The Clique Problem with Multiple-Choice Constraints under a Cycle-Free Dependency Graph

Andreas Bärmann[1], Patrick Gemander[2] and Maximilian Merkert[3]

[1] Andreas.Baermann@fau.de
Lehrstuhl für Wirtschaftsmathematik,
Department Mathematik,
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstraße 11, 91058 Erlangen, Germany

[2] Patrick.Gemander@fau.de
Gruppe Data Science and Optimization
Fraunhofer-Arbeitsgruppe für Supply Chain Services SCS,
Fraunhofer-Institut für Integrierte Schaltungen IIS,
Nordostpark 93, 90411 Nürnberg, Germany

[3] Maximilian.Merkert@ovgu.de
Institut für Mathematische Optimierung,
Fakultät für Mathematik,
Otto-von-Guericke-Universität Magdeburg,
Universitätsplatz 2, 39106 Magdeburg, Germany

First draft online: 26 January 2018
Revision date: 14 November 2019

Abstract

The clique problem with multiple-choice constraints (CPMC) represents a very common substructure in many real-world applications, for example scheduling problems with precedence constraints. It consists in finding a clique in a graph whose nodes are partitioned into subsets, such that exactly one node from each subset is chosen. Even though we can show that (CPMC) is NP-complete, we will be able to identify and characterize a new interesting subclass which is solvable in polynomial time. It arises whenever there are no circular dependencies between the subsets in the partition when it comes to the compatibilities of their elements. We will be able to show that this subclass is equivalent to the clique problem on a perfect graph with certain additional structure. This finding will allow us to give a full convex-hull description. Although the convex hull can have exponentially-many facets, we can state an efficient and practical separation algorithm. In an application from underground and railway timetabling, we show that, in comparison to a naive formulation, these results lead to significant reductions in computation time.

Keywords: Clique Problem, Multiple-Choice Constraints, Perfect Graph, Convex-Hull Description, Scheduling

Mathematics Subject Classification: 90C27 - 90C57 - 90C35 - 90C90

1 Introduction

The clique problem with multiple-choice constraints (CPMC) is a combinatorial optimization problem which is prevalent as a substructure in scheduling problems with precedence constraints and other problems where compatibilities between decision alternatives play a role. It can be stated as follows: let $G = (V, E)$ be an undirected graph and $\mathcal{V} = \{V_1, \dots, V_m\}$ a partition of its node set V into m disjoint subsets such that each subset V_i is a stable set in G. The clique problem with multiple-choice constraints then asks to find a clique in G that contains exactly one node from each subset V_i. In other words, we would like to find an m-clique in an m-partite graph.

Despite its importance in the field of scheduling alone, this problem has been relatively little studied as an interesting combinatorial problem by itself in the literature. [GIZ+02] deals with the enumeration of all m-cliques in an m-partite graph and uses the developed algorithm in the context of an application from textile engineering. In [MK13], the authors present an improved version of this enumeration method. (CPMC) has also been considered under a complement viewpoint, where edges represent conflicts rather than compatibilities. What is a clique of size m in our context is then referred to as an *independent transversal* or an *independent system of representatives*. [Kin11] proves a criterion that guarantees the existence of an independent transversal while [ABZ07] approaches a colourability conjecture. The framework of *systems of disjoint representatives* for a family of hypergraphs studied in [AH00] also fits this setting; the authors prove a hypergraph version of Hall's theorem using topological arguments. For an introduction to the general clique problem and related concepts, including polyhedral properties, we refer to [GLS88]. A broad survey on the maximal clique problem can be found in [BBPP99].

To the best of our knowledge, [BGMS18] for the first time studies polyhedral properties of (CPMC). This paper investigates a special case of the problem, referred to as *staircase compatibility*. It imposes certain conditions on the structure of the subgraphs that are induced by the union $V_i \cup V_j$ of any two distinct subsets V_i and V_j. Under these conditions, the authors derive totally unimodular representations of the problem as a linear constraint system of polynomial size, whose existence shows that this special case is solvable in polynomial time.

In the present paper, we first show the NP-completeness of (CPMC) in general and then proceed to a new special case which is also solvable in polynomial time. It does not rely on the structure of the subgraph induced by two subsets, but restricts which pairs of subsets in $\mathcal{V}$ may have mutually incompatible elements. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the *dependency graph* of a (CPMC)-instance $(G, \mathcal{V})$, which contains an edge between two subsets V_i and V_j if and only if there exist two nodes $v \in V_i$ and $w \in V_j$ such that $\{v, w\} \notin E$. We will be able to show that (CPMC) is solvable in polynomial time if the dependency graph is a forest. This result stems from our finding that in this case, to which we refer as (CPMCF), the graph G is perfect. Furthermore, to describe the convex hull of the set of feasible solutions to (CPMCF), it is sufficient to consider only the stable-set inequalities induced by subgraphs on $V_i \cup V_j$ for any edge $\{V_i, V_j\} \in \mathcal{E}$ of the dependency graph. In other words, the separation problem reduces to a maximum-weight stable-set problem in a bipartite graph. This property will make it possible to derive non-trivial upper bounds on the number of stable-set inequalities needed to describe the convex hull. Moreover, it enables us to use a network flow algorithm to efficiently solve the corresponding separation problem. In a computational case study, we will show that there is significant benefit from exploiting these properties.

This paper is organized as follows: Section 2 begins with some context on the problem from the literature and shows its NP-completeness before giving a formal definition of the special case (CPMCF). We also state a dynamic-programming method to solve it in polynomial time for linear objective functions. Afterwards, Section 3 presents the full polyhedral description as well as graph-theoretic properties. Section 4 closes the theoretical part of this paper with the derivation of a non-trivial exponential upper bound for the number of facets of the convex

hull and a worst-case example in which an exponential number of facets is indeed attained. In Section 5, we will apply our theoretical results to an optimization problem from the field of energy-efficient underground and railway timetabling where (CPMC) and (CPMCF) arise as substructures to model the timetable constraints. We will be able to demonstrate that, in comparison to a naive formulation, they lead to significant reductions in computation time for instances originating from real-world data. We finish with our conclusions in Section 6.

2 Problem Definition and Basic Properties

In this section, we will provide a formal definition of the clique problem with multiple-choice constraints and prove its NP-completeness in general. Afterwards we will define the aforementioned special case (CPMCF), which is the focus of this paper. We will see that this special case is solvable in polynomial time.

Definition 2.1. *(Clique Problem with Multiple-Choice Constraints) Let $G = (V, E)$ be a simple undirected graph and $\mathcal{V} = \{V_1, \dots, V_m\}$ a partition of V such that each $V_i \in \mathcal{V}$ is a stable set in G. The* clique problem with multiple-choice constraints *(CPMC) is then given by the task to find a clique of size m in G. An instance of (CPMC) is consequently given by the corresponding pair $(G, \mathcal{V})$, where we refer to G as the* compatibility graph *of the (CPMC)-instance.*

Remark 2.2. *It is clear from this definition that (CPMC) is a subclass of the general clique problem. More precisely, the problem consists in finding an m-clique in an m-partite graph, which is due to $\mathcal{V}$ being a partition of V into m stable sets in G. Note that for any fixed m, the problem is polynomially solvable in time $O(|V|^m)$ by enumeration. In contrast, the decision variant of the problem is NP-complete in general but remains polynomially solvable if $|V_i| \leq 2$ holds for all $i = 1, \dots, m$. This will be shown by Theorem 2.3 and Remark 2.4.*

(CPMC) asks for a clique that contains exactly one node from each of the subsets in a given partition of the node set, which can, from a polyhedral point of view, be interpreted as the addition of multiple-choice constraints to a general clique problem. (CPMC) has been described in [BGMS18] as a combinatorial structure which frequently occurs in scheduling problems with precedence constraints, such as the project scheduling problem (cf. [SZ15], which provides a broad survey on project scheduling, including problem variants, modelling and solution approaches as well as applications). However, the authors also give an example where (CPMC) can be used to model piecewise-linear flow costs in a network. This application has been studied in more detail in [LM16]. [BGMS18] also introduces a special case of (CPMC), called *staircase compatibility*, in which there are certain restrictions on the structure of the subgraphs induced by the union of any two subsets in the partition. The authors show the polynomial solvability of this special case by providing totally unimodular formulations for the problem which have polynomial size. [LM16] shows the same by explicitly proving that the graph G is perfect in the context of the studied application.

In [Kar72] it has been proved that the clique problem is NP-complete. In particular, the proof shows that the satisfiability problem can be reduced to finding an m-clique in an m-partite graph with known partition, as is the case for (CPMC). Therefore, the decision variant of (CPMC) is NP-complete. Before we present another special case in which (CPMC) is solvable in polynomial time, we will give an explicit proof of the NP-completeness of the decision variant of its general version.

Theorem 2.3. *The decision variant of (CPMC) is NP-complete.*

Proof. We prove this result by showing that the problem of graph colourability for a given number k of colours can be reduced to the decision variant of (CPMC). The k-colourability problem is well known to be NP-complete for $k \geq 3$ (see [GJS74], [GJ79, Problem GT4]). Let

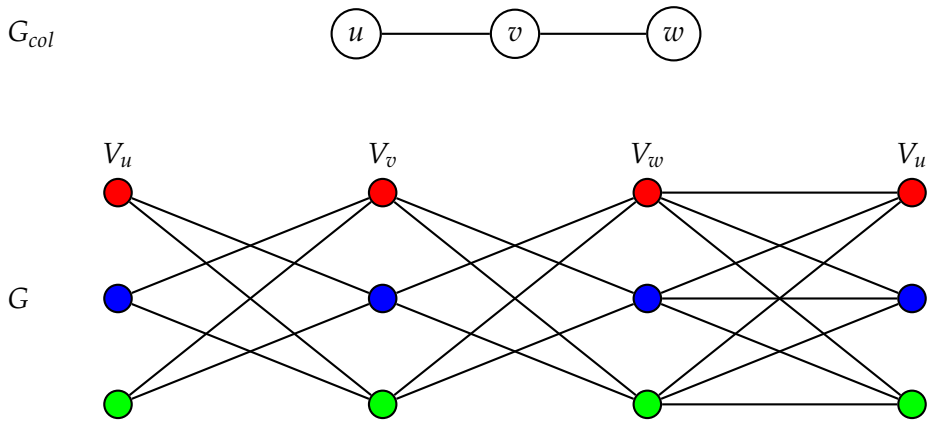


Figure 1: Illustration of the reduction of the k-colourability problem to (CPMC) as described in the proof of Theorem 2.3, here for an example instance with $k = 3$ colours and a graph G_{col} consisting of three nodes and two edges. Note that V_u is displayed twice in order to show its interconnectedness with V_v and V_w more clearly.

$G_{col} = (V_{col}, E_{col})$ be a graph to be coloured and let k be the number of available colours. We can model this problem as (CPMC) with a graph G as follows: create a set of nodes V_i for each node $i \in V_{col}$ with nodes $V_i = \{v_{ic} \mid 1 \leq c \leq k\}$, with the interpretation that choosing v_{ic} corresponds to colouring $i \in V_{col}$ with colour c. As illustrated by Figure 1, an edge $\{v_{ic}, v_{jc'}\}$ is contained in the edge set of G if and only if

$$\{i, j\} \notin E_{col} \text{ or } c \neq c'$$

holds. This fully describes the colourability problem in terms of the decision variant of (CPMC), and the transformation clearly is polynomial. □

Remark 2.4. *The proof of Theorem 2.3 implies that the decision variant of (CPMC) is NP-complete even if the size of each subset V_i is at most three. However, when restricted to two elements per subset, the decision variant of (CPMC) becomes solvable in polynomial time. This can be shown by a reduction to 2-satisfiability, which is solvable in polynomial time (see [Kro67]).*

The above theorem shows that the additional structure of (CPMC) compared to the general clique problem does not make it easier to solve. In the present work, we will investigate a subcase in which there is no circular dependency between the subsets in the partition. The following notations will be helpful in this respect.

