# A parameterized point of view on forming small coalitions

Foivos Fioravantes, Harmender Gahlawat, Nikolaos Melissinos

# A parameterized point of view on forming small coalitions

Foivos Fioravantes[1], Harmender Gahlawat[2], and Nikolaos Melissinos[1]

[1]Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

[2]Combinatorial Optimization group of G-SCOP, Grenoble-INP

**Abstract**

Imagine we want to split a group of agents into teams in the most *efficient* way, considering that each agent has their own preferences about their teammates. This scenario is modeled by the extensively studied Coalition Formation problem. Here, we study a version of this problem where each team must additionally be of bounded size.

We conduct a systematic algorithmic study, providing several intractability results as well as multiple exact algorithms that scale well as the input grows (FPT), which could prove useful in practice.

Our main contribution is an algorithm that deals efficiently with tree-like structures (bounded *treewidth*) for "small" teams. We complement this result by proving that our algorithm is asymptotically optimal. Particularly, even considering star-like structures (bounded *vertex cover number*) cannot result in an algorithm with a better running time, under reasonable theoretical assumptions.

**Keywords:** Coalition formation, additive separable hedonic games, parameterized complexity

## 1 Introduction

Coalition Formation is a central topic in Computational Social Choice and economic game theory [14]. The goal is to partition a set of agents into coalitions to *optimize* some *utility* function. One well-studied notion in Coalition Formation is *Hedonic Games* [24], where the utility of an agent depends solely on the coalition it is placed in. Due to their extremely general nature that captures numerous scenarios, hedonic games are intensively studied in computer science [2, 6, 11, 13, 16, 27, 39, 50, 54], and are shown to have applications in social network analysis [51], scheduling group activities [18], and allocating tasks to wireless agents [53].

Due to its general nature, most problems concerning the computational complexity of hedonic games are hard [52]. In fact, even encoding the preferences of agents, in general, takes exponential space, which motivates the study of succinct representations for agent preferences. One of the most-studied such class of games is Additive Separable Hedonic Games [12], where the agents are represented by the vertices of a weighted graph and the weight of each edge represents the *utility* of the agents joined by the edge for each other (see also Weighted Graphical Games model of [19]). Variants where the agent preferences are asymmetric are modeled using directed graphs. Here, the utility of an agent for a group of agents is *additive* in nature. Additive Separable Hedonic Games are well-studied in the literature [1, 3, 7].

Most literature in the Additive Separable Hedonic Games considers the agents to be *selfish* in nature and hence, the notion used to measure the efficiency is that of *stability* [52], including *core stability*, *Nash Stability*, *individual stability*, etc. Semi-altruistic approaches where the agents are concerned about their *relative*'s utility along with theirs are also studied [47]. A standard altruistic approach in computational social choice is that of *utilitarian social welfare*, where the goal is to maximize the total sum of utility of all the agents. Observe that if all edge weights are positive, then the maximum

utilitarian utility is achieved by putting all agents in the same coalition. But there are many practical scenarios, for example, forming office teams to allocate several projects and allocating cars/buses to people for a trip, where we additionally require that each coalition should be of a bounded size.

We consider the Additive Separable Hedonic Games with an additional constraint on the maximum allowed size of a coalition (denoted by $\mathcal{C}$), with the goal to maximize the total sum of utility of all the agents. We formally define the problem definition, along with other preliminaries, in Section 2. This game is known to be NP-hard even when $\mathcal{C} = 3$ [46] (and hence W-hard parameterized by $\mathcal{C}$). Therefore, we consider the *parameterized complexity* of this problem through the lens of various structural parameters of the input graph and present a comprehensive analysis of its computational complexity. In parameterized complexity, the goal is to restrict the exponential blow-up of running time to some *parameter* of the input (which is usually much smaller than the input size) rather than the whole input size. Due to its practical efficiency, the paradigm of parameterized complexity has been used extensively to study problems arising from Computational Social Choice and Artificial Intelligence [5, 8, 15] (including hedonic games [33, 37]).

It is worth mentioning that $\mathcal{C}$-CF (defined later) has been studied from an approximation perspective and is shown to have applications in Path Transversals [44]. Moreover, [4] considered a Weighted Graphical Game to maximize social welfare and provided constant-factor approximation for restricted families of graphs. Finally, [29] considered the online version of several Weighted Graphical Games (aiming to maximize utilitarian social welfare), in one of which they also consider coalitions of bounded size.

## Our contribution

In this paper we study the $\mathcal{C}$-Coalition Formation problem, which is a version of the Coalition Formation problem with the added constraint that each coalition should be of size at most $\mathcal{C}$. We consider two distinct variants of this problem according to the possibilities for the utilities of the agents. In the *unweighted* version, the utilities of all the pairs of agents are either 0 (there is no edges connecting them) or 1. In the *weighted* version, the utilities of all pairs of agents are given by natural numbers. We will refer to the former as the $\mathcal{C}$-CF and the latter as the $\mathcal{C}$-CF$w$ problems, respectively.

Recall that the $\mathcal{C}$-CF is, generally speaking, a computationally hard problem. To combat this, we propose two algorithms that are efficient in the case where the input graph has a tree-like (bounded treewidth) or a star-like (bounded vertex cover number) structure. Such structures may seem restrictive at first glance, but it is often the case that inputs stemming from real-world applications do exhibit them (recall the small world phenomenon [25]). In particular, we show that:

**Theorem 1.1.** *The $\mathcal{C}$-CF$w$ problem can be solved in time $(\mathrm{tw}\mathcal{C})^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$, where* tw *is the treewidth of the input graph.*

The complexity in the above algorithm depends on $\mathcal{C}$. It is natural to wonder whether there can be an efficient algorithm that avoids this. We answer this question negatively.

**Theorem 1.2.** *The $\mathcal{C}$-CF problem is W[1]-hard when parameterized by the tree-depth of the input graph.*

Nevertheless, we do achieve such an algorithm by allowing the input to have a star-like structure. In the following statements, vc denotes the vertex cover number of the input graph.

**Theorem 1.3.** *The $\mathcal{C}$-CF$w$ problem can be solved in time $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$.*

Then, we prove that both of the above algorithms are, essentially, optimal, *i.e.*, we do not expect a drastic improvement in their running times.

**Theorem 1.4.** *There is no algorithm that solves the $\mathcal{C}$-CF problem in time $(\mathcal{C}\text{vc})^{o(\text{vc}+\mathcal{C})}n^{\mathcal{O}(1)}$, unless the ETH fails.*

We then slightly shift our approach and attack this problem using the toolkit of *kernelization*. Intuitively, our goal is to "peel off" the useless parts of the input (in polynomial time) and solve the problem for the "small" part of the input remaining, known as the *kernel*. Due to its profound impact, kernelization was termed "the lost continent of polynomial time" [28]. It is specifically useful in practical applications as it has shown tremendous speedups in practice [34, 35, 49, 55].

**Theorem 1.5.** *$\mathcal{C}$-CF admits a kernel with $\mathcal{O}(\text{vc}^2\mathcal{C})$ vertices.*

We complement the above result by proving that, unfortunately, there can be no such kernel for the weighted version.

**Theorem 1.6.** *It is highly unlikely to construct a poly$\{\text{vc}+\mathcal{C}\}$ size kernel that solves the $\mathcal{C}$-CF$w$ problem.*

We close our study by considering additional structural parameters for the unweighted case. We postpone the formal definition of these parameters until Section 2.2.

**Theorem 1.7.** *The $\mathcal{C}$-CF problem can be solved in FPT time when parameterized by the vertex integrity of the input graph.*

**Theorem 1.8.** *The $\mathcal{C}$-CF problem is W[1]-hard when parameterized by the twin-cover number of $G$.*

The choice to focus our attention to the above two parameters is not arbitrary. Let $G$ be a graph with vertex integrity vi, twin-cover number twc and vertex cover number vc. Then, $\text{vi} \leq \text{twc} + \omega(G)$ and $\text{twc} \leq \text{twc} + \omega(G)$, where $\omega(G)$ is the clique number of $G$. Finally, $\text{twc} + \omega(G) \leq f(\text{vc})$, for some computable function $f$. Taking the above into consideration, our Theorems 1.7 and 1.8 provide a clear dichotomy of the tractability of $\mathcal{C}$-CF when considering these parameters.

# 2 Preliminaries

We follow standard graph-theoretic notation [20]. For any integer and $n$, we denote $[n]$ the set of all integers between 1 and $n$. That is, $[n] = \{1, \ldots, n\}$.

Formally, the input of the $\mathcal{C}$-CF$w$ consists of a graph $G = (V, E)$ and an edge-weight function $w : E \to \mathbb{N}$. Additionally, we are given a *capacity* $\mathcal{C} \in \mathbb{N}$ as part of the input. Our goal is to find a *$\mathcal{C}$-partition* of $V$, that is, a partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ such that $|C_i| \leq \mathcal{C}$ for each $i \in [p]$. For each $i \in [p]$, let $E_i$ denote the edges of $G[C_i]$. Let $E(\mathcal{P})$ be the set of edges of the partition $\mathcal{P}$, *i.e.*, $E(\mathcal{P}) = \bigcup_{i=1}^{p} E(G[C_i])$. The *value* of a $\mathcal{C}$-partition $\mathcal{P}$ is: $v(\mathcal{P}) = \sum_{i=1}^{p} \sum_{e \in E(C_i)} w(e)$. We are interested in computing an *optimal* $\mathcal{C}$-partition, *i.e.*, a $\mathcal{C}$-partition of maximum value. Note that we will also use the defined notations for general (not necessarily $\mathcal{C}$-)partitions.

We are also interested in the *unweighted* version of the $\mathcal{C}$-CF problem, where each edge of the input graph has a weight of 1; in such cases, the input of the problem will only consist of the graph and the required capacity.

## 2.1 Parameterized Complexity - Kernelization

*Parameterized complexity* is a computational paradigm that extends classical measures of time complexity. The goal is to examine the computational complexity of problems with respect to an additional measure, referred to as the parameter. Formally, a parameterized problem is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$, where $k$ is called the parameter of the instance. A parameterized problem is *Fixed-Parameter Tractable* (FPT) if it can be solved in $f(k)|x|^{\mathcal{O}(1)}$ time for an arbitrary computable function $f\colon \mathbb{N} \to \mathbb{N}$. According to standard complexity-theoretic assumptions, a problem is not in FPT if it is shown to be W[1]-hard. This is achieved through a *parameterized reduction* from another W[1]-hard problem, a reduction, achieved in polynomial time, that also guarantees that the size of the considered parameter is preserved.

A *kernelization algorithm* is a polynomial-time algorithm that takes as input an instance $(I, k)$ of a problem and outputs an *equivalent instance* $(I', k')$ of the same problem such that the size of $(I', k')$ is bounded by some computable function $f(k)$. The problem is said to admit an $f(k)$ sized kernel, and if $f(k)$ is polynomial, then the problem is said to admit a polynomial kernel. It is known that a problem is FPT if and only if it admits a kernel.

Finally, the *lower bounds* we present are based on the so-called Exponential Time Hypothesis (ETH for short) [40], a weaker version of which states that 3-SAT cannot be solved in time $2^{o(n+m)}$, for $n$ and $m$ being the number of variables and clauses of the input formula respectively.

We refer the interested reader to classical monographs [17, 48, 30, 21, 31] for a more comprehensive introduction to this topic.

## 2.2 Structural parameters

Let $G = (V, E)$ be a graph. A set $U \subseteq V$ is a *vertex cover* of $G$ if for every edge $e \in E$ it holds that $U \cap e \neq \emptyset$. The *vertex cover number* of $G$, denoted $\mathrm{vc}(G)$, is the minimum size of a vertex cover of $G$.

A *tree-decomposition* of $G$ is a pair $(T, \mathcal{B})$, where $T$ is a tree, $\mathcal{B}$ is a family of sets assigning to each node $t$ of $T$ its *bag* $B_t \subseteq V$, and the following conditions hold:

- for every edge $\{u, v\} \in E(G)$, there is a node $t \in V(T)$ such that $u, v \in B_t$ and
- for every vertex $v \in V$, the set of nodes $t$ with $v \in B_t$ induces a connected subtree of $T$.

The *width* of a tree-decomposition $(T, \mathcal{B})$ is $\max_{t \in V(T)} |B_t| - 1$, and the treewidth $\mathrm{tw}(G)$ of a graph $G$ is the minimum width of a tree-decomposition of $G$. It is well known that computing a tree-decomposition of minimum width is fixed-parameter tractable when parameterized by the treewidth [42, 9], and even more efficient algorithms exist for obtaining near-optimal tree-decompositions [43].

A tree-decomposition $(T, \mathcal{B})$ is *nice* if every node $t \in V(T)$ is exactly of one of the following four types:

**Leaf:** $t$ is a leaf of $T$ and $|B_t| = 0$.

**Introduce:** $t$ has a unique child $c$ and there exists $v \in V$ such that $B_t = B_c \cup \{v\}$.

**Forget:** $t$ has a unique child $c$ and there exists $v \in V$ such that $B_c = B_t \cup \{v\}$.

**Join:** $t$ has exactly two children $c_1, c_2$ and $B_t = B_{c_1} = B_{c_2}$.

Every graph $G = (V, E)$ admits a nice tree-decomposition that has width equal to $\mathrm{tw}(G)$ [10].

The *tree-depth* of $G$ can be defined recursively: if $|V| = 1$ then $G$ has tree-depth 1. Then, $G$ has tree-depth $k$ if there exists a vertex $v \in V$ such that every connected component of $G[V \setminus \{v\}]$ has tree-depth at most $k - 1$.

The graph $G$ has *vertex integrity* $k$ if there exists a set $U \subseteq V$ such that $|U| = k' \leq k$ and all connected components of $G[V \setminus U]$ are of order at most $k - k'$. We can find such a set in FPT-time parameterized by $k$ [23].

A set $S$ is a *twin-cover* [32] of $G$ $V$ can be partitioned into the sets $S, V_1, \ldots, V_p$, such that for every $i \in [p]$, all the vertices of $V_i$ are twins. The size of a minimum twin-cover of $G$ is the *twin-cover number* of $G$.

Let $A$ and $B$ be two parameters of the same graph. We will write $A \leq_f B$ to denote that the parameter $A$ is upperly bounded by a function of parameter $B$. Let $G$ be a graph with treewidth tw, vertex cover number vc, tree-depth td, twin-cover number twc and vertex integrity vi. We have that that twc $\leq_f$ vc. Moreover, tw $\leq_f$ td $\leq_f$ vi $\leq_f$ vc, but twc is incomparable to tw.

# 3 Bounded Tree-width or Vertex Cover Number

This section includes both the positive and negative results we provide for graphs of bounded tree-width or bounded vertex cover number.

## 3.1 Graphs of bounded tree-width

**Theorem 3.1.** *Given a weighted graph $G$, as well as a nice tree decomposition of $G$ of width* tw*, there exists an algorithm that computes an optimal $\mathcal{C}$-partition of $G$ in time* $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}n^{\mathcal{O}(1)}$.

*Proof.* As the techniques we are going to use are standard, we are sketching some of the introductory details. For more details on tree decompositions (definition and terminology), see [22]. Assuming that we have a *nice tree decomposition* $\mathcal{T}$ of the graph $G$ rooted at a node $r$, we are going to perform dynamic programming on the nodes of $\mathcal{T}$. For a node $t$ of $\mathcal{T}$, we denote by $B_t$ the bag of this node and by $B_t^{\downarrow}$ the set of vertices of the graph that appears in the bags of the nodes of the subtree with $t$ as a root. Observe that $B_t \subseteq B_t^{\downarrow}$.

In order to simplify some parts of the proof, we assume that the $\mathcal{C}$-partitions we look into are allowed to include empty sets. In particular, whenever we consider a $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ of a graph $G[B_t^{\downarrow}]$, we assume that is in the following form:

- $p \geq \text{tw} + 1$,
- for any set $C_j \in \mathcal{P}$, if $j \in [\text{tw}+1]$ then either $C_j = \emptyset$ or $C_j \cap B_t \neq \emptyset$ and
- for any set $C_j \in \mathcal{P}$, if $j > \text{tw}+1$ then $C_j \neq \emptyset$ and $C_j \cap B_t = \emptyset$.

Note that any $\mathcal{C}$-partition can be made to fit such a form without affecting its value. Also, for any node $t$ of the tree decomposition and any $\mathcal{C}$-partition of $G[B_t^{\downarrow}]$, no more than tw $+ 1$ sets of the $\mathcal{C}$-partition can intersect with $B_t$. Thus, we do not need to store more sets of $\mathcal{P}$ intersecting with $B_t$.

For all nodes $t$ of the tree decomposition, we will create all the $\mathcal{C}$-partitions of $G[B_t^{\downarrow}]$ that are needed in order to find an optimal $\mathcal{C}$-partition; this will be achieved by storing only $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}$ $\mathcal{C}$-partitions for each bag. In order to decide which $\mathcal{C}$-partitions we need to keep, we first define types of $\mathcal{C}$-partitions of $G[B_t^{\downarrow}]$ based on their intersection with $B_t$ and the size of their sets. In particular, let $Col$ be a *coloring function* $Col : B_t \to [tw+1]$ and $S$ be a table of size $|tw+1|$ such that $0 \leq S[i] \leq \mathcal{C}$ for all $i \in [\text{tw}+1]$. We will say that a $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ is of *type* $(Col, S)_t$ if:

- $\mathcal{P}$ is a $\mathcal{C}$-partition of $G[B_t^{\downarrow}]$,
- for any $i \leq \text{tw}+1$ and $u \in B_t$, $Col(u) = i$ if and only if $u \in C_i \cap B_t$ and
- $S[i] = |C_i|$ for all $i \in [tw+1]$.

For any $\mathcal{C}$-partition $\mathcal{P}$ of type $(Col, S)_t$, the function $Col$ describes the way that $\mathcal{P}$ partitions the set $B_t$. Also, the table $S$ gives us the sizes of the sets of $\mathcal{P}$ that intersect with $B_t$.

Finally, for any node $t$, a $\mathcal{C}$-partition of type $(Col, S)_t$ will be called *important* if it has value greater or equal to the value of any other $\mathcal{C}$-partition of the same type. Notice that any optimal $\mathcal{C}$-partition of the given graph is also an important $\mathcal{C}$-partition of the root of the tree decomposition. Therefore, to compute an optimal $\mathcal{C}$-partition of $G$, it suffices to find an important $\mathcal{C}$-partition of maximum value among the all important $\mathcal{C}$-partitions of the root of the given tree decomposition of $G$.

We now present the information we will keep for each node. Let $t$ be a node of the tree decomposition, $Col : B_t \to [tw+1]$ be a function and $S$ be a table of size $|tw+1|$ such that $0 \leq S[i] \leq \mathcal{C}$ for all $i \in [\text{tw}+1]$.

If there exists an important $\mathcal{C}$-partition of type $(Col, S)_t$, then we store a tuple $(Col, S, W, \mathcal{P})$ for $t$, where $\mathcal{P}$ is an important $\mathcal{C}$-partition of type $(Col, S)_t$ and $W$ is its value. Observe that $W$ is the value of a partition of the whole subgraph induced the vertices belonging to $B_t^{\downarrow}$.

We now explain how to deal with each kind of node of the nice tree decomposition.

**Leaf Nodes.** Since the leaf nodes contain no vertices, we do not need to keep any non-trivial coloring. Also, all the positions of the tables $S$ are equal to 0. Finally, we keep a $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_{\mathrm{tw}+1}\}$ where $C_i = \emptyset$ for all $i \in [\mathrm{tw}+1]$.

**Introduce Nodes.** Let $t$ be an introduce node with $c$ being its child node and $u$ be the newly introduced vertex. We will use the tuples we have computed for $c$ in order to build one important $\mathcal{C}$-partition for each type of $\mathcal{C}$-partition that exists for $t$. For each tuple $(Col, S, W, \mathcal{P})$ of $c$, we create at most $\mathrm{tw}+1$ tuples for $t$ as follows. For each color $i \in [\mathrm{tw}+1]$ we consider two cases: either $0 \leq S[i] < \mathcal{C}$ or $S[i] = \mathcal{C}$. If $0 \leq S[i] < \mathcal{C}$, then we set $Col(u) = i$, increase $S[i]$ by one, extend the $\mathcal{C}$-partition $\mathcal{P}$ by adding $u$ into the set $C_i$ and increase $W$ by $\sum_{uv \in E, v \in C_i} w(uv)$. If $S[i] = \mathcal{C}$ then we cannot color $u$ with the color $i$ as the corresponding set is already of size $\mathcal{C}$.

First, we need to prove that, this way, we create at least one important $\mathcal{C}$-partition for $t$ for each type of $\mathcal{C}$-partition of $G[B_t^{\downarrow}]$. Assume that for a type $(Col, S)_t$ there exists an important $\mathcal{C}$- partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ of $B_t^{\downarrow}$.

Let $\mathcal{P}_c$ be the $\mathcal{C}$-partition we defined by the restriction of $\mathcal{P}$ on the vertex set $B_c^{\downarrow}$. That is, $\mathcal{P}_c = \{C_1^c, \ldots, C_p^c\}$ where $C_i^c = C_i \cap B_c^{\downarrow}$ for all $i \in [p]$. Notice that, since $c$ is the child of an introduce node, there exists a $k \in [\ell]$ such that $C_k^c = C_k \setminus \{u\}$, and $C_i^c = C_i$ for all $i \in [p] \setminus \{k\}$. Also, note that $C_k^c$ may be empty. Since $\mathcal{P}$ is a $\mathcal{C}$-partition of $G[B_t^{\downarrow}]$, we have that $\mathcal{P}_c$ is a $\mathcal{C}$-partition of $G[B_c^{\downarrow}]$. Furthermore, let $Col' : B_c \to [\mathrm{tw}+1]$ such that $Col'(u) = Col(u)$ for all $u \in B_c$ and $S'$ be a table where $S'[i] = S[i]$ for all $i \in [\mathrm{tw}+1] \setminus k$ and $S'[k] = S[k] - 1$. Observe that $\mathcal{P}_c$ is of type $(Col', S')_c$.

