 or u'_1), ..., ($u_{\ell-1}$ or $u'_{\ell-1}$). Note that all such combinations and their subsets are irredundant, otherwise MMCS would not generate them. Hence, the MMCS tree has exponential size. Note that while the mentioned partial solutions are irredundant, they are not *extendable*, i.e. they cannot be extended to a minimal hitting set by adding vertices. This is because $\{w\}$ is the only minimal hitting set. MMCS only recognizes this when it selects the edge $\{w\}$ as the last edge of every branch, upon which all other vertices lose their private edge, causing MMCS to discard the considered partial solution as redundant. As a short remark, we note that pruning all non-extendable partial solutions is computationally infeasible, since checking extendability is $W[3]$-complete, as shown in [Blä+22, Theorem 2]. This is why the pruning of partial solutions is based on irredundancy instead of extensibility. In the following lemma, we now formalize the intuition to prove that the MMCS tree grows exponentially. We use the notation $U_I := \{u_i \mid i \in I\} \sqcup \{u'_i \mid i \in I\}$ for index sets $I \subseteq \{0, \dots, \ell - 1\}$. Also, we write $I^\complement := \{0, \dots, \ell - 1\} \setminus I$.

Lemma 4.3 We consider MMCS with a non-min-heuristic, with or without violator pruning. Then for $k \leq \ell$, the k-th level of the MMCS tree contains 2^k nodes of the form (S, C), where there is an index set $I \subseteq \{0, ..., \ell - 1\}$ consisting of k indices such that $S \subseteq U_I$ is irredundant and $C \supseteq U_{I^\complement}$.

For example, the following illustrates how the MMCS tree for $\mathcal{H}_3$ might look like, omitting candidate sets for clarity.



A possible MMCS tree for $\mathcal{H}_3$

Proof. The proof goes by induction over the levels of the MMCS tree.

Base case. The statement holds for the 0-th level, which consists only of the start node $(\emptyset, V)$. The empty set $\emptyset$ is irredundant and the conditions are met with the empty index set $I = \emptyset$.

Inductive step. We assume that the statement holds for the $(k-1)$-th level, where $k \leq \ell$. We consider an arbitrary node (S, C) in the $(k-1)$-th level and investigate its child nodes.

By the induction assumption, we know that there is an index set $I \subseteq \{0, ..., \ell - 1\}$ containing $k-1$ indices, such that $S \subseteq U_I$ is irredundant and $C \supseteq U_{I^\complement}$.

To find out, what edge is selected in this situation, we recall that a non-min-heuristic selects an unhit edge E for which $|E \cap C|$ is not minimum, if such an edge exists. The unhit edges are exactly the edge $\{w\}$ and all edges of the form $\{w, u_i, u'_i\}$ with $i \in I^\complement$. There is an edge of the latter form, because $|I| = k - 1 < \ell$. Since the candidate set C contains $U_{I^\complement}$, we know that $|\{w, u_i, u'_i\} \cap C| \geq 2$. Because $|\{w\} \cap C| \leq 1$, the non-min-heuristic selects an edge $\{w, u_j, u'_j\}$ for some $j \in I^\complement$.

MMCS creates the partial solutions $S \cup \{u_j\}$ and $S \cup \{u'_j\}$ as child nodes. Both are irredundant, since u_j and u'_j hit only one edge of the hypergraph, which implies that all vertices of S keep their private edges.

Which vertices end up in a candidate set, depends on which child node the candidate set belongs to and, if violator pruning is enabled, which vertices are violating vertices. But in any case, MMCS removes at most the vertices of the selected edge from the candidate set, implying that the child nodes' candidate sets are supersets of $C \setminus \{w, u_i, u'_i\} \supseteq U_{(I \cup \{j\})^\complement}$.

Therefore, the two generated child nodes satisfy the conditions with the index set $I \cup \{j\}$. Since this construction works for all the 2^{k-1} nodes on the $(k-1)$-th level, we can conclude that the k-th level contains 2^k nodes of the required form. ∎

Theorem 4.1 now directly follows from Lemma 4.2 and Lemma 4.3, as argued above. As mentioned in the beginning of this section, the runtime changes drastically if we switch to the min-heuristic, which is the default. This is demonstrated in the following lemma.

Lemma 4.4 With the min-heuristic, with or without violator pruning, MMCS processes $\mathcal{H}_\ell$ in linear time and the MMCS tree has constant size.

Proof. The min-heuristic selects the edge $\{w\}$ as a first edge because all other edges are supersets of $\{w\}$. Since no edges remain unhit, MMCS is already done. While the MMCS tree consists only of one node, MMCS still takes linear time to process this node. In particular, finding the edge $\{w\}$ among all ℓ edges takes linear time. ∎

In this section, we have seen a particular class of hypergraphs on which MMCS is output-polynomial with the min-heuristic, but exhibits exponential runtime with a non-min-heuristic. However, this behavior can be exactly reversed, as we will see this in the next section.

4.2 MMCS with the Min-Heuristic and No Violator Pruning

In this section, we make a step closer to MMCS' default configuration by considering the min-heuristic, which is the default heuristic. As introduced in Definition 2.8, it always selects an unhit edge, for which the intersection with the current candidate set has minimal cardinality. Yet, we still differ from the default configuration of MMCS in that we consider violator pruning disabled. We construct a particular class of hypergraphs on which MMCS in the mentioned configuration is not output-polynomial. Interestingly, it becomes output-polynomial if we switch back to a non-min-heuristic, implying that the min-heuristic is not always optimal.

The goal of this section is to prove the following theorem. It assumes the worst case regarding ordering of vertices. That is, we assume the worst case in how MMCS orders the vertices of $E \cap C$, where C is the candidate set of a certain tree node and E is the edge selected for the generation of child nodes. The ordering of vertices only affects the candidate sets of the child nodes. Recall that MMCS does not specify the ordering of vertices.

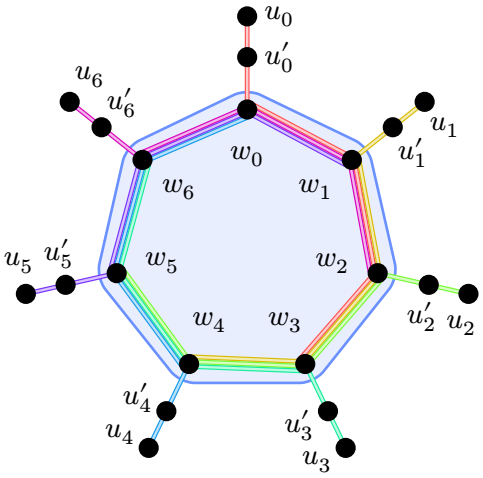
Theorem 4.5 MMCS with the min-heuristic and without violator pruning is not output-polynomial, where we assume the worst case regarding ordering of vertices.

Again, we prove this by explicitly constructing a class of hypergraphs for which the MMCS tree grows exponentially, while the number of minimal hitting sets only grows polynomially. Of course the number of vertices and edges may grow at most polynomially too. Then we can conclude that the MMCS tree is not output-polynomial,

implying that the runtime is not output-polynomial either. We consider the following class of hypergraphs, parameterized by a positive integer ℓ. Let $W = \{w_0, \dots, w_{\ell-1}\}$, $U = \{u_0, \dots, u_{\ell-1}\}$ and $U' = \{u'_0, \dots, u'_{\ell-1}\}$. All indices are modulo ℓ.

$$\begin{aligned} \mathcal{H}_\ell &:= \{\{u_i, u'_i\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\} \mid i \in \{0, \dots, \ell-1\}\} \cup \{W\} \\ V(\mathcal{H}_\ell) &:= W \sqcup U \sqcup U' \end{aligned}$$

The following visualizes $\mathcal{H}_7$, where the colored area represents the edge W. Every colored line represents one of the other edges and should not be confused for a path consisting of multiple edges of size 2.