Definition 2.5. *(Subgraphs G_{ij}, Neighbourhoods $N_j(U)$ and Dependency Graph $\mathcal{G}$) Let $\mathcal{I} = (G, \mathcal{V})$ with $G = (V, E)$ and $\mathcal{V} = \{V_1, \ldots, V_m\}$ be an instance of (CPMC). For two subsets $V_i, V_j \in \mathcal{V}$ with $i \neq j$, we write*

$$G_{ij} := (V_i \cup V_j, E_{ij})$$

for the subgraph of G induced by $V_i \cup V_j$, where E_{ij} is the corresponding edge set. Note that all subgraphs G_{ij} are bipartite. Furthermore, the neighbourhood *$N_j(U) \subseteq V_j$ of a subset $U \subseteq V$ in V_j is given by*

$$N_j(U) := \{v \in V_j \mid (\exists u \in U)\, \{u, v\} \in E\}.$$

It represents those nodes in V_j for which there is a compatible node in U. Finally, we call $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with

$$\mathcal{E} := \{\{V_i, V_j\} \subseteq \mathcal{V} \mid (\exists u \in V_i)(\exists v \in V_j)\, \{u, v\} \notin E\}$$

the dependency graph *of G. Note that $\{V_i, V_j\} \in \mathcal{E}$ is equivalent to G_{ij} not being a complete bipartite graph.*

The name "dependency graph" is motivated by the fact that for choosing a clique in G, the choice of a node from V_i may directly restrict the choice of a node from V_j if the edge $\{V_i, V_j\}$ is contained in $\mathcal{G}$.

In the above terms, we can give a general formulation of (CPMC) as a binary feasibility problem. We introduce a variable $x_v \in \{0, 1\}$ which takes a value of 1 if node $v \in V$ is selected and 0 otherwise. A possible model formulation then looks as follows:

$$
\begin{aligned}
&\text{find} \quad x \\
&\text{s.t.} \quad \sum_{v \in V_i} x_v &= 1 \qquad & (\forall V_i \in \mathcal{V}) && \text{(1a)} \\
&\qquad x_v &\leq \sum_{\substack{w \in V_j: \\ \{v,w\} \in E}} x_w \qquad & (\forall (V_i, V_j) \in \mathcal{V} \times \mathcal{V} \colon \{V_i, V_j\} \in \mathcal{E})(\forall v \in V_i) && \text{(1b)} \\
&\qquad x &\in \{0,1\}^{|V|}. && && \text{(1c)}
\end{aligned}
$$

The multiple-choice constraints (1a) ensure the selection of exactly one node v per subset $V_i \in \mathcal{V}$. The compatibility constraints (1b) enforce that the chosen nodes form a clique in G. Finally, the constraints (1c) restrict variable x to binary values. Note that a compatibility constraint $x_v \leq \sum_{w \in V_j : \{v,w\} \in E} x_w$ for a node v in subset V_i and a second subset V_j is redundant to constraints (1a) if $\{v, w\} \in E$ for all $w \in V_j$, as the right-hand side sums up to 1 in this case. Furthermore, for each edge $\{V_i, V_j\} \in \mathcal{E}$ in the dependency graph, model (1) contains compatibility constraints (1b) for both ordered pairs (V_i, V_j) and (V_j, V_i) although only one set of inequalities for each $\{V_i, V_j\} \in \mathcal{E}$ would suffice for a correct integer programming model. However, this second set of inequalities improves the linear programming (LP) relaxation, which is why we keep it in the model. Altogether, we refer to model (1) as the *naive* formulation for (CPMC).

We now define a new special case of (CPMC) for which it is possible to improve the above formulation significantly.

Definition 2.6. *((CPMC) under a Cycle-Free Dependency Graph) If the dependency graph $\mathcal{G}$ of a given (CPMC)-instance $\mathcal{I}$ is a* forest*, i.e. it does not contain cycles, then we call $\mathcal{I}$ an instance of the* clique problem with multiple-choice constraints under a cycle-free dependency graph (CPMCF).

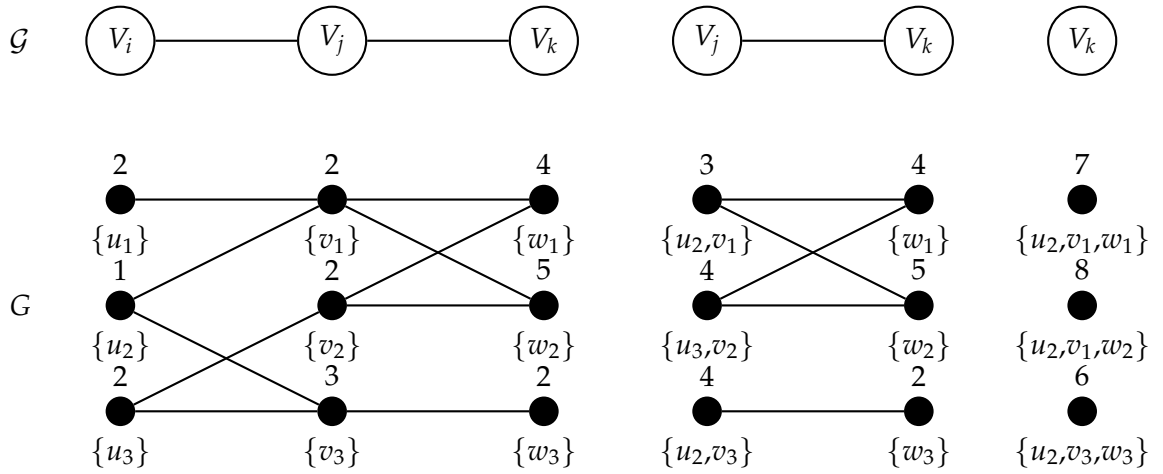
In the remainder of this paper, we will give a thorough investigation of the properties of (CPMCF) and study an indicative application in which it occurs as a substructure.

Before exploring the polyhedral structure of (CPMCF), we want to show that (CPMCF) can be solved in polynomial time by a dynamic-programming algorithm for any linear objective function. The key insight is that we can construct partial solutions by iteratively contracting leaf nodes into their respective parent node, which is possible because the dependency graph $\mathcal{G}$ does not contain cycles.

A node $v \in V_i$ with an empty neighbourhood $N_j(\{v\})$ for some $j \neq i$ cannot be part of a feasible solution and can thus be removed entirely. Therefore, we can assume w.l.o.g. that each node in any given subset has at least one compatible node in each adjacent subset in the dependency graph:

$$(\forall V_i \in \mathcal{V})(\forall v \in V_i)(\forall \{V_i, V_j\} \in \mathcal{E})\, N_j(\{v\}) \neq \emptyset. \tag{2}$$

(CPMCF) can then be solved in time $\mathcal{O}(|E|)$ by Algorithm 1, which considers each edge of the graph G for contraction at most once. An illustration of the algorithm is given in Figure 2.



(a) Initial graph (b) Graph after first iteration (c) Graph after second iteration

Figure 2: Illustration of the dynamic program described in Algorithm 1. Figure 2a shows the initial graph. Figure 2b and Figure 2c show the graphs after first contracting V_i into V_j and then V_j into V_k respectively. Note that for each node $w \in V_k$, the corresponding set C_w finally contains a cost-minimal clique in G containing w. With $\mathcal{G}$ consisting of one connected component only, Algorithm 1 simply returns $C = \{u_2, v_3, w_3\}$.

Algorithm 1 A dynamic-programming algorithm for (CPMCF) with linear node costs

Input A (CPMCF)-instance $\mathcal{I} = (G, \mathcal{V})$ satisfying condition (2) with costs $c_v \in \mathbb{R}$ for $v \in V$
Output A weight-minimal clique C in G satisfying the multiple-choice constraints

1: **for** $v \in V$ **do**
2: $\quad C_v \leftarrow \{v\}$
3: **end for**
4: **while** $\mathcal{E} \neq \emptyset$ **do**
5: $\quad$ Select $\{V_i, V_j\} \in \mathcal{E}$ with $\deg_{\mathcal{G}}(V_i) = 1$
6: $\quad$ **for** $v \in V_j$ **do**
7: $\quad\quad$ Select $u \in V_i$ with $c_u = \min_{w \in V_i : \{w,v\} \in E}\{c_w\}$
8: $\quad\quad$ $c_v \leftarrow c_v + c_u$
9: $\quad\quad$ $C_v \leftarrow C_v \cup C_u$
10: $\quad$ **end for**
11: $\quad$ $G \leftarrow (V \setminus V_i, E \setminus \{\{u, v\} \mid u \in V_i\})$
12: $\quad$ $\mathcal{G} \leftarrow (\mathcal{V} \setminus \{V_i\}, \mathcal{E} \setminus \{\{V_i, V_j\}\})$
13: **end while**
14: $C \leftarrow \emptyset$
15: **for** $V_i \in \mathcal{V}$ **do**
16: $\quad$ Select $u \in V_i$ with $c_u = \min_{w \in V_i}\{c_w\}$
17: $\quad$ $C \leftarrow C \cup C_u$
18: **end for**
19: **return** C

Lemma 2.7. *Algorithm 1 is correct.*

Proof. Together with the fact that $\mathcal{G}$ is cycle-free, condition (2) ensures that for each $v \in V$ there exists a solution C with $v \in C$. In particular, each C_v from Line 2 can be part of a feasible solution, and each C_v contains the vertices required to build a weight-minimal solution containing v.

