Generalized Set and Graph Packing Problems

by

Jazmín Romero

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2015

Abstract

Many complex systems that exist in nature and society can be expressed in terms of networks (e.g., social networks, communication networks, biological networks, Web graph, among others). Usually a node represents an entity while an edge represents an interaction between two entities. A *community* arises in a network when two or more entities have common interests, e.g., related proteins, industrial sectors, groups of people, documents of a collection. There exist applications that model a community as a fixed graph H [98, 10, 119, 2, 142, 136]. Additionally, it is not expected that an entity of the network belongs to only one community; that is, communities tend to share their members.

The *community discovering* or *community detection* problem consists on finding all communities in a given network. This problem has been extensively studied from a practical perspective [61, 137, 122, 116]. However, we believe that this problem also brings many interesting theoretical questions. Thus in this thesis, we will address this problem using a more rigorous approach. To that end, we first introduce graph problems that we consider capture well the community discovering problem. These graph problems generalize the classical $\mathcal{H}$-Packing problem [88] in two different ways.

In the *$\mathcal{H}$-Packing with t-Overlap problem*, the goal is to find in a given graph G (the network) at least k subgraphs (the communities) isomorphic to a member of a family of graphs $\mathcal{H}$ (the community models) such that each pair of subgraphs overlaps in at most t vertices (the shared members). On the other hand, in *the $\mathcal{H}$-Packing with t-Membership problem* instead of limiting the pairwise overlap, each vertex of G is contained in at most t subgraphs of the solution. For both problems each member of $\mathcal{H}$ has at most r vertices and m edges. An instance of the $\mathcal{H}$-Packing with t-Overlap and t-Membership problems corresponds to an instance of the $\mathcal{H}$ -Packing problem for $t = 0$ and $t = 1$, respectively. We also restrict the overlap between the edges of the subgraphs in the solution instead of the vertices (called $\mathcal{H}$-Packing with t-Edge Overlap and t-Edge Membership problems).

Given the closeness of the r-Set Packing problem [87] to the $\mathcal{H}$-Packing problem, we also consider overlap in the problem of packing disjoint sets of size at most r. As usual for set packing problems, given a collection $\mathcal{S}$ drawn from a universe $\mathcal{U}$, we seek a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ consisting of at least k sets subject to certain disjointness restrictions. In the r-Set Packing with t-Membership, each element of $\mathcal{U}$ belongs to at most t sets of $\mathcal{S}'$ while in the r-Set Packing with t-Overlap each pair of sets in $\mathcal{S}'$ overlaps in at most t elements. For both problems, each set of $\mathcal{S}$ has at most r elements. We refer to all the problems introduced in this thesis simply as packing problems with overlap. Also, we group as the *family of t-Overlap problems*: $\mathcal{H}$-Packing with t-Overlap, $\mathcal{H}$-Packing with t-Edge Overlap, and r-Set Packing with t-Overlap. While we call the *family of t-Membership* problems: $\mathcal{H}$-Packing with t-Membership, $\mathcal{H}$-Packing with t-Edge Membership, and r-Set Packing with t-Membership.

The classical $\mathcal{H}$-Packing and r-Set Packing problems are NP-complete [87, 88]. We will show in this thesis that allowing overlap in a packing does not make the problems "easier". More precisely, we show that the $\mathcal{H}$-Packing with t-Membership and the r-Set Packing with t-Membership are NP-complete when $\mathcal{H} = \{H\}$ and H is an arbitrary connected graph with at least three vertices and $r \geq 3$, respectively.

Parameterized complexity, introduced by Downey and Fellows [44], is an exciting and interesting approach to deal with NP-complete problems. The underlying idea of this approach is to isolate some aspects or parts of the input (known as *the parameters*) to investigate whether these parameters make the problem tractable or intractable. The main goal of this thesis is to study the parameterized complexity of our packing problems with overlap. We set up as a parameter k the size of the solution (number of communities), and we consider as fixed-constants r, m and t.

We show that our problems admit polynomial kernels via two types of techniques: polynomial parametric transformations (PPTs) [16] and classical reduction algorithms [43]. PPTs are mainly used to show lower bounds and as far as we know they have not been used as extensively to obtain kernel results as classical kernelization techniques [96, 42]. Thus, we believe that employing PPTs is a promising approach to obtain kernel reductions for other problems as well. On the other hand, with non-trivial generalizations of kernelization algorithms for the classical $\mathcal{H}$-Packing problem [114], we are able to improve our kernel sizes obtained via PPTs. These improved kernel sizes are equivalent to the kernel sizes for the disjoint version when $t = 0$ and $t = 1$ for the t-Overlap and t-Membership problems, respectively.

We also obtain fixed-parameter algorithms for our packing problems with overlap (other than running brute force on the kernel). Our algorithms combine a search tree and a greedy localization technique and generalize a fixed-parameter algorithm for the problem of packing disjoint triangles [54]. In addition, we obtain faster FPT-algorithms by transforming our overlapping problems into an instance of the disjoint version of our problems.

Finally, we introduce the Π-Packing with $\alpha()$-Overlap problem to allow for more complex overlap constraints than the ones considered by the t-Overlap and t-Membership problems and also to include more general communities definitions. This problem seeks at least k induced subgraphs in a graph G subject to: each subgraph has at most r vertices and obeys a property Π (a community definition) and for any pair of subgraphs H_i, H_j, with $i \neq j$, we have that $\alpha(H_i, H_j) = 0$ holds (an overlap constraint).

We show that the Π-Packing with $\alpha()$-Overlap problem is fixed-parameter tractable provided that Π is computable in polynomial time in n and $\alpha()$ obeys some natural conditions. Motivated by practical applications we give several examples of $\alpha()$ functions which meet those conditions.

Acknowledgments

The journey to complete this thesis has been a really long one, but finally I have arrived to the goal line. I would like to thank to all the people that contributed in some way or another to make it possible. I really wanted to say thank you all from the bottom of my heart.

First, I would like to thank my supervisor Alejandro López Ortiz for giving me a second opportunity and have faith on me, when nobody else did (not even myself). This thesis would not have existed without him. Thank you Alex, I am and I will be forever grateful to you.

I would like to thank the members of my committee, Therese Biedl, Dan Brown, Joseph Cheriyan, Alejandro López Ortiz, Ian Munro and Saket Saurabh for taking their time to read my thesis and for their invaluable feedback about my research. I am also thankful to Henning Fernau and Alejandro López Ortiz for their collaboration in this project. I have been really fortunate to work and learn with you. Thanks as well to Naomi Nishimura for her research guidance in early stages of this project.

Special thanks goes to Wendy Rush and Margaret Towell for their help and assistance always that I needed it.

I am very grateful to the National Council of Science and Technology (CONACYT, Mexico) for granting me with a scholarship to support four years of my studies. I would also like to thank the financial support that I received from Brock Solutions and the Government of Alberta (Child Care Subsidy) which indirectly helped me to finish this thesis.

I have been blessed to have many good friends by my side in my life. Thanks to all of them for making this journey more enjoyable.

To my little family: Ricardo and Laurita. Ricardo thank you for your love, support and patience during this project and in our life together in general. Laura, among all the roles that I have in my life, the one I love the most is being your mother. Thank you for filling our lives with so much joy. To my parents, Nancy and Eduardo: nothing of what I have accomplished in my life would have been possible without you. Thank you for always being there for me. Last but not least, thanks to God. I go to bed every night counting all the blessings of my life, and I realize that I do not have enough fingers to count. I do not know what I have done to be loved so much.

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

List of Algorithms

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation: Discovering Communities in Networks

Many complex systems that exist in nature and society can be expressed in terms of networks. In these networks, each node is an entity, for example a protein, a paper, or a person, and each edge represents a relationship between two entities, e.g., protein interaction, co-citation, or communication [104, 122].

Real networks have different properties compared to random graphs. They present a high level of order and organization. For example, many vertices with low degree coexist with some vertices with large degree. In addition, there are more edges within groups of vertices than edges connecting these groups. This feature of real networks is called *community structure* [66].

A *community* arises in a network when two or more entities have common interests. We can see examples of communities in different contexts. For example, in society we can think of groups of friends, families, or co-workers as communities [29, 63]. In protein-interaction networks, communities could be a group of proteins having the same or similar function in the same cell [26, 124]. On the other hand, in the World Wide Web a community may represent a group of web pages related to the same topic [40]. In this way, the members of a community tend to share several properties. For example, in social networks, people that belong to a community are likely to have relative homogeneity of customs, beliefs, and interests [51, 112]. Figure 1.1 shows a Protein-Interaction network and the communities formed in this network.

Extracting these communities is helpful to understand the structure of the network, the interactions among members, and the way that information spreads through the network. For example, benefits in the World Wide Web are to automate the process of creating Web directories and to visualize search results grouped by categories [7]. On the other hand, identifying communities of customers with similar purchasing interests can help in providing more efficient recommendation systems [97]. The problem of extracting these communities is known as the *community discovering problem* or *community detection problem.* With the advent of the Internet, the problem of discovering communities in online social networks (such as Facebook or LinkedIn) has received even more attention [113, 135, 107].

Intuitively, we could consider a community as a part of the network in which the nodes are highly interconnected with each other and loosely connected to the rest of the network [112]. However, there is

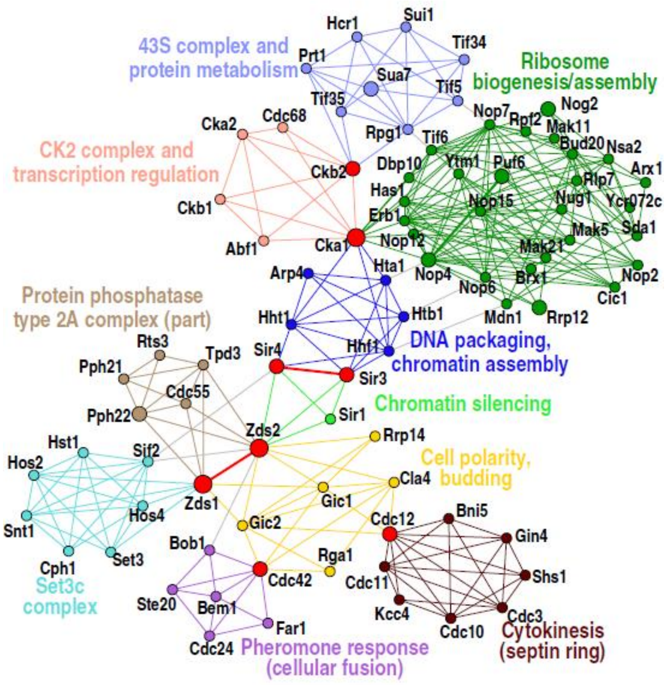
43S complex and protein metabolism
Ribosome biogenesis/assembly
CK2 complex and transcription regulation
Protein phosphatase type 2A complex (part)
DNA packaging, chromatin assembly
Chromatin silencing
Cell polarity, budding
Set3c complex
Pheromone response (cellular fusion)
Cytokinesis (septin ring)

Figure 1.1: Overlapping communities in a Protein-Interaction network. Figure obtained from [119].

no universal definition of a community. In fact, the way a community is modeled is highly dependent on the application being modeled by the network. There are applications that model a community as a fixed graph H [98, 10, 119, 2, 142, 136].

Early approaches to solve the community discovering problem consider member disjoint communities [46, 66, 59, 7]. Naturally, one person can have different groups of friends, and one protein can belong to more than one protein complex. Therefore, in realistic scenarios, communities can share members [119, 112, 136, 122, 107]. In Figure 1.1 we can observe that the overlapping proteins are highlighted in red.

The community discovery problem has been extensively studied in practice [61, 137, 122, 116]. To our knowledge, few works [111, 5] have considered the problem of discovering overlapping communities from a more theoretical or rigorous approach. This is not a negative situation at all, as it is a problem that arises in practice. However, we believe that there are many interesting questions that need a more theoretical treatment. For example, to determine the computational complexity of the problem, propose particular classes of networks or community models that make the problem "easier", determine which aspects of the problems (number of communities, degree or diameter of the network, etc) could make this problem more approachable, among others.

In this work, we focus on treating the community discovering problem from a theoretical viewpoint. To that end, in the next section, we introduce graph problems that we believe abstract and formalize the community discovering problem.

1.2 Packing Problems Allowing Overlap

Given a graph G and a family of graphs $\mathcal{H}$, *the $\mathcal{H}$-Packing problem* asks for the maximum number of vertex-disjoint subgraphs in G where each one is isomorphic to a member of the family $\mathcal{H}$. This set of vertex-disjoint subgraphs is called a *packing* of G. This problem arises in practical applications such as scheduling [89, 9, 76], wireless sensor tracking [11], wiring board design and code optimization [88], among others, and it has been widely studied (see Section 2.4 for a condensed list on results on this problem).

If we take a closer look at the definition of this problem, and we consider that the input graph G represents a network while each member in $\mathcal{H}$ represents a community model (recall that there exists applications that model communities as fixed-graphs [2, 119, 142, 98]), then each subgraph in a packing of G could represent a community of the network. However, the $\mathcal{H}$-Packing fails to abstract the community discovering problem as it will output only member-disjoint communities (subgraphs in a packing are vertex-disjoint), and we learned in the previous section that overlap between communities is an inherent characteristic of the network structure.

On the other hand, despite about 40 years of study in the $\mathcal{H}$-Packing problem, no generalization that consider overlap between the subgraphs in a packing has been proposed (as far as we are aware). Therefore in this thesis, we generalize the $\mathcal{H}$-Packing problem to consider overlap in a packing in two different ways.

The *$\mathcal{H}$-Packing with t-Overlap problem* seeks for the maximum collection of subgraphs (the communities) in a graph G (the network) such that: i. each subgraph is isomorphic to a member of $\mathcal{H}$ (the community models) and ii. each pair of subgraphs overlaps in at most t vertices, for some $t \geq 0$ (the shared members). Notice that for $t = 0$, we are back to the classical $\mathcal{H}$-Packing problem.

The $\mathcal{H}$-Packing with t-Overlap problem restricts the overlap between a pair of communities. Such type of overlap can be found in certain clustering algorithms as in [8]. Bounding the pairwise overlap is not the only way of allowing overlap in a packing. Thus, we formulate a different generalization of the the $\mathcal{H}$-Packing problem which instead of restricting the pairwise overlap, bounds the number of times a vertex of G is contained in a subgraph of a packing. This generalization is called the $\mathcal{H}$-Packing with t-Membership problem and is defined next.

Given a graph G, the *$\mathcal{H}$-Packing with t-Membership problem* seeks the maximum collection of subgraphs, each one isomorphic to a member of $\mathcal{H}$, such that each vertex of G (a member of the network) belongs to at most t subgraphs of the collection, for some $t \geq 1$. Once again, for $t = 1$ this problem is the classical *$\mathcal{H}$-Packing*.

Based on the facts that a community can be represented as a fixed graph H (thus a family of graphs $\mathcal{H}$ is more flexible as it allows for more than one community model) and that overlap is a necessary feature in the community discovering problem, we believe our problems capture well the community discovering problem. In addition, we formulate variants for these problems that regulate the overlap in terms of edges instead of vertices or consider induced subgraphs in G as well. All these problems are formally defined in Chapter 2.

Packing problems also arise in the context of sets. We are referring to the classical Set Packing problem. Given a collection of sets $\mathcal{S}$ drawn from a universe $\mathcal{U}$, the *Set Packing problem* looks for the sub-collection of $\mathcal{S}$ of maximum cardinality such that all sets in that sub-collection are pairwise-disjoint. Such sub-collection is called a *packing* of $\mathcal{S}$. The pair $\mathcal{S}$ and $\mathcal{U}$ can be treated as an hypergraph, where the vertices are the members of $\mathcal{U}$ and the hyper-edges are the members of $\mathcal{S}$. Thus a packing of an hypergraph asks for a subset of disjoint-edges of maximum cardinality. When every element of $\mathcal{S}$ has at most r elements, the problem is denoted as the *r-Set Packing problem*.

Given the closeness between the r-Set Packing and the $\mathcal{H}$-Packing problems, we also formulate generalizations for the r-Set Packing problem that consider overlap. More precisely, the *r-Set Packing with t-Overlap problem* seeks for the sub-collection of $\mathcal{S}$ of maximum cardinality such that each pair of sets overlaps in at most t elements. In contrast, in the *r-Set Packing with t-Membership problem*, each member of $\mathcal{U}$ is contained in at most t sets of the sub-collection. These generalizations are born from purely theoretical motivation but we believe that are interesting on their own.

1.3 Thesis Overview

As we shall show in Chapter 3, some of our packing problems that allow overlap are NP-complete. This implies that unless P=NP, there is no algorithm that can provide a solution for our problems in polynomial time. There exists several alternatives to deal with NP-complete problems: approximation algorithms [6, 81], average-case analysis [82], randomized algorithms [115], and heuristics methods [109].

Parameterized complexity (or multivariate complexity) introduced by Downey and Fellows [44, 43] (about 25 years ago) is an exciting approach to cope with NP-complete problems. The underlying idea is to isolate some aspects or parts of the input (known as *the parameter*) of a problem and find a solution in time exponential depending only on the parameter and polynomial with respect to the input. Contrary to other approaches to deal with NP-complete problems, parameterized complexity could give more insight about which aspect of the problem make it tractable or intractable. Thus, the main goal of this thesis is the study of the parameterized complexity of packing problems allowing overlap.

The first aspect of the input of the packing problems that we could consider as the parameter is the size of the solution. That is, instead of finding all communities in the network, we are interested on determining only k. In this way, k will be our parameter, and our problem becomes a parameterized problem. After that we endeavor to determine whether our parameterized problems admit or not algorithms that run in $f(k)n^{O(1)}$ (where f is a computable function, possible exponential, and n is the size of the network). Ideally, one hopes that these algorithm would be of practical use for small values of k.

The remainder of this thesis is organized as follows. In Chapter 2, we outline the basic notation and terminology, and the formal definitions of our parameterized problems. That chapter also includes a list on related problems to our packing problems with overlap. In Chapter 3, we show the computational complexity of our problems. After that we divide our thesis in two main objectives: reduce our parameterized problems to problem kernels (i.e. reduced instances with size bounded by $O(f(k))$ and determine algorithms that solve our problems in $O(f(k)n^{O(1)})$ time. Even though finding a kernel for our problems would imply immediate FPT-algorithms (by running a brute force on the kernel), we would like to determine faster FPT-algorithms for our problems. Achieving any of these goals would imply that our parameterized problems are *fixed-parameter tractable*. We dedicate Chapters 4-7 and Appendix A to the first goal, while we explore the second one in Chapter 8. Finally in Chapter 9, we generalize our packing problems even more. We believe that this generalization captures many more realistic scenarios. We show that our generalization remains fixed-parameter tractable when subject to some natural conditions. Appendix B is a compendium of all the parameterized problems considered in this thesis.

Most of the results presented in this thesis have appeared in the following articles and/or manuscripts:

- *The $\mathcal{G}$-Packing with t-Overlap Problem.* In Proceedings of the 8th International Workshop on Algorithms and Computation (WALCOM 2014) [126].
- *A Parameterized Algorithm for Packing Overlapping Subgraphs.* In Proceedings of the 9th International Computer Science Symposium (CSR 2014) [127]
- *Parameterized algorithms for the H-Packing with t-Overlap problem.* Journal of Graph Algorithms and Applications [103].
- *Kernelization algorithms for packing problems allowing overlaps.* In Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC 2015) [56].
- *Using parametric transformations toward polynomial kernels for packing problems allowing overlaps.* ACM Transactions on Computation Theory (Extended version of [56]) [57].
- *Arbitrary overlap constraints in graph packing problems.* Manuscript 2015. [102].

Chapter 2

Preliminaries

2.1 Notation and Terminology

In the following section, we introduce the notation and terminology adopted in this thesis. Recall that we study problems that deal with graphs or sets. Thus, it may be the case that some terms apply for both types of problems. Let us start with the terminology related to graphs. The reader is referred to the book of Diestel [38] for a complete review of graph theory concepts.

Graphs are undirected and simple unless specified otherwise. For a graph G, $V(G)$ and $E(G)$ denote its sets of vertices and edges, respectively. We use the letter n to denote the size of $V(G)$. The neighborhood of a vertex v is denoted by $N_G(v) = \{u \mid (u, v) \in E(G)\}$. In the same way, for a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \mid (u, v) \in E(G), u \in S; v \notin S\}$. The degree of a vertex v is $|N_G(v)|$ and will be denoted as $deg_G(v)$. The distance between two vertices u and v in G, denoted as $dist_G(u, v)$, is the length of a shortest path from u to v in G. We drop the index G whenever it is clear from the context. A subgraph of G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The induced subgraph of G by a set $S \subseteq V(G)$ is denoted by $G[S]$; $G[S]$ has vertex set S and edge set $\{(u, v) \in E(G[S]) \mid \ u, v \in S, \ (u, v) \in E(G)\}$.

We adopt the following notation for some graph classes that are used in this document. A path on s vertices and $s - 1$ edges is written as P_s. We denote a star on s vertices and $s - 1$ edges as S_s. A clique or complete graph on s vertices is denoted as K_s. A cycle on s vertices and s edges is denoted as C_s. A bipartite graph is a graph G whose set of vertices $V(G)$ can be partitioned into two disjoint sets, $V(G) = A \cup B$, such that any edge of G has an endpoint in A and the other in B. A complete bipartite graph or a bipartite clique, written $K_{p,q}$ where $p = |A|$ and $q = |B|$, is a bipartite graph in which every two vertices from different partitions are adjacent. A graph is *chordal* if it does not contain a cycle on n vertices as an induced subgraph for $n > 3$. A *cograph* is a graph which does not contain an induced path of three edges. Thus, in a connected cograph the greatest distance between any pair of vertices is at most two; that is, the diameter of a connected cograph is at most two. A graph is *planar* if it can be drawn in the plane in such a way that no edges cross each other.

Given a graph G, let $b : V(G) \to \mathbb{N}$ a degree constraint for every vertex. A subset $M \subseteq E(G)$ is called a *b-matching* of G, if for all $v \in V(G)$ the number of edges in M incident to v is at most $b(v)$. The

Cardinality b-Matching problem refers to the problem of finding a b-matching of G of maximum cardinality and can be solved in polynomial time [84]. A b-matching of maximum cardinality is called a *maximum cardinality b-matching.* Observe that for $b(v) = 1$ for all $v \in V(G)$, we are back to the classical maximum matching problem. A vertex v is *s-matched* if the number of edges of M incident to v is s, for $1 \leq s \leq b(v)$. Otherwise, v is *unmatched.* An *M-alternating path* in G is a path in G which starts with an unmatched vertex and then contains alternately edges from $E(G)\backslash M$ and M. That is, a path $p = v_1 v_2 \dots v_p$, where v_1 is an unmatched vertex, $v_2 \dots v_{p-1}$ are s-matched vertices, and v_p could be s-matched or unmatched. An M-alternating path is an *M-augmenting path* if it contains at least one s-matched vertex v with $s < b(v)$ in the path $v_2 \dots v_p$ or v_p is unmatched. The following lemma is a generalization of the well-known fact that a maximum matching of a graph G does not contain an augmenting path.

Lemma 1. *[67] Any maximum cardinality b-matching M of a graph G does not contain an M-augmenting path.*

A set of vertices $V' \subseteq V(G)$ or edges $E' \subseteq E(G)$ is *contained* in a subgraph H, if $V' \subseteq V(H)$ or $E' \subseteq E(H)$, respectively. On the other hand, $V(E')$ denotes the set of vertices corresponding to the end-points of all the edges in E', i.e., $V(E') = \{u, v \mid (u, v) \in E'\}$. For a collection of subgraphs $\mathcal{Q}$, $|\mathcal{Q}|$ is the number of subgraphs in $\mathcal{Q}$ while $V(\mathcal{Q}) = \bigcup_{i=1}^{|Q|} V(Q_i)$ where $Q_i \in \mathcal{Q}$.

Two subgraphs H_i and H_j are *vertex-disjoint* if they do not share vertices, i.e., $V(H_i) \cap V(H_j) = \emptyset$. Otherwise, we say that H_i and H_j *overlap* in $|V(H_i) \cap V(H_j)|$ vertices. In addition, $G[V(H_i) \cap V(H_j)]$ is the *overlap region* between H_i and H_j. Similarly, H_i and H_j are *edge-disjoint* if they do not share edges, i.e., $E(H_i) \cap E(H_j) = \emptyset$. We also use the term overlap for the edge-version, i.e., H_i and H_j overlap in $|E(H_i) \cap E(H_j)|$ edges.

All our graph problems deal with a family of graphs $\mathcal{H}$ where each $H \in \mathcal{H}$ is an arbitrary graph. Let $r(\mathcal{H}) = \max\{|V(H)| : H \in \mathcal{H}\}$ denote the order of the biggest graph in $\mathcal{H}$. Observe that in order to have this as a reasonable notion, we always (implicitly) require $\mathcal{H}$ to be a finite set. We simply write r instead of $r(\mathcal{H})$ if $\mathcal{H}$ is clear from the context. Similarly, $m(\mathcal{H}) = \max\{|E(H)| : H \in \mathcal{H}\}$, and we write m whenever it is clear from the context. When the family of graphs $\mathcal{H}$ is composed of only one graph H, we replace $\mathcal{H}$ by H in the notation that involves $\mathcal{H}$.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijection $f : V \to V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$. A subgraph of G that is isomorphic to some $H \in \mathcal{H}$ is called an *$\mathcal{H}$-subgraph.* We denote as $\mathcal{H}_G$ the set of all $\mathcal{H}$-subgraphs in G, in this way, $|\mathcal{H}_G| = O(n^{r^2})$.

A *packing* of a graph G is a set of vertex-disjoint subgraphs of G each one isomorphic to a graph in $\mathcal{H}$. An *edge-packing* of G is a set of edge-disjoint subgraphs each one isomorphic to a graph in $\mathcal{H}$. In this thesis, we will consider problems that modify this notion of packing by allowing overlap between the subgraphs.

Let $\mathcal{V}$ be a collection of sets of vertices each of which forms a subgraph (not necessarily induced) isomorphic to some $H \in \mathcal{H}$. That is, for each $V \in \mathcal{V}$, there is an $H' \subseteq G[V]$ that is an $\mathcal{H}$-subgraph, (here $\subseteq$ also refers to the subgraph relation). Hence, for each $V \in \mathcal{V}$, $|V| \leq r$, and $|\mathcal{V}| = O(n^r)$. Also, let $\mathcal{E}$ be a collection of sets of edges each of which induces a subgraph isomorphic to some $H \in \mathcal{H}$. That is, for each $E \in \mathcal{E}$, there is an $H \subseteq G[V(E)]$ that is an $\mathcal{H}$-subgraph. $V(E)$ denotes the set of vertices corresponding to the end-points of the edges E. For each $E \in \mathcal{E}$, $|E| \leq m$, and $|\mathcal{E}| = O(n^m)$. Notice that there could be two H-subgraphs H_1, H_2 in $\mathcal{H}_G$ with the same element $V \in \mathcal{V}$ but different set of edges $E_1, E_2 \in \mathcal{E}$. In this way, $V(E_1) = V(E_2)$ but $E_1 \neq E_2$.

We will now provide the notation and terminology involving sets. The universe is $\mathcal{U} = \{u_1, \ldots, u_n\}$, and we denote the size of the universe by n. Let $\mathcal{S} = \{S_1, \ldots, S_l\}$ be a collection of sets drawn from $\mathcal{U}$. We usually denote as u, S, s an element of $\mathcal{U}$, a set of $\mathcal{S}$, and a subset of $\mathcal{U}$, respectively. For $\mathcal{S}' \subseteq \mathcal{S}$, $val(\mathcal{S}')$ denotes the union of all elements in $\mathcal{S}'$. For $P \subset \mathcal{U}$, P is *contained* in $S \in \mathcal{S}$, if $P \subseteq S$. For a subset $s \subset \mathcal{U}$, $\mathcal{S}(u)$ collects all the sets in $\mathcal{S}$ that contain s.

Two sets S_i and S_j are element-disjoint if they do not share elements, i.e., $S_i \cap S_j = \emptyset$. Two sets S,$S' \in \mathcal{S}$ *overlap in* $|S \cap S'|$ elements and they *conflict* if $|S \cap S'| \geq t + 1$. For any two sets $S, S' \in \mathcal{S}$, $|S \cap S'|$ is the *overlap size* while $S \cap S'$ is the *overlap region.*

A *packing* of $\mathcal{U}$ is a collection of element-disjoint sets of $\mathcal{S}$. In this work, we will consider a modification to this definition by allowing overlap in the sets of a packing.

2.2 Basics on Classical and Parameterized Complexity

We will provide in this section some basic notions on classical computational complexity. The reader is referred to Garey and Johnson [65] and Papadimitriou [121] for excellent sources about this topic.

In this thesis, we assume a generic one-processor random access machine (RAM) as our underlying model of computation. In the RAM model, instructions (arithmetic, memory access, control, among others) are executed one after another and each one takes a unit time.

Informally, the *running time* of an algorithm on a particular input is the number of steps executed. The *time complexity* of an algorithm A is the function $t : \mathbb{N} \to \mathbb{N}$, where $t(n)$ is the maximum number of steps that A uses on any input of length n. Thus, the running time of A is $t(n)$.

A *problem* or a *language* is a set L of strings of length at most n over a finite alphabet Σ ($L \subseteq \Sigma^n$). A language is *decided* by an algorithm A, if A can determine whether the string s is in L or not, for every string $s \in \Sigma^n$.

For a given function $g(n)$, we denote $g(n) = O(f(n))$ if there exists constants c_0 and n_0 such that $g(n) \leq c_0 f(n)$ for all $n \geq n_0$. For a given a function $g(n, k)$ we write $g(n, k) = O^*(f(k))$ if there exists k_0, c, and n_0 such that $g(n, k) \leq f(k)n^c$ for all $k \geq k_0$ and $n \geq n_0$.

The time complexity class P contains the problems that are solvable in polynomial-time. More specifically, they are problems that can be solved in time $O(n^c)$ for some constant c, n is the input size of the problem. On the other hand, the class *NP* consists of those problems that are "verifiable" in polynomial time. That is, given a "certificate" of a solution, there exists an algorithm that can verify in polynomial time whether the certificate is correct or not.

A language L is *NP-complete*, if it satisfies the following conditions: 1. L is in NP and 2. Every language L' in NP is polynomial time reducible to L. If a language L satisfies property 2 (but not necessarily property 1), L is NP-hard. A language is in co-NP if and only if its complement is in NP.

In Chapter 1, we motivate the use parameterized complexity to cope with the intractability of our packing problems with overlap. We give next some general notions of this approach as introduced by Downey and Fellows [44]. We refer to the reader to [44, 117, 43, 33] for comprehensive reviews.

Definition 1. *A* parameterized problem *is a language* $L \subseteq \Sigma^* \times \mathbb{N}$, *where* Σ *is a finite alphabet. The second component is called the* parameter.

Intuitively, a parameterized problem is a decision problem where some aspect of its input constitutes a parameter. That is, the input to a parameterized problem is a pair $(x, k) \in \Sigma^* \times \mathbb{N}$ where x is the instance of the problem and k is the parameter (we assume here that the parameter is an integer).

Definition 2. *A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is* fixed-parameter tractable *(or belongs to the class FPT), if there is an algorithm A, a polynomial p, and a computable function f such that A decides for any input $(x, k) \in \Sigma^* \times \mathbb{N}$ whether or not $(x, k) \in L$ in at most $f(k)p(|x|)$ time, where $|x|$ denotes the length of the input x. The algorithm A is called a* fixed-parameter algorithm *or simply an* FPT-algorithm.

A parameterized problem is reduced to a kernel, if any instance can be reduced to a smaller instance such that: the reduction is in polynomial time, the size of the new instance is bounded by a function depending only on an input parameter, and the smaller instance has a solution if and only if the original instance has one. Formally,

Definition 3. *A* kernelization *or a* kernelization algorithm *for a parameterized problem L is a polynomial time transformation that maps an instance (x, k) to an instance (x', k') (the* kernel*) such that:*

(i) $(x, k) \in L$ if and only if $(x', k') \in L$,

(ii) $k' \leq k$,

(iii) $|x'| \leq f(k)$ for some arbitrary computable function f on non-negative integers.

We say that L has *kernel size* $f(k)$. The size of the kernel could be exponential or worse. When f is bounded by a polynomial in k, L has a *polynomial kernel*. Bodlaender et al. [16] introduce the notion of polynomial parameter transformations to show that some parameterized problems do not admit polynomial kernels unless NP⊆coNP/poly.

Definition 4. *Let L and Q be parameterized problems. We say that L is* polynomial time and parameter reducible *to Q, if there exists a polynomial time computable function f and a polynomial p that for all $x \in \{0, 1\}^*$ and $k \in \mathbb{N}$, if $f(x, k) = (x', k')$, then the following hold: (i) $(x, k) \in L$ if and only if $(x', k') \in Q$, and (ii) $k' \leq p(k)$.*

We say that f is a polynomial parametric transformation *from L to Q.*

Every problem that is in FPT has a kernelization algorithm [117, 44], and every problem that is reduced to a kernel is in FPT. This last follows because after reducing a problem to a kernel, we can run a brute-force algorithm on the kernel. Since the kernel has size bounded by a parameter, then the brute-force algorithm (running on the kernel) is an FPT algorithm.

Unfortunately, not every parameterized problem is fixed-parameter tractable. $W[1]$ is the lowest class of parameterized intractability. The fundamental conjecture that $FPT \neq W[1]$ is analogous to the conjecture that $P \neq NP$. In this way, a problem that is in $W[1]$ does not allow for a fixed-parameter algorithm unless $FPT = W[1]$. This problem is called *fixed-parameter intractable.*

The running time of a fixed-parameter algorithm is polynomial in the input size n but possibly exponential or worse in the parameter k. Therefore, one expects for small values of k that $f(k)$ leads to a fixed-parameter algorithm of practical use.

2.3 Our Parameterized Problems

We consider packing problems that allow overlap in a solution in two different ways. In the *family of t-Membership problems*, the *overlap condition* bounds the number of times an *object* is contained in a *group* of a solution. On the other hand, in the *family of t-Overlap problems*, that condition bounds the number of *objects* that a pair of *groups* of a solution shares. Depending on the type of problem (packing sets or packing graphs) an object is an element, a vertex or an edge, and a group is a set or a subgraph.

We next introduce the formal definitions of the parameterized versions of the t-Membership and t-Overlap problems considered in this work. Overloading the terminology, we will use the same problem name for the parameterized decision versions of our problems as their maximization versions introduced in Chapter 1. Let k be the parameter size of the solution and r, m and t be fixed constants in all the following definitions.

2.3.1 Pairwise Overlap Problems

Our t-Overlap problems are defined next. In these definitions, $r \geq 1$ and $t \geq 0$ are constants.

The r-Set Packing with t-Overlap problem
Instance: A collection $\mathcal{S}$ of distinct sets, each of size at most r, drawn from a universe $\mathcal{U}$ of size n, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a (k, r, t)-*set packing*, i.e., a collection of at least k sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where $|S_i \cap S_j| \leq t$, for any pair S_i, S_j with $i \neq j$?

The $\mathcal{H}$-Packing with t-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-$\mathcal{H}$-*packing*, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph from $\mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|V(H_i) \cap V(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The Induced-$\mathcal{H}$-Packing with t-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-*induced-$\mathcal{H}$-packing*, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph from $\mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|V(H_i) \cap V(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The $\mathcal{H}$-Packing with t-Edge-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, m, t)-*edge-$\mathcal{H}$-packing*, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to a graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$, and $|E(H_i) \cap E(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

2.3.2 Membership Problems

We next introduce the family of t-Membership problems. For these problems, $r \geq 1$ and $t \geq 1$ are constants.

The r-Set Packing with t-Membership problem
Input: A collection $\mathcal{S}$ of distinct sets, each of size at most r, drawn from a universe $\mathcal{U}$ of size n, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a (k, r, t)-*set membership*, i.e., at least k sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where each element of $\mathcal{U}$ is in at most t sets of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Membership problem.
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-*$\mathcal{H}$-membership*, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$ for $i \neq j$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

The Induced-$\mathcal{H}$-Packing with t-Membership problem.
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-*induced-$\mathcal{H}$-membership*, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$ for $i \neq j$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Edge Membership problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, m, t)-*edge-$\mathcal{H}$-membership*, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$ for $i \neq j$, and every edge in $E(G)$ belongs to at most t subgraphs of $\mathcal{K}$?

An instance of the set packing problems with membership and overlap will be denoted as $(\mathcal{S},\mathcal{U},k)$ while an instance for the graph packing problems will consists of (G,k).

Sometimes the size k will be dropped from the notation (k,r,t)-set membership, (k,r,t)-$\mathcal{H}$-membership, (k,r,t)-set packing and (k,r,t)-$\mathcal{H}$-packing and be simply denoted as (r,t)-set membership, (r,t)-$\mathcal{H}$-membership, (r,t)-set packing and (r,t)-$\mathcal{H}$-packing, respectively.

An (r,t)-$\mathcal{H}$-membership P of G is *maximal* if every $\mathcal{H}$-subgraph of G that is not in P has at least one vertex v contained in t $\mathcal{H}$-subgraphs of P. Similarly, an (r,t)-set packing $\mathcal{M}$ of $\mathcal{S}' \subseteq \mathcal{S}$ is *maximal* if any set of $\mathcal{S}'$ that is not already in $\mathcal{M}$ conflicts with some set in $\mathcal{M}$. That is, for each set $S \in \mathcal{S}'$ where $S \notin \mathcal{M}$, $|S \cap S'| \geq t+1$ for some $S' \in \mathcal{M}$.

The parameterized versions of the $\mathcal{H}$-Packing and r-Set Packing problems (when they are parameterized by size of the solution) are included in Appendix B. Notice that when $t=1$ and $t=0$ for the t-Membership and t-Overlap problems, respectively, we are back to the classical r-Set Packing, $\mathcal{H}$-Packing and Edge-$\mathcal{H}$-Packing problems.

2.4 Overview of Related Results

We have introduced theoretical formalizations for the problem of discovering communities in networks. These formalizations are gathered in the graph packing problems that allow overlap: namely, the $\mathcal{H}$-Packing with t-Overlap and t-Membership problems. We have also proposed similar formulations for packing hypergraphs allowing overlap, i.e., the r-Set Packing with t-Overlap and t-Membership.

To the best of our knowledge, our formalizations are newly introduced to the literature. Thus, we will provide a short list of results on the problems that are closest to ours: the $\mathcal{H}$-Packing and r-Set Packing problems. Our emphasis will be on classical complexity and parameterized complexity. We also point out other problems that could be related to our packing problems with overlap as well.

2.4.1 Results on the $\mathcal{H}$-Packing Problem

In this section we first discuss the results related to the computational complexity of the $\mathcal{H}$-Packing problem. After that, we list some of the results related to the parameterized complexity of this problem.

Hell and Kirkpatrick [88] proved that the H-Packing problem is NP-complete, if H has a component with at least three vertices. This result implies that the only possible polynomial time solvable case of the H-Packing problem is when H is an edge, unless P = NP. In this case, the K_2-Packing problem is better known as the maximum matching problem [32, 48, 49].

Dyer and Frieze [47] showed that the H-Packing remains NP-complete even when G is a planar graph and H is an arbitrary connected graph on at least three vertices. This complexity result holds when G is planar bipartite and H is a connected graph, a tree, or a path on at least three vertices [47]. In addition, Berman et al. [12] showed that this problem is still NP-complete if both G and H are planar graphs. Caprara and Rizzi [18] focused on the particular case of $H = K_3$. The authors shows that the K_3-Packing problem is NP-hard for planar graphs if the maximum degree of the input graph G is at least 4. Otherwise, if G has maximum degree at most 3, the problem is solvable in polynomial time.

Due to the intractability of the H-packing problem, an alternative way to study it is to consider a specific graph class for the graph G. Masuyama and Ibaraki [105] presented a linear-time algorithm for the H-packing problem when G is a tree and H is a path of length k. Lingas [100] has shown that the H-Packing can be solved in $O(|V(G)|^{5/2})$ when G is a tree of arbitrary size and thus H is a tree as well. Kovacs and Lingas [94] present an $O((|V(G)||V(H)|)^2)$ algorithm for the H-packing problem when both G and H are biconnected outerplanar graphs. Biconnected outerplanar graphs are partial 2-trees; Dessmark et al. [37] generalize the previous result for partial k-trees. They present an $O(|V(G)|^{k+1}|V(H)|+|V(H)|^k)$ algorithm to solve the H-packing problem when the G is a partial k-tree and H is a k-connected partial k-tree.

We now list the works for the $\mathcal{H}$-Packing problem that have considered more than one graph in the family $\mathcal{H}$. In general, $\mathcal{H}$-Packing is polynomial-time solvable, if every component of every graph in $\mathcal{H}$ has at most two vertices. When $\mathcal{H}$ contains only complete graphs, then $\mathcal{H}$-Packing problem is NP-complete, if all $H \in \mathcal{H}$ have at least three vertices [88, 78]. However, if $\mathcal{H} = \{K_2, K_3\}$ the $\mathcal{H}$-packing problem is solvable in polynomial [78].

For a family of complete bipartite graphs, the $\mathcal{H}$-Packing problem is polynomially solvable only if $\mathcal{H}$ is composed of the family of stars $\{S_1, S_2, \dots\}$. For any other family of complete bipartite graphs, the $\mathcal{H}$-packing problem is NP-complete or harder [77, 79].

In the case that $\mathcal{H}$ is the family of cycles, the $\mathcal{H}$-packing problem is NP-complete unless $\mathcal{H}$ is composed of all cycles, i.e., $\mathcal{H} = \{C_3, C_4, \dots\}$ [49, 80], all cycles except C_3, i.e., $\mathcal{H} = C_4, C_5, \dots$ [80], or all cycles except C_4, i.e., $\mathcal{H} = \{C_3, C_5, C_6 \dots\}$ [4].

A graph H is *hypomatchable* if H has no perfect matching but $H \backslash v$ has a perfect matching for all $v \in H$ [78]. A hypomatchable graph on n vertices is denoted by F_n. In the case in which $\mathcal{H} = \{K_2, F_1, \dots, F_k\}$, the $\mathcal{H}$-Packing problem is polynomially solvable [120].

A k-piece is a connected graph with maximum degree k [75]. When $\mathcal{H}$ is the family of all k-pieces, the $\mathcal{H}$-packing problem can be solved efficiently [72].

On the Parameterized Complexity

The H-Packing problem has been extensively studied in the area of parameterized complexity, mainly when the family of graphs $\mathcal{H}$ contains a prescribed graph H. Unless specified otherwise, in all the results we list next, the input parameter k is the size of the solution. Let us start the discussion with results that focus on prescribed graphs H.

Fellows et al. [54] give an $O(2^{2k \log k + 1.86k} n^2)$ algorithm for the K_3-packing problem. This algorithm combines a search tree with a greedy algorithm. The authors also reduce the K_3-Packing problem to an $O(k^3)$ kernel by using a rule based on crown decompositions [28]. This kernel result was later on improved to $O(k^2)$ by H. Moser [114].

Prieto and Sloper [123] give an $O(k^2)$ kernel when H is a star with s leaves ($H = S_{s+1}$). Basically, the kernelization algorithm consists of obtaining a maximal collection of disjoint S_{s+1} and reducing the vertices that are not in that collection with reduction rules based on crown decompositions. In the special case of $s = 2$, the authors obtain a kernel with $15k$ vertices and an $O(2^{5.301k}k^{2.5} + n^3)$ algorithm. The FPT-algorithm is based on running a brute-force algorithm on the kernel but instead of searching for all possible selections of k S_3's, the authors search only for the *centers* of the stars. The kernel size of the

S_3-Packing (or equivalently P_3-Packing) was improved to $7k$ by Wang et al. [134]. The improvement comes from analyzing in more detail the vertices which are not in the maximal collection. Again the vertices which are not in the maximal collection are reduced using crown decompositions-based reduction rules.

Fernau and Raible [58] consider the P_3-Packing problem as well. In this work, the authors give an $O^*(2.482^{3k})$ algorithm for this problem by using a WIN-WIN approach. The basic idea is to construct a solution of size j by using some vertices of a solution of size $j-1$. In addition, the algorithm interleaves kernelization with the search of that solution.

Chen et al. [24] transform kernelization results for packing problem to their dual covering problems (not defined here). With that dual notion and with general crown decompositions, they obtain a kernel reduction for the T_r-Packing problem, where T_r is a tree with r edges. For the particular case of $r = 2$ (i.e., the P_3-Packing problem) the authors obtain a kernel with $6k$ vertices. This result improves the $7k$ kernel obtained by Wang et al. [134].

We now switch our attention to results concerning an arbitrary graph H. The first FPT-algorithm for the H-Packing problem was achieved by Fellows et al. [54]. This algorithm generalizes the search tree algorithm for the K_3-Packing problem (obtained in the same article) and runs in $O(2^{k|H|\log k+2k|H|\log|H|}n^H)$. Subsequently by using color-coding and dynamic programming, Fellows et al. [53] dramatically improved this running time to $O^*(2^{10rk})$. Later on, Kneis et al. [90] introduced a new technique called *divide and color* which improves the running time to $O^*(2^{4rk})$.

Regarding kernelization, the H-Packing problem was first reduced to an $O(r^r k^r)$ kernel by Fellows et al. [54]. Even though reduction rules based on crown decompositions have been very successful at obtaining kernels for packing problems for small Hs, it seems not to be the case for arbitrary graphs H. Fellows et al. [54] use a different technique which basically consists on bounding the number of subgraphs that contain a specific set of vertices. This technique is called *sunflower kernelization* [35].

Moser [114] achieved the current smallest kernel for the H-Packing problem. He reduces this problem to an $O(r^r k^{r-1})$ kernel. This kernelization algorithm is similar to the one in [54] in the sense that the input instance is reduced by bounding the number of subgraphs in G that contain a specific set of vertices. However, there is one additional step in this algorithm which allows to reduce the exponent from r to $r-1$. Roughly speaking, this extra step consists on computing a maximal collection of subgraphs that overlaps in at most $r-1$ vertices and reducing the vertices that are not in that maximal collection using a maximum matching reduction (which resembles reduction rules based on crown decompositions).

There has recently been a growing interest on proving lower-bounds on the kernel sizes of parameterized problems [23, 15, 95, 36, 55, 17]. Concerning packing problems, Dell and Marx [35] have shown that the K_r-Packing problem does not admit a problem kernel with $O(k^{r-1-\epsilon})$ vertices for any $\epsilon > 0$ under certain complexity assumption (unless coNP $\subseteq$ NP/poly). This leads to the question whether the current $O(k^{r-1})$ kernel for the H-Packing problem is a tight bound for an arbitrary graph H or not, which remains open.

Regarding family of graph classes $\mathcal{H}$, Jansen and Marx [85] give a characterization of the family $\mathcal{H}$ such that the $\mathcal{H}$-Packing problem admits a polynomial kernel when parameterized by $k|V(H)|$ where H is in $\mathcal{H}$. The authors show that if a hereditary class $\mathcal{H}$ has the property that every component of every graph in $\mathcal{H}$ either has bounded size or is a bipartite graph with one of the sides having bounded size, then the $\mathcal{H}$-Packing problem admits a polynomial kernel, and has no polynomial kernel otherwise, unless the polynomial hierarchy collapses. Furthermore, if $\mathcal{H}$ does not have this property, then $\mathcal{H}$-Packing is either $WK[1]$-hard, $W[1]$-hard, or Long Path-hard, giving evidence that it does not admit polynomial Turing kernels either. Bodlaender et al. [17] have obtained lower bounds for the problem of finding disjoint cycles and paths.

2.4.2 Results on the Edge-$\mathcal{H}$-Packing Problem

We now consider the Edge-$\mathcal{H}$-Packing problem. Once again we focus on the classical complexity of this problem, and after that we move to its parameterized complexity. Unfortunately, the Edge-$\mathcal{H}$-Packing problem has not been studied as extensively as its vertex-version.

As with the vertex-version, the Edge-H-Packing problem is NP-complete for an arbitrary connected graph H that has at least three edges [30, 39]. When H has at most two edges, the Edge-H-Packing problem is solved in polynomial time [105, 131].

This problem is still NP-complete when G is an arbitrary connected graph or a planar graph and H is a star [47], a path [47], a tree [30], a cycle [83], and an acyclic graph [30, 39, 74] on at least three edges.

Heath and Vergara [73] obtain several interesting results for this problem. They show that the problem is polynomial solvable if G is a tree, and in linear time if H is a star or a path. If G is an outerplanar graph and H is either a cycle or a star with s leaves, the problem can be solved in linear time.

The Edge-K_3-Packing problem is solvable in polynomial time if the maximum degree of G is at most 4, and if G has maximum degree at least 5 then is NP-hard for planar graphs [18].

Unfortunately, the landscape of the complexity for this problem when the family of graphs $\mathcal{H}$ has more than one graph seems unexplored. Let us now discuss the results concerning the parameterized complexity of this problem.

On the Parameterized Complexity

Mathieson et al. reduce the Edge-K_3-Packing problem to a kernel bounded by $4k$. This appears to be the only result considering prescribed graphs H for the edge-version of the packing problem.

Fellows et al. [54] obtain a kernelization algorithm for the Edge-H-Packing problem, for an arbitrary graph H on r edges. This algorithm works in the same way as with the vertex-version of the problem and achieves an $O(r^r k^r)$ kernel. In addition, the authors show an FPT-algorithm based on color-coding and dynamic programming with $O^*(2^{10rk})$ running time. Currently, this problem can be solved in $O^*(2^{4rk})$ [90].

2.4.3 Results on the r-Set Packing Problem

In this section, we present a survey on the results to the r-Set Packing problem. Some of these algorithms solve the more general weighted version of the r-Set Packing problem which considers that every set in $\mathcal{S}$ has associated a weight (a real number) and the question is whether $\mathcal{S}$ admits or not a $(k, r, 0)$-set packing of maximum weight.

The r-Set Packing problem was shown to be NP-complete, for $r \geq 3$, among the famous Karp's 21 NP-complete problems [87]. The known parameterized algorithms for the r-Set Packing have used either the technique of greedy localization [86], the technique of color-coding [3] plus dynamic programming [25, 53, 92], color-coding plus the divide-and-conquer method [25, 90], the narrow sieves technique [13, 14], and the combinatorial representative sets technique [60]. Being the last three, the ones that led to the current fastest algorithms for the r-Set Packing and other combinatorial problems.

The history of the FPT-algorithms for the r-Set Packing problem for the particular case of $r = 3$ seems to start with [86]. Jia et al. [86] give an $O((5.7k)^k n)$ for the 3-Set Packing problem using a greedy localization algorithm. The fastest algorithms for the parameterized versions of the 3-Set Packing problems (and other graph problems as well) have been obtained using color-coding, which was introduced by Alon et al. [3]. The basic idea of this technique is to randomly color the elements and expect that the sets on a packing contains elements with different colors. Usually, the running time of algorithms obtained via color-coding are of the form $O^*(c^k)$ for some constant c. Note the big improvement on FPT-algorithms with running time $O^*(k^k)$. However, this constant c can be very large. De-randomizing these algorithms results on algorithms with running time of the same form but with even larger constants [90].

Koutis [92] proposed a randomized algorithm for the 3-Set Packing problem that takes time $O^*(10.88^{3k})$ and a deterministic algorithm that runs in $O^*(3200^{3k})$. Kneis et al. [90] improve this running time with a deterministic algorithm of time $O^*(16^{3k})$ using randomized divide and conquer (in combination with color-coding). Using divide and conquer as well, Chen et al. [25] provide a randomized algorithm that solves the 3-Set Packing problem in $O^*(2.52^{3k})$ whose deterministic algorithm is of time complexity $O^*(12.83^k)$. Liu et al. [101] gave a deterministic algorithm of time $O^*(4.613^{3k})$ based on greedy localization and color coding. Wang and Feng [133] further improve this running time. They provide a randomized and deterministic algorithm both with $O^*(3.52^{3k})$. The current best algorithm for the 3-Set Packing problem runs in $O^*(8.097^k)$ and it was obtained by Zehavi [141].

We now list the results on the general case of the r-Set Packing problem, for $r \geq 3$. Chen et al. [21] provided an $O^*((rk)^{O(rk)})$ deterministic algorithm for this problem. Later on, Downey, Fellows, and Koblitz [41] presented an $O^*((rk)^{O(rk)})$ deterministic algorithm for the weigthed r-Set Packing problem. Subsequently, Fellows et al. [53] obtained an $O^*(2^{O(rk)})$ deterministic algorithm for the weighted version as well. Koutis [92] provided both deterministic and randomized algorithm for the r-Set Packing problems with running times $O^*(2^{O(rk)})$ and $O^*(10.874^{rk})$, respectively. Subsequently, Chen et al. [27] presented an $O^*(5.44^{rk})$ deterministic algorithm for the r-Set Packing problem. Chen et al. [22] give deterministic and randomized algorithms for the weighted r-Set Packing problem with running times $O^*(4^{rk+o(rk)})$ and $O^*(4^{(r-1)k+o(rk)})$, respectively. Chen et al., [20] designed a deterministic algorithm based on *universal sets* with improved running times of $O^*(4^{(r-0.5)k+o(rk)})$ for the weighted r-Set Packing. Koutis [93] obtained a randomized algorithm for the r-Set Packing problem which runs in $O^*(2^{rk})$. Using representative sets, Zehavi [140] provided a deterministic algorithm for the weighted r-Set Packing problem whose running time is $O^*((0.563 \cdot 2.851^r)^k)$. Later on, Goyal et al., [69] designed an improved deterministic algorithm for this problem combining representative sets and dynamic programming. This improved algorithm runs in time $O^*(2.851^{(r-0.5501)k})$. Table 2.1 summarizes these results.

On the other hand, kernelization algorithms for the r-Set Packing problem have been obtained in [53, 1, 140]. Fellows et al. [53] reduce this problem to a kernel with both $O(r^r k^r)$ elements and sets in the reduced universe and collection of sets. The underlying idea of this reduction is to bound the number of sets in $\mathcal{S}$ that contain a specific subset of elements of $\mathcal{U}$. Further kernel size improvement to $O(e^r r k^r)$ (on the number of sets) was achieved by Zehavi [140]. Dell and Marx [35] showed that the r-Set Packing does not admit a problem kernel with $O(k^{r-\epsilon})$ sets for any $\epsilon > 0$ unless coNP$\subseteq$NP/poly. Interestingly, the universe can be further reduced as Abu-Khzam [1] gives a kernelization algorithm that returns a kernel with $O(r^r k^{r-1})$ elements in the universe.

Reference	Problem	Algorithm	Running Time
Chen et al. [21]	r-Set Packing	Deterministic	$O^*((rk)^{O(rk)})$
Downey et al. [41]	Weighted-r-Set Packing	Deterministic	$O^*((rk)^{O(rk)})$
Fellows et al. [53]	Weighted-r-Set Packing	Deterministic	$O^*(2^{O(rk)})$
Koutis [92]	r-Set Packing	Deterministic	$O^*(2^{O(rk)})$
Koutis [92]	r-Set Packing	Randomized	$O^*(10.874^{rk})$
Chen et al. [27]	r-Set Packing	Deterministic	$O^*(5.44^{rk})$
Chen et al. [22]	Weighted-r-Set Packing	Deterministic	$O^*(4^{rk+o(rk)})$
Chen et al. [22]	Weighted-r-Set Packing	Randomized	$O^*(4^{(r-1)k+o(rk)})$
Chen et al. [20]	Weighted-r-Set Packing	Deterministic	$O^*(4^{(r-0.5)k+o(rk)})$
Koutis [93]	r-Set Packing	Randomized	$O^*(2^{rk})$
Zehavi [140]	Weighted-r-Set Packing	Deterministic	$O^*((0.563 \cdot 2.851^r)^k)$
Goyal et al. [69]	Weighted-r-Set Packing	Deterministic	$O^*(2.851^{(r-0.5501)k})$

Table 2.1: Algorithms for the r-Set Packing problem and its weighted version

Results on the r-Set Packing with t-Membership Problem

Recently, the r-Set packing with t-Membership problem received attention by Gabizon et al. [64]. Using representative sets, the authors obtain a deterministic algorithm that runs in $O(2^{O(rk\frac{\log t}{t})}|\mathcal{S}|n^{O(1)})$. Given that representative sets are useful to guarantee disjoint sets, the authors extend the concept of representative sets to multisets. A multiset is a set where an element could occur more than once. The idea behind the generalization of representative sets to consider multisets is as follows. If $\mathcal{P}$ is a family of multisets, with all sets having the same size k (counting duplicates), then a subfamily $\mathcal{P}' \subseteq \mathcal{P}$ *k-represents* $\mathcal{P}$, if for every multiset A of size r, whenever there exists a $P \in \mathcal{P}$ such that no element appears more than t times in $P + A$, then there also exists a set $P' \in \mathcal{P}'$ such that no element appears more than t times in $P' + A$. The authors show that every family $\mathcal{P}$ of multisets has a relatively small k-representative family $\mathcal{P}'$ and provide an efficient algorithm to compute such family.

The connection between the r-Set Packing with t-Membership problem and this generalized notion of representative sets is as follows. Basically, a (k, r, t)-set membership $\mathcal{K} = \{S_1, S_2, \ldots, S_k\}$ is viewed as a multiset $P = S_1 + S_2 + \cdots + S_k$. This multiset has size at most $k\dot{r}$ and every element in $\mathcal{U}$ appears at most t times in P. The family $\mathcal{P}$ is therefore the collection of all (k, r, t)-set membership of $\mathcal{S}$. The algorithm designed by Gabizon et al. [64] solves the more general weighted version of the r-Set packing with t-Membership problem.

Observe that this result could imply that the r-Set Packing with t-Overlap is more difficult to solve than the r-Set Packing with t-Membership problem. For the t-Overlap version, every element of $\mathcal{U}$ could appear at most k times in a solution (a multiset). However, not every multiset that contains an element at most k times will be a solution for the r-Set Packing with t-Overlap problem.

2.4.4 Other Related Problems

In this section, we briefly mention other graph problems that have some relationship to our packing with overlap problems. The *cluster editing problem* consists of modifying a graph G by adding or deleting edges such that the modified graph is composed of a vertex-disjoint union of cliques. Some works have considered overlap in the cluster editing problem [34, 52]. Fellows et al. [52] allow each vertex of the modified graph to be contained in at most t maximal cliques. This version is related to the $\mathcal{H}$-packing with t-Membership, when $\mathcal{H}$ is the family of cliques. The *graph clustering problem* is also related to our problems. However, most of the approaches to graph clustering seek disjoint clusters [50].

The problem of finding one community of size at least r in a given network is also related to our packing problems with overlap. The most studied community models are cliques and some relaxations of cliques. Parameterized complexity results for this problem can be found in [71, 91, 128]. Overlap has not yet been considered under this setting.

As indicated in Chapter 1, the $\mathcal{H}$-Packing with t-Overlap and t-Membership attempt to formalize the community discovering problem. Thus, we refer to the reader to the following comprehensive surveys [61, 137, 122, 116] for approaches to the latter problem.

2.5 Conclusions

In this chapter, we introduced the notation and terminology that will be used during this thesis as well as some basic concepts on parameterized complexity. We also introduced the formal definitions of our parameterized problems. Finally, we investigated the existing literature on the problems that theoretically are most closely related to our problems: the $\mathcal{H}$-Packing (in all its forms) and the r-Set Packing problems. These problems have a long history of results from different methodologies; we focused here only on classical computational complexity and parameterized complexity results. Interestingly enough, overlap has not been considered for those problems in all those years of study (as far as we know). In the following chapters, we aim to start building the landscape of results for our packing problems that allow overlaps.

Chapter 3

On the Complexity of the Packing Problems with Overlap

We have proposed the $\mathcal{H}$-Packing with t-Overlap and t-Membership problems as abstractions for the community discovering problem. Also, we learned that these problems generalize the well-studied H-packing problem. Generally speaking, all these problems are in NP, and some of them are NP-complete. This follows since every instance of the $\mathcal{H}$-Packing problem, which is NP-complete [88], is mapped to an instance of these problems by setting $t = 0$ and $t = 1$, respectively.

However strictly speaking, we have introduced whole families of problems. Each $\mathcal{H}$ and each t specify a particular $\mathcal{H}$-Packing with t-Overlap (respectively t-Membership problem). Notice that the constant r is a property of $\mathcal{H}$. Thus, in this chapter we would like to investigate the computational complexity of our problems for other values of t besides zero and one.

The content of this chapter is as follows. In Section 3.1, we focus on the complexity of the t-Membership problems. In particular, we show that the $\mathcal{H}$-Packing with t-Membership problem is NP-Complete for all values of $t \geq 1$ when $\mathcal{H} = \{H\}$ and H is an arbitrary connected graph with at least three vertices, but polynomial-time solvable for smaller graphs. On the other hand, in Section 3.2 we prove that for any $t \geq 0$, there always exists a connected graph H such that the H-Packing problem with t-Overlap is NP-Complete.