Visualization of $\mathcal{H}_7$

Before we investigate how MMCS handles this class of hypergraphs, we first show that the number of minimal hitting sets grows at most polynomially in the following lemma.

Lemma 4.6 The hypergraph $\mathcal{H}_\ell$ has at most n^4 minimal hitting sets, where $n = 3\ell$ denotes the number of vertices of $\mathcal{H}_\ell$.

Proof. We use the notation $E_i := \{u_i, u'_i\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}$. As before, all indices are modulo ℓ.

Because of the edge $W \in \mathcal{H}_\ell$, any minimal hitting set must contain w_j for at least one $j \in \{0, \dots, \ell-1\}$. The vertex w_j alone already hits all edges except for the three edges E_{j+1}, E_{j+2} and E_{j+3}. Since all vertices in a minimal hitting set have their own private edge, i.e. an edge that is hit by that vertex and by no other vertex from the set, any minimal hitting set can consist of at most 4 vertices. Therefore, there are at most $\binom{n}{4} \leq n^4$ minimal hitting sets. ∎

Before we investigate the min-heuristic, we first show that MMCS with a non-min-heuristic is output-polynomial on $\mathcal{H}_\ell$.

Lemma 4.7 With a non-min-heuristic, with or without violator pruning, MMCS processes $\mathcal{H}_\ell$ in output-polynomial time.

Proof. Recall that a non-min-heuristic selects, whenever possible, an unhit edge E for which $|E \cap C|$ is not minimum, where C is the current candidate set. In the first step, it therefore selects the edge $W \in \mathcal{H}_\ell$, since all other edges have minimum cardinality. Consequently, every partial solution on the first level consists of a vertex from W. As argued in the proof of Lemma 4.6, only 3 edges remain unhit. When going from one level to the next, at least one previously unhit edge must become hit. As a result, the MMCS tree has depth at most 4 and therefore polynomial size, implying polynomial runtime. ∎

Now we shift our attention back to the min-heuristic. We show that with it, the runtime is exponential, where we consider violator pruning disabled for now. Together with Lemma 4.7, this shows that the min-heuristic is not always optimal. This is not entirely surprising, because the min-heuristic minimizes the number of branches greedily by minimizing the number of child nodes of the current node. However, that is not to say that we should prefer a non-min-heuristic, since we had the exact opposite behavior in the previous section. On the particular class of hypergraphs that we investigated there, the runtime with a non-min-heuristic is exponential while the runtime with the min-heuristic is polynomial.

Before proving the exponential runtime in the following lemma, we want to build some intuition. Note that all the edges $E_i = \{u_i, u_i'\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}$ are selected by the min-heuristic, before it selects the largest edge W. Just as in the last section, this causes MMCS to generate, among others, all combinations of the form $(u_0 \text{ or } u_0')$, $(u_1 \text{ or } u_1')$, ..., $(u_{\ell-1} \text{ or } u_{\ell-1}')$. Note that all of these and their subsets are irredundant, otherwise MMCS would not generate them. Hence, the MMCS tree grows exponentially. Note that the mentioned partial solutions are not extendable, because all minimal hitting sets have size at most 4. But since MMCS only checks for irredundancy and not for extendability, it discards the mentioned partial solutions only after the last remaining edge W is selected. Because then a vertex of W is added which makes the partial solution redundant. To formalize this intuition, we require a specific ordering of vertices. This is the reason why we need to assume the worst case regarding the ordering of vertices in Theorem 4.5. For the proof, we use the same approach as in the proof of Lemma 4.3. That is, we use induction over the levels of the MMCS tree to keep track of how many nodes of a certain form are in each level. Again, we use the notation $U_I := \{u_i \mid i \in I\} \sqcup \{u_i' \mid i \in I\}$ and $I^\complement := \{0, \dots, \ell-1\} \setminus I$ for index sets $I \subseteq \{0, \dots, \ell-1\}$.

Lemma 4.8 We consider MMCS with the min-heuristic and disabled violator pruning and assume that the vertices of $E \cap C$ are always ordered so that child nodes corresponding to vertices from U and U' get the largest candidate sets.
Then for $k \leq \ell$, the k-th level of the MMCS tree contains 2^k nodes of the form (S, C), where there is an index set $I \subseteq \{0, \dots, \ell-1\}$ consisting of k indices, such that $S \subseteq U_I$ is irredundant, $C \supseteq U_{I^\complement}$, and $C \supseteq W$.

Proof. The proof goes by induction over the levels of the MMCS tree.

Base case. The statement holds for the 0-th level, which consists only of the start node $(\emptyset, V)$. The empty set $\emptyset$ is irredundant and the conditions are met with the empty index set $I = \emptyset$.

Inductive step. We assume that the statement holds for the $(k-1)$-th level, where $k \leq \ell$. We consider an arbitrary node (S, C) in the $(k-1)$-th level and investigate its child nodes.

By the induction assumption, we know that there is an index set $I \subseteq \{0, ..., \ell - 1\}$ containing $k-1$ indices, such that $S \subseteq U_I$ is irredundant, $C \supseteq U_{I^\complement}$ and $C \supseteq W$.

The unhit edges are exactly the edge W and E_i for all $i \in I^\complement$, where we again denote $E_i = \{u_i, u_i'\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}$. There is an edge of the latter form, because $|I| = k - 1 < \ell$. Since the candidate set C contains both $U_{I^\complement}$ and W by assumption, we know that for every i, $|E_i \cap C| = 2 + |W| - 3 = \ell - 1$. For the edge W, we have $|W \cap C| = |W| = \ell$. Therefore, the min-heuristic selects an edge E_j for some $j \in I^\complement$. Note that we need the assumption $C \supseteq W$ in order for E_j to be selected. If we had $w_{i-1}, w_{i-2}, w_{i-3} \notin C$, then the edge W would be selected, because we could calculate $|E_i \cap C| = |\{u_i, u_i'\}| + |(W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}) \cap C| = 2 + |W \cap C| > |W \cap C|$.

Since E_j is selected, MMCS creates the partial solutions $S \cup \{u_j\}$ and $S \cup \{u_j'\}$ as child nodes. Both are irredundant, since u_j and u_j' hit only one edge of the hypergraph, which implies that all vertices of S keep their private edges.

By assumption, the child nodes corresponding to vertices of U and U' get the largest candidate sets. Since violator pruning is disabled, these are the candidate sets, where at most u_j and u_j' are removed. Therefore, the child nodes' candidate sets contain not just $U_{(I \cup \{j\})^\complement}$ but also W. Note that without that assumption, $S \cup \{u_j\}$ and $S \cup \{u_j\}$ might get candidate sets that do not fully contain W, which would break the induction.

Altogether, we see that the two generated child nodes satisfy the conditions with the index set $I \cup \{j\}$. Since this construction works for all the 2^{k-1} nodes on the $(k-1)$-th level, we can conclude that the k-th level contains 2^k nodes of the required form. ∎

Theorem 4.5 now directly follows from Lemma 4.6 and Lemma 4.8. Note, however, that we have not yet considered MMCS in its default configuration, since we disabled violator pruning. If we enable it, MMCS processes this $\mathcal{H}_\ell$ in polynomial time, as shown in the following lemma.

Lemma 4.9 With the min-heuristic and with violator pruning enabled, MMCS processes $\mathcal{H}_\ell$ in polynomial time. More precisely, the MMCS tree has depth at most 11.

Proof. If some partial solution S contains a vertex from W, the subtree under the node (S, C), for any candidate set C, has depth at most 4. This is because after having selected a vertex from W, at most 3 edges of $\mathcal{H}_\ell$ can remain unhit, while every vertex added in a step of MMCS hits at least one edge that was previously unhit. In the proof of Lemma 4.6, we used the same argument to show that minimal hitting sets of $\mathcal{H}_\ell$ contain at most 4 vertices.