Line 5 then selects a leaf V_i of $\mathcal{G}$ with some parent V_j, which is always possible as $\mathcal{G}$ contains no cycle. Consequently, each vertex $v \in V_i$ is compatible to all vertices $u \in (\bigcup_{V_k \in \mathcal{G}} V_k) \setminus V_j$ of the remaining graph $\mathcal{G}$. Hence, the aforementioned properties of C_v still hold after Line 9 as the algorithm chose the neighbour with minimal cost in Line 7. Since $\mathcal{G}$ is a forest, either a leaf node to be selected in Line 4 exists, or each connected component in $\mathcal{G}$ is already contracted to a single vertex. In the latter case, the algorithm selects a weight-minimal solution for each connected component of $\mathcal{G}$, which yields a feasible weight-minimal solution to the entire problem. □

As Algorithm 1 offers a weight-minimal solution w.r.t. given node costs, it also allows to solve the standard (CPMCF) and therefore leads to the following theorem:

Theorem 2.8. *(CPMCF) is solvable in polynomial time.*

We now compare (CPMCF) with a second polynomial-time-solvable special case of (CPMC) which has been introduced in [BGMS18]:

Remark 2.9. *(Comparison of (CPMCF) to (CPMC) under Staircase Compatibility) In [BGMS18], the authors study another special case of the clique problem with multiple-choice constraints which is solvable in polynomial time: (CPMC) under staircase compatibility, or (CPMCS) in short. An instance $\mathcal{I} = (G, \mathcal{V})$ of (CPMC) falls into this class if each subset $V_i \in \mathcal{V}$ can be ordered according to some total order $<_i$ such that the following two conditions hold for all subgraphs $G_{ij} = (V_i \cup V_j, E_{ij})$:*

1. *If $u \in V_i$ and $v_1 <_j v_2 <_j v_3 \in V_j$ with $\{u, v_1\}, \{u, v_3\} \in E_{ij}$, then $\{u, v_2\} \in E_{ij}$.*
2. *If $u_1 <_i u_2 \in V_i$ and $v_1 <_j v_2 \in V_j$ with $\{u_1, v_2\}, \{u_2, v_1\} \in E_{ij}$, then $\{u_1, v_1\}, \{u_2, v_2\} \in E_{ij}$.*

The subgraph G_{uv} from Figure 1 is an example which shows that (CPMCF) and (CPMCS) are indeed distinct special cases of (CPMC). To fulfil the staircase conditions, we would need to find total orders of V_u and V_v such that all three nodes in V_u are pairwise successive, and the same for V_v, which is obviously impossible. On the other hand, it is clear that not all (CPMCS)-instances belong to (CPMCF) as the former can have arbitrary cycles in the dependency graph. As a conclusion, (CPMCS) really only requires a special structure of the subgraphs G_{ij}, while (CPMCF) only requires a special structure of the dependency graph.

3 Polyhedral Description and Graph-Theoretic Properties

In the following, we will introduce the (CPMCF)-polytope, for which we will be able to give a full description in terms of linear constraints. This will rely on the fact that the graph corresponding to an instance of (CPMCF) is perfect. We will also see that (CPMCF) allows for an efficient separation of the facet-inducing maximal stable-set constraints.

Given an instance $\mathcal{I} = (G, \mathcal{V})$ of (CPMCF), we denote the (CPMCF)-polytope by

$$P_{CPMCF}(\mathcal{I}) \coloneqq \operatorname{conv}(\{x \in \{0,1\}^{|V|} \mid x \text{ is an incidence vector of a solution to } \mathcal{I}\}).$$

The main result of this section is the following theorem, giving a full polyhedral description of the (CPMCF)-polytope:

Theorem 3.1. *Let $\mathcal{I} = (G, \mathcal{V})$ be an instance of (CPMCF). Then the (CPMCF)-polytope $P_{CPMCF}(\mathcal{I})$ is determined by the constraints*

$$\sum_{v \in V_i} x_v = 1 \quad (\forall V_i \in \mathcal{V}) \tag{3a}$$

$$\sum_{v \in S} x_v \leq 1 \quad (\forall \text{ stable sets } S \subseteq V) \tag{3b}$$

$$x_v \geq 0 \quad (\forall v \in V). \tag{3c}$$

We will refer to model (3) as the *stable-set formulation*. The stable-set inequalities (3b) together with the non-negativity constraints (3c) already hint that perfect graphs play a major role in our proof of the above theorem. Recall that perfect graphs have originally been defined by [Ber63] as graphs with the property that the clique number and the colouring number coincide for all induced subgraphs. The following theorem, proved in [GLS88], states an equivalent characterization of perfect graphs from a polyhedral point of view.

Theorem 3.2. *A graph G is perfect if and only if its* clique polytope

$$P_{clique} := \text{conv}(\{x \in \{0,1\}^{|V|} \mid x \text{ is an incidence vector of a clique in } G\})$$

is determined by

$$\sum_{v \in S} x_v \leq 1 \quad (\forall \text{ stable sets } S \subseteq V) \tag{4a}$$

$$x_v \geq 0 \quad (\forall v \in V). \tag{4b}$$

Note that the naive formulation is itself a "stable-set formulation" in the sense that the multiple-choice constraints ((1a) and (3a) respectively) are present in both formulations and, using the constraints (1a), the compatibility constraints (1b) can be reformulated as stable-set constraints. However, this does not improve the naive formulation as it still contains only a subset of all required stable-set constraints. Finally, this also implies that the LP relaxation of the stable-set formulation cannot be worse than the LP relaxation of the naive formulation.

The following definition introduces a property which has been proved in [Hoà87] to be a sufficient condition for a graph to be perfect.

Definition 3.3. *(Hole, Antihole, Alternately Colourable) A* hole *in a graph G is an induced subgraph isomorphic to a cycle with at least four nodes, and an* antihole *is the complement of a hole. A graph $G = (V, E)$ is* alternately colourable *if the edges of G can be coloured with two colours such that no hole of G contains adjacent edges of the same colour.*

For the remainder of this section, let $\mathcal{I} = (G, \mathcal{V})$ be a given instance of (CPMCF) and $\mathcal{G}$ its dependency graph. Further, let $\bar{G} = (V, \bar{E})$ be the complement graph of G. We will first determine the structure of all holes in $\bar{G}$. With an almost canonical colouring it then follows that $\bar{G}$ is alternately colourable.

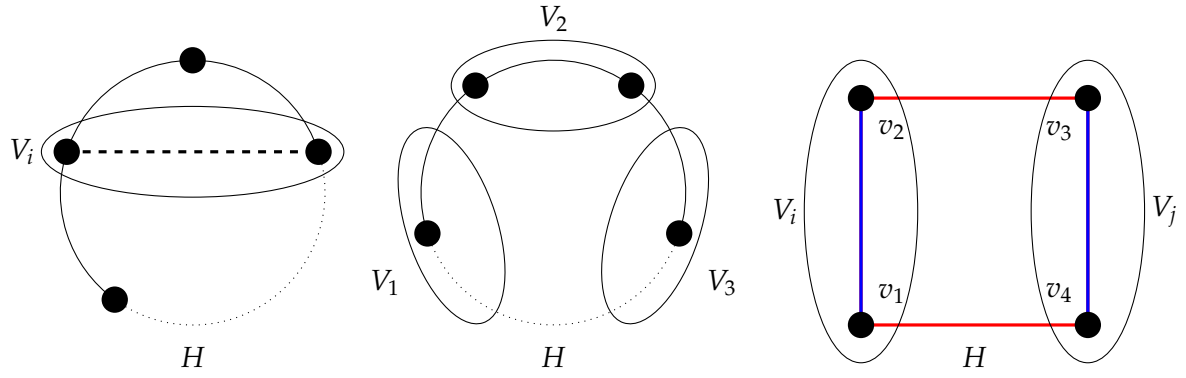
Lemma 3.4. *Any hole H in $\bar{G}$ is of length four, and $|V_i \cap H| = |V_j \cap H| = 2$ holds for some $V_i, V_j \in \mathcal{V}, i \neq j$.*

Proof. Suppose H is a hole in $\bar{G}$ and define $\mathcal{U} := \{V_i \in \mathcal{V} \mid V_i \cap H \neq \emptyset\}$, i.e. the subset of $\mathcal{V}$ containing all $V_i \in \mathcal{V}$ that have a non-empty intersection with the hole H. Since each $V_i \in \mathcal{V}$ is a clique in $\bar{G}$, any two nodes $u, v \in V_i \cap H, V_i \in \mathcal{V}$ must be neighbours in H, and we have $|V_i \cap H| \leq 2$ for all $V_i \in \mathcal{V}$. In other words, as Figure 3a illustrates, the hole H traverses each subset $V_i \in \mathcal{U}$ exactly once, and H intersects each V_i in at most two nodes.

After potentially renaming the elements of $\mathcal{V}$ accordingly, we can assume w.l.o.g. that H traverses $V_1, \ldots, V_{|\mathcal{U}|}$ in this order. Therefore, $\mathcal{E}$ must, by the definitions of $\bar{G}$ and $\mathcal{G}$, contain the edges $\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{|\mathcal{U}|-1}, V_{|\mathcal{U}|}\}, \{V_{|\mathcal{U}|}, V_1\}$. As Figure 3b illustrates, this immediately leads to a contradiction to $\mathcal{G}$ being a forest if $|\mathcal{U}| \geq 3$.

Hence, the hole H must be of length four, and $|V_1 \cap H| = |V_2 \cap H| = 2$. This completes the proof. □

Note that by Lemma 3.4, all holes in $\bar{G}$ must be of the form shown in Figure 3c as any other order would lead to a chord in H due to V_i and V_j being cliques in $\bar{G}$. By colouring "blue" all edges $\{u, v\}$ of $\bar{G}$ with $u, v \in V_i$ for some $V_i \in \mathcal{V}$ and colouring "red" all remaining edges, we then get, as Figure 3c further illustrates, the following result:



(a) Invalid intersection $V_i \cap H$ (b) H induces a cycle in $\mathcal{G}$ (c) Hole in $\bar{G}$ with colouring

Figure 3: Illustrations of the structure of a hole H in $\bar{G}$

Theorem 3.5. *$\bar{G}$ is alternately colourable.*

Together with the next two theorems, proved by [Hoà87] and [Lov72] respectively, this allows us to complete the proof of Theorem 3.1.

Theorem 3.6. *Alternately colourable graphs are perfect.*

Theorem 3.7. *The complement of a perfect graph is itself perfect.*

Proof of Theorem 3.1. From Theorem 3.5, Theorem 3.6 and Theorem 3.7, it directly follows that G is perfect as the complement of an alternately colourable graph.

Theorem 3.2 then yields that the polytope $P_0 := P_{Clique}(G)$ is described by the stable-set inequalities (3b) and the non-negativity constraints (3c). Moreover, each $V_i \in \mathcal{V}$ is a stable set in G, such that the inequalities $\sum_{v \in V_i} x_v \leq 1$, $i = 1, \ldots, m$, are valid for P_0 and therefore induce faces. Iteratively assuming equality in these constraints yields rational integral polyhedra $P_i := \{x \in P_{i-1} \mid \sum_{v \in V_i} x_v = 1\}$, $i = 1, \ldots, m$. In particular, P_m is a rational integral polyhedron.

It is left to show that P_m coincides with $P_{CPMCF}(\mathcal{I})$. Let $\bar{x} \in \{0,1\}^{|V|}$. This vector is contained in P_m if and only if it is the incidence vector of a clique C such that $|C \cap V_i| = 1$ for all $i \in \mathcal{V}$. The same holds for $P_{CPMCF}(\mathcal{I})$. As a result, we have $P_m = P_{CPMCF}(\mathcal{I})$, which completes the proof. □

It has been shown in [GLS84] that the weighted version of the clique problem on perfect graphs is solvable in polynomial time by an algorithm based on the ellipsoid method and semidefinite programming. This can also be applied to (CPMCF) since it is, as follows from Theorem 3.5, a special case of the clique problem on a perfect graph. However, this may not be an efficient approach in practice. In any case, this yields an alternative proof of Theorem 2.8, which stated that (CPMCF) is solvable in polynomial time.

Remark 3.8. *Lemma 3.4 directly implies that G does not contain an odd antihole as all holes in $\bar{G}$ are of length four. It is possible, but somewhat involved, to show that G cannot contain an odd hole either. Together with the Strong Perfect Graph Theorem (see [CRST06]), this would provide an alternative proof of the perfectness of G.*

4 Bounds for the Number of Facets

In this section, we will show that the facet-inducing stable-set inequalities of the (CPMCF)-polytope all correspond to stable sets whose nodes come from a single subgraph G_{ij}. This allows us to state an upper bound for the number of these facet-inducing stable sets. Additionally, we

will present a class of instances for which this bound is tight up to a constant factor, which leads to an exponential number of facets in the worst case. However, we will see that it is still possible to separate these facets efficiently.