Since $\mathcal{P}_c$ is of type $(Col', S')_c$, we know that we have stored a tuple $(Col', S', W', \mathcal{P}')$ for $c$, where $\mathcal{P}' = \{C_1', \ldots, C_{p'}'\}$ is an important $\mathcal{C}$-partition of $G[B_c^{\downarrow}]$. Note that $\mathcal{P}'$ is not necessarily the same as $\mathcal{P}_c$, but both of these $\mathcal{C}$-partitions are of the same type. While constructing the tuples of $t$, at some point the algorithm will consider the tuple $(Col', S', W', \mathcal{P}')$. At this stage, the algorithm will add the vertex $u$ on any set of $\mathcal{P}'$ of size at most $\mathcal{C}-1$, creating a different tuple for each option. These options include the set colored by $k$; let $(Col_t, S_t, W_t, \mathcal{P}_t)$ be the corresponding tuple, where $\mathcal{P}_t = \{C_1^t, \ldots, C_{p'}^t\}$. Observe that in this case, $u$ is colored $k$ (*i.e.* $Col_t(u) = k = Col(u)$), $S'[k]$ is increase by one (*i.e.* $S_t[k] = S'[k] + 1 = S[k]$) and $u$ is added to $C_k'$ (*i.e.* $C_k^t = C_k' \cup \{u\}$). Notice that $Col'(v) = Col(v)$ for all $v \in B_t$ and $S'[i] = S[i]$ for all $i \in [\mathrm{tw}+1]$. Therefore, it suffices to show that $\mathcal{P}_t$ is also an important $\mathcal{C}$-partition of $G[B_t^{\downarrow}]$. Indeed, this would indicate that $\mathrm{val}(\mathcal{P}) = \mathrm{val}(\mathcal{P}_t)$, since $\mathcal{P}$ and $\mathcal{P}_t$ would both be important partitions of the same type.

On the one hand, we have that:

$$\mathrm{val}(\mathcal{P}) = \mathrm{val}(\mathcal{P}_c) + \sum_{uv \in E, v \in C_k} w(uv) =$$

$$= \mathrm{val}(\mathcal{P}_c) + \sum_{uv \in E, v \in B_t \text{ and } Col(v) = k} w(uv)$$

On the other hand, we have that:

$$\mathrm{val}(\mathcal{P}_t) = \mathrm{val}(\mathcal{P}') + \sum_{uv \in E, v \in C_k'} w(uv) =$$

$$= W' + \sum_{uv \in E, v \in B_t \text{ and } Col'(v) = k} w(uv)$$

Since $Col(v) = Col'(v)$ for all $v \in B_t$, we have that the two above sums are equal. Therefore we need to compare $W'$ with $\mathrm{val}(\mathcal{P}_c)$. Note that $\mathcal{P}_c$ and $\mathcal{P}'$ are both $\mathcal{C}$-partitions of $G[B_c^{\downarrow}]$ of the same type. Thus, $W' = \mathrm{val}(\mathcal{P}') \geq \mathrm{val}(\mathcal{P}_c)$. It follows that $\mathrm{val}(\mathcal{P}) \leq \mathrm{val}(\mathcal{P}_t)$, and since $\mathcal{P}$ is important, we have that $\mathrm{val}(\mathcal{P}_t) = \mathrm{val}(\mathcal{P})$ and that $\mathcal{P}_t$ is also important.

**Forget Nodes.** Let $t$ be an forget node, with $c$ being its child node and $u$ be the newly introduced vertex. We will use the tuples we have computed for $c$ in order to build one important $\mathcal{C}$-partition for each type of $\mathcal{C}$-partition that exists for $t$. For each tuple $(Col, S, W, \mathcal{P})$ of $c$ we create one tuple $(Col', S', W', \mathcal{P}')$ for $t$ as follows. Let $Col(u) = i$. We consider two cases: either $C_i \cap B_t = \emptyset$ or not. In the former, we have that the color $i$ does not appear on any vertex of $B_c \setminus \{u\} = B_t$. Therefore, are free to reuse this color. To do so, we set $S'[i] = 0$ and we modify $\mathcal{P}$. In particular, if $\mathcal{P} = \{C_1, \ldots, C_k\}$, we create a new $\mathcal{C}$-partition $\mathcal{P}' = \{C'_1 \ldots, C'_{k+1}\}$ where $C'_j = C_j$ for all $j \in [k] \setminus \{i\}$, $C'_i = \emptyset$ and $C'_{k+1} = C_i$. Also, we define $Col'$ as the restriction of the function $Col$ to the set $B_t$. Finally, $W' = W$. In the latter case, it suffices to restrict $Col$ to the set $B_t$. We keep all the other information the same.

We will now prove that, for any type of $\mathcal{C}$-partition of $t$, if there exists a $\mathcal{C}$-partition of that type, we have created an important $\mathcal{C}$-partition of that type. Assume that for a type $(Col, S)_t$ there exists an important $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ of $B_t$ of value $W$. We consider two cases: either $u \in C_\ell$ for some $\ell \leq \text{tw} + 1$ or $u \in C_\ell$ for some $\ell > \text{tw} + 1$.

**Case 1: $u \in C_\ell$ for some $\ell \leq \text{tw} + 1$.** In this case, $C_\ell \cap B_t \neq \emptyset$. This follows from the assumption that any $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ we consider is such that for any set $C_j \in \mathcal{P}$, if $j \in [\text{tw} + 1]$ then either $C_j = \emptyset$ or $C_j \cap B_t \neq \emptyset$ and because $\{v \mid v \in B_c \setminus \{u\}$ and $Col(v) = \ell\} \neq \emptyset$. Let $Col_c : B_c \to [\text{tw}+1]$ be such that $Col_c(u) = \ell$ and $Col_c(v) = Col(v)$ for all $v \in B_t$. Notice that $\mathcal{P}$ is of type $(Col_c, S)_c$. Let $(Col_c, S, W', \mathcal{P}')$ be the tuple that is stored in $c$ for the $\mathcal{C}$-partition $\mathcal{P}' = \{C'_1, \ldots, C'_{p'}\}$ of type $(Col_c, S)_c$.

While creating the tuples of $t$, at some point, the tuple $(Col_c, S, W', \mathcal{P}')$ was considered. Let $(Col'_c, S', W', \mathcal{P}')$ be the tuple that was created at that step. Notice that, since $Col_c$ is an extension of $Col$ to the set $B_c$ and $C_\ell \cap B_t = \{v \in B_t \mid Col(v) = \ell\} \cap B_t \neq \emptyset$, we have that $\{v \in B_t \mid Col_c(v) = \ell\} \cap B_t \neq \emptyset$. Therefore, $\{v \in B_t \mid Col'_c(v) = \ell\} \cap B_t = \{v \in B_t \mid Col_c(v) = \ell\} \cap B_t \neq \emptyset$. Thus, it follows from the construction of $(Col'_c, S', W', \mathcal{P}')$ that $Col'_c(v) = Col(v)$ for all $v \in B_t$. Also, since $\{v \in B_t \mid Col'_c(v) = \ell\} \cap B_t \neq \emptyset$, the vertex $u$ was not the only vertex colored with $\ell$. Therefore, $S'$ is the same as $S$. This gives us that $\mathcal{P}'$ and $\mathcal{P}$ are of the same type in $t$. That is, $(Col'_c, S')_t = (Col, S)_t$ and we have stored a tuple for this type.

It remains to show that $\mathcal{P}'_t$ is an important partition of its type in $t$. This is indeed the case as $\mathcal{P}$ and $\mathcal{P}'$ have the same type in $c$ and $\mathcal{P}'$ is an important partition of this type in $c$. Since the value of the two partitions does not change in $t$ and they remain of the same type, we have that $\mathcal{P}'_t$ is an important partition of its type in $t$.

**Case 2: $u \in C_\ell$ for some $\ell > \text{tw} + 1$.** In this case we have that $C_\ell \cap B_t = \emptyset$ and $C_\ell \cap B_c = \{u\}$. Notice that, at least one of the $C_i$s, $i \in [\text{tw} + 1]$, must be empty. Indeed, since $C_i \cap B_c = C_i \cap B_t \neq \emptyset$, for all $i \in [\text{tw} + 1]$, we have $\text{tw} + 2$ sets intersecting $B_c$ (including $C_\ell$). This is a contradiction as these sets must be disjoint and $|B_c| \leq \text{tw} + 1$.

First, we need to modify the partition $\mathcal{P}$ so that it respects the second item of the assumptions we have made for the $\mathcal{C}$-partitions in $c$. To do so, select any $k \in [\text{tw}+1]$ such that $C_k = \emptyset$ and set $C_k = C_i$. Then, set $C_k = C_{k+1}$, for all $k \in [p-1] \setminus [i-1]$, and remove $C_p$. Let $\mathcal{P}_c = \{C_{c,1}, \ldots, C_{c,p-1}\}$ be the resulting $\mathcal{C}$-partition of $c$. We define $Col_c : B_c \to [\text{tw} + 1]$ such that, for all $v \in B_c$, $Col_c(v) = i$ if and only if $v \in C_{c,i}$. Notice that $Col$ is the restriction of $Col_c$ on the vertex set $B_t$. Also, we define $S_c$ to be the table of size $\text{tw} + 1$ such that, for all $i \in \text{tw} + 1$, $S_c[i] = |C_{c,i}|$. Notice that for all $i \in [\text{tw} + 1] \setminus \{k\}$, we have $S[i] = S_c[i]$ and $S_c[k] \neq 0$ and $S[k] = 0$.

Observe that $\mathcal{P}$ is of type $(Col, S)_t$ and $\mathcal{P}_c$ is of type $(Col_c, S_c)_c$. Therefore, let $(Col_c, S_c, W', \mathcal{P}')$ be the tuple we have stored in $c$, where $\mathcal{P}'$ is an important partition of type $(Col_c, S_c)_c$. While constructing the tuples of $t$, at some point, we consider the tuple $(Col_c, S_c, W', \mathcal{P}')$ and create a tuple $(Col_t, S_t, W', \mathcal{P}_t)$ for $t$. We claim that $\mathcal{P}_t$ is of the same type as $\mathcal{P}$ and that $\mathcal{P}_t$ is an important partition of that type. Notice that $u$ is the only vertex of $B_c$ such that $Col_c(u) = k$. It follows that $(Col_t, S_t, W', \mathcal{P}_t)$ was created by setting:

- $Col_t$ to be the restriction of $Col_c$ on the set $B_t$,
- $S_t[k] = 0$ and $S_t[i] = S_c[i]$, for $i \in [\text{tw} + 1] \setminus \{k\}$ and
- we modify the $\mathcal{P}_c$ following the steps described by the algorithm.

Notice that, $\mathcal{P}'$ and $\mathcal{P}_t$ are the same $\mathcal{C}$-partition, presented in a different way. By the construction of $Col'_t$ and $S'_t$, we have that $(Col_t, S_t)_t$ is the same as $(Col, S)_t$. It follows that there exists a tuple $(Col, S, W', \mathcal{P}')$ stored in $t$, where $\mathcal{P}'$ is of type $(Col, S)_t$.

It remains to show that $\mathcal{P}_t$ is an important partition of its type. Notice that $\mathcal{P}$ and $\mathcal{P}_c$ are the same $\mathcal{C}$-partition. Therefore, they have the same value. The same holds for $\mathcal{P}'$ and $\mathcal{P}_t$. Finally, since $\mathcal{P}'$ and $\mathcal{P}_c$ have the same type in $c$ and $\mathcal{P}'$ is an important partition, we have that $\text{val}(\mathcal{P}') \geq \text{val}(\mathcal{P}_c)$. So, $\text{val}(\mathcal{P}_t) = \text{val}(\mathcal{P}') \geq \text{val}(\mathcal{P}_c) = \text{val}(\mathcal{P})$, from which follows that $\mathcal{P}_t$ is also an important partition.

**Join Nodes.** Let $t$ be a join node, with $c_1$ and $c_2$ being its children nodes. We will use the tuples we have computed for $c_1$ and $c_2$ in order to build one important $\mathcal{C}$-partition for each type of $\mathcal{C}$-partition that exists for $t$. For any pair of tuples $(Col_1, S_1, W_1, \mathcal{P}_1)$ and $(Col_2, S_2, W_2, \mathcal{P}_2)$, of $c_1$ and $c_2$ respectively, we will create a tuple $(Col, S, W, \mathcal{P})$ for $t$ if:

- $Col_1(u) = Col_2(u)$ for all $u \in B_t$ (which is the same as $B_{c_1}$ and $B_{c_2}$),
- for all $i \in [\text{tw}+1]$, $S_1[i] + S_2[i] - |C_i \cap B_t| \leq \mathcal{C}$,

where $C_i$ is the $i^{th}$ set of $\mathcal{P}_1$. Note that the choice of $\mathcal{P}_1$ here is arbitrary because of the first condition. Indeed, the first condition guarantees that $\mathcal{P}_1$ and $\mathcal{P}_2$ "agree" on the vertices of $B_t$. That is, the vertices of $B_t$ are partitioned in the same sets according to $\mathcal{P}_1$ and $\mathcal{P}_2$. The second conditions guarantees that the sets created for $\mathcal{P}$ are of size at most $\mathcal{C}$. The tuple $(Col, S, W, \mathcal{P})$ is created as follows. We set:

- $Col(u) = Col_1(u)$ for all $u \in B_t$,
- $S[i] = S_1[i] + S_2[i] - |C_i \cap B_t|$ for all $i \in [\text{tw}+1]$, and
- $W = W_1 + W_2 - \sum_{uv \in E(G[B_t]), Col(u)=Col(v)} w(uv)$.

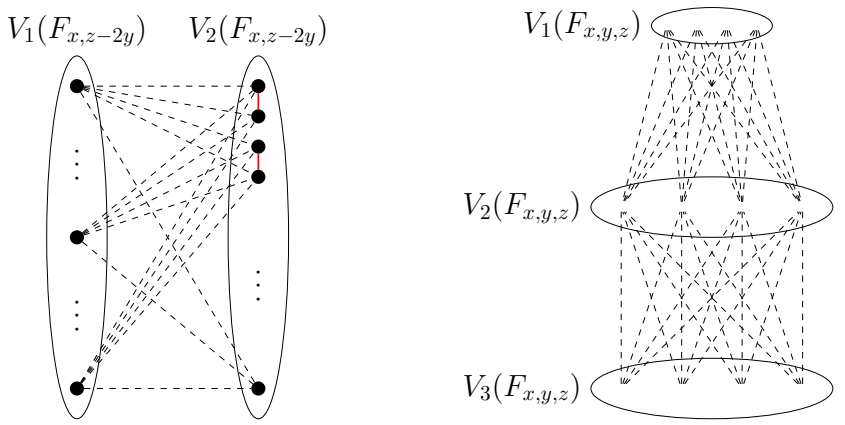
Once more, $C_i$ is chosen w.l.o.g. to be the $i^{th}$ set of $\mathcal{P}_1$. Finally, we define $\mathcal{P}$. Let $\mathcal{P}_1 = \{C_1^1, \ldots, C_p^1\}$ and $\mathcal{P}_2 = \{C_1^2, \ldots, C_{p'}^2\}$; we create the $\mathcal{C}$-partition $\mathcal{P} = \{C_1 \ldots, C_{p+p'-\text{tw}-1}\}$ as follows. For any $i \in [\text{tw}+1]$, set $C_i = C_i^1 \cup C_i^2$. For any $i \in [p] \setminus [\text{tw}+1]$, set $C_i = C_i^1$. Last, for any $i \in [p'] \setminus [\text{tw}+1]$, set $C_{p+i} = C_i^2$. This completes the construction of the tuple we keep for $t$, for each pair of tuples that are stored for $c_1$ and $c_2$.

We will now prove that, for any type of $\mathcal{C}$-partition of $t$, if there exists a $\mathcal{C}$-partition of that type, we have created an important $\mathcal{C}$-partition of that type. We assume that for a type $(Col, S)_t$ of $t$, there exists an important $\mathcal{C}$-partition $\mathcal{P} = \{C_1, \ldots, C_p\}$ of $G[B_t^{\downarrow}]$. Let $\mathcal{P}_1 = \{C_1 \cap B_{c_1}^{\downarrow}, \ldots, C_p \cap B_{c_1}^{\downarrow}\}$ and $\mathcal{P}_2 = \{C_1 \cap B_{c_2}^{\downarrow}, \ldots, C_p \cap B_{c_2}^{\downarrow}\}$. Notice that $\mathcal{P}_1$ and $\mathcal{P}_2$ are $\mathcal{C}$-partitions of $G[B_{c_1}^{\downarrow}]$ and $G[B_{c_2}^{\downarrow}]$, respectively. Let $(Col, S_1)_{c_1}$ and $(Col, S_2)_{c_2}$ be the types of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively (recall that, by construction, $Col_1 = Col_2 = Col$). The existence of $\mathcal{P}_1$ (respectively $\mathcal{P}_2$) guarantees that there is a tuple $(Col, S_1, W_1, \mathcal{P}'_1)$ (resp. $(Col, S_2, W_2, \mathcal{P}'_2)$) stored for the node $c_1$ (resp. $c_2$). By the definition of $\mathcal{P}_1$ and $\mathcal{P}_2$, we have that $S[i] = S_1[i] + S_2[i] - |C_i \cap B_t| \leq \mathcal{C}$ for all $i \in [\text{tw}+1]$. It follows that while constructing the tuples of $t$, the algorithm considered, at some point, the pair of tuples $(Col, S_1, W_1, \mathcal{P}'_1)$ and $(Col, S_2, W_2, \mathcal{P}'_2)$, and created the tuple $(Col, S', W', \mathcal{P}')$ for $t$. Notice that, by the construction of $S'$, we have that $S[i] = S'[i]$ for all $i \in [\text{tw}+1]$. Therefore the type $(Col, S')_t$ is the same as $(Col, S)_t$.

It remains to show that $\mathcal{P}'$ is an important partition of its type. Notice that, $\text{val}(\mathcal{P}') = W' = W_1 + W_2 - \sum_{uv \in E(G[B_t]), Col(u)=Col(v)} w(uv)$ and
$\text{val}(\mathcal{P}) = \text{val}(\mathcal{P}_1) + \text{val}(\mathcal{P}_2) - \sum_{uv \in E(G[B_t]), Col(u)=Col(v)} w(uv)$. Since $W_1$ is the weight of an important partition of the same type as $\mathcal{P}_1$ in $c_1$, we have that $\text{val}(\mathcal{P}_1) \leq W_1$. Also, $W_2$ is the weight of an important partition of the same type as $\mathcal{P}_2$ in $c_2$. It follows that $\text{val}(\mathcal{P}_2) \leq W_2$. Overall: $\text{val}(\mathcal{P}') = W_1 + W_2 - \sum_{uv \in E(G[B_t]), Col(u)=Col(v)} w(uv) \geq \text{val}(\mathcal{P}_1) + \text{val}(\mathcal{P}_2) - \sum_{uv \in E(G[B_t]), Col(u)=Col(v)} w(uv) = \text{val}(\mathcal{P})$.

Thus, $\mathcal{P}'$ is an important partition of its type in $t$. This finishes the description of our algorithm, as well as the proof of its correctness.

It remains to compute the running time of our algorithm. First we calculate the number of different types of $\mathcal{C}$-partitions for a node $t$. We have at most $(\text{tw}+1)^{\text{tw}+1}$ different functions $Col$ and $(\mathcal{C}+1)^{\text{tw}+1}$ different tables $S$. Therefore, we have $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}$ different types for each node. Since we are storing




($\alpha$) the gadget $F_{x,z-2y}$ ($\beta$) the gadget $F_{x,y,z}$

Figure 1: The gadgets used in the proof of Theorem 3.2

one tuple per type, we are storing $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}$ tuples for each node of the tree decomposition. Moreover, for the leaf nodes, we need to create just one tuple. For the introduce and forget nodes, we need to consider each tuple of their children once. Therefore, we can compute all tuples for these nodes in time $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}$. For the join nodes, in the worst case, we may need to consider all pairs of tuples of their children that share the same coloring function. This still does not result in more than $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}$ combinations. Finally, as all the other calculations remain polynomial to the number of vertices, the total time that is required is $(\text{tw}\mathcal{C})^{\mathcal{O}(\text{tw})}|V(G)|^{\mathcal{O}(1)}$. □

**Theorem 3.2.** *Let $G$ be an unweighted graph, and $\mathcal{C}$ and $v^*$ be two integers . Deciding if there exists a $\mathcal{C}$-partition $\mathcal{P}$ of $G$ with $v(\mathcal{P}) \geq v^*$ is W[1]-hard when parameterized by the tree-depth of $G$.*

*Proof.* We present a reduction from the General Factors problem. In this problem, we are given a graph $H = (V, E)$ and a list function $L : V \to \mathcal{P}(\{0, \ldots, \Delta(H)\})$ that specifies the available degrees for each vertex $u \in V$. The question is whether there exists a set $E' \subseteq E$ such that $d_{H-E'}(u) \in L(u)$ for all $u \in V$. It is known that the General Factors problem is W[1]-hard, even on bipartite graphs when parameterized by the size of the smallest bipartition [36]. Let $(H, L)$ be an instance of the General Factors problem where $H = (V_L, V_R, E)$ is a bipartite graph ($V(H) = V_L \cup V_R$ and $E(H) = E$) and $L : V_L \cup V_R \to \mathcal{P}([|V(H)|])$ gives the list of degrees for each vertex. Notice that, normally, $|L(u)| \leq d(u) \leq |V(H)|$. Nevertheless, we can assume that $|L(u)| = |V(H)|$ as we can allow $L(u)$ to be a multiset. Hereafter, we assume that the size for the smallest bipartition is $m$ and the total number of vertices is $n = |V(H)|$. Note that, $m \leq n/2$. We can also assume that $m > 2$ as otherwise, we could answer whether $(H, L)$ is a yes-instance of the General Factors problem in polynomial time.