We will show that selecting a vertex from W becomes inevitable within a constant number of levels. To build some intuition, we recall from the proof of Lemma 4.8 that we can only guarantee that the min-heuristic selects an edge of the form $E_j = \{u_j, u'_j\} \sqcup W \setminus \{w_{j-1}, w_{j-2}, w_{j-3}\}$ for some j, if the current candidate set contains all vertices from W. Otherwise, the edge W might be selected, causing all child nodes to contain a vertex from W. Therefore, we will track the candidate sets over few levels and show that almost all vertices from W are pruned as violating vertices.

Let us consider an irredundant partial solution S on the fourth level containing no vertex from W. We do not need to consider redundant partial solutions, since they are already leaves in the MMCS tree. Without loss of generality, we assume that the next selected edge is $E_j = \{u_j, u'_j\} \sqcup W \setminus \{w_{j-1}, w_{j-2}, w_{j-3}\}$, because if the next selected edge would be W, then all child nodes would contain a vertex from W.

Having selected E_j, MMCS branches on the vertices u_j and u'_j, as well as the vertices from $W \setminus \{w_{j-1}, w_{j-2}, w_{j-3}\}$. To track the candidate sets of the child nodes, we observe that $S \cup \{w_i\}$ is redundant for all i. This is because the 4 vertices that are not in W previously had private edges, while w_i hits all but 3 edges, therefore making at least one of the 4 vertices lose their private edge. As a consequence, all vertices that MMCS considers for branching that are part of W are violating and thus removed from all child nodes' candidate sets. From W only the vertices w_{j-1}, w_{j-2} and w_{j-3} might remain in the candidate set, since they are not part of the selected edge E_j.

Only in that way we might get partial solutions on the fifth level that still contain no vertices from W. We consider such a node and assume again without loss of generality that the next selected edge is $E_k = \{u_k, u'_k\} \sqcup W \setminus \{w_{k-1}, w_{k-2}, w_{k-3}\}$. As before, we can see that all vertices that MMCS considers for branching that are part of W are violating and thus removed from all child nodes' candidate sets. Since $E_j \neq E_k$, at least one of the vertices w_{j-1}, w_{j-2} and w_{j-3} that has previously remained in the candidate set is now removed as a violating vertex. Repeating this one more time, we lose yet another such vertices from the candidate set.

We can conclude that if there is an irredundant partial solution S in the seventh level containing no vertex from W, its candidate set C may contain at most one vertex from W. To conclude our proof, we can see that the min-heuristic will select the edge W in that situation. This is because for any unhit edge of the form $E_i = \{u_i, u'_i\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}$ for some i, we can do the following calculation, in which we use that C contains at most one vertex from W.

$$\begin{aligned} |E_i \cap C| &= |\{u_i, u'_i\} \sqcup (W \setminus \{w_{i-1}, w_{i-2}, w_{i-3}\}) \cap C| \\ &= 2 + |W \cap C| - |\{w_{i-1}, w_{i-2}, w_{i-3}\} \cap C| \\ &\geq 2 + |W \cap C| - 1 > |W \cap C| \end{aligned}$$

In conclusion, all irredundant partial solutions in the eighth level must contain a vertex from W. We can conclude that the MMCS tree has depth at most 11, which implies that it contains at most n^{12} nodes, where $n = 3\ell$ is the number of vertices. ∎

We want to mention as a short remark that it is possible to generalize the class of hypergraphs by introducing a *gap parameter* $g \in \mathbb{N}$. It represents how many vertices of W an edge leaves out. Above we had $g = 3$.

$$\mathcal{H}_\ell^g := \left\{\{u_i, u_i'\} \sqcup W \setminus \{w_{i-1}, w_{i-2}, \dots, w_{i-g}\} \mid i \in \{0, \dots, \ell - 1\}\right\} \sqcup \{W\}$$
$$V(\mathcal{H}_\ell^g) := W \sqcup U \sqcup U'$$

Now in the general case, $\mathcal{H}_\ell^g$ has at most n^{g+1} minimal hitting sets, cf. Lemma 4.6. The proof works in the same way. The central argument that a vertex $w_j \in W$ hits all except for g edges, works for all $g \in \mathbb{N}$, not just for $g = 3$. Likewise, in the general case but with the restriction $g \geq 3$, the MMCS tree has exponential size, cf. Lemma 4.8. We need $g \geq 3$ to ensure that the min-heuristic does not select the edge W before all other edges have been selected. With that observation the proof works exactly like above. Also, enabling violator pruning reduces the runtime to being polynomial in the general case, cf. Lemma 4.9. The proof works like before, but it may take more levels until the candidate sets contain at most one vertex from W. For completeness, it should be stated that this result still considers the min-heuristic and disabled violator pruning and assumes the worst case regarding the ordering of vertices, as before.

In this section, we have seen a particular class of hypergraphs, on which MMCS with the min-heuristic and without violator pruning is not output-polynomial, while it becomes output-polynomial when we enable violator pruning. Interestingly, it also becomes output-polynomial if we switch the min-heuristic for a non-min-heuristic, contrasting the behavior we have seen for a different class of hypergraphs in the previous section.

4.3 MMCS in Its Default Configuration

In this section, we finally consider MMCS in its default configuration, i.e. with the min-heuristic and with violator pruning enabled. We construct a class of hypergraphs and a class of graphs on which MMCS is not output-polynomial in its default configuration. The insight gained through these classes, suggests augmenting MMCS with a heuristic for ordering vertices, which we motivate in this section. By constructing these classes, we prove the following two theorems. As in the last section, we assume the worst case regarding the ordering of vertices, cf. notes before Theorem 4.5. For the second theorem, we additionally assume the worst case regarding the selection of edges. That is, we assume the worst case in how MMCS selects an edge among all the edges that satisfy the min-heuristic.

Theorem 4.10 MMCS in its default configuration, i.e. with the min-heuristic and violator pruning enabled, is not output-polynomial, where we assume the worst case regarding the ordering of vertices.

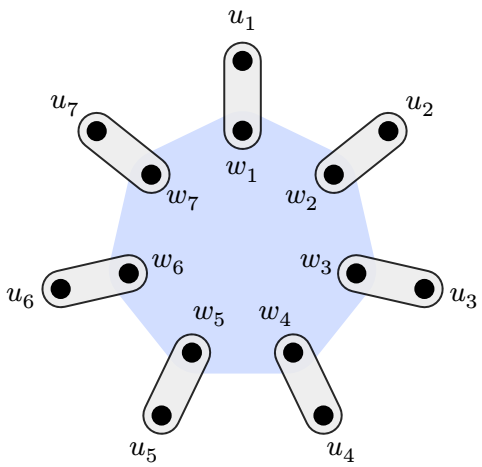
Theorem 4.11 MMCS in its default configuration is not even output-polynomial on graphs, where we assume the worst case regarding the ordering of vertices and selection of edges.

Recall that minimal hitting sets on graphs are better known as vertex covers. While it is currently unknown whether there are output-polynomial algorithms enumerating all minimal hitting sets of hypergraphs, it is known that there are output-polynomial algorithms enumerating all vertex covers of graphs [TIAS77 | JYP88]. Theorem 4.11 shows that MMCS is not one of them.