4.1 An Upper Bound for the Number of Facets

In the following, let $\mathcal{I} = (G, \mathcal{V})$ be an instance of (CPMCF) with $G = (V, E)$ and $\mathcal{V} = \{V_1, \dots, V_m\}$. We begin our discussion by stating the following result for the general clique polytope, which has been proved in [Pad73].

Lemma 4.1. *The stable-set inequality*

$$\sum_{v \in S} x_v \leq 1$$

is facet-inducing for the general clique polytope if and only if $S \subseteq V$ is a maximal stable set in G.

The next lemma establishes the crucial fact that all facet-inducing stable-set inequalities in the case of (CPMCF) correspond to nodes from within a single subgraph G_{ij} each.

Lemma 4.2. *Let $S \subseteq V$ be a stable set in G. Then there exists a subgraph G_{ij} of G such that $S \subseteq V_i \cup V_j$.*

Proof. We prove this result by contradiction. Suppose $|\{V_i \in \mathcal{V} \mid V_i \cap S \neq \emptyset\}| \geq 3$. Let w.l.o.g. $\{V_1, V_2, V_3\} \subseteq \{V_i \in \mathcal{V} \mid V_i \cap S \neq \emptyset\}$. Then we can find elements $u \in V_1 \cap S$, $v \in V_2 \cap S$ and $w \in V_3 \cap S$ with $\{u, v\}, \{v, w\}, \{w, u\} \notin E$. This induces the edges $\{V_1, V_2\}, \{V_2, V_3\}, \{V_3, V_1\} \in \mathcal{E}$ in the dependency graph $\mathcal{G}$ of G. However, these edges form a cycle in $\mathcal{G}$, which is a contradiction. □

Put differently, we can fully describe the (CPMCF)-polytope using only stable-set inequalities corresponding to maximal stable sets in G, and each necessary stable set S in G is already contained in some subgraph G_{ij}. With these results in mind, we can find facet-inducing stable-set inequalities more efficiently and also deduce an upper bound for the number of required stable-set inequalities. The general idea is to create maximal stable sets from small subsets of a given $V_i \in \mathcal{V}$ and then lift a bound for the maximal size of these small subsets to a bound for the number of facet-inducing stable-set inequalities.

Definition 4.3. *Let G_{ij} be a subgraph of G and $U \subseteq V_i$. We define*

$$S_{ij}(U) := \{v \in V_i \mid N_j(\{v\}) \subseteq N_j(U)\} \cup (V_j \setminus N_j(U)).$$

Note that $S_{ij}(U)$ is, by construction, a maximal stable set in G_{ij}.

The following lemma shows that for all maximal stable sets S which are not equal to some subset in the partition $\mathcal{V}$, there exists a $U \subset V_i$, $V_i \in \mathcal{V}$, with $S_{ij}(U) = S$.

Lemma 4.4. *Let S be a maximal stable set in some subgraph G_{ij} of G with*

$$S \cap V_i \neq \emptyset \wedge S \cap V_j \neq \emptyset. \tag{5}$$

Then there exist subsets $U_i \subseteq V_i$ and $U_j \subseteq V_j$ with

$$S_{ij}(U_i) = S_{ji}(U_j) = S.$$

Proof. Let $U_i := V_i \cap S$ and $U_j := V_j \cap S$. Since S is a maximal stable set, we have $(V_j \setminus N_j(U_i)) = V_j \cap S$ and $\{v \in V_i \mid N_j(v) \subseteq N_j(U_i)\} = V_i \cap S$. Hence, $S_{ij}(U_i) = S$ holds by construction. Analogously, we get $S_{ji}(U_j) = S$. □

Remark 4.5. *Condition* (5) *in Lemma 4.4 serves two purposes:*

1. *It ensures that a maximal stable set S in G_{ij} with condition* (5) *is also maximal in G.*
2. *The stable-set inequality $\sum_{v \in S} x_v \leq 1$ corresponding to S is not redundant to the multiple-choice constraint corresponding to some V_i.*

Given a maximal stable set S satisfying condition (5), Lemma 4.4 ensures that there exist subsets $U_i \subseteq V_i$ and $U_j \subseteq V_j$ from which we can construct S. Hence, we can also find minimal subsets $U_i' \subseteq U_i$ and $U_j' \subseteq U_j$ with $S_{ij}(U_i') = S_{ji}(U_j') = S$. These minimal subsets now have an additional property that allows us to state an upper bound for the number of maximal stable sets S in G_{ij} satisfying condition (5). To describe this additional property, we need two more definitions.

Definition 4.6. *Within a subgraph G_{ij} of G, a set of nodes $U \subseteq V_i$* dominates *another set of nodes $W \subseteq V_i$ with respect to G_{ij} if $N_j(W) \subseteq N_j(U)$.*

Definition 4.7. *Let G_{ij} be a subgraph of G and $U \subseteq V_i$. Then we call U a* non-dominating set *in G_{ij} if there is no node $u \in U$ such that $\{u\}$ is dominated by $U \setminus \{u\}$ with respect to G_{ij}.*

The next lemma links the above definitions to maximal stable sets satisfying condition (5).

Lemma 4.8. *Let S be a maximal stable set in G_{ij} satisfying condition* (5)*. Further, let U be a minimal subset of V_i with $S_{ij}(U) = S$. Then U is a non-dominating set.*

Proof. Suppose U is not a non-dominating set. Then we can find $u \in U$ such that $\{u\}$ is dominated by $U \setminus \{u\}$. Consequently, we have $S_{ij}(U) = S_{ij}(U \setminus \{u\})$, since $N_j(U) = N_j(U \setminus \{u\})$. This is a contradiction to U being minimal. □

Since G is an undirected graph, we have $G_{ij} = G_{ji}$. If we can find upper bounds for the sizes of non-dominating subsets of V_i or V_j in G_{ij}, we can lift these to upper bounds for the number of maximal stable sets S in G_{ij} satisfying condition (5).

Lemma 4.9. *For each subgraph G_{ij} of G, let n_{ij} and n_{ji} be upper bounds for the sizes of non-dominating subsets of V_i and V_j respectively. Then there exist at most*

$$\min \left\{ \sum_{k=1}^{n_{ij}} \binom{|V_i|}{k}, \sum_{k=1}^{n_{ji}} \binom{|V_j|}{k} \right\}$$

different maximal stable sets S in G_{ij} satisfying condition (5)*.*

Proof. Let S be a maximal stable set in G_{ij} satisfying condition (5). Due to Lemma 4.4, we can find minimal sets $U_i \subseteq V_i$ and $U_j \subseteq V_j$ with $S = S_{ij}(U_i) = S_{ji}(U_j)$. Then U_i and U_j are non-dominating sets with respect to G_{ij} by Lemma 4.8.

Since the cardinality of a non-dominating set in V_i with respect to G_{ij} is, by assumption, less than or equal to n_{ij} and each non-dominating subset corresponds to a single maximal stable set, there can be at most

$$\sum_{k=1}^{n_{ij}} \binom{|V_i|}{k}$$

maximal stable sets S in G_{ij} satisfying condition (5). Analogously lifting the bound n_{ji} completes the proof. □

As the following theorem proves, summing up the bounds from Lemma 4.9 for all non-complete subgraphs G_{ij} yields an upper bound for the number of stable-set inequalities required to describe the (CPMCF)-polytope $P_{CPMCF}(\mathcal{I})$.

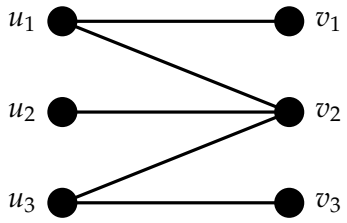


Figure 4: The stable-set inequality for the maximal stable set $S = \{u_2, v_1, v_3\}$ is not facet-defining, as it is a linear combination of the stable-set inequalities corresponding to $\{u_1, u_2, v_3\}$ and $\{u_2, u_3, v_1\}$ as well as the equation $x_{u_1} + x_{u_2} + x_{u_3} = 1$.

Theorem 4.10. *For each non-complete subgraph G_{ij}, $i < j$, let n_{ij} and n_{ji} be upper bounds for the sizes of non-dominating subsets of V_i and V_j respectively. Then the number of facets of the (CPMCF)-polytope $P_{CPMCF}(\mathcal{I})$ induced by stable-set inequalities is bounded from above by*

$$\sum_{\{V_i,V_j\}\in\mathcal{E}} \min\left\{\sum_{k=1}^{n_{ij}} \binom{|V_i|}{k}, \sum_{k=1}^{n_{ji}} \binom{|V_j|}{k}\right\}.$$

Proof. By Lemma 4.1 it suffices to include the stable-set inequalities corresponding to maximal stable sets in G. Moreover, a stable set S with $S \subseteq V_i$ yields a stable-set inequality which is redundant to the multiple-choice constraint corresponding to V_i. Together with Lemma 4.2, it follows that only maximal stable sets S in subgraphs G_{ij} satisfying condition (5) can induce facets of the (CPMCF)-polytope $P_{CPMCF}(\mathcal{I})$. Applying Lemma 4.9 to each non-complete subgraph G_{ij} completes the proof. □

There are cases in which it can be seen from the structure of an instance that non-dominating sets must be of small size. As an example, consider the following case: let $V_i := \{u_1, \ldots, u_n\}$ with the property $N_j(\{u_1\}) \subseteq N_j(\{u_2\}) \subseteq \ldots \subseteq N_j(\{u_n\})$ for a suitable ordering of the u_i. In this case, any $U \subseteq V_i$ is dominated by $\{u_{\max}\}$, where $u_{\max} \in U$ is the element with maximum index. Hence, non-dominating sets have size 1, and by Lemma 4.9, we obtain an upper bound for the number of facet-defining stable-set constraints that is linear in $\max\{|V_i| \mid V_i \in \mathcal{V}\}$ and $|\mathcal{V}|$. This structure is indeed present in the underground instance discussed in Section 5.

We emphasize that not all maximal stable sets in a subgraph G_{ij} necessarily induce facet-defining inequalities for the (CPMCF)-polytope $P_{CPMCF}(\mathcal{I})$. As a counterexample, consider the bipartite graph $G = (V, E)$ with $V := \{u_1, u_2, u_3, v_1, v_2, v_3\}$ and $E := \{\{u_1, v_1\}, \{u_1, v_2\}, \{u_2, v_2\}, \{u_3, v_2\}, \{u_3, v_3\}\}$ under the partition $\mathcal{V} = \{\{u_1, u_2, u_3\}, \{v_1, v_2, v_3\}\}$, see Figure 4. The set $S := \{u_2, v_1, v_3\}$ is a maximal stable set. However, its stable-set inequality is implied by those of the stable sets $\{u_1, u_2, v_3\}$, and $\{u_2, u_3, v_1\}$ together with the multiple-choice constraint for $\{u_1, u_2, u_3\}$. Indeed, the inequality $x_{u_2} + x_{v_1} + x_{v_3} \leq 1$ can be derived by adding the inequalities $x_{u_1} + x_{u_2} + x_{v_3} \leq 1$ and $x_{u_2} + x_{u_3} + x_{v_1} \leq 1$ and subtracting the multiple-choice constraint $x_{u_1} + x_{u_2} + x_{u_3} = 1$.