3.1 Complexity of the Packing Problems with Membership

Let us present first one concrete $\mathcal{H}$-Packing with t-Membership problem that is hard for all possible values of t.

Theorem 1. *For all $t \geq 1$, the P_3-Packing with t-Membership problem is NP-complete.*

Proof. P_3-Packing with t-Membership in NP is easy to see for any fixed t. Moreover, the assertion is well-known for $t = 1$.

We are now showing a reduction from P_3-Packing with t-Membership to P_3-Packing with $(t+1)$-Membership, which proves the claim by induction.

Let (G, k) be an instance of the P_3-Packing with t-Membership problem, where $G = (V, E)$, with $V = \{v_0, \ldots, v_{n-1}\}$. Without loss of generality, assume that n is divisible by $t+1$; otherwise, simply add some isolated vertices to obtain an equivalent instance. Let $U = \{u_0, \ldots, u_{(2n)/(t+1)-1}\}$ be a set of new vertices, i.e., $V \cap U = \emptyset$. Create a graph $G' = (V', E')$ as follows:

- $V' = V \cup U$;
- $E' = E \cup \{v_{i+j}u_{2i/(t+1)}, u_{2i/(t+1)}u_{2i/(t+1)+1} \mid 0 \leq i < n, i \pmod{t+1} \equiv 0, 0 \leq j \leq t\}$.

(See Figure 3.1)

We now prove the following claim: G has a $(3,t)$-P_3-membership of size at least k if and only if G' has a $(3,t+1)$-P_3-membership of size at least $n+k$, where $n = |V|$.

If P is a $(3,t)$-P_3-membership for G, then $P' = P \cup X$ is a $(3,t+1)$-P_3-membership for G', where

$$X = \{v_{i+j}u_{2i/(t+1)}u_{2i/(t+1)+1} \mid 0 \leq i < n, i \pmod{t+1} \equiv 0, 0 \leq j \leq t\}$$

Clearly, $|P'| = |P| + |X| = |P| + n$.

Conversely, let P' be a $(3,t+1)$-P_3-membership for $G' = (V', E')$. Let us denote by $oc_{P'} : V' \to \{0, \ldots, t+1\}$ the *occupancy* of v, meaning that

$$oc_{P'}(v) = \{p \in P' \mid v \text{ occurs on path } p\}.$$

Clearly, $oc_{P'}(u_{2i/(t+1)}) \geq oc_{P'}(u_{2i/(t+1)+1})$ for all $0 \leq i < n$, $i \pmod{t+1} \equiv 0$, $0 \leq j \leq t$. If, for some i, $oc_{P'}(u_{2i/(t+1)}) > oc_{P'}(u_{2i/(t+1)+1})$, then there must be some $p \in P'$ that looks like $p = v_{i+j}u_{2i/(t+1)}v_{i+\ell}$ or $p = v_{i+j}v_{i+\ell}u_{2i/(t+1)}$ or $p = v_{i+\ell}v_{i+j}u_{2i/(t+1)}$ with $j < \ell$, or like $p = v_{i'+j'}v_{i+\ell}u_{2i/(t+1)}$ for some i', j' such that $i' \neq i$.

In either case, replace p by $v_{i+\ell}u_{2i/(t+1)}u_{2i/(t+1)+1}$. Doing this consecutively wherever possible, we end up with a P' whose occupancy function satisfies $oc_{P'}(u_{2i/(t+1)}) = oc_{P'}(u_{2i/(t+1)+1})$ for all i; obviously this new $(3,t+1)$-P_3-membership is not smaller than the original one.

Now, P' contains only two types of paths: those (collected in P_G) completely consisting of vertices from G, and those (collected in P_X) containing exactly one vertex from G (and two from U). We will maintain this invariant in the following modifications. Also, we will maintain that $|P'| \geq n+k$ and that $oc_{P'}(u_{2i/(t+1)}) = oc_{P'}(u_{2i/(t+1)+1})$ for all i. We are now describing a loop in which we gradually modify P'.

(i) It might be that P' is not maximal at this stage, meaning that more P_3's can be added to P' without destroying the property that P' forms a $(3,t+1)$-P_3-membership. So, we first greedily add paths to P' to ensure maximality. For simplicity, we keep the names P_X and P_G for the possibly new sets.

Consider some vertex v with $oc_{P_G}(v) < t+1$. As P' is maximal, $oc_{P_X}(v) = 1$. But, if $oc_{P_G}(v) = t+1$, then $oc_{P_X}(v) = 0$, as P' is a $(3,t+1)$-P_3-membership.

If there is no $v \in V$ with $oc_{P_X}(v) = 0$, then we quit this loop. Otherwise, there is some $v \in V$ with $oc_{P_X}(v) = 0$. By maximality of P', this means that $oc_{P_G}(v) = t+1$. Pick some $p_v \in P_G$ that contains v.

Assume that $v = v_{i+j}$ for some i, j with $i \pmod{t+1} \equiv 0$ and $0 \le j \le t$. Delete p_v from P' (and hence from P_G) and replace it by $v_{i+j}u_{2i/(t+1)}u_{2i/(t+1)+1}$. As (previously) $oc_{P_X}(v) = 0$, the replacement was not contained in P' before. This new $(3, t+1)$-P_3-membership of G', called again P' for the sake of simplicity, is hence not smaller than the previous one, and we can recurse, re-starting at (i); the new P' decomposes again into P_G and into P_X.

Finally, we arrive at a $(3, t+1)$-P_3-membership P' of G' such that for each vertex $v \in V$, $oc_{P_X}(v) > 0$, i.e., $oc_{P_X}(v) = 1$. As clearly $t + 1 \ge oc_{P'}(v) = oc_{P_X}(v) + oc_{P_G}(v)$, this implies that P_G is a $(3, t)$-P_3-membership of G.

(See Figure 3.2)

Hence, P' decomposes into a $(3, t)$-P_3-membership P_G of G and a set of paths P_X containing exactly one vertex from G. If $|P'| \ge n + k$, then $|P_G| = |P'| - |P_X| \ge n + k - |P_X| \ge k$ as required. □

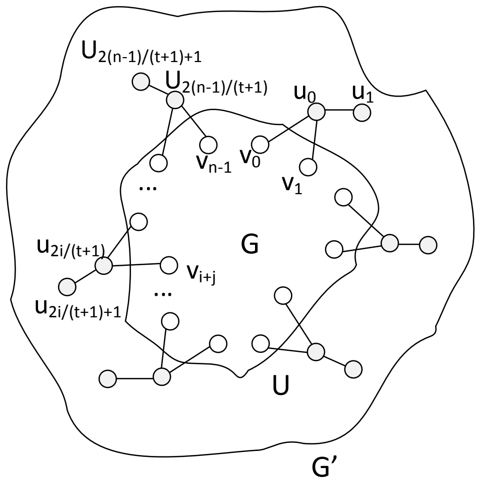

Figure 3.1: An illustration of the construction of G' from G and U in proof of Theorem 1. In this example $t = 1$. Edges of G were omitted for clarity.

We can construct a similar reduction for the H-Packing with t-Membership problem for each H containing at least three vertices based on results of Kirkpatrick and Hell [88].

Theorem 2. *Let H be a connected graph with at least three vertices. For all $t \ge 1$, the H-Packing with t-Membership problem is NP-complete.*

Proof. (Sketch) We add to G a collection U of $\frac{n}{t+1}$ vertex-disjoint H^*s where $H^* = H[V(H)\backslash u]$ for some vertex $u \in H$. For example, if H is a K_5, H^* is a K_4, or if H is a S_5 (star of five vertices), H^* will be either a S_4 or an independent set of 4 vertices. After that we "attach" each H^* to $t + 1$ vertices in G to form $t + 1$ Hs in G'. As with the P_3's construction, those $t + 1$ vertices would be connected only to one H^*. In this way, we can form n H's of G' (collected in X) where each vertex in each H^* is contained in

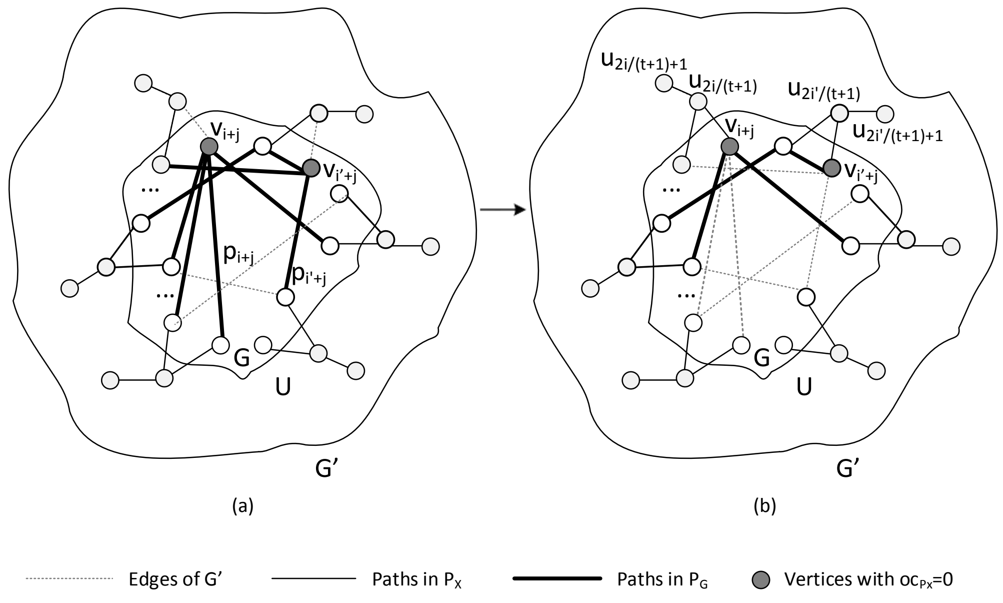

Figure 3.2: A $(3, t+1)$-P_3-membership P' of G' (for $t = 1$) in proof of Theorem 1. Paths in P_X and P_G are indicated in black and bold black, respectively. Edges of G' are indicated with dashed gray lines. In this example, there are two vertices v_{i+j} and $v_{i'+j}$ with $oc_{P_X}(v_{i+j}) = 0$ and $oc_{P_X}(v_{i'+j}) = 0$. Arbitrarily, the paths p_{i+j} and $p_{i'+j}$ are removed from P' and replaced by $v_{i+j}u_{2i/(t+1)}u_{2i/(t+1)+1}$ and $v_{i'+j}u_{2i'/(t+1)}u_{2i'/(t+1)+1}$, respectively (see (b)). Observe that P_G is now a $(3, t)$-P_3-membership of G.

$t + 1$ H's of X and each vertex of G is contained in only one H of X. Thus, a (k, r, t)-H-membership P of G can be altered into an $(n + k, r, t + 1)$-H-membership P' of G by only adding to P the collection X. Conversely, let P' be a $(r, t + 1)$-H-membership for $G' = (V', E')$ of size at least $n + k$. We gradually modify P' in two separate steps, in both of them we always preserve its size and the property that is an $(r, t + 1)$-H-membership for G'. First, we can always *alter* P' by replacing some H's to end up with P' containing only two types of H's: those completely consisting of vertices from G (P_G), and those containing exactly one vertex from G and one H^* from U (P_X). After that, we can gradually alter P' once more to force that every vertex in G is contained in exactly one H of P_X. This would imply that $P_G = P' \backslash P_X$ would be an (r, t)-H-membership G of size $|P_G| = |P'| - |P_X| \geq n + k - |P_X| \geq k$. □

In particular, this applies to packing complete graphs, with $K_3 = C_3$ being the simplest example. Since in this case, the vertex-induced subgraph version coincides with the subgraph version, we can conclude:

Corollary 1. *For each $r \geq 3$, there exists a connected graph H on r vertices such that, for all $t \geq 1$, the Induced-H-Packing with t-Membership problem is NP-complete.*

As also the r-Set Packing problem is known to be NP-complete for any $r \geq 3$, we can adapt the construction of Theorem 1 to obtain the following result:

Theorem 3. *For all $t \geq 1$ and $r \geq 3$, the r-Set Packing with t-Membership problem is NP-complete.*

3.1.1 A Polynomial Solvable Case for the Vertex Membership Problem

On the positive side, we next show that the P_2-Packing with t-Membership problem can be solved in polynomial time.

A $(k, 2, t)$-P_2-membership P of a graph G is a subset of at least k edges of $E(G)$ such that every vertex of G is contained in at most t edges of P. Let us denote as G^* the subgraph of G induced by P, i.e., $G^* = (V_{G^*}, P)$ where $V_{G^*} = V(P)$ (the end-points of the edges in P).

Lemma 2. *The graph G^* has maximum degree t.*

Proof. Assume otherwise that there is a vertex $v \in V_{G^*}$ with $deg_{G^*}(v) > t$. This implies that there are more than t edges of G^* incident to v. Since $E(G^*) = P$, this is equivalent to say that v is contained in more than t edges of P, a contradiction since P is a (k, t)-P_2-Membership. □

Let $b : V(G) \to \mathbb{N}$ be a degree constraint for every vertex. The problem of finding a subgraph G^* of G such that each vertex $v \in V(G^*)$ has degree at most $b(v)$ in G^*, i.e., $deg_{G^*}(v) \leq b(v)$ and the number of edges in G^* is maximized is known as the *degree-constrained subgraph problem* [129]. Let us refer to G^* as a *degree-constrained subgraph.* Y. Shiloach [129] constructed a graph G' from G as follows.

1. For every vertex $v \in V(G)$ add $deg_G(v)$ vertices in G' (called *copies of* v)
2. An edge (u, v) in G will be replaced in G' by an edge composed of a copy of u and a copy of v. Each copy participates in only one edge. That is, the edge $(u, v) \in E(G)$ is *represented* by an edge (u_i, v_j) where u_i is a copy of u and v_j is a copy of v_j for some $1 \leq i \leq deg_G(u)$ and $1 \leq j \leq deg_G(j)$.
3. Add t vertices in G' for each vertex v in G. These t vertices are completely connected to the $deg_G(v)$ copies of v. We will identify these vertices as $\bar{v}_1 \ldots \bar{v}_t$, and we will call them the *t-vertices* of v.

Figure 3.3 illustrates an example of this construction.

Theorem 4. *G has a degree-constrained subgraph with k edges if and only if G' has a maximum matching of size at least $|E(G)| + k$ [129].*

By Lemma 2, we can find a $(k, 2, t)$-P_2-membership P of G, by solving the degree-constrained subgraph problem with $b(v) = t$, for all $v \in V(G)$. The maximum matching problem can be solved in general graphs in $O(\sqrt{|V|}|E|)$ running time by running the Micali-Vazirani algorithm [108].

In our case, we need to determine $|V(G')|$ and $|E(G')|$, where G' is the graph constructed from G. From the construction of G', we know that the number of vertices in G' is given by:

- For every vertex $v \in V(G)$, we added $deg_G(v)$ vertices in G' (called *copies of* v). This implies that there are $\sum_{v \in V(G)} deg_G(v) = 2|E(G)|$ copies.

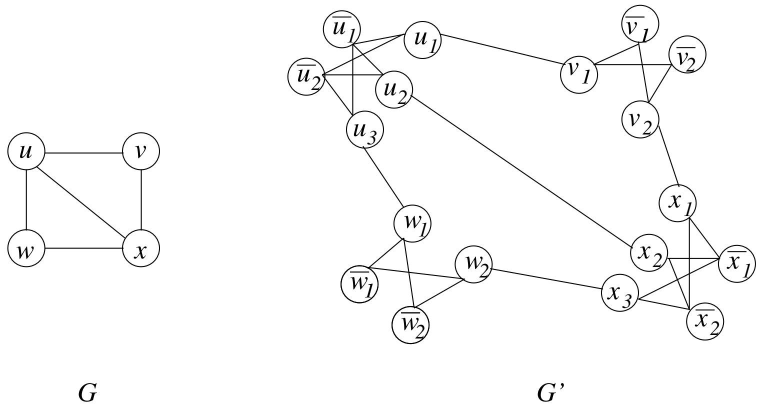

Figure 3.3: An illustration of the construction of G' from G as in [129]. In this example $t = 2$.

- In addition, we added t vertices in G' for each vertex v in G (called the t-vertices). This implies that there are $t|V(G)|$ t-vertices.

In total, we have $|V(G')| = 2|E(G)| + t|V(G)|$.

Similarly the number of edges in G' corresponds to:

- Each edge $(u, v) \in E(G)$ is *represented* by an edge (u_i, v_j) where u_i is a copy of u and v_j is a copy of v_j for some $1 \leq i \leq deg_G(u)$ and $1 \leq j \leq deg_G(j)$. This implies that there are $|E(G)|$ of these edges in G'
- The t-vertices are completely connected to the $deg_G(v)$ copies of v. This implies that there are $\sum_{v \in V(G)} tdeg_G(v)$= $2t|E(G)|$ of these edges.

In total, we have $|E(G')| = 2t|E(G)| + t|E(G)|$=$3t|E(G)|$.

Thus, the maximum matching can be solved in $O(\sqrt{2|E(G)| + t|V(G)|}3t|E(G)|)$ by running Micali and Vazirani's algorithm [108]. Hence, we can state:

Corollary 2. *Let $t \geq 1$. P_2-Packing with t-Membership can be solved in time that is polynomial both in the size of the input graph G and in t.*

We can summarize Theorem 2 and Corollary 2 by stating the following dichotomy result that is analogous to the classical one due to Kirkpatrick and Hell [88].

Theorem 5. *(Dichotomy Theorem) Let $t \geq 1$. Assuming that P is not equal to NP, then the H-Packing with t-Membership problem can be solved in polynomial time if and only if $|V(H)| \leq 2$.*

3.1.2 Complexity of the Edge Membership

We now turn to the edge version of the $\mathcal{H}$-Packing with t-Membership problem. Notice that it is known that for each connected graph H on at least three edges, the problem of finding an Edge-H-Packing of size k in a graph G is NP-complete, as finally shown by Dor and Tarsi [39].

We only present one concrete hardness result.

Theorem 6. *For any $t \geq 1$, the C_3-Packing with t-Edge Membership problem is NP-complete.*

Proof. Membership in NP is obvious. For the hardness, we proceed again by induction. For $t = 1$, the result was shown by Holyer [83].

Assume that it is known that the C_3-Packing with t-Edge Membership problem is NP-hard for some concrete $t \geq 1$. We are going to show then that the C_3-Packing with $(t+1)$-Edge Membership problem is NP-hard.

Let $G = (V, E)$ and k represent an instance of the C_3-Packing with t-Edge Membership problem. We are going to construct an instance (G', k') of the C_3-Packing with $(t+1)$-Edge Membership problem as follows:

- $G' = (V', E')$, where $V' = V \cup E$ and
$$E' = E \cup \{ve : v \in V, e \in E, v \in e\},$$
- $k' = k + |E|$.

Namely, if P is a $(k, 3, t)$-edge-C_3-membership of G, we can get a $(k', 3, t+1)$-edge-C_3-membership of G' by adding all triangles of the form $\{uv, u, v\}$ to P, where $u, v \in V$ and $uv \in E'$.

Conversely, let P' be a $(k', 3, t+1)$-edge-C_3-membership of G'. Observe that (i) the edges $(u, uv) \in E'$ with $u, v \in V$ and $uv \in E$ can only be made use of in the triangle $\{uv, u, v\}$. Moreover, any $(k', t+1)$-edge-C_3-membership P' that does not use this triangle could be changed in a $(3, t+1)$-edge-C_3-membership with the same number of triangles (or more) by adding this triangle, possibly at the expense of deleting another triangle using the edge uv. Hence, we can assume that all $|E|$ triangles of the form $\{uv, u, v\}$ are in the $(k', 3, t+1)$-edge-C_3-membership of G' that we consider. Every edge of G is used exactly once in this way. Thanks to observation (i), the remaining $k' - |E| = k$ triangles in P' form a (k, t)-edge-C_3-membership of G. $\square$

3.2 Complexity of the Packing Problems with Pairwise Overlap

For the packing problems with pairwise overlap, the complexity landscape is known even less. For instance, we do not know of any concrete graph H such that the corresponding H-Packing with t-Overlap problem is NP-complete for all $t \geq 0$. However, the next theorem explains at least that there are NP-hard H-Packing with t-Overlap problems for each value $t \geq 0$.

Theorem 7. *For any $t \geq 0$, there exists a connected graph H_t such that the H_t-Packing with t-Overlap problem is NP-complete.*

Proof. (Sketch) According to Caprara and Rizzi [18], the problem of packing at least k triangles with zero vertex overlap in a graph of maximum degree four is NP-complete. We reduce from this problem and add, to each vertex v of the triangle packing instance graph G, a $K_{1,5}$, call it S^v that is connected to v by one edge to its center c'. This gives the new graph G'. Likewise, the graph H is a triangle, where we add three $K_{1,5}$'s. Clearly, any triangle packing of G translates into an H-packing of G' of the same size, and vice versa. However, as the only vertices in G' that are of degree six are the centers v' in the S^v that are connected to v via a bridge, these must host the degree six vertices of H. Hence, there is no way of exploiting possible overlaps, as long as $t \leq 6$. For bigger t, we can change the construction by adding bigger stars $K_{1,t-1}$ to each vertex. This finally yields the claim. □

3.3 Conclusions

We begin in this chapter the study of the complexity of the packing problems allowing overlaps. For the H-Packing with t-Membership, we concluded that this problem is NP-complete when H is a connected graph with $|V(H)| \geq 3$. An analogous result is stated for r-Set Packing with t-Membership when $r \geq 3$. For the packing problems with overlap, we showed that there is always at least one graph H for which the H-Packing with t-Overlap problem is NP-complete. Since polynomial time algorithms for the general cases seem unlikely, we will explore other approaches to provide a solution for our problems.

Part II

Kernelization Algorithms

Chapter 4

A Case Study: Crown Decompositions to Obtain Kernels for Packing Cliques with Pairwise Overlap

We concluded in the previous chapter that several of our packing problems allowing overlaps are NP-complete. Our direction is therefore to study these problems from a parameterized point of view. One of our goals is to obtain *kernels*, i.e., reduced instances whose size is bounded by an input parameter (we use as the parameter the size of the solution).

A crown decomposition is a technique that at a high level partitions the vertices of a graph into three sets each one subject to specific conditions [28]. Reduction rules based on crown decompositions have been designed to obtain kernels for several packing problems. For example: packing triangles [106, 54], packing paths with three vertices [123, 134] and packing trees with r edges [24]. Therefore, it is natural to wonder if crown decompositions can also help to obtain kernels for generalized packing problems that allow overlap. We will investigate that question in this chapter.

This chapter is organized as follows. In Section 4.1, we introduce a generalized crown decomposition called the *clique-crown decomposition*. Based on this decomposition in Section 4.2, we design a reduction rule for the specific case of the $\mathcal{H}$-Packing with t-Overlap problem, when $\mathcal{H} = \{K_r\}$. Using that reduction rule, we reduce this problem to a kernel when $t = r - 2$ in Section 4.3. Finally, we briefly discuss in Section 4.4, the limitations to obtain kernels for this problem for any $t \geq 0$.

4.1 The Clique-Crown Decomposition

In this section, we introduce a graph decomposition called the *clique-crown decomposition* which is a generalization of the crown decomposition technique. This technique was introduced by Chor et al. [28], and it has been adapted to obtain kernels for graph packing problems [54, 106, 123, 134, 24].

Definition 5. *A* crown decomposition *(H, C, R) in a graph G is a partitioning of $V(G)$ into three sets H, C, and R that have the following properties:*

1. *$C = C_m \cup C_u$ (the crown) is an independent set in G.*
2. *H (the head) is a separator in G such that there are no edges in G between vertices belonging to C and vertices belonging to R.*
3. *R is the rest of the graph, i.e., $R = V(G) \backslash (C \cup H)$.*
4. *There is a perfect matching between C_m and H.*

Generally, vertices in C_m and in H are part of a desired solution while vertices in C_u can be removed from G.

A crown decomposition can be computed in polynomial time for a graph G given certain conditions.

Lemma 3. *[28] If a graph $G = (V, E)$ has an independent set of vertices $I \subseteq V(G)$ such that $|I| \geq |N(I)|$, then G has a crown decomposition, where $C \subseteq I$ and $H \subseteq N(I)$, that can be found in time $O(|V(G)| + |E(G)|)$, given I.*

In our clique-crown decomposition, we have cliques in both the head H and the crown C. We say that a clique A *is completed* with a clique B, if $D = G[V(A) \cup V(B)]$ is a clique and $|V(D)| = r$. In this way, we require that each clique in H be completed by at least one clique in C.

Definition 6. *A* clique-crown decomposition *(H, C, R) is a partition of G that has the following properties:*

1. *$C = C_m \cup C_u$ (the crown) is a set of cliques in G where each clique has size at most $r - (t + 1)$. Cliques in C are denoted with letters $\alpha, \beta, \ldots$.*
2. *H (the head) is a set of cliques in G where each clique has size at least $t + 1$ and at most $r - 1$. Cliques in H are denoted with letters $\mathbb{A}, \mathbb{B}, \ldots$. The head satisfies the following conditions.*
   i. *Each $\mathbb{A} \in H$ is completed by at least one clique in C. Furthermore, no subgraph $\mathbb{A}'$ of $\mathbb{A}$ is completed by a clique in C.*
   ii. *Each pair of cliques $\mathbb{A}$ and $\mathbb{B}$ in H overlaps in at most t vertices.*
   iii. *Each $\mathbb{A} \in H$ is completed by a clique in R, defined below. In addition, the size of any subgraph $\mathbb{A}'$ of $\mathbb{A}$ that is completed by a clique in R is at most t.*
3. *R is the rest of the graph, i.e., $R = G[V(G) \backslash (V(C) \cup V(H))]$.*
4. *The set of vertices of the cliques in H, $V(H)$, is a separator such that there are no edges in G from C to R.*
5. *There exists an injective function f mapping each clique $\mathbb{A} \in H$ to a distinct clique $\alpha \in C_m$ such that α completes $\mathbb{A}$. In this way, the subgraph $G[V(\mathbb{A}) \cup V(\alpha)]$ (denoted as $\mathbb{A} \cdot \alpha$) is a K_r that we call* a mapped K_r. *We impose the condition that any pair of the mapped K_rs overlaps in at most t vertices.*

The values $r \geq 1$ and $t \geq 0$ in our clique-crown decomposition are considered fixed-constants.

Figure 4.1 shows a clique-crown decomposition of a graph G with $r = 4$ and $t = 1$. Cliques that belong to the head H are $G[\{3, 4\}]$, $G[\{8, 9, 10\}]$, and $G[\{11, 12\}]$. These cliques are highlighted with thicker lines. Cliques in the crown C are $G[\{1, 2\}]$, $G[\{5, 6\}]$, $G[\{7\}]$ and $G[\{13, 14\}]$. The mapped K_rs in this example are $G[\{1, 2, 3, 4\}]$, $G[\{7, 8, 9, 10\}]$, and $G[\{11,12, 13,14\}]$. Note that the clique $G[\{5, 6\}]$ is in C_u.

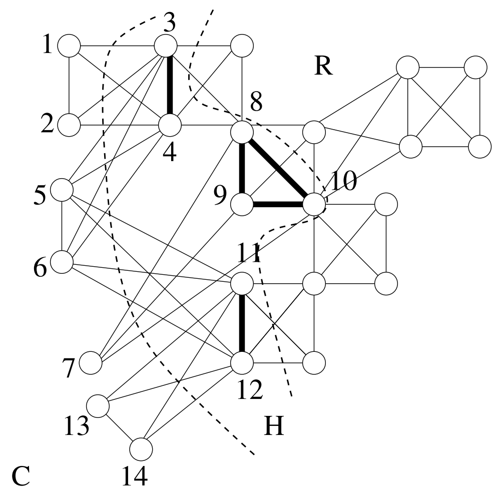

Figure 4.1: Example of a clique-crown decomposition with $r = 4$ and $t = 1$.

4.1.1 A Method for Computing Clique-Crown Decompositions

We next present a method to find a clique-crown decomposition in a graph G given two sets of cliques of G: O and $\mathsf{Cliques}(O)$. These sets of cliques follow the next conditions.

Definition 7. *A* pseudo-crown O *and a* super-head $\mathsf{Cliques}(O)$ *are sets of cliques of G that satisfy the following properties:*

1. *Each clique in O has size at most $r - (t + 1)$, and the cliques in O are pairwise vertex-disjoint.*
2. *Any pair of cliques in $\mathsf{Cliques}(O)$ overlaps in at most t vertices.*
3. *Every clique $\mathbb{A} \in \mathsf{Cliques}(O)$ should be completed by at least one clique in O.*
4. *No subgraph $\mathbb{A}'$ of $\mathbb{A}$ is completed by a clique in O, and the size of any subgraph $\mathbb{A}'$ of $\mathbb{A}$ that is completed by a clique in $G[V(G) \backslash (V(O) \cup V(\mathsf{Cliques}(O)))]$ is at most t. Observe that the size of $\mathbb{A}$ is at least $t + 1$ and at most $r - 1$.*

The following method is a generalization of the method used to compute a crown-decomposition for the Edge-K_3-Packing problem [123].

Lemma 4. *Any graph G with sets of cliques O and $\mathsf{Cliques}(O)$ which follow properties of Definition 7 and $|O| \geq |\mathsf{Cliques}(O)|$ has a clique-crown decomposition (H, C, R) where $O \subseteq C$ and $H \subseteq \mathsf{Cliques}(O)$, that can be found in $O(|V(G)| + |E(G)|)$ time, given O and $\mathsf{Cliques}(O)$.*

Proof. First, we construct a graph G' from G as follows. We initialize $V(G') = V(G)$ and $E(G') = E(G)$. We contract in G' each clique $\alpha \in O$ into a single vertex v_α, and we denote the set of contracted cliques as O_{cont}. After that for each clique $\mathbb{A} \in \mathsf{Cliques}(O)$, we add a vertex $v_{\mathbb{A}}$ to $V(G')$, i.e., *a representative vertex*; we denote as *Rep* the set of all representative vertices. We say that v_α "completes" $v_{\mathbb{A}}$ if the clique α completes the clique $\mathbb{A}$. For every vertex $v_\alpha \in O_{cont}$ that completes $v_{\mathbb{A}}$, add $(v_\alpha, v_{\mathbb{A}})$ to $E(G')$. After that, add to $E(G')$ an edge from $v_{\mathbb{A}}$ to each vertex of $\mathbb{A}$. Finally, remove from $E(G')$ the edges from v_α to $\mathbb{A}$.

We next show that G' has a crown decomposition (H', C', R'). In G', the set of contracted cliques O_{cont} is an independent set. By the construction of G', we know that $N(O_{cont})$ is the set of representative vertices *Rep*. Since we introduced a representative vertex per clique in $\mathsf{Cliques}(O)$, then $|N(O_{cont})| = |\mathsf{Cliques}(O)|$. Thus, since $|O| \geq |\mathsf{Cliques}(O)|$, then $|O_{cont}| \geq |N(O_{cont})|$ in G'. By Lemma 3, G' admits a crown decomposition (H', C', R') that is computed in polynomial time, where $C' \subseteq O_{cont}$ and $H' \subseteq N(O_{cont}) =$ *Rep*.

Now, we use the crown decomposition (H', C', R') of G' to construct the clique-crown decomposition (H, C, R) of G where $H \subseteq \mathsf{Cliques}(O)$.

1. For each vertex $v_\alpha \in C' \subseteq O_{cont}$, add the clique α to C. The size of each clique in C is at most $r - (t + 1)$. This follows because $\alpha \in O$ and each clique in O has size at most $r - (t + 1)$.
2. For each vertex $v_{\mathbb{A}} \in H'$, where $H' \subseteq$ *Rep*, we assign to H the clique that this vertex represents, i.e., $\mathbb{A}$. Since $H \subseteq \mathsf{Cliques}(O)$, then each clique in H has size at least $t + 1$ and at most $r - 1$. Likewise, properties i-iii from Definition 6 follow.
3. $R = G[V(G) \backslash (V(C) \cup V(H))]$.
4. The set of vertices of the cliques in H, $V(H)$, is a separator. This follows since cliques in C complete only cliques on H; thus, vertices in $V(C)$ are only adjacent to vertices in $V(H)$.
5. We make the perfect matching between C' and H' to correspond to the injective function f in the following way. For any matched edge $(v_\alpha, v_{\mathbb{A}})$ complete $\mathbb{A}$ with α. For any pair of mapped K_rs $\mathbb{A} \cdot \alpha$ and $\mathbb{B} \cdot \beta$ completed in this way, $|V(\mathbb{A} \cdot \alpha) \cap V(\mathbb{B} \cdot \beta)| \leq t$. This follows because α and β are vertex-disjoint, and $|V(\mathbb{A}) \cap V(\mathbb{B})| \leq t$ by assumption in the set $\mathsf{Cliques}(O)$.

Thus, if $|O| \geq |\mathsf{Cliques}(O)|$, then G admits a clique-crown decomposition. □

Clearly, the main difficulty of applying this method is to obtain a pseudo-crown O and a super-head $\mathsf{Cliques}(O)$ (Definition 7).

4.2 The Clique-Crown Reduction Rule for the K_r-Packing with t-Overlap Problem

In this section, we present the clique-crown reduction rule for the K_r-Packing with t-Overlap problem which is based on our clique-crown decomposition. Let us first recapitulate the definition of this problem. Recall that r is considered a fixed-constant.

K_r-Packing with t-Overlap problem
Instance: A graph G and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-K_r-*packing* i.e., a set of at least k cliques each one with r vertices $\mathcal{K} = \{S_1, \ldots, S_k\}$ where $|V(S_i) \cap V(S_j)| \leq t$, for any pair S_i, S_j with $i \neq j$?

To design the clique-crown reduction rule, we use an annotated version of the K_r-Packing with t-Overlap problem. In this annotated version, any K_r in the solution overlaps in at most t vertices with any clique from a set $\mathcal{F}$ given as part of the input. Again, let t be fixed in the following definition.

Definition 8. ***Annotated K_r-Packing with t-Overlap problem***

Instance*: A graph G, a set of cliques $\mathcal{F}$ from G each one has size at least $t+1$ and at most $r-1$ and every pair of cliques in $\mathcal{F}$ overlaps in at most t vertices, and a non-negative integer k.*

Parameter*: $k - |\mathcal{F}|$*

Question*: Does G contain an* annotated $(k - |\mathcal{F}|, r, t)$-$K_r$-packing*, i.e, a set of at least $k - |\mathcal{F}|$ K_rs $\mathcal{K} = \{S_1, S_2, ..., S_k\}$ where $|V(S_i) \cap V(S_j)| \leq t$, for any pair S_i, S_j $(i \neq j)$, and $|V(S) \cap V(C)| \leq t$ for any $S \in \mathcal{K}$ and $C \in \mathcal{F}$?*

We now introduce our clique-crown reduction rule.

Reduction Rule 1. The Clique-Crown Reduction. *If G admits a clique-crown decomposition (H, C, R), then reduce G as $G' = G[V(G) \backslash V(C)]$ and $k = k - |H|$. Make H be the set of cliques $\mathcal{F}$ of the annotated K_r-Packing with t-Overlap problem.*

The goal of the clique-crown reduction is to make the mapped K_rs part of the solution and remove unnecessary vertices from G. As part of the correctness of Rule 1, we prove first that the vertices in $V(C_u) \backslash V(C_m)$ are not included in any K_r of the solution.

Lemma 5. *G has a (k, r, t)-K_r-packing if and only if $(G \backslash (V(C_u) \backslash V(C_m))$ has a (k, r, t)-K_r-packing.*

Proof. Cliques in C_u only complete cliques from the set H; otherwise $V(H)$ would not be a separator. However, every clique in H is mapped to a clique in C_m by the injective function. On the other hand, by Definition 6, cliques in H cannot be partitioned in more cliques than $|H|$ that could be completed by cliques in C_u. □

We use the next observation for the proof of correctness of the following lemmas.

Observation 1. *Any clique $\mathbb{A}'$ of $\mathbb{A} \in H$, where $|V(\mathbb{A}')| \geq t+1$, is an induced subgraph of at most one K_r of any solution, since the K_r-Packing with t-Overlap problem allows overlap at most t.*

Lemma 6. *If G admits a clique-crown decomposition (H, C, R), then the set of mapped K_rs is an $(|H|, r, t)$-K_r-packing in G.*

Proof. Follows from Definition 6. □

The input graph G' for the annotated K_r-packing with t-Overlap problem is obtained by removing C from G. Since a clique $\mathbb{A} \in H$ has size at least $t+1$, it can be in at most one K_r of any solution (Observation 1). Hence, we make the set H to be the set of cliques $\mathcal{F}$ in the annotated K_r-packing with t-Overlap problem. In the example of Figure 4.1, $\mathcal{F}$ would be equal to $G[\{3,4\}]$, $G[\{8,9,10\}]$ and $G[\{11,12\}]$.

The correctness of our clique-crown reduction rule is finally shown in the next lemma.

Lemma 7. *Let $G' = G[V(G)\backslash V(C_m \cup C_u)]$. G has a (k, r, t)-K_r-packing if and only if G' has an annotated $(k-|H|, r, t)$-K_r-packing.*

Proof. Assume by contradiction that G admits a clique-crown decomposition (H, C, R) and has a (k, r, t)-K_r-packing, but $(G', H, k-|H|)$ does not have an annotated solution. By Lemma 5, vertices in $V(C_u)\backslash V(C_m)$ are redundant. That is, they do not belong to any K_r in the solution. By Observation 1, every $\mathbb{A} \in H$ is in at most one K_r in the solution. Therefore, we cannot form more than $|H|$ K_rs by completing each clique of H with cliques in R rather than with cliques in C_m. The only case that we could have more than $|H|$ K_rs is if there is a clique $\mathbb{A} \in H$ that has at least two cliques $\mathbb{A}', \mathbb{A}''$ each of size at least $t+1$ that are completed by some clique in R. However, that is not possible by Definition 6.

Assume now that $(G', H, k-|H|)$ has an annotated $(k-|H|, r, t)$-K_r-packing, but G does not have a (k, r, t)-K_r-packing. This would imply that the sets H and C form more than $|H|$ K_rs which is a contradiction by Lemma 6. □

To conclude this section, we show how a solution of the original instance can be obtained using an annotated solution of the reduced graph G'.

Claim 1. *Let $\mathcal{K}'$ be an annotated $(k-|H|, r, t)$-K_r-packing of G' and $\mathcal{H}$ be the set of mapped K_rs in G found with the clique-crown decomposition. The set $\mathcal{K}' \cup \mathcal{H}$ is a (k, r, t)-K_r-packing of G.*

Proof. Since $G' = G[V(G)\backslash V(C)]$, $\mathcal{K}'$ is a $(k-|H|, r, t)$-K_r-packing of G as well. By Lemma 6, the set $\mathcal{H}$ is an $(|H|, r, t)$-K_r-packing in G. Since H becomes the set of cliques $\mathcal{F}$ in the annotated instance, then no K_r of $\mathcal{K}'$ overlaps in more than t vertices with any K_r of $\mathcal{H}$. Therefore, $\mathcal{K}' \cup \mathcal{H}$ is a (k, r, t)-K_r-packing in G. □

In Figure 4.2, the set of mapped cliques of Figure 4.1 are highlighted with thicker lines. An annotated solution for the reduced graph is indicated with dashed lines. We can see how the set of mapped cliques and the annotated solution form a $(5, 4, 1)$-K_4-packing ($k = 5$, $r = 4$, and $t = 1$).

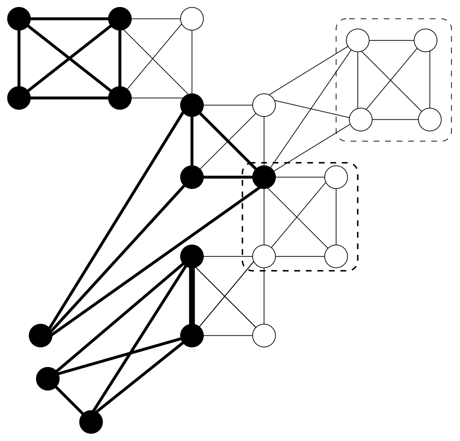

Figure 4.2: An annotated solution for an instance of the K_4-Packing with 1-Overlap problem is indicated with dashed lines ($k = 5$, $r = 4$, and $t = 1$).

4.3 An Application of Clique-Crown Decompositions: A Kernel for the K_r-Packing with $(r-2)$-Overlap Problem

Using the clique-crown reduction rule, we introduce an algorithm to obtain a kernel for the K_r-Packing with $(r-2)$-Overlap problem. That is, when every pair of K_rs in the solution can overlap in at most $r-2$ vertices.

The steps of our kernelization algorithm are outlined in Algorithm 1. We start by reducing our graph G with the following reduction rule.

Reduction Rule 2. *Delete any vertex v and any edge e that are not included in a K_r of G.*

After that, we compute a maximal solution $\mathcal{M}$ for the K_r-packing with $(r-2)$-Overlap problem. If the size of $\mathcal{M}$ is at least k, then the algorithm stops. Otherwise, we will reduce the graph G using our clique-crown reduction rule.

Recall from Lemma 4 that to compute a clique-crown decomposition of G, we need two sets of cliques: a pseudo-crown O and a super-head $\mathsf{Cliques}(O)$ (Definition 7). Next, we show that such sets O and $\mathsf{Cliques}(O)$ are composed of vertices that are outside and inside, respectively, of the maximal solution. Specifically, those sets are computed as in Line 7.

We next introduce a series of lemmas that characterize the sets O and $\mathsf{Cliques}(O)$ defined in Algorithm 1.

Claim 2. *$O = V(G) \backslash V(\mathcal{M})$ is an independent set.*

Proof. Assume by contradiction that there exists an edge (u, v) in $G[O]$. After applying Reduction Rule 2, each edge in the reduced graph is included in at least one K_r; thus, (u, v) belongs to at least one clique S'. S' is not in $\mathcal{M}$; otherwise u, v would not be in O. S' is not in O; otherwise as S' would be disjoint

Algorithm 1 Kernelization Algorithm by Clique-Crown - K_r-Packing with $(r-2)$-Overlap

```
1: Input: A graph $G = (V, E)$ and a non-negative integer $k$.
2: Reduce $G$ by Reduction Rule 2.
3: Greedily, find a maximal $(r, r-2)$-$K_r$-packing $\mathcal{M}$ in $G$.
4: if $|\mathcal{M}| \geq k$ then
5:    $\mathcal{M}$ is a $(k, r, r-2)$-packing and stop
6: else
7:    Let $O$ be $V(G) \backslash V(\mathcal{M})$ and $\mathsf{Cliques}(O)$ be the set of cliques in $\mathcal{M}$ completed by vertices in $O$.
8:    if $|O| \geq |\mathsf{Cliques}(O)|$ then
9:       Apply the clique-crown reduction rule in $G$ (Reduction Rule 1).
10:   end if
11: end if
```

from $\mathcal{M}$, and hence, S' could be added to $\mathcal{M}$, contradicting the maximality of $\mathcal{M}$. Thus, S' should overlap with at least one clique $S \in \mathcal{M}$, for $S \neq S'$.

Since u, v are both in O, the overlap with S is at most $r-2$, i.e., $|V(S) \cap V(S')| = r-2$, but in this case, S' could be added to $\mathcal{M}$ contradicting the maximality of $\mathcal{M}$. □

Claim 3. *Each clique T (on $r-1$ vertices) completed by any $u \in O$ is contained in a clique $S \in \mathcal{M}$.*

Proof. We first show $V(T) \cap V(O) = \emptyset$. Assume for contradiction that there is a vertex $v \in V(T)$ contained in O. However, since $T \cdot u$ forms a K_r this would imply that there is an edge (u, v) in O which is a contradiction since O is an independent set. Thus, $V(T) \subset V(\mathcal{M})$.

We claim that $|V(T) \cap V(S)| = r-1$ for some $S \in \mathcal{M}$, i.e., $V(T) \subset V(S)$. Suppose otherwise that $|V(T) \cap V(S)| < r-1$, for any $S \in \mathcal{M}$. Since $u \in O$, this would imply that $|V(T \cdot u) \cap V(S)| \leq r-2$ for every $S \in \mathcal{M}$, and $T \cdot u$ could be added to $\mathcal{M}$ as the K_r-Packing with $(r-2)$-Overlap problem allows overlap at most $r-2$, contradicting the assumption of maximality of $\mathcal{M}$. □

Claim 4. *Each clique $\mathbb{A} \in \mathsf{Cliques}(O)$ is completed by at least one clique in O and no subgraph $\mathbb{A}'$ of $\mathbb{A}$ is completed by a clique in O. In addition, the size of any subgraph $\mathbb{A}'$ of $\mathbb{A}$ completed by a clique in $G[V(G) \backslash (V(O) \cup V(\mathsf{Cliques}(O)))]$ is at most t. Furthermore, any pair of cliques in $\mathsf{Cliques}(O)$ overlaps in at most t vertices.*

Proof. Assume by contradiction that there is a clique $\mathbb{A}' \subset \mathbb{A}$ of size $s < r-1$ completed by a clique in $G[V(O)]$. This would imply that there is a K_{r-s} in $G[V(O)]$, a contradiction since O is an independent set (Claim 2).

The second and third parts of the claim follows because the size of each clique in $\mathsf{Cliques}(O)$ is $t+1 = r-1$ (Claim 3) and therefore the size of the largest subgraph of $\mathbb{A}$ is at most $r-2 = t$. □

We can see that by Claims 1 and 4, the sets O and $\mathsf{Cliques}(O)$ meet the properties of Definition 7. Thus, the method described in proof of Lemma 4 can be used to compute a clique-crown decomposition in G. Next, we prove that the size of the reduced instance is bounded by a function of the parameter k.

Claim 5. *The set of vertices O completes at most $rk - r$ cliques.*

Proof. By Claim 3, vertices in O only complete K_{r-1}'s contained in cliques in $\mathcal{M}$. There are r K_{r-1}'s in a K_r and at most $k-1$ K_rs in $\mathcal{M}$; thus, there are at most $rk-r$ K_{r-1}'s that can be completed by vertices in O. ☐

Claim 6. $|O| < rk - r$

Proof. In Algorithm 1, if $|O| \geq |\mathsf{Cliques}(O)|$, O is reduced by the clique-crown reduction rule (Rule 1). Since $|\mathsf{Cliques}(O)| < rk - r$, then $|O| < rk - r$, after applying that rule. ☐

Lemma 8. *If $|V(G)| > 2(rk-r)$, then Algorithm 1 will either find a $(k, r, r-2)$-K_r-packing, or it will reduce G.*

Proof. Assume by contradiction that $|V(G)| > 2(rk-r)$, but the algorithm neither finds a $(k, r, r-2)$-K_r-packing nor reduces the graph G. Any vertex $v \in V(G)$ that was not reduced by Rule 2 is in $V(\mathcal{M})$, or it is in $O = V(G) \backslash V(\mathcal{M})$; thus, $|V(G)| = |V(\mathcal{M})| + |O|$.

The size of $\mathcal{M}$ is at most $k-1$; thus, $|V(\mathcal{M})|$ is at most $rk-r$, and by Claim 6 we know that an upper bound for $|O|$ is $rk-r$.

In this way, the size of the instance is at most $2(rk-r)$ which contradicts the assumption that $|V(G)| > 2(rk-r)$. ☐

Theorem 8. *The K_r-packing with $(r-2)$-Overlap problem admits a $2(rk-r)$ kernel which can be found in $O(n^r)$ time, where r is a fixed-constant.*

Proof. By Lemma 8, the reduced instance has size at most $2(rk-r)$. Rule 2 is computed in time $O(n^r)$, which is also the same time to compute the maximal solution $\mathcal{M}$ and $\mathsf{Cliques}(O)$. Lemma 4 shows that the clique-crown decomposition is computed in polynomial time given the set of cliques O and $\mathsf{Cliques}(O)$. The set O corresponds to the independent set $V(G) \backslash V(\mathcal{M})$ (Claim 2), and the set $\mathsf{Cliques}(O)$ is the set of K_{r-1}'s completed by vertices in O. By Claim 3, all these K_{r-1}'s are contained in the K_rs of $\mathcal{M}$. Moreover, in Rule 2, we already compute all K_{r-1}'s that a vertex completes. Thus, the time to obtain $\mathsf{Cliques}(O)$ is $O(n^r)$. ☐

Computing a solution for the K_r-Packing with $(r-2)$-Overlap problem

We have obtained a $2(rk-r)$ kernel for the K_r-Packing with $(r-2)$-Overlap problem. We can now apply a brute-force algorithm on the kernel to find a solution. First, remember that the instance was reduced by the clique-crown reduction rule (Rule 1) and the cliques in the head H becomes the set of cliques $\mathcal{F}$ in the annotated version. Therefore, we are now looking for an annotated solution in the reduced instance G' (Definition 8).

The brute-force algorithm first finds all K_rs $\mathcal{W}$ in the reduced instance G' and after that evaluates if each possible selection of $k - |H|$ K_rs from $\mathcal{W}$ is an annotated solution in G'.

By Claim 1, we can obtain a $(k, r, r-2)$-packing in the original instance combining an annotated solution in G' with the set of mapped K_rs. The time of this brute-force algorithm is $O((2rk-2r)^{rk})$ and the K_r-Packing with $(r-2)$-Overlap problem can be decided in time $O((2rk-2r)^{rk} + n^r)$.

Corollary 3. *The K_r-Packing with $(r-2)$-Overlap problem can be solved in $O((2rk-2r)^{rk} + n^r)$ time.*

4.4 Obtaining Kernels Beyond Overlap of size $r - 2$

In this section, we will briefly discuss the difficulties that we face on reducing the K_r-Packing with t-Overlap problem to a kernel for any overlap value $t \geq 0$ using the clique-crown reduction rule.

Observe first that we showed the correctness of our clique-crown reduction rule for any instance of the K_r-Packing with t-Overlap problem, that is, our rule is general enough to handle any value of $t \geq 0$. However, the main problem is how to actually compute such decomposition for a graph G for any $t \geq 0$. In our method to compute the clique-crown (Lemma 4), the set of cliques $O \supseteq C$ should be composed of vertex-disjoint cliques. However, that is not a necessary condition of C in Definition 6. If we remove such condition, then a perfect matching in the computation of the clique-crown in Lemma 4 will not correspond to the injective function required by the clique-crown (Definition 6).

Let us assume that we are able to generalize our Lemma 4 to state that there exists an Algorithm CliqueCrownCompute that can compute a clique-crown decomposition of G given two sets of cliques of G: $O \supseteq C$ and $\mathsf{Cliques}(O) \supseteq H$, if $|O| \geq |\mathsf{Cliques}(O)|$. For this generalized lemma, O and $\mathsf{Cliques}(O)$ do not need to follow properties of Definition 7 but rather only the properties of the crown and the head (Definition 6), respectively.

Assuming the Algorithm CliqueCrownCompute exists and with some modifications, we could obtain a kernel problem for the K_r-Packing with t-Overlap problem in similar way as in Section 4.3. However, we still do not know how to obtain such algorithm.

4.5 Conclusions

In this chapter, we proposed the clique-crown decomposition technique which is based on the crown decomposition [28]. Using our clique-crown decomposition, we designed a reduction rule for the K_r-Packing with t-Overlap problem. We also gave an actual kernel for the K_r-Packing with $(r-2)$-Overlap problem using that reduction rule. Unfortunately, it is unclear how to overcome the difficulties of extending this kernel for values of t others than $r - 2$. This is perhaps because crown decompositions rules alone may not be viable alternatives to achieve this goal.

Despite of this, crown decompositions still play a key role in our kernelization results. In specific, our second step in the kernelization algorithms for our $\mathcal{H}$-Packing with t-Overlap and t-Membership problems in Chapter 7 consists of reduction rules inspired by crown decompositions.

Chapter 5

From Packing Graphs to Packing Sets: A Kernelization Framework

Previously, we introduced kernelization with a specific case of the $\mathcal{H}$-Packing with t-Overlap problem. By using a generalization of the crown-decomposition [28], we obtained a linear kernel when the family $\mathcal{H}$ consists of a single clique of size r ($\mathcal{H} = \{K_r\}$) and $t = r - 2$. However, we also concluded in that chapter that we need to explore different approaches to kernelization as crown-decompositions by themselves may not be the path to follow to obtain a kernel for the general case.

In this chapter, we introduce a general framework that will allow us to obtain kernelization results for the $\mathcal{H}$-Packing with t-Overlap and t-Membership problems, for an arbitrary family $\mathcal{H}$ of graphs and any non-negative integer value of t. Our framework consists of simply transforming an instance of $\mathcal{H}$-Packing with t-Membership and t-Overlap to an instance of r-Set Packing with t-Membership and t-Overlap, respectively. After this by reducing to a kernel the constructed instances of the set packing problems, we can achieve a reduction to a kernel for the original graph packing problems. The existence of kernelization algorithms for the r-Set Packing with t-Membership and t-Overlap problems is fundamental to these frameworks.

We clarify that in this chapter we are not obtaining kernel sizes per se, but only giving a systematic procedure that will allow us to obtain such kernel sizes in subsequent chapters. The framework described here will be employed in two different chapters in this thesis. Specifically, in Chapter 6, we achieve a kernelization algorithm for both the r-Set Packing with t-Membership and t-Overlap via polynomial parametric transformations [16]. In addition in Chapter 7, we improve the kernel sizes for these problems using classical reduction algorithms [44, 117]. Thus with these frameworks, we will obtain immediate kernel results for the graph versions. Furthermore in Chapter 8, we develop FPT-algorithms for the r-Set Packing with t-Membership and t-Overlap problems which will result in fixed-parameter algorithms for the graph versions as well.

This chapter is organized as follows. In Section 5.1, we give our framework for the $\mathcal{H}$-Packing with t-Membership and t-Edge Membership problems, while in Section 5.2, we focus on the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap problems.

5.1 Kernelization Frameworks for Packing Graphs with Membership

Let us start the description of our kernelization frameworks for the $\mathcal{H}$-packing with t-Membership and t-Edge Membership problems. Recall that $r, t \geq 1$ are fixed-constants.

5.1.1 From Packing Graphs with Vertex-Membership to Packing Sets with Membership

In the $\mathcal{H}$-Packing with t-Membership, the goal is to seek for a collection of at least k $\mathcal{H}$-subgraphs of an input graph G such that every vertex of G belongs to at most t subgraphs of that collection. In addition, for every pair of $\mathcal{H}$-subgraphs H_i, H_j, $V(H_i) \neq V(H_j)$.

The next theorem shows that the $\mathcal{H}$-Packing with t-Membership is $W[1]$-Complete when parameterized by k and r. Therefore, we need to consider r as a fixed-constant.

Theorem 9. *The $\mathcal{H}$-Packing with t-Membership is $W[1]$-Complete when parameterized by k and r.*

Proof. The problem of finding a clique of size r in a graph G (i.e., *the Clique problem*) is W[1]-Complete when parameterized by r. There exists a very simple parameterized reduction from Clique to the $\mathcal{H}$-Packing with t-Membership problem. The input graph G and r remains the same. By making $k = 1$ and $\mathcal{H}$ equivalent to the family of cliques, we know that G has a clique of size r if and only if G has a (k, r, t)-$\mathcal{H}$-Membership. □

Before giving our transformation to the r-Set Packing with t-Membership, we will preprocess the input instance. Note that there could exist in G more than one $\mathcal{H}$-subgraph with the same set of vertices (but different set of edges). However by the explicit condition that all $\mathcal{H}$-subgraphs in a (k, r, t)-set membership should have different set of vertices, only one of those $\mathcal{H}$-subgraphs can be in the solution.

Lemma 9. *Let H_i and H_j be a pair of $\mathcal{H}$-subgraphs in G such that $V(H_i) = V(H_j)$ but $E(H_i) \neq E(H_j)$. Any (k, r, t)-$\mathcal{H}$-membership of G that contains H_i does not contain H_j (and vice versa). Furthermore, we can replace H_i by H_j in such membership.*

Proof. Follows because for every pair of $\mathcal{H}$-subgraphs H_i, H_j in a (k, r, t)-$\mathcal{H}$-Membership $V(H_i) \neq V(H_j)$, and every vertex of G is contained in at most t $\mathcal{H}$-subgraphs of a (k, r, t)-$\mathcal{H}$-Membership. □

Recall that $\mathcal{H}_G$ denotes the set of all $\mathcal{H}$-subgraphs in G (see Chapter 2); thus, $|\mathcal{H}_G| = O(|\mathcal{H}| n^{r(\mathcal{H})^2})$. We can find a (k, r, t)-$\mathcal{H}$-membership from G by selecting k $\mathcal{H}$-subgraphs from $\mathcal{H}_G$ such that every vertex of $V(G)$ is contained in at most t of those subgraphs. By Lemma 9, we can apply the next reduction rule to $\mathcal{H}_G$.

Reduction Rule 3. *For any pair of $\mathcal{H}$-subgraphs H_i, H_j in $\mathcal{H}_G$ such that $V(H_i) = V(H_j)$ (but $E(H_i) \neq E(H_j)$), we arbitrary select one and remove the other from $\mathcal{H}_G$.*

It is important to clarify that we only apply Reduction Rule 3 to $\mathcal{H}_G$ for the vertex and induced version of the $\mathcal{H}$-Packing with t-Membership problem. Thus, after applying this rule, $|\mathcal{H}_G| = O(|\mathcal{H}|n^{r(\mathcal{H})})$. Now, we proceed to construct an instance for the r-Set Packing with t-Membership as follows.

Transformation 1. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r*

Let $r = r(\mathcal{H})$.

The universe $\mathcal{U}$ equals $V(G)$.

There is a set S in $\mathcal{S}$ for each $\mathcal{H}$-subgraph H in $\mathcal{H}_G$ and $S = V(H)$.

In this way, $|\mathcal{U}| = O(n)$ and $|\mathcal{S}| = |\mathcal{H}_G|$. After applying Rule 3, $|\mathcal{H}_G| = O(|\mathcal{H}|n^{r(\mathcal{H})})$. Thus, $|\mathcal{S}| = O(|\mathcal{H}|n^r)$. Each set in $\mathcal{S}$ has size at most $r = r(\mathcal{H})$.

Lemma 10. *Transformation 1 runs in $O(n^r)$ time, where $r = r(\mathcal{H})$.*

Lemma 11. *G has a (k, r, t)-$\mathcal{H}$-membership if and only if $\mathcal{S}$ has a (k, r, t)-set membership.*

Proof. We build a (k, r, t)-set membership $\mathcal{K}_S$ from a (k, r, t)-$\mathcal{H}$-membership $\mathcal{K}$. For each $\mathcal{H}$-subgraph H_i in $\mathcal{K}$, we add a set $S_i = V(H_i)$ to $\mathcal{K}_S$. By construction, $S_i \in \mathcal{S}$. Given that every vertex of $V(G)$ is contained in at most t $\mathcal{H}$-subgraphs of $\mathcal{K}$, each element of $\mathcal{U}$ will be contained in at most t sets of $\mathcal{K}_S$.

Given a (k, r, t)-set membership $\mathcal{K}_S$, we build a (k, r, t)-$\mathcal{H}$-membership $\mathcal{K}$ of G. For each set S_i in $\mathcal{K}_S$, we add an $\mathcal{H}$-subgraph $H_i \subseteq G[S_i]$. By construction, H_i is an $\mathcal{H}$-subgraph of G. Each vertex of G will be contained in at most t $\mathcal{H}$-subgraphs of $\mathcal{K}$; otherwise there would be at least one element of $\mathcal{U}$ contained in more than t sets of $\mathcal{K}_S$. □

Let us suppose that there exists a kernelization algorithm KernelAlg-M for the r-Set Packing with t-Membership. That is, KernelAlg-M receives as input an instance of the r-Set Packing with t-Membership ($\mathcal{U}$, $\mathcal{S}$ and k) and outputs in polynomial time a reduced universe $\mathcal{U}'$ and a reduced collection $\mathcal{S}'$ with $|\mathcal{U}'| = O(g(k))$ and $|\mathcal{S}'| = O(h(k))$, for some computable functions g and h. In this way, we run KernelAlg-M on the constructed instance of the r-Set Packing with t-Membership. After that we use the reduced universe $\mathcal{U}'$ to obtain a reduced graph G' for the original graph packing problem as follows.

Transformation 2. ***Input:*** *G, $\mathcal{U}'$;* ***Output:*** *G'*

Return $G' = G[\mathcal{U}']$.

Lemma 12. *G' has a (k, r, t)-$\mathcal{H}$-membership if and only if $\mathcal{S}'$ has a (k, r, t)-set membership.*

By Transformation 2, $|V(G')| = |\mathcal{U}'|$. Since KernelAlg-M is a kernelization algorithm for the r-Set Packing with t-Membership problem, and $|\mathcal{U}'| = O(g(k))$ (for some computable function g), we can conclude.