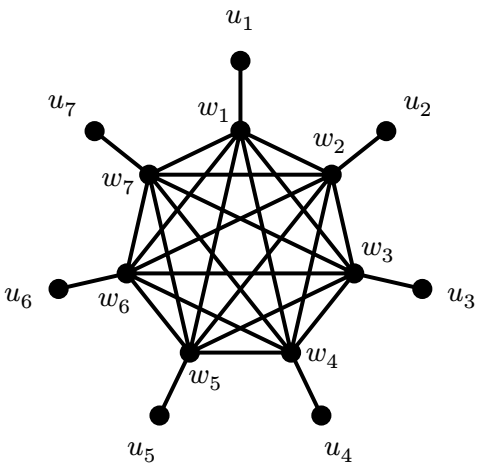
As in the previous two sections we prove these theorems by constructing a class of hypergraphs for which the MMCS tree grows exponentially, while the number of minimal hitting sets only grows polynomially. To simplify notation, we denote the complete p-uniform hypergraph on a vertex set W as $\binom{W}{p} := \{W' \subseteq W \mid |W'| = p\}$. With that notation, we define, for every integer constant $p \geq 2$, the following class of hypergraphs, parameterized by a positive integer $\ell > p$. Let $W = \{w_1, \dots, w_\ell\}$ and $U = \{u_1, \dots, u_\ell\}$.

$$\mathcal{H}_\ell^p := \binom{W}{p} \sqcup \{\{u_i, w_i\} \mid i \in \{1, \dots, \ell\}\}$$
$$V(\mathcal{H}_\ell^p) := \{w_1, \dots, w_\ell, u_1, \dots, u_\ell\} = W \sqcup U$$

Note that in the case $p = 2$, the class of hypergraphs $\mathcal{H}_\ell^2$ is actually a class of graphs. In that case $\binom{W}{2}$ is the complete graph on the vertex set W, that is usually denoted as K_ℓ. The following are visualizations with $\ell = 7$ and $p = 3$ or $p = 2$, respectively. For $p = 3$, we symbolize $\binom{W}{3}$ with a colored area for readability. In the case of $p = 2$, we draw all edges as simple lines, since $\mathcal{H}_\ell^2$ is a graph.



Visualization of $\mathcal{H}_7^3$



Drawing of $\mathcal{H}_7^2$

First, we prove that $\mathcal{H}_\ell^p$ has at most polynomially many minimal hitting sets. Recall that p is a constant in our setting.

Lemma 4.12 $\mathcal{H}_\ell^p$ has at most $n^{2(p-1)}$ minimal hitting sets, where $n = 2\ell$ is the number of vertices.

Proof. To hit all the edges $E \in \binom{W}{p}$, a vertex set may omit at most $p-1$ vertices of W. If it omitted p vertices, say $w_1, \dots, w_p \in W$, then the edge $\{w_1, \dots, w_p\} \in \binom{W}{p}$ would be unhit.

Therefore, every minimal hitting set of $\mathcal{H}_\ell^p$ must have a subset $T' \subseteq W$ consisting of all except for $p-1$ vertices of W. But that subset T' already hits all edges of $\mathcal{H}_\ell^p$ except for $p-1$ edges, namely the edges $\{w_i, u_i\}$ for all $w_i \notin T'$. That implies that there are at most n^{p-1} ways to extend T' to a minimal hitting set, where $n = 2\ell$ is the number of vertices. This is because every vertex added to T' must have a private edge, which in particular must be unhit by T', which implies that there may only be at most $p-1$ vertices in a hitting set in addition to T'.

To conclude the proof, we see that there are exactly $\binom{\ell}{p-1}$ sets in $\binom{W}{\ell-(p-1)}$. Since every minimal hitting set is a superset of such a set, and there are only n^{p-1} ways to extend such a set to a minimal hitting set, we can conclude that there are at most $\binom{\ell}{p-1}n^{p-1}$ minimal hitting sets. Using $\binom{\ell}{p-1} \leq \ell^{p-1} \leq n^{p-1}$, we can finally conclude $|\mathrm{Tr}(\mathcal{H}_\ell^p)| \leq n^{2(p-1)}$. ∎

Before we prove the exponential growth of the MMCS tree, let us first build some intuition. The min-heuristic selects all edges of the form $\{u_i, w_i\}$ with some i, before it selects any edge of $\binom{W}{p}$. For $p = 2$ this holds only due to our worst-case assumption regarding the selection of edges among the edges satisfying the min-heuristic. For $p > 2$, however, we do not need that assumption. Selecting all edges of that form causes MMCS to build up all combinations of the form (u_1 or w_1), (u_2 or w_2), ..., (u_ℓ or w_ℓ). Note that all such combinations and their subsets are irredundant, otherwise MMCS would not consider them. Hence, the MMCS tree grows exponentially. To develop this intuition in a formal proof, we require a specific ordering of vertices, which is why we assume the worst case regarding the ordering of vertices in Theorem 4.10 and Theorem 4.11. As in the previous two chapters, the proof uses induction over the levels to keep track of how many nodes of a certain form are in each level.

Lemma 4.13 We consider MMCS with the min-heuristic and enabled violator pruning. We assume that when an edge $\{u_j, w_j\}$ is selected, the vertices are ordered so that the child node corresponding to u_j gets the larger candidate set. In the case $p = 2$, we further assume that whenever an edge of the form $\{u_j, w_i\}$ for some j satisfies the min-heuristic, then such an edge is selected and not an edge from $\binom{W}{2}$.
Then for $k < \ell$, the k-th level of the MMCS tree contains 2^k nodes of the form (S, C) where there is an index set $I \subseteq \{1, \dots, \ell\}$ with $|I| = k$ so that for all $i \in I$, the partial solution S contains either u_i or w_i but not both and no other vertices. Furthermore, every vertex of W is either a candidate or included in the partial solution, and the candidate sets C contains all vertices u_i for $i \notin I$.

Proof. The proof goes by induction over the levels of the MMCS tree. However, before we start with the induction, we first note that a partial solution S that satisfies the conditions of the induction hypothesis is irredundant. By assumption, S can contain at most one vertex from the pair $\{u_i, w_i\}$ for all $i \in \{1, \dots, \ell\}$. If one of them is contained in S, it therefore has $\{u_i, w_i\}$ as a private edge. Consequently, every vertex of S has a private edge, implying that S is irredundant.

Base case. The statement holds for the 0-th level, which only consists of the start node $(\emptyset, V)$. All conditions are met with the empty index set $I = \emptyset$.

Induction step. We assume that the statement holds for the $(k-1)$-th level, where $k < \ell$ and consider a node (S, C) on this level. We denote its corresponding index set as $I \subseteq \{1, \dots, \ell\}$.

To investigate which edge the min-heuristic selects, we look at $|E \cap C|$ for an unhit edge $E \in \binom{W}{p}$. Since E is unhit, all $w \in E$ are not in S. By the induction assumption, every $w \in E$ must therefore be in the candidate set C, allowing us to conclude that $|E \cap C| = p$. On the other hand, we have $|\{u_i, w_i\} \cap C| = 2$ for every unhit edge of the form $\{u_i, w_i\}$ with some i. This is because the induction assumption guarantees $u_i \in C$ for all $i \notin I$, which is the case since $\{u_i, w_i\}$ is unhit. Moreover, $w_i \in C$ follows from the same argument as before.

In the case $p \geq 3$, we can therefore conclude that the min-heuristic will select an unhit edge $\{u_j, w_j\}$ for some j. Note that since $|I| = k - 1 < \ell$, at least one such edge is still unhit and thus available for selection. In the case $p = 2$, we need the additional assumption that the particular implementation of MMCS selects such an edge even if the unhit edges from $\binom{W}{2}$ also satisfy the min-heuristic.

Having selected the edge $\{u_j, w_j\}$, MMCS considers the partial solutions $S \cup \{w_j\}$ and $S \cup \{u_j\}$. By assumption, the latter gets assigned the larger candidate set, which in this case, is $C \setminus \{u_j\}$, while the former receives $C \setminus \{u_j, w_j\}$ as candidate set. No vertices are pruned as violators, because both considered partial solutions are irredundant and non-hitting. They are irredundant, because they satisfy the conditions of the induction hypothesis for the index set $I \cup \{j\}$, and we have shown above that we can conclude irredundancy from these conditions. Since $|I \cup \{j\}| = k < \ell$, there is still an unhit edge, implying that the partial solutions are not yet hitting sets.