However, in the following we will give an example where the bound for the number of facet-defining inequalities from Theorem 4.10 is asymptotically tight up to a factor of 2.

4.2 A Worst-Case Example with an Exponential Number of Facets

In this section, we will show that the (CPMCF)-polytope can indeed have exponentially-many facets. To this end, we consider a specific class of instances which requires an exponential number of stable-set inequalities to completely describe the (CPMCF)-polytope.

Definition 4.11. *For a given $n \in \mathbb{N}$, $n \geq 2$, we define the (CPMCF)-instance $\mathcal{I}_{exp}$ as follows:*

$$\begin{aligned} V_1 &:= \{a, u_1, \dots, u_n\} \\ V_2 &:= \{b, v_1, \dots, v_n\} \\ V &:= V_1 \cup V_2 \\ E &:= \{\{a, v\} \mid v \in V_2\} \cup \{\{u, b\} \mid u \in V_1\} \cup \{\{u_i, v_i\} \mid i \in \{1, \dots, n\}\} \\ G &:= (V, E) \\ \mathcal{V} &:= \{V_1, V_2\} \\ \mathcal{I}_{exp} &:= (G, \mathcal{V}). \end{aligned}$$

The key to getting exponentially-many stable sets such that the resulting stable-set inequalities induce facets lies in the choice of E. The node a is connected to all elements in V_2, and the node b is connected to all elements in V_1. In combination with the ladder-like structure of the edges $\{u_i, v_i\}$, this ensures the existence of exponentially-many maximal stable sets in G corresponding to facet-inducing inequalities. Figure 5a shows the graph G for $n = 3$. For the remainder of this section, we consider $\mathcal{I}_{exp}$ for some fixed $n \in \mathbb{N}$, $n \geq 2$.

Definition 4.12. *Let $S \subseteq V$ be a stable set in G. We then write*

$$F_S := \left\{ x \in P_{CPMCF}(\mathcal{I}) \;\middle|\; \sum_{v \in S} x_v = 1 \right\}$$

for the face of $P_{CPMCF}(\mathcal{I})$ induced by the stable-set inequality corresponding to S.

The following lemma identifies a large class of subsets $U \subseteq V$ such that the faces $F_{S_{1,2}(U)}$ are facets of $P_{CPMCF}(\mathcal{I}_{exp})$.

Lemma 4.13. *Let $U \subsetneq \{u_1, \dots, u_n\}$ be a non-empty subset and*

$$\begin{aligned} S_{1,2}(U) &= \{u \in V_1 \mid N_2(\{u\}) \subseteq N_2(U)\} \cup (V_2 \setminus N_2(U)) \\ &= U \cup \{v_i \in V_2 \setminus \{b\} \mid u_i \notin U\} \end{aligned}$$

the corresponding maximal stable set as in Definition 4.3. Then $F_{S_{1,2}(U)}$ is a facet of $P_{CPMCF}(\mathcal{I}_{exp})$.

Proof. An example for $n = 3$ can be seen in Figure 5b. For ease of notation, we just write S for $S_{1,2}(U)$ in this proof. Note that S neither contains a nor b and contains exactly one of the two nodes u_i and v_i for each $i \in \{1, \dots, n\}$. We show that F_S is non-empty and that there exists a vector $\bar{x}$ such that the stable-set inequality $\sum_{v \in S} x_v \leq 1$ corresponding to S is the only inequality in the convex-hull description presented in Theorem 3.1 to separate $\bar{x}$ from $P_{CPMCF}(\mathcal{I}_{exp})$.

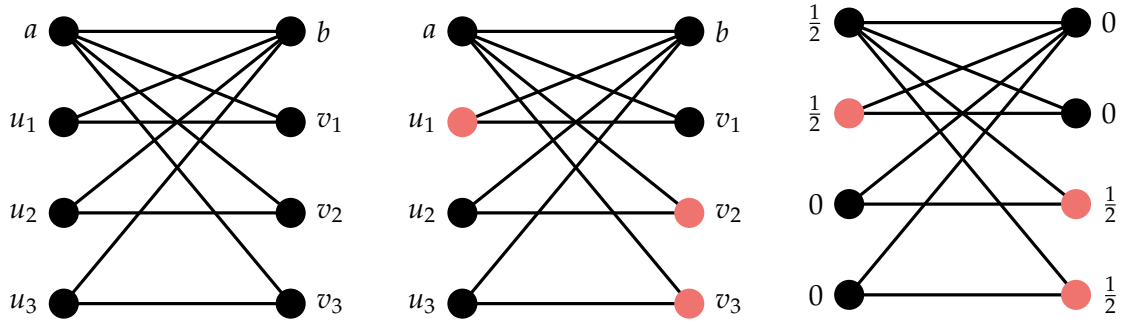
The face F_S is non-empty as it contains at least the incidence vectors of cliques C of the form $C = \{a, v_i\}$, $v_i \in V_2 \cap S$. Now define the vector $\bar{x} \in [0,1]^{|V|}$ with

$$\bar{x}_v := \begin{cases} \frac{1}{|S|-1}, & \text{if } v \in S \\ 1 - \frac{|S \cap V_1|}{|S|-1}, & \text{if } v = a \\ 1 - \frac{|S \cap V_2|}{|S|-1}, & \text{if } v = b \\ 0, & \text{otherwise.} \end{cases}$$

An example is given in Figure 5c. The vector $\bar{x}$ satisfies the multiple-choice constraints (3a) by construction. Furthermore, we have $|S \cap V_1| \leq |S| - 1$ and $|S \cap V_2| \leq |S| - 1$, and thus $\bar{x} \geq 0$ holds. Suppose there exists a stable set $S' \subseteq S$ with

$$\sum_{v \in S'} \bar{x}_v > 1.$$

The stable set S' cannot contain a or b as $\sum_{v \in V_1} \bar{x}_v = \sum_{v \in V_2} \bar{x}_v = 1$. Further, it has to contain all nodes $v \in S$ as otherwise $\sum_{v \in S'} \bar{x}_v \leq 1$ would hold. Finally, the stable set S is maximal by definition, and therefore we have $S' = S$. □



(a) Compatibility graph G for $n = 3$ (b) Maximal stable set *(red)* in G (c) Vector $\bar{x}$ for marked stable set

Figure 5: Illustration for $n = 3$. Figure 5a shows the initial graph G. Figure 5b shows a maximal stable set in G with the properties stated in Definition 4.12, and Figure 5c shows the vector $\bar{x}$ as specified in the proof of Lemma 4.13.

Lemma 4.14. *Let $U, U' \subseteq \{u_1, \dots, u_n\}$, $U \neq U'$, be non-empty subsets. Then $F_{S_{1,2}(U)}$ and $F_{S_{1,2}(U')}$ are different facets of $P_{CPMCF}(\mathcal{I}_{exp})$.*

Proof. W.l.o.g. let $u_1 \in U \setminus U'$. By Lemma 4.13, $F_{S_{1,2}(U)}$ and $F_{S_{1,2}(U')}$ induce facets of $P_{CPMCF}(\mathcal{I}_{exp})$. However, the incidence vector corresponding to $C = \{u_1, b\}$ is contained in $F_{S_{1,2}(U)} \setminus F_{S_{1,2}(U')}$, and hence $F_{S_{1,2}(U)} \neq F_{S_{1,2}(U')}$. □

We have proved that for each non-empty subset $U \subsetneq \{u_1, \dots, u_n\}$, the face $F_{S_{1,2}(U)}$ is a different facet of $P_{CPMCF}(\mathcal{I}_{exp})$. This allows us to prove the following theorem.

Theorem 4.15. *For $n \in \mathbb{N}$, $n \geq 2$, the polyhedron $P_{CPMCF}(\mathcal{I}_{exp})$ has exponentially-many facets.*

Proof. Let $U \subsetneq V_1$ be a non-empty proper subset of $\{u_1, \dots, u_n\}$. By Lemma 4.2, $S_{1,2}(U)$ is a maximal stable set in G, and by Lemma 4.13, $F_{S_{1,2}(U)}$ is a facet of $P_{CPMCF}(\mathcal{I}_{exp})$. For $U \neq U'$, we have $F_{S_{1,2}(U)} \neq F_{S_{1,2}(U')}$ by Lemma 4.14. Since there are $\sum_{k=1}^{n-1} \binom{n}{k} = 2^n - 2$ different non-empty proper subsets of $\{u_1, \dots, u_n\}$, the number of facets of $P_{CPMCF}(\mathcal{I}_{exp})$ is in $\Theta(2^n)$, whereas the number of variables and inequalities in $\mathcal{I}_{exp}$ is in $\mathcal{O}(n)$. □

As the number of required stable-set inequalities to describe the convex hull of (CPMCF) is exponential in general, their complete enumeration might not be feasible in practice. The natural approach is then to separate the stable-set inequalities, which is possible in polynomial time in a perfect graph (see [GLS84]). However, this algorithm for general perfect graphs is not combinatorial and numerically unstable and is therefore not well suited for practical use. In the case of (CPMCF), Lemma 4.2 tells us that we only have to separate stable-set inequalities induced by maximal stable sets within the bipartite subgraphs G_{ij}. This readily gives us an alternative algorithm for their separation: given a feasible solution $\bar{x}$ to the LP relaxation of model (1), we solve the maximum-stable-set problem within each subgraph G_{ij}, where the weight of a node v is given by the value $\bar{x}_v$. Solving the maximum-stable-set problem in a bipartite graph is possible via transformation to a maximum-flow problem (see e.g. [FF06]). The latter can be solved in time in $\mathcal{O}((|V_i| + |V_j|)^3)$ via the FIFO preflow-push algorithm (see [AMO93]). As the dependency graph is a forest for (CPMCF), there are $\mathcal{O}(|\mathcal{V}|)$ subgraphs G_{ij} to check. Assuming the nodes of V are evenly distributed across the partitions with n nodes per partition, which approximately is the case for the instances considered in Section 5, the whole separation is possible in time in $\mathcal{O}(|\mathcal{V}|n^3) = \mathcal{O}(|V|n^2)$.

5 Computational Results

In the following, we will show that our theoretical results on the (CPMCF)-polytope can lead to substantial computational benefits when applied to real-world problem settings. To this end, we compare the stable-set formulation to the naive formulation at the example of a timetabling problem arising in underground and railway traffic, where (CPMC) is used to model feasible timetables. For the stable-set formulation (3) we have enumerated the necessary stable-set constraints before optimization, using Lemma 4.2 to limit the enumerative effort. The naive formulation is handed to the solver as it is given in (1).

We will see that the stable-set formulation, which is based on the complete knowledge of the convex hull of the (CPMCF)-polytope, leads to significant reductions in computation time compared to the naive formulation. This even applies when the (CPMCF)-system of timetable constraints has a few cycles in its corresponding dependency graph. What is more, it even leads to a reduction in the number of constraints necessary to model the problem presented here, as the occurring non-dominating sets are relatively small.

For the implementation of our models, we have used the Python-API of *Gurobi 7.5.1* (see [Gur17]). The computations have been performed on a server with Intel Xeon E5-2690 3.00 GHz processors with 25 MB cache and 128 GB RAM. Each computation has been run on 5 threads with a time limit of 10 hours; other than that, we have used Gurobi's standard parameter settings throughout.