Lemma 13. *The $\mathcal{H}$-Packing with t-Membership problem can be reduced to a problem kernel with $|\mathcal{U}'|$ vertices.*

The induced version of this problem seeks for at least k induced $\mathcal{H}$-subgraphs in G where each vertex in $V(G)$ is contained in at most t of these $\mathcal{H}$-subgraphs. To achieve a problem kernel, we redefine $\mathcal{S}$ in Transformation 1, adding a set $S = V(H)$ per each induced $\mathcal{H}$-subgraph H of G.

5.1.2 From Packing Graphs with Edge-Membership to Packing Sets with Membership

Similarly to the vertex-membership version, we will reduce the $\mathcal{H}$-Packing with t-Edge Membership problem to the r-Set Packing with t-Membership problem. Recall that in the $\mathcal{H}$-Packing with t-Edge Membership, we seek a collection of at least k $\mathcal{H}$-subgraphs of an input graph G such that every edge of G belongs to at most t subgraphs of the collection, and $E(H_i) \neq E(H_j)$ for any pair of subgraphs H_i, H_j.

For the edge-membership version, we assume that each $H \in \mathcal{H}$ has no isolated vertices. Otherwise, we can replace each $H \in \mathcal{H}$ that has a set H_I of isolated vertices with a graph $H' = H[V(H)\backslash H_I]$. Let $\mathcal{H}'$ denotes the family of subgraphs with these modified graphs. Furthermore, let I denotes the set of isolated vertices contained in G.

Lemma 14. *G has a (k, r, t)-edge-$\mathcal{H}$-membership if and only if $G\backslash I$ has a (k, r, t)-edge-$\mathcal{H}'$-membership.*

Henceforth, for the remainder of this subsection we assume that each $H \in \mathcal{H}$ and hence G as well do not contain isolated vertices. We create next an instance of the r-Set Packing with t-Membership problem using an instance of the $\mathcal{H}$-Packing with t-Edge Membership problem.

Transformation 3. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$ and r.*

Let $r = m(\mathcal{H})$.

The universe $\mathcal{U}$ equals to $E(G)$.

There is a set S in $\mathcal{S}$ for each $\mathcal{H}$-subgraph H in $\mathcal{H}_G$, and $S = E(H)$.

Recall that $\mathcal{H}_G$ is not reduced by Rule 3 for this version of the problem. In this way, $|\mathcal{U}| = O(n^2)$ and $|\mathcal{S}| = |\mathcal{H}_G| = O(|\mathcal{H}|n^{m(\mathcal{H})})$. Each set in $\mathcal{S}$ has size at most r, where $r = m(\mathcal{H})$.

Lemma 15. *Transformation 3 runs in $O(n^r)$ time, where $r = m(\mathcal{H})$.*

Lemma 16. *G has a (k, r, t)-edge-$\mathcal{H}$-membership if and only if $\mathcal{S}$ has a (k, r, t)-set membership.*

Proof. We build a (k, r, t)-set membership $\mathcal{K}_S$ from a (k, r, t)-edge-$\mathcal{H}$-membership $\mathcal{K}$ of G. For each $\mathcal{H}$-subgraph H_i in $\mathcal{K}$, we add a set $S_i = E(H_i)$ to $\mathcal{K}_S$. Since every edge of $E(G)$ is contained in at most t $\mathcal{H}$-subgraphs of $\mathcal{K}$, every element of $\mathcal{U}$ will be contained in at most t $\mathcal{H}$-subgraphs of $\mathcal{K}_S$.

Given a (k, r, t)-set membership $\mathcal{K}_S$, we build a (k, r, t)-edge-$\mathcal{H}$-membership $\mathcal{K}$ of G. For each set S_i in $\mathcal{K}_S$, we add an $\mathcal{H}$-subgraph H_i with set of vertices $V(S_i)$ and set of edges S_i. Each edge of G will be contained in at most t $\mathcal{H}$-subgraphs of $\mathcal{K}$. Otherwise, $\mathcal{K}_S$ would not be a (k, r, t)-set membership. ☐

Once again, we suppose the existence of an algorithm KernelAlg-M that reduces the r-Set Packing with t-Membership problem to a kernel. Let $\mathcal{U}'$ and $\mathcal{S}'$ be the reduced universe and collection of sets, respectively, obtained with Algorithm KernelAlg-M. The reduced graph G' for the original instance of the $\mathcal{H}$-Packing with t-Edge Membership problem is obtained with the following transformation.

Transformation 4. ***Input:*** *$G, \mathcal{U}'$;* ***Output:*** *G'*

Return $G' = G[V(\mathcal{U}')]$.

Lemma 17. *G' has a (k, r, t)-edge-$\mathcal{H}$-membership if and only if $\mathcal{S}'$ has a (k, r, t)-set membership.*

Given that G does not contain isolated vertices (Lemma 14) and by Transformation 4, $|V(G')| \leq 2|\mathcal{U}'|$. Hence, we can state.

Lemma 18. *The $\mathcal{H}$-Packing with t-Edge Membership problem can be reduced to a problem kernel with $2|\mathcal{U}'|$ vertices.*

5.2 Kernelization Frameworks for Packing Graphs with Pairwise Overlap

We now switch our attention to the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap problems. Again for these problems, $r > 0$ and $0 \leq t \leq r - 1$ are fixed-constants.

5.2.1 From Packing Graphs with Vertex-Overlap to Packing Sets with Pairwise Overlap

In the $\mathcal{H}$-Packing with t-Overlap problem, the goal is to find a collection of at least k $\mathcal{H}$-subgraphs of G such that every pair of those $\mathcal{H}$-subgraphs overlaps in at most t vertices. Additionally, for every pair of subgraphs H_i, H_j, $V(H_i) \neq V(H_j)$. Here, we transform an instance of the $\mathcal{H}$-Packing with t-Overlap problem to an instance of the r-Set Packing with t-Overlap problem. Before giving such transformation, we would like to show why we assume that r is considered a fixed-constant. In a similar way as with the t-Membership variant (Theorem 9), we can reduce the Clique problem to the $\mathcal{H}$-Packing with t-Overlap problem.

Theorem 10. *The $\mathcal{H}$-Packing with t-Overlap is $W[1]$-Complete when parameterized by k and r.*

We assume that each $H \in \mathcal{H}$ is an arbitrary graph with at least $t + 1$ vertices. Otherwise, we just add the set of $\mathcal{H}$-subgraphs each with at most t vertices straight to a (k, r, t)-$\mathcal{H}$-Packing and decrease the parameter k by the size of that set.

Once again note that there could exist in G more than one $\mathcal{H}$-subgraph with the same set of vertices (but different set of edges). However, by the explicit condition that all $\mathcal{H}$-subgraphs in a (k, r, t)-$\mathcal{H}$-Packing should have different set of vertices we claim that only one of those $\mathcal{H}$-subgraphs can be in a solution.

Lemma 19. *Let H_i and H_j be a pair of $\mathcal{H}$-subgraphs in G such that $V(H_i) = V(H_j)$ but $E(H_i) \neq E(H_j)$. Any (k, r, t)-$\mathcal{H}$-packing of G that contains H_i does not contain H_j (and vice versa). Furthermore, we can replace H_i by H_j in such packing.*

Proof. Let H_i, H_j be a pair of $\mathcal{H}$-subgraphs with $V(H_i) = V(H_j)$ and $E(H_i) \neq E(H_j)$. Either H_i or H_j are in the solution. This follows because for every pair of $\mathcal{H}$-subgraphs H_i, H_j in a (k, r, t)-$\mathcal{H}$-packing $V(H_i) \neq V(H_j)$. Now suppose that H_i is in a (k, r, t)-$\mathcal{H}$-packing $\mathcal{K}$ and that we cannot replace H_i by H_j in such packing. This would only be possible if H_j overlaps in more than t vertices with some H' in $(\mathcal{K} \backslash H_i)$. A contradiction since $V(H_i) = V(H_j)$. □

We can find a (k,r,t)-$\mathcal{H}$-packing from G by selecting k $\mathcal{H}$-subgraphs from $\mathcal{H}_G$ (the collection of all $\mathcal{H}$-subgraphs of G) that pairwise overlap in at most t vertices. By Lemma 19, we apply Reduction Rule 3 to $\mathcal{H}_G$. In this way, $|\mathcal{H}_G| = O(|\mathcal{H}|n^{r(\mathcal{H})})$.

Next, we create an instance of the r-Set Packing with t-Overlap problem as in Transformation 1. In this way, $|\mathcal{U}| = O(n)$ and $|\mathcal{S}| = |\mathcal{H}_G| = O(|\mathcal{H}|n^r)$. Each set in $\mathcal{S}$ has size at most r.

Lemma 20. *G has a (k,r,t)-$\mathcal{H}$-packing if and only if $\mathcal{S}$ has a (k,r,t)-set packing.*

Proof. We build a (k,r,t)-set packing $\mathcal{K}_S$ from a (k,r,t)-$\mathcal{H}$-packing $\mathcal{K}$. For each $\mathcal{H}$-subgraph H_i in $\mathcal{K}$, we add a set $S_i = V(H_i)$ to $\mathcal{K}_S$. By construction, $S_i \in \mathcal{S}$. Given that every pair H_i, H_j overlaps in at most t vertices, every pair of sets in $\mathcal{K}_S$ overlaps in at most t elements.

Given a (k,r,t)-set packing $\mathcal{K}_S$, we build a (k,r,t)-$\mathcal{H}$-packing $\mathcal{K}$ of G. For each set S_i in $\mathcal{K}_S$ we add an $\mathcal{H}$-subgraph $H_i \subseteq G[S_i]$. By construction, H_i is an $\mathcal{H}$-subgraph of G. Any pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ will overlap in at most t vertices; otherwise there would be at least one pair of sets in $\mathcal{K}_S$ sharing more than t elements. $\square$

Let us suppose that KernelAlg-O is a kernelization algorithm for the r-Set Packing with t-Overlap problem. We obtain a reduced universe $\mathcal{U}'$ and a reduced collection of sets $\mathcal{S}'$ for the constructed instance of the r-Set Packing with t-Overlap with Algorithm KernelAlg-O. After that, the reduced graph G' for the original graph problem is obtained as in Transformation 2.

Lemma 21. *G' has a (k,r,t)-$\mathcal{H}$-packing if and only if $\mathcal{S}'$ has a (k,r,t)-set packing.*

Therefore, we can state.

Lemma 22. *The $\mathcal{H}$-Packing with t-Overlap problem can be reduced to a problem kernel with $|V(G')| = |\mathcal{U}'|$ vertices.*

The induced version seeks for at least k induced $\mathcal{H}$-subgraphs in G that pairwise overlap in at most t vertices. To achieve a problem kernel, we redefine $\mathcal{S}$ of Transformation 1 by adding a set $S = V(H)$ per each induced $\mathcal{H}$-subgraph H of G.

5.2.2 From Packing Graphs with Edge-Overlap to Packing Sets with Pairwise Overlap

In the $\mathcal{H}$-Packing with t-Edge Overlap problem, we search for a collection of at least k $\mathcal{H}$-subgraphs such that every pair of them overlaps in at most t edges. For the edge version of this problem, we assume that each $H \in \mathcal{H}$ has no isolated vertices. Otherwise, we can replace each $H \in \mathcal{H}$ that has a set H_I of isolated vertices with a graph $H' = H \backslash H_I$. Let $\mathcal{H}'$ denote the family of subgraphs with these modified graphs. Furthermore, let I denote the set of isolated vertices contained in G.

Lemma 23. *G has a (k,r,t)-edge-$\mathcal{H}$-overlap if and only if $G \backslash I$ has a (k,r,t)-edge-$\mathcal{H}'$-overlap.*

Henceforth, for the remainder of this section we assume that each $H \in \mathcal{H}$ and hence G as well do not contain isolated vertices. Next, we create an instance of the r-Set Packing with t-Overlap problem with Transformation 3. In this way, $|\mathcal{U}| = O(n^2)$ and $|\mathcal{S}| = |\mathcal{H}_G| = O(|\mathcal{H}|n^{m(\mathcal{H})})$. Each set in $\mathcal{S}$ has size $O(r)$, where $r = m(\mathcal{H})$.

Lemma 24. *G has a (k,r,t)-edge-$\mathcal{H}$-packing if and only if $\mathcal{S}$ has a (k,r,t)-set packing.*

Proof. We build a (k,r,t)-set packing $\mathcal{K}_S$ of $\mathcal{S}$ from a (k,r,t)-edge-$\mathcal{H}$-packing $\mathcal{K}$ of G. For each $\mathcal{H}$-subgraph H_i in $\mathcal{K}$, we add a set $S_i = E(H_i)$ to $\mathcal{K}_S$. By our construction, S_i exists in $\mathcal{S}$. Given that every pair H_i, H_j overlaps in at most t edges, every pair of sets in $\mathcal{K}_S$ overlaps in at most t elements.

Given a (k,r,t)-set packing $\mathcal{K}_S$ we build a (k,r,t)-edge-$\mathcal{H}$-packing $\mathcal{K}$ of G. For each set S_i in $\mathcal{K}_S$ we add an $\mathcal{H}$-subgraph H_i with set of vertices $V(S_i)$ and the set of edges S_i. Any pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ will overlap in at most t edges; otherwise there would be two sets in $\mathcal{K}_S$ sharing more than t elements. □

Once again we obtain a reduced universe $\mathcal{U}'$ and a reduced collection $\mathcal{S}'$ for the constructed instance of the r-Set Packing with t-Overlap using the kernelization algorithm KernelAlg-O. After that, the reduced graph for the original instance is obtained with Transformation 4. Given that G does not contain isolated vertices (Lemma 23), $|V(G')| \leq 2|\mathcal{U}'|$. Thus,

Lemma 25. *The $\mathcal{H}$-Packing with t-Edge Overlap problem can be reduced to a problem kernel with $|V(G')| \leq 2|\mathcal{U}'|$ vertices.*

We summarize the frameworks employed in the previous sections in Algorithm 2. This algorithm besides receiving as input G, $\mathcal{H}$ and k, it also has four additional parameters: "Problem π", [Transformation], [KernelAlg], and [Reinterpretation]. The last three parameters are placeholders for specific routines that vary according to the type of problem that we are reducing. Notice that the parameter k stays the same, as it is not changed in any of our transformations.

Algorithm 2 Kernelization Framework - $\mathcal{H}$-Packing with t-Membership/t-Overlap

```
Input: $G$, $\mathcal{H}$, $k$, "Problem $\pi$", [Transformation], [KernelAlg], and [Reinterpretation].
Output: $G'$
1: Construct an instance of [Problem $\pi$] (a universe $\mathcal{U}$ and a collection $\mathcal{S}$) by running
   [Transformation($G$,$\mathcal{H}$)]
2: Obtain $\mathcal{U}'$ and $\mathcal{S}'$ by running [KernelAlg($\mathcal{U}$,$\mathcal{S}$,$k$)] (i.e., a kernelization algorithm of [Problem $\pi$])
3: Reinterpret $G'$ by running [Reinterpretation($G$,$\mathcal{U}'$)]
4: Return $G'$
```

5.3 Conclusions

We introduced in this chapter a systematic framework to obtain kernelization algorithms for graph packing with pairwise overlap and membership problems. In short, this framework consists of transforming $\mathcal{H}$-Packing with t-Membership and t-Overlap instances to instances of r-Set Packing with t-Membership and t-Overlap, respectively. Algorithm 2 formalizes such approach.

An essential assumption in these frameworks is the existence of kernelization algorithms for the r-Set Packing with t-Membership and t-Overlap problems. In Chapters 6 and 7, we develop two different kernelization algorithms for these latter problems. We will obtain in those chapters immediate kernelization results for our graph packing problems using the kernelization frameworks introduced here.

Chapter 6

Using Polynomial Parametric Transformations Towards Kernelization

In the previous chapter, we described a systematic framework to obtain kernelization results for the $\mathcal{H}$-Packing with t-Membership and t-Overlap problems. In a nutshell, that framework consists on transforming these graph packing problems to the r-Set Packing with t-Membership and t-Overlap, respectively. After we only need to focus on obtaining kernelization algorithms for the set packing problems, as per the frameworks these algorithms will imply kernelization results for the graph packing versions as well.

In this chapter, we obtain kernelization algorithms for both the r-Set Packing with t-Membership and t-Overlap. Instead of using standard reduction rules [44, 117, 33] for this goal, we will employ polynomial parametric transformations (PPTs). Informally, a PPT from a parameterized problem P to a parameterized problem Q is a polynomial time computable function f that transforms any instance of P to an instance of Q subject to the following crucial properties: the instance of P has a solution if and only if the transformed instance of Q has a solution, and the size of the parameter of P grows at most polynomially in the transformed instance. A formal definition of PPT is included in Chapter 2. Even though these transformations have been used mainly for showing the existence of kernelization lower bounds [16], we believe that this is a powerful technique to obtain kernelization algorithms as well. In contrast to classical reductions for example, we obtained those algorithms in a much faster way.

Our kernelization algorithms for the r-Set Packing with t-Membership and t-Overlap are sketched in three steps. First, we develop PPTs to transform an instance of set packing problems that allows overlap (either membership or pairwise) to an instance of the classical set packing problem (i.e., packing disjoint sets). More precisely, we will transform the r-Set Packing with t-Membership and t-Overlap to the r'-Set Packing problem for some $r' > r$. Second, we run a kernelization algorithm for the r'-Set Packing problem in the transformed instances. Finally, we re-interpret the kernel result of the r'-Set Packing problem into kernels of the original r-Set Packing with t-Membership and t-Overlap problems (see Figure 6.1).

By employing the kernelization framework described in the previous chapter, we use our newly obtained kernelization algorithms for the r-Set Packing with t-Membership and t-Overlap to obtain kernelization results for the $\mathcal{H}$-packing with t-Overlap and t-Membership problems.

This chapter is organized as follows. Section 6.1 provides our results for all our membership problems. In Section 6.1.1, we give a PPT from the r-Set Packing with t-Membership problem to the $(r+1)$-Set Packing problem. This leads into an $O((r+1)^r k^r)$ kernel for the first problem. We also discuss how to reduced the weighted version of the r-Set Packing with t-Membership to a kernel using a similar approach as the unweighted version. In Section 6.1.2, we obtain an $O((r+1)^r k^r)$ kernel, where $r = r(\mathcal{H})$, for $\mathcal{H}$-Packing with t-Membership via the kernelization framework of Section 5.1.1. Also in this section, we introduce the ISV variant of that problem and obtain an $O((r+1)^r k^r)$ kernel as well, but via PPTs, as we will discuss, it is not possible to do it within the framework of Section 5.1.1. Similarly in Section 6.1.3, we obtain an $O((r+1)^r k^r)$ kernel, where $r = m(\mathcal{H})$, for $\mathcal{H}$-Packing with t-Edge Membership via the kernelization framework of Section 5.1.2. We also consider the NISV variant of the Edge Membership problem and obtain an $O((r+1)^r k^r)$ kernel as well ($r = m(\mathcal{H})$) with a PPT to $(r+1)$-Set Packing instead of the framework of Section 5.1.2.

In Section 6.2.1, we focus our attention in the pairwise overlap problems. We provide a PPT from r-Set Packing with t-Overlap to the $(\binom{r}{t+1}+1)$-Set Packing problem which allows us to obtain an $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ kernel for the first problem. We also consider the weighted variant of the r-Set Packing with t-Overlap problem. By employing this PPT and by the framework of Section 5.2.1, we obtain a kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = r(\mathcal{H})$ for the $\mathcal{H}$-Packing with t-Overlap problem. In similar way an $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ kernel, where $r = m(\mathcal{H})$, is obtained for the edge-overlap version in Section 6.2.3. Finally in that section, we also consider the NISV variant of the $\mathcal{H}$-Packing with t-Edge Overlap problem and obtain the same kernel size as for the original version but using a polynomial parametric transformation instead. Table 6.1 summarizes the results obtained in this chapter.

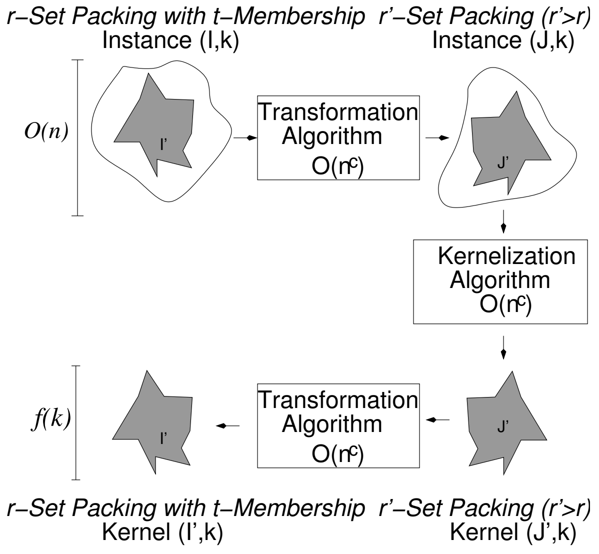
r–Set Packing with t–Membership
Instance (I,k)
r'–Set Packing (r'>r)
Instance (J,k)
O(n)
I'
Transformation Algorithm O(n^c)
J'
Kernelization Algorithm O(n^c)
f(k)
I'
Transformation Algorithm O(n^c)
J'
r–Set Packing with t–Membership
Kernel (I',k)
r'–Set Packing (r'>r)
Kernel (J',k)

Figure 6.1: Schematic of the Kernelization Algorithm by Polynomial Parametric Transformation for the r-Set Packing with t-Membership problem

Problem	Kernel Size	Section
r-Set Packing with t-Membership	$O((r+1)^r k^r)$ elements	6.1.1
$\mathcal{H}$-Packing with t-Membership	$O((r(\mathcal{H})+1)^{r(\mathcal{H})} k^{r(\mathcal{H})})$ vertices	6.1.2
$\mathcal{H}$-Packing with t-Membership (ISV)	$O((r(\mathcal{H})+1)^{r(\mathcal{H})} k^{r(\mathcal{H})})$ vertices	6.1.2
$\mathcal{H}$-Packing with t-Edge Membership	$O((m(\mathcal{H})+1)^{m(\mathcal{H})} k^{m(\mathcal{H})})$ vertices	6.1.3
$\mathcal{H}$-Packing with t-Edge Membership (NISV)	$O((m(\mathcal{H})+1)^{m(\mathcal{H})} k^{m(\mathcal{H})})$ vertices	6.1.3
r-Set Packing with t-Overlap	$O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ elements	6.2.1
$\mathcal{H}$-Packing with t-Overlap	$O((r(\mathcal{H})^{t+1}+1)^{r(\mathcal{H})^{t+1}} k^{r(\mathcal{H})^{t+1}})$ vertices	6.2.2
$\mathcal{H}$-Packing with t-Edge Overlap	$O((m(\mathcal{H})^{t+1}+1)^{m(\mathcal{H})^{t+1}} k^{m(\mathcal{H})^{t+1}})$ vertices	6.2.3
$\mathcal{H}$-Packing with t-Edge Overlap (NISV)	$O((m(\mathcal{H})^{t+1}+1)^{m(\mathcal{H})^{t+1}} k^{m(\mathcal{H})^{t+1}})$ vertices	6.2.3

Table 6.1: Summary of results presented in Chapter 6

6.1 PPTs for Packing Problems with Membership

In this section, we introduce a polynomial parametric transformation from the r-Set Packing with t-Membership problem to the $(r+1)$-Set Packing problem. We obtain a kernel result for the r-Set Packing with t-Membership problem by running a kernelization algorithm on the constructed instance of the $(r+1)$-Set Packing problem. This compression result can be turned into a proper kernel result by re-interpreting the $(r+1)$-Set Packing kernel within the original problem.

To obtain kernel results for the $\mathcal{H}$-Packing with t-Membership and t-Edge Membership problems, we transform each of these problems to an instance of the r-Set Packing with t-Membership problem as explained in Section 5.1. Recall that $r, t \geq 1$ are fixed constants.

6.1.1 From Packing Sets with Membership to Packing Disjoint Sets

Let us begin with the r-Set Packing with t-Membership problem. The objective of this problem is to obtain at least k distinct sets from a collection of sets $\mathcal{S}$ drawn from a universe $\mathcal{U}$ such that every element of $\mathcal{U}$ is contained in at most t of these sets, if one exists.

Recall that the $(r+1)$-Set Packing problem searches for at least k pairwise disjoint sets from a given collection of sets. We create an instance for this problem (a universe $\mathcal{U}^T$ and a collection $\mathcal{S}^T$) using an instance of the r-Set Packing with t-Membership problem as follows. Notice that size of a set in $\mathcal{S}$ is at most r while the size of a set in $\mathcal{S}^T$ is at most $r+1$.

Transformation 5. ***Input:*** *$\mathcal{S}, \mathcal{U}$;* ***Output:*** *$\mathcal{S}^T, \mathcal{U}^T$*

The universe $\mathcal{U}^T$ equals $(\mathcal{U} \times \{1, \ldots, t\}) \cup \mathcal{S}$.

The collection $\mathcal{S}^T$ contains all subsets of $\mathcal{U}^T$ each with at most $r+1$ elements of the following form: $\{\{(u_1, j_1), \ldots, (u_i, j_i), \ldots, (u_{r'}, j_{r'}), S\} \mid S \in \mathcal{S},\ S = \{u_1, \ldots, u_{r'}\}$, for each $1 \leq j_i \leq t$ and $1 \leq i \leq r'$, where $r' \leq r\}$.

Intuitively, since each element in $\mathcal{U}$ can be in at most t sets of a (k, r, t)-set membership, we duplicate each element t times. This would imply that there are t^r sets in $\mathcal{S}^T$ representing the same set $S \in \mathcal{S}$. All of those sets have a common element which is precisely the set of elements in S, i.e., some $S \in \mathcal{S} \subset \mathcal{U}^T$. This is why the upper bound on the number of elements per set in $\mathcal{S}^T$ increases by one with respect to the sets in $\mathcal{S}$.

Example 1. *For an instance of the r-Set Packing with t-Membership with universe* $\mathcal{U} = \{a, b, c, d, e, f, g, h\}$, $\mathcal{S} = \{\ \{a, b, c, d\},\ \{b, c, e, f\},\ \{b, c, g, h\}\ \}$, $k = 2$ *and* $t = 2$, *the constructed instance of the* $(4 + 1)$*-Set Packing is as follows.*

$$\begin{aligned} \mathcal{U}^T &= \{(a,1),(b,1),(c,1),(d,1),(e,1),(f,1),(g,1),(h,1),\\ &\quad\ (a,2),(b,2),(c,2),(d,2),(e,2),(f,2),(g,2),(h,2)\}\\ &\cup\ \ \mathcal{S}. \end{aligned}$$

Some sets of the collection $\mathcal{S}^T$ *are* $\{(a,1),\ (b,1),\ (c,1),\ (d,1),\ \{a,b,c,d\}\}$, $\{(a,1),\ (b,1),\ (c,1),\ (d,2),\ \{a,b,c,d\}\}$, $\{(a,2),\ (b,2),\ (c,2),\ (d,2),\ \{a,b,c,d\}\}$, $\{(b,1),\ (c,2),\ (e,1),\ (f,2),\ \{b,c,e,f\}\}$, $\{(b,1),\ (c,1),\ (g,2),\ (h,2),\ \{b,c,g,h\}\}$.

A $(2, 4 + 1, 0)$*-set packing is*

$$\begin{gathered} \{(a,1),(b,1),(c,1),(d,1),\{a,b,c,d\}\},\\ \{(b,2),(c,2),(e,1),(f,1),\{b,c,e,f\}\}. \end{gathered}$$

This corresponds to the $(2, 4, 2)$*-set membership* $\{\{a, b, c, d\},\ \{b, c, e, f\}\}$.

The size of $\mathcal{U}^T$ is bounded by $|\mathcal{U}| \cdot t + |\mathcal{S}| < tn + rn^r = O(n^r)$. Each set in $\mathcal{S}^T$ has size at most $r + 1$. For each $S \in \mathcal{S}$, we can form at most t^r sets with the tr ordered pairs from the elements in S. In this way, for each $S \in \mathcal{S}$ there are t^r sets in $\mathcal{S}^T$, and $|\mathcal{S}^T| = t^r|\mathcal{S}| = O(t^r n^r)$. This leads us to the following result.

Lemma 26. *Transformation 5 can be computed in* $O(t^r n^r)$ *time.*

Note that the parameter k stays the same in this transformation, and t only influences the running time of the whole construction, as the $(r + 1)$-Set Packing instance will grow if t gets bigger.

Lemma 27. $\mathcal{S}$ *has a* (k, r, t)*-set membership if and only if* $\mathcal{S}^T$ *has a* $(k, r+1, 0)$*-set packing (i.e., k disjoint sets).*

Proof. We build a $(k, r + 1, 0)$-set packing $\mathcal{K}_{\mathcal{S}}$ from a (k, r, t)-set membership $\mathcal{K}$. For each $S_j \in \mathcal{K}$, we add a set S_j^T to $\mathcal{S}^T$ with $|S_j| + 1 \leq r + 1$ elements. By construction, S_j^T has one element from $\mathcal{S}$ and at most r ordered pairs. As $S_j \in \mathcal{S} \subset \mathcal{U}^T$, we can choose $S_j^T \cap \mathcal{S} = \{S_j\}$. For each element $u_i \in S_j$ $(1 \leq i \leq |S_j| \leq r)$, we add an ordered pair $(u_i, l + 1)$ to S_j where l corresponds to the number of sets in $\{S_1, \ldots, S_{j-1}\} \subset \mathcal{K}$ that contain u_i. Since each element of $\mathcal{U}$ is a member of at most t sets of $\mathcal{K}$ (otherwise $\mathcal{K}$ would not be a solution), l is at most $t - 1$ and $(u_i, l + 1)$ always exists in $\mathcal{U}^T$. Thus, $S_i^T \in \mathcal{S}^T$. It remains to show that the sets in $\mathcal{K}_{\mathcal{S}}$ are pairwise disjoint. By construction, none of the sets in $\mathcal{K}_{\mathcal{S}}$ will share the element S. In addition, no pair S_i^T, S_j^T shares an ordered pair (u, l) for some $u \in \mathcal{U}$ and $1 \leq l \leq t$. Namely, if S_i^T and S_j^T share a pair (u, l), this would imply that there are two sets $S_i, S_j \in \mathcal{K}$ that share the same element u.

Without lost of generality, assume that S_i appears before S_j in $\mathcal{K}$. If (u, l) is contained in both S_i^T and S_j^T, then u is a member of $l-1$ sets from both $\{\dots, S_i, \dots\}$ and $\{\dots, S_i, \dots, S_j\}$. This is a contradiction, since both S_i and S_j contain u.

We construct a (k, r, t)-set membership $\mathcal{K}$ using a $(k, r+1, 0)$-set packing $\mathcal{K}_S$. Each set $S_i^T \in \mathcal{K}_S$ has an element $S \in \mathcal{S}$. Let us denote that element as S_i. For each set $S_i^T \in \mathcal{K}_S$, we add S_i to $\mathcal{K}$. Since the sets in $\mathcal{K}_S$ are pairwise disjoint, $S_i \neq S_j$. Now, we need to show that each element of $\mathcal{U}$ is a member of at most t sets in $\mathcal{K}$. Assume otherwise for the sake of contradiction. If two sets in $\mathcal{K}$ share an element u, then there are two sets in $\mathcal{K}_S$ each one with a pair (u, i) and (u, j), respectively. Since the sets in $\mathcal{K}_S$ are disjoint, $i \neq j$. If an element $u \in \mathcal{U}$ is a member of more than t sets in $\mathcal{K}$, then there are at least $t+1$ ordered pairs $(u, 1), \dots, (u, t+1)$ contained in $t+1$ different sets in $\mathcal{K}_S$. However, by the construction of $\mathcal{U}^T$, there are at most t ordered pairs that contain u. $\square$

Up to this point we have introduced a polynomial parametric transformation from the r-Set Packing with t-Membership to the $(r+1)$-Set Packing. Next, we run the currently best kernelization algorithm for the $(r+1)$-Set Packing problem [1] on the constructed instance. This algorithm would leave us with a new universe $\mathcal{U'}^T$ with at most $2r!((r+1)k-1)^r$ elements, as well with a collection $\mathcal{S'}^T$ of subsets. The next property is borrowed from [1].

Lemma 28. *$\mathcal{S}^T$ has a $(k, r+1, 0)$-set packing if and only if $\mathcal{S'}^T$ has a $(k', r+1, 0)$-set packing, where $k' \leq k$.*

Next, we construct the reduced universe $\mathcal{U}'$ and $\mathcal{S}'$ for our original membership problem using $\mathcal{U'}^T$ and $\mathcal{S'}^T$ as follows.

Transformation 6. ***Input:*** $\mathcal{S'}^T$, $\mathcal{U'}^T$; ***Output:*** $\mathcal{S}'$, $\mathcal{U}'$

The reduced universe $\mathcal{U}'$ equals to the union of elements u for each $(u, i) \in \mathcal{U'}^T$.

The reduced collection $\mathcal{S}'$ contains a set which corresponds to the element S in each set in $\mathcal{S'}^T$.

$\mathcal{U}'$ and $\mathcal{S}'$ together with k, give the reduced r-Set Packing with t-Membership instance we are looking for. By construction, $\mathcal{U}'$ will have at most $2r!((r+1)k-1)^r$ elements. This reduction property allows us to state:

Theorem 11. *The r-Set Packing with t-Membership has a problem kernel with $O((r+1)^r k^r)$ elements from the given universe.*

By Transformations 5 and 6, each set in $\mathcal{S}'$ must be contained in the collection $\mathcal{S} \in \mathcal{U'}^T$. Since $|\mathcal{U'}^T| = O((r+1)^r k^r)$, we can conclude:

Theorem 12. *The r-Set Packing with t-Membership has a problem kernel with $O((r+1)^r k^r)$ sets.*

Our kernelization algorithm is summarized in Algorithm 3.

Algorithm 3 Kernelization Algorithm by PPT - r-Set Packing with t-Membership

```
Input: $\mathcal{U}$ and $\mathcal{S}$
Output: Reduced $\mathcal{U}'$ and $\mathcal{S}'$
1: Construct a universe $\mathcal{U}^T$ and collection $\mathcal{S}^T$ using Transformation 5.
2: Run the $(r+1)$-Set Packing kernelization algorithm [1] on $\mathcal{U}^T$ and $\mathcal{S}^T$.
3: Using Transformation 6, re-interpret the kernel $\mathcal{S}'^T$, $\mathcal{U}'^T$ as reduced $\mathcal{S}'$, $\mathcal{U}'$, respectively.
4: Return $\mathcal{U}'$, $\mathcal{S}'$.
```

Packing Weighted Sets with Membership using PPTs

Next, we briefly discuss how to employ our PPT approach to reduce the weighted version of the r-Set Packing with t-Membership problem to a kernel. Let us start first by defining this variant. In the *weighted r-Set packing with t-Membership problem*, every set in $\mathcal{S}$ has associated a weight (a real number), and the goal is to find a (k, r, t)-set membership of maximum weight.

To reduce the weighted variant, we construct again an instance of the r-Set Packing using Transformation 5. Recall that there will be t^r sets in $\mathcal{S}^T$ (the constructed collection of sets) representing the same set $S \in \mathcal{S}$. Thus, we assign the weight of S to each of these t^r sets. By using similar arguments as in Lemma 27, we can show that the instance $(\mathcal{U}, \mathcal{S}, k)$ has a (k, r, t)-set membership of maximum weight if and only if the constructed instance $(\mathcal{U}^T, \mathcal{S}^T, k)$ has a $(k, r+1, 0)$-set packing of maximum weight. After that, we will run the kernelization algorithm of Zehavi [140] for the weighted version of the r-Set Packing problem in our constructed instance. Subsequently, we re-intepret back the kernel returned by Algorithm [140] as in Transformation 6. Given that Algorithm [140] returns a reduced kernel of size $O(e^{r'}r'k^{r'})$ where $r' = r+1$ (on the number of sets), we can hence conclude:

Theorem 13. *The weighted r-Set Packing with t-Membership has a problem kernel with $O(e^{r+1}(r+1)k^{r+1})$ sets.*

6.1.2 A Kernel Reduction for the $\mathcal{H}$-Packing with t-Membership and t-Membership (ISV) Problems

Recall that the $\mathcal{H}$-Packing with t-Membership problem seeks for at least k $\mathcal{H}$-subgraphs where each vertex of G belongs to at most t of them. In addition, $V(H_i) \neq V(H_j)$ for every pair H_i, H_j of those $\mathcal{H}$-subgraphs.

To reduce this problem to a kernel, we will use the framework explained in Section 5.1.1. In that framework, we show how to reduce $\mathcal{H}$-Packing with t-Membership to a kernel problem by transforming it to an instance of the r-Set Packing with t-Membership, assuming the existence of a kernelization algorithm KernelAlg-M for the latter problem. In this chapter, KernelAlg-M will correspond to Algorithm 3.

Let us first recapitulate our kernelization framework. We first transform an instance of the $\mathcal{H}$-Packing with t-Membership problem to an instance of the r-Set Packing with t-Membership (Transformation 1). That is, we construct a universe $\mathcal{U}$ and a collection $\mathcal{S}$. The parameter k remains the same, and $r = r(\mathcal{H})$. Recall that $r(\mathcal{H})$ is the order of the largest graph in the family of graphs $\mathcal{H}$.

After that we reduced to a kernel this constructed instance for the r-Set packing with t-Membership problem with Algorithm 3, i.e., we obtain a reduced universe $\mathcal{U}'$ and a reduced collection $\mathcal{S}'$. Finally, we re-interpret the reduced universe $\mathcal{U}'$ as a reduced instance of the original $\mathcal{H}$-Packing with t-Membership problem, i.e. a reduced graph G' (Transformation 2).

According to Transformation 2, $G' = G[\mathcal{U}']$, and by Algorithm 3 $|\mathcal{U}'| = O((r+1)^r k^r)$. Lemma 13 allows us to conclude:

Theorem 14. *$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O((r+1)^r k^r)$ vertices, where $r = r(\mathcal{H})$.*

Also in Section 5.1.1, we describe how to reduce the induced version of the $\mathcal{H}$-Packing with t-Membership to a problem kernel in a similar way as the non-induced version. Thus, we obtain the next result.

Theorem 15. *Induced-$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O((r+1)^r k^r)$ vertices, where $r = r(\mathcal{H})$.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Membership that we described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing with t-Membership*, Transformation 1, Algorithm 3, and Transformation 2.

In Appendix A.1.1, we include a polynomial parametric transformation from the $\mathcal{H}$-Packing with t-Membership problem to the $(r+1)$-Set Packing problem, where $r = r(\mathcal{H})$ (instead to the r-Set Packing with t-Membership problem). That transformation leads to the same kernel size as in Theorem 14.

The $\mathcal{H}$-Packing with t-Membership (ISV) Problem

Recall that one condition of the $\mathcal{H}$-Packing with t-Membership problem is that every pair H_i, H_j of $\mathcal{H}$-subgraphs in the solution should have different set of vertices, i.e., $V(H_i) \neq V(H_j)$. In this section, we discuss a more flexible definition of the $\mathcal{H}$-Packing with t-Membership problem that allows $\mathcal{H}$-subgraphs in a (k, r, t)-$\mathcal{H}$-membership with identical sets of vertices but requires different sets of edges. We indicate this variant by adding ISV (identical set of vertices) to the problem name.

The $\mathcal{H}$-Packing with t-Membership (ISV) problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-*$\mathcal{H}$-membership* (ISV), i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

Note that every solution for the original problem is a solution for the ISV variant, but the converse is not necessarily true.

Example 2. *Consider the input graph of Figure 6.2 and suppose that $\mathcal{H} = \{C_4, K_4\}$, $k = 2$, and $t = 2$. A (k, r, t)-$\mathcal{H}$-membership (ISV) is*

$$\begin{aligned}\{H_1 =&(\{b,c,f,e\},\{bc,cf,ef,be\}),\\ H_2 =&(\{b,c,f,e\},\{bc,cf,ef,be,bf,ce\})\}.\end{aligned}$$

However, this is not a (k, r, t)-$\mathcal{H}$-membership.

Notice that we cannot transform the ISV variant to the r-Set Packing with t-Membership (as we did with the original version) because Lemma 9 and Lemma 11 do not hold for the ISV version. Therefore, to achieve a kernelization algorithm for the $\mathcal{H}$-Packing with t-Membership (ISV), we introduce a polynomial parametric transformation from the $\mathcal{H}$-Packing with t-Membership problem (ISV) to an instance of the $(r+1)$-Set Packing problem (packing disjoint sets) directly.

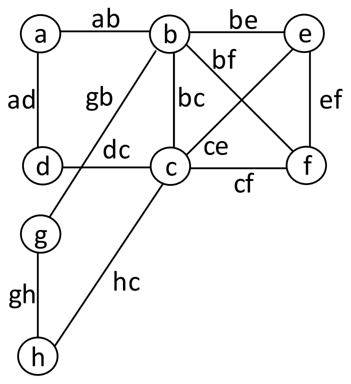

Figure 6.2: An input graph G for the $\mathcal{H}$-Packing with t-Membership problem.

Recall from Chapter 2 that $\mathcal{H}_G$ is the set of all $\mathcal{H}$-subgraphs in G, and $\mathcal{E}$ defines a collection of sets of edges each of which induces a subgraph isomorphic to some $H \in \mathcal{H}$. We clarify that for this transformation, we do not apply Reduction Rule 3 to $\mathcal{H}_G$. Next, we create an instance for the $(r+1)$-Set Packing problem (a universe $\mathcal{U}$ and a collection $\mathcal{S}$) as follows.

Transformation 7. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r.*

Let $r = r(\mathcal{H})$.

The universe $\mathcal{U}$ equals $(V(G) \times \{1, \ldots, t\}) \cup \mathcal{E}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ each with at most $r+1$ elements

$$\{\{(v_1, j_1), \ldots, (v_i, j_i), \ldots, (v_{r'}, j_{r'}), E\} \mid E \in \mathcal{E},$$
$$V(E) = \{v_1, \ldots, v_{r'}\}, \textit{ for each } 1 \leq j_i \leq t \textit{ and } 1 \leq i \leq r', \textit{ where } r' \leq r\}$$

Intuitively, each set in $\mathcal{S}$ would represent an $\mathcal{H}$-subgraph in G. Since each vertex in $V(G)$ can be in at most t $\mathcal{H}$-subgraphs, we replicate each vertex t times. To allow $\mathcal{H}$-subgraphs with the same set of vertices, we add the element E to the sets. In this way, $\mathcal{H}$-subgraphs with identical set of vertices will have a different element E.

Example 3. *Let us consider the graph of Figure 6.2. $\mathcal{H} = \{C_4, K_4\}$, $t = 3$, and $k = 3$.*

$$\begin{aligned}\mathcal{E} =&\{\{ab, ad, bc, dc\}, \{bg, bc, gh, hc\}, \{be, ef, cf, bc\},\\ &\{be, ce, cf, bf\}, \{be, bc, ef, cf, bf, ce\}\}\end{aligned}$$

and

$$\begin{aligned}&\mathcal{U} = \{(a,1),(b,1),(c,1),(d,1),(e,1),(f,1),(g,1),(h,1),\\ &(a,2),(b,2),(c,2),(d,2),(e,2),(f,2),(g,2),(h,2)\}\\ &\cup \mathcal{E}.\end{aligned}$$

Some sets of the collection $\mathcal{S}$ are

$$\begin{aligned}&\{(a,1),(b,1),(c,1),(d,1),\{ab,ad,bc,bd\}\},\\ &\{(a,1),(b,1),(c,1),(d,2),\{ab,ad,bc,bd\}\},\\ &\{(b,1),(c,1),(e,1),(f,1),\{be,ef,cf,bc\}\},\\ &\{(b,2),(c,2),(e,2),(f,2),\{be,ce,cf,bf\}\},\\ &\{(b,3),(c,3),(e,3),(f,3),\{be,bc,ef,cf,bf,ce\}\}.\end{aligned}$$

A $(3, 4+1, 0)$-set packing is

$$\begin{aligned}&\{\{(b,1),(c,1),(e,1),(f,1),\{be,ef,cf,bc\}\},\\ &\{(b,2),(c,2),(a,1),(d,1),\{ab,ad,bc,bd\}\},\ \textit{and}\\ &\{(b,3),(c,3),(g,3),(h,3),\{be,bc,ef,cf,bf,ce\}\}\}.\end{aligned}$$

This corresponds to the $(3, 4, 3)$-$\mathcal{H}$-membership (ISV)

$$\begin{aligned}&\{H_1 = (\{b,c,e,f\},\{be,ef,cf,bc\}),\\ &H_2 = (\{a,b,c,d\},\{ab,ad,bc,bd\}),\ \textit{and}\\ &H_3 = (\{b,c,e,f\},\{be,bc,ef,cf,bf,ce\})\}.\end{aligned}$$

The size of $\mathcal{U}$ is upper-bounded by $|\mathcal{U}| = |V(G)| \times t + |\mathcal{E}| \leq tn + n^{m(\mathcal{H})} = O(n^{r^2})$. Each set in $\mathcal{S}$ has size at most $r+1$ (where $r = r(\mathcal{H})$) and $|\mathcal{S}| \leq t^r|\mathcal{H}|n^{m(\mathcal{H})}$. Thus,

Lemma 29. *Transformation 7 runs in $O(t^r|\mathcal{H}|n^{m(\mathcal{H})})$ time*

Lemma 30. *G has a (k, r, t)-$\mathcal{H}$-membership (ISV) if and only if $\mathcal{S}$ has a $(k, r+1, 0)$-set packing.*

Proof. We build a $(k, r+1, 0)$-set packing $\mathcal{K}_{\mathcal{S}}$ from a (k, r, t)-$\mathcal{H}$-membership (ISV) $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $|V(H_j)| + 1 \leq r + 1$ elements as follows. The element E in S_j corresponds to the set of edges $E(H_j) = \{e_1, \ldots, e_{|E|}\}$. Note that $E(H_j) \in \mathcal{E} \subseteq \mathcal{U}$. For each vertex $v_i \in V(H_j)$ $(1 \leq i \leq |V(H_j)| \leq r)$, we add an ordered pair $(v_i, l+1)$ to S_j, where l corresponds to the number of $\mathcal{H}$-subgraphs in $\{H_1, \ldots, H_{j-1}\} \subseteq \mathcal{K}$ that v_i is contained in. Since each vertex is contained in at most t $\mathcal{H}$-subgraphs, $0 \leq l \leq t-1$ and $(v_i, l+1)$ always exists in $\mathcal{U}$. By construction of $\mathcal{S}$, $S_j \in \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_{\mathcal{S}}$ are pairwise disjoint. Since $E(H_i) \neq E(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_{\mathcal{S}}$ will share the same element E. In addition, no pair S_i, S_j shares an ordered pair (v, l) for some $v \in V(G)$ and $1 \leq l \leq t$. If S_i and S_j share a pair (v, l) this would imply that there are two $\mathcal{H}$-subgraphs H_i and H_j that share the same vertex v. Without lost of generality, assume that H_i appears before H_j in $\mathcal{K}$. If (v, l) is contained in both S_i and S_j, then v is a member of $l-1$ $\mathcal{H}$-subgraphs from both $\{\ldots, H_i, \ldots\}$ and $\{\ldots, H_i, \ldots, H_j\}$, a contradiction, since both H_i and H_j contain v.

We construct a (k, r, t)-$\mathcal{H}$-membership (ISV) $\mathcal{K}$ using a $(k, r+1, 0)$-set packing $\mathcal{K}_{\mathcal{S}}$. Each set $S_i \in \mathcal{K}_S$ has an element $E_{S_i} \in \mathcal{E}$ which corresponds to a set of edges in $E(G)$ that forms a subgraph isomorphic to $H \in \mathcal{H}$ in G. In addition, we take the vertex v_l that appears in each $(v_l, j) \in S_i$ (for $1 \leq l \leq r$ and $j \in [1 \ldots t]$) and we add it to a set V_{S_i}. In this way, for each set $S_i \in \mathcal{K}_S$, we add to $\mathcal{K}$ an $\mathcal{H}$-subgraph H_{S_i} with a set of vertices V_{S_i} and a set of edges E_{S_i}, i.e., $H_{S_i} = (V_{S_i}, E_{S_i})$. Since the sets in $\mathcal{K}_S$ are pairwise disjoint, $E(H_{S_i}) \neq E(H_{S_j})$. Now, we need to show that each vertex of $V(G)$ is a member of at most t $\mathcal{H}$-subgraphs in $\mathcal{K}$. Assume otherwise by contradiction. If two $\mathcal{H}$-subgraphs in $\mathcal{K}$ share a vertex v, then there are two sets in $\mathcal{K}_S$ where each one has a pair (v, i) and (v, j), respectively. Since the sets in $\mathcal{K}_S$ are disjoint then $i \neq j$. If a vertex $v \in V(G)$ is a member of more than t $\mathcal{H}$-subgraphs in $\mathcal{K}$ then there are at least $t+1$ ordered pairs $(v, 1), \ldots, (v, t+1)$ contained in $t+1$ different sets in $\mathcal{K}_S$. However, by construction of $\mathcal{U}$, there are at most t ordered pairs that contain v. $\square$

Given a reduced universe $\mathcal{U}' \subseteq \mathcal{U}$, we construct a graph G' as follows.

Transformation 8. ***Input:*** *G, $\mathcal{U}'$;* ***Output:*** *G'*

We take each vertex v that appears in each $(v, i) \in \mathcal{U}'$ and the vertices in each $E \in \mathcal{E}'$ and consider the graph G' that is induced by all these in G.

Lemma 31. *G' has a (k, r, t)-$\mathcal{H}$-membership (ISV) if and only if $\mathcal{S}'$ has a $(k, r+1, 0)$-set packing.*

Our kernelization algorithm for $\mathcal{H}$-Packing with t-Membership (ISV) problem corresponds to Algorithm 2 with input: G, $\mathcal{H}$, k, $(r+1)$-*Set Packing*, Transformation 7, kernelization algorithm [1], and Transformation 8. Given that the universe $\mathcal{U}'$ returned by Algorithm [1] has at most $2r!((r+1)k-1)^r$ elements, then by construction, G' has at most $O((r+1)^r k^r)$ vertices.

Theorem 16. *The $\mathcal{H}$-Packing with t-Membership problem (ISV) has a kernel with $O((r+1)^r k^r)$ vertices, where $r = r(\mathcal{H})$.*

6.1.3 A Kernel Reduction for the $\mathcal{H}$-Packing with t-Edge Membership and t-Edge Membership (NISV) Problems

In the $\mathcal{H}$-Packing with t-Edge Membership problem, we seek for a collection of least k $\mathcal{H}$-subgraphs where each edge of G belongs to at most t $\mathcal{H}$-subgraphs of that collection. Similarly as with the $\mathcal{H}$-Packing with t-Membership problem (i.e., the vertex version), we will use the kernelization framework described in Section 5.1.2.

Using an instance of the $\mathcal{H}$-Packing with t-Edge Membership problem, we construct an instance of the r-Set Packing with t-Membership: a universe $\mathcal{U}$ and a collection $\mathcal{S}$ (Transformation 3). The parameter k remains the same and $r = m(\mathcal{H})$. Recall that $m(\mathcal{H})$ is the size of the largest subgraph in the family of graphs $\mathcal{H}$. After that we reduced to a kernel this constructed instance for the r-Set packing with t-Membership problem with Algorithm 3, i.e., we obtain reduced universe $\mathcal{U}'$ and collection $\mathcal{S}'$. Finally, we re-interpret the reduced universe $\mathcal{U}'$ as a reduced instance of the original $\mathcal{H}$-Packing with t-Edge Membership problem, i.e. a reduced graph G' (Transformation 4).

By Transformation 4, $G' = G[V(\mathcal{U}')]$ and $|V(G')| \leq 2|\mathcal{U}'|$ (G does not contain isolated vertices by Lemma 14).

By Lemma 18, we can state.

Theorem 17. *The $\mathcal{H}$-Packing with t-Edge Membership problem has a kernel with $O((r+1)^r k^r)$ vertices where $r = m(\mathcal{H})$.*

Our kernelization algorithm for the $\mathcal{H}$-Packing with t-Edge Membership problem corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing with t-Membership*, Transformation 3, Algorithm 3, and Transformation 4.

For completeness, we include a polynomial parametric transformation from the $\mathcal{H}$-Packing with t-Edge Membership problem to the $(r+1)$-Set Packing, where $r = m(\mathcal{H})$ (instead to the r-Set Packing with t-Membership) in Appendix A.1.2. That transformation leads to the same kernel size as in Theorem 17.

The $\mathcal{H}$-Packing with t-Edge Membership (NISV) Problem

We present a stricter definition of the $\mathcal{H}$-Packing with t-Edge Membership problem to avoid $\mathcal{H}$-subgraphs in a (k, m, t)-edge-$\mathcal{H}$-membership with identical sets of vertices. This is equivalent to adding the condition $V(H_i) \neq V(H_j)$ to the $\mathcal{H}$-Packing with t-Edge Membership problem definition. We will indicate this variant by adding NISV (non identical set of vertices) to the problem name.

The $\mathcal{H}$-Packing with t-Edge Membership (NISV) problem
Input: A graph G, and a positive integer k.
Parameter: k
Question: Does G contain a (k, r, t)-edge-$\mathcal{H}$-membership, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$, and every edge in $E(G)$ belongs to at most t subgraphs of $\mathcal{K}$?

Note that every solution for the NISV variant is a solution for the original problem, but the converse is not necessarily true.

Example 4. *Let us consider the graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $t = 2$, and $k = 2$. A $(2, 4, 2)$-edge-$\mathcal{H}$-membership (NISV) corresponds to*

$H_1 = (\{a,b,c,d\}\{ab,ad,bc,dc\}), H_2 = (\{b,e,c,f\}\{bc,be,ef,cf\})$.

This is also a $(2,4,2)$-edge-$\mathcal{H}$-membership.

On the other hand, $H_1 = (\{b,e,c,f\}\{be,ef,cf,bc\}), H_2 = (\{b,e,c,f\}\{be,ce,cf,bf\})$ is a $(2,4,2)$-edge-$\mathcal{H}$-membership but is not a $(2,4,2)$-edge-$\mathcal{H}$-membership (NISV).

We cannot use the same kernelization approach for the NISV variant as we did with the original $\mathcal{H}$-Packing with t-Edge Membership problem. That is, transforming the NISV variant to an instance of the r-Set Packing with t-Membership problem. This follows because not every (k,r,t)-set membership can be transformed into a (k,r,t)-$\mathcal{H}$-membership (NISV), and Lemma 16 would not hold.

Therefore, we introduce a polynomial parametric transformation that reduces the $\mathcal{H}$-Packing with t-Edge Membership problem (NISV) directly to $(r+1)$-Set Packing (packing disjoint sets).

As defined in Chapter 2, $\mathcal{V}$ is a collection of sets of vertices each of which forms a subgraph (not necessarily induced) isomorphic to some $H \in \mathcal{H}$, and $\mathcal{E}$ is a collection of sets of edges each of which induces a subgraph isomorphic to some $H \in \mathcal{H}$.

Transformation 9. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r.*

Let $r = m(\mathcal{H})$.

The universe $\mathcal{U}$ equals $(E(G) \times \{1,\dots,t\}) \cup \mathcal{V}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ each with at most $r+1$ elements:

$$\begin{aligned}&\{\{(e_1,j_1),\dots,(e_{m'},j_{m'}),V(E)\} \mid E \in \mathcal{E},\\ &E = \{e_1,\dots e_{m'}\} \textit{ for each } 1 \le j_i \le t \textit{ and } 1 \le i \le m', \textit{ where } m' \le m(\mathcal{H})\}.\end{aligned}$$

The intuition behind our transformation is that each set in $\mathcal{S}$ would represent an $\mathcal{H}$-subgraph in G. Since each edge in $E(G)$ can be in at most t $\mathcal{H}$-subgraphs, we replicate each edge t times. This would imply that there are t^r sets in $\mathcal{S}$ representing the same $\mathcal{H}$-subgraph H of G. All of those sets have a common element which is precisely the set of vertices in $V(H)$, i.e., some $V \in \mathcal{V} \subset \mathcal{U}$. This will avoid to choosing sets that correspond to $\mathcal{H}$-subgraphs with identical set of vertices.

Example 5. *For the graph of Figure 6.2, $\mathcal{H} = \{C_4\}$, $k = 3$, and $t = 2$.*

$$\begin{aligned}\mathcal{V} &= \{\{a,b,c,d\},\{b,c,e,d,f\},\{b,c,g,h\}\}\\ \mathcal{U} &= \{(ab,1),(ad,1),(dc,1),(bc,1),(be,1),(ef,1),(cf,1),(bf,1),(ec,1),(gb,1),\\ &\quad (dc,1),(hc,1),(ab,2),(ad,2),(dc,2),(bc,2),(be,2),(ef,2),(cf,2),(bf,2),\\ &\quad (ec,2),(gb,2),(dc,2),(hc,2)\}\\ &\cup\ \mathcal{V}.\end{aligned}$$

Some sets of the collection $\mathcal{S}$ are

$$\begin{gathered}\{(ad,1),(ab,1),(bc,1),(dc,1),\{a,b,c,d\}\},\\ \{(bc,1),(cf,1),(ef,1),(be,1),\{b,c,e,f\}\},\\ \{(bf,1),(cf,1),(ec,1),(be,1),\{b,c,e,f\}\},\\ \{(bc,1),(bf,1),(ef,1),(ec,1),\{b,c,e,f\}\},\\ \{(gh,1),(dc,1),(bc,1),(hc,1),\{b,c,g,h\}\}.\end{gathered}$$

A $(3,4+1,0)$*-set packing is*

$$\begin{aligned}&\{\{(ab,1),(ad,1),(dc,1),(bc,1),\{a,b,c,d\}\},\\ &\{(be,1),(bf,1),(cf,1),(ec,1),\{b,c,e,f\}\} \textit{ and}\\ &\{(bc,2),(gb,1),(hc,1),(gh,1),\{b,c,g,h\}\}\}.\end{aligned}$$

This corresponds to the following $(3,4,2)$*-edge-*$\mathcal{H}$*-membership (NISV)*

$$\begin{aligned}\{H_1 &= (\{a,b,c,d\},\{ab,ad,dc,bc\}),\\ H_2 &= (\{b,c,e,f\},\{be,bf,cf,ec\}) \textit{ and}\\ H_3 &= (\{b,c,g,h\},\{bc,gb,gh,hc\})\}.\end{aligned}$$

Lemma 32. *G has a (k,r,t)-edge-$\mathcal{H}$-membership (NISV) if and only if $\mathcal{S}$ has a $(k,r+1,0)$-set packing.*

Proof. We build a $(k,r+1,0)$-set packing $\mathcal{K}_{\mathcal{S}}$ from a (k,r,t)-edge-$\mathcal{H}$-membership (NISV) $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $|E(H_j)|+1 \leq r+1$ elements as follows. The element V in S_j corresponds to the set of vertices $V(H_j)$. Note that $V(H_j) \in \mathcal{V} \subseteq \mathcal{U}$. For each edge $e_i \in E(H_j)$ $(1 \leq i \leq |E(H_j)| \leq m)$, we add an ordered pair $(e_i, l+1)$ to S_j where l corresponds to the number of $\mathcal{H}$-subgraphs in $\{H_1,\dots,H_{j-1}\} \subset \mathcal{K}$ that e_i belongs to. Since each edge is a member of at most t $\mathcal{H}$-subgraphs, $0 \leq l \leq t-1$ and $(e_i, l+1)$ always exists in $\mathcal{U}$. By construction of $\mathcal{S}$, $S_i \in \mathcal{K}_{\mathcal{S}} \subseteq \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_{\mathcal{S}}$ are pairwise disjoint. Since $V(H_i) \neq V(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_{\mathcal{S}}$ will share the same element V. On the other hand, no pair S_i, S_j shares an ordered pair (e,l) for some edge $e \in E(G)$ and some integer $1 \leq l \leq t$. If S_i and S_j share a pair (e,l), this would imply that there are two $\mathcal{H}$-subgraphs H_i and H_j that share the same edge e. Without lost of generality, assume that H_i appears before H_j in $\mathcal{K}$. If (e,l) is contained in both S_i and S_j, then e is a member of $l-1$ $\mathcal{H}$-subgraphs from both $\{\dots,H_i,\dots\}$ and $\{\dots,H_i,\dots,H_j\}$. This is a contradiction since both H_i and H_j contain e.