**The construction.** Starting from $(H, L)$, we will construct a graph $G$ such that any $\mathcal{C}$-partition of $G$, for $\mathcal{C} = 100n^3$, has a value exceeding a threshold if and only if $(H, L)$ is a yes-instance of the General Factors problem. We start by carefully setting values that so that our reduction works. We define the values $A = n^2$, $B = 5n^2 + 3m + 4$ and $D = 2m + 5$ which will be useful for the constructions and calculations that follow.

We now describe the two different gadgets denoted by $F_{x,5A-2y}$ and $F_{x,\mathcal{C}-y,z}$. The $F_{x,5A-2y}$ gadget is defined for $4xy < 5A - 2y$. It is constructed as follows (illustrated in Figure 1(a)):

- We create two independent sets $U$ and $V$ of size $x$ and $5A - 2y$ respectively,
- we add all edges between vertices of $U$ and $V$ and

- we add $2xy$ edges between vertices of $V$ such that the graph induced by the vertices incident to these edges is an induced matching (we have enough vertices because we assumed that $4xy < 5A - 2y$).

Hereafter, for any gadget $F_{x,5A-2y} = F$ we will refer to $U$ as $V_1(F)$ and to $V$ as $V_2(F)$.

The construction of $F_{x,\mathcal{C}-y,z}$ is as follows(illustrated in Figure 1(b)):

- We create three independent sets $U$, $V$ and $W$ of size $x$, $\mathcal{C} - y$ and $z$ respectively,
- we add all edges between vertices of $U$ and $V$ and all edges between vertices of $U$ and $V$,

Hereafter, for any gadget $F_{x,\mathcal{C}-y,z} = F$ we will refer to $U$ as $V_1(F)$, to $V$ as $V_2(F)$ and to $W$ as $V_3(F)$ Before we continue, notice that $|E(F_{x,5A-2y})| = 5xA$ and $|E(F_{x,\mathcal{C}-y,z})| = (x + z)(\mathcal{C} - y)$.

We are now ready to describe the construction of the graph $G$, illustrated in Figure 2. First, for each vertex $v \in V(H)$, we create a copy $F^v$ of the $F_{4,\mathcal{C}-B,2m+10}$ gadget; we say that this is a *vertex-gadget*. We also fix a set $U(F^v) \subset V_1(F^v)$ such that $|U(F^v))| = 2$. Now, for any vertex $v \in V(H)$ and integer $\alpha \in L(v)$, we create a copy $F^{\alpha(v)}$ of the $F_{m+6,5A-2\alpha}$ gadget; we say that this is a *list-gadget*. We add all the edges between $V_2(F^{\alpha(v)})$ and $U(F^v)$. Recall that we have assumed $|L(v)| = |V(H)|$, for all $v \in V(H)$. So, for each vertex $v$ of $H$, in addition to $F^v$, we have created $|V(H)| = n$ gadgets (one for each element in the list). Finally for each edge $e = uv \in E(H)$, where $u \in V_L$ and $v \in V_R$, we create a copy $F^e$ of the $F_{1,\mathcal{C}-D,2m}$ gadget; we say that this is an *edge-gadget*. Then, we add a set of vertices $V_e = \{w^e_{L1}, w^e_{L2}, w^e_{R1}, w^e_{R2}\}$. We add all the edges between $V_1(F^u)$ and $V_e$, all the edges between $V_1(F^u)$ and $\{w^e_{L1}, w^e_{L2}\}$, all the edges between $V_1(F^v)$ and $\{w^e_{R1}, w^e_{R2}\}$ and the edges $w^e_{Li}, w^e_{Rj}$ for all $i, j \in [2]$ (*i.e.*, $V_e$ induces a $K_{2,2}$). Hereafter, let $V_E = \bigcup_{e\in E(H)} V_e$ and by $U_E = \bigcup_{e\in E(H)} V(F^e)$. This completes the construction of $G$.

Before we continue let us introduce some notation. Observe that all the vertex-gadgets contain the same number of edges. For every vertex $v \in V(H)$, let $m_v = |E(F)|$, where $F$ is any vertex-gadget. Similarly, all edge-gadgets contain the same number of edges. For every edge $e \in E(H)$, let $m_e = |E(F)|$, where $F$ is any edge-gadget. Finally, the same holds for the list-gadgets; let $m_\ell = |E(F)|$, where $F$ is any list-gadget.

Our goal is to show that an optimal $\mathcal{C}$-partition $\mathcal{P}$ of $G$ has value $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$ if and only if $(H, L)$ is a yes-instance of the General Factors problem.

To do so, we start by proving some properties of the optimal partitions of $G$.

**Properties of optimal $\mathcal{C}$-partitions of $G$.** Assume that $\mathcal{P}$ an optimal $\mathcal{C}$-partition of $G$. First, we will show that for every gadget $F$, there exists a $C \in \mathcal{P}$ such that $V(F) \subseteq C$. Then we will prove that for any vertex-gadget $F^v$, there exists one list-gadget $F$ that represents an element of the list $L(v)$ (*i.e.* any $u \in U(F^v)$ and $w \in V_2(F)$ are adjacent) and there exists a $C \in \mathcal{P}$ such that $V(F^v) \cup V(F) \subseteq C$. Finally, we will show that, in order for $\mathcal{P}$ to be optimal, *i.e.*, $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$, the vertices of $V_E$ will be partitioned such that:

- the set that includes $V(F^e)$, either includes all the vertices of $V_e$ or none of them and
- the set that includes $V(F^v)$ and a gadget $V(F)$, for a list-gadget $F$ representing the value $\alpha \in L(v)$, will also include $2\alpha$ vertices from $V_E$.

We will show that if both the above conditions hold, then $\mathcal{P}$ is optimal and $(H, L)$ is a yes-instance of the General Factors problem. In particular, the edges $E'$ of the solution of the General Factors problem are exactly the edges $e \in E(H)$ such that $F^e$ and $V_e$ are in the same set of $\mathcal{P}$.

Let $\mathcal{P}$ be a $\mathcal{C}$-partition of $G$. For every $C \in \mathcal{P}$, we can assume that $G[C]$ is connected as otherwise we could consider each connected component of $G[C]$ separately. We start with the following lemma:

**Lemma 3.3.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $F$ be a vertex or edge-gadget. There exists a set $C \in \mathcal{P}$ such that $C \supseteq V(F)$.*

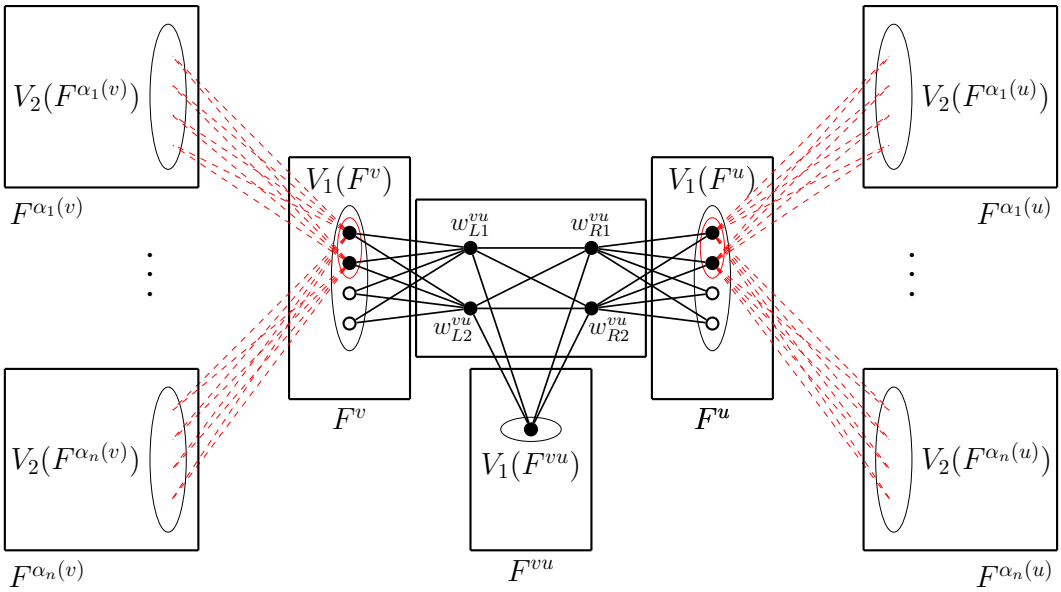



Figure 2: The graph $G$ constructed in the proof of Theorem 3.2

*Proof.* Assume that this is not true and let $F$ be a vertex or edge-gadget such that $C \cap V(F) \neq V(F)$ for all $C \in \mathcal{P}$. We first show that $\max_{C\in\mathcal{P}}\{|C \cap V_2(F)|\} = x \geq 2|V_2(F)|/3$. Assume that $\max_{C\in\mathcal{P}}\{|C \cap V_2(F)|\} = x < 2|V_2(F)|/3$. We consider the partition $\mathcal{P}' = \{V(F), C_1 \setminus V(F), \ldots, C_p \setminus V(F)\}$. We will show that $v(\mathcal{P}) < v(\mathcal{P}')$. Notice that any edge that is not incident to a vertex of $V(F)$ is either in both sets or in none of them. Therefore, we need to consider only edges incident to at least on vertex of $V(F)$. Also, since all edges in $E(F)$ are included in $\mathcal{P}'$ we only need to consider edges incident to $V_1(F)$ (as any other vertex is incident to edges in $E(F)$).

For any vertex $v \in V_1(F)$, let $d = d(v) - |V_2(F)|$ ($d$ is the same for any $v \in V_1(F)$) and $C \in \mathcal{P}$ be the set such that $v \in \mathcal{C}$. We have that $|C \cap N(v)| \leq d + x$, from which follows that $|E(\mathcal{P}) \setminus E(\mathcal{P}')| \leq 4d$. Notice that, regardless of which gadget $F$ and vertex $v \in V_1(F)$ that we consider, we have that $d \leq 5nA + 2n < 6nA$ (since $A = n^2$). Indeed, if $F$ is an edge-gadget then $d = 4$. Also, if $F$ is a vertex-gadget, then any $v \in V_1(F)$ has at most $5A$ neighboring vertices in each of the $n$ list-gadgets related to it (if it is in $U(F)$) and at most $2n$ in $V_E$.

We will now calculate $|E(\mathcal{P}') \setminus E(\mathcal{P})|$. Consider a $v \in V_1(F) \cup V_3(F)$ and let $C \in \mathcal{P}$ such that $v \in C$. Notice that $|C \cap V_2(F)| = x$. Therefore, we have at least $|V_2(F)| - x$ edges incident to $v$, which belong in $E(\mathcal{P}') \setminus E(\mathcal{P})$. Since $V_1(F) \cup V_3(F)$ is an independent set, it follows that $|E(\mathcal{P}') \setminus E(\mathcal{P})| \geq |V_1(F) \cup V_3(F)|(|V_2(F)| - x) > (2m+4)|V_2(F)|/3 > |V_2(F)|$. Now, in order to show that $v(\mathcal{P}') > v(\mathcal{P})$, it suffices to show that $4d < |V_2(F)|$. This is indeed the case as $|V_2(F)| = \mathcal{C} - B = 100n^3 - (5n^2 + 3m + 4) > 24n^3 = 24nA > 4d$. Thus, we can assume that $\max_{C\in\mathcal{P}}\{|C \cap V_2(F)|\} = x \geq 2|V_2(F)|/3$.

Let $C \in \mathcal{P}$ be the set such that $|C \cap V_2(F)| \geq 2|V_2(F)|/3$. We will show that $C \cap V_3(F) = V_3(F)$. Assume that this is not true and let $v \in V_3(F)$ such that $v \notin C$. Notice that at most $y = |V_2(F)| - x \leq |V_2(F)|/3$ edges incident to $v$ are included in $E(\mathcal{P})$. If $|C| < \mathcal{C}$, then moving $v$ from its set to $C$ increases the number of edges in $E(\mathcal{P})$ by $x - y \geq |V_2(F)|/3$. Therefore, we can assume that $|C| = \mathcal{C}$. Since $G[C]$ is connected and $|C| = \mathcal{C}$, we have that $C$ must include at least one vertex from $V_1(F)$ and at least one vertex from $N(V_1(F)) \setminus V_2(F)$. Notice that any vertex $u \in N(V_1(F)) \setminus V_2(F)$ has degree at most $m + 10$ (regardless of the value of $m$). Therefore, by replacing a vertex $u \in C \cap (N(V_1(F)) \setminus V_2(F))$ in $C$ by $v$, we increase the number of edges in $E(\mathcal{P})$ by at least $x - y - d(u) \geq |V_2(F)|/3 - d(u) > 0$. This contradicts the optimality of $\mathcal{P}$. Thus, we can assume that $C \cap V_3(F) = V_3(F)$.

We will show that $C \cap V_2(F) = V_2(F)$. Assume that there exists a vertex $v \in V_2(F) \setminus C$. Since $C \cap V_3(F) = V_3(F)$, we have that $|N(v) \cap C| \geq |V_3(F)| = 2m + 14$ and $N(v) \setminus C \leq 4$. otherwise If $|C| < \mathcal{C}$, then moving $v$ from its set to $C$ increases the number of edges in $E(\mathcal{P})$ (recall that $m > 2$).

Thus we can assume that $|C| = \mathcal{C}$. Since $G[C]$ is connected and $|C| = \mathcal{C}$, we have that $C$ must include at least one vertex in $V_1(F)$ and one from $N(V_1(F)) \setminus V_2(F)$. Notice that any vertex $u \in N(V_1(F)) \setminus V_2(F)$ has degree at most $m + 10$. Therefore, by replacing a vertex $u \in C \cap (N(V_1(F)) \setminus V_2(F))$ in $C$ by $v$, we increase the number of edges in $E(\mathcal{P})$ by at least $2m + 10 - (m + 10) = m$. This contradicts the optimality of $\mathcal{P}$. Thus, we can assume that $C \cap V_2(F) = V_2(F)$.

We now show that $C \cap V_1(F) = V_1(F)$. Assume that this is not true and let $v \in V_1(F)$ such that $v \notin C$. Notice that we may have up to $d(v) - |V_2(F)|$ edge in $E(\mathcal{P})$ that are incident to $v$. If $|C| < \mathcal{C}$, then moving $v$ from its set to $C$ increases the number of edges in $E(\mathcal{P})$ by at least $2|V_2(F)| - d(v) > 0$ (since $V_2(F) \subset C$ and $|V_2(F)| = \mathcal{C} - B$). Thus, we can assume that $|C| = \mathcal{C}$. Since $G[C]$ is connected and $|C| = \mathcal{C}$, we have that $C$ must include at least one vertex in $V_1(F)$ and one from $N(V_1(F)) \setminus V_2(F)$. Any vertex $u \in N(V_1(F)) \setminus V_2(F)$ can contribute at most $m+10$ edges in $E(\mathcal{P})$. Therefore, by replacing $u$ in $C$ by $v$, we increase the number of edges in $E(\mathcal{P})$ by at least $2|V_2(F)| - d(v) - (m + 10) > 0$. This contradicts the optimality of $\mathcal{P}$. This finishes the proof of the lemma. □

Next, we will show that the same holds for the list-gadgets. In order to do so, we first need the two following intermediary lemmas.

**Lemma 3.4.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $F$ be a list-gadget in $G$. There exists a set $C \in \mathcal{P}$ such that $|C \cap V_2(F)| \geq 3|V_2(F)|/4$.*

*Proof.* Assume that this is not true and let $max_{C \in \mathcal{P}}\{|C \cap V_2(F)|\} = x < 3|V_2(F)|/4$. We create a new partition $\mathcal{P}' = \{V(F), C_1 \setminus V(F), \ldots, C_p \setminus V(F)\}$. We will show that $v(\mathcal{P}) < v(\mathcal{P}')$. Notice that any edge that is not incident to a vertex of $V_2(F)$ is either in both $\mathcal{P}$ and $\mathcal{P}'$ or in neither of them. Therefore, we need to consider only the edges that are incident to a vertex of $V_2(F)$. Observe that any edge in $G[V(F)]$ is included in $E(\mathcal{P}')$. Thus, $E(\mathcal{P}) \setminus E(\mathcal{P}') \subseteq E(G[V_2(F) \cup U(F^v)] \setminus E(G[V_2(F)])$ (recall that $N[V_2(F)] \cap V_1(F^v) = U(F^v)$ and $N[V_2(F)] \setminus V_1(F^v) \subseteq V(F)$). Since $max_{C \in \mathcal{P}}\{|C \cap V_2(F)|\} = x < 3|V_2(F)|/4$, we have at most $3|V_2(F)|/2$ edges of $E(G[V_2(F) \cup V_1(F^v)] \setminus E(G[V_2(F)])$ in $E(\mathcal{P}) \setminus E(\mathcal{P}')$. Thus, $E(\mathcal{P}) \setminus E(\mathcal{P}') \leq 3|V_2(F)|/2$. We will now calculate the size of $E(\mathcal{P}') \setminus E(\mathcal{P})$. Since $max_{C \in \mathcal{P}}\{|C \cap V_2(F)|\} = x < 3|V_2(F)|/4$, for each vertex $v \in V_1(F)$ there are at least $|V_2(F)|/4$ edges incident to $v$ that are included in $E(\mathcal{P}') \setminus E(\mathcal{P})$. Therefore, $|E(\mathcal{P}') \setminus E(\mathcal{P})| \geq |V_1(F)||V_2(F)|/4$. Since $|V_1(F)| = m + 6 > 6$ we have that $v(\mathcal{P}) < v(\mathcal{P}')$, which contradicts to the optimality of $\mathcal{P}$. □

**Lemma 3.5.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $F$ a list-gadget in $G$. There exists a set $C \in \mathcal{P}$ such that $|C \cap V_2(F)| \geq 3|V_2(F)|/4$ and $V_1(F) \subseteq C$.*

*Proof.* By Lemma 3.4, we have that there exists a $C \in \mathcal{P}$ such that $|C \cap V_2(F)| \geq 3|V_2(F)|/4$. Assume that there exists a $v \in V_1(F) \setminus C$. We can assume that $|C| = \mathcal{C}$, as otherwise we could move $v$ into $C$ which would result in a $\mathcal{C}$-partition with higher value. Since $|C| = \mathcal{C}$ and $G[C]$ is connected, we know that $C$ includes vertices from $V_1(F^v)$, where $F^v$ is a vertex-gadget in $G$. Also, by Lemma 3.3, we know that $C \supseteq V(F^v)$. Since $|C| = \mathcal{C}$ and $G[C]$ is connected, we also have a vertex $u \in C \cap N[V_1(F^v)] \setminus (V_2(F^v) \cup V_2(F))$. Notice that $d(u) \leq m + 10$. We claim that replacing $u$ in $C$ by $v$ to $C$ will result in a $\mathcal{C}$-partition with higher value. Indeed, since $d(u) \leq m + 10$, removing $u$ from $C$ reduces the value of the partition by at most $m+10$. Moreover, $v$ has $3|V_2(F)|/4$ neighbors in $C$. Therefore, moving $v$ into $C$ increases the value of $\mathcal{P}$ by at least $|V_2(F)|/2$. Since $|V_2(F)|/2 > m + 10$, this is a contradiction to the optimality of $\mathcal{P}$. Thus $V_1(F) \subseteq C$. □

We are now ready to show that the vertices of any list-gadget will belong to the same set in any optimal $\mathcal{C}$-partition of $G$.

**Lemma 3.6.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $F$ be a list-gadget in $G$. There exists a set $C \in \mathcal{P}$ such that $V(F) \subseteq C$.*

*Proof.* By Lemma 3.5 we have that there exists a $C \in \mathcal{P}$ such that $|C \cap V_2(F)| \geq 3|V_2(F)|/4$ and $V_1(F) \subseteq C$. Assume that there exists a vertex $u \in C \setminus V_2(F)$. We can assume that $|C| = \mathcal{C}$ as otherwise we could include $u$ into $C$ and this would result in a $\mathcal{C}$-partition with a higher value (as most of the neighbors of $u$ are in $C$). Since $|C| = \mathcal{C}$ and $G[C]$ is connected, we know that $C$ includes vertices from $V_1(F^v)$, where $F^v$ is a vertex-gadget in $G$. Also, by Lemma 3.3, we know that $C \supseteq V(F^v)$.

Since $C \supseteq V(F^v)$, we can conclude that there is no other list-gadget $F'$ in $G$ such that $V_1(F') \cap C \neq \emptyset$. Indeed, since $V_1(F') \cap C \neq \emptyset$ and by Lemma 3.5, we have that $|C \cap V_2(F')| \geq 3|V_2(F')|/4$ and, thus, that $|C| > \mathcal{C}$. Since $|C| = \mathcal{C}$ and $G[C]$ is connected, we need to include vertices from $N(V_1(F^v)) \setminus (V_2(F^v) \cup V_2(F))$ in $C$. Also, since we have concluded that there is no list-gadget $F'$ in $G$ such that $V_1(F') \cap C \neq \emptyset$, any vertex $w \in C$ such that $w \in N(V_1(F^v)) \setminus (V_2(F^v) \cup V_2(F))$ has $|N(w) \cap C| \leq 6$. We claim that replacing $u$ in $C$ by $v$ will result in a $\mathcal{C}$-partition with a higher value. Indeed, since $|N(u) \cap C| \leq 6$, removing $u$ from $C$ will reduce the value of the partition by at most 4. Also, since $v$ has at least $d(v) - 1$ of its neighbors in $C$, moving it into $C$ increases the value of the partition by at least $d(v) - 1 \geq m + 6 + 2 - 1 > 6$. This is a contradiction to the optimality of $\mathcal{P}$. Thus, $V_1(F) \subseteq C$. □

As we already mentioned, it follows from Lemmas 3.3 and 3.6 that for any optimal $\mathcal{C}$-partition $\mathcal{P}$ of $G$, any set $C \in \mathcal{P}$ that includes a vertex-gadget $F^v$ can also include at most one list-gadget $F$. We will show that any such set $C$ must, actually, include exactly one list-gadget.