5.1 An Application from Energy-Efficient Timetabling

We will perform our computational study on the following benchmark problem, which is an interesting combination of timetable optimization and minimization of energy consumption. Given an initial timetable draft, the task is to adjust the departure times of the trains in the stations and their velocity profiles on the lines in order to minimize the total energy consumption during the planning horizon while maintaining the basic structure of the timetable draft. More precisely, we allow to shift the departure times within small intervals around the departure times stated in the draft and allow the choice of suitable velocity profiles from a discrete set of given alternatives for each train on a given line. The choice of a different velocity profile naturally has an effect on the travel time on a line, such that dwell times in the stations might need to be adjusted as well. To ensure that passengers have enough time to get on and off the train, we require adequate minimum dwell times. With respect to safety, we enforce suitable minimum headway times between consecutive trains on the same track. In doing so, we require that the trains in the optimized timetable pass the tracks in the same order as in the draft, i.e. the order is not subject to optimization. This is a significant simplification of the problem, but not overly conservative given the small scope of our changes to the timetable draft.

The power consumption of each train during its travel is measured in each second. A braking train can feed energy back into the power network, which translates into a negative consumption. This recuperated energy can be used by other trains in the network. However, if there is no other train requiring energy in that moment, the recuperated energy is lost. Therefore, the aspired reduction in total energy consumption stems from two effects: energy-efficient driving profiles for each individual train and a synchronization of braking trains with accelerating trains. Reducing the energy consumption of a train generally corresponds to driving more slowly. Of course this has to be balanced with passenger service. Indeed, we will be able to show that alterations at a scale hardly noticeable by the passenger already lead to major reductions in energy consumption.

In our description of the problem, we use the following expressions: a journey of a single train from one station to the next one to be scheduled is called a *leg*. A possible combination of departure time and travel time for a given leg is called a *departure configuration*. In short, the

task can be described as choosing a departure configuration for each leg, such that the timetable remains feasible and the total energy consumption in the system is minimized.

For further background on energy-efficient railway timetabling based on combinatorial optimization, we refer the interested reader to the following works in the literature. In [BGMS18], the authors present a simpler variant of the above problem, where departure times may be shifted but travel times and power profiles are fixed. It is treated as an application of (CPMC) under staircase compatibility, a case that is much simpler to solve due to the more compact complete convex-hull description that it allows. The preceding work [BMS17] compares different objective functions for the same problem (without exploiting the staircase property) with respect to the power consumption patterns they produce as well as the computational complexity they entail. [SG97] develops a decomposition heuristic for a similar problem and performs a case study for the underground system of Montréal, Canada. Indicative examples of approaches based on non-linear optimization, among others, are given in the broad survey [SGK17] while state-of-the-art approaches for general railway timetabling can be found in [CT12]. Finally, we remark that the (CPMC)-structure is not only found in railway timetabling but also in runway scheduling, see e.g. [ABMV17].

We will proceed by explaining the underlying timetabling and energy consumption model and will then come to the description of several real-world instances, one from underground traffic and the other ones from railway traffic. The first instance is an example where the constraints for a feasible timetable have (CPMCF)-structure, while they are only near-(CPMCF) for the second type. Finally, we will see that our improved formulation for (CPMCF) significantly outperforms the naive formulation from the beginning.

5.2 Mathematical Model

Let the set T be the planning horizon with a resolution in seconds. During this planning horizon, we have a set of trains R travelling through a network. For each train $r \in R$, we have a set A^r representing the legs it travels. Any such leg $a \in A := \bigcup_{r \in R} A^r$ has a set of permitted departure times J^a and a set of admissible travel times D^a. Furthermore, the set W contains all pairs of directly successive legs of the same train. For any $(a_1, a_2) \in W$, the minimum dwell time between a train completing leg a_1 and departing for the subsequent leg a_2 is denoted by $c^{a_1 a_2}$. In a similar fashion, the set L contains all pairs of legs of two different trains passing the same track in direct succession. For a pair $(a_1, a_2) \in L$, the trains on these two legs have to meet a safety distance which is given by the minimum headway time $s^{a_1 a_2}$. We assume that it is sufficient to enforce this minimum headway between the respective times of departure and arrival of the two trains.

The set of feasible timetable adaptations can now be represented as an instance of (CPMC) by defining a suitable graph $G = (V, E)$ which models the compatibilities between the departure configurations of the legs to be scheduled. For each such departure configuration $(j, d) \in J^a \times D^a$ for a leg $a \in A$, we introduce a corresponding node in the graph. The set of all nodes V is thus given by

$$V := \{(a, j, d) \mid a \in A, j \in J^a, d \in D^a\}.$$

For ease of exposition, we describe E via the set of edges $\bar{E}$ of the complement graph $\bar{G}$ of G, which models incompatibilities between the departure configurations of the legs. In order to choose only one departure configuration per leg $a \in A$, we have to define the edge set

$$\bar{E}^a_{MC} := \{\{(a, j_1, d_1), (a, j_2, d_2)\} \mid j_1, j_2 \in J^a, d_1, d_2 \in D^a, (j_1, d_1) \neq (j_2, d_2)\}.$$

For each pair of successive legs $(a_1, a_2) \in W$ of the same train, we have to respect the minimum

dwell time constraints, which leads to the edges

$$\bar{E}_{DT}^{a_1a_2} := \{\{(a_1, j_1, d_1), (a_2, j_2, d_2)\} \mid j_1 \in J^{a_1}, j_2 \in J^{a_2}, d_1 \in D^{a_1}, d_2 \in D^{a_2}, \\ j_2 < j_1 + d_1 + c^{a_1a_2}\}.$$

Finally, for any two trains with a pair of successive legs $(a_1, a_2) \in L$ on the same track, we have to meet the minimum headway times. This leads to the following edges:

$$\bar{E}_{HW}^{a_1a_2} := \{\{(a_1, j_1, d_1), (a_2, j_2, d_2)\} \mid j_1 \in J^{a_1}, j_2 \in J^{a_2}, d_1 \in D^{a_1}, d_2 \in D^{a_2}, \\ j_2 < j_1 + \max\{d_1 - d_2, 0\} + s^{a_1a_2}\}.$$

Combining these three types of edges yields the edge set $\bar{E}$ of the complement graph $\bar{G}$:

$$\bar{E} := \bigcup_{a \in A} \bar{E}_{MC}^{a} \cup \bigcup_{(a_1,a_2) \in W} \bar{E}_{DT}^{a_1a_2} \cup \bigcup_{(a_1,a_2) \in L} \bar{E}_{HW}^{a_1a_2}.$$

The graph $G = (V, E)$ now represents the compatibilities between the departure configurations of the legs to be scheduled. By definition of $\bar{E}_{MC}^{a}$, each set $V^a \subseteq V$ containing all departure configurations of the leg $a \in A$ is a stable set in G. Accordingly, the partition $\mathcal{V}$ of V is given by

$$\mathcal{V} := \{V^a \mid a \in A\}.$$

This yields the (CPMC)-instance $\mathcal{I} = (G, \mathcal{V})$, whose feasible set we denote by $X(\mathcal{I})$. The decision variable $x_{jd}^{a} \in \{0, 1\}$ then models the choice of a node $(a, j, d) \in V$ and consequently the choice of the departure configuration $(j, d) \in J^a \times D^a$ for the leg a. Furthermore, the multiple-choice constraints require the choice of exactly one departure configuration per leg, and the edges E represent the compatibilities between the departure configurations.

It remains to model the objective function representing the total energy consumption during the planning horizon. Let the power consumption of train $r \in R$ travelling a leg $a \in A$ with departure configuration $(j, d) \in J^a \times D^a$ in second $t \in T$ be given by p_{jd}^{at}. This allows to compute the total power consumption $P(x, t)$ induced by a feasible timetable $x \in X(\mathcal{I})$ in second $t \in T$:

$$P(x, t) = \sum_{a \in A} \sum_{j \in J^a} \sum_{d \in D^a} p_{jd}^{at} x_{jd}^{a}.$$

As recuperated energy that is not immediately used is lost, we have to cut off the power consumption in each second at zero when computing the total energy consumption over the planning horizon. Altogether, this leads to the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{t \in T} \max(P(x, t), 0) \\ \text{s.t.} \quad & x \in X(\mathcal{I}). \end{aligned}$$

In order to make it accessible for an MIP solver like Gurobi, we linearize the objective function by introducing continuous auxiliary variables:

$$\begin{aligned} \min \quad & \sum_{t \in T} z_t & & \\ \text{s.t.} \quad & z_t & \geq \quad P(x, t) \qquad & (\forall t \in T) \\ & x & \in \quad X(\mathcal{I}) & \\ & z_t & \geq \quad 0 \qquad & (\forall t \in T). \end{aligned}$$

In this final model, which we use for all subsequent computations, variables z_t represent the net energy consumption in the system at each instant $t \in T$. Overall, the presented timetabling problem amounts to optimizing a convex piecewise-linear objective function over the (CPMC)-polytope. For the latter, we will compare the computational performance of the naive formulation versus the stable-set formulation. Note that this requires the x-variables to be binary in both formulations, even when the dependency graph is cycle-free.

5.3 Description of the Instances

The data sets for this computational study have been provided by two industry partners with whom we work together on energy-efficient timetabling. The data on underground traffic comes from VAG, the operator of public transport in the city of Nürnberg, Germany, while the data on railway traffic comes from Deutsche Bahn AG, the most important operator of railway traffic in Germany.

5.3.1 The Underground Instance

Our partner VAG has provided us with the 2017 schedules of the Nürnberg underground system and adequate values for the remaining schedule-relevant parameters as well as typical data on the energy consumption of the trains. For this computational study – intended to estimate the potential of the overall approach – we restrict ourselves to the longest line in the system, which is U1, and to the morning rush hour interval between 5 a.m. and 9 a.m. The number of trains running in this time frame is 124, with a total of 2581 departures. Legs crossing the boundaries of the planning horizon were fixed to the configuration of the current timetable while interchange times between underground lines were disregarded.

We allow departure time shifts of up to ± 15 seconds around the current departure time for each leg, going in increments of 5 seconds, and allow the model to choose from three different travel times per leg. The shortest of the three travel times is the one where the train drives as fast as possible for as long as possible. The intermediate travel time is the current travel time or, if this coincides with the shortest travel time, 1.5% slower. The longest travel time is 3% longer than the current travel time if the latter does not coincide with the shortest travel time and 4.6% longer if it does. To compute the corresponding velocity and power consumption profiles, we used an optimization model to solve the underlying optimal control problem of driving the given distance in the given time as energy-efficient as possible, based on typical underground train characteristics. Note that this problem itself is not easy to solve, so we used a heuristic to accomplish the task in acceptable time. Finally, we assume that each possible travel time is compatible with each possible departure time, as long as all timetabling constraints are met.