We construct a (k,r,t)-edge-$\mathcal{H}$-membership (NISV) $\mathcal{K}$ using a $(k,r+1,0)$-set packing $\mathcal{K}_{\mathcal{S}}$. Each set $S_i \in \mathcal{K}_S$ has an element $V_{S_i} \in \mathcal{V}$ which corresponds to a set of vertices in $V(G)$ that forms a subgraph in G that is isomorphic to some $H \in \mathcal{H}$. In addition, we take the edge e_l that appears in each $(e_l, j) \in S_i$ (for $1 \leq l \leq m$ and $j \in [1...t]$) and we add it to a set E_{S_i}. In this way, for each set $S_i \in \mathcal{K}_S$, we add to $\mathcal{K}$ an $\mathcal{H}$-subgraph H_{S_i} with a set of vertices V_{S_i} and a set of edges E_{S_i}, i.e., $H_{S_i} = (V_{S_i}, E_{S_i})$. Since the sets in $\mathcal{K}_S$ are pairwise disjoint, $V(H_{S_i}) \neq V(H_{S_j})$. Now, we need to show that each edge of $E(G)$ is a member of at most t $\mathcal{H}$-subgraphs in $\mathcal{K}$. Assume the contrary for the sake of contradiction. If two $\mathcal{H}$-subgraphs in $\mathcal{K}$ share an edge e, then there are two sets in $\mathcal{K}_S$ where each one has a pair (e, i) and (e, j), respectively. Since the sets in $\mathcal{K}_S$ are disjoint, $i \neq j$. If an edge $e \in E(G)$ is a member of more than t $\mathcal{H}$-subgraphs in $\mathcal{K}$, then there are at least $t + 1$ ordered pairs $(e, 1), \ldots, (e, t + 1)$ contained in $t + 1$ different sets in $\mathcal{K}_S$. However, by construction of $\mathcal{U}$, there are at most t ordered pairs that contain e. □

The size of the constructed instance is upper-bounded by

$$|\mathcal{U}| = |E(G)| \times t + |\mathcal{V}| \leq tn^2 + n^{r(\mathcal{H})} = O(n^{r(\mathcal{H})}) \,.$$

Each set in $\mathcal{S}$ has size $r + 1$ (where $r = m(\mathcal{H})$) and $|\mathcal{S}| \leq t^r |\mathcal{H}| n^{m(\mathcal{H})}$.

We construct a reduced graph G' from a reduced universe $\mathcal{U}' \subseteq \mathcal{U}$ as follows.

Transformation 10. ***Input:*** *$G, \mathcal{U}'$;* ***Output:*** *G'*

We take the end-points of each edge e that appears in each $(e, i) \in \mathcal{U}'$ and the vertices in each $V \in \mathcal{V}'$ and consider the graph G' that is induced by all these in G.

Lemma 33. *G' has a (k, r, t)-$\mathcal{H}$-membership (NISV) if and only if $\mathcal{S}'$ has a $(k, r + 1, 0)$-set packing.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Edge Membership (NISV) problem corresponds to Algorithm 2 with input: G, $\mathcal{H}$, k, $(r + 1)$-*Set Packing*, Transformation 9, kernelization algorithm [1], and Transformation 10. The size of $\mathcal{U}'$ is at most $2r!((r + 1)k - 1)^r$ [1]; hence $|V(G')| = O((r + 1)^r k^r)$. Therefore, we can state:

Theorem 18. *The $\mathcal{H}$-Packing with t-Edge Membership (NISV) problem has a kernel with $O((r + 1)^r k^r)$ vertices, where $r = m(\mathcal{H})$.*

6.2 PPTs for Packing Problems with Pairwise Overlap

In this section, we present kernelization results for the packing problems with pairwise overlap using similar methodology as with the membership problems. First, we introduce a polynomial parametric transformation from the r-Set Packing with t-Overlap problem to the $(\binom{r}{t+1} + 1)$-Set Packing problem. After that we reduce to a kernel the constructed instance of the $(\binom{r}{t+1} + 1)$-Set Packing problem. Finally, we re-interpret this kernel into a proper kernel for the original r-Set Packing with t-Overlap problem.

We also obtain kernel results for our packing graphs with pairwise overlap problems. To that end, we use the framework explained in Chapter 5 that consists on transforming $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap problems into an instance of the r-Set packing with t-Overlap problem. Recall that $r \geq 1$ and $0 \leq t \leq r - 1$ are fixed-constants.

6.2.1 From Packing Sets with Pairwise Overlap to Packing Disjoint Sets

In the r-Set packing with t-Overlap problem, we seek for a collection of at least k distinct sets from a collection $\mathcal{S}$ (drawn from $\mathcal{U}$) such that each pair of sets overlaps at most t elements.

We create an instance of the $(\binom{r}{t+1}+1)$-Set Packing problem, (a universe $\mathcal{U}^T$ and a collection $\mathcal{S}^T$) using an instance of the r-Set Packing with t-Overlap problem as follows. Notice that the sets in $\mathcal{S}$ have size at most r while sets in $\mathcal{S}^T$ has size at most $\binom{r}{t+1}+1$.

Transformation 11. ***Input:*** *$\mathcal{S}, \mathcal{U}$;* ***Output:*** *$\mathcal{S}^T$, $\mathcal{U}^T$*

The universe $\mathcal{U}^T$ equals $\mathcal{S} \cup \{I \mid I \subset \mathcal{U}, |I| = t+1\}$.

The collection $\mathcal{S}^T$ contains all subsets of $\mathcal{U}^T$ with at most $(\binom{r}{t+1}+1)$ elements, of the following form:

$\{S = \{u_1, \ldots, u_{|S|}\}, \{I_1\}, \ldots, \{I_{\binom{|S|}{t+1}}\} \mid S \in \mathcal{S},$ *and each* $I_i \subset S,$ *where* $|I_i| = t+1\}$.

Intuitively, each set in $\mathcal{S}$ will be represented by a set in $\mathcal{S}^T$ with the addition of *forbidden* subsets. Given that a pair of sets in a (k, r, t)-set packing can overlap in at most t elements, we consider every subset of $t+1$ elements of S as forbidden. This would imply that a pair of sets of $\mathcal{S}$ that overlaps in at most t elements corresponds to a pair of sets in $\mathcal{S}^T$ that are pairwise-disjoint or without common forbidden subsets.

Example 6. *Let us consider the next instance of the r-Set Packing with t-Overlap.*

$$\begin{aligned}
\mathcal{U} &= \{a, b, c, d, e, f, g, h\},\\
\mathcal{S} &= \{a, b, c, d\}, \{b, d, e, f\}, \{a, f, g, h\}, \{e, f, g\},\\
\textit{and } k &= 2, \quad t = 1, \quad r = 4.
\end{aligned}$$

The constructed instance for the $(\binom{r}{t+1}+1)$-Set Packing problem is as follows.

$$\begin{aligned}
\mathcal{U}^T = \{&\{a, b, c, d\}, \{b, d, e, f\}, \{a, f, g, h\}, \{e, f, g\},\\
&\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{a, g\}, \{a, h\},\\
&\{b, c\}, \{b, d\}, \{b, e\}, \{b, f\}, \{b, g\}, \{b, h\},\\
&\{c, d\}, \{c, e\}, \{c, f\}, \{c, g\}, \{c, h\},\\
&\{d, e\}, \{d, f\}, \{d, g\}, \{d, h\},\\
&\{e, f\}, \{e, g\}, \{e, h\},\\
&\{f, g\}, \{f, h\}\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S}^T = \{&\{\{a, b, c, d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\},\\
&\{\{b, d, e, f\}, \{b, d\}, \{b, e\}, \{b, f\}, \{d, e\}, \{d, f\}, \{e, f\}\},\\
&\{\{a, f, g, h\}, \{a, f\}, \{a, g\}, \{a, h\}, \{f, g\}, \{f, h\}, \{g, h\}\},\\
&\{\{e, f, g\}, \{e, f\}, \{e, g\}, \{f, g\}\}\}
\end{aligned}$$

A $(2, 7, 0)$-set packing for the constructed instance is

$$\{\{\{a,b,c,d\},\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{b,d\},\{c,d\}\}$$
$$\{\{a,f,g,h\},\{a,f\},\{a,g\},\{a,h\},\{f,g\},\{f,h\},\{g,h\}\}\}$$

This corresponds to the $(2,4,1)$-set packing $\{\{a,b,c,d\},\{a,f,g,h\}\}$.

The size of $\mathcal{U}^T$ is bounded by $n^r + \binom{n}{t+1} = O(n^r + n^{t+1})$. Each set $S^T \in \mathcal{S}^T$ has at most $\binom{r}{t+1}+1$ elements of $\mathcal{U}^T$. Observe that there is exactly one set in $\mathcal{S}^T$ per set in $\mathcal{S}$. Thus, the size of $\mathcal{S}^T$ is $O(n^r)$. In this way,

Lemma 34. *Transformation 11 can be computed in $O(n^r + n^{t+1})$ time.*

Lemma 35. *$\mathcal{S}$ has a (k,r,t)-set packing if and only if $\mathcal{S}^T$ has a $(k, \binom{r}{t+1}+1, 0)$-set packing, i.e., k disjoint sets.*

Proof. We build a $(k, \binom{r}{t+1}+1, 0)$-set packing $\mathcal{K}_\mathcal{S}$ from a (k,r,t)-set packing $\mathcal{K}$. For each $S_j \in \mathcal{K}$, we add a set S_j^T to $\mathcal{K}_\mathcal{S}$. This set S_j^T has at most $\binom{r}{t+1}+1$ elements from $\mathcal{U}^T$ of the following form. One element (the element S in Transformation 11) corresponds to the set S_j, and there are $\binom{r}{t+1}$ elements each one corresponding to a subset of size $t+1$ from S_j. By construction of $\mathcal{S}^T$, each $S_j^T \in \mathcal{K}_\mathcal{S} \subseteq \mathcal{S}^T$. It remains to show that the sets in $\mathcal{K}_\mathcal{S}$ are pairwise disjoint. Since $S_i \neq S_j$ for each pair $S_i, S_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_\mathcal{S}$ will share the same element S. On the other hand, no pair S_i^T, S_j^T shares an element I. If S_i^T and S_j^T would share an element $I = \{u_1, \ldots, u_{t+1}\}$ for some set of $t+1$ elements of $\mathcal{U}$, then there would be two sets S_i and S_j that overlap in at least $t+1$ elements, a contradiction.

We construct a (k,r,t)-set packing $\mathcal{K}$ using a $(k, \binom{r}{t+1}+1, 0)$-set packing $\mathcal{K}_\mathcal{S}$. We simply add to $\mathcal{K}$ a set S_i for each set $S_i^T \in \mathcal{K}_S$. That set S_i is initialized with the element S of the set S_i^T. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $S_i \neq S_j$. Now, we need to show that each pair of sets in $\mathcal{K}$ overlaps in at most t elements. Assume otherwise by contradiction. If a pair of sets in $\mathcal{K}$ shares $t+1$ elements $u_1, \ldots, u_{t+1}$, then there are two sets S_i^T, S_j^T in $\mathcal{K}_S$ that have an element $I = \{u_1 \ldots u_{t+1}\}$ in common. However, this would imply that S_i^T and S_j^T are not pairwise-disjoint, a contradiction. □

Up to this point, we have provided a polynomial parametric transformation for the r-Set packing with t-Overlap problem to an instance of the $(\binom{r}{t+1}+1)$-Set Packing problem. Notice that the parameter k is not altered by our transformation.

We now run the kernelization algorithm [1] for the $(\binom{r}{t+1}+1)$-Set Packing problem on the constructed instance ($\mathcal{U}^T$ and $\mathcal{S}^T$). This algorithm returns a reduced universe with at most $2(r'-1)!(r'k-r')^{r'-1}$ where r' is an upper-bound on the size of the sets of the input instance of that algorithm. By our Transformation 11, each set in $\mathcal{S}^T$ has size at most $r^{t+1}+1$. That is, $r' = r^{t+1}+1$ and Algorithm [1] will leave us with a reduced universe $\mathcal{U'}^T$ with $O((r^{t+1}+1)k - (r^{t+1}+1))^{r^{t+1}}$ elements as well as with a reduced collection $\mathcal{S'}^T$ of sets. The next property follows by [1].

Lemma 36. *The collection $\mathcal{S}$ drawn from $\mathcal{U}$ has a $(k, (\binom{r}{t+1}+1), 0)$-set packing if and only if the set $\mathcal{S'}^T$ drawn from $\mathcal{U'}^T$ has a $(k', (\binom{r}{t+1}+1), 0)$-set packing, where $k' \leq k$.*

Next, we construct the reduced universe $\mathcal{U}'$ and $\mathcal{S}'$ for our original r-Set Packing with t-Overlap problem using ${\mathcal{U}'}^T$ and ${\mathcal{S}'}^T$ as follows.

Transformation 12. ***Input:*** *${\mathcal{S}'}^T$, ${\mathcal{U}'}^T$;* ***Output:*** *$\mathcal{S}'$, $\mathcal{U}'$*

The reduced universe $\mathcal{U}'$ equals to the union of elements u in each $S \in {\mathcal{U}'}^T$.

The reduced collection $\mathcal{S}'$ contains a set which corresponds to the element S in each set in ${\mathcal{S}'}^T$.

$\mathcal{U}'$ and $\mathcal{S}'$ together with k, give the reduced r-Set Packing with t-Overlap instance we are looking for. Since each member of ${\mathcal{U}'}^T$ is a set with at most r elements from $\mathcal{U}$, we can state:

Theorem 19. *The r-Set Packing with t-Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ elements from the given universe.*

By Transformations 11 and 12, each set in $\mathcal{S}'$ is contained in the collection $\mathcal{S} \in {\mathcal{U}'}^T$. In this way, we can conclude:

Theorem 20. *The r-Set Packing with t-Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ sets.*

Our kernelization algorithm for the r-Set Packing with t-Overlap via PPT is summarized in Algorithm 4.

Algorithm 4 Kernelization Algorithm by PPT - r-Set Packing with t-Overlap

```
Input: $\mathcal{U}$ and $\mathcal{S}$
Output: $\mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{S}' \subseteq \mathcal{S}'$
1: Construct a universe $\mathcal{U}^T$ and collection $\mathcal{S}^T$ using Transformation 11.
2: Run the $(\binom{r}{t+1}+1)$-Set Packing kernelization algorithm [1] on $\mathcal{U}^T$ and $\mathcal{S}^T$.
3: Using Transformation 12, re-interpret the kernel ${\mathcal{U}'}^T, {\mathcal{S}'}^T$ as reduced $\mathcal{U}'$, $\mathcal{S}'$, respectively.
4: Return $\mathcal{U}'$, $\mathcal{S}'$.
```

Packing Weighted Sets with Pairwise Overlap using PPTs

We discuss in this section how to reduce to a kernel the weighted version of the r-Set packing with t-Overlap by using our PPTs. Let us introduce first the definition of the *weighted r-Set packing with t-Overlap problem*. In this problem, every set in $\mathcal{S}$ has associated a weight (a real number), and the goal is to find a (k, r, t)-set packing of maximum weight.

We construct an instance of the r-Set Packing problem (packing disjoint sets) using Transformation 11. Given that each set $S \in \mathcal{S}$ corresponds to a set $S' \in \mathcal{S}^T$, we assign the weight of S to S'. After that in similar way as with the weighted r-Set Packing with t-Membership, we run the kernelization algorithm of Zehavi [140] (which reduces the weighted version of the r-Set Packing problem) in our constructed instance. Finally, we re-intepret back the kernel returned by Algorithm [140] as a kernel for the original weighted r-Set Packing with t-Overlap using Transformation 12. Algorithm [140] returns a reduced kernel of size $O(e^{r'} r' k^{r'})$ where $r' = r^{t+1} + 1$ (on the number of sets), we can hence conclude:

Theorem 21. *The weighted r-Set Packing with t-Overlap has a problem kernel with $O(e^{r^{t+1}+1}(r^{t+1}+1)k^{r^{t+1}+1})$ sets.*

6.2.2 A Kernel Reduction for the $\mathcal{H}$-Packing with t-Overlap Problem

In the $\mathcal{H}$-Packing with t-Overlap, we seek for at least k $\mathcal{H}$-subgraphs of G such that each pair of those subgraphs overlaps in at most t vertices. To achieve a reduction to the kernel for this problem, we will use the kernelization framework described in Section 5.2.1. In that framework, we show how to reduce the $\mathcal{H}$-Packing with t-Overlap to a kernel by transforming it to the r-Set Packing with t-Overlap. Also, it is assumed the existence of a kernelization algorithm KernelAlg-O for the r-Set Packing with t-Overlap. In this chapter, Algorithm 4 takes the place of KernelAlg-O.

We start by constructing an instance of the r-Set Packing with t-Overlap with the instance of the $\mathcal{H}$-Packing with t-Overlap with Transformation 1, i.e., a universe $\mathcal{U}$ and a collection of sets $\mathcal{S}$. The parameter k is not altered and $r = r(\mathcal{H})$. After that we reduce to a kernel the constructed instance of the r-Set Packing with t-Overlap with Algorithm 4. That is, we obtain a reduced universe $\mathcal{U}'$ and a reduced collection of sets $\mathcal{S}'$ as well. Finally, we re-interpret $\mathcal{U}'$ as a reduced instance of the $\mathcal{H}$-Packing with t-Overlap (Transformation 2), i.e., a reduced graph G'.

By Transformation 2, $G' = G[\mathcal{U}']$, and by Algorithm 4, $|\mathcal{U}| = O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$. By Lemma 2, we can hence conclude.

Theorem 22. *$\mathcal{H}$-Packing with t-Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = r(\mathcal{H})$.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Overlap problem described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing with t-Overlap*, Transformation 1, Algorithm 4, and Transformation 2.

In similar fashion, we obtain the next result for the induced version of the problem.

Theorem 23. *Induced-$\mathcal{H}$-Packing with t-Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = r(\mathcal{H})$.*

In Appendix A.2.1, we include a polynomial parametric transformation from the $\mathcal{H}$-Packing with t-Overlap problem to the $(\binom{r}{t+1}+1)$-Set Packing, where $r = r(\mathcal{H})$ (instead of to the r-Set Packing with t-Overlap). That transformation leads to the same kernel problem size as in Theorem 22.

6.2.3 A Kernel Reduction for the $\mathcal{H}$-Packing with t-Edge Overlap and t-Edge Overlap (NISV) Problems

The $\mathcal{H}$-Packing with t-Edge Overlap seeks in G for a collection of at least k $\mathcal{H}$-subgraphs where each pair of them overlaps in at most t edges. We will achieve a kernel reduction for this problem with similar methodology as for the vertex-version. This methodology is fully detailed in Section 5.2.2.

Using an instance of the $\mathcal{H}$-Packing with t-Edge Overlap, we construct an instance of the r-Set Packing with t-Overlap with Transformation 3, i.e., a universe $\mathcal{U}$ and collection of sets $\mathcal{S}$. The parameter k remains the same, and $r = m(\mathcal{H})$. We reduce to a kernel the constructed instance of the r-Set Packing with t-Overlap using Algorithm 4 obtaining in this way a reduced universe $\mathcal{U}'$ and a reduced collection $\mathcal{S}'$. Finally with Transformation 4, we re-interpret this reduced universe as a reduced instance of the $\mathcal{H}$-Packing with

t-Edge Overlap. That is, a reduced graph G', where $G' = G[V(\mathcal{U}')]$. Recall that G does not have isolated vertices (Lemma 23).

By Algorithm 4, $|\mathcal{U}'| = O((r^{(t+1)} + 1)^{r^{t+1}} k^{r^{t+1}})$. Lemma 25 allows us to conclude.

Theorem 24. *The $\mathcal{H}$-Packing with t-Edge-Overlap has a problem kernel with $O((r^{t+1} + 1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = m(\mathcal{H})$.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Edge Overlap problem that we just described corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing with t-Overlap*, Transformation 3, Algorithm 4, and Transformation 4.

In Appendix A.2.2, we include a polynomial parametric transformation from the $\mathcal{H}$-Packing with t-Edge Overlap problem to the $(\binom{r}{t+1} + 1)$-Set Packing, where $r = m(\mathcal{H})$ (instead of to the r-Set Packing with t-Overlap). That transformation leads to the same kernel size as in Theorem 24.

The $\mathcal{H}$-Packing with t-Edge Overlap (NISV) Problem

Similarly to the $\mathcal{H}$-Packing with t-Edge Membership problem (NISV), we present a stricter definition of the $\mathcal{H}$-Packing with t-Edge Overlap problem to avoid $\mathcal{H}$-subgraphs in a (k, m, t)-edge-$\mathcal{H}$-overlap with identical sets of vertices. This is equivalent to adding the condition $V(H_i) \neq V(H_j)$ to the $\mathcal{H}$-Packing with t-Edge Overlap problem definition. We indicate this variant by adding NISV (non identical set of vertices) to the problem name.

The $\mathcal{H}$-Packing with t-Edge Overlap (NISV) problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, m, t)-edge-$\mathcal{H}$-packing (NISV), i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to a graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|E(H_i) \cap E(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

We cannot transform the NISV variant to an instance of the r-Set Packing with t-Overlap as we did with the original version of the $\mathcal{H}$-Packing with t-Edge Overlap problem. Lemma 24 does not hold for this version of the problem. This is because not every solution for the r-Set Packing with t-Overlap problem can be transformed into a solution of the $\mathcal{H}$-Packing with t-Edge Overlap (NISV) problem.

Therefore, we reduce the $\mathcal{H}$-Packing with t-Edge Overlap (NISV) problem directly to $(\binom{r}{t+1} + 1)$-Set Packing (disjoint sets), instead of reducing to r-Set Packing with t-Overlap. Before introducing our transformation, the reader is referred to Chapter 2 for a quick reminder of the definitions of $\mathcal{V}$ and $\mathcal{E}$.

Transformation 13. ***Input:*** *G, $\mathcal{V}$, $\mathcal{E}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r*

Let $r = m(\mathcal{H})$.

The universe $\mathcal{U}$ is defined as $\mathcal{V} \cup \{I \mid I \subset E(G), |I| = t + 1\}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ with at most $(\binom{r}{t+1} + 1)$ elements, of the following form:

$\{V(E), \{I_1\}, \ldots, \{I_{\binom{|E|}{t+1}}\} \mid E \in \mathcal{E}$ *and each* $I_i \subset E$ *where* $|I_i| = t + 1\}$.

The idea is that each $\mathcal{H}$-subgraph in G will be represented by a set S in $\mathcal{S}$. Since any pair of $\mathcal{H}$-subgraphs in a solution cannot overlap in more than t edges, we add all possible subsets of $t+1$ edges to a set S. This would imply that a pair of $\mathcal{H}$-subgraphs that overlaps in at most t edges corresponds to a pair of sets in $\mathcal{S}$ that are pairwise-disjoint. All $\mathcal{H}$-subgraphs with the same set of vertices will correspond to sets in $\mathcal{S}$ that share the element $V(E)$.

Example 7. *Consider the graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $k = 2$, $r = 4$, and $t = 2$. The instance of the $(\binom{r}{t+1} + 1)$-Set Packing problem is as follows.*

$$\begin{aligned}
&\mathcal{V} = \{\{a,b,c,d\},\{b,c,e,f\},\{b,c,g,h\}\}\\
&\mathcal{U} = \mathcal{V}\ \cup\\
&\quad \{ab,ad,bc\},\{ab,ad,dc\},\{ab,ad,be\},\ldots,\\
&\quad \{gb,gh,hc\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} = \{&\{S_1 = \{a,b,c,d\},\{ab,ad,bc\},\{ab,ad,cd\},\{ad,bc,cd\},\{ab,bc,cd\}\},\\
&\{S_2 = \{b,c,e,f\},\{bc,gh,hc\},\{bc,gh,bg\},\{gh,hc,bg\},\{bc,hc,gc\}\},\\
&\{S_3 = \{b,c,g,h\},\{bc,cf,ef\},\{bc,cf,eb\},\{cf,ef,eb\},\{bc,ef,eb\}\},\\
&\{S_4 = \{b,c,g,h\},\{bc,cf,ef\},\{bc,cf,eb\},\{cf,ef,eb\},\{bc,ef,eb\},\ldots,\{eb,bf,ce\}\}\}
\end{aligned}$$

A $(2,21,0)$-set packing for this instance is $\{S_1, S_4\}$. This corresponds to the $(2,4,2)$-$\mathcal{H}$-packing $H_1 = (\{a,b,c,d\},\{ab,ad,bc,dc\}))$ and $H_2 = (\{b,e,c,f\},\{be,cf,ef,eb,bf,ce\}$.

The size of our constructed instance is bounded by $|\mathcal{U}| \le n^r + \binom{n^2}{t+1} = O(n^r + n^{2(t+1)})$. Each set $S \in \mathcal{S}$ has at most $\binom{r}{t+1} + 1$ elements. One of these elements corresponds to a set of $\mathcal{S}$ while the other $\binom{r}{t+1}$ elements correspond to subsets of $t+1$ elements from $\mathcal{U}$. There is exactly one set per $\mathcal{H}$-subgraph in G. Since $|\mathcal{H}_G| = O(n^{m(\mathcal{H})})$, the size of $\mathcal{S}$ is $O(n^r)$. Recall that $r = m(\mathcal{H})$.

Lemma 37. *Transformation 13 can be computed in $O(n^r + n^{2(t+1)})$ time, where $r = m(\mathcal{H})$.*

Lemma 38. *G has a (k,r,t)-edge-$\mathcal{H}$-packing if and only if $\mathcal{S}$ has $(k, (\binom{r}{t+1} + 1, 0)$-set packing (i.e., at least k disjoint sets).*

Proof. We build a $(k, \binom{r}{t+1} + 1, t)$-set packing $\mathcal{K}_\mathcal{S}$ from a (k,r,t)-edge-$\mathcal{H}$-packing $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $\binom{|E(H_j)|}{t+1} + 1$ elements as follows. The element $V(E)$ in S_j corresponds to the set of vertices $V(H_j)$. We add an element to S_j for each subset of $t+1$ edges from $E(H_j)$. Thus, we are adding $\binom{|E(H_j)|}{t+1}$ sets each of size $t+1$ to S_j. By construction of $\mathcal{S}$, each $S_j \in \mathcal{K}_\mathcal{S} \subseteq \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_\mathcal{S}$ are pairwise disjoint. Since $V(H_i) \neq V(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_\mathcal{S}$ will share the element $V(E)$. On the other hand, no pair S_i, S_j shares a set I with $t+1$ elements. Otherwise, there would be two $\mathcal{H}$-subgraphs H_i and H_j that overlap in at least $t+1$ edges, a contradiction.

We construct a (k,r,t)-edge-$\mathcal{H}$-packing $\mathcal{K}$ using a $(k, \binom{r}{t+1}+1, t)$-set packing $\mathcal{K}_{\mathcal{S}}$. Each set $S_i \in \mathcal{K}_S$ has an element $V_{S_i} \in \mathcal{V}$ which corresponds to a set of vertices in $V(G)$ that forms an $\mathcal{H}$-subgraph H_{S_i}. Furthermore, the union of the elements I in S_i, (denoted as E_{S_i}), corresponds to the edges of the $\mathcal{H}$-subgraph H_{S_i}. In this way, we add to $\mathcal{K}$ an $\mathcal{H}$-subgraph H_{S_i} with edges E_{S_i} for each set $S_i \in \mathcal{K}_S$. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $V(H_{S_i}) \neq V(H_{S_j})$. Now, we need to show that each pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ overlaps in at most t edges. Assume otherwise by contradiction. If two $\mathcal{H}$-subgraphs in $\mathcal{K}$ share $t+1$ edges $e_1, \ldots, e_{t+1}$, then there are two sets S_i, S_j in $\mathcal{K}_S$ that has an element $I = \{e_1 \ldots e_{t+1}\}$ in common. However, this would imply that S_i and S_j are not pairwise-disjoint, a contradiction. □

We could now run a kernelization algorithm for the $(\binom{r}{t+1}+1)$-Set Packing problem which will leave us with a reduced universe $\mathcal{U}'$. We construct a reduced graph G' from a reduced universe $\mathcal{U}' \subseteq \mathcal{U}$ as follows.

Transformation 14. ***Input:*** *$G, \mathcal{U}'$;* ***Output:*** *G'*

We take the end-points of each edge e that appears in $\mathcal{U}'$ and the vertices in each $V \in \mathcal{V}'$ and consider the graph G' that is induced by all these in G.

Lemma 39. *G' has a (k,r,t)-$\mathcal{H}$-packing (NISV) if and only if $\mathcal{S}'$ has a $(k, r+1, 0)$-set packing.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Edge Overlap (NISV) problem corresponds to Algorithm 2 with input: G, $\mathcal{H}$, k, $(r+1)$-*Set Packing*, Transformation 13, kernelization algorithm [1], and Transformation 14. The size of $\mathcal{U}'$ is $O(((r^{t+1}+1)k - (r^{t+1}+1))^{r^{t+1}})$ [1]; hence $|V(G')| = O(r^{r^{t+1}} k^{r^{t+1}})$. Therefore, we can state:

Theorem 25. *The $\mathcal{H}$-Packing with t-Edge Overlap (NISV) has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = m(\mathcal{H})$.*

6.3 Conclusions

We introduced in this chapter polynomial parametric transformations from the problem of packing sets with membership and pairwise overlap to the problem of packing disjoint sets. These transformations allowed us to obtain polynomial size kernels for our r-Set Packing with t-Membership and t-Overlap problems and for the $\mathcal{H}$-Packing with t-Membership and t-Overlap problems as well. We also discussed how to obtain polynomial kernel sizes for the weighted versions of our set packing problems by slightly changing our PPTs.

In the case of the r-Set Packing with t-Membership problem, our kernel size $O((r+1)^r k^r)$ is very close to the smallest kernel size for the r-Set Packing problem $O(r^r k^{r-1})$ [1]. Unfortunately, our PPT does not have the same effect in the kernel size for the r-Set Packing with t-Overlap problem. Only for $t = 0$, our $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ kernel is close to the kernel size of the disjoint version. In addition, our kernel grows as the size of the overlap value t is increased. This does not agree with the intuition that we should have a smaller kernel when more overlap is allowed in the solution.

Despite that our kernels for the pairwise overlap problems are not as good as the kernels for the membership problems, all of our kernels are still polynomial size. In addition, our PPTs are relatively simple, and the analysis of correctness is fairly intuitive once the transformations are understood. Therefore, we believe that the use of polynomial parametric transformations could consists on an effective and fast technique to obtain kernelization algorithms for other parameterized problems as well.

Chapter 7

Improved Kernel Results by Classical Reduction

As we learned in the previous chapter, kernelization by polynomial parametric transformations may provide a fast and effective way to obtain kernel results for a parameterized problem. However, we also noticed that this simplicity comes with a price: the kernel results may not be as a good as the ones of the problem that we are transforming to. Therefore in this chapter, we attempt to improve our previous kernel results by tackling the r-Set Packing with t-Overlap and t-Membership problems directly.

The kernelization algorithms for r-Set Packing with t-Overlap and t-Membership that we develop here are based on the classical approach of reduction [44, 117]. That is, in polynomial time, apply a set of reduction rules to an input instance in such a way that the reduced instance has a solution if and only the original instance has one. In addition, the reduced instance has size $O(f(k))$, for some computable function f, where k is a parameter. Recall that an instance of our set packing problems is a collection of sets $\mathcal{S}$, a universe $\mathcal{U}$, and a parameter k. In general, our kernelization algorithms reduce the input instances of our set packing problems by removing sets of $\mathcal{S}$ and elements from $\mathcal{U}$ that are not needed to form a solution. These algorithms are inspired on the ideas of kernelization algorithms for packing disjoint graphs [53, 114].

The organization of this chapter is as follows. In Section 7.2, we show our kernel results for all the pairwise overlap problems: r-Set Packing with t-Overlap, $\mathcal{H}$-Packing with t-Overlap, and $\mathcal{H}$-Packing with t-Edge Overlap. We develop a kernelization algorithm for the r-Set Packing with t-Overlap problem in Section 7.2.1. This algorithm leaves us with a reduced universe with $O(r^r k^{r-t-1})$ elements. In addition in Section 7.2.2, we apply the kernelization framework of Chapter 5 altogether with the above algorithm to obtain $O(r^r k^{r-t-1})$ kernels for the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap where $r = r(\mathcal{H})$ and $r = m(\mathcal{H})$, respectively. Note our kernel size improvement for all our pairwise overlap problems with respect to the $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ kernels obtained with PPTs in the previous chapter (again $r = r(\mathcal{H})$ and $r = m(\mathcal{H})$ for vertex-overlap and edge-overlap, respectively).

Section 7.3 provides our results for our membership problems: r-Set Packing with t-Membership, $\mathcal{H}$-Packing with t-Membership, and $\mathcal{H}$-Packing with t-Edge Membership. In Section 7.3.1, we obtain a kernelization algorithm that reduces the r-Set Packing with t-Membership to an $O\big(r^3 \binom{r}{t}^{r-3} k^{r-1}\big)$ kernel.

Problem	Kernel Size	Section
r-Set Packing with t-Overlap	$O(r^r k^{r-t-1})$ elements	7.2.1
$\mathcal{H}$-Packing with t-Overlap	$O(r(\mathcal{H})^{r(\mathcal{H})} k^{r(\mathcal{H})-t-1})$ vertices	7.2.2
$\mathcal{H}$-Packing with t-Edge Overlap	$O(m(\mathcal{H})^{m(\mathcal{H})} k^{m(\mathcal{H})-t-1})$ vertices	7.2.2
r-Set Packing with t-Membership	$O\big(r^3 \big(\frac{r}{t}\big)^{r-3} k^{r-1}\big)$ elements	7.3.1
$\mathcal{H}$-Packing with t-Membership	$O\big(r(\mathcal{H})^3 \big(\frac{r(\mathcal{H})}{t}\big)^{r(\mathcal{H})-3} k^{r(\mathcal{H})-1}\big)$ vertices	7.3.2
$\mathcal{H}$-Packing with t-Edge Membership	$O\big(m(\mathcal{H})^3 \big(\frac{m(\mathcal{H})}{t}\big)^{m(\mathcal{H})-3} k^{m(\mathcal{H})-1}\big)$ vertices	7.3.2

Table 7.1: Summary of results presented in Chapter 7

This algorithm together with the kernelization framework in Section 5.1 let us to obtain $O\big(r^3 \big(\frac{r}{t}\big)^{r-3} k^{r-1}\big)$ kernels as well for the $\mathcal{H}$-Packing with t-Membership, and $\mathcal{H}$-Packing with t-Edge Membership problems with $r = r(\mathcal{H})$ and $r = m(\mathcal{H})$, respectively. These kernel results show a more modest but still an improvement with respect to the $O((r+1)^r k^r)$ kernels obtained with PPTs ($r = r(\mathcal{H})$ and $r = m(\mathcal{H})$ for vertex-membership and edge-membership, respectively). Table 7.1 summarizes the results obtained in this chapter.

7.1 A Common Ground

The algorithms that we develop in this chapter follow a common framework, but they still have significant differences that lead to an independent analysis of correctness and kernel sizes. Therefore, before providing the specifics, we will give a general overview of such a common base.

The input instance for the r-Set Packing with t-Membership and t-Overlap problems consists of a universe $\mathcal{U}$ and a collection of sets $\mathcal{S}$ (drawn from $\mathcal{U}$), and a parameter k. The output of the kernelization algorithms is $\mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{S} \subseteq \mathcal{S}'$ where $|\mathcal{U}'| = O(f(k))$, for some computable function f. These algorithms reduce an instance of these problems to a kernel in two major and independent steps: by removing *extra sets* from $\mathcal{S}$ and by removing *extra elements* from $\mathcal{U}$. A set $S \in \mathcal{S}$ or element $u \in \mathcal{U}$ is *extra*, if there is a solution (for either problem) that does not contain S or u.

Intuitively to determine which sets are extra, we count the number of sets in $\mathcal{S}$ that intersect in a specific subset of elements of $\mathcal{U}$. At most a specific number of those sets (which varies according to the type of problem) is kept in $\mathcal{S}$ to find a solution. The rest of the sets (which is chosen arbitrarily) are extra. On the other hand, to determine which elements are extra, we need to compute an auxiliary bipartite graph and perform a reduction rule based on a matching problem, which again will vary according to the type of problem we are dealing with.

Algorithm 5 depicts the general framework behind Algorithms 6 and 9 which are kernelization algorithms for the r-Set packing with t-Overlap and t-Membership, respectively. First, a maximal collection of sets $\mathcal{R} \subseteq \mathcal{S}$ such that every pair of sets overlaps in $\mathcal{R}$ in at most $r-2$ elements is computed. After that with the use of a tailored algorithm for either the t-Overlap or t-Membership problem, extra sets are identified in $\mathcal{R}$. The collections $\mathcal{S}$ and $\mathcal{R}$ are updated accordingly. Once there are no more extra sets in $\mathcal{S}$, the second reduction happens in Line 6. An auxiliary bipartite graph B is constructed using $\mathcal{U} \backslash val(\mathcal{R})$ and $\mathcal{R}$. Once again, depending on the type of problem a reduction rule based on a matching problem is applied to B. In the case of t-Overlap will be a maximum matching while for t-Membership is a cardinality

t-matching (see Chapter 2). The reduced universe $\mathcal{U}'$ and $\mathcal{S}'$ are obtained after the computation of that rule.

Algorithm 5 Kernelization by Reduction - Framework

```
Input: An instance $\mathcal{U}, \mathcal{S}$
Output: A reduced instance $\mathcal{U}', \mathcal{S}'$
1: repeat
2:     Compute a maximal collection of sets $\mathcal{R} \subseteq \mathcal{S}$ such that every pair of sets in $\mathcal{R}$ overlaps in at most
       $r - 2$ elements.
3:     Identify extra sets $\mathcal{E}$ in $\mathcal{R}$ and make $\mathcal{R} = \mathcal{R} \backslash \mathcal{E}$ and $\mathcal{S} = \mathcal{S} \backslash \mathcal{E}$.
4: until No more extra sets are identified, i.e., $\mathcal{E} = \emptyset$.
5: Construct an auxiliary bipartite graph $B$ from $\mathcal{U} \backslash val(\mathcal{R})$ and $\mathcal{R}$.
6: Remove extra elements in $\mathcal{U} \backslash val(\mathcal{R})$ by computing a matching problem-based reduction rule on $B$.
7: Return $\mathcal{U}'$ and $\mathcal{S}'$ as outputted by the matching-based reduction rule.
```

In the next sections, we will develop the details behind each kernelization algorithm separately.

7.2 Improved Kernel Results for Packing Problems with Pairwise Overlap

In this section, we focus on our packing problems with pairwise overlap: r-Set Packing with t-Overlap, $\mathcal{H}$-Packing with t-Overlap, and $\mathcal{H}$-Packing with t-Edge Overlap. We start in Section 7.2.1 by developing a kernelization algorithm (Algorithm 6) for the r-Set Packing with t-Overlap problem. This algorithm permits us to improve the kernel result obtained in Section 6.2.1 from $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ to $O(r^r k^{r-t-1})$. In Section 7.2.2, we will obtain improved kernel results for the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap by applying our kernelization framework from Chapter 5.

7.2.1 A Reduction Algorithm for Packing Sets with Pairwise Overlap

Recall from the definition of the r-Set Packing with t-Overlap problem that the goal of this problem is to seek a collection of at least k sets from $\mathcal{S}$ such that each pair of sets on that collection overlaps in at most t elements, if one exists.

We assume that each set in $\mathcal{S}$ has size at least $t+1$. Otherwise, we can add the sets with size at most t straight to a (k, r, t)-set packing and decrease the parameter k by the number of those sets. We start with a simple reduction rule.

Reduction Rule 4. *Remove any element of $\mathcal{U}$ that is not contained in at least one set of $\mathcal{S}$.*

The next theorem shows that the r-Set packing with t-Overlap problem is easy for $t \geq r-1$. Thus, for the remainder of this section, we will assume that $t \leq r-2$.

Theorem 26. *The r-Set Packing with t-Overlap can be solved in polynomial time for $t \geq r-1$.*

Proof. The algorithm consists on simply evaluating the size of $\mathcal{S}$. That is, if $|\mathcal{S}| \geq k$ then $\mathcal{S}$ is a (k, r, t)-set packing; in the other case, $\mathcal{S}$ does not have a solution. The correctness follows because every pair of sets in $\mathcal{S}$ overlaps in at most $r - 1$ elements (even if $S' \subset S$ for some $S', S \in \mathcal{S}$). Recall that the the overlap restriction asks only for $|S_i \cap S_j| \leq t$. □

We will divide the task of explaining our kernelization algorithm for the r-Set Packing with t-Overlap problem in three steps: first, we focus on explaining the algorithm itself; next, we prove that the reduced instance by our algorithm has a solution if and only if the original instance has one; and finally, we show that the reduced instance is actually a kernel of the original instance.

The Algorithm

Algorithm 6 reduces an instance of the r-Set Packing with t-Overlap problem to a kernel in two separate steps. First, the goal is to identify sets of $\mathcal{S}$ that may not be needed (*extra*) to form a (k, r, t)-set packing. After that, *unnecessary* elements are removed from $\mathcal{U}$ by triggering a reduction.

Let us explain the ideas behind the first step of reduction which occurs in Lines 3-9. In Line 4, we compute a maximal $(r, r-2)$-set packing $\mathcal{R}$ of $\mathcal{S}$, i.e., a maximal collection of sets from $\mathcal{S}$ such that every pair of sets in $\mathcal{R}$ overlaps in at most $r - 2$ elements. See Example 8 for an illustration of this maximal solution.

Example 8. *Consider the following instance of the r-Set Packing with t-Overlap problem, where $r = 4$, $t = 2$, and $k = 2$.*

$$\mathcal{U} = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y\} \text{ and}$$
$$\mathcal{S} = \{\{a, b, c, e\}, \{b, c, d, e\}, \{e, f, g, i\}, \{a, e, f, i\}, \{e, g, i, h\}, \{i, j, n, m\},$$
$$\{i, j, m, k\}, \{i, j, m, l\}, \{o, q, n, p\}, \{q, p, r, s\}, \{q, p, t, u\}, \{q, p, u, v\}, \{q, p, v, w\},$$
$$\{q, p, x, y\}\}.$$

Figure 7.1 shows the hypergraph constructed from $\mathcal{U}$ and $\mathcal{S}$.

A maximal $(r, r-2)$-set packing is

$$\begin{aligned}\mathcal{R} = \{&\{b, c, d, e\}, \{e, f, g, i\}, \{i, j, n, m\},\\ &\{o, n, p, q\}, \{q, p, r, s\}, \{q, p, t, u\},\\ &\{q, p, v, w\}, \{q, p, x, y\}\}.\end{aligned}$$

This maximal set is highlighted with bold lines in Figure 7.1.

Algorithm 7 (applied in Line 6 in Algorithm 6) identifies *extra* sets in $\mathcal{R} \subseteq \mathcal{S}$. We could just use $\mathcal{R} = \mathcal{S}$ as input of Algorithm 7, (i.e., a maximal $(r, r-1)$-set packing instead). However, a smaller set $\mathcal{R}$ will lead to a smaller problem kernel.

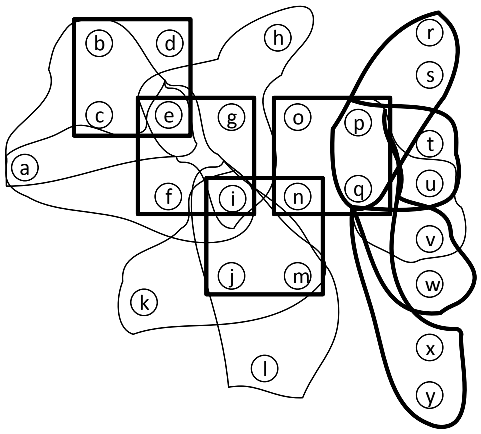

Figure 7.1: The hypergraph constructed with $\mathcal{S}$ and $\mathcal{U}$ from Example 8

Lines 3-9 in Algorithm 6 basically keep reducing the set $\mathcal{R}$ using Algorithm 7 while there are no more sets from $\mathcal{S}\backslash(\mathcal{R}\cup\mathcal{E})$ to add to $\mathcal{R}$. We need to run Algorithm 7 possibly more than once in order to preserve the maximality of $\mathcal{R}$. That is, after Line 4, each set in $\mathcal{S}\backslash(\mathcal{R}\cup\mathcal{E})$ overlaps in $r-1$ elements with at least one set in $\mathcal{R}$. After Line 7, $\mathcal{R}$ is reduced by $\mathcal{E}'$, and $\mathcal{E}$ is updated with $\mathcal{E}'$. Therefore, all sets in $\mathcal{S}\backslash(\mathcal{R}\cup\mathcal{E})$ that were overlapping in $r-1$ elements only with sets in $\mathcal{E}'$ can now be added to $\mathcal{R}$.

Before explaining the rest of Algorithm 6, we will detail how Algorithm 7 actually finds the extra sets of $\mathcal{R}$. The basic idea of Algorithm 7 is that if there is more than a specific number of members of $\mathcal{S}$ that pairwise overlap in exactly the same subset of elements $P \subseteq \mathcal{U}$, then some of those members are extra. This approach has been applied to obtain kernels for the H-Packing problem [53, 114]. To our knowledge, this is the first generalization for packing problems with arbitrary overlap.

For each set S in $\mathcal{R}$, Algorithm 7 takes every subset of elements $P \subsetneq S$ from sizes from ini to $t+1$ in decreasing order (Lines 3-5). The variable ini is defined as the maximum overlap value between any pair of sets in $\mathcal{R}$ (Line 1). Thus, given the construction of $\mathcal{R}$ in Algorithm 6, $ini = r-2$. Note that $|S| > i$ to run Lines 5-13 of Algorithm 7; otherwise S will be considered at a later iteration of the for-loop of Line 4. For every P, we count the number of sets in $\mathcal{R}$ that contain P (collected in $\mathcal{P}$, Line 6) and we keep in $\mathcal{R}$ a specific number of them (Line 7). The rest of the sets will be removed (Lines 8-12). The size of P is determined by the variable i in Line 3, and the threshold $f(i)$ is defined as $f(i) = (r-t)(k-1)f(i+1)+1$ where $f(ini+1)$ is initialized to one (Line 2). The function $f(i)$ basically bounds the number of sets in $\mathcal{R}$ that contains a subset P with i elements. Observe that $f(i) = (r-t)(k-1)f(i+1)+1 = \sum_{j=0}^{ini-i+1}[(r-t)(k-1)]^j$. Algorithm 7 returns $\mathcal{E}$ which is the set of extra sets in $\mathcal{R}$ (Line 16).

Example 9. *Consider iteration $i = r-2$, $S = \{o,n,p,q\}$, and $P = \{p,q\}$ in Algorithm 7 for the instance of Example 8. In this case,*

$$\mathcal{P} = \{\{o,n,p,q\},\{q,p,r,s\},\{q,p,t,u\},\{q,p,v,w\},\{q,p,x,y\}\}.$$

Since $k=2$, for $i=r-2$, $f(r-2)=3$. Thus, two sets of $\mathcal{P}$ will be removed in Line 9.

Algorithm 6 Kernelization Algorithm by Reduction - r-Set Packing with t-Overlap

```
Input: An instance $\mathcal{U}, \mathcal{S}$
Output: A reduced instance $\mathcal{U}', \mathcal{S}'$

1:  Apply Reduction Rule 4
2:  $\mathcal{R} = \emptyset$, $\mathcal{E} = \emptyset$
3:  repeat
4:    Greedily add sets from $\mathcal{S}\backslash(\mathcal{R} \cup \mathcal{E})$ to $\mathcal{R}$ such that every pair of sets in $\mathcal{R}$ overlaps in at most $r-2$
      elements (i.e., a maximal $(r, r-2)$-set packing).
5:    if at least one set was added to $\mathcal{R}$ then
6:      $\mathcal{E}'$ = Algorithm 7($\mathcal{R}$)
7:      $\mathcal{R} = \mathcal{R}\backslash\mathcal{E}'$, $\mathcal{E} = \mathcal{E} \cup \mathcal{E}'$
8:    end if
9:  until no more sets have been added to $\mathcal{R}$
10: Reduce $\mathcal{S} = \mathcal{S}\backslash\mathcal{E}$ and re-apply Reduction Rule 4
11: Compute a maximal $(r, t)$-set packing $\mathcal{M}$ of $\mathcal{R}$
12: if $|\mathcal{M}| \geq k$ then
13:   Let $\mathcal{S}'$ be any subset of $\mathcal{M}$ of size $k$
14:   Return $\mathcal{U}$ and $\mathcal{S}'$
15: end if
16: Let $\mathcal{U}'$ and $\mathcal{S}'$ be the reduced universe and collection of sets, respectively, after applying Reduction
    Rule 5.
17: Return $\mathcal{U}'$ and $\mathcal{S}'$
```

Algorithm 7 Extra Sets Reduction - r-Set Packing with t-Overlap

```
Input: A collection $\mathcal{R} \subseteq \mathcal{S}$ of sets
Output: A collection $\mathcal{E} \subseteq \mathcal{R}$ of sets
1:  $ini = \max\{|S_i \cap S_j| : S_i, S_j \in \mathcal{R}, i \neq j\}$
2:  $f(ini + 1) = 1$, $\mathcal{E} = \emptyset$
3:  for $i = ini$ downto $t + 1$ do
4:    for each $S \in \mathcal{R}$ such that $|S| > i$ do
5:      for each $P \subsetneq S$ where $|P| = i$ do
6:        $\mathcal{P} = \{S' \in \mathcal{R} : S' \supsetneq P\}$
7:        $f(i) = (r - t)(k - 1)f(i + 1) + 1$
8:        if $|\mathcal{P}| > f(i)$ then
9:          Choose any $\mathcal{P}' \subset \mathcal{P}$ of size $f(i)$
10:         Set $\mathcal{E} \leftarrow \mathcal{E} \cup (\mathcal{P}\backslash\mathcal{P}')$ (extra sets)
11:         $\mathcal{R} \leftarrow \mathcal{R}\backslash(\mathcal{P}\backslash\mathcal{P}')$
12:       end if
13:     end for
14:   end for
15: end for
16: Return $\mathcal{E}$
```

Let us now resume explaining the rest of Algorithm 6. After no more extra sets can be found, we reduce $\mathcal{S}$ by removing $\mathcal{E}$ in Line 10 of Algorithm 6. As a consequence, we re-apply Reduction Rule 4. Then, we compute a maximal (r,t)-set packing in $\mathcal{R}$. As we will see in proof of Lemma 45, this maximal solution will help us to determine an upper-bound for the number of sets in $\mathcal{R}$. From now on, we assume that the maximal solution was not a (k,r,t)-set packing and Algorithm 6 continues executing.

We now proceed to explain the second step of reduction of Algorithm 6 which occurs in Line 16. An element $u \in \mathcal{U}$ is *extra*, if there is a (k,r,t)-set packing of $\mathcal{S}$ where $u \notin S$ for each set $S \in \mathcal{K}$. We will identify extra elements in $\mathcal{U}\backslash val(\mathcal{R})$, denoted as O henceforth.

To reduce O, we first construct an auxiliary bipartite graph $B = (V_O, V_P, E)$ as in Algorithm 8. After that, we compute a maximum matching in B, the unmatched vertices in the partition V_O correspond to extra elements in O and will be removed from $\mathcal{U}$. As a consequence the collection $\mathcal{S}$ will be properly updated. This reduction is detailed in Reduction Rule 5. The output of this rule is a reduced universe $\mathcal{U}'$ and a reduced collection of sets $\mathcal{S}'$.

Reduction Rule 5. *Construct a bipartite graph $B = (V_O, V_P, E)$ with Algorithm 8. Compute a maximum matching M in B. Let V'_O be the set of unmatched vertices of V_O and O' be the elements of $O \subset \mathcal{U}$ corresponding to those vertices. Likewise, let $\mathcal{S}(O')$ be the sets in $\mathcal{S}\backslash\mathcal{R}$ that contain elements of O'. Reduce to $\mathcal{U}' = \mathcal{U}\backslash O'$ and $\mathcal{S}' = \mathcal{S}\backslash\mathcal{S}(O')$.*

Example 10. *For the instance of Example 8, $O = \{a,h,k,l\}$, and*

$$\mathcal{S}\backslash\mathcal{R} = \{\{b,c,e,a\},\{e,i,f,a\},\{e,g,i,h\},\{i,j,m,k\},\{i,j,m,l\}\}.$$

According to our construction, there will be one vertex in V_P for the subsets $\{b,c,e\}$, $\{e,i,f\}$, $\{e,g,i\}$ and $\{i,j,m\}$. Figure 7.2 depicts the bipartite graph constructed as described above. The left column represents the vertices in V_O while the right column represents the vertices in V_P. A maximum matching in this bipartite graph is highlighted with bold lines. Observe that the element k (or alternatively l) will be removed from $\mathcal{U}$ by Reduction Rule 5.

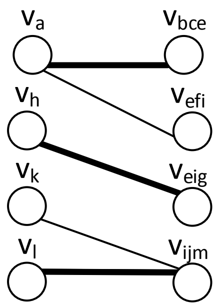

Figure 7.2: The bipartite graph constructed with O and $\mathcal{S}\backslash\mathcal{R}$ from Example 10

Algorithm 8 Compute Auxiliary Bipartite Graph

```
Input: $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{R}$
Output: A bipartite graph $B = (V_O, V_P, E)$
1: Make $O = \mathcal{U} \backslash val(\mathcal{R})$
2: Add a vertex $v_o$ to $V_O$ of each element in $O$
3: For each set $S$ in $\mathcal{S} \backslash \mathcal{R}$ and each subset $P \subsetneq S$ where $|P| = r - 1$, if there is at least one element $o \in O$
   such that $\{o\} \cup P \in \mathcal{S} \backslash \mathcal{R}$, add a vertex $v_p$ to $V_P$
4: Add an edge $(v_o, v_p)$ to $E$ for each pair $v_o, v_p$, if $\{o\} \cup P \in \mathcal{S} \backslash \mathcal{R}$
5: Return $B$
```

Correctness

We now proceed to show the correctness of Algorithm 6. Recall that this algorithm reduces the instance in two general steps: by removing extra sets of $\mathcal{S}$ and by removing extra elements of $\mathcal{U}$. The first step happens via executing Algorithm 7 in the loop of Lines 4-9 in Algorithm 6. The correctness of such step will be proved with Lemmas 40 and 41. The second step occurs via Reduction Rule 5 in Line 16 of Algorithm 6 and its correctness will be proved with Lemma 43.

Let us begin by showing that Algorithm 7 correctly reduces a given maximal set $\mathcal{R}$.

Lemma 40. *Let $\mathcal{E} \subset \mathcal{R}$ be the collection of extra sets returned by Algorithm 7. $\mathcal{R}$ has a (k, r, t)-set packing if and only if $\mathcal{R} \backslash \mathcal{E}$ has a (k, r, t)-set packing.*

Proof. Since $\mathcal{R} \backslash \mathcal{E} \subseteq \mathcal{R}$, we know that if $\mathcal{R} \backslash \mathcal{E}$ has a (k, r, t)-set packing, then $\mathcal{R}$ has one. Let $\mathcal{K}$ be a (k, r, t)-set packing of $\mathcal{R}$. If $\mathcal{K}$ is also a (k, r, t)-set packing of $\mathcal{R} \backslash \mathcal{E}$, the lemma follows. Otherwise, there is at least one set $S_e \in \mathcal{K}$ that is extra, ($S_e \in \mathcal{E}$ by Line 10); that is, $\mathcal{K} \cap \mathcal{E} \neq \emptyset$. We claim that we can replace each set in $\mathcal{K} \cap \mathcal{E}$ (an extra set) by a set in $\mathcal{R} \backslash \mathcal{E}$.

For any set S_e removed from $\mathcal{R}$ (i.e., $S_e \in \mathcal{E}$), there is at least one subset of elements $P \subset S_e$ such that $|\mathcal{P}| > f(i)$ (Line 8). Since the variable i determines the size of P and $f(i)$ is an upper bound for the number of sets containing P, we will prove the lemma by induction on i.

Let $i = ini$ (first iteration of loop of Line 3) and let P be a subset with ini elements such that $|\mathcal{P}| > f(i)$, where $\mathcal{P}$ is the set of all sets in $\mathcal{R}$ that contain P (Line 6).

Given that $|P| = ini \geq t+1$ (Line 2), at most one set in $\mathcal{P}$ can be in a (k, r, t)-set packing $\mathcal{K}$. Otherwise, there would be a pair of sets in $\mathcal{K}$ overlapping in more than t elements. Assume that set is S_e and is in $\mathcal{E}$ (Lines 9-10).

We claim that we can replace S_e in $\mathcal{K} \cap \mathcal{E}$ by a set in $\mathcal{P}'$ (the members of $\mathcal{S}$ that contain P that were kept in $\mathcal{R}$, Line 11).

Claim 7. *The $k - 1$ sets in $\mathcal{K} \backslash S_e$ conflict with at most $(r - t)(k - 1)$ sets in $\mathcal{P}'$.*

Proof. A set $S^* \in \mathcal{K} \backslash S_e$ conflicts with a set $S' \in \mathcal{P}'$, if they overlap in more than t elements, i.e., $|S^* \cap S'| \geq t + 1$.

Since both S^* and S_e are in $\mathcal{K}$, $|S_e \cap S^*| \leq t$ (otherwise $\mathcal{K}$ would not be a (k, r, t)-set packing). Moreover, $P \subset S_e$. Thus, $|P \cap S^*| = l \leq t$. Therefore, at least $t + 1 - l$ elements from S^* should be in $S' \backslash P$ to have a conflict between S^* and S'. That is, $|(S' \backslash P) \cap S^*| \geq t+1-l$. Let us denote as $u_1 \dots u_{t+1-l}$ the elements shared between $S' \backslash P$ and S^*.

Every set in $\mathcal{P}'$ that contains the subset of elements $P \cup \{u_1 \dots u_{t+1-l}\}$ will conflict with S^*. The sets in $\mathcal{P}'$ pairwise overlap exactly in the set P. Otherwise, there would be a pair of sets in $\mathcal{P} \subset \mathcal{R}$ overlapping in more than ini elements ($|P| = ini$), a contradiction to Line 2. This implies that there is at most one set in $\mathcal{P}'$ that contains the subset of elements $P \cup \{u_1 \dots u_{t+1-l}\}$.

It remains to count the number of disjoint subsets of S^* with $t+1-l$ elements. Notice that only disjoint sets are relevant for counting as we showed that the sets in $\mathcal{P}'$ intersect in exactly P. From the $|S^*| - l \leq r-l$ elements (l are already in P) we can have at most $\frac{r-l}{t+1-l}$ disjoint sets of $t+1-l$ elements. Hence, there are at most $\frac{r-l}{t+1-l}$ sets in $\mathcal{P}'$ that could conflict with S^*. Repeating this argument, the $k-1$ sets in $\mathcal{K} \backslash S_e$ conflict with at most $\frac{r-l}{t+1-l}(k-1)$ sets in $\mathcal{P}'$. This expression is maximum and equal to $(r-t)(k-1)$ when $l = t$. □

Given that $|\mathcal{P}'| = (r-t)(k-1)f(ini+1)+1$ (where $f(ini+1) = 1$), there is at least one set in $\mathcal{P}'$ that will not conflict with any set in $\mathcal{K} \backslash S_e$ and it can replace S_e.

Now, we will develop the same argument for $i < ini$.

Let $i < ini$ and P be a subset of elements of size i such that $|\mathcal{P}| > f(i)$. Since $|P| \geq t+1$ and all sets in $\mathcal{P}$ share at least P, at most one set of $\mathcal{P}$ is in a (k,r,t)-set packing. Assume that set is S_e and is extra, i.e., $S_e \in \mathcal{P} \backslash \mathcal{P}'$.

Claim 8. *The $k-1$ sets in $\mathcal{K} \backslash S_e$ conflict with at most $(r-t)(k-1)f(i+1)$ sets in $\mathcal{P}'$.*

Proof. A set $S^* \in \mathcal{K} \backslash S_e$ conflicts with a set $S' \in \mathcal{P}'$, if $|S^* \cap S'| \geq t+1$. Given that $P \subset S_e$ and $|S_e \cap S^*| \leq t$, $|P \cap S^*| = l \leq t$. Therefore, at least $t+1-l$ elements from S^* should be in $S' \backslash P$ to have a conflict between S^* and S'. That is, $|(S' \backslash P) \cap S^*| \geq t+1-l$. Let us denote as $u_1 \dots u_{t+1-l}$ the set of elements shared between $S' \backslash P$ and S^*.

Notice that all sets in $\mathcal{P}'$ that contain the subset of elements $P \cup \{u_1 \dots u_{t+1-l}\}$ will conflict with S^*. There are $f(i+(t+1-l))$ sets in $\mathcal{R}$ that contain a subset of elements of size $i+(t+1-l)$. Therefore, there are at most $f(i+(t+1-l))$ sets in $\mathcal{P}'$ that contain the subset $P \cup \{u_1 \dots u_{t+1-l}\}$ and will conflict with S^*.