**Lemma 3.7.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $F^v$ a vertex-gadget in $G$. Let $C \in \mathcal{P}$ be the set such that $V(F^v) \subseteq C$. There exists a list-gadget $F$ such that $N(V_1(F^v)) \cap V(F) \neq \emptyset$ and $V(F) \subseteq C$.*

*Proof.* By Lemma 3.3 we have that, for any vertex-gadget $F^v$, there exists a set $C \in \mathcal{P}$ such that $V(F^v) \subseteq C$. We will show that $C$ also includes a list-gadget $F$ and that $N(V_1(F^v)) \cap V(F) \neq \emptyset$. Assume that this is not true, and let $F$ be any list-gadget such that $N(V_1(F^v)) \cap V(F) \neq \emptyset$. We can assume that $|C| \geq \mathcal{C} - |V(F)|$ as otherwise we could include $V(F)$ in $C$ and create a $\mathcal{C}$-partition of higher value than $\mathcal{P}$. By the size of $F^v$, the assumption that $|C| \geq \mathcal{C} - |V(F)|$, and Lemma 3.3, we have that $C \setminus V(F^v) \subseteq V_E$. Let $S = V(F) \cup V(F^v)$ and $\mathcal{P}' = \{S, C_1 \setminus S, \ldots, C_p \setminus S\}$. We claim that $v(\mathcal{P}) < v(\mathcal{P}')$. We will calculate the values $|E(\mathcal{P}) \setminus E(\mathcal{P}')|$ and $|E(\mathcal{P}') \setminus E(\mathcal{P})|$. Notice that the only edges that may belong in $E(\mathcal{P}) \setminus E(\mathcal{P}')$ are the edges between $V_1(F^v)$ and $V_E$. This means that $|E(\mathcal{P}) \setminus E(\mathcal{P}')| \leq 8n$ (since there are less than $n$ edges incident to $v$ in $H$ and 8 edges between $V(F^v)$ and $V_e$, for any $e$ incident to $v$). As for $|E(\mathcal{P}') \setminus E(\mathcal{P})|$, since the edges between $|U(F^v)|$ and $|V_2(F)|$ do not contribute to $\mathcal{P}$, we have that $|E(\mathcal{P}') \setminus E(\mathcal{P})| \geq |U(F^v)| \cdot |V_2(F)|$. Since $|V_2(F)| > 5A - 2n > 3n$ (for any list-gadget, and sufficiently large $n$) and $|V_1(F^v)| = 4$, we can conclude that $|V_1(F^v)| \cdot |V_2(F)| > 8n$. Therefore, $|E(\mathcal{P}) \setminus E(\mathcal{P}')| < |E(\mathcal{P}') \setminus E(\mathcal{P})|$, which contradicts the optimality of $\mathcal{P}$. □

Finally, we will show that any vertex $u \in V_e$ must be in a set that includes either vertices from $V(F^e)$ or vertices from $V(F^u) \cup V(F^v)$, where $e = uv$. Formally:

**Lemma 3.8.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$, $w \in V_e$, for some $e = uv \in E(H)$, and $w \in C$ for some $C \in \mathcal{P}$. If $V(F^u) \cap C = \emptyset$ and $V(F^v) \cap C = \emptyset$ then $V(F^e) \cup \{w\} \subseteq C$.*

*Proof.* It follows by Lemma 3.3 that there exists a $C' \in \mathcal{P}$ such that $V(F^e) \subseteq C' \subseteq V(F^e) \cup V_e$. Indeed, assuming otherwise, $C'$ would include vertices from a vertex-gadget, thus, and $|C'| > \mathcal{C}$, a contradiction. Assume that $V(F^u) \cap C = \emptyset$ and $V(F^v) \cap C = \emptyset$. If $C' \neq C$ then $w$ contributes 0 edges to the value of $\mathcal{P}$ since $N(w) \cap C = \emptyset$. Now, since $C' \subseteq V(F^e) \cup V_e$, and $|V(F^e)| = \mathcal{C} - 4$ we know that we always can move $w$ to $C'$ and increase the value of the partition. Therefore, $C' = C$. □

Next, we will calculate the absolute maximum value of any $\mathcal{C}$-partition of $G$. Notice that, in any optimal $\mathcal{C}$-partition, we have two kind of sets; those that include vertices of vertex or list-gadget and those that include vertices from edge-gadgets. We separate the sets of any optimal $\mathcal{C}$-partition of $G$ based on that. In particular, for an optimal $\mathcal{C}$-partition $\mathcal{P}$, we define $\mathcal{P}_V$ and $\mathcal{P}_E$ as follows. We set $\mathcal{P}_E \subseteq \mathcal{P}$ such that $C \in \mathcal{P}_E$ if and only if there exists an edge-gadget $F^e$ such that $V(F^e) \subseteq C$. Then, we set $\mathcal{P}_V = \mathcal{P} \setminus \mathcal{P}_E$.

It is straightforward to see that the previous lemmas also hold optimal $\mathcal{C}$-partitions $\mathcal{P}'$ of $G[V(\mathcal{P}_E)]$ and $\mathcal{P}''$ of $G[V(\mathcal{P}_V)]$. Indeed, assuming otherwise, we could create a $\mathcal{C}$-partition for $G$ of higher value since $\mathcal{P}$ is the concatenation of $\mathcal{P}'$ and $\mathcal{P}''$.

Notice now that for any vertex in $V(G) \setminus V_E$, we know whether it belongs in $V(\mathcal{P}_V)$ or in $V(\mathcal{P}_E)$. However, this is not true for the vertices of $V_E$. We will assume that $V(\mathcal{P}_V)$ includes $x$ vertices from $V_E$ and we will use this in order to provide an upper bound to the value of $|E(\mathcal{P}_V)|$ and $|E(\mathcal{P}_E)|$.

Let us now consider an optimal partition $\mathcal{P}$, let $S = V(\mathcal{P}_E) \cap V_E$, $x = |S|$ and $y = |V_E \setminus S|$.

We start with the upper bound of $|E(\mathcal{P}_V)|$. Let $F^v$ be a vertex-gadget. Recall that, there are $n$ list-gadgets adjacent to $F^v$ and, by Lemma 3.7, exactly one of them is in the same set as $F^v$ in any optimal $\mathcal{C}$-partition. Let $F$ be a list-gadget that is not in the same set as $F^v$ in $\mathcal{P}$ (and thus in $\mathcal{P}_V$). By Lemma 3.6, we know that all vertices of $F$ are in the same set of $\mathcal{P}_V$. Thus, for each one of them, we have $m_\ell$ edges in $E(\mathcal{P}_V)$. Since there are $n-1$ such list-gadgets for each one of the $n$ vertex-gadgets, in total we have $n(n-1)m_\ell$ edges that do not belong in the same set as a vertex-gadget.

Now, let $F$ be the list-gadget such that the vertices of $V(F)$ and $V(F^v)$ are in the same set $C \in \mathcal{P}_V$. Let $\alpha_v$ be the value represented by the list-gadget $F$. Since $|V(F^v) \cup V(F)| = \mathcal{C} - 2\alpha_v$, at most $2\alpha_v$ of the $y$ vertices of $V_E$ can be in $C$. Let $|C \cap V_E| = y_v \leq 2\alpha_v \leq y$. Since these vertices must be incident to $V_1(F^v)$, we have that $|E(G[C])| = m_v + m_\ell + 4y_v + 2(5A - 2\alpha_v) = m_v + m_\ell + 10A + 4y_v - 4\alpha_v$; the $2(5A - 2\alpha_v)$ term comes from the fact that exactly 2 vertices of $V_1(F^v)$ are adjacent to all the vertices of $V_2(F)$. By counting all sets that include vertices from vertex-gadgets we have that

$$m_v n + m_\ell n + 10A + 4 \sum_{v \in V(H)} y_v - 4 \sum_{v \in V(H)} \alpha_v$$

In total:

$$|E(\mathcal{P}_V)| = m_v n + m_\ell n^2 + 10nA + 4 \sum_{v \in V(H)} y_v - 4 \sum_{v \in V(H)} \alpha_v,$$

where $n = |V(H)|$.

Now, we will calculate an upper bound of $|E(\mathcal{P}_E)|$ and we will give some properties that must be satisfied in order to achieve this maximum. Let $S = V_E \cap V(\mathcal{P}_E)$. By Lemma 3.8, we have that $\mathcal{P}_E$ consists of the vertex sets of the connecting components of $G[V(\mathcal{P}_E)]$. Thus, in order compute an upper bound of $|E(\mathcal{P}_E)|$, it is suffices to find an upper bound of the number of edges in $G[S' \cup \bigcup_{e \in E(H)} V(F^e)]$, for any set $S' \subseteq V_E$ where $|S'| = |S|$.

For any $G[S' \cup \bigcup_{e \in E(H)} V(F^e)]$, where $S' \subseteq V_E$, we define types of its connected components based on the size of their intersection with $V_E$. In particular, let $\mathcal{X} = \{C_1 \ldots, C_p\}$ be the vertex sets of the connected component of $G[S \cup \bigcup_{e \in E(H)} V(F^e)]$. For any set $C \in \mathcal{X}$, we have $|C \cap V_E| = i$, where $i \in \{0, 1, 2, 3, 4\}$. We set $\mathcal{X}_i = \{C \in \mathcal{X} \mid |C \cap V_E| = i\}$. Notice that, $\sum_{i=0}^{4} |\mathcal{X}_i| = |E(H)|$.

We claim that, in order to maximize the number of edges in $G[S \cup \bigcup_{e \in E(H)} V(F^e)]$, we would like to have as many sets in $\mathcal{X}_4$ as possible. Formally:

**Lemma 3.9.** *For any $0 \leq x \leq |V_E|$, let $S$ be a subset of $V_E$ such that $|S| = x$ and $|E(G[U_E \cup S])| = \max_{S' \subseteq V_E, |S'| = x} |E(G[U_E \cup S'])|$. Assume that $\mathcal{X} = \{C_1 \ldots, C_p\}$ are the vertex sets of the connected components of $G[U_E \cup S]$. Then, we have that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 0$ if $x \bmod 4 = 0$ and $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 1$ otherwise.*

*Proof.* First, we will prove that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| \leq 1$. Assume that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| > 1$. We will show that there exists a set $S'$ such that $|S'| = x$ and $|E(G[U_E \cup S])| < |E(G[U_E \cup S'])|$. Let $C^1$ and $C^2$ be two sets in $\mathcal{X}$ such that $C^1, C^2 \notin \mathcal{X}_0 \cup \mathcal{X}_4$. Let $C^1 \in \mathcal{X}_{\ell_1}$ and $C^2 \in \mathcal{X}_{\ell_2}$. We consider two cases: first, $\ell_1 + \ell_2 \leq 4$ and then $\ell_1 + \ell_2 > 4$.

**Case 1. $\boldsymbol{\ell_1 + \ell_2 \leq 4}$.** Let $F^{e_1}$ and $F^{e_2}$ be the edge-gadgets such that $V(F^{e_1}) \subseteq C_1$ and $V(F^{e_2}) \subseteq C_2$. We modify the sets $C^1$ and $C^2$ as follows:

- We replace $C^1$ with $C^{1'} = C^1 \setminus V_{e_1}$ and
- we replace $C^2$ with $C^{2'} = C^2 \setminus V_{e_2} \cup Y$ where $Y \supseteq \{w^e_{L1}, w^e_{R1}\}$ and $|Y| = \ell_1 + \ell_2$.

Let $S' = Y \cup (S \setminus (V_{e_1} \cup V_{e_2}))$ and let us denote the resulting partition by $\mathcal{X}'$. Notice that $\ell_1 + \ell_2 \geq 2$. So we can always have $Y \supseteq \{w^e_{L1}, w^e_{R1}\}$ and $|Y| = \ell_1 + \ell_2$. Also, the number of vertices from $|V(\mathcal{X}) \cap V_E| = |V(\mathcal{X}') \cap V_E| = x$. It remains to show that $|E(\mathcal{X}')| > |E(\mathcal{X})|$. It suffices to show that $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$. To achieve that, we need to consider three sub-cases, $\ell_1 + \ell_2 = 2$, $\ell_1 + \ell_2 = 3$ or $\ell_1 + \ell_2 = 4$.

**Case 1.a.** $\boldsymbol{\ell_1 + \ell_2 = 2}$. Since $\ell_1 \geq 1$ and $\ell_2 \geq 1$, we have that $\ell_1 = \ell_2 = 1$. Thus, by the construction of $G$, $|E(G[C^1])| = |E(G[C^2])| = |E(G[F^{e_1}])| + 1$ (as all edge-gadgets have the same number of edges). Also, since $\ell_1 + \ell_2 = 2$, we get that $Y = \{w^e_{L1}, w^e_{R1}\}$. This, by the construction of $G$, gives us that $|E(G[C^{1'}])| = |E(G[F^{e_1}])|$ and $|E(G[C^{2'}])| = |E(G[F^{e_2}])| + 3$. Therefore, $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$.

**Case 1.b.** $\boldsymbol{\ell_1 + \ell_2 = 3}$. Since $\ell_1 \geq 1$ and $\ell_2 \geq 1$ we have that either $\ell_1 = 2$ and $\ell_2 = 1$ or $\ell_1 = 1$ and $\ell_2 = 2$. Assume, w.l.o.g., that $\ell_1 = 2$ and $\ell_2 = 1$. By the construction of $G$, we have that $|E(G[C^1])| \leq |E(G[F^{e_1}])| + 3$ and $|E(G[C^2])| \leq |E(G[F^{e_2}])| + 1$. Also, since $\ell_1 + \ell_2 = 3$ and $Y \supseteq \{w^e_{L1}, w^e_{R1}\}$, we have that $|E(G[C^{1'}])| = |E(G[F^{e_1}])|$ and $|E(G[C^{2'}])| = |E(G[F^{e_2}])| + 5$. Therefore, $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$.

**Case 1.c.** $\boldsymbol{\ell_1 + \ell_2 = 4}$. Since $\ell_1 \geq 1$ and $\ell_2 \geq 1$, we have that either $\ell_1 = \ell_2 = 2$ or one of the $\ell_1$ and $\ell_2$ is 1 and the other 3. In the first case, $|E(G[C^1])| = |E(G[C^2])| \leq |E(G[F^{e_1}])| + 3$ while in the second, $|E(G[C^1])| = |E(G[F^{e_1}])| + 1$ and $|E(G[C^1])| = |E(G[F^{e_1}])| + 5$. In both cases, $|E(G[C^1])| + |E(G[C^2])| \leq 6$. Also, since $\ell_1 + \ell_2 = 4$, $Y = V_{e_2}$, we have that $|E(G[C^{1'}])| = |E(G[F^{e_1}])|$ and $|E(G[C^{2'}])| = |E(G[F^{e_2}])| + 8$. Therefore, $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$.

**Case 2.** $\boldsymbol{\ell_1 + \ell_2 > 4}$. Let $F^{e_1}$ and $F^{e_2}$ be the edge-gadgets such that $V(F^{e_1}) \subseteq C_1$ and $V(F^{e_2}) \subseteq C_2$. We modify the sets $C^1$ and $C^2$ as follows:

- We replace $C^1$ with $C^{1'} = C^1 \cup V_{e_1}$ and
- we replace $C^2$ with $C^{2'} = C^2 \setminus V_{e_2} \cup Y$ where $Y \subseteq \{w^e_{L1}, w^e_{R1}\}$ and $|Y| = \ell_1 + \ell_2 - 4$.

Indeed, it suffices to have $Y \subseteq \{w^e_{L1}, w^e_{R1}\}$ as $2 \leq \ell_1, \ell_2 \leq 3$, and thus, $\ell_1 + \ell_2 - 4 < 3$. We need to consider two cases, either $\ell_1 + \ell_2 = 5$ or $\ell_1 + \ell_2 = 6$

**Case 2.a.** $\boldsymbol{\ell_1 + \ell_2 = 5}$. In this case, we have that one of $\ell_1, \ell_2$ is equal to 2 while the other is equal to 3. W.l.o.g. let $\ell_1 = 2$ By the construction of $G$, we get that $|E(G[C^1])| \leq |E(G[F^{e_1}])| + 3$ (as all edge-gadgets have the same number of edges). Also, $|E(G[C^1])| \leq |E(G[F^{e_1}])| + 5$. Now, observe that $|E(G[C^{1'}])| = |E(G[F^{e_1}])| + 8$ and $|E(G[C^{1'}])| = |E(G[F^{e_1}])| + 1$. Therefore, $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$.

**Case 2.b.** $\boldsymbol{\ell_1 + \ell_2 = 6}$. In this case, we have that one of $\ell_1 = \ell_2 = 3$. By the construction of $G$, we obtain that $|E(G[C^1])| = |E(G[C^2])| = |E(G[F^{e_1}])| + 5$ (as all edge-gadgets have the same number of edges). We also have that $|E(G[C^{1'}])| = |E(G[F^{e_1}])| + 8$ and $|E(G[C^{1'}])| = |E(G[F^{e_1}])| + 3$ (since $Y = \{w^e_{L1}, w^e_{R1}\}$ in this case). Therefore, $|E(G[C^{1'}])| + |E(G[C^{2'}])| > |E(G[C^1])| + |E(G[C^2])|$.

To sum up, we have that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| \leq 1$. We will now show that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 0$ if $x \ mod \ 4 = 0$ and $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 1$ otherwise.

Assume that $x \ mod \ 4 = 0$. Notice that $4|\mathcal{X}_4| + 3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1| + 0|\mathcal{X}_0| = x$; therefore $(4|\mathcal{X}_4| + 3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1| + 0|\mathcal{X}_0|) \ mod \ 4 = 0 \implies (3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1|) \ mod \ 4 = 0$. This implies that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 0$. Indeed, assuming otherwise we get that $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 1$, and thus $(3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1|) \ mod \ 4 = i$, for an $i \in [3]$. This is a contradiction to $(3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1|) \ mod \ 4 = 0$.

Next, assume that $x \ mod \ 4 = i$ for $i \in [3]$. Then we have that $4|\mathcal{X}_4| + 3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1| + 0|\mathcal{X}_0| = x \implies (4|\mathcal{X}_4| + 3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1| + 0|\mathcal{X}_0|) \ mod \ 4 = i \implies (3|\mathcal{X}_3| + 2|\mathcal{X}_2| + |\mathcal{X}_1|) \ mod \ 4 = i$. If $|\mathcal{X}_1| + |\mathcal{X}_2| + |\mathcal{X}_3| = 0$ then the previous implies that $i = 0$ which is a contradiction. This finishes the proof of this lemma. □

It follows that the maximum value of $\max_{S' \subseteq V_E, |S'| = x} |E(G[U_E \cup S'])|$ is

- $m_e|E(H)| + 8x/4$, when $x \ mod \ 4 = 0$
- $m_e|E(H)| + 8(x - i)/4 + x_i$, when $x \ mod \ 4 = i$

where $x_1 = 1$, $x_2 = 3$, $x_3 = 5$. Notice that $\max_{S' \subseteq V_E, |S'| = x} |E(G[U_E \cup S'])| \leq m_e|E(H)| + 2x$ where the equality holds only when $x \ mod \ 4 = 0$.

Thus, we have the following:

**Corollary 3.10.** *Given that $x = |S|$, the maximum value of $\mathcal{P}$ is: $v(\mathcal{P}) \leq m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)|+10A|V(H)|+8|E(H)|$. Also, this can be achieved only when $x$ mod $4 = 0$ and $x = |V_E|-y = \sum_{v\in H(v)} 2\alpha_v$.*

**The reduction.** Let $\mathcal{P}$ be an optimal $\mathcal{C}$-partition of $G$. We will prove that the following two statements are equivalent:

- $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$
- $(H, L)$ is a yes-instance of the General Factors problem.