To conclude the induction step, we need to check that the candidate sets of both child nodes satisfy the conditions for the index set $I \cup \{j\}$. Since we removed at most u_j and w_j from the candidate sets, both of them contain all u_j and w_i for all $i \notin I \cup \{j\}$. The more interesting condition is that every vertex of W is either in the partial solution or the corresponding candidate set. For the child node $(S \cup \{u_j\}, C \setminus \{u_j\})$, this is clear since swapping u_j from the candidate set to the index set does not affect any vertex of W. For the child node $(S \cup \{w_j\}, C \setminus \{u_j, w_j\})$ the vertex w_j is swapped and therefore still either in the partial solution or the candidate set. Since all other vertices of W are not affected by swapping w_j, we can conclude that the condition is satisfied for this child node too.

Altogether, we see that the two generated child nodes satisfy the conditions with the index set $I \cup \{j\}$. Since this construction works for all the 2^{k-1} nodes on the $(k-1)$-th level, we can conclude that the k-th level contains 2^k nodes that satisfy the required conditions. ∎

Note that the assumption about the ordering of vertices really matters in this example. Let us consider the case $p = 2$ for that. If for some node (S, C) the vertices of the selected edge $\{w_j, u_j\}$ are ordered so that w_j is the first vertex, then w_j gets excluded from candidate set of the partial solution $S \cup \{u_j\}$, which is $C' \coloneqq C \setminus \{u_j, w_j\}$. That implies that the vertex w_j is neither in the partial solution $S \cup \{u_j\}$ nor in its

corresponding candidate set C'. This in turn means that for all $w_i \notin S \cup \{w_j\}$, the edge $\{w_j, w_i\}$ is unhit and $|\{w_j, w_i\} \cap C'| = 1$. This causes MMCS to select all these vertices one after the other, without branching into two nodes at any step. In that way, MMCS reaches either the minimal hitting set $\{u_j, w_1, \dots, w_{j-1}, w_{j+1}, \dots, w_\ell\}$ or a superset of it. In both cases the branch is finished. With some further thoughts, we generalize this argument for every $p \geq 2$ in the following lemma.

Lemma 4.14 If MMCS in its default configuration, upon selecting an edge $\{w_j, u_j\}$ of the input graph $\mathcal{H}_\ell^p$, always orders the vertices so that the child node corresponding to w_j gets the larger candidate set, then the size of the MMCS tree is at most $(2\ell)^p$. In particular MMCS is output-polynomial on $\mathcal{H}_\ell^p$ in that case.

Proof. For a node (S, C), we define $W_{\text{ex}} := W \setminus (S \cup C)$. The proof relies on the following observation. If for a node (S, C), the set W_{ex} has size $p-1$, then the subtree under the node (S, C) is trivial, meaning that every descendant has exactly one child until a redundant set or a minimal hitting set is found. In other words, there are no branching nodes in the subtree under (S, C), i.e. nodes with two or more children. To see this, consider $w \in W \setminus W_{\text{ex}}$. It has an edge $E := \{w\} \cup W_{\text{ex}} \in \binom{W}{p}$ for which $E \cap C$ consists of only one vertex, namely w. Since the min-heuristic always selects an edge for which that intersection has minimum cardinality, all $w \in W \setminus W_{\text{ex}}$ must be selected before MMCS can branch into two or more nodes again. Similarly, since $\{u_i, w_i\} \cap C = \{u_i\}$ for all $w_i \in W_{\text{ex}}$, all these vertices u_i must be selected before MMCS can branch into two or more nodes again. Note that $|W_{\text{ex}}| > p-1$ would imply that there is an unhit edge that has an empty intersection with the candidate set, which is not possible by Lemma 2.10. Hence, after adding all vertices mentioned above, the resulting partial solution S' must contain all vertices of W except for the $p-1$ vertices that W_{ex} consists of. Additionally, it must contain all vertices u_i for which it does not contain w_i, i.e. for all $w_i \in W_{\text{ex}}$. In that case, however, S' is already a hitting set and hence either a minimal hitting set or redundant. In both cases MMCS is finished with that branch.

Taking the point of view of an adversary that tries to maximize the number of branching nodes in the MMCS tree, we call adding a vertex to W_{ex} a *bad choice*. In these terms, MMCS can only make $p-1$ bad choices until no further branching points can occur.

Whenever MMCS selects an edge $\{u_i, w_i\}$ for a node (S, C) with $u_i, w_i \in C$, it generates the child node $(S \cup \{u_i\}, C \setminus \{u_i, w_i\})$. Here we used the assumption that the child node corresponding to w_i gets the larger candidate set. Note that w_i now is unselected and excluded from the candidate set, i.e. MMCS made a bad choice for that child node. That implies that MMCS makes one bad choice for one child node. Also, consider the case where an edge $E \in \binom{W}{p}$ is selected. W.l.o.g., its intersection with the candidate set is $E \cap C = \{w_1, \dots, w_k\}$. Then MMCS generates the child nodes $(S \cup \{w_i\}, C \setminus \{w_1, \dots, w_i\})$. That implies that for one child node, namely $(S \cup \{w_1\}, C \setminus \{w_1\})$, no bad choice is being made while for the next child node $(S \cup \{w_2\}, C \setminus \{w_1, w_2\})$, one bad choice is being made with w_1, whereas for the third child node, two bad choices are being made, and so on until the last child node $(S \cup \{w_k\}, C \setminus \{w_1, \dots, w_k\})$ makes $k-1$ bad choices with $w_1, \dots w_{k-1}$. So whatever edge is selected, if there are k child nodes, then one makes no bad choice, the next makes one bad choice, the next makes

two bad choices, and so on. Intuitively, this implies that in most branches, it only takes few branching nodes until the $p-1$ bad choices have been made, at which point no further branching nodes can occur.

To calculate how large the MMCS tree can be, we consider the number $N_b(d)$ which is defined as the maximum number of branching nodes that can occur in a tree that has depth d and in which there cannot be any branching nodes after b bad choices. The observation above about the number of bad choices per child node can be written as the following recurrence formula, where $d \in \mathbb{N}_0$ and $b \in \mathbb{N}$.

$$N_b(d+1) \leq 1 + \sum_{i=0}^{b-1} N_{b-i}(d) \tag{4.1}$$

The 1 comes from the root node of the tree and the sum goes over all its child nodes. For the child node with index i, we make i bad choices, so the subtree under that child nodes has only $b-i$ bad choices left until no further branching nodes may appear. Also, that subtree has depth d, since its parent has depth $d+1$. For all child nodes with index $i \geq b$ that might be generated, all b bad choices are exhausted, which is why the sum only goes up to $i = b-1$. Of course, there might be fewer than b child nodes, but we do not aim for a tight bound here. To connect this back to the size of the MMCS tree, note that the number of branching nodes in the MMCS tree can be bounded by $N_{p-1}(2\ell)$. This is because the MMCS tree has depth at most 2ℓ and admits at most $p-1$ bad choices before no further branching nodes can occur.

We claim that $N_b(d) \leq d^b$ holds for all $b \in \mathbb{N}$ and $d \in \mathbb{N}_0$. Assume for an induction proof, that this holds for some fixed $d \in \mathbb{N}_0$. Then we can do an inductive step using Equation (4.1) and the binomial theorem.

$$\begin{aligned} N_b(d+1) &\leq 1 + \sum_{i=0}^{b-1} N_{b-i}(d) \\ &\leq 1 + \sum_{i=0}^{b-1} d^{b-1} \\ &\leq 1 + \sum_{i=0}^{b-1} \binom{b}{i} d^{b-i} = (d+1)^b \end{aligned}$$

To conclude the induction, note that $N_b(0) = 0 \leq 0^b$ holds for all $b \in \mathbb{N}$.