It remains to compute the number of subsets with $t+1-l$ elements from S^*. There are at most $\binom{r-l}{(t+1)-l}$ ways of selecting $t+1-l$ elements from the $|S^* - l| \leq r-l$ elements of S^*. Repeating this argument, we find that the $k-1$ sets in $\mathcal{K} \backslash S_e$ can conflict with at most

$$\binom{r-l}{t+1-l}(k-1)f(i+(t+1-l)) \tag{7.1}$$

sets of $\mathcal{P}'$.

The function $f(i+j)$ is equivalent to $\sum_{j=0}^{ini-(i+j)+1}[(r-t)(k-1)]^j$ which is upper-bounded by $2[(r-t)(k-1)]^{ini-i-j+1}$. This implies that $f(i) > f(i+1) > f(i+2) > \dots f(i+j)$. Therefore Expression 7.1 is maximum when $l = t$ and upper-bounded by $(r-t)(k-1)f(i+1)$. □

Since there are $(r-t)(k-1)f(i+1)+1$ sets in $\mathcal{P}'$, there is at least one set in $\mathcal{P}'$ that will not conflict with any set in $\mathcal{K}\backslash S_e$ and it can replace S_e. □

We prove next that the sets collected in $\mathcal{E}$ (Lines 3-9 of Algorithm 6) are extra.

Lemma 41. *After running Lines 3-9 of Algorithm 6, $\mathcal{S}$ has a (k,r,t)-set packing if and only if $\mathcal{S}\backslash\mathcal{E}$ has a (k,r,t)-set packing.*

Proof. Since $(\mathcal{S}\backslash\mathcal{E}) \subseteq \mathcal{S}$, if $\mathcal{S}\backslash\mathcal{E}$ has a (k,r,t)-set packing, then $\mathcal{S}$ has one. Now, assume for contradiction that $\mathcal{S}$ has a (k,r,t)-set packing $\mathcal{K}$ but $\mathcal{S}\backslash\mathcal{E}$ does not have one. This implies that there is an $S \in \mathcal{K}$ that is in $\mathcal{E}$. If that set S is in $\mathcal{E}$, then in some iteration of Algorithm 6 (Lines 3-9) S was in $\mathcal{R}$ in Line 5 and after that S was not in $\mathcal{R}$ but in $\mathcal{E}'$. However, by Lemma 40 we know that $\mathcal{R}$ has a (k,r,t)-set packing if and only if $\mathcal{R}\backslash\mathcal{E}'$ has one. □

With Lemmas 40 and 41, we have proved the correctness of Algorithm 6 until Line 10. Lines 11-15 consist on computing a maximal (r,t)-set packing $\mathcal{M}$. If the size of $\mathcal{M}$ is at least k, then a subset of size k of $\mathcal{M}$ is output as $\mathcal{S}'$ and the algorithm stops. Therefore, it remains to show the correctness of Reduction Rule 5. Recall that this rule computes a maximum matching in the bipartite graph $B=(V_O,V_P,E)$. Since the construction of this graph heavily relies on the set $O=\mathcal{U}\backslash val(\mathcal{R})$, we will give a characterization of this set that will be key to prove the correctness of such rule.

Lemma 42. *Let $O=\mathcal{U}\backslash val(\mathcal{R})$. (i) Each element in O is contained in at least one set S. (ii) Only sets in $\mathcal{S}\backslash\mathcal{R}$ contain elements from O. (iii) No pair of different elements in O is contained in the same set S for any $S \in \mathcal{S}\backslash\mathcal{R}$. (iv) Each $S \in \mathcal{S}\backslash\mathcal{R}$ contains one element in O and $r-1$ elements of some set in $\mathcal{R}$.*

Proof. After re-applying Reduction Rule 4, each element in $\mathcal{U}$ is in at least one set S in the reduced collection $\mathcal{S}$. Since $O \subseteq \mathcal{U}$, (i) follows.

To prove claims (ii) and (iii), assume for the sake of contradiction that there is a pair of elements $u_i, u_j \in O$, $i \neq j$, such that $u_i, u_j \in S$. The set S is not in $\mathcal{R}$; otherwise, u_i, u_j would not be in O. In addition, if S is disjoint from $\mathcal{R}$, then S could be added to $\mathcal{R}$, contradicting its maximality. Therefore, $S \in \mathcal{S}\backslash\mathcal{R}$ and S overlaps with at least one set S' in $\mathcal{R}$. Since by assumption, u_i, u_j are both in S, $|S \cap S'| \leq r-2$. However, once again S can be added to $\mathcal{R}$ contradicting its maximality. This proves claims (ii) and (iii).

To prove claim (iv), assume there is a set $S \in \mathcal{S}\backslash\mathcal{R}$ such that $u_i \in S$ and S overlaps in at most $r-2$ elements with each set in $\mathcal{R}$. This once more contradicts the fact that each pair of sets in $\mathcal{R}$ overlaps in at most $r-2$ elements. □

We show next that Reduction Rule 5 correctly reduces $\mathcal{U}$ and $\mathcal{S}$.

Lemma 43. *Let $\mathcal{S}'$, $\mathcal{U}'$ be the reduced collection of sets and universe, respectively, output by Reduction Rule 5. $\mathcal{S}$ has a (k,r,t)-set packing if and only if $\mathcal{S}'$ has a (k,r,t)-set packing.*

Proof. Since $\mathcal{S}' \subseteq \mathcal{S}$, if $\mathcal{S}'$ has a (k,r,t)-set packing, then $\mathcal{S}$ has one as well. Suppose that $\mathcal{S}$ has a (k,r,t)-set packing $\mathcal{K}$. If $\mathcal{K}$ is not a solution in $\mathcal{S}'$, then at least one set in $\mathcal{K}$ contains an element in O' (the unmatched elements).

We next show that we can transform $\mathcal{K}$ into a (k, r, t)-set packing of $\mathcal{S}'$. Let $\mathcal{K}' = \{S_{o'_1}, \ldots, S_{o'_l}\}$ $(l \leq k)$ be the sets of $\mathcal{K}$ that contain unmatched elements. By Lemma 42, each $S_{o'_i} \in \mathcal{K}'$ contains only one element of O. Let us denote by o'_i the element in O' contained in the set $S_{o'_i}$.

We next show that for each o'_i, there exists an element in $O \backslash O'$ that can replace o'_i to form a new set. By our construction of B, there is a vertex $v_{p_i} \in V_P$ representing the subset $P_i = S_{o'_i} \backslash \{o'_i\}$, and this vertex v_{p_i} is adjacent to $v_{o'_i} \in V_O$ (the representative vertex of o'_i). The vertex v_{p_i} is matched. That is, there exists a matched edge (v_{o_i}, v_{p_i}) where $v_{o_i} \in V'_O$. Otherwise, we could add the edge $(v_{o'_i}, v_{p_i})$ to M contradicting the assumption that M is maximum. By our construction, $S_{o_i} = \{o_i\} \cup P_i$ is a set in $\mathcal{S} \backslash \mathcal{R}$. Thus, for each o'_i there exists an element o_i in $O \backslash O'$ that can replace o'_i to form the new set $S_{o_i} = (S_{o'_i} \backslash \{o'_i\}) \cup \{o_i\}$.

Let us denote as $\mathcal{K}^*$ by these new sets $\{S_{o_1}, \ldots, S_{o_l}\}$. We now claim that these sets pairwise overlap in at most t elements. For the sake of contradiction, suppose that there is a pair S_{o_i}, S_{o_j} that conflicts, i.e., $|S_{o_i} \cap S_{o_j}| \geq t + 1$. Since $S_{o_i} = P_i \cup \{o_i\}$, $S_{o_j} = P_i \cup \{o_j\}$ and $o_i \neq o_j$, the conflict is only possible if $|P_i \cap P_j| \geq t + 1$. Each set S_{o_i} is replacing a set $S_{o'_i}$ in $\mathcal{K}$ (a (k, r, t)-set packing). Since $S_{o'_i} \cap S_{o_i} = P_i$ and $S_{o'_j} \cap S_{o_j} = P_j$, $|P_i \cap P_j| \leq t$, as otherwise $\mathcal{K}$ would not be a (k, t) set packing.

We next show that we can form a solution of $\mathcal{S}'$ by altering $\mathcal{K}$. First, we replace $\mathcal{K}'$ by $\mathcal{K}^*$ in $\mathcal{K}$. If each set in $\mathcal{K}^*$ overlaps in at most t elements with each set in $\mathcal{K} \backslash \mathcal{K}'$, then $\mathcal{K}$ is a (k, r, t)-set packing and we are done. Assume that there is a set $S_{o_i} \in \mathcal{K}^*$ that overlaps in more than t elements with some set in $\mathcal{K} \backslash \mathcal{K}^*$ (we already showed that the sets in $\mathcal{K}^*$ pairwise overlap in at most t elements). Since $S_{o_i} = P_i \cup \{o_i\}$, P_i shares at most t elements with any set in $\mathcal{K} \backslash S_{o_i}$. Therefore, if $S_{o_i} = P_i \cup \{o_i\}$ conflicts with some set $S^* \in \mathcal{K} \backslash \mathcal{K}^*$, $o_i \in S^*$. Once again by our construction of B, $P^* = S^* \backslash \{o_i\}$ is represented by a vertex $v_{p^*} \in V_P$, and v_{p^*} is adjacent to v_{o_i}. We claim that v_{p^*} is a matched vertex, that is, there exists an edge $(v_{o^*}, v_{p^*}) \in M$. Assume for contradiction that v_{p^*} is unmatched. This would imply that we can increase the size of the matching M by adding the edges $(v_{o'_i}, v_{p_i})$ and (v_{o_i}, v_{p^*}) and remove the matched edge (v_{o_i}, v_{p_i}). However, this contradicts the assumption that M is maximum. In this way, we can change S^* by removing o_i and adding o^*, i.e., S^* becomes $(S^* \backslash \{o_i\}) \cup \{o^*\}$. We can repeat this argument, if the new S^* conflicts with another set in $\mathcal{K} \backslash \mathcal{K}^*$. Applying the same argument we can iterate replacing sets in $\mathcal{K}$ until we have a (k, r, t)-set packing in $\mathcal{S}'$.

□

We are now ready to state the correctness of Algorithm 6.

Theorem 27. *Let $(\mathcal{S}', \mathcal{U}', k)$ be the reduced instance of the r-Set Packing with t-Overlap returned by Algorithm 6. The collection $\mathcal{S}$ has a (k, r, t)-set packing if and only if $\mathcal{S}'$ has one.*

Kernel Size

We have proved so far the correctness of Algorithm 6. To actually show that the reduced instance returned by this algorithm is a kernel, we need to show two remaining steps: first that Algorithm 6 runs in polynomial time in n, and second that the size of the reduced instance is bounded by a computable function f depending only on the parameter k i.e., $f(k)$ (recall that r and t are fixed-constants).

Lemma 44. *Algorithm 6 runs in polynomial time.*

Proof. Reduction Rule 4 can be computed in $O(n^r)$ time. An upper bound to compute the set $\mathcal{R}$ is $O(n^r)$. Algorithm 7 runs in time $O(n^{r^2})$. In addition, a maximum matching can be computed in polynomial time. □

It remains to determine an upper bound for the number of elements in the reduced universe $\mathcal{U}'$. Given that $\mathcal{U}' = val(\mathcal{S}')$ (Reduction Rule 5), we will use $\mathcal{S}'$ to upper-bound $\mathcal{U}'$.

By Line 4 of Algorithm 6, $\mathcal{R} \subseteq \mathcal{S}$. Thus, $\mathcal{S} = (\mathcal{S}\backslash\mathcal{R}) \cup \mathcal{R}$. After Reduction Rule 5, some members in $\mathcal{S}\backslash\mathcal{R}$ were removed, and we obtained the reduced collection $\mathcal{S}' = \mathcal{S}\backslash\mathcal{S}(O')$. Thus, $\mathcal{S}' = (\mathcal{S}'\backslash\mathcal{R}) \cup \mathcal{R}$.

Since $\mathcal{S}'\backslash\mathcal{R} \subseteq \mathcal{S}\backslash\mathcal{R}$, by Lemma 42 we know that each set in $\mathcal{S}'\backslash\mathcal{R}$ contains one element in O and $r-1$ elements in a set of $\mathcal{R}$. Thus, $val(\mathcal{S}') = O \cup val(\mathcal{R})$. The elements in O' were removed from O (Reduction Rule 5), therefore, $val(\mathcal{S}') = (O\backslash O') \cup val(\mathcal{R}) = \mathcal{U}'$.

We next provide an upper bound for the size of $\mathcal{R}$.

Lemma 45. *The size of $\mathcal{R}$ is at most $2r^{r-1}k^{r-t-1}$ and $|val(\mathcal{R})| \leq 2r^r k^{r-t-1}$.*

Proof. Let $\mathcal{M}$ be a maximal (r,t)-set packing of $\mathcal{R}$. Each set $S \in \mathcal{R}$ is either in $\mathcal{M}$ or it overlaps with at least one set $S' \in \mathcal{M}$ in at least $t+1$ elements (i.e., $|S \cap S'| \geq t+1$). Otherwise, $\mathcal{M}$ would not be a maximal (r,t)-set packing.

For each set $S \in \mathcal{R}$, each subset of S with $t+1$ elements is contained in at most $f(t+1)$ sets in $\mathcal{R}$ (a direct consequence of Lemma 40). Therefore, we can use the subsets of size $t+1$ contained in the members of $\mathcal{M}$ to upper-bound the number of sets in $\mathcal{R}$.

For each set in $\mathcal{M}$, there are $\binom{r}{t+1}$ subsets of size $t+1$. Thus, $|\mathcal{R}| \leq |\mathcal{M}| \binom{r}{t+1} f(t+1)$.

If $|\mathcal{M}| \geq k$, $\mathcal{M}$ is a (k,r,t)-set packing of $\mathcal{S}$, and the algorithm outputs a subset of $\mathcal{M}$ and stops in Line 14. Thus, $|\mathcal{M}| \leq k-1$. On the other hand, $f(t+1) = \sum_{j=0}^{ini-(t+1)+1}[(r-t)(k-1)]^j < 2((r-t)(k-1))^{ini-(t+1)+1}$.

In this way, $|\mathcal{R}| \leq r^{t+1}\ k-1\ 2((r-t)(k-1))^{ini-(t+1)+1}$. Given that $ini = r-2$, $|\mathcal{R}| \leq 2r^{r-1}k^{r-t-1}$ and $|val(\mathcal{R})| \leq 2r^r k^{r-t-1}$. □

Now, we bound the size of $O\backslash O'$.

Lemma 46. *After applying Reduction Rule 5, there are at most $2r^r k^{r-t-1}$ elements in $O\backslash O'$.*

Proof. After applying Reduction Rule 5, $|V'_O| \leq |V_P|$. By our construction of the set V_P, there are at most $\binom{r}{r-1}|\mathcal{R}| = r|\mathcal{R}|$ vertices in V_P. By Lemma 45, $|\mathcal{R}| \leq 2r^{r-1}k^{r-t-1}$ and the lemma follows. □

Theorem 27, Lemma 44, the $2r^r k^{r-t-1}$ elements in $O\backslash O'$ together with the $2r^r k^{r-t-1}$ elements in $\mathcal{R}$ give the following result.

Theorem 28. *The r-Set Packing with t-Overlap has a problem kernel with $O(r^r k^{r-t-1})$ elements from the given universe.*

The size of the reduced collection $\mathcal{S}'$ is given by $|\mathcal{R}| + |\mathcal{S}\backslash\mathcal{R}|$. By our construction of the bipartite graph B (Algorithm 8) each edge in B corresponds to a set in $\mathcal{S}$. In fact, $|\mathcal{S}\backslash\mathcal{R}| = |E(B)|$.

Therefore, $|\mathcal{S}'| = |E(B')| + |\mathcal{R}|$, where B' is the reduced graph B after Reduction Rule 5. This rule removes unmatched vertices but not unmatched edges. Therefore, B' could be a complete bipartite graph and $|E(B')| \leq |V'_O||V_P|$. After applying Reduction Rule 5, $|V'_O| \leq |V_P|$, and there are $O(r^r k^{r-t-1})$ vertices in V_P. Thus, $|E(B)| = O(r^{r^2} k^{(r-t-1)^2})$ and $|\mathcal{S}'| = |E(B')| + |\mathcal{R}| = O(r^{r^2} k^{(r-t-1)^2}) + r^{r-1}k^{r-t-1} = O(r^{r^2} k^{(r-t-1)^2})$

Theorem 29. *Algorithm 6 returns a reduced collection $\mathcal{S}'$ with $O(r^{r^2} k^{(r-t-1)^2})$ sets.*

By running directly Algorithm 7 with input $\mathcal{R} = \mathcal{S}$ (instead of Algorithm 6), we can improve this result to $O(r^r k^{r-t})$ (it would follow by Lemma 45 with $ini = r-1$). Nevertheless, this improvement on $\mathcal{S}'$ would imply a slightly bigger reduced universe $\mathcal{U}'$ with $O(r^r k^{r-t})$ elements.

Improving the Kernel Size for the Weighted r-Set Packing with t-Overlap problem

We discussed in Section 6.2 how to reduce the weighted version of the r-Set Packing with t-Overlap problem to a kernel. Recall that in the weighted variant, every set in $\mathcal{S}$ has a weight (a real number) and the question is to find a (k, r, t)-set packing of maximum weight.

We believe that we can modify Algorithm 7 to reduce the weighted r-Set Packing with t-Overlap problem to a kernel. In Line 7 of Algorithm 7, we arbitrarily leve in $\mathcal{R}$ $f(i) = (r-t)(k-1)f(i+1)+1$ sets that contain a subset of size i. Instead of doing that choice arbitrarily, we keep in $\mathcal{R}$ the sets with maximum weight. In this way, the kernelization algorithm for the weighted r-Set Packing with t-Overlap will correspond to Algorithm 7 with input $\mathcal{R} = \mathcal{S}$ and Line 7 modified as explained. By omitting technical details, we can state that:

Theorem 30. *The weighted r-Set Packing with t-Overlap can be reduced to a problem kernel with $O(r^r k^{r-t})$ sets.*

This kernel size is an improved kernel with respect to the $O(e^{r^{t+1}+1}(r^{t+1}+1)k^{r^{t+1}+1})$ kernel obtained with PPTs in Section 6.2.

7.2.2 Smaller Kernel Sizes for $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap Problems

We will now apply our kernelization framework of Chapter 5 to obtain kernel results for our packing graphs with pairwise overlap problems. By improving our kernel result for the r-Set Packing with t-Overlap problem, we are automatically improving the kernel results for the graph problems as well.

Vertex Pairwise-Overlap

In the $\mathcal{H}$-Packing with t-Overlap, we seek for at least k $\mathcal{H}$-subgraphs of G such that each pair of those subgraphs overlaps in at most t vertices. To reduce this problem to a kernel, we will transform it to the r-Set

Packing with t-Overlap as in Section 5.2.1. However, to actually obtain a kernel for the graph version, one must have a kernelization algorithm for the r-Set Packing problem. In the framework described in Chapter 5, we assume that there exists a kernelization algorithm called KernelAlg-O for the r-Set Packing with t-Overlap. In this chapter, Algorithm 6 takes the place of KernelAlg-O and that is precisely what allows us to improve the kernel result for our graph packing with t-Overlap problems with respect to the kernels obtained in Chapter 6.

Briefly summarizing, we construct an instance (a universe $\mathcal{U}$ and a collection of sets $\mathcal{S}$) of the r-Set Packing with t-Overlap with the instance of the $\mathcal{H}$-Packing with t-Overlap as in Transformation 1. The parameter k is not altered and $r = r(\mathcal{H})$. We obtain a reduced universe $\mathcal{U}'$ and a reduced collection of sets $\mathcal{S}'$ (i.e., a kernel problem) by running Algorithm 6 on $\mathcal{U}$ and $\mathcal{S}$. Finally, we re-interpret the reduced universe $\mathcal{U}'$ as a reduced graph G' of the $\mathcal{H}$-Packing with t-Overlap with Transformation 2.

By Transformation 2, $G' = G[\mathcal{U}']$, and by Algorithm 6, $|\mathcal{U}'| = O(r^r k^{r-t-1})$. By Lemma 2, we can hence conclude.

Theorem 31. *The $\mathcal{H}$-Packing with t-Overlap has a problem kernel with $O(r^r k^{r-t-1})$ vertices, where $r = r(\mathcal{H})$.*

Our kernelization algorithm for the $\mathcal{H}$-Packing with t-Overlap problem corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing with t-Overlap*, Transformation 1, Algorithm 6, and Transformation 2.

In similar fashion, we obtain the next result for the induced version of the problem.

Theorem 32. *Induced-$\mathcal{H}$-Packing with t-Overlap has a problem kernel with $O(r^r k^{r-t-1})$ vertices, where $r = r(\mathcal{H})$.*

Edge Pairwise-Overlap

We introduce the $\mathcal{H}$-Packing with t-Edge Overlap problem to regulate the overlap between the edges of the $\mathcal{H}$-subgraphs instead of the vertices. Recall that there is an upper bound $m(\mathcal{H})$ on the number of edges of each graph occurring in $\mathcal{H}$.

By following a similar scheme as in the vertex overlap version, we obtain a kernel problem (a graph G') for the $\mathcal{H}$-Packing with t-Edge Overlap with the following kernelization algorithm: Algorithm 2 with input: G, $\mathcal{H}$, k, *r-Set packing with t-Overlap*, Transformation 3, Algorithm 4, and Transformation 4.

By Transformation 4, $G' = G[V(\mathcal{U}')]$, and by Algorithm 6, $|\mathcal{U}'| = O(r^r k^{r-t-1})$. By Lemma 4, we can hence conclude.

Theorem 33. *The $\mathcal{H}$-Packing with t-Edge-Overlap has a problem kernel with $O(r^r k^{r-t-1})$ vertices, where $r = m(\mathcal{H})$.*

7.3 Improved Kernel Results for Packing Problems with Membership

We now switch our attention to the packing problems with membership: r-Set Packing with t-Membership, $\mathcal{H}$-Packing with t-Membership, and $\mathcal{H}$-Packing with t-Edge Membership. In Section 7.3.1, we develop an algorithm that reduces the r-Set Packing with t-Membership problem to a kernel. Our algorithm lets us to improve the kernel result obtained in Section 6.2.1 from $O((r+1)^r k^r)$ to $O\big(r^3 \big(\frac{r}{t}\big)^{r-3} k^{r-1}\big)$. In Section 7.3.2, we also obtain improved kernel sizes for the $\mathcal{H}$-Packing with t-Membership and t-Edge Membership with respect to our kernels obtained via PPTs from Chapter 6.

7.3.1 A Reduction Algorithm for Packing Sets with Membership

We will follow a similar organization as in Section 7.2.1. We give first a general explanation of the kernelization algorithm for the r-Set Packing with t-Membership. After that, we focus on proving that this algorithm correctly reduces an instance of this problem, and we finally show that this reduced instance is a kernel.

The Algorithm

Our kernelization algorithm for the r-Set Packing with t-Membership is depicted in Algorithm 9. This algorithm follows similar ideas as our kernelization algorithm for the r-Set Packing with t-Overlap problem (Algorithm 6), but it has important differences that lead to separate analysis of correctness and kernel size as well.

Similarly as Algorithm 6, Algorithm 9 reduces r-Set Packing with t-Membership to a kernel problem in two major steps. First, we identify in Lines 3-9 the sets of $\mathcal{S}$ that *may not be needed* (*extra*) to form a (k, r, t)-set membership, and second in Line 16, we remove *redundant* elements of $\mathcal{U}$.

Algorithm 9 starts by cleaning up the instance of the r-Set Packing with t-Membership using Reduction Rule 4. Next, this algorithm greedily obtains a maximal collection $\mathcal{R} \subseteq \mathcal{S}$ such that every pair of sets in $\mathcal{R}$ overlaps in at most $r-2$ elements. After that, Algorithm 10 is called in Line 6 to determine which sets of $\mathcal{R}$ are *extra*. Lines 3-9 in Algorithm 9 basically keep reducing $\mathcal{R}$ using Algorithm 10 while there are no more sets from $\mathcal{S}\backslash(\mathcal{R} \cup \mathcal{E})$ to add to $\mathcal{R}$. We need to repeat these steps to preserve the maximality of $\mathcal{R}$.

Before continuing with the explanation of the rest of Algorithm 9, we will describe how Algorithm 10 finds the extra sets of $\mathcal{R}$. This algorithm counts the number of sets in $\mathcal{R}$ that pairwise overlap exactly in the same subset of elements $P \subset \mathcal{U}$ and keeps in $\mathcal{R}$ a specific number of them. Specifically, for $i = ini$ to 1 and each set $S \in \mathcal{R}$, Algorithm 10 takes every subset of elements $P \subsetneq S$ of size i (Lines 3-5). The number of sets in $\mathcal{R}$ that contain P (collected in $\mathcal{P}$, Line 6) is counted, and at most $f(i)$ of them are kept in $\mathcal{R}$ (Line 7-11). The rest of the sets is added to the collection $\mathcal{E}$. The threshold $f(i)$ is defined as $f(i) = \frac{r}{t}(k-1)\ f(i+1) + t$ where $f(ini+1)$ is initialized to one (Line 2).

The function $f(i)$ basically bounds the number of sets in $\mathcal{R}$ that contains a subset P with i elements. Observe that $f(i) = \frac{r}{t}(k-1)\ f(i+1) + t \leq \sum_{j=0}^{ini-i+1} t[(\frac{r}{t})(k-1)]^j$. Algorithm 10 returns $\mathcal{E}$ which is the collection of extra sets in $\mathcal{R}$ (Line 16).

Recall that the kernelization algorithm for the r-Set Packing with t-Overlap (Algorithm 6) also calls a routine to identify extra sets from $\mathcal{R}$ (Algorithm 7). The difference between Algorithm 10 and Algorithm 7 is that the first one runs from $ini = r-2$ to 1 while the second one stops at $t+1$. In addition, in Algorithm 10, $f(i)$ is defined as $f(i) = \frac{r}{t}(k-1)\ f(i+1) + t$ while in Algorithm 7 $f(i) = (r-t)(k-1)f(i+1) + 1$. We could have written a generic routine for both problems and pass these values as parameters. However,

Algorithm 9 Kernelization Algorithm by Reduction - r-Set Packing with t-Membership

```
Input: An instance $\mathcal{U}, \mathcal{S}$
Output: A reduced instance $\mathcal{U}', \mathcal{S}'$
1: Apply Reduction Rule 4
2: $\mathcal{R} = \emptyset$, $\mathcal{E} = \emptyset$
3: repeat
4:   Greedily add sets from $\mathcal{S}\backslash(\mathcal{R} \cup \mathcal{E})$ to $\mathcal{R}$ such that the sets in $\mathcal{R}$ pairwise overlap in at most $r - 2$
     elements.
5:   if at least one set was added to $\mathcal{R}$ then
6:     $\mathcal{E}'$ = Algorithm 10($\mathcal{R}$)
7:     $\mathcal{R} = \mathcal{R}\backslash\mathcal{E}'$, $\mathcal{E} = \mathcal{E} \cup \mathcal{E}'$
8:   end if
9: until no more sets have been added to $\mathcal{R}$
10: Reduce $\mathcal{S} = \mathcal{S}\backslash\mathcal{E}$ and re-apply Reduction Rule 4
11: Compute a maximal $(r, t)$-set membership $\mathcal{M}$ of $\mathcal{R}$
12: if $|\mathcal{M}| \geq k$ then
13:   Let $\mathcal{S}'$ be any subset of $\mathcal{M}$ of size $k$
14:   Return $\mathcal{U}$ and $\mathcal{S}'$
15: end if
16: Let $\mathcal{U}'$ and $\mathcal{S}'$ be the reduced universe and collection of sets, respectively, after applying Reduction
    Rule 6.
17: Return $\mathcal{U}'$ and $\mathcal{S}'$
```

Algorithm 10 Extra Sets Reduction - r-Set Packing with t-Membership

```
Input: A collection $\mathcal{R} \subseteq \mathcal{S}$ of sets
Output: A collection $\mathcal{E} \subseteq \mathcal{R}$ of sets
1: $ini = \max\{|S_i \cap S_j| : S_i, S_j \in \mathcal{R}, i \neq j\}$
2: $f(ini + 1) = 1$, $\mathcal{E} = \emptyset$
3: for $i = ini$ downto 1 do
4:   for each $S \in \mathcal{R}$ such that $|S| > i$ do
5:     for each $P \subsetneq S$ where $|P| = i$ do
6:       $\mathcal{P} = \{S' \in \mathcal{R} : S' \supsetneq P\}$
7:       $f(i) = \frac{r}{t}(k-1)\ f(i+1) + t$
8:       if $|\mathcal{P}| > f(i)$ then
9:         Choose any $\mathcal{P}' \subsetneq \mathcal{P}$ of size $f(i)$
10:        Set $\mathcal{E} \leftarrow \mathcal{E} \cup (\mathcal{P}\backslash\mathcal{P}')$ (extra sets)
11:        $\mathcal{R} \leftarrow \mathcal{R}\backslash(\mathcal{P}\backslash\mathcal{P}')$
12:      end if
13:    end for
14:  end for
15: end for
16: Return $\mathcal{E}$
```

we decided to keep them separate for explanatory purposes and to help the exposition of our analysis of correctness and kernel sizes.

We now continue explaining the rest of Algorithm 9. After Algorithm 9 finishes executing the repeat-loop of Lines 3-9, the collection $\mathcal{S}$ is updated by removing $\mathcal{E}$ in Line 10, and Reduction Rule 4 is re-applied. After that, Algorithm 9 computes a maximal (r, t)-set membership $\mathcal{M} \subseteq \mathcal{R}$. This maximal solution will help us to determine an upper-bound for the number of sets in $\mathcal{R}$ in Lemma 51. If $|\mathcal{M}| \geq k$, Algorithm 9 outputs a subset of size k from $\mathcal{M}$ and stops executing. Let us assume for explanatory purposes that $\mathcal{M}$ was not a (k, r, t)-set membership and Algorithm 9 continues executing.

In our second step of reduction in Algorithm 9, we determine *extra* elements from $\mathcal{U} \backslash val(\mathcal{R})$ that may not be needed in a (k, r, t)-set membership.

We will identify extra elements in $\mathcal{U} \backslash val(\mathcal{R})$, denoted as O from now on. Similarly as for H-Packing [114] and r-Set Packing with t-Overlap (Algorithm 6), we will construct an auxiliary bipartite graph $B = (V_O, V_P, E)$ to reduce O. In the kernelization algorithms of those problems, the reduction of O happens via computing a maximum matching in B. More precisely, the elements of O that correspond to the unmatched vertices of V_O are removed from $\mathcal{U}$. Given that in a t-Membership problem, an element of $\mathcal{U}$ can be included in up to t sets of a solution, we will compute a generalization of the maximum matching problem instead.

We construct the bipartite graph $B = (V_O, V_P, E)$ with Algorithm 8. Let $b : V(B) \to \mathbb{N}$ a degree constraint for every vertex in B. For our reduction purposes, we will assign $b(v) = 1$ for all $v \in V_O$, and $b(v) = t$ for all $v \in V_P$. After that we will compute a cardinality b-Matching on B (see Chapter 2). Intuitively, every vertex of V_P is matched up to t vertices from V_O. However, every vertex in V_O is only matched to at most one vertex of V_P. After that we will remove the elements of O corresponding to unmatched vertices of V_O. Our reduction rule is formally described next.

Reduction Rule 6. *Construct an auxiliary bipartite graph $B = (V_O, V_P, E)$ with Algorithm 8. Assign $b(v) = 1$ for all $v \in V_O$, and $b(v) = t$, for all $v \in V_P$ and compute a cardinality b-matching M_t. Let $V_{O'} \subsetneq V_O$ be the set of unmatched vertices of V_O and $O' \subsetneq O$ be the elements corresponding to those vertices. Likewise, let $\mathcal{S}(O')$ be the sets in $\mathcal{S} \backslash \mathcal{R}$ that contain elements of O'. Reduce $\mathcal{U}' = \mathcal{U} \backslash O'$ and $\mathcal{S}' = \mathcal{S} \backslash \mathcal{S}(O')$.*

Example 11. *Let us assume an instance of the r-Set Packing with t-Membership with $k = 4$, $r = 4$ and $t = 2$.*

$$\begin{aligned}\mathcal{S} =&\{\{b,c,d,m\},\{a,b,c,d\},\{b,e,f,q\},\{a,b,e,f\},\{h,i,j,m\},\\ &\{g,h,i,j\},\{l,h,i,j\},\{t,h,i,j\},\{m,e,q,i\},\{p,m,e,q\},\\ &\{r,m,e,q\},\{s,m,e,q\},\{l,m,e,q\}\}\end{aligned}$$

Let us suppose the following set $\mathcal{R}$ after reduced by the repeat-loop of Lines 3-9 in Algorithm 9.

$$\mathcal{R} =\{\{b,c,d,m\},\{b,e,f,q\},\{h,i,j,m\},\{m,e,q,i\}\}$$

In this way, $O = \{a, g, l, p, r, s, t\}$. The bipartite graph $B = (V_O, V_P, E)$ constructed as in Algorithm 8 is shown in Figure 7.3. The degree constraint for every vertex is $b(v) = 1$ for all $v \in V_O$, and $b(v) = 2$, for all $v \in V_P$. A maximum cardinality-b-matching of B is highlighted. In this example, the elements s and t are identified as extra and will be removed from $\mathcal{U}$. As a consequence the sets $\{s, m, e, q\}$ and $\{t, h, i, j\}$ will be removed from $\mathcal{S}$.

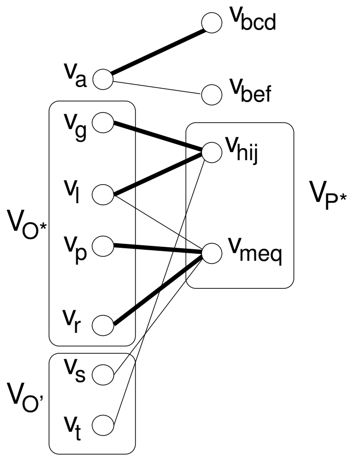

Figure 7.3: The bipartite graph constructed from Example 11. A maximum cardinality b-matching is highlighted.

Correctness

Next, we prove the correctness of Algorithm 9. In general this algorithm reduces an instance of the r-Set Packing with t-Membership in two steps: by removing extra sets of $\mathcal{S}$ and by removing extra elements of $\mathcal{U}$. The first step of reduction occurs by executing Algorithm 10 in the loop of Lines 4-9 in Algorithm 9. We show that this step is correct with Lemmas 47 and 48. Reduction Rule 6 in Line 16 of Algorithm 9 triggers the second reduction step. The correctness of this rule is proved in Lemma 49.

Lemma 47. *Let $\mathcal{E} \subset \mathcal{R}$ be the collection of extra sets returned by Algorithm 10. $\mathcal{R}$ has a (k, r, t)-set membership if and only if $\mathcal{R} \backslash \mathcal{E}$ has a (k, r, t)-set membership.*

Proof. Since $\mathcal{R} \backslash \mathcal{E} \subseteq \mathcal{R}$, if $\mathcal{R} \backslash \mathcal{E}$ has a (k, r, t)-set membership then $\mathcal{R}$ has one. Let $\mathcal{K}$ be a (k, r, t)-set membership of $\mathcal{R}$. If $\mathcal{K}$ is also a (k, r, t)-set membership of $\mathcal{R} \backslash \mathcal{E}$ then the lemma follows. Otherwise, there is at least one set in $\mathcal{K}$ that is in $\mathcal{E}$ (Line 10); that is, $\mathcal{K} \cap \mathcal{E} \neq \emptyset$. We claim that we can replace each set in $\mathcal{K} \cap \mathcal{E}$ (an extra set) by a set in $\mathcal{R} \backslash \mathcal{E}$.

For any set $S_e \in \mathcal{E}$ (removed from $\mathcal{R}$), there is at least one subset of elements $P \subsetneq S_e$ such that $|\mathcal{P}| > f(i)$ (Line 8). Recall that $\mathcal{P}$ collects all the sets in $\mathcal{R}$ that contain P. Since the variable i determines the size of P and $f(i)$ is an upper bound for the number of sets containing P, we will prove the lemma by induction on i.

Let $i = ini$ (first iteration of loop of Line 3) and let P be a subset with ini elements such that $|\mathcal{P}| > f(ini)$, where $\mathcal{P}$ is the collection of all sets in $\mathcal{R}$ that contain P (Line 6).

Observation 2. *At most t sets of $\mathcal{P}$ are in $\mathcal{K}$; otherwise, there would be at least $t+1$ sets in $\mathcal{K}$ containing the subset P and $\mathcal{K}$ would not be a (k, r, t)-set membership.*

Let us denote as $\mathcal{S}^e$ the sets of $\mathcal{P}$ that are in $\mathcal{K}$ and are extra, i.e., $\mathcal{S}^e = \mathcal{P} \cap (\mathcal{K} \cap \mathcal{E})$. By Observation 2, $|\mathcal{S}^e| = m \leq t$.

We claim that we can replace each set in $\mathcal{S}^e$ by a set in $\mathcal{P}' \subseteq \mathcal{R} \backslash \mathcal{E}$ ($\mathcal{P}'$ is the subset of $\mathcal{P}$ that was kept in $\mathcal{R}$, Line 11).

Claim 9. *The collection $\mathcal{P}'$ has at most $\frac{r}{t}(k-m)$ sets that could conflict with the $k-m$ sets in $\mathcal{K} \backslash \mathcal{S}^e$.*

Proof. Let S' be a set in $\mathcal{P}'$ that *conflicts* with the collection $\mathcal{K} \backslash \mathcal{S}^e$. (i) That is, there is at least one element $u \in S'$ that is contained in exactly t sets of $\mathcal{K} \backslash \mathcal{S}^e$. This would imply that S' cannot be added to $\mathcal{K} \backslash \mathcal{S}^e$. More precisely, $(\mathcal{K} \backslash \mathcal{S}^e) \cup S'$ is not an (l, r, t)-set membership where $l = |(\mathcal{K} \backslash \mathcal{S}^e) \cup S'|$.

We next show that the element u is contained only in the set S', i.e., $u \in S'$ and $u \notin S$ for each $S \in \mathcal{P}'$ with $S \neq S'$.

Recall that all sets in $\mathcal{P}$ contain a subset P, thus $P \subset S'$. Since $\mathcal{S}^e \subset \mathcal{P}$, S' shares at least P with every $S_e \in \mathcal{S}^e$.

First, we show that $u \notin P$. Assume otherwise by contradiction. By assumption (i), u is in t sets of $\mathcal{K} \backslash \mathcal{S}^e$. (ii) Since $P \subset S_e$ for each $S_e \in \mathcal{S}^e$, this would imply that u is also in all the sets in $\mathcal{S}^e$, and therefore in at least $t+1$ sets of $\mathcal{K}$ (recall that $S_e \in \mathcal{K}$), a contradiction since $\mathcal{K}$ is a (k, r, t)-set membership.

In this way, if u is in another set $S \neq S'$ of $\mathcal{P}$, u must be in $S \backslash P$. However, this implies that $|S \cap S'| \geq |P| + 1 = ini + 1$, a contradiction to Line 1 from Algorithm 10. Therefore, the element u is only contained in S', and u accounts for only one *conflicting set* in $\mathcal{P}'$.

Since by assumption (i), u is in t sets of $\mathcal{K} \backslash \mathcal{S}^e$, there are at most $r(k-m) - t$ elements "left" in $val(\mathcal{K} \backslash \mathcal{S}^e)$ to count for possible conflicting sets in $\mathcal{P}'$. We can repeat up to j times the argument above before we "run out" of elements in $val(\mathcal{K} \backslash \mathcal{S}^e)$; that is, when $r(k-m) - jt = 0$

This implies that $\mathcal{K} \backslash \mathcal{S}^e$ can conflict with at most $j = \frac{r}{t}(k-m)$ sets in $\mathcal{P}'$. □

By the claim above, $\mathcal{P}'$ should have at least $\frac{r}{t}(k-m) + m$ to replace the m sets in $\mathcal{S}^e$ by sets in $\mathcal{P}'$. Since $1 \leq m \leq t$, $\frac{r}{t}(k-m)+m$ is upper-bounded by $\frac{r}{t}(k-1)+t$. Given that $f(ini) = 1$, $|\mathcal{P}'| = \frac{r}{t}(k-1)+t$, and there always will be m sets in $\mathcal{P}'$ to replace the m sets in $\mathcal{S}^e$.

We next develop the same argument for $i < ini$. Similarly, let P be a subset with i elements such that $|\mathcal{P}| > f(i)$ and $\mathcal{S}^e = \mathcal{P} \cap (\mathcal{K} \cap \mathcal{E})$.

Once again, we claim that we can replace each set in $\mathcal{S}^e$ by a set in $\mathcal{P}'$, and we denote $|\mathcal{S}^e|$ by $m \leq t$ (Observation 2).

Claim 10. *The collection $\mathcal{P}'$ has at most $\frac{r}{t}(k-m)f(i+1)$ sets that could conflict with the $k-m$ sets in $\mathcal{K} \backslash \mathcal{S}^e$.*

Proof. Let S' be a set in $\mathcal{P}'$ conflicting with $\mathcal{K}\backslash\mathcal{S}^e$. That is, as before, there is at least one element $u \in S'$ that is in exactly t sets of $\mathcal{K}\backslash\mathcal{S}^e$.

The element u is contained in at most $f(i+1)$ sets in $\mathcal{P}'$. This follows because $f(i+1)$ bounds the number of sets that contain a subset of $i+1$ elements. Each set in $\mathcal{P}'$ contains P and $|P| = i$, thus there are at most $f(i+1)$ sets of $\mathcal{P}'$ containing $P \cup u$ (we showed before in (ii) that u cannot be in P).

In this way, the element u accounts for $f(i+1)$ *conflicting sets* in $\mathcal{P}'$. Once again, we can repeat up to j times the argument above before we "run out" of elements in $val(\mathcal{K}\backslash\mathcal{S}^e)$ that is when $r(k-m) - jt = 0$. Hence, the sets in $\mathcal{K}\backslash\mathcal{S}^e$ can conflict with at most $j = \frac{r}{t}(k-m)f(i+1)$ sets in $\mathcal{P}'$. □

By the above claim, $\mathcal{P}'$ should have at least $\frac{r}{t}(k-m)f(i+1) + m$ to replace the m sets in $\mathcal{S}^e$ by sets in $\mathcal{P}'$. Since $1 \leq m \leq t$, $\frac{r}{t}(k-m)f(i+1) + m \leq \frac{r}{t}(k-1)f(i+1) + t$, there will always be t sets in $\mathcal{P}'$ to replace the extra sets in $\mathcal{S}^e$. □

From the previous lemma, we can derive the following corollary which will be required in the proof of Lemma 51.

Corollary 4. *Every element of a set in $\mathcal{R}\backslash\mathcal{E}$ is contained in at most $f(1) = \sum_{j=0}^{ini} t[\frac{r}{t}(k-1)]^j < 2t$ $(\frac{r}{t}(k-1))^{ini}$ sets in $\mathcal{R}\backslash\mathcal{E}$.*

Next, we show that the collection of sets $\mathcal{E}$ found in Lines 3-9 of Algorithm 9 is extra.

Lemma 48. *After running Lines 1-9 of Algorithm 9, $\mathcal{S}$ has a (k, r, t)-set membership if and only if $\mathcal{S}\backslash\mathcal{E}$ has a (k, r, t)-set membership.*

Proof. Since $(\mathcal{S}\backslash\mathcal{E}) \subseteq \mathcal{S}$, if $\mathcal{S}\backslash\mathcal{E}$ has a (k, r, t)-set membership then $\mathcal{S}$ has one. Now, assume for contradiction that $\mathcal{S}$ has a (k, r, t)-set membership $\mathcal{K}$ but $\mathcal{S}\backslash\mathcal{E}$ does not have one. This implies that there is an $S \in \mathcal{K}$ that is in $\mathcal{E}$. If that set S is in $\mathcal{E}$, then in some iteration of Algorithm 9 (Lines 3-9) S was in $\mathcal{R}$ in Line 5 and after that S was not in $\mathcal{R}$ but in $\mathcal{E}'$. However, by Lemma 47 we know that $\mathcal{R}$ has a (k, r, t)-set membership if and only if $\mathcal{R}\backslash\mathcal{E}'$ has one. □

Lemmas 47 and 48 show the correctness of Algorithm 9 until Line 10. It remains to show the correctness of Reduction Rule 6. Before doing that, it is fundamental in the correctness of this rule to notice that the set O ($O = \mathcal{U}\backslash val(\mathcal{R})$) computed in this section follows Lemma 42.

We show next that Reduction Rule 6 correctly reduces $\mathcal{U}$ and $\mathcal{S}$.

Lemma 49. *Let $\mathcal{S}'$, $\mathcal{U}'$ be the reduced collection of sets and universe, respectively, output by Reduction Rule 6. The collection $\mathcal{S}$ has a (k, r, t)-set membership if and only if $\mathcal{S}'$ has a (k, r, t)-set membership.*

Proof. Since $\mathcal{S}' \subseteq \mathcal{S}$, if $\mathcal{S}'$ has a (k, r, t)-set membership, then $\mathcal{S}$ has one as well. Suppose that $\mathcal{S}$ has a (k, r, t)-set membership $\mathcal{K}$. If $\mathcal{K}$ is not a solution in $\mathcal{S}'$, then at least one set in $\mathcal{K}$ contains an element in O' (unmatched elements). We next show that there is always a (k, r, t)-set membership that does not contain any element from O'.

Let $V_{O'} \subsetneq V_O$ be the set of all unmatched vertices of V_O. Let $\mathcal{P}$ be the collection of all M_t-alternating paths in B each one starting in a vertex in $V_{O'}$. Now, let M_t^* be the set of matched edges in these paths, i.e., $M_t^* = (E(\mathcal{P}) \cap M_t)$, where M_t is the maximum cardinality b-matching computed in Rule 6. The set $V_{O^*} = V(M_t^*) \cap (V_O \backslash V_{O'})$ corresponds to the end-points of the edges in M_t^* that are in $V_O \backslash V_{O'}$. Similarly, $V_{P^*} = V(M_t^*) \cap (V_P)$. (See Figure 7.3)

Claim 11. *i. Vertices in $V_{O'} \cup V_{O^*}$ are only adjacent to vertices in V_{P^*}.*

ii. Every vertex in V_{P^*} is matched to exactly t different vertices in V_{O^*}, and each vertex in V_{O^*} is matched to exactly only one vertex in V_{P^*}. We call this a $(1,t)$-perfect matching *between V_{O^*} and V_{P^*}.*

Proof. Let us start first with the proof of (i). Assume for contradiction that there exists an edge $(v_{o'}, v_p)$ with $v_{o'} \in V_{O'}$ but $v_p \notin V_{P^*}$. Notice first that v_p is a t-matched vertex. Otherwise, we could add the edge $(v_{o'}, v_p)$ to M_t contradicting that M_t is maximum. Since $v_{o'}$ is unmatched, there is an M_t-alternating path v_o, v_p, v_o^* (for some $v_o^* \in V_O$), and v_p would be in V_{P^*} by our definition of V_{P^*}.

Once again by contradiction, assume that there exists an edge (v_o, v_p) with $v_o \in V_O^*$ but $v_p \notin V_{P^*}$. Notice that v_p must be t-matched. Otherwise, the edge (v_o, v_p) and the M_t-alternating path $P \in \mathcal{P}$ starting in some vertex $v_o' \in V_O'$ and ending in v_o would form an M_t-augmenting path. A contradiction because M_t is maximum (Lemma 1). In this way, $v_p \in V_P^*$. This concludes the proof of (i).

Since every vertex in V_O' is adjacent to only vertices in V_{P^*}, and there are no M_t-augmenting paths in B, every vertex in V_{P^*} is matched to exactly t vertices in V_O^* and each vertex in V_O^* is matched to only one vertex and that vertex is in V_{P^*}. Furthermore, vertices in V_{O^*} are only adjacent to vertices in V_{P^*}. □

We refer to $O^* \subseteq (O \backslash O')$ as the set of elements corresponding to the vertices in V_O^*. Similarly, P^* is the collection of subsets each one represented by a vertex in V_{P^*}. By the construction of B, Claim 11 relates to the collection $\mathcal{S}$ and O in the following way.

Claim 12. *(i) The elements in $O' \cup O^*$ are contained only in sets in $\mathcal{S} \backslash \mathcal{R}$. Furthermore each of these sets contains a subset of P^*. More precisely, for any set $S \in \mathcal{S} \backslash \mathcal{R}$ containing one element $o \in (O' \cup O^*)$ (by Lemma 42 is only one), the subset $P = (S \backslash o)$ is in P^*.*

(ii) There are at least $t|P^*|$ sets in $\mathcal{S} \backslash \mathcal{R}$ of the form $P_i \cup o_l^*$ where $P_i \in P^*$ and $o_l^* \in O^*$. Furthermore, for each pair $P_i \cup o_l^*$ and $P_j \cup o_n^*$ of these sets, $o_l^* \neq o_n^*$.

Now we need to show that there is always a (k, r, t)-set membership that does not include unmatched elements (elements from O'). Let $\mathcal{K}^* \subseteq \mathcal{K}$ be all sets of $\mathcal{K}$ that contain elements from O^*. Similarly, $\mathcal{K}' \subseteq \mathcal{K}$ is the collection of sets of $\mathcal{K}$ that contain elements from O'. Each set in $(\mathcal{K}^* \cup \mathcal{K}')$ contains a subset in P^* (Claim 12 (i)). We claim that we cannot form more sets of $\mathcal{K}$ by forming sets using subsets in P^* with elements of $O^* \cup O'$ rather than using O^* only.

Any element in $val(P^*)$ is contained in at most t sets of $\mathcal{K}$; otherwise, $\mathcal{K}$ would not be a solution by (ii). Therefore, each member of P^* can be in at most t sets of $\mathcal{K}$ and $|\mathcal{K}' \cup \mathcal{K}^*| \leq t|P^*|$.

Since by Claim 12 (ii), there are $t|P^*|$ sets in $\mathcal{S}$ containing each one an element of O^* and a subset of P^*, we know that we can replace all the sets in $\mathcal{K}' \cup \mathcal{K}^*$ by sets that contain a subset in P^* and an element in O^*. Let $\bar{\mathcal{K}}$ the collection of these newly formed sets. Every element of $val(\bar{\mathcal{K}})$ is contained in at most t sets of $(\mathcal{K} \backslash (\mathcal{K}' \cup \mathcal{K}^*)) \cup \bar{\mathcal{K}}$. This follows because: 1. each element in $val(P^*)$ is contained in at most t sets of $\mathcal{K}$ (see **), and 2. the elements in $val(\bar{\mathcal{K}}) \backslash val(P^*)$ are elements from O^* which by Claim 12 (ii) are all different. Furthermore, elements from O^* form sets of $\mathcal{S}$ (more precisely sets of $\mathcal{S} \backslash \mathcal{R}$) using only subsets of P^*. □

We now state the correctness of Algorithm 9.

Theorem 34. *Let $(\mathcal{S}', \mathcal{U}', k)$ be the reduced instance of the r-Set Packing with t-Membership returned by Algorithm 9. The collection $\mathcal{S}$ has a (k, r, t)-set membership if and only if $\mathcal{S}'$ has one.*

Kernel Size

Up to this point, we have shown that the reduced instance by Algorithm 9 has a (k, r, t)-set membership if and only if the original instance has one. To prove that this instance is a kernel, we need to show that Algorithm 9 runs in polynomial time in n and the size of the reduced instance is bounded by a computable function f depending only on the parameter k. Recall that t and r are fixed-constants.

Lemma 50. *Algorithm 9 runs in polynomial time.*

Proof. Reduction Rule 4 can be computed in $O(n^r)$ time. An upper bound to compute the set $\mathcal{R}$ is $O(n^r)$. Algorithm 10 runs in time $O(r^r n^r)$. In addition, a cardinality b-matching can be computed in polynomial time [129, 108]. □

It remains to determine an upper bound for the number of elements in the reduced universe $\mathcal{U}'$. Given that $\mathcal{U}' = val(\mathcal{S}')$ (Reduction Rule 6), we will use $\mathcal{S}'$ to upper-bound $\mathcal{U}'$.

The collection $\mathcal{S}' = (\mathcal{S}' \backslash \mathcal{R}) \cup \mathcal{R}$. Since $\mathcal{S}' \backslash \mathcal{R} \subseteq \mathcal{S} \backslash \mathcal{R}$, by Lemma 42 we know that each set in $\mathcal{S}' \backslash \mathcal{R}$ contains one element in O and $r - 1$ elements in a set of $\mathcal{R}$. Thus, $val(\mathcal{S}') = O \cup val(\mathcal{R})$. The elements in O' were removed from O (Reduction Rule 6), therefore, $val(\mathcal{S}') = (O \backslash O') \cup val(\mathcal{R})$.

We next provide an upper bound for the size of $\mathcal{R}$.

Lemma 51. *The number of sets in $\mathcal{R}$ is at most $2r\left(\frac{r}{t}\right)^{r-2} k^{r-1}$, and $|val(\mathcal{R})| \leq 2r^2 \left(\frac{r}{t}\right)^{r-2} k^{r-1}$.*

Proof. (i) Let $\mathcal{M}$ be a maximal (r, t)-set membership of $\mathcal{R}$. Each set $S \in \mathcal{R}$ is either in $\mathcal{M}$ ($S \in \mathcal{M}$), or it shares one element with at least t sets in $\mathcal{M}$, i.e., $S \notin \mathcal{M}$ but $|S \cap S_i| \geq 1$ for some collection $\{S_1 \dots S_t\} \subseteq \mathcal{M}$. Otherwise, $\mathcal{M}$ would not be a maximal (r, t)-set membership.

There are $|val(\mathcal{M})|$ elements in $\mathcal{M}$. Assuming each one of them is contained in a set of $\mathcal{R} \backslash \mathcal{M}$, and by Corollary 4, we know that this element will be in at most $f(1)$ sets in $\mathcal{R}$, we have that $|\mathcal{R}| \leq |val(\mathcal{M})| f(1)$.

Strictly by (i), at most $\frac{|val(\mathcal{M}|)}{t}$ elements of $\mathcal{M}$ could be intersecting with sets in $\mathcal{R} \backslash \mathcal{M}$, thus $|\mathcal{R}| \leq \frac{|val(\mathcal{M}|)}{t} f(1)$.

As each set in $\mathcal{S}$ has at most r elements, $|val(\mathcal{M})| \leq r|\mathcal{M}|$. If $|\mathcal{M}| \geq k$ then $\mathcal{M}$ is a (k, r, t)-set membership, and Algorithm 9 outputs a subset of $\mathcal{M}$ and stops in Line 14. Thus, $|\mathcal{M}| \leq k - 1$.

On the other hand, $f(1) \leq 2t(\frac{r}{t}(k-1))^{ini}$ (Corollary 4).

Combining these upper-bounds, $|\mathcal{R}| \leq \frac{r}{t}(k-1) 2t(\frac{r}{t}(k-1))^{ini}$.

Since $ini = r-2$ by Line 1 of Algorithm 10 (consequence of Line 4 of Algorithm 9), $|\mathcal{R}| \leq 2r\left(\frac{r}{t}\right)^{r-2} k^{r-1}$, and $|val(\mathcal{R})| \leq r|\mathcal{R}| \leq 2r^2\left(\frac{r}{t}\right)^{r-2} k^{r-1}$.

□

Now, we bound the size of $O \backslash O'$.

Lemma 52. *After applying Reduction Rule 6, there are at most $2r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}$ elements in O.*

Proof. After applying Reduction Rule 49, $|O| = |V_O| \leq t|V_P|$.

By our construction of V_P, there are at most $\binom{r}{r-1}|\mathcal{R}| = r|\mathcal{R}|$ vertices in V_P. By Lemma 51, $|\mathcal{R}| \leq 2r^2\left(\frac{r}{t}\right)^{r-2}k^{r-1}$ and the lemma follows. □

Theorem 34, Lemma 50, the $2r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}$ elements in O together with the $2r^2\left(\frac{r}{t}\right)^{r-2}k^{r-1}$ elements in $\mathcal{R}$ give the following result.

Theorem 35. *The r-Set Packing with t-Membership has a problem kernel with $O\left(r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}\right)$ elements from the given universe.*

The size of the reduced collection $\mathcal{S}'$ is given by $|\mathcal{R}| + |\mathcal{S}\backslash\mathcal{R}|$. By our construction of the bipartite graph B (Algorithm 8) each edge in B is a set in $\mathcal{S}$. In fact, $|\mathcal{S}\backslash\mathcal{R}| = |E(B)|$.

Therefore, the size of $\mathcal{S}' = |E(B')| + |\mathcal{R}|$, where B' is the reduced graph B after Rule 6. This rule removes unmatched vertices but not unmatched edges. Therefore, B' could be a complete bipartite graph and $|E(B')| \leq |V'_O||V_P|$. There are $2r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}$ vertices in V'_O and $2r^2\left(\frac{r}{t}\right)^{r-2}k^{r-1}$ vertices in V_P. In this way, $\mathcal{S}' = (2r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1})^2 + 2r\left(\frac{r}{t}\right)^{r-2}k^{r-1}$.

Theorem 36. *Algorithm 9 returns a reduced collection $\mathcal{S}'$ with $O\left(\left(r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}\right)^2\right)$ sets.*

Similarly as with the r-Set Packing with t-Overlap problem, by running directly Algorithm 10 with input $\mathcal{R} = \mathcal{S}$ (instead of Algorithm 9), we can improve this result to $O(2r\left(\frac{r}{t}\right)^{r-1}k^r)$ (it would follow by Lemma 51, $ini = r-1$). However, this improvement on $\mathcal{S}'$ would imply a slightly bigger reduced universe $\mathcal{U}'$ with $O(2r\left(\frac{r}{t}\right)^{r-1}k^r)$ elements.

Improving the Kernel Size for the Weighted r-Set Packing with t-Membership problem

We next discuss how to improve the kernel size for the weighted r-Set Packing with t-Membership problem with respect to kernel obtained in Section 6.1. The input of this improved kernelization algorithm consists on running Algorithm 10 with input $\mathcal{R} = \mathcal{S}$. We also modify Line 7 of Algorithm 10 by keeping on $\mathcal{R}$, $f(i) = \frac{r}{t}(k-1)\ f(i+1) + t$ sets of maximum weight (instead of an arbitrary selection). With this modification we obtain the following result.

Theorem 37. *The weighted r-Set Packing with t-Membership has a problem kernel with $O(2r\left(\frac{r}{t}\right)^{r-1}k^r)$ sets.*

This kernel result slightly improves the $O(e^{r+1}(r+1)k^{r+1})$ kernel obtained with PPTs in Section 6.1.

7.3.2 Smaller Kernel Sizes for $\mathcal{H}$-Packing with t-Membership and t-Edge Membership Problems

As with the pairwise overlap problems in Section 7.2.2, we will improve the kernel sizes for our $\mathcal{H}$-Packing with t-Membership and t-Edge Membership problems applying the framework from Chapter 5. In that framework, we basically transform an instance of $\mathcal{H}$-Packing with t-Membership and t-Edge Membership to instances of the r-Set Packing with t-Membership problem. By reducing the constructed instances of the set packing version to a kernel, we obtain a kernel for the original graph problems.

The reader will notice that this is the process we applied in Sections 6.1.2 and 6.1.3 as well. The difference relies in that now we use Algorithm 9 (instead of Algorithm 3) to reduce to a kernel the constructed instances. This change allows us to improve the kernel sizes for our graph packing with membership problems.

Vertex Membership

Let us briefly summarize how we apply the framework from Section 5.1.1 to reduce the $\mathcal{H}$-Packing with t-Membership to a kernel. We construct an instance (a universe $\mathcal{U}$ and a collection of sets $\mathcal{S}$) of the r-Set Packing with t-Membership problem from the instance of the $\mathcal{H}$-Packing with t-Membership problem with Transformation 1. After that, we reduce that constructed instance to a kernel with a kernelization algorithm for the r-Set Packing with t-Membership problem. In Chapter 5, we call that algorithm KernelAlg-M. In this chapter, Algorithm 9 takes the place of KernelAlg-M in that approach. Since Algorithm 9 improves the kernel obtained by Algorithm 3, this lets us to also improve the kernel for our graph problems. Once that we obtained the reduced universe $\mathcal{U}'$ and collection of sets $\mathcal{S}'$ with Algorithm 9, we can re-interpret $\mathcal{U}'$ towards a reduced graph G' for the original graph problem with Transformation 2.