Assume that $(H, L)$ is a yes-instance of the General Factors problem and let $E' \subseteq E(H)$ be the edge set such that, for any vertex $v \in V(H)$, we have $d_{H-E'}(v) \in L(v)$. We will create a $\mathcal{C}$-partition of $G$ that has value $m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$. For each edge $e \in E(H) \setminus E'$ we create a set $C_e = V(F^e)$ and for each $e \in E'$ we create a set $C_e = V(F^e) \cup V_e$. For each $v \in V(H)$, let $F$ be the list-gadget that represents the value $d_{H-E'}(v)$. The existence of such a list-gadget is guaranteed since $d_{H-E'}(v) \in L(v)$. Also, let $U_v$ be the subset of $V_E$ such that, for any $u \in U_v$, there exists an edge $e \in E(H) \setminus E'$ such that $u \in V_e$ and $u$ is incident to the vertices of $V_1(F^v)$ (this means that $e$ is incident to $v$ in $H$). Notice that the vertices in $U_v$ have not been included in any set $C_e$, for $e \in E(H)$, that we have created this far. Now, for each $v \in V(H)$, we create a set $C_v = V(F^v) \cup V(F) \cup U_v$. It remains to deal with the list-gadgets that have not yet been included in any set. We create the sets $C_1, \ldots, C_{n(n-1)}$, one for each one of them. We claim that $\mathcal{P} = \{C_e \mid e \in E(H)\} \cup \{C_v \mid v \in V(H)\} \cup \{C_1, \ldots, C_{n(n-1)}\}$ is a $\mathcal{C}$-partition of $G$ and $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$. Notice that any of the sets $C \in \{C_e \mid e \in E(H)\} \cup \{C_1, \ldots, C_{n(n-1)}\}$ have size at most $\mathcal{C}$ as they are either vertex sets of a list-gadget or a subset of $V(F^e) \cup V_e$, for some $e \in E(H)$. Thus we only need to show that $|C_v| \leq \mathcal{C}$ for all $v \in V(H)$. We have that $|V(F^v) \cup F| \subseteq C_v$ where $F$ is the list-gadget that represents the value $d_{H-E'}(v) \in L(v)$. Therefore, $|V(F^v) \cup F| = \mathcal{C} - 2d_{H-E'}(v)$. We claim that $|U_v| = 2d_{H-E'}(v)$. Recall that $U_v$ contains the vertices of $V_E$ for which there exists an edge $e \in E(H) \setminus E'$ such that $u \in V_e$ and $u$ is incident to the vertices of $V_1(F^v)$. Actually, there are exactly $d_{H-E'}(v)$ edges incident to $v$ from $E(H) \setminus E'$. Also, for each such edge $e$, two vertices of $V_e$ are incident to $V_1(F^v)$ (the vertices $w^e_{L1}, w^e_{L2}$ if $v \in V_L$ and $w^e_{R1}, w^e_{R2}$ if $v \in V_R$). Thus $|U_v| = 2d_{H-E'}(v)$ and $|C_v| = \mathcal{C}$.

It remains to argue that $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$. First, notice that, the vertex set $V(F)$ of any gadget $F$ belongs to one set. Thus, every edge of $E(G[V(F)])$, contributes in the value of $\mathcal{P}$. This give us $m_v$ edges for each vertex-gadget, $m_e$ edges for each edge-gadget and $m_\ell$ edges for each list-gadget. Since we have $|V(H)|$ vertex-gadgets, $|E(H)|$ edge-gadgets and $|V(H)|^2$ list-gadgets, this gives $m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)|$ edges (up to this point). We also need to compute the number of edges in $E(\mathcal{P})$ that do not belong in any set $E(G[V(F)])$, for any gadget $F$. Let $S$ be the set $E(\mathcal{P}) \setminus \bigcap_{F \text{ is any gadget}} E(G[V(F)])$. Notice that, for any $C \in \mathcal{P}$, we have $S \cap C \neq \emptyset$ if and only if there exists a vertex or edge-gadget $F$ such that $V(F) \subseteq C$.

First we consider a set $C$ that includes a vertex-gadget. By construction, we have that $C$ includes the vertices of a vertex-gadget $F^v$, the vertices of a list-gadget $F$ that represents an integer $\alpha_v$ and $2\alpha_v$ vertices from the set $V_E$. There are exactly $|U^v| \cdot |V_1(F)|$ edges between $F^v$ and $F$. Also, for any vertex $u \in C \cap V_E$, we have that $N(u) \cap C = V_1(F^v)$. Thus, we have $|U^v| \cdot |V_1(F)| + |C \cap V_E| \cdot |V_1(F^v)| = 2(5A - 2\alpha) + 2\alpha \cdot 4 = 10A + 4\alpha_v$ edges. Also, notice that, by construction, $C \cap V_E$ is an independent set. Thus we have no other edges to count. This give us $10nA + 4\sum_{v\in V(H)} \alpha_v$.

Now we consider a set $C$ that includes an edge-gadget. By the construction of $C$ we have that there exists an edge $e \in E(H)$ such that either $C = V(F^e)$ or $C = V(F^e) \cup V_e$. Therefore, if $e \in E'$ then $C = V(F^e) \cup V_e$ and $E(G[C])$ includes 8 edges incident to vertices of $V_E$, while if $e \notin E'$, then $C = V(F^e)$ and $E(G[C])$ does not include edges incident to vertices of $V_E$. This give us $8|E'|$ extra edges.

In order to complete the calculation of $|S|$ we need to observe that the values $\alpha_v$, $v \in V(H)$ and $|E'|$ are related. In particular, by the selection of $\alpha_v$, we have that $\sum_{v\in V(H)} d_{H-E'}(v) = \sum_{v\in V(H)} \alpha_v = 2|E(H) \setminus E'|$. It follows that: $|S| = 10nA + 4\sum_{v\in V(H)} \alpha_v + 8|E'| = 10nA + 8|E(H) \setminus E'| + 8|E'| = 10nA + 8|E(H)|$.

In total, $|E(\mathcal{P})| = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$.

For the reverse direction, assume that we have a $\mathcal{C}$-partition $\mathcal{P}$ of $G$ such that $v(\mathcal{P}) = m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$. By the calculated upper bounds, we have that,

- $|E(\mathcal{P}_E)| = m_e|E(H)| + 2x$ and
- $|E(\mathcal{P}_V)| = m_v n + m_\ell n^2 + 10nA + 4\sum_{v\in V(H)} y_v - 4\sum_{v\in V(H)} \alpha_v$.

Also, in order to achieve $m_v|V(H)| + m_\ell|V(H)|^2 + m_e|E(H)| + 10A|V(H)| + 8|E(H)|$, we have that $\sum_{v\in V(H)} \alpha_v = (|V_E| - x)/2$. Recall that in order to achieve the maximum value, for any edge $e \in E(H)$, either $V_e \subset \mathcal{P}_V$ or $V_e \subset \mathcal{P}_E$. Let $E' = \{e \in E(H) \mid V_e \subset \mathcal{P}_E\}$. We claim that for any $v \in V(H)$, we have $d_{H-E'}(v) \in L(v)$. Let $V(F^v) \subseteq C$, for some $C \in \mathcal{P}$, $F$ be the list-gadget such that $F \subseteq C$ and $|C \cap V_E| = x_v$. By Corollary 3.10 we obtain that $2\alpha_v = x_v$ where $a_v$ is the value represented by $F$, if the partition is of optimal value. Observe that, for any edge $e \in E(H) \setminus E'$ incident to $v$, two vertices of $V_e$ are in $C \cap V_E$. Thus, $2d_{H-E'}(v) = x_v$. Since $2\alpha_v = x_v$ and $\alpha_v \in L(v)$ (by the construction of list-gadgets) we obtain that $d_{H-E'}(v) = \alpha_v \in L(v)$. Thus $(H, L)$ is a yes-instance of the General Factors problem.

**The tree-depth of G is bounded.** The only thing that remains to be shown is that the tree-depth of $G$ is bounded by a computable function of $m$. Recall that $m$ is the size of one of the bipartitions of $H$. W.l.o.g., assume that $|V_L| = m$. We start by deleting the set $V_1(F^v)$, for all $v \in V_L$. This means that we have deleted $4m$ vertices. Now, we will calculate an upper bound of the tree-depth of the remaining graph. In the new graph, there are connected components that include vertices from vertex-gadgets $F^v$, for $v \in V_R$, but no connected component includes two such gadgets. For each such a component, we delete the vertices $V_1(F^v)$, for each $v \in V_R$. Since these deletion are in different components, they are increasing the upper bound of the tree-depth of the original graph by 4. Also, after these deletion, any connected component that remains is:

- either a list-gadget $F$,
- or isomorphic to $G[V_e \cup V(F^e)]$ (for any $e \in E(H)$),
- or isomorphic to $G[V_2(F^v) \cup V_3(F^v)]$ (for any $e \in E(H)$).

We claim that, in any of these cases, the tree-depth of this connected component is at most $\mathcal{O}(m)$. Consider a list-gadget $F$. Any $G[V(F)]$ had tree-depth at most $m + 1$. This holds because, if we remove $V_1(F)$, we remain with a set of independent vertices plus a matching. Consider a connected component isomorphic to $G[V_e \cup V(F^e)]$. Observe that $G[V_e \cup V(F^e)]$ has tree-depth $2m + 5$ because removing $V_1(F^e) \cup V_3(F^e) \cup V_e$ results to an independent set and $|V_1(F^e) \cup V_3(F^e) \cup V_e| = 2m + 5$ for all $e \in E(H)$. Finally, consider a connected component isomorphic to $G[V_2(F^v) \cup V_3(F^v)]$. In this case, the tree-depth of this component is upperly bounded by $2m$ since by deleting $V_3(F^v)$, we end up with an independent set.

In total, the tree-depth of $G$ is upper bounded by $3m + 9$. This completes the proof. □

## 3.2 Graphs of bounded vertex cover number

**Theorem 3.11.** *Given a weighted graph $G = (V, E)$ with vertex cover number* vc*, there exists an algorithm that computes an optimal $\mathcal{C}$-partition of $G$ in time* $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}n^{\mathcal{O}(1)}$.

*Proof.* Let $U$ be a vertex cover of $G$ of size vc and let $I$ be the independent set $V \setminus U$. If such a vertex cover is not provided as input, we can compute one in time $2^{\mathrm{vc}}n^{\mathcal{O}(1)}$ time [17]. First, observe that there can be at most vc many coalitions in $G$ which can have a positive contribution (since the contribution comes from edges and each edge in $G$ is incident to some vertex in $U$). Next, we guess $\mathcal{P}' = \{C_1, \ldots, C_p\}$ (here, $p \leq \mathrm{vc}$), the intersection of the sets of an optimal $\mathcal{C}$-partition of $G$ with $U$; let $W = v(\mathcal{P}')$. Notice that we can enumerate all $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$ partitions of $U$ in $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$ time.

Next, for each $\mathcal{P}'$ we do the following (in $n^{\mathcal{O}(1)}$ time). We create a new graph $G'$ as follows. First, we create the vertex sets $S_i$, where $|S_i| = \mathcal{C} - C_i$ for each $i \in [p]$. Then, we add all the edges between

the vertices of $x \in I$ and $S_i$ if $v \in N(C_i)$, for every $i \in [p]$. Formally, $S_i = \{u_1^i, \ldots, u_{S_i}^i\}$ for $i \in [p]$, $V(G') = \bigcup_{i \in [p]} S_i$, and $E(G') = \{u_j^i x \mid x \in N(C_i) \cap I, i \in [p], j \in [S_i]\}$. Finally, for every edge $xy$, with $x \in S_i$ and $y \in I$, we set the weight $w(xy) = \sum_{u \in C_i \wedge uy \in E} w(uy)$, i.e., to be equal with to much $W$ would increase if $y$ was added to $C_i$. Now, observe that in order to compute an optimal $\mathcal{C}$-partition of $G$ whose intersection with $U$ is $\mathcal{P}'$, it suffices to find a maximum weighted matching of $G'$, which can be done in polynomial time [26]. Since we do this operation for each possible intersection of the $\mathcal{C}$-partition with $U$, all of which can be enumerated in $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})}$ time, we can compute an optimal $\mathcal{C}$-partition of $G$ in time $\mathrm{vc}^{\mathcal{O}(\mathrm{vc})} n^{\mathcal{O}(1)}$. □

The following theorem establishes that our algorithm from Theorem 3.11, which enumerates all possible intersections of a smallest size vertex cover with an optimal solution, is optimal asymptotically assuming ETH.

**Theorem 3.12.** *Given an unweighted graph $G$ of vertex cover number* vc*, there is no algorithm that computes an optimal $\mathcal{C}$-partition of $G$ in time $(\mathcal{C}\mathrm{vc})^{o(\mathrm{vc})} n^{\mathcal{O}(1)}$, unless the ETH fails.*

*Proof.* We will present a reduction from a restricted version of 3-SAT problem.

**Definition 3.13** (R3-SAT)**.** *In this version of SAT, the input consists of a 3-SAT formula $\phi$ defined on a set of variables $X$ and a set of clauses $C$. Additionally, we have that:*

- *each variable appears at most four times in $C$ and*
- *the variable set $X$ is partitioned into $X_1 \cup X_2 \cup X_3$, such that every clause includes at most one variable from each one of the sets $X_1$, $X_2$ and $X_3$.*

*The question is whether there exists a truth assignment to the variables of $X$ that satisfies $\phi$.*

**Lemma 3.14.** *The R3-SAT problem is NP-hard. Also, under the ETH, there is no algorithm that solves this problem in time $2^{o(n+m)}$, where $n$ is the number of variables and $m$ is the number of clauses.*

*Proof.* The reduction is from 3-SAT. First we make sure that each variable appears at most four times. Assume that variable $x$ appears $k > 3$ times. We create $k$ new variables $x_1, \ldots, x_k$ and replace the $i$-th appearance of $x$ with $x_i$. Finally, we add the clauses $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \ldots (x_k \vee \neg x_1)$. This procedure is repeated until there is no variable that appear more than 3 times.

Next, we create an instance where the variables are partitioned in the wanted way. First, we fix the order that the variables appear in each clause. Let $x$ be any variable that appears in the formula. If $x$ appears only in the $i$-th position of every clause it is part of (for some $i \in [3]$), then we add $x$ into $X_i$. Otherwise, we create three new variables $x_1, x_2, x_3$ and, for each clause $c \in C$, if $x$ appears in the $i$-th position of $c$, we replace it with $x_i$. Notice that, at the moment, $x_i$ appears at most twice for each $i \in [3]$. We add $x_i$ in the set $X_i$, for all $i \in [3]$. Also, we add the clauses $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1)$. Thus, in any satisfying assignment of the formula, the variables $x_1$, $x_2$ and $x_3$ have the same assignment. Notice that in each one of the original clauses, the $i$-th literal contains a variable from $X_i$. Therefore, each one of the original clauses have at most one variable from $X_i$ for each $i \in [3]$. This is also true for all the clauses that were added during the construction.

It is easy to see that the constructed formula is satisfiable if and only if $\phi$ is also satisfiable.

Finally, notice that the number of variables and clauses that were added is linear in regards to $n+m$. Therefore, we cannot have an algorithm that runs in $2^{o(n+m)}$ and decides whether the new instance is satisfiable unless the ETH is false. □

**The construction.** Let $(X, C)$ be an instance of the R3-SAT problem, and let $X = X_1 \cup X_2 \cup X_3$ be the partition of $X$ as it is defined above. We may assume that $|X_1| = |X_2| = |X_3| = n = 2^k$ for some $k \in \mathbb{N}$. If this is not the case, we can add enough dummy variables that are not used anywhere just to make sure that this holds. We can also assume that $k$ is an even number; if not, we can double the variables to achieve that. Notice that the number of additional dummy variables is at most

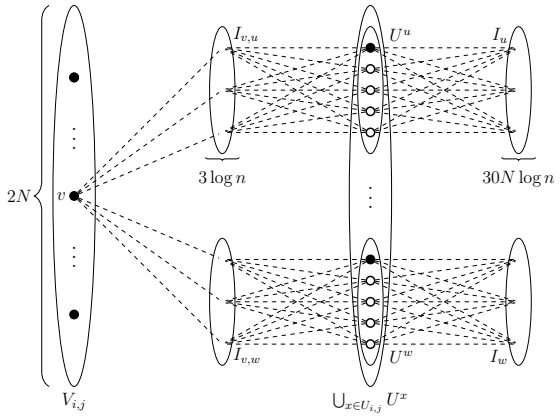



Figure 3: The gadget $G_{i,j}$ used in the construction of Theorem 3.12

$2max\{|X_1|, |X_2|, |X_3|\}$, so that the number of variables still remains linear in regards to $n + m$. The construction is illustrated in Figure 3.

We start by partitioning each variable set $X_i$ in to $k = \log n$ sets $X_{i,1} \ldots, X_{i,k}$, with $|X_{i,j}| \leq \lceil n/\log n \rceil$ for every $j \in [k]$.

For each set $X_{i,j}$, we construct a variable gadget $G_{i,j}$ as follows:

- First, we create a vertex set $V_{i,j}$ with $2N = 2\lceil n/\log^2 n \rceil$ vertices. Each vertex in $V_{i,j}$ represents at most $\frac{\log n}{2}$ variables. To see that we have enough vertices to achieve this, observe that $X_{i,j}$ represents a set of $\lceil n/\log n \rceil$ variables. Thus: $\left\lceil \frac{\lceil \frac{n}{\log n} \rceil}{2\lceil \frac{n}{\log^2 n} \rceil} \right\rceil = \left\lceil \frac{n}{2\log n \lceil \frac{n}{\log^2 n} \rceil} \right\rceil \leq \left\lceil \frac{n}{2\log n \frac{n}{\log^2 n}} \right\rceil = \left\lceil \frac{\log n}{2} \right\rceil = \frac{\log n}{2}$ where the last equality holds because $\log n$ is assumed to be an even number. Hereafter, let $X(v)$ be the variable set that is represented by $v$. Also notice that $X(v) \subseteq X_{i,j} \subseteq X_i$ for all $v \in V_{i,j}$.
- Then we create the set of vertices $U_{i,j} = \{u_\ell \mid \ell \in [\sqrt{n}]\}$. Hereafter, we will call these *assignment vertices*. Now, for each vertex $v \in V_{i,j}$ and each assignment over the variable set $X(v)$, we want to have a vertex of $U_{i,j}$ represent this assignment. Since $|X(v)| \leq \frac{\log n}{2}$, there are at most $2^{\frac{\log n}{2}}$ different assignments over the variable set $X(v)$. Therefore, we can select the variables of $U_{i,j}$ to represent the assignments over $X(v)$ in a way such that each assignment is represented by at least one vertex and no vertex represents more than one assignment. Notice that $U_{i,j}$ contains enough vertices to achieve this since $|U_{i,j}| = \sqrt{n}$. We are doing the same for all vertices in $V_{i,j}$.
- We proceed by creating four copies $u^1, \ldots, u^4$ of each vertex $u \in U_{i,j}$. For each assignment vertex $u$, let $U^u$ be the set $\{u, u^1, \ldots, u^4\}$. For each set $U^u$, we add an independent set $I_u$ of size $30N \log n$. Then, for each vertex $v \in I_u$ we add all the edges between $v$ and the vertices of $U^u$.
- Finally, for each pair $(v, u) \in V_{i,j} \times U_{i,j}$, we create an independent set $I_{v,u}$ of $3\log n$ vertices and, for all $w \in I_{v,u}$ and $x \in U^u \cup \{v\}$, we add the edge $wx$.

This concludes the construction of $G_{i,j}$, which corresponds to the set $X_{i,j}$. We repeat this process for all sets $X_{i,j}$.

Let $V_C$ be the set of clause vertices, which contains a vertex $v_c$ for each $c \in C$. We add the vertices of $V_C$ to the graph we are constructing. The edges incident to the vertices of $V_C$ are added as follows.

Let $c \in C$, $l$ be a literal that appears in $c$, $x$ be the variable that appears in $l$ and $v$ the variable vertex such that $x \in X(v)$. We first add the edge $v_c v$. Now, consider the $(i, j) \in [3] \times [\log n]$ such that $v \in V_{i,j}$. For each vertex $u$ in $U_{i,j}$ we add the edge $uv_c$ if and only if $l$ becomes true by the assignment over $X(v)$ represented by $u$.

Let $G$ be the resulting graph. Finally, set $\mathcal{C} = 42N \log n$ to be the capacity of the cars, where, recall, $N = \lceil n/\log^2 n \rceil$. This finishes our construction.

**Properties of optimal $\mathcal{C}$-partitions of $G$.** First, we identify the structural properties of any optimal $\mathcal{C}$-partition of $G$ that are going to be used in the reduction. In particular we will show that for any optimal $\mathcal{C}$-partition $\{C_1, \ldots, C_p\}$ of $G$, we have that:

- for any $k \in [p]$, if $\{u\} \subseteq C_k$ for some assignment vertex $u$, then $U^u \subseteq C_k$;
- for any $k \in [p]$, $|C_k \cap \bigcup_{(i,j)\in[3]\times[\log n]} U_{i,j}| \leq 1$;
- for any $(i,j) \in [3] \times [\log n]$ and $v \in V_{i,j}$, if $v \in C$ then $C \cap U_{i,j} \neq \emptyset$.

**Lemma 3.15.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. Let $u \in U_{i,j}$ for some $(i,j) \in [3] \times [\log n]$ and $u \in C_k$ for some $k \in [p]$. Then, $U^u \cap C_k = U^u$.*

*Proof.* Assume that, for some $(i,j) \in [3] \times [\log n]$, there exists a set $U^u$ for an assignment vertex $u \in U_{i,j}$ such that $u \in C_k$, for some $k \in [p]$, and $U^u \cap C_k \neq U^u$. We will show that, in this case, $\mathcal{P}$ is not an optimal $\mathcal{C}$-partition of $G$. Indeed, consider the following $\mathcal{C}$-partition of $G$. First set $C = U^u \cup N(U^u) \setminus V_C$. Then, let $\mathcal{P}' = \{C, C_1 \setminus C, \ldots, C_p \setminus C\}$. Notice that $\mathcal{P}'$ is a $\mathcal{C}$-partition. Indeed, $|C| = 5 + 30N \log n + 2N3 \log n \leq 42N \log n = \mathcal{C}$ and $|C_i \setminus C| \leq |C_i| \leq \mathcal{C}$ as $C_i \in \mathcal{P}$ for all $i \in [p]$.

Now, we will show that $v(\mathcal{P}') > v(\mathcal{P})$. First observe that for every $v \in C$, we have that:

- $v \in U^u$, or
- $v \in V_u$ where $V_u = \{v \mid N(v) = U^u\}$, or
- $v \in V'_u$ where $V'_u = \{v \mid N(v) = U^u \cup \{v\}$ for some $v \in V_{i,j}\}$.

By construction, we know that $|V_u| = 30N \log n$ and $|V'_u| = 2 \cdot 3N \log n$.

We now consider $\mathcal{P}$. Observe that the vertices of $U^u$ are assigned to different components of $\mathcal{P}$. Thus, we have that:

- at most $4 \cdot 30N \log n = 120N \log n$ of the edges incident to vertices of $V_u$ are included the $E(\mathcal{P})$, and
- at most $5 \cdot 2N \cdot 3 \log n = 30N \log n$ of the edges incident to vertices of $V'_u$ are included in $E(\mathcal{P})$.