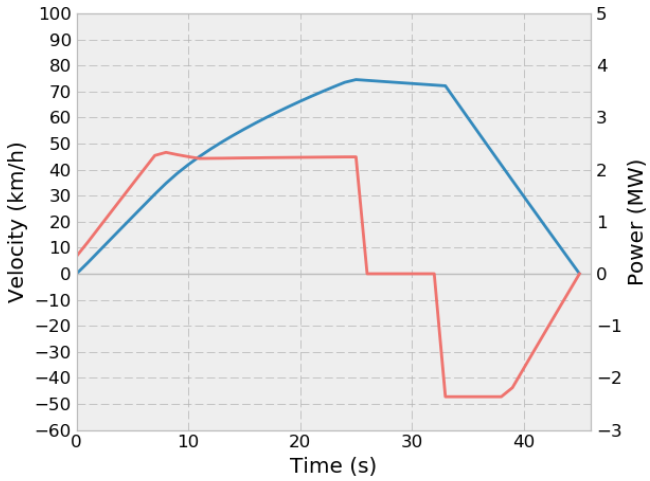


Figure 6: Example diagram for velocity (*blue*) and power consumption profiles (*red*)

Figure 6 shows the velocity and power consumption profiles on a leg with a length of 594 m driven within 45 seconds. As we can see, we have a positive power consumption during the acceleration phase between 0 and 25 seconds, no power consumption during the coasting phase between 25 and 32 seconds and energy recuperation during the braking phase.

Table 1 gives statistics on the average fastest, intermediate and slowest profiles over all legs in the planning horizon. The average intermediate profile is about 2.4% slower than the fastest

Average profile statistics	Fastest profile	Intermediate profile	Slowest profile
Travel time	55.49 s	56.82 s	58.89 s
Gross energy consumption	12.62 kWh	10.96 kWh	9.82 kWh
Recuperated energy	5.14 kWh	4.45 kWh	3.97 kWh

Table 1: Profile statistics of the *Nürnberg* underground instance

profile and consumes about 13% less energy. Using the slowest profile takes about 6.1% longer than the fastest one while its consumption is about 22% lower. In all cases, about 40% of the energy can be recuperated. The typical profile in the current schedule is the intermediate one.

In the underground instance, the headway time constraints are redundant due to a relatively low amount of trains in the system. This makes the dependency graph a forest as it is only determined by the minimum dwell time constraints. Furthermore, non-dominating sets only have size 1, as any departure configuration (j_1, d_1) for some leg a_1 that is compatible to a departure configuration (j_2, d_2) for the subsequent leg a_2 of the same train is also compatible to all configurations $(j_2', d_2') \in J^{a_2} \times D^{a_2}$ with $j_2' \geq j_2$. As a consequence, the number of maximal stable-set constraints is linear in the number of departure configurations, and they can easily be enumerated.

5.3.2 The Railway Instances

The railway instances we consider in the following are based on data provided by Deutsche Bahn AG which contains the 2015 timetable for the passenger traffic operated by them as well as typical train characteristics. We highlight the results for one instance, *Nürnberg*, which has roughly as many departures as the *Nürnberg* underground instance, and then present a summary of the results for 30 more instances of different sizes. The scope of the railway instances we created was the railway traffic – regional and long-distance traffic – passing through certain subnetworks of Germany between 5 a.m. and 9 a.m. Here we considered larger shifts in the departure times than in the underground instance. We assumed that train departure times could be shifted by up to ± 180 seconds around the current departure time in both directions in increments of 60 seconds. Furthermore, train profiles were not as sophisticated in this case, as they only contained one acceleration phase followed by a cruising phase and then a braking phase, not necessarily driving in an energy-optimal fashion.

In the railway instances, the headway time constraints are not redundant, but the dependency graph typically still has relatively few cycles, and the non-dominating sets corresponding to the headway time constraints have a size of at most 3. The latter stems from two facts. Firstly, we can choose among three velocity profiles for each leg. Secondly, any departure configuration (j_1, d_1) for some leg a_1 of a given train that is compatible to a departure configuration (j_2, d_2) for the leg a_2 of the train directly following on the same track is also compatible to all configurations $(j_2', d_2') \in J^{a_2} \times D^{a_2}$ with $j_2' \geq j_2$ and $d_2' \geq d_2$. As a result, the number of stable-set constraints in the stable-set formulation is at most cubic per subgraph G_{ij}, which is still tractable to enumerate.

Note that these railway instances are closely related to those used in [BMS17] as they are derived from the same timetable and cover the same time frame in the morning. However, here these instances have been significantly extended. While in [BMS17] the decision space only included shifting the departure times of the stations, it now also includes the choice of driving profile for each train on each of its legs. This does not only enlarge the corresponding

compatibility graph, it also makes its structure fundamentally more complex. Adjusting only the departure times leads to the special case (CPMCS), which allows a linear-size complete description of the feasible set and is therefore typically much simpler to solve. A detailed analysis of this special case can be found in [BGMS18], where the same basic data has been used to show that shifting departures can be solved for very large-scale instances spanning a planning horizon of a whole day. Altogether, from the above discussion, it can be expected that the instances used here are much more difficult to solve than those in the two mentioned publications.

5.4 Computational Results for the Underground Instance

We now compare the performance of our two (CPMCF)-formulations for the underground instance we created. Table 2 shows the size of the two models after preprocessing. Clearly, the

Model statistics	Naive	Stable set
Model construction time	21.29 s	27.78 s
#total variables	66,368	66,414
#continuous variables	14,606	14,606
#binary variables	51,762	51,808
#constraints	70,225	26,659
#non-zeros	2,358,430	1,889,765

Table 2: Model construction time of the *Nürnberg* underground instance and size after preprocessing

number of variables is about the same for both models, while the number of constraints is significantly lower for the stable-set formulation. To understand this, recall that both models are formulated in the same variables and have the same constraints for the linearization of the objective function as well as the same multiple-choice constraints. As the headway constraints are redundant for this instance, the difference comes from a much lower number of dwell time constraints needed for the stable-set formulation. This is because of the small size of non-dominating sets in this instance, which we already pointed out, as well as the fact that the stable-set formulation groups departure configurations according to common compatibilities to form joint inequalities. Gurobi's preprocessing then is about equally efficient for both formulations when it comes to reducing the number of variables and constraints. We remark here that enumerating all relevant stable-set constraints for the corresponding formulation is not an overly high effort, as is visible from the comparison of model construction times. Indeed, for both formulations, the largest part of the effort goes into the construction of the objective function, not the timetabling contraints. We will see that the same holds for all considered instances of the problem, which is why we did not consider the model construction times as part of the solution time of an instance; especially they did not count towards the time limit.

Table 3 gives an overview on the solution progress for the instance in both formulations. It shows that solving the LP relaxation of either model can be done in well under a minute, where Gurobi's barrier algorithm yielded the best result in both cases. Here we already see a computational advantage of the more compact stable-set formulation, which only takes one fourth of the time the naive formulation needs. Looking at the optimal objective value of the LP relaxation, we can see that the stable-set formulation also provides a tighter bound. Indeed, this initial bound is already about as good as the best bound the naive formulation produces within 10 hours of computation. Although neither formulation enabled us to solve the in-

	Naive	Stable set
LP solution time	28.13 s	6.21 s
LP bound	15.320 MWh	15.694 MWh
Bound after 1 h	15.520 MWh	15.699 MWh
Objective after 1 h	15.942 MWh	15.805 MWh
Gap after 1 h	2.64%	0.67%
Bound after 10 h	15.696 MWh	15.702 MWh
Objective after 10 h	15.817 MWh	15.771 MWh
Gap after 10 h	0.77%	0.44%

Table 3: The solution progress of the *Nürnberg* underground instance

stance to optimality within the time limit, the stable-set formulation is very good at reducing the optimality gap quickly. We observed that 4 minutes into the optimization, the stable-set formulation already obtains a gap below 1%, while the naive formulation takes over 6 hours to achieve this. The best solution found by the stable-set formulation within 1 hour is already better than the best solution found via the naive formulation within 10 hours. At the point when the time limit is reached we see that the two formulations reach comparable results, with a certain advantage on the side of the stable-set formulation.

In Figure 7, we compare the current timetable (left) and the best solution found by the stable-set formulation (right) in terms of energy consumption. The *blue* curve shows the total power

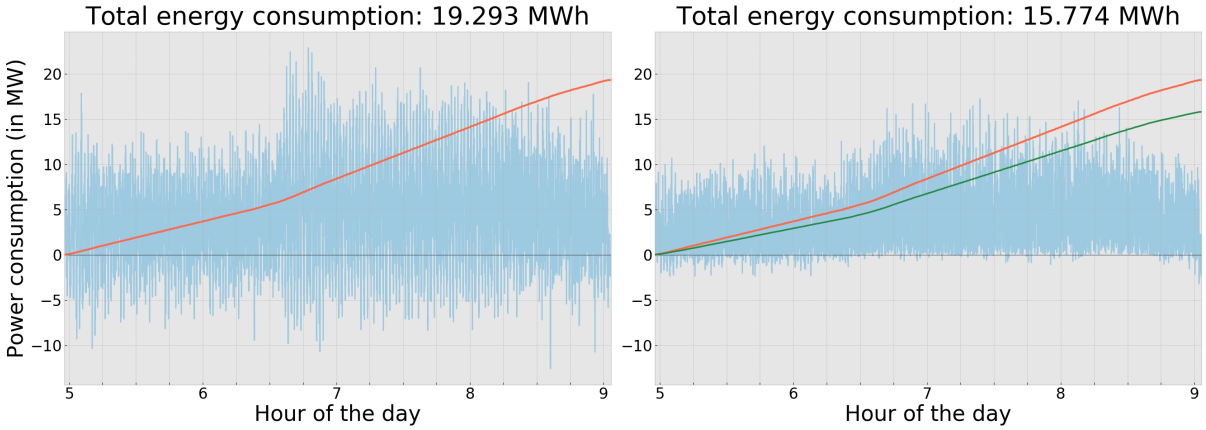


Figure 7: A comparison of the underground timetable before (left) and after optimization (right). The power consumption in each second is shown in *blue*, the total energy consumption over time before optimization is shown in red, after optimization in *green*.

consumption of all trains in each second while the *red* curve shows the accumulated energy consumption up to a given point in time, starting just before 5 a.m. In the right image, this is accompanied by the accumulated energy consumption after optimization (green). Note that negative total power consumptions correspond to lost recuperation energy, which is therefore not subtracted when computing the accumulated energy consumption. The diagrams clearly show that the optimized solution reduces the overall energy consumption by about 18%, which illustrates the high potential of this approach. This effect is achieved by a reduction in gross energy consumption as well as a better exploitation of recuperated energy. On the right-hand side we see that peak power consumption values are significantly lower while the same applies

to losses in recuperated energy.

We remark that these improvements have been achieved with only minor prolongations in travel time for the passenger. The average travel time on a leg increases from 57.0 seconds to 58.5 seconds, which is partly offset by the average decrease in station dwell time from 26.7 seconds to 25.9 seconds. The average overall extension of a journey in the system is thus only 0.7 seconds per travelled leg. As could have been expected, judging from the goal of better synchronization of accelerating trains with braking trains, there are about as many preponed as postponed departures.

5.5 Computational Results for the Railway Instances

In this section, we briefly review the results on the railway instances as an example for non-(CPMCF)-instances. As for the underground instance, we begin with similar statistics for the *Nürnberg* railway instance on the problem size after preprocessing in Table 4. Another effect

Model statistics	Naive	Stable set
Model construction time	81.31 s	91.01 s
#total variables	61,249	61,507
#continuous variables	13,748	13,749
#integer variables	651	650
#binary variables	47,501	47,108
#constraints	89,216	31,906
#nonzeros	15,863,815	15,571,881

Table 4: Model construction time of the *Nürnberg* railway instance and size after preprocessing

that carries over from the underground instance is that the number of inequalities necessary to represent the dwell time constraints is much lower in the stable-set formulation. For both formulations, the number of headway time constraints is below 1000, and the efficiency of Gurobi's preprocessing again is about equal. Interestingly, Gurobi finds it beneficial to discretize some of the continuous variables, such that we also find a few integer variables in the preprocessed model. Note that although the number of variables and constraints lies in the same order as for the underground instance, the number of nonzeros is much higher as the legs have longer travel times, which leads to a much denser linearization of the objective function. Thus, we again have similar model construction times for both formulations.