By Transformation 2, $G' = G[\mathcal{U}']$, and by Algorithm 9 $|\mathcal{U}'| = O\big(r^3 \left(\frac{r}{t}\right)^{r-3} k^{r-1}\big)$. Lemma 13 permits us to conclude:

Theorem 38. *$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O\big(r^3 \left(\frac{r}{t}\right)^{r-3} k^{r-1}\big)$ vertices, where $r = r(\mathcal{H})$.*

In similar fashion, we obtain the next result.

Theorem 39. *Induced-$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O\big(r^3 \left(\frac{r}{t}\right)^{r-3} k^{r-1}\big)$ vertices, where $r = r(\mathcal{H})$.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Membership that we described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, *r-SetPacking with t-Membership*, Transformation 1, Algorithm 9, and Transformation 2.

Edge Membership

With similar ideas as with the vertex membership version, we reduce the $\mathcal{H}$-Packing with t-Edge Membership to a kernel with the following kernelization algorithm: Algorithm 2 with input G, $\mathcal{H}$, k, *r-Set Packing* *with t-Membership*, Transformation 3, Algorithm 9, and Transformation 4. The output of this algorithm is a reduced graph G'.

By Transformation 4, $G' = G[V(\mathcal{U}')]$, and by Algorithm 9 $|\mathcal{U}'| = O\big(r^3 \left(\frac{r}{t}\right)^{r-3} k^{r-1}\big)$. By Lemma 18, we can state.

Theorem 40. *The $\mathcal{H}$-Packing with t-Edge Membership problem has a kernel with $O\big(r^3 \ \left(\frac{r}{t}\right)^{r-3} \ k^{r-1}\big)$ vertices where $r = m(\mathcal{H})$.*

7.4 Conclusions

In this chapter, we obtained improved kernel sizes for all our pairwise overlap and membership problems with respect to the kernels obtained via PPTs in the previous chapter. This improvement comes from reducing the problems directly instead of transforming them to the r'-Set Packing problem ($r' > r$). As expected, tailored kernelization algorithms for the particular problems could be more involved and the analysis of correctness and kernel sizes is also more technical.

Although t is considered a fixed-constant, it is still interesting to see the effect of its value in both kernel sizes. In the $O(r^r k^{r-t-1})$ kernels for the pairwise overlap problems, this value of t affects directly the exponent of parameter k, while in the $O\big(r^3 \left(\frac{r}{t}\right)^{r-3} k^{r-1}\big)$ kernels of the membership problems, the value of t only affects as a divisor factor. For both cases, this effect of t matches the intuition that the larger the overlap or membership in the solution, the smaller the kernel should be. That is, we can "shrink" more the instance as we are allowing for more overlap or membership between the members of the solution.

Finally notice that for $t = 0$ and $t = 1$ for pairwise overlap and membership problems, respectively, our kernel sizes match the $O(r^r k^{r-1})$ [114, 1] kernel size of packing disjoint sets of size at most r (or graphs of order at most r). Recall that for those values of t our problems become equivalent to packing disjoint objects (sets or graphs, depending on the problem).

It remains of course to answer the question of how "good" the kernels obtained in these sections really are. Specifically, it would be ideal to develop kernelization lower bounds for all of our problems. In addition, it would be interesting to develop kernelization algorithms for other parameter combinations.

Part III

Fixed-Parameter Algorithms

Chapter 8

On Fixed-Parameter Algorithms

Previously, we focused on reducing packing problems with pairwise overlap and membership to a kernel problem. We obtained such kernels with two different techniques: polynomial parametric transformations [16] and classical reduction [44, 117]. Obtaining a problem kernel implies obtaining an immediate fixed-parameter algorithm, i.e., running a brute-force algorithm on the kernel.

In this chapter, however, we are interested on designing better algorithms than brute-force algorithms. Instead of exploring all possible solutions, we will construct a search tree that will contain a solution for our problems at one of the leaves of the tree (if one exists). To achieve an FPT-algorithm with a search tree, the size of the tree (depth and branch-in) must be bounded by $f(k)$, where f is a computable function and k is the parameter. In addition, the time spend at each node must be polynomial in n. Bounded search trees consists on a powerful technique for designing fixed-parameter algorithms [44, 117].

We obtain additional FPT-algorithms for our packing problems in a more indirect way. Recall that in Chapter 6, we transform the instances of the packing problems with overlap into instances of the problem of packing disjoint sets. Thus in this chapter, we show that we can obtain a solution for our problems by running FPT-algorithms for the r-Set Packing problem (packing disjoint sets).

This chapter is organized as follows. We analyze the running time of brute-force algorithms on our problem kernels for the r-Set Packing with t-Overlap and t-Membership in Sections 8.1.1 and 8.2.1, respectively. Although these results are straightforward, we still included them for completeness. In Sections 8.1.2 and 8.2.2, we design FPT-algorithms for the r-Set Packing with t-Overlap and t-Membership, respectively, which are based on bounded search trees in combination with a greedy algorithm. Additionally in Sections 8.1.3 and 8.2.3, we discuss how to achieve efficient (the best ones from this chapter) FPT-algorithms for our problems via FPT-algorithms for the problem of packing disjoint sets. Table 8.1 summarizes the running times obtained for each algorithm. Finally in Sections 8.1.4 and 8.2.4, we point out how our FPT-algorithms for the r-Set Packing with t-Overlap and t-Membership translate into FPT-algorithms for the graph versions of those problems.

r-Set Packing with	FPT-Algorithm	Running Time
	Brute-Force on kernel	$O(r^{2k}k^{(r-t-1)^2k} + n^r)$
t-Overlap	Bounded Search Tree (on kernel)	$O(r^{r^2+rk}k^{(r-t-1)(k+r)+2} + n^r)$
	"Color-Coding"	$O(2^{O((c+1)(r^{t+1}+1)k)} + n^r + n^{t+1})$
	Brute-Force on kernel	$O\left(\left(r^3\left(\frac{r}{t}\right)^{r-3}k^{r-1}\right)^{2k} + n^r\right)$
t-Membership	Bounded Search Tree (on kernel)	$O((2^r t^{k-(r(r-3))}(rk)^{r^2+rk}) + n^r)$
	"Color-Coding"	$O(2^{O((c+1)(r+1)k)} + n^r)$

Table 8.1: Summary of results presented in Chapter 8

8.1 On FPT-Algorithms for Packing Problems with Pairwise Overlap

In this section, we will develop fixed-parameter algorithms for the r-Set Packing and $\mathcal{H}$-Packing with t-Overlap problems. First, we simply provide the running time of a brute-force search algorithm on the kernel obtained in Section 7.2.1. After that, we develop an FPT-algorithm using a bounded search tree in combination with a greedy algorithm. This algorithm does not necessarily run on a kernel. Finally, we point out how to obtain a faster FPT-algorithm by running an FPT-algorithm on the constructed instances of the $(\binom{r}{t+1}+1)$-Set Packing problem.

8.1.1 A Naive Algorithm

A simple way to obtain a (k, r, t)-set packing of the instance $(\mathcal{U}, \mathcal{S}, k)$ is by running a brute-force algorithm on $\mathcal{S}$. That is, finding all possible combinations of size k on $\mathcal{S}$ and check whether a combination is a (k, r, t)-set packing or not.

In order that this algorithm be fixed-parameter tractable by the parameter k, the size of $\mathcal{S}$ must be bounded by $f(k)$, where f is a computable function. Thus, we will run this algorithm on the reduced $\mathcal{S}$ from Theorem 29. The size of the reduced $\mathcal{S}$ is $O(r^{r^2}k^{(r-t-1)^2})$. Hence, the running time of the brute-force algorithm on the kernel is $O(r^{r^2k}k^{(r-t-1)^2k})$.

Lemma 53. *The r-Set Packing with t-Overlap problem can be solved with brute force in $O(r^{r^2k}k^{(r-t-1)^2k} + n^r)$ running time, when $|\mathcal{S}| = O(r^{r^2}k^{(r-t-1)^2})$.*

8.1.2 A Bounded Search Tree with Greedy Localization Algorithm

We will develop in this section a search tree algorithm for the r-Set Packing with t-Overlap problem. This algorithm uses a greedy routine to bound the number of children at each node of the tree. The FPT-algorithm (called BST-ALGORITHM t-OVERLAP) for the r-Set Packing with t-Overlap has three main components: INITIALIZATION, GREEDY, and BRANCHING.

Let us explain first the global structure of BST-ALGORITHM t-OVERLAP. This algorithm starts by computing a maximal (r, t)-set packing $\mathcal{M}$ from $\mathcal{S}$. If $|\mathcal{M}| \geq k$, then $\mathcal{M}$ is a (k, r, t)-set packing, and the

BST-ALGORITHM t-OVERLAP stops. Otherwise, we create a search tree T where at each node i, there is a collection of sets $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_k\}$ with $s^i_j \subseteq S$ for some $S \in \mathcal{S}$. The children of the root of T are created according to a procedure called INITIALIZATION.

After that for each node i of T, a routine called GREEDY will attempt to find a (k, r, t)-set packing using $\mathbf{Q^i}$. If GREEDY succeeds, then BST-ALGORITHM t-OVERLAP stops. Otherwise, the next step is to create children of the node i using the procedure BRANCHING. Each child l of the node i will have a collection $\mathbf{Q^l}$ that is the same as $\mathbf{Q^i}$ with the difference that one of the sets of $\mathbf{Q^l}$ is increased by one element. The BST-ALGORITHM t-OVERLAP will repeat GREEDY in these children. Eventually, BST-ALGORITHM t-OVERLAP either finds a solution at one of the leaves of the tree or determines that it is not possible to find one.

We next explain the three main components of BST-ALGORITHM t-OVERLAP individually. Let us start with the INITIALIZATION routine. As we will see in Lemma 54, if there is a solution $\mathcal{K} = \{S^*_1, \ldots, S^*_k\}$, each S^*_j contains at least $t+1$ elements which are contained in a set $S \in \mathcal{M}$ (recall that $\mathcal{M}$ is a maximal (r, t)-set packing). Furthermore, these $t+1$ elements are only contained in S^*_j (and S); otherwise it would be a pair of sets in $\mathcal{K}$ (and $\mathcal{M}$) overlapping in more than t elements. Thus, we create a collection $\mathcal{M}_{t+1}$ of sets that contains all possible subsets of $t+1$ elements of S, for each $S \in \mathcal{M}$. The root of the search tree will have a child i for each possible combination of k sets from $\mathcal{M}_{t+1}$.

At each node i, the GREEDY routine returns a collection of sets $\mathbf{Q^{gr}}$. Initially, $\mathbf{Q^{gr}} = \emptyset$ and $j = 1$. At iteration j, GREEDY searches for a set S that contains s^i_j and that S does not overlap in more than t elements with any set in $\mathbf{Q^{gr}}$. GREEDY searches for S in a collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$. $\mathcal{S}(s^i_j, \mathbf{Q^i}, t) \subseteq \mathcal{S}(s^i_j)$ is the subcollection of $\mathcal{S}(s^i_j)$ where for each $S' \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ each pair of sets in $(\mathbf{Q^i} \backslash s^i_j) \cup S'$ does not overlap in more than t elements. If such set S exists, GREEDY adds S to $\mathbf{Q^{gr}}$, i.e., $\mathbf{Q^{gr}} = \mathbf{Q^{gr}} \cup S$ and continues with iteration $j = j+1$. Otherwise, GREEDY stops executing and returns $\mathbf{Q^{gr}}$. If $|\mathbf{Q^{gr}}| = k$, then $\mathbf{Q^{gr}}$ is a (k, r, t)-set packing and BST-ALGORITHM t-OVERLAP stops. If the collection $\mathbf{Q^i}$ cannot be propagated into a solution (Lemma 56), GREEDY returns $\mathbf{Q^{gr}} = \infty$.

The BRANCHING procedure executes every time that GREEDY does not return a (k, r, t)-set packing but $\mathbf{Q^i}$ could be propagated into one. That is, $\mathbf{Q^{gr}} \neq \infty$ and $|\mathbf{Q^{gr}}| < k$. Let $j = |\mathbf{Q^{gr}}| + 1$ and s^i_j be the *j-th* set in $\mathbf{Q^i}$. GREEDY stopped at j because each set $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ overlaps in more than t elements with at least one set in $\mathbf{Q^{gr}}$. We will use the conflicting elements between $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ and $\mathbf{Q^{gr}}$ to create children of the node i. Let I^* be the set of those conflicting elements which are obtained as $I^* = (\bigcup (S \cap S')) \backslash (val(\mathbf{Q^i}))$ for each pair $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ and $S' \in \mathbf{Q^{gr}}$ that overlap in more than t elements. BRANCHING creates a child l of the node i for each element $u_l \in I^*$. The collection $\mathbf{Q^l}$ of child l is the same as the collection $\mathbf{Q^i}$ of its parent i with the update of the set s^i_j as $s^i_j \cup u_l$, i.e., $\mathbf{Q^l} = \{s^i_1, \ldots, s^i_{j-1}, s^i_j \cup u_l, s^i_{j+1}, \ldots, s^i_k\}$.

Example 12. *Consider the following instance of the r-Set Packing with t-Overlap problem (*$r = 4$*,* $t = 1$*,* $k = 4$*).*

$$\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$$

$$\begin{aligned}\mathcal{S} = \{&S_1 = \{1, 2, 3, 4\}, S_2 = \{1, 3, 9, 10\}, S_3 = \{1, 3, 10, 11\},\\ &S_4 = \{3, 4, 11, 12\}, S_5 = \{5, 6, 7, 8\}, S_6 = \{5, 6, 7, 4\},\\ &S_7 = \{6, 8, 13, 14\}\}\end{aligned}$$

A maximal (r,t)-set packing is $\mathcal{M} = \{S_1, S_5\}$. Thus,

$$\begin{aligned}\mathcal{M}_{t+1} =& \{\{1,3\},\{1,4\},\{1,2\},\{2,3\},\{2,4\},\{3,4\},\\ &\{5,6\},\{5,7\},\{5,8\},\{6,7\},\{6,8\},\{7,8\}\}\end{aligned}$$

One example of a child of the root is a node i with collection $\mathbf{Q^i} = \{s_1^i = \{1,3\}, s_2^i = \{3,4\}, s_3^i = \{5,6\}, s_4^i = \{6,8\}\}$. Consider the execution of Greedy *at this node. At $j = 1$,* Greedy *tries to find a set that contains $s_1^i = \{1,3\}$. Recall that* Greedy *searches in the collection $\mathcal{S}(s_1^i, \mathbf{Q^i}, t)$. All the sets in $\mathcal{S}$ that contain s_1^i are $\mathcal{S}(s_1^i) = \{S_1, S_2, S_3\}$. Observe however that S_1 overlaps in $t+1 = 2$ elements with $s_2^i = \{3,4\}$. Thus, $\mathcal{S}(s_1^i, \mathbf{Q^i}, t) = \{S_2, S_3\}$. Now, assume that* Greedy *chooses S_3 and $\mathbf{Q^{gr}} = \{S_3\}$. Let us look at the next execution of* Greedy*, $j = 2$.* Greedy *searches for a set in $\mathcal{S}(s_2^i, \mathbf{Q^i}, t) = \{S_4\}$ (S_1 also includes s_2^i but conflicts with s_1^i). However, S_4 overlaps in $t+1$ elements ($\{3,11\}$) with the set S_3 in $\mathbf{Q^{gr}}$. Thus,* Greedy *stops executing.*

After that Branching *executes at that node and computes I^*. In this case, there is only one set in $\mathbf{Q^{gr}} = \{S_3\}$ and $\mathcal{S}(s_2^i, \mathbf{Q^i}, t) = \{S_4\}$. Since $|S_4 \cap S_3| = |\{3,11\}| \geq t+1$, $I^* = \{11\}$. Note that the element 3 is not in I^* as it is already in s_2^i. After that,* Branching *creates a children l of the node i with collection $\mathbf{Q^l} = \{\{1,3\},\{3,4,11\},\{5,6\},\{6,8\}\}$.*

Correctness.

We next show that BST-algorithm t-Overlap finds a (k,r,t)-set packing, if $\mathcal{S}$ has one. Let us begin with a lemma that shows how a maximal (r,t)-set packing and a (k,r,t)-set packing intersect. This will help us to show the correctness of the Initialization routine.

Lemma 54. *Let $\mathcal{M}$ and $\mathcal{K}$ be a maximal (r,t)-set packing and a (k,r,t)-set packing, respectively. We claim that any $S^* \in \mathcal{K}$ overlaps with some $S \in \mathcal{M}$ in at least $t+1$ elements, i.e., $|S^* \cap S| \geq t+1$. Furthermore, there is no pair $S_i^*, S_j^* \in \mathcal{K}$ for $i \neq j$ that overlaps in the same set of elements with S for all S with $|S_i^* \cap S| \geq t+1$, i.e., $(S_i^* \cap S) \neq (S_j^* \cap S)$.*

Proof. Assume for contradiction that there is a set $S^* \in \mathcal{K}$ such that for any set $S \in \mathcal{M}$, the overlap between them is at most t, i.e., $|S^* \cap S| \leq t$. However, in this case, we could add S^* to $\mathcal{M}$, and $\mathcal{M} \cup S^*$ is a $(|\mathcal{M}|+1, r, t)$ set packing contradicting the maximality of $\mathcal{M}$.

To prove the second part of the lemma, assume for contradiction that there is a pair $S_i^*, S_j^* \in \mathcal{K}$ that overlaps in the same set of vertices for all $S \in \mathcal{M}$. However, by the first part of the lemma, we know that there is at least one $S \in \mathcal{M}$ such that $|S_i^* \cap S| \geq t+1$. This would imply that $|S_i^* \cap S_j^*| \geq t+1$, a contradiction since the r-Set Packing with t-Overlap problem does not allow overlap greater than t. $\square$

A collection $\mathbf{Q^i} = \{s_1^i, \ldots, s_j^i, \ldots, s_k^i\}$ is a *partial-solution* of a (k,r,t)-set packing $\mathcal{K} = \{S_1^*, \ldots, S_k^*\}$ if and only if $s_j^i \subseteq S_j^*$, for $1 \leq j \leq k$. Now, we show the correctness of the Initialization routine.

Lemma 55. *If there exists at least one (k,r,t)-set packing of $\mathcal{S}$, at least one of the children of the root will have a partial-solution.*

Proof. By Lemma 54, every set in $\mathcal{K}$ contains at least one set of $t+1$ elements of a set in $\mathcal{M}$. Furthermore, this set of $t+1$ elements will be contained in only one set of $\mathcal{K}$. The collection $\mathcal{M}_{t+1}$ contains each possible subset of $t+1$ elements for each set in $\mathcal{M}$. Since we created a node for each selection of k sets from $\mathcal{M}_{t+1}$, i.e., $\binom{\mathcal{M}_{t+1}}{k}$, the lemma follows. ☐

The next lemma states that BST-ALGORITHM t-OVERLAP correctly stops attempting to propagate a collection $\mathbf{Q^i}$.

Lemma 56. *The collection $\mathbf{Q^i}$ is not a partial-solution if either: i. there is a pair of distinct sets in $\mathbf{Q^i}$ that overlaps in more than t elements, or ii. for some $s^i_j \in \mathbf{Q^i}$, $\mathcal{S}(s^i_j, \mathbf{Q^i}, t) = \emptyset$.*

Proof. (i) Suppose otherwise that $\mathbf{Q^i}$ is a partial-solution, but s^i_j, s^i_l $(j \neq l)$ overlap in more than t elements. Since $\mathbf{Q^i}$ is a partial-solution, $s^i_j \subseteq S^*_j$ and $s^i_l \subseteq S^*_l$ where S^*_j,$S^*_l \in \mathcal{K}$ and $\mathcal{K}$ is a (k, r, t)-set packing. However, our assumption implies that S^*_j and S^*_l also overlap in more than t elements, a contradiction.

(ii) To prove the second part of the lemma, we will prove the next stronger claim.

Claim 13. *If $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a partial-solution, then $S^*_j \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ for each $1 \leq j \leq k$.*

Proof. Assume for contradiction that $\mathbf{Q^i}$ is a partial-solution but $S^*_j \notin \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ for some j. If $\mathbf{Q^i}$ is a partial-solution then $s^i_j \subseteq S^*_j \in \mathcal{K}$ and $((\mathbf{Q^i} \backslash s^i_j) \cup S^*_j)$ is a partial-solution as well. The set $S^*_j \in \mathcal{S}(s^i_j)$ and $\mathcal{S}(s^i_j, \mathbf{Q^i}, t) \subseteq \mathcal{S}(s^i_j)$. The only way that S^*_j would not be in $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ is if there is at least one pair of sets in $((\mathbf{Q^i} \backslash s^i_j) \cup S^*_j)$ that overlaps in more than t elements but then $((\mathbf{Q^i} \backslash s^i_j) \cup S^*_j)$ would not be a partial-solution, a contradiction to (i). ☐

☐

BRANCHING creates at least one child whose collection is a partial-solution, if the collection of the parent is a partial-solution as well.

Lemma 57. *If $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a partial-solution, then there exists at least one $u_l \in I^*$ such that $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j \cup u_l, \ldots, s^i_k\}$ is a partial-solution.*

Proof. Assume to the contrary that $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a partial-solution but that there exists no element $u_l \in I^*$ such that $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j \cup u_l, \ldots, s^i_k\}$ is a partial-solution. This can only be possible if $(S^*_j \backslash s^i_j) \cap I^* = \emptyset$.

First, given that $\mathbf{Q^i}$ is a partial-solution $S^*_j \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ (Claim 13). In addition, there exists at least one set $S' \in \mathbf{Q^{gr}}$ such that $|S^*_j \cap S'| \geq t+1$; otherwise, S^*_j would have been selected by GREEDY. By the computation of I^*, $(S^*_j \cap S') \subseteq I^*$.

Now it remains to show that $|s^i_j \cap S'| \leq t$. In this way, at least one element of $(S^*_j \backslash s^i)$ will be in I^*. Assume otherwise by contradiction. Recall that S' contains some set s^i_h of $\mathbf{Q^i}$ (for some $h \leq j$). If S' was selected by GREEDY, then $S' \in \mathcal{S}(s^i_h, \mathbf{Q^i}, t)$. However, this immediately implies that $|s^i_j \cap S'| \leq t$. ☐

By the previous lemmas, we can state:

Theorem 41. *The* BST-ALGORITHM t-OVERLAP *finds a (k, r, t)-set packing of $\mathcal{S}$, if $\mathcal{S}$ has at least one.*

Running Time

Lemma 58. *The root has at most* $\left(\frac{e^2 r}{t+1}\right)^{k(t+1)}$ *children.*

Proof. There are $|\binom{S}{t+1}|$ distinct sets of $t+1$ elements chosen from a set $S \in \mathcal{M}$. The collection $\mathcal{M}_{t+1}$ contains all possible subsets of $t+1$ elements from S, for each $S \in \mathcal{M}$. Since $|\mathcal{M}| \leq k-1$ and $|S| \leq r$, there are at most $(k-1)\binom{r}{t+1}$ sets in $\mathcal{M}_{t+1}$. Since we have $1 \leq t+1 \leq r$, then from the Stirling's approximation, $\binom{r}{t+1} \leq \left(\frac{er}{t+1}\right)^{t+1}$. Thus, there are at most $(k-1)\left(\frac{er}{t+1}\right)^{t+1}$ sets in $\mathcal{M}_{t+1}$.

From the root of the search tree, we create a node i for each possible selection of k sets from $\mathcal{M}_{t+1}$, i.e., $|\binom{\mathcal{M}_{t+1}}{k}|$. Hence, $\binom{(k-1)\left(\frac{er}{t+1}\right)^{t+1}}{k} \leq \left(\frac{e(k-1)\left(\frac{er}{t+1}\right)^{t+1}}{k}\right)^k \leq \left(\frac{e^2 r}{t+1}\right)^{k(t+1)}$. □

Lemma 59. *The height of the search tree is at most* $(r-t-1)k$.

Proof. If Greedy does not return a (k,r,t)-set packing but $\mathbf{Q^{gr}} \neq \infty$, Branching creates at least one child l of a node i. Recall that one set in the collection of this child gets increased by one element. In other words, each level on the tree corresponds to one element added to a set of $\mathbf{Q^i}$. Since at the first level of the tree each set has $t+1$ elements, any set of $\mathbf{Q^i}$ could be completed into a set of $\mathcal{S}$ in at most $r-(t+1)$ levels (which are not necessarily consecutive).

Therefore, we need at most $(r-t-1)k$ levels to propagate $\mathbf{Q^i}$ into a (k,r,t)-set packing. □

Lemma 60. *A node i at level h can have at most $r(k-1)$ children if $h \leq (r-t-2)k$ and at most $r(k-m-1)$ children where $m = h-(r-t-2)k$ if $h > (r-t-2)k$.*

Proof. There is a child of the node i for each element in I^*. Recall that $I^* = \bigcup (S \cap S')$ for each pair $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ and $S' \in \mathbf{Q^{gr}}$ that overlaps in more than t elements. Therefore, $|I^*| \leq r|\mathbf{Q^{gr}}|$.

The greedy algorithm needs to complete only the sets in $\mathbf{Q^i}$ that are not sets of $\mathcal{S}$. Therefore, $|\mathbf{Q^{gr}}| \leq |\mathbf{Q^i}| - m$ where m is the number of sets of $\mathbf{Q^i}$ that are already sets from $\mathcal{S}$.

Assuming all the sets of $\mathbf{Q^i} - m$ were completed by greedy but the last one, i.e., $|\mathbf{Q^{gr}}| \leq |\mathbf{Q^i}| - m - 1$, then $|I^*| \leq r|\mathbf{Q^{gr}}| \leq r(|\mathbf{Q^i}| - m - 1) \leq r(k-m-1)$.

Now, we need to determine how many sets of $\mathbf{Q^i}$ are already sets of $\mathcal{S}$ at level h, i.e., the value of m. Since a set s^i_j can be completed into a set in at most $r-(t+1)$ levels but these are not necessarily consecutive, then we cannot guarantee that $\frac{h}{r-(t+1)}$ is the number of sets of $\mathcal{S}$ at level h. Therefore, in the worst-case one element is added to each set of $\mathbf{Q^i}$ level by level. In this way, in at most $(r-t-2)k$ levels every set of $\mathbf{Q^i}$ could have $r-1$ elements, and at level $(r-t-2)k+1$ we could obtain the first set of $\mathcal{S}$. After that level, the remaining sets of $\mathbf{Q^i}$ are completed into sets of $\mathcal{S}$ by adding one element. □

Theorem 42. *The r-Set Packing with t-Overlap problem can be solved with* BST-algorithm t-Overlap *in* $O(r^{rk} k^{(r-t-1)k+2} n^r)$ *time.*

Proof. By Lemmas 59 and 60, the size of the tree is

$$\left(\frac{e^2 r}{t+1}\right)^{k(t+1)} \prod_{i=1}^{(r-t-2)k} r(k-1) + \prod_{i=1}^{k-r+t} r(k-i)$$

$$< \left(\frac{e^2 r}{t+1}\right)^{k(t+1)} (r(k-1))^{(r-t-1)k-1}.$$

A maximal solution $\mathcal{M}$ can be computed in time $O(krn^r)$ which is also the required time to compute the collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$. The greedy algorithm runs in $O(k^2rn^r)$. □

We can run BST-ALGORITHM t-OVERLAP on the $O(r^r k^{r-t-1})$ kernel obtained in Section 7.2.1. This will imply an $O(r^{r^2+rk} k^{(r-t-1)(k+r)+2} + n^r)$ running time.

8.1.3 A "Color-Coding" Algorithm

The color-coding technique introduced by N. Alon, R. Yuster and U. Zwick [3] has been successfully applied to obtain algorithms with $O(2^k)$ running time for several problems, including the r-Set Packing problem. The basic idea behind this technique applied to r-Set Packing is to randomly color the elements of the universe and "hope" that all elements in a $(k, r, 0)$-set packing receive a different color.

We would like to highlight the observation that we could run an FPT-algorithm based on this technique (or any other technique) in our constructed instances for the r'-Set Packing problem, where $r' > r$ (Section 6.2.1). However, we will be paying a price in the running time.

There exist several FPT-algorithms involving color-coding for the r-Set Packing problem [53, 90, 101, 25, 133]. With exception of [53], all of these algorithms are stated and analyzed for the specific case of $r = 3$. Therefore, we will discuss how to run algorithm [53] on our constructed instances for the r'-Set Packing problem. It is important to mention that the analysis of the running time of the FPT-algorithm [53] is based on running this FPT-algorithm on a problem kernel, which is also obtained in [53].

First, we will transform an instance $(\mathcal{S},\mathcal{U},k)$ of the r-Set Packing with t-Overlap to an instance $(\mathcal{S}^T,\mathcal{U}^T,k)$ of r'-Set Packing problem with $r' = \binom{r}{t+1}+1$ by running Transformation 11. After that, we reduce $(\mathcal{S}^T,\mathcal{U}^T,k)$ to a kernel $(\mathcal{S'}^T,\mathcal{U'}^T,k)$ with kernelization algorithm [53]. Finally, we obtain a $(k, r', 0)$-set packing of $\mathcal{S'}^T$ by running the color-coding algorithm [53]. By Lemma 35, we can transform a $(k, r', 0)$-set packing of $\mathcal{S}^T$ into a (k, r, t)-set packing of $\mathcal{S}$. Algorithm 11 summarizes these steps. Since algorithm [53] runs in $O(|\mathcal{U}^T| + 2^{O((c+1)(r'k))})$ (where $c \geq 131.76$ [101]) and by Transformation 11 $|\mathcal{U}^T| = O(n^r + n^{t+1})$, we can hence state:

Lemma 61. *The r-Set Packing with t-Overlap problem can be solved with Algorithm 11 in* $O((n^r + n^{t+1}) + 2^{O((c+1)(r^{t+1}+1)k)})$.

This methodology would work as well by running any other FPT-algorithm for r'-Set Packing instead of [53]. Also, it is not necessary to reduce the instance of the r'-Set Packing to a kernel before running such an algorithm. Algorithm 11 does it in that way because the analysis of the running time of [53] considers that assumption.

Algorithm 11 "Color-Coding" - r-Set Packing with t-Overlap

```
Input: An instance $(\mathcal{S}, \mathcal{U}, k)$ of the $r$-Set Packing with $t$-Overlap problem
Output: A $(k, r, t)$-set packing $\mathcal{K} \subseteq \mathcal{S}$, if $\mathcal{S}$ has one.
1: Transform the instance $(\mathcal{S}, \mathcal{U}, k)$ of $r$-Set Packing with $t$-Overlap to an instance $(\mathcal{S}^T, \mathcal{U}^T, k)$ of $r'$-Set Packing problem, where $r' = (\binom{r}{t+1} + 1)$, by running Transformation 11.
2: Reduce the instance $(\mathcal{S}^T, \mathcal{U}^T, k)$ to a kernel $(\mathcal{S}'^T, \mathcal{U}'^T, k)$ by running kernelization algorithm [53].
3: Obtain a $(k, r', 0)$-set packing of $\mathcal{S}'^T$ by running the color-coding algorithm [53].
4: Re-interpret this $(k, r', 0)$-set packing as a $(k, r, t)$-set packing $\mathcal{K}$ of $\mathcal{S}$ as in Lemma 35.
5: Return $\mathcal{K}$
```

8.1.4 FPT-Algorithms for the Graph Packing Versions

To obtain FPT-algorithms for the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap problems, we will follow a similar methodology as with the kernelization algorithms in Sections 6.2.2 and 7.2.2. First, we transform an instance of the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap problems to an instance of the r-Set Packing with t-Overlap problem using Transformations 1 and 3, respectively. After that, we run any of the FPT-algorithms discussed previously. Any of these algorithms outputs a (k, r, t)-set packing, if the input instance has one. By Lemmas 20 and 24, we can transform a (k, r, t)-set packing into a (k, r, t)-$\mathcal{H}$-Packing and a (k, r, t)-Edge-$\mathcal{H}$-Packing, respectively. Hence, we can state:

Lemma 62. *The $\mathcal{H}$-Packing with t-Overlap problem can be solved in $O((n^r + n^{t+1}) + 2^{O((c+1)(r^{t+1}+1)k)})$, where $r = r(\mathcal{H})$.*

Lemma 63. *The $\mathcal{H}$-Packing with t-Edge Overlap problem can be solved in $O((n^r+n^{t+1})+2^{O((c+1)(r^{t+1}+1)k)})$, where $r = m(\mathcal{H})$.*

8.2 On FPT-Algorithms for Packing Problems with Membership

This section follows similar structure as Section 8.1. First, we analyze a brute-force algorithm for the r-Set Packing with t-Membership problem. After that, we adapt the search tree algorithm of Section 8.1.2 to provide a (k, r, t)-set membership. We also discuss how we can obtain immediate FPT-algorithms for the r-Set Packing with t-Membership via FPT-algorithms for the problem of packing disjoint sets. We conclude this section showing how to solve the $\mathcal{H}$-Packing with t-Membership and t-Overlap using the FPT-algorithms for the set version of the problem.

8.2.1 A Naive Algorithm

A naive algorithm for the r-Set Packing with t-Membership simply searches for all possible combinations of size k from $\mathcal{S}$. When this algorithm runs on a kernel, it becomes a fixed-parameter tractable algorithm. The collection $\mathcal{S}$ is reduced to $O\big(\big(r^3\big(\frac{r}{t}\big)^{r-3}k^{r-1}\big)^2\big)$ by Algorithm 9 (Theorem 36). In this way,

Lemma 64. *The r-Set Packing with t-Membership problem can be solved with brute force in $O\big(\big(r^3\big(\frac{r}{t}\big)^{r-3}k^{r-1}\big)^{2k} + n^r\big)$ running time, when $|\mathcal{S}| = O\big(\big(r^3\big(\frac{r}{t}\big)^{r-3}k^{r-1}\big)^2\big)$.*

8.2.2 A Bounded Search Tree with Greedy Localization Algorithm

We will adapt BST-ALGORITHM t-OVERLAP to return a (k, r, t)-set membership of $\mathcal{S}$ (if exists) instead of a (k, r, t)-set packing. We will call this modified algorithm BST-ALGORITHM t-MEMBERSHIP. Even though the general structure of both algorithms is the same, there are clear differences in the individual routines of each of them. For explanatory purposes, we will explain this modified algorithm entirely instead of re-using previous explanations.

Let us briefly recall how the algorithm works at high level. Once again, we compute first a maximal solution, i.e., a maximal (r, t)-set membership $\mathcal{M}$ from $\mathcal{S}$. If $|\mathcal{M}| \geq k$, then $\mathcal{M}$ is a (k, r, t)-set membership, and BST-ALGORITHM t-MEMBERSHIP stops. If not, we create again a search tree T where at each node i, there is a collection of sets $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_k\}$ with $s^i_j \subseteq S$ for some $S \in \mathcal{S}$. The children of the root of T will be created with INITIALIZATION.

After that for each node i of T, GREEDY tries to find a (k, r, t)-set membership using $\mathbf{Q^i}$. If GREEDY finds one then BST-ALGORITHM t-MEMBERSHIP stops. In the other case, this algorithm will create children of the node i using BRANCHING. Each child l of the node i will have a collection $\mathbf{Q^l}$ that is the same as $\mathbf{Q^i}$ with the difference that one of the sets of $\mathbf{Q^l}$ is increased by one element. BST-ALGORITHM t-MEMBERSHIP will repeat GREEDY in these children. Eventually, BST-ALGORITHM t-MEMBERSHIP either finds a solution at one of the leaves of the tree or determines that it is not possible to find one.

We now explain each routine of BST-ALGORITHM t-MEMBERSHIP separately. Let us start with the INITIALIZATION routine. By Lemma 65, if there is a solution $\mathcal{K} = \{S^*_1, \ldots, S^*_k\}$ then each S^*_j contains at least one element from $val(\mathcal{M})$, and each element of $val(\mathcal{M})$ could be in at most t sets of $\mathbf{Q^i}$ ($t \leq k$).Therefore, to initialize the children of the root, we create a set $\mathcal{M}_t$ that contains t copies of each element in $val(\mathcal{M})$. That is, per each element $u \in val(\mathcal{M})$ there are t copies $u_1 \ldots u_t$ in $\mathcal{M}_t$, in this way $|\mathcal{M}_t| = t|val(\mathcal{M})|$. The root will have a child i for each possible selection of k elements from $\mathcal{M}_t$. The collection $\mathbf{Q^i}$ is initialized with this selection. After this initialization, we remove the indices from the elements in the sets of each $\mathbf{Q^i}$. Thus, if for example a collection $\mathbf{Q^i} = \{\{u_1\}, \{u_2\}, \{u_t\}, \{a_1\}, \{b_1\}\}$ after removing the indices $\mathbf{Q^i} = \{\{u\}, \{u\}, \{u\}, \{a\}, \{b\}\}$.

At each node i, the GREEDY routine returns a collection of sets $\mathbf{Q^{gr}}$. Initially, $\mathbf{Q^{gr}} = \emptyset$ and $j = 1$. At iteration j, GREEDY searches for a set S that contains s^i_j subject to three conditions: (i) S is not equal to any set in $(\mathbf{Q^i} \backslash s^i_j)$, (ii) S is not already in $\mathbf{Q^{gr}}$, (iii) and each element of S is contained in at most t sets in $(\mathbf{Q^{gr}} \cup S \cup \bar{\mathbf{Q}}^{\mathbf{i}})$, where $\bar{\mathbf{Q}}^{\mathbf{i}}$ is the sub-collection of $\mathbf{Q^i}$ defined as $\bar{\mathbf{Q}}^{\mathbf{i}} = \{s^i_{j+1}, \ldots, s^i_k\}$. The first two conditions guarantees that all sets in a solution will be distinct while the third one ensures that each element will be contained in at most t sets of a solution. GREEDY searches for S in the collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$. $\mathcal{S}(s^i_j, \mathbf{Q^i}, t) \subseteq \mathcal{S}(s^i_j)$ is the subcollection of $\mathcal{S}(s^i_j)$ where for each $S' \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$, each element in $val((\mathbf{Q^i} \backslash s^i_j) \cup S')$ is contained in at most t sets of $((\mathbf{Q^i} \backslash s^i_j) \cup S')$. If such set S exists, GREEDY adds S to $\mathbf{Q^{gr}}$, i.e., $\mathbf{Q^{gr}} = \mathbf{Q^{gr}} \cup S$ and continues with iteration $j = j + 1$. Notice that at each iteration $\mathbf{Q^{gr}}$ is a $(|\mathbf{Q^{gr}}|, r, t)$-set membership. Otherwise, GREEDY stops executing and returns $\mathbf{Q^{gr}}$. If $|\mathbf{Q^{gr}}| = k$ then $\mathbf{Q^{gr}}$ is a (k, r, t)-set membership and BST-ALGORITHM t-MEMBERSHIP stops. If $\mathbf{Q^i}$ cannot be propagated into a solution (Lemma 67), GREEDY returns $\mathbf{Q^{gr}} = \infty$. Observe that the set S that will be added to $\mathbf{Q^{gr}}$ follows different conditions than the conditions in the GREEDY routine of BST-ALGORITHM t-OVERLAP. In addition, the meaning of the collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ is different.

The BRANCHING procedure executes every time that GREEDY does not return a (k, r, t)-set membership but $\mathbf{Q^i}$ could be propagated into one. That is, $\mathbf{Q^{gr}} \neq \infty$ and $|\mathbf{Q^{gr}}| < k$. Let $j = |\mathbf{Q^{gr}}| + 1$ and s^i_j be the

j-*th* set in $\mathbf{Q^i}$. Greedy stopped at j because each set $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$ failed one of the three conditions specified above. For the first condition, Greedy returns $\mathbf{Q^{gr}} = \infty$ (Lemma 67). Failing the second condition implies that each element of S is contained in one set in $\mathbf{Q^{gr}}$ while failing the third one implies that at least one element $u \in S$ is contained in more than t sets of $(\mathbf{Q^{gr}} \cup S \cup \bar{\mathbf{Q}}^{\mathbf{i}})$. Let I^* be the set of elements in S (for each $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, t)$) either contained in a set of $\mathbf{Q^{gr}}$ or contained in $t+1$ sets in $(\mathbf{Q^{gr}} \cup S \cup \bar{\mathbf{Q}}^{\mathbf{i}})$. Branching creates a child l of the node i for each element $u_l \in I^*$. The collection $\mathbf{Q^l}$ of child l is the same as the collection $\mathbf{Q^i}$ of its parent i with the update of the set s^i_j as $s^i_j \cup u_l$, i.e., $\mathbf{Q^l} = \{s^i_1, \ldots, s^i_{j-1}, s^i_j \cup u_l, s^i_{j+1}, \ldots, s^i_k\}$. Contrasting with Branching in the BST-algorithm t-Overlap, the set I^* is computed differently.

Example 13. *Consider the following instance of the r-Set Packing with t-Membership problem ($r = 3$, $t = 2$, $k = 4$)*

$$\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$$

$$\begin{aligned}\mathcal{S} = \{&S_1 = \{1, 2, 3, 4\}, S_2 = \{1, 2, 5, 6\}, S_3 = \{1, 9, 11, 12\},\\ &S_4 = \{1, 9, 13, 14\}, S_5 = \{1, 15, 16, 17\}, S_6 = \{1, 15, 16, 18\},\\ &S_7 = \{2, 9, 10, 19\}, S_8 = \{2, 9, 7, 10\}\}\end{aligned}$$

A maximal (r,t)-set membership $\mathcal{M} = \{S_1, S_2\}$. Thus,

$$\mathcal{M}_t = \{1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6\}$$

One example of a child of the root is a node i with collection $\mathbf{Q^i} = \{s^i_1 = \{1\}, s^i_2 = \{1\}, s^i_3 = \{2\}, s^i_4 = \{2\}\}$. Let us consider the execution of Greedy *at this node. At $j = 1$,* Greedy *will try to find a set in $\mathcal{S}(s^i_1, \mathbf{Q^i}, t)$ that contains $s^i_1 = \{1\}$. The collection $\mathcal{S}(s^i_1, \mathbf{Q^i}, t)$ is a subcollection of $\mathcal{S}(s^i_1) = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. A set in $\mathcal{S}(s^i_1)$ would be in $\mathcal{S}(s^i_1, \mathbf{Q^i}, t)$, if each element of that set is in at most t sets of $\mathbf{Q^i} \backslash s^i_1$. In this way, $\mathcal{S}(s^i_1, \mathbf{Q^i}, t) = \{S_3, S_4, S_5, S_6\}$. Assume* Greedy *chooses S_3 and add it to $\mathbf{Q^{gr}}$. For the next iteration $j = 2$, assume that* Greedy *chooses S_4 from $\mathcal{S}(s^i_2, \mathbf{Q^i}, t) = \{S_3, S_4, S_5, S_6\}$. Thus, at the end of that iteration $\mathbf{Q^{gr}} = \{S_3, S_4\}$. For iteration $j = 3$,* Greedy *tries to choose a set from $\mathcal{S}(s^i_3, \mathbf{Q^i}, t) = \{S_7, S_8\}$. However, both of these sets contain the element 9 which is contained in $t = 2$ sets of $\mathbf{Q^{gr}}$. Thus,* Greedy *stopped at iteration $j = 3$. After that* Branching *computes the set $I^* = \{9\}$ and creates a children l of the node i with collection $\mathbf{Q^l} = \{\{1\}, \{1\}, \{2, 9\}, \{2\}\}$.*

Correctness

Let us start first providing an analogous lemma to Lemma 54 which shows how a maximal (r,t)-set membership and a (k,r,t)-set membership intersect. This is fundamental to the correctness of the Initialization routine.

Lemma 65. *Let $\mathcal{M}$ and $\mathcal{K}$ be a maximal (r,t)-set membership and a (k,r,t)-set membership, respectively. We claim that each $S^* \in \mathcal{K}$ is either in $\mathcal{M}$ or it shares at least one element with at least t sets of $\mathcal{M}$. Furthermore, each element of $val(\mathcal{M})$ can be in at most t sets of $\mathcal{K}$.*

Proof. Assume that there is a set $S^* \in \mathcal{K}$ that is not in $\mathcal{M}$ and each element $u \in S^*$ is contained in at most $t-1$ sets of $\mathcal{M}$. However, we could add S^* to $\mathcal{M}$ contradicting the assumption that is maximal. Now, assume that there is a set $S^* \in \mathcal{K}$ that is also in $\mathcal{M}$ and S^* shares at least one element with at least t sets of $\mathcal{M}$ (excluding itself). However, in this case $\mathcal{M}$ would not be a $(|\mathcal{M}|,t)$-set membership contradicting our assumption. The second claim follows by definition of a (k,r,t)-set membership. ☐

We will adapt the partial-solution term introduced in Section 8.1.2 for the r-Set Packing with t-Membership problem. A collection $\mathbf{Q^i} = \{s_1^i, \ldots, s_j^i, \ldots, s_k^i\}$ is a *partial-solution* of a (k,r,t)-set membership $\mathcal{K} = \{S_1^*, \ldots, S_k^*\}$ if and only if $s_j^i \subseteq S_j^*$, for $1 \leq j \leq k$.

Lemma 66. *If there exists at least one (k,r,t)-set membership of $\mathcal{S}$, at least one of the children of the root will have a partial-solution.*

Proof. By Lemma 65, every set in $\mathcal{K}$ contains at least one element of $val(\mathcal{M})$. Furthermore, every element of $val(\mathcal{M})$ will be contained in at most t sets of $\mathcal{K}$ ($t \leq k$). The set $\mathcal{M}_t$ contains t copies of each element of $val(\mathcal{M})$. Since we created a node for each selection of k elements from $\mathcal{M}_t$, i.e., $\binom{\mathcal{M}_t}{k}$, and each set of the collection $\mathbf{Q^i}$ is initialized with one of these k elements, the lemma follows. ☐

Recall that BST-algorithm t-Membership stops branching a node with collection $\mathbf{Q^i}$, when Greedy returns $\emptyset$. This routine returns that value when $\mathbf{Q^i}$ is not a partial-solution.

Lemma 67. *The collection $\mathbf{Q^i}$ is not a partial-solution either: i. if there is an element in $val(\mathbf{Q^i})$ that is contained in more than t sets of $\mathbf{Q^i}$, ii. if each set in $\mathcal{S}(s_j^i, \mathbf{Q^i}, t)$, for some s_j^i, is equal to some set in $\mathbf{Q^i} \backslash s_j^i$, or iii. if for some $s_j^i \in \mathbf{Q^i}$, $\mathcal{S}(s_j^i, \mathbf{Q^i}, t) = \emptyset$,*

Proof. (i) Suppose otherwise that $\mathbf{Q^i}$ is a partial-solution, but there is an element $u \in val(\mathcal{M})$ that is contained in at least $t+1$ sets of $\mathbf{Q^i}$. Let $\{s_1^i \ldots s_{t+1}^i\}$ be those $t+1$ sets. Since $\mathbf{Q^i}$ is a partial-solution, this would imply that $s_j^i \subset S_j^*$ for $1 \leq j \leq t+1$. However, this would imply that there are $t+1$ sets in $\mathcal{K}$ that contain the element u a contradiction because $\mathcal{K}$ is a (k,r,t)-set membership.

(ii) This follows because all sets in a (k,r,t)-set membership are distinct.

(iii) To prove the third part of the lemma, we will prove the next stronger claim.

Claim 14. *If $\mathbf{Q^i} = \{s_1^i, \ldots, s_j^i, \ldots, s_k^i\}$ is a partial-solution, then $S_j^* \in \mathcal{S}(s_j^i, \mathbf{Q^i}, t)$ for each $1 \leq j \leq k$.*

Proof. Assume by contradiction that $\mathbf{Q^i}$ is a partial-solution but $S_j^* \notin \mathcal{S}(s_j^i, \mathbf{Q^i}, t)$ for some j. If $\mathbf{Q^i}$ is a partial-solution, then $s_j^i \subseteq S_j^* \in \mathcal{K}$ and $((\mathbf{Q^i} \backslash s_j^i) \cup S_j^*)$ is a partial-solution as well. The set $S_j^* \in \mathcal{S}(s_j^i)$ and $\mathcal{S}(s_j^i, \mathbf{Q^i}, t) \subseteq \mathcal{S}(s_j^i)$. The only way that S_j^* would not be in $\mathcal{S}(s_j^i, \mathbf{Q^i}, t)$ is if there is at least one element u in $val((\mathbf{Q^i} \backslash s_j^i) \cup S_j^*)$ that is contained in more than t sets in $((\mathbf{Q^i} \backslash s_j^i) \cup S_j^*)$ but then $((\mathbf{Q^i} \backslash s_j^i) \cup S_j^*)$ would not be a partial-solution a contradiction to (i). ☐

□

Branching creates at least one child whose collection is a partial-solution, if the collection of the parent is a partial-solution as well.

Lemma 68. *If $\mathbf{Q^i} = \{s_1^i, \dots, s_j^i, \dots, s_k^i\}$ is a partial-solution, there exists at least one $u_l \in I^*$ such that $\mathbf{Q^i} = \{s_1^i, \dots, s_j^i \cup u_l, \dots, s_k^i\}$ is a partial-solution.*

Proof. Assume to the contrary that $\mathbf{Q^i} = \{s_1^i, \dots, s_j^i, \dots, s_k^i\}$ is a partial-solution but that there exists no element $u_l \in I^*$ such that $\mathbf{Q^i} = \{s_1^i, \dots, s_j^i \cup u_l, \dots, s_k^i\}$ is a partial-solution. This can only be possible, if $(S_j^* \backslash s_j^i) \cap I^* = \emptyset$.

Since $\mathbf{Q^i}$ is a partial-solution, $S_j^* \in \mathcal{S}(s_j^i, \mathbf{Q^i}, t)$ by Claim 14. In addition, either S_j^* was equal to a set in $\mathbf{Q^{gr}}$, or there exists at least one element $u \in S_j^*$ that is contained in $t+1$ sets in $\mathbf{Q^{gr}} \cup S_j^* \cup \bar{\mathbf{Q}}^{\mathbf{i}}$ otherwise S_j^* would have been selected by Greedy. The first case directly implies that $(S_j^* \backslash s_j^i) \cap I^* \neq \emptyset$. In the second case we know by the computation of I^* that $u \in I^*$. Thus, it remains to show, that $u \in (S_j^* \backslash s_j^i)$.

To that end, we first show that u must be contained in at least one set in $\mathbf{Q^{gr}}$. Suppose otherwise that u is not in a set in $\mathbf{Q^{gr}}$. This would imply that u is in $t+1$ sets in $S_j^* \cup \bar{\mathbf{Q}}^{\mathbf{i}}$. Recall that $\bar{\mathbf{Q}}^{\mathbf{i}}$ is the subcollection of $\mathbf{Q^i}$ with $\{s_{j+1}^i, \dots, s_k^i\}$. Since $\bar{\mathbf{Q}}^{\mathbf{i}} \subseteq \mathbf{Q^i}$, u would be in at least $t+1$ sets in $((\mathbf{Q^i} \backslash s_j^i) \cup S_j^*)$. However, in that case S_j^* would not have been in $\mathcal{S}(s_j^i, \mathbf{Q^i}, t)$.

Next we prove that $u \notin s_j^i$. This will guarantee that the node i would have at least one children. Suppose that $u \in s_j^i$. We previously showed that there exists at least one set S in $\mathbf{Q^{gr}}$ that contains u. Thus by our assumptions, u is in t sets in $\{s_j^i, s_{j+1}^i, \dots, s_k^i\} \subseteq \mathbf{Q^i}$. Recall that S was selected by Greedy at some iteration $l < j$, and that S contains a subset $s_l^i \in \mathbf{Q^i}$. In this way, S must be in $\mathcal{S}(s_l^i, \mathbf{Q^i}, t)$. By definition of this collection, each element of S (including u) is contained in at most t sets of $(\mathbf{Q^i} \backslash s_l^i) \cup S_l$. However, this would imply that u is contained in at most $t-1$ sets of $\{s_{l+1}^i \dots s_k^i\} \subseteq \mathbf{Q^i}$, a contradiction. □

By the previous lemmas, we can state:

Theorem 43. *The* BST-algorithm t-Membership *finds a (k, r, t)-set membership of $\mathcal{S}$, if $\mathcal{S}$ has at least one.*

Running Time

The number of children of the root is given by $|\binom{\mathcal{M}_t}{k}| \leq \binom{t(r(k-1))}{k} = O((trk)^k)$ and the height of the tree is at most $(r-1)k$. The number of children of each node at level h is given by $|I^*|$. We know from the proof in Lemma 68 that each element in I^* is contained in at least one set of $\mathbf{Q^{gr}}$, and the number of elements in $\mathbf{Q^{gr}}$ is at most $r(k-1)$, $|I^*| \leq (r)(k-1)$. Thus, the size of the tree is given by:

$$\binom{t(r(k-1))}{k} \prod_{h=1}^{(r-1)k} r(k-1) = O(t^k (rk)^{rk})$$

.

Theorem 44. *The r-Set Packing with t-Membership problem can be solved with* BST-algorithm t-Membership *in $O(t^k(rk)^{rk}n^r)$ time.*

We can run this algorithm in the kernel obtained in Section 7.3. The total running time of the algorithm would be $O(2^r t^{k-(r(r-3))}(rk)^{r^2+rk} + n^r)$ which is not better than running a brute-force algorithm on the kernel.

8.2.3 A "Color-Coding" Algorithm

Recall from Section 6.1.1 that we can transform an instance of the r-Set Packing with t-Membership to an instance of the $(r+1)$-Set Packing problem. Similar as in Section 8.1.3, we will apply the FPT-algorithm [53] in such constructed instance.

First, we will transform the instance $(\mathcal{S},\mathcal{U},k)$ of the r-Set Packing with t-Membership to an instance $(\mathcal{S}^T,\mathcal{U}^T,k)$ of r'-Set Packing problem with $r' = r+1$ by running Transformation 5. After that, we reduce $(\mathcal{S}^T,\mathcal{U}^T,k)$ to a kernel $(\mathcal{S}'^T,\mathcal{U}'^T,k)$ with kernelization algorithm [53]. Finally, we obtain a $(k,r',0)$-set packing of $\mathcal{S}'^T$ by running the color-coding algorithm [53]. By Lemma 27, we can transform a $(k,r',0)$-set packing of $\mathcal{S}^T$ into a (k,r,t)-set membership of $\mathcal{S}$ (see Algorithm 12 for a summary of this approach). Algorithm [53] runs in $O(|\mathcal{U}^T| + 2^{O((c+1)(r'k))})$ (where $c \geq 131.76$ [101]) and Transformation 5 returns $|\mathcal{U}^T| = O(n^r)$, we can hence state:

Lemma 69. *The r-Set Packing with t-Membership problem can be solved with Algorithm 12 in $O(n^r + 2^{O((c+1)(r+1)k)})$.*

Algorithm 12 "Color-Coding" - r-Set Packing with t-Membership problem

Input: An instance $(\mathcal{S},\mathcal{U},k)$ of the r-Set Packing with t-Membership problem
Output: A (k,r,t)-set membership $\mathcal{K} \subseteq \mathcal{S}$, if $\mathcal{S}$ has one.

```
1: Transform the instance $(\mathcal{S},\mathcal{U},k)$ of $r$-Set Packing with $t$-Membership to an instance $(\mathcal{S}^T,\mathcal{U}^T,k)$ of $r'$-Set Packing problem, where $r' = r+1$, by running Transformation 5.
2: Reduce the instance $(\mathcal{S}^T,\mathcal{U}^T,k)$ to a kernel $(\mathcal{S}'^T,\mathcal{U}'^T,k)$ by running kernelization algorithm [53].
3: Obtain a $(k,r',0)$-set packing of $\mathcal{S}'^T$ by running the color-coding algorithm [53].
4: Re-interpret this $(k,r',0)$-set packing as a $(k,r,t)$-set membership $\mathcal{K}$ of $\mathcal{S}$ as in Lemma 27.
5: Return $\mathcal{K}$
```

8.2.4 FPT-Algorithms for the Graph Packing Versions

We obtain FPT-algorithms for the $\mathcal{H}$-Packing with t-Membership and t-Edge Membership problems by transforming an instance of these problems to an instance of the r-Set Packing with t-Membership (Transformations 1 and 3, respectively). After that we run any of the FPT-algorithm previously discussed to obtain a (k,r,t)-set membership on the constructed instances. Finally, we re-interpret back that (k,r,t)-set membership as a solution for the original graph problems (Lemmas 11 and 16, respectively). Hence, we can state:

Lemma 70. *The $\mathcal{H}$-Packing with t-Membership problem can be solved in $O(n^r + 2^{O((c+1)(r+1)k)})$, where $r = r(\mathcal{H})$.*

Lemma 71. *The $\mathcal{H}$-Packing with t-Edge Membership problem can be solved in $O(n^r + 2^{O((c+1)(r+1)k)})$, where $r = m(\mathcal{H})$.*

8.3 Conclusions

We developed in this chapter fixed-parameter algorithms for our packing problems with overlap in three ways: a brute-force on a kernel, a bounded search tree, and by using an FPT-algorithm for the r'-Set Packing problem ($r' > r$). Our fastest FPT-algorithm, which has $O(2^{rk})$ running time instead of $O(k^{rk})$, consists of simply running an FPT algorithm (based on color-coding) on the constructed instance of the r'-Set Packing problem ($r' > r$) from Chapter 6. This is a powerful result because we believe that other algorithms can be obtained in a similar fashion.

On the other hand, even though our search tree algorithm asymptotically behaves like a brute-force algorithm on a problem kernel, we will see in the next chapter that this algorithm is very flexible and effective. In particular, it will allow us to show fixed-parameter tractable results for much more general packing problems allowing overlap.

Chapter 9

Arbitrary Overlap Constraints in Packing Problems

In the previous chapters, we studied the $\mathcal{H}$-Packing with t-Overlap and t-Membership problems from a parameterized point of view. The main motivation behind these problems is to capture the fact that communities in real networks may overlap by sharing one or more of their members. In the specific case of the $\mathcal{H}$-Packing with t-Overlap problem, there is even an explicit upper-bound in the overlap between a pair of communities.

Limiting the pairwise overlap is not an unrealistic scenario [62, 111, 31]. Similarly, Gossen et al. [68] and Adamcsek et al. [2] suggest that the overlap should be low (at least in social networks). Nevertheless, in some cases the type of overlap between a pair of communities may be more complex. For example, it has been observed by Yang and Leskovec [138] that overlapping regions are more densely connected than the rest of the community. Gossen et al. [68] suggested that overlapping regions should contain nodes which have strong membership to all the communities they belong to. Also, Youssef et al. [139] consider only *boundaries nodes* in the overlapping regions.

The study of the properties of the overlapping vertices or regions is not as developed as the properties of the communities themselves. However, it is not unlikely to consider that overlapping nodes could present common properties [112]. It would be more logical to expect that nodes that are "closer" to each other belong to the same pair of communities than nodes that are far away within the same community, for example. Motivated by this, we generalize the $\mathcal{H}$-Packing with t-Overlap to restrict the pairwise overlap by a function $\alpha()$ rather than by an upper-bound t. More precisely, we will say that for any pair of $\mathcal{H}$-subgraphs H_i, H_j in the solution, $\alpha(H_i, H_j) = 0$. In this way, we could fix α to return zero (or *no-conflict*) when H_i and H_j satisfy an overlap restriction. For example, return zero when the distance between every pair of vertices in $H_i \cap H_j$ is at most d_t (for some constant d_t).

In our problem generalization, we also consider other community models (represented through a graph property Π) besides a family of graphs $\mathcal{H}$. The scope of community definitions is vast, see [61, 137]. Thus in this chapter, we introduce the much more general problem Π-*Packing with* $\alpha()$-*Overlap*. We also propose a similar generalization for the problem of packing sets with pairwise overlap that we call the r-*Set Packing with* $\alpha()$-*Overlap problem*. In addition, we show that these problems are fixed-parameter tractable when the constraint function $\alpha()$ meets some natural conditions (a *well-conditioned* $\alpha()$) and when the parameter is the size of the solution.

This chapter is organized as follows. In Section 9.1, we provide an $O(r^{rk}\, k^{(r+1)\,k}\, n^r)$ algorithm for the r-Set Packing with $\alpha()$-Overlap problem, when $\alpha()$ meets specific requirements. This algorithm generalizes our FPT-algorithm based on bounded search trees and greedy localization techniques from Chapter 8.

In Section 9.2, we first define the Π-Packing with $\alpha()$-Overlap problem and then we transform this problem to its set version. This allows us to achieve an algorithm with $O(r^{rk}\, k^{(r+1)\,k}\, n^r)$ running time, provided that $\alpha()$ is well-conditioned.

In Section 9.3, we give specific examples of well-conditioned $\alpha()$ functions, some motivated by practical applications while others by theoretical considerations. Specifically, a well-conditioned $\alpha()$ can restrict: i) the size of the overlap, ii) the weight in the overlap region, (assuming as input a weighted graph), iii) the maximum distance between any pair of vertices in the overlap, iv) *the pattern* in the overlap region, i.e. the induced subgraph in the overlap should be isomorphic to a graph in $\mathcal{F}$, where $\mathcal{F}$ is a graph class that is hereditary, v) that all overlapping vertices must satisfy a specific property ξ, and vi) finally, that the overlap region should have a specific density.