Also, since $|N(u) \cap V_C| \leq 4N \log n$ and $|N(u^i) \cap V_C| = 0$, for all $i \in [4]$, we have that $E(\mathcal{P})$ contains at most $4N \log n$ edges between $U^u$ and $V_C$. Therefore, by removing $C$ for all $C_i$, $i \in [p]$, we have reduced the value of $\mathcal{P}$ by at most $(120 + 30 + 4)N \log n = 154N \log n$. Let us now count the number of edges in $G[C]$. Since $U^u \cup V_u \subseteq C$, we have that $E(G[C])$ includes all the $150N \log n$ edges between vertices of $U^u$ and $V_u$. Also, we have that $E(G[C])$ contains $\frac{5}{6}$ of the edges incident to $V'_u$. Indeed, $N(V'_u) \cap C = U_u$. This gives another $5 \cdot 2N3 \log n = 30N \log n$ edges. Furthermore, no other edge appears in $G[C]$. Thus, $E(G[C])$ contains $(150 + 30)N \log n = 180N \log n$ edges. Therefore, we have that $v(\mathcal{P}') \geq 26N \log n + v(\mathcal{P})$. This is a contradiction to the optimality of $\mathcal{P}$, as $v(\mathcal{P}') > v(\mathcal{P})$. □

**Lemma 3.16.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. For any $k \in p$, there is no pair $(u, u')$ of vertices such that:*

- *$u \in U_{i,j}$ for some $(i,j) \in [3] \times [\log n]$,*
- *$u' \in U_{i',j'}$ for some $(i',j') \in [3] \times [\log n]$ (it is not necessary that $(i,j) \neq (i',j')$) and*
- *$\{u, u'\} \subseteq C_k$.*

*Proof.* Assume that this is not true and let $k \in [p]$ be an index for which such a pair $(u, u')$ exists in $C_k$. By the optimality of $\mathcal{P}$ and Lemma 3.15, we have that $U^u \cup U^{u'} \subseteq C_k$. By construction, we have that $|I_u| = |I_{u'}| = 30N \log n$. Since $u$ and $u'$ belong in the same $C_k$ and $\mathcal{C} = 42N \log n$, we know that there are at least $(2 \cdot 30 - 42)N \log n = 18N \log n$ vertices from the sets $I_u$ and $I_{u'}$ that do not belong in $C_k$. Notice that these vertices do not contribute at all to the value of $\mathcal{P}$ as they are not in the same partition as any of their neighbors. Consider the sets $C^1 = U^u \cup I_u$, $C^2 = U^{u'} \cup I_{u'}$ and $C = C^1 \cup C^2$.

We create the $\mathcal{C}$-partition $\mathcal{P}' = \{C^1, C^2, C_1 \setminus C, \ldots, C_p \setminus C\}$. Notice that $\mathcal{P}'$ is indeed a $\mathcal{C}$-partition as $|C^1| = |C^2| = 5 + 30N \log n \leq 42N \log n$ and $|C_i \setminus C| \leq |C_i| \leq 42N \log n$ as $C_i \in \mathcal{P}$ for all $i \in [p]$. We will show that $v(\mathcal{P}) < v(\mathcal{P}')$.

First, we will deal with the edges incident to vertices of $I_u$ and $I_{u'}$. Notice that $C^1$ and $C^2$ include all the edges between $U^u$ and $I_u$ as well as the edges between $U^{u'}$ and $I_{u'}$. Therefore, $|E(\mathcal{P}') \setminus E(\mathcal{P})| \geq 5(2 \cdot 30 - 42)N \log n = 90N \log n$. Indeed, each vertex of $I_u \cup I_{u'}$ has exactly five neighbors in the set $C^1 \cup C^2$ and at least $18N \log n$ edges do not contribute any value to $\mathcal{P}$. Now we consider the edges incident to vertices in $W = N(U^u \cup U^{u'}) \setminus (I_u \cup I_{u'})$. Observe that, in the worst case, all the edges between vertices of $W$ and $U^u \cup U^{u'}$ are included in $E(\mathcal{P})$ while none of them is included in $\mathcal{P}'$. Also, any edge that is not incident to $U^u \cup U^{u'}$ is either included in both $E(\mathcal{P})$ and $E(\mathcal{P}')$ or in none of them. Notice that any vertex in $U^u$ (respectively in $U^{u'}$) has $2N \cdot 3 \log n$ neighbors in $V(G_{i,j}) \setminus I_u$ (resp. in $V(G_{i',j'}) \setminus I_{u'}$). Furthermore, $u$ (resp. $u'$) has at most $4N \log n$ neighbors in $V_C$. Also, there are no other neighbors of these vertices to be considered. Therefore, in the worst case, $|E(\mathcal{P}) \setminus E(\mathcal{P}')| = 2N \cdot 3 \log n + 8N \log n = 68N \log n$. Since $90N \log n > 68N \log n$ we have that $v(\mathcal{P}') > v(\mathcal{P})$ which contradicts the optimality of $\mathcal{P}$. □

**Lemma 3.17.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. For any $(i,j) \in [3] \times [\log n]$ and $u \in U_{i,j}$, if $u \in C$ then any $v \in N(u) \cap V(G_{i,j})$ also belongs in $C$.*

*Proof.* Assume that for an $(i,j) \in [3] \times [\log n]$ there exists a $u \in U_{i,j}$ and $w \in (N(u) \cap V(G_{i,j}))$ such that $u \in C_k$ and $w \notin C_k$. We will show that $\mathcal{P}$ is not optimal.

It follows from Lemma 3.15 that $U^u \subseteq C_k$. We will consider two cases, either $|C_k| < \mathcal{C}$ or not.

**Case 1: $|C_k| < \mathcal{C}$.** In this case, either $w \in I_u$ or $w \in I_{v,u}$ for some $v \in V_{i,j}$. Since $w$ has at most one neighbor that does not belong in $C_k$, moving $w$ to the partition of $C_k$ will create a $\mathcal{C}$-partition that includes more edges than $\mathcal{P}$. This is a contradiction to the optimality of $\mathcal{P}$.

**Case 2: $|C_k| = \mathcal{C}$.** In this case, it is safe to assume that $G[C_k]$ is connected as otherwise we can partition it in to its connected components. This does not change the value of the partition, and the resulting set that contains $u$ has a size less than $\mathcal{C}$. We proceed by considering two sub-cases, either $C_k \cap V_C \neq \emptyset$ or not.

**Case 2.a: $C_k \cap V_C \neq \emptyset$.** We claim that, in this case, either there exists a vertex $c \in C_k \cap V_C$ such that $u \notin N(c)$ or $G[C_k]$ has a leaf $x$ such that $u \notin N(x)$. Indeed, in the second case, $|C_k \cap V_C|$ (by construction) and the existence of $w$ is guaranteed by the fuct that no other assignment vertex can be in $C_k$. In the first case we set $y = c$ while in the second $y = x$ We create a new partition as follows:

- we remove $y$ from $C_k$,
- move $w$ from its set to $C_k$ and
- add a new set $C = \{c\}$ in the partition.

Let $\mathcal{P}'$ be this new $\mathcal{C}$-partition. We have that $v(\mathcal{P}') > v(\mathcal{P})$. Indeed, $w$ has at most one neighbor that does not belong in $C_k$. Therefore, moving $w$ to $C_k$ increases the number of edges by at least 4 ($w$ is adjacent to all vertices of $U^u$ and $U^u \subseteq C_k$). We consider the case where $y$ is a vertex $c \in C_k \cap V_C$ such that $u \notin N(c)$. Since $u$ is the only assignment vertex in $C_k$, and there are at most 3 edges connecting $c$ to variable vertices, removing $c$ from $C_k$ reduces the value of $\mathcal{P}$ by at most 3. Therefore, $v(\mathcal{P}') - v(\mathcal{P}) = |E(\mathcal{P}')| - |E(\mathcal{P})| \geq 1$. This is a contradiction to the optimality of $\mathcal{P}$. Similarly, in the case where $y$ a leaf such that $u \notin N(y)$, removing $y$ from $C_k$ reduces the value of $\mathcal{P}$ by at most 1. This again contradicts the optimality of $\mathcal{P}$.

**Case 2.b: $C_k \cap V_C = \emptyset$.** Notice that, since $G[C_k]$ is connected, $|N(u) \cup V_{i,j}| < \mathcal{C}$ and $C_k \cap V_C = \emptyset$, there exists a pair $(v,x) \in V_{i,j} \times U_{i,j}$ such that $x \neq u$ and $C_k \cap I_{v,x} \neq \emptyset$. Also, by Lemmas 3.16 and 3.15, we have that $U^x \cap C_k = \emptyset$. Therefore, any vertex $y \in C_k \cap I_{v,x}$ contributes at most one edge in $E(\mathcal{P})$. We create a new partition as follows:

- select a vertex $y \in C_k \cap I_{v,x}$ and remove it from $C_k$,
- move $w$ for its set to $C_k$ and

- add a new set $C = \{y\}$ in the partition.

This is a contradiction to the optimality of $\mathcal{P}$, since the removal of $y$ from $C_k$ reduces the value of the partition by at most 1, while moving $w$ to $C_k$ increases the value by at least 4. □

Summing up the previous lemmas, we can observe that in any optimal $\mathcal{C}$-partition $\mathcal{P}$ of $G$, there is one component for each vertex $u \in \bigcup_{(i,j)\in[3]\times[\log n]} U_{i,j}$ and if $u \in C$, for some $C \in \mathcal{P}$, then $N(u) \setminus V_C \subseteq C$.

**Lemma 3.18.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. For any $(i,j) \in [3] \times [\log n]$ and $v \in V_{i,j}$, if $v \in C_k$, for some $k \in [p]$, then $|C_k \cap U_{i,j}| = 1$*

*Proof.* Recall that by Lemma 3.16, $|C_k \cap U_{i,j}|$ is either 1 or 0. Assume that there exist $(i,j) \in [3]\times[\log n]$ and $v \in V_{i,j}$ such that $v \in C_k$ and $|C_k \cap U_{i,j}| = 0$. By this assumption and Lemma 3.17, we can conclude that $N(v) \cap C_k \subseteq V_C$. Also, since each variable has at most 4 appearances and $v$ represents at mots $\frac{\log n}{2}$ variables, we have that $|N(v) \cap C_k| \leq 2 \log n$.

Let $u \in U_{i,j}$ be an arbitrary assignment vertex. Also, let $C_\ell \neq C_k$ be the set of $\mathcal{P}$ such that $u \in C_\ell$. By Lemma 3.17, we know that $C_\ell \cap I_{v,u} = I_{v,u}$. Now we consider two cases, either $|C_\ell| < \mathcal{C}$ or not.

**Case 1: $|C_\ell| < \mathcal{C}$.** We create a new partition as follows:

- remove $v$ for $C_k$ and
- add $v$ for $C_\ell$.

Let $\mathcal{P}'$ be the new partition; notice that this is a $\mathcal{C}$-partition as $|C_\ell \cup \{v\}| < \mathcal{C} + 1$. Also, the removal of $v$ from $C_k$ reduces the value of the partition by at most $2 \log n$ while the addition of $v$ to $C_k$ increases the value by $3 \log n$. This is a contradiction to the optimality of $\mathcal{P}$.

**Case 2: $|C_\ell| = \mathcal{C}$.** Similarly to the proof of Lemma 3.17, we assume that $G[C_\ell]$ is connected. Also, since any set $I_{v',x}$, for $(v', x) \in V_{i,j} \times U_{i,j}$ is a subset of the set of the partition that includes $x$, we have that $C_\ell \cap V_C \neq \emptyset$. Indeed, assuming otherwise we get that either $|C_\ell| < \mathcal{C}$ or $G[C_\ell]$ is not connected. We create a new partition as follows:

- select (arbitrarily) a vertex $c \in C_\ell \cap V_C$ and remove it from $C_\ell$,
- move $v$ from $C_k$ to $C_\ell$ and
- add a new set $C = \{c\}$ in the partition.

We will show that the value of the new partition is greater than the original. First, notice that $c$ has at most four neighbors in $C_\ell$, as $C_\ell$ can include only one assignment vertex, and $v$ has at most $2 \log n$ neighbors in $C_k$, as $N(v) \cap C_k \subseteq V_C$). Therefore, removing $c$ from $C_\ell$ and $v$ from $C_k$ reduces the value of the partition by at most $2 \log n + 4$. Also, since $u \in C_\ell$ and by Lemma 3.17, we get that $I_{v,u} \subseteq C_\ell$. Thus, moving $v$ into $C_\ell$ increases the value of the partition by $|I_{v,u}| = 3 \log n > 2 \log n + 4$. This is a contradiction to the optimality of $\mathcal{P}$. □

Now, we compute the range in which the value of any optimal $\mathcal{C}$-partition belongs in.

**Lemma 3.19.** *Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. We have that $3N \log^2 n(180\sqrt{n} + 6) \leq v(\mathcal{P}) \leq 3N \log^2 n(180\sqrt{n} + 6) + 2m$, where $m = |V_C|$. Furthermore, if a vertex $c \in V_C$ belongs to $C \in \mathcal{P}$ and $|N(c) \cap C| = 2$, then $N(c) \cap C = \{v, u\}$, where $v \in V_{i,j}$ and $u \in U_{i,j}$, for some $(i,j) \in [3] \times [\log n]$.*

*Proof.* First, we calculate the number of edges that $E(\mathcal{P})$ includes from any $G_{i,j}$. Notice that $W = V_{i,j} \cup U_{i,j}$ is a vertex cover of $G_{i,j}$ and no edge is incident to two vertices of this set. Therefore, we can compute $|E(\mathcal{P}) \cap E(G_{i,j})|$ by counting the edges of $\mathcal{P}$ that are incident to a vertex of $W$. First, for any vertex $u \in U_{i,j}$, if $u \in C$, for a $C \in \mathcal{P}$, we have that $N(u) \cap V_C \subseteq N(u) \cap C$. Also, we know that $U^u \subseteq C$. Therefore, all the edges that are incident to vertices in $U^u$, are in $E(\mathcal{P})$. So, for each $u \in U_{i,j}$ we have $5(30 + 2 \cdot 3)N \log n = 180N \log n$ edges in $E(\mathcal{P})$ that are incident to vertices in $U^u$. Also, it follows by Lemma 3.18 that for any vertex $v \in V_{i,j}$, there exists a (unique) $u \in U_{i,j}$ such that $\{v, u\} \subseteq C$ for some $C \in \mathcal{P}$. Furthermore, by Lemma 3.17, we have that $N(v) \cap C \subseteq I_{v,u}$. Thus, for

each $v \in V_{i,j}$, $E(\mathcal{P})$ includes $3 \log n$ edges and no other edge (from $E(G_{i,j})$) is incident to it. Since we have not counted any edge more than once, we have that $|E(\mathcal{P}) \cap E(G_{i,j})| = (180\sqrt{n}+6)N \log n$ for any $(i,j) \in [3] \times [\log n]$. Therefore, for any optimal $\mathcal{C}$-partition $\mathcal{P}$ of $G$, we have that $|E(\mathcal{P}) \cap \bigcup_{(i,j)\in[3]\times[\log n]} E(G_{i,j})| = 3N \log^2 n(180\sqrt{n}+6)$.

Since there are no edges between $V(G_{i,j})$ and $V(G_{i',j'})$ for $(i,j) \neq (i',j')$, it remains to count the edges incident to vertices of $V_C$. For any $(i,j) \in [3] \times [\log n]$ and any $c \in V_C$, we have that $|N(c) \cap V_{i,j}| \leq 1$ as the clause represented by $c$ has at most one variable from the vertex set $X_i$ and the vertices of any $V_{i,j}$ represent variables from $X_i$. Assume that $c \in C$, for $C \in \mathcal{P}$. If $C \cap U_{i,j} = \emptyset$ for all $(i,j) \in [3] \times [\log n]$, then $c$ has no neighbors in $C$. Indeed, by Lemma 3.18 we have that any variable vertex appears in the same set as one assignment vertex. Now, assume that $C$ includes a $u \in U_{i,j}$ for some $(i,j) \in [3] \times [\log n]$. By Lemma 3.16, there is no other assignment vertex in $C$. Also, by Lemma 3.18, only variable vertices from $V_{i,j}$ can be in $C$. Therefore, $c$ has at most 2 neighbors in $C$ (one variable vertex and one assignment vertex). Since the sets of edges are disjoint, we have at most 2 extra edges per clause vertex $c \in V_C$. This concludes the proof of this lemma. □

**The reduction.** We are now ready to show that the starting formula $\phi$ is satisfiable if and only if $G$ has a $\mathcal{C}$-partition of value $3N \log^2 n(180\sqrt{n}+6) + 2m$, where recall that $\mathcal{C} = 42N \log n$ and $N = \lceil n/\log^2 n \rceil$.

Assume that $\phi$ is satisfiable and let $\alpha : X \to \{true, false\}$ be a satisfying assignment. We will construct a $\mathcal{C}$-partition of $G$ of the wanted value.

First, for each assignment vertex $u$, create a set $C_u = U^u \cup (N(u) \setminus V_C)$. We then extend these sets as follows. Consider a variable vertex $v$ and restrict the assignment $\alpha$ on the vertex set $X(v)$. By construction, there exists an assignment vertex $u$ that represents this restriction of $\alpha$. Notice that there may exist more than one such vertex; in this case we select one of them arbitrarily. We add $v$ into the set $C_u$ that corresponds to $u$. We repeat the process for all variable vertices. Next, we consider the vertices in $V_C$. Let $c \in V_C$ be a vertex that represents a clause in $\phi$. Since $\alpha$ is a satisfying assignment, there exists a literal in this clause that is set to true by $\alpha$. Let $x$ be the variable of this literal. We find the set $C_u$ such that $v \in C_u$ and $x \in X(v)$. We add $c$ in $C_u$, and we repeat this for the rest of the vertices in $V_C$.

We claim that the partition $\mathcal{P} = \{C_u \mid u \text{ is an assignment vertex}\}$ is an optimal $\mathcal{C}$-partition of $G$. We first show that this is indeed a $\mathcal{C}$-partition. By construction, for any $C \in \mathcal{P}$ we have a pair $(i,j) \in [3] \times [\log n]$ and a vertex $u \in U_{i,j}$ such that $C \subseteq V_{i,j} \cup N[U^u] \cup V_C$. Notice that $|V_{i,j} \cup N[U^u]| = 2N + 2N \cdot 3 \log n + 30N \log n + 5$. We now calculate $|C \cap V_C|$. By construction, if $c \in C \cap V_C$, there exists a vertex $v \in V_{i,j}$ such that $v \in C$. Therefore, $N(V_{i,j}) \cap V_C \supseteq C \cap V_C$. Since each $v \in V_{i,j}$ represents $\frac{\log n}{2}$ variables and each variable appears in at most 4 clauses, we have that $|N(V_{i,j}) \cap V_C| \leq |V_{i,j}| 2 \log n \leq 4N \log n$. Thus $|C| \leq 2N + 2N3 \log n + 30N \log n + 4N \log n + 5 < 42N \log n = \mathcal{C}$ for sufficiently large $n$.

We now need to argue about the optimality of $\mathcal{P}$. Using the same arguments as in Lemma 3.19, we can show that $E(\mathcal{P}) \cap E(G_{i,j})$ includes exactly $3N \log n(180\sqrt{n}+6)$ edges. Thus, $|E(\mathcal{P}) \cap \bigcup_{(i,j)\in[3]\times[\log n]} E(G_{i,j})| = 3N \log^2 n(180\sqrt{n}+6)$. Therefore, we need to show that there are $2m$ additional edges in $E(\mathcal{P})$ that are incident to vertices of $V_C$. Notice that, for any $c \in V_C$, there exists a $C_u$ such that $c \in C_u$ and there exist vertices $v, u$ in $C_u$ that are both incident to $c$ (which holds by the selection of $C_u$), with $v$ being a variable vertex and $u$ an assignment vertex. Finally, by construction, there are at most 2 edges incident to $c$ in $E(\mathcal{P})$. Therefore, the $v(\mathcal{P}) = 3N \log^2 n(180\sqrt{n}+6) + 2m$.

For the reverse direction, assume that we have a $\mathcal{C}$-partition $\mathcal{P}$ of $G$, with $v(\mathcal{P}) = 3N \log^2 n[180\sqrt{n}+6] + 2m$. By Lemma 3.19 we have that each vertex $c \in V_C$ must be in a set $C \in \mathcal{P}$ such that:

- $|N(c) \cap C| = 2$ and
- there exist $(i,j) \in [3] \times [\log n]$ such that $v \in V_{i,j} \cap C$, $u \in U_{i,j} \cap C$ and $\{v,u\} \subseteq N(c)$.

We construct an assignment $\alpha$ of $\phi$ that corresponds to this partition as follows. For each variable $x$, consider the variable vertex $v$ such that $x \in X(v)$. By Lemma 3.18 there exists a unique assignment vertex $u$ such that $v$ and $u$ belong in the same component of $\mathcal{P}$. Let $\sigma_{v,u}$ be the assignment represented by $u$ for $X(v)$. We set $\alpha(x) = \sigma_{v,u}(x)$. Notice that each variable appears in the set of one variable

vertex and for each such vertex we have selected a unique assignment (represented by the assignments vertex in its set). Therefore the assignment we create in this way it is indeed unique.