Finally, note that between any two branching nodes, there can be at most 2ℓ non-branching nodes. Given that the MMCS tree contains at most $N_{p-1}(2\ell) \leq (2\ell)^{p-1}$ branching nodes, we can conclude that it contains at most $(2\ell)^p$ nodes in total. ∎

The fact that the ordering of vertices determines whether MMCS needs exponential or polynomial time on $\mathcal{H}_\ell^p$ suggests that it might be beneficial to augment MMCS with a heuristic for ordering vertices. To develop our consideration above into a heuristic for ordering vertices, note that for almost all partial solutions S, the vertices of W have a larger degree in the subhypergraph $\operatorname{unhit}(S)$ than the vertices of U. With the *degree in the subhypergraph* $\operatorname{unhit}(S)$, we mean the number of yet unhit edges that the given vertex is part of. Let us consider $p = 2$ here. For every $w_i \in W$, as long

as there are still three vertices of W missing from S, there are two unhit edges that w_i is part of. At the same time, any vertex $u_i \in U$ is part of at most one unhit edge. Note that if both vertices of a selected edge are non-violating, one of them is excluded only from its own child node's candidate set, while the other one is only also excluded from the other child node's candidate set. In $\mathcal{H}_\ell^2$, the large degree of a vertex $w_i \in W$ is precisely why excluding it from the candidate set of its sibling node is beneficial. Because then, the intersection of every unhit edge containing w_i with the candidate set gets smaller by one. This is precisely how many edges get to have an intersection of size one when excluding w_i from the candidate set of the sibling node. As we have seen above, these intersections of size one cause MMCS to finish quickly. For edges of size two, we therefore propose to order the two vertices so that the vertex of larger degree in unhit(S) is excluded from both child nodes' candidate sets. In the following and final chapter, we investigate this heuristic rigorously and generalize it to the case where there are more than two vertices to branch on. Unsurprisingly, we can see that for every $p \geq 2$, this heuristic enables MMCS to process $\mathcal{H}_\ell^p$ in polynomial time.

Corollary 4.15 With the heuristic for vertex ordering that is described above and later formally defined in Definition 5.1, MMCS solves $\mathcal{H}_\ell^p$ in polynomial time for every $p \geq 2$, where we consider the min-heuristic and enabled violator pruning.

Proof. Upon selecting an edge $\{w_j, u_j\}$, the heuristic *almost* always orders the two vertices so that the child node corresponding w_j gets the larger candidate set. That is precisely the condition of Lemma 4.14, which implies polynomial runtime. However, there is one case, where the other child node could get the larger candidate set. This happens only when the current partial solution S already contains all vertices of W besides w_j and $p-1$ others, because then the degree of w_j in unhit(S) is one. This, however, is not a problem for the polynomial runtime, since MMCS can only run into such partial solutions polynomially often and the subtree under them is polynomial. Therefore, the entire MMCS tree can only be polynomially larger than it would be under the conditions of Lemma 4.14. To see why MMCS can only run into such nodes polynomially often, note that for any $w_i \in S$, the vertex u_i cannot be selected by S, otherwise S would be redundant. Hence, there are at most $\binom{\ell}{p}2^p$ such partial solutions, which is polynomial in ℓ. To see why the subtree under such nodes is polynomial, note that S hits all edges of $\binom{W}{p}$ except for one edge, namely the edge consisting of the p unselected vertices. Also, at most p of the remaining edges are unhit, namely the edges $\{w_i, u_i\}$ for which neither w_i nor u_i is selected. With only a constant number of edges remaining unhit, the MMCS tree under the considered node has polynomial size, as we have argued before several times, e.g., in the proof of Lemma 4.12. ∎

Summarizing the most important results of this chapter, we have seen that MMCS is not output-polynomial in its default configuration, where we had to assume the worst case regarding the ordering of vertices. In fact, MMCS in its default configuration is not even output-polynomial on graphs, if we additionally assume the worst case regarding edge selection among all edges satisfying the min-heuristic. Moreover, we have seen that the min-heuristic can lead to exponentially better or worse runtime than a non-min-heuristic depending on the input. We have also seen that disabling

violator pruning can exponentially increase the runtime. Finally, we have developed an idea for a heuristic for ordering vertices that reduces the runtime from exponential to polynomial on the class $\mathcal{H}_\ell^p$ from this section.

5 A Heuristic for Ordering Vertices

In this chapter, we investigate the heuristic for ordering vertices that arose in Section 4.3. We prove that it is equivalent to minimizing the number of *potential branching points* among all direct child nodes. What this means exactly is defined later.

Before stating the heuristic, let us first recall the role of the ordering of vertices. When MMCS selects an edge $E \in \text{unhit}(S)$, we look at the intersection $E \cap C$, where S is the current partial solution and C the corresponding candidate set. We denote the vertices of $E \cap C$ as $v_1, \dots, v_k$. For the rest of this chapter, we use the convention that the chosen ordering $v_1, \dots, v_k$ results in the following candidate sets. For every i, the partial solution $S \cup \{v_i\}$ gets the candidate set $C \setminus \{v_1, \dots v_i\}$. If there are any violating vertices, all of them have to be removed from all candidate sets. In particular, the first vertex v_1 is excluded from the most candidate sets while the last vertex v_k is excluded from the fewest. The same convention is used implicitly in the original paper [MU14]. To avoid confusion, we should note that the ordering of vertices we discuss here is independent of the order in which an implementation chooses to recursively process the child nodes. That said, we propose the following heuristic, which is based on our intuition from Section 4.3.

Definition 5.1 The *degree heuristic for ordering vertices* sorts the vertices of $E \cap C$ descendingly by their degree in the subhypergraph $\text{unhit}(S)$, where E is the selected edge, S the current partial solution and C the corresponding candidate set. That means the vertex with the largest degree in $\text{unhit}(S)$ gets the largest candidate set.

Here, we use the notation $\deg_{\text{unhit}(S)}(v) = |\{E \in \text{unhit}(S) \mid v \in S\}|$ for the degree of a vertex v in the subhypergraph $\text{unhit}(S) \subseteq \mathcal{H}$.

In the rest of this chapter, we prove that the degree heuristic is equivalent to minimizing the number of *potential branching points* among all child nodes.

Definition 5.2 A *potential branching point* of a (child) node (S', C') for which S' is irredundant and non-hitting, is a combination (E', v) of an edge $E' \in \text{unhit}(S')$ and a vertex $v \in E' \cap C'$. For S' being redundant or a hitting set, we say that the (child) node (S', C') has no potential branching points, since such nodes are always leaves in the MMCS tree.

The intuition behind a potential branching point (E', v) is that MMCS will create a new branch for v if the edge E', which is available for selection, is selected. Using this definition, we can state the following theorem about the degree heuristic for ordering vertices.

Theorem 5.3 Upon branching on an edge $E \in \text{unhit}(S)$ for a node (S, C), ordering the vertices of $E \cap C$ according to the degree heuristic for ordering vertices minimizes the number of potential branching points among all direct child nodes of (S, C).

Proof. We prove the slightly stronger statement, that ordering the vertices in such a way that minimizes the number of potential branching points among all direct child nodes is equivalent to ordering the non-violating vertices descendingly by their degree, while violating vertices can appear at any place in the ordering.

Let us consider a node (S, C) for which we already have selected an edge $E \in \text{unhit}(S)$. We consider some ordering of the vertices $E \cap C = \{v_1, \dots, v_k\}$ and investigate how many potential branching points there are among all child nodes.

First, observe that any child node (S', C') has fewer potential branching points than its parent (S, C). We look at how much smaller the set of potential branching points of a child node (S', C') is compared to (S, C). Of course, we get rid of all potential branching points involving an edge $E' \in \text{unhit}(S)$ which is now hit by S'. But since this does not depend on the candidates, the ordering of vertices has no effect on it, so we can safely ignore this effect in our analysis. The candidate set C' only affects the number of potential branching points in the following way. If a candidate $v \in C$ gets excluded from the child's candidate set C', then we get rid of any potential branching points (E', v) with an edge E' that includes v. In other words by excluding v, we get rid of $\deg_{\text{unhit}(S')}(v)$ potential branching points.