Motivated by clustering applications, we study the r-Set Packing with $\alpha()$-Overlap (PCH) problem in Section 9.4. In this setting, every set in the solution must contain a specific set of elements from a given collection of sets $\mathcal{C}$. This problem remains fixed-parameter tractable if $|\mathcal{C}| = O(g(k))$ for some computable function g dependent on k and independent of n.

9.1 Packing Sets with Overlap Constraints

In this section, we generalize the BST-ALGORITHM t-OVERLAP from the previous chapter to allow arbitrary overlap constraints between a pair of sets. Let us begin with the formal definition of our newly introduced problem. Once more, r is a fixed-constant.

The r-Set Packing with $\alpha()$-Overlap problem
Input: A collection of sets $\mathcal{S}$ each one of size at most r, drawn from a universe $\mathcal{U}$, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a $(k, \alpha())$-*set packing*, i.e., at least k distinct sets $\mathcal{K} = \{S_1^*, \ldots, S_k^*\}$ where for each pair S_i^*, S_j^* $(i \neq j)$ $\alpha(S_i^*, S_j^*) = 0$, where $\alpha : \mathcal{S} \times \mathcal{S} \rightarrow \{0, 1\}$?

The FPT-algorithm provided in this section assumes that the function $\alpha()$ meets specific requirements that we call it a *well-conditioned* $\alpha()$.

Definition 9. *Let $\mathcal{U}^r$ be all possible subsets of elements of $\mathcal{U}$ each of size at most r, and $\alpha : \mathcal{U}^r \times \mathcal{U}^r \rightarrow \{0,1\}$ be a function. A pair of sets $s_i, s_j \in \mathcal{U}^r$* α-conflict *if $\alpha(s_i, s_j) = 1$ else they* do not α-conflict. *If $\alpha()$ satisfies the following requirements, we say $\alpha()$ is* well-conditioned.

i) $\alpha()$ is hereditary. *Specifically, if s_i and s_j do not α-conflict ($\alpha(s_i, s_j) = 0$), then $\alpha(s'_i, s'_j) = 0$ for any pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$.*

ii) If s_i and s_j α-conflict ($\alpha(s_i, s_j) = 1$), then $|s_i \cap s_j| \geq 1$. Furthermore, for any pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with $\alpha(s'_i, s'_j) = 0$ we have $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$. The elements in $s_i \cap s_j$ are referred to as the conflicting elements*.*

iii) α is computable in polynomial time in n.

We can relax the condition (iii) in the well-conditioned $\alpha()$ when the properties that $\alpha()$ will evaluate are given as part of the input. Given that each set in $\mathcal{S}$ has at most r elements (and r is considered a fixed-constant), evaluating $\alpha(S_i, S_j)$ will take constant time. Therefore, in the specific case that $\alpha()$ only evaluates the properties in S_i andS_j, we can omit condition (iii) or only asks to be computed in $O(r)$ time.

We introduced in Chapter 8 a search tree algorithm for the r-Set Packing with t-Overlap. The FPT-algorithm of this section basically generalizes that algorithm. Before, we used to look to a specific type a conflict between a pair of sets: overlap larger than t. Here, we will consider the more general α-conflicts.

As with the BST-ALGORITHM t-OVERLAP, the FPT-algorithm (called BST $\alpha()$-ALGORITHM) for r-Set Packing with $\alpha()$-Overlap has three main components: INITIALIZATION, GREEDY, and BRANCHING. The main differences with the BST-ALGORITHM t-OVERLAP are the INITIALIZATION and GREEDY routines, a different analysis of correctness and running time.

Once more, we start by computing a maximal collection of sets $\mathcal{M}$ from $\mathcal{S}$ such that for each $S \notin \mathcal{M}$, S α-conflicts with some $S' \in \mathcal{M}$, i.e., *a maximal* $\alpha()$*-set packing* $\mathcal{M}$ of $\mathcal{S}$. If $|\mathcal{M}| \geq k$, then $\mathcal{M}$ is a $(k, \alpha())$-set packing, and the BST $\alpha()$-ALGORITHM stops. Otherwise, we create a search tree T where at each node i, there is a collection of sets $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_k\}$ with $s^i_j \subseteq S$ for some $S \in \mathcal{S}$. The children of the root of T are created according to a procedure called INITIALIZATION.

After that for each node i of T, a routine called GREEDY will attempt to find a $(k, \alpha())$-set packing using $\mathbf{Q^i}$. If GREEDY succeeds, then the BST $\alpha()$-ALGORITHM stops. Otherwise, the next step is to create children of the node i using the procedure BRANCHING. Each child l of the node i will have a collection $\mathbf{Q^l}$ that is the same as $\mathbf{Q^i}$ with the difference that one of the sets of $\mathbf{Q^l}$ is increased by one element. The BST $\alpha()$-ALGORITHM will repeat GREEDY in these children. Eventually, the BST $\alpha()$-ALGORITHM either finds a solution at one of the leaves of the tree or determines that it is not possible to find one. The pseudocode of this algorithm is detailed in Algorithm 13.

We next explain the three main components of the BST $\alpha()$-ALGORITHM individually. Let us start with the INITIALIZATION routine. By Lemma 72, if there is a solution $\mathcal{K} = \{S^*_1, \ldots, S^*_k\}$, each S^*_j contains at least one element of $val(\mathcal{M})$. Notice that each element of $val(\mathcal{M})$ could be in at most k sets of $\mathcal{K}$. Thus, we create a set $\mathcal{M}_k$ that contains k copies of each element in $val(\mathcal{M})$. That is, per each element $u \in val(\mathcal{M})$ there are k copies $u_1 \ldots u_k$ in $\mathcal{M}_k$ and $|\mathcal{M}_k| = k|val(\mathcal{M})|$. The root will have a child i for each possible combination of k elements from $\mathcal{M}_k$. A set of $\mathbf{Q^i}$ is initialized with one element of that combination. For example, if the combination is $\{u_1, u_2, u_k, a_1, b_1\}$, $\mathbf{Q^i} = \{\{u_1\}, \{u_2\}, \{u_k\}, \{a_1\}, \{b_1\}\}$. After that, we remove the indices from the elements in $\mathbf{Q^i}$, e.g., $\mathbf{Q^i} = \{\{u\}, \{u\}, \{u\}, \{a\}, \{b\}\}$. See Algorithm 14 for details.

At each node i, the GREEDY routine (Algorithm 15) returns a collection of sets $\mathbf{Q^{gr}}$. Initially, $\mathbf{Q^{gr}} = \emptyset$ and $j = 1$. At iteration j, GREEDY searches for a set S that contains s^i_j (recall that s^i_j is the jth set of $\mathbf{Q^i}$) subject to two conditions: (1) S is not already in $\mathbf{Q^{gr}}$ and (2) S does not α-conflict with any set in $\mathbf{Q^{gr}}$. If such set S exists, GREEDY adds S to $\mathbf{Q^{gr}}$, i.e., $\mathbf{Q^{gr}} = \mathbf{Q^{gr}} \cup S$ and continues with iteration $j = j + 1$. Otherwise, GREEDY stops executing and returns $\mathbf{Q^{gr}}$. If $|\mathbf{Q^{gr}}| = k$, then $\mathbf{Q^{gr}}$ is a $(k, \alpha())$-set packing and the BST $\alpha()$-ALGORITHM stops. If $\mathbf{Q^i}$ cannot be propagated into a solution (Lemma 74), GREEDY returns $\mathbf{Q^{gr}} = \infty$. GREEDY searches for the set S in the collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) \subseteq \mathcal{S}(s^i_j)$ which is obtained as follows: add a set $S' \in \mathcal{S}(s^i_j)$ to $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$, if S' does not α-conflict with any set in $(\mathbf{Q^i} \backslash s^i_j)$ and S' is distinct of each set in $(\mathbf{Q^i} \backslash s^i_j)$ (see Algorithm 17). Observe that this last condition can only be true for sets in $\mathbf{Q^i}$ that are already sets of $\mathcal{S}$.

The BRANCHING procedure (Algorithm 16) executes every time that GREEDY does not return a $(k, \alpha())$-set packing but $\mathbf{Q^i}$ could be propagated into one. That is, $\mathbf{Q^{gr}} \neq \infty$ and $|\mathbf{Q^{gr}}| < k$. Let $j = |\mathbf{Q^{gr}}| + 1$ and s^i_j be the j-*th* set in $\mathbf{Q^i}$. GREEDY stopped at j because each set $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ either it was already contained in $\mathbf{Q^{gr}}$, or it α-conflicts with at least one set in $\mathbf{Q^{gr}}$ (see **). We will use the conflicting elements between $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ and $\mathbf{Q^{gr}}$ to create children of the node i. Let I^* be the set of those conflicting elements. BRANCHING creates a child l of the node i for each element $u_l \in I^*$. The collection $\mathbf{Q^l}$ of child l is the same as the collection $\mathbf{Q^i}$ of its parent i with the update of the set s^i_j as $s^i_j \cup u_l$, i.e., $\mathbf{Q^l} = \{s^i_1, \ldots, s^i_{j-1}, s^i_j \cup u_l, s^i_{j+1}, \ldots, s^i_k\}$. (Line 12, Algorithm 16). The set I^* is obtained as $I^* = I^* \cup ((S \backslash s^i_j) \cap S')$ for each pair $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ and $S' \in \mathbf{Q^{gr}}$ that α-conflict or that $S = S'$ (see Lines 4-8 of Algorithm 16).

Correctness.

Let us re-define the partial-solution term for the r-Set Packing with $\alpha()$-Overlap problem. A collection $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a *partial-solution* of a $(k, \alpha())$-set packing $\mathcal{K} = \{S^*_1, \ldots, S^*_j, \ldots, S^*_k\}$ if and only if $s^i_j \subseteq S^*_j$, for $1 \leq j \leq k$.

We will show the correctness of our algorithm in two steps: first, we show that at least one child of the root will have a partial solution, and second, if a node is a partial-solution, at least one of its children will be a partial-solution as well. In this way, a $(k, r, \alpha())$-set packing will be at one of the leaves of the tree. We use extensively properties (i) and (ii) of the well-conditioned $\alpha()$ for show such correctness. The third property of a well-conditioned $\alpha()$ is used for running time computation.

The INITIALIZATION procedure uses a maximal $\alpha()$-set packing to create the children of the root. The correctness of that procedure basically follows because as we will see in the next lemma every set of a $(k, \alpha())$-set packing shares at least one element with at least one set of a maximal $\alpha()$-set packing. This intersection is due to the (ii) property of a well-conditioned $\alpha()$.

Lemma 72. *Given an instance $(\mathcal{U}, \mathcal{S}, k)$ of r-Set Packing with $\alpha()$-Overlap, where $\alpha()$ is well-conditioned, let $\mathcal{K}$ and $\mathcal{M}$ be a $(k, \alpha())$-set packing and a maximal $\alpha()$-set packing, respectively. For each $S^* \in \mathcal{K}$, S^* shares at least one element with at least one set $S \in \mathcal{M}$.*

Proof. If $S^* \in \mathcal{M}$, the lemma simply follows. Assume by contradiction that there is a set $S^* \in \mathcal{K}$ such that $S^* \notin \mathcal{M}$ and there is no set $S \in \mathcal{M}$ α-conflicting with S^*. However, we could add S^* to $\mathcal{M}$, contradicting its maximality. Thus, there exists at least one $S \in \mathcal{M}$ α-conflicting with S^*. Since $\alpha()$ is well-conditioned, by Definition 9 (ii) $|S \cap S^*| \geq 1$. □

Lemma 73. *If there exists at least one $(k, \alpha())$-set packing of $\mathcal{S}$, at least one of the children of the root will have a partial-solution.*

Proof. By Lemma 72, every set in $\mathcal{K}$ contains at least one element of $val(\mathcal{M})$. It is possible that the same element be in at most k different sets of $\mathcal{K}$. Therefore, we replicated k times each element in $val(\mathcal{M})$ collected in $\mathcal{M}_k$. Since we created a node for each selection of k elements from $\mathcal{M}_k$, i.e., $\binom{\mathcal{M}_k}{k}$, the lemma follows. $\square$

The next lemma states that the BST $\alpha()$-Algorithm correctly stops attempting to propagate a collection $\mathbf{Q^i}$. Due to the hereditary property of the well-conditioned $\alpha()$, we can immediately discard a collection $\mathbf{Q^i}$, if it has a pair of sets that α-conflicts. In addition, the collection $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ contains all sets from $\mathcal{S}(s^i_j)$ that are not α-conflicting with any set in $\mathbf{Q^i}$ (excluding s^i_j). So again, if $\mathbf{Q^i}$ is a partial-solution, due to the hereditary property of $\alpha()$, $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ cannot be empty.

Lemma 74. *Assuming $\alpha()$ is well-conditioned, $\mathbf{Q^i}$ is not a partial solution either: i. if there is a pair of sets in $s^i_j, s^i_l \in \mathbf{Q^i}$ ($j \neq l$) that α-conflict, or ii. if for some $s^i_j \in \mathbf{Q^i}$, $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) = \emptyset$.*

Proof. (i) Suppose otherwise that $\mathbf{Q^i}$ is a partial-solution, but s^i_j, s^i_l α-conflict. Since $\mathbf{Q^i}$ is a partial-solution, $s^i_j \subseteq S^*_j$ and $s^i_l \subseteq S^*_l$ where S^*_j,$S^*_l \in \mathcal{K}$ and $\mathcal{K}$ is a $(k, \alpha())$-set packing.

The pair S^*_j,S^*_l does not α-conflict, otherwise, $\mathcal{K}$ would not be a solution. However, $\alpha()$ is hereditary, $s^i_j \subseteq S^*_j$, and $s^i_l \subseteq S^*_l$, thus, s^i_j and s^i_l do not α-conflict either.

(ii) To prove the second part of the lemma, we will prove the next stronger claim.

Claim 15. *If $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a partial-solution, $S^*_j \in \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ for each $1 \leq j \leq k$.*

Proof. Assume by contradiction that $\mathbf{Q^i}$ is a partial-solution but $S^*_j \notin \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ for some j.

If $\mathbf{Q^i}$ is a partial-solution, $s^i_j \subseteq S^*_j \in \mathcal{K}$ and $((\mathbf{Q^i} \backslash s^i_j) \cup S^*_j)$ is a partial-solution as well. The set $S^*_j \in \mathcal{S}(s^i_j)$ and $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) \subseteq \mathcal{S}(s^i_j)$ (see Algorithm 17 for the computation of $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$). The only way that S^*_j would not be in $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ is if there is at least one set S in $(\mathbf{Q^i} \backslash s^i_j)$ that α-conflicts with S^*_j or if S^*_j is equal to a set in $(\mathbf{Q^i} \backslash s^i_j)$ but then $((\mathbf{Q^i} \backslash s^i_j) \cup S^*_j)$ would not be a partial-solution a contradiction to (i). $\square$

$\square$

Branching creates at least one child whose collection is a partial-solution, if the collection of the parent is a partial-solution as well. Recall that I^* is computed when Greedy stopped its execution at some $j \leq k$, i.e., it could not add a set that contains s^i_j to $\mathbf{Q^{gr}}$. If $\mathbf{Q^i}$ is a partial solution, $s^i_j \subseteq S^*_j$ and $S^*_j \in \mathcal{K}$. Given property (ii) for a well-conditioned $\alpha()$, S^*_j must be intersecting in at least one element with at least on set in $\mathbf{Q^{gr}}$. Therefore, at least one element of S^*_j will be in I^*.

Lemma 75. *If $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j, \ldots, s^i_k\}$ is a partial-solution, there exists at least one $u_l \in I^*$ such that $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_j \cup u_l, \ldots, s^i_k\}$ is a partial-solution.*

Proof. Assume to the contrary that $\mathbf{Q^i} = \{s_1^i, \ldots, s_j^i, \ldots, s_k^i\}$ is a partial-solution but that there exists no element $u_l \in I^*$ such that $\mathbf{Q^i} = \{s_1^i, \ldots, s_j^i \cup u_l, \ldots, s_k^i\}$ is a partial-solution. This can only be possible if $(S_j^* \backslash s_j^i) \cap I^* = \emptyset$.

First, given that $\mathbf{Q^i}$ is a partial-solution $S_j^* \in \mathcal{S}(s_j^i, \mathbf{Q^i}, \alpha)$ (Claim 15).

In addition, either S_j^* is already in $\mathbf{Q^{gr}}$ (i.e, it is equal to some set $S' \in \mathbf{Q^{gr}}$) or S_j^* must be α-conflicting with at least one set $S' \in \mathbf{Q^{gr}}$; otherwise, S_j^* would have been selected by Greedy. Any of these situations implies that $|S_j^* \cap S'| \geq 1$ (Definition 9 (ii)). By the computation of I^* (Algorithm 16), $S_j^* \cap S' \subseteq I^*$.

Now it remains to show, that at least one element of $S_j^* \cap S'$ is in $S_j^* \backslash s_j^i$. This will guarantee that the set s_j^i will be increased by one element at the next level of the tree. This immediately follows if $S_j^* = S'$.

Thus, we will show it for the case that $S_j^* \neq S'$ but S_j^* α-conflicts with S'. Suppose that $(S_j^* \cap S') \cap (S_j^* \backslash s_j^i) = \emptyset$ by contradiction. Recall that S' contains some set s_h^i of $\mathbf{Q^i}$ (for some $h \leq j$). Furthermore, $S' \in \mathcal{S}(s_h^i, \mathbf{Q^i}, \alpha)$; otherwise S' would not have been selected by Greedy.

If S' is α-conflicting with S_j^* but S' is not α-conflicting with s_j^i (otherwise S' would not have been in $\mathcal{S}(s_h^i, \mathbf{Q^i}, \alpha)$), then $(S' \cap (S_j^* \backslash s_j^i)) \neq \emptyset$ by property (ii) in Definition 9. □

By the previous lemmas, we can state:

Theorem 45. *The* BST $\alpha()$-Algorithm *finds a* $(k, \alpha())$*-set packing of* $\mathcal{S}$*, if* $\mathcal{S}$ *has at least one and* $\alpha()$ *is well-conditioned.*

Running Time.

The number of children of the root is given by $|\binom{\mathcal{M}_k}{k}| \leq \binom{k(r(k-1))}{k} = O((rk^2)^k)$ and the height of the tree is at most $(r-1)\,k$. The number of children of each node at level h is equivalent to the size of I^* at each level h. We know from Lemma 75 that $|I^*| \leq |val(\mathbf{Q^{gr}})|$. The number of elements in $val(\mathbf{Q^{gr}})$ is at most $r(k-1)$, thus, $|I^*| \leq r(k-1)$. Therefore, the size of the tree is given by $\binom{k\,(r(k-1))}{k} \prod_{h=1}^{(r-1)k} r\,(k-1)$ which is $O(r^{rk}\, k^{(r+1)\,k})$. In addition, $\alpha()$ is computable in polynomial time in n (Definition 9, (iii)).

Theorem 46. *The r-Set Packing with* $\alpha()$*-Overlap problem can be solved in* $O(r^{rk}\, k^{(r+1)\,k}\, n^r)$ *time, when* $\alpha()$ *is well-conditioned.*

Pseudocode.

Algorithm 13 BST $\alpha()$-Algorithm

```
1: Compute a maximal $\alpha()$-set packing $\mathcal{M}$
2: if $|\mathcal{M}| \geq k$ then Return $\mathcal{M}$ end if
3: $T$=Initialization($\mathcal{M}$)
4: for each node $i$ of $T$ do
5:     Let $\mathbf{Q^i}$ be the collection of sets at node $i$
6:     $\mathbf{Q^{gr}}$ =Greedy($\mathbf{Q^i}$)
7:     if $\mathbf{Q^{gr}}! = \infty$ then
8:         if $|\mathbf{Q^{gr}}| = k$ then Return $\mathbf{Q^{gr}}$ end if
9:         Branching($T$,node $i$,$\mathbf{Q^i}$,$\mathbf{Q^{gr}}$)
10:    end if
11: end for
```

Algorithm 14 Initialization($\mathcal{M}$)

```
1: Replicate $k$ times each element $u \in val(\mathcal{M})$ and identify them as $u_1, \ldots, u_k$.
2: Let $\mathcal{M}_k$ be the enlarged set $val(\mathcal{M})$
3: $i = 0$, $T = null$
4: while $i < |\binom{\mathcal{M}_k}{k}|$ do
5:     Let $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_k\}$ be the $i$th combination of $\binom{\mathcal{M}_k}{k}$
6:     CreateNode($T$,root,node $i$,$\mathbf{Q^i}$)
7:     $i = i + 1$
8: end while
9: Return $T$
```

Algorithm 16 Branching(T,node i,$\mathbf{Q^i}$,$\mathbf{Q^{gr}}$)

```
1: Let $s^i_j$ be the first set of $\mathbf{Q^i}$ not completed by Greedy, i.e.,
2: $j = |\mathbf{Q^{gr}}| + 1$ and $s^i_j = \mathbf{Q^i}[j]$
3: $I^* = \emptyset$
4: for each $S \in \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ do
5:     for each $S' \in \mathbf{Q^{gr}}$ do
6:         if $\alpha(S, S') == 1$ OR $(S == S')$ then $I^* = I^* \cup ((S \backslash s^i_j) \cap S')$ end if
7:     end for
8: end for
9: $l = 0$
10: while $l \leq |I^*|$ do
11:     Let $u_l$ be the $l$th element of $I^*$
12:     $\mathbf{Q^l} = \{s^i_1, s^i_2, \ldots, s^i_j \cup u_l, \ldots, s^i_k\}$
13:     CreateNode($T$,node $i$,node $l$,$\mathbf{Q^l}$)
14:     $l = l + 1$
15: end while
```

Algorithm 15 GREEDY($\mathbf{Q^i}$)

```
[H]
 1: $\mathbf{Q^{gr}} = \infty$
 2: //Check if $\mathbf{Q^i}$ could not be a partial solution
 3: if there is no pair $s^i_f$,$s^i_g$ in $\mathbf{Q^i}$ $(f \neq g)$ with $\alpha(s^i_f, s^i_g) = 1$ then
 4:    $\mathbf{Q^{gr}} = \emptyset$; $j = 0$
 5:    repeat
 6:       Let $s^i_j$ be the $j$th set of $\mathbf{Q^i}$
 7:       if $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) = \emptyset$ then
 8:          $\mathbf{Q^{gr}} = \infty$
 9:       else
10:          //Choose arbitrarily a set $S$ from $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$ such that
11:          //$S$ does not $\alpha$-conflict with any set in $\mathbf{Q^{gr}}$
12:          //and $S$ is not already in $\mathbf{Q^{gr}}$
13:          $f = 0$
14:          while $f < |\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)|$ do
15:             Let $S_f$ be the $f$-th set in $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$
16:             $Conflicts = 0$
17:             for each $S' \in \mathbf{Q^{gr}}$ do
18:                if $(\alpha(S_f, S') == 1)$ OR $(S_f == S')$ then
19:                   $Conflicts = Conflicts + 1$
20:                end if
21:             end for
22:             if $Conflicts == 0$ then $S = S_f$; $f = |\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)| + 1$ end if
23:          end while
24:          //Add the set $S$ to $\mathbf{Q^{gr}}$
25:          if such set $S$ does not exist then
26:             $j = k + 1$
27:          else
28:             $\mathbf{Q^{gr}} = \mathbf{Q^{gr}} \cup S$
29:          end if
30:          $j = j + 1$
31:       end if
32:    until $(j \geq k)$ OR $(\mathbf{Q^{gr}} = \infty)$
33: end if
34: Return $\mathbf{Q^{gr}}$
```

Algorithm 17 COMPUTE $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$

```
1: $l = 0$, $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) = \emptyset$
2: while $l < |\mathcal{S}(s^i_j)|$ do
3:   Let $S_l$ be the $l$-th set in $\mathcal{S}(s^i_j)$
4:   $f = 0$, $conflicts = 0$
5:   while $f < |\mathbf{Q^i}|$ do
6:     if $f \neq j$ then
7:       if $\alpha(s^i_f, S_l) == 1$ OR $(s^i_f == S_l)$ then
8:         $conflicts = conflicts + 1$
9:       end if
10:     end if
11:     $f = f + 1$
12:   end while
13:   if $conflicts == 0$ then $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) = \mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha) \cup S_l$ end if
14:   $l = l + 1$
15: end while
16: Return $\mathcal{S}(s^i_j, \mathbf{Q^i}, \alpha)$
```

9.2 Packing Graphs with Overlap Constraints

The Π-Packing with $\alpha()$-Overlap problem generalizes the $\mathcal{H}$-Packing with t-Overlap problem by including other community definitions in addition to prescribed graphs and by allowing more complex overlap restrictions. Let us start with the formal definition of this problem. Once more, r is a fixed-constant.

The Π-Packing with $\alpha()$-Overlap problem
Input: A graph G and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, α)-Π-packing, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1^*, \dots, H_k^*\}$, subject to the following conditions: i) each H_i^* has at most r vertices and obeys the property Π, and ii) for any pair H_i, H_j, with $i \neq j$, $\alpha(H_i^*, H_j^*) = 0$ and $V(H_i) \neq V(H_j)$?

In this problem, we represent a community through a graph property Π. Intuitively, if a subgraph H of order at most r has the property Π (called a Π-*subgraph*), H is a community. Observe that we are explicitly bounding the size of a community by a constant r. To obtain an FPT algorithm, we require however that Π be verifiable in polynomial time in n, where $n = |V(G)|$.

Examples of properties Π that could represent communities are the following. Let S be an induced subgraph of G with at most r vertices. S is a community if it has a density of at least t ($|E(S)| \geq t$) and the number of edges connecting S to rest of the network is at most a specific value [111]. S is a community if every vertex in S is adjacent to at least $|V(S)| - c$ vertices in S (for some constant c). Observe that with our property Π, we still can use a family of graphs $\mathcal{H}$ to represent a community. In that case, Π would correspond to the condition that S is a community if S is isomorphic to a graph H in $\mathcal{H}$.

In the Π-Packing with $\alpha()$-Overlap problem, we regulate the pairwise overlap with a function $\alpha()$. More precisely, we say that two induced Π-subgraphs H_i and H_j α-*conflict* if $\alpha(H_i, H_j) = 1$; otherwise they do not α-conflict. To provide a solution for the Π-Packing with $\alpha()$-Overlap problem, we will basically follow the approach of reducing this problem to the set version, i.e., to the r-Set Packing with $\alpha()$-Overlap problem.

To this end, we first compute the collection of all induced Π-subgraphs of G, and we collect them in Π_G. This is done by naively testing all sets of at most r vertices from G. We highlight that we are not asking to compute the largest subgraph of G that follows Π, but rather only verifying whether a specific induced subgraph of at most r vertices satisfies Π or not. In this way, $|\Pi_G| = O(n^r)$.

Next, we construct an instance of r-Set Packing with $\alpha()$-Overlap as follows.

Transformation 15. ***Input:*** *G* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$*

The universe $\mathcal{U}$ equals to $V(G)$.

There is a set S in $\mathcal{S}$ for each induced Π-subgraph H in Π_G and $S = V(H)$.

Furthermore, $\alpha(S_i, S_j) = \alpha(H_i, H_j)$.

In this way, the size of $\mathcal{U}$ is $O(n)$ while the size of $\mathcal{S}$ is $|\Pi_G| = O(n^r)$. Each set in $\mathcal{S}$ has at most r elements.

Lemma 76. *The collection $\mathcal{S}$ has a $(k, \alpha())$-set packing if and only if G has a $(k, \alpha())$-Π-packing.*

Proof. We build a $(k, \alpha())$-set packing $\mathcal{K}_S$ from a $(k, \alpha())$-Π-packing. For each Π-subgraph H_i in $\mathcal{K}$, we add a set $S_i = V(H_i)$ to $\mathcal{K}_S$. By construction, $S_i \in \mathcal{S}$. Every pair of sets S_i, S_j $\mathcal{K}_S$ does not α-conflict. This follows because every pair $H_i, H_j \in \mathcal{K}_S$ does not α-conflict, i.e., and $\alpha(H_i, H_j) = 0$.

Given a $(k, \alpha())$-set packing $\mathcal{K}_S$, we build a $(k, \alpha())$-Π-packing $\mathcal{K}$ of G. For each set S_i in $\mathcal{K}_S$, we add a Π-subgraph $H_i = (S_i, E(G[S_i]))$. By construction, H_i is an induced Π-subgraph of G. Any pair of Π-subgraphs in $\mathcal{K}$ does not α-conflict; otherwise, there would be a pair of sets in $\mathcal{K}_S$ α-conflicting. $\square$

We assume once more that the function $\alpha()$ is well-conditioned. After that, we run Algorithm 13 of Section 9.1 on the constructed instance. By Lemma 76, we can transform back the set packing instance to the original graph problem.

Theorem 47. *The Π-Packing with $\alpha()$-Overlap problem can be solved in $O(r^{rk}\, k^{(r+1)k}\, n^r)$ time, when $\alpha()$ is well-conditioned and Π is polynomial time verifiable.*

9.3 Well-Conditioned Overlap Constraints

We provide next several examples of well-conditioned $\alpha()$ functions. In the first section, we focus on functions concerning the r-Set Packing with $\alpha()$-Overlap problem that by Transformation 15 could be used to restrict the overlap for the graph version as well. After that in Section 9.3.2, we provide functions that consider properties exclusively from graphs.

9.3.1 Restricting the Overlap Between Sets

Constant Overlap. Upper-bounding the overlap size by a constant t (as in the r-Set Packing with t-Overlap problem) where $0 \leq t \leq r-1$ is a well-conditioned $\alpha()$ function. More precisely, let α-Constant(s_i, s_j) be a function that returns 0 (*no-conflict*) if $|s_i \cap s_j| \leq t$, otherwise returns 1 (α-*conflict*).

```
Function $\alpha$-Constant$(s_i, s_j)$
Result =no-conflict
if $|s_i \cap s_j| > t$ then Result =$\alpha$-conflict end if
Return Result
EndFunction
```

Lemma 77. *The function α-Constant is well-conditioned.*

Proof. (i) α-Constant is *hereditary.* For any pair of sets s_i,s_j with $\alpha(s_i, s_j) = 0$, $|s_i \cap s_j| \leq t$. Thus, there cannot be a pair of subsets $s'_i \subseteq s_i$, $s'_j \subseteq s_j$ with $|s'_i \cap s'_j| > t$.

(ii) Since $t \geq 0$, for any pair s_i, s_j with $\alpha(s_i, s_j) = 1$, $|s_i \cap s_j| \geq t + 1 \geq 1$. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be a pair of subsets with $\alpha(s'_i, s'_j) = 0$. Note that at most one $s'_i = s_j$ or $s'_j = s_j$; otherwise $\alpha(s'_i, s'_j) = 1$. Since $\alpha(s'_i, s'_j) = 0$, $|s'_i \cap s'_j| \leq t$. Therefore, $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) Finally, we can decide in constant time whether $|s_i \cap s_j| > t$. $\square$

Weighted Overlap. Let us assume that each $u \in \mathcal{U}$ has associated a non-negative weight $w(u)$. We could restrict the overlap region by its weight. The function α-Weight(s_i, s_j) returns *no-conflict* if $w(s_i \cap s_j) = (\sum_{u \in (s_i \cap s_j)} w(u)) \leq w_t$ where $w_t \geq 0$ is a constant, else returns *α-conflict*.

```
Function $\alpha$-Weight$(s_i, s_j)$
Result =no-conflict
if  $w(s_i \cap s_j) = (\sum_{u \in s_i \cap s_j} w(u)) > w_t$ then Result =$\alpha$-conflict end if
Return Result
EndFunction
```

Lemma 78. *The function α-Weight is well-conditioned.*

Proof. (i) α-Weight is *hereditary*. For any pair of sets s_i,s_j with $w(s_i \cap s_j) \leq w_t$ ($\alpha(s_i, s_j) = 0$), there is no pair of subsets $s'_i \subseteq s_i, s'_j \subseteq s_j$ with $w(s'_i \cap s'_j) > w_t$ ($\alpha(s_i, s_j) = 1$). For the sake of contradiction, suppose otherwise. Notice that $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$. Thus, if $w(s_i \cap s_j) \leq w_t$ but $w(s'_i \cap s'_j) > w_t$, then there must be some elements in $(s_i \cap s_j) \backslash (s'_i \cap s'_j)$ with negative weights, a contradiction because the weights in $\mathcal{U}$ are non-negative.

(ii) If $\alpha(s_i, s_j) = 1$, then $w(s_i \cap s_j) > w_t$ and $(s_i \cap s_j) \neq \emptyset$. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be any pair of subsets with $\alpha(s'_i, s'_j) = 0$, (i.e., $w(s'_i \cap s'_j) \leq w_t$). Note that at most one $s'_i = s_j$ or $s'_j = s_j$; otherwise $\alpha(s'_i, s'_j) = 1$. Since $w(s_i \cap s_j) > w_t$, $w(s_i \cap s_j) - w(s'_i \cap s'_j) > 0$. Therefore, $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) Finally, we can determine in $O(r)$ time, if $w(s_i \cap s_j) > w_t$. □

Notice that we could use α-Weight(s_i, s_j) to upper-bound the overlap size by a constant t. To this end, we make $w(u) = 1$ for each $u \in \mathcal{U}$, and $w_t = t$. However, we have defined these functions separately to combine them into a well-conditioned $\alpha()$ function that restricts the overlap by size and weight.

```
Function $\alpha$-ConstantAndWeight$(s_i, s_j)$
Result =no-conflict
if $|s_i \cap s_j| > t$ OR $w(s_i \cap s_j) > w_t$ then Result =$\alpha$-conflict end if
Return Result
EndFunction
```

Lemma 79. *The function α-ConstantAndWeight is well-conditioned.*

Measures Overlap. A *measure* of a set S is a function μ that satisfies the following conditions: i. (non-negative) $\mu(S) \geq 0$, ii. (zero for the empty set) $\mu(S) = 0$ if $S = \emptyset$, and iii. (additive) $\mu(S) = \mu(S_1) + \mu(S_2)$ for any disjoint sets of S. The last property also implies that for any $S' \subseteq S$, $\mu(S') \leq \mu(S)$. Let μ be a measure on each set $s_i \cap s_j$, and $t \geq 0$ a constant value. The function α-Measure(s_i, s_j) returns *no-conflict* if $\mu(s_i \cap s_j) \leq t$ otherwise returns *α-conflict*.

```
Function $\alpha$-Measure$(s_i, s_j)$
Result =no-conflict
if $\mu(s_i \cap s_j) > t$ then Result =$\alpha$-conflict end if
Return Result
EndFunction
```

Lemma 80. *The function α-Measure is well-conditioned.*

Proof. (i) α-Measure is *hereditary*. Assume for contradiction that $\mu(s_i \cap s_j) \leq t$ but there is a pair of subsets $s'_i \subseteq s_i, s'_j \subseteq s_j$ with $\mu(s'_i \cap s'_j) > t$. Let $S = (s'_i \cap s'_j)$. First $S \neq \emptyset$, otherwise, $\mu(S) = 0$ and since $t \geq 0$ there would be a contradiction. Second $S \subseteq (s_i \cap s_j)$, thus by the additive property of μ, $\mu(S) \leq \mu(s_i \cap s_j)$. Since $\mu(s_i \cap s_j) \leq t$, the claim holds.

(ii) If $\mu(s_i \cap s_j) > t$, then $|s_i \cap s_j| \geq 1$. This follows because $\mu(\emptyset) = 0$ and $t \geq 0$. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be a pair of subsets with $\alpha(s'_i, s'_j) = 0$, (i.e., $\mu(s'_i \cap s'_j) \leq t$). Note that at most one $s'_i = s_i$ or $s'_j = s_j$; otherwise $\alpha(s'_i, s'_j) = 1$. Since $\mu(s_i \cap s_j) > t$, $\mu(s_i \cap s_j) - \mu(s'_i \cap s'_j) > 0$. In this way, $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) The function μ is computed in polynomial time; thus, we can verify in constant time whether $\mu(s_i \cap s_j) > t$ or not. □

Metric Overlap. Let us assume that $\mathcal{U}$ is a metric space. That is, there is a *metric* or *a distance function* that defines a distance between each pair of elements u, v of $\mathcal{U}$, subject to the following conditions: $dist_{\mathcal{U}}(u, v) \geq 0$, $dist_{\mathcal{U}}(u, v) = 0$ if $(u = v)$, $dist_{\mathcal{U}}(u, v) = dist_{\mathcal{U}}(v, u)$ and $dist_{\mathcal{U}}(u, w) \leq dist_{\mathcal{U}}(u, v) + dist_{\mathcal{U}}(v, w)$. For a constant $d_t > 0$, we define the function α-Metric(s_i, s_j) which returns *no-conflict* if $|s_i \cap s_j| \leq 1$ or $dist_{\mathcal{U}}(u, v) \leq d_t$ for each pair u, v $(u \neq v)$ in $s_i \cap s_j$, else it returns *α-conflict*.

```
Function $\alpha$-Metric$(s_i, s_j)$
if $|s_i \cap s_j| \leq 1$ OR $(dist_{\mathcal{U}}(u, v) \leq d_t$ for each pair $u, v$ in $(s_i \cap s_j), u \neq v)$ then
    Result =no-conflict
else
    Result =$\alpha$-conflict
end if
Return Result
EndFunction
```

Lemma 81. *The function α-Metric is well-conditioned.*

Proof. (i) α-Metric is hereditary. For any pair of sets s_i,s_j that does not α-conflict (i.e., $\alpha(s_i, s_j) = 0$), there is no pair of subsets $s'_i \subseteq s_i$, $s'_j \subseteq s_j$ with $\alpha(s'_i, s'_j) = 1$. Assume the opposite by contradiction. $|s'_i \cap s'_j| \geq 1$, otherwise s'_i and s'_j would not α-conflict. Observe that $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$. In addition, since $\mathcal{U}$ is a metric-space and $\alpha(s_i, s_j) = 0$, $dist_{\mathcal{U}}(u, v) \leq t$ for each pair u, v $(u \neq v)$ in $(s_i \cap s_j)$. Given that we are using $dist_{\mathcal{U}}$ and not $dist_{s'_i \cap s'_j}$, there is no pair of elements in $(s'_i \cap s'_j)$ with $dist_{\mathcal{U}}(u, v) > t$.

(ii) Since $d_t > 0$, for any pair s_i, s_j with $\alpha(s_i, s_j) = 1$, $|s_i \cap s_j| > 1$. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be a pair of subsets with $\alpha(s'_i, s'_j) = 0$. Note that at most one $s'_i = s_i$ or $s'_j = s_j$; otherwise $\alpha(s'_i, s'_j) = 1$. $dist_{\mathcal{U}}(u, v) \leq t$ for each pair u, v $(u \neq v)$ in $(s'_i \cap s'_j)$. Since $\alpha(s_i, s_j) = 1$ but $\alpha(s'_i, s'_j) = 0$, then it must exists at least one element u in $((s_i \cap s_j) \backslash (s'_i \cap s'_j))$ such that $dist_{\mathcal{U}}(u, v) > t$ for some v in $(s_i \cap s_j)$. In this way, $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) Assuming as input a metric space $\mathcal{U}$, α-Metric is verified in $O(r^2)$ time. □

9.3.2 Restricting the Overlap Between Subgraphs

Prescribed Pattern. It has been observed in social networks that the overlap region is often more densely connected than the rest of the community [138]. Inspired by this, we will allow pairwise-overlap in a $(k, \alpha())$-Π-packing, if the overlap region has a specific pattern, for example, it is a clique. More precisely, we say that a pair of Π-subgraphs H_i,H_j does not α-conflict, if $G[V(H_i) \cap V(H_j)]$ is isomorphic to a graph F in a class $\mathcal{F}$. To define a well-conditioned $\alpha()$, $\mathcal{F}$ is a graph class that is hereditary (i.e., it is closed under taking induced subgraphs). To preserve our FPT results, any graph in $\mathcal{F}$ should be polynomial time verifiable. Examples of $\mathcal{F}$ are cliques, planar and chordal graphs. Indeed this applies to any graph class that is closed under minors, since this is hereditary and by the Robertson-Seymour theorem the graph is polynomially testable by checking for the forbidden minors [125]. We define the function α-Pattern(s_i, s_j) that returns *no-conflict* if $|s_i \cap s_j| = 0$ or if $G[s_i \cap s_j]$ is isomorphic to a graph F in $\mathcal{F}$; otherwise, it returns *α-conflict*.

```
Function $\alpha$-Pattern$(s_i, s_j)$
if $|s_i \cap s_j| = 0$ OR $G[s_i \cap s_j]$ is isomorphic to some $F \in \mathcal{F}$ then
    Return no-conflict
else
    Return $\alpha$-conflict
end if
EndFunction
```

Lemma 82. *The function α-Pattern is well-conditioned.*

Proof. (i) α-Pattern is hereditary. Assume for contradiction that there is pair s_i,s_j with $\alpha(s_i, s_j) = 0$ but there is a pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with $\alpha(s'_i, s'_j) = 1$. If $\alpha(s'_i, s'_j) = 1$, this implies that $G[s'_i \cap s'_j]$ is not isomorphic to a graph F in $\mathcal{F}$. Notice that $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$. In addition, $G[s_i \cap s_j]$ is isomorphic to a graph $F \in \mathcal{F}$ (otherwise, $\alpha(s_i, s_j) = 1$). Since $\mathcal{F}$ is a graph class that is hereditary, $G[s'_i \cap s'_j]$ is also isomorphic to F and s'_i and s'_j do not α-conflict.

(ii) It follows by definition of α-Pattern(s_i, s_j) that for any pair s_i, s_j that α-conflict (i.e., $\alpha(s_i, s_j) = 1$) $|s_i \cap s_j| \geq 1$. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be a pair of non-empty subsets where $\alpha(s'_i, s'_j) = 0$. This implies that $G[s'_i \cap s'_j]$ is isomorphic to a graph $F \in \mathcal{F}$. Recall that $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$. Since $G[s_i \cap s_j]$ is not isomorphic to a graph in $\mathcal{F}$ but $G[s'_i \cap s'_j]$ is and $\mathcal{F}$ is closed under taking induced subgraphs, $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) Finally, α-Pattern is computed in polynomial time as it is a constraint of the class $\mathcal{F}$. □

Distance. In [139], overlapping nodes occur only in the *boundary* regions of overlapping communities in sensor networks. Motivated by this, we consider in the overlap region only nodes that are "closer" to each other. In this way, two Π-subgraphs H_i,H_j do not α-conflict if the distance in G between any pair of vertices u, v in $V(H_i) \cap V(H_j)$ is at most a constant $d_t > 0$, i.e., $dist_G(u, v) \leq d_t$. Recall that a Π-subgraph H_i is represented by a set $S_i = V(H_i)$ in $\mathcal{S}$. Since the graph distance is a metric on $V(G)$, we use the function α-Metric defined previously (Lemma 81). Notice that using $dist_{G[H_i \cap H_j]}$ instead is not a well-conditioned $\alpha()$ function. Specifically, it does not follow the hereditary property.

Property. There are several vertex properties that are relevant in the analysis of real networks: vertex strength [19, 110], vertex weight [99], and disparity [110], among others. On the other hand, weights in the edges of a network may better reflect the strength of a relationship in social and communication networks [70]. Moreover, Onnela et al., [118] suggest that edges in the overlapping regions have large weights.

Hence, we suggest considering only overlapping nodes or edges that present the same property ξ (or properties). For example, overlapping nodes or edges each with at least a specific strength or weight. We assume however that the properties values for each vertex or edge are given as part of the input.

Recall that by Transformation 15, we have a set $S = V(H)$ per each induced Π-subgraph H in G. To consider edges properties in the overlap region, we would need to slightly modify that transformation and make $\mathcal{U} = E(G)$ and $S = E(H)$ instead. This only increases the size of the sets in $\mathcal{S}$ by a factor of r and $|\mathcal{S}| = O(n^{r^2})$.

We define α-Property(s_i, s_j) which simply returns *no-conflict* either if $|s_i \cap s_j| = 0$ or if each element u in $(s_i \cap s_j)$ satisfies ξ. Otherwise, it returns *α-conflict*.

```
Function $\alpha$-Property$(S_i, S_j)$
if $|S_i \cap S_j| = 0$ OR (for each element $u$ in $(S_i \cap S_j)$ $u$ satisfies $\xi$ ) then
  Return no-conflict
else
  Return $\alpha$-conflict
end if
EndFunction
```

Lemma 83. *The function α-Property is well-conditioned.*

Proof. (i) α is *hereditary.* Assume by contradiction that there is pair s_i,s_j with $\alpha(s_i, s_j) = 0$ but there is a pair of subsets $\alpha(s'_i, s'_j) = 1$ where $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$. If $\alpha(s_i, s_j) = 0$, every element in $s_i \cap s_j$ satisfies the property ξ. Since $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$, every element in $s'_i \cap s'_j$ satisfies ξ as well.

(ii) By definition of α-Property any pair of disjoint sets do not α-conflict. Let $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ be a pair of subsets with $\alpha(s'_i, s'_j) = 0$. If $\alpha(s_i, s_j) = 1$ but $\alpha(s'_i, s'_j) = 0$, there must exists at least one element in $((s_i \cap s_j) \backslash (s'_i \cap s'_j))$ that does not follow ξ and therefore $((s_i \cap s_j) \backslash (s'_i \cap s'_j)) \neq \emptyset$.

(iii) The property ξ for each element of $\mathcal{U}$ is given as part of the input. Thus, we can verify in constant time whether s_i, s_j α-conflict or not. □

Dense Overlap. We design another $\alpha()$ function to model the behavior that the overlap region is densely connected. To that end, we define α-DenseOverlap(s_i, s_j) that returns *no-conflict* if $|s_i \cap s_j| = 0$ or $|E(G[s_i \cap s_j])| \geq \frac{O(O-1)}{2} - c$, where $O = |s_i \cap s_j|$ and $c \geq 0$ is a constant; otherwise, it returns *α-conflict.*

```
Function $\alpha$-DenseOverlap$(s_i, s_j)$
Set $O = |s_i \cap s_j|$.
if $|s_i \cap s_j| = 0$ or $|E(G[s_i \cap s_j])| \geq \frac{O(O-1)}{2} - c$ then
  Return no conflict
end if
Return $\alpha$-conflict
EndFunction
```

Lemma 84. *The function α-DenseOverlap is well-conditioned.*

Proof. (i) α-DenseOverlap is hereditary. Assume by contradiction that there is pair of sets s_i, s_j with $\alpha(s_i, s_j) = 0$ but there is a pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with $\alpha(s'_i, s'_j) = 1$. Notice that $(s'_i \cap s'_j) \neq \emptyset$; otherwise, $\alpha(s'_i, s'_j) = 0$. Therefore, if $\alpha(s'_i, s'_j) = 1$, then $|E(G[s'_i \cap s'_j])| < \frac{O'(O'-1)}{2} - c$, where $O' = |s'_i \cap s'_j|$. However since $G[s'_i \cap s'_j]$ is an induced subgraph of $G[s_i \cap s_j]$, $|E(G[s_i \cap s_j])| < \frac{O(O-1)}{2} - c$, a contradiction.

(ii) If $\alpha(s_i, s_j) = 1$, $|s_i \cap s_j| \geq 1$. Furthermore, for any pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with $\alpha(s'_i, s'_j) = 0$, $(s_i \cap s_j)\backslash(s'_i \cap s'_j) \neq \emptyset$. Assume otherwise by contradiction. If $\alpha(s'_i, s'_j) = 0$ then $|E(G[s'_i \cap s'_j])| \geq \frac{O'(O'-1)}{2} - c$, where $O' = |s'_i \cap s'_j|$. Thus, if $(s_i \cap s_j)\backslash(s'_i \cap s'_j) = \emptyset$, then $|E(G[s_i \cap s_j])| \geq \frac{O(O-1)}{2} - c$, where $O = |s_i \cap s_j|$, a contradiction to our assumption that $\alpha(s_i, s_j) = 1$.

(iii) We can verify in polynomial time this condition. □

9.4 Packing Problems with Predetermined Cluster Heads

The problem of discovering communities in networks has been tackled with clustering algorithms as well [116]. Many of these algorithms consider as part of the input a collection of sets of vertices $\mathcal{C} = \{C_1, \ldots, C_l\}$ where each set $C_i \subset V(G)$ is called a *cluster head*. The objective is to find a set of clusters in G where each cluster contains exactly one cluster head. In addition, clusters should not share elements of the cluster heads [130, 19, 99, 31, 139].

Motivated by this, we introduce the r-Set Packing with $\alpha()$-Overlap problem (PCH), where PCH stands for Predetermined Clusters Heads.

The r-Set Packing with $\alpha()$-Overlap problem (PCH)
Input: A collection $\mathcal{S}$ of sets each one of size at most r, drawn from a universe $\mathcal{U}$, a non-negative integer k and a collection $\mathcal{C} = \{C_1, \ldots, C_l\}$ of sets of elements where $C_i \subset \mathcal{U}$.
Parameter: k
Question: Does $\mathcal{S}$ contain a $(k, \alpha())$-set packing (PCH), i.e., a set of at least k distinct sets $\mathcal{K} = \{S^*_1, \ldots, S^*_k\}$ subject to the following conditions: each S^*_i contains at least one set of $\mathcal{C}$; for any pair S^*_i,S^*_j with $i \neq j$, $((S^*_i \cap S^*_j) \cap val(\mathcal{C})) = \emptyset$, and S^*_i, S^*_j do not α-conflict?

Recall that a Π-subgraph (or a cluster) is represented by a set in $\mathcal{S}$ (Transformation 15, Section 9.2). Thus, this problem translates into a PCH variation for our Π-Packing problem as well. Also notice that our PCH variation is more relaxed in the sense that each cluster could contain more than one cluster head.

To solve the r-Set Packing with α-Overlap problem (PCH), we need to do two modifications to the BST $\alpha()$-Algorithm described in Section 9.1.

First, we redefine the routine that creates the children of the root of the search tree, and we call it Initialization (PCH) (Algorithm 18). By Lemma 72, a maximal solution $\mathcal{M}$ is used to determine the children of the root. In the (PCH)-variation, we no longer compute $\mathcal{M}$ but rather we use $\mathcal{C}$ to compute those children. That is, the root will have a child i for each possible combination of $\binom{\mathcal{C}}{k}$. Recall that a node i has a collection $\mathbf{Q^i} = \{s^i_1, \ldots, s^i_k\}$. Each set of $\mathbf{Q^i}$ is initialized with set of that combination.

Lemma 85. *If there exists at least one $(k, \alpha())$-set packing (PCH) of $\mathcal{S}$, at least one of the children of the root will have a partial-solution.*

Proof. It follows by the explicit condition that each set in a $(k, \alpha())$-set packing (PCH) should contain at least one set from $\mathcal{C}$ and because the routine Initialization (PCH) tries all possible selections of size k from $\mathcal{C}$ to create the children of the root. □

The second modification to the BST $\alpha()$-ALGORITHM consists on redefining the $\alpha()$ function as $\alpha()$-PCH. Since the sets of a $(k, \alpha())$-set packing (PCH) cannot share elements of $val(\mathcal{C})$, we say that two sets s_i, s_j also α-conflict if there is at least one element of $(s_i \cap s_j)$ in $val(\mathcal{C})$. This new function returns *α-conflict* if $((s_i \cap s_j) \cap val(\mathcal{C})) \neq \emptyset$; otherwise executes the original $\alpha()$ function and returns $\alpha(s_i, s_j)$.

```
Function $\alpha$-PCH($s_i, s_j$)
if $((s_i \cap s_j) \cap val(\mathcal{C})) \neq \emptyset$ then
  Return $\alpha$-conflict
else
  Call the original $\alpha()$ function
  Return $\alpha(s_i, s_j)$
end if
EndFunction
```

Lemma 86. *If the function $\alpha()$ is well conditioned, the function $\alpha()$-PCH is also well-conditioned.*

Proof. (i) α-PCH is hereditary. Assume that α-PCH$(s_i, s_j) = 0$, and there is pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with α-PCH$(s'_i, s'_j) = 1$. Since α is well conditioned, this is only possible if $((s'_i \cap s'_j) \cap val(\mathcal{C})) \neq \emptyset$. However, $(s'_i \cap s'_j) \subseteq (s_i \cap s_j)$ and by our assumption $((s_i \cap s_j) \cap val(\mathcal{C})) = \emptyset$, a contradiction.

(ii) If α-PCH$(s_i, s_j) = 1$, $|s_i \cap s_j| \geq 1$. Assume otherwise by contradiction. Since α is well-conditioned, this is only possible if the extra condition in α-PCH returns *α-conflict* when $s_i \cap s_j = \emptyset$. However, in that case there cannot be an intersection with the set $val(\mathcal{C})$, and α-PCH$(s_i, s_j) = 0$ instead. It remains to show that for any pair of subsets $s'_i \subseteq s_i$ and $s'_j \subseteq s_j$ with α-PCH$(s'_i, s'_j) = 0$, $(s_i \cap s_j) \backslash (s'_i \cap s'_j) \neq \emptyset$. Assume otherwise by contradiction, but in that case again there cannot be an intersection with $val(\mathcal{C})$ and α-PCH$(s_i, s_j) = 0$.

(iii) Since α is well-conditioned, and it takes $O(r)$ time to verify the extra condition in α-PCH, α-PCH is verified in polynomial time. □

The above modifications guarantee that Algorithm 13 will find a $(k, \alpha())$-Set Packing (PCH) if $\mathcal{S}$ has at least one. To maintain our running time, $|\mathcal{C}| = O(g(k))$, where g is a computable function dependent only on k and possibly r but independent of n. Observe that if there is at least one cluster head in $\mathcal{C}$ with more than r elements, we simply output that the instance does not have a solution. Hence, we can state:

Theorem 48. *If $\alpha()$ is well-conditioned, the r-Set Packing with $\alpha()$-Overlap problem (PCH) is solved in $O((g(k))^k (rk)^{(r-1)k} n^r)$ time, where $|\mathcal{C}| = g(k)$ and each set in $\mathcal{C}$ has size at most $r - 1$.*

We could also omit the

condition that clusters cannot share members of the cluster heads as in [45]. In that case, we do not need to redefine the function $\alpha()$.

Algorithm 18 INITIALIZATION (PCH)($\mathcal{C}$)

```
1: $i = 0$, $T = null$
2: while $i < |\binom{\mathcal{C}}{k}|$ do
3:     Let $\{C_1^i, \ldots, C_k^i\}$ be the $i$th combination of $\binom{\mathcal{C}}{k}$
4:     Make $\mathbf{Q^i} = \{s_1^i, \ldots, s_k^i\}$ equal to $\{C_1^i, \ldots, C_k^i\}$, i.e, $s_j^i = C_j^i$
5:     CREATENODE($T$,ROOT,NODE $i$,$\mathbf{Q^i}$)
6:     $i = i + 1$
7: end while
8: Return $T$
```

9.5 Conclusions

In this chapter, we introduced the r-Set Packing and Π-Packing with $\alpha()$-Overlap problems as more universal versions for the problem of packing sets and graphs subject to overlap constraints modeled by a function $\alpha()$. These generalizations could capture a much larger percentage of potential real life applications.

In addition, we obtain fixed-parameter tractable results for our problems when the $\alpha()$ function is subject to specific conditions (Definition 9). It would be interesting to obtain FPT-algorithms for any function α which not necessarily meets Definition 9. For example, when the overlap is bounded by a percentage of the sizes of the communities or when the overlap size has a lower-bound instead of an upper-bound.

Finally, it remains to explore other techniques that could help to design more practical fixed-parameter algorithms for these problems.

Part IV

Conclusions

Chapter 10

Conclusions and Future Work

We have introduced in this thesis generalizations for the classical problems of packing graphs ($\mathcal{H}$-Packing) and packing sets (r-Set Packing). Our generalizations allow overlap in a solution in two different ways: by bounding the pairwise overlap between sets or subgraphs (*family of t-Overlap problems*) and by bounding the number of times an element or a vertex is contained in a set or a subgraph (*family of t-Membership problems*). The motivation of allowing overlap in a packing arises in the context of discovering communities in networks.

We start in Chapter 3 with the formal study of the computational complexity of our packing problems. In that chapter, we provide a dichotomy theorem (analogous to the one of Kirkpatrick and Hell [88]) which shows that for any graph H, the H-Packing with t-Membership is NP-complete if $|V(H)| \geq 3$ and solvable in polynomial time otherwise. A similar result can be stated for the r-Set Packing with t-Membership. We also show that there always exists at least one graph H such that the H-Packing with t-Overlap is NP-complete. We consider these results as a starting point on the study of the computational complexity of our problems as several questions remain open. The complexity of the disjoint-version of our problems ($\mathcal{H}$-Packing) has been studied for specific families of graphs (cycles, cliques, stars, among others - see Section 2.4.1). It would be interesting therefore to investigate the complexity of our problems under those families of graphs. Furthermore, we know the existence of polynomial solvable cases for the H-Packing problem for specific input graphs G (trees and partial k-trees). It is also another research path to determine if those specific graph classes (or perhaps others) consist on polynomial solvable instances for our packing problems with overlap as well.

In Chapter 6, we reduce our packing problems to polynomial kernels using polynomial parametric transformations (PPTs). PPTs were previously used in [96, 42] to obtain kernels (for different problems). In this way, our results strength the suggestion that PPTs could be considered as an alternative kernelization technique. In Chapter 7, we improve our kernel results by using the classical approach of reduction. Thus, the next question arises: are these improved kernel sizes tight? To answer this question, one need to determine kernelization lower bounds for our problems as the ones by Dell and Marx [35] for the r-Set Packing problem. Other relevant topics for future research could be to consider prescribed graphs H or G to see if smaller kernels (perhaps even linear) could be obtained. Paths or trees could be good candidates.

After reducing a problem to a kernel, the problem can be solved (in FPT-time) by running brute-force on the kernel. However, one always wonder if there is a faster way than brute-force. We attempt to solve that question in Chapter 8. In that chapter, we obtain fixed-parameter algorithms for our problems based on search trees and greedy localization techniques. Our algorithms run in time $O(k^{ck})$ (where c is a constant) which asymptotically is equivalent to running brute-force on the kernel. We are able to obtain an $O(2^{ck})$ algorithm by means of an FPT-algorithm [53] for the disjoint version of our problems (r-Set Packing). This algorithm involves the color-coding technique. However, this approach does not solve our problem directly; we need to transform our packing problems with overlap to an instance of the disjoint-version, obtain a solution for the constructed instance (disjoint-version), and then re-interpret this solution to a solution for the original (overlapping) instance. It is therefore crucial to obtain an $O(c^k)$ algorithm that solves our problems directly. Very recently, Gabizon et al. [64] provided an $O(2^{O(rk\frac{\log t}{t})}|\mathcal{S}|n^{O(1)})$ deterministic algorithm for the r-Set Packing with t-Membership problem by using an extended notion of representative sets. Such generalization does not apply for the r-Set Packing with t-Overlap. This result could imply that the t-Overlap variant may be more difficult to solve than the t-Membership version.

Considering overlap in packing problems was introduced to abstract the problem of discovering overlapping communities in networks. Even though it is realistic to expect a small overlap between a pair of communities [62, 111, 31], we wanted to capture situations where the overlap satisfies more complex conditions. This motivated us to pose our problems in an even more general way, and in Chapter 9, we introduce the Π-Packing with $\alpha()$-Overlap problem. In this way, if for example it is expected that the overlap region between a pair of communities contains only nodes with small distance within each other, we could represent that condition through $\alpha()$. We show that when the function $\alpha()$ obeys specific properties (Definition 9), Π-Packing with $\alpha()$-Overlap is in FPT. We also give several $\alpha()$ functions that meet those conditions. A next research step for this problem could be to determine either an FPT-algorithm that solves the problem for functions other than those as in Definition 9, or perhaps determine which conditions in $\alpha()$ make the problem fixed-parameter intractable.

We introduce our packing problems that allow overlap to give a more theoretical treatment to the community discovering problem. We consider that this thesis takes the first steps towards that goal. It would be very appealing to obtain more practical solutions for these problems without leaving a theoretical analysis behind. Perhaps considering specific properties for the graph G that mimic the behavior of real networks (such as degeneracy, average degree node, average path length, among others) could help to achieve that goal. Curiously, even in online social networks which are dynamic networks, there are graph theoretical properties of the network that remain stable over time [132, 135].