We claim that $\alpha$ is a satisfying assignment. Consider a clause of $\phi$ and assume that $c$ is the corresponding clause vertex in $V_C$. Assume that $c \in C$ for some $C \in \mathcal{P}$. By Lemma 3.19 we have that $|N(c) \cap C| = 2$ and there exist $(i,j) \in [3] \times [\log n]$ such that $v \in V_{i,j} \cap C$, $u \in U_{i,j} \cap C$ and $\{v,u\} \subseteq N(c)$. Since $v \in N(c)$, we know that there exists a variable $x \in X(v)$ that appears in a literal $l$ of the clause represented by $c$. Observe that $v$ is unique. Moreover, since $u, v \in V(G_{i,j})$, and $u \in N(c)$, we have that $\sigma_{v,u}(l) = \alpha(l)$ satisfies the clause represented by $c$. This finishes the reduction.

In order for the claimed lower bound to hold, we need to bound $\mathrm{vc}(G) = \mathrm{vc}$, *i.e.*, the size of the vertex cover number of $G$, appropriately. Notice that the vertex set containing the $V_{i,j}$s, the $U_{i,j}$s and the copies of the vertices in the $U_{i,j}$s, for every $(i,j) \in [3] \times [\log n]\}$, is a vertex cover of the graph. Therefore, $\mathrm{vc} \leq 3 \log n(2N + 5\sqrt{n}) \in \mathcal{O}(\frac{n}{\log n})$. Additionally, $\mathcal{C} \in \mathcal{O}(\frac{n}{\log n})$.

To sum up, if we had an algorithm that computed an optimal solution of the $\mathcal{C}$-CF problem in time $(\mathcal{C}\mathrm{vc})^{o(\mathcal{C}+\mathrm{vc})}$, we would also solve the R3-SAT problem in time $\left(\frac{n}{\log n}\right)^{o(\frac{n}{\log n})}$. This contradicts the ETH since $\left(\frac{n}{\log n}\right)^{o(\frac{n}{\log n})} = 2^{(\log n - \log\log n)o(\frac{n}{\log n})} = 2^{o(n - \frac{n \log\log n}{\log n})} = 2^{o(n)}$. □

# 4 Kernelization

In this section, we establish that $\mathcal{C}$-CF admits a polynomial kernel parameterized by $\mathrm{vc} + \mathcal{C}$. We will use an auxiliary bipartite graph $H$ that we construct as follows. Let $U$ be a vertex cover in $G$ and let $I = V(G) \setminus U$. Then, $V(H)$ contains two partitions $X$ and $Y$ such that $V(Y) = I$ and for each $u \in U$, we add $t = \mathrm{vc} \times \mathcal{C} + \mathcal{C}$ many vertices $u_1, \ldots, u_t$. Moreover, if $uv \in E(G)$ such that $u \in U$ and $v \in I$, we add the edge $u_i v$ in $H$ for each $i \in [t]$. Now, we compute a maximum matching $\mathcal{M}$ in $H$. Let $Y' \subseteq Y$ be the set of vertices that are not matched in $\mathcal{M}$. We have the following reduction rule (RR).

**(RR):** Delete an arbitrary vertex $w \in Y'$ from $I$.

**Lemma 4.1.** *RR is safe.*

*Proof.* First, observe that in any $\mathcal{C}$-partition $\mathcal{P}$ of $G$, at most $\mathrm{vc} \times \mathcal{C}$ many vertices can participate in sets $C \in \mathcal{P}$ such that $C \cap U \neq \emptyset$ and these are the only vertices of $I$ that can contribute in the value of $\mathcal{P}$.

Now, let $G' = G[V(G) \setminus \{w\}]$. Since any $\mathcal{C}$-partition, $\mathcal{P}'$ of $G'$ can be easily extended to a $C$-partition of $G$ by adding to it a singleton set $C = \{w\}$, it suffices to show that the value of the optimal partition of $G$ and value of the optimal partition of $G'$ are equal.

Let $\mathcal{P} = \{C_1, \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. We claim that there exists a $\mathcal{C}$-partition $\mathcal{P}^*$ such that $\{w\} \in \mathcal{P}^*$ and $v(\mathcal{P}^*) = v(\mathcal{P})$. Notice that, by proving that $\mathcal{P}^*$ exists, we also prove that any optimal $\mathcal{C}$-partition of $G'$ has the same value as any optimal $\mathcal{C}$-partition of $G$. Indeed, $\mathcal{P}^* \setminus \{w\}$ is a $\mathcal{C}$-partition of $G'$ and $v(\mathcal{P}^*) = v(\mathcal{P}^* \setminus \{w\}) = v(\mathcal{P})$; thus $\mathcal{P}^* \setminus \{w\}$ is a $\mathcal{C}$-partition of $G'$ (otherwise, $\mathcal{P}$ is not an optimal $\mathcal{C}$-partition of $G$). It remains to prove that such a $\mathcal{C}$-partition exists.

In the case that $\{w\}$ is a singleton in $\mathcal{P}$ then $\mathcal{P}^* = \mathcal{P}$. Therefore, we assume that $w$ participates in some set $C \neq \{w\}$ of $\mathcal{P}$. Let $x \in U \cap C$ such that $xw \in E(G)$. Then, observe that $x_1, \ldots, x_t$ are matched to $t = \mathrm{vc} \times \mathcal{C} + \mathcal{C}$ many vertices of $I$ by the maximum matching in $H$ (as $w \notin Y$); let $S_x$ be the set of these vertices.

Observe that at least $\mathcal{C}$ of the vertices in $S_x$ are not contributing in the value of $\mathcal{P}$ (since at most $\mathrm{vc} \times \mathcal{C}$ many vertices can participate in sets $C \in \mathcal{P}$ such that $C \cap U \neq \emptyset$). We create a new set $C'$ by moving $\mathcal{C} - 1$ of these vertices (which contain vertices that are connected to $x$) into $C'$ and move $x$ from $C$ to $C'$. Observe that after this step, we have that $v(\mathcal{P}') \geq v(\mathcal{P})$ as we remove at most $\mathcal{C} - 1$ edges incident on $x$ in $C$ and add exactly $\mathcal{C} - 1$ edges incident on $x$ in $C'$. We can keep repeating this step until $w$ is no longer connected to any vertex in $C$, and at this point, we can delete $w$ from $C$ (since its contribution is 0) and add it as a singleton. Hence, we get the $\mathcal{P}^*$. This completes our proof. □

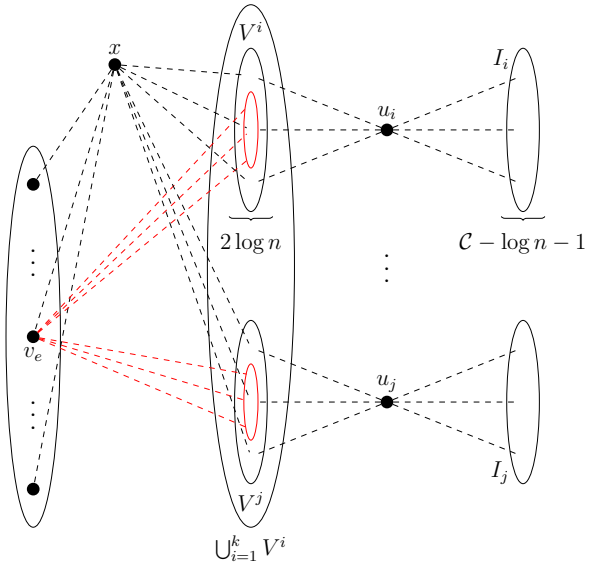



Figure 4: The graph $G$ constructed in the proof of Theorem 4.3

We argue about the size of our kernel in the next lemma.

**Lemma 4.2.** *Once we cannot apply RR anymore,* $|V(G)| = \mathcal{O}(\mathrm{vc}^2\mathcal{C})$.

Hence, we have our poly kernel parameterized by $\mathrm{vc} + \mathcal{C}$. We want to mention that this kernel can be further improved to $\mathcal{O}(\mathrm{vc}\mathcal{C})$ vertices by matching each vertex of $U$ to at most $\mathcal{C}$ vertices in $I$, but the analysis is highly non-trivial. Hence, in this preliminary version, we provide the above, simpler, kernel.

**Theorem 4.3.** *Given an edge-weighted graph* $G$ *of vertex cover number* vc*, it is highly unlikely to find a poly*$\{\mathrm{vc} + \mathcal{C}\}$ *size kernel that computes an optimal* $\mathcal{C}$*-partition of* $G$.

*Proof.* The reduction is from the $k$-Multicolored Clique problem, where given a graph $H = (V, E)$ and a partition $(V_1, \ldots, V_k)$ of $V$ into $k$ independent sets, the question is whether there exists a set $S \subseteq V$ such that $G'[S]$ is a clique. We may additionally assume that $|V_1| = \ldots = |X_k| = n = 2^m$ for some $m \in \mathbb{N}$ as, otherwise, we can add independent vertices in each set. It is known that $k$-MC does not admit a kernel of size $poly(k + \log n)$, unless the Polynomial Hierarchy collapses [38].

**The construction.** We construct an instance of $\mathcal{C}$-Coalition Formation, for $\mathcal{C} = \binom{k}{2} + k\log n + 1$, as follows (illustrated in Figure 4). For each set $V_i$, we first create a clique of $2\log n$ vertices $V^i = \{u^i_j, v^i_j \mid j \in [\log n]\}$. We proceed by creating a vertex $u_i$ and an independent set $I_i$ of size $\mathcal{C} - \log n - 1$. Finally, we add all edges between $u_i$ and vertices from $V^i \cup I_i$.

Before we continue, we will relate the vertices for each set $V_i$ with a subset of vertices of $V^i$. Let $v_i, \ldots, v_n$ be an enumeration of the vertices in $V_i$. We assign to each $v_j \in V_i$ a binary string of length $\log n$ such that the string assigned to $v_j$ represents the number $j$ in binary form. Let $s(v)$ be the string assigned to a vertex $v$ of the original graph. Also, for each $i$ and $v \in V_i$, we use $s(v)$ in order to define a set $S(v) \subseteq V^i$ as follows; for each $\ell \in [\log n]$,

- if the $\ell$-th letter of $s(v)$ is 0, we add $u^i_\ell$ in $S(v)$,

- otherwise, we add $v_\ell^i$ in $S(v)$.

We continue by creating one vertex $v_e$ for each edge $e \in E$. We call this set $V_e$. We add edges incident to $v_e$ as follows: Let $e = uv$ where $u \in V_i$ and $v \in V_j$ for some $i, j \in [k]$; notice that $i \neq j$. We add all edges $v_e w$, where $w \in S(u) \cup S(v)$.

Finally, we add a vertex $x$ add we add edges between $x$ and any vertex in $V_e \cup \bigcup_{i\in[k]} V^i$. We will call this new graph $G$.

We complete the construction by defining the weight function $w : E(G) \to \mathbb{N}$ as follows.

- For any edge $e = u_i v$ where $v \in I_i$, we set $w(e) = 4\mathcal{C}^3$.
- For any edge $e = u_i v$ where $v \in V^i$, we set $w(e) = 3\mathcal{C}^2$.
- For any edge $e = xv$ where $v \in V^i$, we set $w(e) = 2\mathcal{C}$.
- For any other edge $e$ we set $w(e) = 1$.

We will show that finding a clique of order $k$ in $H$ is equivalent of finding a $\mathcal{C}$-partition $\mathcal{P}$ of $G$ of value $v(\mathcal{P}) = k\big(4\mathcal{C}^3(\mathcal{C} - \log n - 1) + (3\mathcal{C}^2 + 2\mathcal{C})\log n + 2\binom{\log n}{2}\big) + \binom{k}{2}(2\log n + 1)$.

**Properties of optimal $\mathcal{C}$-partitions of $G$.** Before we proceed with the reduction, we will prove some properties for any optimal partition of $G$.

**Lemma 4.4.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. For any $i \in [k]$, there exists a $j \in [p]$ such that $u_i \cup I_i \subseteq C_j$.*

*Proof.* Assume that there exists an $i \in [k]$ such that $C_j \cap (u_i \cup I_i) \neq (u_i \cup I_i)$, for any $j \in [p]$. Let $C$ be the set $\{u_i\} \cup I_i$. We claim that the partition $\mathcal{P}' = \{C_1 \setminus C, \ldots C_p \setminus C, C\}$ has higher value than $\mathcal{P}$. Indeed, by separating $C$ from the rest of the partition, we only lose the weights of the edges incident to $u_i$ and vertices of $V^i$ (as all the other neighbors of $u_i$ are in $C$). Since all these edges have weight $3\mathcal{C}^2$, we are reducing the value of the partition by at most $\mathcal{C}3\mathcal{C}^2$. On the other hand, notice that there exists at least one edge $e = u_i v$ for some $v \in I_i$ such that $w(e)$ is counted in $\mathcal{P}'$ but not in $\mathcal{P}$. Since $4\mathcal{C}^3 > \mathcal{C}3\mathcal{C}^2$, we have that $v(\mathcal{P}') > v(\mathcal{P})$, which contradicts the optimality of $\mathcal{P}$. □

**Lemma 4.5.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$. If $u_i \in C_j$, for some $(i, j) \in [k] \times [p]$, then $|C_j \cap V^i| = \log n$.*

*Proof.* Assume that there exists a $u_i \in C_j$, for some $(i, j) \in [k] \times [p]$ such that $|C_j \cap V^i| < \log n$. Notice that $|C_j \cap V^i|$ is at most $\log n$ by Lemma 4.4. Select (arbitrarily) a set $U$ such that $V^i \supset U \supset C_j \cap V^i$ and $|U| = \log n$. We set $C = U \cup I_i \cup \{u_i\}$. Them, we create the partition $\mathcal{P}' = \{C_1 \setminus C, \ldots, C_p \setminus C, C\}$. We claim that $v(\mathcal{P}') > v(\mathcal{P})$. Indeed, there is at least one edge $u_i v \in E(\mathcal{P}') \setminus E(\mathcal{P})$. Also, this edge has weight $3\mathcal{C}^2$. Now, consider an edge in $e \in E(\mathcal{P}) \setminus E(\mathcal{P}')$. It is not hard to see that $w(e) = 1$ or $w(e) = 2\mathcal{C}$. Also, since any edge with weight $2\mathcal{C}$ is incident to $x$, we may have less than $\mathcal{C} - 1$ such edges in $E(\mathcal{P}) \setminus E(\mathcal{P}')$. Thus, the total weight of the edges in $E(\mathcal{P}) \setminus E(\mathcal{P}')$ is less than $\mathcal{C}2\mathcal{C} + \binom{\mathcal{C}}{2} < 3\mathcal{C}^2$. Therefore, $v(\mathcal{P}') > v(\mathcal{P})$, which contradicts the optimality of $\mathcal{P}$. □

**Lemma 4.6.** *Let $\mathcal{P} = \{C_1 \ldots, C_p\}$ be an optimal $\mathcal{C}$-partition of $G$ and $x \in C_\ell$, for an $\ell \in [p]$. Then $|C_\ell \cap V^i| = \log n$ for all $i \in [k]$.*

*Proof.* It follows from Lemmas 4.4 and 4.5 that, for each $i \in [k]$, we have exactly $\log n$ vertices from $V_i$ that are in the same set as $\{u_i\} \cup I_i$. Let $C^i \in \mathcal{P}$ be the set that includes $\{u_i\}$. Observe that $|C^i| = \mathcal{C}$. Thus, no other vertex has been included to $C_i$. Therefore, for all $i \in [k]$, there are exactly $\log n$ vertices that are not in the same set as $u_i$; let $S_i$ be this subset of $V^i$. That is, $S_i = V^i \setminus C^i$. We will show that, for all $i \in [k]$, we have that $S_i \subseteq C_\ell$. Assume that there exists an $i$ such that $S_i \nsubseteq C_\ell$ and let $u \in S_i \setminus C_\ell$. We consider two cases, either $|C_\ell| < \mathcal{C}$ or not.

**Case 1 $|C_\ell| < \mathcal{C}$:** Then we create the following partition.

- Remove $u$ from its current set and

- add $u$ to $C_\ell$.

Since $u$ was not in the same set as $u_i$ or $x$ in $\mathcal{P}$, any edge $e \in E(\mathcal{P})$ that is incident to $u$ has weight 1. Also, since $\mathcal{P}$ is a $\mathcal{C}$-partition, we have at most $\mathcal{C}-1$ neighbors of $u$ in the same set as $u$ in $\mathcal{P}$. Thus, moving $u$ to a different set reduces the value of the partition by at most $\mathcal{C}-1$. On the other hand, in $E(\mathcal{P}')$ we have at least included the edge $xu$ and $w(xu) = 2\mathcal{C} > \mathcal{C}$. This contradicts the optimality of $\mathcal{P}$.

**Case 2 $|C_\ell| = \mathcal{C}$:** Then we have at least one edge vertex $v_e$ in $\mathcal{C}_\ell$. We create a new $\mathcal{C}$-partition by swapping $u$ and $v_e$. Again, moving $u$ to a different set, reduces the value of the partition by at most $\mathcal{C}-1$. Also, recall that by construction, $d(v_e) = 2\log n + 1$ and all of theses edges are of weight 1. Therefore, moving $v_e$ to a different set reduces the value of the partition by at most $2\log n + 1$. The fact that $E(\mathcal{P}')$ includes at least the edge $xu$ and $w(xu) = 2\mathcal{C} > \mathcal{C} + 2\log n + 1$ leads to a contradiction to the optimality of $\mathcal{P}$. □

**The reduction.** We are now ready to prove the theorem. In particular, we will show that $H$ has a clique of order $k$ if and only if any optimal $\mathcal{C}$-partition $\mathcal{P}$ for $G$ has value at least $v(\mathcal{P}) = k\big(4\mathcal{C}^3(\mathcal{C}-\log n-1) + (3\mathcal{C}^2+2\mathcal{C})\log n + 2\binom{\log n}{2}\big) + \binom{k}{2}(2\log n+1)$.

Assume that $H$ has a clique of order $k$ and let $v^i$ be the vertex of this clique that also belongs to $V_i$, for each $i \in [k]$. For each $i \in [k]$, we create the set $C_i = \{u_i\} \cup I_i \cup (V_i \setminus S(v^i))$. Then we create a set $C = \{x\} \cup \bigcup_{i\in[k]} S(v^i) \cup \{v_e \mid e = v^iv^j \text{ for all } 1 \le i < j \le k\}$. Finally we add one set for each remaining vertex $v_e$. Let $\mathcal{P} = \{C_1, \ldots, C_p\}$, $p > k+1$, be the resulting partition. We claim that $v(\mathcal{P}) = k\big(4\mathcal{C}^3(\mathcal{C}-\log n-1) + (3\mathcal{C}^2+2\mathcal{C})\log n + 2\binom{\log n}{2}\big) + \binom{k}{2}(2\log n+1)$.

Indeed, we have that, for any $i \in [k]$, the sum of the weights of the edges of $G[C_i]$ is exactly $4\mathcal{C}^3(\mathcal{C}-\log n-1) + 3\mathcal{C}^2\log n + \binom{\log n}{2}$. Also, by construction, the sum of the weights of the edges of $G[C]$ is exactly $k2\mathcal{C}\log n + k\binom{\log n}{2} + \binom{k}{2}(2\log n+1)$. Finally, all the other sets are singletons. Thus $v(\mathcal{P}) = k\big(4\mathcal{C}^3(\mathcal{C}-\log n-1) + (3\mathcal{C}^2+2\mathcal{C})\log n + 2\binom{\log n}{2}\big) + \binom{k}{2}(2\log n+1)$.

For the reverse direction, assume that we have a partition $\mathcal{P}$ that has value $v(\mathcal{P}) = k\big(4\mathcal{C}^3(\mathcal{C}-\log n-1) + (3\mathcal{C}^2+2\mathcal{C})\log n + 2\binom{\log n}{2}\big) + \binom{k}{2}(2\log n+1)$. By Lemmas 4.4 and 4.5, we know that, for each $i \in [k]$, there exists a set $C \in \mathcal{P}$ such that $C \supseteq \{u_i\} \cup I_i$ and $C \setminus (\{u\} \cup I_i) \subseteq V^i$. Let us reorder the sets of $\mathcal{P}$ such that $\mathcal{P} = \{C_1, \ldots, C_p\}$ and $u_i \in C_i$, for all $i \in [k]$. First, we calculate the maximum value of $\{C_1, \ldots, C_k\}$. Notice that for any $i$, $C_i$ includes exactly $\log n$ vertices from $V^i$ and the set $\{u_i\} \cup I_i$. Therefore, we need to take into account:

- $\binom{\log n}{2}$ edges of weight 1, between the vertices of $V^i$,
- $\log n$ edges of weight $3\mathcal{C}^2$, between the vertices of $V^i$ and $u_i$ and
- $\mathcal{C} - \log n$ edges of weight $4\mathcal{C}^3$ between the vertices of $I_i$ and $u_i$.

In total, this gives us a value of $\binom{\log n}{2} + \log n 3\mathcal{C}^2 + (\mathcal{C}-\log n)4\mathcal{C}^3$, and this holds for all $i \in [k]$.

By Lemma 4.6, we also know that there exists a set $C$ in $\mathcal{P}$ that includes the vertex $x$ together with the remaining vertices from the sets $V^i$, $i \in [k]$. Notice that $C$ may also include up to $\binom{k}{2}$ vertices from $V_e$. Actually, $C$ must include all these vertices, as otherwise the edges incident to them will contribute nothing to the value of $\mathcal{P}$.

We will calculate the value of the edges in $E(C \setminus V_e)$. Notice that these edges are either between two vertices in the same set $V^i$ or between a set $V^i$ and $x$. Since for each $i \in [k]$ we have $\log n$ vertices from $V^i$, we have:

- $\binom{\log n}{2}$ edges of weight 1, between the vertices of $V^i$, for each $i \in [k]$ and
- $\log n$ edges of weight $2\mathcal{C}$, between the vertices of $V^i$ and $x$, for each $i \in [k]$.