Table 5 shows the most important statistics from the solution progress of the *Nürnberg* railway instance. The LP relaxation of this instance has been solved fastest with the primal simplex algorithm for both formulations. We see that this can be done more than 6 times faster for the stable-set formulation than for the naive formulation. After just 4 minutes, the stable-set formulation has enabled Gurobi to solve the problem to its default optimality tolerance of 0.01%. In contrast, the solution of the naive formulation has made much less progress as Gurobi still shows an optimality gap of more than 20%, which is reduced to less than 1% within about half an hour. Even after reaching the time limit of 10 hours, optimality could not be fully proven for the instance via the naive formulation, with a remaining optimality gap of 0.02%. Note that the railway instance was easier to solve overall as with more trains in the system, minimizing the loss of recuperated energy becomes less complex.

In Table 6, we give an overview of all the 31 railway instances we created. They vary markedly in size – from small stations with few passing trains, over bigger regions of Germany

	Naive	Stable set
LP solution time	200.20 s	34.02 s
LP bound	108.540 MWh	111.225 MWh
Bound after 4 min	108.540 MWh	111.226 MWh
Objective after 4 min	140.229 MWh	111.237 MWh
Gap after 4 min	22.60%	<0.01%
Bound after 10 h	111.157 MWh	-
Objective after 10 h	111.208 MWh	-
Gap after 10 h	0.02%	-

Table 5: The solution progress of the *Nürnberg* railway instance

and the German long-distance traffic up to the complete German passenger traffic. The upper section of the table shows instances based on the traffic through individual stations, the second section shows the traffic within bigger regions of Germany, and the last section shows nationwide traffic: long-distance traffic (*Fernverkehr*), short-distance traffic (*Regionalverkehr*) and both combined (*Deutschland*). It becomes clear at first sight that the stable-set formulation yields the better performance for each single instance. While most instances were not solved to optimality within the time limit by either formulation, the stable-set formulation solves two instances more, has vastly shorter solution times for the instances solved by both formulations and has solved twice as many instances to an optimality gap below 1%. From the last column, we see that the energy consumption implied by the timetable can be reduced substantially in almost all cases. Only for the largest instance, we could not improve on the initial solution within the considered time limit. We remark that compared to the time limit we used, model construction times for the two models were neglectable for all instances except for the largest two. For those, they were between 30 and 45 minutes for both models.

Altogether, our results show that there is significant benefit in exploiting the (CPMCF)-(sub)structures in this timetabling problem. Typically, the overall size of the model decreases, the bounds improve, finding good solutions early is facilitated, and the overall solution time decreases. One thing we have to point out is that enumeration of the necessary stable-set constraints in the corresponding formulation sometimes means a noticeable overhead in the model building time, as it actually amounts to subsuming the complexity of the stable-set separation problem under model construction. However, the results from Section 4.1 help to limit the enumerative effort necessary. It can be expected that separating the constraints of the stable-set formulation will further speed up the solution process.

Instance	#Trains	#Legs	Computation time or gap Naive	Stable set	Energy savings [%]
Bayreuth Hbf	14	71	7.45%	**5.95%**	33.85%
Zeil	15	165	23.28%	**20.38%**	35.30%
Passau	23	220	7.26%	**5.80%**	29.93%
Jena Paradies	25	244	0.49%	**0.40%**	20.74%
Lichtenfels	33	380	1.19%	**0.67%**	24.79%
Erlangen	45	717	0.99%	**0.43%**	24.24%
Bamberg	58	776	0.58%	**0.37%**	23.16%
Aschaffenburg	72	823	0.13%	**0.09%**	19.37%
Kiel Hbf	80	458	0.44%	**0.36%**	25.21%
Leipzig Hbf (tief)	95	1514	5.89%	**1.76%**	30.73%
Würzburg Hbf	106	928	30750.99 s	**1597.59 s**	19.98%
Dresden	123	1603	0.62%	**0.17%**	22.58%
Ulm Hbf	134	1357	0.09%	**0.04%**	19.80%
Berlin Hbf (S-Bahn)	172	3696	20.35%	**1.47%**	21.19%
Stuttgart Hbf (tief)	185	3083	66.14%	**6.76%**	64.29%
Hamburg-Altona(S)	186	2932	39.95%	**3.47%**	39.38%
Frankfurt(Main)Hbf	231	2042	26710.23 s	**465.12 s**	16.87%
Nürnberg	276	2903	0.02%	**212.44 s**	20.67%
S-Bahn Hamburg	320	4208	26.91%	**1.31%**	39.44%
Regio Nord	410	3155	1.27%	**0.04%**	21.23%
Regio Nordost	428	3988	2.14%	**0.08%**	18.11%
Regio Hessen	460	6439	13.50%	**0.23%**	30.47%
Regio Südwest	535	5961	4.58%	**0.11%**	22.44%
Regio Südost	666	7659	4.76%	**0.04%**	24.11%
Regio BW	690	7744	9.69%	**0.10%**	34.81%
S-Bahn Berlin	688	12558	15.23%	**0.21%**	22.17%
Regio NRW	827	11316	7.95%	**0.05%**	25.58%
Regio Bayern	1015	12246	6.74%	**0.03%**	28.06%
Fernverkehr	231	1260	5027.4 s	**205.83 s**	17.02%
Regionalverkehr	6041	75287	100.0%	**33109.53 s**	23.92%
Deutschland	6272	76547	100.00%	**20.89%**	0.00%

Table 6: Computational results for the two problem formulations for the railway instances

6 Conclusions

In this paper, we have discussed theoretical aspects of the clique problem with multiple-choice constraints under a cycle-free dependency graph (CPMCF) and have demonstrated its practical relevance. After showing the NP-completeness of the general clique problem with multiple-choice constraints (CPMC), we gave a formal definition of the new special case (CPMCF) and differentiated it from the known polynomial-time solvable special case under staircase compatibility (CPMCS) from [BGMS18]. Via a study of the underlying compatibiliy graph – which turned out to be perfect – we could derive a full description of the (CPMCF)-polytope and could determine that (CPMCF) is in P as well. We then provided necessary criteria for facet-inducing stable-set inequalities as well as a non-trivial exponential upper bound for their total number via so-called non-dominating sets. In a worst-case example, we showed that an exponential number of facets can actually be attained and outlined an efficient separation algorithm for the stable-set inequalities.

Applying our theoretical results to the problem of energy-efficient timetabling for underground and railway systems, we found that exploiting the knowledge of the complete convex hull of (CPMCF) can lead to significantly lower computation times.

Regarding further research on theoretical aspects, we may ask whether the special cases (CPMCS) and (CPMCF) can be further generalized to other subclasses of (CPMC) that are still in P. An interesting case might be series-parallel or bounded-treewidth dependency graphs. In these cases, G is not perfect in general, but dynamic-programming algorithms similar to Algorithm 1 might still be applicable, enabling compact extended formulations. On the application side, there can be further study on how the stable-set formulation performs on other real-world problems, e.g. runway scheduling.

References

[ABMV17] Pasquale Avella, Maurizio Boccia, Carlo Mannino, and Igor Vasilyev. Time-indexed formulations for the runway scheduling problem. *Transportation Science*, 51(4):1196–1209, 2017.

[ABZ07] Ron Aharoni, Eli Berger, and Ran Ziv. Independent systems of representatives in weighted graphs. *Combinatorica*, 27(3):253–267, May 2007.

[AH00] Ron Aharoni and Penny Haxell. Hall's theorem for hypergraphs. *J. Graph Theory*, 35(2):83–88, October 2000.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, Inc., 1993.

[BBPP99] Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. *The Maximum Clique Problem*, pages 1–74. Springer US, Boston, MA, 1999.

[Ber63] Claude Berge. Some classes of perfect graphs. *Six Papers on Graph Theory [related to a series of lectures at the Research and Training School of the Indian Statistical Institute, Calcutta*, pages 1–21, 1963.

[BGMS18] Andreas Bärmann, Thorsten Gellermann, Maximilian Merkert, and Oskar Schneider. Staircase compatibility and its applications in scheduling and piecewise linearization. *Discrete Optimization*, 29:111–132, 2018.

[BMS17] Andreas Bärmann, Alexander Martin, and Oskar Schneider. A comparison of performance metrics for balancing the power consumption of trains in a railway network by slight timetable adaptation. *Public Transport*, 9(1-2):95–113, 2017.

[CRST06] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006.

[CT12] Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

[FF06] Ulrich Faigle and Gereon Frahling. A combinatorial algorithm for weighted stable sets in bipartite graphs. *Discrete Applied Mathematics*, 154(9):1380–1391, 2006.

[GIZ+02] Tore Grünert, Stefan Irnich, Hans-Jürgen Zimmermann, Markus Schneider, and Burkhard Wulfhorst. Finding all k-cliques in k-partite graphs, an application in textile engineering. *Computers & operations research*, 29(1):13–31, 2002.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[GJS74] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 47–63, New York, NY, USA, 1974. ACM.

[GLS84] Martin Grötschel, Laszlo Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, 21:325–356, 1984.

[GLS88] Martin Grötschel, Laszlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

[Gur17] Gurobi Optimization, Inc. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2017.

[Hoà87] Chính T Hoàng. Alternating orientation and alternating colouration of perfect graphs. *Journal of Combinatorial Theory, Series B*, 42(3):264–273, 1987.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[Kin11] Andrew D. King. Hitting all maximum cliques with a stable set using lopsided independent transversals. *Journal of Graph Theory*, 67(4):300–305, 2011.

[Kro67] M. R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.

[LM16] Frauke Liers and Maximilian Merkert. Structural investigation of piecewise linearized network flow problems. *SIAM Journal on Optimization*, 26(4):2863–2886, 2016.

[Lov72] László Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95–98, 1972.

[MK13] Mohammad Mirghorbani and P Krokhmal. On finding k-cliques in k-partite graphs. *Optimization Letters*, 7(6):1155–1165, 2013.

[Pad73] Manfred W. Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[SG97] Brunilde Sansó and Pierre Girard. Instantaneous power peak reduction and train scheduling desynchronization in subway systems. *Transportation science*, 31(4):312–323, 1997.

[SGK17] Gerben M. Scheepmaker, Rob M. P. Goverde, and Leo G. Kroon. Review of energy-efficient train control and timetabling. *European Journal of Operational Research*, 257(2):355–376, 2017.

[SZ15] Christoph Schwindt and Jürgen Zimmermann, editors. *Handbook on Project Scheduling (Vol. 1 + Vol. 2)*. Springer, 2015.

Acknowledgements

We thank Valentina Cacchiani and Juan-José Salazar-González for the fruitful discussions we had on the topic. We also gratefully acknowledge the computing resources provided by the group of Michael Jünger in Cologne. In particular, we thank Thomas Lange for his technical support. Furthermore, we acknowledge financial support by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of „BAYERN DIGITAL II“. Moreover, we thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for their support within GRK 2297 MathCoRe as well as Project Z01 in CRC TRR 154. Last but not least, we thank the anonymous referees for their insightful comments, which led to a substantial improvement of the paper.