On the other hand, the $\mathcal{H}$-Packing and r-Set Packing problems (disjoint version) have been studied for almost four decades. There exists a myriad of results for these problems from different methodologies (classical and parameterized complexity, polynomial solvable instances, approximation algorithms, among others). Not surprisingly, most of those results do not transfer directly to our packing problems allowing overlap. The techniques heavily rely on the assumption that overlap is not allowed in a packing. Thus besides the practical motivation behind of our problems, it is relevant to see now the implication of such overlap at least on the classical and parameterized complexity of packing problems. We believe that there is value on our proposed generalized packing problems and that this thesis could lead to further work on these problems.

Appendix A

Additional Polynomial Parametric Transformations

This appendix contains alternative polynomial parametric transformations for the $\mathcal{H}$-Packing with t-Membership and t-Overlap problems. In Chapter 6, we reduce these problems to a kernel by transforming them to the r-Set Packing with t-Membership and t-Overlap, respectively. Here, we obtain the same kernel sizes as the ones obtained in that chapter, but we transform these problems to the r'-Set Packing (for some $r' > r$) instead.

Recall that the r'-Set Packing problem identifies at least k disjoint sets from a collection $\mathcal{S}$, if possible. Each set of $\mathcal{S}$ is drawn from a universe $\mathcal{U}$ and has at most r' elements. All our PPTs use the collections $\mathcal{V}$ and $\mathcal{E}$. The reader is referred to Chapter 2 for the definitions of those collections.

A.1 From Packing Graphs with Membership to Packing Disjoint Sets

We introduce in this section polynomial parametric transformations from the $\mathcal{H}$-Packing with t-Membership and t-Edge Membership problems to the $(r+1)$-Set Packing problem. These transformations allow us to obtain alternative kernelization algorithms to the ones provided in Chapter 6.

A.1.1 An Alternative PPT for $\mathcal{H}$-Packing with t-Membership

Let us begin with the $\mathcal{H}$-Packing with t-Membership problem. Recall that this problem looks for a set of at least k $\mathcal{H}$-subgraphs such that each vertex of G belongs to at most t of those $\mathcal{H}$-subgraphs. In addition, for each pair of $\mathcal{H}$-subgraphs H_i, H_j, $V(H_i) \neq V(H_j)$.

We create an instance for the $(r+1)$-Set Packing problem, (a universe $\mathcal{U}$ and a collection $\mathcal{S}$) using an instance of the $\mathcal{H}$-Packing with t-Membership problem.

Transformation 16. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r*

Let $r = r(\mathcal{H})$.

The universe $\mathcal{U}$ equals $(V(G) \times \{1, \dots, t\}) \cup \mathcal{V}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ each with at most $r + 1$ elements

$\{\{(v_1, j_1), \dots, (v_i, j_i), \dots, (v_{r'}, j_{r'}), V\} \mid V \in \mathcal{V}$, $V = \{v_1, \dots, v_{r'}\}$, for each $1 \leq j_i \leq t$ and $1 \leq i \leq r'$, where $r' \leq r\}$.

The intuition behind our transformation is that each set in $\mathcal{S}$ would represent an $\mathcal{H}$-subgraph in G. Since each vertex in $V(G)$ can be in at most t $\mathcal{H}$-subgraphs, we duplicate t times each vertex. This would imply that there are t^r sets in $\mathcal{S}$ representing the same $\mathcal{H}$-subgraph H of G. All of those sets have a common element which is precisely the set of vertices in $V(H)$, i.e., some $V \in \mathcal{V} \subset \mathcal{U}$.

Example 14. *Consider the graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $k = 2$, $r = 4$, and $t = 2$. The constructed instance for the $(r + 1)$-Set packing is as follows.*

$$\begin{aligned}
\mathcal{V} =& \{\{a, b, c, d\}, \{b, c, e, f\}, \{b, c, g, h\}\} \\
\mathcal{U} =& \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1), (f, 1), (g, 1), (h, 1), \\
& (a, 2), (b, 2), (c, 2), (d, 2), (e, 2), (f, 2), (g, 2), (h, 2) \\
& \cup \mathcal{V}\}.
\end{aligned}$$

Some sets of the collection $\mathcal{S}$ are

$$\begin{aligned}
&\{(a, 1), (b, 1), (c, 1), (d, 1), \{a, b, c, d\}\}, \\
&\{(a, 1), (b, 1), (c, 1), (d, 2), \{a, b, c, d\}\}, \\
&\{(a, 2), (b, 2), (c, 2), (d, 2), \{a, b, c, d\}\}, \\
&\{(b, 1), (c, 2), (e, 1), (f, 2), \{b, c, e, f\}\}, \\
&\{(b, 1), (c, 1), (g, 2), (h, 2), \{b, c, g, h\}\}.
\end{aligned}$$

A $(2, 5, 2)$-set packing for this instance is

$$\begin{aligned}
\{&\{(a, 1), (b, 1), (c, 1), (d, 1), \{a, b, c, d\}\}, \\
&\{(b, 2), (c, 2), (e, 1), (f, 1), \{b, c, e, f\}\}\}
\end{aligned}$$

This corresponds to the following $(2, 4, 2)$-$\mathcal{H}$-packing

$$\begin{aligned}
&\{H_1 \subseteq G[\{a, b, c, d\}], \\
&H_2 \subseteq G[\{b, c, e, f\}]\}.
\end{aligned}$$

Note that it does not matter which C_4 from the K_4 $G[\{b, c, e, f\}]$ we assign to H_2.

The size of $\mathcal{U}$ is bounded by $|\mathcal{U}| = |V(G)| \times t + |\mathcal{V}| \leq tn + n^r = O(n^r)$. Each set in $\mathcal{S}$ has size at most $r+1$. For each $V \in \mathcal{V}$, we can form at most t^r sets with the tr ordered pairs from the vertices in V. In this way, for each $V \in \mathcal{V}$ there are t^r sets in $\mathcal{S}$, and $|\mathcal{S}| = t^r|\mathcal{V}| = O(t^r n^r)$. This leads us to the following result.

Lemma 87. *Transformation 16 can be computed in $O(t^r n^r)$ time.*

Lemma 88. *G has a (k, r, t)-$\mathcal{H}$-membership if and only if $\mathcal{S}$ has a $(k, r+1, 0)$-set packing (i.e., at least k disjoint sets).*

Proof. We build a $(k, r+1, t)$-set packing $\mathcal{K}_\mathcal{S}$ from a (k, r, t)-$\mathcal{H}$-membership $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $|V(H_j)| + 1$ elements as follows. The element V in S_j corresponds to the set of vertices $V(H_j)$. Note that $V(H_j) \in \mathcal{V} \subset \mathcal{U}$. For each vertex $v_i \in V(H_j)$ $(1 \leq i \leq |V(H_j)| \leq r)$, we add an ordered pair $(v_i, l+1)$ to S_j where l corresponds to the number of $\mathcal{H}$-subgraphs in $\{H_1, \ldots, H_{j-1}\} \subset \mathcal{K}$ that v_i is contained. Since each vertex is contained in at most t $\mathcal{H}$-subgraphs, $0 \leq l \leq t-1$ and $(v_i, l+1)$ always exists in $\mathcal{U}$. By our construction of $\mathcal{S}$, each $S_i \in \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_\mathcal{S}$ are pairwise disjoint. Since $V(H_i) \neq V(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_\mathcal{S}$ will share the same element V. In addition, no pair S_i, S_j shares an ordered pair (v, l) for some $v \in V(G)$ and $1 \leq l \leq t$. If S_i and S_j share a pair (v, l) this would imply that there are two $\mathcal{H}$-subgraphs H_i and H_j that share the same vertex v. Without lost of generality assume that H_i appears first than H_j in $\mathcal{K}$. If (v, l) is contained in both S_i and S_j then v is a member of $l-1$ $\mathcal{H}$-subgraphs from both $\{\ldots, H_i, \ldots\}$ and $\{\ldots, H_i, \ldots, H_j\}$. A contradiction since both H_i and H_j contain v.

We construct a (k, r, t)-$\mathcal{H}$-membership $\mathcal{K}$ using a $(k, r+1, t)$-set packing $\mathcal{K}_\mathcal{S}$. Each set $S_i \in \mathcal{K}_S$ has an element $V_{S_i} \in \mathcal{V}$ which corresponds to a set of vertices in $V(G)$ that forms an isomorphic subgraph to H in G. We add to $\mathcal{K}$ an $\mathcal{H}$-subgraph $H_{S_i} \subseteq G[V_{S_i}]$ for each set $S_i \in \mathcal{K}_S$. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $V(H_{S_i}) \neq V(H_{S_j})$. Now, we need to show that each vertex of $V(G)$ is a member of at most t $\mathcal{H}$-subgraphs in $\mathcal{K}$. Assume otherwise by contradiction. If two $\mathcal{H}$-subgraphs in $\mathcal{K}$ share a vertex v, then there are two sets in $\mathcal{K}_S$ where each one has a pair (v, i) and (v, j), respectively. Since the sets in $\mathcal{K}_S$ are disjoint, then $i \neq j$. If a vertex $v \in V(G)$ is a member of more than t $\mathcal{H}$-subgraphs in $\mathcal{K}$, then there are at least $t+1$ ordered pairs $(v, 1), \ldots, (v, t+1)$ contained in $t+1$ different sets in $\mathcal{K}_S$. However, by our construction of $\mathcal{U}$, there are at most t ordered pairs that contain v. □

After that we run the kernelization algorithm for the $(r+1)$-Set Packing problem [1]. This algorithm would leave us with a new universe $\mathcal{U}'$ with at most $2(r-1)!(rk-1)^{r-1}$ elements, as well with a collection $\mathcal{S}'$ of subsets.

Lemma 89. *$\mathcal{S}$ has a $(k, r+1, 0)$-set packing if and only if $\mathcal{S}'$ has a $(k, r+1, 0)$-set packing.*

Next we construct a reduced graph G' using $\mathcal{U}'$.

Transformation 17. ***Input:*** *$G, \mathcal{U}'$;* ***Output:*** *G'*

Take each vertex v that appears in each $(v, i) \in \mathcal{U}'$ and the vertices in each $V \in \mathcal{V}'$ and consider the graph G' induced by all these in G.

By this construction, G' has at most $2(r-1)!(rk-1)^{r-1}$ vertices.

Lemma 90. *G' has a (k,r,t)-$\mathcal{H}$-membership if and only if $\mathcal{S}'$ has a $(k,r+1,0)$-set packing.*

Theorem 49. *$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O(r^r k^{r-1})$ vertices, where $r = r(\mathcal{H})$.*

The induced version of the problem obtains at least k induced $\mathcal{H}$-subgraphs in G where each vertex in $V(G)$ is contained in at most t of these $\mathcal{H}$-subgraphs (if possible). To achieve a problem kernel, we only need to redefine $\mathcal{V}$ as a collection of sets of vertices each of which must induce an $\mathcal{H}$-subgraph in G.

Theorem 50. *Induced-$\mathcal{H}$-Packing with t-Membership has a problem kernel with $O(r^r k^{r-1})$ vertices, where $r = r(\mathcal{H})$.*

In this way, the kernelization algorithm for the $\mathcal{H}$-Packing with t-Membership that we described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, $(r+1)$-*SetPacking*, Transformation 16, Algorithm [1], and Transformation 17.

A.1.2 An Alternative PPT for $\mathcal{H}$-Packing with t-Edge Membership

In the $\mathcal{H}$-Packing with t-Edge Membership problem, the goal is to obtain a collection of at least k $\mathcal{H}$-subgraphs in G such that each edge of $E(G)$ belongs to at most t of those subgraphs. In addition, for each pair of $\mathcal{H}$-subgraphs, $E(H_i) \neq E(H_j)$.

We transform an instance of the $\mathcal{H}$-Packing with t-Edge Membership problem to an instance of the $(r+1)$-Set Packing problem.

Transformation 18. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r*

Let $r = m(\mathcal{H})$.

The universe $\mathcal{U}$ equals $(E(G) \times \{1, \ldots, t\}) \cup \mathcal{E}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ each with at most $r+1$ elements

$\{\{(e_1, j_1), \ldots, (e_{m'}, j_{m'}), E\} \mid E \in \mathcal{E},\ E = \{e_1, \ldots e_{m'}\}$*for each* $1 \leq j_i \leq t$ *and* $1 \leq i \leq m'$*, where* $m' \leq m\}$.

Example 15. *Consider that graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $k = 2$, $r = 4$, and $t = 2$. The instance for the $(r+1)$-set packing looks like follows.*

$$\begin{aligned}
\mathcal{E} =& \{\{ab, ad, bc, dc\}, \{gh, hc, bc, dc\}, \{bc, cf, ef, be\}, \{bf, cf, ec, be\}, \{bc, bf, ef, ec\}\} \\
\mathcal{U} =& \{(ab,1), (ad,1), (dc,1), (bc,1), (be,1), (ef,1), (cf,1), (bf,1), (ec,1), (gb,1), (dc,1), (hc,1), \\
& (ab,2), (ad,2), (dc,2), (bc,2), (be,2), (ef,2), (cf,2), (bf,2), (ec,2), (gb,2), (dc,2), (hc,2) \\
& \cup \mathcal{E}\}.
\end{aligned}$$

Some sets of the collection $\mathcal{S}$ are

$$\begin{aligned}&\{(ad,1),(ab,1),(bc,1),(dc,1),\{ad,ab,bc,dc\}\},\\&\{(bc,1),(cf,1),(ef,1),(be,1),\{bc,cf,ef,be\}\},\\&\{(bf,1),(cf,1),(ec,1),(be,1),\{bf,cf,ec,be\}\},\\&\{(bc,1),(bf,1),(ef,1),(ec,1),\{bc,bf,ef,ec\}\},\\&\{(gh,1),(dc,1),(bc,1),(hc,1),\{gh,dc,bc,hc\}\}.\end{aligned}$$

An $(2,5,2)$-set packing for this instance is

$$\begin{aligned}&\{\{(bf,1),(cf,1),(ec,1),(be,1),\{bf,cf,ec,be\}\},\\&\ \{(bc,1),(bf,2),(ef,1),(ec,2),\{bc,bf,ef,ec\}\}\}\end{aligned}$$

This corresponds to the following $(2,4,2)$-edge-$\mathcal{H}$-membership

$$\begin{aligned}\{H_1&=(\{b,e,c,f\},\{bf,cf,ec,be\}),\\H_2&=(\{b,e,c,f\},\{bf,cf,ec,be\}).\end{aligned}$$

Notice that the subgraphs have identical set of vertices. The NISV version of the $\mathcal{H}$-Packing with t-Edge Membership problem avoids this situation (see Section 6.1.3).

The size of our constructed instance is bounded by $|\mathcal{U}| = |E(G)| \times t + |\mathcal{E}| \leq tn + n^r = O(n^r)$. Each set in $\mathcal{S}$ has size $r+1$, and $|\mathcal{S}| \leq t^r n^r$. Recall that $r = m(\mathcal{H})$.

Lemma 91. *Transformation 18 can be computed in $O(t^r n^r)$.*

Lemma 92. *G has an (k,r,t)-edge-$\mathcal{H}$-membership if and only if $\mathcal{S}$ has a $(k,r+1,0)$-set packing.*

Proof. We build a $(k,r+1,t)$-set packing $\mathcal{K}_{\mathcal{S}}$ from a (k,r,t)-edge-$\mathcal{H}$-membership $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $|E(H_j)|+1$ elements as follows. The element E in S_j corresponds to the set of edges $E(H_j)$. Note that $E(H_j) \in \mathcal{E} \subset \mathcal{U}$. For each edge $e_i \in E(H_j)$ $(1 \leq i \leq |E(H_j)| \leq m)$, we add an ordered pair $(e_i, l+1)$ to S_j where l corresponds to the number of $\mathcal{H}$-subgraphs in $\{H_1,\dots,H_{j-1}\} \subset \mathcal{K}$ that e_i is a member. Since each edge is a member of at most t $\mathcal{H}$-subgraphs, $0 \leq l \leq t-1$ and $(e_i, l+1)$ always exists in $\mathcal{U}$. By our construction of $\mathcal{S}$, each $S_i \in \mathcal{K}_{\mathcal{S}} \subseteq \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_{\mathcal{S}}$ are pairwise disjoint. Since $E(H_i) \neq E(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_{\mathcal{S}}$ will share the same element E. On the other hand, no pair S_i, S_j shares an ordered pair (e,l) for some edge $e \in E(G)$ and some integer $1 \leq l \leq t$. If S_i and S_j share a pair (e,l) this would imply that there are two $\mathcal{H}$-subgraphs H_i and H_j that share the same edge e. Without lost of generality assume that H_i appears first than H_j in $\mathcal{K}$. If (e,l) is contained in both S_i and S_j then e is a member of $l-1$ $\mathcal{H}$-subgraphs from both $\{\dots,H_i,\dots\}$ and $\{\dots,H_i,\dots,H_j\}$. A contradiction since both H_i and H_j contain e.

We construct a (k, r, t)-edge-$\mathcal{H}$-membership $\mathcal{K}$ using a $(k, r+1, t)$-set packing $\mathcal{K}_{\mathcal{S}}$. Each set $S_i \in \mathcal{K}_S$ has an element $E_{S_i} \in \mathcal{E}$ which corresponds to a set of edges in $E(G)$ that forms an isomorphic subgraph to H in G. In this way, for each set $S_i \in \mathcal{K}_S$, we add to $\mathcal{K}$ an $\mathcal{H}$-subgraph H_{S_i} with a set of vertices $V(E_{S_i})$ and a set of edges E_{S_i}, i.e., $H_{S_i} = (V(E_{S_i}), E_{S_i})$. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $E(H_{S_i}) \neq E(H_{S_j})$. Now, we need to show that each edge of $E(G)$ is a member of at most t $\mathcal{H}$-subgraphs in $\mathcal{K}$. Assume otherwise by contradiction. If two $\mathcal{H}$-subgraphs in $\mathcal{K}$ share an edge e, then there are two sets in $\mathcal{K}_S$ where each one has a pair (e, i) and (e, j), respectively. Since the sets in $\mathcal{K}_S$ are disjoint, $i \neq j$. If an edge $e \in E(G)$ is a member of more than t $\mathcal{H}$-subgraphs in $\mathcal{K}$ then there are at least $t+1$ ordered pairs $(e, 1), \ldots, (e, t+1)$ contained in $t+1$ different sets in $\mathcal{K}_S$. However, by our construction of $\mathcal{U}$, there are at most t ordered pairs that contain e. □

Similarly as in the vertex-version, the kernelization algorithm [1] will leave us with a reduced universe $\mathcal{U}'$ with at most $O((r+1)^r k^r)$ elements.

Lemma 93. *$\mathcal{S}$ has a $(k, r+1, 0)$-set packing if and only if $\mathcal{S}'$ has a $(k, r+1, 0)$-set packing.*

We now obtain a reduced graph G' using $\mathcal{U}'$ as follows.

Transformation 19. ***Input:*** *$G, \mathcal{U}'$;* ***Output:*** *G'*

We take the end-points of each edge e that appears in each $(e, i) \in \mathcal{U}'$ and the vertices in each $V \in \mathcal{V}'$ and consider the graph G' that is induced by all these in G.

Lemma 94. *G' has a (k, r, t)-edge-$\mathcal{H}$-membership if and only if $\mathcal{S}'$ has a $(k, r+1, 0)$-set packing.*

Theorem 51. *The $\mathcal{H}$-Packing with t-Edge Membership problem has a kernel with $O((r+1)^r k^r)$ vertices, where $r = m(\mathcal{H})$.*

The kernelization algorithm for the $\mathcal{H}$-Packing with t-Edge Membership that we described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, $(r+1)$-*Set Packing*, Transformation 18, Algorithm [1], and Transformation 19.

A.2 From Packing Graphs with Pairwise Overlap to Packing Disjoint Sets

This section introduces polynomial parametric transformations from the $\mathcal{H}$-Packing with t-Overlap and t-Edge Overlap to the $(\binom{r}{t+1}+1)$-Set Packing problem. These transformations allow us to obtain alternative kernelization algorithms for these problems with the same kernel sizes as the ones presented in Chapter 6.

A.2.1 An Alternative PPT for $\mathcal{H}$-Packing with t-Overlap

In this section, we will transform an instance of the $\mathcal{H}$-Packing with t-Overlap problem to an instance of the $(\binom{r}{t+1}+1)$-Set Packing problem.

Transformation 20. ***Input:*** *G, $\mathcal{V}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r*

Let $r = r(\mathcal{H})$

The universe $\mathcal{U}$ is defined as $\mathcal{V} \cup \{I \mid I \subset V(G), |I| = t+1\}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ with at most $\binom{r}{t+1} + 1$ elements, of the following form:

$\{V = \{v_1, \ldots, v_{|V|}\}, \{I_1\}, \ldots, \{I_{\binom{|V|}{t+1}}\} \mid V \in \mathcal{V}$ and each $I_i \subset V$ where $|I_i| = t+1\}$.

Intuitively, there is a set S in the collection $\mathcal{S}$ per each $\mathcal{H}$-subgraph H in G. The set S contains one element V that represents the set of vertices $V(H)$, and $\binom{|V(H)|}{t+1}$ elements each one representing a subset of $t+1$ vertices from $V(H)$. In this way, any pair sets in $\mathcal{S}$ that are disjoint corresponds to a pair of $\mathcal{H}$-subgraphs that overlaps in at most t vertices.

Example 16. *Consider the graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $k = 2$, $r = 4$, and $t = 2$. The instance of the $(\binom{r}{t+1} + 1)$-Set Packing problem corresponds to.*

$$\begin{aligned}
\mathcal{V} =& \{\{a,b,c,d\},\{b,c,e,f\},\{b,c,g,h\}\} \\
\mathcal{U} =& \{\mathcal{V} \ \cup \\
& \{a,b,c\},\{a,b,d\},\{b,c,d\},\{b,c,g\},\{b,c,h\},\{c,g,h\},\ldots, \\
& \{b,c,e\},\{b,c,f\},\{e,c,f\}\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} =& \{S_1 = \{\{a,b,c,d\},\{a,b,c\},\{a,b,d\},\{b,c,d\}\}, \\
& S_2 = \{\{b,c,g,h\},\{b,c,g\},\{b,c,h\},\{c,g,h\}\}, \\
& S_3 = \{\{b,c,e,f\},\{b,c,e\},\{b,c,f\},\{e,c,f\}\}\}
\end{aligned}$$

A $(2,4,2)$-set packing of $\mathcal{S}$ is $\{S_1, S_2\}$. This corresponds to the $(2,4,2)$-$\mathcal{H}$-packing $H_1 \subseteq G[\{a,b,c,d\}]$ and $H_2 \subseteq G[\{b,c,g,h\}]$.

The size of our constructed instance is bounded by $|\mathcal{U}| \leq n^r + \binom{n}{t+1} = O(n^r + n^{t+1})$. Each set $S \in \mathcal{S}$ has at most $\binom{r}{t+1} + 1$ elements, and there is one set in $\mathcal{S}$ per set in $\mathcal{V}$. Thus, the size of $\mathcal{S}$ is $O(n^r)$.

Lemma 95. *Transformation 20 can be computed in $O(n^r + n^{t+1})$ time.*

Lemma 96. *G has a (k,r,t)-$\mathcal{H}$-packing if and only if $\mathcal{S}$ has a $(k, \binom{r}{t+1} + 1, 0)$-set packing (i.e., at least k disjoint sets).*

Proof. We build a $(k, \binom{r}{t+1} + 1, 0)$-set packing $\mathcal{K}_{\mathcal{S}}$ from a (k,r,t)-$\mathcal{H}$-packing $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j with $\binom{|V(H_j)|}{t+1} + 1$ elements as follows. The element V in S_j corresponds to the set of vertices $V(H_j)$. We add one element to S_j for each subset of $t+1$ vertices from $V(H_j)$. Thus, we are adding $\binom{|V(H_j)|}{t+1}$ elements to S_j each one corresponding to a set of $t+1$ vertices. By our construction of $\mathcal{S}$, each $S_j \in \mathcal{K}_{\mathcal{S}} \subseteq \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_{\mathcal{S}}$ are pairwise disjoint. Since $V(H_i) \neq V(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_{\mathcal{S}}$ will share the same element V. On the other hand, no pair S_i, S_j shares a set I with $t+1$ elements. Otherwise, if S_i and S_j would share an element $\{v_1, \ldots, v_{t+1}\}$ for some set of $t+1$ vertices of $V(G)$, then there are two $\mathcal{H}$-subgraphs H_i and H_j that overlap in at least $t+1$ vertices, a contradiction.

We construct a (k, r, t)-$\mathcal{H}$-packing $\mathcal{K}$ using a $(k, \binom{r}{t+1} + 1, 0)$-set packing $\mathcal{K}_{\mathcal{S}}$. Each set $S_i \in \mathcal{K}_S$ has an element $V_{S_i} \in \mathcal{V}$ which corresponds to a set of vertices in $V(G)$ that forms an isomorphic subgraph to H in G. We add to $\mathcal{K}$ an $\mathcal{H}$-subgraph $H_{S_i} \subseteq G[V_{S_i}]$ for each set $S_i \in \mathcal{K}_S$. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $V(H_{S_i}) \neq V(H_{S_j})$. Now, we need to show that each pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ overlaps in at most t vertices. Assume otherwise by contradiction. If a pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ shares $t+1$ vertices ($\{v_1, \ldots, v_{t+1}\}$), then there is a pair of sets S_i, S_j in $\mathcal{K}_S$ that has an element $I = \{v_1 \ldots v_{t+1}\}$ in common. However, this would imply that S_i and S_j are not pairwise-disjoint, a contradiction. □

We could now run a kernelization algorithm for the $(\binom{r}{t+1} + 1)$-Set Packing problem which will leave us with a reduced universe $\mathcal{U'}^T$ with at most $2r^{t+1}((r^{t+1}+1)k - (r^{t+1}+1))^{r^{t+1}}$ elements as well with a reduced collection of sets $\mathcal{S}'$.

Lemma 97. *$\mathcal{S}$ has a $(k, \binom{r}{t+1} + 1, 0)$-set packing if and only if $\mathcal{S}'$ has a $(k, \binom{r}{t+1} + 1, 0)$-set packing.*

Next we construct a reduced graph G' using $\mathcal{U}'$.

Transformation 21. ***Input:*** *G, $\mathcal{U}'$;* ***Output:*** *G'*

Take each vertex v that appears in each $\mathcal{U}'$ and consider the graph G' induced by all these in G.

By this construction, G' has at most $2r^{t+1}((r^{t+1}+1)k - (r^{t+1}+1))^{r^{t+1}}$ vertices.

Lemma 98. *G' has a (k, r, t)-$\mathcal{H}$-packing if and only if $\mathcal{S}'$ has a $(k, \binom{r}{t+1} + 1, 0)$-set packing.*

Theorem 52. *$\mathcal{H}$-Packing with t-Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = r(\mathcal{H})$.*

The induced version of the problem aims to obtain at least k induced $\mathcal{H}$-subgraphs in G where each pair of $\mathcal{H}$-subgraphs shares at most t vertices, if possible. To achieve a transformation from this problem to the $(\binom{r}{t+1} + 1)$-Set Packing problem, we only need to redefine $\mathcal{V}$ as a collection of sets of vertices each of which must induce an $\mathcal{H}$-subgraph in G.

In this way, the kernelization algorithm for the $\mathcal{H}$-Packing with t-Overlap that we described above corresponds to Algorithm 2 with input G, $\mathcal{H}$, k, $(\binom{r}{t+1} + 1)$-*Set Packing*, Transformation 20, Algorithm [1], and Transformation 21.

A.2.2 An Alternative PPT for $\mathcal{H}$-Packing with t-Edge Overlap

The transformation for this version of the problem follows closely the one for the vertex-version. The difference is that now we are having all possible sets of $t+1$ edges from an $\mathcal{H}$-subgraph (instead of sets of $t+1$ vertices) in $\mathcal{U}$ and $\mathcal{S}$.

Transformation 22. ***Input:*** *G, $\mathcal{H}$;* ***Output:*** *$\mathcal{U}$, $\mathcal{S}$, and r.*

Let $r = m(\mathcal{H})$

The universe $\mathcal{U}$ is defined as $\mathcal{E} \cup \{I \mid I \subset E(G), |I| = t+1\}$.

The collection $\mathcal{S}$ contains all subsets of $\mathcal{U}$ with at most $\binom{r}{t+1} + 1$ elements, of the following form:

$\{E, \{I_1\}, \dots, \{I_{\binom{|E|}{t+1}}\} \mid E \in \mathcal{E}$ and each $I_i \subset E$ where $|I_i| = t+1\}$.

Example 17. *Consider the graph of Figure 6.2 with $\mathcal{H} = \{C_4\}$, $k = 2$, $r = 4$, and $t = 2$. The instance of the $(\binom{r}{t+1} + 1)$-Set Packing problem is as follows.*

$$\begin{aligned}
\mathcal{E} &= \{\{ab, ad, bc, dc\}, \{gh, hc, bc, dc\}, \{bc, cf, ef, be\}, \{bf, cf, ec, be\}, \{bc, bf, ef, ec\}\} \\
\mathcal{U} &= \mathcal{E} \ \cup \ \{ab, ad, bc\}, \{ab, ad, dc\}, \{ab, ad, be\}, \dots, \\
&\quad \{gb, gh, hc\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} = \{&\{S_1 = \{ab, ad, bc, dc\}, \{ab, ad, bc\}, \{ab, ad, cd\}, \{ad, bc, cd\}, \{ab, bc, cd\}\}, \\
&\{S_2 = \{bc, gh, hc, bg\}, \{bc, gh, hc\}, \{bc, gh, bg\}, \{gh, hc, bg\}, \{bc, hc, gc\}\}, \\
&\{S_3 = \{bc, cf, ef, eb\}, \{bc, cf, ef\}, \{bc, cf, eb\}, \{cf, ef, eb\}, \{bc, ef, eb\}\}, \\
&\{S_4 = \{be, cf, ef, eb, bf, ce\}, \{bc, cf, ef\}, \{bc, cf, eb\}, \dots, \{eb, bf, ce\}\}\}
\end{aligned}$$

A $(2, 21, 2)$-set packing for this instance is $\{S_1, S_4\}$. This corresponds to the $(2, 4, 2)$-$\mathcal{H}$-packing $H_1 = (\{a, b, c, d\}, \{ab, ad, bc, dc\}))$ and $H_2 = (\{b, e, c, f\}, \{be, cf, ef, eb, bf, ce\}$.

Lemma 99. *Transformation 22 can be computed in $O(n^{r^2} + n^{2(t+1)})$ time.*

Lemma 100. *G has a (k, r, t)-edge-$\mathcal{H}$-packing if and only if $\mathcal{S}$ has a $(k, \binom{r}{t+1} + 1, 0)$-set packing (i.e., at least k disjoint sets).*

Proof. We build a $(k, \binom{r}{t+1}, t)$-set packing $\mathcal{K}_\mathcal{S}$ from a (k, r, t)-$\mathcal{H}$-edge-packing $\mathcal{K}$. For each $H_j \in \mathcal{K}$, we add a set S_j to $\mathcal{K}_\mathcal{S}$ with $\binom{|E(H_j)|}{t+1} + 1$ elements as follows. The element E in S_j corresponds to the set of edges $E(H_j)$. We add an element to S_j for each subset of $t+1$ edges from $E(H_j)$. Thus, we are adding $\binom{|E(H_j)|}{t+1}$ elements to S_j each one is a set of size $t+1$. By our construction of $\mathcal{S}$, each $S_j \in \mathcal{K}_\mathcal{S} \subseteq \mathcal{S}$. It remains to show that the sets in $\mathcal{K}_\mathcal{S}$ are pairwise disjoint. Since $E(H_i) \neq E(H_j)$ for each pair $H_i, H_j \in \mathcal{K}$, none of the sets in $\mathcal{K}_\mathcal{S}$ will share the same element E. On the other hand, no pair S_i, S_j shares an element I. If S_i and S_j would share an element $I = \{e_1, \dots, e_{t+1}\}$ for some set of $t+1$ edges of $E(G)$, there would be two $\mathcal{H}$-subgraphs H_i and H_j that overlap in at least $t+1$ edges, a contradiction.

We construct a (k, r, t)-edge-$\mathcal{H}$-packing $\mathcal{K}$ using a $(k, \binom{r}{t+1}, t)$-set packing $\mathcal{K}_\mathcal{S}$. Each set $S_i \in \mathcal{K}_S$ has an element $E_{S_i} \in \mathcal{E}$ which corresponds to a set of edges in $E(G)$ that forms an $\mathcal{H}$-subgraph H_{S_i}. In this way, we add to $\mathcal{K}$ an $\mathcal{H}$-subgraph H_{S_i} with edges E_{S_i} for each set $S_i \in \mathcal{K}_S$ and vertices $V(E_{S_i})$. Since the sets in $\mathcal{K}_S$ are pairwise-disjoint, $E(H_{S_i}) \neq E(H_{S_j})$. Now, we need to show that each pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ overlaps in at most t edges. Assume otherwise by contradiction. If a pair of $\mathcal{H}$-subgraphs in $\mathcal{K}$ would share at least $t+1$ edges ($\{e_1, \ldots, e_{t+1}\}$), then there would be a pair of sets S_i, S_j in $\mathcal{K}_S$ that has an element $I = \{e_1 \ldots e_{t+1}\}$ in common. However, this would imply that S_i and S_j are not pairwise-disjoint, a contradiction. $\square$

Using similar steps as with the $\mathcal{H}$-Packing with t-Overlap (vertex-version), we can reduce this transformed instance to a kernel.

Theorem 53. *$\mathcal{H}$-Packing with t-Edge Overlap has a problem kernel with $O((r^{t+1}+1)^{r^{t+1}} k^{r^{t+1}})$ vertices, where $r = m(\mathcal{H})$.*

Appendix B

Compendium of Parameterized Problems

We list in this appendix all the parameterized problems studied in this thesis. For the set problems, each set in $\mathcal{S}$ has at most r elements while for the graph problems, each graph in $\mathcal{H}$ has at most r vertices and m edges.

B.1 Set Problems

The r-Set Packing
Input: A collection $\mathcal{S}$ of distinct sets, each of size at most r, drawn from a universe $\mathcal{U}$ of size n, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a $(k, r, 0)$-set packing, i.e., at least k sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where $|S_i \cap S_j| = 0$, for any pair S_i, S_j with $i \neq j$?

The r-Set Packing with t-Membership problem
Input: A collection $\mathcal{S}$ of distinct sets, each of size at most r, drawn from a universe $\mathcal{U}$ of size n, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a (k, r, t)-set membership, i.e., at least k sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where each element of $\mathcal{U}$ is in at most t sets of $\mathcal{K}$?

The r-Set Packing with t-Overlap problem
Instance: A collection $\mathcal{S}$ of distinct sets, each of size at most r, drawn from a universe $\mathcal{U}$ of size n, and a non-negative integer k.
Parameter: k.
Question: Does $\mathcal{S}$ contain a (k, r, t)-set packing, i.e., a collection of at least k sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where $|S_i \cap S_j| \leq t$, for any pair S_i, S_j with $i \neq j$?

The r-Set Packing with $\alpha()$-Overlap problem
Input: A collection $\mathcal{S}$ of sets each of size at most r, drawn from a universe $\mathcal{U}$, and a non-negative integer k.
Parameter: k
Question: Does $\mathcal{S}$ contain a $(k, \alpha())$-set packing, i.e., at least k distinct sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ where for each pair S_i, S_j $(i \neq j)$ $\alpha(S_i, S_j) = 0$?

The r-Set Packing with $\alpha()$-Overlap (PCH) problem
Input: A collection $\mathcal{S}$ of sets each of size at most r, drawn from a universe $\mathcal{U}$, a non-negative integer k and a collection $\mathcal{C} = \{C_1, \ldots, C_l\}$ of sets of elements where $C_i \subset \mathcal{U}$.
Parameter: k
Question: Does $\mathcal{S}$ contain a $(k, \alpha())$-set packing (PCH), i.e., a set of at least k distinct sets $\mathcal{K} = \{S_1, \ldots, S_k\}$ subject to the following conditions: each S_i^* contains at least one set of $\mathcal{C}$; for any pair S_i,S_j with $i \neq j$, $((S_i \cap S_j) \cap val(\mathcal{C})) = \emptyset$, and S_i, S_j do not α-conflict?

B.2 Graph Problems

The $\mathcal{H}$-Packing problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a $(k, r, 0)$-$\mathcal{H}$-packing, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph from $\mathcal{H}$ and $|V(H_i) \cap V(H_j)| = 0$ for any pair H_i, H_j with $i \neq j$?

The Edge-$\mathcal{H}$-Packing problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a $(k, m, 0)$-edge-$\mathcal{H}$-packing, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, and $|E(H_i) \cap E(H_j)| = 0$ for any pair H_i, H_j with $i \neq j$?

The $\mathcal{H}$-Packing with t-Membership problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k,r,t)-$\mathcal{H}$-*membership*, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$ for $i \neq j$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

The Induced-$\mathcal{H}$-Packing with t-Membership problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k,r,t)-induced-$\mathcal{H}$-membership, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$ for $i \neq j$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Membership (ISV) problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k,r,t)-$\mathcal{H}$-membership (ISV), i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$, and every vertex in $V(G)$ is contained in at most t subgraphs of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Edge Membership problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k,m,t)-edge-$\mathcal{H}$-membership, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$ for $i \neq j$, and every edge in $E(G)$ belongs to at most t subgraphs of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Edge Membership (NISV) problem
Input: A graph G, and a positive integer k.
Parameter: k
Question: Does G contain a (k,r,t)-edge-$\mathcal{H}$-membership (NISV), i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$, and every edge in $E(G)$ belongs to at most t H-subgraphs of $\mathcal{K}$?

The $\mathcal{H}$-Packing with t-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k,r,t)-$\mathcal{H}$-packing, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \ldots, H_k\}$ where each H_i is isomorphic to some graph from $\mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|V(H_i) \cap V(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The Induced-$\mathcal{H}$-Packing with t-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, r, t)-induced-$\mathcal{H}$-packing, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1, \dots, H_k\}$ where each H_i is isomorphic to some graph from $\mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|V(H_i) \cap V(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The $\mathcal{H}$-Packing with t-Edge-Overlap problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, m, t)-edge-$\mathcal{H}$-packing, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \dots, H_k\}$ where each H_i is isomorphic to a graph $H \in \mathcal{H}$, $E(H_i) \neq E(H_j)$, and $|E(H_i) \cap E(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The $\mathcal{H}$-Packing with t-Edge Overlap (NISV) problem
Input: A graph G, and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, m, t)-edge-$\mathcal{H}$-packing, i.e., a set of at least k subgraphs $\mathcal{K} = \{H_1, \dots, H_k\}$ where each H_i is isomorphic to a graph $H \in \mathcal{H}$, $V(H_i) \neq V(H_j)$, and $|E(H_i) \cap E(H_j)| \leq t$ for any pair H_i, H_j with $i \neq j$?

The Π-Packing with $\alpha()$-Overlap problem
Input: A graph G and a non-negative integer k.
Parameter: k
Question: Does G contain a (k, α)-Π-packing, i.e., a set of at least k induced subgraphs $\mathcal{K} = \{H_1, \dots, H_k\}$, subject to the following conditions: i. each H_i has at most r vertices and obeys the property Π, and ii. for any pair H_i, H_j, with $i \neq j$, $\alpha(H_i, H_j) = 0$ and $V(H_i) \neq V(H_j)$?

References

[1] Faisal N. Abu-Khzam. An improved kernelization algorithm for r-Set Packing. *Information Processing Letters*, 110(16):621–624, 2010. 17, 50, 51, 55, 59, 61, 62, 66, 91, 130, 131, 133, 135

[2] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006. iii, 3, 4, 107

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. 16, 17, 99

[4] R. Anstee, 1996. Personal Communication to Pavol Hell. 14

[5] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 37–54. ACM, 2012. 3

[6] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation.* Springer-Verlag, 1999. 5

[7] Hemant Balakrishnan and Narsingh Deo. Detecting communities using bibliographic metrics. In *Proceedings of the IEEE International Conference on Granular Computing*, pages 293–298. IEEE, 2006. 2, 3

[8] Suman Banerjee and Samir Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proceedings of 20th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 2, pages 1028–1037. IEEE Society Press, 2001. 4

[9] Reuven Bar-Yehuda, Magnús M. Halldórsson, Joseph (Seffi) Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 732–741. Society for Industrial and Applied Mathematics, 2002. 4

[10] Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismail. Efficient identification of overlapping communities. In Paul Kantor, Gheorghe Muresan, Fred Roberts, Daniel D. Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *Intelligence and Security Informatics*, volume 3495 of *Lecture Notes in Computer Science*, pages 27–36. Springer Berlin Heidelberg, 2005. iii, 3

[11] Ramon Bejar, Bhaskar Krishnamachari, Carla Gomes, and Bart Selman. Distributed Constraint Satisfaction in a Wireless Sensor Tracking System. Workshop on Distributed Constraint Reasoning, International Joint Conference on Artificial Intelligence, 2001. 4

[12] Fran Berman, David Johnson, Tom Leighton, Peter W. Shor, and Larry Snyder. Generalized planar matching. *Journal of Algorithms*, 11(2):153–184, 1990. 13

[13] Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM Journal on Computing*, 43(1):280–299, 2014. 17

[14] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. 17

[15] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. 15

[16] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Analysis of Data Reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008. iv, 10, 39, 46, 93

[17] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeoc. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412:4570–4578, 2011. 15, 16

[18] Alberto Caprara and Romeo Rizzi. Packing triangles in bounded degree graphs. *Information Processing Letters*, 84(4):175 – 180, 2002. 13, 16, 27

[19] Duanbing Chen, Mingsheng Shang, Zehua Lv, and Yan Fu. Detecting overlapping communities of weighted networks via a local algorithm. *Physica A: Statistical Mechanics and its Applications*, 389(19):4177 – 4187, 2010. 120, 122

[20] J Chen, Q Feng, Y Liu, S Lu, and J Wang. Improved deterministic algorithms for weighted matching and packing problems. *Theor. Comput. Sci.*, 412(23):2503–2512, 2011. 17, 18

[21] J Chen, D Friesen, W Jia, and I Kanj. Using nondeterminism to design effcient deterministic algorithms. *Algorithmica*, 40(2):83–97, 2004. 17, 18

[22] J Chen, J Kneis, S Lu, D Molle, S Richter, P Rossmanith, S H Sze, and F Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM J. on Computing*, 38(6):2526–2547, 2009. 17, 18

[23] Jianer Chen, Henning Fernau, Iyad A. Kanj, and Ge Xia. Parametric Duality and Kernelization: Lower Bounds and Upper Bounds on Kernel Size. *SIAM Journal on Computing*, 37(4):1077–1106, 2007. 15

[24] Jianer Chen, Henning Fernau, Peter Shaw, Jianxin Wang, and Zhibiao Yang. Kernels for Packing and Covering Problems. In Jack Snoeyink, Pinyan Lu, Kaile Su, and Lusheng Wang, editors, *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, volume 7825 of *LNCS*, pages 199–211. Springer, Heidelberg, 2012. 15, 29, 30

[25] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 298–307, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 17, 99

[26] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast proteinprotein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006. 2

[27] Shenshi Chen and Zhixiang Chen. Faster deterministic algorithms for packing, matching and t-dominating set problems. *CoRR*, abs/1306.3602, 2013. 17, 18

[28] Benny Chor, Mike Fellows, and David Juedes. Linear Kernels in Linear Time, or How to Save k Colors in $\mathcal{O}(n^2)$ Steps. In *Proceedings of the 30th Workshop on Graph Theoretic Concepts in Computer Science (WG 2004)*, volume 3353, pages 257–269, 2004. 14, 29, 30, 38, 39

[29] James S. Coleman. *Introduction to Mathematical Sociology.* New York: Free Press, 1978. 2

[30] Derek G. Corneil, Shigeru Masuyama, and S. Louis Hakimi. Edge-disjoint packings of graphs. *Discrete Applied Mathematics*, 50(2):135–148, 1994. 16

[31] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 277–288. ACM, 2013. 107, 122, 127

[32] William H. Cunningham and A.B. Marsh. A primal algorithm for optimum matching. In M.L. Balinski and A.J. Hoffman, editors, *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 50–72. Springer Berlin Heidelberg, 1978. 13

[33] Narek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer International Publishing, 2015. 9, 46

[34] Peter Damaschke. Fixed-Parameter Tractable Generalizations of Cluster Editing. In *The 6th International Conference on Algorithms and Complexity (CIAC)*, volume 3998, pages 344–355, 2006. 19

[35] Holger Dell and Dániel Marx. Kernelization of packing problems. In Y. Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 68–81. SIAM, 2012. 15, 17, 126

[36] Holger Dell and Dieter Van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. 15

[37] Anders Dessmark, Andrzej Lingas, and Andrzej Proskurowski. Maximum packing for k-connected partial k-trees in polynomial time. *Theoretical Computer Science*, 236(1–2):179–191, 2000. 14

[38] Reinhard Diestel. *Graph Theory.* Springer-Verlag, Heidelberg, 2005. 7

[39] Dorit Dor and Michael Tarsi. Graph decomposition is NP-complete: A complete proof of Holyer's conjecture. *SIAM Journal on Computing*, 26(4):1166–1187, 1997. 16, 26

[40] Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense implicit communities in the Web graph. *ACM Transactions on the Web*, 3(2):1–36, 2009. 2

[41] R. G. Downey, M. R. Fellows, and M. Koblitz. Techniques for exponential parameterized reductions in vertex set problems. (Unpublished, reported in [44], §8.3). 17, 18

[42] Rodney G. Downey, Judith Egan, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. Dynamic Dominating Set and Turbo-Charging Greedy Heuristics. *Tsinghua Science and Technology*, 19(4):329–337, 2014. iv, 126

[43] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag London, 2013. iv, 5, 9

[44] Rodney G. Downey and Mike R. Fellows. *Parameterized Complexity*. Springer, 1999. iii, 5, 9, 10, 39, 46, 67, 93, 145

[45] Jan Dreier, Philipp Kuinke, Rafael Przybylski, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Overlapping communities in social networks. *CoRR*, abs/1412.4973, 2014. 123

[46] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 291–299. Society for Industrial and Applied Mathematics, 1999. 3

[47] Martin Dyer and Alan Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10:139–153, 1985. 13, 16

[48] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. 13

[49] Jack Edmonds and Ellis L. Johnson. Matching: A well-solved class of integer linear programs. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 27–30. Springer Berlin Heidelberg, 2003. 13, 14

[50] Schaeffer S. Elisa. Survey: Graph clustering. *Journal Computer Science Review*, 1(1):27–64, 2007. 19

[51] Lucia Falzon. Determining Groups from the Clique Structure in Large Social Networks. *Social Networks*, 22(2):159–172, 2000. 2

[52] Michael R. Fellows, Jiong Guob, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. Graph-based data clustering with overlaps. *Discrete Optimization*, 8(1):2–17, 2011. 19

[53] Michael R. Fellows, Christian Knauer, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Ulrike Stege, Dimitrios M. Thilikos, and Sue Whitesides. Faster Fixed-Parameter Tractable Algorithms for Matching and Packing Problems. *Algorithmica*, 52(2):167–176, 2008. 15, 17, 18, 67, 71, 99, 100, 105, 127

[54] Mike Fellows, Pinar Heggernes, Frances Rosamond, Christian Sloper, and Jan Arne Telle. Finding k Disjoint Triangles in an Arbitrary Graph. In Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel, editors, *The 30th Workshop on Graph-Theoretic Concepts in Computer Science*, volume 3353 of *LNCS*, pages 235–244. Springer, Heidelberg, 2004. iv, 14, 15, 16, 29, 30

[55] Henning Fernau, Fedor V. Fomin, Geevarghese Philip, and Saket Saurabh. The Curse of Connectivity: t-Total Vertex (Edge) Cover. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics (COCOON)*, volume 6196 of *Lecture Notes in Computer Science*, pages 34–43. Springer Berlin Heidelberg, 2010. 15

[56] Henning Fernau, Alejandro López-Ortiz, and Jazmín Romero. Kernelization algorithms for packing problems allowing overlaps. In Rahul Jain, Sanjay Jain, and Frank Stephan, editors, *Theory and Applications of Models of Computation (TAMC)*, volume 9076 of *LNCS*, pages 415–427. Springer, 2015. 6

[57] Henning Fernau, Alejandro López-Ortiz, and Jazmín Romero. Using Parametric Transformations Toward Polynomial Kernels for Packing Problems Allowing Overlaps. *ACM Transactions on Computation Theory*, 7(3), 2015. 6

[58] Henning Fernau and Daniel Raible. A parameterized perspective on packing paths of length two. *Journal of Combinatorial Optimization*, 18(4):319–341, 2009. 15

[59] Gary William Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004. 3

[60] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 142–151. SIAM, 2014. 17

[61] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010. iii, 3, 19, 107

[62] Linton C. Freeman. The Sociological Concept of "Group": An Empirical Test of Two Models. *American Journal of Sociology*, 98(1):152–166, 1992. 107, 127

[63] Linton C. Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science*. BookSurge Publishing, 2004. 2

[64] Ariel Gabizon, Daniel Lokshtanov, and Micha Pilipczuk. Fast algorithms for parameterized problems with relaxed disjointness constraints. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 545–556. Springer Berlin Heidelberg, 2015. 18, 127

[65] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. 9

[66] Michelle Girvan and Mark Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002. 2, 3

[67] A. J. Goldman. Optimal matchings and degree-constrained subgraphs. *Journal of Research of the National Bureau of Standards*, 68B(1):27–29, 1964. 8

[68] Tatiana Gossen, Michael Kotzyba, and Andreas Nürnberger. Graph clusterings with overlaps: Adapted quality indices and a generation model. *Neurocomputing*, 123:13–22, 2014. 107

[69] Prachi Goyal, Neeldhara Misra, Fahad Panolan, and Meirav Zehavi. Deterministic algorithms for matching and packing problems based on representative sets. *SIAM Journal on Discrete Mathematics*, 29(4):1815–1836, 2015. 17, 18

[70] Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973. 120

[71] Sepp Hartung, Christian Komusiewicz, and André Nichterlein. On structural parameterizations for the 2-club problem. In *Proceedings of the 39th Interanational Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 233–243, 2013. 19

[72] David Hartvigsen, Pavol Hell, and Jácint Szabó. The k-piece packing problem. *Journal of Graph Theory*, 52(4):267–293, 2006. 14

[73] Lenwood S. Heath and John Paul C. Vergara. Edge-Packing in Planar Graphs. *Theory of Computing Systems*, 31(6):629–662, 1998. 16

[74] Lenwood S. Heath and John Paul C. Vergara. Edge-packing planar graphs by cyclic graphs. *Discrete Applied Mathematics*, 81(1-3):169 – 180, 1998. 16

[75] Pavol Hell. Graph Packings. *Electronic Notes in Discrete Mathematics*, 5:170–173, 2000. 14

[76] Pavol Hell and David G. Kirkpatrick. Schedulling, matching and coloring. *Colloquia Math Society*, 25:273–279, 1978. 4

[77] Pavol Hell and David G. Kirkpatrick. Star factors and star packings. Technical Report 82-6, Simon Fraser University, 1982. 14

[78] Pavol Hell and David G. Kirkpatrick. Packings by cliques and by finite families of graphs. *Discrete Mathematics*, 49(1):45–59, 1984. 14

[79] Pavol Hell and David G. Kirkpatrick. Packings by complete bipartite graphs. *SIAM Journal on Algebraic and Discrete Methods*, 7:199–209, 1986. 14

[80] Pavol Hell, David G. Kirkpatrick, Jan Kratochvíl, and I. Kříž. On restricted two-factors. *SIAM Journal on Discrete Mathematics*, 1(4):472–484, 1988. 14

[81] Dorit S. Hochbaum. *Approximation algorithms for NP-hard problems.* Course Technology, 1997. 5

[82] Micha Hofri. *Analysis of Algorithms: Computational Methods and Mathematical Tools.* Oxford University Press, 1995. 5

[83] Ian Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981. 16, 26

[84] Stefan Hougardy. Linear Time Approximation Algorithms for Degree Constrained Subgraph Problems. In William Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 185–200. Springer Berlin Heidelberg, 2009. 8

[85] Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and turing kernels. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 616–629. SIAM, 2015. 15

[86] Weijia Jiaa, Chuanlin Zhanga, and Jianer Chenc. An efficient parameterized algorithm for m-set packing. *Journal of Algorithms*, 50(1):106–117, 2004. 16, 17

[87] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972. iii, 16

[88] David G. Kirkpatrick and Pavol Hell. On the completeness of a generalized matching problem. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 240–245, 1978. iii, 4, 13, 14, 20, 22, 25, 126

[89] Morton Klein and Hiroshi Takamori. Parallel line assignment problems. *Management Science*, 19(1):1–10, 1972. 4

[90] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer Berlin Heidelberg, 2006. 15, 16, 17, 99

[91] Christian Komusiewicz and Manuel Sorge. Finding dense subgraphs of sparse graphs. In *Parameterized and Exact Computation - 7th International Symposium (IPEC)*, pages 242–251, 2012. 19

[92] Ioannis Koutis. A faster parameterized algorithm for set packing. *Information Processing Letters*, 94:7–9, 2005. 17, 18

[93] Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In Luca Aceto, Ivan Damgøard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586. Springer Berlin Heidelberg, 2008. 17, 18

[94] Tomas Kovacs and Andrzej Lingas. Maximum Packing for Biconnected Outerplanar Graphs. *Discrete Applied Mathematics*, 100(1–2):85–94, 2000. 14

[95] Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. In Jianer Chen and FedorV. Fomin, editors, *Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 264–275. Springer Berlin Heidelberg, 2009. 15

[96] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 450–459, Washington, DC, USA, 2012. IEEE Computer Society. iv, 126

[97] P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In Subhash Bhalla, editor, *Databases in Networked Information Systems*, volume 2544 of *Lecture Notes in Computer Science*, pages 188–200. Springer Berlin Heidelberg, 2002. 2

[98] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 31:1481–1493, 1999. iii, 3, 4

[99] Junqiu Li, Xingyuan Wang, and Justine Eustace. Detecting overlapping communities by seed community in weighted complex networks. *Physica A: Statistical Mechanics and its Applications*, 392(23):6125 – 6134, 2013. 120, 122

[100] Andrzej Lingas. Maximum tree-packing in $O(n^{5/2})$. *Theoretical Computer Science*, 181(2):307–316, 1997. 14

[101] Yang Liu, Songjian Lu, Jianer Chen, and Sing-Hoi Sze. Greedy localization and color-coding: Improved matching and packing algorithms. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation*, volume 4169 of *Lecture Notes in Computer Science*, pages 84–95. Springer Berlin Heidelberg, 2006. 17, 99, 105

[102] Alejandro López-Ortiz and Jazmín Romero. Arbitrary overlap constraints in graph packing problems. Manuscript, 2005. 6

[103] Alejandro López-Ortiz and Jazmín Romero. Parameterized algorithms for the H-packing with t-overlap problem. *Journal of Graph Algorithms and Applications*, 18(4):515–538, 2014. 6

[104] Feng Luo, James Z. Wang, and Eric Promislow. Exploring local community structures in large networks. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 233–239. IEEE Computer Society, 2006. 2

[105] Shigeru Masuyama and Toshihide Ibaraki. Chain packing in graphs. *Algorithmica*, 6:826–839, 1991. 14, 16

[106] Luke Mathieson, Elena Prieto, and Peter Shaw. Packing edge disjoint triangles: A parameterized view. In *Proceedings of the First International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, pages 127–137, 2004. 29, 30

[107] Julian Mcauley and Jure Leskovec. Discovering social circles in ego networks. *ACM Trans. Knowl. Discov. Data*, 8(1):4:1–4:28, 2014. 2, 3

[108] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, SFCS '80, pages 17–27. IEEE Computer Society, 1980. 24, 25, 88

[109] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics.* Springer-Verlag, 2000. 5

[110] Giovanna Miritello. *Temporal Patterns of Communication in Social Networks.* Springer Theses. Springer International Publishing, 1 edition, 2013. 120

[111] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E. Tarjan. Clustering social networks. In Anthony Bonato and Fan R.K. Chung, editors, *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 56–67. Springer Berlin Heidelberg, 2007. 3, 107, 116, 127

[112] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 251–260, New York, NY, USA, 2010. ACM. 2, 3, 107

[113] Alan E Mislove. *Online social networks: measurement, analysis, and applications to distributed information systems*. ProQuest, 2009. 2

[114] Hannes Moser. A Problem Kernelization for Graph Packing. In Mogens Nielsen, Antonín Kučera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tůma, and Frank Valencia, editors, *SOFSEM 2009*, volume 5404 of *LNCS*, pages 401–412. Springer, Heidelberg, 2009. iv, 14, 15, 67, 71, 83, 91

[115] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. 5

[116] Chiheb-Eddine Ben N'Cir, Guillaume Cleuziou, and Nadia Essoussi. Overview of overlapping partitional clustering methods. In M. Emre Celebi, editor, *Partitional Clustering Algorithms*, pages 245–275. Springer International Publishing, 2015. iii, 3, 19, 122

[117] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford, 2006. 9, 10, 39, 46, 67, 93

[118] J-P Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A.-L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007. 120

[119] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. iii, xii, 3, 4

[120] Sarah Pantel. Graph packing problems. Master's thesis, Simon Fraser University, Burnaby, B.C., Canada, 1998. 14

[121] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. 9

[122] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012. iii, 2, 3, 19

[123] Elena Prieto and Christian Sloper. Looking at the stars. *Theoretical Computer Science*, 351(3):437–445, 2006. 14, 29, 30, 31

[124] Alexander W. Rives and Timothy Galitski. Modular organization of cellular networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(3):1128–1133, 2003. 2

[125] Neil Robertson and Paul Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–327, 2004. 120

[126] Jazmín Romero and Alejandro López-Ortiz. The $\mathcal{G}$-Packing with t-Overlap Problem. In Sudebkumar Prasant Pal and Kunihiko Sadakane, editors, *8th International Workshop on Algorithms and Computation (WALCOM 2014)*, volume 8344 of *LNCS*, pages 114–124. Springer, Heidelberg, 2014. 6

[127] Jazmín Romero and Alejandro López-Ortiz. A Parameterized Algorithm for Packing Overlapping Subgraphs. In Edward A. Hirsch, Sergei O. Kuznetsov, Jean-Éric Pin, and Nikolay K. Vereshchagin, editors, *9th International Computer Science Symposium in Russia (CSR 2014)*, volume 8476 of *LNCS*, pages 325–336. Springer, Heidelberg, 2014. 6

[128] Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012. 19

[129] Yossi Shiloach. Another look at the degree constrained subgraph problem. *Information Processing Letters*, 12(2):89–92, 1981. xii, 24, 25, 88

[130] Chao Tong, Zhongyu Xie, Xiaoyun Mo, Jianwei Niu, and Yan Zhang. Detecting overlapping communities of weighted networks by central figure algorithm. In *Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE*, pages 7–12, Oct 2014. 122

[131] John Paul C. Vergara and Lenwood S. Heath. Edge packing by isomorphic subgraphs. Technical Report TR-91-03, Virginia Polytechnic Institute and State University, 1991. 16

[132] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM. 127

[133] Jianxin Wang, Qilong Feng, and Jianer Chen. An $o^*(3.523^k)$-time parameterized algorithm for the 3-set packing problem. *Theoretical Computer Science*, 412(18):1745 – 1753, 2011. Theory and Applications of Models of Computation (TAMC 2008). 17, 99

[134] Jianxin Wang, Dan Ning, Qilong Feng, and Jianer Chen. An improved kernelization for p_2-packing. *Information Processing Letters*, 110:188–192, 2010. 15, 29, 30

[135] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, pages 205–218, New York, NY, USA, 2009. ACM. 2, 127

[136] Kai Wu, Yasuyuki Taki, Kazunori Sato, Yuko Sassa, Kentaro Inoue, Ryoi Goto, Ken Okada, Ryuta Kawashima, Yong He, Alan C. Evans, and Hiroshi Fukuda. The overlapping community structure of structural brain network in young healthy individuals. *PLoS ONE*, 6(5):e19608, 05 2011. iii, 3

[137] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):43, 2013. iii, 3, 19, 107

[138] Jaewon Yang and Jure Leskovec. Structure and overlaps of ground-truth communities in networks. *ACM Transactions on Intelligent Systems and Technology*, 5(2):1–35, 2014. 107, 119

[139] Moustafa A. Youssef, Adel Youssef, and Mohamed F. Younis. Overlapping multihop clustering for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(12):1844–1856, 2009. 107, 120, 122

[140] Meirav Zehavi. Deterministic parameterized algorithms for matching and packing problems. CoRR abs/1311.0484, 2013. 17, 18, 51, 62

[141] Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 1037–1049. Springer, 2015. 17

[142] Elena Zotenko, Katia S Guimaraes, Raja Jothi, and Teresa M Przytycka. Decomposition of overlapping protein complexes: A graph theoretical method for analyzing static and dynamic protein associations. *Algorithms for Molecular Biology*, 1(7):1–11, 2006. iii, 3, 4