In the following, we assume that there are no violating vertices or equivalently that all child nodes are irredundant and non-hitting. Firstly, we can ignore all child nodes that are redundant or hitting, since by definition, they have no potential branching points. Secondly, we can ignore that violating vertices are excluded from the candidate sets of all child nodes. While this can actually decrease the number of potential branching vertices in all child nodes, this decrease does not depend on the ordering of vertices. This is because violating vertices are excluded from all candidate sets regardless of the ordering of vertices. We can conclude that only the ordering of non-violating vertices matters.

We now count how many potential branching points we get rid of among all direct child nodes. For that, let us consider the child node $(S \cup \{v_i\}, C \setminus \{v_1, \dots, v_i\})$ for some i. Since the vertices $v_1, \dots, v_i$ are excluded from the child node's candidate set, we get rid of the following number of potential branching points.

$$\sum_{j=1}^{i} \deg_{\text{unhit}(S \cup \{v_i\})}(v_j)$$

Summing over all child nodes, we get rid of the following number N of potential branching points in total.

$$N = \sum_{i=1}^{k} \sum_{j=1}^{i} \deg_{\text{unhit}(S \cup \{v_i\})}(v_j) \tag{5.1}$$

Only one step is missing to see why we maximize this number by ordering the vertices according to the degree heuristic. This step involves the following observation, where $\mathcal{E}_{ij} \coloneqq \{E \in \operatorname{unhit}(S) \mid v_i, v_j \in E\}$.

$$\deg_{\operatorname{unhit}(S \cup \{v_i\})}(v_j) = \deg_{\operatorname{unhit}(S)}(v_j) - \left|\mathcal{E}_{ij}\right|$$

This follows from $\operatorname{unhit}(S) = \operatorname{unhit}(S \cup \{v_i\}) \sqcup \{E \in \operatorname{unhit}(S) \mid v_i \in E\}$ by intersecting both sides with $\{E \in \mathcal{H} \mid v_j \in E\}$. Using this we can decompose the number N of potential branching points that we get rid of from Equation (5.1) into the following.

$$N = \underbrace{\sum_{i=1}^{k} \sum_{j=1}^{i} \deg_{\operatorname{unhit}(S)}(v_j)}_{\text{(I)}} - \underbrace{\sum_{i=1}^{k} \sum_{j=1}^{i} \left|\mathcal{E}_{ij}\right|}_{\text{(II)}}$$

We can see that (II) does not depend on the ordering of vertices. The key observation for this is that $|\mathcal{E}_{ij}| = |\mathcal{E}_{ji}|$ for all i, j, which follows directly from the definition of $\mathcal{E}_{ij}$. We can therefore write $\text{(II)} = \sum_{i<j} |\mathcal{E}_{ij}| = \frac{1}{2} \sum_{i \neq j} |\mathcal{E}_{ij}|$. This is now independent of the ordering of vertices, since all combinations of indices appear. We can conclude that only (I) matters if we want to find an ordering of vertices that maximizes the number of potential branching points that we get rid of. Since in (I), the summand is now independent of the summation index i, we can rewrite (I) to the following.

$$\text{(I)} = k \deg_{\operatorname{unhit}(S)}(v_1) + \cdots + 2 \deg_{\operatorname{unhit}(S)}(v_{k-1}) + 1 \deg_{\operatorname{unhit}(S)}(v_k)$$

This representation shows that in order to maximize the number of potential branching points that we get rid of, we have to order the vertices decreasingly by their degree in the hypergraph $\operatorname{unhit}(S)$. That means we must choose v_1 to be the vertex with the largest degree in $\operatorname{unhit}(S)$, v_2 must have second-largest degree, and so on. ∎

As a final remark, we shortly note that minimizing the number of potential branching points among all direct child nodes does not necessarily minimize the actual number of grandchildren. Intuitively, we are trying to get as many intersections $E \cap C$ as small as possible, whereas for the actual number of grandchildren only the smallest such intersection of every child node matters.

In conclusion, we have seen that the degree heuristic, that arose from the example $\mathcal{H}_\ell^2$ in Section 4.3, minimizes the number of potential branching points among all child nodes. This suggests that the degree heuristic might be beneficial in general, not just on this particular example.

6 Conclusion

We showed exponential lower bounds for various configurations of MMCS even when there are only polynomially many minimal hitting sets. Most importantly, we showed that MMCS is not output-polynomial in its default configuration, i.e. with the min-heuristic and enabled violator pruning. This result even holds when we restrict the inputs to graphs instead of general hypergraphs, but for that case, we assumed the worst case regarding the selection of edges among all edges satisfying the min-heuristic. This shows that, while there are output-polynomial algorithms enumerating all vertex covers of graphs, MMCS is not one of them. Furthermore, we saw that MMCS with a non-min-heuristic can take exponential time, while the min-heuristic reduces the runtime to being polynomial on that particular class of hypergraphs. This behavior can also be exactly reversed, as we saw with a class of hypergraphs that is solved in polynomial time using a non-min-heuristic but requires exponential time when using the min-heuristic and disabling violator pruning. Enabling violator pruning in that case makes the runtime polynomial again, demonstrating that violator pruning can make a significant difference. Some of these results, especially the lower bound for MMCS in its default configuration, assume the worst case regarding the ordering of vertices. The ordering of vertices is not specified in MMCS, and it determines the candidate sets of the child nodes. We showed that a certain ordering of vertices reduces the runtime from exponential to polynomial on the class of hypergraphs that disproves output-polynomiality for MMCS. Based on the intuition from this example, we propose to augment MMCS with a heuristic for ordering vertices based on their degree among all edges that remained unhit so far. We showed that this heuristic minimizes the number of potential branching points among all child nodes, suggesting that it could improve the performance of MMCS in general, not just on the particular class of hypergraphs from which it arose. Besides lower bounds, we showed that MMCS is output-polynomial on paths and cycles due to their exponential number of minimal hitting sets. As secondary findings, we made two interesting observations about MMCS that, to our knowledge, have not been published before. Firstly, we showed that the min-heuristic prevents MMCS from running into the situation where the candidate set is not sufficient to hit all remaining edges. Secondly, we showed that while only inclusion-minimal edges matter for the transversal hypergraph, MMCS behaves differently on a hypergraph $\mathcal{H}$ and on its minimization $\min(\mathcal{H})$ due to redundancy checks.

Future Work

While we have seen that the use of the min-heuristic can cause the runtime to increase from being polynomial to being exponential in certain cases, and the opposite in other cases, we have not shown a similar result for violator pruning. We saw a class of

hypergraphs on which enabling violator pruning decreased the runtime from being exponential to being polynomial. It might be interesting to investigate if there are hypergraphs on which the opposite behavior is displayed.

It is currently still unknown whether there are output-polynomial algorithms enumerating all minimal hitting sets of hypergraphs. We showed that MMCS, which on many instances is the most efficient algorithm for enumerating minimal hitting sets, is not output-polynomial. That said, augmenting MMCS with our proposed heuristic for ordering vertices could, in principle, make MMCS output-polynomial. It is an interesting topic for future research to see if and how one could modify the hypergraph $\mathcal{H}_\ell^p$ from Section 4.3 to cause MMCS to need exponential time even with that new heuristic in place. If this is successful, it might give further hints on how to modify MMCS to improve its performance in practice or even make it output-polynomial.

While we showed that the new heuristic for ordering vertices minimizes the number of potential branching points among all direct child nodes, it would be a very interesting topic of future research to empirically evaluate the effects of the new heuristic on many instances in practice.