Therefore, by adding these with the value from the sets $C_i$, $i \in [k]$, we have calculate a value of $k\big(4\mathcal{C}^3(\mathcal{C}-\log n) + (3\mathcal{C}^2+2\mathcal{C})\log n + 2\binom{\log n}{2}\big)$.

Observe that the assumed value of $\mathcal{P}$ higher than the one that we have calculate for the moment, by $\binom{k}{2}(2\log n+1)$. Notice also that this extra value can be added only by the vertices from $V_e$ that

can be in the same set as $x$. Finally, any vertex $v \in V_e \cap C$ can contribute at most $2\log n + 1$ since $d(v) = 2\log n + 1$ and all these edges have weight 1. Therefore, in order to achieve the wanted value, we have that $|C \cap V_e| = \binom{k}{2}$ and for each vertex $v \in C \cap V_e$, $N(v) \subseteq C$.

Next, we will show that there is no pair $(i, j)$ for which there exist two edges $e, e'$ such that $\{v_e, v_{e'}\} \subseteq C$, $e = uv$, where $u \in V_i$ and $v \in V_j$, $e' = u'v'$, where $u' \in V_i$ and $v' \in V_j$. Notice that, $N(v_e) \subseteq C$ and $N(v_e) = S(u) \cup S(v) \cup x$. Therefore, $C \cap V^i = S(u)$ and $C \cap V^j = S(v)$. Since the same holds for $v_{e'}$, we can conclude that $S(u) = S(u')$ and $S(v) = S(v')$. Thus, $e = e'$. This cannot happen because these vertices represent edges of $H$ and there are no parallel edges in $H$. We can conclude that no two of vertices $v_e$ and $v'_e$ in $C$ can represent edges between vertices of the same sets. Also, since we have $\binom{k}{2}$ such vertices, for each pair $(i, j)$ we have a vertex $v_e$ that represents an edge $uv$ where $u \in V_i$ and $v \in V_j$.

Now, consider the set of vertices $U = \{v \in V(H) \mid S(v) = C \cap V^i \text{ for some } i \in [k]\}$. We claim that $U$ is a clique of order $k$ in $H$. We will first show that for each $i \in [k]$, we have that $C \cap V^i = S(v)$ for some $v \in V_i$. As we mentioned, for each pair $(i, j)$ there exists one $e = uv$, where $u \in V_i$, $v \in V_j$ and $v_e \in C$. Also, $N(v_e) \subseteq C$ and $N(v_e) = S(u) \cup S(v) \cup x$. Therefore, $C \cap V^i = S(u)$. Since this holds for any $i \in [k]$, we have that $U$ indeed represents a set of $k$ vertices in $H$. We need to show that $U$ induces a clique. Consider two vertices $u, v \in U$ and let $u \in V_i$ and $V_j$. Recall that for each pair $(i, j)$, we have a vertex $v_e \in C$ such that $e = u'v'$, $u \in V_i$ and $v \in V_j$. Also, we have shown that $S(u') = C \cap V^i$ and $S(v') = C \setminus V^j$. Therefore, $S(u') = S(u)$ and $S(v') = S(v)$, from which follows that $e = uv$. Thus, there exists an edge between the two vertices. Since we have selected $u$ and $v$ arbitrarily, we have that $U$ is indeed a clique.

To fully prove the statement, it remains to be shown that the parameter that we are considering is bounded by a polynomial of $k + \log n$. Notice that the set $U = \{x\} \cup \bigcup_{i \in [k]}(V^i \cup \{u_i\}$ is a vertex cover of $G$. Also, $|V^i| = 2\log n$ for all $i \in [k]$. Therefore, we have that $|U| \in O(k \log n)$. Recall that we have set $\mathcal{C} = 1 + \binom{k}{2} + k \log n$. Therefore, $\text{vc} + \mathcal{C} \in \mathit{poly}(k + \log n)$. □

# 5 Additional Structural Parameters

**Theorem 5.1.** *Given an unweighted graph $G = (V, E)$ with vertex integrity $k$, there exists an FPT algorithm that computes an optimal $\mathcal{C}$-partition of $G$, parameterized by $k$.*

*Proof.* Let $U \subseteq V$ be such that $|U| = k' \leq k$ and $S_1, \ldots, S_m$ be the vertex sets of the connected components of $G[V \setminus U]$. It follows that $|S_j| \leq k$, $j \in [m]$. Let $\mathcal{P}' = \{C'_1, \ldots, C'_p\}$ be the strict restriction [1] of an optimal $\mathcal{C}$-partition $\mathcal{P}$ of $G$ on the set $U$ (there are at most $|U|^{|U|} \leq k^k$ possible restrictions of $\mathcal{P}$ on $U$). We will extend $\mathcal{P}'$ into an optimal $\mathcal{C}$-partition of $G$. To do so, we will organize the connected components of $G[V \setminus U]$ into a bounded number of different types, and run an ILP.

We begin by defining the types.Two graphs $G_i = G[U \cup S_i]$ and $G_j = G[U \cup S_j]$, $i, j \in [m]$, are of the same *type* if there exists a bijection[2] $f : U \cup S_i \to U \cup S_j$ such that $f(u) = u$ for all $u \in U$ and $N_{G_i}(u) = \{f^{-1}(v) \mid v \in N_{G_j}(f(u))\}$ for all $u \in S_i$. Note that if such a function exists, then $G_i$ is isomorphic to $G_j$.

Let $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$ be the different types that were defined. Observe that $\ell$ is at most a function of $k$ since $|U| \leq k$. For each $i \in [\ell]$, we define the *representative of* $\mathcal{T}_i$ to be any connected component of $G[V \setminus U]$ that is contained in a graph of type $\mathcal{T}_i$; we will denote this graph by $G_{\mathcal{T}_i}$. For each $i \in [\ell]$, we will store a set of vectors $\tau^i_j$, for $j \in [q]$, which contain all possible orderings of all possible partitions of $V(G_{\mathcal{T}_i})$ into $p + k$ sets (some of which may be empty). If $G_{\mathcal{T}_i}$ *follows* the vector $\tau^i_j = (\alpha_1, \ldots, \alpha_{p+1}, \ldots, \alpha_{p+k})$, then $\alpha_1, \ldots, \alpha_{p+k}$ is a partition of $V(G_{\mathcal{T}_i})$, and $\mathcal{P}^i_j = \{C'_1 \cup \alpha_1, \ldots, C'_p \cup \alpha_p, \alpha_{p+1}, \ldots, \alpha_{p+k}\}$ is a possible extension of $\mathcal{P}'$ including the vertices that belong in any component of type $i$, according to the vector $\tau^i_j$.

[1] a restriction is *strict* if it only contains non-empty sets.

[2] Recall that a function $f : A \to B$ is a *bijection* if, for every $a_1, a_2 \in A$ with $a_1 \neq a_2$, we have that $f(a_1) \neq f(a_2)$ and for every $b \in B$, there exists an $a \in A$ such that $f(a) = b$. Recall also that the *inverse* function of $f$, denoted as $f^{-1}$, exists if and only if $f$ is a bijection, and is such that $f^{-1} : B \to A$ and for each $b \in B$ we have that $f^{-1}(b) = a$, where $f(a) = b$.

For every $i \in [\ell]$ and $j \in [q]$, let $E^i_j = \{E(\mathcal{P}^i_j) \setminus E(\mathcal{P}')\}$ be the *important edges according to* $\tau^i_j$. be the edges of the subgraph of $G$ induced by $\mathcal{P}^i_j$. All that remains to be done is to search through these vectors and find the optimal ones among those that result in $\mathcal{C}$-partitions. This is achieved through the following ILP.

---

Variables

| | | |
|---|---|---|
| $x_i$ | $i \in [\ell]$ | number of components of type $i$ |
| $y_{i,j}$ | $i \in [\ell], j \in [q]$ | number of important edges according to $\tau^i_j$ |
| $v_{i,j,l}$ | $i \in [\ell], j \in [q], l \in [p]$ | number of vertices in the $l^{th}$ position of vector $\tau^j_i$ |
| $z_{i,j}$ | $i \in [\ell], j \in [q]$ | number of components of type $i$ following the vector $\tau^i_j$ |

Constants

| | | |
|---|---|---|
| $w_l$ | $l \in [p]$ | number of vertices in $\mathcal{C}'_l$ |

Objective

$$\max \sum_{i=1}^{\ell} \sum_{j=1}^{q} y_{i,j} z_{i,j} \tag{5.1}$$

Constraints

$$\sum_{j=1}^{q} z_{i,j} = x_i \qquad \forall i \in [\ell] \tag{5.2}$$

$$\sum_{i=1}^{\ell} \sum_{j=1}^{q} v_{i,j,z} z_{i,j} + w_l \leq \mathcal{C} \qquad \forall l \in [p] \tag{5.3}$$

---

In the above model, the constraint 5.2 is used to make sure that every component of type $i$ follows exactly one vector $\tau^i_j$. Then, the constraint 5.3 is used to make sure that the resulting partition is indeed a $\mathcal{C}$-partition. Finally, since the number of variables of the model is bounded by a function of $k$, we can and obtain a solution in FPT time, parameterized by $k$ (by running for example the Lenstra algorithm [45]). □

**Theorem 5.2.** *Let $G$ be an unweighted graph and $\mathcal{C}$ and $v^*$ be two integers. Deciding if there exists a $\mathcal{C}$-partition $\mathcal{P}$ of $G$ with $v(\mathcal{P}) \geq v^*$ is W[1]-hard when parameterized by the twin-cover number of $G$.*

*Proof.* The reduction is from the Unary Bin Packing (UBP for short) problem. This problem takes as input a set of items $A = \{a_1, \ldots, a_n\}$, a *size function* $s : A \to \mathbb{N}$ which returns the size of each item in unary encoding, and two integers $B$ and $k$. The question that interests us is whether the items of $A$ can fit into $k$ bins, so that every bin contains items of total size exactly $B$, and every item is assigned to exactly one bin. This problem was shown to be W[1]-hard when parameterized by $k$ in [41].

Let $(A, s, B, k)$, where $A = \{a_1, \ldots, a_n\}$, be an instance of UBP. We construct an instance of $\mathcal{C}$-CF as follows: for each $j \in [n]$, construct the clique $K^j$, which is of order $s(a_j)$. Then, for each $i \in [k]$, add one vertex $b_i$ and all the edges between $b_i$ and all the vertices of the cliques $K^j$, for all $j \in [n]$.

Let $G$ be the resulting graph, and set $\mathcal{C} = B + 1$. Observe that the twin-cover number of $G$ is at most $k$, as the set $\{b_1, \ldots, b_k\}$ is a twin-cover of $G$. We will show that any optimal partition $\mathcal{P}$ of $(G, \mathcal{C})$ has value $v(\mathcal{P}) = \sum_{j=1}^{n} \frac{s(a_j)(s(a_j)-1)}{2} + kB$ if and only if $(A, s, B, k)$ is a yes-instance of UBP.

For the first direction of the reduction, let $(A, s, B, k)$ be a yes-instance of UBP and let $f : A \to [k]$ be the returned function assigning items to bins, such that every bin contains items with total size exactly equal to $B$. We define a partition $\mathcal{P}$ of $V(G)$ into $k$ sets $C_1, \ldots, C_k$ as follows. For every $i \in [k]$, the set $C_i$ contains $b_i$ and all the vertices of the clique $K^j$ such that $f(a_j) = i$, for all $j \in [n]$. Clearly, $|C_i| = B + 1 = \mathcal{C}$ for every $i \in [k]$ and, thus, $\mathcal{P}$ is a $\mathcal{C}$-partition of $(G, \mathcal{C})$. Moreover, $E(\mathcal{P})$ contains all the edges that belong in the clique $K^j$, for every $j \in [n]$, and exactly $B$ edges incident to $b_i$, for each $i \in [k]$. In total, $v(\mathcal{P}) = |E(\mathcal{P})| = \sum_{j=1}^{n} \frac{s(a_j)(s(a_j)-1)}{2} + kB$.

For the reverse direction, let $(G, \mathcal{C})$ be an instance of $\mathcal{C}$-CF and $\mathcal{P} = C_1, \ldots, C_p$ be a partition of $(G, \mathcal{C})$ with value $v(\mathcal{P}) = \sum_{j=1}^{n} \frac{s(a_j)(s(a_j)-1)}{2} + kB$. Let $G' = G - \{b_1, \ldots, b_k\}$ and observe that $|E(G')| = \sum_{j=1}^{n} \frac{s(a_j)(s(a_j)-1)}{2}$.

**Claim 5.3.** *For each $i \in [k]$, there exists a unique $\ell \in [p]$ such that $b_i \in C_\ell$. Moreover, $p = k$ and $|C_\ell| = B + 1$.*

*Proof of the claim.* In order for $v(\mathcal{P})$ to have the correct value, and by the construction of $G$, each one of the vertices $b_1, \ldots, b_k$ contributes exactly $B$ edges to $v(\mathcal{P})$. Indeed, since $|C_i| \leq B + 1$, $i \in [p]$, no vertex can contribute more than $B$ edges towards $v(\mathcal{P})$. Assume now that there exist $i < i' \in [k]$ and $\ell \in [p]$ such that $b_i$ and $b_{i'}$ both belong to $C_\ell$. Then, since $\mathcal{C} = B + 1$ and by the construction of $G$, we have that $C_\ell$ contains at most $B - 1$ edges incident to $b_i$ and $b_{i'}$, which is a contradiction. Finally, for all $i \in [p]$, if $C_i$ contains a vertex from $\{b_1, \ldots, b_k\}$, then $|C_i| = B + 1$. It also follows that $p = k$. $\diamond$

**Claim 5.4.** *For each $j \in [n]$, all the vertices of $K^j$ belong in the same set of $\mathcal{P}$.*

*Proof of the claim.* In order for $v(\mathcal{P})$ to have the correct value, and by the construction of $G$, we have that for each $j \in [n]$, each vertex of $K^j$ contributes all of its incident edges in $G'$ towards $v(\mathcal{P})$. $\diamond$

We are now ready to show that $(A, s, B, k)$ is a yes-instance of the UBP problem. Let $\mathcal{P}$ be an optimal partition of $(G, \mathcal{C})$. It follows from Claim 5.3 that $\mathcal{P}$ consists of $k$ sets $\{C_1, \ldots, C_k\}$. We create the bins $B_1, \ldots, B_k$. For each $j \in [n]$, we insert the item $a_j$ in the bin $B_i$, $i \in [k]$, if and only if $K^j \subseteq C_i$. It follows from Claim 5.4 that each item of $A$ has been assigned to exactly one bin. Recall that for each $j \in [n]$, the item $a_j$ has size equal to the order of $K^j$ (by construction). Moreover, for each $j \in [n]$, the item $a_j$ corresponds exactly to the clique $K^j$. Thus, from Claim 5.4, we have that for each $i \in k$, $|C_i|$ is equal to the sum of the orders of the cliques contained in $C_i$, which is exactly equal to $B$.

It remains to show that $\sum_{a_j \in B_\ell} s(a_j) = B$ for all $\ell \in [k]$. Recall that $|V(K^j)| = s(a_j)$, for $j \in [n]$. Let $\ell \in [k]$. We have that $\sum_{a_j \in B_\ell} s(a_j) = \sum_{a_j \in B_\ell} |V(K^j)|$. Also, $|C_\ell| = \sum_{a_j \in B_\ell} |V(K^j)| + 1$ since $C_\ell$ contains the cliques that correspond to the items contained in $B_\ell$ and one vertex from $\{b_1, \ldots, b_k\}$. Thus, $\sum_{a_j \in B_\ell} s(a_j) = |C_\ell| - 1 = B$. $\square$

# 6 Conclusion

In this paper, we studied the $\mathcal{C}$-Coalition Formation problem, considering both its weighted and unweighted versions, through the lens of parameterized complexity. The main takeaway message is that the problems behave relatively well in regards to many widely used parameters, despite the multiple intractability results that we provided. On the one hand, our intractability results provide motivation towards a more heuristic-oriented approach. On the other hand, there are many rather interesting theoretical questions that are born from our research. In particular, we are wondering about the existence of an FPT algorithm for $\mathcal{C}$-CF$w$ parameterized by the vertex integrity. Other examples of interesting parameters that are left untouched by our work are the neighborhood diversity and the feedback vertex set of the input graph.

# References


[1] Alessandro Aloisio, Michele Flammini, and Cosimo Vinci. The impact of selfishness in hypergraph hedonic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1766–1773, 2020.

[2] Haris Aziz, Florian Brandl, Felix Brandt, Paul Harrenstein, Martin Olsen, and Dominik Peters. Fractional hedonic games. *ACM Transactions on Economics and Computation (TEAC)*, 7(2):1–29, 2019.

[3] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195:316–334, 2013.

[4] Yoram Bachrach, Pushmeet Kohli, Vladimir Kolmogorov, and Morteza Zadimoghaddam. Optimal coalition structure generation in cooperative graph games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 81–87, 2013.

[5] Christer Bäckström, Yue Chen, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider. The complexity of planning revisited—a parameterized analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1735–1741, 2012.

[6] Nathanaël Barrot, Kazunori Ota, Yuko Sakurai, and Makoto Yokoo. Unknown agents in friends oriented hedonic games: Stability and complexity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1756–1763, 2019.

[7] Nathanaël Barrot and Makoto Yokoo. Stable and envy-free partitions in hedonic games. In *IJCAI*, pages 67–73, 2019.

[8] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, Claude Guy Quimper, and Toby Walsh. The parameterized complexity of global constraints. In *AAAI Conference on Artificial Intelligence*, pages 235–240. AAAI Press, 2008.

[9] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[10] Hans L. Bodlaender. A partial $k$-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.

[11] Niclas Boehmer and Edith Elkind. Individual-based stability in hedonic diversity games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1822–1829, 2020.

[12] Anna Bogomolnaia and Matthew O Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[13] Felix Brandt, Martin Bullinger, and Anaëlle Wilczynski. Reaching individually stable coalition structures. *ACM Transactions on Economics and Computation*, 11(1-2):1–65, 2023.

[14] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[15] Robert Bredereck, Jiehua Chen, Rolf Niedermeier, and Toby Walsh. Parliamentary voting procedures: Agenda control, manipulation, and uncertainty. *Journal of Artificial Intelligence Research*, 59:133–173, 2017.

[16] Martin Bullinger and Stefan Kober. Loyalty in cardinal hedonic games. In *IJCAI*, pages 66–72, 2021.

[17] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[18] Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. Group activity selection problem with approval preferences. *International Journal of Game Theory*, 47:767–796, 2018.

[19] Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

[20] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[21] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[22] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[23] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[24] Jacques H Dreze and Joseph Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica: Journal of the Econometric Society*, pages 987–1003, 1980.

[25] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[26] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[27] Angelo Fanelli, Gianpiero Monaco, Luca Moscardelli, et al. Relaxed core stability in fractional hedonic games. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 182–188, 2021.

[28] M. R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. IWPEC'06, page 276–277, Berlin, Heidelberg, 2006. Springer-Verlag.

[29] Michele Flammini, Gianpiero Monaco, Luca Moscardelli, Mordechai Shalom, and Shmuel Zaks. Online coalition structure generation in graph games. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1353–1361, 2018.

[30] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[31] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[32] Robert Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *Parameterized and Exact Computation - 6th International Symposium, IPEC 2011*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2011.

[33] Robert Ganian, Thekla Hamm, Dušan Knop, Šimon Schierreich, and Ondřej Suchý. Hedonic diversity games: A complexity picture with more than two colors. *Artificial Intelligence*, 325:104017, 2023.

[34] Yong Gao. Data reductions, fixed parameter tractability, and random weighted d-cnf satisfiability. *Artificial Intelligence*, 173(14):1343–1366, 2009.

[35] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[36] Gregory Z. Gutin, Eun Jung Kim, Arezou Soleimanfallah, Stefan Szeider, and Anders Yeo. Parameterized complexity results for general factors in bipartite graphs with an application to constraint programming. *Algorithmica*, 64(1):112–125, 2012.

[37] Tesshu Hanaka and Michael Lampis. Hedonic games and treewidth revisited. In *30th Annual European Symposium on Algorithms, ESA 2022*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2022.

[38] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015.

[39] Ayumi Igarashi, Kazunori Ota, Yuko Sakurai, and Makoto Yokoo. Robustness against agent failure in hedonic games. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 364–370, 2019.

[40] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[41] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

[42] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[43] Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 528–541. ACM, 2023.

[44] Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1546–1558. SIAM, 2017.

[45] H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[46] Chaya Levinger, Amos Azaria, and Noam Hazon. Social aware coalition formation with bounded coalition size. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2667–2669, 2023.

[47] Gianpiero Monaco, Luca Moscardelli, and Yllka Velaj. Additively separable hedonic games with social context. *Games*, 12(3):71, 2021.

[48] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[49] Rolf Niedermeier and Peter Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73(3-4):125–129, 2000.

[50] Kazunori Ohta, Nathanaël Barrot, Anisse Ismaili, Yuko Sakurai, and Makoto Yokoo. Core stability in hedonic games among friends and enemies: Impact of neutrals. In *IJCAI*, pages 359–365, 2017.

[51] Martin Olsen. Nash stability in additively separable hedonic games and community structures. *Theory of Computing Systems*, 45:917–925, 2009.

[52] Dominik Peters and Edith Elkind. Simple causes of complexity in hedonic games. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 617–623, 2015.

[53] Walid Saad, Zhu Han, Tamer Basar, Mérouane Debbah, and Are Hjorungnes. Hedonic coalition formation for distributed task allocation among wireless agents. *IEEE Transactions on Mobile Computing*, 10(9):1327–1344, 2010.

[54] Jakub Sliwinski and Yair Zick. Learning hedonic games. In *IJCAI*, pages 2730–2736, 2017.

[55] Karsten Weihe. Covering trains by stations or the power of data reduction. *Proceedings of Algorithms and Experiments, ALEX*, pages 1–8, 1998.