Another interesting idea would be to add another strategy for candidate pruning. Note that when we add a vertex u_i from the hypergraph $\mathcal{H}_\ell^p$ from Section 4.3, then we can remove the vertex w_i from the candidate set of $S \cup \{u_i\}$. This is because adding u_i only hits a single previously unhit edge $\{w_i, u_i\}$, and that edge contains w_i. So adding w_i to $S \cup \{u_i\}$ later, would cause u_i to lose its only private edge. With that pruning technique, MMCS solves $\mathcal{H}_\ell^p$ in polynomial time for the same reason that is given in the proof of Lemma 4.14, even without the new degree heuristic for vertex ordering. In general, we propose the following strategy for candidate pruning. Upon adding a vertex v to a partial solution S, we check if v hits only one previously unhit edge. If that is the case, then we remove all vertices of that edge from the candidate set of $S \cup \{v\}$. Note that this check can easily be integrated in the original implementation of MMCS [MU14], since that implementation already iterates over all edges that are hit by v and stores if those edges have remained unhit so far.

Lastly, it might be interesting to find larger classes of graphs or hypergraphs on which MMCS is output-polynomial. A potential candidate would be the class of trees. Here, the approach that worked for paths and cycles does not work, because trees do not necessarily have exponentially many minimal hitting sets. For example, the star $K_{1,n}$ consisting of a center vertex that is adjacent to all other n vertices, has exactly two minimal vertex covers for every $n \in \mathbb{N}$. Nevertheless, there is a limit to how general such classes of hypergraphs can be, since we could show that MMCS is not output-polynomial on the class of all graphs.

Acknowledgment I want to thank my advisor, Martin Schirneck, for his guidance and his detailed feedback on all parts of this thesis, as well as for answering all my questions related to this thesis and the academic world in general during our relaxed regular meetings. Also, I want to thank Thomas Bläsius for joining one of our meetings and bringing up the idea to augment MMCS with a heuristic for ordering vertices. Finally, I want to thank my friends, some of whom have taken the time to give me some feedback.

Bibliography

[Ber84] C. Berge. *Hypergraphs: Combinatorics of Finite Sets*. Elsevier, May 1, 1984. ISBN: 978-0-08-088023-5.

[BI95] J. C. Bioch and T. Ibaraki. "Complexity of Identification and Dualization of Positive Boolean Functions". In: *Information and Computation* Volume 123 (Nov. 15, 1995), pp. 50–63. ISSN: 0890-5401. DOI: *10.1006/inco.1995.1157*.

[Bir+20] Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. "Hitting set enumeration with partial information for unique column combination discovery". In: *Proc. VLDB Endow.* Volume 13 (July 1, 2020), pp. 2270–2283. ISSN: 2150-8097. DOI: *10.14778/3407790.3407824*.

[Blä+22] Thomas Bläsius, Tobias Friedrich, Julius Lischeid, Kitty Meeks, and Martin Schirneck. "Efficiently enumerating hitting sets of hypergraphs arising in data profiling". In: *Journal of Computer and System Sciences* Volume 124 (Mar. 1, 2022), pp. 192–213. ISSN: 0022-0000. DOI: *10.1016/j.jcss.2021.10.002*.

[Ble+24] Tobias Bleifuß, Thorsten Papenbrock, Thomas Bläsius, Martin Schirneck, and Felix Naumann. "Discovering Functional Dependencies through Hitting Set Enumeration". In: *Proc. ACM Manag. Data* Volume 2 (Mar. 26, 2024), 43:1–43:24. DOI: *10.1145/3639298*.

[DV87] János Demetrovics and Vu Duc Thi. "Keys, antikeys and prime attributes". In: *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Sectio Computatorica* Volume 8 (1987), pp. 35–52.

[EG95] Thomas Eiter and Georg Gottlob. "Identifying the Minimal Transversals of a Hypergraph and Related Problems". In: *SIAM Journal on Computing* Volume 24 (Dec. 1995), pp. 1278–1304. ISSN: 0097-5397. DOI: *10.1137/S0097539793250299*.

[FK96] Michael L. Fredman and Leonid Khachiyan. "On the Complexity of Dualization of Monotone Disjunctive Normal Forms". In: *Journal of Algorithms* Volume 21 (Nov. 1, 1996), pp. 618–628. ISSN: 0196-6774. DOI: *10.1006/jagm.1996.0062*.

[GV17] Andrew Gainer-Dewar and Paola Vera-Licona. "The Minimal Hitting Set Generation Problem: Algorithms and Computation". In: *SIAM Journal on Discrete Mathematics* Volume 31 (Jan. 2017), pp. 63–100. ISSN: 0895-4801. DOI: *10.1137/15M1055024*.

[Hag09] Matthias Hagen. “Lower bounds for three algorithms for transversal hypergraph generation”. In: *Discrete Applied Mathematics* Volume 157 (Apr. 6, 2009), pp. 1460–1469. ISSN: 0166-218X. DOI: *10.1016/j.dam.2008.10.004*.

[JYP88] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. “On generating all maximal independent sets”. In: *Information Processing Letters* Volume 27 (Mar. 25, 1988), pp. 119–123. ISSN: 0020-0190. DOI: *10.1016/0020-0190(88)90065-8*.

[KBEG07] Leonid Khachiyan, Endre Boros, Khaled Elbassioni, and Vladimir Gurvich. “On the dualization of hypergraphs with bounded edge-intersections and other related classes of hypergraphs”. In: *Theoretical Computer Science* Volume 382 (Aug. 31, 2007), pp. 139–150. ISSN: 0304-3975. DOI: *10.1016/j.tcs.2007.03.005*.

[KLMN14] Mamadou Moustapha Kanté, Vincent Limouzy, Arnaud Mary, and Lhouari Nourine. “On the Enumeration of Minimal Dominating Sets and Related Notions”. In: *SIAM Journal on Discrete Mathematics* Volume 28 (Jan. 2014), pp. 1916–1929. ISSN: 0895-4801. DOI: *10.1137/120862612*.

[KPN22] Jan Kossmann, Thorsten Papenbrock, and Felix Naumann. “Data dependencies for query optimization: a survey”. In: *The VLDB Journal* Volume 31 (Jan. 1, 2022), pp. 1–22. ISSN: 0949-877X. DOI: *10.1007/s00778-021-00676-3*.

[Mar26] Arnaud Mary. *Enumeration of minimal transversals of hypergraphs of bounded VC-dimension*. Jan. 30, 2026. arXiv: *2407.00694[math]*.

[MR87] Heikki Mannila and Kari-Jouko Räihä. “Dependency Inference”. In: *Proceedings of the 13th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Sept. 1, 1987, pp. 155–158. ISBN: 978-0-934613-46-0.

[MU14] Keisuke Murakami and Takeaki Uno. “Efficient algorithms for dualizing large-scale hypergraphs”. In: *Discrete Applied Mathematics* Volume 170 (June 19, 2014), pp. 83–94. ISSN: 0166-218X. DOI: *10.1016/j.dam.2014.01.012*.

[Rei87] Raymond Reiter. “A theory of diagnosis from first principles”. In: *Artificial Intelligence* Volume 32 (Apr. 1, 1987), pp. 57–95. ISSN: 0004-3702. DOI: *10.1016/0004-3702(87)90062-2*.

[Tak08] Ken Takata. “A Worst-Case Analysis of the Sequential Method to List the Minimal Hitting Sets of a Hypergraph”. In: *SIAM Journal on Discrete Mathematics* Volume 21 (Jan. 2008), pp. 936–946. ISSN: 0895-4801. DOI: *10.1137/060653032*.

[TIAS77] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. “A New Algorithm for Generating All the Maximal Independent Sets”. In: *SIAM Journal on Computing* Volume 6 (Sept. 1977), pp. 505–517. ISSN: 0097-5397. DOI: *10.1137/0206036*.